В.Ф.Заклякoв

Информатика

Учебник. Пятое издание, переработанное и дополненное

От автора:

Жизнь не стоит на месте. Вы держите в руках бумажную или читаете электронную версию обновлённого пятого издания учебника, созданного на базе: *Грошев А. С., Закляков П. В. Информатика: учеб. для вузов – 4-е изд., перераб. и доп. – М.: ДМК Пресс, 2018. – 672 с.: цв. ил. ISBN 978-5-97060-638-4.* Авторы последнего, в силу своей занятости (или по другим причинам), не изъявили желания заняться обновлением материала в соавторстве, поэтому пришлось взять этот процесс в свои руки и формально оказаться новым и единственным автором. С другой стороны, читатель в тексте найдёт упоминание «мы» не один раз, означающее, что никто не забыт и общее мнение нашего виртуального коллектива совпало. Прошу не судить строго, если что-то где-то по тексту Вам, дорогой читатель, не понравится.

От редакции:

Поскольку желающих отрецензировать данное издание в короткие сроки не нашлось, редакция выпустила данный учебник на свой страх и риск (как это было в своё время со вторым изданием), полагая, что он будет интересен широкому кругу читателей, в том числе:

- студентам вузов;
- школьникам старших классов;
- преподавателям и аспирантам;
- системным администраторам;
- всем, кто в силу своей природной любознательности интересуется устройством компьютера и принципами его работы.

Читатели и представители образовательных учреждений заинтересованные в наличии формальной рецензии могут прислать её в редакцию (dmkpress@gmail.com) для размещения на сайте или высказать свои пожелания и замечания напрямую автору (zvf305@yandex.ru).

Москва, 2021

УДК 004
ББК 32.97

Закляков В. Ф.
Информатика: учеб. для вузов – 5-е изд., перераб. и доп. – М.: ДМК Пресс, 2021. – 750 с.: цв. ил.

ISBN 978-5-97060-921-7

Компьютеры окружают нас повсюду – сегодня они стали такой же будничной реалией, как бытовые электроприборы. В этой книге подробно объясняется что ныне понимается под информацией и как компьютер её обрабатывает.

Для этого, сначала даётся академическое изложение материала в стиле учебника, приводятся базовые сведения из области информатики: основы кодирования текстов, чисел, преобразования аналогового сигнала в цифровой вид и т. д. Излагаются основы вычислений и булевой алгебры. В последующей (условно справочной) части освещается история ЭВМ от ламп и транзисторов до нетбуков и планшетов, рассматриваются аппаратное устройство и программное обеспечение современного компьютера.

В процессе чтения вы узнаете много интересных фактов и случаев из истории, примеров эффективного применения свободного ПО на базе ОС GNU/Linux, несомненно станете мудрее (поскольку в тексте содержится много различных life hack'ов). В отдельную главу вынесены актуальные выдержки из законодательства РФ, раскрывающие юридические аспекты работы с информацией. Приложения к главам содержат лабораторные работы, рекомендуемые для самостоятельного выполнения и закрепления изученного материала.

Информация по базам данных и программированию в книге отсутствует.

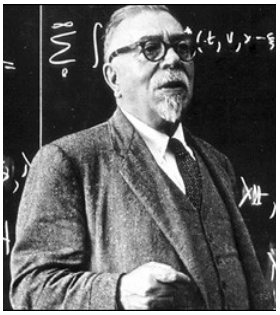
Ил. 540. Табл. 258.

УДК 004
ББК 32.97

ISBN 978-5-97060-921-7

Введение

Выделение информатики как самостоятельной области человеческой деятельности связано в первую очередь с развитием компьютерной техники. Слово «*информатика*» (*франц. informatique*) появилось в начале 1960-х годов как гибрид двух французских слов – «*information*» (*информация*) и «*automatique*» (*автоматика*) и дословно означает «*информационная автоматика*». Во Франции и других странах Европы данный термин применяется для обозначения области научных знаний, связанных с автоматизацией обработки информации с помощью ЭВМ. Фактическое его использование немного шире, потому как до изобретения электронной вакуумной лампы и транзистора также существовали разнообразные вычислительные машины, скорее механические и электромеханические, нежели электронные.



Но́рберт Ви́нер

Возможно, чтобы скрасить сей недочёт, а может и по другим причинам, в 70-е годы в англоязычной литературе наука о переработке информации на основе вычислительной техники утвердилась под названием «*Computer Science*», что буквально означает «*компьютерная наука*».

В нашей стране в 60-е годы XX века вопросы, связанные с разработкой, функционированием и применением автоматизированных систем обработки информации, объединялись термином «*кибернетика*[1]», хотя это было не вполне верно, поскольку, по определению Н. Винера [2], кибернетика – это наука о законах управления в живой и неживой природе [1], то есть сфера её интересов охватывает лишь часть (хотя обширную и важную) вопросов, связанных с информацией и информационными процессами. К началу 80-х годов термин «*кибернетика*» постепенно ушёл из широкого распространения.

Виктор Михайлович Глушков

Так что же такое *информатика*?

Академик В. М. Глушков [3] в письме президенту Академии наук СССР по поводу создания нового отделения Академии ссылается на определение, данное Международным конгрессом по информатике, проходившим в Японии в 1978 году: «*Понятие информатики охватывает области, связанные с разработкой, созданием, использованием и материально-техническим обслуживанием систем обработки информации, включая машины, оборудование, математическое обеспечение, организационные аспекты, а также комплекс промышленного, коммерческого, административного, социального и политического воздействия*». Это очень широкое определение, его нужно кон-

[1] От греч. kiber – над, nautus – моряк, кормчий, управляющий рулём, предположительно, отсюда у Н.Винера появилось «управление». Впервые термин «кибернетика» встречается в работах древнегреческого философа Платона (ок. 427-347 гг. до н. э.), в которых он обозначил правила управления обществом. Через две с лишним тысячи лет французский физик А. И. Ампер (1775–1836) в своей работе «Опыт философских наук» (1834) термин «кибернетика» также применил к науке об управлении обществом. [8]

[2] Но́рберт Ви́нер (англ. Norbert Wiener; 1894–1964) – американский учёный, выдающийся математик и философ, основоположник кибернетики и теории искусственного интеллекта.

кретизировать, но в нём выделено главное – понятие о вычислительной технике и автоматизации, которые не существуют отдельно друг от друга [3]. После, в 1983 году на сессии годичного собрания Академии наук СССР, и было организовано новое «Отделение информатики, вычислительной техники и автоматизации». Информатика стала рассматриваться как «*комплексная научная и инженерная дисциплина, изучающая все аспекты разработки, проектирования, создания, оценки, функционирования основанных на ЭВМ систем переработки информации, их применения и воздействия на различные области социальной практики*». Были созданы институты данного профиля, информатика стала обязательным предметом в системе среднего и высшего образования.

Мы предложим читателям следующее определение, взятое из [4]:

Информатика – наука, изучающая общие свойства информации, закономерности и способы её создания, хранения, поиска, преобразования и использования с помощью компьютерных систем.

На сайте Министерства образования и науки Российской Федерации приведена «*Примерная программа по информатике и информационным технологиям*» [4] для среднего (полного) общего образования и «*Примерная программа дисциплины информатика*» [5], составленная в соответствии с государственными образовательными стандартами высшего профессионального образования для направлений «*Технические науки*», «*Образование*» и прочих. Обе программы содержат разделы:

- общие сведения об информации,
- аппаратное и программное обеспечение информационных систем,
- технологии обработки текстовой, числовой, графической и мультимедийной информации,
- работа с базами данных,
- алгоритмизация и программирование,
- локальные и глобальные компьютерные сети.

Сравнение двух программ показывает, что школьный и вузовский курсы информатики достаточно близки по содержанию. Следовательно, задача вуза – углубить знания по информатике, полученные в школе, познакомить студентов со всеми новостями в области технического и программного обеспечения информационных процессов, а также с использованием информационных технологий в их будущей профессиональной деятельности. При выполнении практических работ студент должен усовершенствовать свои навыки в работе на компьютере в процессах обработки, поиска и хранения всех видов информации.

Смеем предположить, что компьютер как для грамотного специалиста, так и для обычного человека должен стать привычным инструментом автоматизации его интел-

[3] Ви́ктор Миха́йлович Глушко́в (1923–1982) – советский математик, кибернетик. Решил обобщённую пятую проблему Гильберта. Был инициатором и главным идеологом разработки и создания ОГАС, предназначенной для автоматизированного управления всей экономикой СССР. Для этого им была разработана система алгоритмических алгебр и теория для управления распределёнными базами данных. Также под его руководством в 1966 году была разработана первая персональная ЭВМ «МИР-1» (машина для инженерных расчётов).

[4] *http://window.edu.ru/window_catalog/files/r37206/09-1-s.pdf*.

[5] *http://www.edu.ru/db/portal/spe/prog/htm/mf0201.htm*.

лектуального труда. На удивление, выделенные В. А. Острейковским [6] более десятилетия назад основные признаки информационного общества [4] находят всё большее отражение в нашей повседневной жизни, и вы без труда узнаете их сами:

1. Большинство работающих в информационном обществе (около 80%) занято в информационной сфере, то есть сфере производства информации и информационных услуг.

2. Обеспечены техническая, технологическая и правовая возможности доступа любому члену общества практически в любой точке территории и в приемлемое время к нужной ему информации (за исключением военных и государственных секретов, точно оговорённых в соответствующих законодательных актах).

3. Информация становится важнейшим стратегическим ресурсом общества и занимает ключевое место в экономике, образовании и культуре.

Информатизация – необходимое условие научно-технического, социального, экономического и политического прогресса в обществе. Неизбежность информатизации обусловлена следующими причинами:

- беспрецедентным усложнением социально-экономических процессов в результате увеличения масштабов и темпов общественного производства, углубления разделения труда и его специализации в научно-технической революции;
- необходимостью адекватно реагировать на возникающие проблемы в динамично меняющейся обстановке, присущей постоянно развивающемуся обществу;
- повышением степени самоуправления предприятий, территорий, регионов.

Конечно, процесс перехода от индустриального общества к информационному происходит не только не одновременно в различных странах, но и внутри России имеет разную скорость и темпы развития.

Полагаем, что читатели и без авторов заметили, что в нашей жизни буквально за несколько лет появились такие, вещи как «электронные билеты», когда для посадки на самолёт или поезд нужен лишь паспорт. Различные «электронные приёмные» от уровня Президента страны до глав управ и районов. Всё больше появляется терминалов для оплаты пластиковыми картами (особенно в мегаполисах), повсеместно появляются терминалы для приёма денег. Сотовая связь и её сервисы покрывают большинство мест проживания граждан. На рекламах вместо телефонов и адресов мы видим непонятые квадратики с точками (QR-коды). Электронные книги вытесняют бумажные издания.

Надеемся, что наш бумажный учебник окажется вам полезным.

[6] Острейковский Владислав Алексеевич (1932 г. р.) – д. т. н., заслуженный деятель науки и техники РФ, академик Международной академии информатизации. В 1993 году получил медаль Минатома РФ «*За заслуги в повышении безопасности атомных станций*». В 1995 году по представлению Стокгольмского королевского университета был удостоен диплома Кембриджского университета (Великобритания) «*Человек года в области кибернетики*». С 1985 года является членом Ядерного общества, редколлегии журнала «Ядерная энергетика», Президиума объединенного учебно-методического совета Министерства образования РФ.

Глава 1. Общие сведения об информационных процессах

1.1. Понятие информации, её виды и свойства

Вся жизнь человека так или иначе связана с накоплением и обработкой информации, которую он получает из окружающего мира, используя пять органов чувств – зрение, слух, вкус, обоняние и осязание. Как научная категория «*информация*» составляет предмет изучения для самых различных дисциплин: ***информатики***, ***теории связи, теории информации, кибернетики, философии, физики, биологии, семиотики*** (науки о знаках и знаковых системах), ***теории массовой коммуникации*** (исследование средств массовой информации (СМИ) и их влияния на общество), ***соционики*** (теории информационного метаболизма индивидуальной и социальной психики), ***информодинамики*** (науки об открытых информационных системах), ***информациологии*** (науки о получении, сохранении и передаче информации для различных множеств объектов) и т. д. И всё же общепринятого строгого научного определения информации до настоящего времени не существует, а вместо него обычно используют понятие об информации. В разных областях науки и техники в это понятие вкладывают разный смысл, чтобы этот смысл в наибольшей степени соответствовал задачам соответствующей предметной области. Имеется множество определений понятия информации – от наиболее общего философского (информация есть отражение реального мира) до наиболее частного прикладного (информация есть сведения, являющиеся объектом переработки). Вот некоторые из них:

- сообщение, осведомление о чём-либо;
- сведения об окружающем мире, протекающих в нём процессах и т. д., которые воспринимают живые организмы, управляющие машины и другие информационные системы в процессе жизнедеятельности и работы;
- сведения о лицах, предметах, фактах, событиях, явлениях и процессах независимо от формы их представления;
- результат отражения реальности в сознании человека, представленный на его внутреннем языке;
- сведения, содержащиеся в сообщении и рассматриваемые как объект передачи, хранения и обработки;
- знания, переданные кем-то другим или приобретенные путём собственного исследования и изучения;
- содержание сигнала, сообщения;
- содержательное описание объекта или явления;
- семантика или прагматика синтаксиса языка представления данных;
- продукт научного познания, средство изучения реальной действительности;
- основное содержание отображения;
- результат выбора;

- мера разнообразия;
- отражённое разнообразие;
- сущность, сохраняющаяся при вычислимом изоморфизме;
- мера сложности структур, мера организации;
- вечная категория, содержащаяся во всех без исключения элементах и системах материального мира, проникающая во все «поры» жизни людей и общества;
- бесконечный законопроцесс триединства энергии, движения и массы с различными плотностями кодовых структур бесконечно-беспредельной Вселенной.

Наряду с названными существуют сотни других, иногда противоречащих друг другу или взаимоисключающих определений информации. По мнению профессора Игоря Н. Бекмана [9], к настоящему времени различных определений этого термина накопилось где-то около пятисот, но список не закрыт, так как дать однозначного и чёткого определения информации до сих пор никому не удалось, и вряд ли когда удастся. По крайней мере, в наше время довольно распространено мнение, что информация наряду с материей и энергией является первичным понятием нашего мира и поэтому в строгом смысле не может быть определена. Многообразие этих определений свидетельствует о широте подхода к понятию информации и свидетельствует о том, что становление концепции информации в современной науке ещё не завершено.

В Федеральном законе Российской Федерации от 27 июля 2006 г. № 149-ФЗ *«Об информации, информационных технологиях и о защите информации»* [7] даётся следующее определение этого термина:

«*информация – сведения (сообщения, данные) независимо от формы их представления*».

Толковый словарь русского языка [8] Сергея Ивановича О́жегова [9] приводит два определения слова *информация*:

1) Сведения об окружающем мире и протекающих в нём процессах, воспринимаемые человеком или специальными устройствами.
2) Сообщения, осведомляющие о положении дел, о состоянии чего-нибудь. (*Научно-техническая и газетная информация, СМИ – печать, радио, телевидение, кино, интернет – прим. авторов*).

В ***информатике*** наиболее часто используется следующее определение этого термина:

Информация – это осознанные сведения об окружающем мире, которые являются объектом хранения, преобразования, передачи и использования. *Сведения* – это знания, выраженные в сигналах, сообщениях, известиях, уведомлениях и т. д.

Каждого человека в мире окружает море информации различных видов. Стремление зафиксировать, сохранить надолго своё восприятие информации было всегда свойственно человеку. Мозг человека хранит множество информации и использует для хра-

[7] http://www.rg.ru/2006/07/29/informacia-dok.html

[8] Ожегов С. И. Словарь русского языка: ок. 57 000 слов / под ред. чл.-корр. АН СССР Н. Ю. Шведовой. - 20-е изд., стереотип. – М.: Рус. яз., 1988. – 750 с. ISBN 5-200-00313-X.

[9] Серге́й Ива́нович О́жегов (1900–1964) – лингвист, лексикограф, доктор филологических наук, профессор. Ударение в фамилии *О́жегов* ставится на первом слоге на букве «о», так как эта фамилия является производной от слова «о́жег», что означало палку, которую в старину окунали в расплавленный металл, чтобы определить степень его готовности к разливке.

нения её свои способы, основа которых – скорее всего, двоичный код [4], как и у компьютеров. Человек всегда стремился иметь возможность поделиться своей информацией с другими людьми и найти надёжные средства для её передачи и долговременного хранения. Для этого в настоящее время изобретено множество способов хранения информации на внешних (относительно мозга человека) носителях и её передачи на огромные расстояния.

Основные **виды информации** по её форме представления, способам её кодирования и хранения, что имеет наибольшее значение для информатики это:

- **графическая** или **изобразительная** – первый вид, для которого был реализован способ хранения информации об окружающем мире в виде наскальных рисунков, а позднее в виде картин, фотографий, схем, чертежей на бумаге, холсте, мраморе и других материалах, изображающих картины реального мира;
- **звуковая** – мир вокруг нас полон звуков, и задача их хранения и тиражирования была решена с изобретением звукозаписывающих устройств в 1877 г. [10]; её разновидностью является **музыкальная** информация – для этого вида был изобретён способ кодирования с использованием специальных символов (музыкальных нот), что делает возможным хранение её аналогично графической информации;
- **текстовая** – способ кодирования речи человека специальными символами – буквами, причём разные народы имеют разные языки и используют различные наборы букв для отображения речи; особенно большое значение этот способ приобрёл после изобретения бумаги и книгопечатания;
- **числовая** – количественная мера объектов и их свойств в окружающем мире; особенно большое значение приобрела с развитием торговли, экономики и денежного обмена; аналогично текстовой информации для её отображения используется метод кодирования специальными символами – цифрами, причём системы кодирования (счисления) могут быть разными;
- **видеоинформация** – способ сохранения движущихся картин окружающего мира, появившийся с изобретением кино.

Существуют также виды информации, для которых до сих пор не изобретено эффективных способов их кодирования и хранения, – это **тактильная** информация, передаваемая ощущениями, **органолептическая**, передаваемая запахами и вкусами, и другие виды, для которых современная наука даже не нашла признанных всеми терминов определения (например, экстрасенсорная информация).

Для передачи информации на большие расстояния первоначально использовались кодированные световые сигналы, с изобретением электричества – передача закодированного определённым образом сигнала по проводам, позднее – с использованием радиоволн.

Хранение информации при использовании компьютеров осуществляется на магнитных дисках или лентах, на лазерных дисках (**CD, DVD, BD**), специальных устройствах энергонезависимой памяти (флэш-память и прочее). Эти методы постоянно совершенствуются, изобретаются новые устройства и носители информации. Обработку информации (воспроизведение, преобразование, передачу, запись на внешние носители) выполняет процессор компьютера. С помощью компьютера возможно создание и

[10] Рекомендуем ознакомиться с экспозицией «История звукозаписи» в музее радио имени А. С. Попова (г. Екатеринбург, ул. Розы Люксембург/Энгельса, 9/11, филиал Свердловского областного краеведческого музея, +7 (343) 371-50-60, http://uole-museum.ru).

хранение новой информации любых видов, для чего служат специальные программы, используемые на компьютерах, и устройства ввода информации.

Особым видом информации в настоящее время можно считать информацию, представленную в глобальной сети интернет. Здесь используются особые приёмы хранения, обработки, поиска и передачи распределённой информации больших объёмов и особые способы работы с различными видами информации.

Постоянно совершенствуется программное обеспечение, благодаря которому становится возможным не только коллективно работать с информацией, взять хотя бы *GoogleDocuments*, но и для многих программ появляется возможность сохранения «в облако» (сервисы *Яндекс.Диск, Amazon S3, Box.com, Copy.com, Dropbox, DVCS-Autosync, Google Drive, iCloud, iDrive, ownCloud, Rackspace Cloud Files, Selectel Cloud Storage, SkyDrive, SparkleShare, SugarSync, Ubuntu One, Windows Azure Blob и др.*). Смеем предположить, что чувства, испытываемые нашими внуками при виде «флэшек», будут сродни тем, что испытывает читатель при виде перфокарт сегодня.

1.2. Свойства информации

Как и всякий объект, информация обладает свойствами. Характерной отличительной особенностью информации от других объектов природы и общества является дуализм: на свойства информации влияют как свойства исходных данных, составляющих её содержательную часть, так и свойства методов, фиксирующих эту информацию.

С точки зрения информатики наиболее важными представляются следующие ***общие качественные свойства***: объективность, достоверность, полнота, точность, актуальность, полезность, ценность, своевременность, понятность, доступность, краткость и прочие.

1) Объективность информации. Объективный – существующий вне и независимо от человеческого сознания. Информация – это отражение внешнего объективного мира. Информация объективна, если она не зависит от методов её фиксации, чьего-либо мнения, суждения.

Пример. Сообщение «На улице тепло» несёт субъективную информацию, а сообщение «На улице 22 °С» – объективную, но с точностью, зависящей от погрешности средства измерения.

Объективную информацию можно получить с помощью измерительных приборов. Отражаясь в сознании конкретного человека, информация перестаёт быть объективной, так как преобразовывается (в большей или меньшей степени) в зависимости от мнения, суждения, опыта, знаний конкретного субъекта.

2) Достоверность информации. Информация достоверна, если она отражает истинное положение дел. Объективная информация всегда достоверна, но достоверная информация может быть как объективной, так и субъективной. Достоверная информация помогает принять нам правильное решение.

Недостоверной информация может быть по следующим причинам:

- преднамеренное искажение (дезинформация) или непреднамеренное искажение субъективного свойства;
- искажение в результате воздействия помех и недостаточно точных средств измерений.

3) Полнота информации. Информацию можно назвать полной, если её достаточно для понимания и принятия решений. Неполная информация может привести к ошибочному выводу или решению.

4) Точность информации определяется степенью её близости к реальному состоянию объекта, процесса, явления (погрешностью средства измерения).

5) Актуальность информации – важность для настоящего времени, злободневность, насущность. Иногда только вовремя полученная информация может быть полезна.

6) Полезность (ценность) **информации** может быть оценена применительно к нуждам конкретных её потребителей и оценивается по тем задачам, которые можно решить с её помощью.

7) Относимость информации показывает возможность её использования в узкой области, применимость для решения какой-либо конкретной задачи.

Многие свойства могут как дополнять друг друга, так и быть синонимами. В различных ситуациях важны разные свойства. Немного подумав, читатели без труда вспомнят и добавят к этому списку ещё несколько свойств информации.

Самая ценная информация – объективная, достоверная, полная и актуальная. При этом следует учитывать, что и необъективная, недостоверная информация (например, художественная литература) имеет большую значимость для человека. Важно чтобы об этом было сообщено заранее: сказка, мультфильм, фантастический рассказ или используются спецэффекты, все герои сюжета выдуманы. В противном случае, это будет дезинформация, обман, фейковая новость, со своими последствиями.

Социальная (общественная) информация обладает ещё и дополнительными свойствами:

- имеет семантический (смысловой) характер, то есть понятийный, так как именно в понятиях обобщаются наиболее существенные признаки предметов, процессов и явлений окружающего мира;
- имеет языковую природу (кроме некоторых видов эстетической информации, например изобразительного искусства). Одно и то же содержание может быть выражено на разных естественных (разговорных) языках, записано в виде математических формул и т. д.

Процессы и действия в отношении информации: накопление, старение, копирование, размножение, удаление, уничтожение, блокирование, модификация

С течением времени количество информации растёт, информация накапливается, происходят её систематизация, оценка и обобщение. Это свойство назвали ростом и кумулированием информации. (Кумуляция – от лат. *cumulatio* – увеличение, скопление).

Старение информации заключается в уменьшении её ценности с течением времени. Старит информацию появление новой информации, которая уточняет, дополняет или отвергает полностью или частично более раннюю. Научно-техническая информация стареет быстрее, эстетическая (произведения искусства) – медленнее. (Со своей стороны мы постарались наполнить данный учебник по большей части информацией, менее подверженной старению. К сожалению, сделать учебник полностью таковым не удалось.)

Процессы копирования и размножения информации схожи как в теории, так и с точки зрения обывателя, однако, в юридическом аспекте между терминами искусственно внесена разница. Особенно это заметно в отношении компьютерной информации, где копирование – это повторение и устойчивое запечатление её на машинном или ином носителе. Оно может быть осуществлено путём записи содержащегося во внутренней памяти ЭВМ файла на дискету, диск, флешку, его распечатки и т. д. С другой же стороны, из диспозиции (дословного толкования содержания) ст. 272 УК РФ к копированию компьютерной информации не относится копирование от руки, путём фотографирования текста с экрана дисплея, а также считывание информации путём перехвата излучений ЭВМ, расшифровки шумов принтера и пр. (Также см. ниже ещё одно определение копирования.)

Размножение информации (с точки зрения УК РФ), отличается от копирования информации тем, что информация повторяется не на обособленном от оригинального носителе, а на оригинальном носителе (например, в памяти ЭВМ заводится несколько файлов одного и того же содержания), либо на однородном носителе, оставшемся в распоряжении пользователя (например, копия заводится в памяти ЭВМ, образующей с данным компьютером систему, либо на дискете, сознательно оставленной в компьютере).

Как избавиться от информации? Забывание, удаление и уничтожение информации – это одно и тоже или нет? А чем информация отличается от данных? Последние понятия зачастую в теории используются как синонимы, но между ними существуют и принципиальные различия.

Данные – это совокупность сведений, которые зафиксированы на каком-либо носителе – бумаге, диске, флешке, плёнке и т.п. Эти сведения должны быть в форме, пригодной для хранения, передачи и обработки. Дальнейшее преобразование данных позволяет получить информацию. Таким образом, информацией можно назвать результат анализа и преобразования данных.

Уничтожение информации (destruction of information): любое условие, делающее информацию непригодной для использования независимо от причины.[11]

В юридической практике устоялись следующие определения[12]:

а) уничтожение информации – это приведение информации или её части в непригодное для использования состояние независимо от возможности её восстановления. Уничтожением информации не является переименование файла, где она содержится, а также само по себе автоматическое «вытеснение» старых версий файлов последними по времени;

б) блокирование информации – результат воздействия на компьютерную информацию или технику, последствием которого является невозможность в течение некоторого времени или постоянно осуществлять требуемые операции над компьютерной информацией полностью или в требуемом режиме, то есть совершение действий, приводящих к ограничению или закрытию доступа к компьютерному оборудованию и находящимся на нём ресурсам, целенаправленное затруднение доступа законных пользователей к компьютерной информации, не связанное с её уничтожением;

[11] «Финансовые услуги. Рекомендации по информационной безопасности. ГОСТ Р ИСО/ТО 13569-2007» (утв. Приказом Ростехрегулирования от 27.12.2007 № 514-ст), http://docs.cntd.ru/document/1200068821

[12] «Методические рекомендации по осуществлению прокурорского надзора за исполнением законов при расследовании преступлений в сфере компьютерной информации» (утв. Генпрокуратурой России 14 апреля 2014, https://genproc.gov.ru/upload/iblock/a77/мет рек ПКИ.doc)

в) модификация информации – внесение изменений в компьютерную информацию (или ее параметры). Законом установлены случаи легальной модификации программ (баз данных) лицами, правомерно владеющими этой информацией, а именно: модификация в виде исправления явных ошибок; модификация в виде внесения изменений в программы, базы данных для их функционирования на технических средствах пользователя; модификация в виде частной декомпиляции программы для достижения способности к взаимодействию с другими программами;

г) копирование информации – создание копии имеющейся информации на другом носителе, то есть перенос информации на обособленный носитель при сохранении неизменной первоначальной информации, воспроизведение информации в любой материальной форме – от руки, фотографированием текста с экрана дисплея, а также считывания информации путем любого перехвата информации и т.п.

Все аспекты в отношении информации невозможно рассмотреть коротко, поэтому завершим параграф[13] очевидным: логичность, компактность, удобная форма представления облегчают понимание и усвоение информации, не зря говорят: «краткость – сестра таланта».

1.3. Характеристики информации

Рассмотрим вопросы с связанные с количественными и качественными характеристиками информации, а так же как и в чём их оценивать (измерять).

1.3.1. Формальные единицы измерения информации в технике

В теории кодирования и передачи сообщений под количеством информации понимают количество кодируемых, передаваемых или хранимых символов. При этом используют простой способ определения количества информации как число использованных символов. (Подробнее об этом методе оценки см. п. 1.3.3.1.) Он основан на подсчёте числа символов в сообщении, то есть связан с его длиной и не учитывает содержания. Скорее это не измерение количества информации, а потенциально возможная верхняя оценка информационной вместимости носителя или сообщения. Для упрощения и формализации процесса оценки в вычислительной технике символы исходного алфавита сообщения обычно кодируются двоичными числами, то есть с использованием нулей и единиц. Как следствие появились и стандартные единицы измерения: бит (*binary digit*) и байт (*byte*).

Бит – минимальная единица измерения информации – величина, которая может принимать одно из двух значений (в математическом представлении 0 или 1).

Байт – единица количества информации в системе СИ. Байт – восьмиразрядный двоичный код, с помощью которого наиболее часто представляют один символ текста (*о кодировании чисел и текста байтами рассказано ниже*).

Информационный объём сообщения (информационная ёмкость сообщения) – количество информации в сообщении, измеренное в битах, байтах или производных единицах (Кбайтах, Мбайтах и т. д.).

[13] Отдельные юридические аспекты в отношении информации будут рассмотрены в Главе 2.

1.3.2. Обозначение одного байта по ГОСТ 8.417–2002

Обозначение одного байта по ГОСТ 8.417–2002 «*Государственная система обеспечения единства измерений. Единицы величин*»: «байт» или «Б». При этом: 2^{10} байт = = 1024 байта = 1 Кбайт = 1 КБ; 2^{20} байт = 1 048 576 байт = 1 Мбайт = = 1 МБ = 2^{10} КБ = 1024 КБ = 1024 Кбайта и т. д.

Обратите внимание, что используется буква «К» (большая, заглавная, прописная), в то время как в международной системе единиц СИ (SI, фр. *Le **S**ystème **I**nternational d'Unités*, не имеет никакого отношения в языку Си (англ. C)) приставка «кило-», означающая 1000 = 10^3, в русском языке обозначается буквой «к» (маленькая, строчная). Чтобы запомнить этот нюанс, приведём короткий анекдот:

«*Физик ошибочно думает, что в одном килобайте (КБ) тысяча (1000) байт, а программист – что в одном килограмме (кг) тысяча двадцать четыре (1024) грамма.*»

В зарубежной практике сегодня используются рекомендации стандарта IEEE 1541–2002, согласно которым бит (англ. bit) обозначается как «b», а байт (анг. byte) – «B». При этом, чтобы не путать «килобайты с килограммами», используются приставки:

- kibi (обозн. «Ki») = 2^{10} = 1024;
- mebi (обозн. «Mi») = 2^{20} = 1 048 576;
- gibi (обозн. «Gi») = 2^{30} = 1 073 741 824 и др.

Надо заметить, что несмотря на цифру 2002 в названии, стандарт был встречен обществом «двояко» и лишь 27 марта 2008 г. был окончательно переутверждён как действующий. Скорее всего, он был «скопирован» с утверждённого ещё в 1999 г. МЭК стандарта IEC 60027-2. Отличие последнего в том, что в нём для обозначения бита вместо «b» предлагается использовать слово «bit».

Поскольку в массе своей покупатели и продавцы не сведущи в обозначениях сокращений по стандартам, благодаря стараниям маркетологов приставки kibi и прочие не используются – занижать визуально наблюдаемый объём невыгодно. Например, диск с этикеткой «1000 у. е.» будет продаваться лучше, чем с «932 ГБ», хотя в абсолютном исчислении 932*1024*1024*1024 даже чуть больше, чем 1000*1000*1000*1000. Не техническому специалисту ориентироваться в подобных нюансах сложно, потому как в «сегодняшних» прайс-листах и технических спецификациях (том, что у всех на виду) запросто можно увидеть ёмкости жёстких дисков в «попугаях» (*См. мультфильм «38 попугаев»*).

1.3.3. Методы и модели оценки количества информации

По мнению В. А. Острейковского [4]:

Для теоретической информатики информация играет такую же роль, как и вещество в физике. И подобно тому, как веществу можно приписывать довольно большое количество характеристик: массу, заряд, объём, и т. д., – так и для информации имеется пусть и не столь большой, но достаточно представительный набор характеристик единицы измерения, что позволяет некоторой порции приписывать числа – количественные характеристики информации.

На сегодняшний день известно много методов измерения информации, рассмотрим некоторые из них, в том числе наиболее часто встречающиеся:

- объёмный;
- энтропийный;
- алгоритмический.

1.3.3.1. Объёмный метод

Объёмный метод – является самым простым и грубым методом измерения информации. Соответствующую количественную оценку информации естественно назвать объёмом информации. Объём информации в сообщении – это количество символов в сообщении. Поскольку, например, одно и то же число может быть записано многими разными способами (с использованием различных алфавитов): «двадцать один», «twenty one», «einundzwanzig», 21, 11001_2, 15_{16}, XXI, то этот способ чувствителен к форме представления (записи) сообщения.[14]

1.3.3.2. Энтропийный метод (информация как снятая неопределённость)

В теории информации и кодирования применяется энтропийный метод измерения информации. Этот метод измерения исходит из следующей модели. Получатель информации имеет определённые представления о возможности наступления некоторых событий. Эти представления в общем случае достоверны и выражаются вероятностями, с которыми он ожидает то или иное событие (*например, сигнал – прим. авт.*). Общая мера неопределённости (энтропия) характеризуется математической зависимостью от совокупности этих вероятностей. Количество информации в сообщении определяется тем, насколько уменьшится эта мера после получения сообщения.

Создателем ***общей теории информации*** и основоположником цифровой связи считается Клод Шеннон [15]. Он ввёл понятие информационной энтропии – меры хаотичности информации, определяющей неопределённость появления какого-либо символа первичного алфавита. При отсутствии информационных потерь численно равна количеству информации на символ передаваемого сообщения.

Замечание. Приводимая в конце параграфа формула для информационной энтропии по своему внешнему виду была известна физикам со времён классических исследований Максвелла, Больцмана и Гиббса в области статистической механики. Больцман обозначил такой тип суммирования символом *H* и показал, что в идеализированных системах эта сумма пропорциональна термодинамической величине энтропии. Те же исследователи понимали, что существует тесная связь между энтропией и информацией; однако количественной теории информации они не разработали. Они и не могли её разработать, ибо в те времена не приходилось сталкиваться с

[14] В вычислительной технике вся обрабатываемая и хранимая информация вне зависимости от её природы (число, текст, отображение, изображение и прочее) представлена в двоичной форме (с использованием алфавита, состоящего из двух символов 0 и 1). Такая стандартизация позволила ввести две стандартные единицы измерения: бит и байт. Байт – это восемь бит. Подробнее см. раздел «Представление информации в компьютере».

[15] Клод Э́лвуд Ше́ннон (англ. *Claude Elwood Shannon*; 1916–2001) – американский инженер и математик, его работы являются синтезом математических идей с конкретным анализом чрезвычайно сложных проблем их технической реализации. Является основателем теории информации, нашедшей применение в современных высокотехнологических системах связи. Шеннон внёс огромный вклад в теорию вероятностных схем, теорию автоматов и теорию систем управления – области наук, входящие в понятие «кибернетика». В 1948 году в статье «*Математическая теория связи*» (*A Mathematical Theory of Communication*) ввёл термин «*бит*» для обозначения наименьшей единицы информации и обосновал возможность применения двоичного кода для передачи информации, что и принесло ему всемирную известность.

передачей информации техническими средствами.[10] Вполне естественно, что Шеннон использовал те же обозначения, которые сейчас приняты повсеместно.

Количеством информации называют числовую характеристику сигнала, не зависящую от его формы и содержания и характеризующую неопределённость, которая исчезает после получения сообщения в виде данного сигнала – в этом случае количество информации зависит от вероятности получения сообщения о том или ином событии. Оценка количества информации основывается на законах теории информации.

Рассмотрим пример [4], предложенный В. А. Острейковским:

Пусть имеется колода карт, содержащая 32 различные карты. Вероятность выбора одной карты из колоды равна 1/32. Априори (доопытно, до произведения выбора) естественно предположить, что наши шансы выбрать некоторую определённую карту одинаковы для всех карт колоды. Произведя выбор, мы устраняем эту априорную неопределённость. Нашу априорную неопределённость можно было бы охарактеризовать количеством возможных равновероятностных выборов. Если теперь определить количество информации как меру устраненной неопределённости, то и полученную в результате выбора информацию можно охарактеризовать числом 32.

Однако в теории информации получила использование другая количественная оценка H, рассчитываемая по формуле Ральфа Хартли [16], а именно – логарифм от описанной выше оценки по основанию 2:

$$H = \log_2 m,$$

где m – число возможных равновероятных выборов (при $m = 2$, $H = 1$). То есть для выбора из колоды имеем следующую оценку количества информации, получаемую в результате выбора:

$$H = \log_2 32 = 5.$$

Полученная оценка имеет интересную интерпретацию (на примере урезанной игральной колоды из 32 карт: «7», «8», «9», «10», «Валет», «Дама», «Король», «Туз» четырёх привычных мастей (♠, ♣, ♥, ♦). Она характеризует число «двоичных» вопросов, ответы на которые позволяют выбрать либо «да», либо «нет». Например, для выбора дамы пик такими вопросами могут быть:

№	Вопрос	Ответ	Интерпретация ответа
1	Карта красной масти?	Нет	0
2	Трефы?	Нет	0
3	Одна из четырёх старших?	Да	1
4	Одна из двух старших?	Нет	0
5	Дама?	Да	1

Таким образом, чтобы устранить неопределённость (сделать выбор нужной карты) в колоде карт, необходимо получить в диалоговом режиме пять вопросов – пять от-

[16] Ральф Хартли (англ. Ralph Vinton Lyon Hartley, 1888–1970) – американский учёный-электронщик. Он предложил генератор Хартли, преобразование Хартли и сделал вклад в теорию информации, введя в 1928 году логарифмическую меру $H = K \cdot \log_2 N$ (в нашем случае $K = 1$), которая называется хартлиевским количеством информации, содержащейся в сообщении.

ветов (сообщений). По колонке интерпретации ответов этот выбор можно описать последовательностью из пяти двоичных символов 00101. Количество информации такого сообщения составляет 5 бит (равно числу вопросов в нашем случае).

Сообщение, уменьшающее неопределённость знаний в два раза, несёт 1 бит информации.

На первый взгляд может показаться, что эта интерпретация не годится в случае, когда количество выборов не равно степени двойки, так как получается нецелое количество вопросов, к примеру если взять колоду из 36 карт (добавлены шестёрки), то можно заметить, что для того, чтобы выяснить у участника «эксперимента», какую карту он выбрал, в ряде случаев понадобится 5 вопросов, как и в предыдущем случае, а в ряде случаев – и 6 вопросов. Усреднение по случаям и даёт получаемую по формуле нецелую величину.

Двоечник и отличник

Рассмотрим ещё один пример. Допустим, что имеется двое обучаемых с обоснованно присвоенными общественными характеристиками «двоечник» и «отличник». Мы подсчитали количество полученных ими оценок за учебный год и занесли данные в таблицу.

Оценки	«Двоечник»	«Отличник»
«Отлично» (5)	3	85
«Хорошо» (4)	15	9
«Удовлетворительно» (3)	20	5
«Неудовлетворительно» (2)	60	1
«Плохо» (1)	2	0

Так получилось, что суммарно они получили каждый по 100 оценок (это ни на что не влияет, но зато упростит наши последующие расчёты). Подсчитаем некоторые характеристики, а именно вероятности получения оценок. Приближённо [17] вероятностью будем считать отношение числа тех или иных оценок, поделённое на общее число выставленных оценок в год. В итоге получим:

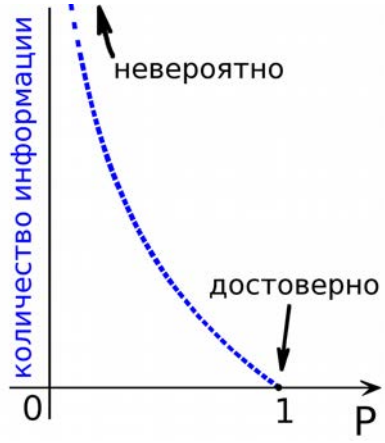
Вероятность какого события	«Двоечник»	«Отличник»
вероятность получения пятёрки	3/100 =0,03	85/100 = 0,85
вероятность получения четвёрки	15/100 =0,15	9/100 = 0,09
вероятность получения тройки	20/100 = 0,2	5/100 = 0,05
вероятность получения двойки	60/100 = 0,6	1/100 = 0,01
вероятность получения единицы	2/100 = 0,02	0/100 = 0

[17] Мы предполагаем, что результаты опыта сводятся к схеме случаев, при которой вероятность события A вычисляется по формуле $P(A) = m/n$, где n – общее число случаев, а m – число случаев, благоприятных событию A (См. стр.5. *Вентцель Е. С., Овчаров Л. А. Теория вероятностей. – 2-е изд., стер. – М.: «Наука» Главная редакция физико-математической литературы, 1973 – 368 с.: ил.*).

Логично предположить, что сообщение вида «*Мама, я получил оценку 5 по информатике*» от отличника ожидаемо и несёт мало новой информации для родителей. В то же время, если то же самое скажет двоечник, в его семье будет праздник. То есть:

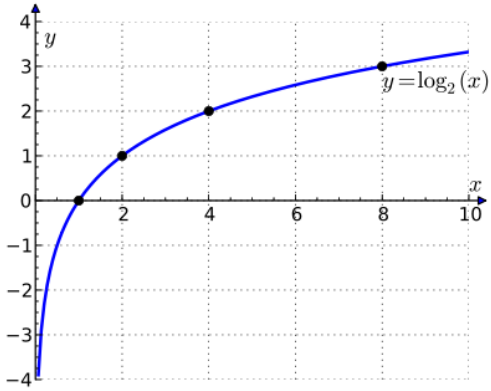
чем выше вероятность события, тем меньше информации оно несёт	и	чем ниже вероятность события, тем больше информации оно несёт.

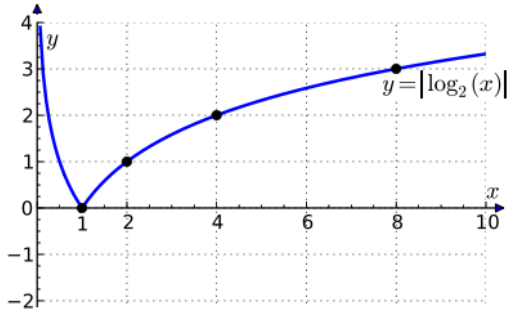
Сам факт того, что ребёнком получена оценка в школе обычно родителей не интересует так сильно как то, какая именно. Этому есть объяснение. Если предположить что в какой-то школе «Ы» оценки ставят ежедневно (или еженедельно), то наличие поставленной оценки за отведённый период означает, что учитель обязательно какую-то оценку да поставил. Или, вероятностным языком, произошло одно из пяти событий: «получена пятёрка», «получена четвёрка», «получена тройка», «получена двойка», «получена единица». Как вы понимаете, раз других оценок быть не может, то такое событие (как выбор одной оценки из возможных) будет достоверным событием, вероятность которого можно принять за единицу, а информацию от этого сообщения можно считать нулевой.



Невероятное событие – это то, у которого вероятность стремиться к нулю, а количество полученной при этом информации велико (на графике асимптотически стремиться к бесконечности). Скажем, прилетели инопланетяне и поэтому все уроки отменили.

Мы не берёмся оценить (как и доказать) правдоподобность степени кривизны отображённой нами линии, как и сделать выводы относительно её выпуклости (впуклости). Предположим, что наиболее подходящей по ряду качеств будет функция «модуль логарифма». См. Рисунок 1.1.





как видно, $y=|\log_2(x)|=-\log_2(x)$,
при $x \in (0,1]$

Рисунок 1.1. Графики логарифма и модуля логарифма

В защиту нашего предположения о выборе в пользу логарифма вероятности приведём взятые из теории информации следующие доводы:

1. Необходимо, чтобы количество информации было аддитивно. Например если у нас последовательно переданы два независимых сообщения x_1 и x_2, то количество информации в суммарном сообщении должно равняться сумме количества информации каждого сообщения по отдельности.

2. Количество информации содержащееся в достоверном сообщении равно нулю (новую информацию заранее известное сообщение не несёт).

3. Информационная мера не должна зависеть от степени восприятия информации конкретным получателем (информация должна быть объективной).

Таким образом, наши интуитивные представления можно описать более строго.

Определим количество информации h_i, содержащееся в каждом сообщении о полученной оценке, исходя из предложенной выше статистики оценок. Для этого напомним свойство логарифма (известное читателям из школьного курса алгебры), что $\log_b \frac{1}{a} = \log_b a^{-1} = -\log_b a$ и воспользуемся формулой $h_i = \log_2\left(\frac{1}{p_i}\right)$,

где p_i – вероятность получения i-й (по значению, а не порядку) оценки.

Таким образом, количество информации [18]:

Событие	«Двоечник»	«Отличник»
получена оценка 5	$log_2(1/0{,}03) \approx 5{,}059$	$log_2(1/0{,}85) \approx 0{,}234$
получена оценка 4	$log_2(1/0{,}15) \approx 2{,}737$	$log_2(1/0{,}09) \approx 3{,}474$
получена оценка 3	$log_2(1/0{,}2) \approx 2{,}322$	$log_2(1/0{,}05) \approx 4{,}322$
получена оценка 2	$log_2(1/0{,}6) \approx 0{,}737$	$log_2(1/0{,}01) \approx 6{,}644$
получена оценка 1	$log_2(1/0{,}02) \approx 5{,}644$	$log_2(1/0) = \infty$

Как видим, ранее предположенное утверждение, что количество информации в сообщении о некотором событии зависит от вероятности этого события, подтвердилось. Чем меньше вероятность, тем больше информации.

Обычно передаваемое сообщение не состоит из одного передаваемого символа. Оно может состоять из нескольких независимых символов (появление символа – событие), тогда ***информационная энтропия*** для m независимых случайных событий с n возможными состояниями (от 1 до n) рассчитывается по формуле Шеннона:

$$H(m) = -m\sum_{i=1}^{n} p(i)\log_2 p(i) \quad ,$$

где $p(i)$ – вероятность i-го события. Пользоваться этой формулой в отношении оценки количества содержащейся информации в используемых нами словах и предложениях (текстах) невозможно поскольку буквы в них оказываются зависимыми друг от друга.

Замечание 1. Откуда в формуле взялся минус? – См. Рисунок 1.1.

Замечание 2. В примере выше основание логарифма равно двум, но в ряде случаев оно может не указываться, поскольку это не существенно в целом. Изменение основания приведёт лишь к изменению размерности.

[18] Если у вас под рукой не оказалось калькулятора или логарифмической линейки, то предлагаем воспользоваться сайтом http://kalkulyatoronline.ru/.

1.3.3.3. Алгоритмический метод

В теории информации существует алгоритмический метод оценки количества информации в сообщении. В. А. Острейковский предлагает этот метод охарактеризовать следующими рассуждениями.

Под алгоритмом всегда понималась процедура, которая позволяла путём выполнения последовательных элементарных шагов (действий) получать однозначный результат (независимо от того, кто выполнял эти шаги) или за конечное число шагов прийти к выводу о том, что решение не существует.

Каждый согласится, что слово 0101010101…01 сложнее слова 0000000000…00, а слово, где 0 и 1 выбираются из эксперимента – бросания монеты (где 0 – герб, 1 – аверс[19]), сложнее обоих предыдущих.

Компьютерная программа, производящая слово из одних нулей, крайне проста: печатать один и тот же символ. Для получения 010101 … 01 нужна чуть более сложная программа, печатающая символ, противоположный только что напечатанному. Случайная, не обладающая никакими закономерностями последовательность не может быть произведена никакой «короткой» программой. Длина программы, производящей хаотическую последовательность, должна быть близка к длине последней.

Приведённые рассуждения позволяют предположить, что любому сообщению можно приписать количественную характеристику, отражающую сложность программы, которая позволяет её произвести.

Так как имеется много разных вычислительных машин и разных языков программирования (разных способов задания алгоритма), то для определённости задаются некоторой конкретной вычислительной машиной, например машиной Тьюринга [20].

1.3.4. Семантическая (смысловая) мера информации

정보의 의미론적 척도는 전술한 정보측정방법에 포함시킬 수 없는데, 의미론적 수준에서의 정보량 추정의 결과는 정보를 받는 사람, 예를 들어 객관적이지 않은 상황을 제시하는 특정한 사람의 특성에 의존하기 때문이다.

Если вы пропустили предыдущий абзац и начали сразу читать данную строчку, то вы попались на уловку придуманную авторами, ещё раз подтвердив истинность утверждения, что:

Семантическая мера информации не может быть отнесена к вышеописанным методам измерения информации, поскольку результат оценки количества информации на семантическом уровне зависит от свойств её получателя, например конкретного человека, что даёт не объективную картину.

Если среди читателей, их родственников или знакомых случайно найдутся знатоки корейского языка, то они подтвердят, что первый и третий абзацы данного параграфа идентичны по смыслу, то есть несут одинаковое количество информации[21] для чита-

[19] Аве́рс (фр. *avers*, лат. *adversus* – «обращённый лицом») – лицевая, главная сторона монеты.

[20] Маши́на Тью́ринга – абстрактный исполнитель (абстрактная вычислительная машина). Была предложена Аланом Тьюрингом в 1936 году для формализации понятия алгоритма.

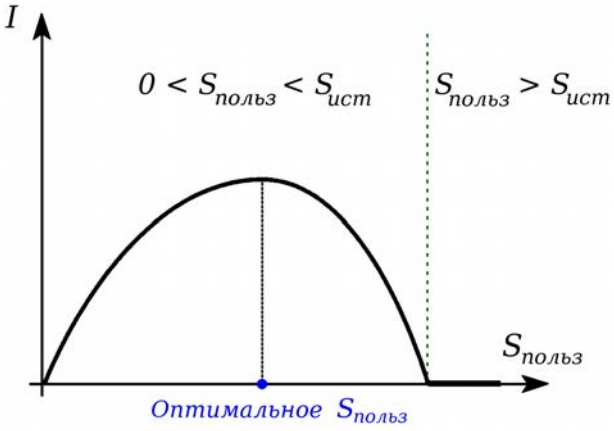
[21] Заметим, что из равенства количества информации на семантическом уровне (в нашем случае у двух предложений одинаковый смысл) не следует, что объёмы памяти ЭВМ занимаемые этими же предложениями будут равны. Подробнее о кодировании текстовой информации см. параграф 2.2. Представление текстовой информации в ЭВМ.

теля, однако воспринять эту информацию получится не у всех. Подумайте на досуге, что бы могла означать используемая в обиходе фраза «читайте между строк» с научной точки зрения.

По мнению Акулова О.А., Н.В.Медведева [5] и Макаровой Н.В., Волкова В.Б. [6], для измерения смыслового содержания информации, то есть её количества на семантическом уровне, наибольшее признание получила тезаурусная мера, которая связывает семантические свойства информации со способностью пользователя понимать поступившее сообщение. Для этого используется понятие тезауруса пользователя.

Теза́урус [22] – множество смысловыражающих единиц некоторого языка с заданной на нём системой семантических отношений. Тезаурус фактически определяет семантику языка (национального языка, языка конкретной науки или формализованного языка для автоматизированной системы управления) [7]. В отличие от толкового словаря, тезаурус позволяет выявить смысл не только с помощью определения, но и посредством соотнесения слова с другими понятиями и их группами. Более коротко:

Тезаурус – это совокупность сведений, которыми располагает пользователь или система [6].



В зависимости от соотношения между смысловым содержанием информации источника $S_{ист}$ и тезаурусом пользователя $S_{польз}$ изменяется количество семантической информации I, воспринимаемой пользователем (и включаемой им в дальнейшем в свой тезаурус) [5, 6].

Если зафиксировать $S_{ист}$, то характер зависимости $I(S_{польз})$ представлен на Рисунке 1.2.

Рисунок 1.2. Зависимость количества воспринимаемой пользователем информации I от его тезауруса $S_{польз}$ ($S_{ист}$ – const)

Отметим, что когда тезаурус пользователя стремится к нулю, то и воспринимаемая им информация также стремится к нулю. Согласитесь, что попытка объяснить высшую математику только научившемуся говорить двухлетнему ребёнку практически бесполезное занятие.

Также, когда $S_{польз} \geq S_{ист}$, источник не несёт новой информации пользователю, как если рассказывать про количество букв в алфавите или про таблицу умножения выпускникам школы.

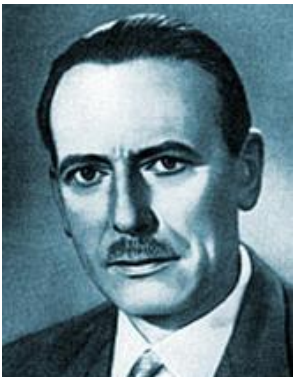
Максимальное количество семантической информации потребитель приобретает при согласовании её смыслового содержания со своим тезаурусом (См. значение «*Оптимальное $S_{польз}$*» на Рисунке 1.2 соответствующее экстремуму графика), когда поступающая информация понятна пользователю и несёт ему ранее неизвестные (отсутствующие в его тезаурусе) сведения [5, 6].

[22] От греч. θησαυρός (thēsaurós) – сокровище, сокровищница

Вывод: Народное шутливое утверждение, что «*Гениальному писателю нужен гениальный читатель.*» возможно подвергнуть сомнению, поскольку последний, если окажется умнее первого, может заскучать при чтении, так как не откроет для себя ничего нового.

1.3.5. Прагматическая мера информации (мера полезности)

После прочтения раздела подумайте, почему в народе говорят: «Гусь свинье не товарищ» или «Дорога ложка к обеду»?



Александр Александрович Харкевич (1904-1965), советский учёный в области электросвязи и электроакустики. Его «Очерки общей теории связи» (1955) стали одной из основополагающих работ в теории информации.

Довольно интересный подход к измерению меры полезности информации был предложен отечественным учёным А. А. Харкевичем, который предложил принять за меру ценности информации приращение вероятности достижения некоторой цели. Зачастую это более понятно обывателю, нежели какая-то там вероятность сообщения или энтропия.

Например, вы школьник средних классов и мечтаете лечить людей от болезней. Это ваша цель. Для её достижения у вас может быть несколько путей, заняться фундаментальными исследованиями в медицине и в итоге создавать новые лекарства, придумывать новые методы лечения болезней, либо стать врачом и использовать уже имеющиеся препараты и методы. В любом случае, в старших классах вам предстоит сделать выбор своего жизненного пути и, возможно, по окончании школы поступить в какой-нибудь вуз. Ваш друг, как ему кажется, напротив, далёк от медицины, интересуется компьютерами и планирует поступать в какой-нибудь технический университет. Он сообщает вам, что ездил на выходных в интересующий его вуз и рассказывает подробную информацию о работе в нём с абитуриентами.

Если не брать в рассмотрение социальные и психологические аспекты общения людей, как то желание двух друзей поговорить, умение преподнести информацию, вопросы вежливости и т.п., то сообщённая вам информация скорее всего будет бесполезна, поскольку она не приблизит вас к вашей цели, хотя она будет вам полностью понятна (воспринята), полна, достоверна, актуальна и т.д. Объективно ситуация видится следующим образом: Если до получения сообщения вероятность достижения вами цели равнялась p_0, а после её получения p_1, то ценность информации определяется как:

$$I = lop\, p_1 - \log p_0 = \log \frac{p_1}{p_0}$$

Причём эта формула допускает и отрицательную ценность информации, в случае снижения вероятности достижения цели после приёма сообщения. В конкретном случае это может произойти по той причине, что ваш друг убедит вас пойти с ним учиться на программиста в его вуз за компанию, а курсе на третьем, например, он женится, ста-

нет семейным человеком и будет общаться с вами реже, а вы вспомните, что первоначальная жизненная цель у вас была совсем другая.

В общем случае, отрицательная ценность может говорить о том, что сообщение содержит дезинформацию. Создана такая ситуация кем-то специально или так получилось случайно, – это уже другой вопрос.

Дальнейшее развитие данного подхода базируется на статистической теории информации и теории решений. Сущность метода состоит в том, что, кроме вероятностных характеристик неопределённости объекта, после получения информации вводятся функции штрафов или потерь, и оценка информации производится с учётом возможной минимизации потерь. Максимальной ценностью обладает информация такого сообщения, которое уменьшает потери до ноля при достижении поставленной цели.

1.4. Заключение

Всё в жизни крутится вокруг нас, людей, даже информация измеряется относительно нас и для нас. Иначе откуда взяться таким расплывчатым понятиям как «обилие информации», «избыток информации», «информационная зависимость», «информационная диета» или «гиперинформированность»?

Обсуждение различных информационных вопросов с гуманитарной точки зрения не менее интересно, но к сожалению выходит за возможности данного учебника, поэтому лишь порекомендуем читателям самостоятельно изучить интересное на наш взгляд интервью с Алексеем Козыревым[23] по теме «Изобилие информации: это благо или бремя?» [11].

Что же вопросов ***измерения информации***, то в жизни мы чаще имеем дело не с абстрактной информацией, а с конкретными физическими объектами и явлениями вокруг нас, поэтому все наши интересы сводятся к измерению тех или иных физических величин. Прямо или косвенно. Если не углубляться в вопросы квантовой физики и метрологии, то измерения получаются двух видов:

- либо счётные (точно измеримые): например 2 карандаша, один ластик;
- либо непрерывные – измеряемые примерно.

С первыми обычно всё понятно, а вот со вторыми возникают трудности. Информация как характеристика какого-либо физического объекта, например о цвете яблока созревающего на дереве, сильно зависит от времени получения этой информации. Поэтому нам важно понять (договориться) когда и как измерять. Смеем предположить, что в июне и в июле результаты будут разными. А ёлка, как мы знаем с детства, наоборот, зимой и летом окажется одним цветом. По сути, в примере с яблоком, как цвет есть непрерывная величина плавно переходящая от зелёного к красному[24], так и момент его измерения может быть растянут во времени. Вот тут и возникает трудность: как «бинарному» по своей сути компьютеру, используя «нули» и «единицы», записать в себя что-то непрерывное (бесконечное) и желательно абсолютно точно? Это невозможно, но, поскольку очень хочется, можно приблизиться и значение физической величины (способной принимать бесконечное множество значений на некотором диапа-

[23] Деканом философского факультета МГУ на момент интервьюирования.
[24] Будем считать, что при созревании яблоко окрашивается равномерно по всем направлениям.

зоне[25]) брать (замерять) в определённые моменты времени с конечной точностью, заведомо огрубляя измеренное значение. Часто это вполне допустимо. Измерили один раз, скажем 12 июля, а цвет определили как ближайший к семи цветам радуги: «*в день Петра и Павла яблоко было красным*».

Момент измерения. И так, при измерении аналоговых величин измерение может быть дискретным: разовым или взятым счётное число раз, а может быть непрерывным (по времени). Последнее возможно в теории информации, но невозможно для цифровых ЭВМ исходя из их природы.

Независимо от момента измерения, ***измеренное значение***, аналогично, может быть точным (в теории и не существующим в понятии цифровой ЭВМ), а может быть дискретным, то есть одним значением из набора возможных состояний.

Выше было оценено количество информации содержащееся в дискретном сообщении о полученной оценке. Понятно, что и энтропия считалась для дискретного источника. Энтропию для непрерывного источника какого-либо сигнала мы сознательно не стали рассматривать в учебнике. То есть, мы сознательно упростили подачу материала, если необходимо подробнее, то см. [12, 13]

В последующей главе мы обсудим как можно осуществить кодирование, то есть перевод «в нули и единицы» понятных ЭВМ дискретных и аналоговых сообщений. Дискретными будут цифры (счётные множества) и тексты, а аналоговыми, – множество чисел $\mathbb{R}$, изображение, звук (в общем случае аналоговый сигнал).

Мы считаем последующую главу важной, поскольку согласно одному из принципов, сформулированных Джоном фон Нейманом, вся информация, которую хранят, обрабатывают и передают по сетям современные ЭВМ (будь то персональные компьютеры, планшеты, телефоны, MP3-плееры, цифровые фотоаппараты, очки, электронные карты и прочие «гаджеты»), представляется в виде двоичных чисел, фактически последовательностью «0» и «1».

Замечание о выборе основания для системы кодирования или «почему двоичная система удобна?»

Перед тем как вы перейдёте к чтению следующей главы, о кодировании различной информации, всё же следует сказать несколько слов в пользу выбора основания кода в котором будет производиться кодирование.

Пытаясь хоть как-то обосновать свой выбор наверно следует поставить вопрос формально: «какая система счисления (а значит и код на её базе) наиболее экономична в записи?» и ответить на него доказательством теоремы «о представлении некоторого числа n минимальным набором символов в определённой системе счисления». Затем, найдя экстремум функции $f(x)=x^{\frac{n}{x}}$ в точке $x = e = 2.718281828...$, как это сделал в своё время Джон фон Нейман, заняться выбором между 2 и 3.

Отметим, что интерес к троичной логике и арифметике возник задолго до появления первых компьютеров в связи с замечательными свойствами симметричного кода чисел. В последнее время этот интерес отчасти возрождается, во многом благодаря новым возможностям микро- и нано- электроники.

Несмотря на различные положительные аргументы высказанные в своё время Н.П.Брусенцовым (создателем первого уникального троичного компьютера «Сетунь» не только в СССР, но и в мире) в отношении выбора троичного кода и троичной логики, мы всё же выберем двоичную основу для кодирования, поскольку:

[25] Математики для оперирования такими величинами используют специальный термин - континуум.

Во-первых, большинство современных систем используют двоичное кодирование и наша цель слегка разобраться как они работают.

Во-вторых, можно привести следующие аргументы (См. [14, § 5, стр. 14–15]): главное достоинство двоичной системы – простота алгоритмов сложения, вычитания умножения и деления. Таблица умножения в ней совсем не требует ничего запоминать: ведь любое число, умноженное на нуль, равно нулю, а умноженное на единицу, равно самому себе. И при этом никаких переносов в следующие разряды, а они есть даже в троичной системе. Таблица деления сводится к двум равенствам $0 / 1 = 0$, $1 / 1 = 1$, благодаря чему деление столбиком многозначных двоичных чисел делается гораздо проще, чем в десятичной системе, и по существу сводится к многократному вычитанию.

Таблица сложения, как ни странно, чуть сложнее, потому что $1 + 1 = 10$ и возникает перенос в следующий разряд. В общем виде операцию сложения однобитовых чисел можно записать в виде $x + y = 2w + v$, где w, v – биты результата. Внимательно посмотрев на таблицу сложения, можно заметить, что бит переноса w – это просто произведение xy, потому что он равен единице, лишь когда x и y равны единице (можно записать $w = x$ AND y). А вот бит v равен $x + y$, за исключением случая $x = y = 1$, когда он равен не 2, а 0. Операцию, с помощью которой по битам x, y вычисляют бит v ($v = x$ XOR y), называют по-разному («XOR», «исключающее или»). Мы будем использовать для неё название «*сложение по модулю 2*» и символ ⊕ ($v = x \oplus y$). Таким образом, сложение битов выполняется фактически не одной, а двумя операциями. Если отвлечься от технических деталей, то именно с помощью этих операций и выполняются все операции в компьютере.

Для выполнения сложения однобитовых чисел обычно даже создают специальный отдельный логический элемент с двумя входами x, y и двумя выходами w, v, как бы составленный из элемента умножения (его часто называют конъюнкцией, чтобы не путать с умножением многозначных чисел) и элемента сложения по модулю 2. Этот элемент часто называют полусумматором.

Подробнее все эти преимущества будут заметны при рассмотрении построения аппаратной части ЭВМ на базе элементов логики в Главе 4.

На этом завершим первую главу следующим афоризмом:

Все люди делятся на 10 типов:
те, кто понимают двоичную систему счисления, и те, кто нет [25].

1.5. Контрольные вопросы к главе 1

1. От каких слов произошло слово «информатика», какой у него смысл?
2. Как называется информатика в англоязычных странах?
3. В каких годах термин «кибернетика» постепенно перестал использоваться?
4. Перечислите основные признаки информационного общества.
5. Что такое информатизация и какими причина она обусловлена?
6. Существует ли общепринятое научное определение информации?
7. Что понимается под информацией российским законодательством?
8. Как термин «информация» трактовал Сергей Иванович Ожегов?
9. Что такое «информация», какие виды её вы знаете?
10. Какими свойствами обладает информация?
11. Какие процессы могут происходить с информацией?
12. Какие существуют подходы (оценки) по измерению объёма компьютерной информации?
13. Как обозначается байт и его производные по ГОСТ 8.417–2002?

14. В чём сущность энтропийного метода оценки количества информации?
15. Как выглядит и зачем нужна формула Ральфа Хартли?
16. Как выглядит и зачем нужна формула Шеннона?
17. Объясните смысл минуса перед логарифмом вероятности?
18. Как можно измерить (оценить) смысловое содержание информации?
19. В каких случаях, в контексте количества информации на семантическом уровне, студенты начинают скучать на лекциях?
20. Как, в контексте преподнесения информации, должен читать лекцию профессор, заинтересованный в том, чтобы его слушали?
21. Что есть прагматическая мера информации и кем она была предложена?
22. Как в рамках прагматической меры информации можно оценить народную поговорку «гусь свинье не товарищ»?
23. Какую формулу в отношении информации предложил использовать Александр Александрович Харкевич?

1.6. Литература к главе 1

1. *Винер Н.* Кибернетика, или Управление и связь в животном и машине / пер. с англ. И. В. Сольовьёва и Г. Н. Поварова, под ред. Г. Н. Поварова. – 2-е изд. – М.: Сов. радио, 1968. – 328 с.
2. *Острейковский В. А.* Информатика: учеб. для вузов / В. А. Острейковский. - 4-е изд., стер. - М.: Высш. шк., 2007. - 511 с.: ил. ISBN 978-5-06-003533-9.
3. *Беликов Е. П.* Об организации в Академии наук СССР работ по информатике, вычислительной технике и автоматизации // Вестник АН СССР. – 1983. – № 6.
4. *Грошев А. С.* Информатика: учеб. для вузов / А. С. Грошев. – Архангельск,: Арханг. гос. техн. ун-т, 2010. – 470 с. ISBN 978-5-261-00480-6.
5. *Акулов О. А., Медведев Н. В.* Информатика. Базовый курс. Учебник. – М.: Омега-Л, 2006. – 560 с.: ил. ISBN 5-365-00293-8.
6. *Макарова Н. В., Волков В. Б.* Информатика: Учебник для вузов. – СПб.: Питер, 2011. – 576 с.: ил. ISBN 978-5-496-00001-7.
7. Большая советская энциклопедия. Том 25. – Москва: Советская энциклопедия, 1976.
8. *Федотова Е. Л., Федотов А. А.* Информатика. Курс лекций: учеб. пособие. – М.: ИД «ФОРУМ»: ИНФРА-М, 2011. – 480 с.: ил. ISBN 978-5-8199-0448-0, 978-5-16-004571-9.
9. *Бекман И. Н.* Информатика. Курс лекций. – М.: Рим – 2009, http://profbeckman.narod.ru/InformLekc.htm.
10. *Голдман С.* Теория информации / пер. с англ. Б. Г. Белкина, под ред. В. В. Фурдуева. – М.: Издательство иностранной литературы, 1957. – 446 с.
11. *Выжутович В.* Изобилие информации: благо или бремя? // Российская газета. – 2014. – № 129 (6401), http://www.rg.ru/2014/06/10/kozyrev.html.
12. *Яглом А.М., Яглом И.М.* Вероятность и информация – М.: Наука, 1973. – 512 с.
13. *Голдман С.* Теория информации / пер. с англ. Б. Г. Белкина, под ред. В. В. Фурдуева. – М.: Издательство иностранной литературы, 1957. – 446 с.
14. *Гашков С. Б.* Системы счисления и их применение. – М.: МЦНМО, 2004. ISBN 5-94057-146-8.

Глава 2. Представление информации в компьютере

Материал, изложенный в данной главе, важен для понимания основ работы современных ЭВМ. Он будет полезен не только будущим программистам, но и системным администраторам и прочему техническому персоналу.

Существуют международные стандарты и методы кодирования числовой, текстовой, звуковой, графической и видеоинформации в ЭВМ. Знание основных принципов кодирования очень важно не только для правильного «чтения» информации, но и для компактного её хранения в памяти ЭВМ. Обсуждению форматов представления чисел в памяти компьютера посвящено большое количество литературы. Вопросы, касающиеся систем счисления, частично изучаются даже в средней школе. Уровень изложения при этом обычно поверхностно-ознакомительный из-за чего может сформироваться ложное представление, что материал прост и понятен, а обучаемый, основываясь на своей интуиции, может самостоятельно разобраться во всех нюансах. Позднее, при программировании, а тем более администрировании, данному вопросу не уделяется достаточно времени, а если и уделяется, то ознакомленными с данной темой оказываются не все, а отдельные программисты и математики. Уровень изложения материала оказывается высоким (пестрит ссылками и формулами), многие моменты и доказательства опускаются как очевидные, а язык изложения всё чаще оказывается английским.

Первая часть главы (раздел 2.1) по представлению чисел в памяти ЭВМ нацелена на то, чтобы заполнить нишу между школьным курсом информатики и уровнем компьютерных стандартов. В отличие от встречающихся в Сети (и литературе) материалов по данной теме, наш материал дополнен примерами программ[26], с помощью которых вы, читатели, можете самостоятельно на практике проверить изложенные теоретические сведения. Если вы не умеете программировать, не пугайтесь кода программ, а используйте их, как если бы они были теоремы без доказательства. Представьте, что вы не знаете (или подзабыли) как доказать теорему Пифагора, это ведь не значит, что вы не сможете ею воспользоваться. Для читателей слабо знакомых с операционной системой (ОС) Linux дано пояснение как компилировать программы и запускать их из командной сроки указанной ОС.

Во второй части главы (начиная с раздела 2.2) приводится информация о существующих текстовых кодировках и о том, как звук, графика и видео превращаются в памяти ЭВМ в «нули» и «единицы».

Два примера к разделу 2.1. (вместо предисловия)

Рассмотрим две простые программы[27] на языке С. Если вы заранее можете правильно сказать, что они выведут, то смело переходите к разделу 2.2.

Первая программа считает разницу двух целых чисел 123456789 и 123456788, вторая – вызывает в цикле функцию (в нашем случае печати) со значениями x от $x_н = 0{,}1$ до $x_к = 0{,}3$ с шагом 0,1.

[26] Тексты программ доступны для скачивания на сайте http://learn2prog.ru в разделе «Информатика».
[27] Представленные в том виде как есть, без оценки их правильности, корректности, нужности и т.п.

Программа 1. Вычитание чисел.

```
$ cat 123456789-123456788.c
#include <stdio.h>

void main (void)
{
    float a,b,f;
    a=123456789;
    b=123456788;
    f=a-b;
    printf ("a-b=%f\n",f);
}
$ gcc 123456789-123456788.c &&./a.out
a-b=8.000000
```

Программа 2. Цикл.

```
$ cat for_0.1_0.3.c
#include <stdio.h>

void main (void)
{
  float x=0.1;
  for ( ; x<=0.3; x+=0.1)
  {
    printf ("x=%f\n",x);
  }
}
$ gcc for_0.1_0.3.c && ./a.out
x=0.100000
x=0.200000
```

Если у вас другое мнение (неправильное) по результатам работы программ, то понять почему в первом случае разница оказалась равна восьми и куда исчезла третья строчка с выводом x=0.3 вы сможете, если изучите раздел 2.1.

2.1. Основные теоретические сведения представления чисел в ЭВМ

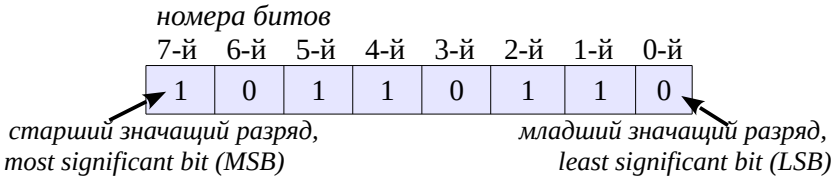
Примем как данное, что к моменту появления компьютерной техники (вычислительных машин и конечных автоматов) человечество уже использовало такие понятия, как «число», «натуральное число» ($\mathbb{N}$, $\mathbb{N}+$) и «целое число» ($\mathbb{Z}$), известные читателям ещё из школьного курса математики, в связи с чем и возникла потребность в частичном или полном представлении обозначаемых ими объектов в виде каких-то состояний внутри аппаратуры ЭВМ. В этой главе мы не будем рассматривать аппаратное устройство узлов и блоков ЭВМ, а также вникать в то, как осуществляются математические операции над числами. (Примем как данное один из принципов Джона фон Неймана[28] об использования двоичной системы счисления по представлению информации в памяти ЭВМ, из которого вытекает, что все хранимые числа в конечном итоге должны быть представимы в виде набора «0» и «1».) Попытаемся порассуждать на тему, как было бы разумнее и эффективнее хранить конечные множества чисел в памяти.

Замечание. В параграфе 2.1.1 рассматривается представление конечного подмножества множества целых чисел $\mathbb{N}$ и нуля (представление положительных чисел), а также конечное подмножество множества целых чисел $\mathbb{Z}$ (представление положительных и отрицательных чисел вместе). Не будут рассматриваться не целые числа, представимые как действительные (вещественные) $\mathbb{R}$ (рациональные $\mathbb{Q}$ и иррациональные $\mathbb{I} = \mathbb{Q}\backslash\mathbb{R}$), комплексные числа ($\mathbb{C}$), кватернионы ($\mathbb{H}$) и другие числа. ЭВМ, работающие в троичной (например, «Сетунь»), десятичной (например, «ENIAC») и иных системах счисления, также рассматриваться не будут. Представление вещественных чисел рассматривается ниже в подтеме «*Числа с плавающей точкой/запятой согласно стандарту IEEE754*» (см. стр. 48).

[28] Подробнее о принципах вычислительной машины Джона фон Неймана см. стр. 228.

2.1.1. Положительные целые числа (машинное представление без знаковых целых чисел)

Обычно числа располагаются в памяти компьютера в двоичном формате в последовательных ячейках памяти, при этом минимальный набор двоичных ячеек, к которому применимы операции адресации (как следствие и другие операции), составляет 8 двоичных ячеек, или 8 бит.



Ранее такой набор назывался машинным словом и имел размер 1 байт = 8 бит. Сейчас размер машинного слова больше и зависит от характеристик аппаратной части ПК.

При рассмотрении вопроса о представлении различных чисел в памяти ЭВМ удобно записывать эти числа в десятичной (естественной для нас), в двоичной (естественной для компьютера) и шестнадцатеричной (более компактной) формах записи.

Основная сложность в понимании излагаемого материала, на взгляд авторов, состоит в том, что производимые операции следует рассматривать синхронно для восьми-девяти различных сущностей (см. рис. 2.1), поэтому попробуем показать «условное направление» движения мысли стрелкой. Большинство других авторов, стремясь к компактности изложения темы, зачем-то «из большой и красивой картины» делают «фрагментарный рисунок», что и создаёт в сознании читателей (слушателей) путаницу, усложняет понимание и запоминание изложенного материала. А самое главное, читателям не даются ответы на потенциальные вопросы вроде: «*А зачем всё это нужно?*», «*Зачем столько сущностей, в какой-то мере дублирующих друг друга?*» На какие-то вопросы ответы очевидны, на какие-то нет. Для каждой сущности имеются свои правила, где-то они схожи с соседними, а где-то различны. Попробуем разобраться со всем этим.

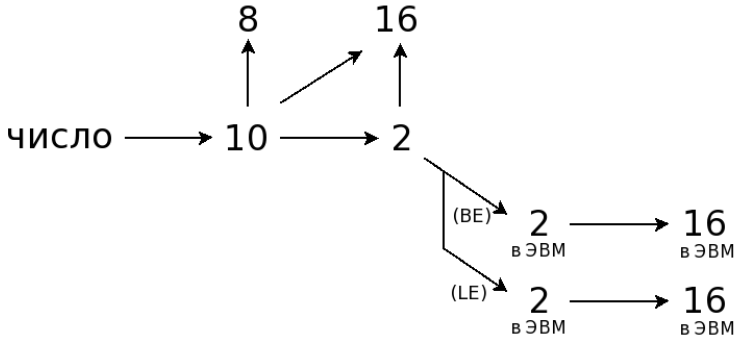


Рисунок 2.1. Предлагаемое условное направление движения мыслей между сущностями (цифры обозначают используемую систему счисления)

Следует полагать, что первично в жизни существует число (числа), некоторая абстракция отображающая реальность, а далее люди представляют числа какой-либо мо-

делью. Обычно в качестве модели мы используем десятичную систему счисления, которой нас учат в детстве родители, а позднее мы изучаем её в школе. В старших классах к нашим познаниям добавляются системы счисления с основаниями 2, 8, 16 и мы учимся преобразовывать числа из одной системы в другую. Иные системы счисления, как и нижние две ветки рисунка 2.1 с отметками «в ЭВМ», обычно не рассматриваются за ненадобностью, либо их изучение поверхностно.

Рассмотрим «нижнюю» ветку рис. 2.1 на примере числа 361 и запишем наши мысленные шаги с пояснениями в виде табл. 2.1.

Таблица 2.1. Категории сущностей. (стрелки указывают направление движения мыслей)

число →	10 →	2 →	8 в ЭВМ (LE) →	16 в ЭВМ (LE)
Число	десятичная система счисления	двоичная система счисления	двоичный формат хранения чисел в памяти ЭВМ	шестнадцатеричная форма записи двоичного формата хранения чисел в памяти ЭВМ
Изначальная математическая абстракция	привычная человеку форма записи чисел	математическая абстракция	фактическое хранение чисел в памяти ЭВМ (зависит от архитектуры и точности)	фактически bin2hex("2 в ЭВМ"), компактная форма записи чисел двоичного формата
	361 ×	101101001 × 	[01101001 00000001 00000000 00000000]	[69 01 00 00]

Замечание. Обратите внимание, что шестнадцатеричная форма записи двоичных чисел может быть применена как к «абстрактным двоичным числам», так и к записи, выполненной в «двоичном формате хранения чисел в памяти ЭВМ». Иногда в литературе для обозначения «представлений в ЭВМ» над двоичной или шестнадцатеричной записью могут использоваться квадратные скобки. Например «[69010000]». Если предположить, что двоичные записи (числа в двоичной системе счисления и в представлении в памяти ЭВМ) могут отличаться, то и компактные, шестнадцатеричные записи указанных сущностей также будут отличаться. Разные множества чисел (положительные целые, отрицательные целые и вещественные разных диапазонов) имеют различные двоичные форматы.

2.1.1.1. Зачем столько двоичных форматов? [29]

Если при работе с ЭВМ использование таких систем счисления как «10», «2», «16» объясняется удобством использования в конкретной ситуации (см. выше), то следует логичный вопрос: «Зачем нужно множить двоичные форматы?» а именно, зачем

[29] «Представьте себе, сколько усилий было бы сэкономлено, если бы Homo sapiens имел 8 или 16 пальцев!» [5, стр. 500]. Кстати, если в отношении рук, вы спросите у коренного англичанина: «How many fingers do you have, sir?», то в ответ услышите: «I have eight fingers.», потому как наши большие пальцы рук у них называются thumbs.

помимо «2» придумали ещё несколько форматов «2 в ЭВМ» (варианты «2 в ЭВМ (LE)» и «2 в ЭВМ (BE)» для каждой из разрядностей 16, 32, 64 и т.д.)?

Ответом может быть:

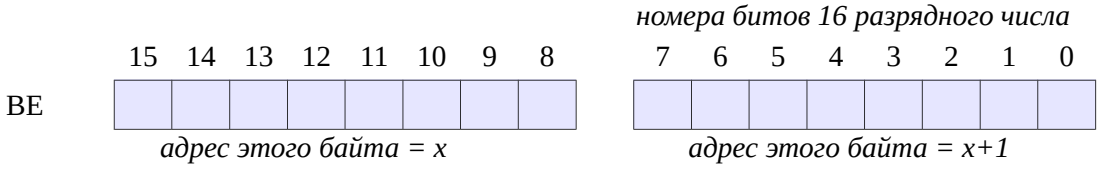
- с целью экономии, то есть для уменьшения числа операций, затрачиваемых компьютером для осуществления операций сложения (и умножения), либо для упрощения схемотехники, реализующей эти операции; (Позднее вы выясните, что процессору складывать положительные и отрицательные числа, представленные в дополнительном коде проще, поскольку для этого нет необходимости делать какие-либо предварительные преобразования над слагаемыми.)
- чтобы учитывать особенности архитектур процессоров, существующих на сегодня, например порядок следования байтов в многобайтовых машинных словах: «`LITTLE_ENDIAN`» или «`BIG_ENDIAN`» (см. выше в скобках LE и BE);
- чтобы учитывать особенности хранения одного и того же числа в переменных разных форматов (например целого типа, но разных размеров).

Замечание. Следует заметить, что мы не рассматриваем двоично-десятичный код [5, стр. 499] и другие, коих великое множество, а ограничимся представлением чисел в современном ПК, оснащённом процессором фирмы AMD или Intel с порядком Little Endian.

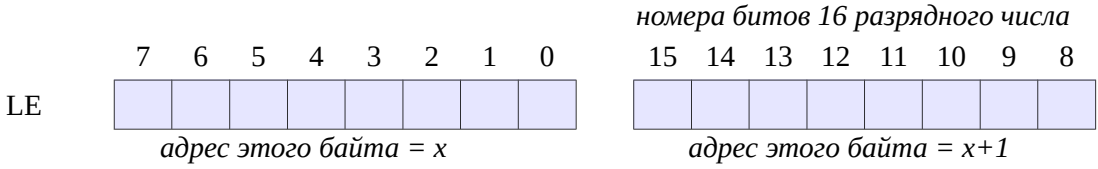
Замечание 2. Ряд специфических (на сегодня) систем и процессоров могут работать как в порядке от младшего к старшему, так и в обратном, причём режим работы может изменяться как аппаратно, за счёт перемычек на плате, так и программно. Переключаемый порядок байтов иногда называют англ. *bi-endian*. Также в литературе можно встретить ранее существовавшие и использовавшиеся другие форматы порядка, такие как *Middle-Endian* и *PDP-Endian*, но на сегодня они все не являются актуальными.

Замечание 3. Чтобы лучше запомнить различие между терминами big-endian и little-endian, посмотрите на варёное яйцо. Эти термины также можно отнести и к нему, поскольку у него один конец будет более тупым чем другой. А понять почему мы попросили вас посмотреть именно на варёное яйцо вы сможете перечитав сатирический рассказ «Путешествия Гулливера» Джонатана Свифта.

Ниже приведено абстрактное расположение битов одного и того же двухбайтового числа (16 разрядов) в памяти ЭВМ для порядков Big Endian:



и Little Endian:



В примере выше рассмотрены двухбайтовые слова. Схожим образом в памяти переставляются местами байты более длинных слов 32-х, 64-х и более разрядных систем.

2.1.2. Основы кодирования, представления и записи целых чисел

Запись первых 15 натуральных чисел и 0 в упомянутых выше системах счисления (не формах представления чисел в ЭВМ) будет выглядеть следующим образом (см. табл. 2.2).

Таблица 2.2. Запись первых пятнадцати натуральных чисел и нуля в различных системах счисления

Десятичная	Двоичная	Шестнадцатеричная
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A или a
11	1011	B или b
12	1100	C или c
13	1101	D или d
14	1110	E или e
15	1111	F или f

Для того чтобы различать, в какой системе счисления записано число, удобно указывать систему счисления нижним индексом в записи числа, например:

$7_{10} = 0111_2 = 7_{16}$ или со скобками $(7)_{10} = (0111)_2 = (7)_{16}$

$13_{10} = 1101_2 = D_{16}$ $(13)_{10} = (1101)_2 = (D)_{16}$

Обозначения, используемые при записи чисел

Индекс «2», записанный от числа справа снизу, означает, что число записано в двоичной системе счисления, например $0{,}01_2$.

Индекс «10» означает десятичную систему счисления, например $0{,}01_{10}$.

Индекс «16» - шестнадцатеричную систему счисления, например $110A_{16}$. Также шестнадцатеричные числа обозначают префиксом «0x», например 0x00A0.

Замечание. Учитывая, что для более-менее детального исследования вопроса представления целых чисел в памяти ЭВМ необходим доступ к ячейкам памяти, будем иллюстрировать наши рассуждения примерами на языке Си (англ. C). Программы предназначены для углублённого изучения рассматриваемых вопросов. Если Вы не умеете программировать, то не расстраивайтесь, в будущем у Вас будет шанс (в других курсах, либо самостоятельно) подробно познакомиться с основами программирования и синтаксисом данного языка. Сейчас лишь кратко рассмотрим структуру одной программы на языке Си и приведём необходимые пояснения. Все последующие программы в учебнике (возможно написанные и на других языках программирования) столь подробно рассматриваться не будут.

Коды программ сознательно приводятся в тексте, а не вынесены в приложения, так как многие из них очень просты и наглядны. Если же Вы пока не умеете программировать и «никакая» наглядность кода Вас не убеждает нам поверить, то – компилируйте и запускайте программы, проверяйте изложенный материал, понимание исходных кодов в деталях для этого не потребуется.

Здесь и далее в учебнике приводится множество примеров для командной строки ОС Linux. Поскольку использование командного интерпретатора bash стало стандартом de facto для CLI интерфейса этой и ряда других ОС, авторы не считают нужным углубляться в её изучение. Большинство команд интуитивно понятны, к более сложным даются пояснения. Выбор дистрибутива ОС Linux не важен, поскольку командная строка везде едина. В учебнике используется традиционное обозначение приглашения командной строки: строки начинающиеся со знака $ выполняются от обычного пользователя, а со знака # – от суперпользователя.

Существенным моментом является то, что для заданий ниже должны быть установлены средства разработки и компиляторы gcc, g++, fpc. Для работы примеров на php нужен пакет php-cli. Для работы с текстами нужны пакеты iconv, dos2unix, unix2dos.

Рассмотрим программу, которая определяет сколько места занимает в памяти компьютера переменная типа unsigned char и какое максимальное значение может принимать число, хранимое в переменной этого типа. Выведем текст этой программы и разберём как она работает.

```
$ cat unsigned_char_max.c
#include <stdio.h>
#include <limits.h>
int main()
{
   printf("unsigned char size: %d\n", sizeof(unsigned char));
   unsigned char a = ~0; // все биты a равны 1
   printf("unsigned char max value: %d\n", a);
   printf("unsigned char max: %u\n", UCHAR_MAX);
   printf("char bit count: %u\n", CHAR_BIT);
   printf("char size: %d\n", sizeof(char));
}
```

Структурно эта программа состоит из:

- директив компилятору[30] языка Си о подключении дополнительных заголовочных файлов библиотек (`#include <stdio.h>` и `#include <limits.h>`), содержащих объявления вызываемых программой функций, констант и переменных. Файл `stdio.h` является системным заголовочным файлом, который является частью стандартной библиотеки языка Си и расположен по стандартному пути, известному компилятору. Данный файл содержит объявление функции `printf()`, которая предназначена для вывода строк на устройство стандартного вывода (консоль) программы. Реализация функции `printf()` находится в стандартной библиотеке языка Си, которая неявно подключается в ходе компоновки программы, в процессе которой осуществляется связывание сгенерированных объектных модулей и используемых ими библиотек. Назначение параметров функции `printf()` подробнее рассматривается ниже. Файл `limits.h` включается для получения значений констант `UCHAR_MAX` и `CHAR_BIT`;
- главной точки входа приложения функции `main() {...}`. Каждая функция внутри фигурных скобок содержит набор операторов, реализующих алгоритм её работы;

[30]Компилятор отвечает за перевод исходного файла в объектный модуль, содержащий машинный код. Впоследствии компоновщик объединяет объектные модули в исполняемые программы или библиотеки.

- других функций (в данном случае их нет).

В программе используется всего лишь одна функция `printf()` для вывода данных, которая вызывается несколько раз. Функции передаётся один или более параметров, второй и последующие из которых являются опциональными, т. е. могут отсутствовать. Первый параметр функции `printf()` определяет строку форматирования, которая помимо обычного текста может содержать специальные подстановочные символы (%f, %d, %s), которые будут заменены функцией `printf()` на значение её второго, третьего и т. д. аргумента в соответствии с порядком их следования. Разные подстановочные символы используются для указания функции `printf()` типа переданных ей аргументов и для способа их форматирования. Например, %f – как число с плавающей запятой типа float, %d – как целое десятичное число, %s – как символ или строку и т. д. Рассмотрим, например, вызов функции.

```
printf("%d %f", 1, 3.0);
```

Данная функция выведет на консоль.

```
1 3.000000
```

Подробнее о существующих символах и форматах вывода функции `printf()` можно прочитать, набрав в консоли ОС Linux «`man 3 printf`» или воспользовавшись справочником языка программирования или используемой среды разработки.

Рассмотрим первый вызов функции printf() в нашей программе:

```
printf("unsigned char size: %d\n", sizeof(unsigned char));
```

«`\n`» в строке форматирования означает символ перевода на следующую строку и используется для того, чтобы последующий вывод на консоль производился с новой строки. Оператор `sizeof(unsigned char)` возвращает количество байт, занимаемых переменными типа unsigned char. Для хранения временного значения в программе используются переменная *a*.

```
unsigned char a = ~0;
```

Переменные бывают различных типов. Для правильной работы программы указание типа обязательно. Тип задаётся при создании переменной. В нашем случае это «`unsigned char`». Вместе с созданием переменой её можно инициировать, т. е. записать в ячейки памяти отведённые для неё какие-то значения. В нашем примере мы задали в качестве значения переменной *a* битовую инверсию нуля – то есть число у которого все биты равны единице.

К этому моменту должно быть понятно, что

```
printf("unsigned char max value: %d\n", a);
```

выведет значение, которое принимает переменная *a*.

Выше уже упоминалось, что файл `limits.h` содержит определение константы `UCHAR_MAX`, которая соответствует максимальному значению, которое может принимать переменная типа `unsigned char`.

```
printf("unsigned char max: %u\n", UCHAR_MAX);
```

При запуске программы ожидается, что значение *a* и `UCHAR_MAX` совпадут.

В предпоследнем запуске функции (3-я с конца строчка)

```
printf("char bit count: %u\n", CHAR_BIT);
```

выводится значение константы `CHAR_BIT`, содержащей количество бит в переменной типа `char`.

Переменные типа `unsigned char` занимают в памяти столько же байт сколько и переменные типа `char`, что должен показать вывод следующего вызова функции `printf()`.

```
printf("char size: %d\n", sizeof(char));
```

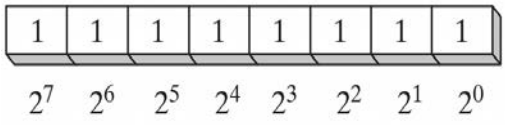
После того как программа написана её следует скомпилировать, скомпоновать («слинковать») и запустить. Во время компиляции мы задаём имя выходной программы ключом «-o». Если этого не сделать, то будет создан файл «a.out». Подробнее о компиляции, сборке и примеры программ под ОС Linux см. в [10].

```
$ gcc unsigned_char_max.c -o unsigned_char_max
```

После запустим программу, выполнив файл unsigned_char_max.

```
$ ./unsigned_char_max
unsigned char size: 1
unsigned char max value: 255
unsigned char max: 255
char bit count: 8
char size: 1
```

Как видим, число занимает в памяти 1 байт, что составляет 1*8=8 бит. Максимальное число получится тогда, когда все 8 бит будут выставлены в «1».



При подсчёте получим

$$2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 255 = 256 - 1 = 2^8 - 1$$

Проверим:

```
$ echo "2^7+2^6+2^5+2^4+2^3+2^2+2^1+2^0"|bc
255
$ echo "2^8-1"|bc
255
```

Замечание. bc – от англ. bash calculator, – хороший консольный калькулятор запросто считающий, например 2^{256}-1 и даже больше.

2.1.2.1. Прямой код

Возьмём другую программу и посмотрим, как хранится в памяти число 13.

```
$ cat unsigned_char_representation.c
void PrintBinByte(unsigned char c);

int main()
{
   unsigned char a = 13;
   PrintBinByte(a);
}
```

В этой программе используется написанная нами функция `PrintBinByte()`, которая находится в отдельном файле.

```
$ cat PrintBinByte.cpp
#include <stdio.h>
#include <limits.h>

void PrintBinByte(unsigned char c)
{
   unsigned char highBit = 1 << (CHAR_BIT - 1);     // 128 = 2^7 = (10000000)
   for (int i = 0; i < CHAR_BIT; ++i)
   {  bool isSet = (c & highBit) != 0;
      printf("%d", isSet ? 1 : 0);
      c <<= 1;
   }
```

```
}
```

Поскольку в языке Си нет типа bool, скомпилируем программу с помощью компилятора g++, а не gcc.

```
$ g++ unsigned_char_representation.c PrintBinByte.cpp
      -o uchar_representation
```

и запустим

```
$ ./uchar_representation
00001101
```

Как видим,

$$0 + 0 + 0 + 0 + 1*2^3 + 1*2^2 + 0 + 1*2^0 = 8 + 4 + 1 = 13 .$$

То есть мы проверили, что представление целых неотрицательных чисел типа unsigned char не отличается от канонической формы представления двоичных чисел. Действительно,

$$(b_{n-1}\, b_{n-2} \ldots\, b_1\, b_0)_2 = b_{n-1} * 2^{n-1} + b_{n-2} * 2^{n-2} \ldots\, b_1 * 2^1 + b_0 * 2^0,$$

где b_i принимают значения из {0; 1}. Такую форму представления целого неотрицательного числа называют *прямым кодом*.

2.1.3. Отрицательные целые числа (машинное представление любых целых, как со знаком, так и без)

Если с положительными натуральными числами и нулём всё довольно просто (о знаке можно не думать), то с отрицательными целыми дело обстоит чуть сложнее. Для них действует та же самая схема рассуждений, что и для положительных чисел (см. рис. 2.1), но используются более сложные алгоритмы преобразований.

Во-первых, никто не хранит только отрицательные числа, так как для этой цели можно использовать положительные (тип unsigned), мысленно добавляя знак «–». В связи с этим интерес представляет не само хранение отрицательных чисел, а хранение отрицательных и положительных чисел вместе.

То есть чтобы выделенная переменная могла позволить представление какого-то подмножества множества $\mathbb{Z}$ (как из положительной области чисел, так и из отрицательной).

В этом случае переход от абстрактного «числа» к его десятичной (как и к двоичной) форме обычно не вызывает проблем. Преобразование положительных чисел остаётся неизменным, а преобразование отрицательных чисел производится так же, как и положительных чисел, но по завершении операций преобразования спереди записи ставится знак минус «–».

Во-вторых, сложности начинаются при попытке записать это число (а именно «минус») в памяти компьютера. Вариантов, как это сделать, много, и вопрос в том, как выбрать самый оптимальный.

Рассмотрим некоторые наиболее очевидные варианты

Если мы задействуем дополнительный бит для знака, то получим числа размером в 8 + 1 = 9 бит, 32 + 1 = 33 бита и 64 + 1 = 65 бит, где будет записаны знак и модуль числа, что не очень удобно с точки зрения записи чисел в памяти и операций над ними (особенно с разными знаками). Также мы получим ситуацию, когда у нас будет два нуля, «+0» и «−0».

Попробуйте ответить самостоятельно: «Сколько операций потребуется, чтобы сложить −4 и +4 ? А −4 и +5 ? Какой получится ответ, если произвести операции над указанными числами, представленными в двоичном виде (знак и модуль числа) с точки зрения обычной арифметики?»

Возможное решение указанных проблем: ограничить максимальный и минимальный размеры чисел, выделив для их представления на один бит меньше, а с наличием двух нулей смириться.

В [6] данный формат называется как «*sign and magnitude*», что дословно можно назвать «*знак и значение*» или «*знак и модуль*». Удобство хранения чисел в таком формате, с точки зрения человека, приведёт к излишним преобразованиям при сложении чисел компьютером. Если вы не поленились и постарались ответить на вопросы выше, то заметили, что обычные правила арифметики ($\oplus$ – сложение по модулю 2), когда мы складываем числа с разными знаками «в столбик» работать не будут.

Чтобы обойти этот недостаток, можно представлять числа со сдвигом (как это делается при хранении порядка у чисел с плавающей точкой (запятой)), либо «таблично», в этом случае мы избавимся от двух нулей, но удобств для осуществления арифметических операций это не добавит.

На практике вопрос хранения отрицательных чисел в памяти ЭВМ решают использованием дополнительного кода. Рассмотрим подробнее, почему используется этот формат, и проверим программой, так ли это.

2.1.3.1. Дополнительный код

В зарубежной литературе (в частности, в [4, 6, 11]) вместо «*дополнительный код*» можно встретить «*N-bit two's complement*» и «*two's-complement*», что дословно можно перевести «*как дополнение в n-битах до двух*» или, коротко, «*дополнение до двух*». Рассмотрим, что это означает, взяв алгоритмы преобразования из [6].

Мы уже увидели, что положительные числа в компьютере хранятся в двоичной системе счисления.

Каждый компьютер имеет свою архитектуру и определённый процессор, а последний – в свою очередь, фиксированное число разрядов, называемое «*словом*» *(word)* и определяющее размер порции данных, которую процессор может обработать за один раз. В общем случае размер слова зависит от используемого процессора и режима его работы (который, в свою очередь, определяется ОС), но в большинстве современных архитектур он не меньше 32 бит [13]. Обычно при программировании придерживаются следующих правил:

для «очень больших» чисел применяется тип long long,

для «обычных» – int,

для «небольших» – short,

а для «совсем маленьких» – char. Правда, увлекаться экономией памяти ради экономии не стоит: выигрыш чаще всего невелик, работа с «невыровненными» данными, не кратными по размеру слову, может негативно повлиять на производительность.

При хранении в переменной чисел обоих знаков в двоичной форме записи этой переменной самый левый (старший) бит отводится под знак («знаковый бит»). Первое правило: если он равен 0, то, значит, мы имеем дело с положительным числом, если в знаковом бите стоит «1», то мы имеем дело с отрицательным числом. Далее, в зависи-

мости от этого бита, действуют разные алгоритмы. С положительными числами всё просто, а вот для отрицательных существует несколько различных схем, упрощающих человеку их преобразование в «машинный формат» представления.

Дополнительный код

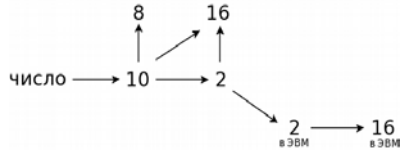
Положительное число	Отрицательное число
Первый бит в записи равен 0 (0xxxxx....xxxxx)	Первый бит в записи равен 1 (1xxxxx....xxxxx)
Дополнительный код есть само число в прямом коде	Дополнительный код определяется любым из рассмотренных алгоритмов

Благодаря дополнительному коду процессор может работать с положительными и отрицательными числами одинаково хорошо: $a + (-a)$ всегда даёт 0, и в семантику представления арифметический блок процессора вообще не вникает.

Первое, что приходит в голову, – это формат «знака и значения», рассмотренный нами выше, но, как мы выяснили, это не лучший способ. Лучший способ, как показали практика и время, – это хранить числа в дополнительном коде (формате дополнения «до двух», *см. врезку*). Этот способ основывается на базовом принципе арифметики, что число a, сложенное со своим отрицанием $-a$, должно давать в сумме 0. Если это правило действует для первой сущности «число» (см. рис. 2.1), то почему бы его не перенести на сущность «2 в ЭВМ».

То есть в двоичной арифметике для компьютера также должно выполняться правило, что дополнительный код положительного числа, сложенный с дополнительным кодом отрицания этого же числа, должен давать в сумме 0 (в формате двоичного представления внутри компьютера).

Фрагмент рисунка 2.1



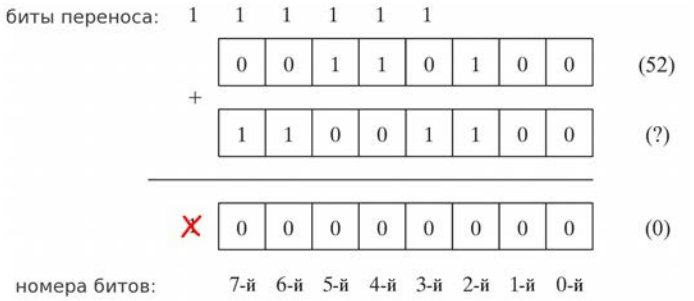
Дополнительный код для положительных и отрицательных чисел высчитывается по-разному.

В случае положительных чисел всё действительно просто. Спереди от числа, преобразованного в двоичный вид, дописываются незначащие нули, чтобы общая длина двоичной записи числа соответствовала размеру отводимой под него памяти. Иными словами, дополнительный код положительного числа «совпадает» с этим числом.

Например, если взять «число 52», то на следующем этапе, согласно рис. 2.1, мы берём это число в десятичной системе счисления 52_{10}, после чего преобразуем его в двоичный вид 110100_2, а затем дописываем спереди нули. В случае хранения числа в одном байте памяти дописываем нули до 8 разрядов, то есть для «2 в ЭВМ» получим «00110100», а для случая использования типа *integer* (занимаемый объём 4 байта) придётся дописать на 24 нуля больше, а запись «2 в ЭВМ (LE)» будет другой (о том, что и как переставляется, рассматривается ниже).

Теперь рассмотрим способ получения дополнительного кода для отрицательных целых чисел. Данный способ базируется на двух основных свойствах: на том, как компьютер осуществляет сложение, и на факте, что число $-a$ есть то число, которое вы должны добавить к a, чтобы получить 0. Например, если $a = 3$, то $-a = -3$, так как $3 + (-3) = 0$, и если $a = -4$, то тогда $-a = 4$, так как $-4 + 4 = 0$.

Возьмём два числа 00110100 и 11001100, произведём их сложение «в столбик», в результате чего получим 0. Последний (левый) 9-й бит «переноса» мы отбросили, так как сумма должна поместиться в 8 битах. Также заметим, что первым числом было как раз 52_{10}.



Из получившегося результата (сумма равна 0) и свойств, предложенных выше, делаем вывод, что битовая комбинация 11001100, использовавшаяся нами при сложении с первым числом, есть дополнительный код (для 8-битового представления) для числа −52, так как −52 + 52 в сумме даёт 0.

Теперь давайте разберёмся, как такое получилось. С одной стороны, вы можете сказать, вы же сами выбрали числа таким образом, чтобы их двоичная сумма была нулём. Это верно, но давайте подробнее рассмотрим, как мы производили операцию сложения этих чисел.

Мы начали складывать биты в записи в столбик справа налево и, если было необходимо, осуществляли перенос единицы на следующий разряд влево. А далее мы могли заметить, что, складывая два дополнительных кода (дополнительный код положительного числа есть прямой код), мы получали при сложении каждого бита всегда ноль. Наверняка эта закономерность бросилась в глаза, если вы честно сложили числа, а не пропустили этот шаг как очевидный.

Замечание. Чтобы лучше прочувствовать «закономерность дополнительного кода» попробуйте выполнить следующее несложное задание. Для этого выпишите в столбик все возможные комбинации ячеек памяти для некоторой абстрактной 4-разрядной ЭВМ (см. таблицу справа). У вас получится всего 16 комбинаций от 0000, 0001, 0010 и т.д. до 1111. Можно взять и не абстрактную, а реальную 8- или 16-разрядную ЭВМ, но тогда таких комбинаций будет $2^8 = 256$ или $2^{16} = 65536$ и вся наглядность формата потеряется. Сопоставьте первым восьми комбинациям значения от 0 до 7. Это полностью согласуется с реальной картиной происходящего и с уровнем начальной школы по представлению положительных целых чисел в ЭВМ. У вас получится таблица, приведённая справа от этого предложения.

Коды чисел (они же комбинации ячеек памяти ЭВМ в двоичном виде)	Сопоставленное десятичное значение числа
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	
1001	
1010	
1011	
1100	
1101	−3
1110	
1111	

Заполните пустые ячейки

Далее, постарайтесь самостоятельно определить значения пустых ячеек (например, методом перебора всех возможных значений) и заполните таблицу до конца.

Ожидаемый алгоритм рассуждений и действий читателей следующий:

Во-первых, следует исходить из следующих разумных положений:

- все недостающие значения отрицательные;
- операции сложения как отрицательного числа с положительным, так и их кодов должны давать правильный результат (собственно это основная цель создания дополнительного кода).

Во-вторых, если предположить что для человека операции с нулём проще, а большая часть модулей от непроставленных значений будет лежать в диапазоне от 0 до 7, начать заполнение лучше где-то с середины, то есть с определения позиции числа −3 или −4.

В-третьих, для определения положения числа в таблице (например, мы выбрали число −3) должны выполняться следующие равенства (удобнее всего операции сложения записать в столбик): 3 + (−3) = 0 – в десятичной системе счисления и 0011 + xxxx = 0000 – в двоичной, где xxxx − искомый дополнительный код числа (−3), причём, внимательно изучив таблицу, можно сказать, что его первый разряд уже известен − это 1. Так, в худшем случае, за восемь переборов всех комбинаций от 1000 до 1111 у вас должен найтись правильный результат и определиться место в таблице. Для располагающегося рядом с −3 числом (−2 или −4) потребуется уже семь переборов. Затем шесть. А затем пять и т.д., вплоть до 8*7*6*5*...=8!, если вовремя не заметить закономерность и не сократить число переборов.

Попробуем наблюдаемую закономерность преобразовать в алгоритм вычисления дополнительного кода.

2.1.3.2. Алгоритмы вычисления дополнительного кода

Алгоритм № 1 поиска дополнительного кода (*n-bit two's complement*) представления числа $-m$ ($m > 0$).

1. Находим двоичное представление числа m.
   (Например, для $m = 52_{10}$ получим 110100_2.)
2. Дописываем слева от записи нули в необходимом количестве, для того чтобы у нас получилось всего n бит. (Из 110100_2 при $n = 8$ получим 00110100_2.)
3. Двигаемся справа налево и дополняем каждый бит записи с первой встретившейся слева единицы (не затрагивая её) единицей. Для последующих разрядов, как легко заметить, будет действовать правило, что если был 0, то в дополнении будет 1, а если была 1, то в дополнении будет 0.

Таблица 2.3. Дополнение каждого бита записи левее первой встретившейся единицы справа (не затрагивая её)

0	0	1	1	0	1	0	0	исходное число 52_{10}
⊕ 1	⊕ 1	⊕ 1	⊕ 1	⊕ 1	–	–	–	дополнения разрядов по модулю 2
1	1	0	0	1	1	0	0	результат

Заметьте, что если мы так будем дополнять число 0, то ожидаемая единица никогда не встретится, в результате получим также 0. Так что не зря в поговорке говорится, «*как ни крути, ноль – он и в Африке ноль*».

Другими словами, если вы возьмёте дополнительный код любого целого числа, как мы сделали выше, причём не важно, положительного или отрицательного, и прибавите «1» к каждому его разряду левее первой встретившейся единицы справа, то вы получите дополнение этого числа (что в десятичной системе счисления соответствует отрицанию числа). Так, если возьмём исходно в дополнительном коде 11001100 (что есть -52_{10}) и прибавим к каждому разряду «1» по модулю 2 после самой правой единицы, если двигаться налево, то получим 00110100 (что будет 52_{10} – отрицание числа «-52_{10}»). См. табл. 2.4.

Таблица 2.4. Сложение разрядов с «1» слева от первой встретившейся единицы справа

1	1	0	0	1	1	0	0	исходное число -52_{10}
⊕ 1	⊕ 1	⊕ 1	⊕ 1	⊕ 1	–	–	–	дополнения разрядов по модулю 2
0	0	1	1	0	1	0	0	результат

Существует и другой способ получения *дополнительного кода*.

Если бы сложение выше было не поразрядным, а, как обычно, с переносом (см. иллюстрацию ниже), то при сложении числа с его дополнением мы бы получили в конечном итоге $100000000_2=256_{10}$. См. табл. 2.5.

Таблица 2.5. Другой способ сложения

1	1	1	1	1	1				биты переноса	
	1	1	0	0	1	1	0	0	слагаемое$_1$ (-52)	204_{10}
+										+
	0	0	1	1	0	1	0	0	слагаемое$_2$ ($+52$)	52_{10}
1	0	0	0	0	0	0	0	0	сумма (0)	256_{10}

Следующее наблюдение даёт нам способ получения дополнения. Обратите внимание, что $11001100_2 = 204_{10}$ и что 52 + 204 = 256, см. в таблице правый столбец. То есть дополнительным кодом числа –52 будет представление в двоичном виде числа ($2^8 - 52 = 204$). В математической терминологии мы говорим, что число 204 есть дополнение до нуля числа 52 по модулю 256 (mod 256), что означает, что это то число, в сумме с которым 52 даст 0. (Сложение по модулю 256 означает, что мы складываем числа как обычно, а после берём остаток от деления на 256. В нашем случае это 0.)

В общем случае, то есть не для 8-битных чисел, а для n-разрядных чисел, мы получим следующий порядок действий.

Если в компьютере используется n бит для хранения чисел, то логично, что при выполнении операций с числами следует использовать арифметику по модулю 2^n, потому как размер чисел будет ограничен представлением последних n-разрядами памяти. При этом оказывается, что операция «отбрасывания» старшего бита переноса, выходящего за сетку сложения, по сути, и есть деление результата сложения на 2^n и получение остатка от деления.

Коротко сказанное можно записать следующим алгоритмом.

Алгоритм № 2 нахождения дополнительного кода (*n-bit two's complement*) представления числа $-m$ ($m > 0$) есть представление целого положительного числа $2^n - m$ в двоичной системе счисления. Просто вычисляем это число.

Рассмотрим критические значения, 0, +128, −128. Кстати, несмотря на то что $m > 0$, если окажется $m = 0$, то эффект будет тот же, что и раньше: «ноль – он и в Африке ноль», потому как 256 в 8 битах не запишется (девятая единица числа 100000000_2 окажется за пределами восьми разрядов, отводимых под хранение, и будет отброшена). Что же касается чисел +128 и −128, то тут мы правильного результата не получим, но и использовать эти числа нельзя, так как «+128» не существует (при восьмибитном хранении числа указанное значение выходит за допустимые пределы), ведь мы старший бит числа отводим под знак числа, а для положительных чисел старший разряд будет 0. А «−128» однозначно не удовлетворяет нашему условию ($m > 0$).

Кстати, для получения дополнительного кода числа по модулю 2^n можно воспользоваться ещё одним алгоритмом, который чем-то напоминает первый. На практике используется довольно часто, так как инвертировать побитно и суммировать визуально оказывается проще, чем искать первую встретившуюся единицу (как в *алгоритме* № 1) или возводить в степень и вычитать большие числа друг из друга (как в *алгоритме* № 2).

Алгоритм № 3 поиска дополнительного кода числа $-m$ ($m > 0$):

1. Найдите двоичное представление модуля рассматриваемого десятичного числа. (Дословно отбросьте знак и переведите в двоичную систему счисления. Например, для $m = 52_{10}$ получим 110100_2.)
2. В двоичном написании дополните это число спереди нулями до n-разрядного представления.
3. Инвертируйте его запись, заменив все «0» в записи на «1», а «1» на «0».

0	0	1	1	0	1	0	0	отбросили знак 52
1	1	0	0	1	0	1	1	инверсия $\overline{52}$

4. Прибавьте к числу 1.
5. Отбросьте выходящий за пределы n-разрядов результат ($n = 8$).

0	0	0	0	0	0	1	1		биты переноса при сложении
	1	1	0	0	1	0	1	1	инверсия $\overline{52}$
								1	+1
0	1	1	0	0	1	1	0	0	результат $\overline{52} + 1$
	1	1	0	0	1	1	0	0	отбрасываем старший разряд

Как легко догадаться, рассуждения, приводимые выше, применимы для любых n-разрядных чисел. Посмотрим, как на практике осуществляется хранение 32-разрядных чисел в памяти ЭВМ на примере типов unsigned int и int.

Возьмём программу `unsigned_int_representation.c`

```
$ cat unsigned_int_representation.c
#include <stdio.h>
#include <limits.h>
void PrintBinNumber(unsigned char *bytes, size_t count);
void PrintBinBytes(unsigned char *bytes, size_t count);
int main()
```

```
{ unsigned int a = 0;
  a = ~a; // все биты a равны 1
  printf("unsigned int max value: %u\n", a);
  printf("unsigned int max: %u\n", UINT_MAX);
  printf("unsigned int bit count: %u\n",CHAR_BIT * sizeof(unsigned int));

  unsigned int b = 13;
  printf("%u двоичное представление: \n", b);
  PrintBinNumber((unsigned char *)&b, sizeof(b));
  printf("\n");

  printf("%u в памяти: \n", b);
  PrintBinBytes((unsigned char *)&b, sizeof(b));
}
```

Где к ранее ведённой функции `PrintBinByte()` добавились `PrintBinNumber()` и `PrintBinBytes()`, описанные во вспомогательных файлах:

```
$ cat PrintBinNumber.cpp
#include <stdio.h>

void PrintBinByte(unsigned char c);
void PrintBinNumber(unsigned char *number, size_t bytesCount)
{
   for (int i = bytesCount - 1; i >=0; --i)
   {
      PrintBinByte(number[i]);
      bool isLast = i == 0;
      if (!isLast)
         printf(" ");
   }
}
$ cat PrintBinBytes.cpp
#include <stdio.h>

void PrintBinByte(unsigned char c);
void PrintBinBytes(unsigned char *bytes, size_t count)
{
   for (size_t i = 0; i < count; ++i)
   {
      PrintBinByte(bytes[i]);
      bool isLast = i == count - 1;
      if (!isLast) printf(" ");
   }
}
```

Скомпилируем программу.

```
$ g++ unsigned_int_representation.c PrintBinNumber.cpp PrintBinBytes.cpp
PrintBinByte.cpp -o unsigned_int_representation
```

Запустим и проверим результат.

```
$ ./unsigned_int_representation
unsigned int max value: 4294967295
unsigned int max: 4294967295
unsigned int bit count: 32
13 двоичное представление:
00000000 00000000 00000000 00001101
13 в памяти:
00001101 00000000 00000000 00000000
```

```
$ echo "2^32-1"|bc
4294967295
```

Как видим, максимальное значение рассчитано правильно, также правильно выведено число 13 в двоичном виде.

Можно заметить, что байты в памяти хранятся в обратном порядке (LITTLE_ENDIAN; для процессоров с архитектурой BIG_ENDIAN, например IBM POWER, программу придётся модифицировать: оставляем это читателю в качестве упражнения). Сделано это для того, чтобы не прибегать к дополнительным преобразованиям. Если мы ошибочно обратимся к числу типа *unsigned int* как к типу *unsigned char*, то, если число укладывается в диапазон *unsigned char*, результат будет правильным. Данная особенность была важна для обратной совместимости в коде 8-, 16- и 32-битных приложений, для обратной совместимости на пути апгрейда при «скачкé» процессоров 8085 → 8086; а потом на этапе 80286 → 80386. Кстати, исходя из этой особенности, при написании программ, работающих с протоколом TCP, может использоваться функция htons() для преобразования номера порта (двухбайтовое число) в число с сетевым порядком следования байтов [10, стр. 119]. Модифицируем программу, чтобы можно было передавать числа через командную строку, и посмотрим на отрицательное число в памяти.

```
$ cat int_representation.c
#include <stdio.h>
#include <stdlib.h>

void PrintBinNumber(unsigned char *bytes, size_t count);
void PrintBinBytes(unsigned char *bytes, size_t count);
void PrintHexNumber(unsigned char *number, size_t bytesCount);
void PrintHexMemory(void * pointer, size_t bytesCount);
int main(int argc, char *argv[])
{
   if (argc != 2)
   {
      printf("Использование:\n %s number.\n", argv[0]);
      return -1;
   }

   int number = atoi(argv[1]);

   printf("\t\tИсходное число: %d\n", number);

   printf("Двоичное представление: ");
   PrintBinNumber((unsigned char *)&number, sizeof(number));
   printf("\n");

   printf(" 16-чное представление: ");
   PrintHexNumber((unsigned char *)&number, sizeof(number));

   printf("Представление в памяти: ");
   PrintBinBytes((unsigned char *)&number, sizeof(number));
   printf("\n");
   PrintHexMemory(&number,sizeof(number));

   return 0;
}
$ cat PrintHexNumber.cpp
#include <stdio.h>

void PrintHexByte(unsigned char c);

void PrintHexNumber(unsigned char *number, size_t bytesCount)
{
   for (int i = bytesCount - 1; i >=0; --i)
   {
```

```
      PrintHexByte(number[i]);
      bool isLast = i == 0;
      if (!isLast) printf(" ");
   }
}
```

```
$ cat PrintHexByte.cpp
#include <stdio.h>

void PrintHexByte(unsigned char c)
{
   printf("   %02X   ", c);
}
```

```
$ cat PrintHexMemory.c
#include <stdio.h>

void PrintHexMemory(void * pointer,
                    size_t bytesCount)
{   unsigned short int i;
   char *a;
   for (i=0;i<bytesCount;i++)
   {
     a=( (char *) pointer)+i;
     printf ("по адресу %p: ",a);
     printf ("%02hhX\n",*a);
   }
   printf("\n");
}
```

Компилируем и запускаем.

```
$ g++ int_representation.c PrintBinBytes.cpp PrintBinByte.cpp \
      PrintBinNumber.cpp PrintHexNumber.cpp PrintHexByte.cpp \
      PrintHexMemory.c -o int_representation
$ ./int_representation -1013
        Исходное число: -1013
Двоичное представление: 11111111 11111111 11111100 00001011
 16-чное представление:    FF       FF       FC       0B
Представление в памяти: 00001011 11111100 11111111 11111111
по адресу 0x7fff8f5952fc: 0B
по адресу 0x7fff8f5952fd: FC
по адресу 0x7fff8f5952fe: FF
по адресу 0x7fff8f5952ff: FF
```

Как видим, практика не расходится с теорией.

2.1.3.3. Ошибки «переполнения» целых

Представление чисел в дополнительном коде, а также то, что у большинства компиляторов нет дополнительных проверок при работе с переменными типа int и unsigned int на пограничных значениях, хранит в себе опасность (не отлавливается ситуация переполнения). Определим на примере 32-битных чисел типа int минимально и максимально возможные числа для данного типа, а также посмотрим, как они хранятся в памяти ПК.

```
$ cat int_limits.c
#include <stdio.h>
#include <limits.h>

int main()
{
   printf("int max: %d\n",INT_MAX);
   printf("int min: %d\n",INT_MIN);
}
$ gcc int_limits.c -o int_limits
$ ./int_limits
int max: 2147483647
int min: -2147483648

$ ./int_representation 2147483647
        Исходное число: 2147483647
Двоичное представление: 01111111 11111111 11111111 11111111
 16-чное представление:    7F       FF       FF       FF
Представление в памяти: 11111111 11111111 11111111 01111111
```

```
по адресу 0x7fff5010242c: FF
по адресу 0x7fff5010242d: FF
по адресу 0x7fff5010242e: FF
по адресу 0x7fff5010242f: 7F
$ ./int_representation -2147483648
        Исходное число: -2147483648
Двоичное представление: 10000000 00000000 00000000 00000000
...
Представление в памяти: 00000000 00000000 00000000 10000000
...
```

Ошибки могут возникнуть при выполнении операций сложения, вычитания и умножения, при которых результат будет выходить за предельные значения. Приведём небольшой пример.

```
$ cat int_overflow.c
#include <stdio.h>

void main (void)
{  int a=INT_MAX; // 2147483647
   int b=4;
   int c;
   c=a+b;
   printf ("%d+%d=%d\n",a,b,c);
}
```

```
$ gcc int_overflow.c &&./a.out
2147483647+4=-2147483645
```

хотя результат должен быть:

```
$ echo "2147483647+4"|bc
2147483651
```

Как видим, программа при своей работе ни ошибок, ни предупреждений не выдаёт. Аналогичное поведение наблюдается и у ряда других компиляторов, например g++ семейства mingw (http://mingw.org/) для архитектуры IA32 обладает той же особенностью.

Что происходит в памяти в этот момент? Где возникает ошибка? Посмотрим содержимое памяти:

```
$ ./int_representation -2147483645
        Исходное число: -2147483645
Двоичное представление: 10000000 00000000 00000000 00000011
 16-чное представление:    80       00       00       03
Представление в памяти: 00000011 00000000 00000000 10000000
по адресу 0x7fff6f24872c: 03
по адресу 0x7fff6f24872d: 00
по адресу 0x7fff6f24872e: 00
по адресу 0x7fff6f24872f: 80
```

Получается, проблема в том, что процессор складывает не числа, а представление этих чисел в двоичном виде. Причём арифметика для цифр представления правильная:

```
 01111111 11111111 11111111 11111111
+00000000 00000000 00000000 00000100
----------------------------------------
 10000000 00000000 00000000 00000011
```

С точки зрения процессора, ошибки в сложении нет, так как он не делает различий между int и unsigned int (он просто складывает переданные ему числа как беззнаковые и выставляет соответствующие флаги, если диагностирует возможные ошибки при трактовании полученных данных). Ошибка возникает при трактовании полученного результата программой и библиотеками языка С. Если бы в программе выше типы всех переменных были бы unsigned int, то результат получился бы правильным. (Арифметические операции на уровне процессора – отдельная тема, выходящая за рамки курса «Информатика».)

```
$ cat unsigned_int.c
#include <stdio.h>
void main (void)
{
   int a=2147483647; // 2^31-1
   int b=4, c;
   unsigned int d;
   c=a+b;
   d=a+b;
   printf ("%d+%d=%u\n",a,b,c);
   printf ("%d+%d=%u\n",a,b,d);
}
```

```
$ gcc unsigned_int.c &&./a.out
2147483647+4=2147483651
2147483647+4=2147483651
```

Обратите внимание, что получить правильный результат можно, не только используя беззнаковые типы (то есть меняя интерпретацию числа программой), как в случае с переменной *d*, но и вынуждая библиотеку C трактовать целое со знаком как беззнаковое (вывод переменной *c*).

Многое зависит от директив компилятора

Если расширить эксперимент с ошибками и написать аналогичную программу во FreePascal, то результат будет тем же. Тип Integer заменён на Longint для наглядности при сравнении результатов работы программ, так как во FreePascal первый использует лишь 16 бит, а второй – 32 бита. Следует отметить, что рассматриваемый эффект присущ и 16-битным переменным.

```
$ cat int_sum.pas
program program10_int_sum;
var  a,b,c :Longint;
begin
     a:=2147483647;
     b:=4;
     c:=a+b;
     writeln(a,'+',b,'=',c);
end.
```

```
$ fpc -oint_sum.elf -Tlinux int_sum.pas
Free Pascal Compiler version 2.4.0 [2010/05/05]
for x86_64
Copyright (c) 1993-2009 by Florian Klaempfl
Target OS: Linux for x86-64
Compiling int_sum.pas
Linking int_sum.elf
/usr/bin/ld: warning: link.res contains out-
put sections; did you forget -T?
11 lines compiled, 0.0 sec

$ ./int_sum.elf
2147483647+4=-2147483645
```

Если ознакомиться с документацией, доступной на странице http://www.freepascal.org/docs.var, то в разделе 1.1.63 руководства для программистов [7] можно узнать, что у компилятора есть директива $R или $RANGECHECKS, отвечающая за проверку диапазонов. По умолчанию проверка не делается, что мы и наблюдали выше. Однако, если в коде включить проверку через директиву {$R+}, при компиляции в программе будут добавлены дополнительные проверки, и если во время работы программы будет выход за пределы отведённых значений для переменной, то будет выдана run-time ошибка, а программа завершится с кодом возврата 201.

```
$ cat int_sum2.pas
program int_sum2;
{$R+}
var  a,b,c :Longint;

begin
    a:=2147483647;
    b:=4;
    c:=a+b;
    writeln(a,'+',b,'=',c);
end.
```

```
$ fpc -oint_sum2.elf -Tlinux int_sum2.pas
...
$ ./int_sum2.elf
Runtime error 201 at $0000000000400227
  $0000000000400227
  $00000000004001A8

$ echo $?
201
```

Вышеупомянутый пример был приведён для языка FreePascal, так как аналогичной директивы для компилятора gcc нет; по крайней мере, авторам она неизвестна, а рассмотрение вопросов формата хранения переменных в памяти ПК не должно ограничиваться одной реализацией. Читатели (слушатели) должны смотреть на проблему шире. Следует отметить, что для компилятора g++ (языка C++) проблема также существует.

```
$ cat int_overflow.cpp
#include <iostream>
#include <climits>
using namespace std;

int main( int argc, char** argv)
{
  int a,b;
  cout<<"Минимальное значение "<<INT_MIN<<"\n";
  cout<<"Максимальное значение "<<INT_MAX<<endl;
  a=INT_MAX; b=4;
  cout<<INT_MAX<<"+"<<b<<"="<<a+b<<endl;
  return 0;
}

$ g++ int_overflow.cpp && ./a.out
Минимальное значение -2147483648
Максимальное значение 2147483647
2147483647+4=-2147483645
```

А вот PHP 5.2.12, обработав аналогичный код,

```
<?php
echo PHP_INT_MAX;
$a=2147483647; $b=4; $c=0;
if (is_int($a)) { echo "<br>a-int "; }
if (is_int($b)) { echo "<br>b-int "; }
if (is_int($c)) { echo "<br>c-int "; }
$c=$a+$b;
echo "<br>".$a."+".$b."=".$c;
if (is_int($a)) { echo "<br>a-int "; }
           else { echo "<br>a-not int "; }
if (is_int($b)) { echo "<br>b-int "; }
           else { echo "<br>b-not int "; }
if (is_int($c)) { echo "<br>c-int "; }
elseif (is_float($c)) { echo "<br>c-float "; }
 else { echo "<br>c-???(not int, not float)"; }
?>
```

выдал довольно интересный результат, показав свою лояльность к ошибкам программиста:

```
2147483647
a-int
b-int
c-int
2147483647+4=2147483651
a-int
b-int
c-float
```

Как видим, чуда, что в 32-битной переменной оказалось значение, превышающее максимальное, не произошло. PHP отработал в соответствии с документацией: «*если результат операции лежит за границами типа integer, он будет преобразован в float*» [14], а для последнего «*в PHP обычно используется формат двойной точности IEEE 754*» [15]. Поэтому результат был подсчитан и выведен правильно. Также заметим, что журнал веб-сервера (файл error_log) ни на строчку не пополнился.

2.1.4. Целые и не целые (числа с плавающей точкой/запятой)

Смеем предположить, что хранение целых чисел в памяти ЭВМ не является сложным для среднестатистического читателя, большей частью материал, изложенный выше, принимается интуитивно, так как сильно пересекается со школьным курсом, рассматривающим разные системы счисления. Кажущаяся на первый взгляд простота приводит разум к логичному вопросу: зачем подробнее изучать то, что и так понятно, разве можно узнать что-то новое? Как следствие, формат хранения чисел в памяти ЭВМ для многих, в том числе и будущих программистов, представляет собой «чёрный ящик»: создал переменную, присвоил значение, выполнил вычисления, считал результат.

До тех пор пока не требуются точные вычисления, работа «ящика» оказывается предсказуемой и устраивает большинство из нас. Если язык программирования не типизирован, то можно перестать думать о форматах данных вообще, так как последний будет преобразовывать их «на лету». (Собственно, есть за что любить тот же PHP.) В противном случае вам придётся из какой-нибудь книжки по программированию узнать о существовании различных типов данных и их пограничных значениях. Но и тут если не требуется что-то большее, чем сложение чисел с точностью не более чем двух знаков после запятой (например, рубли, копейки), а цифры в расчётах не более чем семизначные, то мир прекрасен и написанные нами программы работают. Однако, как только мы делаем что-то более специфичное, начинаются «фокусы», в ряде которых мы и попробуем разобраться.

В предыдущем разделе был рассмотрен более простой вопрос – представление целых чисел, но и там были свои проблемы, например переполнение на пограничных значениях. В этом разделе мы коснёмся вопросов хранения и обработки вещественных чисел, фактически заглянем во внутренности «чёрного ящика» всех современных ЭВМ. Смеем предположить, что «подводных камней» будет намного больше, чем сейчас можно представить.

Замечание. Плавающая запятая или плавающая точка?

В России традиционно целая часть числа от дробной отделяется запятой, также по сей день действует *ГОСТ 16325–88 «Машины вычислительные электронные цифровые общего назначения. Общие технические требования»*, где используется термин «*плавающая запятая*». Так как на сегодня наша промышленность, электроника и наука «скорее не развивается, чем развивается», наблюдается отставание в мировом масштабе. Информационный вакуум быстро компенсируется зарубежными коллегами выпуском своих стандартов, документаций и прочего, которые после часто переводятся непрофильными переводчиками. Так, у нас появился, если не сказать что был навязан, термин «*плавающая точка*». В современной литературе эти понятия являются синонимами. Что же касается «запятой» в качестве разделителя целого и дробной части, последняя используется в более чем 60 стран [24].

2.1.4.1. Кратко о стандартах IEEE 754-2008 и IEEE 854-1987

Для представления действительных чисел в компьютерах используется формат с плавающей запятой (точкой). Актуальными являются два стандарта для машинной арифметики с плавающей точкой.

IEEE 754-2008 IEEE Standard for Floating-Point Arithmetic определяет представление и операции для чисел с плавающей точкой в компьютерных системах.

Рассматривает форматы хранения, правила арифметики (в том числе и правила округления), стандартные и расширенные функции для типов одинарной (single), двойной (double), расширенной (extended) и расширяемой (extendable) точности, а также рекомендует форматы для обмена данными. В рамках используемых форматов определяет:

- как представлять нормализованные положительные и отрицательные числа с плавающей запятой;
- как представлять денормализованные положительные и отрицательные числа с плавающей запятой;
- как представлять «нулевые» числа;
- как представлять специальные величины «плюс бесконечность» и «минус бесконечность» (±Infinity, ±∞);
- как представлять специальные величины «Не число» (NaN, NaNs, not a number).

IEEE 854-1987 IEEE Standard for Radix-Independent Floating-Point Arithmetic обобщает стандарт ANSI/IEEE Std 754-1985, IEEE Standard for Binary Floating-Point Arithmetic (в 1987 году стандарт IEEE 754-2008 ещё не вышел) с целью убрать из него зависимость от основания системы счисления и длины машинного слова. Он описывает числа с основаниями 2 и 10 («*the radix shall be either 2 or 10 and shall be the same for all supported precisions*»), но, в отличие от IEEE 754-2008, не описывает точное представление таких чисел по битам в памяти ЭВМ. «Radix-Independent» в названии, похоже, стоит понимать лишь как независимость от одного конкретного основания.

Так как перед нами не стоит задача досконального изучения стандартов, перейдём непосредственно к представлению действительных чисел в памяти компьютера на примере одного из типов данных, проверяя при этом отдельные положения небольшими программами.

Во-первых, определимся с терминологией.

Во-вторых, для приводимых примеров будет использоваться язык С. Рассмотрим имеющиеся в нём типы данных.

2.1.4.2. Типы данных в языке С (для хранения действительных чисел)

Как пишет Бьярне Строуструп [31]: «*...типы с плавающей точкой представлены тремя размерами: float (одинарной точности), double (двойной точности) и long double (расширенной точности). Точный смысл каждого типа зависит от реализации. Выбор нужной точности в реальных задачах требует хорошего понимания природы машинных вычислений с плавающей точкой. Если у вас его нет, либо проконсультируйтесь с кем-нибудь, либо изучите проблему сами, либо используйте double и надейтесь на лучшее*» [21, стр. 112]. «*Целью существования... нескольких чисел с плавающей точкой является представление программисту возможности эффективно использовать аппаратные средства. На многих машинах объём памяти, время доступа и скорость вычисления существенно зависят от выбора типа*» [21, стр.113].

[31] *Бьярне Строуструп*, дат. *Bjarne Stroustrup* – автор языка С++.

Согласно документу draft ISO/IEC JTC1 SC22 WG14 N1312 [29], типы float, double и long double для языка C являются базовыми. Компилятором для них отводится 4, 8 и 16 байт или 32, 64, 128 бит соответственно. В старой литературе вы можете встретить информацию, что тип long double занимал в памяти 10 байт или 80 бит, сейчас это не так. Наглядно данную информацию можно представить следующей таблицей.

Тип языка C	float	double	~~long double~~	long double
Размер, байт	4	8	~~10~~	16
Размер, бит	32	64	~~80~~	128

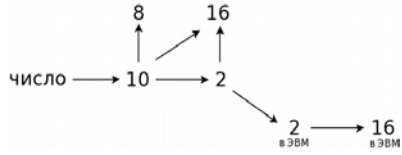
Проверить сколько занимает тот или иной тип можно простой программой:

```
$ cat types_check.c
#include <stdio.h>
void main(void)
{
  printf("Размеры занимаемые переменными (в байтах).\n");
  printf("float: %d\n",sizeof(float));
  printf("double: %d\n",sizeof(double));
  printf("long double: %d\n",sizeof(long double));
}
$ gcc types_check.c && ./a.out
Размеры занимаемые переменными (в байтах).
float: 4
double: 8
long double: 16
```

Замечание. Если у вас на компьютере больше гигабайта оперативной памяти и вы считаете, что для современных ЭВМ вопрос выбора формата хранения данных при программировании неактуален, напомним, что также существует мир микроконтроллеров, по сути тех же ЭВМ, чьи объёмы памяти на 1-2 порядка меньше. Даже SIM-карту телефона (или банковскую карту с чипом) можно в какой-то мере рассматривать как специфическую ЭВМ, не говоря уже о том, что во многих автомобилях управление устройствами происходит по CAN-шине, работа с которой не обходится без ЭВМ.

Какой тип выбрать?

Фрагмент рисунка 2.1



Перед началом использования типов попробуем понять, что для представления действительных чисел также существует несколько «сущностей» (по аналогии с рис. 2.1 сущность – это некоторая модель представления действительных чисел и соответствующий ей формат записи числа на бумаге или в памяти ЭВМ) (см. рис. 2.2 *а*).

По мнению авторов, направление «движения мыслей» наиболее естественно происходит по стрелкам, хотя ничто не мешает вам двигаться и в обратном направлении, даже «перескакивать» через отдельные сущности. Так как за каждой сущностью фактически стоит одна или несколько моделей представления действительных чисел, то высказать однозначно мнение, что какая-то сущность (и как следствие – какой-то формат записи) лучше и что надо использовать только её, нельзя.

Рисунок 2.2 а Типичные направления движения мыслей между сущностями(рассмотрен лишь один вариант перехода для записи в память ЭВМ, подразумевая что он будет для порядка LittleEndian)

«Пробежимся» по графу, изображённому выше на рис. 2.2 *а* от «числа» до его представления в памяти ЭВМ на примере показания термометра (см. табл. 2.6).

Рисунок 2.2 б Типичные направления движения мыслей между сущностями

После рассмотрения примера выделим общие закономерности преобразований и конкретные механизмы переходов между ячейками таблицы.

Опираясь на основы моделирования (представление чисел – это модель) и глядя в таблицу легко догадаться, что:

- каждая модель имеет свои ограничения;
- достоверность (точность) представления чисел для каждой модели различна;
- в некоторых сущностях числа могут иметь несколько форм записи;
- преобразования чисел из одной сущности в другую могут быть неоднозначными.

2.1.4.3. Пример «36,6» (один частный случай)

Все мы когда-либо измеряли температуру и знаем, что такое градусник. Смеем предположить, что если дома в руках не держали, то уж в поликлинике ртутный градусник видели. Допустим, что последний показал какое-то значение, например 36,6 градуса по шкале Цельсия. Мы опустим из рассмотрения философские вопросы, выходящие за рамки статьи: «*что и как измерил градусник*?» и «*каковы относительная и абсолютные погрешности измерения*?», разрешив на них ответить специалистам по физике и метрологии. Фактически мы рассмотрели модель, что-то измерили и записали на бумаге в десятичной системе показания измерительного прибора – термометра, в нашем случае как 36,6. На самом деле то, что мы реально измерили, мы не знаем, ведь мы могли измерить и 36,5999999999999.... или другое значение, то есть мы неизбежно столкнулись с точностью – новой проблемой, которой при записи целых чисел [20] не возникало. В нашей модели мы интуитивно выбрали точность до одного знака после запятой. Выбери мы больше знаков для представления результатов измерения (факт, что нет физического смысла записывать число точнее, чем погрешность измерения, опустим) – столкнулись бы с ограничением в записи числа на листочке бумаги... например, листочек шириной в 33 клетки, а запись по одному знаку в клетке – вот вам и ограничение.

Таблица 2.6. Форматы числа в разных сущностях

Сущность	Описание, форматы	Значение
Действительное число	мера чего-то, зависит от того, в чём и как измеряют (в нашем случае это высота столбика ртути в условных единицах пропорционально измеренной температуре)	
Десятичная система счисления	● естественная форма записи ● нормализованный экспоненциальный вид ● денормализованный экспоненциальный вид	36,6 $3{,}66 \cdot 10^{1}$ $0{,}366 \cdot 10^{2}$
Двоичная система счисления	● естественная форма записи ● форма с плавающей запятой в экспоненциальном нормализованном виде ● форма с плавающей запятой в экспоненциальном денормализованном виде	100100,100110011001100110... $1{,}00100100110011001100110... \cdot 2^{101}$ $0{,}10010010011001100110011... \cdot 2^{110}$

Сущность	Описание, форматы	Значение
Промежуточный вид IEEE 754	Преобразование двоичного нормализованного числа в binary32 формат стандарта IEEE 754	0 10000100 00100100110011001100110
Двоичная форма представления в ЭВМ	непосредственное хранение в памяти ЭВМ (для архитектуры LITTLE_ENDIAN байты хранятся в обратном порядке)	01100110 01100110 00010010 01000010
Шестнадцатеричная форма представления ячеек памяти в ЭВМ	Компактная форма записи содержимого памяти	66 66 12 42

Точная запись числа, полученного в одной системе измерений и записанная в какой-то системе счисления, может не иметь однозначного представления в другой системе счисления, как если бы вас попросили точно записать периодическую дробь 1/3.

Действительное число может быть записано несколькими способами. Рассмотрим различные виды (формы) записи. Число 36,6 мы можем записать как $0{,}366 \cdot 10^2$. Различие форматов записи в десятичной системе счисления наглядно можно увидеть в продуктовом магазине. Где-то на одной пачке товара может быть написано 1 кг (считай $1 \cdot 10^3$ г), а на другой 1000 г, а где-то будет 0,5 кг. А измерь мы вес какого-нибудь товара, например, в тройских унциях, явно придётся округлять значения или отбрасывать разряды, оставляя лишь *n* знаков после запятой.

И как же все эти числа записать в памяти ЭВМ, например чтобы после сложить и узнать общую массу всех товаров? Разнообразие форм в записи одного числа может послужить причиной затруднений для работы цифрового устройства. Во избежание этого нужно:

- либо заранее преобразовывать все числа к единой форме,
- либо создать специальные алгоритмы распознавания формата числа,
- либо указывать каждый раз его форму записи.

Все три варианта в той или иной мере используются.

В первом случае мы можем все вычисления делать с типом float (либо другим). Во-втором случае рассмотрите фрагмент кода, который PHP-интерпретатор правильно распознает, преобразует форматы, а потом правильно посчитает и выведет сумму.

```
<?php
  $a='1';   $b=0.5;    $c=4;
  $d=$a+$b+$c;
  echo $d;
?>
```

А для третьего случая приведём код на C с явным указанием типов:

```
int a=1;
float b=0.5;
```

Из примеров выше может показаться, что интерпретируемые программы удобнее, но за удобство приходится платить – они работают явно медленнее, чем те же алгоритмы в скомпилированном виде.

Переход от десятичной системы счисления к двоичной не рассматриваются, так как это школьный курс информатики. Если забыли, см. *«Схема Горнера и перевод из одной позиционной системы в другую»* [26, *§ 14*, стр. 37–39].

2.1.4.3.1. Терминология

Если не брать в рассмотрение представление нуля, бесконечности и прочих исключений как чисел с плавающей запятой, то можно практически считать синонимами термины «*число с плавающей запятой*» и «*число, записанное в экспоненциальной форме*». В связи с чем числа с плавающей запятой в общем случае можно представить в экспоненциальном виде: $v = M \cdot q^p$,
где M – мантисса, q – основание системы счисления, а p – показатель степени или порядок числа v.

Замечание. В ряде источников литературы используются другие (аналогичные) обозначения:

$$v = M \cdot q^e \Leftrightarrow F = M \cdot q^p$$
$$v = F\ ;\ e = p\ ;\ w = b\ ;\ t = n$$

Пара слов про термины [25]:

- мантиссу англичане называют по-разному: *significand*, *mantissa*, *fraction* (точнее, *fraction* – это дробная часть – не совсем мантисса, но близко). По-русски – только *мантисса;*
- показатель степени по-английски будет *exponent*, реже *extent*. Русских вариантов два: собственно «*показатель степени*» и «*порядок*» числа. В некоторых неграмотно переведённых с английского языка текстах может встретиться третий, скорее ошибочный вариант, полученный транслитерированием – «*экспонента*». Во избежании путаницы с графиком функции $f(x) = e^x$, изучаемым в школе и называемым экспонентой, использовать это слово не стоит.

То, что таким образом можно записать любое число, достаточно очевидно хотя бы потому, что обычное представление – это частный случай экспоненциального – когда показатель степени равен 0. Дальше несложно заметить, что такое представление неоднозначно: мы можем делить/умножать мантиссу на основание, соответственно меняя показатель степени [25], например в десятичной системе счисления будет верно:

$0{,}123 \cdot 10^6 = \mathbf{1{,}23 \cdot 10^5} = 12{,}3 \cdot 10^4 = 123 \cdot 10^3 = 1230 \cdot 10^2 = 123000 = 1230000 \cdot 10^{-1} = \ldots,$

при этом выделенное жирным – **нормализованная форма записи**, все остальные – не нормализованные формы записи этого же самого числа.

Аналогично для двоичной системы счисления:

$0{,}110100 \cdot 2^6 = \mathbf{1{,}10100 \cdot 2^5} = 11{,}0100 \cdot 2^4 = 110100 \cdot 2^0 = 1101000{,}0 \cdot 10^{-1} = 11010000 \cdot 10^{-2}.$

Показатели степени в строчке выше записаны в десятичной системе счисления для удобства, тоже самое, но с двоичными значениями в показателях степени выглядит так:
$0{,}110100 \cdot 2^{110} = \mathbf{1{,}10100 \cdot 2^{101}} = 11{,}0100 \cdot 2^{100} = 110100 \cdot 2^0 = 1101000{,}0 \cdot 10^{-1} = 11010000 \cdot 10^{-10}.$

«*Subnormal numbers*» (*субнормальные числа*), описываемые в стандарте ANSI/IEEE Std 854-1987, есть то же самое, что и «*денормализованные числа*» (*denormalized numbers*), описываемые в стандарте ANSI/IEEE Std 754-1985. Будем использовать последний термин.

2.1.4.3.2. Алгоритм преобразования для одного числа из примера

Преобразование и форматы – вещи полезные, но всё же как записать температуру в памяти компьютера? «Чтобы не изобретать каждый раз велосипед», воспользуемся

разделом 3.5.2 стандарта IEEE 754-2008; тем, кто «не дружит» с английским языком или не имеет стандарта под рукой посоветуем обратиться к [25] или [17].

Для хранения действительных чисел в C часто используется тип float. Взяв справочник по типам данных языка C, можем убедиться, что число 36,6 с запасом укладывается в диапазон отведённых для float значений:

$$(\approx -3{,}40282347\cdot10^{+38}) < 36{,}6 < (\approx 3{,}40282347\cdot10^{+38}).$$

Выберем указанный тип как целевой для последующего преобразования числа 36,6, после чего проверим правильность наших вычислений программно.

Воспользуемся двоичной нормализованной записью, так как она наиболее легко преобразуется к 32-битному формату IEEE 754.

$$36{,}6_{10}\approx100100{,}10011001100110011001_2=$$
$$=1{,}00100100110011001100110\cdot2^5=$$
$$=1{,}00100100110011001100110\cdot2^{101}.$$

Последнее и есть экспоненциальный нормализованный вид. Заметим, что в этой записи «1,00100100110011001100110» – есть мантисса, а «101» – степень числа 2, записанная в двоичной системе счисления.

Согласно стандарту IEEE 754, преобразование будет происходить по следующему алгоритму («знак», «смещённый порядок», «хвост мантиссы»):

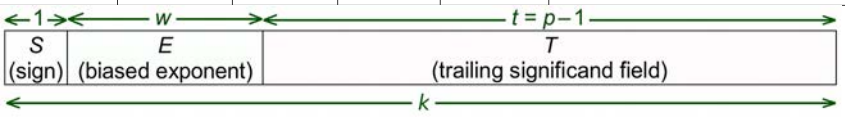
1. Определяемся со знаком числа. Если знак «+», то первый (он же самый старший) бит 32-битной записи числа в памяти ЭВМ будет равен 0. Если знак числа «−», запишем «1». Заметим, что есть сходство с представлением целых знаковых чисел в памяти ЭВМ с помощью дополнительного кода[4], когда получалось, что старший бит в памяти также отвечал за знак числа.
   Получим [0xxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx].

2. Смотрим в табл. 2.7 (стр.56), полученную из стандарта, и определяем, сколько бит отведено под хранение степени (колонка binary32, строка с параметром *w*). Из таблицы получаем 8 разрядов. Число, находящееся в степени, может быть как числом со знаком «+», так и со знаком «−». Согласно стандарту, для определения знака не отводят отдельного бита, а используют хранение в формате «со смещением» Смещение также определяется из табл. 2.6 (см. колонку binary32, строка *bias*). Численно оно равно целой части «половины» от 8 бит $\lfloor(2^8-1)/2\rfloor$, отведённых под число, то есть $+127_{10}(0111\ 1111_2)$. В нашем примере степень $p=101_2=+5_{10}$, следовательно, после смещения получим $E=+5+127=132_{10}=10000100_2$. Полученное «смещённое значение» записывают в отведённые 8 битовых позиций (поле *E* в стандарте). При этом, когда для вычислений нужно будет получить из записанного кода (смещённого порядка) обратно «реальный» *p*, мы просто отнимем 127_{10} из значения *E*. Получим [x100 0010 0xxx xxxx xxxx xxxx xxxx xxxx].

3. Оставшиеся 23 бита отводят для хранения битов мантиссы. Хранятся не все биты мантиссы, а лишь их часть. Этот факт отмечен использованием буквы *T* в соответствующем битовом поле, вместо буквы *M*. Из записи выбрасывается первый бит, поскольку у нормализованной двоичной мантиссы он всегда равен 1, так как число лежит в диапазоне $1 \le M < 2$. (О том, как «нормализовать» или записать «0», – отдельный вопрос. Ниже мы рассмотрим, как в стандарте IEEE754 решают вопрос с нулём, а пока успокоимся, ведь $36{,}6 \neq 0$). Нет смыла каждый

раз записывать первую единицу в отведённых 23 разрядах и использовать только оставшиеся 22 разряда, лучше «отдать первый разряд под точность», записывая во все отведённые разряды остаток от мантиссы. Естественно внутри процессора, для правильных математических вычислений «выброшенная» единица возвращается «обратно». Если оставшиеся биты мантиссы (после отброса первой единицы) занимают меньше 23 разрядов, то последние дополняются нулями. Получим [xxxx xxxx x001 0010 0110 0110 0110 0110].

Таблица 2.7. Параметры двоичных форматов представления чисел согласно стандарту IEEE 754-2008

Параметр и его описание	**Формат стандарта IEEE 754-2008**				
	binary16	**binary32**	**binary64**	**binary128**	**binary{k}**
k, storage width in bits; объём занимаемой памяти в битах	16	32	64	128	$k \geq 128$; k кратно 32
p, precision in bits; точночть в битах	11	24	53	113	$k - \text{round}(4 \times \log_2(k)) + 13$
e_{max}, maximum exponent *e;* максимальное значение для переменной *e*, используемой в формуле представления числа $v = M \times 2^e$	15	127	1023	16383	$2^{(k-p-1)}-1$
Параметры, используемые для кодирования					
bias, смещение ($E = e + bias \Rightarrow bias = E - e$)	15	127	1023	16383	e_{max}
sign bit; число бит для знака	1	1	1	1	1
w, exponent field width in bits ширина поля смещённого порядка в битах	5	8	11	15	$\text{round}(4 \times \log_2(k)) - 13$
t, trailing significand field width in bits; ширина поля для хранения хвоста мантиссы в битах = точночть *p* минус один мнимый разряд	10	23	52	112	$k-w-1$
k, storage width in bits; объём занимаемой памяти в битах	16	32	64	128	$1+w+t$;
наглядно, в битах					

В результате десятичное число 36,6, представленное в формате IEEE 754 с одинарной точностью (binary32), равно `66 66 12 42`$_{16\_BE}$, а в памяти ЭВМ с архитектурой LITTLE_ENDIAN его байты хранится в «перевёрнутом» виде как `42 12 66 66`$_{16\_LE}$.

Проверим наши теоретические выкладки опытным путём с помощью программы float_representation.c. (Недостающие файлы см. в разделе представления целых чисел.)

```
$ cat float_representation.c
#include <stdio.h>
#include <stdlib.h>

void PrintBinNumber(unsigned char *bytes, size_t count);
void PrintBinBytes(unsigned char *bytes, size_t count);
void PrintHexNumber(unsigned char *number, size_t bytesCount);
void PrintFloat(float f);
void PrintHexMemory(void * pointer, size_t bytesCount);

int main(int argc, char *argv[])
{
   if (argc != 2)
   {
      printf("Использование:\n %s number.\n", argv[0]);
      return -1;
   }

   float number = atof(argv[1]);
   printf("\t\tИсходное число: %f\n", number);
   printf("\t   IEEE-754 формат: "); PrintFloat(number); printf("\n");

   printf("Двоичное представление: ");
   PrintBinNumber((unsigned char *)&number, sizeof(number)); printf("\n");

   printf(" 16-чное представление: ");
   PrintHexNumber((unsigned char *)&number, sizeof(number)); printf("\n");

   printf("Представление в памяти: ");
   PrintBinBytes((unsigned char *)&number, sizeof(number)); printf("\n");
   PrintHexMemory(&number,sizeof(number));

   return 0;
}
$ cat PrintFloat.cpp
#include <stdio.h>
#include <limits.h>

void PrintFloat(float f)
{
   unsigned char highBit = 1 << (CHAR_BIT - 1); // 128 = 2^7 = (10000000)

   const int signBit = 0;
   const int biasedExponentEnd = 8;

   unsigned char *p = (unsigned char *)&f;
   int bitNumber = 0;
   for (int i = sizeof(f) - 1; i >=0; --i)
   {
      unsigned char nextByte = p[i];
      for (int i = 0; i < CHAR_BIT; ++i)
      {
         bool isSet = (nextByte & highBit) != 0;
         printf("%d", isSet ? 1 : 0);
         if (bitNumber == signBit || bitNumber == biasedExponentEnd)
            printf(" ");
         nextByte <<= 1;
         ++bitNumber;
      }
   }
}
```

Программу компилируем и запускаем:

```
$ g++ float_representation.c PrintBinBytes.cpp PrintBinByte.cpp PrintBinNumber.cpp PrintHexNumber.cpp PrintHexByte.cpp PrintHexMemory.c PrintFloat.cpp -o float_representation
$ ./float_representation 36.6
        Исходное число: 36.599998
       IEEE-754 формат: 0 10000100 00100100110011001100110
Двоичное представление: 01000010 00010010 01100110 01100110
 16-чное представление:    42       12       66       66
Представление в памяти: 01100110 01100110 00010010 01000010
по адресу 0x7fff36fec51c: 66
по адресу 0x7fff36fec51d: 66
по адресу 0x7fff36fec51e: 12
по адресу 0x7fff36fec51f: 42
```

2.1.4.4. Общий случай для действительных чисел: 10 диапазонов, 5 алгоритмов преобразований, 2 формулы!

Практическая проверка правильности вычислений показала, что предыдущие преобразования верны. Если проводимые вычисления не показались вам сложными и вы считаете, что уже умеете преобразовывать любые числа по формату IEE754, то спешим вас огорчить, рано радуетесь. Если бы мы говорили о формате хранения целых беззнаковых чисел, то можно было бы «остановиться», так как для них используется всего лишь одна формула преобразований. Если добавить знак, то диапазонов, как и формул, станет два. В случае же с действительными числами забежим вперёд и скажем, что диапазонов будет аж 10, поэтому для наглядности они сопоставлены с цветами. Алгоритмов в два раза меньше, так как для «+» и «−» они практически одинаковые. В примере выше мы использовали лишь один диапазон, ниже разберём остальные. В этом, наверное, и кроется основная проблема изучения формата хранения чисел, когда обучаемыми рассматриваются лишь некоторые случаи преобразований, а все остальные уходят из рассмотрения.

Понять сказанное проще, проведя рассуждения «от обратного», а именно «от памяти к числу». Так как для хранения данных одной переменной типа float у нас задействовано 32 двоичных разряда, то максимальное число вариантов будет 2^{32}, начиная 32 «нулями» и заканчивая всеми «единицами». Записывать числа удобнее в более компактной шестнадцатеричной форме, где начало и конец будут 00 00 00 00 и FF FF FF FF, соответственно. См. табл. 2.8 *а*.

Таблица 2.8 а. Все варианты 32-х разрядов, интерпретируемых как число типа float

№ комбинации	Комбинация битов в памяти (прямая последовательность записи, Big Endian)	Комбинация сокращённо (Big Endian)	Комбинация в порядке LittleEndian
1	00000000 00000000 00000000 00000000	00 00 00 00	00 00 00 00
2	00000000 00000000 00000000 00000001	00 00 00 01	01 00 00 00
3	00000000 00000000 00000000 00000010	00 00 00 02	02 00 00 00
4	00000000 00000000 00000000 00000011	00 00 00 03	03 00 00 00
...	...	...	...
$2^{32}-1$	11111111 11111111 11111111 11111110	FF FF FF FE	FE FF FF FF
2^{32}	11111111 11111111 11111111 11111111	FF FF FF FF	FF FF FF FF

Если попытаться сопоставить коду хранящемуся в памяти (двоичному значению 4-байтовой переменной) действительное число которое этим кодом кодируется (по стандарту IEEE-754), то удобнее будет получившуюся «таблицу соответствия» поделить на 10 последовательных диапазонов (множеств), по числу формул которыми они могут быть преобразованы в действительные числа. (См. табл. 2.8 б.)

Таблица 2.8 б. Все варианты 32-х разрядов, интерпретируемых как число типа float

Диапазон	Сопоставленное значение числа	Цвет	№ комбинации = запись в памяти (BE)	Запись в памяти (Little Endian)
1	+0		00 00 00 00	00 00 00 00
2	$\approx 1{,}40129846 \cdot 10^{-45}$		00 00 00 01	01 00 00 00
	...	...	...	...
	$\approx 1{,}17549421 \cdot 10^{-38}$		00 7F FF FF	FF FF 7F 00
3	$\approx 1{,}17549435 \cdot 10^{-38}$		00 80 00 00	00 00 80 00
	...	...	...	...
	...	...	...	...
	...	...	...	...
	...	...	...	...
	$\approx 3{,}40282347 \cdot 10^{+38}$		7F 7F FF FF	FF FF 7F 7F
4	+∞		7F 80 00 00	00 00 80 7F
5	NaN (не число)		7F 80 00 01	01 00 80 7F
	...	...	...	...
	NaN (не число)		7F FF FF FF	FF FF FF 7F
6	−0		80 00 00 00	00 00 00 80
7	$\approx -1{,}40129846 \cdot 10^{-45}$		80 00 00 01	01 00 00 80
	...		...	...
	$\approx -1{,}17549421 \cdot 10^{-38}$		80 7F FF FF	FF FF 7F 80
8	$\approx -1{,}17549435 \cdot 10^{-38}$		80 80 00 00	00 00 80 80
	...	...	...	...
	...	...	...	...
	...	...	...	...
	...	...	...	...
	$\approx -3{,}40282347 \cdot 10^{+38}$		FF 7F FF FF	FF FF 7F FF
9	−∞		FF 80 00 00	00 00 80 FF
10	NaN (не число)		FF 80 00 01	01 00 80 FF
	...		...	...
	NaN (не число)		FF FF FF FF	FF FF FF FF

Далее, если попытаться мысленно разместить более-менее адекватно все диапазоны на числовой оси (как мы привыкли в школе слева направо видеть ось X), то получится примерно следующая картинка, взятая из [17]. см. рис. 2.3.

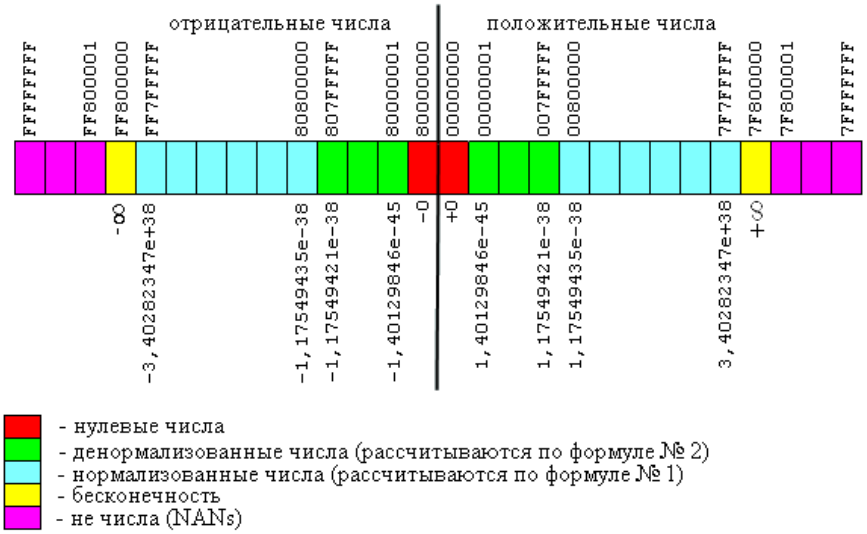


Рисунок 2.3. Диапазон чисел формата одинарной точности (32 бита), представленных по стандарту IEEE 754 [32]

Что мы видим на рисунке? Над воображаемой осью координат «X» записаны значения ячеек памяти в формате big endian. Под осью записаны значения чисел, а цветом указаны алгоритмы, по которым можно осуществить преобразования, причём с целью упрощения и исходя из симметрии их число (как и цветов на рисунке) сокращено до пяти. «Пробежимся» по положительной части оси и разберём их, но перед этим посмотрим на рисунок 2.3 и в таблицу *2.8 б. Все варианты 32-х разрядов, интерпретируемых как число типа float* на стр. 59, заметим, что для формата binary32 диапазон нормализованных чисел самый большой среди всех остальных и он занимает более >99% от всех возможных комбинаций. Для других форматов доля немного отличается, но она также велика и составляет не менее девяти десятков процентов при грубой оценке снизу.

формат	binary16	binary32	binary64	binary128
w	5	8	11	15
формула расчёта $(2^w-2)/2^w$	$\frac{2^5-2}{2^5}$	$\frac{2^8-2}{2^8}$	$\frac{2^{11}-2}{2^{11}}$	$\frac{2^{15}-2}{2^{15}}$
~% нормализованных чисел от числа всех комбинаций	93,75	99,22	99,90	99,99

Поэтому, начнём рассмотрение с самого большого диапазона, соответствующих ему алгоритма и формулы, предзначенных для кодирования чисел представимых в нормализованном виде.

[32] http://www.softelectro.ru/ieee754_14.gif. Обозначение 1,23e+45 = $1{,}23\cdot10^{+45}$.

2.1.4.4.1. Диапазон нормализованных чисел

Рассмотрим общую формулу преобразования и какие из неё следуют ограничения.

Стандарт IEEE 754 по этому поводу предлагает следующее хранение данных в памяти (см. рис. 2.4), где переменные берутся из таблицы 2.7, в зависимости от желаемой точности.

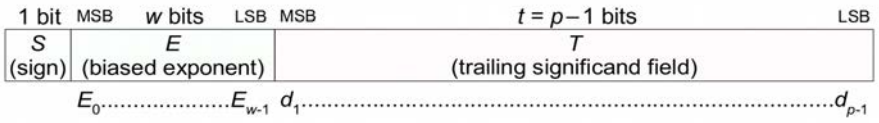


Рисунок 2.4. Распределение битов в памяти ЭВМ для хранения чисел с плавающей запятой по стандарту IEEE754 (раскраска цветами авторская)

Работа с первоисточниками есть основа знаний, но не всегда они удобны для восприятия рядовыми читателями, в связи с этим предлагаем обратиться к «адаптированным фрагментам» стандарта от Владимира Яшкардина в [17] и от Сергея Холодилова в [25], где указанные вопросы изложены намного лучше, так как на ряду с наглядными картинками приводятся различные примеры. К минусам этих работ можно отнести отличные от стандарта обозначения. Мы постараемся сохранить наглядность и избавимся от недостатков.

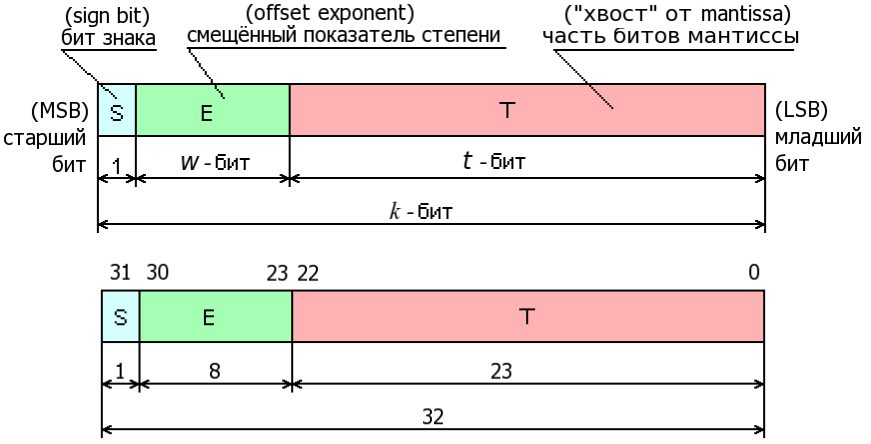


Рисунок 2.5. Распределение битов памяти по формату IEEE754 при точности binary32 (w = 8, t = 23, k = 32) (верх – общая формула; низ – binary32)

Обозначения, используемые на рис. 2.5:

- S – бит знака, $S = \begin{cases} 0 - \text{положительное число «+»} \\ 1 - \text{отрицательное число «−»} \end{cases}$;
- E – смещённый показатель степени двоичного числа;
- T – остаток мантиссы двоичного нормализованного числа с плавающей запятой (в [17] и [25] и др. литературе может обозначается через M, что вносит путаницу, а для T есть удобное совпадение в англ. *tail* означает «хвост»).

Дадим некоторые пояснения. Переход от хранимого в памяти смещённого показателя E к реальному выполняется вычитанием смещения, которое в обозначениях стандарта равно $e = (2^{(w-1)} - 1)$, ~~что ≡ $(2^{(b-1)} - 1)$ в обозначениях из [17]~~.

Замечание. Просьба не путать с основанием натурального логарифма (числом Непера), обозначаемом тоже через e, и равном результату вычисления второго замечательного предела.)

То есть, для расчётов реальная степень числа 2 вычисляется как «хранимое в памяти значение E» минус смещение e, то есть $E - (2^{(w-1)}-1)$, для *binary32* получится $E-127$.

Дополнительная сложность для читателей в восприятии материала состоит в том, что через букву T обозначен остаток мантиссы (хвост), а не сама мантисса. Почему так оказывается удобнее, будет разъяснено позднее. Нормализация подразумевает, что для мантиссы будет выполняться следующее условие:

$$1 \le | M_{реальная} | < Z$$

, где Z – основание системы счисления. Так как мы рассматриваем положительные числа, то модуль в условии можно опустить. Для отрицательной мантиссы всё аналогично с точностью до знака, который, согласно формату, хранится в первом бите записи числа. Поскольку мы используем двоичную систему счисления, её основание $Z = 2$. Следовательно опиванное выше условие будет выглядеть следующим образом:

$$1_{10} \le M_{реальная\ 10} < 2_{10}.$$

Запишем эти же самые ограничения условий в двоичной системе счисления. Получим $1_2 \le M_{реальная\ 2} < 10_2$. Допишем не значащие (с точки зрения математики) нули для наглядности. Получим $1{,}00000000..._2 \le M_{реальная\ 2} < 10{,}00000000..._2$, то есть $M_{реальная\ 2}$ есть число вида 1,xxxxxxxx..., где, как уже легко заметить, первая цифра в записи всегда равна 1.

Если мы хотим записать в память t знаков дробной части числа 1,xxxxxxxx..., придётся вычесть из него единицу (*Какой смысл её хранить, занимая ячейку памяти, если она всегда равна 1?*), после чего мы получим 0,xxxxxxxx..., далее следует «передвинуть» запятую на t разрядов вправо, что для двоичной системы равносильно умножению числа на 2^t.

То есть, общая формула преобразования будет $T = (M_{реальная} - 1) \cdot 2^t$, выразим из этого выражения $M_{реальная}$:

$$M_{реальная} = \left(1 + \frac{T}{2^t}\right) \quad .$$

Сопоставив изложенное выше, получим общую формулу, связывающую представляемое число и хранимые в памяти ЭВМ данные:

$$v = (-1)^S \cdot 2^{(E - 2^{(w-1)} + 1)} \cdot \left(1 + \frac{T}{2^t}\right) \quad , \qquad \text{(формула № 1)}$$

для формата *binary32* из табл. 2.7 подставим следующие значения размеров полей: $w = 8$, $t = 23$, $k = 32$, в результате чего получим:

$$v = (-1)^S \cdot 2^{(E - 127)} \cdot \left(1 + \frac{T}{2^{23}}\right) \quad . \qquad \text{(формула № 1) для } binary32$$

Замечание. Напомним, что в некоторых источниках литературы вместо переменной v используется F, так и другие эквивалентные обозначения $w \equiv b$, $t \equiv n$.

Замечание 2. В большинстве источников литературы, поступают примерно также, начиная с рассмотрения общего вида экспоненциальной записи нормализованных чисел и формулы №1 для представления нормализованных чисел, тем самым сразу закрывая первые 80% от общего объёма информации по данному вопросу. Где-то этим изложением ограничиваются, оно и понятно, ведь по закону Парето, известному как «80/20», на оставшиеся и не рассмотренные 20% материала уйдёт времени в 4 раза больше. Продолжив читать дальше, вы в этом сами убедитесь.

2.1.4.4.2. Ноль (нуль)

Вспоминая когда-то и где-то услышанное «*Ноль – он и в Африке ноль*», так и хочется возразить, спросив, а так ли это на самом деле? Ведь, формально, используя нормализованную форму записи в двоичной системе счисления, как и формулу № 1, невозможно записать число 0.

Давайте проверим это и запишем в биты хвоста мантиссы все нули – получим мантиссу равную единице, точнее 1,00000000... И в поле смещённого показателя степени тоже запишем какое-нибудь число дающее, после вычитания из него смещения, так же ноль. Тогда, получим порядок равный 2^0, что по определению есть 1, следовательно, формула № 1, где используется произведение этих чисел $2^0 \times 1{,}00000000...$, даст в результате единицу, а не ноль.

Обмануть решили?!, – скажет наблюдательный читатель, а если мы используем ранее представленную программу float_representation и с её помощью посмотрим на представление числа 0 в памяти реальной ЭВМ, то ведь мы увидим нули?

```
$ ./float_representation 0
     Исходное число: 0.000000
      IEEE-754 формат: 0 00000000 00000000000000000000000
Двоичное представление: 00000000 00000000 00000000 00000000
 16-чное представление:    00       00       00       00
Представление в памяти: 00000000 00000000 00000000 00000000
```

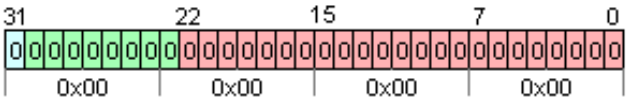
Конечно нули! Нет основания не верить своим глазам и программе.

Раз, так,... – скажет читатель, – *давайте проверим и подставим эти проверенные практикой нули, полученные из памяти реально работающей ЭВМ, в уже проверенную примером «36,6» и только что рассмотренную формулу № 1!*

И что же дальше? Подсчитав результат, мы получим $5{,}87747175411144 \cdot 10^{-39}$ – самое минимальное положительное число, представимое такой формулой. Аналогично (с точностью до наоборот – макимальное отрицательное), для числа со знаком «минус»: $-5{,}87747175411144 \cdot 10^{-39}$.

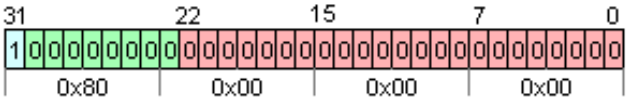
Из изложенного выше делаем вывод: это те случаи когда формула № 1 не работает напрямую и люди, создававшие стандарт IEEE754, договорились сделать из неё первое исключение, а именно, считать для формата binary32 получившиеся выше, в результате расчётов по формуле № 1, числа нулями системы. Логично, что подобными подсчётами никто не занимается и никакую замену в памяти не делает, поскольку для форматов отличных от binary32 числа получаются уже другие.

На примере binary32 стандарт IEEE754 гласит, что фрагмент памяти ЭВМ, содержащий значения $00\ 00\ 00\ 00_{16}$ (следует учитывать, что реальная последовательность хранения зависит от архитектуры, см. стр. 43, в данном случае записи для порядков BE и LE совпадают), будучи интерпретирован как число формата binary32 IEEE754-2008, считается числом «+0».



«+0»

Если рассматривать «отрицательную область», то там тоже будет «свой» ноль.



«−0»

Из рисунков видно, что при шестнадцатеричной записи в памяти отображение «–0» будет как $80\ 00\ 00\ 00_{16\_BE}$ или $00\ 00\ 00\ 80_{16\_LE}$.

Фактически именно эти числа являются представлением «+0» и «−0» в модели IEEE754. Ну а то, что число 0 есть в памяти $00\ 00\ 00\ 00_{16}$, как мы договорились выше, есть красивое совпадение. В двоичной записи это будет комбинация нулей во всех 32 разрядах. Согласитесь, что это тривиально и легко запомнить.

Самостоятельно экстраполировать изложенное для других форматов стандарта не составит труда.

Замечание. При кодировании целых чисел одним байтом от значения «−0» отказались в пользу ещё одного отрицательного числа, вот почему при 256 вариантах мы пользуемся не симметричным диапазоном [−127, +127], а асимметричным [−128, +127], как и [−32768, +32767] и т. д. Смеем предположить, что «+0» и «−0», с точки зрения математики, если не рассматривать пределы справа и слева, есть одно и то же – «0» (+0 = −0 = 0), а в чём различие «+» и «−» – вопрос философский. Однако, из факта существования двух нулей технически можно сделать интересные выводы, например для операции сравнения. Если ранее, при использовании целых чисел, из равенства «представления чисел в памяти ЭВМ» следовало и равенство самих чисел, то для типов с плавающей запятой одним сравнением уже не обойтись. Например, при побитном сравнении фрагментов памяти $80\ 00\ 00\ 00_{16\_BE}$ и $00\ 00\ 00\ 00_{16\_BE}$ (или $00\ 00\ 00\ 80_{16\_LE}$ и $00\ 00\ 00\ 00_{16\_LE}$), хранящих числа с плавающей запятой, видно, что они различаются, а значит, можно ошибочно предположить, что числа, представленные данными записями, разные, по факту же мы понимаем, что числа равны.

Замечание, открытый вопрос. Почему комбинация $80\ 00\ 00\ 00_{16\_BE}$ не была использована для представления какого-нибудь ещё числа, как это было сделано с целыми типами?

Смеем предположить, что с целыми типами, при использовании 8 бит, «выигрыш» оказался существен, плюс несложность добавления числа «на край» отрицательных чисел подсказала простое решение, как использовать это значение. Целые типы с большим числом разрядов просто унаследовали данный подход. Что же касается чисел с плавающей запятой, видимо, даже при разработке стандарта IEEE754 сочли вклад $1/2^{16}$ для формата Binary16 несущественным, не говоря уже о меньшем вкладе $1/2^{32}$ для формата binary32 и т. д. Следует отметить, что в диапазон для NaN, о котором написано ниже, заложено больше неиспользуемых значений.

2.1.4.4.3. Диапазон денормализованных чисел

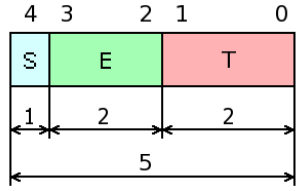
(*англ. denormalized numbers, subnormal numbers*)

Диапазон денормализованных чисел – это следующий случай, где формула № 1 не работает и в отношении него следует договориться о ещё одном исключении.

Перед тем как мы придём к заключению, что для данного диапазона формулу № 1 использовать нельзя, ответим на вопрос: «*А что же будет, если всё же это сделать?*», а также на ещё один – «*В чём преимущество денормализованных чисел?*»

Рассуждения ниже по большей части основаны на материале, взятом из [19], который был переработан и дополнен, устранены ошибки.

Для упрощения изложения рассмотрим «игрушечный» формат «*несуществующий binary5*», в котором возможно не только представить, но и перебрать все $2^5 = 32$ вариантов чисел для большей ясности. В процессе изучения созданного псевдоформата мы получим, а значит и читатели узнают, ответы на поставленные вопросы.



Начнём. Цифра 5 у binary5 говорит о том, что для хранения чисел в памяти будет использоваться пять бит «xyyzz». Распределим их следующим образом: отведём один бит на кодирование знака (поле «S»), два бита на кодирование показателя степени (поле «E» – смещённый показатель степени), два бита для хранения части мантиссы (поле «T» – для двух первых битов хвоста мантиссы). Для поля знака S имеем всего два варианта: «0» и «1». Пусть «0» кодирует знак «+», а «1» – знак «−». Представим типовые параметры формата «*несуществующий binary5*» в табл. 2.9 на ряду со значениями существующего формата binary32 стандарта IEEE 754-2008.

Точность числа определяется количеством знаков в мантиссе. Так как нормализованная мантисса для двоичных чисел всегда имеет вид $1{,}xxxxxxxx_2$ (ноль мы не рассматриваем). При записи этого числа в двоичные разряды, получается, что первый разряд всегда оказывается равен 1. Хранить его в условной физической 1-битовой ячейке памяти нет смысла, поэтому подразумевается что он хранится «неявно», то есть не хранится вообще, а, при необходимости выполнить вычисления, восстанавливается алгоритмом. Поскольку под кодирование хвоста мантиссы мы отвели 2 разряда, общая точность *p* будет на один (тот самый, мнимый) разряд больше и получится равной 3.

Далее, по формуле $2^{(k-p-1)} - 1$ рассчитывается максимальное значение показателя степени $emax = 2^{(5-3-1)} - 1 = 1$, а из него определяется смещение $bias = 1$.

В результате получим следующие параметры для кодирования см. табл. 2.9.

Таблица 2.9. Параметры формата «несуществующий binary5»

Параметр	∄ binary5	binary32
k, storage width in bits	5	32
p, precision in bits, (*t*+1)	3 = 2 + 1	24
e_{max}, maximum exponent	1	127
bias	1	127
sign bit	1	1
w, exponent field width in bits	2	8
t, trailing significand field width in bits	2	23

В предложенном 5-битовом формате, если отказаться от всех требуемых ограничений реальных форматов (о невозможности представления нуля; *e* должно быть $\le e_{max}$; нет никаких бесконечностей и не_чисел), и исключительно следовать формуле № 1, то, перебирая все возможные комбинации битов мы получим 16 чисел со знаком «+» (см. табл. 2.10) и симметрично столько же со знаком «−». Размещение первых на числовой оси представлено на рис. 2.6.

Таблица 2.10. Соответствие кодов положительных чисел их математически рассчитанным результатам по формуле для нормализованных чисел (формула № 1)

Код числа	Вычисление степени	Мантисса	Расчёт числа	Результат
00000	$00_2=0_{10}$; 0-1=-1	$1,00_2$	$+2^{-1}\cdot 1,00_2 = 0,100_2$	$0,5_{10}$

Код числа (S E T)	Вычисление степени, *e* (*E* −*смещение* = *E* −$(2^{(w-1)}-1)$)	Мантисса (M=1, T)	Расчёт числа (± 2^e × M)	Результат (в 10 сист. счисл.)	
00000	$00_2=0_{10}$; 0−1=−1	$1,00_2$	$+2^{-1}\cdot 1,00_2 = 0,100_2$	0,5	«уйдут»
00001	$00_2=0_{10}$; 0−1=−1	$1,01_2$	$+2^{-1}\cdot 1,01_2 = 0,101_2$	0,625	
00010	$00_2=0_{10}$; 0−1=−1	$1,10_2$	$+2^{-1}\cdot 1,10_2 = 0,110_2$	0,75	
00011	$00_2=0_{10}$; 0−1=−1	$1,11_2$	$+2^{-1}\cdot 1,11_2 = 0,111_2$	0,875	
00100	$01_2=1_{10}$; 1−1=0	$1,00_2$	$+2^{0}\cdot 1,00_2 = 1,00_2$	1	останутся
00101	$01_2=1_{10}$; 1−1=0	$1,01_2$	$+2^{0}\cdot 1,01_2 = 1,01_2$	1,25	
00110	$01_2=1_{10}$; 1−1=0	$1,10_2$	$+2^{0}\cdot 1,10_2 = 1,10_2$	1,5	
00111	$01_2=1_{10}$; 1−1=0	$1,11_2$	$+2^{0}\cdot 1,11_2 = 1,11_2$	1,75	
01000	$10_2=2_{10}$; 2−1=1	$1,00_2$	$+2^{1}\cdot 1,00_2 = 10,0_2$	2	
01001	$10_2=2_{10}$; 2−1=1	$1,01_2$	$+2^{1}\cdot 1,01_2 = 10,1_2$	2,5	
01010	$10_2=2_{10}$; 2−1=1	$1,10_2$	$+2^{1}\cdot 1,10_2 = 11,0_2$	3	
01011	$10_2=2_{10}$; 2−1=1	$1,11_2$	$+2^{1}\cdot 1,11_2 = 11,1_2$	3,5	
01100	$11_2=3_{10}$; 3−1=2	$1,00_2$	$+2^{2}\cdot 1,00_2 = 100,0_2$	4	«уйдут»
01101	$11_2=3_{10}$; 3−1=2	$1,01_2$	$+2^{2}\cdot 1,01_2 = 101,0_2$	5	
01110	$11_2=3_{10}$; 3−1=2	$1,10_2$	$+2^{2}\cdot 1,10_2 = 110,0_2$	6	
01111	$11_2=3_{10}$; 3−1=2	$1,11_2$	$+2^{2}\cdot 1,11_2 = 111,0_2$	7	

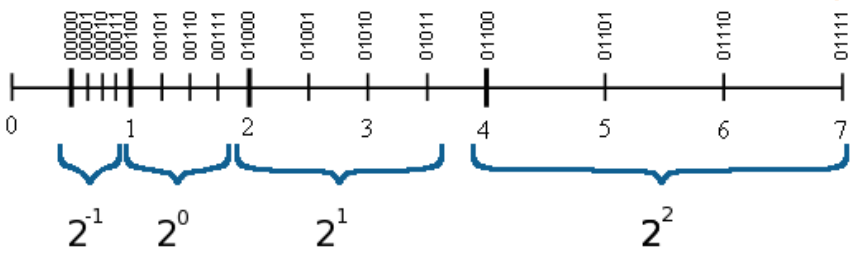


Рисунок 2.6. Отображение битового представления кодов чисел на числовую ось [33]

Легко заметить, что соседние битовые представления, в отличие от представления целых чисел, разместились на числовой оси неравномерно. Крупными штрихами на рис. 2.6 показаны числа с мантиссой, равной $1,00_2$, так как при переходе через них наблюдается изменение степени в представлении числа, а значит, скачкообразное (в 2 раза – по основанию системы счисления) изменение плотности размещения чисел на числовой оси.

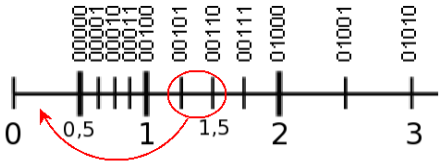
Невооружённым глазом видно, что расстояние между двумя соседними числами

[33] Произведён перебор всех «положительных» вариантов. Расчёт значений выполнен по формуле № 1 для нормализованных чисел (см. табл. 2.10).

слева «от места скачка» оказывается в 2 раза «ближе» чем между двумя соседними числами идущими справа. Например, сравните визуально от числа 2 слева на оси расстояние «$1{,}50_{10} \div 1{,}75_{10}$» и справа – «$2{,}50_{10} \div 3{,}00_{10}$».

Также, обратите внимание, на другое место на числовой оси. Расстояние от нуля до ближайшего к нему числа (0 ÷ 0,5) больше, чем от этого числа к следующему (0,5 ÷ 0,625). Причём, оно оказывает больше чем в 2 раза!

На практике это значит, что разница двух любых чисел от $0{,}5_{10}$ до 1_{10} даст 0, даже если эти числа не равны. Что ещё хуже, в «пропасть» между 0,5 и 0 попадает разница чисел больших 1. Например, «1,5 − 1,25 = 0».



Неформально этот эффект называют «нулевой или околонулевой ямой».

Логичным решением данной проблемы было бы заполнение «ямы» от 0 до 0,5 числами, следующими так же часто, как и от 0,5 до 1. Но увы, «память у компьютера не резиновая», количество отображаемых в модели чисел напрямую определяется отведённым для их хранения числом бит. Точнее, числом их уникальных комбинаций. В примере выше, если не брать в рассмотрение знаковый бит, мы имеем 4 разряда, что и определяет $2^4 = 16$ вариантов. В этом случае вопрос состоит в том, как грамотно отобразить все 16 возможных вариантов (для *float* без знака будет 2^{31}) на числовую ось, чтобы избежать «нулевой ямы». Визуально выход напрашивается один – «растянуть значения диапазона 0,5 – 1 к нулю» (см. рис. 2.7).

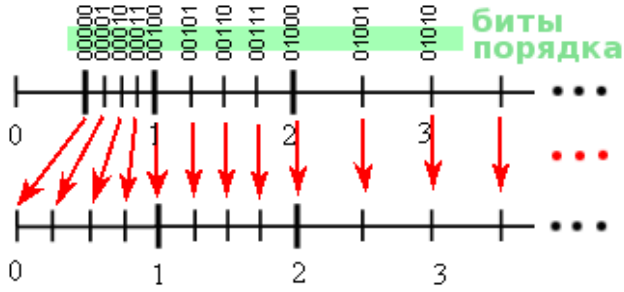
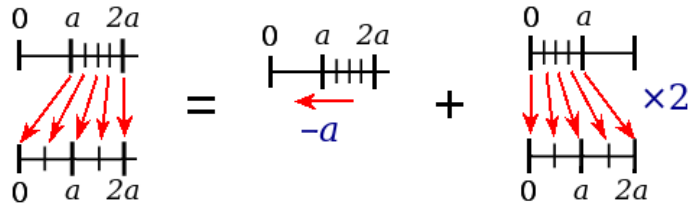


Рисунок 2.7. Решение проблемы «околонулевой ямы»

Если внимательно проанализировать представленное на рис. 2.7 решение, то становится ясно, что «перемещению» подверглись все числа, у которых в битах порядка стоит «00» (такие числа называются *денормализованными* (*англ. denormalized numbers, subnormal numbers*)). [19] Это означает, что для формального преобразования рассматриваемого битового формата в десятичное число нужны два разных алгоритма или две разных формулы.

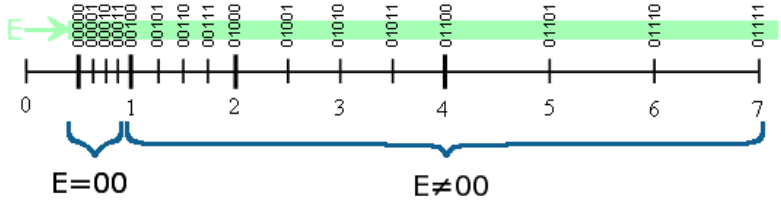
Для чисел, биты порядка которых равны нулю – одну, а для всех остальных – другую. Позже мы увидим это в табл. 2.11.



Указанная стрелками операция перемещения части чисел идентична для них двум действиям: смещению указанных чёрточек влево – этому следует вычитание выливающееся в «замену у мантиссы мнимой единицы на мнимый ноль» и изменению масштаба следования чёрточек на оси – этому следует умножение на 2 или что тоже самое – увеличение показателя степени числа 2 на +1.

Актуальность проблемы «околонулевой ямы» проявилась ещё в 70-х годах прошлого века: в то время в среднем каждый компьютер сталкивался с ней один раз в месяц [19]. Учитывая, что компьютеры приобретали массовость, разработчики «K-C-S» (Уильям Кэхэн, Джероми Кунен и Гарольд Стоун, сокращение из первых букв их фамилий, записанных на английском языке) посчитали эту проблему достаточно серьёзной, чтобы решить её на аппаратном уровне. Схема на рис. 2.7 иллюстрирует суть этого решения, которое в стандарте IEEE754-2008 представлено следующим образом.

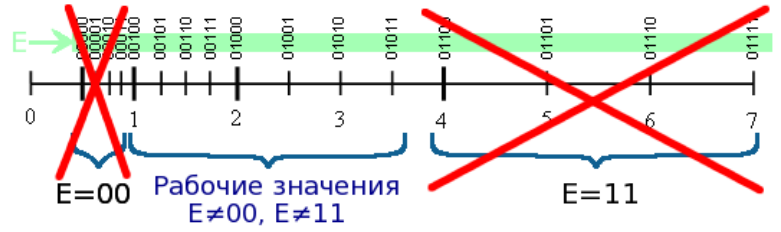
Замечание. Прежде чем сообщить некоторую информацию из стандарта, отметим следующее. Из табл. 2.10 и рис. 2.6 видно, что на всём диапазоне представимых чисел в их кодах у некоторых поле E оказывается полностью нулевым (00 для binary5), а у некоторых оно отлично от нуля. При этом, так совпало что уже введённые нами два ограничения по использованию формулы № 1 (для нуля и денормализованных чисел, кстати, рассматриваемых нами сейчас) как раз приходятся на ситуацию когда поле E=0.



Для всех остальных значений E, то есть E ≠ 0, формула № 1 успешно работает как можно предположить из ожидаемого расположению «чёрточек», соответствующих закодированным числам, на оси выше.

В последующих параграфах на выбор E при использовании формулы № 1 будут наложены дополнительные ограничения. В частности формула № 1 не будет работать в отношении чисел у которых в поле E их кода записаны все единицы (E=11 для binary5).

То есть, рабочими для формулы № 1 являются лишь те значения E, которые не являются ни всеми нулями, ни всеми единицами. В табл. 2.10 половина значений оказываются не верны.



Если $1 \le E \le 2^w - 2$ (число нормализованное, случаи $E \ne 0$ и E состоит из всех единиц исключены), то число представимо тремя независимыми значениями:

$$(S, (E - bias), (1 + 2^{1-p} \times T)).$$

из которых можно без труда выделить соответствующие поля кода (S,E,T – выделены цветом), а значит, составить и получить весь двоичный код, так и получить численное значение соответствующего числа с плавающей запятой, вычислив его по формуле:

$$v = (-1)^S \times 2^{E-bias} \times (1 + 2^{1-p} \times T).$$

Из написанного видно, что нормализованные числа имеют «подразумевающуюся» («мнимую») единицу при кодировании мантиссы. Вот она.

Эта часть осталась без изменений.

Если $E = 0$ и $T \ne 0$ (число денормализованное и исключён случай «нуля»), тогда число представимо тремя независимыми значениями:

$$(S, e_{min}, (0 + 2^{1-p} \times T)).$$

Замечание. Появившаяся новая переменная e_{min} равна минимальному значению e, при котором заканчиваются исключения из формулы № 1 и она начинает работать. Поскольку вначале диапазона исключается только случай E=0, вычислить следующее за ним работающее значение (01 для binary 5), а из него и показатель степени e, который и будет как раз e_{min} не составит труда. Напомним что $e = E - смещение = E - (2^{(w-1)} - 1)$. Для binary5 подставим $w = 2$ и получим $e_{min} = 0$ (тоже самое для формата IEEE754-2008 binary32: $w = 8$, $e_{min} = -127$). Из рис. 2.6 и 2.7 наглядно видно, что рассчитанное таким образом значение верно. А именно, при проверке исходим из утверждения, что значение показателя степени отвечает за плотность расположения чисел на прямой. Рис. 2.6 наглядно демонстрирует этот факт. Визуальная плотность расположения точек для денормализованных чисел после «растягивания и смещения» на рис.2.7 соответствует плотности интервалов между числами для «2^0» рисунка 2.6. Это также сочетается с обсуждавшимся «растягиванием» в 2 раза, так как 2^0 и 2^{-1} отличаются ровно в 2 раза.

Значение соответствующего числа с плавающей запятой вычисляется по формуле:

$$v = (-1)^s \times 2^{e_{min}} \times (0 + 2^{1-p} \times T) \equiv (-1)^s \times 2^{e_{min}} \times \left(0 + \frac{T}{2^{p-1}}\right).$$

Это значит, что поле смещённого порядка двоичного представления денормализованного числа заполнено нулевыми битами, значение несмещённого порядка равно минимальному значению несмещённого порядка для нормализованных чисел (фактически на единицу больше от расчётного недействующего значения), а неявный старший бит мантиссы имеет нулевое значение («мнимый ноль»).

Таким образом, числа с плавающей запятой можно представить следующим образом:

- нормализованные: $(-1)^S \times 1{,}T \times 2^{E - bias}$, если $1 \le E \le 2^w - 2$;
- денормализованные: $(-1)^S \times 0{,}T \times 2^{1 - bias}$, если $E = 0$;
- иные числа (см. последующие параграфы), если в E побитно записаны все 1.

Эти формулы позволяют представить визуальное «растяжение», показанное на рис. 2.7, и внести изменения, представленные в табл. 2.11.

Таблица 2.11. Соответствие кодов чисел их новым результатам полученным по формуле для денормализованных чисел, ранее рассчитанные неправильные значения зачёркнуты

Код числа	Вычисление степени	Мантисса	Расчёт числа	Результат
00000	~~$00_2=0_{10}$; 0−1=−1~~ $E = 00_2=0_{10}$; => формально формула со смещением не работает, поэтому берём e_{min}; $e_{min} = E - bias + 1 = 0$	~~$1{,}00_2$~~ $0{,}00_2$	~~$+2^{-1}\cdot 1{,}00_2 = 0{,}100_2$~~ не используется, т.к. это исключение; если посчитать, то будет $+2^0\cdot 0{,}00_2 = 0{,}00_2$	~~$0{,}5_{10}$~~ 0 (оговорённое исключение, результат исключения совпал с рассчитанным значением)
00001	~~$00_2=0_{10}$; 0−1=−1~~ $00_2=0_{10}$; => 0	~~$1{,}00_2$~~ $0{,}01_2$	~~$+2^{-1}\cdot 1{,}01_2 = 0{,}101_2$~~ $+2^0\cdot 0{,}01_2 = 0{,}01_2$	~~$0{,}625_{10}$~~ $0{,}25_{10}$
00010	~~$00_2=0_{10}$; 0−1=−1~~ $00_2=0_{10}$; => 0	~~$1{,}00_2$~~ $0{,}10_2$	~~$+2^{-1}\cdot 1{,}10_2 = 0{,}110_2$~~ $+2^0\cdot 0{,}10_2 = 0{,}10_2$	~~$0{,}75_{10}$~~ $0{,}50_{10}$
00011	~~$00_2=0_{10}$; 0−1=−1~~ $00_2=0_{10}$; => 0	~~$1{,}00_2$~~ $0{,}11_2$	~~$+2^{-1}\cdot 1{,}11_2 = 0{,}111_2$~~ $+2^0\cdot 0{,}11_2 = 0{,}11_2$	~~$0{,}875_{10}$~~ $0{,}75_{10}$

В результате получаем новое представление для блока денормализованных чисел (см. Рис. 2.8).

Рисунок 2.8. Распределение чисел с плавающей запятой по числовой оси в случае использования двух формул: для нормализованных и денормализованных чисел

Интервал от 0 до 1 заполняют денормализованные числа, что даёт возможность «не проваливаться в 0» в рассмотренном выше примере (разность 1,5 − 1,25). Использование другой формулы для диапазона денормализованных чисел сделало модель чисел более устойчивой к ошибкам округления для чисел, близких к нулю.

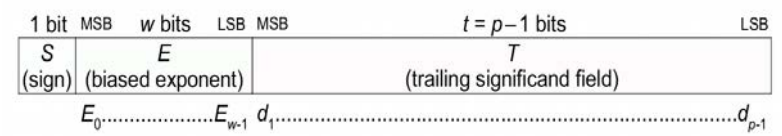
Замечание. Роскошь использования денормализованного представления чисел в процессоре не даётся бесплатно. Из-за того, что такие числа нужно обрабатывать по-другому во всех арифметических операциях, сделать работу в такой арифметике эффективной задача не из простых. Это накладывает дополнительные сложности при реализации АЛУ в процессоре. И хоть денормализованные числа очень полезны, они не являются панацеей, и за округлением до нуля всё равно нужно следить.

Итак, пояснив на примере несуществующего формата способ представления денормализованных чисел, подведём общий итог.

Денормализованные числа – это числа, мантиссы которых лежат в диапазоне $0_2 < M_2 < 1_2$, для вычисления которых используется формула:

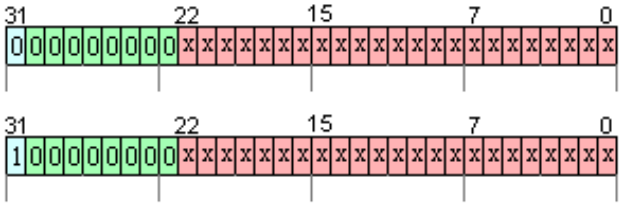
$$v = (-1)^S \times 2^{2-t-2^{w-1}} \times T \qquad \text{(формула № 2)},$$

где значения переменных *S*, *t* и *w* представлены на рис. 2.4. Ниже (на следующей странице) его копия, для удобства читателей.



Замечание. Естественно, формула № 2 может быть всевозможными способами преобразована и записана иначе. По большей части, именно этим и больше ничем, разные источники информации по данному вопросу отличаются друг от друга. Споры о том какая запись отношения и какие обозначения, нагляднее, эффективнее и лучше оставим для читателей.

Если смотреть на числа с точки зрения их записи в памяти (порядок BE), то денормализованными числами будут те, у которых в битах показателя степени окажутся все нули (кроме случая с представлением нуля, рассмотренного в самом начале). Наглядно это можно отобразить следующим образом:



Указанные числа считаются денормализованными числами, за исключением чисел «−0» и «+0» (когда все ячейки с записями «x» окажутся нулевыми).

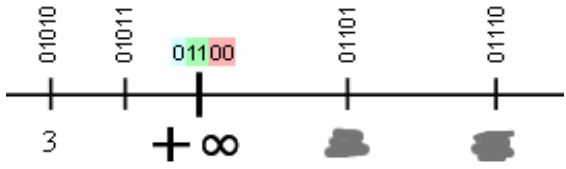
Другими словами, в двоичном представлении такого числа все биты поля смещённого порядка *E* имеют нулевое значение, а в поле хвоста денормализованной мантиссы *T* имеется по крайней мере один единичный бит.

2.1.4.4.4. Бесконечность (∞)

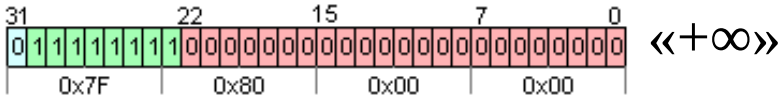
Так как ни по формуле № 1, ни по формуле № 2 представить число «+∞» не представляется возможным, при создании формата представления чисел с плавающей запятой в памяти ЭВМ, стали договариваться о включении данного числа в представимые форматом IEEE754 в качестве ещё одного исключения.

Если ноль, как было обсуждено выше, представляется всеми нулями, то логично было бы представить бесконечность единицами во всех разрядах (кроме знака). Однако так не поступили, возможно из-за потенциально сложных технических решений, или по иным соображениям, предложив рассматривать числа с максимально возможной степенью отдельно.

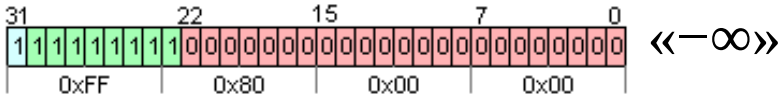
Так, числа, у которых по представлению рис. 2.6 в смещённом показателе степени окажутся все единицы, следует рассматривать отдельно (для формата «binary5» это *E* = 11 и числа 4, 5, 6 и 7), а наименьшее из этих чисел, согласно формуле № 1, то есть то, где все биты мантиссы будут равны нулю, следует считать числом «+∞».



Для формата *binary32* IEEE754 это будет число записанное в памяти как $7F\ 80\ 00\ 00_{16\_BE}$ или $00\ 00\ 80\ 7F_{16\_LE}$. Наглядное для запоминмания битовое представление бесконечности выглядит следующим образом.



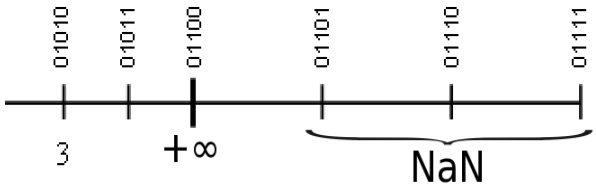
Аналогично со знаком «−», фрагмент памяти, содержащий значения $FF\ 80\ 00\ 00_{16\_BE}$ (или $00\ 00\ 80\ FF_{16\_LE}$), будучи интерпретирован по формату binary32 IEEE754, будет считаться ЭВМ как −∞.



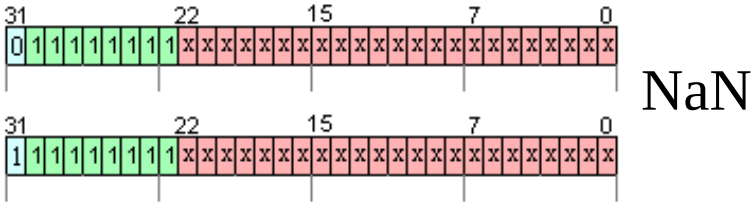
2.1.4.4.5. Не числа (NaN, Not a Number)

Если рассмотреть оставшиеся комбинации битов памяти, являющиеся также исключением из формулы № 1, то их интерпретация, согласно стандарту IEEE754-2008, составит множество «не чисел». Пишется *Not a Number* или сокращённо NaN. С одной стороны, может показаться, что для них выделен довольно большой диапазон комбинаций, который можно было бы отдать диапазону действительных чисел, расширив его. Так, в нашем примере выше с придуманным 5-битным форматом мы получили 16 положительных вариантов, где действительные числа лишись сразу 3 чисел из положительной области, которые стали «NaN».

В нашем небольшом диапазоне «NaN» оказались бы на месте пяти, шести и семи.



Именно поэтому в Табл. 2.10 у последних её трёх строчек была установлена пометка «уйдут». В отрицательной области, которую мы не рассматривали, потери аналогичны. Если же рассмотреть формат binary32 (float), то формат чисел NaN представлен рисунком ниже:



а его общие потери составляют аж целых $2\times(2^{23}-1) = 16\ 777\ 214$ чисел. На рис. 2.3 и в табл. 2.8 б также хорошо видно, что это значительный диапазон от общего числа вариантов $2^{32} = 4\ 294\ 967\ 296$.

Хочется спросить: зачем столько много? К сожалению, ответ на данный вопрос выходит не только за рамки книги, но и за рамки стандарта, и дело вот в чём. О том, что числа бывают разные, было сказано в начале раздела о целых числах, даже если вы его не читали, школьный курс любой уважающей себя школы подразумевает поверхностное ознакомление с таким расширением действительных чисел, как комплексные числа. А операции с такими числами в ЭВМ стандартом IEEE754-2008 не определяются. Скажем лишь, что последний подразумевает поддержку двух видов «не чисел»: *signaling NaN*, *sNaN*, «*сигнальные не числа*» и *quiet NaN*, *qNAN*, «*тихие не числа*», которые должны поддерживаться во всех операциях с плавающей запятой. Мы не будем их рассматривать, но приведём, что по этому поводу говорится в разделе 6.2 стандарта ***Operations with NaNs*:**

Signaling NaNs afford representations for uninitialized variables and arithmetic-like enhancements (such as complex-affine infinities or extremely wide range) that are not in the scope of this standard. Quiet NaNs should, by means left to the implementer's discretion, afford retrospective diagnostic information inherited from invalid or unavailable data and results. To facilitate propagation of diagnostic information contained in NaNs, as much of that information as possible should be preserved in NaN results of operations.

2.1.4.5. Интересные наблюдения

Теперь, когда мы поверхностно ознакомились с форматом хранения действительных чисел приведём несколько интересных наблюдений и перейдём к экспериментам с целью проверки на практике полученных знаний.

Наблюдение 1

Как было замечено в [19], одна из удивительных особенностей представления чисел в формате IEEE754 состоит в том, что порядок и мантисса расположены друг за другом таким образом, что вместе образуют последовательность целых чисел {n}, для которых выполняется:

$$n < n+1 \;\Rightarrow\; F(n) < F(n+1),$$

где $F(n)$ – число с плавающей запятой, образованное от целого n разбиением его битов на «смещённый порядок» и «часть битов мантиссы».

Поэтому, если взять положительное число с плавающей запятой, преобразовать его к двоичному коду, код к целому числу, затем прибавить «1», и сделать преобразования обратно к числу, мы получим следующее число, которое представимо в этой арифметике.

На примере табл. 2.10 это было хорошо заметно. Увидеть «следующее» число формата float можно с помощью следующей программы:

```
$ cat next_float.c
#include <stdio.h>
#include <stdlib.h>
 // !!! Этот код будет работать только на архитектуре с 32-битным int.
int main (int argc, char *argv[])
{
   if (argc != 2)
   {
```

```
    printf("Использование:\n %s number.\n", argv[0]);
    return -1;
  }
  float a = atof(argv[1]);
  printf("Исходное число: %f = %e\n", a, a);  // %e - scientific format
  int n = *(int *)&a + 1;
  float b = *(float *)&n;
  double diff = (double)b - (double)a;
  printf("После %e следующее число: %e, разница (%e)\n", a, b, diff);
  return 0;
}
```

Скомпилируем её и запустим с параметром «0.5».

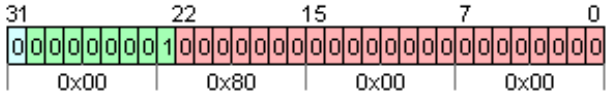
```
$ gcc next_float.c -o next_float
$ ./next_float 0.5
Исходное число: 0.500000 = 5.000000e-01
После 5.000000e-01 следующее число: 5.000001e-01, разница (5.960464e-08)
```

Заодно указанный код измеряет и расстояние до следующего числа. О том, что числа распределены неравномерно, было показано выше. Ниже по этому поводу приводится ещё одно наблюдение.

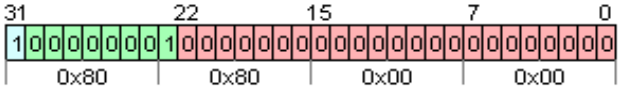
Наблюдение 2

Зная формат чисел с одинарной точностью стандарта IEEE 754 (binary32), Яшкардин Владимир в [17] предлагает посчитать границы диапазона представления действительных чисел в этом формате. Для этого подставим значения максимальных и минимальных абсолютных чисел IEEE 754 в формулы № 1 и № 2.

Минимальными по модулю нормализованными числами будут

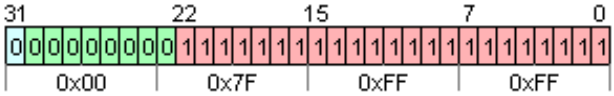


(для него $00\ 80\ 00\ 00_{16\_BE} \sim 2^{-126}\cdot(1+0/2^{23}) = 2^{-126} \approx 1{,}17549435\cdot10^{-38}$) и

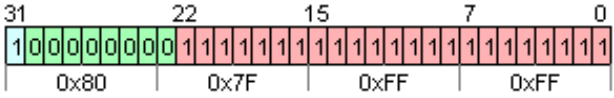


(для него $80\ 80\ 00\ 00_{16\_BE} \sim -2^{-126}\cdot(1+0/2^{23}) = -2^{-126} \approx -1{,}17549435\cdot10^{-38}$)

Максимальными по модулю денормализованными числами будут



(для него $00\ 7F\ FF\ FF_{16\_BE} \sim 2^{-126}\cdot(1-2^{-23}) \approx 1{,}17549421\cdot10^{-38}$) и



(для него $80\ 7F\ FF\ FF_{16\_BE} \sim -2^{-126}\cdot(1-2^{-23}) \approx -1{,}17549421\cdot10^{-38}$).

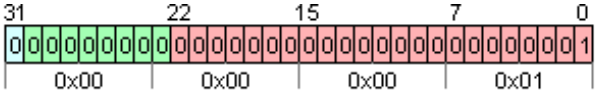
Отсюда видно, что минимальное нормализированное число граничит с максимальным денормализированным и «разрыва» между диапазонами нет, поскольку нет изменения плотности следования чисел (расстояний между двумя соседними числами). Собственно это также наглядно видно на примере формата binary5.

Как следствие, большинство начинающих программистов, использующих тип float (IEEE754-2008 формат binary32) «как чёрный ящик», могут и не знать, что несмотря на «условную непрерывность» следования представимых форматом чисел существуют две различные формулы № 1 и № 2 обеспечивающих этот результат.

Оценим границы нового «объёдинённого» диапазана, ответив на вопрос: какие минимальное и максимальные числа по модулю можно представить форматом binary32?

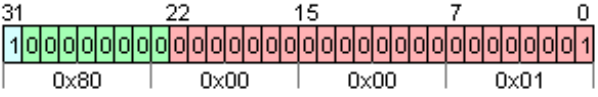
min

Если говорить о минимальных по модулю числах, то следует рассмотреть минимальные денормализованные числа, которые для указанного формата будут выглядеть как



«+min»

($01\ 00\ 00\ 00_{16\_LE} = 00\ 00\ 00\ 01_{16\_BE} \sim 2^{-126}\cdot 2^{-23} = 2^{-149} \approx 1{,}40129846\cdot 10^{-45}$) и



«−min»

($01\ 00\ 00\ 80_{16\_LE} = 80\ 00\ 00\ 01_{16\_BE} \sim -2^{-126}\cdot 2^{-23} = 2^{-149} \approx -1{,}40129846\cdot 10^{-45}$)

Оба числа граничат с нулём. Напишем небольшую программу для проверки.

```
$ cat bits_to_float.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void PrintBinNumber(unsigned char *bytes, size_t count);
void PrintBinBytes(unsigned char *bytes, size_t count);
void PrintHexNumber(unsigned char *number, size_t bytesCount);
void PrintFloat(float f);
void PrintHexMemory(void * pointer, size_t bytesCount);

void PrintUsage(char *programName)
{
   printf ("Передайте в командной строке бит знака,
            8 бит показателя степени, 23 бита мантиссы.\n");
   printf ("Например: %s 0 00011100 00000001111111100000000\n", programName);
}

int BuildFloat(int current, char *bits, int highBitNumber)
{
   int length = strlen(bits);
   for (int i = 0; i < length; ++i)
   {
      if (bits[i] == '1')  current += 1 << (highBitNumber - i);
   }
   return current;
}
int main(int argc, char *argv[])
```

```
{  // !!! Этот код будет работать только на архитектуре с 32-битным int.
   if (argc != 4)
   {
      PrintUsage(argv[0]);
      return -1;
   }

   const size_t SIGN_LEN = 1;
   const size_t BIASED_EXPONENT_LEN = 8;
   const size_t MANTISSA_WITHOUT_FIRST_BIT_LEN = 23;

   const size_t HIGH_FLOAT_BIT_NUM = SIGN_LEN + BIASED_EXPONENT_LEN +
                                     MANTISSA_WITHOUT_FIRST_BIT_LEN - 1;
   char *sign = argv[1];
   char *biasedExponent = argv[2];
   char *mantissaWithoutFirstBit = argv[3];

   if (strlen(sign) != SIGN_LEN ||
       strlen(biasedExponent) != BIASED_EXPONENT_LEN ||
       strlen(mantissaWithoutFirstBit) != MANTISSA_WITHOUT_FIRST_BIT_LEN)
   {
      PrintUsage(argv[0]);
      return -1;
   }

   printf("       Бит знака: %s.............................\n", sign);
   printf("    Биты степени: .%s.....................\n", biasedExponent);
   printf("  Хвост мантиссы: .........%s\n", mantissaWithoutFirstBit);
   printf("    Общая запись: %s%s%s\n", sign, biasedExponent,
                                        mantissaWithoutFirstBit);

   unsigned int a = 0; //переменная для хранения числа

   a = BuildFloat(a, sign, HIGH_FLOAT_BIT_NUM);
   a = BuildFloat(a, biasedExponent, HIGH_FLOAT_BIT_NUM - SIGN_LEN);
   a = BuildFloat(a, mantissaWithoutFirstBit,
                  HIGH_FLOAT_BIT_NUM - SIGN_LEN - BIASED_EXPONENT_LEN);

   float number = *(float*)&a;

   printf("        Исходное число: %e\n", number);

   printf("       IEEE-754 формат: ");
   PrintFloat(number);
   printf("\n");

   printf("Двоичное представление: ");
   PrintBinNumber((unsigned char *)&number, sizeof(number));
   printf("\n");

   printf(" 16-чное представление: ");
   PrintHexNumber((unsigned char *)&number, sizeof(number));
   printf("\n");

   printf("Представление в памяти: ");
   PrintBinBytes((unsigned char *)&number, sizeof(number));
   printf("\n");
   PrintHexMemory(&number,sizeof(number));
   return 0;
}
```

Скомпилируем и запустим.

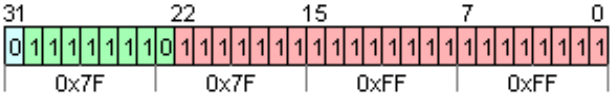
```
$ g++ bits_to_float.c PrintFloat.cpp PrintBinNumber.cpp
      PrintHexNumber.cpp PrintHexByte.cpp PrintBinBytes.cpp
```

```
      PrintBinByte.cpp PrintHexMemory.c -o bits_to_float
$ ./bits_to_float 1 00000000 00000000000000000000001
            Бит знака: 1.............................
          Биты степени: .00000000.....................
       Хвост мантиссы: .........00000000000000000000001
         Общая запись: 10000000000000000000000000000001
       Исходное число: -1.401298e-45
      IEEE-754 формат: 1 00000000 00000000000000000000001
Двоичное представление: 10000000 00000000 00000000 00000001
 16-чное представление:    80       00       00       01
Представление в памяти: 00000001 00000000 00000000 10000000
по адресу 0x7fff77ec6e70: 01
по адресу 0x7fff77ec6e71: 00
по адресу 0x7fff77ec6e72: 00
по адресу 0x7fff77ec6e73: 80
```

Как видим, результат получился ожидаемым.

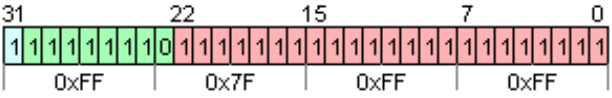
max

Максимальным числом для формата *float* (*binary32*) будет число, в записи которого будут все единицы, кроме младшего разряда степени (разряд знака также не рассматривается).



«+max»

(FF FF 7F $7F_{16\_LE}$ = 7F 7F FF FF_{16_BE} ~ $(+1)\cdot(2^{127}\cdot(2-2^{-23})$ ≈ 3,40282347·10^{+38})



«−max»

(FF FF 7F FF_{16_LE} = FF 7F FF FF_{16_BE} ~ $(-1)\cdot(2^{127})\cdot(2-2^{-23})$ ≈ −3,40282347·10^{+38})

Эти числа для binary32 граничат с +∞ и −∞, соответственно.

Проверим той же программой, что была написана выше.

```
$ ./bits_to_float 0 11111110 11111111111111111111111
            Бит знака: 0.............................
          Биты степени: .11111110.....................
       Хвост мантиссы: .........11111111111111111111111
         Общая запись: 01111111011111111111111111111111
       Исходное число: 3.402823e+38
      IEEE-754 формат: 0 11111110 11111111111111111111111
Двоичное представление: 01111111 01111111 11111111 11111111
 16-чное представление:    7F       7F       FF       FF
Представление в памяти: 11111111 11111111 01111111 01111111
по адресу 0x7fff0bcc6440: FF
по адресу 0x7fff0bcc6441: FF
по адресу 0x7fff0bcc6442: 7F
по адресу 0x7fff0bcc6443: 7F
```

Вопрос читателю ха засыпку: Какой у данной ЭВМ порядок байтов: Little Endian или Big Endian?

Наблюдение 3

Сергей Холодилов в [25] предлагает следующую математическую модель.

Если множество действительных чисел – это прямая, то множество чисел, представимых в виде <*подставьте название любого конечного формата*>, – это конечное

множество точек на прямой. Точки упорядочены, переход от точки к точке – прибавление/вычитание единицы к младшему разряду и (для форматов с плавающей запятой), если необходимо, нормализация.

Пример числовой прямой для формата с фиксированной запятой показан на рис. 2.9, представимые числа отмечены красными точками (формат 4:1 – четыре знака перед запятой, один после).

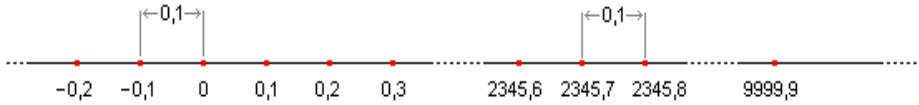


Рисунок 2.9. Пример множества чисел, представимых в формате с фиксированной запятой (расстояние между соседними числами одинаково)

Характерный признак – постоянный шаг. Два соседних числа с фиксированной запятой всегда отличаются на одну и ту же дельту – в данном случае на 0,1. Этим они похожи на целые числа.

У чисел с плавающей запятой картина совершенно иная. На рис. 2.10 кусочек числовой прямой изображён в «истинном» масштабе (формат немного игрушечный: мантисса – 1 десятичный знак, показатель степени – от −3 до +4).

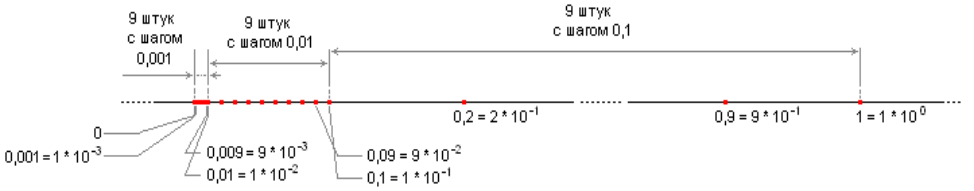


Рисунок 2.10. Пример множества чисел, представимых в формате с плавающей запятой, истинный масштаб

Видно, что у каждой следующей части шаг увеличивается в 10 раз, это естественное следствие увеличения показателя степени. То, что в данном случае шаг равен всем предыдущим частям, вместе взятым, – совпадение, если бы мантисса была длиннее, шаг был бы меньше.

На рис. 2.11 менее игрушечный формат (мантисса – 3 десятичных знака, показатель степени – от −3 до +4, для маленьких значений разрешена денормализация) изображён с переменным масштабом.

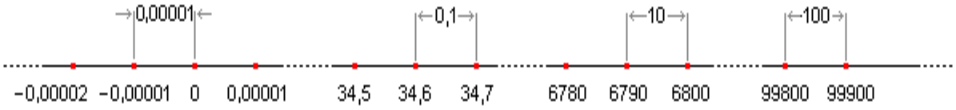


Рисунок 2.11. Пример множества чисел, представимых в формате с плавающей запятой, переменный масштаб

В данном случае шаг меняется от 0,00001 до 100. Если не считать денормализованных значений, то постоянно количество значащих цифр – в примере их всегда три, по размеру мантиссы.

В общем-то, в этих картинках – вся суть, остальное – просто следствия – примеры применения этой модели, демонстрирующие проявляющиеся на практике отличия от «математических» чисел.

2.1.5. Ошибки представления (точность)

Попытаемся ответить на вопрос почему две программы, приведённые в качестве примера перед разделом 2.1, работают неправильно и как может показаться «обманывают» нас, выдавая результатом первой аж 8 вместо 1.

Что интересно, аналогичная программа, написанная на PHP работает «лучше» и выдаёт интуитивно ожидаемую единицу.

```
$ cat 123456789_example2.php
<?php
    $a=floatval(123456789);
    $b=floatval(123456788);
    $f=floatval($a-$b);
    if (is_float($a) and is_float($b) and is_float($f))
       echo 'Переменные $a,$b,$f типа float.';
    echo "\na-b=$f\n";
?>
$ php 123456789_example2.php
Переменные $a,$b,$f типа float.
a-b=1
```

Если программу на C немного модифицировать, чтобы переменные были типа int, то результат будет ожидаемым.

```
$ cat 123456789_int_example.c
#include <stdio.h>

void main (void)
{
    int a,b,f;
    a=123456789;
    b=123456788;
    f=a-b;
    printf ("a-b=%d\n",f);
}
$ gcc 123456789_int_example.c &&./a.out
a-b=1
```

Хотя сравнивать в данном случае не совсем корректно (переменные хранятся как строки), аналогичная программа на bash и результат её работы будут выглядеть следующим образом:

```
$ cat 123456789_example.sh
#!/bin/bash
a=123456789
b=123456788
let f=$a-$b
echo "a-b=$f"
$ sh 123456789_example.sh
a-b=1
```

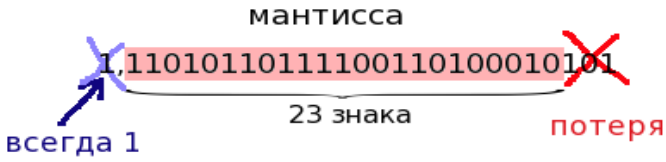
Странно получается: один и тот же тип float в PHP и в C «работает» по-разному. Если быть честными, то сравнивать PHP и C в данном случае тоже не совсем корректно, но если вы задались вопросами «*почему?*» или «*а что же происходит в программе на самом деле?*», или «*откуда берётся ошибка в 700% при смене типа?*», то продолжайте читать дальше.

Вспомним только что прочитанную вами теорию и попытаемся разместить значение 123456789 в отведённые для хранения в памяти переменной *a* (в первой программе) четыре байта.

Для этого нам придётся вычислить биты отвечающие за знак числа, показатель степени (смещённый) и часть битов мантиссы, а затем заполнить полученными значениями соответствующие поля согласно стандарту binary32 IEEE 754-2008.

Что же, приступим, преобразование будет происходить по следующему алгоритму [23, 17]:

1. Определяемся со знаком числа. Знак «+», значит первый (он же самый старший) бит 32 битной записи числа в памяти ЭВМ будет равен 0. Получим [0xxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx]
2. Определяем значение показателя степени. Для этого переведём число 123456789 в двоичную систему счисления
   $123456789_{10}=11101011011110011010001010 1_2=$
   ---------- 27 знаков ----------
   $=11101011011110011010001010 1\cdot 2^0=11101011011110011010001010 1,0\cdot 2^0$
   Сдвигаем запятую на 26 позиций ($26_{10}=11010_2$) – это и есть степень двойки.
   $1,11101011011110011010001010 1\cdot 2^{11010}$
   Согласно стандарту хранение показателя степени происходит «со смещением», поэтому, смотрим в Таблицу 2.7, и определяем сколько бит отведено под хранение степени (колонка binary32, строка с параметром *w*) и каково смещение показателя степени (колонку binary32, строка *bias*). Получим 8 и $127=+127_{10}=0111\ 1111_2$ соответственно. Следовательно, после смещения получим $26+127=132_{10}=10011001_2$. Дополнять нулями полученное значение не нужно. Смещённое значение записывают в отведённые для него 8 бит и получают [x1001100 1xxx xxxx xxxx xxxx xxxx xxxx]
3. Определяем мантиссу в нашем примере она равна 1,11101011011110011010001010 1. Мантисса хранится в нормализованном виде, поэтому её первый бит всегда получается равен 1 (и, как следствие не записывается), а далее возникает казус. Для записи оставшихся 26 разрядов после запятой в памяти по стандарту отведено лишь 23



Получим для битов мантиссы [xxxx xxxx x110 1011 0111 1001 1010 0010]
Сверим конечно полученный итоговый теоретический результат
[0 10011001 11101011011110011010001 0]

с результатом работы программы float_representation.c на практике:

```
$ ./float_representation 123456789
 Исходное число: 123456792.000000
IEEE-754 формат: 0 10011001 11010110111100110100011
   Двоичный вид: 10100011 01111001 11101011 01001100
по адресу 0x7fff692e479c: A3
по адресу 0x7fff692e479d: 79
по адресу 0x7fff692e479e: EB
по адресу 0x7fff692e479f: 4C
```

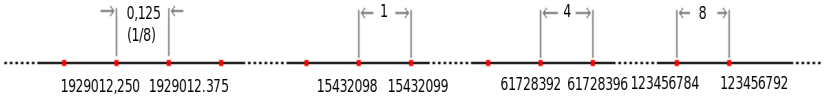
Как видим, различие лишь в последнем разряде.

Заметим, что программа переводила число 123456792, а не 123456789, потому как последнее в формате float записать невозможно.

Давайте попробуем произвести обратное действие для обоих чисел – запишем теоретически полученное значение и практическое (с помощью программы) в память и посмотрим на результат. Для этого воспользуемся программой bits_to_float.c.

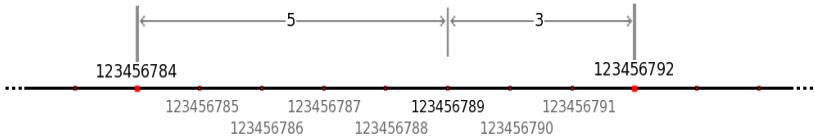
```
$ ./bits_to_float 0 10011001 11010110111100110100011
     Бит знака: 0...............................
  Биты степени: .10011001.......................
Хвост мантиссы: .........11010110111100110100011
  Общая запись: 01001100111010110111100110100011
Шестнадцатеричная запись: 4c eb 79 a3
В памяти ЭВМ
по адресу 0x7ffe46ded7f4: A3
по адресу 0x7ffe46ded7f5: 79
по адресу 0x7ffe46ded7f6: EB
по адресу 0x7ffe46ded7f7: 4C

Число float= 1.234568e+08
Число float= 1.234568e+08
Число float= 123456792.000000
$ ./bits_to_float 0 10011001 11010110111100110100010
     Бит знака: 0...............................
  Биты степени: .10011001.......................
Хвост мантиссы: .........11010110111100110100010
  Общая запись: 01001100111010110111100110100010
Шестнадцатеричная запись: 4c eb 79 a2
В памяти ЭВМ
по адресу 0x7ffe6a5adb74: A2
по адресу 0x7ffe6a5adb75: 79
по адресу 0x7ffe6a5adb76: EB
по адресу 0x7ffe6a5adb77: 4C

Число float= 1.234568e+08
Число float= 1.234568e+08
Число float= 123456784.000000
```

Как видно, были получены два числа 123456792 и 123456784. Рассмотрим как указанные числа расположены на «числовой оси формата binary32 IEEE754 (float)».



А также посчитаем насколько они «отстают» от искомого числа 123456789.

$|123456784-123456789| = 5$ $|23456792-123456789| = 3$



То есть, первое (полученное на практике число 123456792) явно ближе к искомому результату (даёт меньшую ошибку). А второе, полученное нами в теории – 123456784, отличается больше, то есть, заведомо даст большую ошибку вычислений. Получается, что операции «отбрасывания» лишних разрядов и «округления» хоть и похожи, но могут приводить к разным результатам.

Если вы не специалист по преобразованию чисел, то чем руководствоваться? Что говорит по этому поводу стандарт IEEE754-2008? Однозначного ответа на первый вопрос нет, так как ответ на второй вопрос предусматривает четыре способа округления чисел [22]:

1. Округление стремящееся к ближайшему целому.
2. Округление стремящееся к нулю.
3. Округление стремящееся к $+\infty$.
4. Округление стремящееся к $-\infty$.

Отсюда можем заключить, что теория и практика часто могут расходиться из-за мелочей. Лишь «хакеры» (в смысле профессионалы своего дела) могут знать подобные нюансы и способны точно предугадать поведение ЭВМ. Надеемся, вы теперь догадались почему первоначальная программа, вычитая числа 123456789 и 123456788, представленные в формате float, получила результат равный 8?

Ситуация со второй программой аналогичная. Запустите её под отладчиком и вы увидите как в памяти представлены числа 0,1 и 0,3. Думает, что отладчик не понадобится, поскольку сейчас, после прочтения информации данного раздела, вы можете сделать все преобразования на листе бумаги, то есть, без компьютера вообще.

Напоследок добавим, что проблема точности математических вычислений в ЭВМ не нова и давно уже существует понятие машинного ε (эпсилон), в отношении которого хочется привести красивый график функции ошибки точности представления чисел в формате IEEE754, который читатели наверняка запомнят лучше чем формулы (см. рис. 2.12).

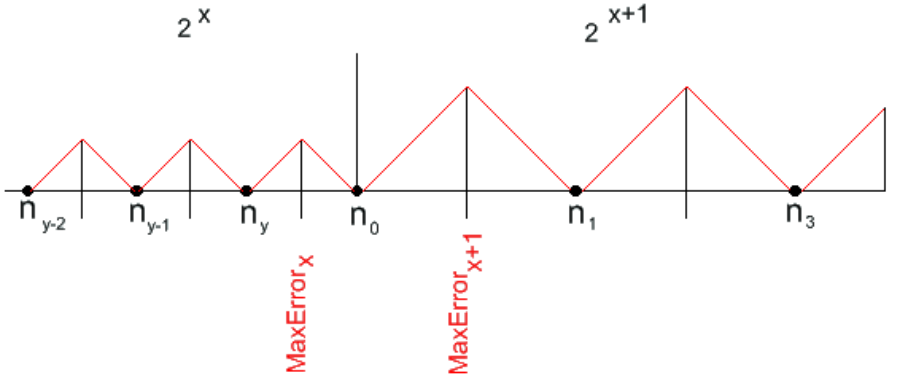


Рисунок 2.12. Функция ошибки точности представления числа в формате IEEE754

Выводы о точности и ошибках вычислений

Вы, наверно, и сами догадались глядя на рисунок 2.12 (впрочем как и на рисунки 2.9-2.11, ранее), что числа представленные в формате IEEE754 представляют конечное множество, на которое отображается бесконечное множество вещественных чисел. Мощности множеств явно не равны, поэтому исходное число может быть представлено в формате IEEE754 с ошибкой. (В некоторых случаях может повезти и ошибка будет равно нулю.) Абсолютная максимальная ошибка для числа в формате IEEE754 равна в пределе половине шага чисел, это тот самый случай, что получился в первой программе. Шаг чисел удваивается с увеличением показателя степени двоичного числа на единицу, то есть при удвоении числа. Фактически видно, что, чем дальше от нуля, тем шире шаг чисел в формате IEEE754 по числовой оси.

Рассмотрим два интересных случая из практики: про циклическую дыру и приведём пример С.М.Лавренова из [41] по подсчёту результата вычисления следующей формулы

$$\frac{(a+b)^2-(a^2+2ab)}{b^2},$$

где для хранения переменных *a* и *b* предлагается использовать тип *float*.

Результат вычислений значения формулы должен быть всегда равен 1. Это легко проверить, если раскрыть скобки и упростить формулу. При значениях $a = 95$ и $b = 1$ оно так и есть, результат вычислений на практике – единица, однако, при $a = 95$ и $b = 0{,}1$ нижеследующая программа выдаёт интересный результат:

```
$ cat Lavrenov\'s_example.c
#include <stdio.h>

void main (void)
{
  float a,b,c,d,e,f;
  a=95;  b=0.1;
  c=((a+b)*(a+b)-(a*a+2*a*b))/(b*b);
  printf ("Результат вычислений: %f\n",c);
}

$ gcc Lavrenov\'s_example.c && ./a.out
Результат вычислений: 0.976562
```

Если поэкспериментировать с параметрами, то при разных значениях *a* и *b* будут получаться разные результаты, хотя мы знаем, что результат всегда должен быть один – равный единице (см. Табл. 2.13).

Таблица 2.13. Влияние выбора параметров a и b на результат вычисления

a	*b*	результат
95	1	1,000000
95	0,1	0,976562
95	0,05	1,171875
95	0,04	1,220703
95	0,01	0,000000
5	0,01	1,010895
5	0,1	0,999832

Ещё один довольно интересный пример ошибки, вызванной сдвигом мантисс, рассматривается в [19]:

Циклические дыры

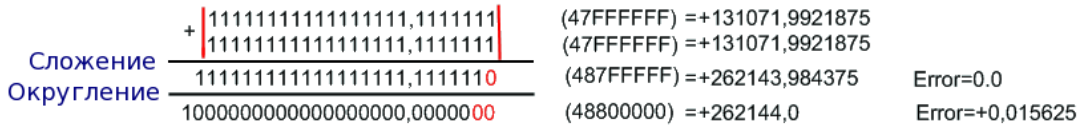
Эти ошибки связаны с потерей точности результата при неполном пересечении мантисс чисел на числовой оси. Если мантиссы чисел не пересекаются на числовой оси, то операции сложения и вычитания между этими числами невозможны (происходят с ошибкой). К примеру, возьмём число $+131071{,}9921875_{10}$

```
$ ./float_representation 131071.9921875
 Исходное число: 131071.992188
IEEE-754 формат: 0 10001111 11111111111111111111111
   Двоичный вид: 11111111 11111111 11111111 01000111
по адресу 0x7fff4104c48c: FF
по адресу 0x7fff4104c48d: FF
по адресу 0x7fff4104c48e: FF
по адресу 0x7fff4104c48f: 47
```

Несмотря на то, что число 131071,9921875 в двоичной системе точно представляется как: +11111111111111111,1111111 в программе при выводе оно округляется до 6 знаков после запятой и выводится как 131071,992188.

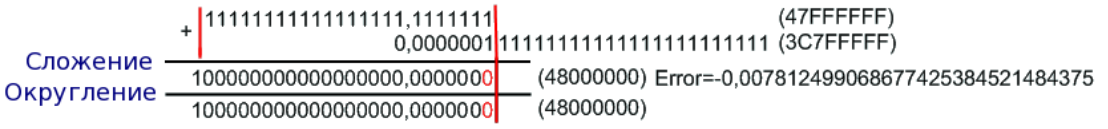
Покажем несколько взятых из [17] компьютерных операций сложений с этим числом в формате IEEE 754-2008 binary32. Значащих цифр в мантиссе двоичного числа в формате binary32 не более 24 (23+1). Красным цветом показаны цифры выходящие за этот предел.

1. Сложение с одинаковым числом (ошибка сдвига = 0,0).

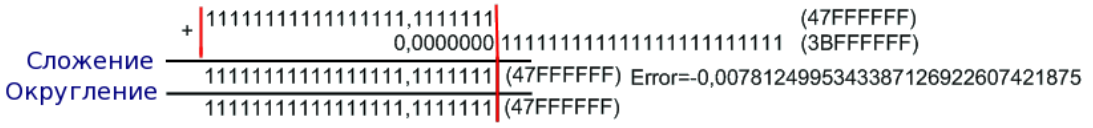


2. Сложение с числом меньшим в 2 раза (ошибка сдвига = −0,00390625).

3. Сложение с числом меньшим в 2^{23} раза (ошибка сдвига = −0,007812).



4. Сложение с числом меньшим в 2^{24} раза (ошибка сдвига= −0.007812).



В последнем примере мантиссы чисел разошлись, и арифметические операции с этими числами становятся бессмысленными.

Как видно из приведённых примеров ошибка сдвига возникает если у исходных нормализованных чисел различаются показатели степени. Если числа отличаются более чем в 2^{23} (для binary32) и 2^{52} (для binary64) раз, то операции сложения и вычитания между этими числами невозможны.

Максимальная относительная погрешность результата операции равна примерно $5{,}96 \cdot 10^{-6}$ %, что не превышает относительную погрешность представления числа (Подробнее см.п.9.1 в [19]).

Хотя, с относительной погрешностью здесь всё в порядке, есть и другие проблемы.

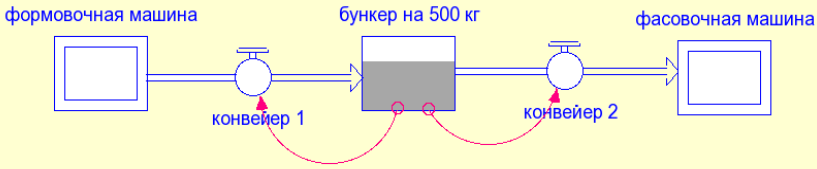
Во-первых, работать с числами можно только в узком диапазоне числовой оси, где мантиссы пересекаются.

Во-вторых, для каждого исходного числа существует предел выполнения цикла называемый «***Циклической дырой***». (Термин предположительно введён Владимиром Яшкардиным в [17]) Поясним, если существует цикл в котором исходное число суммируется к сумме, то существует численный предел суммы для этого числа. То есть, сумма, достигнув определённой величины, перестает увеличиваться от сложения её с исходным числом.

Пример циклической дыры

Рассмотрим пример циклической дыры, взятый из [17], в автоматической системе управления:

Имеется фармацевтический цех, производящий таблетки весом 10 мг, состоящий из: формовочной машины, накопительного бункера на 500 кг, фасовочной машины, автоматической системы управления.



Формовочная машина подаёт в бункер по 10 таблеток одновременно. Фасовочная машина забирает по одной таблетке.

Автоматическая система управления учитывает таблетки поступившие в бункер от формовочной машины и взятые из бункера фасовочной машиной. То есть, существует программа, которая показывает заполнения бункера продукцией в кг. Когда в бункере будет более 500 кг продукции, формовочная машина встаёт на паузу, она включится когда в бункере станет 200 кг продукции. Фасовочная машина остановиться если в бункере будет менее 10 кг и запустится когда в бункере будет более 100 кг продукции.

Обе машины могут периодически останавливаться для обслуживания, не зависимо друг от друга (благодаря бункеру).

Как вы понимаете, такая система работает в бесконечном цикле.

Допустим в один из дней фасовочная машина простояла слишком долго и бункер заполнился до 300 кг.

Что произойдёт после её включения?

Смоделируем ситуацию на выходном конвейере.

```
$ cat model.c
#include <stdio.h>
int main (void)
{
  float a=0.00001; // вес таблетки в кг
  float c;  // продукции в бункере в кг
  c = 300;  // исходный вес продукции в бункере
  for (; c>0 ;)
  {
    c = c - a ;   // забрали одну таблетку из бункера
    printf ("изменённый вес бункера %f\n",c);
  }
  return 0;
}
```

```
$ gcc model.c && ./a.out
изменённый вес бункера 300.000000
изменённый вес бункера 300.000000
изменённый вес бункера 300.000000
...
```

выполнение прервано по нажатию «CTRL+C»

Как видим, конвейер «повис» в бесконечном цикле. Измените вес таблетки на 0,1 кг и вы увидите как программа model.c довольно скоро завершит своё выполнение, считай наша условная фасовочная машина остановится. Как понимаете, таблеток по 100 г не бывает, поэтому изменим программу, добавив счётчик.

```
$ cat model2.c
#include <stdio.h>
int main (void)
{

  float a=0.00001; // вес таблетки в кг
  float c;         // продукции в бункере в кг
  long int i;      // счётчик циклов
  c = 300;         // исходный вес продукции в бункере

  for (i=1; c>0 && i<=100000; i++)
  {
    c = c - a ;    // забрали одну таблетку из бункера
    printf ("шаг %d, изменённый вес бункера %f\n",i,c);
  }
  return 0;
}
```

Всё равно происходит «ошибка», в данном примере фасовочная машина забрала из бункера 1 кг продукции, а вес продукции в бункере не изменился.

Почему не изменился? Потому, что мантиссы чисел 300 и 0,00001 не пересекаются для формата binary32 (напомним, что для языка C это float).

А что дальше?

Далее формовочная машина доведёт вес бункера до 500 кг и остановится. Фасовочная машина заберёт все таблетки из бункера и тоже остановится. (*Если добавлять к массе контейнера по 10 таблеток, то сумма увеличивается, а если отнимать по од-*

ной – не уменьшается.) Программа будет показывать вес в бункере 500 кг. Прибегут специалисты, проверят датчики, провода, компьютер и скажут, что программа зависла. Но, программа не зависала, она продолжает работать без сбоев и любой контроль это подтвердит. Просто число 0,0001 попало в циклическую дыру и выйти из него не может.

На самом деле опытный программист никогда не будет делать циклическое вычитание (или суммирование) таким образом. Этот пример вымышлен специально. По мнению Владимира Яшкардина [17] так считать нельзя, хотя с точки зрения математика здесь всё безупречно. Эта ошибка свойственна математикам и начинающим программистам. Основная работа программиста заключается в борьбе с погрешностями, а не в математических решениях поставленной задачи.

2.2. Представление текстовой информации в ЭВМ

Так как большинство современных ЭВМ (если не сказать, что почти все) удовлетворяет принципам вычислительной машины фон Неймана, логично предположить, что информация, которую они хранят, обрабатывают и передают по сетям (в том числе и тексты), удовлетворяет принципу двоичного кодирования, а именно – представима в виде двоичных чисел.

Существуют международные стандарты и методы кодирования текстовой, числовой, изобразительной, звуковой и видеоинформации.

Если вопросы представления чисел в памяти ЭВМ мы рассмотрели в предыдущем разделе, то данный раздел посвящён проблемам хранения текстовой информации.

Понимание принципов кодирования и знание основных кодовых таблиц очень важны для правильного чтения информации, добываемой из интернета, а в ряде случаев и из получаемой вами электронной почты или файлов.

Если мы перенесёмся во времена, когда ЭВМ только появлялись, а их базовые принципы ещё не были чётко сформулированы, то первое, что приходит на ум, – закрепить за каждым символом алфавита числовой номер – код. Например, 1 для буквы «а», 2 пусть обозначает букву «б», 3 – букву «в» и т. д. Так как мы не были первыми, кто использовал ЭВМ и так думал, первично кодированию подверглись латинские символы, цифры и различные знаки препинания, а не кириллица. Так, в англоязычных странах используются 26 прописных и 26 строчных букв (A… Z, a… z), 9 знаков препинания (. , : ! " ; ? ( )), пробел, 10 цифр, 5 знаков арифметических действий (+, −, *, /, ^) и ряд специальных символов (№, %, _, #, $, &, >, <, |, \); все они и были закодированы – всего набралось чуть больше 100 символов. Для кодирования такого числа вариантов можно ограничиться 7-разрядным двоичным числом (от 0 до 1111111, в десятичной системе счисления – от 0 до 127), что и было сделано.

2.2.1. Кодовые таблицы

Для кодирования букв и других символов, используемых в печатных документах, необходимо закрепить за каждым символом числовой номер – код.

Первой 7-разрядной кодовой таблицей была ***ASCII*** (American Standard Code for Information Interchange), опубликованная как стандарт ASA X3.4-1963 в 1963 году американской организацией по стандартизации American Standards Association (ASA), которая позднее некоторое время именовалась ***ANSI*** [34]. Таблица содержала 32 кода команд или управляющих символов (от 0 до 31), большая часть которых сегодня не используется, и 95 кодов (от 33 до 127) для различных знаков, достаточных для работы с английскими текстами, как показано в табл. 2.14. Там же символы построчно имеют следующие коды в шестнадцатеричной системе счисления (в скобках – в десятичной):

- 1-я строка с 00 по 0F и далее с 10 по 1F (0–15, 16–31),
- 2-я строка с 20 по 2F и 30 – 3F (32–47, 48–63),
- 3-я строка с 40 по 4F и 50 – 5F (64–79, 80–95),
- 4-я строка с 60 по 6F и 70 – 7F (96–111, 112–127).

В данной таблице для преобразования прописных букв в строчные достаточно к коду буквы прибавить $32_{10} = 100000_2 = 00100000_2$, и наоборот, для преобразования строчных в прописные – из кода буквы вычесть 32_{10}.

$$a + 32_{10} = A,\ b + 32_{10} = B,\ \dots,\ z + 32_{10} = Z$$

$$A - 32_{10} = A,\ b - 32_{10} = B,\ \dots,\ z - 32_{10} = Z$$

На практике чаще используют не сложение и вычитание, а «побитовое ИЛИ» с числом 00100000_2 или «побитовое И» с числом 11011111_2 , соответственно, поскольку повторное применение побитовых логических операций не оказывает никакого влияния на результат, чего не скажешь про операции сложения и вычитания.

Направление преобразования регистра букв	Варианты действий над кодом буквы (символа)	
	арифметика, $\pm 32_{10}$	побитовые логические операции
a → A	прибавить $32_{10} = 00100000_2$	«побитовое ИЛИ» с $00100000_2 = 32_{10} = 20_{16}$
A → a	вычесть $32_{10} = 00100000_2$	«побитовое И» с $11011111_2 = 223_{10} = DF_{16}$

Единый шаг между буквами различного регистра наглядно можно увидеть, если кодовую таблицу символов разместить в таблице с 32 колонками. Символы в таблицу записываются слева направо, сверху вниз, начиная от символа с кодом 0 и заканчивая символом с кодом 127_{10}. Тогда в строках вышестоящие символы от нижестоящих будут отличаться ровно на 32 позиции (см. табл. 2.14 а).

Таблица 2.14 а. 7-битная кодовая таблица ASCII (ANSI)

	☺	☻	♥	♦	♣	♠	•	◘	○	◙	♂	♀	♪	♫	☼	►	◄	↕	‼	¶	§	▬	↨	↑	↓	→	←	∟	↔	▲	▼
	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

[34] American National Standards Institute, http://www.ansi.org/, поэтому данную 7-битную кодовую таблицу по появлении называли также ANSI, однако на сегодня термин «кодировка ANSI» не устоялся, в ряде случаев под ним могут пониматься совсем другие кодировки, вплоть до ISO/IEC 8859-1.

Однако, на практике использовать такую таблицу для преобразований кодов символов в их вид и обратно не очень удобно, поэтому чаще можно увидеть таблицу с 16 колонками, обрамлённую номерами строк и столбцов записанных в шестнадцатеричном виде (см. табл. 2.14 б).

Таблица 2.14 б. 7-битная кодовая таблица ASCII (ANSI)

№№ столбцов →	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1 строка	0		☺	☻	♥	♦	♣	♠	●	◘	○	◙	♂	♀	♪	♬	☼
2 строка	1	►	◄	↕	!!	¶	§	▬	↨	↑	↓	→	←	∟	↔	▲	▼
3 строка	2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
4 строка	3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
5 строка	4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
6 строка	5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
7 строка	6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
8 строка	7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

В такой таблице находить символы по их шестнадцатеричном коду, как и определять шестнадцатеричнй код, исходя из начертания символов, довольно просто. Первая позиция в записи XX_{16} – отвечает за выбор строки, а вторая – за выбор столбца в этой строке. Например, если код символа $4C_{16}$, то искать его следует в строке с подписью «4» (т.е. пятой по счёту сверху, поскольку нумерация начинается с нуля), где он будет располагаться на 13-й позиции (в пересечении со столбцом имеющем подпись «C», так как $C_{16} + 1 = 12_{10} + 1_{10} = 13_{10} = 13$).

Как вы успели заметить пересчитывать и использовать десятичные номера строк и столбцов в подобных таблицах не очень удобно и легко ошибиться, поэтому они даже не подписываются. Намного проще ориентироваться по шестнацатеричным подписям, а в случае редко возникающей необходимости шестнадцатеричное значение можно легко преобразовать в десятичное.

В последующем данная таблица ***ASCII*** была принята как стандарт ведущими международными организациями по стандартизации:

- ***ISO/IEC 646:1991*** (***ISO*** – *http://www.iso.org/* – International Organization for Standardization; ***IEC*** – *http://www.iec.ch/* – International Electrotechnical Commission – ведущие международные организации по стандартизации, в области электротехники – совместные стандарты);
- ***ITU-T Recommendation T.50 (09/92)*** (The International Telecommunication Union – *http://www.itu.int/*), ***ECMA*-6** (European Computer Manufacturers Association).

2.2.1.1. Русификация

В таблице ASCII (см. табл. 2.14), отсутствуют русские буквы (кириллица). Поэтому для нашей страны, а также и для многих других стран необходимо было добавить в кодовую таблицу символы национальных алфавитов. Для этого было предложено использовать 8-битную кодовую таблицу, которая могла содержать дополнительно ещё 128 символов (с 128 по 255).

В дальнейшем был принят стандарт на 8-битную таблицу ***ASCII – ISO/IEC 8859***, в которой первые 128 символов оставались теми же, что и в 7-битной таблице, а символы с 128 по 255 отводились для неанглийских символов.

Существует несколько частей этого стандарта:

- ISO/IEC 8859-1:1998 – Part 1: Latin alphabet № 1,
- ISO/IEC 8859-5:1999 – Part 5: Latin/Cyrillic alphabet,
- ISO/IEC 8859-6:1999 – Part 6: Latin/Arabic alphabet,
- ISO/IEC 8859-7:2003 – Part 7: Latin/Greek alphabet,
- ISO/IEC 8859-8:1999 – Part 8: Latin/Hebrew alphabet и т. д.

В табл. 2.15 представлена вторая половина кодовой таблицы (коды 128–255) для стандарта ***ISO 8859-5***, где подписи строк записаны со смещением (+8).

Таблица 2.15. Кодовая таблица ISO 8859-5 (коды с 128_{10} по 255_{10})

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8																
A		Ё	Ђ	Ѓ	Є	S	I	Ї	J	Љ	Њ	Ћ	Ќ	-	Ў	Џ
C	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
E	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
9																
B	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
D	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
F	№	ё	ђ	ѓ	є	s	i	ї	j	љ	њ	ћ	ќ	§	ў	џ

Первые русские ЭВМ использовали 7-битную кодировку символов ***КОИ-7*** (Код Обмена Информацией семибитный – табл. 2.16), в которой присутствовали прописные латинские буквы, а на месте строчных латинских были русские прописные буквы (кириллица).

*Таблица 2.16. 7-битная таблица символов **КОИ-7***

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
6	Ю	А	Б	Ц	Д	Е	Ф	Г	Х	И	Й	К	Л	М	Н	О

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1																
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
7	П	Я	Р	С	Т	У	Ж	В	Ь	Ы	З	Ш	Э	Щ	Ч	Ъ

На тот период времени это было первое самое простое и очевидное решение по русификации, хотя и не самое удачное. Поскольку в русском алфавите 33 буквы против 26 в латинском, пришлось пожертвовать латинским нижним регистром и несколькими дополнительными символами.

Позднее на первых отечественных персональных компьютерах использовалась так называемая «Основная кодировка ВЦ Академии наук СССР», в руководствах к старым матричным принтерам обозначаемая просто как **«ГОСТ»** – 8-битная кодовая таблица. Первая её половина полностью повторила ASCII, а вторая – уже содержала символы псевдографики, русские прописные и строчные буквы (коды с 128 по 255 – табл. 2.17).

*Таблица 2.17. Кодировка символов «**ГОСТ**» (коды с 128_{10} по 255_{10})*

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	╨	╤	╥	╙	╘	╒	╓	╫	╪	┘	┌	█	▄	▌	▐	▀
A	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
C	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
E	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
9	╔	╕	╖	╗	╘	╙	╚	╛	╜	╝	╞	╟	▓	╪	╫	╬
B	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
D	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
F	Ё	ё	/	\	/	\	→	←	↑	↓	÷	±	№	¤	■	

В дальнейшем основной кодировкой русских букв для первых дисковых операционных систем для ПК (MS-DOS в частности) стала «Альтернативная кодировка ВЦ Академии наук СССР» (вторая половина таблицы для кодов 128–255 приведена в табл. 2.18). Она также содержит псевдографику (позволяющую в текстовом режиме рисовать рамки из одинарных и двойных линий). Существует несколько модификаций, отличающихся символами в последних 14 позициях. Кроме этого она зарегистрирована в IANA (*Internet Assigned Numbers Authority* [35] – организации, отвечающей за административное управление в интернете) как IBM 866 или ***CP866***.

*Таблица 2.18. Таблица символов DOS Cyrillic (**CP866**, коды с 128_{10} по 255_{10})*

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
A	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
C	└	┴	┬	├	─	┼	╞	╟	╚	╔	╩	╦	╠	═	╬	╧
E	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
9	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
B	░	▒	▓	│	┤	╡	╢	╖	╕	╣	║	╗	╝	╜	┘	┐
D	╨	╤	╥	╙	╘	╒	╓	╫	╪	┘	┌	▉	▄	▌	▐	▀
F	Ё	ё	Є	є	Ї	ї	Ў	ў	°	•	·	√	№	¤	■	□

В связи с широким распространением в операционных системах того времени национальных локализаций для второй половины таблицы ASCII появилось понятие «кодовая страница» (code page, **CP**). В табл. 2.19 приведены некоторые из них.

Таблица 2.19. Некоторые национальные кодовые страницы (CP – code page) и их номера

Локализация	ANSI CP	Mac CP	DOS (OEM) Primary CP	DOS (OEM) Secondary CP
English (United States)	1252	10000	437	850
English (Britain, Canada и др.)	1252	10000	850	437
French (Standard)	1252	10000	850	437
German (Standard)	1252	10000	850	437
Russian	**1251**	**10007**	**866**	**855**

По мере развития операционных систем (ОС), в появившейся ОС Windows некоторое время использовалась **CP1251**, которая представлена в табл. 2.20.

Таблица 2.20. Таблица символов CP1251 (коды с 128_{10} по 255_{10})

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	Ђ	Ѓ	‚	ѓ	„	…	†	‡	€	‰	Љ	‹	Њ	Ќ	Ћ	Џ
A		Ў	ў	Ј	¤	Ґ	¦	§	Ё	©	Є	«	¬		®	Ї
C	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
E	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
9	ђ	‘	’	“	”	•	–	—		™	љ	›	њ	ќ	ћ	џ
B	°	±	І	і	ґ	µ	¶	·	ё	№	є	»	ј	S	s	ї
D	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
F	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я

В операционной системе Linux (как и в большинстве ОС UNIX) для представления русских букв исторически использовалась кодировка ***КОИ-8R*** (табл. 2.21), зарегистрированная в IANA как ***KOI8-R*** и описанная в RFC 1489 *(см. дополнительную информацию на http://koi8.pp.ru/)*, однако на сегодня большинство дистрибутивов ОС Linux используют кодировку **Unicode** представленную в формате **UTF-8**, о которой будет рассказано ниже.

[35] http://www.iana.org.

*Таблица 2.21. Таблица символов **KOI8-R** (коды с 128_{10} по 255_{10})*

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	─	│	┌	┐	└	┘	├	┤	┬	┴	┼	▀	▄	█	▌	▐
A	═	║	╒	ё	╓	╔	╕	╖	╗	╘	╙	╚	╛	╜	╝	╞
C	ю	а	б	ц	д	е	ф	г	х	и	й	к	л	м	н	о
E	Ю	А	Б	Ц	Д	Е	Ф	Г	Х	И	Й	К	Л	М	Н	О

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
9	░	▒	▓	⌠	■	·	√	≈	≤	≥	□	⌡	°	²	·	÷
B	╟	╠	╡	Ё	╢	╣	╤	╥	╦	╧	╨	╩	╪	╫	╬	©
D	п	я	р	с	т	у	ж	в	ь	ы	з	ш	э	щ	ч	ъ
F	П	Я	Р	С	Т	У	Ж	В	Ь	Ы	З	Ш	Э	Щ	Ч	Ъ

Стандартизован и зарегистрирован также украинский клон ***КОИ-8*** – ***KOI8-U*** (табл. 2.22), имеющий отличия от ***KOI8-R*** во второй строке символов псевдографики.

*Таблица 2.22. Таблица символов **KOI8-U** (коды с 128_{10} по 255_{10})*

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	─	│	┌	┐	└	┘	├	┤	┬	┴	┼	▀	▄	█	▌	▐
A	═	║	╒	ё	є	╔	і	ї	╗	╘	╙	╚	╛	ґ	╝	╞
C	ю	а	б	ц	д	е	ф	г	х	и	й	к	л	м	н	о
E	Ю	А	Б	Ц	Д	Е	Ф	Г	Х	И	Й	К	Л	М	Н	О

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
9	░	▒	▓	⌠	■	·	√	≈	≤	≥	□	⌡	°	²	·	÷
B	╟	╠	╡	Ё	Є	╢	І	Ї	╤	╧	╨	╩	╪	Ґ	╬	©
D	п	я	р	с	т	у	ж	в	ь	ы	з	ш	э	щ	ч	ъ
F	П	Я	Р	С	Т	У	Ж	В	Ь	Ы	З	Ш	Э	Щ	Ч	Ъ

Кириллица Macintosh (компьютеров фирмы Apple), она же **CP10007**, довольно близка к **CP1251**. Не зарегистрирована в IANA, но часто обозначается как **x-mac-cyrillic** (табл. 2.23).

*Таблица 2.23. Таблица символов Macintosh Cyrillic, **CP10007** (коды с 128_{10} по 255_{10})*

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
A	†	°	Ґ	£	§	•	¶	І	®	©	™	Ђ	ђ	≠	Ѓ	ѓ
C	j	S	¬	√	ƒ	≈	Δ	«	»	…		Ћ	ћ	Ќ	ќ	s
E	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
A	∞	±	≤	≥	і	µ	ґ	J	Є	є	Ї	ї	Љ	љ	Њ	њ
C	–	—	“	”	‘	’	÷	„	Ў	ў	Џ	џ	№	Ё	ё	я
E	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	¤

Замечание. Отметим, что все представленные выше кодовые таблицы (и кодировки на их основе, за исключением 7-битной кодовой таблицы ASCII) являются однобайтовыми.

2.2.1.2. Кодовая таблица Unicode

Развитие обмена текстами через: дискеты и модемы (BBS'ки, эхоконференции FidoNet, электронная почта через UUCP) показало недальновидность придуманной ранее схемы (с кодовыми страницами). Например, использование одновременно кириллицы и греческих символов в текстах было невозможным, так как они кодировались одними и теми же кодами.

В основе этой проблемы лежит принцип однобайтовости: для кодирования 1 символа используется 1 байт. 1 байт = 8 бит или 8 комбинаций, то есть можно адресовать лишь $2^8 = 256$ комбинаций, т.е. столько же различных символов.

Для решения проблемы в 1991 году была создана некоммерческая организация **Unicode Consortium**, в которую вошли представители многих компьютерных фирм (Borland, IBM, Lotus, Microsoft, Novell, Sun, WordPerfect и др.). Сегодня она тесно взаимодействует с рабочей группой ISO/IEC/JTC1/SC2/WG2, которая занимается разработкой международного стандарта 10646 (ISO/IEC 10646)[36], занимается развитием и

[36] Между стандартом Юникода и ISO/IEC 10646 установлена синхронизация, хотя каждый стандарт использует свою терминологию и систему документации.

внедрением стандарта «The Unicode Standard». Последний состоит из двух основных частей: универсального набора символов (UCS от англ. universal character set) и семейства кодировок (UTF от англ. unicode transformation format). Универсальный набор символов перечисляет допустимые по стандарту Юникод символы и присваивает каждому символу код в виде неотрицательного целого числа, записываемого обычно в шестнадцатеричной форме с префиксом U+, например, U+040F. Семейство кодировок определяет различные способы преобразования UCS кодов символов для более компактного и удобного их хранения или передачи.

В новой кодовой таблице от принципа однобайтовости ушли частично. Стандарт кодирования символов **Unicode** стал доминирующим в интернациональных программных многоязычных средах. Microsoft Windows NT и его потомки Windows 2000, 2003, XP, Vista, 7, 8, 10 используют **Unicode**, точнее основанный на ней формат **UTF-16** для внутреннего представления текстов. UNIX-подобные операционные системы типа Linux, BSD и Mac OS X, iOS, Android и др. приняли **Unicode** в формате **UTF-8** как основное представление многоязычного текста. Сейчас, в 2020 году можно смело отметить, что последний формат пришёл в мир ЭВМ чтобы стать унифицирующим.

Простой переход с однобайтовых кодировок на двубайтовые не выход, поскольку $2^{16} = 65536$ всё же больше чем 256, но недостаточный в перспективе. Можно было бы придумать трёх- и более байтовое кодирование, но у такого решения появились бы иные проблемы: обратной совместимости со старым оборудованием и программным обеспечением, а также рост объёмов всех ранее закодированных текстов при их перекодировании в новый формат. (Подробнее об этом см. в следующем параграфе.)

*Таблица 2.24. Фрагмент со знаками кириллицы из кодовой таблицы **Unicode***

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
040	Ѐ	Ё	Ђ	Ѓ	Є	Ѕ	І	Ї	Ј	Љ	Њ	Ћ	Ќ	Ѝ	Ў	Џ
041	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
042	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
043	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
044	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
045	ѐ	ё	ђ	ѓ	є	ѕ	і	ї	ј	љ	њ	ћ	ќ	ѝ	ў	џ
046	Ѡ	ѡ	Ѣ	ѣ	Ю	ю	Ꙗ	ꙗ	Ѥ	ѥ	Ж	ж	Ѫ	ѫ	Ѯ	ѯ
047	Ѱ	ѱ	Ѳ	ѳ	Ѵ	ѵ	Ѷ	ѷ	Ѹ	ѹ	Ѻ	ѻ	Ѽ	ѽ	Ѿ	ѿ
048	Ҁ	ҁ	≠	◌҃	◌҄	◌҅	◌҆	◌҇	◌̾	◌̽	Ҋ	ҋ	Ь	ь	Ҏ	ҏ
049	Ґ	ґ	Ғ	ғ	Ҕ	ҕ	Җ	ж	Ҙ	ҙ	Қ	қ	Ҝ	ҝ	Ҟ	ҟ
04A	Ҡ	ҡ	Ң	ң	Ҥ	ҥ	Ҧ	ҧ	Ҩ	ҩ	Ҫ	ҫ	Ҭ	ҭ	Ү	ү
04B	Ұ	ұ	Ҳ	ҳ	Ҵ	ҵ	Ҷ	ҷ	Ҹ	ҹ	Һ	һ	Ҽ	ҽ	Ҿ	ҿ
04C	І	Ӂ	ӂ	Ӄ	ӄ	Ӆ	ӆ	Ӈ	ӈ	Ӊ	ӊ	Ӌ	ӌ	Ӎ	ӎ	Ӏ
04D	Ӑ	ӑ	Ӓ	ӓ	Æ	æ	Ӗ	ӗ	Ә	ә	Ӛ	ӛ	Ӝ	ӝ	Ӟ	ӟ
04E	Ӡ	ӡ	Ӣ	ӣ	Ӥ	ӥ	Ӧ	ӧ	Ө	ө	Ӫ	ӫ	Ӭ	ӭ	Ӯ	ӯ
04F	Ӱ	ӱ	Ӳ	ӳ	Ӵ	ӵ	Ӷ	ӷ	Ы	ы	Ӹ	ӹ	Ӽ	ӽ	Ӿ	ӿ

Новая кодировка **Unicode** была разработана с учётом возможного большого числа представляемых ею символов. Она резервирует 1 114 112 ($2^{20}+2^{16}$) позиций для различных символов, в настоящее время используются более 96 000. Первые 256 кодов символов точно соответствуют таковым ***ISO 8859-1***, наиболее популярной 8-разрядной таблицы символов «западного мира»; в результате первые 128 символов также идентичны таблице ***ASCII***.

В табл. 2.24 показан русский блок **Unicode** (коды от 0400_{16_BE} до $04FF_{16\_BE}$; хотя и хочется не писать лишний 0, идущий перед всеми записями, так не поступают).

Замечание. Естественно полная таблица намного больше и в не показанных ячейках хранятся другие символы или ничего (т. е. в используемой версии кодировки символы не определены).

Кодовое пространство стандарта **Unicode** разделено на 17 планов («planes»), и каждый план имеет 65 536 (= 2^{16}) точек кода (символов). Также планы ещё называют плоскостями.

Нумерация планов идёт с нуля, поэтому первый план –

План 0, – основной многоязычный план (BMP – *Basic Multilingual Plane*) – тот, в котором описано большинство символов. BMP содержит символы почти для всех современных языков и большое количество специальных символов.

Ещё три плана (1, 2, 3) используются для «графических» символов.

План 1. Дополнительный многоязычный план (SMP – Supplementary Multilingual Plane) главным образом используется для исторических символов, а также для музыкальных и математических символов.

План 2. Дополнительный иероглифический план (SIP – Supplementary Ideographic Plane), используется для приблизительно 40 000 редких китайских иероглифов.

План 3. Третичный иероглифический план (TIP – Tertiary Ideographic Plane) используется (зарезервирован) для архаичных китайских иероглифов.

Планы 4 – 13, временно не используются.

План 14. Дополнительный план особого назначения (SSP – Supplementary Special-purpose Plane), отведён для символов, используемых по особому назначению (теги, селекторы написания и др.).

План 15 и ***План 16*** открыты для любого частного использования.

2.2.2. Однобайтное и многобайтное кодирование текстов, кодировки переменной длины

В зависимости от размера[37] кодовой таблицы, для кодирования одного символа может потребоваться один или несколько байтов. Как вы заметили ранее, первоначальные кодовые таблицы (ASCII + расширение) укладывались в 256 вариантов различных символов и им хватало 8 бит (одного байта) для адресации. Для унификации более короткие, например 7-разрядные кодовые таблицы, обычно дополнялись ещё одним нулевым разрядом до восьми и использовались как 8-битные.

С появлением Unicode вариантов символов стало больше и возникла необходимость кодировать их более чем одним байтом. Например это можно было бы сделать двумя, тремя или четырьмя байтами, в зависимости от числа символов, – наиболее

[37] Под размером кодовой таблицы будем понимать число уникальных символов представленных в ней.

простое решение, приходящее на ум. То есть, вместо однобайтовой использовалась бы многобайтовая кодировка. Собственно эта идея была реализована в кодировках UCS-2 и UCS-4 (Universal Character Set длиной в 2 и 4 байта, соответственно).

Однако здравый смысл говорит, что:

- не все символы будут использоваться одинаково часто;
- в результате использования, например трёхбайтовой, кодировки размеры всех однобайтовых текстов в «новом формате» увеличатся в три раза.

Более оптимальное использование памяти могли бы дать кодировки переменной длинны с кодированием символов в зависимости от частоты их встречаемости в тексте (см. алгоритм Хаффмана на стр. 153). По данному пути в чистом виде не пошли, вероятнее всего потому, что захотели обеспечить какую-то обратную совместимость с существовавшими кодировками, да и сбор статистики по всем текстам мира был бы затруднителен. В качестве компромисса на этапе придумывания кодировки Unicode было предложено несколько форматов (Unicode Transformation Format) переменной длинны для кодирования её символов: UTF-7, UTF-8, UTF-16 и UTF-32.

Заключение. Для кодирования одного символа может использоваться один или несколько байтов. Условно по этому признаку все существующие текстовые кодировки можно разделить (классифицировать) по количеству байт затрачиваемых для кодирования одного символа. Если для любого символа достаточного одного байта – кодировка считается однобайтовой, если требуется несколько – многобайтовой.

Кодировки переменной длины используют для разных групп символов разную длину. В них одни символы могут быть однобайтовыми, а другие двух-, трёх- и даже четырёхбайтовыми. В этом случае среднее значение указанного параметра может быть дробным и оно сильно зависит от содержания кодируемых текстов.

2.2.3. Недостатки многобайтовых кодировок

К недостаткам многобайтовых кодировок можно отнести:

1. Излишнюю избыточность (один символ кодируется бóльшим числом битов);

2. Низкую помехоустойчивость (в случае потери одного байта из многобайтового кода все последующие байты смещаются и весь последующий текст перестаёт правильно читаться);

3. У существующего ПО возможны сложности «при виде» байтов с кодами 0÷31 (в кодировке ASCII это служебные коды). Подобно двоичным файлам, например кодировка Unicode мало подходит для непосредственной передачи по сети – байты в тексте вполне могут приходиться на область управляющих символов;

4. Может использоваться разный порядок байт: LITTLE_ENDIAN и BIG_ENDIAN.

Для нивелирования указанных недостатков обычно применяются другие кодировки, а точнее, форматы кодирования, основанные на Unicode. К ним относятся упомянутые выше форматы переменной длины, обозначаемые как UTF (Unicode Transformation Format):

- 7-битная UTF-7 (RFC 2152, 1997 г., зарегистрирована в IANA как UTF-7);

- 8-битная UTF-8 (RFC 2279, 1998 г., зарегистрирована в IANA как UTF-8);
- 16-битная UTF-16 (RFC 2781, 2000 г., зарегистрирована в IANA как UTF-16, UTF-16BE, UTF-16LE);
- 32-битная UTF-32 (в прошлом UCS-4, RFC 3629, 2003 г., зарегистрирована в IANA как UTF-32, UTF-32BE, UTF-32LE).

Для интересующихся историей можно привести уже не поддерживаемые стандартом Unicode форматы UTF-1 и UTF-EBCDIC.

Замечание. Формально ни один из форматов выше (Unicode Transformation Format) не является ни кодировкой, ни кодовой таблицей символов, а является лишь алгоритмом «упаковывания» битов представляющих символы кодировки Unicode (программно распознаваемым кодом относительно исходного Unicode), но, несмотря на это, UTF-7 и UTF-8 зарегистрированы именно как кодировки, наравне с ISO 8859-5 или KOI8-R. Естественно, обе не являются специфически «русскими», а пригодны для написания «сколько угодно»-язычного письма.

Обидно, что даже среди программистов находятся те, кто не «видит» разницу между Unicode и, например UTF-8, ошибочно считая это одним и тем же.

2.2.3.1. Краткое сравнение UCS-2 и UTF-16

UCS-2 и UTF-16 представляют собой две схемы двухбайтового кодирования символов. Цифры 2 и 16 в названиях схем говорят нам о том, что для представления одного символа в них используются 2 байта и 16 бит, соответственно. В ряде случаев между схемами нет разницы, но иногда разница есть и её важно знать.

UCS-2	UTF-16
Старая схема кодирования.	Новая схема кодирования.
Кодирование с фиксированной шириной (два байта для каждого символа => в теории можно адресовать не более 216=65536 символов).	Кодирование с переменной шириной, использует минимум 2 байта и максимум 4 байта для каждого символа. Это позволяет UTF-16 представлять любой символ в Юникоде, используя минимальное пространство для наиболее часто используемых символов.
Для большинства (~65 тыс.) символов UCS-2 и UTF-16 имеют одинаковые кодовые точки, поэтому они в значительной степени эквивалентны. Это позволяет приложениям, поддерживающим UTF-16, правильно интерпретировать коды UCS-2. В обратную сторону утверждение не всегда верно из-за многих улучшений в UTF-16.	
Поддержка направления письменности только в традиционном варианте слева направо (обычное направление для большинства видов письменностей, в частности для латиницы и кириллицы).	Поддержка направлений письменностей как «слева направо», так и «справа налево» (письменность на арабском, иврите, персидском, урду и синдхи).
Нормализация текстов перед сравнением – не поддерживается.	Нормализация текстов перед сравнением – поддерживается.

Последняя функция не всем известна, и в ряде случаев позволяет успешно обходить простые или старонаписанные спам-фильтры[38].

Замечание. Поиск и фильтрация для современных мультиязычных текстов не такая и простая задача, как может сначала показаться. Регулярными выражениями просто не отделаться, потому как дело не только в различных пробельных символах и регистре. Кроме всяких ударений, шляпок, дужек, кружочков и иных подписей у букв есть и другие проблемы. Например, знатоки немецкого языка скажут, что мляуты можно не писать, пользуясь следующей схемой замен букв в словах ü = ue, ö = oe, ä = ae, ß = ss. Знатоки английского языка сообщат, что в предложениях слова «cannot» и «can't» есть одно и тоже, а также приведут с десяток других, подобных этому случаев. Знактоки …. языка, скажут что …. и т.д.

2.2.4. Кодирование символов таблицы Unicode в формате UTF-8

В **UTF-8** [33] все символы разделены на несколько групп по значению первых битов. Символы с кодами менее 128_{10} кодируются одним байтом, первый бит которого равен нулю, а последующие 7 бит в точности соответствуют $2^7 = 128$ символам 7-битной таблицы ***ASCII*** (см. табл. 2.14), следующие $2^{11}-128 = 1920$ символов – двумя байтами (*Greek*, *Cyrillic*, *Coptic*, *Armenian*, *Hebrew* и *Arabic*-символы). Последующие символы кодируются тремя и четырьмя байтами. Попробуйте сами рассчитать сколько их будет.

*Таблица 2.25. Принцип кодирования символов в **UTF-8***

Диапазон адресуемых кодов (*hexadecimal*)	Байт в записи символа	Вид UTF-8 (*binary*), комментарий
00– 7F 000000–00007F	1	0aaaaaaa Первый бит 0, следующие 7 соответствуют таблице **ASCII**
0080– 07FF 000080–0007FF	2	110xxxxx 10xxxxxx Первые 3 бита 110 – всего используется 2 байта, второй байт начинается с 10
0800– FFFF 000800–00FFFF	3	1110xxxx 10xxxxxx 10xxxxxx Первые 4 бита 1110 – всего используется 3 байта, второй и третий байты начинаются с 10
010000–10FFFF	4	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx Первые 5 бит 11110 – всего используется 4 байта, второй, третий и четвёртый байты начинаются с 10
и так далее, для большего числа байт (пока не используются)		

Рассмотрим **алгоритм** записи кода одного символа Unicode в UTF-8 следующий:

1. Если размер символа в кодировке UTF-8 = 1 байт, то его код имеет вид (0aaaaaaa), где «0» – просто ноль, остальные биты «a» – это побитный код символа в кодировке ASCII;

2. Если размер символа в кодировке UTF-8 больше 1-го байта (то есть от 2 до 6):

[38] См. Номрализация Unicode *http://habr.com/ru/post/45489/*.

2.1. Первый байт символа содержит информацию об общем количестве байтов этого символа. Число подряд идущих единичных битов от начала байта определяет общее число байтов, и, наоборот. См. табл. 2.25, или более коротко:

```
11xx xxxx  => 2 байта:  .... ....
111x xxxx  => 3 байта:  .... .... .... ....
1111 xxxx  => 4 байта:  .... .... .... .... .... ....
1111 1xxx  => 5 байт:   .... .... .... .... .... .... .... ....
1111 11xx  => 6 байт:   .... .... .... .... .... .... .... ....И Т.Д.
```

2.2. Затем следует «0» – бит терминатор, означающий завершение кода размера;

```
2 — 110x xxxx
3 — 1110 xxxx
4 — 1111 0xxx
5 — 1111 10xx
6 — 1111 110x
```

2.3. Затем идут значащие биты до конца текущего байта;

```
2 — 110x xxxx
3 — 1110 xxxx
4 — 1111 0xxx
5 — 1111 10xx
6 — 1111 110x
```

2.4. Далее идут один или несколько значащих байтов, которые имеют вид (`10xx xxxx`), где «`10`» – биты признака продолжения, а `x` – значащие биты.

В общем случае варианты представления одного символа в кодировке UTF-8 выглядят так:

(1 байт) 0aaa aaaa
(2 байта) 110x xxxx 10xx xxxx
(3 байта) 1110 xxxx 10xx xxxx 10xx xxxx
(4 байта) 1111 0xxx 10xx xxxx 10xx xxxx 10xx xxxx
(5 байт) 1111 10xx 10xx xxxx 10xx xxxx 10xx xxxx 10xx xxxx
(6 байт) 1111 110x 10xx xxxx 10xx xxxx 10xx xxxx 10xx xxxx 10xx xxxx

Пример. Закодируем русскую букву «а» в UTF-8.

1. Смотрим в таблицу Unicode (см. табл. 2.24, стр.204) и определяем Unicode-код буквы «а» – 0430_{16_BE}.
2. Переводим 0430_{16_BE} в 2-ичный вид (см. табл. 2.2, стр. 31): 0000 0100 0011 0000.
3. Так как русские буквы кодируются 2 символами, то, исходя из определения длины кода UTF-8, получаем 11 значащих бит (см. [6]) и выделяем их с конца 0000 0100 0011 0000.
4. Перегруппировываем биты из `100 0011 0000`
   в `10000 110000`.
5. Соотносим данные биты с местом, куда они будут записаны в 2-х байтовой записи:

```
110x xxxx 10xx xxxx      получаем    110x xxxx 10xx xxxx
   10000    110000       ------->       1 0000   11 0000
```

6. Подставляем их и записываем полученный результат `1101 0000 1011 0000`.
7. Переводим получившееся двоичное число в шестнадцатеричный вид (см. табл. 2.2, стр. 31), результат готов: D0 B0.

Повторим шаги 1-6 для последующих букв нижнего регистра русского алфавита и запишем результат в виде таблицы:

Буква	а	б	в	г	д	е	ё	ж	з	и	й	к	л
unicode	0430	0431	0432	0433	0434	0435	0451	0436	0437	0438	0439	043A	043B
UTF-8	D0 B0	D0 B1	D0 B2	D0 B3	D0 B4	D0 B5	D1 91	D0 B6	D0 B7	D0 B8	D0 B9	D0 BA	D0 BB
Буква	м	н	о	п	р	с	т	у	ф	х	ц	ч	...
unicode	043C												
UTF-8													

Предлагаем читателям самостоятельно завершить заполнение таблицы, в том числе для букв верхнего регистра и некоторых часто используемых знаков препинания.

Замечание 1. Совершать подобные манипуляции каждый раз вручную (в том числе и для других символов) неудобно, поэтому в жизни программисты обычно прибегают к различным упрощающим жизнь ранее рассчитанным справочным таблицам, либо программам их рассчитывающих.

Замечание 2. Проверить полученный результат, в случае использования локальной кодировки utf-8 в системе (см. 2.2.7.1. Локализация в ОС Linux на стр.107), довольно просто

```
$ echo -n "а"|od -tx1
0000000 d0 b0
```

Замечание 3. Проверить полученный результат, в случае использования другой локальной кодировки, например koi8-r, также довольно просто

```
$ echo -n "а"|iconv -f koi8-r -t utf-8|od -tx1
0000000 d0 b0
```

2.2.5. Транслитерация, ASCII art

Особняком стоит 7-битная, русская «кодировка» – **транслитерация**, или **транскириллица**, когда русские буквы передаются похожими по звучанию или написанию английскими (primerno takim obrazom). Первоначально такой способ кодирования использовался в первых ПК привозимых из зарубежа, когда установить программы русификации не было возможности.

Как правильнее: подбирать буквы по звучанию (например «ш» – «sh», «ч» – «ch») или схожему начертанию («ш» – «w», «ч» – «4») ответить сложно, потому как это не стандарт, а очередная придумка наших находчивых граждан, «псевдокодировка». Если при фонетическом соответствии задача написать программу конвертирующую однозначно тексты «туда» и «обратно» кажется реальной, то при визуальном способе подбора соответствий задача нерешаема простой заменой.

Вторую жизнь в «псевдокодировку» вдохнули SMS'ки. Максимальный размер сообщения в стандарте GSM – 140 байт (1120 бит). Поэтому, такой способ кодирования можно встретить когда нужно уложиться в отведённые 160 символов или 140 байт. Максимальное число символов «влезает» в сообщение при 7-битном кодировании. При использовании 8-битной кодировки (например, немецкий и французский языки использующие точки над буквами (мляуты) и др. значки) можно отправлять сообщения длиной до 140 символов. Для поддержки других национальных алфавитов (китайского,

арабского, русского и др.) используется 2-байтовая (16-битная) кодировка UCS-2. Таким образом, SMS, написанное кириллицей, не может превышать 70 знаков. В ряде телефонов прошлого поддерживались 8-битные кодировки кириллицы – KOI8-R и CP1251, но при использовании их возникали проблемы с совместимостью с другими телефонами, где данных кодовых таблиц нет. Естественно, при таком выборе кодовой таблицы невозможна отправка сообщений с использованием других алфавитов, кроме кириллицы и латиницы.

Замечание. В «обществе потребления» не выгодно экономить поэтому, «корпорациям добра» предоставляющим услуги сотовой связи, проще переложить дополнительную финансовую нагрузку на плечи абонентов. Вот и получается что для передачи русских букв аппараты по умолчанию используют невыгодную, с точки получаемого конечного размера сообщений, кодировку UCS-2. Ответом на эту несправедливость в России даже были попытки законодательно защитить абонентов, уровняв размеры оплаты за сообщения длинной в 160 символов, независимо от используемого алфавита: кириллицы или латиницы, но они быстро сошли на нет.

По аналогии с принципом похожести символов (по написанию) используемом при транслитерации, ASCII символы могут схожим образом использоваться для изображения графики. Так, при использовании моноширинных шрифтов, одни символы оказываются темнее, а другие светлее, например сравните «.» и «Ш». Библиотека ASCII art library[39] использует этот принцип и позволяет смотреть на текстовых экранах (консолях) видео. Например, с помощью программы mplayer (см. рис. 2.13).

Рисунок 2.13. Просмотр видео на текстовом терминале

Если монитором, текстовым режимом, алгоритмом кодека и исходным файлом видео поддерживается цвет – то можно смотреть и в цвете.

```
$ aafire
...
$ mplayer -vo aa -monitorpixelaspect 0.5  видеофайл.avi
```

2.2.6. Специальные символы

Наличие большого числа количество кодовых таблиц, хотя бы для той же кириллицы, объясняется не только первоначальной слабой заинтересованностью в сотрудничестве друг с другом между разными разработчиками компьютеров и программного обеспечения, но и некоторым прогрессом в развитии сетевых технологий и самой вычислительной техники. Как можно догадаться, кодовые таблицы (кодировки) не являются чем-то незыблемым, поскольку это всего лишь упрощающие жизнь договорённости между людьми вида: под символом с номером *n* понимать букву «*М*» и они, как

[39] См. http://aa-project.sourceforge.net/aalib/, либо пакет aalib из репозитария EPEL.

и любые другие договорённости, могут быть не только уточнены и дополнены, в зависимости от меняющихся потребностей общества, но и пересмотрены. Приведём несколько примеров.

2.2.6.1. Знаки валют

Так, в прошлом веке с появлением валюты «евро» возникла необходимость в использовании на компьютерах знака этой валюты, поэтому в конце 1997 года ряд кодовых таблиц подверглись ревизии с целью включения в них нового символа евровалюты – "€" (Евро, EUR), он был помещён в позицию 128 (0x80) большинства таблиц и в позицию 136 (0x88) русской таблицы CP1251.

В этом веке ситуация повторилась, 11 декабря 2013 года Банк России утвердил графическое обозначение рубля (символ рубля, валюта RUR) в виде прописной буквы «Р» кириллического алфавита, дополненной в нижней части горизонтальной чертой. [34] Немногим позднее, 16 июня 2014 года была принята очередная версия Unicode – версии 7.0, включившая в себя изображения знака российского рубля. [35] Таким образом, с того момента и до версии 12 (последней на момент подготовки учебника в 2020 году) в Unicode символ рубля находится на позиции 20 BD_{16}.[40]

Замечание. Регулярному обновлению подвергаются отнюдь не все существующие кодовые таблицы (а за ними шрифты, программы, системы и устройства), поэтому новые символы в непересмотренных таблицах (или в старых версиях) вы не найдёте. На таких системах, вновь введённые значки и символы не будут отображаться корректно, вместо них, скорее всего, пользователю будет показан небольшой квадратик. Он будет либо пустой, либо с кодом символа, для того чтобы разные квадратики всё же можно было как-то отличать друг от друга.

2.2.6.2. Смайлики (значки, маленькие цветные картинки, иконки, эмодзи)

Отдельного упоминания заслуживает группа символов «эмóдзи»[41], – в простонародье смайлики и иные небольшие картинки различной тематической направленности. Если традиционно шрифт воспринимается как подстановка по номеру символа из таблицы его чёрно-белого начертания в необходимом размере, то, что мешает придумать и добавить в таблицы новых шрифтов символы не похожие на традиционные элементы существующих письменностей, ни на знаки препинания, ни на цифры, ни на буквы, ни на иероглифы, ни на вязь из нескольких буквы, ни на что-нибудь другое. Вы уже точно видели несколько подобных символов и значков (например, в первой главе масти карт: ♠, ♣, ♥, ♦ на стр. 15) среди текста, особенно, если занимались изучением учебника путём перелистывания вперёд и назад. Двум значкам мы специально добавили привычный для них цвет – красный, поскольку следующим логичным вопросом за этим действием будет: А что мешает символы в системах и программах с графическим интерфейсом выполнять двухцветными или многоцветными? Одно дело традиционная полиграфия и желание сэкономить при печати хотя бы на краске (взять наш учебник: он цветной и поэтому не дешевле своих аналогов), а другое дело современные телевизоры, мониторы и экраны мобильных устройств.

[40] В HTML существует ещё несколько способов для отображения знака валюты рубля – графикой (например с помощью картинки тегом <img src="...">), либо с помощью кода ₽

[41] От японского 絵 – картинка и 文字 – знак, символ; в англ. emoji.

Следующий логичный шаг в развитии данного направления, – использование анимации (наверняка вы уже видели и подмигивающий смайлик, и тот у которого над головой кружатся маленькие звёздочки, обозначающие головокружение и много других анимаций), а это по сути аналогичная замена некоторого номера на его отображение из некоторой базы (библиотеки) цветных, возможно и анимированных, изображений (иконок, значков, начертаний).

Первые эмодзи, появившиеся в 1990-х годах, никак не были связаны с unicode и представляли собой значки определяемые матрицей из 12×12 точек. Впоследствии (по времени синхронно с появлением системы коротких сообщений у операторов сотовой связи), они стали плавно заменять в культуре написания текстов использовавшиеся во времена FIDO некоторые сокращения для выражения эмоций (типа LOL – громкий смех и др.) и ASCII-эмотиконы, более привычные и понятные нам как текстовые смайлики: :-) :'-(-))) ;-) >:-(:-p и др. Возможно, на развитие эмодзи повлияло и то, что короткие текстовые сообщения набирались пользователями уже на сотовых телефонах, а не на стационарных компьютерах с большими клавиатурами и экранами. В какой-то момент обошлось не без желающих заработать на картинках за счёт их лицензирования и последующей продаже прав на их использование.

Однако, несмотря на появляющиеся то здесь, то там различные библиотеки картинок (в том числе и платные), жирную и объединяющую точку среди всех желающих использовать эту технологию регулярно ставит unicode консорциум, постепенно, от версии к версии, добавляющий новые значки в наборы доступных пользователям символов. Так в разные годы (с момента появления и до сегодняшних дней) новые версии unicode добавляли новые значки. Когда-то их число было больше, когда-то меньше. Рекордное добавление (+722 значка) оказалось в 6.0 версии, а самое маленькое (+2) в 3.0 версии. На сегодня в версии 13.0 насчитывается 1329 значков.

В основном это символы с кодами лежащими в диапазонах U+1F300 … U+1F5FF, U+263A, U+1F600 … U+1F64F, причём им всем стандартом ISO 15924 даже выделен свой код письменности – «Zsye» с номером 993.

Просмотреть как выглядит тот или иной символ можно с помощью стандартных программ.

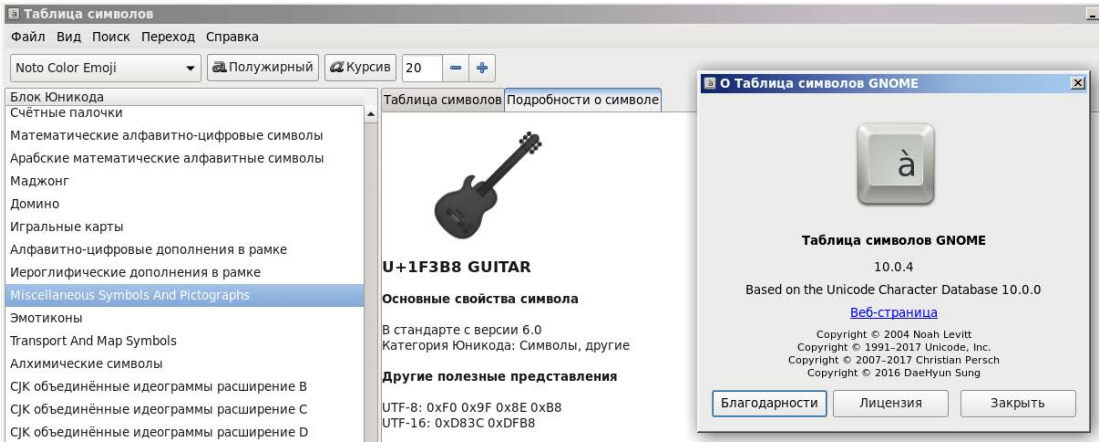
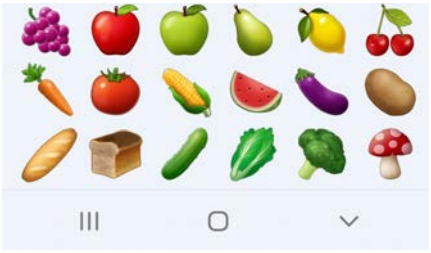


Рисунок 2.14. Просмотр отдельных символов у шрифтов в программе gucharmap

Мобильные устройства, их операционные системы и приложения развиваются очень быстро, поэтому сообщение о том, что начиная с версии 4.4 в ОС Android имеется поддержка эмодзи в стандартной виртуальной клавиатуре скорее вызовет улыбку (тут как раз не хватает нужного эмодзи в тексте, например вроде этих 😉 😊 😋), потому как было это аж в 2013 году, поскольку доля мобильных устройств с этим годом выпуска на рынке минимально и год от года продолжает уменьшаться и найти такие работающие устройства всё сложнее и сложнее. Что же программного обеспечения, то выпущенные в 2020 году версии многих (если не всех) популярных приложений для мгновенного обмена сообщениями, такие как WhatsApp, Telegram, ВКонтакте и др. естественно позволяют использовать наборы эмодзи.



Указанные символы получили широкое распространение не только из-за красоты самих картинок (хотя вопрос красоты очень спорный), а, скорее, потому, что при их хранении (и передаче по каналам связи) по сравнению с традиционными графическими анимированными форматами требуется в разы меньше места. Например, символу с кодом «`f0 9f 92 bb`» (формат UTF-8) соответствует цветной значок с изображением компьютера. Если в системе для диапазона кодов символов, включающих данный код, установлен шрифт[42], то при его печати в текстовой консоли графической среды (окне терминала) будет отображена соответствующая картинка, а в тех системах, где указанного шрифта нет – будет изображён квадратик содержащий unicode код символа.

```
$ echo $'\xf0\x9f\x92\xbb'
💻
$ echo $'\xf0\x9f\x92\xbb'
01F
4BB
```

В завершение отметим, что, в некотором роде суть эмодзи наглядно демонстрирует следующий анекдот:

Анекдот. *Поручик Ржевский (в оригинале или любая другая личность – прим. авт.) заходит в развлекательное заведение (что-то типа современного антикафе – прим авт.) и видит знакомую девушку в незнакомой компании. Он проявляет любопытство и присоединяется к ним. Кто-то из компании произносит: Тридцать пять! Все смеются. Через какое-то – кто-то другой: «Семнадцать.» Все в компании опять смеются. «Это да, а, сто девятый!» – перебивает его другой и все смеются пуще прежнего. А что вы делаете?, – недоумеваючи интересуется поручик. Мы рассказываем анекдоты, – отвечает ему один из компании, – просто мы все давно знакомы, не один день встречаемся и все из нас уже все анекдоты выучили. Рассказывать каждый раз неинтересно и долго, поэтому, с целью экономии времени, мы их пронумеровали и, если хотим рассказать анекдот, то просто называем его номер.*

Интересно, интересно, – понимающе кивает поручик, – а можно мне попробовать? Да, конечно, попробуйте, – слышит он в ответ. Ни минуты не задумавшись: Двадцать второй! – выдаёт поручик, первое, что ему приходит на ум. Компания притихла, никто не смеётся, некоторое замешательство, все опустили глаза. Поручик в недоумении. Неловкая пауза. Одна женщина подходит и даёт пощёчину поручику. Как? За что? – возмущается он, не понимая, что произошло.

Извините, поручик, – кто-то вежливо отвечает из компании, – но такие пошлые анекдоты мы при дамах не рассказываем.

[42] Например «noto», для CentOS 7 это пакет google-noto-emoji-color-fonts-20180508-4.el7.rpm. Программы также должны поддерживать цветные шрифты, например LibreOffice поддерживает их начиная с версии 6.3.2. Экспортированные из него PDF-файлы также требуют обновлённой программы просмотра.

2.2.6.3. Специальный символ «BOM»

Глядя на последовательность шестнадцатеричных чисел, так и по косвенным признакам, определить, в какой кодовой таблице представлен текст, конечно, можно, но в некоторых случаях, особенно с UTF, сделать это однозначно правильно невозможно. Отчасти и потому, что при одних и тех же кодировках в разных системах в многобайтовых записях используется разный порядок байт. Поэтому, чтобы упростить жизнь пользователям (и программам), был придуман специальный символ «BOM», который можно хранить или передавать вместе с текстом. Аббревиатура BOM расшифровывается как *Byte Order Mark* (*отметка о порядке байтов*) – Unicode-символ, используемый для индикации порядка байтов текстового файла. По спецификации его использование не является обязательным, однако если BOM используется, то он должен быть установлен в начале текстового файла (тестового блока). Помимо своего конкретного использования в качестве указателя порядка байтов, символ может также указать, каким форматом кодировки Unicode закодирован текст. Например, см. Табл. 2.26.

Таблица 2.26. Символ BOM и обозначаемые им формат кодировки и порядок байт

BOM-символ	Формат кодировки	Порядок байт
`EF BB BF`	UTF-8	для много-байтовых знаков не определён
`FE FF`	UTF-16 (BE)	Big Endian
`FF FE`	UTF-16 (LE)	Little Endian
`00 00 FE FF`	UTF-32 (BE)	Big Endian
`FF FE 00 00`	UTF-32 (LE)	Little Endian

Согласно спецификации Unicode, символ «U+FEFF» (для случая UTF-16 (BE)) в середине потока данных должен интерпретироваться как «нулевой ширины неразрывный пробел» (по существу, отсутствующий символ). Однако Unicode 3.2 настоятельно рекомендует использовать в этом качестве символ «U+2060» («Word Joiner»), а «U+FEFF» использовать только как «BOM».

В большинстве случаев использование символа «BOM» для пользователя прозрачно, например тот же «блокнот» (notepad.exe для ОС Windows), открывая файл с «BOM»-символом вначале текста, никак его не покажет. Узнать о его наличии можно или косвенно – например, файл будет на 2 байта больше, или используя hex-режим просмотра файлов.

Замечание. Данный вопрос наглядно демонстрирует, что неграмотно составленные тесты не могут дать объективную оценку знаниям экзаменуемого. Допустим, вам попался вопрос: сколько будет занимать текстовый файл, содержащий слово «мама», написанное кириллицей? Как вы догадываетесь, правильным ответом может быть и 4 байта, и 8 байт, и 10 байт, и 4096 байт и др. Как отвечать? Какие кодировки подразумевались? Знал ли разработчик теста о том, что могут использоваться один или несколько символов «BOM»? Имелся ли размер используемого места файлом на диске или чистый размер содержимого?

2.2.6.4. Символ(ы) перевода (конца) строки

DOS, Windows	UNIX, Linux, Mac OS
`0D 0A`	`0A`

Во времена матричных принтеров существовало несколько команд, посылаемых данным устройствам при печати, а именно, когда каретка заканчивала печатать строку ей посылалась команда Carriage Return (CR, 13_{10}, 0x0D) на возврат её в начало строки,

после чего, по получению принтером кода Line Feed (LF, 10_{10}, 0x0A), происходило прокручивание валика подачи бумаги на 1 интервал вперёд и тем самым под каретку «подъезжала» новая строчка бумаги. Так как указанные коды обычно посылались на принтер вместе (естественно последовательно), то в текстовых файлах, подготавливаемых для отправки на печать в текстовых редакторах операционных систем DOS и Windows за указанными двумя кодами закрепился один символ «новой строки» (new line), часто встречаемый в программах как «\n». Иногда этот символ называют «символом конца строки», что не совсем корректно.

В ОС Linux и UNIX для перевода строк используется лишь один символ, эквивалентный символу (line feed, LF, 10, 0x0A). Проблем с печатью в указанных системах не наблюдалось, демон печати (обычно lpd) выполнял «конвертирование» текстовых файлов в команды конкретного принтера на лету.

Что интересно в макинтошах до ОС Mac OS X в качестве символа перевода строки в текстовых файлах изначально использовался символ (carriage return, CR, 13, 0x0D). Так как Mac OS X основана на UNIX системе FreeBSD, то в ней используется один символ (line feed, LF, 10, 0x0A).

Для конвертирования файлов из одного формата в другой используются утилиты dos2unix и unix2dos. Например,

```
$ dos2unix 2.txt
dos2unix: converting file 2.txt to UNIX format ...
```

Многие современные редакторы умеют работать со несколькими форматами перевода строки. Например даже такой простой редактор как «встроенный в Far Manager по клавише F4», автоматически определяет формат перевода строки из открываемого файла и продолжает сохранять изменения в таком же формате.

2.2.6.5. Другие специальные символы

Следует отметить, что существует ряд коррекционных символов, например один из них «Right-To-Left Override» (UTF-16BE код `20 2E`, UTF-8 код `E2 80 AE`), подробнее он рассмотрен на стр. 552.

Существуют и другие редко используемые обычным пользователем символы и значки не относимые ни к каким группам, причём некоторые из них визуально схожи с традиционными ASCII-символами. Все их знать невозможно, но помнить, что они есть и не пугаться, когда они вам встретяться, стоит. Например, символ косой черты «/» (UTF-8 код `E2 81 84`), рассмотренный также на стр. 552.

2.2.7. Работа с текстовыми данными на практике

Для работы с текстами в составе большинства ОС Linux имеется большое количество всевозможных консольных утилит, не говоря уже о том, что при взаимодействии с операционной системой через интерфейс командной строки пользователь набирает команды в виде текстов, а на консоли (окне терминала) ему в ответ выдаются также текстовые данные. О большинстве из них с примерами хорошо рассказано в [2].

Полезными могут оказаться консольные команды cat, od, echo, cp, mv, rm, sed, grep, cut, iconv, dd (пакет coreutils), dos2unix и unix2dos (пакет dos2unix), strings (пакет binutils) и др. Некоторые из них представлены в кратком справочнике приведённом в табл. 2.27.

Таблица 2.27. Краткий справочник консольных команд ОС Linux для работы с текстами

Команда с параметрами и её краткое описание
`touch file1.txt` создать пустой файл с именем file1.txt в текущей директории, если файл имеется – модифицировать время последнего доступа к нему
`echo "мама" >file2.txt` создать файл file2.txt и записать в него слово «мама» и символ перевода строки (\n, 0x0A), если файл уже имеется – предварительно стереть всё в нём
`echo -n "папа" >file3.txt` создать файл file3.txt и записать в него слово «папа», если файл уже имеется – предварительно стереть всё в нём
`echo "сестра" >>file4.txt` дозаписать слово «сестра» и символ перевода строки (\n, 0x0A) в конец файла file4.txt, если файла нет, то создать его
`echo -n "брат" >>file5.txt` дозаписать слово «брат» в конец файла file5.txt, если файла нет, то создать его
`cat file6.txt` вывести содержимое файла file6.txt на консоль ***Замечание.*** Вывод происходит в первый поток, он же STDOUT. Будьте внимательны, таким образом можно вывести любой файл, в том числе и двоичный, содержащий служебные символы с кодами 0-31, что может повлиять на внешний вид консоли.

`od -t x1 file7`
посмотреть содержимое файла file7 в hex формате
`rm file8.odt`
удалить файл file8.odt

`iconv -l` посмотреть список кодировок, поддерживаемых командой iconv (см.разд. 2.2.7.2)
`iconv -f CP1251 -t UTF-8 < input.txt > output.txt` прочитать содержимое файла input.txt в кодировке CP1251 и сохранить в формате UTF-8 в файл output.txt
`unix2dos < in.txt | iconv -f UTF-8 -t UTF-16 > out.txt` Преобразовать файл in.txt из Unix-формата и кодировки UTF-8 (Unicode) в новый файл out.txt в формате Windows (DOS) и кодировке UTF-16 (Unicode) , если файл out.txt существует, то он будет перезаписан
`dd if=file2.txt of=file9.txt bs=1 count=4 skip=2` скопировать 4 байта (4 блока размером в 1 байт) из файла file2.txt в файл file9.txt начиная с третьего байта (пропустить 2 от начала) (результат работы команды зависит от кодировки, если файл file2.txt содержит слово «мама» в формате UTF-8, то в файле file9.txt окажется «ам»)
`echo -n $'\33'>>file11.txt` Дозаписать символ с кодом 33_8 в файл file11.txt
`echo -n $'\x1c'>>file12.txt` Дозаписать в файл file12.txt символ с шестнадцатеричным кодом «`1c`»
`echo -n $'\xd0\xb0'>>file13.txt` Дозаписать два 16-ричных символа «`d0`» и «`b0`» в файл file13.txt (русская буква «а» в UTF-8)
`echo -n $'\x41'` Вывести на консоль символ с шестнадцатеричным кодом 41 – латинская буква «A»
`cat file1>>file2` Дозаписать file1 в конец файла file2. Если file2 отсутствует – создать его.

2.2.7.1. Локализация в ОС Linux

При выводе текстовой информации на экран ЭВМ используется та или иная кодовая таблица. До загрузки ОС используется таблица взятая из BIOS'a видеокарты, обычно там прошита ASCII. Как легко догадаться, любая ОС, в том числе и Linux, при выводе сообщений на консоль использует свои собственные настройки. (Настройки графической среды X-Window и оконных менеджеров мы рассматривать не будем.) Выполнение простой команды, выводящей содержимое файла на консоль

```
$ cat file.txt
```

будет сильно варьироваться от настроек, называемых «локализацией».

Локализация означает приспособление программы или ОС к кодировке и стилям печати времени, даты, денежных единиц принятых в данной стране. Стандарт описан, например, в документах POSIX 1.c Расширения потоков (IEEE Std 1003.1c-1995). Ниже приведены отдельные моменты.

Если не задано иное, то системные программы в ОС Linux для определения вида кодировки и языка сообщений используют установленные переменные окружения. К таким переменным можно отнести переменные LANG и LC_*, посмотреть которые можно командой

```
$ env|grep "LANG\|LC_"
LANG=ru_RU.utf8
```

Если переменные с именем вида LC_* в окружении не установлены, то берутся их значения по умолчанию, доступные через команду locale:

```
$ locale
LANG=ru_RU.utf8
LC_CTYPE="ru_RU.utf8"
LC_NUMERIC="ru_RU.utf8"
LC_TIME="ru_RU.utf8"
LC_COLLATE="ru_RU.utf8"
LC_MONETARY="ru_RU.utf8"
LC_MESSAGES="ru_RU.utf8"
LC_PAPER="ru_RU.utf8"
LC_NAME=ru_RU.utf8
LC_ADDRESS="ru_RU.utf8"
LC_TELEPHONE="ru_RU.utf8"
LC_MEASUREMENT="ru_RU.utf8"
LC_IDENTIFICATION="ru_RU.utf8"
LC_ALL=
```

Детально параметры локализации можно посмотреть через команду locale, указав интересующие параметры в ключе -k

```
$ locale -k LC_PAPER
height=297                width=210                paper-codeset="UTF-8"
```

Значительная часть параметров локализации не относятся к кодированию текстов, поэтому их полный список вы можете изучить самостоятельно.

Отметим лишь, что определить какие в вашей ОС Linux имеются файлы для локализации, можно заглянув в директорию /usr/share/locale/, либо запустив команду

```
$ locale -a
```

Общий вид строк, которые присваиваются переменным окружения, должен быть примерно таким: «язык_территория.кодировка», например «ru_RU.utf8».

2.2.7.2. Перекодирование текстов, ошибки

Давным давно возникала потребность читать различные тексты...

Витязь на распутье. 1882. – В.М. Васнецов

Они были написаны не только с использованием разных языков и шрифтов, но и наверно кодировок, если бы тогда это слово (понятие) существовало.

В современном мире для перекодирования текстовых файлов и строк из одной кодировки в другую в ОС Linux (и большинстве UNIX-систем) существует утилита iconv, поддерживающая довольно большое число всевозможных кодировок.

```
$ iconv -l
437, 500, 500V1, 850, 851, 852, 855, 856, 857, 860, 861, 862, 863, 864, 865, 866,
866NAV, 869, 874, 904, 1026, 1046, 1047, 8859_1, 8859_2, 8859_3, 8859_4, 8859_5,
8859_6, 8859_7, 8859_8, 8859_9, 10646-1:1993, 10646-1:1993/UCS4, ANSI_X3.4-1968,
ANSI_X3.4-1986, ANSI_X3.4, ANSI_X3.110-1983, ANSI_X3.110, ARABIC, ARABIC7, ARMSCII-8,
ASCII, ASMO-708, ASMO_449, BALTIC, BIG-5, BIG-FIVE, BIG5-HKSCS, BIG5, BIG5HKSCS,
BIGFIVE, BRF, BS_4730, CA, CN-BIG5, CN-GB, CN, CP-AR, CP-GR, CP-HU, CP037, CP038,
... пропущено несколько страниц вывода ...
LATINGREEK1, MAC-CENTRALEUROPE, MAC-CYRILLIC, MAC-IS, MAC-SAMI, MAC-UK, MAC,
MACCYRILLIC, MACINTOSH, MACIS, MACUK, MACUKRAINIAN, MIK, MS-ANSI, MS-ARAB, MS-CYRL,
MS-EE, MS-GREEK, MS-HEBR, MS-MAC-CYRILLIC, MS-TURK, MS932, MS936, MSCP949, MSCP1361,
MSMACCYRILLIC, MSZ_7795.3, MS_KANJI, NAPLPS, NATS-DANO, NATS-SEFI, NATSDANO,
NATSSEFI, NC_NC0010, NC_NC00-10, NC_NC00-10:81, NF_Z_62-010, NF_Z_62-010_(1973),
NF_Z_62-010_1973, NF_Z_62010, NF_Z_62010_1973, NO, NO2, NS_4551-1, NS_4551-2,
NS_45511, NS_45512, OS2LATIN1, OSF00010001, OSF00010002, OSF00010003, OSF00010004,
OSF00010005, OSF00010006, OSF00010007, OSF00010008, OSF00010009, OSF0001000A,
OSF00010020, OSF00010100, OSF00010101, OSF00010102, OSF00010104, OSF00010105,
OSF00010106, OSF00030010, OSF0004000A, OSF0005000A, OSF05010001, OSF100201A4,
OSF100201A8, OSF100201B5, OSF100201F4, OSF100203B5, OSF1002011C, OSF1002011D,
OSF1002035D, OSF1002035E, OSF1002035F, OSF1002036B, OSF1002037B, OSF10010001,
OSF10010004, OSF10010006, OSF10020025, OSF10020111, OSF10020115, OSF10020116,
OSF10020118, OSF10020122, OSF10020129, OSF10020352, OSF10020354, OSF10020357,
OSF10020359, OSF10020360, OSF10020364, OSF10020365, OSF10020366, OSF10020367,
OSF10020370, OSF10020387, OSF10020388, OSF10020396, OSF10020402, OSF10020417, PT,
PT2, PT154, R8, R9, RK1048, ROMAN8, ROMAN9, RUSCII, SE, SE2, SEN_850200_B,
SEN_850200_C, SHIFT-JIS, SHIFT_JIS, SHIFT_JISX0213, SJIS-OPEN, SJIS-WIN, SJIS,
SS636127, STRK1048-2002, ST_SEV_358-88, T.61-8BIT, T.61, T.618BIT, TCVN-5712, TCVN,
TCVN5712-1, TCVN5712-1:1993, THAI8, TIS-620, TIS620-0, TIS620.2529-1, TIS620.2533-0,
TIS620, TS-5881, TSCII, TURKISH8, UCS-2, UCS-2BE, UCS-2LE, UCS-4, UCS-4BE, UCS-4LE,
UCS2, UCS4, UHC, UJIS, UK, UNICODE, UNICODEBIG, UNICODELITTLE, US-ASCII, US, UTF-7,
UTF-8, UTF-16, UTF-16BE, UTF-16LE, UTF-32, UTF-32BE, UTF-32LE, UTF7, UTF8, UTF16,
```

```
UTF16BE, UTF16LE, UTF32, UTF32BE, UTF32LE, VISCII, WCHAR_T, WIN-SAMI-2, WINBALTRIM,
WINDOWS-31J, WINDOWS-874, WINDOWS-936, WINDOWS-1250, WINDOWS-1251, WINDOWS-1252,
WINDOWS-1253, WINDOWS-1254, WINDOWS-1255, WINDOWS-1256, WINDOWS-1257, WINDOWS-1258,
WINSAMI2, WS2, YU
```

Список выше содержит практически все известные авторам встроенные кодировки. Это не обязательно означает, что можно использовать любые комбинации данных имён в параметрах командной строки FROM и TO (ключи -f и -t, соответственно). Одна и та же кодировка может быть перечислена под несколькими именами (псевдонимами). Прим. авт.

Перекодировать файл довольно несложно:

```
$ iconv -f WINDOWS-1251 -t UTF8 < input.txt > output.txt
```

Если текст несколько раз в случайных комбинациях перекодировать, не придавая значения его первоначальной кодировке и содержимому, то может получиться ситуация, напоминающая карикатуру ниже. (К сожалению, найти её автора не удалось.)

Во избежание подобной ситуации, т.е. для успешного перекодирования требуется явное указание правильной исходной кодировки текста с помощью ключа «`-f`». Однако, как обычно случается в жизни, исходная кодировка текста часто оказывается неизвестной и её как раз и требуется определить для осуществления конвертирования. В этом случае, можно использовать утилиты enca и enconv (из пакета enca репозитария EPEL). Первая занимается определением (угадыванием) кодировки любого текстового файла:

```
$ enca file1.txt
KOI8-R Cyrillic
  LF line terminators
```

А вторая позволяет не заниматься последующей подстановкой полученных результатов в параметр «`-f`» утилиты iconv, совершая для этого последовательно два действия: определение кодировки и перекодирование файла в кодировку текущей локали (см. locale в 2.2.7.1. Локализация в ОС Linux на стр.107).

Честно говоря, изображённая автором рисунка сцена больше похожа на случаи отсутствия необходимых шрифтов, и, это уже скорее вопрос отображения информации (из разделов графики или печати, нежели кодирования текстов).

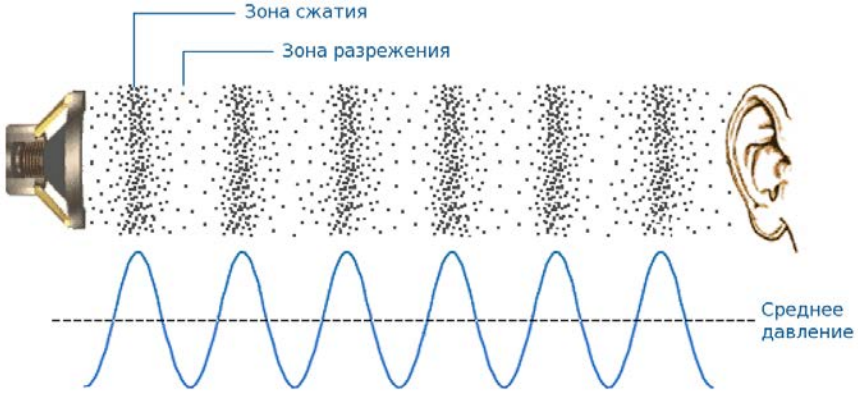
Следует понимать, что в случае ошибочного кодирования (или конвертирования, но не отображения) могут возникнуть потери информации, которые приведут к тому, что правильно раскодировать записанный текст уже не предоставится возможным, однако это бывает редко, поэтому выходов из ситуации существует два: либо использовать перебор всевозможных вариантов и надеяться, что правильное сочетание нескольких перекодирований будет найдено, либо использовать статистические свойства закодированного текста, сведя задачу к раскрытию сообщения, закодированного с помощью шифра простой замены.

В любом случае, проблема не нова. Полезными в её решении могут оказаться навыки программирования либо давно написанные (и не поддерживаемые разработчиками) программы: codepage для DOS [43] или Shtirlitz для Windows, так и более современные типа PuntoSwither, xneur и др.

[43] Для ОС Linux существует эмулятор DOSBox, для iOS – приложение-эмулятор iDOS, а для Android – aDosBox (репозиторий http://f-droid.org).

2.3. Кодирование звуковой и аналоговой информации

Задача кодирования звуковой информации является частной задачей кодирования (восстановления) аналоговых сигналов. Аналоговая информация окружает нас повсеместно и представляет собой показания, снимаемые какими-либо датчиками на протяжении какого-то времени, при этом снимаемые показания могут меняться быстро или, наоборот, медленно. В первом приближении, если измерять перемещения мембраны или геометрические размеры какого-либо тела (внутри микрофона), как следствие череды сжатий и разрежений воздушного пространства, то можно сделать предположение, что мы говорим о звуке. Обычно под звуком понимают такие колебания, которые оказываются слышны человеческому уху.



Спектрально-амплитудный диапазон слышимости нормального человека отображён на рис. 2.15.

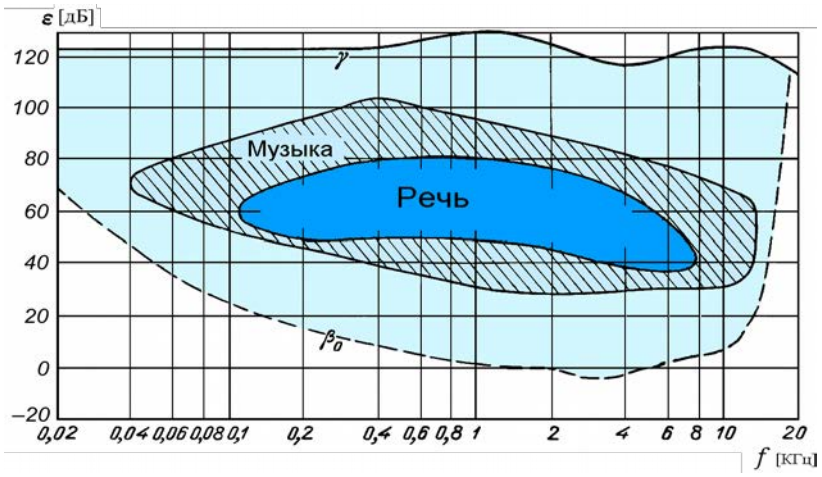
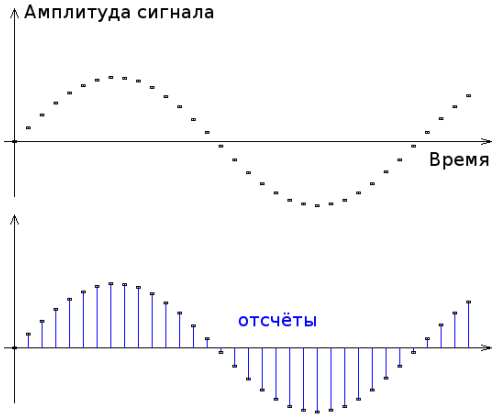


Рисунок 2.15. Спектрально-амплитудный диапазон слышимости среднестатистическим человеком

Учёные Владимир Александрович Котельников и Гарри Найквист [44] независимо показали, что аналоговые сигналы, в силу того, что измеряемые показатели меняются плавно, обладают тем свойством, что по своей форме кривые изменений (справа это синусоида) могут быть восстановлены не только при непрерывном снятии показаний, но и по некоторым выборочным точкам (отсчётам).

Они показали, что не имеет смысла снимать точки чаще, чем происходят «изменения в графике».



Владимир Александрович Котельников (1908–2005). Советский и российский учёный в области радиотехники, радиосвязи и радиолокации планет. Наиболее важные научные достижения: открытие теоремы отсчётов, создание теории потенциальной помехоустойчивости, давшей учёным и инженерам инструмент для синтеза оптимальных систем обработки сигналов в системах связи, радиолокации, радионавигации, разработка планетарных радиолокаторов и проведение с их помощью фундаментальных астрономических исследований. Академик РАН по Отделению технических наук (радиотехника) с 1953 года. В 1999 году Международный научный фонд Эдуарда Рейна (Германия) признал приоритет В. А. Котельникова, наградив его премией в номинации «за фундаментальные исследования» за впервые математически точно сформулированную и доказанную в аспекте коммуникационных технологий теорему отсчётов.

Теорема В.А.Котельникова *(теорема Найквиста, теорема Шеннона–Найквиста, теорема об отсчётах)* гласит, что *любой аналоговый сигнал с ограниченным спектром может быть представлен конечным числом дискретных отсчётов взятых с частотой в 2 раза чаще чем максимальная частота спектра сигнала*.

Замечание 1. Доказательство теоремы основано на преобразовании Фурье и приводить его здесь не имеет смысла. Формулировка теоремы без потерь смысла адаптирована специально для данного учебника. Оригинал см. в [43].

Замечание 2. Уважаемый читатель, прочитав теорему, подумайте и ответьте: как сочетаются на рисунке с предыдущей страницы частота синусоидального сигнала и частота взятых с него отсчётов? Отсчёты идут слишком редко, слишком часто или в самый раз?

Указанная теорема несколькими десятилетиями позднее позволила свести задачу представления звуковой информации в памяти ЭВМ к измерению интенсивности звука через заданный интервал времени (например, 48 раз за 0,001 секунды). Принцип такого представления изображён на рис. 2.16.

[44] né Harry Theodor Nyqvist (1889–1976), родился в Швеции, в 1907 г. переехал в США, один из основоположников «Теории информации». Его работы по определению ширины частотного диапазона, требуемого для передачи информации, заложили основы для последующих успехов в разработке теории информации. Исторически считается одним из авторов теоремы Найквиста–Шеннона, в сущности аналогичной теореме отсчётов Котельникова.

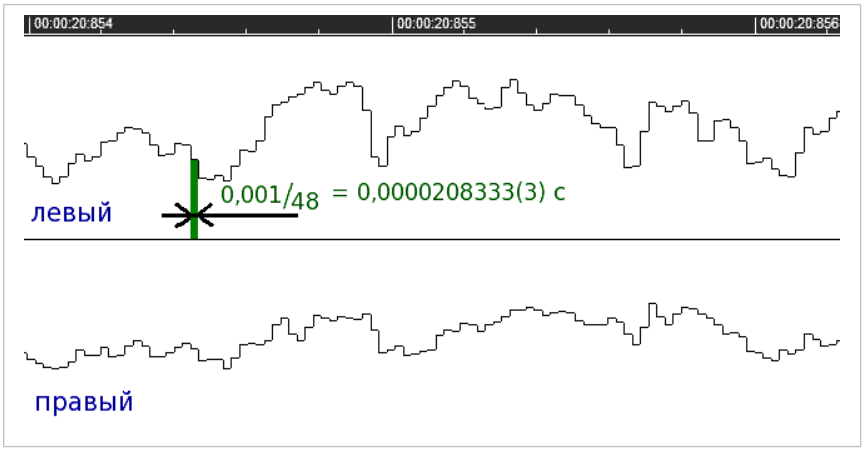


Рисунок 2.16. Диаграмма стереозвука в музыкальном редакторе (верхняя ось – время ~ от 20,854 до 20,856 сек., то есть 0,002 сек.)

В предложенном примере каждое измеренное значение представлено в виде «столбика» шириной 1/48 от 0,001 секунды и высотой пропорциональной уровню сигнала в момент считывания показаний. Своей «угловатостью» изображение напоминает гистограмму. По факту же, как аудио-редакторы при отображении формы сигнала на экран, так и цифро-аналоговые преобразователи (ЦАП'ы), при воспроизведении сигнала на аудиовыход используют интерполяцию, в результате чего мы видим более естественную и красивую картинку (см. рисунок 2.17).

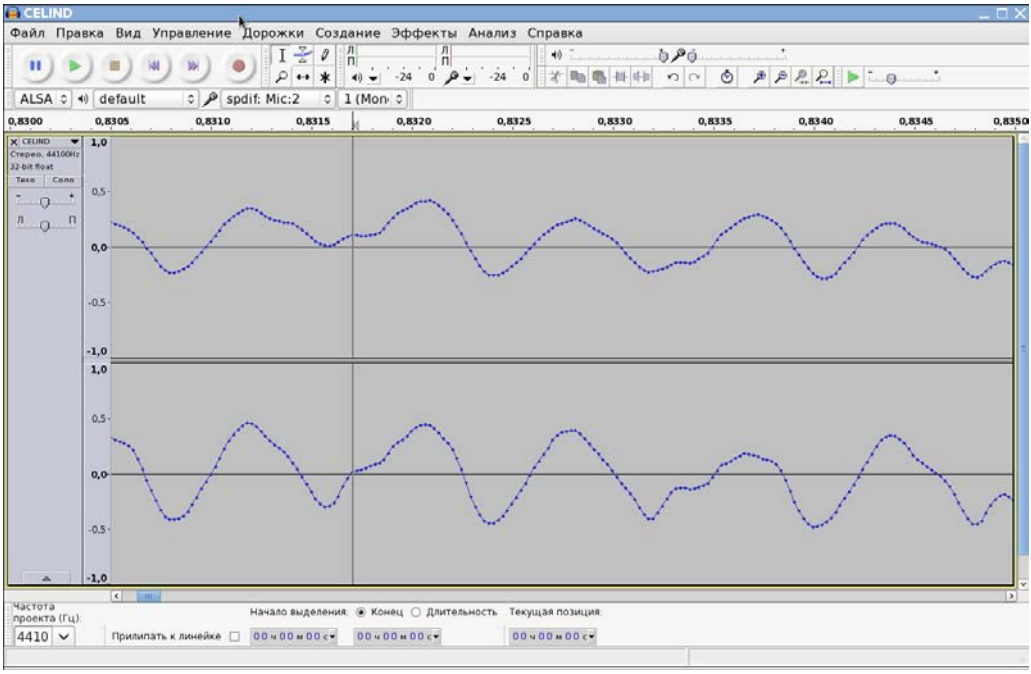


Рисунок 2.17. Окно аудиоредактора Audacity

На рис. 2.17. приведён снимок окна популярного аудиоредактора Audacity (свободное ПО), моменты измерений звукового сигнала и их значения обозначены точками. Для удобства восприятия точки соединены интерполированной кривой. Ощущение же получается обратным: взяли кривую и на ней поставили точки... но это не так. Грубый «прямоугольный вид» скорее нужен для более лёгкого понимания процесса «оцифрования» аналогового сигнала, а именно, что этот процесс происходит в три простых шага:

Первый шаг. На выходе каждого измерения получается некоторое числовое (аналоговое) значение амплитуды.

Непрерывная звуковая волна разбивается на отдельные участки по времени, для каждого устанавливается своя величина амплитуды. Каждой ступеньке присваивается свой уровень громкости звука, который можно рассматривать как набор возможных аналоговых состояний. Этот процесс называется дискретизацией.

Дискретизация – преобразование непрерывной функции в дискретную. (Получение из непрерывной функции значений её отсчётов, взятых в отдельные моменты времени, определяемые частотой дискретизации.)

Второй шаг. Так как информация в ЭВМ не хранится в аналоговом виде, то значения измерений (амплитуды отсчётов) квантуются (по факту измеряются с конечной точностью, которая определяется разрядностью аналого-цифрового преобразователя, – АЦП). Результат этих измерений уже можно представить в виде некоторого конечного числа (которое и является значением указанного отсчёта), представимого в памяти ЭВМ в цифровом виде.

Квантова́ние (англ. quantization) – в информатике – разбиение диапазона значений непрерывной или дискретной величины (например, амплитуды уровня сигнала) на конечное число интервалов.

Третий шаг – это кодирование. В ряде случаев кодирование как таковое отсутствует и все полученные на этапе квантования значения записываются в память одно за другим по мере проведения измерений.

Если действия для всех трёх шагов собрать вместе, то в литературе этот алгоритм называются как «Импульсно-кодовая модуляция» (ИКМ, англ. *Pulse Code Modulation*, PCM) или линейной ИКМ[45].

Как вы можете догадаться, указанный метод кодирования довольно простой и быстрый, но память при его использовании расходуется не эффективно. Эффективности можно добиться или последующим сжатием данных, либо использованием другого способа кодирования на этапе 3. Например, можно записывать в память не значение измеренного отсчёта, допустим $n = 8$ (или 16) бит, а разницу, по отношению к последнему измерению. Поскольку соседние отсчёты обычно мало отличаются, то для кодирования разницы можно использовать заведомо меньше 8 (или 16) бит. Так были придуманы сначала «Дифференциальная ИКМ», а позднее (в начале 1970-х) «Адаптивная

[45] Линейная, потому как используется линейная шкала для измерения амплитуд. В ряде случае возможно использование логарифмической шкалы и тем самым получение более широкого динамического диапазона. Думаем, что вы наверняка слышали про систему Dolby, работающую схожим образом.

дифференциальная ИКМ» (АДИКМ, англ. *Adaptive differential pulse-code modulation*, ADPCM).

Более сложные «алгоритмы оцифрования» аналогового сигнала чаще называются алгоритмами сжатия или «кодеками», потому процесс всегда можно поделить на два шага: первый – непосредственно получение ИКМ-последовательности (этапы 1,2,3) и второй шаг (этап 4) – сжатие полученных значений тем или иным алгоритмом (кодеком). К сожалению, рассмотрение алгоритмов работы кодеков выходит за рамки тем рассматриваемых в учебнике.

Так как хранимыми в памяти ЭВМ аналоговыми сигналами чаще всего оказывается звук, то рассмотрим «звуковую» терминологию.

2.3.1. Терминология

Основные характеристики качества звука:

1) ***Точность выборки*** (англ. *sampling size*[46])***, или глубина кодирования звука*** – количество бит на одно измерение величины звукового сигнала или количество возможных значений амплитуды (поскольку это связанные вещи).

Современные недорогие звуковые карты обеспечивают 16-битную глубину кодирования звука. Количество уровней (градаций амплитуды) можно рассчитать по формуле: $N = 2^I = 2^{16} = 65\,536$ уровней сигнала (градаций амплитуды).

2) ***Частота выборки*** (англ. sampling rate), ***или частота дискретизации*** – это количество измерений уровня звукового сигнала за 1 секунду.

Одно измерение в 1 секунду соответствует частоте 1 Гц.

1000 измерений в 1 секунду – 1 кГц. Количество измерений обычно находится в диапазоне от 8000 до 48 000 (8 кГц – 48 кГц).

8 кГц используется в телефонных сетях общего пользования при уплотнении телефонных каналов (считается, что разборчивая речь укладывается в диапазоне 0,3–3,4 кГц, поэтому фильтры настраивают на полосу пропускания 0–4 кГц, а дискретизация берётся как удвоенная максимальная частота).

48 кГц соответствует качеству звучания, хранимого звуковыми компакт-дисками.

На практике значения частоты дискретизации, применяемые в звуковых системах, могут настраиваться; часто они равны 44,1 кГц или 48 кГц.

Чем больше частота дискретизации, тем более качественный звук мы можем кодировать.

Описанный выше способ кодирования не подразумевает сжатия данных (звука) с целью экономии дискового пространства. Обычно кодирование с целью сжатия рассматривается отдельно, потому как способов сжатия (как с потерями, так и без) существует великое множество.

3) ***Битрейт*** – характеристика сжатого звука и видео использующая понятие количества бит, необходимых для хранения или передачи одной секунды потока данных. Величина измеряется в килобитах в секунду (kbps). Битрейт характеризует как плотность упаковки информации, так и её качество. Обычно указанная характеристика приводится в сочетании с каким-либо используемым алгоритмом сжатия. Например, из двух MP3-файлов, сжатых с разным битрейтом, более качественный (близкий к

[46] size – по англ. размер. Больше значений – больше точность и больше размер. Точность по англ. будет accuracy, precision.

оригиналу) звук будет у файла с большим битрейтом. В то же время файл другого формата, при равном битрейте, может дать как лучшее, так и худшее качество звука. Стандартов кодирования двухканальной и многоканальной (5.1 и 7.1) аудиоинформации насчитывается несколько десятков, наименования некоторых из них, используемых в современных методах записи мультимедиа информации, приведены далее в табл. 2.28 (см. стр.136).

2.3.2. Формат кодирования FLAC

Выше (в первом грубом приближении) мы узнали, что звуковая волна для ЭВМ представляется в виде ступенчатой кривой. Ширина ступеньки тем меньше, чем больше частота выборки, а высота ступеньки тем меньше, чем больше точность АЦП.

Возможности наиболее распространённой современной аппаратуры предусматривают работу с частотой выборки до 48 кГц (48 тысяч раз в секунду!), что позволяет правильно описывать звук частотой до 22,05 кГц (формат CD Digital Audio)[47]. В формате DVD-Audio для записи звука применяется кодирование линейной ИКМ с разрядностью 16, 20 или 24 бит и частотой дискретизации 44,1 , 48 , 88,2 , 96 , 176,4 или 192 кГц, что позволяет записывать звуки (формально уже и ультразвуки тоже) с большим частотным диапазоном.

У оптических дисков есть один недостаток, – их физический размер. Они хотя и меньше чем грампластинки, но заметно уступают современной флеш-памяти. Поэтому на замену указанным форматам приходит формат FLAC[48] (англ. Free Lossless Audio Codec) – популярный свободный кодек, предназначенный для сжатия аудиоданных без потерь. Файлы в формате FLAC можно хранить во флэш-памяти, передавать по сети, да и также записывать на оптические носители, но в режиме «данных». По сравнению с WAV форматом, использующим ИКМ для записи звука без потерь, файлы получаются намного меньше. А по сравнению с аудио-кодеками, обеспечивающими сжатие с потерями (MP3, AAC, WMA, Ogg Vorbis) FLAC, как и любой другой «lossless-кодек», не удаляет никакой информации из аудиопотока обеспечивая самое высокое качество. Он подходит как для прослушивания музыки на высококачественной звуковоспроизводящей аппаратуре, так и для архивирования аудиоколлекций.

Сегодня формат FLAC поддерживается множеством устройств от портативных аудиоплееров и автомагнитол до стационарных аудио систем[49]. Естественно, программных реализаций для работы с FLAC великое множество.

2.3.3. Основы звукообработки

«Чисто писано в бумаге,
да забыли про овраги,
как по ним ходить.»
Л.Н. Толстой

Почему мы говорим «звуковая карта», а не ЦАП + АЦП?

Если вы думаете, что теоретической информации, которую вы прочитали несколькими параграфами выше, достаточно чтобы разбираться в кодировании звуков, то сме-

[47] Из одной цифры совсем логично не следует другая, но сейчас не об этом речь.
[48] Формат FLAC https://xiph.org/flac/
[49] Устройства поддерживающие формат FLAC https://xiph.org/flac/links.html

ем вас уверить, что описанное выше, отличается от реальности как парниковые помидоры от грунтовых. Поэтому, если вы гурман, точнее, в нашем случае – меломан, то этот параграф о проблемах звукообработки для вас.

Рассмотрим более подробно, что происходит с аналоговым сигналом и его спектром[50] при дискретизации, как это описано Дмитрием Симаненковым в [36]. На рисунках 2.18 и 2.19 вы видите исходный аналоговый сигнал и его спектр, соответственно.

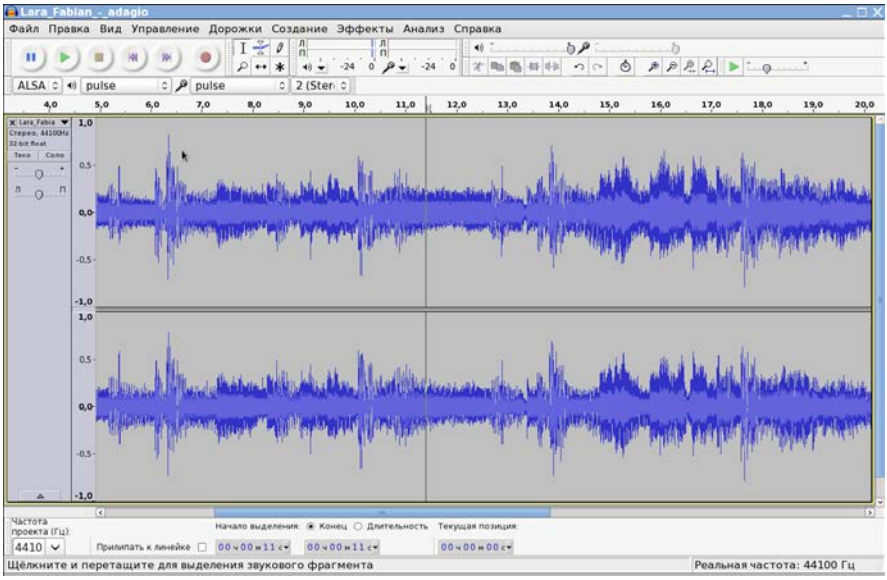


Рисунок 2.18. Пример звукового сигнала, открытого в редакторе Audacity

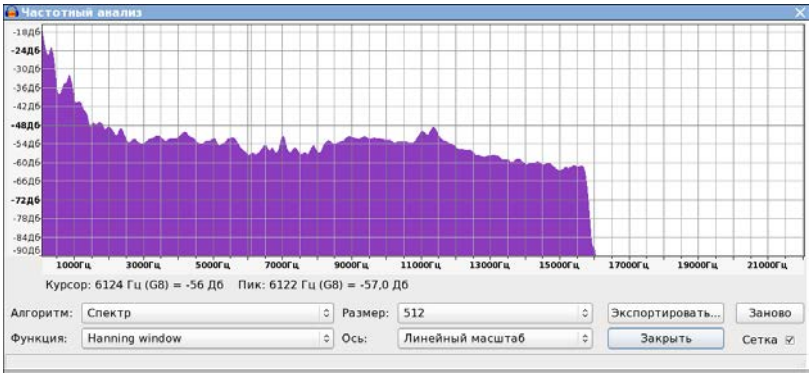


Рисунок 2.19. Спектр сигнала изображённого на рисунке 2.26

В процессе дискретизации частотный спектр существенно изменяется. Что и является камнем преткновения в данном вопросе. Исходный аналоговый звуковой сигнал обычно имеет спектр, в основном лежащий в полосе частот от 20 Гц до 20 кГц. Нижняя граница обычно опускается до нуля, несмотря на то, что у сабвуферов она около 5 Гц. Кроме того, обычно в сигнале содержатся помехи частотой до нескольких

[50] Спектр (лат. spectrum «виде́ние») в физике — распределение значений физической величины (обычно энергии, частоты или массы). Графическое представление такого распределения называется спектральной диаграммой. Математически определение спектра даётся через преобразование Фурье.

сот килогерц, вызванные трудно устранимыми наводками от промышленного и бытового электрооборудования. (Не стоит забывать про электросеть с её напряжением питания частотой в 50Гц и, соответственно, 100 Гц, так как иногда это приносит неприятности (в случае плохих блоков питания). Может появиться дрожание фазы[51]. В своё время этому явлению уделяли много внимания в производстве модемов, даже некоторые рекомендации серии «v.» давались исходя из того как минимизировать указанную проблему.

После дискретизации сигнал представляет собой последовательность узких импульсов разной амплитуды и с очень широким спектром – до нескольких мегагерц (математический факт: чем уже импульс, тем шире его спектр). Поэтому и в целом спектр такой последовательности импульсов расширяется до тех же нескольких мегагерц. Установленный факт: спектр дискретизированного сигнала значительно шире спектра исходного аналогового сигнала.

Рассмотрим подробнее, как формируется новый широкий спектр [36]. Существует два важных процесса. Во-первых, свёртка всего первоначального спектра аналогового сигнала, простирающегося примерно от 0 Гц до нескольких сот килогерц, внутрь полосы частот от 0 Гц до половины частоты дискретизации. Этот процесс изображён на рисунке 2.20.

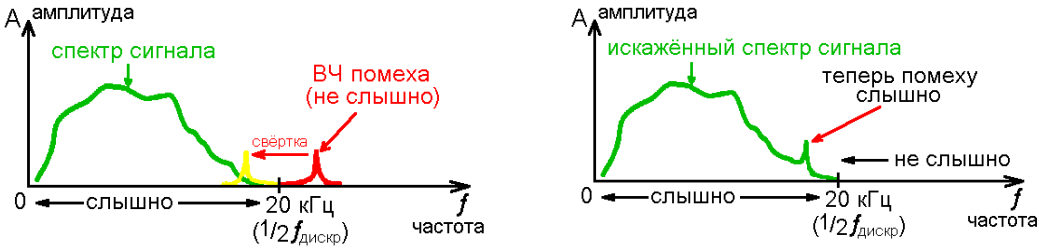


Рисунок 2.20. Процесс свёртки спектра внутрь частотного диапазона 0–20 кГц

Свёртка означает, что все составляющие исходного аналогового сигнала с частотами выше половины частоты дискретизации (а это в основном неслышимые помехи) попадают в слышимый человеческим ухом диапазон частот от 20 Гц до половины частоты дискретизации, то есть неслышимые помехи становятся слышимыми, и, таким образом, может резко ухудшиться отношение сигнал/шум.

Всё это выглядит весьма непривычно, если не сказать, что вообще противоречит здравому смыслу! Получается, что происходит дискретизация высокочастотных сигналов с составляющими, лежащими значительно выше не только половины частоты дискретизации, но и самой частоты дискретизации. На первый взгляд, это даже противоречит упомянутой выше теореме Котельникова, но взгляните ниже.

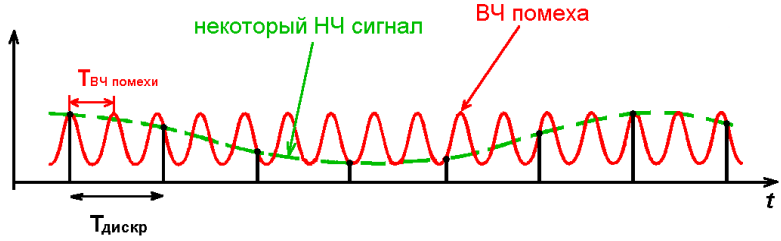


Рисунок 2.21. Дискретизация высокочастотной синусоидальной помехи

[51] Явление дрожания фазы также называют явлением Джиттера.

На рисунке 2.21 показан процесс дискретизации высокочастотного синусоидального сигнала на частоте, которая более чем вдвое ниже частоты дискретизации самого сигнала. Хорошо видно, что период колебаний высокочастотной помехи более чем вдвое меньше, нежели период дискретизации. Поскольку период обратно пропорционален частоте, частота помехи действительно больше частоты дискретизации. Кроме того, дискретные отсчёты (толстые чёрные столбики) высокочастотной помехи полностью совпадают с дискретными отсчётами некоторого низкочастотного сигнала (плавная прерывистая зелёная кривая на рисунке 2.29). Таким образом, после дискретизации высокочастотного сигнала мы получили низкочастотный сигнал. Это и есть эффект свертки высокочастотных помех внутрь частотного диапазона от 0 Гц до половины частоты дискретизации. [36]

Это не противоречит теореме В.А.Котельникова, поскольку там указано, что частота дискретизации должна быть выше частоты сигнала. В данном же примере от нашего внимания ушло, что дискретизируемым сигналом является полезный сигнал плюс ВЧ помеха, которые в своей сумме и выходят за рамки поставленных теоремой ограничений. Надеемся, что вы теперь догадываетесь, зачем стремятся повысить частоту дискретизации, что порой она превышает двойную частоту максимально слышимого человеком сигнала.

Во-вторых, изменение, которому подвергается спектр, заключается в его расширении. Этот факт уже не противоречит здравому смыслу и вполне очевиден. Как было написано выше, относительно низкочастотный исходный аналоговый сигнал в процессе дискретизации преобразуется в последовательность очень узких импульсов. Так как они имеют широкий спектр, то и вся последовательность, разумеется, будет иметь примерно такой же широкий спектр. Вследствие того, что весь исходный спектр свернулся в полосу частот от 0 Гц до половины частоты дискретизации, логично и естественно, что расширение спектра происходит дублированием спектра из полосы от 0 Гц до половины частоты дискретизации на ширину спектра узкого импульса дискретов. То есть окончательно спектр дискретизированного сигнала есть несколько десятков сдвинутых по частоте копий спектров, полученных в результате свёртки спектра исходного аналогового сигнала внутрь полосы частот от 0 Гц до половины частоты дискретизации. Рассмотрим пример, см. рисунок 2.22.

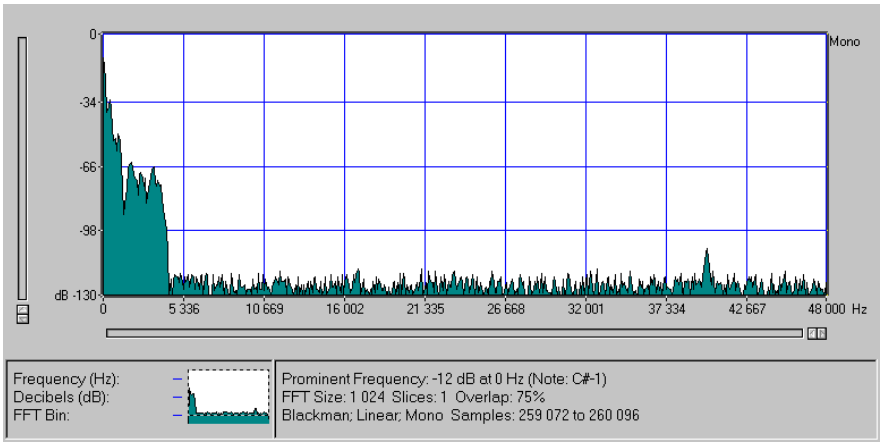


Рисунок 2.22 а. Спектр исходного сигнала (до расширения)

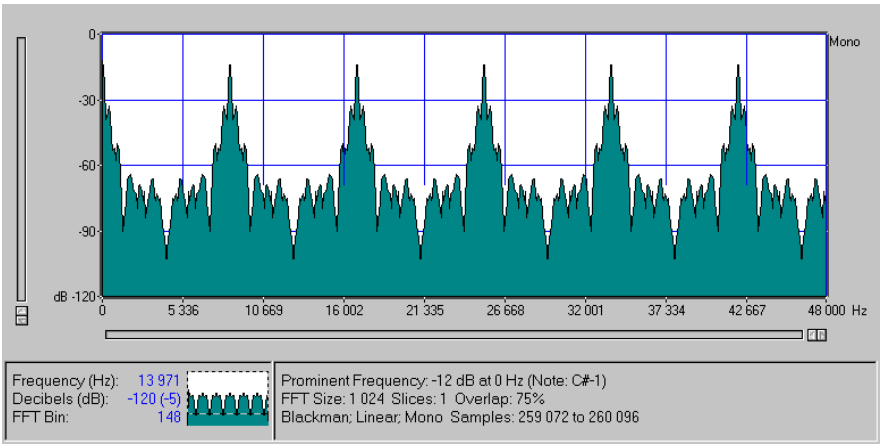
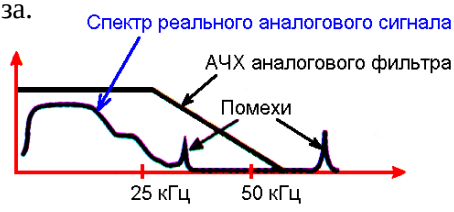


Рисунок 2.22 б. Спектр сигнала дискретизированного на частоте 8 кГц

Видно, что речевой сигнал, заведомо ограниченный по спектру (<4 кГц), при выборе частоты дискретизации 8 кГц проявляет описанный эффект. Если же взять все слышимые нами звуки, то и их высокочастотные составляющие, «плохо» оцифровываются на казалось бы достаточных и стандартных частотах дискретизации в 44,1 кГц или 48 кГц. Конечно, можно использовать и более высокую частоту дискретизации если звуковая карта позволяет, но пропорционально увеличению частоты дискретизации возрастёт:

- интенсивность потока цифровых данных, особенно если записывается или воспроизводится несколько каналов одновременно;
- вычислительная нагрузка на цифровые процессоры эффектов, как-то: «ревербератор», «хорус», «флэнжер» и так далее; (их вычислительные возможности не безграничны, а на дешёвых картах, типа «ЦАП+АЦП», их вообще нет);
- объём памяти, требующийся для хранения цифрового сигнала.

Поэтому, чтобы сильно не увеличивать частоту дискретизации, перед дискретизацией нужно провести аналоговую фильтрацию, которая представляет собой довольно сложную задачу. Аналоговые фильтры не могут пропустить, скажем, все частоты от 0 Гц до 24 кГц и подавить все частоты выше 24 кГц. Аналоговый фильтр низких частот подавляет высокие частоты, начиная с некоторой частоты, называемой частотой среза.

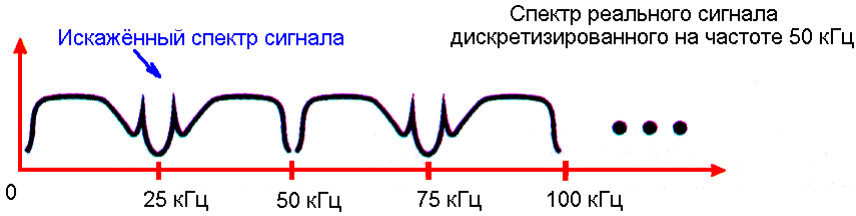


Подавление плавно усиливается с ростом частоты. Поэтому, чтобы добиться отсутствия частот выше 24 кГц, нужно устанавливать частоту среза фильтра равной примерно 16– кГц, а это уже плохо, так как будут ослаблены полезные частоты в слышимом человеческим ухом диапазоне 16–20 кГц (хотя, для не звукового использования это вполне может подойти). [36]

Кроме того, ещё одна неприятность заключается в том, что чем уже мы пытаемся сделать переходную область между полосой пропускания и полосой подавления

(например, используя несколько фильтров, фильтры Чебышева, Баттерворта, Бесселя и др.), тем сильнее выражаются вносимые фазовые искажения, дольше идёт переходный процесс (фильтр начинает «звенеть») и тем сложнее и капризнее в настройке такой аналоговый фильтр.

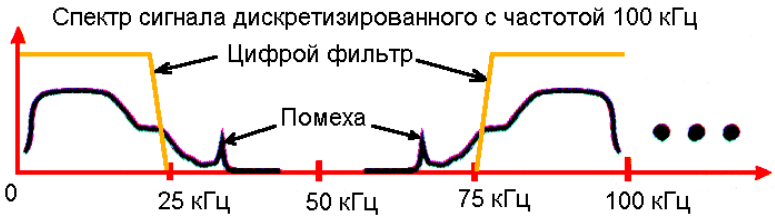
В результате из-за неэффективности фильтров «слышны помехи»:



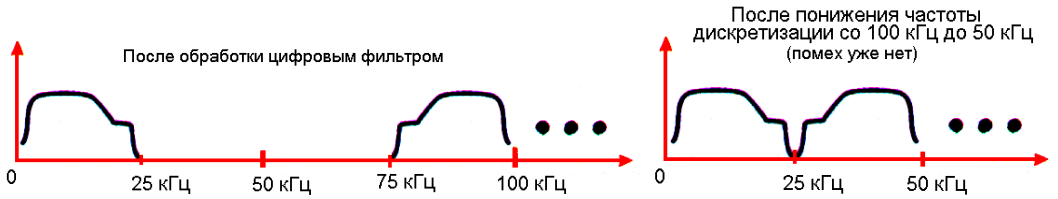
В современных звуковых картах эта проблема решается «оверсэмплингом» (методом дискретизация сигналов с запасом по частоте дискретизации, то есть с частотой в несколько раз превышающей частоту В.А.Котельникова, от англ. oversampling). По этому методу диапазон частот входного аналогового сигнала ограничивается с помощью сравнительно несложного аналогового фильтра. Причём частота среза фильтра выбирается значительно выше полезной высшей частоты, а переходная полоса фильтра делается достаточно широкой. Таким образом, исключаются и завал полезных высших частот, и фазовые искажения, характерные для аналоговых фильтров с узкой переходной полосой. [36]

Далее, отфильтрованный, с ограниченным по частоте спектром, сигнал дискретизируется на достаточно высокой частоте, исключающей наложение и искажение спектра («алиазинг» – от англ. aliasing). Затем дискретные отсчёты сигнала преобразуются в последовательность чисел с помощью АЦП. После этого мы имеем поток цифровых данных, представляющих аналоговый сигнал, включая как полезные, так и нежелательные высокочастотные компоненты и помехи. Эти цифровые данные пропускаются через цифровой фильтр с очень узкой переходной полосой и очень большим подавлением нежелательных высокочастотных компонентов.

Сегодня расчёт и создание таких цифровых фильтров, к тому же не вносящих никаких фазовых искажений, не представляет больших трудностей. Увеличим частоту дискретизации с 50 кГц до 100 кГц и посмотрим на результат:



После цифрового фильтра представленный в цифровом виде сигнал имеет спектр, правильно ограниченный по частоте. Применяя к такому сигналу теорему В.А.Котельникова, мы можем резко понизить частоту его дискретизации до удвоенной величины наивысшей полезной частотной составляющей, – чего мы и хотели добиться.



Надо отметить, что часто цифровые фильтры размещаются в той же микросхеме, что и другие узлы АЦП, так что пользователь может не подозревать, какие сложные процессы протекают в ней!

2.3.3.1. Обратный путь: из цифры в аналог

Применяется дискретизация на повышенной частоте и в цифро-аналоговых преобразователях (ЦАП). В ЦАП также существует проблема сложности аналоговых восстанавливающих (интерполирующих) фильтров. Ведь сразу после простейшего ЦАП сигнал представляет собой серию узких импульсов, имеющих многочисленные «алиазинговые» спектральные компоненты. На аналоговый фильтр в этом случае возлагается задача полностью пропустить сигнал нужного частотного диапазона (скажем, 0–24 кГц) и по возможности наиболее полно подавить ненужные высокочастотные компоненты. Конечно, чисто аналоговому фильтру выполнить такие противоречивые требования не под силу. Поэтому цифровой сигнал сначала интерполируют, то есть вставляют дополнительные отсчёты, вычисленные по специальным алгоритмам, и тем самым резко увеличивают частоту дискретизации. При этом исходный спектр полезного сигнала не искажается, но сигнал уже дискретизирован на значительно более высокой частоте. Это приводит к тому, что «алиазинговые» спектральные компоненты на выходе ЦАП далеко отстоят от частотных компонентов основного сигнала и, чтобы отфильтровать их, достаточно простого аналогового фильтра.

Рассмотрим процесс реконструкции сигнала более подробно. При цифровом представлении в нашем распоряжении имеется информация о величине сигнала только в определённые моменты времени. Мы считаем, что мы не имеем дополнительной информации о форме сигнала между отсчётами. Восстановление формы (или интерполяция) сигнала между отсчётами и является задачей цифро-аналогового преобразования. Интерполяция в современных ЦАП может выполняться нелинейными и линейными (цифровая фильтрация) методами в сочетании с аналоговыми («антиалиазинговыми») фильтрами высоких частот. [36]

Простейшие нелинейные методы интерполяции вполне очевидны. Допустим, мы имеем несколько дискретных отсчётов синусоидального сигнала частотой 100 Гц, взятых через 1/350 секунды (с частотой 350 Гц), то есть частота дискретизации в три с половиной раза выше частоты синусоиды. Конечно, такое соотношение частоты сигнала и частоты дискретизации «лучше», чем теоретический предел 1:2, но будем усложнять задачу постепенно. Построим параболу через три последовательных отсчёта. На рисунке 2.23 вы видите высокую степень совпадения формы параболы и начальной формы синусоиды (соотношение частоты синусоиды и частоты дискретизации не имеет в данном случае принципиального значения, 1:3,5 , или 1:4 , или 1:3,75 существенным образом на вид картинки не повлияет).

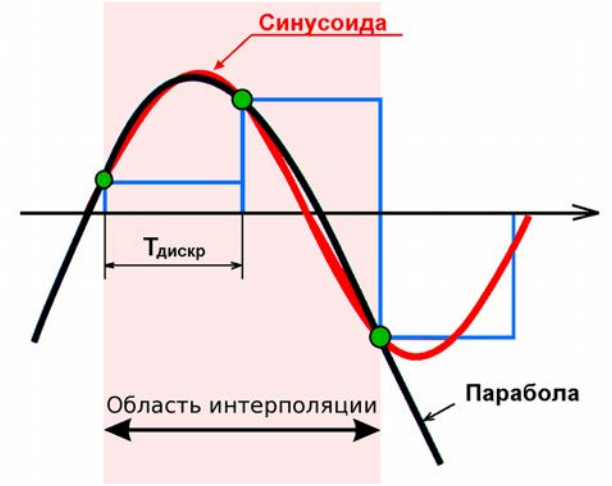


Рисунок 2.23. Параболическая интерполяция сигнала

Таким образом, построив параболу через три последовательных дискрета синусоиды и найдя все коэффициенты её уравнения at^2+bt+c, где a, b, c – коэффициенты, t – время, мы можем вычислить значение сигнала в любой момент времени между дискретами и интерполировать (реконструировать) сигнал между отсчётами с весьма высокой точностью. Оставшиеся небольшие искажения формы восстановленного сигнала (в данном случае синусоиды), по сути, являются нелинейными искажениями. То есть реконструированный сигнал состоит из суммы истинной синусоиды и нескольких её гармоник.

Теперь очевидно, что уменьшить искажения формы восстановленного сигнала можно, просто отфильтровав эти высокочастотные гармоники (вторую, третью и так далее). Цифровые и аналоговые фильтры, применяемые для этих целей даже в дешёвых ЦАП, позволяют подавить вторую гармонику на 80 и более дБ. Это позволяет довести коэффициент нелинейных искажений формы реконструированного сигнала до 0,01–0,02 %. (Это уже почти уровень hi-fi.) [36]

Чтобы получить аналогичные результаты для соотношения частоты сигнала и частоты дискретизации 1:3, нужно применять интерполяцию по четырём последовательным отсчётам гиперболой или использовать сплайны третьего порядка. Если задаться целью интерполировать сигналы, содержащие наивысшую частоту 20 кГц при частоте дискретизации 44,1 кГц, то есть для соотношения частоты сигнала и частоты дискретизации примерно 1:2,2 при коэффициенте нелинейных искажений формы реконструированного сигнала не более 0,01%, то необходимо использовать кривые на основе полиномов 15–20-го порядка. При современном уровне развития схемотехники цифровых сигнальных процессоров (DSP) это вполне возможно, но несколько дороговато.

Линейные методы восстановления формы сигнала по его дискретным отсчётам основаны на использовании цифровых фильтров с конечной и бесконечной импульсной реакцией [36]. К сожалению, их работа не столь наглядна, желающие изучить вопрос подробнее могут начать с [37]. Если коротко, то внутри них происходит следующее: в исходную последовательность отсчётов сигнала между дискретами вставляются нулевые отсчёты. Новая полученная последовательность подаётся на интерполирующий цифровой фильтр. После него нулевые отсчёты «превращаются» в точно ре-

конструированные отсчёты исходного сигнала. Затем для окончательного сглаживания и восстановления сигнал подаётся на простой аналоговый фильтр. Этот метод позволяет получить довольно низкий коэффициент нелинейных искажений формы реконструированного сигнала. Практически все ЦАП'ы современных звуковых карт работают по методу цифровой фильтрации.

Полагаем, что для основ звукообработки изложенного материала достаточно, однако, если вам интересно узнать более, например о том, что такое «джиттер» (англ. jitter), «дизеринг» (dithering, англ. dither от староанглийского didderen – дрожать) и noise shaping, то см. [36].

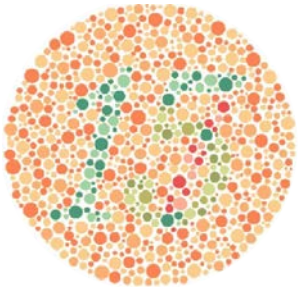
2.4. Кодирование графической и видеоинформации

Лучше 1 раз увидеть, чем 100 раз услышать.
(народная мудрость)

Интересно, а у ЭВМ дело с восприятием обстоит также как и у людей? А соотношение 1:100 имеет место тоже быть или оно другое, более точно подсчитанное? Говорить об этом пока рано и стоит отделить «восприятие» с использованием искуственного интеллекта на базе свёрточных (или возможно уже иных) нейронных сетей от простой попытки «закодировать» воспринимаемое избражение, то есть перевести его в какие-то числовые значения («оцифровать»), чтобы из затем сохранить в памяти ЭВМ. Это может быть «неудобным» для последующего эффективного использования (например поиска объектов на изображениях), но не об этом речь.

Современные компьютеры, в том числе и различные мобильные устройства (планшеты, навигаторы, видеорегистраторы, телефоны, фотоаппараты, MP3-плееры и прочие «гаджеты») запросто справляются с графикой, отображая нам на своих цветных графических дисплеях фотографии и другие изображения. Что стоит за этой простотой?

Во-первых, отметим, что восприятие цветов и изображений людьми не так однозначно как восприятие чисел и знаков (текстовых символов, иероглифов). Наверняка вы в жизни сталкивались с тем, что среднестатистически женщины лучше или точнее «ощущают» цвета. Вероятно, тут есть связь с тем, что генетически они с меньшей вероятностью подвержены заболеванию дальтонизма (H53.5), при наличии которо-

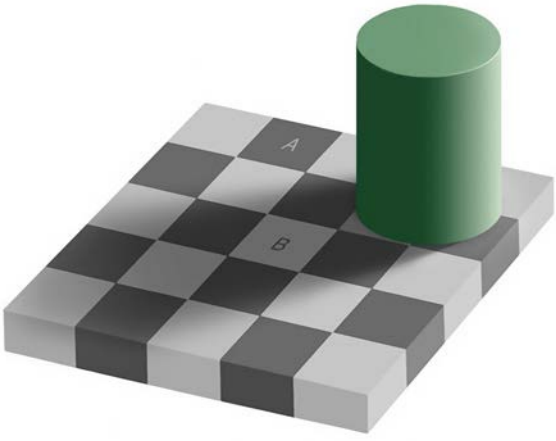


го спорить об адекватности восприятия цветов не приходится. Искажение цветов при известном заболевании объясняется тем, что среднестатистический человек способен воспринять миллионы разнообразных цветовых оттенков благодаря строению глаза, в сетчатке которого имеются специальные клетки для восприятия цвета – колбочки; а для лучшего восприятия яркости при плохой освещённости – палочки. Яркость цвета[52] связана с нашим личным восприятием световых видимых лучей колбочками трёх типов S, M, L. Поскольку их три, обычных людей называют трихроматами. В свою очередь, дальтоники – дихроматы имеют лишь два типа. К слову, дихроматами являются собаки, обезьяны и практически все животные. Многое зависит от общей яркости изображения (как-то коррелирующей или связанной с яркостью освещения). При её уменьшении меняется режим работы глаза человека. Представьте себе ясный день и сравните свои визуальные ощущения с аналогичными в сумерки или в ночь. В восприятии цвета ночью вы уступите даже дихроматам в ясный день и будете всё цветное за счёт «палочек» воспринимать в чёрно-белых (чёрно-серых) оттенках. Также существуют люди (скорее исключения, поскольку их очень мало, обладающие редкими генами, если не мутациями от нормы), в арсенале которых целых четыре колбочки, – тетрахроматы. Естественно, что они способны увидеть ещё больший цветовой диапазон. Поскольку это явление редкое и мало влияющее на повседневную жизнь, основная часть людей даже не догадываются об их существовании, а многие тетрахроматы живущие среди нас и не догадываться о том, что наделены уникальными возможностями.

По аналогии с природой работает и техника занимающаяся переводом «увиденного» в снимки. Последние разделяют на моно-, мульти- и гипер-спектральные, как и оборудование предназначенное для их создания. Наиболее ярко это заметно в описании характеристик спутниковых оптических систем предназначенных для получения данных дистанционного зондирования земной поверхности. Также в быту можно встретить различного рода тепловизоры захватывающие диапазон инфракрасного света и переводящих его в видимый человеческим глазом диапазон.

Так что, бытовое понятие «цветное изображение», вроде как понятное всем, не очень-то и применимо без его адаптации для технических устройств и ЭВМ.

Кроме проблем связанных с кодированием цвета и яркости обозначим ещё одну. Так, отметим, что даже у здоровых людей могут наблюдаться оптические иллюзии. Например, наиболее интересна «иллюзия шахматной доски», предположительно предложенная Эдвардом Адельсоном (Edward H. Adelson). На изображении фрагмента шахматной доски справа квадраты *A* и *B*, хотя и кажутся нам чёрными и белыми, скорее являются параллелограммами, причём одного цвета одинаковой яркости. Последнее можете проверить, если сомневаетесь.



[52] Следует различать понятие яркость света (физическая величина) и яркость цвета (биологическая величина).

Замечание. Для тех, кто не поленился и закрыл, например несколькими листками бумаги мешающие объективному восприятию лишние части изображения, но всё ещё продолжает не верить своим глазам – предлагаем посмотреть на эту тему специально снятый видеоролик http://www.youtube.com/watch?v=z9Sen1HTu5o. Также, для души рекомендуем ознакомиться с работами Морица Корнелиса Эшера (Maurits Cornelis Escher) (1898 – 1972) и на досуге почитать: [44] и *Даниэль Канеман* Думай медленно... решай быстро – М.:АСТ, 2014, ISBN 978-5-17-080053-7.

Учитывая выше изложенное, будем оперировать исходя из среднестатистических возможностей зрения здорового человека. Естественно доля субъективности есть и при восприятии звуков, но там она проявляется не так наглядно.

Обращаем внимание, что «компьютерное зрение» (узнавание и распознавание компьютером различных образов, а также «компьютерная графика» (отображение и прорисовка тех или иных объектов в удобном для человека виде) не входят в рассматриваемые нами вопросы. Наша цель – дать общее понятие читателям, что с достаточной степенью точности та или иная графическая картинка из жизни может быть представлена в памяти ЭВМ некоторой моделью, которая в свою очередь, представима последовательностью «нулей» и «единиц».

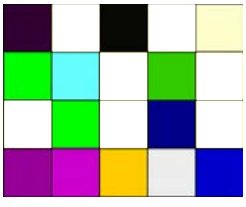
Для более глубокого изучения вопросов компьютерной графики рекомендуем читателям ознакомиться с наглядными публикациями Скарубина Алексея: «Свет и цвет: основы основ» [38] и «Температура цвета» [39], а также с книгой Р. Гонсалеса и Р. Вудса «Цифровая обработка изображений» [40].

2.4.1. Растровая графика

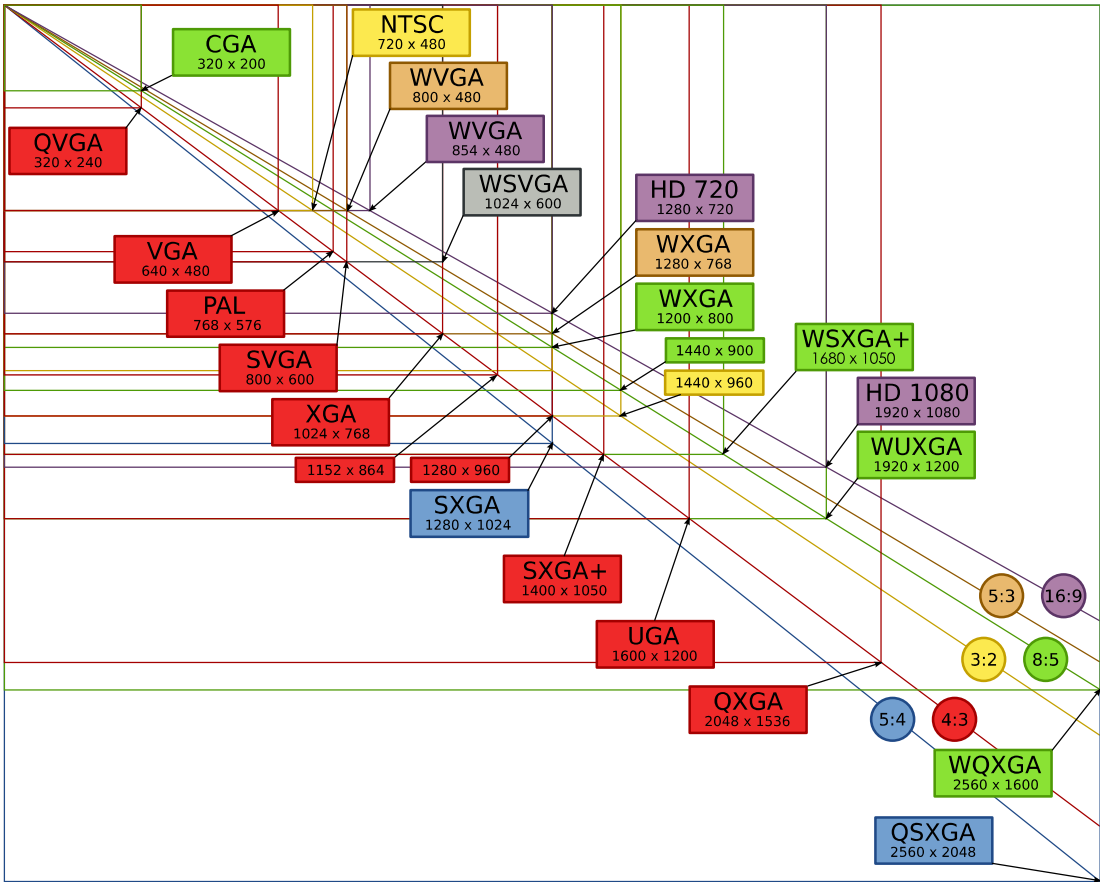
Для понимая того как с помощью растровой графики графическая информация хранится в памяти ЭВМ введём ряд определений:

Точка (pixel, пиксель) – в некоторой степени абстрактный безмасштабный минимальный элемент изображения. Каждой точке присущ набор свойств: горизонтальная и вертикальная координаты (для определения её положения относительно других точек), цвет, прозрачность. Внутри ЭВМ точка не имеет физической размерности, но при отображении её реальными устройствами (монитор, принтер) её размеры могут пересчитываться исходя из разрешающей способности отображающих её устройств. При таком отображении точка приобретает измеримые размеры обычно в виде квадратной области, реже прямоугольной, овальной или иной.

Изображение – набор точек принадлежащих некоторой двумерной области.



Также множество точек называется ***растром*** (bit map, dot matrix, raster), а изображение, которое формируется на основе растра, называется ***растровым***. (Программы, работающие с указанным множеством точек, называются программами для работы с «*растровой графикой*».) На экране монитора всегда формируется растровое изображение, однако для хранения может использоваться и векторное представление информации, где изображение представлено в виде набора графических объектов с их координатами и свойствами (прямоугольник, треугольник, полигон, линия, овал, текст и т. п.). Думаем, как выглядит экран, все и так представляют.



Разрешение изображения *(разрешение, resolution)* – размер прямоугольной или квадратной области растрового изображения представимый некоторым количеством точек в ширину и высоту. На мониторах и экранах это также число пикселей по горизонтали и по вертикали. Наиболее часто используются 1024×768, 1280×800, 1280×1024 (для 15″, 17″, 19″), если первые мониторы имели соотношение 4:3, то на сегодня они всё больше похожи на вытянутый прямоугольник с соотношением 16:9. Также часто можно видеть разрешения 1920×1080, 1280×720, 1333×768 хорошо подходящие под телевидение высокой чёткости HDTV – стандарты 1080i, 1080p и 720p и воспроизведение дисков Blue-ray формата.

Несмотря на короткую историю, назвать указанные разрешения высокими сложно, потому как сегодня запросто можно встретить дисплеи с разрешениями 2560×1536, 2560×1600, 2880×1800, 3840×2160, 4096×2160 и даже выше.

В классификации видеоразрешений появились новые UHD разрешения, рекомендованные для телевидения сверхвысокой чёткости (Ultra High Definition Television, UHDTV): 4K (2160p) – 3840×2160 и 8K (4320p) 7680×4320. Два цифровых формата с указанными разрешениями картинки были предложены японской телекомпанией NHK Science & Technical Research Laboratories и приняты Международным союзом электросвязи в августе 2012 года как Рекомендация ITU-R BT.2020 [53]. Следует отметить, что указанная выше телекомпания NHK реализовала на практике систему, работающую в

[53] http://www.itu.int/rec/R-REC-BT.2020-0-201208-I/en.

формате 7680×4320, а в 2013 году фирмой Sharp был выставлен на продажу телевизор с разрешением 8K. По аналогии с 4K и 8K появилось понятие 5K, обозначающее разрешение 5120×2160 (или 5120×2880).

Замечание. Увеличивать разрешение можно и больше, главное чтобы техника его поддерживала. На данный момент, 16K (15360×8640) видео может быть без труда воспроизведено на панели состоящей из нескольких мониторов при помощи технологии AMD Eyefinity или NVIDIA Surround. Действующий выставочный образец 100-дюймового 16K8K S-UHD (15360 × 8640) дисплейного модуля был продемонстрирован в августе 2018 года фирмой Innolux в рамках выставки Touch Taiwan. В 2019 году фирмой Sony на выставке NAB Show был представлен дисплей с аналогичным разрешением. Назвать его дисплеем сложно, так как его диагональ равнялась 20 м. Также в 2019 году появился стандарт DisplayPort 2.0, поддерживающий устройства с таким разрешением. Массовое использование устройств с таким большим разрешением стоит ожидать разве что через 5-10 лет.

Обычно каждый пиксель изображения нумеруется начиная с нуля слева направо и сверху вниз.

Если размер пикселя на экране дисплея по его геометрическим размерам довольно мал, настолько мал, что два соседних пикселя не различаются невооружённым глазом среднестатистического человека, указанный дисплей с подачи фирмы Apple называют «Retina display» обычно это в районе 200–350 ppi (pixels per inch, точек на дюйм).

Для записи растровой графической информации (как с потерями в качестве изображения, так и без) существует множество форматов, наиболее популярные среди них это jpg, bmp, tiff и др.

Некоторые растровые графические форматы содержат помимо информации о самом изображении некоторые дополнительные данные, которые часто называют метаданными. Форматы записей метаданных определяются следующими стандартами и спецификациями Exif, IPTC, XMP и др.

2.4.1.1. Стандарт EXIF хранения метаданных об изображении

EXIF (англ. Exchangeable Image File Format) – стандарт, позволяющий добавлять к изображениям и прочим медиафайлам дополнительную информацию (метаданные), комментирующую этот файл, описывающий условия и способы его получения, авторство, GPS/Глонасс – координаты и т. п. Получил широкое распространение в связи с появлением цифровых фотокамер. Информация, записанная в этом формате, может использоваться как пользователем, так и различными устройствами, например принтером. Стандарт EXIF является чрезвычайно гибким и допускает широкое развитие – как правило, фотоаппараты добавляют к файлу информацию, специфичную только для данной конкретной камеры. Правильно интерпретировать такую информацию могут только программы от изготовителя фотоаппарата. Разработчик формата – Japan Electronics and Information Technology Association (JEITA). EXIF является частью более широкого стандарта DCF.

Для работы с EXIF-данными в .jpg-файлах в ОС Linux используется консольная утилита exiv2. Например, посмотреть данные можно командой

```
$ exiv2 IMG_1420.JPG
Имя файла: IMG_1420.JPG   Размер файла: 1698808 Байт   Тип MIME: image/jpeg
Размер изображения: 2816 x 2112              Camera make     : XXXXXXX
Модель камеры: XXXXXXX DIGITAL XXXX XXX
Отметка времени снимка: 2007:08:04 10:43:09
Номер снимка: 103-1420
```

```
Время экспозиции: 1/100 s
Диафрагма: F2.8
Смещение экспозиции: 0 EV
Вспышка  : Нет, авто
Flash bias       : 0 EV
Фокусное расстояние: 5.8 mm
Расстояние до объекта: 243
Светочувствительность: 100
Режим экспозиции: Easy shooting (Auto)
Режим замера: Multi-segment
Режим макросъёмки: Выкл
Качество изображения: Fine
Разрешение Exif: 2816 x 2112
Баланс белого: Авто
Эскиз       : image/jpeg, 5632 Байт
Авторские права:
Комментарий Exif:
```

Замечание. Для ОС Windows существует бесплатная программа просмотра и удаления EXIF-информации файла – IrfanView (http://www.irfanview.com/).

Стереть все данные:

```
$ exiv2 rm IMG_1420.JPG
```

Изменить какой-нибудь параметр, например «*Комметарий Exif*»:

```
$ exiv2 -M"set Exif.Photo.UserComment charset=Ascii тест123" IMG_1420.JPG
```

2.4.1.2. Цветовая модель

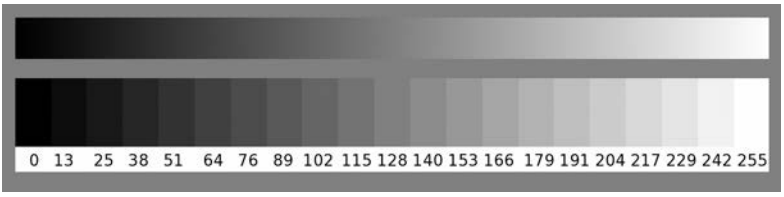
Хотя компьютеры и не подвержены всему тому, что присуще человеку при работе с цветом, кодирование цветов – это отдельный большой вопрос при работе с изображениями. Наука, занимающаяся этим вопросом, называется ***колометрией***, название которой произошло от лат. *color* – цвет и *греч.* μετρεω – измеряю.

Цветовая модель (color model, *цветовое пространство, система цветов*) – это правило, по которому может быть определён цвет точки.

Назначение цветовой модели должно быть ясно из определения и состоит в том, чтобы сделать возможным описание цветов некоторым стандартным, общепринятым образом. По существу, цветовая модель определяет некоторую систему координат и подпространство внутри этой системы, в котором каждый цвет представляется единственной точкой. [40]

Примеры моделей: RGB, CMY, CMYK, HSI (BHS, HSB), HLS, LAB, GreySCale (8 бит) и 1-битовая[54].

Самая простая двухцветная модель – ***1-битовая***. 0 1 В ней для описания цвета каждого пикселя (чёрного или белого) используется всего один бит.

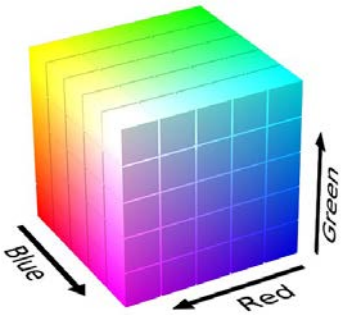


Более сложная ***модель GreyScale***. В ней цветность представлена градациями серого, а точнее числом от 0 до 255.

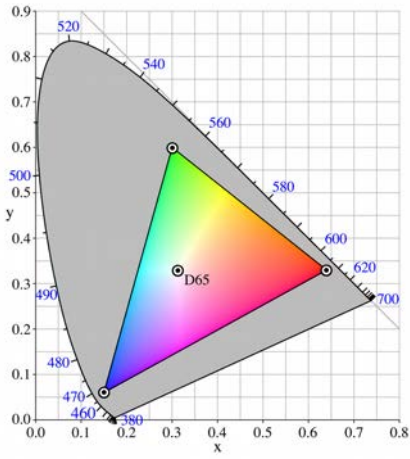
[54] R – Red, G – Green, B – Blue, C – Cyan, M – Magenta, Y – Yellow, K – blacK, H – Hue, S – Saturation, I – Intensity, B – Brightness, L – Lightness, в LAB: L – Luminance, A – цветовая гамма от зелёного до пурпурного, а B – цветовая гамма от голубого до жёлтого.

На рисунке серого цвета верхняя градиентная полоска отображает всю палитру данной цветовой схемы. Рамка вокруг сделана цветом № 128. Нижние прямоугольники окрашены интенсивностью в соответствии с подписями под ними.

Для представления «полноцветных» изображений используются несколько более сложных моделей. При первом приближении большинство цветов представимо как сумма яркости трёх основных цветов: красного, зелёного и синего. В основе модели лежит декартова система координат. Если интенсивность каждого цвета представить числом, то любой цвет можно разложить на три оси и представить как точку, принадлежащую кубу.



В памяти ЭВМ при этом будут храниться три указанных числа в двоичном виде. Так определяется наиболее известная цветовая ***RGB**-модель* (Red-Green-Blue). На каждое число отводится один байт (8 бит), таким образом, можно представить 2^{24} цвета, то есть примерно 16,7 млн цветов. Сейчас данная модель повсеместно используется в HTML-документах [55] и веб-дизайне, не говоря уже о графических форматах. Популярность её объясняется простотой понимания и охватом достаточного числа цветов (на картинке[56] слева они представлены треугольником), чтобы с помощью них получать близкие к реальности яркие и сочные картинки. Реально же видимая цветовая область несколько шире – она захватывает ещё и серую область за пределами треугольника. Естественно, вместо серого там находятся те или иные цвета, которые при печати отразить невозможно.



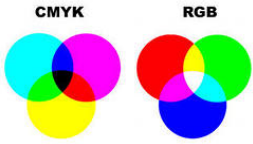
Белый цвет в этой модели представляется как #FFFFFF, чёрный – #000000, красный – #FF0000, синий – #0000FF. Жёлтый цвет является комбинацией красного и зелёного и потому представляется как #FFFF00. Знак «#» в некоторых случаях может опускаться.

Цветовая модель ***RGB*** (в варианте без оттенков, когда 1 – есть цвет, а 0 – нет) была стандартизирована в 1931 г. и впервые использована в цветном телевидении. Но, поскольку телевидение было аналоговым, то между 0 и 1 позднее появились оттенки. Модель ***RGB*** является ***аддитивной*** моделью, то есть цвет получается в результате сложения базовых цветов. Существуют и другие цветовые модели, которые для ряда задач оказываются более предпочтительными, чем ***RGB***-модель.

Например, для представления цвета в принтерах и полиграфии чаще используется ***субтрактивная*** **CMYK**-модель (Cyan-Magenta-Yellow-blacK), цвет в которой получается в результате вычитания базовых цветов из белого цвета.

Соответствие цветов в CMYK-модели:

- Белый (0,0,0,0)
- Чёрный (0,0,0,1)
- Голубой (1,0,0,0)
- Сиреневый (0,1,0,0)
- Жёлтый (0,0,1,0)



[55] HTML (от англ. *HyperText Markup Language* – «язык разметки гипертекста»).

[56] Цветовой охват RGB https://ru.wikipedia.org/wiki/RGB#/media/File:CIExy1931_sRGB_gamut_D65.png.

В цветовой модели ***HSV*** (Hue-Saturation-Value) цвет представляется через цвет, насыщенность и значение, а в модели ***HLS*** (Hue-Lightness-Saturation) – через оттенок, яркость и насыщенность.

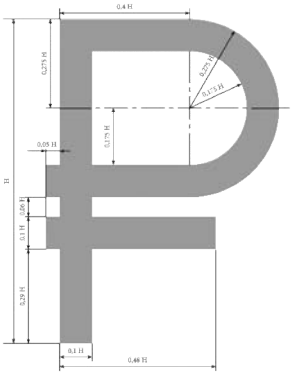
В цветовой модели **LAB** цвет состоит из: освещённости (Luminance) – это совокупность понятий яркость (lightness) и интенсивность (chrome) и гамм A и B. То есть двумя показателями в совокупности определяется цвет и одним показателем определяется его освещённость. LAB – это аппаратно-независимая цветовая модель, то есть она не зависит от способа передачи нам цвета. Она содержит в себе цвета как RGB так и CMYK, и grayscale, что позволяет ей с минимальными потерями конвертировать изображение из одной цветовой модели в другую.

Современные графические редакторы (GIMP[57], Photoshop и др.), как правило, могут работать с несколькими цветовыми моделями.

Независимо от цветовой модели, хранить не сжатое растровое изображение выходит накладно. Достаточно перемножить параметры разрешения на количество бит отводимых цветовой схемой для кодирования цвета и вы получите впечатляющие объёмы. Даже современных винчестеров большого объёма не на много хватит.

Несомненно, можно воспользоваться сторонними программами по сжатию, но эффективнее сжимать «на лету», то есть на этапе создания изображения или во время работы с ним, заложив такую возможность в графический формат. Поэтому постоянно появляются (придумываются) новые способы кодирования графической информации со сжатием. При этом сжатие бывает как с потерями (уровень потерь может варьироваться настройками формата, задаваемыми пользователем при сохранении), так и без. Теория сжатия изображений давно ушла вперёд, на сегодня используются не только словари, спектры, фракталы и вейвлет-преобразования, но и многое другое.

2.4.2. Векторная графика



Особняком идут графические форматы файлов (и работающие с ними программы), сохраняющие векторную графическую информацию. В противовес хранению растра, указанные форматы хранят изображения не в виде точек, представляющих собой сетку, а в виде некоторых легко масштабируемых примитивов (линии, дуги, прямоугольники, кривые Безье и пр.). По сути изображение состоит из кусочков ломаных и вычисляется в процессе отображения. Это основное отличие векторной графики от растровой. Например, в таком формате оптимально хранить масштабируемые шрифты (TrueType, OpenType[58] и др.). С 2016 года используется новый стандрат OpenType-SVG представления цветных шрифтов.

[57] GNU Image Manipulation Program – кросс-платформенное, доступное, в том числе и на русском языке, СПО для работы с изображениями. На сайте http://www.gimp.org/ доступны для скачивания версии под Linux, Windows, Mac OS X, FreeBSD, OpenBSD и другие ОС.

[58] OpenType – формат файла шрифтов, поддерживающий Unicode-кодировку, разработанный совместно Microsoft и Adobe для применения в различных операционных системах. OpenType обладает бóльшими, по сравнению с TrueType, возможностями допечатной подготовки и поддерживает бóльший набор символов при меньшем размере файла. Файлы шрифтов имеют расширение .ttf (шрифты, основанные на TrueType) или .otf (шрифты, основанные на PostScript).

Например такими цветными шрифтами в современных программах и на сайтах отображаются эмодзи и другие красивые надписи. (см. рис. 2.24)

Рисунок 2.24. Применение векторной графики в шрифтах (примеры изображения букв взяты с сайта http://colorfonts.wtf, слева направо: обычным (чёрно-белый) шрифтом, цветным векторным шрифтом, цветным векторно-растровым шрифтом)

Не все компьютерные шрифты являются векторными или смешанными (когда поверх векторной фигуры накладывается некоторое растровое изображение). Текстовые видеорежимы работы видеоадаптеров (видеокарт) даже у современных компьютеров или печать у матричных принтеров в текстовом режиме неизбежно используют растровые шрифты записанные (запрограммированные, зашитые) в памяти этих устройств.

Для рисования структурных блок схем, всевозможных планов, диаграмм и иных простых векторных рисунков удобно использовать программу Dia [59]. Также диаграммы можно рисовать с помощью средств, входящих в состав большинства офисных пакетов, например в LibreOffice.

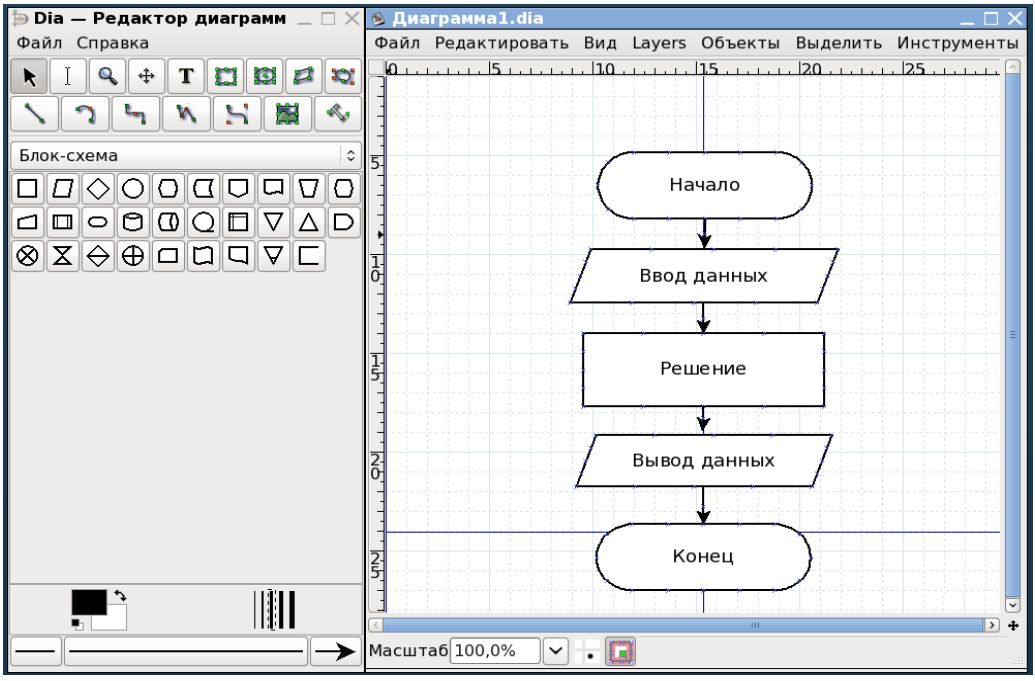


Рисунок 2.25. Окно программы Dia (свободное ПО)

[59] Свободное ПО Dia – программа рисования диаграмм. Доступна для скачивания для GNU/Linux, MacOS X, UNIX и Windows по адресу https://live.gnome.org/Dia. Коммерческий закрытый аналог – MS Visio.

В качестве примера программ, работающих с более сложными векторными изображениями можно привести, InkScape[60], CorelDraw, и др.

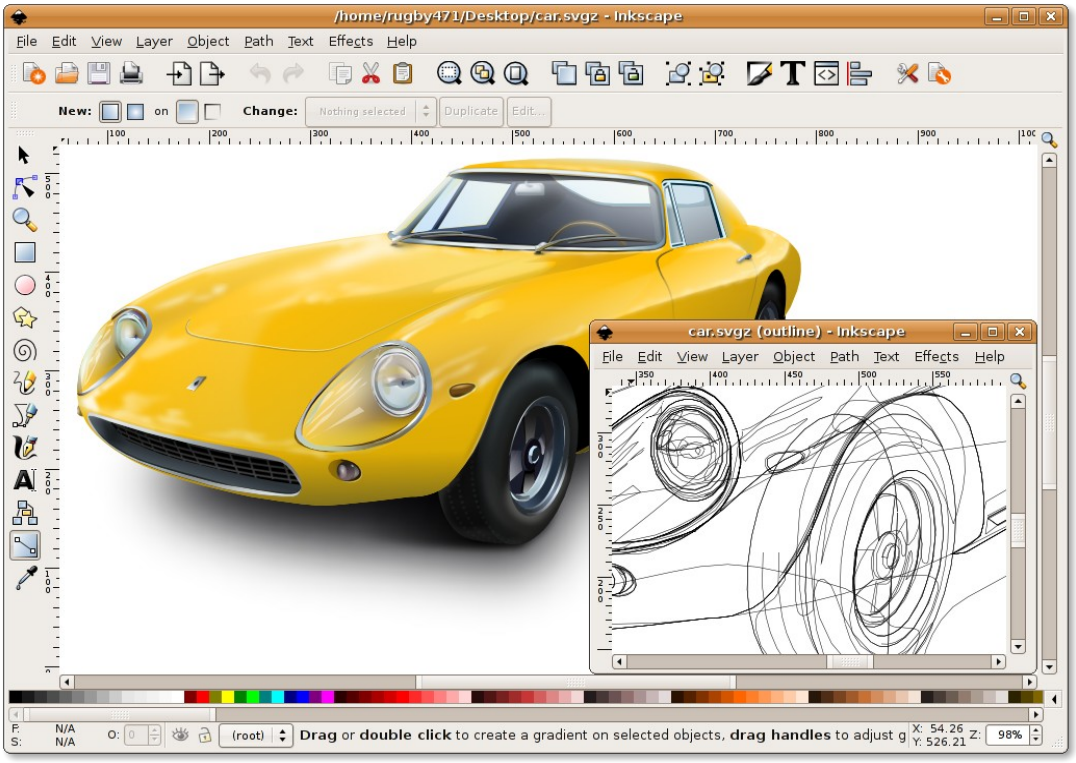


Рисунок 2.26. Окно редактора векторных изображений Inkscape

К сожалению, нарисованные плоские 2D-векторные рисунки невозможно просмотреть под другим углом или повращать в объёме.

2.4.3. Трёхмерная графика (3D-графика)

Логичным развитием векторной графики можно считать трёхмерную графику (в англ. 3D – 3 dimension, 3 измерения), где изображение состоит из определённого набора геометрических объектов (плоских и объёмных), которые являются трёхмерными, то есть описываются тремя координатами. Упрощённо эти координаты можно назвать *Длиной*, *Шириной* и *Высотой*. Четвёртое измерение – *Время* – присутствует только в динамических сценах или сценах, использующих анимацию.

Наиболее известными программами, работающими с трёхмерной графикой, пожалуй, являются Blender[61], 3DSmax, Maya, Lightwave, Cinema4D и др.

[60] Inkscape – открытый редактор векторной графики (СПО), функционально схожий с Illustrator, Freehand, CorelDraw или Xara X и использующий стандарт W3C под названием Scalable Vector Graphics (SVG). В программе поддерживаются такие возможности SVG, как фигуры, контуры, текст, маркеры, клоны, альфа-канал, трансформации, градиенты, текстуры и группировка. На сайте http://inkscape.org/ доступны для скачивания не только исходные тексты программы, но и сборки под Linux, Windows и Mac OS X.

[61] Свободное ПО, см. http://www.blender.org.

3D-графика – мощнейший инструмент для создания эффектов в современном художественном кино, рекламных роликах, компьютерных играх. 3D-графика широко применяется в архитектурных и дизайнерских проектах, в технических конструкторских бюро. С её помощью можно осуществить проектирование объектов и наглядное представление их взаимодействия.

В интернете существует множество бесплатных уроков (в том числе и видео на youtube) как начать работать с Blender, рисовать различные простые объекты, начиная с кубиков, шариков, стаканов, чашек, кружек и заканчивая самолётами и животными, в том числе на русском языке. На рис. 2.27, представлен результат построения модели плюшевого медведя с середине процесса и в конце.

Рисунок 2.27. Окно программы Blender, 3D модель плюшевого медведя

Замечание. С целью расширения кругозора читателей предлагаем самостоятельно поискать информацию в интернете по ключевым словам «3D-принтер» и «3D-сканер», в том числе и среди видеороликов. Возможно, что к моменту прочтения этих строк читателем кто-либо догадается соединить между собой описанные выше два устройства и к возможным вариантам поиска добавится ещё «3D-копир»[62].

Продолжение см. в параграфе 2.4.5.2. 3D-изображение, 3D-видео (стр. 139).

2.4.4. Фрактальная графика

Фрактальная графика, как и векторная, также является вычисляемой, но отличается от неё тем, что никакие объекты в памяти компьютера не хранятся. Изображение строится по уравнению («формуле изображения»), поэтому ничего, кроме неё, хранить не нужно. Занимательный факт: если в формуле поменять коэффициенты, то программа сгенерирует совершенно новую картину. Простейшим фрактальным объектом яв-

[62] См. также http://reprap.org/wiki/RepRap/ru.

ляется фрактальный треугольник. В процессе создания изображения строятся новые объекты, наследующие свойства своих родительских структур, согласно заданному математическим выражением алгоритму. Процесс наследования можно продолжать до бесконечности, меняя при этом различные параметры программы.

Попробуйте самостоятельно поискать в Сети информацию по некоторым построениям фрактальной графики, используя ключевые слова: «треугольник Серпинского», «салфетка Серпинского», «квадрат Серпинского», «ковёр Серпинского [63]», «снежинка Коха [64]», «кривая Коха» и др. Предполагаем, что вы найдёте для себя много интересного.

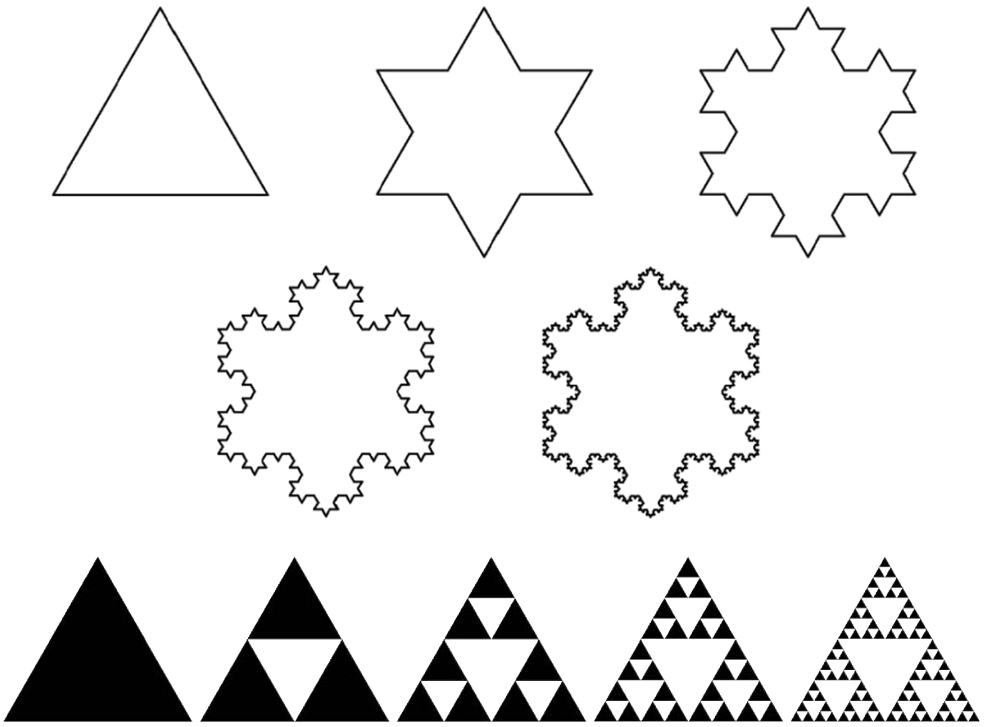


Рисунок 2.28. Построение «снежинки Коха» и «треугольника Серпинского»

Фрактал – объект, отдельные элементы которого наследуют свойства родительских структур. Поскольку более детальное описание элементов меньшего масштаба происходит по простому алгоритму (например, для рисунка выше словесно алгоритм можно выразить так: имеющийся треугольник уменьшаем и переворачиваем и т. д.), описать такой объект можно всего лишь несколькими математическими уравнениями.

Фракталы позволяют описывать целые классы изображений, для детального описания которых требуется относительно мало памяти. С другой стороны, фракталы слабо применимы к изображениям вне этих классов.

[63] Вацлав Франциск Серпинский (польск. Wacław Franciszek Sierpiński, 1882–1969) – выдающийся польский математик. Известен своими трудами по теории множеств, аксиоме выбора, континуум-гипотезе, теории чисел, теории функций, а также топологии.

[64] Нильс Фабиан Хельге фон Кох (швед. Niels Fabian Helge von Koch, 1870–1924) – шведский математик.

Существует множество различных программ фрактальной графики, как коммерческих, так и свободного ПО. Например, Apophysis [65].

2.4.5. Представление видеоинформации в ПК

Видеоинформация – наиболее сложный вид для хранения, обработки и воспроизведения. Впервые движущиеся изображения были сохранены на киноплёнке в виде большого количества отдельных кадров изображения, заснятых через небольшие промежутки времени (24 кадра в секунду). Позднее на ту же плёнку стала записываться и звуковая дорожка (в последующем несколько дорожек для многоканального звука). С этой точки зрения, сложности в хранении видеоинформации нет – выше нами были рассмотрены способы представления графической информации и аналогового сигнала (звука). Читатель может подумать: «записываем подряд 24 картинки – и видеоряд готов». Теоретически да, это так, но... Попробуйте самостоятельно подсчитать, сколько потребуется места для записи в таком формате хотя бы 15-минутного видеоролика без звука для самого маленького разрешения 352 × 288.

Без хитрых способов сжатия явно не обойтись.

Исторически проблема кодирования видеорядов данных в ЭВМ совпала с появлением телевидения с его аналоговой записью движущегося изображения на магнитные ленты (системы телевидения **PAL** и **SECAM** используют 25 кадров в секунду, система **NTSC** – 29,97 кадра в секунду).

С появлением компьютеров (а точнее увеличением их вычислительной мощности) широкое распространение получили цифровые методы записи и кодирования видеоинформации, которые постоянно совершенствуются. В настоящее время каждый может записать видео с использованием мобильных телефонов, цифровых фото- и видеокамер, компьютеров-очков и выполнить монтаж видеофильма на персональных компьютерах, производительности которых достаточно для перекодирования видео высокого разрешения объёмом в несколько гигабайт (но продолжительность кодирования может составлять несколько часов).

Компьютерные цифровые методы кодирования видео могут использовать частоту телевизионных стандартов **PAL/SECAM** или **NTSC**, так как видеозаписи многих цифровых форматов могут воспроизводиться как специальными компьютерными программами, так и бытовыми DVD-плеерами, а также путём подключения телевизора к компьютеру (для передачи видео и звука сейчас всё чаще используется порт HDMI[66]).

Качество видеоизображения в цифровых методах постоянно улучшается. Широкое распространение цифрового видео было связано с появление вначале CD-дисков, затем DVD, далее Blu-Ray дисков, на которых в основном и распространялись кинофильмы и ёмкостью которых ограничивались качественные возможности. В табл. 2.28 приведены характеристики некоторых видеоформатов.

Стандарты кодирования видео разрабатываются группой экспертов в области цифрового видео **MPEG** (Moving Picture Experts Group) – Международной организацией по стандартизации (ISO). Первый стандарт MPEG-1 был представлен в 1992 г., последние стандарты в этой области – MPEG-7 и MPEG-21.

[65] Apophysis (Апофиз) (от греч. αποφυσις, – отросток) – редактор фрактальной графики с открытым исходным кодом для визуализации в ОС Windows.

[66] High-Definition Multimedia Interface, разъём: http://pinouts.ru/Video/hdmi.shtml.

Алгоритмы кодирования видео очень сложны, их описания можно найти в специальной литературе или на сайте *http://www.mpeg.org*.

Все форматы сжатия семейства MPEG (MPEG-1, MPEG-2, MPEG-4, MPEG-7) используют высокую избыточность информации в изображениях, разделенных малым интервалом времени. Между двумя соседними кадрами обычно изменяется только малая часть сцены – например, происходит плавное смещение небольшого объекта на фоне фиксированного заднего плана. В этом случае полная информация о сцене сохраняется выборочно – только для **опорных** кадров. Для остальных кадров достаточно передавать разностную информацию: о положении объекта, направлении и величине его смещения, о новых элементах фона, открывающихся за объектом по мере его движения. Причём эти разности можно формировать не только по сравнению с предыдущими изображениями, но и с последующими (поскольку именно в них по мере движения объекта открывается ранее скрытая часть фона).

Таблица 2.28. Сравнение форматов записи видео на диски

Формат	Разрешение	Стандарт кодирования		Совместимость с DVD-плеером
		видео	аудио	
VCD	352×288 PAL 352×240 NTSC	MPEG-1	MPEG-1	всегда
SVCD	480×576 PAL 480×480 NTSC	MPEG-2	MPEG-1	иногда
DVD	720×576 PAL 720×480 NTSC	MPEG-2	MPEG-1, AC3	всегда
XVCD	720×576 PAL 720×480 NTSC	MPEG-1 или MPEG-2	MPEG-1	иногда
DivX	640×480	MPEG-4	MP3, WMA	иногда
HDTV 720p	1280×720	MPEG-4 H.264	MP3, WMA, AC3 или др.	BD-плеер
HDTV 1080i	1920×1080 (i – чересстрочная развёртка)	MPEG-4 H.264	MP3, WMA, AC3 или др.	BD-плеер
AVCHD 720p	1280×720 (p – прогрессивная развёртка)	MPEG-4 v.10 (AVC/H.264)	PCM (7.1) или AC3 (5.1)	нет
AVCHD 1080i	1920×1080	MPEG-4 v.10 (AVC/H.264)	PCM (7.1) или AC3 (5.1)	нет
4K UHDTV (2160p)	3840×2160	HEVC(H.265)	AAC	нет
8K UHDTV (4320p)	7680×4320	HEVC(H.265)	AAC	нет

Алгоритмы MPEG сжимают только опорные кадры – I-кадры (Intra frame – внутренний кадр). В промежутки между ними включаются кадры, содержащие только изменения между двумя соседними I-кадрами – P-кадры (Predicted frame – прогнозируемый кадр). Для того чтобы сократить потери информации между I-кадром и P-кадром, используются B-кадры (Bidirectional frame – двунаправленный кадр). В них содержится информация, которая берётся из предшествующего и последующего кадров.

MPEG-4 использует технологию фрактального сжатия изображений. Фрактальное (контурно-основанное) сжатие подразумевает выделение из изображения контуров и текстур объектов. Контуры представляются в виде сплайнов (полиномиальных функций) и кодируются опорными точками. Текстуры могут быть представлены в качестве коэффициентов пространственного частотного преобразования (например, дискретного косинусного или вейвлет-преобразования [67]).

Новые версии **MPEG-4** – **AVC/H.264** (Advanced Video Codec, называемый также **H.264**) – стандарт, предназначенный для значительного сжатия видеопотока при сохранении высокого качества и **AVCHD** (Advanced Video Codec High Definition – улучшенный видеокодек для видео высокого разрешения) – цифровой формат записи видеоданных в форматах 720p или 1080i и многоканального звука. Стандарт **AVCHD** был разработан совместно компаниями Sony и Panasonic в 2006 году. За основу был взят кодек AVC/H.264 (Multimedia Framework).

HEVC (High Efficiency Video Coding, ITU-T H.265) – более эффективный стандарт сжатия по сравнению с H.264/MPEG-4 AVC. Первая его версия появилась в 2013 году, вторая – принята в конце 2014 года, опубликована в начале 2015 года.

Форматы файлов Microsoft **AVI** и **MKV** – контейнеры, предназначенные для хранения видеоинформации, синхронизованной с аудиоинформацией. **AVI** может содержать в себе потоки 4 типов – Video, Audio, MIDI, Text. Причём видеопоток может быть только один, тогда как аудио – несколько.

Контейнер **MKV** (Matroska – матрёшка) разрабатывался с учётом современных тенденций и возможных тенденций будущего. Он универсален, так как построен на принципе EBML (Extensible Binary Meta Language, – как XML, но для двоичных данных). В **MKV** можно поместить любое количество аудио-, видеорядов, меню как на DVD, главы, субтитры, шрифты, постеры, тексты, комментарии, описания, фотоальбомы и прочее. Ограничений практически нет. Максимальная совместимость со всеми возможными требованиями к видеоконтейнеру на данный момент и на ближайшее будущее. Используется в настоящее время для переноса информации DVD- и Blu-Ray дисков в один файл *.mkv с сохранением меню, выбора языка воспроизведения, показа субтитров на выбранном языке, показа сцен-фрагментов основного фильма, рекламных роликов диска и прочего.

В октябре 1996 года группа **MPEG** приступила к разработке формата сжатия **MPEG-7**, призванного определить универсальные механизмы описания аудио- и видеоинформации. Этот формат получил название «Мультимедиа-интерфейс для описания содержимого» (Multimedia Content Description Interface). В отличие от предыдущих форматов сжатия семейства **MPEG**, **MPEG-7** описывает информацию, представленную в любой форме (в том числе в аналоговой) и не зависит от среды передачи данных.

Формат сжатия **MPEG-7** использует развитую многоуровневую структуру описаний аудио- и видеоинформации на основе языка этих описаний. Существуют различные типы информации, для которых разработаны схемы описания базовых структур: низкоуровневые аудио-визуальные характеристики, такие как цвет, текстура, движение, уровень звука и т. д.; высокоуровневые семантические объекты, события и абстрактные принципы; описание содержимого, навигации и доступа к аудиовизуальному мате-

[67] Вейвле́ты (от *англ. wavelet*), всплески – математические функции, позволяющие анализировать различные частотные компоненты данных.

риалу и т. д. Одной из отличительных особенностей **MPEG-7** является его способность к определению типа сжимаемой информации. Если это аудио- или видеофайл, то он сначала сжимается с помощью алгоритмов **MPEG-1, MPEG-2, MPEG-4**, а затем описывается при помощи **MPEG-7**.

Разработка формата **MPEG-21** – долговременный проект, который называется «Система мультимедийных средств» (Multimedia Framework). Над разработкой этого формата эксперты начали работать в 2000 г. Задача разработки **MPEG-21** может быть сформулирована следующим образом: определение технологии, необходимой для поддержки пользователей при обмене, доступе, продаже и других манипуляциях цифровыми объектами. При этом предполагается обеспечить максимальную эффективность и прозрачность этих операций.

Форматы файлов Microsoft **AVI** и **MKV** – контейнеры, предназначенные для хранения видеоинформации, синхронизованной с аудиоинформацией. **AVI** может содержать в себе потоки 4 типов – Video, Audio, MIDI, Text. Причём видеопоток может быть только один, тогда как аудио – несколько.

Для просмотра видеосюжетов (в том числе и видеовещания по Сети) на компьютере обычно используются какая-либо встроенная программа и набор кодеков. Рекомендуем читателям кросс-платформенную программу VLC Media Player[68], доступную для ОС Windows, Linux, Mac OS X и в исходных кодах, а также набор кодеков K-Lite Codec Pack[69] для ОС Windows и Perian[70] для Mac OS X. Установка[71] VLC Media Player в ОС Linux обычно автоматически подгружает и устанавливает из репозитария пакеты со всеми необходимыми кодеками и зависимостями (Подробнее см. раздел *5.6.5 «Программы воспроизведения DVD-фильмов и видеофайлов, кодеки»*).

2.4.5.1. Цифровое вещание

По количеству в мире сегодня доминирует цифровое вещание телевизионных каналов (англ. digital video broadcasting, DVB, версии T1, T2). Существуют различные форматы цифрового вещания. В конечном итоге Российская Федерация отказалась от централизованного вещания телевизионного изображения в аналоговом формате и перешла на стандарт вещания DVB-T2 с 14 октября 2019 года. В этот день российский космонавт Александр Скворцов, находящийся на Международной космической станции, вышел на прямую связь с центром управления полётами (г. Королёв Московской обл.) и на фоне флага с изображением символа цифрового ТВ – «цифровой бабочки» символическим нажатием кнопки дал сигнал к отключению в 21 российском регионе аналогового телевещания[72].

Очевидно, что передаваемый цифровой сигнал без труда может быть принят с помощью цифрового приёмника, подключенного к компьютеру, и сохранён или отображён. (Подробнее см. *4.12.6. Платы видеозахвата, TV- и FM-приёмники*, стр. 333.) Форматы кодирования видеосигналов в рамках данного учебника не рассматриваются.

[68] http://www.videolan.org.
[69] http://www.codecguide.com.
[70] http://www.perian.org/.
[71] Например, «yum install vlc» или «aptitude install vlc».
[72] https://ria.ru/20191014/1559752630.html

2.4.5.2. 3D-изображение, 3D-видео

В продолжение параграфа 2.4.3. Трёхмерная графика (3D-графика) (стр. 132) отметим, что построение объёмных изображений объектов в статике или в движении построено на особенностях человеческого зрения. Люди от природы наделены бинокулярным зрением (видят мир двумя глазами), а их мозг, обрабатывает полученные от глаз сигналы, «пересчитывает» и создаёт в нашем воображении изображение. Процесс восприятия происходит бессознательно, мы лишь наслаждаемся итоговым результатом. В этом случае, может показаться, что для хранения 3D изображений в памяти ПК всего-то и надо увеличить объём памяти вдвое, либо поставить два компьютера рядом, либо один поделить на 2 части. Например, уже сейчас можно недорого купить (или изготовить самому) небольшие картонные или пластиковые приспособления с двумя линзами для размещения в них телефона.

Рисунок 2.29. Держатель телефона для создания виртуальной реальности

Если программное обеспечение телефона снабдить функциями опроса датчиков определения положения устройства в пространстве и соответствующего изменения показываемого изображения, то запросто можно добиться эффекта присутствия. При повороте головы у человека будет создаваться ощущение наличия вокруг него виртуальной реальности. Многие игры уже сейчас используют эту возможность.

Привязываться к телефону (или другому мобильному устройству) не обязательно, в компьютерных магазинах давно продаются специальные шлемы виртуальной реальности, подключаемые к компьютеру.

В теории вроде всё красиво, но на практике дело обстоит немного сложнее. Для 3D-съёмки надо иметь две камеры, работающие почти синхронно. Для удобства хранения одного файла изображений вместо двух необходим соответствующий файл-контейнер и, соответственно, его поддержка в программном обеспечении.

Также, отображение 3D-картинки на привычных экранах встречает значительно большие трудности. Для получения разных картинок в правом и левом глазах человека приходится или использовать эффекты поляризации и очки с поляризацией линз, либо использовать очки с цветовыми светофильтрами, либо повышать число кадров в изображении и также использовать кратковременно перекрывающие световой поток фильтры. Судя по киноиндустрии некоторые достижения в этом направлении уже достигнуты.

Логичным продолжением технологии 3D видео является пленоптика, речь о которой пойдёт в следующем параграфе.

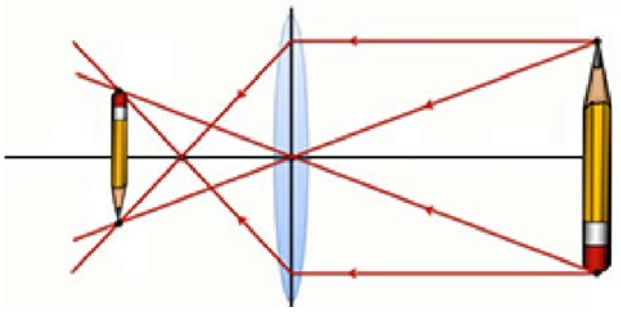
2.4.5.3. Пленоптика (вычисляемое видео)

Наверняка многие читатели заметили тенденцию последних лет от производителей мобильных телефонов – снабжать их бóльшим количеством камер и увеличивать «сырое» разрешение их матриц. В интернете можно найти много информации на эту тему и даже юмор (см. картинку справа, чехол телефона с кадром из импортного к/ф «Коммандос», 1985 г.), но это ещё не пленоптика, а лишь некоторые её отголоски. Отголоски потому, что первопричина этого – разные фокусные расстояния камер и селективная их работа при съёмках «вблизь» и «вдаль», но, когда с помощью программного обеспечения оптические сенсоры работают одновременно и на основе полученных данных высчитывается общее изображение, тогда это шаг в сторону вычисляемых изображений.

Замечание. Благодаря преподавателям (и научным работникам) лаборатории компьютерной графики ВМК МГУ им. М.В.Ломоносова подробнее о данном вопросе можно прочитать в материале «Вычисляемое видео в 755 мегапикселей: пленоптика вчера, сегодня и завтра» https://habr.com/ru/post/440652/. Мы же приведём некоторые выдержки из этого материала.

История вопроса уходит корнями к работам А.А.Гершуна о световом поле[73].

В традиционных видео- и фото- системах сенсор (обычно это ПЗС-матрица или плёнка) располагается в фокальной плоскости линзовой системы объектива, фиксируя перевёрнутый свет от реального изображения и формируя двухмерную картинку.



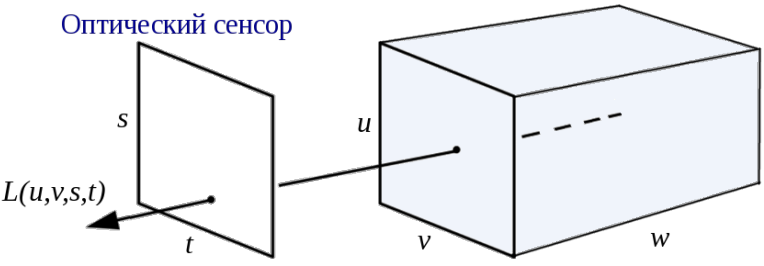
Картинка выше должна быть знакома читателю из раздела оптики школьного курса физики.

Если объект съёмки расположен не в фокусе линзы, то получаемое изображение получается нечётким и мы обычно констатируем, что «ничего не видно».

Гершун Андрей Александрович (1903–1952) – советский учёный в области фотометрии и светотехники, основатель научной школы по гидрооптике (Государственный оптический институт им. С.И. Вавилова). Лауреат двух Сталинских премий (1942, 1949).

Это происходит потому, что оптическая система настроена и работает так, чтобы фиксировать «плоский» (2D) срез объекта по фокальной плоскости.

На самом же деле снимаемая сцена не плоская. Источниками фиксируемого камерой света являются точки некоторого пространственного объёма, допустим, некоторого прямоугольного параллелепипеда с размерами $u \times v \times w$. В этом случае прилетающие в объектив камеры (а в последствии на оптический сенсор) световые лучи от разных точек описанного выше объёма будут падать под разными углами. Ситуация чем-то напоминает голограммы.



Фиксировать углы, то есть работать в полярных координатах, и создавать учитывающие угол оптические сенсоры не очень удобно, поэтому на практике, где область пространства откуда «прилетает» свет ограничена, интенсивность лучей оценивают функцией зависящей от 4-х параметров (для видеосцен – от пяти, где пятый параметр – время), называемой световым полем. Данное понятие является ключевым для плеоптической съёмки и под ним

[73] *А.А. Гершун* Световое поле, 1936. http://books.e-heritage.ru/book/10075261, http://nasledie.enip.ras.ru/ras/view/publication/general.html?id=47282739.

понимается то, что в каждой точке оптического 2D-сенсора фиксируется не цвет пикселя (яркость), а двумерная матрица пикселей, «засветившая» этот пиксель, тем самым превращая старый двумерный кадр в новый четырёхмерный (обычно пока с очень небольшим разрешением по s и t). В целом, в данной модели, глубина *w* оказывается несущественна и в формулах не учитывается (важен ближайший к сенсору *u*×*v*-срез в снимаемом объёме). С добавлением основной линзы объектива ситуация кардинально не меняется.

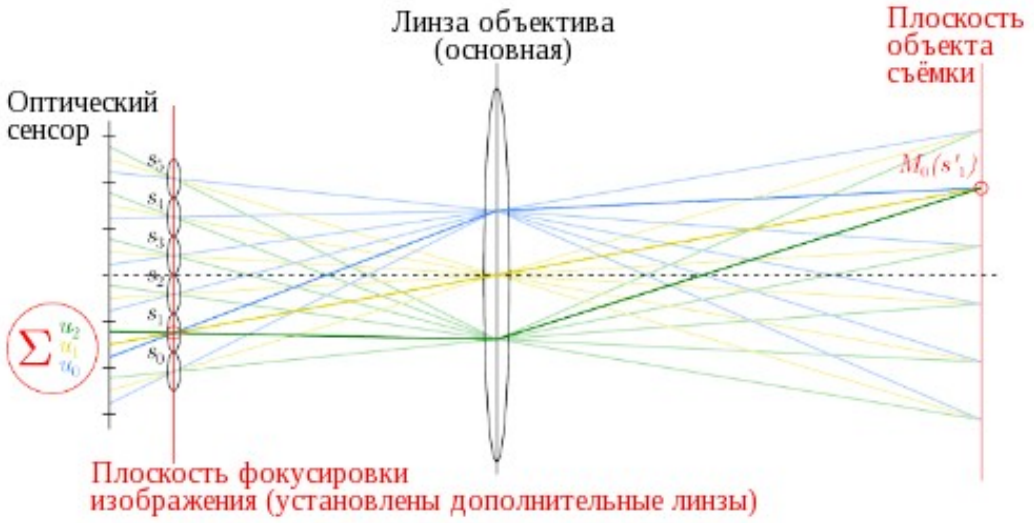


Рисунок 2.30. Прохождение оптических лучей через систему линз

А вот, смещение сенсора от фокальной плоскости объектива немного назад и добавление в неё множества мелких линз кардинально меняют фиксируемую (снимаемую) картинку. Изображение за ними оказывается примерно следующим.

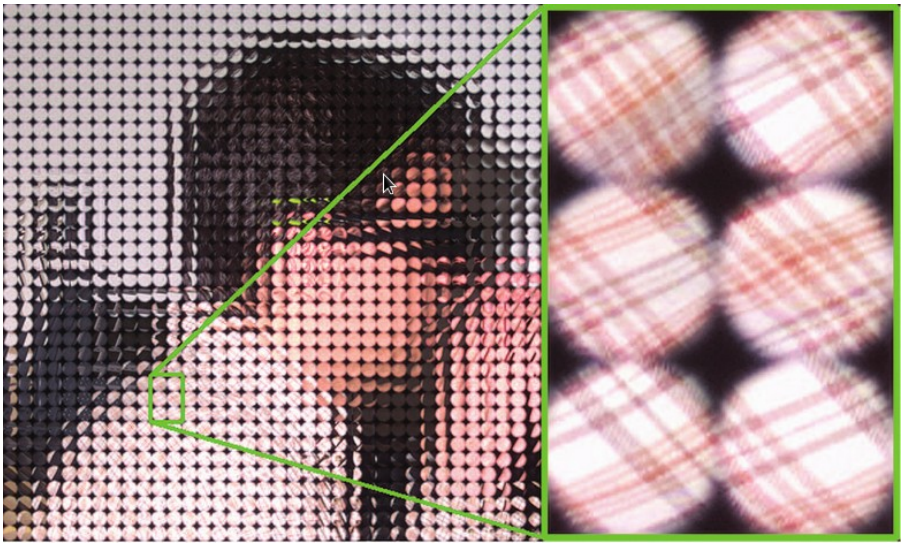


Рисунок 2.31. Изображение после сетки мелких линз (Взято из: https://www.researchgate.net/publication/224207951_Using_Focused_Plenoptic_Cameras_for_Rich_Image_Capture)

Замечание. Изображение 2.31 чем-то напоминает фасеточное зрение присущее некоторым насекомым, ракообразным и др. беспозвоночных, фасеточные глаза которых образованы особыми структурными единицами «омматидиями», роговичная линза которых имеет вид выпуклого шестигранника – фасетки[74].

Подобная оптическая система позволяет на сенсоре фиксировать информацию из плоскостей съёмки, отличных от фокальной. Причём, эта информация находит своё отображение на нескольких различных пикселях оптического сенсора.

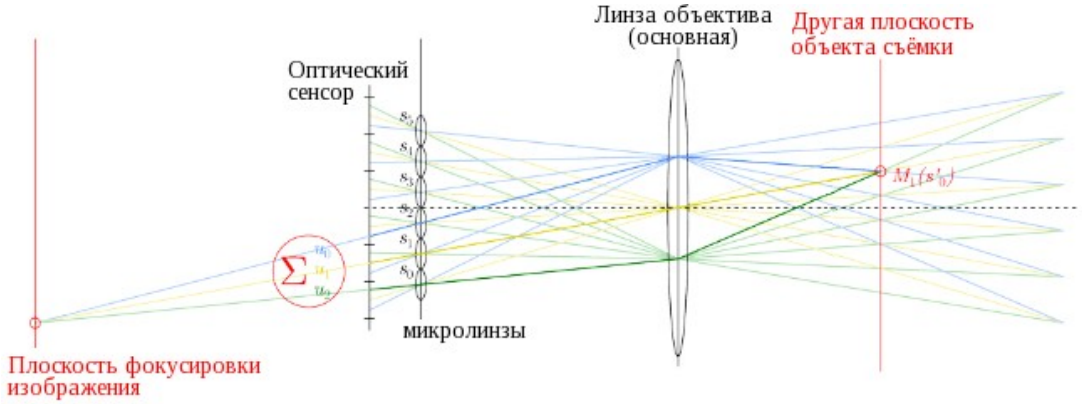


Рисунок 2.32. Прохождение оптических лучей от другой плоскости

В результате, если матрица оптического сенсора будет обладать достаточно большим разрешением, то появляется возможность на основе зафиксированных им данных математически вычислить новое изображение. Поскольку чудес не бывает (новой информации об изображении взяться неоткуда), получаемое с помощью пересчёта изображение оказывается меньшего размера, нежели сырое разрешение оптического сенсора. (В ссылке приведённой в начале параграфа описывается сенсор в 755 мегапикселей!)

Самая известная возможность такого пересчёта (возможность пленоптических камер) – изменение фокусировки уже после того, как сделан кадр! При использовании аналоговых технологий такое и другие эффекты не достижимы в принципе, разве что – голограммы.

В определённых пределах возможен пересчёт точки съёмки, освещения, разрешения, окружения объекта съёмки.

Время экспозиции и форма диафрагмы для вычисляемого изображения тоже могут варьироваться, как если бы съёмка велась на реальной фотокамере с изменением этих параметров.

Помимо этого возможны создание более качественного эффекта «замена фона» (без использования зелёного экрана позади объекта съёмки – хромакей), выделение стерео пар изображений для 3D-снимков и др.

[74] От франц. facette – грань.

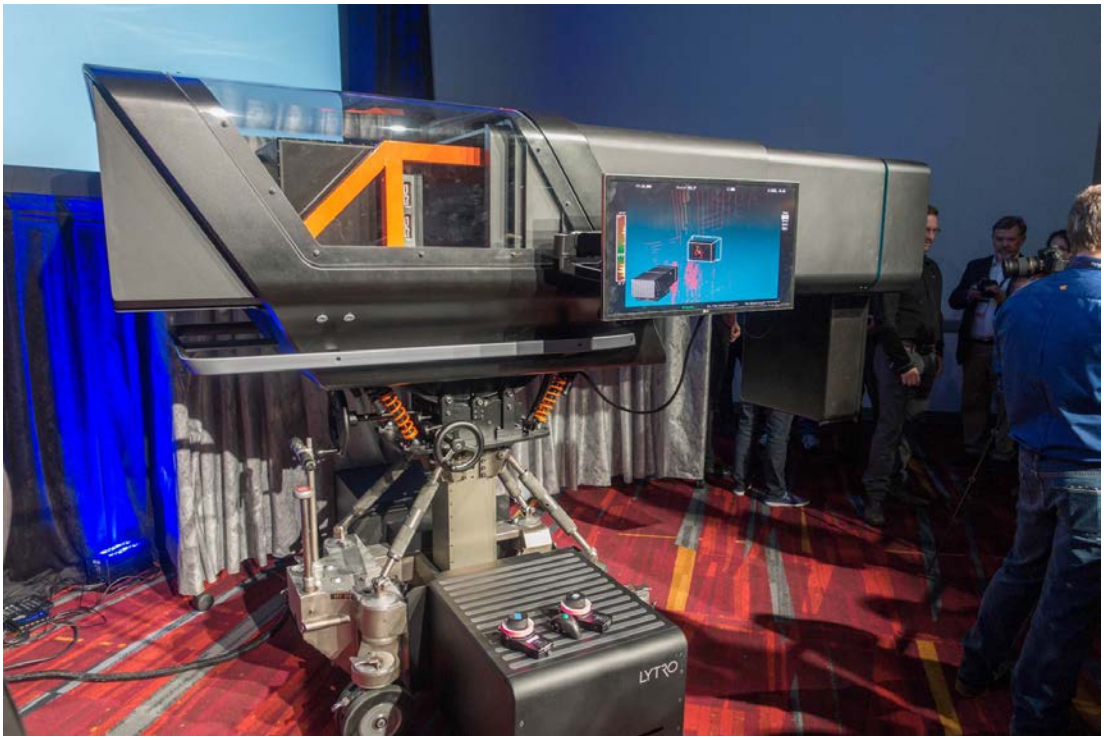


Рисунок 2.33. Плеоптическая камера компании Lytro, 2018 г.

К сожалению, более подробное рассмотрение вопросов в отношении кодирования изображений (графики, видео) выходит за рамки учебника.

2.5. Некоторые виды кодов и кодирования

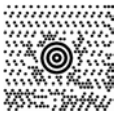
Предыдущие разделы 2.1–2.4 говорят читателям о том, что информация об окружающей нас действительности, событиях, объектах и т.п., в рамках некоторой модели, являющейся по сути отображением картины реального мира, с некоторой точностью может быть представлена в виде чисел, текстов, звуков, аналоговых сигналов, изображений и последовательностей изображений (видео) которые в рамках другой модели могут быть преобразованы в двоичные числа с целью унификации процессов хранения и обработки этих сведений в ЭВМ.

Полученные таким образом цифровые данные могут быть подвергнуты различным дополнительным преобразованиям обычно для последующего упрощения процессов их использования, либо получения иных необходимых выгод (сокращение размеров хранения, уменьшение времени затрачиваемого на обработку и т.п.).

Сведения о подобных преобразованиях, являющихся актуальными по мнению авторов, и связанные с этим вопросы представлены в последующих параграфах.

2.5.1. Штрихкоды






В последнее время мы всё чаще видим в рекламе, на товарах, а также их компонентах, всевозможные одномерные[75] и двумерные штрихкоды.

Штрихкод[76] – это попытка, в удобном для чтения ЭВМ виде с помощью чёрных и белых чёрточек или точек, а также их расположения, записать некоторую текстовую информацию. Чёрный и белый цвета выбраны из-за того, что они обладают максимальной контрастностью по отношению друг к другу и тем самым обеспечивают минимизацию ошибок считывания. Использование других цветов возможно, но часто не оправдано с технической или иной точек зрения.

Идеи кодирования, а затем «штрихкодирования» или «радиокодирования» товаров, документов и иных предметов с целью упрощения их учёта существуют давно. Так, в прошлом веке, под влиянием зарубежных коммерческих компаний была создана «GS1[77]» – международная организация, ведающая вопросами стандартизации учёта и штрихового кодирования логистических единиц. Естественно, что за пользование услугами этой организации (например, чтобы не было двух разных товаров с одним штрихкодом) косвенно приходится платить конечным потребителям, деньги которых косвенным образом уходят в зарубежные и международные компании. Естественно с точки зрения нашей экономики предпочтительно чтобы штаб-квартиры международных стандартизирующих организаций располагались на территории РФ. Также можно не использовать технологии и не платить за это. Согласно законодательству РФ нанесение штрихового кода на продукцию не является обязательным, однако, что интересно, впервые товар народного потребления обзавёлся штрихкодом на своей упаковке в 1974 году на добровольной основе, а сегодня товары практически невозможно продать через торговые сети, если они не обладают штрихкодом. Как ни крути, прогресс остановить невозможно, поэтому рассмотрим технические стороны вопроса.

Количество различных «одномерных» (1D) штрихкодов велико: Code-11, Code-2of5 Inerleaved, Code-39, Code-39 Full ASCII, Code-128, GS1-128 (UCC/EAN-128), Фармакоды и другие. Длина полосы кода состоящей из чёрных и белых чёрточек напрямую влияет на максимальный объём кодируемой информации. Естественно, что чрезмерно длинные штрихкоды неудобны. Для кодирования большего объёма информации были придуманы несколько-полосные или двумерные (2D) штрихкоды: (Micro) QR Code, Data Matrix, Aztec, Codablock-F, Maxicode, (Micro) PDF417, Han Xin и другие.

Какие-то коды кодируют только цифры, иные имеют расширенный набор символов, но ни один не дружит с кириллицей напрямую, обычно для кодирования русских букв используют кодировку Unicode или её формат UTF-8.

Замечание. Наше государство активно внедряет информационные технологии в повседневную жизнь: «*Штриховой код, как технология автоматической идентификации и сбора данных, широко используется при осуществлении платежей физическими лицами. Использова-*

[75] Пример штрихкода EAN-13 см. на тыльной стороне обложки учебника.
[76] Хороший on-line генератор штрихкодов доступен по адресу http://barcode.tec-it.com.
[77] Сайт организации http://www.gs1.org, до 2005 EAN – International-Uniform Code Council (EAN-UCC).

ние символов штрихового кода на платёжном документе позволяет осуществить автоматизированный ввод реквизитов платежа и этим снизить трудоёмкость проведения операции приёма платежа, уменьшить количество ошибок, допускаемых клиентами и сотрудниками организаций, принимающих платежи, и сократить время оформления платежа. Для задания единых правил использования штрихового кода как поставщиками услуг при выставлении счетов (печати платёжных документов), так и принимающими платежи организациями возникла необходимость разработки общего стандарта.[42]», – национальный стандарт Российской Федерации ГОСТ Р 56042-2014 *«Двумерные символы штрихового кода для осуществления платежей физических лиц»*.

Прошло совсем немного лет с момента принятия стандарта, а он уже повсеместно активно используется. Наверняка вы вспомните что недавно видели платёжки со штрихкодами.

2.5.1.1. QR-коды

Из группы 2D штрихкодов благодаря рекламе наиболее узнаваемы QR-коды, разработанные и представленные японской компанией «Denso-Wave» [78] в 1994 году. Несмотря на то что они были придуманы почти как два десятилетия назад, широкое распространение они получили лишь сейчас, по мере заполнения рынка «гаджетов» мобильными устройствами со встроенными фотокамерами (а то и двумя). С первого взгляда, это небольшие прямоугольные квадраты с чёрно-белыми точками внутри наподобие приведённого ниже.

Слева закодировано слово «Информатика».

Картинка создана с помощью генератора, доступного на сайте http://www.qrcoder.ru/. Ещё один веб-генератор/распознаватель QR-кодов доступен по адресу на http://foxtools.ru/QR. Также QR-коды умеет создавать LibreOffice версии 6 и выше.

По сути, это матричный код (или двумерный штрихкод). Обычно более мелкие элементы (составляющие) итогового рисунка кода – это маленькие чёрные точки квадратной формы.

Поскольку QR-код обладает достаточной избыточностью, а считывание его производится лазером на основе яркости (без учёта цвета), стало возможным создавать «дизайнерские» решения на основе QR-кодов без потери их функциональности: с использованием цветов отличных от чёрного и белого, треугольной, круглой, сердцевидной и т. п. формой «точки» (наименьшего элемента изображения). Дополнительно на конечное изображение кода может накладываться тот или иной фильтр (например размытие), а вместо части изображения может использоваться сторонний текст или рисунок.

Название QR-код произошло от английского *quick response* (быстрый отклик).

Основное достоинство QR-кода – это лёгкое распознавание сканирующим оборудованием (в том числе и фотокамерой мобильного телефона, либо камерой компьютера встроенного в оправу очков), что даёт возможность использования в торговле, производстве, логистике. QR-коды больше всего распространены в Японии, стране, где

[78] http://www.denso-wave.com/en/.

штрихкоды пользовались такой большой популярностью, что объём информации, зашифрованной в коде, вскоре перестал устраивать индустрию. Японцы начали экспериментировать с новыми способами кодирования небольших объёмов информации в графической картинке, так и был придуман QR-код.

Рисунок 2.34. Пример размещения QR-кода на информационной табличке с описанием достопримечательности

Максимальное количество символов, помещаемых в один QR-код, варьируется от способа кодирования и составляет: для цифр – 7189 знаков; для цифр и букв (включая кириллицу) – 4296 символов; для двоичного кода – 2953 байт; для иероглифов – 1817 штук. Первоначально QR-коды были широко распространены в странах Азии (особенно в Японии), но, довольно быстро появились у нас и в других странах.

Наибольшее признание они получили среди пользователей мобильной связи – установив программу-распознаватель, абонент может моментально заносить в свой телефон текстовую информацию, добавлять контакты в адресную книгу, переходить по веб-ссылкам, отправлять SMS-сообщения и т. д. С появлением технологии дополненной реальности QR-коды вносят в жизнь новые интересные моменты, например, вы только посмотрели на ценник с QR-кодом в магазине через очки с камерой и дисплеями, как сразу на экране, сбоку на полупрозрачной панели появилась подробная информация о товаре и список ближайших магазинов, где данный товар можно купить дешевле. С учётом того, какими темпами совершенствуются технологии распознавания изображений, QR коды в данном примере могут и не потребоваться.

Что же до использования QR-кодов в промышленности – у них есть один большой недостаток – для успешного считывания метка должна быть разборчивой и находиться в зоне прямой видимости. Данное условие не всегда достижимо, например при движении объекта с кодом, поэтому в ряде отраслей на пятки QR-кодированию очень активно наступает технология NFC [79] и радиочастотная идентификация на базе меток RFID [80], которые уже позволяют осуществлять двухсторонний обмен. Например, в московской области радиометки активно используются для удалённого отслеживания перемещения мусорных контейнеров.

Рисунок 2.35. Радиометка на контейнере

Вы без труда самостоятельно вспомните и другие места применения радиометок, например выходя из какого-нибудь большого супермаркета.

Кроме кодирования товаров и документов в последнее время появился «социальный запрос» получивший ответ в виде de facto стандартизации кодирования таких не материальных вещей как «телефонный номер», «SMS с готовым текстом», «ссылка для

[79] NFC (англ. *Near Field Communication* – «*коммуникация на небольшом расстоянии*») – технология беспроводной высокочастотной связи малого радиуса действия, которая даёт возможность обмена данными между устройствами, находящимися на расстоянии около 10 см. Эта технология – простое расширение стандарта бесконтактных карт (ISO 14443), которая объединяет интерфейс смарткарты и считывателя в единое устройство.

[80] RFID (англ. *Radio Frequency IDentification* – радиочастотная идентификация) – способ автоматической идентификации объектов, в котором посредством радиосигналов считываются или записываются данные, хранящиеся в так называемых транспондерах, или RFID-метках.

like'ания», «адрес профиля социальной сетей», «адрес твита» и тому подобных. Кодирование указанной информации не отличается от кодирования текстов, за тем лишь исключением, что мобильные приложения, производящие распознавание подобных кодовых изображений, по результатам распознавания могут выполнять те или иные действия, как то набор распознанного номера, набор текстового сообщения или отправка URL-строки браузеру или иному приложению.

Леса у нас много, но лучше его беречь, в ряде случаев использование подобных кодов может осуществляться без их печати на бумажные или иные физические носители. Например, отображать коды можно как на стационарный экран монитора для быстрого считывания информации в мобильные устройства, так и на экран мобильного телефона, например при просмотре письма-подтверждения со штрихкодом, для представления последнего автомату для печати бумажных билетов, либо для прохода через турникет или контролёра. Это могут быть как электронные билеты в кино, так и посадочные талоны на поезд или самолёт.

Распространённость и простота реализации штрихового кодирования сделали его очень популярным. Хочешь на экране отображай, хочешь на бумаге, упаковке, плёнке или наклейках печатай, однако, у них всех есть недостаток: вмещают мало информации. Как следствие, требуют подключения к локальной базе данных, для поиска и получения данных по ключу или доступ в интернет, что по сути есть так же самая база данных.

Например, наклейки ЕГАИС (Единой государственной автоматизированной информационной системы, предназначенной для государственного контроля над объёмом производства и оборота этилового спирта, алкогольной и спиртосодержащей продукции) выглядят красиво, имеют несколько степеней защиты, но бесполезны, если у вас нет оборудования для их проверки, а у этого оборудования нет связи с сервером. Также с 1 июля 2017 года в соответствии с Федеральным законом от 03 июля 2016 № 290-ФЗ действует новый порядок применения контрольно-кассовой техники (ККТ). Для проверки чеков, на последних печатаются прямым текстом и/или в виде QR-кодов ссылки на страницы сайта ФНС России, однако, если интернета нет, то проверить чек по этой технологии невозможно.

Размещение полностью автономной информации в штрихкодах затруднено, поскольку в 1-2 килобайта можно уместить разве что небольшой отрывок текста (часто опять ту же же ссылку), а вот небольшое изображение размером чуть побольше иконки или любой офисный документ уже не влезут даже в самый большой код. Исключением будут разве что платёжки, где в формате XML закодированы реквизиты получателя денег для более быстрого их считывания. Но, оцените каков размер этого штрихкода!

Перспективным может быть увеличение объёмов кодируемой информации, сделать это можно банально за счёт увеличения размеров матрицы, добавлением цветов или оттенков. Дополнительно расширить ёмкость можно отказавшись от квадратной формы ячеек. Хороший пример – Jabcode[81]. Явно не универсальное, но решение проблемы.

Замечание 1. Возьмите разрешения экранов и камер современных мобильных телефонов и вы поймёте, что эта технология явно им подходит. На одном устройстве можно последовательно отображать коды, а другое может их считывать.[82] При этом образуется односторонний канал связи который, в отличие от радиоканала, не создаёт помех (например, может использо-

[81] JAB Code (Just Another Bar Code*) https://github.com/jabcode/jabcode*.
[82] Передача файлов по воздуху через камеру смартфона *https://habr.com/ru/company/vdsina/blog/534412/*.

ваться на борту самолёта) и к тому же, данные переданные таким образом довольно сложно перехватить. Например, с помощью txqr (https://github.com/divan/txqr) можно передавать небольшие файлы с одного мобильного устройства на другое со скоростью в десятки килобайт в секунду. Конечно можно и кабелем соединить два устройства, но тут есть много но, как минимум он должен быть под рукой.

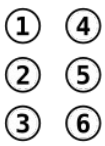
Пример применения: некоторые wi-fi web-камеры настраиваются исключительно с помощью QR-кодов. Последние генерируются специальным мобильным приложением, а экран планшета или телефона с отображённым на нём кодом (в котором, как вы понимаете, закодированы название, пароль сети wifi и наверняка что-то ещё) помещается на удалении 20-30 см от объектива web-камеры. Другой пример описывает выдержка из описания мессенджера: «*Чтобы подключить ваш веб-браузер к клиенту ..., просто откройте https://web.whatsapp.com. Вы увидите QR-код – отсканируйте его с помощью WhatsApp, и вы готовы к работе.*»

Замечание 2. К сожалению, в повсеместном и бездумном использовании 2D-кодов есть и негативный социальный аспект. Так, реклама «с квадратиками» делит общество на тех, кто имеют электронные устройства и могут прочитать то, что написано, и тех, кто этого сделать не могут. Даже при наличии современных алгоритмов и аппаратных возможностей для распознавания изображений использование штрихкодов скорее есть показатель проигранной битвы между человеком и компьютером в пользу последнего, поскольку, вы, читатель, наверняка ещё не научились читать без помощи техники сообщения со штрих- и QR-кодов. Не очень хочется, чтобы буквари наших внуков или правнуков вместо привычного алфавита с картинками содержали информацию о различных методиках штрихового кодирования.

2.5.2. Шрифт Брайля

В 1824 году французом Луи Брайлем (фр. Louis Braille, 1809-1852) был разработан рельефно-точечный тактильный шрифт, предназначенный для письма и чтения незрячими и плохо видящими людьми. Будучи ребёнком Луи поранился в мастерской отца шорным ножом; после из-за начавшегося воспаления глаза мальчик потерял зрение.

Предполагают, что свой шрифт Брайл создал в возрасте 15 лет как альтернативу рельефно-линейному шрифту Валентина Гаюи, вдохновившись простотой «ночного шрифта» капитана артиллерии Шарля Барбье, который в то время использовался военными для записи донесений, которые можно было прочесть в темноте.



Наверняка вы вспомните что когда-то видели «точки» на кнопках лифтов в общественных местах или надписи на отдельных продуктах и товарах.

Для изображения букв в шрифте Брайля используются шесть точек (ощущаемых пальцем выпуклостей). Точки расположены в два столбца. При письме точки прокалываются или продавливаются с обратной стороны листа, таким образом «писать» текст приходится в обратном порядке – справа налево, чтобы читать его можно было как обычно, – слева направо.

Для читающего точки нумеруются по столбцам слева направо и по строкам сверху вниз. Используя традиционный (шеститочечный) шрифт Брайля, можно записать $2^6 = 64$ различных символа: 63 информативных и один пробел. В расширенном (восьмиточечном) шрифте Брайля – $2^8 = 256$ символов: 255 информативных и один пробел.

В настоящее время шрифт Брайля приспособлен для письма на различных языках, включая китайский. Также он используется для нотного письма и в системе математических обозначений. Вы без труда в интернете найдёте правила кодирования и обозначения интересующих вас символов. Для повышения вашей эрудиции отметим, что существует несколько версий (уровней) шрифта Брайля:

Уровень 1. Состоит из 26 стандартных букв алфавита и знаков препинания. Используется только теми людьми, которые впервые начинают читать с помощью шрифта Брайля.

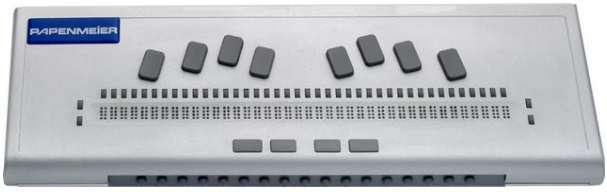
Уровень 2. Состоит из 26 стандартных букв алфавита, знаков препинания и сокращений. Сокращения используются с целью экономии места, поскольку страница со шрифтом Брайля не может уместить столько текста, как обычная печатная страница. В основном, на вывесках в общественных местах, в книгах, меню и большей части других информационных материалов используется 2-й уровень шрифта Брайля.

Уровень 3. В основном, используется при личной переписке, в дневниках и записках, а также в некоторой степени – в произведениях литературы. Это разновидность стенографии, где целые слова сокращаются до нескольких букв.

Интересный факт. Из-за особенностей шрифта Брайля в письме на его основе сделаны некоторые изменения правил набора текста. В результате чего ребёнок, обучавшийся по системе Брайля, впоследствии может допускать ряд характерных ошибок при написании или наборе обычного текста традиционным образом, например на компьютере.

С развитием ЭВМ появился и новый термин «компьютерные тифлотехнологии»[83] – общее название комплекса средств, обеспечивающих незрячим и слабовидящим людям возможность самостоятельного использования обычного персонального компьютера и программ общего назначения.

Так, существуют тактильные брайлевские дисплеи – небольшие прямоугольные коробочки, соединяемые с компьютером через USB или Bluetooth, на верхней стороне которых одновременно отображается 40 или 80 выпуклых символов. Каждая брайлевская ячейка имеет 6 или 8 точек, изготовленных из металла или нейлона. Они поднимаются или опускаются при помощи электронного управления и создают брайлевское представление символа, который отображён на экране компьютера. Управляется брайлевский дисплей программой чтения экрана или специальными кнопками, расположенными на его корпусе. Существуют портативные модели с меньшим числом отображаемых символов. Цена таких тактильных дисплеев варьируется от 100 до 500 тысяч рублей.



2.5.3. Сжатие (архивация) различных видов информации

Дискретное двоичное представление информации обычно имеет некоторую избыточность. Часто в информации присутствуют последовательности одинаковых битов или их групп. Объём информации имеет большое значение не только для хранения, но также непосредственно влияет на скорость передачи информации по компьютерным

[83] От греч. typhlos – слепой.

сетям. Поэтому были разработаны специальные методы (алгоритмы) ***сжатия информации*** (*data compression*), с помощью которых можно существенно уменьшить её объём. Существуют как универсальные алгоритмы, которые рассматривают информацию как простую последовательность битов, так и специализированные, которые предназначены для сжатия информации определённого типа (изображений, текста, звука и видео).

Все алгоритмы сжатия оперируют входным потоком информации, минимальной единицей которой является бит, а максимальной – несколько бит, байт или несколько байтов.

Основными техническими характеристиками процессов сжатия и результатов их работы являются:

степень сжатия (compress rating), или отношение (ratio) объёмов исходного и результирующего потоков;

скорость сжатия – время, затрачиваемое на сжатие некоторого объёма информации входного потока до получения из него эквивалентного выходного потока;

качество сжатия – величина, показывающая, насколько сильно упакован выходной поток при помощи применения к нему повторного сжатия по этому же или иному алгоритму.

Все способы сжатия можно разделить на две категории: **обратимое** и **необратимое** сжатие.

Необратимое сжатие – такое преобразование входного потока информации, при котором выходной поток, основанный на определённом формате информации, представляет собой объект, достаточно похожий по внешним характеристикам на входной поток, однако отличается от него объёмом.

Степень сходства входного и выходного потоков определяется степенью соответствия некоторых свойств объекта (до сжатия и после), представляемого данным потоком информации. Такие подходы и алгоритмы используются для сжатия информации растровых графических файлов, видео и звука. При таком подходе используются свойство структуры данного формата файла и возможность представить информацию, приблизительно схожую по качеству для восприятия человеком. Поэтому, кроме степени или величины сжатия, в таких алгоритмах возникает понятие качества, так как исходная информация в процессе сжатия изменяется. Под качеством можно понимать степень соответствия исходной и результирующей информации, оцениваемого субъективно, исходя из формата информации. Для графических файлов такое соответствие определяется визуально, хотя имеются и соответствующие интеллектуальные алгоритмы и программы. Необратимое сжатие невозможно применять в областях, в которых необходимо иметь точное соответствие информационной структуры входного и выходного потоков. Данный подход реализован в популярных форматах представления фотоинформации – **JPEG, TIFF, GIF, PNG** и др., аудиоинформации – **MP3**, видеоинформации – **MPEG-4**.

Обратимое сжатие всегда приводит к снижению объёма выходного потока информации без изменения его информативности, то есть без потери информационной структуры.

Из выходного потока при помощи восстанавливающего или декомпрессирующего алгоритма можно получить входной, а процесс восстановления называется декомпрессией, или распаковкой, и только после процесса распаковки информация пригодна для использования в соответствии с их внутренним форматом.

2.5.3.1. Способы обратимого сжатия информации

1) Сжатие способом кодирования серий (RLE)

Наиболее известный простой подход и алгоритм сжатия информации обратимым путём – это кодирование серий последовательностей (Run Length Encoding – RLE).

Суть методов данного подхода состоит в замене цепочек или серий повторяющихся байтов или их последовательностей на один кодирующий байт и счётчик числа их повторений.

Например:

44 44 44 11 11 11 11 11 01 33 FF 22 22 – исходная последовательность;

03 44 05 11 00 03 01 33 FF 02 22 – сжатая последовательность.

Первый байт во второй последовательности указывает, сколько раз нужно повторить следующий байт.

Если первый байт равен 00, то затем идёт счётчик, показывающий, сколько за ним следует неповторяющихся байтов информации (00 03).

Данные методы, как правило, достаточно эффективны для сжатия растровых графических изображений (BMP, PCX, TIF, GIF), так как последние содержат достаточно много длинных серий повторяющихся последовательностей байтов. Недостатком метода RLE является достаточно низкая степень сжатия.

2) Алгоритм Хаффмана

Сжимая файл по алгоритму Хаффмана [84], первое, что необходимо сделать, – прочитать файл полностью и подсчитать, сколько раз встречается каждый символ из расширенного набора **ASCII**.

Если учитывать все 256 символов, то не будет разницы в сжатии текстового и EXE-файла.

После подсчёта частоты вхождения каждого символа необходимо сформировать бинарное дерево для кодирования с учётом частоты вхождения символов.

Пример сжатия по алгоритму Хаффмана приведён ниже.

Пусть файл имеет длину 100 байт и в нём присутствуют 6 различных символов. Подсчитаем вхождение каждого из символов в файл и получим следующую таблицу:

Символ	*A*	*B*	*C*	*D*	*E*	*F*
Число вхождений	10	20	30	5	25	10

Отсортируем символы по частоте вхождения:

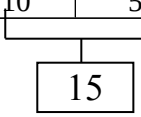
Символ	*C*	*E*	*B*	*F*	*A*	*D*
Число вхождений	30	25	20	10	10	5

Далее возьмём из последней таблицы 2 символа с наименьшей частотой. В нашем случае это *D* (5) и *F* (10) или *A* (10), можно взять любой из них, например *A*.

[84] Дэвид Хаффман (англ. *David Albert Huffman*; 1925–1999) – первопроходец в сфере теории информации. В 1952 году создал алгоритм префиксного кодирования с минимальной избыточностью (известный как алгоритм или код Хаффмана). В 1999 году получил медаль Ричарда Хэмминга за исключительный вклад в теорию информации.

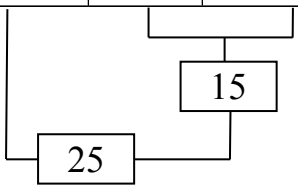
Сформируем из «узлов» *D* и *A* новый «узел», частота вхождения для которого будет равна сумме частот *D* и *A*:

Символ	C	E	B	F	A	D
Число вхождений	30	25	20	10	10	5



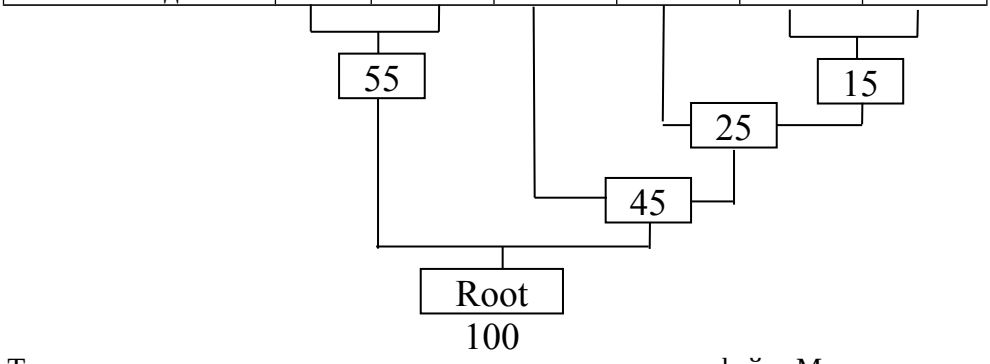
Номер в рамке – сумма частот символов A и D. Теперь мы снова ищем два символа с самыми низкими частотами вхождения, исключая из просмотра D и A и рассматривая вместо них новый «узел» с суммарной частотой вхождения. Самая низкая частота теперь у F и нового «узла». Снова сделаем операцию слияния узлов:

Символ	C	E	B	F	A	D
Число вхождений	30	25	20	10	10	5



Просматриваем таблицу снова для следующих двух символов (B и E). Продолжаем этот режим, пока всё «дерево» не сформировано, то есть пока всё не сведётся к одному узлу.

Символ	C	E	B	F	A	D
Число вхождений	30	25	20	10	10	5



Теперь, когда наше дерево создано, можно кодировать файл. Мы должны всегда начинать из корня (Root). Кодируя первый символ (лист дерева C с наибольшей частотой), прослеживаем вверх по дереву все повороты ветвей, и если делаем левый поворот, то запоминаем бит = 0, и аналогично бит = 1 для правого поворота. Так, для C мы будем идти влево к 55 (и запомним 0), затем снова влево (0) к самому символу. Код Хаффмана для нашего символа C – 00. Для следующего символа (E) получается – лево, право, что выливается в последовательность 01. Выполнив эту процедуру для всех символов, получим:

C = 00 (2 бита) *B* = 10 (2 бита) *A* = 1110 (4 бита)
E = 01 (2 бита) *F* = 110 (3 бита) *D* = 1111 (4 бита)

При кодировании заменяем символы на новые коды, при этом те символы, которые встречаются наиболее часто, имеют самые короткие коды. Таблицу кодирования запоминаем в том же архивном файле для последующей разархивации.

Замечание от авторов: вспомните азбуку морзе, как там кодируются, например, буквы «е» и «щ»?

3) Арифметическое кодирование

Совершенно иное решение предлагает так называемое арифметическое кодирование. Арифметическое кодирование является методом, позволяющим упаковывать символы входного алфавита без потерь при условии, что известно распределение частот этих символов и, что является наиболее оптимальным, так как достигается теоретическая граница степени сжатия.

Каждый символ после кодирования представляется как некоторое дробное число из интервала [0, 1) с весом, пропорциональным вероятности его появления.

4) Алгоритм Лемпеля–Зива–Велча (Lempel-Ziv-Welch – LZW)

Данный алгоритм отличают высокая скорость работы как при упаковке, так и при распаковке, достаточно скромные требования к памяти и простая аппаратная реализация.

Был опубликован Велчем в 1984 году в качестве улучшенной реализации алгоритма LZ78, опубликованного Лемпелем и Зивом в 1978 году. Алгоритм разработан так, чтобы его можно было быстро реализовать, но он не обязательно оптимален, поскольку он не проводит никакого анализа входных данных. Ещё один недостаток – низкая степень сжатия, по сравнению со схемой двухступенчатого кодирования.

5) Двухступенчатое кодирование. Алгоритм Лемпеля–Зива

Гораздо большей степени сжатия можно добиться при выделении из входного потока повторяющихся цепочек – блоков – и кодирования ссылок на эти цепочки с построением хэш-таблиц от первого до *n*-го уровня с последующим кодированием Хаффмана или арифметическим кодированием.

Метод принадлежит Лемпелю и Зиву и обычно называется LZ-compression.

2.5.3.2. Перечень форматов и программ сжатия с кратким указанием алгоритмов их работы

ZIP: метод Shrinked – модифицированный алгоритм LZW с частичной очисткой словаря и переменной длиной кода. Метод Imploded – модифицированный алгоритм Лемпеля–Зива и статическое кодирование Хаффмана.

gzip: алгоритм Лемпеля–Зива (Lempel-Ziv coding, LZ77), описания же формата gzip v 4.3 можно найти в RFC 1952. Открытая библиотека zlib доступна по адресу http://zlib.net/.

bzip2: сжимает файлы, используя преобразование Барроуза [85] – Уилера [86] (Burrows-Wheeler transform, BWT, также исторически называется блочно-сортирующим сжатием, хотя сжатием и не является) и кодирование кодами Хаффмана. Степень сжатия оказывается значительно лучше, чем при стандартном LZ77/LZ78-сжатии, а скорость – на уровне адаптивного статистического алгоритма сжатия данных без потерь, основанно-

[85] Michael Burrows (1963 г. р.) – английский специалист в области компьютерных наук.
[86] David John Wheeler FRS (1927–2004) – учёный в области компьютерных наук.

го на контекстном моделировании и предсказании (PPM, *англ. Prediction by Partial Matching* – предсказание по частичному совпадению).

7z, **7-zip:** LZMA (англ. Lempel-Ziv-Markov [87] chain-Algorithm) – алгоритм сжатия данных, разрабатываемый с 2001 года. Степень сжатия данным алгоритмом на 30–50% лучше, чем алгоритмом, используемым для ZIP-формата.

PKPAK: метод Packed – алгоритм RLE. метод Crunched – алгоритм LZW. метод Squashed – двухпроходное статическое кодирование Хаффмана.

LHArc: алгоритм Лемпеля–Зива и динамическое кодирование Хаффмана.

LHA: алгоритм Лемпеля–Зива и статическое кодирование Хаффмана.

ARJ: алгоритм Лемпеля–Зива и оригинальный метод кодирования.

cabextract: используется для работы с **.cab**-файлами (архивы со сжатием, применяемые в операционных системах семейства Microsoft Windows). Формат поддерживает три метода сжатия данных: DEFLATE, разработанный Филом Кацем, автором формата ZIP; Quantum, лицензированный у David Stafford, автора архиватора Quantum; LZX, разработанный Джонатаном Форбсом (англ. Jonathan Forbes) и Томи Поутаненом (англ. Tomi Poutanen) и полученный Microsoft после того, как Джонатан Форбс присоединился к компании.

RAR, WinRar: используется высокоэффективный авторский алгоритм. Аббревиатура RAR происходит от фамилии автора **R**oshal [88] **AR**chive.

.ipa: формат архивных файлов приложений от Apple для iPhone, iPod Touch и iPad. Как правило, файлы этого формата шифруются с применением технологии Apple's FairPlay DRM. Каждый .ipa-файл – это бинарный файл для ARM-архитектуры, который на деле является ZIP-архивом с деревом каталогов определённой структуры и содержит исполняемый файл, файлы ресурсов, таких как видео, аудио и изображения и т. д. Файлы .ipa могут быть установлены на iPhone, iPod Touch, iPad посредством покупки соответствующих программ в AppStore либо через утилиты типа *VShare*, *PPHelper*, *Zeusmos* и др. (*аналоги бывшего Installous*). Также при желании указанные файлы могут быть установлены и запущены в iPhone/iPad-симуляторах [89] на обычном компьютере.

Apple Disk Image, .dmg: проприетарный формат, в основном используемый для распространения архивов программ через интернет, поддерживается ОС Mac OS X. Использует сжатие алгоритмами LZ77 и bzip2.

Подробнее про архивацию файлов см. раздел 5.6.2. Архивация файлов (стр. 525).

[87] Андре́й Андре́евич Ма́рков (1856–1922) – русский математик, академик, внёсший большой вклад в теорию вероятностей, математический анализ и теорию чисел.

[88] Евге́ний Ла́заревич Роша́л (1972 г. р.) – российский программист, автор известного файлового менеджера FAR Manager (http://www.farmanager.com/), формата сжатия RAR, архиваторов RAR и WinRAR, особенно популярных в России и странах бывшего СССР.

[89] Например, см. http://ipadian.net/.

2.6. Контрольные вопросы к главе 2

1. В какой системе счисления представлена информация при её обработке и хранении на компьютере?
2. Что есть машинное слово, чем определяется его размер?
3. Как выглядит прямой код?
4. Что из себя представляет дополнительный код и зачем он нужен?
5. Какие алгоритмы вычисления дополнительного кода вы знаете?
6. Как отрицательные числа хранятся в памяти ЭВМ?
7. Чем отличаются архитектуры LITTLE_ENDIAN и BIG_ENDIAN, какая используется в домашних ПК?
8. Какие типы данных используются в языке С для хранения чисел с плавающей запятой?
9. Какие ошибки могут возникнуть при сложении двух чисел?
10. Как полностью называется и что описывает стандарт IEEE754-2008?
11. Как полностью называется и что описывает стандарт IEEE 854-1987?
12. Как выглядит естественная форма записи числа?
13. Как выглядит нормализованный экспоненциальный вид записи числа?
14. Как выглядит денормализованный экспоненциальный вид записи числа?
15. Что такое мантисса, как этот термин обозначается по-английски?
16. Что такое показатель степени, как этот термин обозначается по-английски?
17. Что обозначает индекс 16 в записи $110A_{16}$?
18. Как записать число $+\infty$ в памяти ЭВМ?
19. Как записать число $-\infty$ в памяти ЭВМ?
20. Почему в формате с плавающей запятой существуют два способа записи нуля?
21. В чём различие понятий «мантисса» и «хвост мантиссы»?
22. Какие диапазоны чисел рассчитываются по формуле № 1?
23. Какие диапазоны чисел рассчитываются по формуле № 2?
24. Что есть «не числа»?
25. Какие биты отводятся для хранения смещённого показателя степени в формате IEEE754-2008 binary32?
26. Почему вместо показателя степени хранится смещённый показатель степени?
27. Приведите простой код для представления какого-либо текстового алфавита в памяти ЭВМ.
28. В чём разница при кодировании целых и вещественных чисел?
29. Правильно ли утверждение: представимые числа с плавающей запятой распределены равномерно по числовой оси от min до max? Если нет, то почему это не верно?
30. Что значит выражение «околонулевая яма»?
31. Как расшифровывается аббревиатура ACSII?
32. Сколько байтовой является кодировка KOI8-R?
33. Какие кодировки используются для работы с кириллицей?
34. Что такое UTF-8?
35. Чем Unicode отличается от UTF-8?
36. С какого момента в кодировке Unicode появился знак валюты рубля?
37. Зачем нужен специальный символ «BOM»?
38. Какой консольной утилитой в ОС Linux можно переконвертировать текстовый файл из одной кодировки в другую?

39. Как в консоли ОС Linux изменить формат символов перевода строки в текстовом файле?
40. Как аналоговая информация (например, звук) представляется в ЭВМ?
41. Сформулируйте теорему В. А. Котельникова, какое название она имеет в зарубежной литературе?
42. Назовите примеры программ (желательно из области свободного ПО) для работы со звуковой информацией в ПК.
43. Что есть частота выборки или частота дискретизации?
44. Что такое глубина кодирования звука?
45. В каких единицах измеряется битрейт?
46. Что вы знаете про формат FLAC?
47. Какие проблемы возникают при оцифровывании звука?
48. Какие проблемы возникают при восстановлении аналогового звука?
49. Что такое пиксель?
50. Что есть RGB- и CMYK-модели кодирования цвета?
51. Является ли программа GNU Image Manipulation Program кросс-платформенной?
52. Какие способы представления графической информации вы знаете?
53. В чём различие векторной и растровой графики?
54. Бывают ли вектроные шрифты цветными?
55. Приведите примеры программы для работы с векторной графикой.
56. Как в компьютере представляются символы эмодзи (смайлики) графикой или текстом?
57. Приведите примеры изображений, построенных с использованием элементов фрактальной графики.
58. Назовите примеры программ, для работы с векторными графическими форматами представления данных.
59. Назовите примеры программ работающих с трёхмерной графикой.
60. Какие форматы и стандарты кодирования видеоинформации вы знаете?
61. Что такое пленоптика?
62. Какие 1D и 2D штрихкоды вы знаете?
63. Что такое QR-код, как он выглядит, где используется?
64. Зачем используется шрифт Брайля?
65. Что есть сжатие информации?
66. В чём отличие обратимого сжатия от необратимого?
67. Какие способы обратимого сжатия вам известны?
68. Приведите примеры программ-архиваторов, и какие алгоритмы сжатия они используют?
69. EXIF – что это такое?
70. Как удалить EXIF-данные из .jpg-файла?

2.7. Литература к главе 2

1. Дополнительный код для представления целых чисел со знаком // http://www.dpla.ru/celye/2.10072005.htm.
2. Дидактические материалы по информатике // http://comp-science.narod.ru/didakt_i.html.
3. On-line калькулятор: Прямой, дополнительный и обратный коды //

http://planetcalc.ru/747/.
4. Two's complement // http://en.wikipedia.org/wiki/Two%27s_complement.
5. *Хоровиц П.,.Хилл У.* Искусство схемотехники / пер. с англ. – 6-е изд. – М.: Мир, 2003. – 704 с., ил. ISBN 5-03-003395-5.
6. Two's Complement Representation of Integers // http://cs.roanoke.edu/Fall2008/CPSC120B/twoscomplement.html.
7. Порядок байтов // http://ru.wikipedia.org/wiki/Порядок_байтов.
8. Programmer's guide // ftp://ftp.freepascal.org/pub/fpc/docs-pdf/prog.pdf.
9. Функция is_int() в PHP // http://www.php.net/manual/en/function.is-int.php.
10. *Митчел М., Оулдем Д., Самьюэл А.* Программирование для Linux. Профессиональный подход / пер. англ. – М.: Издательский дом «Вильямс», 2002. ISBN 5-8459-0243-6.
11. Kip R. Irvine, Assembly Language for x86 3 Processors (6th Edition), Prentice Hall, 2010, ISBN-10 013602212X, ISBN-13 : 978-0136022121.
12. ГОСТ 28043–89 – Персональные электронные вычислительные машины. Интерфейс накопителей на жёстких несменных магнитных дисках с подвижными головками. Общие требования // http://protect.gost.ru/document.aspx?control=7&id=139432.
13. Integer (computer science) // http://en.wikipedia.org/wiki/Integer_%28computer_science%29.
14. Документация PHP: Целые числа // http://www.php.net/manual/ru/language.types.integer.php.
15. Документация PHP: Числа с плавающей точкой // http://www.php.net/manual/ru/language.types.float.php.
16. *Даймонд Д., Торвальдс Л.* Just for fun. Рассказ нечаянного революционера./ пер. с англ. – М.: Эксмо-Пресс, 2002. ISBN 5-04-009285-7.
17. *Яшкардин В.* IEEE 754 – стандарт двоичной арифметики с плавающей точкой // http://www.softelectro.ru/ieee754.html.
18. IEEE 754 Converter // http://www.h-schmidt.net/FloatApplet/IEEE754.html, http://www.h-schmidt.net/FloatConverter/IEEE754.html.
19. Что нужно знать про арифметику с плавающей запятой // http://habrahabr.ru/post/112953/.
20. *Закляков П.* Представление чисел в памяти ЭВМ. Часть 1: Целые числа // Системный администратор – 2012. – № 3 (112). – С. 90–96; № 5 (114) – С. 88–91.
21. *Страуструп Б.* (*через «а» — прим. авт.*) Язык программирования C++, – Спец. изд. / пер. с англ. – М.; СПб.: БИНОМ; Невский Диалект, 2001. ISBN 5-7989-0223-4, ISBN 5-7940-0064-3.
22. IEEE 754: Standard for Binary Floating-Point Arithmetic // http://grouper.ieee.org/groups/754/.
23. IEEE Standard for Floating-Point Arithmetic // http://ali.ayad.free.fr/IEEE_2008.pdf.
24. Countries using Arabic numerals with decimal comma // http://en.wikipedia.org/wiki/Decimal_separator.
25. *Холодилов С.* Плавающая запятая // http://www.rsdn.ru/article/alg/float.xml.
26. *Гашков С. Б.* Системы счисления и их применение. – М.: МЦНМО, 2004. ISBN

5-94057-146-8.
27. Стандарт языка C: ISO/IEC JTC1/SC22/WG14 – C // http://www.open-std.org/jtc1/sc22/wg14/.
28. TR 24732: Decimal floating point, draft ISO/IEC JTC1 SC22 WG14 N1312 // http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1312.pdf.
29. Интервью с Уильямом Кэхэном про становление стандарта IEEE754 (на англ.) // http://www.eecs.berkeley.edu/~wkahan/ieee754status/754story.html.
30. David Goldberg What Every Computer Scientist Should Know About Floating-Point Arithmetic // http://www.validlab.com/goldberg/paper.pdf, http://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html.
31. Differences Among IEEE 754 Implementations // http://www.validlab.com/goldberg/addendum.html.
32. *Грошев А. С.* Информатика: учеб. для вузов / А. С. Грошев. – Архангельск: Арханг. гос. техн. ун-т, 2010. – 470 с. ISBN 978-5-261-00480-6.
33. UTF-8, a transformation format of ISO 10646 // http://tools.ietf.org/html/rfc2279.
34. 11 декабря 2013 года Банк России утвердил графическое обозначение валюты рубля. // http://www.cbr.ru/today/?prtid=voterub
35. Announcing The Unicode Standard, Version 7.0 // http://blog.unicode.org/2014/06/announcing-unicode-standard-version-70.html
36. *Симаненков Д.* Из аналога в цифру и обратно: немного теории… // Компьютерра – 1998. – № 30-31 (258-259). – С. 20–27, http://old.computerra.ru/1998/258/1481/.
37. *Каппелини В. и др.* Цифровые фильтры и их применение: пер. с англ./ В. Каппелини, А. Дж. Константинидис, П. Эмилиани. – М.:Энергоатомиздат, 1983 – 350 с.
38. *Скарубин А.* Свет и цвет: основы основ // http://geektimes.ru/post/202966/
39. *Скарубин А.* Температура цвета // http://geektimes.ru/post/193142/
40. *Гонсалес Р., Вудс Р.* Цифровая обработка изображений / пер. с англ. – 3-е изд. исп. и доп. – М.: Техносфера, 2012. – 1104 с., ISBN 978-5-94836-331-8.
41. *Подбельский В., Фомин С.* Программирование на языке Си: Учебн. пособие. – 2-е доп. изд. – М.:Финансы и статистика, 2002, ISBN 5-279-02180-6.
42. ГОСТ Р 56042-2014 Национальный стандарт Российской Федерации «Двумерные символы штрихового кода для осуществления платежей физических лиц» // http://docs.cntd.ru/document/1200110981.
43. *Котельников В.А.* О пропускной способности «эфира» и проволоки в электросвязи // Всесоюзн. энерг. ком., Материалы к первому всесоюзному съезду по вопросам реконструкции дела связи и развития слаботочной промышленности (изд. ред. Упр. связи РККА, 1993).
44. *Шапиро Л.* Компьютерное зрение / Л.Шапиро, Дж.Стокман; Пер. с англ. – М.: БИНОМ. Лабораторая знаний, 2006. – 752 с., ISBN 5-94774-384-1, (англ. ISBN 0-13-030796-3).

Глава 3. Законодательство РФ о защите компьютерной информации

Ограничимся цитированием нескольких наиболее важных статей законов, действующих в Российской Федерации, имеющих отношение к данной теме. Текст любых законов, согласно статье 4 ФЗ от 14 июня 1994 г. № 5-ФЗ «*О порядке опубликования и вступления в силу федеральных конституционных законов, федеральных законов, актов палат Федерального Собрания*»: «*Официальным опубликованием федерального конституционного закона, федерального закона, акта палаты Федерального Собрания считается первая публикация его полного текста в "Парламентской газете", "Российской газете", "Собрании законодательства Российской Федерации" или первое размещение (опубликование) на "Официальном интернет-портале правовой информации" (www.pravo.gov.ru*[90]*).»*, можно найти, как и узнать, когда они вступают в силу в вышеуказанных источниках.

Указанный URL-адрес входит в «*Государственную систему распространения правовых актов в электронном виде*» (ГСРПА), а его статус как официального публикатора правовых актов был определён Федеральным законом от 21 октября 2011 года № 289-ФЗ «*О внесении изменений в Федеральный закон "О порядке опубликования и вступления в силу федеральных конституционных законов, федеральных законов, актов палат Федерального Собрания"*».

Также информацию можно найти и в различных коммерческих системах, выпускаемых фирмами ИПК «Кодекс», НПП «Гарант», АО «КонсультантПлюс» и др. Несмотря на то что некоторые из фирм могут быть победителями различных государственных тендеров, например на разработку систем классификации правовых актов России, а интерфейс может быть очень удобен, содержать комментарии и ссылки, отображаемую в них информацию желательно перепроверять в вышеуказанных официальных источниках.

Несомненно и то, что выдержки из законодательных актов, распоряжений и прочих официальных документов можно найти и в интернете, в том числе на официальных сайтах различных государственных структур, таких как Минкомсвязи России, ФСТЭК России и др. [91]

Следует помнить, что законодательство РФ достаточно часто модифицируется и дополняется. Например, вместо отдельных законов, в том числе Закона об авторском праве, в 2006 г. была принята часть четвёртая Гражданского кодекса, в редакцию которой позднее были внесены небольшие изменения другими федеральными законами и Определениями Конституционного Суда РФ.

Не забывайте ставшие уже крылатыми фразы «*незнание законов не освобождает от ответственности*», а «*знание – сила*». Отслеживайте изменения в интересующей вас области самостоятельно.

Замечание. Актуализация нижеизложенного материала на январь 2021 года.

[90] Ранее также был доступен адрес http://pravo.msk.rsnet.ru/.
[91] http://minsvyaz.ru, http://fstec.ru/ и др.

3.1. Уголовный кодекс РФ о преступлениях в сфере компьютерной информации

Глава 28. Преступления в сфере компьютерной информации

Статья 272. Неправомерный доступ к компьютерной информации

1. Неправомерный доступ к охраняемой законом компьютерной информации, если это деяние повлекло уничтожение, блокирование, модификацию либо копирование компьютерной информации, –

наказывается штрафом в размере до двухсот тысяч рублей или в размере заработной платы или иного дохода осуждённого за период до восемнадцати месяцев, либо исправительными работами на срок до одного года, либо ограничением свободы на срок до двух лет, либо принудительными работами на срок до двух лет, либо лишением свободы на тот же срок.

2. То же деяние, причинившее крупный ущерб или совершенное из корыстной заинтересованности, –

наказывается штрафом в размере от ста тысяч до трёхсот тысяч рублей или в размере заработной платы или иного дохода осуждённого за период от одного года до двух лет, либо исправительными работами на срок от одного года до двух лет, либо ограничением свободы на срок до четырёх лет, либо принудительными работами на срок до четырёх лет, либо арестом на срок до шести месяцев, либо лишением свободы на тот же срок.

3. Деяния, предусмотренные частями первой или второй настоящей статьи, совершенные группой лиц по предварительному сговору или организованной группой либо лицом с использованием своего служебного положения, –

наказываются штрафом в размере до пятисот тысяч рублей или в размере заработной платы или иного дохода осуждённого за период до трёх лет с лишением права занимать определённые должности или заниматься определённой деятельностью на срок до трёх лет, либо ограничением свободы на срок до четырёх лет, либо принудительными работами на срок до пяти лет, либо лишением свободы на тот же срок.

4. Деяния, предусмотренные частями первой, второй или третьей настоящей статьи, если они повлекли тяжкие последствия или создали угрозу их наступления, –

наказываются лишением свободы на срок до семи лет.

Примечание 1. Под компьютерной информацией понимаются сведения (сообщения, данные), представленные в форме электрических сигналов, независимо от средств их хранения, обработки и передачи.

Примечание 2. Крупным ущербом в статьях настоящей главы признаётся ущерб, сумма которого превышает один миллион рублей.

Статья 273. Создание, использование и распространение вредоносных компьютерных программ

1. Создание, распространение или использование компьютерных программ либо иной компьютерной информации, заведомо предназначенных для несанкционированного уничтожения, блокирования, модификации, копирования компьютерной информации или нейтрализации средств защиты компьютерной информации, –

наказываются ограничением свободы на срок до четырёх лет, либо принудительными работами на срок до четырёх лет, либо лишением свободы на тот же срок со штрафом в размере до двухсот тысяч рублей или в размере заработной платы или иного дохода осуждённого за период до восемнадцати месяцев.

2. Деяния, предусмотренные частью первой настоящей статьи, совершённые группой лиц по предварительному сговору или организованной группой либо лицом с использованием своего служебного положения, а равно причинившие крупный ущерб или совершённые из корыстной заинтересованности, –

наказываются ограничением свободы на срок до четырёх лет, либо принудительными работами на срок до пяти лет с лишением права занимать определённые должности или заниматься определённой деятельностью на срок до трёх лет или без такового, либо лишением свободы на срок до пяти лет со штрафом в размере от ста тысяч до двухсот тысяч рублей или в размере заработной платы или иного дохода осуждённого за период от двух до трёх лет или без такового и с лишением права занимать определённые должности или заниматься определённой деятельностью на срок до трёх лет или без такового.

3. Деяния, предусмотренные частями первой или второй настоящей статьи, если они повлекли тяжкие последствия или создали угрозу их наступления, –

наказываются лишением свободы на срок до семи лет.

Статья 274. Нарушение правил эксплуатации средств хранения, обработки или передачи компьютерной информации и информационно-телекоммуникационных сетей

1. Нарушение правил эксплуатации средств хранения, обработки или передачи охраняемой компьютерной информации либо информационно-телекоммуникационных сетей и оконечного оборудования, а также правил доступа к информационно-телекоммуникационным сетям, повлекшее уничтожение, блокирование, модификацию либо копирование компьютерной информации, причинившее крупный ущерб, –

наказывается штрафом в размере до пятисот тысяч рублей или в размере заработной платы или иного дохода осуждённого за период до восемнадцати месяцев, либо исправительными работами на срок от шести месяцев до одного года, либо ограничением свободы на срок до двух лет, либо принудительными работами на срок до двух лет, либо лишением свободы на тот же срок.

2. Деяние, предусмотренное частью первой настоящей статьи, если оно повлекло тяжкие последствия или создало угрозу их наступления, –

наказывается принудительными работами на срок до пяти лет либо лишением свободы на тот же срок.

Статья 274[1] Неправомерное воздействие на критическую информационную инфраструктуру Российской Федерации

(введена Федеральным законом от 26.07.2017 № 194-ФЗ)

1. Создание, распространение и (или) использование компьютерных программ либо иной компьютерной информации, заведомо предназначенных для неправомерного воздействия на критическую информационную инфраструктуру Российской Федерации, в том числе для уничтожения, блокирования, модификации, копирования информации, содержащейся в ней, или нейтрализации средств защиты указанной информации, –

наказываются принудительными работами на срок до пяти лет с ограничением свободы на срок до двух лет или без такового либо лишением свободы на срок от двух до пяти лет со штрафом в размере от пятисот тысяч до одного миллиона рублей или в размере заработной платы или иного дохода осуждённого за период от одного года до трёх лет.

2. Неправомерный доступ к охраняемой компьютерной информации, содержащейся в критической информационной инфраструктуре Российской Федерации, в том числе с использованием компьютерных программ либо иной компьютерной информации, которые заведомо предназначены для неправомерного воздействия на критическую информационную инфраструктуру Российской Федерации, или иных вредоносных компьютерных программ, если он повлек причинение вреда критической информационной инфраструктуре Российской Федерации, –

наказывается принудительными работами на срок до пяти лет со штрафом в размере от пятисот тысяч до одного миллиона рублей или в размере заработной платы или иного дохода осуждённого за период от одного года до трёх лет и с ограничением свободы на срок до двух лет или без такового либо лишением свободы на срок от двух до шести лет со штрафом в размере от пятисот тысяч до одного миллиона рублей или в размере заработной платы или иного дохода осуждённого за период от одного года до трёх лет.

3. Нарушение правил эксплуатации средств хранения, обработки или передачи охраняемой компьютерной информации, содержащейся в критической информационной инфраструктуре Российской Федерации, или информационных систем, информационно-телекоммуникационных сетей, автоматизированных систем управления, сетей электросвязи, относящихся к критической информационной инфраструктуре Российской Федерации, либо правил доступа к указанным информации, информационным системам, информационно-телекоммуникационным сетям, автоматизированным системам управления, сетям электросвязи, если оно повлекло причинение вреда критической информационной инфраструктуре Российской Федерации, –

наказывается принудительными работами на срок до пяти лет с лишением права занимать определённые должности или заниматься определённой деятельностью на срок до трёх лет или без такового либо лишением свободы на срок до шести лет с лишением права занимать определённые должности или заниматься определённой деятельностью на срок до трёх лет или без такового.

4. Деяния, предусмотренные частью первой, второй или третьей настоящей статьи, совершённые группой лиц по предварительному сговору или организованной группой, или лицом с использованием своего служебного положения, –

наказываются лишением свободы на срок от трёх до восьми лет с лишением права занимать определенные должности или заниматься определённой деятельностью на срок до трёх лет или без такового.

5. Деяния, предусмотренные частью первой, второй, третьей или четвёртой настоящей статьи, если они повлекли тяжкие последствия, –

наказываются лишением свободы на срок от пяти до десяти лет с лишением права занимать определённые должности или заниматься определённой деятельностью на срок до пяти лет или без такового.

3.2. Гражданский кодекс РФ об информации и её взаимосвязях с гражданами и ЭВМ

Часть 4 Гражданского кодекса РФ определяет права на результаты интеллектуальной деятельности и средства индивидуализации.

Статья 1225. Охраняемые результаты интеллектуальной деятельности и средства индивидуализации

1. Результатами интеллектуальной деятельности и приравненными к ним средствами индивидуализации юридических лиц, товаров, работ, услуг и предприятий, которым предоставляется правовая охрана (интеллектуальной собственностью), являются:

1) произведения науки, литературы и искусства;

2) программы для электронных вычислительных машин (программы для ЭВМ);

3) базы данных;

4) исполнения;

5) фонограммы;

6) сообщение в эфир или по кабелю радио- или телепередач (вещание организаций эфирного или кабельного вещания);

...

2. Интеллектуальная собственность охраняется законом.

Статья 1253.1. Особенности ответственности информационного посредника

1. Лицо, осуществляющее передачу материала в информационно- телекоммуникационной сети, в том числе в сети «Интернет», лицо, предоставляющее возможность размещения материала или информации, необходимой для его получения с использованием информационно-телекоммуникационной сети, лицо, предоставляющее возможность доступа к материалу в этой сети, – ***информационный посредник*** – несёт ответственность за нарушение интеллектуальных прав в информационно-телекоммуникационной сети на общих основаниях, предусмотренных настоящим Кодексом, при наличии вины с учётом особенностей, установленных пунктами 2 и 3 настоящей статьи.

Статья 1256. Действие исключительного права на произведения науки, литературы и искусства на территории Российской Федерации

1. Исключительное право на произведения науки, литературы и искусства распространяется:

1) на произведения, обнародованные на территории Российской Федерации или необнародованные, но находящиеся в какой-либо объективной форме на территории Российской Федерации, и признаётся за авторами (их правопреемниками) независимо от их гражданства;

...

Статья 1261. Программы для ЭВМ

Авторские права на все виды программ для ЭВМ (в том числе на операционные системы и программные комплексы), которые могут быть выражены на любом языке и в любой форме, включая исходный текст и объектный код, охраняются так же, как авторские права на произведения литературы. Программой для ЭВМ является представленная в объективной форме совокупность данных и команд, предназначенных для функционирования ЭВМ и других компьютерных устройств в целях получения определённого результата, включая подготовительные материалы, полученные в ходе разработки программы для ЭВМ, и порождаемые ею аудиовизуальные отображения.

Статья 1262. Государственная регистрация программ для ЭВМ и баз данных

1. Правообладатель в течение срока действия исключительного права на программу для ЭВМ или на базу данных может по своему желанию зарегистрировать такую программу или такую базу данных в федеральном органе исполнительной власти по интеллектуальной собственности.

Программы для ЭВМ и базы данных, в которых содержатся сведения, составляющие государственную тайну, государственной регистрации не подлежат. Лицо, подавшее заявку на государственную регистрацию (заявитель), несёт ответственность за разглашение сведений о программах для ЭВМ и базах данных, в которых содержатся сведения, составляющие государственную тайну, в соответствии с законодательством Российской Федерации.

...

Статья 1280. Право пользователя программы для ЭВМ и базы данных

1. Лицо, правомерно владеющее экземпляром программы для ЭВМ или экземпляром базы данных (пользователь), вправе без разрешения автора или иного правообладателя и без выплаты дополнительного вознаграждения:

1) осуществлять действия, необходимые для функционирования программы для ЭВМ или базы данных (в том числе в ходе использования в соответствии с их назначением), включая запись и хранение в памяти ЭВМ (одной ЭВМ или одного пользовате-

ля сети), внесение в программу для ЭВМ или базу данных изменений исключительно в целях их функционирования на технических средствах пользователя, исправление явных ошибок, если иное не предусмотрено договором с правообладателем;

2) изготовить копию программы для ЭВМ или базы данных при условии, что эта копия предназначена только для архивных целей или для замены правомерно приобретенного экземпляра в случаях, когда такой экземпляр утерян, уничтожен или стал непригоден для использования. При этом копия программы для ЭВМ или базы данных не может быть использована в иных целях, чем цели, указанные в "подпункте 1" настоящего пункта, и должна быть уничтожена, если владение экземпляром таких программы или базы данных перестало быть правомерным.

2. Лицо, правомерно владеющее экземпляром программы для ЭВМ, вправе без согласия правообладателя и без выплаты дополнительного вознаграждения изучать, исследовать или испытывать функционирование такой программы в целях определения идей и принципов, лежащих в основе любого элемента программы для ЭВМ, путем осуществления действий, предусмотренных «подпунктом 1 пункта 1» настоящей статьи.

3. Лицо, правомерно владеющее экземпляром программы для ЭВМ, вправе без согласия правообладателя и без выплаты дополнительного вознаграждения воспроизвести и преобразовать объектный код в исходный текст (декомпилировать программу для ЭВМ) или поручить иным лицам осуществить эти действия, если они необходимы для достижения способности к взаимодействию независимо разработанной этим лицом программы для ЭВМ с другими программами, которые могут взаимодействовать с декомпилируемой программой, при соблюдении следующих условий[92]: …

Статья 1286.1. Открытая лицензия на использование произведения науки, литературы или искусства

1. Лицензионный договор, по которому автором или иным правообладателем (лицензиаром) предоставляется лицензиату простая (неисключительная) лицензия на использование произведения науки, литературы или искусства, может быть заключен в упрощенном порядке (открытая лицензия).

Открытая лицензия является договором присоединения. Все её условия должны быть доступны неопределенному кругу лиц и размещены таким образом, чтобы лицензиат ознакомился с ними перед началом использования соответствующего произведения. В открытой лицензии может содержаться указание на действия, совершение которых будет считаться акцептом её условий (статья 438). В этом случае письменная форма договора считается соблюдённой.

Кроме того, в части 4 ГК РФ присутствуют специальные статьи, посвящённые программам для ЭВМ:

Статья 1296. Произведения, созданные по заказу.

Статья 1297. Произведения, созданные при выполнении работ по договору.

[92] Обращаем внимание, что часто коммерческие фирмы пользуются общим незнанием законов, и лицензии на коммерческие программные продукты могут содержать пункты, заведомо ущемляющие права приобретателей этих продуктов. При желании такие пункты лицензионных соглашений можно признать ничтожными в установленном порядке.

Многие положения других статей относятся ко всем объектам авторского права, которыми являются и программы для ЭВМ и базы данных.

Ответственность за нарушение авторских прав изложена в УК РФ ст. 146 «Нарушение авторских и смежных прав».

3.3. Другие федеральные законы и подзаконные акты

Федеральный закон от 27 июля 2006 года № 149-ФЗ «Об информации, информационных технологиях и о защите информации»

Статья 1. Сфера действия настоящего Федерального закона

1. Настоящий Федеральный закон регулирует отношения, возникающие при:

1) осуществлении права на поиск, получение, передачу, производство и распространение информации;

2) применении информационных технологий;

3) обеспечении защиты информации.

2. Положения настоящего Федерального закона не распространяются на отношения, возникающие при правовой охране результатов интеллектуальной деятельности и приравненных к ним средств индивидуализации, за исключением случаев, предусмотренных настоящим Федеральным законом.

…

Статья 10.1. Обязанности организатора распространения информации в сети «Интернет»

1. Организатором распространения информации в сети «Интернет» является лицо, осуществляющее деятельность по обеспечению функционирования информационных систем и (или) программ для электронных вычислительных машин, которые предназначены и (или) используются для приёма, передачи, доставки и (или) обработки электронных сообщений пользователей сети «Интернет».

2. Организатор распространения информации в сети «Интернет» обязан в установленном Правительством Российской Федерации «порядке» уведомить федеральный орган исполнительной власти, осуществляющий функции по контролю и надзору в сфере средств массовой информации, массовых коммуникаций, информационных технологий и связи, о начале осуществления деятельности, указанной в части 1 настоящей статьи.

3. Организатор распространения информации в сети «Интернет» обязан хранить на территории Российской Федерации:

1) информацию о фактах приёма, передачи, доставки и (или) обработки голосовой информации, письменного текста, изображений, звуков, видео- или иных электронных сообщений пользователей сети «Интернет» и информацию об этих пользователях в течение одного года с момента окончания осуществления таких действий;

2) текстовые сообщения пользователей сети «Интернет», голосовую информацию, изображения, звуки, видео-, иные электронные сообщения пользователей сети «Интернет» до шести месяцев с момента окончания их приёма, передачи, доставки и (или) об-

работки. Порядок, сроки и объём хранения указанной в настоящем подпункте информации устанавливаются Правительством Российской Федерации.

Другие статьи данного закона содержат многочисленные положения правового регулирования в сфере информации, информационных технологий и защиты информации. Большое внимание уделяется вопросам ограничения доступа к информации, создания государственных информационных систем, защиты информации, защите персональных данных.

Федеральный закон от 27.07.2006 г. № 152-ФЗ «О персональных данных»

Статья 1. Сфера действия настоящего Федерального закона

1. Настоящим Федеральным законом регулируются отношения, связанные с обработкой персональных данных, осуществляемой федеральными органами государственной власти, органами государственной власти субъектов Российской Федерации, иными государственными органами (далее – государственные органы), органами местного самоуправления, иными муниципальными органами (далее – муниципальные органы), юридическими лицами и физическими лицами с использованием средств автоматизации, в том числе в информационно-телекоммуникационных сетях, или без использования таких средств, если обработка персональных данных без использования таких средств соответствует характеру действий (операций), совершаемых с персональными данными с использованием средств автоматизации, то есть позволяет осуществлять в соответствии с заданным алгоритмом поиск персональных данных, зафиксированных на материальном носителе и содержащихся в картотеках или иных систематизированных собраниях персональных данных, и (или) доступ к таким персональным данным.

…

В соответствии со статьёй 19 «*Меры по обеспечению безопасности персональных данных при их обработке*» указанного закона **постановлением Правительства РФ от 01.11.2012 г. № 1119** в качестве подзаконного акта были утверждены

Требования к защите персональных данных при их обработке в информационных системах персональных данных

В требованиях определяются три типа угроз для информационных систем:

«*Угрозы 1-го типа актуальны для информационной системы, если для неё в том числе актуальны угрозы, связанные с наличием недокументированных (недекларированных) возможностей в системном программном обеспечении, используемом в информационной системе.*

Угрозы 2-го типа актуальны для информационной системы, если для неё в том числе актуальны угрозы, связанные с наличием недокументированных (недекларированных) возможностей в прикладном программном обеспечении, используемом в информационной системе.

Угрозы 3-го типа актуальны для информационной системы, если для нее актуальны угрозы, не связанные с наличием недокументированных (недекларированных) возможностей в системном и прикладном программном обеспечении, используемом в информационной системе».

При обработке персональных данных в информационных системах устанавливаются 4 уровня защищённости персональных данных.

Федеральный закон Российской Федерации от 29.12.2010 г. № 436-ФЗ «О защите детей от информации, причиняющей вред их здоровью и развитию»

Статья 14. Особенности распространения информации посредством информационно-телекоммуникационных сетей

1. Доступ к информации, распространяемой посредством информационно-телекоммуникационных сетей, в том числе сети «Интернет», в местах, доступных для детей, предоставляется лицом, организующим доступ к сети «Интернет» в таких местах (за исключением операторов связи, оказывающих эти услуги связи на основании договоров об оказании услуг связи, заключённых в письменной форме), другим лицам при условии применения административных и организационных мер, технических, программно-аппаратных средств защиты детей от информации, причиняющей вред их здоровью и (или) развитию.

Для реализации вышеуказанного закона в части статьи 14 **Правительством Российской Федерации было выпущено постановление от 26 октября 2012 г. № 1101** «*О единой автоматизированной информационной системе "Единый реестр*[93] *доменных имён, указателей страниц сайтов в информационно-телекоммуникационной сети "Интернет" и сетевых адресов, позволяющих идентифицировать сайты в информационно-телекоммуникационной сети "Интернет", содержащие информацию, распространение которой в Российской Федерации запрещено*».

Федеральный закон Российской Федерации от 02.07.2013 г. № 187-ФЗ «О внесении изменений в отдельные законодательные акты Российской Федерации по вопросам защиты интеллектуальных прав в информационно-телекоммуникационных сетях»

Данный закон в большей мере нацелен на защиту исключительных прав на кинофильмы, телефильмы, в информационно-телекоммуникационных сетях, в том числе в сети «Интернет». В силу сложности его реализации, в 2013 году закон выглядел как стрельба «из пушки по воробьям». Впоследствии, в 2014 году, с выходом ФЗ № 35-ФЗ от 12.03.2014 были внесены некоторые изменения, в частности в ГК появилось понятие информационного посредника, см. ст. 1253.1 выше.

Распоряжение Правительства Российской Федерации от 31 декабря 2020 года № 3704-р

С 1 апреля 2021 года на ряд технически сложных товаров (компьютеры, ноутбуки, телевизоры, игровые приставки, планшеты, телефоны и др.), произведённых с 1 января 2021 года и проаваемых на территории России, производители будут обязаны

[93] Доступен по адресу http://www.zapret-info.gov.ru/.

предустанавливать российские программы. Полный перечень програм доступен по адресу: *http://publication.pravo.gov.ru/Document/View/0001202101060012*.

Напомним, что согласно Федеральному закону от 2 декабря 2019 года № 425-ФЗ «О внесении изменения в статью 4 Закона Российской Федерации "О защите прав потребителей", на смартфоны, компьютеры и умные телевизоры, произведённые после 1 января 2021 года, должно быть предварительно установлено российское ПО.

3.3.1. Требования к организации рабочих мест пользователей ПК

До 1 января 2021 года главным документом, определявшим данные требования, были Санитарно-эпидемиологические правила и нормативы СанПиН 2.2.2/2.4.2620-10 «*Гигиенические требования к персональным электронно-вычислительным машинам и организации работ*» (изменённые СанПиН 2.2.2/2.4.1340-03). С 1 января 2021 года указанный документ утратил силу на основании постановления Главного государственного санитарного врача Российской Федерации от 28 сентября 2020 года № 28 и отменён на основании постановления Правительства Российской Федерации от 8 октября 2020 года № 1631.

В данный момент, на основании постановления главного государственного санитарного врача российской федерации от 28 сентября 2020 года № 28 новым действующим документом являются санитарные правила СП 2.4.3648-20 «Санитарно-эпидемиологические требования к организациям воспитания и обучения, отдыха и оздоровления детей и молодёжи».

Некоторые самые существенные требования по организации рабочих мест пользователей, о которых не должен забывать простой пользователь, приведены в следующих разделах документа:

2.4.5. Интерактивные доски, сенсорные экраны, информационные панели и иные средства отображения информации, а также компьютеры, ноутбуки, планшеты, моноблоки, иные электронные средства обучения (далее – ЭСО) используются в соответствии с инструкцией по эксплуатации и (или) техническим паспортом. ЭСО должны иметь документы об оценке (подтверждении) соответствия.

Использование ЭСО должно осуществляться при условии их соответствия Единым санитарно-эпидемиологическим и гигиеническим требованиям к продукции (товарам), подлежащей санитарно-эпидемиологическому надзору (контролю).

Минимальная диагональ ЭСО должна составлять для монитора персонального компьютера и ноутбука – не менее 39,6 см, планшета – 26,6 см. Использование мониторов на основе электронно-лучевых трубок в образовательных организациях не допускается.

2.10.2. …

Общая продолжительность использования ЭСО на уроке не должна превышать для интерактивной доски - для детей до 10 лет – 20 минут, старше 10 лет – 30 минут; компьютера – для детей 1-2 классов – 20 минут, 3-4 классов – 25 минут, 5-9 классов – 30 минут, 10-11 классов – 35 минут.

Занятия с использованием ЭСО в возрастных группах до 5 лет не проводятся.

3.5.14. При использовании электронного оборудования, в том числе сенсорного экрана, клавиатуры, компьютерной мыши необходимо ежедневно дезинфицировать их в соответствии с рекомендациями производителя либо с использованием растворов или салфеток на спиртовой основе, содержащих не менее 70% спирта.

3.5.15. В помещении, где организовано рабочее место обучающегося с компьютером (ноутбуком) или планшетом, необходимо предусмотреть естественное освещение и искусственное общее и местное на рабочем столе. Источник местного освещения на рабочем месте обучающегося должен располагаться сбоку от экрана персонального компьютера (ноутбука) или планшета. Освещение не должно создавать бликов на поверхности экрана.

Ранее применявшиеся требования теперь не являются обязательными:

~~3.4. Площадь на одно рабочее место пользователей ПЭВМ с ВДТ на базе электронно-лучевой трубки (ЭЛТ) должна составлять не менее 6 м^2, в помещениях культурно-развлекательных учреждений и с ВДТ на базе плоских дискретных экранов (ЖК, плазменные) – 4,5 м^2.~~

~~При использовании ПЭВМ с ВДТ на базе ЭЛТ (без вспомогательных устройств – принтер, сканер и др.), отвечающих требованиям международных стандартов безопасности компьютеров, с продолжительностью работы менее 4 часов в день допускается минимальная площадь 4,5 м^2 на одно рабочее место пользователя (взрослого и учащегося высшего профессионального образования).~~

~~4.4. В помещениях, оборудованных ПЭВМ, проводится ежедневная влажная уборка и систематическое проветривание после каждого часа работы на ПЭВМ.~~

~~6.1. Рабочие столы следует размещать таким образом, чтобы видеодисплейные терминалы были ориентированы боковой стороной к световым проёмам, чтобы естественный свет падал преимущественно слева.~~

~~9.1. При размещении рабочих мест с ПЭВМ расстояние между рабочими столами с видеомониторами (в направлении тыла поверхности одного видеомонитора и экрана другого видеомонитора) должно быть не менее 2,0 м, а расстояние между боковыми поверхностями видеомониторов – не менее 1,2 м.~~

~~9.4. Экран видеомонитора должен находиться от глаз пользователя на расстоянии 600–700 мм, но не ближе 500 мм с учётом размеров алфавитно-цифровых знаков и символов.~~

~~9.6. Конструкция рабочего стула (кресла) должна обеспечивать поддержание рациональной рабочей позы при работе на ПЭВМ, позволять изменять позу с целью снижения статического напряжения мышц шейно-плечевой области и спины для предупреждения развития утомления. Тип рабочего стула (кресла) следует выбирать с учётом роста пользователя, характера и продолжительности работы с ПЭВМ.~~

~~Рабочий стул (кресло) должен быть подъёмно-поворотным, регулируемым по высоте и углам наклона сиденья и спинки, а также расстоянию спинки от переднего края сиденья, при этом регулировка каждого параметра должна быть независимой, легко осуществляемой и иметь надёжную фиксацию.~~

~~9.7. Поверхность сиденья, спинки и других элементов стула (кресла) должна быть полумягкой, с нескользящим, слабо электризующимся и воздухопроницаемым покрытием, обеспечивающим лёгкую очистку от загрязнений.~~

3.4. Контрольные вопросы к главе 3

1. Где можно получить официальную информацию о действующих и вступающих в силу законах на территории РФ?
2. Как наказывается неправомерный доступ к компьютерной информации и распространение вредоносных программ по законодательству РФ?
3. Как наказывается создание, распространение или использование компьютерных программ либо иной компьютерной информации, заведомо предназначенных для несанкционированного уничтожения, блокирования, модификации, копирования компьютерной информации или нейтрализации средств защиты компьютерной информации?
4. Является ли программой для ЭВМ представленная в объективной форме совокупность данных и команд, предназначенных для функционирования ЭВМ и других компьютерных устройств в целях получения определённого результата?
5. В каких случаях лицо, обладающее программой для ЭВМ, вправе без разрешения автора или иного правообладателя и без выплаты дополнительного вознаграждения внести в программу для ЭВМ или базу данных изменения?
6. Какими законами регулируются отношения, возникающие при осуществлении прав на поиск, получение, передачу, производство и распространение информации?
7. Определите понятие «информационный посредник».
8. Каким документом регулируются отношения, связанные с обработкой персональных данных?
9. Сколько и каких типов угроз для информационных систем определяют «*Требования к защите персональных данных при их обработке в информационных системах персональных данных»*?
10. Какими нормативными документами на территории РФ осуществляется защита детей от информации, причиняющей вред их здоровью и развитию, распространяемой посредством сети «Интернет»?
11. По какому адресу (URL) находится сайт для проверки вхождения того или иного сетевого ресурса в Единый реестр доменных имён, указателей страниц сайтов в сети «Интернет» и сетевых адресов, позволяющих идентифицировать сайты в сети «Интернет», содержащие информацию, распространение которой в Российской Федерации запрещено?
12. Будут ли обязаны с 1 апреля 2021 году производители отдельных технически сложных товаров осуществлять предустановку российских программ?
13. Какие основные положения по организации рабочих мест пользователей ПК вы должны выполнять?
14. При размещении столов в помещении с какой стороны желательно, чтобы свет падал на рабочий стол?
15. Что делать, если на экране монитора имеются блики от естественного освещения?

Глава 4. Аппаратное обеспечение компьютеров

4.1. Введение

В первой главе мы рассмотрели вопросы касаемо того, что в нашей жизни считать информацией, где мы с ней сталкиваемся и какими свойствами она обладает. Всячески пытались заинтересовать читателей понятными жизненными примерами, попутно вводя термины – аксиомы любой науки, чтобы в дальнейшем у читателей не возникало разночтений в компьютерной области.

Во второй главе мы попытались пояснить, что практически всё, что нас окружает, при отображении внутрь компьютера (телефона, планшета, «цифровых очков» и прочего) оказывается оцифрованным, то есть преобразуется в последовательность нулей и единиц, независимо от того, температура это на улице, скорость ветра, последний музыкальный хит или фотография любимой собаки.

Далее, по необходимости, следовала третья глава с выдержками из российского законодательства относительно юридических аспектов обработки информации. Помним: «Незнание законов не освобождает от ответственности», а «знание – сила!» Просите работать за компьютером? – обеспечьте условия.

При этом мы ни разу не коснулись механизмов внутренней работы компьютера. А именно как окружающие нас всевозможные вычислительные устройства обрабатывают нули и единицы, понимают, что мы от них хотим?

Современный компьютер – это система, построенная на базе электронных микросхем и предназначенная для хранения, обработки и передачи различных видов информации.

Информацией в компьютере являются те или иные оцифрованные данные, то есть, грубо говоря, числа или структуры состоящие из наборов нулей и единиц. Лишь мы, люди, приписываем им какой-либо иной, удобным нам вид (отображение) или смысл. Естественно, при просмотре файла «цветок.jpg» мы не восхищаемся красотой цифровой последовательности, а психологически взаимодействуем с отрисованной картинкой, изображённой некоторой программой на экране, на базе этих данных. Для этого внутри компьютера числа обрабатываются в зависимости с заданным алгоритмом (программой) и практически всегда все действия в отношении них сводятся к нескольким простым базовым действиям: копированию, сравнению, побитовым и привычным нам арифметическим операциям. О том как это происходит подробнее будет рассказано в разделе «4.5. Научные основы работы современной ЭВМ» (стр. 204).

В свою очередь, все микросхемы есть набор, соединённых вместе определённым образом «единичных» ключевых элементов, обязанных своим существованием открытию электрического тока. Их работа как раз нацелена на выполнение перечисленных выше действий.

Естественно, что изначально человеку требовалось именно считать, поэтому первые механизмы, а затем и более сложные устройства были «заточены» именно под арифметические действия, что мы отчётливо прослеживаем в корнях таких слов как «арифмометр», «calculator», «computer» и др.

Говоря о вычислительных машинах и механизмах правильным будет спросить: а какие они, компьютеры вообще бывают? Думаем, что наглядный ответ на этот вопрос можно найти в следующей таблице.

<table>
<tr><th>Вычислители</th><th>Аналоговые</th><th>Смешанные</th><th>Цифровые
(на базе двоичной системы счисления)</th></tr>
<tr><td>Электронные
(вычисления проводятся на основе протекания электрического тока)</td><td>+</td><td>+</td><td>Современные персональные и промышленные компьютеры, мобильные устройства</td></tr>
<tr><td>Смешанного типа</td><td>+</td><td colspan="2" rowspan="2">Не рассматриваются,
не получили
широкого распространения</td></tr>
<tr><td>Не электронные
(принцип вычисления не связан напрямую с протеканием электрического тока)</td><td>Счёты, механический арифмометр и др.</td></tr>
</table>

Обратим внимание, что речь идёт именно «о вычислительном ядре» системы, потому как даже в цифровых ЭВМ могут быть аналоговые модули, например для связи, либо в аналоговых машинах электричество может запитывать какой-нибудь мотор.

Замечание. Гидравлические, пневматические, смешанные и иные экзотические компьютеры в данном учебнике не рассматриваются.

4.2. История механических вычислителей

История механических устройств обширна и по своему интересна, поэтому ограничимся лишь небольшой хронологической справкой.

≈ 3000 лет до н. э. – в Древнем Вавилоне были изобретены первые счёты – абак.

≈ 500 лет до н. э. – в Китае появился «улучшенный» вариант абака с косточками на соломинках – суаньпань.

≈ 87 год до н. э. – в Греции изготовлен «антикитерский механизм» – механическое устройство на базе зубчатых передач, представляющее собой специализированный астрономический вычислитель.

1208 год – в Персии создан прибор астролябия для определения положения звёзд на небе и вычисления продолжительности дня и ночи.

В XIII веке Луллий Раймунд создал логическую машину в виде бумажных кругов, построенных по троичной логике.

1492 год – Леонардо да Винчи в одном из своих дневников приводит эскиз 13-разрядного суммирующего устройства с десятизубцовыми кольцами.

XVI век – в России усовершенствовали счёты, – 10 деревянных шариков на проволоке.

Рисунок 4.1. Счёты изготовленные в Российской империи, XIX век[94]

1623 год – Вильгельм Шиккард придумал устройство на основе зубчатых колёс («считающие часы») для сложения и вычитания шестиразрядных десятичных чисел.

1622-1630 гг. – Уильям Отред и Ричард Деламейн создают круговую и прямоугольную логарифмические линейки.

Рисунок 4.2. Логарифмические линейки отечественного производства, выполненные в виде прямоугольной линейки и в виде круглых часов со стрелкой. XX век

[94] Автор карикатуры «Цыплят по осени считают» – Рина Зенюк (*rinazeniuk@mail.ru*. Эта и другие карикатуры «синих», красных и белых котов размещены в учебнике безвозмездно с согласия художницы. Другие её карикатуры вы можете найти на её личной странице *https://vk.com/rinazeniuk*.)

1642 год – Блез Паскаль представляет механическое вычислительное устройство известное как «паскалина» для суммирования и вычитания пятиразрядных десятичных чисел. Позднее разрядность была увеличена до восьми.

1673 год – известный немецкий философ и математик Готфрид Вильгельм Лейбниц построил арифмометр, который выполнял умножение, деление, сложение и вычитание.[95]

Замечание. Примерно в это же время идёт развитие математической науки, Исаак Ньютон закладывает основы математического анализа.

1723 год – немецкий математик и астроном Христиан Людвиг Герстен на основе работ Лейбница создал арифметическую машину. Машина высчитывала частное и число последовательных операций сложения при умножении чисел. Кроме того, в ней была предусмотрена возможность контроля за правильностью ввода данных.

1786 год – немецкий военный инженер Иоганн Мюллер в ходе работ по усовершенствованию механического калькулятора на ступенчатых валиках Лейбница, придуманного его соотечественником Филиппом Хахном[96], выдвигает идею «разностной машины» – специализированного арифмометра для табулирования логарифмов, вычисляемых разностным методом.

1801 год – Жозеф Мари Жаккар строит ткацкий станок с программным управлением, программа работы которого задаётся с помощью комплекта перфокарт.

1820 год – первый промышленный выпуск арифмометров. Первенство принадлежит французу Тома де Кальмару.

1822 год – английский математик Чарльз Бэббидж изобрёл, но не смог построить, первую разностную машину (специализированный арифмометр для автоматического построения математических таблиц).

1840 год – Томас Фаулер (*англ.* Great Torrington) построил деревянную троичную счётную машину с троичной симметричной системой счисления[97].

1855 год – братья Георг и Эдвард Шутц (англ. George & Edvard Scheutz) из Стокгольма построили первую разностную машину на основе работ Чарльза Бэббиджа.

1876 год – русским математиком П.Л. Чебышёвым создан суммирующий аппарат с непрерывной передачей десятков. В 1881 году он же сконструировал к нему приставку для умножения и деления (арифмометр Чебышёва).

1884–1887 годы – Холлерит разработал электрическую табулирующую систему, которая использовалась в переписях населения США 1890 и 1900 годов и Российской империи в 1897 году.

[95] Позже Лейбниц описал двоичную систему счисления и обнаружил, что если записывать определённые группы двоичных чисел одно под другим, то нули и единицы в вертикальных столбцах будут регулярно повторяться, и это открытие навело его на мысль, что существуют совершенно новые законы математики. Лейбниц решил, что двоичный код оптимален для системы механики, которая может работать на основе перемежающихся активных и пассивных простых циклов. Он пытался применить двоичный код в механике и даже сделал чертёж вычислительной машины, работавшей на основе его новой математики, но вскоре понял, что технологические возможности его времени не позволяют создать такую машину. см. Двоичная система счисления Лейбница (по материалам «Explication de l'Arithmetique Binaire») *https://slovo.mosmetod.ru/images/yubileiniy_urok/2018/leibnic/sistema.pdf*.

[96] The calculating machines of Johann Helfrich Müller *https://history-computer.com/MechanicalCalculators/18thCentury/Muller.html*.

[97] См. The ternary calculating machine of Thomas Fowler *https://www.mortati.com/glusker/fowler/*, McKay J. The Thomas Fowler story *https://www.thomasfowler.org.uk*.

1912 год – создана машина для интегрирования обыкновенных дифференциальных уравнений по проекту русского учёного А.Н. Крылова.

1929 год – начат промышленный выпуск самого распространённого в СССР арифмометра «Феликс» (см. рис. 4.3), который выпускался вплоть до 1978 года. Общий суммарный тираж – несколько миллионов штук.

Рисунок 4.3. Арифмометр «Феликс» (назван в честь Ф.Э. Дзержинского)

1948 год – в Лихтенштейне начался промышленный выпуск карманной версии арифмометра австрийского инженера Курта Херцштарка (Curt Herzstark). Вплоть до 1970 года было выпущено порядка 140 тыс. шт. арифмометров «Curta». См. рис. 4.4.

Рисунок 4.4. Арифмометр «Curta» («перечница», «математическая граната»)

Все упомянутые выше устройства имеют общее то, что они приводятся в действие внешним двигателем, коим обычно является рука человека. Понятно, что вместо руки может быть электрический, паровой, бензиновый или иной двигатель. Фактически, заменив руку человека электрическим током мы получим переходные «механические устройства с моторчиками», механические счётные машины. С одной стороны, многие из них могли работать в отсутствии электричества за счёт враще-

ния механической ручки, с другой стороны, при наличии электрического тока они на время становились полноценными ЭВМ.[98]

Яркий пример – кассовые аппараты времён СССР. Они могли механически считать суммарную стоимость, отображать её в окошке как покупателю, так и кассиру, а за одно и печатать на бумажном носителей – чеке. (Если посмотреть формально – двухдисплейная ЭВМ с принтером, только без поддержки QR-кодов и без подключения к интернету.)

В завершение параграфа приведём ещё пару вычислителей с моторчиками. Первый – электромеханический арифмометр ВК-2 (вычислитель клавишный, вторая модель).[99] Он представляет собой дальнейшее усовершенствование механического арифмометра.

Рисунок 4.5. Кассовый аппарат «Ока» – переходное электромеханическое счётное устройство с принтером для чеков и двумя дисплеями (СССР, XX век)

Рисунок 4.6. Электромеханический арифмометр ВК-2 (1974 г.в.)

[98] Кстати, замените электрический двигатель пневматическим, паровым или иным и они уже не ЭВМ!
[99] Электромеханический арифмометр ВК-2 *https://habr.com/ru/post/535896/*.

Усовершенствования направлены прежде всего на ускорение работы и автоматизацию некоторых операций. В частности, одним из самых больших неудобств является то, что в арифмометре «Феликс» число необходимо набирать на барабане с помощью рычажков, по очереди двигая их на соответствующую позицию. Это занимает достаточно много времени и требует внимания оператора. В данном арифмометре (как и в предыдущей модели, ВК-1) числа вводятся намного быстрее - нажатием цифровых клавиш, почти как в современном калькуляторе. Также в нём полностью автоматизированы операции деления и очистки установочного регистра (барабана). Умножение полуавтоматическое.

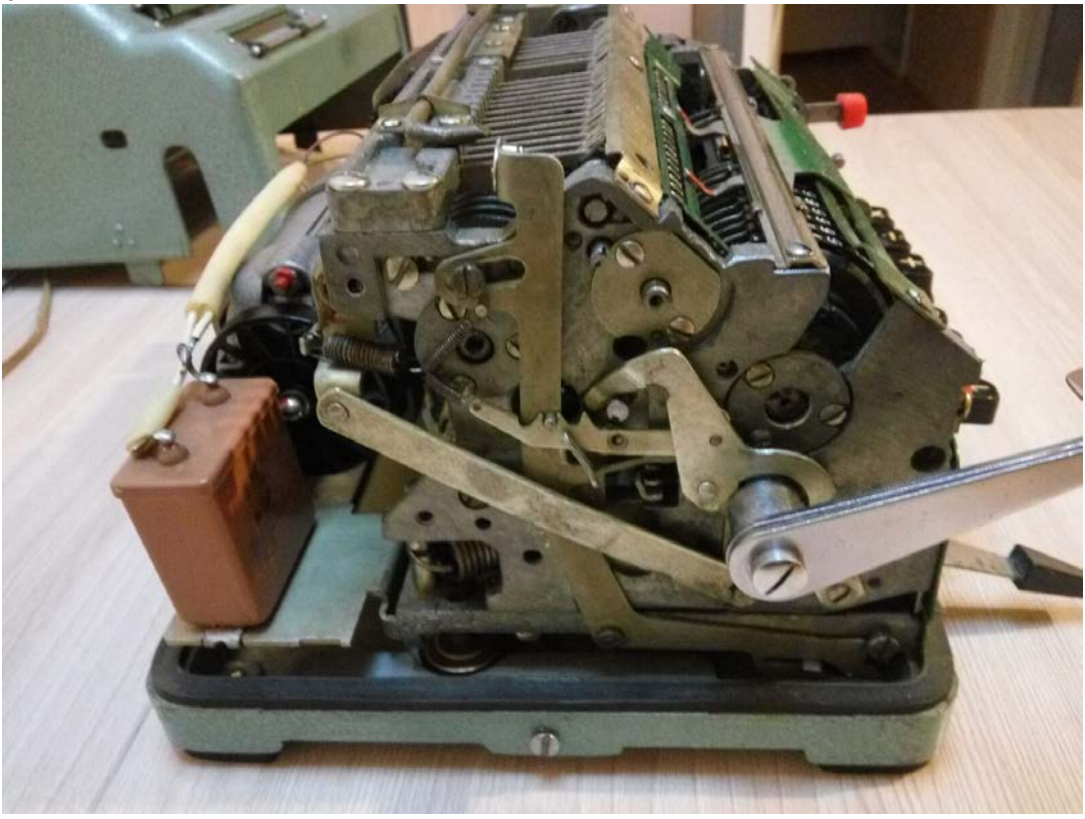


Рисунок 4.7. Внутреннее устройство ВК-2 (вычисления полностью механические)

Второй – механический программируемый компьютер родом из начала пятидесятых – «Аскота 170» (см. рис. 4.8), коих в 1955–1983 гг. было выпущено более трёхсот тысяч штук. Подробнее о его работе см.[100]. Механическая память этой машины составляла ≈ 50 шагов командной памяти (≈ 0,3 КБ) и 56 целочисленных регистров по 12 десятичных разрядов (ещё ≈ 0,3 КБ). Она поддерживает переходы и без труда может посчитать квадратный корень какого-нибудь числа. Видео как она работает: http://youtube.com/watch?v=9sUsiYnHwqI.

[100] Аскота 170 – механический компьютер и советский палеоэндемик *http://habr.com/ru/post/450538/.*

Рисунок 4.8. Бухгалтерская машина Аскота-170

Замечание. Изложенная выше хронология событий не является ни полной, ни строгой с научной точки зрения (мы, авторы, предпочли сэкономить место, если сказать иначе, – поленились привести подтверждающие факты и ссылки на библиографию). Вряд ли она выдумана, поскольку выглядит вполне логично, как следующий анекдот.

Во время раскопок в Америке были найдены кусочки медной проволоки. После долгих исследований учёных было выяснено, что это – остатки древней кабельной сети, которой пользовались индейцы.

Во время раскопок в Германии были найдены кусочки стекла. После долгих исследований было выяснено, что это – остатки оптоволоконной сети древних нибелунгов.

В России долго копали, копали, копали… и не нашли ничего. Так было выяснено, что древние славяне пользовались не только более современными беспроводными локальными сетями, но и спутниковой связью.

4.3. Аналоговые компьютеры (аналоговые вычислители)

На смену механическим вычислительным устройствам в XX веке пришли электрические. Какое-то время они существовали совместно.

Сегодня аналоговая обработка данных не получила широкого распространения в силу её плохой помехозащищённости, особенно, если говорить о передаче аналоговых сигналов на большие расстояния. Тем не менее, можно найти ряд специфических за-

дач, при решении которых, если не гнаться за повышенной точностью, аналоговые вычислители оставляют цифровые электронные компьютеры далеко позади.

Напомним, что понятие «аналоговый» довольно широко и оно включает в себя как электронные устройства, в которых течёт электрический ток, так и те, где электронами и не пахнет[101].

Наиболее кратко о том, что такое аналоговый вычислитель написано в [БСЭ т.1 М.:Издательство «Советская энциклопедия», 1970 с.568-569]:

Аналоговая вычислительная машина (АВМ), вычислительная машина, в которой каждому мгновенному значению переменной величины, участвующей в исходных соотношениях, ставится в соответствие мгновенное значение другой (машинной) величины, часто отличающейся от исходной физической природой и масштабным коэффициентом. Каждой элементарной математической операции над машинными величинами, как правило, соответствует некоторый физический закон, устанавливающий математические зависимости между физическими величинами на выходе и входе решающего элемента (например, законы Ома и Кирхгофа для электрических цепей, выражение для эффекта Холла, лоренцовой силы и т. д.).

Особенности представления исходных величин и построения отдельных решающих элементов в значительной мере предопределяют сравнительно большую скорость работы АВМ, простоту программирования и набора задач, ограничивая, однако, область применения и точность получаемого результата. АВМ отличается также малой универсальностью (алгоритмическая ограниченность) при переходе от решения задач одного класса к другому требуется изменять структуру машины и число решающих элементов.

К первому аналоговому вычислительному устройству относят обычно логарифмическую линейку, появившуюся около 1600 г.

Графики и номограммы – это следующая разновидность аналоговых вычислительных устройств созданных для определения функций нескольких переменных; впервые встречаются в руководствах по навигации в 1791 г. В 1814 году английский учёный Дж. Герман разработал аналоговый прибор планиметр, предназначенный для определения площади, ограниченной замкнутой кривой на плоскости. Планиметр был усовершенствован в 1854 году немецким учёным А. Амслером. Его интегрирующий прибор с катящимся колесом привёл позднее к изобретению английским физиком Дж. Томсоном фрикционного интегратора. В 1876 году другой английский физик У. Томсон применил фрикционный интегратор в проекте гармонического анализатора для анализа и предсказывания высоты приливов в различных портах. Он показал в принципе возможность решения дифференциальных уравнений путём соединения нескольких интеграторов, однако из-за низкого уровня техники того времени идея не была реализована.

Первая механическая вычислительная машина для решения дифференциальных уравнений при проектировании кораблей была построена А.Н. Крыловым в 1904 году. В основу её была положена идея интеграфа – аналогового интегрирующего прибора, разработанного польским математиком Абданк-Абакановичем (1878) для получения интеграла произвольной функции, вычерченной на плоском графике.

Дальнейшее развитие механических интегрирующих машин связано с работами американского учёного В. Буша, под руководством которого была создана чисто механическая интегрирующая машина (1931), а затем её электромеханический вариант (1942). В 1936 году русский инженер Н. Минорский предложил идею электродинамического аналога. Толчок развитию современных АВМ постоянного тока дала разработка Б. Расселом (1942–44 гг., США) решающего усилителя.

Большое значение имели работы советского математика С.А. Гершгорина (1927), заложившие основы построения сеточных моделей. В 1936 в СССР под руководством И.С. Брука были построены механический интегратор и электрический расчётный стол для определения стационарных режимов энергетических систем. В 40-х гг. была начата разработка электро-

[101] В смысле электрической проводимости, а не наличия электронов у атомов.

механического ПУАЗО[102] на переменном токе и первых электронных ламповых интеграторов (Л.И. Гутенмахер). Работы, проведённые под руководством Гутенмахера (1945–46 гг.), привели к созданию первых электронных аналоговых машин с повторением решения. В 1949 году в СССР под руководством В.Б. Ушакова, В.А. Трапезникова, В.А. Котельникова, С.А. Лебедева был построен ряд АВМ на постоянном токе. Эти работы положили начало развитию современной аналоговой вычислительной техники в СССР.

АВМ в основном применяется при решении следующих задач. Контроль и управление. В системах автоматического управления АВМ пользуются, как правило, для определения или формирования закона управления, для вычисления сводных параметров процесса (кпд, мощность, производительность и др.). Если задано математическое выражение, определяющее связь сводного параметра или управляющего воздействия с координатами объекта, АВМ служат для решения соответствующего уравнения. Результат вычислений поступает либо на исполнительный механизм (замкнутая система), либо к оператору. В последнем случае АВМ работает как информационное устройство. Например, АВМ широко распространены для оценки экономической эффективности энергетических систем, и те же АВМ могут управлять исполнительными механизмами, т. е. служить автоматическими регуляторами. Когда закон управления заранее не определён, а заданы лишь некоторый критерий оптимальности и граничные условия, АВМ применяются в системах поиска оптимального управления и служат математической моделью объекта.

Претендовать на полноту изложения вопроса аналоговых вычислений мы не будем, но для понимания сути приведём отдельные примеры некоторых схем для выполнения аналоговых вычислений.

Одно из простейших вычислительных устройств такого рода – сумматор, осуществляющий сложение двух заданных чисел. Его принципиальная схема приведена на рисунке 4.9. В нём используется известная закономерность: сила тока в неразветвлённой части электрической цепи равна сумме токов в отдельных ветвях этой цепи. С помощью реостатов R1 и R2 устанавливают, а амперметрами PA1 и PA2 соответственно контролируют силы тока (в миллиамперах), численно равные значениям первого и второго слагаемого. Амперметр PA3 при этом показывает их сумму. [24, стр.124]

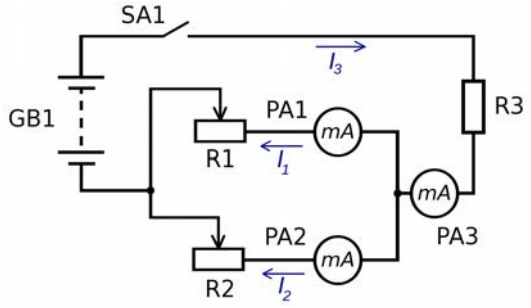


Рисунок 4.9. Аналоговый сумматор тока ($I_3 = I_1+I_2$)

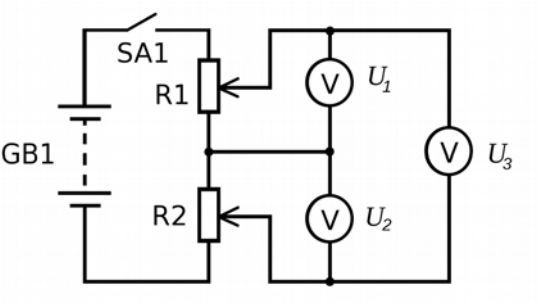
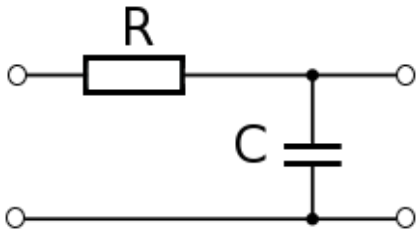


Рисунок 4.10. Аналоговый сумматор напряжения ($U_3 = U_1+U_2$)

Замечание. Указанная схема работоспособна, но не удачна, изменение сопротивления одного резистора незначительно, но влияет на ток в цепи другого. Для выставления значений слагаемых придётся «крутить» обе ручки одновременно, либо устанавливать нужное значение тока в несколько итераций. Схема сумматора на вольтметрах [24, стр 125] см. рис.4.10 в силу высокого внутреннего сопротивления последних лишена этого недостатка.

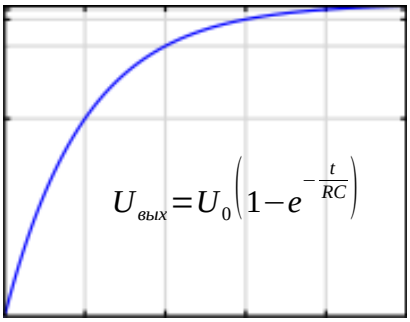
[102] Прибор управления артиллерийским зенитным огнём.

Простая интегрирующая RC-цепь выглядит следующим образом:



Входной сигнал (напряжение) подаётся на клеммы слева, результат снимается – справа.

Реакция такой цепи на единичное ступенчатое воздействие постоянного напряжения с амплитудой V_0 определяется следующей формулой:



Синим цветом показано выходное напряжение $V_{вых}$

На сегодняшний день широкое применение для аналоговых вычислений получили схемы на операционных усилителях, вот несколько простых схем.

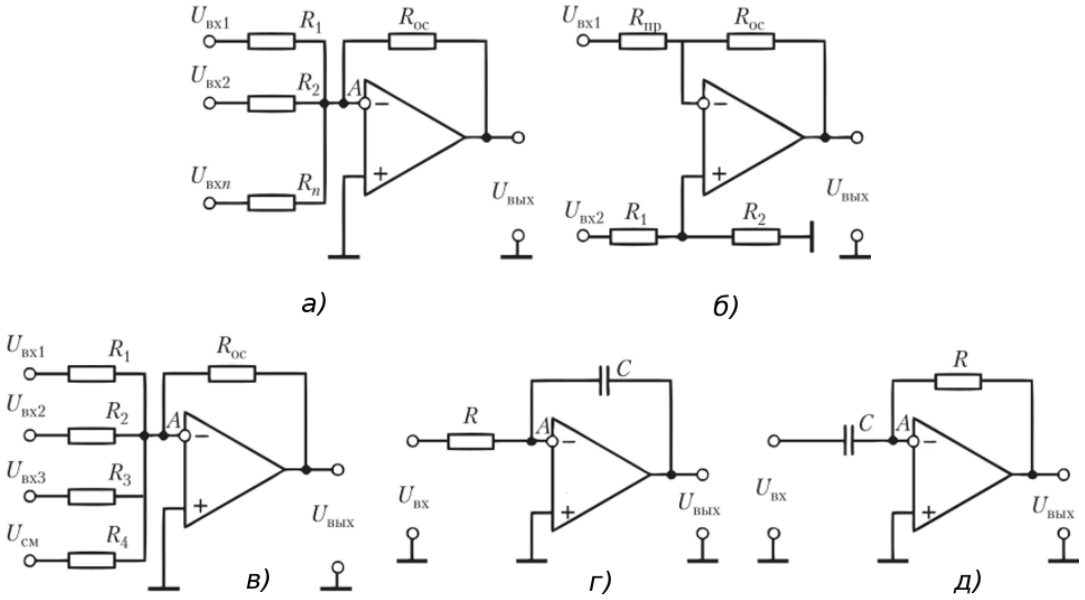
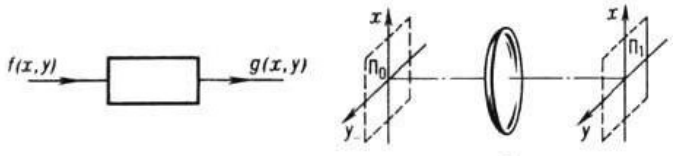


Рисунок 4.11. Схемы выполненные на ОУ: а) суммирования, б) вычитания, в) сумматора трёх напряжений со смещением, г) интегрирования, д) дифференцирования.

В группу аналоговых вычислителей несомненно можно отнести и всевозможные оптические системы, которые могут являться ключевыми элементами для выполнения отдельных математически сложных (с вычислительной точки зрения) операций. Обыкновенная лупа (линза) в силу своих оптических свойств выполняет преобразование Фурье.



Остаётся лишь поместить в её фокусе оптические датчики освещённости и снимать с них готовые показания. Любой школьник старших классов без труда назовёт ещё несколько несложных оптических приборов, пригодных для обсуждаемых целей: оптическая призма, полупрозрачное зеркало, зонная пластина и т.п.

Как можно заметить, термины «компьютер», АВМ и ЭВМ синонимичны, но не идентичны.

Развитие механических вычислительных устройств, относимым к бытовым, до последних дней прошлого века шло по пути разумного увеличения их сложности в параллель с аналоговыми и цифровыми электронными счётными устройствами, после чего соотношение «вычислительная мощность × надёжность / цена» у последних взяло вверх, сделав производство всех остальных типов счётных устройств нерентабельным. На этот процесс также повлияли миниатюризация новой техники, её удобность (лучшая эргономичность) и расширенная функциональность.

4.3. Поколения цифровых ЭВМ

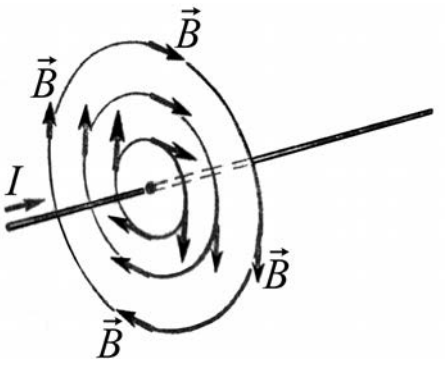
Для повышения точности (а не производительности, хотя она в последующем стала так же расти) потребовалось создавать более сложные механические системы и механизмы. Отчасти это был тупиковый путь, поэтому в различных устройствах стали уходить в сторону унификации, для чего использовать отдельные узловые элементы соединённые электрически. Где-то наоборот, соединяли воедино различные по принципу работы вычислительные блоки, получая при этом вычислительные устройства смешанного типа.

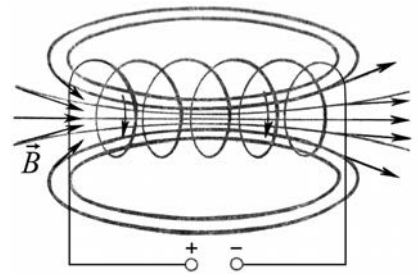
Появились шаговые двигатели, шаговые искатели, схемы с реле и их различные комбинации, – ключевые элементы построения вычислительных техники того времени.

Вряд ли люди думали, что созданные ими относительно простые (по сегодняшним меркам) электрические схемы в будущем можно будет назвать громким словом «ЭВМ», а аналоговая вычислительная составляющая в них будет сведена на нет, поэтому в истории этот этап часто опускается, но мы его назовём нулевым поколением цифровых ЭВМ.

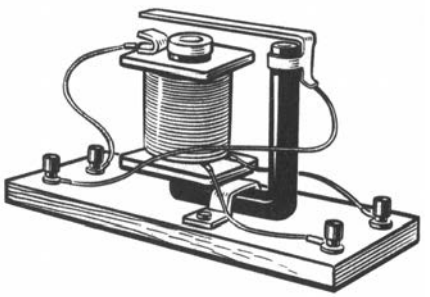
0-е поколение ЭВМ	Электромагнитные реле и механизмы

Физика работы «релейных компьютеров» должна быть понятна читателям из школьного курса общей физики. Кратко она следующая: в 1820 году Хансом Кристианом Эрстедом было обнаружено, что проволока, по которой протекал электрический ток, влияла на магнитную стрелку компаса, [13]. На основании этого было сделано предположение, что проводник с током является источником магнитного поля, которое распределяется вокруг него следующим образом, [14].





Развивая мысль далее, если провод намотать на что-либо цилиндрическое, например на карандаш, получится *соленоид* – однослойная катушка цилиндрической формы, витки которой намотаны вплотную, а длина значительно больше диаметра. Суммарное магнитное поле от соленоида, согласно закону Био-Савара-Лапласа, сформируется как суперпозиция полей от различных небольших участков проводника и, как следствие, распределится следующим образом: (см. выше)



Для усиления магнитного эффекта витки можно намотать в несколько слоёв, а внутрь получившейся катушки вставить стальной сердечник. Тогда при пропускании электрического тока к сердечнику может притягиваться другая железка, которая будет создавать контакт, таким образом с помощью пропускания электрического тока по одной цепи можно будет управлять другой. Ниже приведено изображение самодельного реле.

Если же изготовить реле заводским способом, то оно обычно выглядит примерно так:

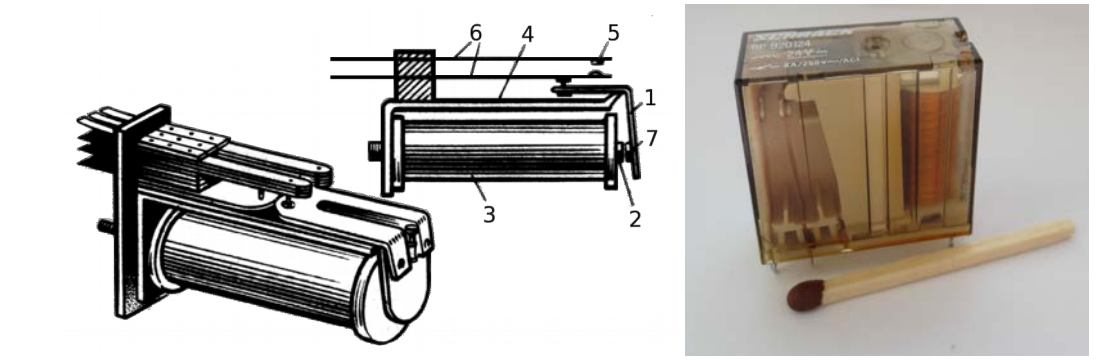


Рисунок 4.12. Электромагнитное механическое реле

У такого реле якорь 1 притягивается к сердечнику электромагнита 2 под действием магнитного поля, создаваемого обмоткой (катушкой) 3, по которой протекает электрический ток. Поворачиваясь на призме ярма 4, якорь замыкает контакты 5. При отключении тока в обмотке якорь отходит от сердечника под действием контактных пружин 6. Латунный штифт 7 предотвращает при этом залипание якоря.

В электромагнитных реле устанавливают контактные пружины не только с замыкающими (нормально разомкнутыми) контактами. Контакты могут быть и размыкающими (нормально замкнутые), и переключающими. Различные реле и сегодня повсеместно используются в бытовой технике, автомобилях и прочем.

2-й закон Ньютона: Ускорение, приобретаемое телами, прямо пропорционально сумме действующих сил на него и обратно пропорционального их массе:

$$\vec{a} = \frac{\sum \vec{F}}{m} \quad .$$

Компьютеру, использующему в качестве ключевых элементов реле, будут присущи следующие недостатки: низкая скорость работы и необходимость частого проведения профилактических работ. Любая «механика», согласно второму закону Ньютона, обладает инерцией. Также контакты имеют тенденцию окисляться, и их надо чистить... а если использовать благородные металлы, то они или очень дороги, или относительно мягкие, или не оптимальны с точки зрения внутреннего сопротивления.

Пример механического компьютера на реле см. на *http://habrahabr.ru/post/220865/*.

1-е поколение ЭВМ	Электронные лампы (~до 1955)

Следующим важным историческим событием стало открытие эффекта термоэлектронной эмиссии (явление испускания электронов нагретыми телами), что позволило создать не только вакуумный диод, но и различные электронные лампы (см. рис. 4.13).

Рисунок 4.13. Эволюция ключевых элементов ЭВМ в масштабе по отношению друг к другу (слева октальная лампа, справа две пальчиковые лампы и набор транзисторов)

Интересное: Название первого поколения ламп – «октальные» – подразумевало использование стандартного электрического разъёма с 8 контактами (часть могла не использоваться или отсутствовать) – октальный цоколь. Следующим поколением на пути миниатюризации были «пальчиковые лампы», получившие своё название из-за их продолговатой цилиндрической формы и относительно малых размеров (диаметр баллона около 2 см, а высота не более 8 см).

Лампы потребляли больше электричества, нежели реле, грелись и быстро выходили из строя, зато скорость их работы была значительно выше, также они позволили создавать более сложные схемы (в том числе и аналоговые[103]), что знаменовало новое поколение ЭВМ.

[103] Небольшие схемы, выполняющие различные арифметические операции, впрочем, как и более сложные тоже, например интегрирование или дифференцирование, давно существуют, а значит, и «аналоговые ЭВМ» имеют право на существование. Аналоговые операции выполняются на порядок быстрее, но не с очень высокой точностью. Исторически «информатика» не рассматривает такие устройства, отдавая их предпочтительно курсу электротехники.

В 1918 году советский учёный Михаил Александрович Бонч-Бруевич изобрёл ламповый триггер [15, стр. 26] (схему переключающего устройства, имеющего два устойчивых рабочих состояния, подробнее про триггеры см. *раздел 4.1.6*), который впоследствии сыграл важную роль в построении электронных цифровых компьютеров.

Позднее, в 1919 г. [15, стр. 26], У. Икклз и Ф. Джордан (США) независимо от Бонч-Бруевича изобрели электронное реле (flip-flop), имеющее также два состояния и получившее позже название триггера.

Физика работы лампы следующая [16, 17]. Представьте обычную лампочку, из которой откачан воздух. В результате нагрева нити накала [104] (например, с помощью с помощью тока от батареи накала, на схеме обозначена как $GB_н$ от *Galvanic Battery*) в ней, как во всяком нагретом металле, сильно активируется хаотическое движение свободных электронов. Многие электроны до того «разбегаются» в металле, что по инерции выскакивают из нити, оказываясь в свободном пространстве (в вакууме, созданном в лампе). Собственно, это и называется термоэлектронной эмиссией. Правда, далеко свободные электроны улететь не могут: у нити накала, которую они покинули, образуется положительный заряд, не дающий далеко электронам улететь. По сути, нить накала, назовём её теперь уже катодом, тянет их обратно. Суммарный заряд катода (нити) ведь остаётся постоянным, а электроны заряжены отрицательно. Как следствие вокруг раскалённого катода существует этакое облачко из электронов, уже вылетевших, но ещё не успевших упасть обратно на катод. Всю эту картину можно сравнить с фонтаном в парке: выброшенная вверх струя воды довольно быстро падает под действием своей силы тяжести, но над фонтаном всё время стоит столб воды, уже поднявшейся и ещё не успевшей упасть.

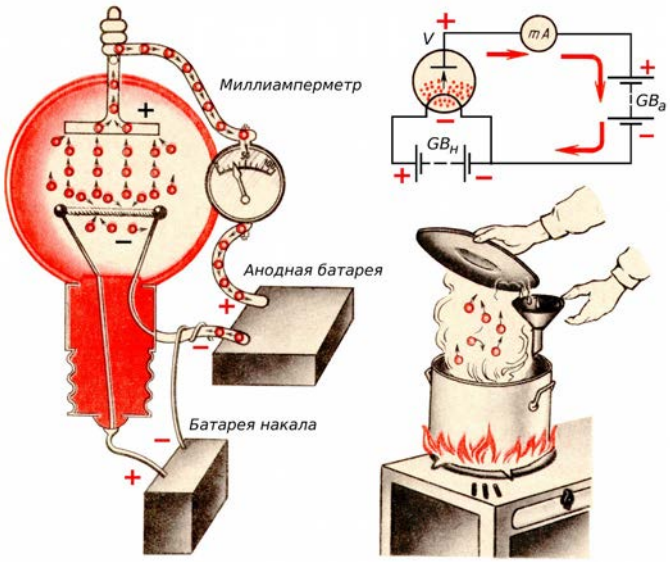


Рисунок 4.14. Принцип работы электронной лампы [16]

Замечание. В некоторых лампах катод и нить накала совпадают. По сути, катодом оказывается тонкая металлическая нить, которую нагревают, пропуская по ней ток, как в нашем случае. Но чаще катод – это металлическая трубочка, внутрь которой опять-таки вставлена нить-подогреватель. Разогрев катода много потребляет, поэтому различными способами пытаются уменьшить затраты энергии на него. Например, в современных лампах катод активируют, покрывают тончайшем слоем специального одноатомного вещества, которое облегчает выход электронов при разогреве. Благодаря этому активированные катоды работают при температурах

[104] В некоторых лампах катод и нить накала совпадают. См. замечание.

1000 °C вместо 2500 °C в чисто металлических катодах. Правда, активированный катод – сооружение довольно нежное, не терпит перегрева, не терпит превышений напряжения накала [17].

Введём в нашу лампочку ещё один электрод сверху и назовём его «анод». Теперь если к нему и катоду подключить батарею (анодную батарею, GB_a), в общем случае источник постоянного анодного напряжения, то положительный заряд анода будет притягивать к себе отрицательно заряженные электроны из «облачка» и начнётся движение электронов в вакууме, как это изображено на рис. 4.14 от катода к аноду. «*Направленное движение заряженных частиц*» по определению и есть ток, в нашем случае назовём его анодным током и, течь он будет в обратном направлении, потому как двигающиеся электроны заряжены отрицательно, а договорились считать, что ток течёт от «+» к «−», то есть от анода к катоду.

Если повернуть анодную батарею и подать на анод «минус», то никакого тока не будет. Электроны будут от него только отталкиваться. Как видите, двухэлектродная лампа – электровакуумный диод – обладает односторонней проводимостью, как и полупроводниковый диод, знакомый большинству из школьного курса физики. Больше того, в вакуумном диоде нет собственных свободных носителей зарядов, и обратного тока в лампе вообще нет.

Следующий шаг – введём в лампу ещё один, третий электрод, так называемую управляющую сетку, и поставим её на пути анодного тока. (см. рис. 4.15.) Название сетка идёт с далёких времён, когда сетка была действительно сеткой; в современных лампах это спираль, окружающая катод на небольшом расстоянии от него. (см. рис. 4.16.)

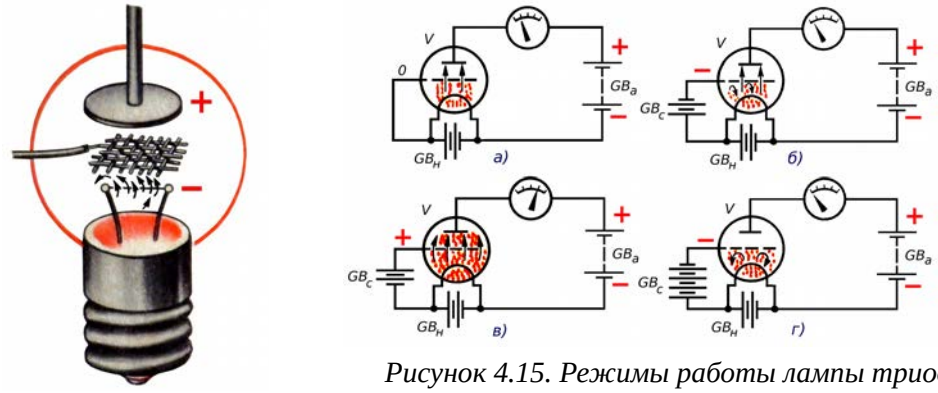


Рисунок 4.15. Режимы работы лампы триода

Управляющая сетка превращает диод в трёхэлектродную усилительную лампу – триод. Когда появились первые транзисторы, их называли «полупроводниковыми триодами» по аналогии, ведь у него тоже три электрода, три рабочие зоны – эмиттер, база, коллектор.

Сетка расположена близко к катоду, и напряжение на ней очень сильно влияет на анодный ток – «плюс» на сетке (см. рис. 4.15 в) подтягивает электроны, увеличивает число свободных электронов, вырвавшихся из облачка вблизи катода и отправившихся в далёкое путешествие к аноду. «Минус» на сетке (см. рис. 4.15 б, г), наоборот, отталкивает электроны к катоду, уменьшает анодный ток. Одним словом, не относительно к ЭВМ, усиливаемый сигнал $U_{вх}$, действуя с командного пункта трёхэлектродной лампы, с сетки, управляет анодным током I_a, а тот, проходя по нагрузке (в нашем случае нагрузкой будет прибор со стрелкой), выделяет в ней мощную копию усиливаемого сигнала.

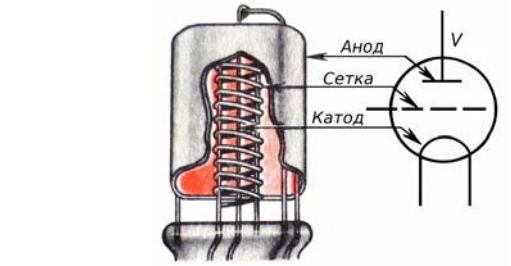


Рисунок 4.16. Электровакуумная лампа триод и её схематическое обозначение [16]

В некоторой высококачественной звуковой технике электронные лампы можно встретить и по сей день, а также найти фирмы, изготавливающие лампы в небольших количествах.

Замечание. Лампа есть прибор, управляемый напряжением, в отличие от транзистора, управляемого током.

Фантазия людей никогда не останавливалась, поэтому были напридуманы различные лампы с большим количеством электродов: «пентод», «лучевой тетрод» и др., но мы их рассматривать не будем, так как нам важно понимание устройства единичного ключевого элемента, которым определялось поколение ЭВМ. Основная цель – показать читателям, что компьютер не есть что-то божественное и неизвестно как работающее, а набор очень большого числа довольно простых элементов, работа которых подчинена простым физическим принципам.

Во временной период этого поколения также произошли следующие знаковые события, в том числе и в области аналоговых вычислений:

1938 год – немецкий инженер Конрад Цузе вскоре после окончания в 1935 году Берлинского политехнического института построил свою первую машину, названную Z1. (В качестве его соавтора упоминается также Гельмут Шрейер (нем. Helmut Schreyer)). Это полностью механическая программируемая цифровая машина. Модель была пробной и в практической работе не использовалась. Её восстановленная версия хранится в Немецком техническом музее в Берлине. В том же году Цузе приступил к созданию машины Z2[105].

1941 год – Конрад Цузе создаёт первую вычислительную машину Z3, обладающую всеми свойствами современного компьютера.

1942 год – в Университете штата Айова Джон Атанасов и его аспирант Клиффорд Берри (англ. Clifford Berry) создали (а точнее – разработали и начали монтировать) первый в США электронный цифровой компьютер ABC. Хотя эта машина так и не была завершена (Атанасов ушёл в действующую армию), она, как пишут историки, оказала большое влияние на Джона Мокли, создавшего двумя годами позже ЭВМ ЭНИАК.

Конец 1943 года – заработала британская вычислительная машина специального назначения Colossus. Машина работала над расшифровкой зашифрованных с помощью Энигмы сообщений фашистской Германии.

Февраль 1944 года – группой американских инженеров под руководством Говарда Эйкена закончена разработка первой американской вычислительной машины Марк I. После монтажа, наладки и испытаний она стала использоваться для выполнения сложных баллистических расчётов американского ВМФ.

1944 год – Конрад Цузе разработал ещё более быстрый компьютер Z4, а также первый язык программирования высокого уровня Планкалкюль.

[105] Сначала эти компьютеры назывались V1 и V2. По-немецки это звучит «Фау-1» и «Фау-2» и чтобы их не путали с ракетами, компьютеры переименовали в Z1 и Z2.

1946 год – публике представлена первая универсальная электронная цифровая вычислительная машина ЭНИАК, разрабатывавшаяся секретно с 1943 года.

4 декабря 1948 года – Государственный комитет Совета министров СССР по внедрению передовой техники в народное хозяйство зарегистрировал за номером 10475 изобретение И.С. Бруком и Б.И. Рамеевым цифровой электронной вычислительной машины.

1950 год – группой С.А. Лебедева из Киевского института электротехники СССР создана первая советская ЭВМ МЭСМ (Малая электронная счётная машина). Она содержала около 6000 электровакуумных ламп и потребляла 15 кВт. Машина могла выполнять около 3000 операций в секунду.

1953 год – на московском заводе счётно-аналитических машин начато производство первой советской серийной ЭВМ «Стрела». Её относят к классу больших универсальных ЭВМ с трёхадресной системой команд и она обладала быстродействием более 2000 операций в секунду с оперативной памятью 2048 43-разрядных слов (время обращения 20 мкс). Машина состояла из 6200 ламп, 60 000 полупроводниковых диодов и потребляла 150 кВт энергии, половина из которой уходила на создание охлаждения.

Замечание. В 1954 году Сталинской премии были удостоены разработчики «Стрелы»: В.В. Александров (8), Ю.Я. Базилевский (9), Д.А. Жучков (10), И.Ф. Лыгин (6), Г.Я. Марков (5), Б.Ф. Мельников (4), Г.М. Прокудаев (3), Б.И. Рамеев (7), Н.Б. Трубников (2), А.П. Цыганкин (11), Ю.Ф. Щербаков (1), Л.А. Ларионова. Главному конструктору машины Ю.Я. Базилевскому также присвоено звание Героя Социалистического Труда.

2-е поколение ЭВМ	Транзистор (1955–1964)

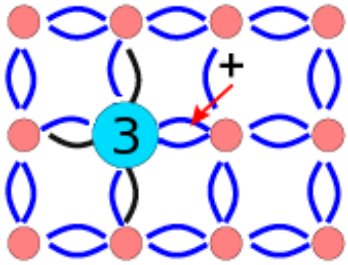
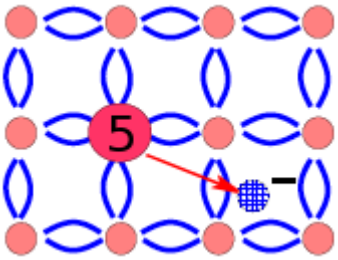
С открытием полупроводников лампы и реле заменили транзисторы. Первым был изобретён полупроводниковый диод (который использовался ещё в первом поколении ЭВМ), а после – биполярный транзистор и, немногим позднее, полевой транзистор.

Полупроводни́к[106] – материал, который по своей удельной проводимости занимает промежуточное место между проводниками и диэлектриками и отличается от проводников сильной зависимостью удельной проводимости от концентрации примесей, температуры и воздействия различных видов излучения. Материалы обладающие свойствами полупроводника: Si, Ge, AsGa, InP, InSb. Самый распространённый и доступный (а как следствие – дешёвый) из них – кремний.

[106] В современной науке физические свойства полупроводников хорошо изучены. Большое количество открытых эффектов (которые не могут быть наблюдаемы ни в проводниках, ни в диэлектриках) прежде всего связанно с устройством зонной структуры полупроводников и наличием достаточно узкой запрещённой зоны. Основной стимул для изучения полупроводников (в первую очередь кремния, а затем и других) – производство полупроводниковых приборов и интегральных микросхем. Они применяются в вычислительной и бытовой технике, освещении, лазерах, ПЗС-матрицах цифровых фото- и видеокамер и т.п.

Физика полупроводников с примесной проводимостью следующая. Атомы полупроводника являются 4-валентными и организуют ровную кристаллическую решётку. При внесении примесей с атомами трёхвалентного или пятивалентного химического элемента появляются новые свойства: дырочные полупроводники (*p*-типа) или электронные полупроводники (*n*-типа). См. таблицу 4.1.

Таблица 4.1. Типы полупроводников

Дырочные, *p*-типа, 3-валентные примеси	Электронные, *n*-типа, 5-валентные примеси
	
Термин «*p*-тип» происходит от слова «positive», обозначающего положительный заряд основных носителей. Этот вид полупроводников, кроме примесной основы, характеризуется дырочной природой проводимости. В четырёхвалентный полупроводник (например, в кремний) добавляют небольшое количество атомов трёхвалентного элемента (например, индия). Каждый атом примеси устанавливает ковалентную связь с тремя соседними атомами кремния. Для установки связи с четвёртым атомом кремния у атома индия нет валентного электрона, поэтому он захватывает валентный электрон из ковалентной связи между соседними атомами кремния и становится отрицательно заряженным ионом, вследствие чего образуется дырка. Примеси, которые добавляют в этом случае, называются ***акцепторными***	Термин «*n*-тип» происходит от слова «negative», обозначающего отрицательный заряд основных носителей. Этот вид полупроводников имеет примесную природу. В четырёхвалентный полупроводник (например, кремний) добавляют примесь пятивалентного полупроводника (например, мышьяка). В процессе взаимодействия каждый атом примеси вступает в ковалентную связь с атомами кремния. Однако для пятого электрона атома мышьяка нет места в насыщенных валентных связях, и он переходит на дальнюю электронную оболочку. Там для отрыва электрона от атома нужно меньшее количество энергии. Электрон отрывается и превращается в свободный. В данном случае перенос заряда осуществляется электроном, а не дыркой, то есть данный вид полупроводников проводит электрический ток подобно металлам. Примеси, которые добавляют в полупроводники, вследствие чего они превращаются в полупроводники *n*-типа, называются ***донорными***

Самое интересное начинается, на стыке двух полупроводников разных типов, поскольку в нём образуется так называемый «гомопереход» (см. рис. 4.17).

Во-первых, за счёт диффузии образуется контактная разность потенциалов и, как следствие, диффузионное электрическое поле, во-вторых, указанный переход начинает обладать новыми электрическими свойствами, которых не было у двух полупроводников разных типов по отдельности. Одна асимметричная вольт-амперная характеристика (ВАХ) *p-n*-перехода чего стоит! Приводить мы её не будем, лишь скажем, что в зависимости от рабочей области этой самой ВАХ одно и тоже устройство с *p-n*-переходом может называться по-разному: выпрямительный диод, стабилитрон, варикап, импульсный диод.

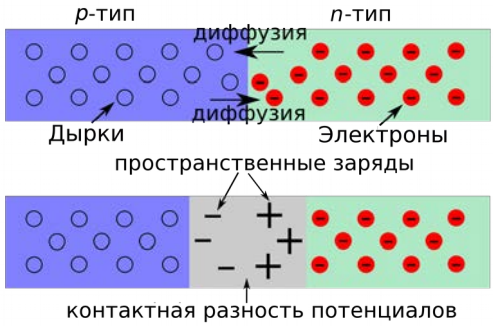


Рисунок 4.17. Гомопереход на стыке двух полупроводников разного типа

Выпрямительное свойство диода (в одну сторону пропускать, а в другую – нет) должно быть известно большинству читателей, так как рассматривается в школьном курсе физики. Помимо непрозрачных корпусов, могут иметь исполнение с прозрачным корпусом, через который даже видно небольшую пластинку полупроводника. (см. рис. 4.18).

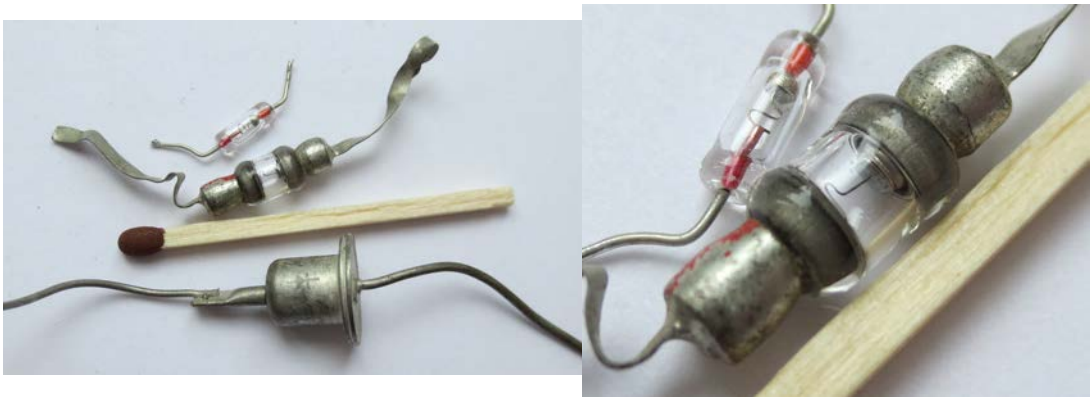


Рисунок 4.18. Выпрямительные диоды, через стеклянную колбу виден полупроводник

Следующим этапом экспериментов с полупроводниками стало их соединение в некоторый «трёхслойных бутерброд». В результате получился транзистор. Схемотехническое обозначение биполярного транзистора приведено на рис. 4.19.

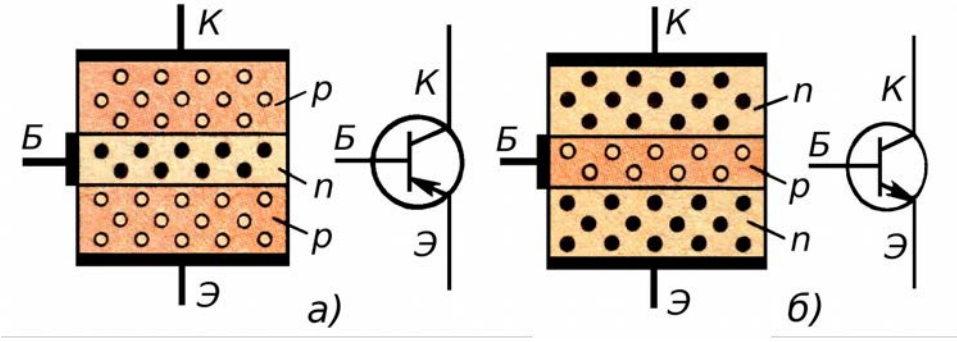


Рисунок 4.19. Устройство биполярного транзистора [16] (у реальных транзисторов зона базы намного уже, на данном рисунке она сильно преувеличена для наглядности трёхслойного построения рисунка)

Выводы транзистора обозначаются по первым буквам слов «база», «коллектор» и «эмиттер» (у другого вида транзисторов – полевых – «исток», «сток» и «затвор»).

Первые отдельные транзисторы изготавливались по сплавной технологии и выглядели следующим образом. (см. рис. 4.20)

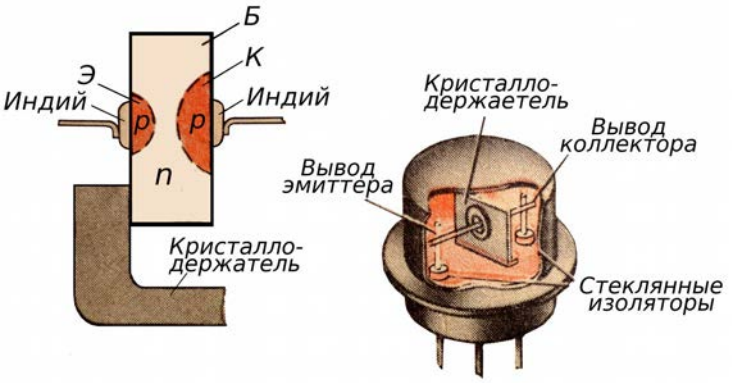


Рисунок 4.20. Устройство и конструкция сплавного транзистора структуры p-n-p [16]

Рисунок 4.21. Фотография транзистор со спиленной крышкой

Работу транзистора довольно просто представить, как её описывают П. Хоровиц и У. Хилл: «транзисторный человек» следит за током базы и регулирует выходной реостат для того, чтобы выходной ток был в $h_{21э}$ раз больше тока базы [1] (см. рис. 4.22).

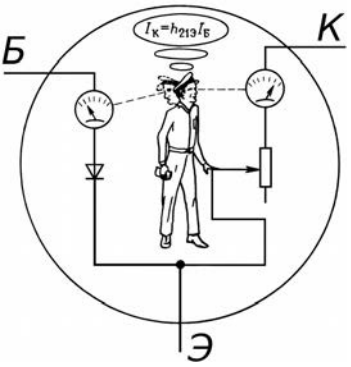


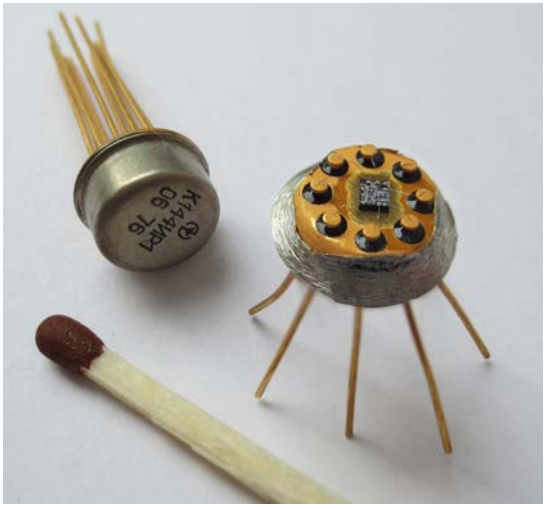
Рисунок 4.22. Внутреннее устройство транзистора [1]

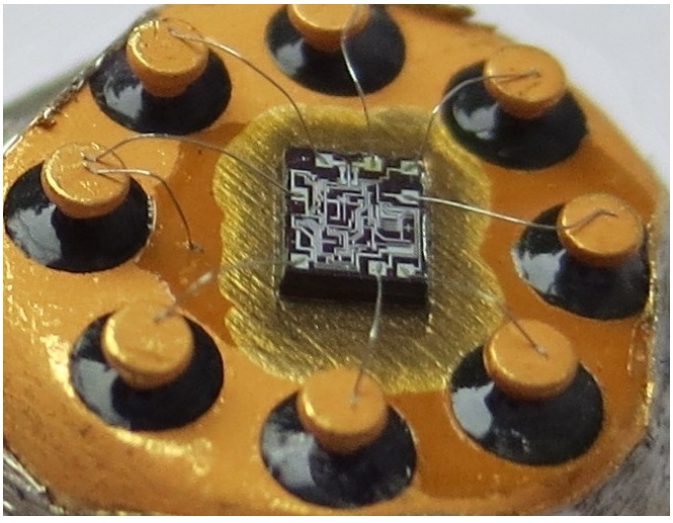
С физической точки зрения происходит следующее. База у транзистора настолько тонка, что дырка или электрон, «пролетев» база-эмиттерный переход, не успевают в области базы задержаться и по инерции залетают в следующую зону база-коллекторного перехода, где подпадают под действие поля, созданного напряжением на коллекторе. Чем-то это напоминает то, как электроны пролетают мимо сетки в электронной лампе.

Современные транзисторы имеют другую технологию изготовления и набор более современных корпусов, на внутренней работе это никак не сказывается. Транзистор стал важным ключевым элементом построения ЭВМ и, по сути, определил развитие ЭВМ не только во 2-м поколении, но и в 3-м и 4-м поколениях.

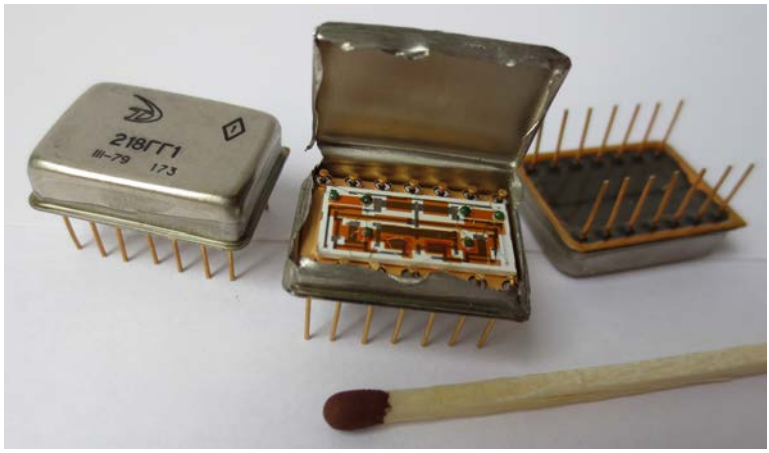
3-е поколение ЭВМ	Интегральные схемы (ИС) (1964–1972)

Через некоторое время после создания транзисторов промышленность пришла к тому, что эффективнее в одном корпусе (и на одном кристалле) изготавливать несколько транзисторов, так называемые транзисторные сборки. Постепенно пошли размышления о том, чтобы разместить на кристалле и другие элементы – сопротивления, ёмкости и прочие. Теперь уже внутри корпуса устройства могла поместиться целая небольшая схема, которую так и назвали микросхемой. А следующее поколение ЭВМ получило основу – интегральные схемы.





(продолжение рисунка с фотографиями см. на следующей странице)



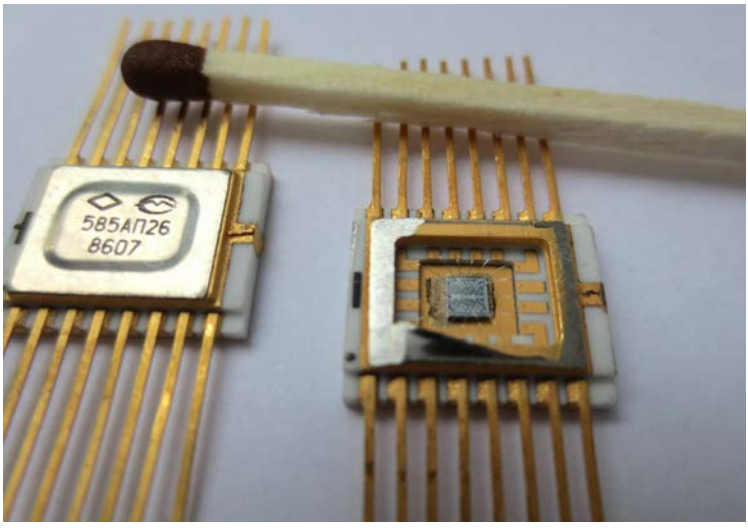


Рисунок 4.23. Внутреннее устройство первых микросхем

4-е поколение ЭВМ	Большие ИС (БИС), сверхбольшие ИС (СБИС) (1972 по наст. время)

В дальнейшем плотность элементов в интегральном исполнении значительно увеличилась и продолжает увеличиться до сих пор за счёт уменьшения технологического процесса (подробнее см. раздел 4.5 *«Процессор»*). В частности, научились выполнять схемы со свойствами индуктивности и другие хитрости. Довольно быстро проскочил этап «больших интегральных схем», перейдя к «сверхбольшим интегральным схемам» (см. рис. 4.24). Сказать, где точно проходит граница, по числу транзисторов и годам довольно сложно, да и не особо нужно, так как оба вида микросхем относят ЭВМ к 4-му поколению, которое существует и по сей день. Даже удивляющие в 2021-м году возможности компьютеров реализованные с помощью искусственного интеллекта на базе нейронных сетей и машинного обучения основаны на производительных компьютерах 4-го поколения объединённых единой сетью.

Возможно в будущем будет использоваться иная классификация поколений.

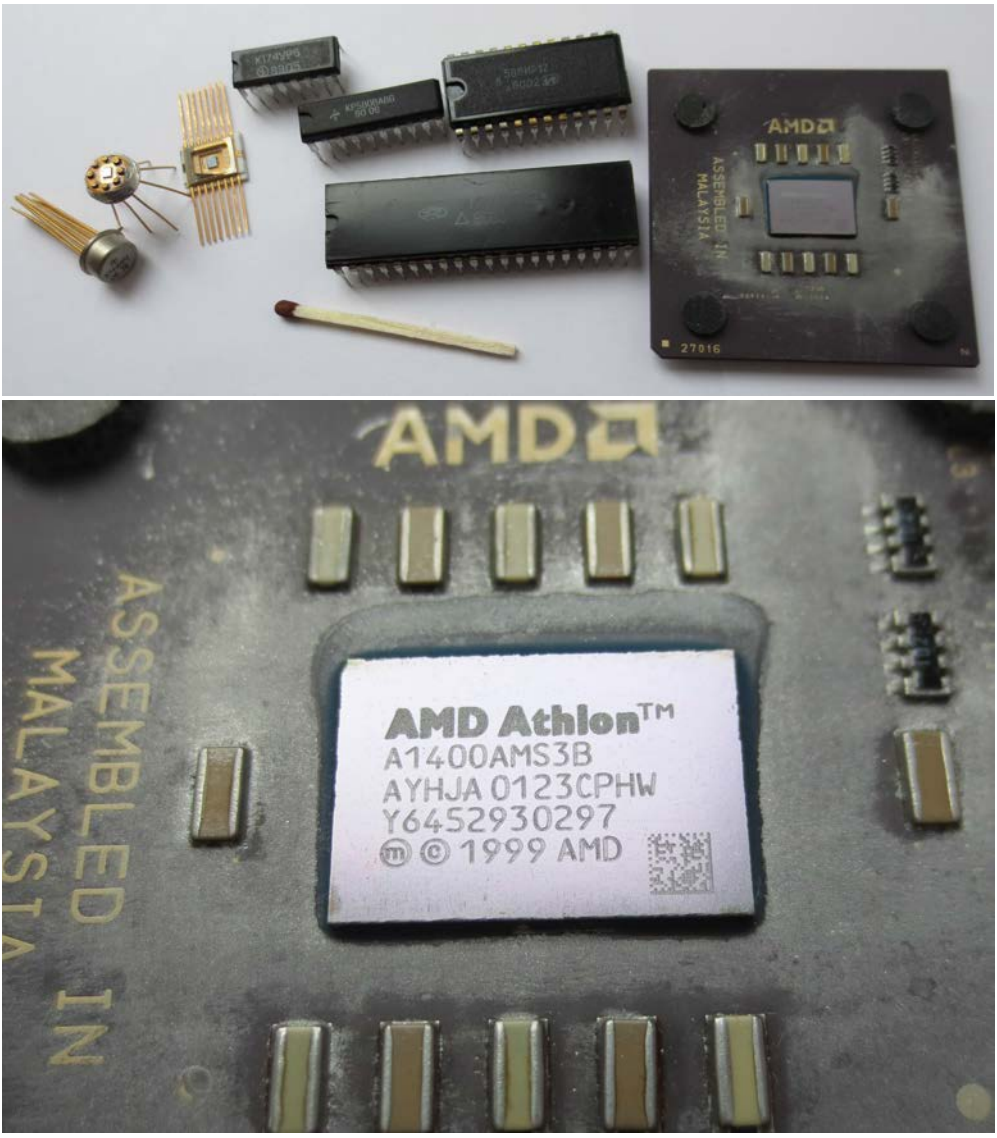


Рисунок 4.24. Переход от «малых» микросхем к БИС и СБИС

5-е поколение ЭВМ	??? г. – квантовые или оптические компьютеры

Когда придёт 5-е поколение ЭВМ? Каким оно будет? Будет ли это поколение основано на использовании кубитов? Какого объёма достигнет память на мемристорах? Какие новые способности появятся у компьютеров в ближайшие 10–20 лет? На эти вопросы можно дать только предположительный ответ, и то мы не можем быть уверены в нём. Именно эта неопределённость делает такой притягательной компьютерную технику.

Живите настоящим, ни одна железка не может принести столько радости, сколько живой человек рядом с вами!

4.4. Другие компьютеры (вычислители)

Ранее были рассмотрены аналоговые вычислители и цифровые ЭВМ, вплоть до современных. В последних обработка и хранение данных ведётся в двоичном коде. Двоичный код является цифровым видом, а вот цифровой вид может быть основан на другой системе счисления, совсем не обязательно двоичной. Несмотря на то, что «двоичный» ≠ «цифровой», указанные слова часто являются синонимами.

Десятичный код использовался в некоторых механических устройствах. Троичная логика и ЭВМ «Сетунь» также упоминались несколько раз ранее (см. Алфавитный указатель). В заключении к Главе 1 (на стр. 23) была обоснована оптимальность выбора двоичной системы счисления в качестве основы для кодирования разных видов информации в современных ЭВМ, в следующем разделе (4.5. Научные основы работы современной ЭВМ) вы сможете увидеть как это удобство сочетается с работой элементов логики современных ЭВМ, различающих как раз два стабильных состояния, насколько просты часто используемые элементарные операции: сложения, умножения, вычитания, деления, сравнения и др.

Также, позднее (из раздела 4.6. Структура классической ЭВМ) вы узнаете, что идейно принцип обработки данных в двоичном виде был предложен Джоном фон Нейманом более семи десятилетий назад. Некоторые наглядные доказательства этому будут представлены ниже в параграфе «4.5.1. Математические основы работы «чёрного ящика»» (см. стр.204). Однако, прежде чем это произойдёт, пытливый ум читателя наверняка уже задался вопросом, если не о существовании, то хотя бы о теоретических принципах работы иных компьютеров, перечислим вытекающие из них виды:

- пневматические и гидравлические компьютеры;
- оптические компьютеры;
- нанокомпьютеры;
- биокомпьютеры;
- квантовые компьютеры.

Рассмотрим последние подробнее.

4.4.1. Квантовые компьютеры (квантовые вычислители)

Наверно квантовые компьютеры можно отнести к аналоговым вычислителям, но если вспомнить о квантовом дуализме волны и частицы, то, говоря у кубитах, ситуация может повториться.

Для тех, кто хочет быстро и поверхностно узнать что это такое, следует рассмотреть две вещи – что есть кубит (q-бит, qubit, кьюбит от англ. quantum bit)[107], а также рассмотреть задачу оптимального размещения трёх пассажиров в двух такси[108], при условии что пассажиры, как в жизни, кто-то с кем-то дружат (обозначим как «1», 😊), а

[107] Кубит – величина, подобно обычному биту, допускающая два собственных состояния, обозначаемых $|0\rangle$ и $|1\rangle$ (обозначения Дирака), но при этом может находиться и в их суперпозиции, то есть в состоянии $A\times|0\rangle + B\times|1\rangle$, где A и B – комплексные числа, удовлетворяющие условию $|A|^2 + |B|^2 = 1$. Физическая реализация устройства с подобными характеристиками – это отдельный непростой вопрос. Подробнее см. *http://de.wikipedia.org/wiki/Qubit*.

[108] *Сидорова Е.* Просто о сложном: Как работает квантовый компьютер *http://ru.ihodl.com/technologies/2020-11-08/prosto-o-slozhnom-kak-rabotaet-kvantovyj-kompyuter*.

кто-то кого-то недолюбливает (обозначим как «0», ☹), а основная цель размещения их по машинам, – найти такое сочетание пассажиров в машинах, чтобы сделать максимальное число людей счастливыми во время поездки.

Математически эту задачу можно представить как высчитывание максимума некоторой интегративной функции счастья (или удовлетворённости), например

$$max\Big(\ f_{\text{счастья}}\big(\ M\ ,\ 🚖_1\ ,\ 🚖_2\ ,\ 🚖_3\big)\ \Big)\ ,$$

для всех пассажиров одновременно, где аргументами этой функции будут: переменные варианты размещения в машине i – $🚖_i$ и постоянная матрица отношений пассажиров друг к другу – M. В случае трёх пассажиров возможна ситуация «*Девочки, против кого дружим?*», то есть, двое дружат между собой и не любят третьего: 1 и 2 дружат, а пары 1-3 и 2-3 скорее враждуют. Например,

$M =$

отношения	Пассажир 1	Пассажир 2	Пассажир 3
Пассажир 1	—	😊	☹
Пассажир 2	😊	—	☹
Пассажир 3	☹	☹	—

В нашем случае, отношения между пассажирами специально упрощены не только до уровня (0/1, а не 😍, 😡, 😋, 😳 и т.д.), но и направления, пассажиры (люди) симметрично относятся друг к другу.[109]

Матрица получается симметричной относительной её главной диагонали и любая из двух её равных половин явно излишняя.

$M =$

отношения	Пассажир 1	Пассажир 2	Пассажир 3
Пассажир 1	—	😊	☹
Пассажир 2	—	—	☹
Пассажир 3	—	—	—

Когда методом простого и короткого перебора всех вариантов это получится сделать, следует при тех же условиях увеличить количество пассажиров и машин такси, скажем до нескольких сотен. Задача станет нерешаемой простым перебором за приемлемое время, не говоря уже о том, что взаимоотношения между людьми намного сложнее чем два варианта 0 и 1.

Задач, подобных этой, сводимых к вычислениям функций, в том или ином виде и поиску их решений, максимумов или минимумов, в жизни можно найти великое множество. Вопрос только в том, что подобные задачи часто интересны только математикам, как в анекдоте:

Анекдот. Шерлок Холмс и доктор Ватсон летят на воздушном шаре. Попадают в густой туман и теряют ориентацию. Тут небольшой просвет – и они видят на земле человека.

[109] В конкретном примере, ситуации, что *Пассажир_N* хочет ехать с *Пассажиром_M*, а *Пассажир_M* не хочет ехать с *Пассажиром_N*, либо наоборот, не рассматривается и исключены как невозможные.

– Уважаемый, подскажите, пожалуйста, где мы находимся?

…

– В корзине воздушного шара.

Тут их относит дальше и они опять ничего не видят.

– Это был математик – говорит Холмс.

– Но, почему?

– Во-первых, он очень долго думал над простым вопросом, во-вторых, он дал нам абсолютно верный ответ, и в-третьих, – нам от него нет никакой пользы.

Поэтому ещё задолго до появления квантовых компьютеров, в отношении обычных вычислительных машин математики и инженеры совместно задавались вопросами формализации представлений «*о том, что некоторые функции поддаются «механическому» вычислению, скажем на ЭВМ, как только для этого составлена надлежащая программа, тогда как другие функции, заданные неэффективным определением, могут требовать творческого подхода...*» [40, стр.15]

С одной стороны, требовалась формализация. Перевод всех задач к вычислению функций, поскольку функции можно посчитать на ЭВМ, с другой стороны, функции бывают разные, в том числе и невычислимые. Появилось ещё два вида научных потребностей – составлять алгоритмы для решения задач (в общем виде – вычисления функций) и возможность оценивания алгоритмов в отношении их конечной вычислимости, чтобы они не превращались в бесконечный процесс вычислений. (Подробнее см. 4.5.5. Алгоритмизация, стр. 220.)

Некоторые умельцы (инженеры-изобретатели) поняли, что математики зашли в тупик в отношении универсального решения сложных, не поддающихся формализации и вычислению задач и стали предлагать свои действенные решения, если не для общего случая, то хотя бы для ряда частных.

Так, силой инженерной мысли и мудростью поколений, многие сложные задачи подверглись решению ещё в средние века. Приведём несколько современных интересных и занимательных примеров с инженерной точки зрения.

Печник *(Побасенка 3.14. Слюна как подручное средство [25, стр. 80]):*

Несколько высококвалифицированных инженеров пытались установить в старую сталеплавильную печь прибор для определения температуры стали в центре печи. За их работой наблюдал старый сталевар. После нескольких неудачных попыток запарившиеся и расстроенные специалисты сделали перерыв.

Тогда старый сталевар спросил их, что они пытаются делать. Выслушав объяснения инженеров, старик сказал, что мог бы показать, как сделать это и без инструментов. Он выплюнул большой комок табачной жвачки на стенку печи и посмотрел на часы. Через несколько секунд он «объявил» температуру. Когда его попросили объяснить, как он её определил, старик сказал, что его предшественник научил его определять время испарения слюны, что он и сделал, а затем умножать его на определённое заданное число. Результат даёт температуру в центре печи. Инженеры снисходительно засмеялись.

Несколько дней спустя, когда им удалось найти способ установить прибор в печь, они решили позабавиться над стариком и проверить его. Но им пришлось об этом сожалеть. Старик оказался прав.

Мораль: годами накопленный опыт может поспорить со знанием.

Пещера

В криптографии, а из неё и в других областях знаний, существует понятие «доказательства с нулевым разглашением информации». Если коротко, то это задача, при которой одна сторона доказывает другой какое-то известное обоим утверждение (например, факт знания пароля

или хэша от него), при этом в результате такого доказательства не передаётся никакой информации в отношении доказываемого утверждения. То есть, если появится третья сторона и ознакомится с записанным протоколом общения первых двух, то она не получит никаких для себя знаний и не сможет выполнить такое же доказательство. Более понятный пример, этого же самого: представьте ситуацию, студент активно работал весь семестр, готовился несколько последних дней и ночей и вот, заветным утром, он рассказывает содержание билета преподавателю и получает заслуженную пятёрку. Рядом сидит другой студент, который совсем не готовился, но ему выпал такой же вопрос, у него замечательная память (также не менее важный талант для достижения целей), он внимательно слушает рассказ своего коллеги по учёбе, затем повторяет его через несколько минут и получает также пятёрку. Как отвечать открыто и громко (иначе могут заподозрить профессора в необъективности) первому студенту, чтобы другие студенты, которые тщательно не готовились, но услышали правильный ответ, не смогли им воспользоваться?[110]

Вся сложность состоит в том, чтобы доказать, что у одной из сторон есть информация, при этом не раскрывая её содержания. Для этого всего существуют, помогающие в той или ной мере, однонаправленные функции, хэши и различные системы и схемы с нулевой передачей знаний. С математической точки зрения они сложны и поэтому мы их не приводим. Вы их без труда найдёте в [26][27], но в одной из работ Жан-Жака Кискатера по этой теме, дословно называющейся «Как объяснить протокол доказательства с нулевым разглашением вашим детям», был приведён пример с пещерой (также доступный в [27, глава 5]), позволяющий объяснить сложное через простое.

Допустим у вас есть волшебная пещера (см. рис. 4.25), внутри которой есть специальная дверь, которая всегда закрыта, но она открывается на некоторое время по произнесении определённого слова возле неё (не важно с какой стороны от двери оно произносится). Человек «Д» (доказывающий) знает это волшебное слово. Человек «П» (проверяющий) его не знает. Как человеку Д доказать человеку П, что он знает слово?

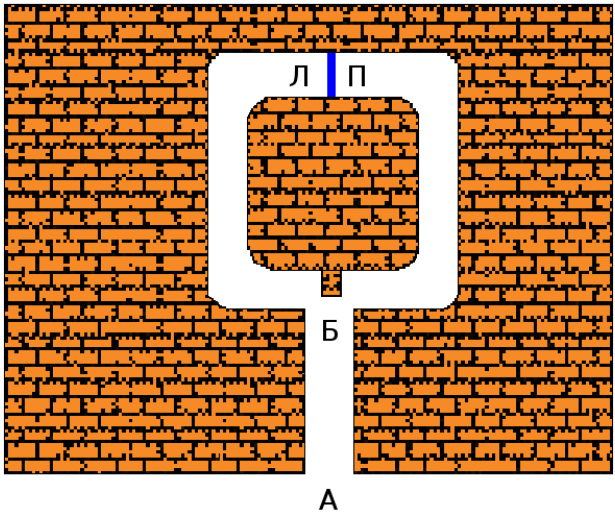


Рисунок 4.25. Волшебная пещера

Алгоритм доказательства (проверки с нулевой передачей знаний): Человеку П даётся видеокамера и он записывает как человек Д уходит в пещеру из точки А. Куда уйдёт человек Д неизвестно, но мы знаем, что он будет или в точке Л или в точке П. Далее человек П переходит к разветвлению в пещере в точке Б (не выключая записи на камере) и громко просит человека Д появиться справа или слева от него (на своё усмотрение). Человек Д появляется с запрошенной стороны. Вероятность выполнения этого действия не зная волшебного слова составляет 50%. Если же Человек Д знает слово, то он может со 100% вероятностью выйти с той стороны, где его попросят. Человек П не получает никакой информации о самом секретном слове. Если вероятности 50% мало, то действие можно повторить, как один, так и лю-

[110] С точки зрения обучения в вузе объективно оба студента хороши, потому как, если была цель дать им реальные знания до наступления формального момента получения ими дипломов, то к качеству обучения в вузе претензий нет и уже не важно ходил ли кто-то из них на лекции или нет. Тут как на войне (или в любви), все средства хороши.

бое число раз. Любой просмотревший полученную видеозапись с несколькими десятками успешных выходов человека Д с запрошенной стороны (Видеозапись считаем подлинной, сговор людей П и Д исключаем.) также будет знать, что человек Д знает волшебное слово, но само слово по прежнему будет знать только Д.

В зарубежной литературе эту методологию ещё в шутку называют «отрежь и дай выбрать другому», потому как математическая её реализация напоминает жизненную ситуацию когда двум сладкоешкам надо справедливо разделить торт на две части, исключив ситуацию, что его оппоненту достанется более вкусный и большой кусок.

Как видно из примеров, если расширить свой кругозор, посмотреть шире на происходящее или уйти в ещё одно изменение, многие задачи поддаются решению и для этого не надо быть детьми с бурной фантазией. Взрослым это тоже под силу, но нужны знания. Например, в электротехнике возможно использование формулы Эйлера и переход от обычного электрического тока и напряжения (измеряемого вольтметром) к комплексным.

$$e^{i\varphi}=\cos\varphi+i\sin\varphi$$

Сложные функции не подающиеся прямому вычислению в большинстве случаев можно разложить в ряд Тейлора и вычислять последний, используя простые арифметические операции.

$$f(x)=f(x_0)+f'(x_0)(x-x_0)+\frac{f''(x_0)}{2!}(x-x_0)^2+\ldots+\frac{f^{(n)}(x_0)}{n!}(x-x_0)^n+\ldots$$

Перейти к квантовым компьютерам, как и к комплексным числам, если вы не знаете что это такое, – сложно.

Это будет сродни ситуации, если строящему с помощью дерева, топора, молотка и гвоздей человеку скворечник или собачью будку предложить промышленный кран для подъёма бетонных плит или насос для прокачки бетона на высоту 100-этажного дома.

Так и в случае квантовых компьютеров, задачи, решаемые с их помощью, довольно специфичны и вряд ли будут интересны бытовым пользователям (разве, что майнинг криптовалют). Квантовые компьютеры не будут быстрее загружать видеоролики и фильмы из интернета, они не сделают телефон и планшеты меньше по размеру, они не уменьшат электропотребление и не будут рисовать более красивые смайлики и вряд ли помогут в решении множества задач. Они призваны снизить вычислительную сложность[111] у ряда сложных задач, а вот решение последних в прикладном аспекте сможет повлиять на нашу бытовую жизнь.

Источником таких «больших» (вычислительной сложных) задач может случить криптография. Например, это могут быть задачи: факторизации натурального числа – разложения его на произведение простых сомножителей, дискретного логарифмирования и логарифмирования на эллиптической кривой, рассмотрение которых выходит за рамки данного учебника.

[111] Вычислительная сложность – понятие в информатике и теории алгоритмов, обозначающее функцию зависимости объёма работы, которая выполняется некоторым алгоритмом, от размера входных данных. С вычислительной сложностью тесно связаны понятия класса сложности. Наиболее часто встречаемые – это P- и NP- классы сложности.

Выводы о реализации, сравнении и совместимости

Физическая реализация квантового компьютера (различные экспериментальные образцы) на сегодня не имеет стандартизации. Источниками хранения кубитов[112] могут служить различные системы, обладающие вероятностными свойствами, описанными выше. Например это могут быть фотоны, электроны, ядра, когерентные состояния света, оптические структуры или решётки, протекание сверхпроводящего тока в состоянии эффекта Джозефсона, а также различные полупроводниковые схемы, выполненные по технологическому процессу с малыми геометрическими размерами и как следствие которого, на работу элементов начинают существенно оказывать воздействие квантовые эффекты.

Упрощённо схему вычислений на квантовом компьютере можно себе представить примерно следующим образом: берётся система кубитов, реализованная физически, на которую записывается её начальное состояние.[113] Затем, состояние системы или её подсистем изменяется посредством унитарных преобразований, выполняющих те или иные логические операции. После этого производится измерение значений, хранимых в кубитах, – это и есть результат работы компьютера. Квантовая система даёт результат только с некоторой вероятностью являющийся правильным, однако, наподобие примера «пещеры Али-Бабы», за счёт небольшого увеличения операций в алгоритме можно сколь угодно приблизить вероятность получения правильного результата к единице.

В сравнении с обычным (классическим) компьютером можно сказать, что проводам находящимися под высоким и низким напряжением или в логических состояниях (например, с выхода триггера) можно соотнести кубиты, а логическим блокам (функциям алгебры логики или их комбинациям) – унитарные преобразования. Такая концепция квантового процессора и квантовых логических вентилей была предложена в 1989 году Дэвидом Дойчем.

Также, по аналогии с минимальным логическим базисом в современных ЭВМ, Дэвид Дойч в 1995 году нашёл универсальный логический блок для квантового компьютера, с помощью которого можно выполнять любые квантовые вычисления. Для построения любого вычисления в квантовом компьютере достаточно двух базовых операций.

Квантовые компьютеры могут иметь и обратную совместимость, поскольку с помощью базовых квантовых операций могут сэмулировать работу обычных логических элементов, то есть обычных компьютеров и, как следствие, решить любую задачу, которая решена сейчас, однако выигрыша по времени от этого вы не получите, как если будете использовать трактор для засеивания небольшого ящика с рассадой, размещаемого на подоконнике.

[112] См. *https://en.wikipedia.org/wiki/Qubit*.

[113] Как это сделать и есть вопрос современных научных исследований, патентований, know-how и технологических секретов одних фирм перед другими. Грубо говоря, одним из примеров может быть некоторая большая молекула или структура из отдельных атомов и/или частиц, созданная в зависимости от хранимого ими состояния, скажем, двоично закодированного текста. Логической операцией может быть облучение этой структуры некоторым электрическим или магнитным полем, либо физико-химическое взаимодействие. Суть взаимодействия будет как раз описываться высчитываемой функцией. Например, как линза делает преобразование Фурье. По завершении взаимодействия, структура станет немного другой и будет хранить в себе новое запомненое состояние (типа результата вычисленной функции), которое останется только как-то считать или косвенно определить.

Прямая совместимость тоже есть: 100 людей с лопатами за день могут выполнить получасовую работу трактора, бульдозера или грейдера.

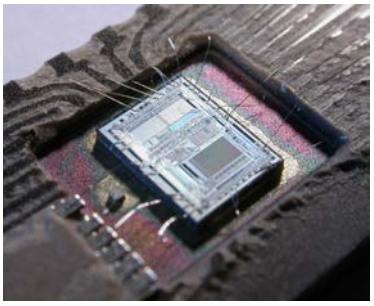
4.5. Научные основы работы современной ЭВМ

Имея представленную выше элементную базу (даже для первых поколений ЭВМ), было бы грех ею не воспользоваться «по полной» на благо общества. Инженеры стали массово придумывать устройства и создавать различные электрические схемы. Всё бы хорошо, но аналоговая обработка данных (сигналов) оказалась сложной и помехонеустойчивой, фактически зашла в тупик. В связи с чем было решено перейти на «двоичные цифровые рельсы», используя которые, ЭВМ можно представить в виде «чёрного ящика», математическое обоснование работы которого приводится далее.

4.5.1. Математические основы работы «чёрного ящика»

Логические состояния «0» и «1» наводят на мысль о следующем историческом персонаже:

Джордж Буль
(1815–1864)



Английский математик и логик, основоположник математической логики.

Его монографии «*Математический анализ логики*», «*Исследование законов мышления*» стали классическими. И сейчас в современной алгебре есть такие понятия, как булевы кольца, булев разброс, булево разложение, булева регулярная точка ядра.

Работы 1847 и 1854 годов положили начало алгебре логики, или булевой алгебре. Буль первым показал, что существует аналогия между алгебраическими и логическими действиями, так как и те, и другие предполагают лишь два варианта ответов – «истина» или «ложь», «0» или «1». Он придумал систему обозначений и правил, пользуясь которыми можно было закодировать любые высказывания[114], а затем манипулировать ими как обычными числами. Булю удалось подробно описать двоичную систему счисления. В своей работе «*Законы мышления*» (1854 г.) Буль окончательно сформулировал основы математической логики. Он также попытался сформулировать общий метод вероятностей, с помощью которого из заданной системы вероятных событий можно было бы определить вероятность последующего события, логически связанного с ними.

Созданный им аппарат математической логики позже стал носить название булевой алгебры. Фактически он позволил представить современный компьютер в виде «чёрного ящика». Рассмотрим подробнее в чём суть этого подхода.

4.5.1.1. Какие они, сколько их?

Рассмотрим некоторое физическое электронно-цифровое устройство (схему с подведённым к ней электропитанием) у которого будет один электрический вход и один выход. Соответственно, ему можно сопоставить некоторую логическую функцию (булеву функцию) от одной переменной $y = f(x)$, где x – это то, что подаётся на вход, а y – это то, что получается на выходе. Ситуация с меньшим количеством входов (переменных) или выходов не интересна, поскольку, она тривиальна.

Замечание 1. Логическая функция отличается от функций изучаемых в школьном курсе алгебры тем, что все возможные значения её входных переменных, как и получаемый результат – логические, то есть, значения называемые и записываемые условно как логический «0» и логическая «1». Такое сильное упрощение приводит к ряду занимательных особенностей не совсем очевидных с первого взгляда.

Замечание 2. Идеальная математическая функция ≠ реальное физическое устройство.

Во-первых, если считать параметры функции входом ящика, а значение функции выходом, то любой реализованный на практике ящик будет работать с некоторой задержкой, хотя бы силу того, что скорость света являющаяся верхним пределом для распространения электрического тока конечна. Если работать с ящиком не непрерывно во времени, а лишь в заданные с некоторой периодичностью моменты времени, то в эти моменты он будет полностью «совпадать» с соответствующей ему функцией, поскольку все переходные процессы будут происходить в тот момент когда мы с ящиком не работаем. (Если профессор в университете приходит на работу раньше студентов, уходит – позже, а обедает когда у студентов занятия по другому предмету, то у последних может сложиться впечатление, что их профессор не спит, не ест и живёт на работе. Собственно, многие также ощущают работу своего компьютера.)

Во-вторых, модель, сопоставляющая «ящику», на деле являющемуся физическим устройством, функцию накладывает на реализацию этого ящика ограничения присущие функции. Функция по сути не видит придуманное нами время (в котором мы живём), она абстрактна и работает мгновенно. На практике это выглядит как то, что значение функции зависит только от входных её параметров. А значит, у ящика не должно быть «памяти» или «задержки», превы-

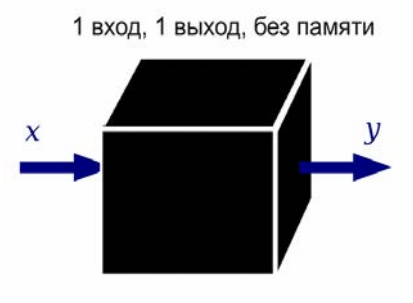
[114] Высказывание - некоторое предложение, о котором можно утверждать, что оно истинно или ложно.

шающей период с которым этот ящик «опрашивается». Вариант ящика с «памятью» впоследствии будет рассмотрен отдельно.

Замечание 3. Будем считать, что наличие напряжения (на входе, выходе или любой точке схемы) более 2,5 В есть логическая «1», а напряжение менее 2,5 В – логический «0». Существует условная договорённость (см. ГОСТ 2.743-82) считать, что логика у которой напряжение «1» больше напряжения «0» называется положительной логикой, а логика у которой напряжение «0» больше напряжения «1» называется отрицательной логикой. Выбор границы двух состояний в +2,5 В ещё одна условность. Скорее это дань исторической моде, – половина напряжения электропитания транзисторно-транзисторной логики (ТТЛ) работавшей от +5 В. С целью снижения тепловыделения напряжение питания современных логических схем оказывается ниже, естественно, снижается напряжение границы разделяющей «0» и «1». Нижний предел у схем также существует, поскольку обычный полупроводниковый p-n переход открывается при напряжении, как минимум 0,6 В.

Если не стоит цель углубляться в схемотехнические решения, то большинство задач выполняемых ЭВМ без труда можно решать на абстрактном уровне логики, оперируя только «0», «1» и логическими функциями.

На один вход ящика (функции) возможно подать счётное число возможных входных комбинаций, а именно две, как и на выходе получить также конечное число вариантов – также два. Возникает вопрос, а каково минимальное число внешне различимых вариантов реализаций чёрных ящиков, чтобы охватывался весь спектр входных и выходных комбинаций?[115] Их оказалось четыре (2 в степени 2 в степени 1).



вариант	значение входа, x	внутри ящика $y = f_i(x)$	значение выхода, y
1	$x = 0$	$y = f_1(x) = 0$	$y = 0$
	$x = 1$		$y = 0$
2	$x = 0$	$y = f_2(x) = x$	$y = 0$
	$x = 1$		$y = 1$
3	$x = 0$	$y = f_3(x) = \sim x$	$y = 1$
	$x = 1$		$y = 0$
4	$x = 0$	$y = f_4(x) = 1$	$y = 1$
	$x = 1$		$y = 1$

Четырём возможным вариантам соответствуют четыре функции. С точки зрения алгебры логики самые простые из них это $f_1(x)$ – абсолютно ложная функция (константа нуля) при любом значении аргумента, и $f_4(x)$ – абсолютно истинная функция (константа единицы). В жизни, устройства, реализующие такие функции получатся, если выход чёрного ящика напрямую соединить с шиной питания: землёй или нулём ($\perp$) и $+U_{num}$, соответственно. Естественно, выход никак не будет зависеть от входа.

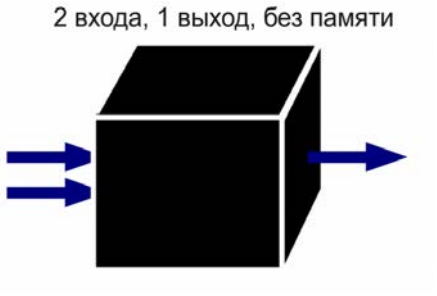
[115] Внутренних различных реализаций одного варианта может быть бесконечное множество. Если при всех входных комбинациях у двух разных ящиков выходы совпадают, то это подсчитывается как один вариант.

Функция $f_3(x)$ повторяет значения входной логической переменной: тождественная функция $f_3(x)\equiv x$ (вход напрямую или через буфер соединён с выходом), а функция $f_4(x) = \neg x = \bar{x} = \sim x$, противоположная по своим значениям x, – логическое отрицание, инвертор или функция «НЕ»[116].

Последующей итерацией стало рассмотрение ящика с двумя входами и одним выходом и соответствующих им логических функций от двух переменных.

x_1	x_2	$f_i(x_1,x_2)$
0	0	$f_i(0,0)$
0	1	$f_i(0,1)$
1	0	$f_i(1,0)$
1	1	$f_i(1,1)$

$i \in \overline{1,16}$



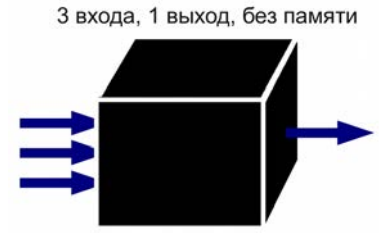
Таких функций оказалось 16 (2 в степени 2 в степени 2), кратко они все представлены в таблице 4.2.

Таблица 4.2. Все логические функции от двух переменных

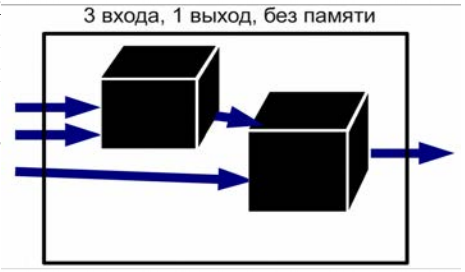
Функция $f_i(x_1,x_2)$	x_1 x_2 00	01	10	11	Примечание
	Результат				
f_1	0	0	0	0	тождественный 0
f_2	0	0	0	1	логическое «И», «AND», конъюнкция, логическое умножение, $x_1 \wedge x_2$ $x_1 * x_2$ $x_1 x_2$
f_3	0	0	1	0	$x_1 \wedge \bar{x}_2$ (запрет x_2)
f_4	0	0	1	1	$x_1 \bar{x}_2 \vee x_1 x_2 = x_1$
f_5	0	1	0	0	$\bar{x}_1 x_2$ (запрет x_1)
f_6	0	1	0	1	$\bar{x}_1 x_2 \vee x_1 x_2 = x_2$
f_7	0	1	1	0	$x_1 \oplus x_2$(сложение по модулю 2)
f_8	0	1	1	1	логическое «ИЛИ», «OR», дизъюнкция, логическое сложение, $x_1 \vee x_2$ $x_1 + x_2$
f_9	1	0	0	0	$\overline{x_1 \vee x_2} = x_1 \downarrow x_2$ (стрелка Пирса, функция Вебба, функция Даггера, «ИЛИ-НЕ»)
f_{10}	1	0	0	1	$x_1 = x_2$ $x_1 \equiv x_2$ (равнозначность, эквивалентность)
f_{11}	1	0	1	0	$\bar{x}_1 \bar{x}_2 \vee x_1 \bar{x}_2 = \bar{x}_2$
f_{12}	1	0	1	1	$x_2 \rightarrow x_1$ (импликация)

[116] Знаки «¬», «~» и надчёркивание выражения сверху – математическое обозначение операции отрицания.

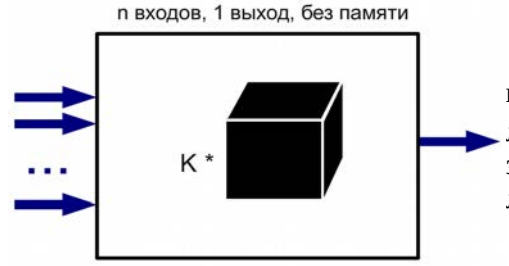
f_{13}	1	1	0	0	$\bar{x}_1\bar{x}_2 \vee \bar{x}_1 x_2 = \bar{x}_1$
f_{14}	1	1	0	1	$x_1 \rightarrow x_2$ (импликация)
f_{15}	1	1	1	0	x_1 / x_2 (функция, штрих Шеффера, «И-НЕ»)
f_{16}	1	1	1	1	тождественная 1



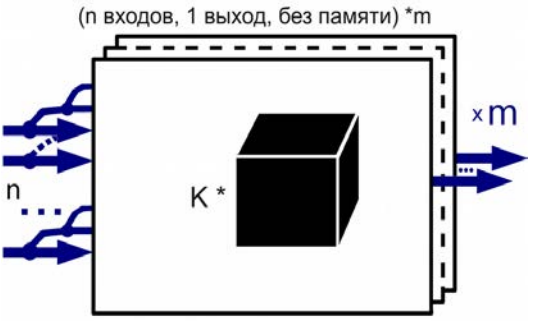
Следующим логичным шагом является рассмотрение чёрного ящика с тремя входами. Добавление всего лишь одного дополнительного входа приводит к тому, что число внутренних его реализаций оказывается 256 (2 в степени 2 в степени 3). Рассмотрение всех возможных комбинаций не имеет смысла, поскольку 3-входовый ящик легко представим внутри двумя более простыми – двухвходовыми ящиками.



Как легко догадаться, указанный опыт представления внутренностей чёрного ящика можно распространить и на *n* входов, грубо нарисовав что-то вида:



В общем случае число реализаций такой конструкции будет 2^{2^n}, при этом, если желаемым будет получить ещё и *m* выходов, указанные ящики придётся размножить, запараллелив входы.



В каком-то понимании такая конструкция будет напоминать «компьютер», например на входе можно подавать закодированный в двоичном виде номер сохранённого ранее видеофильма (или желаемого текста), а на выходе получать одновременно биты с 1 по *m*-й из последовательности закодированного фильма, или текстового произведения, или чего-то ещё. При этом на входе можно подавать разные цифры, а на выходе иметь разный результат.

Любому нормальному человеку такая схема покажется насколько простой, настолько и сложной... если не сказать, что избыточной... и он будет прав, это математическая абстракция, избыточная схема. Современные компьютеры так не работают, хотя что-то от предложенной выше схемы (чёрный ящик с *n* входами и *m* выходами) имеют.

4.5.1.2. Основные положения алгебры логики и вытекающие из этого интересные свойства

Для логических переменных и 4-х основных операций: конъюнкции (логического умножения), дизъюнкции (логического сложения), инверсии (отрицания) и эквивалентности (=) существуют следующие аксиомы.

$0 + 0 = 0$ $0 * 0 = 0$ $0 + 1 = 1 + 0 = 1$ $\sim 0 = 1$

$1 + 1 = 1$ $1 * 1 = 1$ $0 * 1 = 1 * 0 = 0$ $\sim 1 = 0$

Причём, из их тех или иных сочетаний аксиом следуют свойства:

$x + x = x$ $x + 0 = x$ $1 + x = 1$ $x + \sim x = 1$ $\sim(x_0 + x_1) = \sim x_0 * \sim x_1$

$x * x = x$ $0 * x = 0$ $x * 1 = x$ $\sim x * x = 0$ $\sim(x_0 * x_1) = \sim x_0 + \sim x_1$

$\sim(\sim x) = x$

Два самых правых равенства – это принцип двойственности или теорема Де Моргана (инверсия логической суммы равна логическому произведению инверсий и наоборот). Соотношения двойственности для n переменных, часто записывают в виде:

$\sim(x_1 + ... + x_n) = \sim x_1 * ... * \sim x_n$ и $\sim(x_1 * ... * x_n) = \sim x_1 + ... + \sim x_n$

Также на функции конъюнкции и дизъюнкции распространяются обычные алгебраические законы – переместительный, сочетательный и распределительный, которые легко доказываются методом перебора:

$x_1 + x_0 = x_0 + x_1$; $x_1 * x_0 = x_0 * x_1$ – переместительный;

$x_2 + x_1 + x_0 = (x_2 + x_1) + x_0$; $x_2 * x_1 * x_0 = (x_2 * x_1) * x_0$ – сочетательный;

$x_2*(x_1+x_0) = (x_2*x_1) + (x_2*x_0)$; $x_2 + (x_1*x_0) = (x_2+x_1) * (x_2+x_0)$ – распределительный.

Более сложным вытекающим из аксиом свойством является теорема разложения, в соответствии с которой любую логическую функцию $f(x_1, x_2, x_3, ..., x_n)$ от n переменных можно разложить по любой из её переменных. Или, что тоже самое, что вместо логической функции от n переменных можно использовать комбинации других логических функций с числом переменным от которых они зависят на одну меньше (то есть, например представить трёх-входовый чёрный ящик через два двух-входовых).

Представьте логическую функцию $f(x_1, x_2, x_3, ..., x_n)$ от n переменных. Покажем, как можно уменьшить количество входных переменных. Возьмём, например переменную x_2. Согласно введённой аксиоматике выполним несколько несложных преобразований: $f(x_1, x_2, x_3, ..., x_n) = 1 * f(x_1, x_2, x_3, ..., x_n) = (x_2 + \sim x_2) * f(x_1, x_2, x_3, ..., x_n) =$

$$= x_2 * f(x_1, x_2, x_3, ..., x_n) + \sim x_2 * f(x_1, x_2, x_3, ..., x_n) =$$

$$= x_2 * f(x_1, 1, x_3, ..., x_n) + \sim x_2 * f(x_1, 0, x_3, ..., x_n).$$

Выражение верно, поскольку, если $x_2=1$, то её инверсия $\sim x_2=0$. Таким образом, получим $f(x_1, x_2, x_3, ..., x_n) = f(x_1, 1, x_3, ..., x_n)$. Если же $x_2=0$, то её инверсия $\sim x_2=1$. Таким образом, получим $f(x_1, x_2, x_3, ..., x_n) = f(x_1, 0, x_3, ..., x_n)$. При этом, если ввести замены $g_1(x_1, x_3, ..., x_n) = f(x_1, 1, x_3, ..., x_n)$ и $g_0(x_1, x_3, ..., x_n) = f(x_1, 0, x_3, ..., x_n)$, то в выражении $f(x_1, x_2, x_3, ..., x_n) = x_2 * g_1(x_1, x_3, ..., x_n) + \sim x_2 * g_0(x_1, x_3, ..., x_n)$
функция $f()$ слева от знака равенства зависит от n входных переменных, а в выражении справа от знака равенства функции $g_1( )$ и $g_0( )$ зависят от n-1 входной переменной.

Теорему разложения можно применить повторно ещё n-1 раз в отношении оставшихся переменных (x_1, x_3, ... , x_n) функций $g_1(\ )$ и $g_0(\ )$, тогда, в конечном итоге первоначальная логическая функция $f(\ )$ будет разложена по всем своим переменным.

Вывод, который можно сделать из применения теоремы состоит в том, что любая логическая функция, какой-бы она сложной не была, может быть представима набором простых операций. То есть, любую придуманную логическую функцию можно реализовать с помощью физического устройства на практике.

4.5.1.3. Способы описания работы «чёрного ящика»

В связи с тем, что над созданием «чёрного ящика» ЭВМ трудились люди разных взглядов и склада ума, можно выделить следующие подходы к описанию его работы через:

- Произвольное словесное описание;
- Таблицу истинности (табличное описание всех возможных комбинаций входов и выходов);
- Математическую формулу (функцию алгебры логики);
- Условно-графическое обозначение (схемотехнику);
- Другие способы (например, через описание на алгоритмическом языке).

Скорее всего, предпочтения читателей по выбору наиболее удобного метода описания разойдутся. Важно уметь «переходить» от одного описания к другому. К сожалению этот вопрос не рассматриваются в данном учебнике, поэтому мы лишь коротко обозначим типичную схему переходов: на основе компактного словесного описания желаемого результата заполняют таблицу истинности будущей схемы (заполнение может быть неполным, если некоторые комбинации на вход никогда не будут подаваться).

Затем на основе таблицы формируют математическую формулу (описывают сложную логическую функцию). При переходе от табличного описания к функции последняя обычно представляется в виде конъюнкции нескольких членов, каждый из которых является простой дизъюнкцией аргументов или их инверсий. Такая форма представления функции называется нормальной конъюнктивной формой (НКФ). Также функция может быть представима в виде дизъюнкции нескольких членов, каждый из которых является простой конъюнкцией аргументов или их инверсий. Такая форма представления функции называется нормальной дизъюнктивной формой (НДФ).

Нормальные (конъюнктивная, дизъюнктивная) формы не дают однозначного представления функции. Такое представление можно получить только при использовании совершенных нормальных форм.

Форма называется совершенной нормальной дизъюнктивной формой (СНДФ), если в каждом члене НДФ представлены все аргументы (или их инверсии) функции. Аналогично для совершенной нормальной конъюнктивной формы (СНДФ).

Получившаяся функция может оказаться громоздкой и избыточной, поэтому впоследствии её, если это возможно, упрощают (методом Квайна или методом карт Вейча).

На основе получившейся упрощённой логической функции создают условно-графическую схему из простых логических элементов и заменяют условные блоки схемотехническими решениями.

Рассмотрим понятие базиса и основные используемые логические элементы подробнее.

4.5.1.4. Базисы функций алгебры логики

16 логических функций от двух переменных (описанных в таблице 4.2) – это не так много, но оказывается, чтобы представить любую логическую функцию в теории можно обойтись и меньшим числом различных логических функций от двух переменных. Объясняется это тем, что можно выражать одни логические функции через другие логические функции. Почему так произошло в природе? – вопрос открытый.

В теории любая логическая функция может быть представлена в виде суперпозиции функций, образующих базис. Это факт является теоретической основой возможности построения любого цифрового устройства с помощью набора базисных электронных устройств, реализующих функции базиса (этих функций немного, и их физическая реализация в виде электронных устройств достаточно проста).

Базис – набор относительно простых функций алгебры логики (или соответствующих им физических устройств), с помощью комбинации которых в неограниченных количествах можно выразить любую другую функция алгебры логики (или создать соответствующее ей физическое устройство).

Примерами базисов могут служить:

- система функций И, ИЛИ, НЕ (базис № 1);
- система, содержащая функции И, НЕ (базис № 2),
- система, содержащая функции ИЛИ, НЕ (базис № 3),
- функция Шеффера (И-НЕ) (базис № 4);
- функция Пирса (ИЛИ-НЕ) (базис № 5).

Базис № 1 наверняка известен каждому бывшему школьнику. Базисы № 4 и № 5 являются минимальными.

4.5.2. Часто используемые базовые логические элементы

Рассмотрим логические элементы из которых возможно построение базисов. Комбинируя элементы того или иного базиса, можно построить такие часто используемые в цифровой технике устройства, как шифраторы, дешифраторы, мультиплексоры, демультиплексоры, компараторы, сумматоры, триггеры и др.

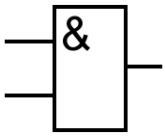
4.5.2.1. И, AND

Логический элемент, реализующий операцию логического умножения «И» (конъюнкцию) двух переменных. Естественно, имеет два входа[117] и один выход. Таблица истинности для данного элемента выглядит так:

Вход 1	Вход 2	Выход
0	0	0
0	1	0
1	0	0
1	1	1

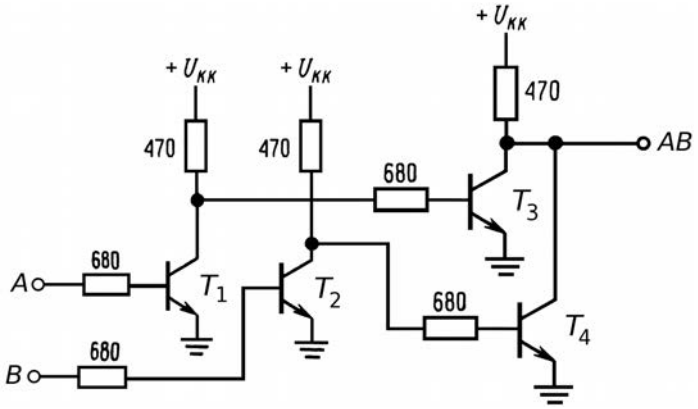
Условно-графическое изображение по стандартам:

ГОСТ 2.743–91 ANSI



[117] Интегральные схемотехнические решения для элементов бывают на три входа и более. Смысла их рассматривать нет, так как внутри они представимы двумя или более элементами с двумя входами.

С помощью транзисторов элемент «И» можно выполнить следующим образом:



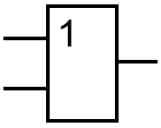
4.5.2.2. ИЛИ, OR

Логический элемент, реализующий операцию логического сложения «ИЛИ» (дизъюнкцию). Количество входов аналогично предыдущему случаю – два. Таблица истинности выглядит следующим образом:

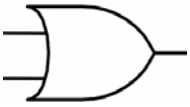
Вход 1	Вход 2	Выход
0	0	0
0	1	1
1	0	1
1	1	1

Условно-графическое изображение схемы по стандартам:

ГОСТ 2.743–91



ANSI



С помощью транзисторов данный элемент реализуется как два последовательно соединённых элемента «ИЛИ-НЕ» и «НЕ».

4.5.2.3. НЕ, NOT

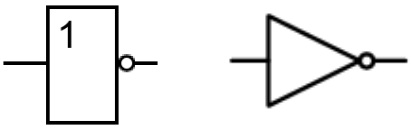
Инвертор – фактически простейший логический элемент, изменяет значение входного сигнала на прямо противоположное. Его функция записывается в следующем виде: $F(x) = \overline{x}$. Таблица истинности выглядит следующим образом (так как вход у этого логического элемента только один, таблица истинности намного проще и состоит только из двух строк):

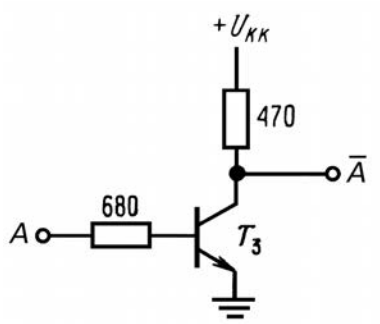
Вход 1	Выход
1	0
0	1

Схемотехнически инверторы могут обладать различным временем распространения сигнала и могут работать на различные виды нагрузки. Они могут быть выполнены на одном

или на нескольких транзисторах, но независимо от внутренней схемы и её параметров они имеют одинаковое условно-графическое обозначение и осуществляют одну и ту же функцию.

ГОСТ 2.743–91 Стандарт ANSI





С помощью транзисторов элемент «НЕ» можно выполнить следующим образом:

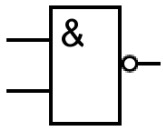
4.5.2.4. И-НЕ, NAND

Данная функция является комбинацией двух логических функций «И» и «НЕ». На практике реализация элемента И-НЕ может оказаться проще (выполняется меньшим количеством элементов схемы), чем реализация отдельно элемента «И». Эту функцию можно записать при помощи *таблицы истинности*, приведённой в таблице ниже:

Вход 1	Вход 2	Выход
0	0	1
0	1	1
1	0	1
1	1	0

Элемент, выполняющий эту функцию, имеет два входа. Вообще говоря, могут использоваться логические элементы, имеющие несколько входов. Условно-графическое изображение логического элемента: (см. справа)

ГОСТ 2.743–91 Стандарт ANSI



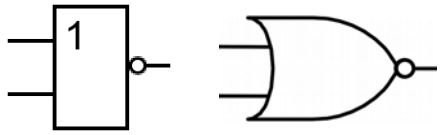
4.5.2.5. ИЛИ-НЕ, NOR

Вход 1	Вход 2	Выход
0	0	1
0	1	0
1	0	0
1	1	0

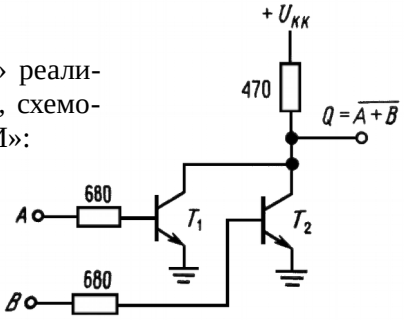
Логический элемент, реализующий операцию логического сложения «ИЛИ-НЕ» оказывается удобен по той же причине, что и предыдущий элемент, и описывается таблицей слева.

Элемент, выполняющий эту функцию, имеет два входа. Вообще говоря, могут использоваться логические элементы, имеющие несколько входов. Условно-графическое изображение схемы, выполняющей функцию логического сложения с последующим отрицанием, приведено справа:

ГОСТ 2.743–91 Стандарт ANSI

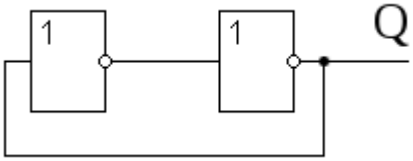


С помощью транзисторов элемент «ИЛИ-НЕ» реализуем следующим образом, и, как легко догадаться, схемотехнически он проще выполним, чем элемент «ИЛИ»:



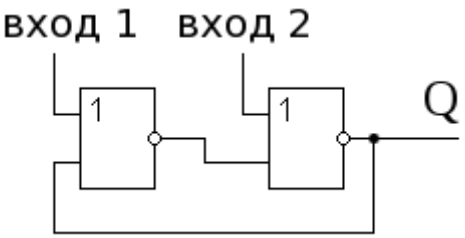
4.5.3. Бистабильные схемы, триггеры

Соедините два элемента «НЕ» следующим несложным образом, мысленно поместите внутрь воображаемого чёрного ящика, так чтобы вывод Q оказался его выходом, после чего ответьте на вопрос: каким будет выходное значение на этом выходе?



Ответ не кажется очевидным, поскольку возможны два правильных, но противоречащих друг другу ответа.

С одинаковой вероятностью на выходе может оказаться как «1», так и «0». Такая схема называется бистабильной. Причём, то или иное установившееся значение будет устойчиво держаться до тех пор пока схема будет электрически запитана.



Если схему снабдить двумя входами № 1 и № 2, то появится возможность переводить её в то или иное необходимое состояние через подачу логических единиц на эти входы.

Имея таблицу истинности элемента «ИЛИ-НЕ», вы без труда можете самостоятельно определить влияние входных значений на выходное.

Подобная бистабильная схема не является единственной.

Наиболее часто элементы «ИЛИ-НЕ» рассмотренной схемы изображают симметрично и в таком виде она встречается в литературе как RS-триггер, самый распространённый из всех существующих схем триггеров.

4.5.3.1. RS-триггер

Триггер – это электронная схема, широко применяемая в регистрах компьютера для надёжного запоминания одного разряда двоичного кода. Фактически это основа хранения данных в памяти. Триггер имеет два устойчивых состояния, одно из которых соответствует двоичной единице, а другое – двоичному нулю.

Термин *триггер* [118] происходит от английского слова *trigger* – защёлка, спусковой крючок. Для обозначения этой схемы в английском языке чаще употребляется термин *flip-flop*, что в переводе означает «хлопанье». Это звукоподражательное название электронной схемы указывает на её способность почти мгновенно переходить («перебрасываться») из одного электрического состояния в другое и наоборот.

Самый распространённый тип триггера – так называемый *RS*-триггер (*S* и *R*, соответственно, от английских *set* – установка и *reset* – сброс). Входы и выходы двух логических элементов «ИЛИ-НЕ» соединены описанным ранее несложным образом. У схемы два входа и один выход, однако, подвести эту схему под какую-либо из ранее рассмотренных 16 логических функций не получится, так как она обладает новым интересным свойством – памятью (объёмом в 1 бит).

В силу симметрии выходов может быть два за счёт добавления второго инверсного выхода – $\overline{Q}$, по большому счёту кардинально на свойства схемы это не влияет.

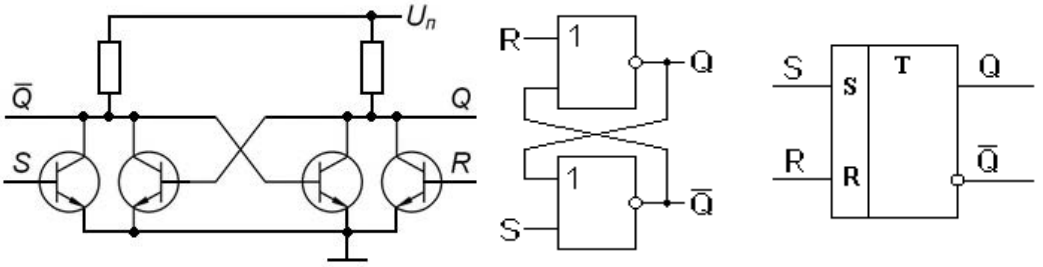


Рисунок 4.26. Триггер RS-типа

Рассмотрим логику работы получившейся схемы подробнее. Вход *S* позволяет устанавливать выход триггера *Q* в единичное состояние. (Устанавливать означает записывать логическую единицу.) Вход *R* позволяет сбрасывать выход триггера *Q* (Quit – выход с англ.) в нулевое состояние.

Переключение триггера из одного устойчивого состояния в другое происходит при подаче активных сигналов на входы. При $R = 1$ триггер устанавливается в состояние, в котором на его выходе $Q = 0$, следовательно, на инверсном выходе $\overline{Q} = 1$, и таким образом, триггер устанавливается в состояние 0. Если триггер до подачи сигнала $R = 1$ находился в состоянии 0, то его состояние не изменится. Если же триггер находился в состоянии 1, то при $R = 1$ произойдёт переключение триггера и на его выходе установится $Q = 0$. Процесс установления триггера в состояние 1 осуществляется при подаче на его вход $S = 1$. Одновременная подача уровней логической «1» на оба входа *R* и *S* не имеет смысла, так как оба выхода триггера (прямой *Q* и инверсный $\overline{Q}$), отображающие его внутреннее состояние, выдадут логический ноль. С математической точки зрения запись $Q = \overline{Q} = 0$ абсурдна и определить по ней что хранит триггер – единицу или ноль невозможно. Если же задаться целью соединить два входа триггера вместе, подать на них «1», затем установить «0» и спросить: что будет после этого на его выходе? Ответ будет: в силу случайных причин триггер может установиться либо в состояние 0, либо в состояние 1. На практике скорее всего результат всё время будет один и тот же, так как сделать два идеально одинаковых элемента и подсоединения невозможно. Поясним выше написанное приведением таблицы переходов для *RS*-триггера.

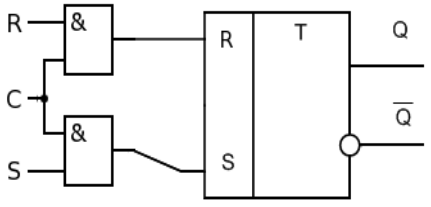
[118] В теории СУБД также используется термин триггер. Там он имеет другой смысл и обозначает программу (внутреннюю функцию), выполняемую по некоторому условию, например изменению какой-либо заранее указанной ячейки в таблице.

Таблица 4.3. Таблица переходов RS-триггера (из момента времени t в момент t+1)

Было в момент t		Подали на входы		Стало на выходах в момент $t+1$			Пояснения
Q	$\overline{Q}$	R	S	Q	Q vs $\overline{Q}$	$\overline{Q}$	
0	1	0	0	0	≠	1	Режим хранения информации $R = S = 0$
1	0	0	0	1	≠	0	
0	1	0	1	1	≠	0	Режим установки единицы $S = 1, R = 0$
1	0	0	1	1	≠	0	
0	1	1	0	0	≠	1	Режим записи нуля $R = 1, S = 0$
1	0	1	0	0	≠	1	
0	1	1	1	0	=	0	Неопределённость, т. к. $Q = \overline{Q}$, а должно быть всегда $Q \neq \overline{Q}$
1	0	1	1	0	=	0	

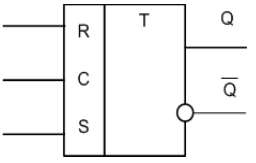
Схема RS-триггера позволяет запоминать состояние логической схемы, но так как в начальный момент времени может возникать переходный процесс[119] (в цифровых схемах этот процесс называется опасными гонками), то запоминать состояния логической схемы нужно только в определённые моменты времени, когда все переходные процессы закончены.

Это означает, что большинство цифровых схем требуют сигнала синхронизации (тактового сигнала). Все переходные процессы в комбинационной логической схеме должны закончиться за время периода синхросигнала, подаваемого на входы триггеров. Схемотехнически получить такой триггер несложно, для этого потребуются дополнительно два элемента «И», которые будут «включать» и «выключить входы» (в какой-то мере можно употребить слова «подключать» и «отключать»). Обозначим дополнительный вход через C [120].



Триггеры, запоминающие входные сигналы только в момент времени, определяемый сигналом синхронизации, называются синхронными. Для того чтобы отличать от них рассмотренные ранее варианты эти триггеры получили название асинхронных.

Обозначение синхронного триггера на принципиальных схемах выглядит следующим образом: (см. справа)



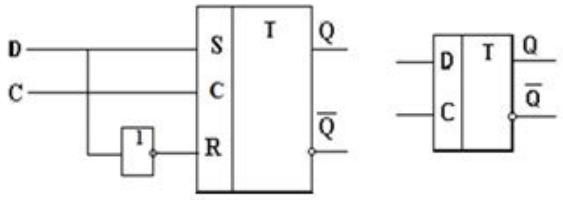
4.5.3.2. D-триггер

В *RS*-триггерах для записи логического нуля и логической единицы требуются разные входы, что не всегда удобно. При записи и хранении данных один бит может принимать значение как нуля, так и единицы. Для его передачи достаточно одного про-

[119] Вы теперь понимаете, зачем на вашем компьютере используется кнопочка Reset? Фактически она устраняет все подобные «неоднозначные» аппаратные состояния внутри вашего компьютера и позволяет запустить его работу заново.

[120] *C* – первая буква от *clock pulse*.

вода. Как мы уже видели ранее, сигналы установки и сброса триггера не могут появляться одновременно, поэтому можно объединить эти входы при помощи инвертора и получить схему *D*-триггера (защёлки).



Такой триггер получил название *D-триггер*. Название происходит от английского слова delay – задержка. Конкретное значение задержки определяется частотой следования импульсов синхронизации.

4.5.3.3. Другие триггеры

В схемотехнической практике для решения разных инженерных задач можно выделить и другие триггеры. Например, на базе элементов «И-НЕ» можно аналогичным образом построить RS-триггер с инверсными входами.

На базе двух триггеров (ведущего и ведомого) можно сделать триггер с двухступенчатым запоминанием информации, например JK-триггер. В общем случае, режим работы T-триггера может быть получен с помощью JK-триггера, либо D-триггера.

Замечание. *Обращаем внимание читателей, что тема триггеров значительно шире, существуют разные виды триггеров и красивые схемотехнические решения от ламп до транзисторов, рассматривать которые мы не ставим своей целью в данной книге.*

4.5.3.4. От триггеров к статической памяти

Последовательное или параллельное соединение триггеров называется регистром. Регистры обычно строятся на основе *D-триггеров*, например см. рис. 4.27.

Объединяя логические схемы вместе (например, даже опытным путём [121]), можно получать различные интересные схемотехнические решения, при этом для упрощения реализации полученных схем можно использовать различные методы минимизации, например Квайна, таблицы Карно и др.

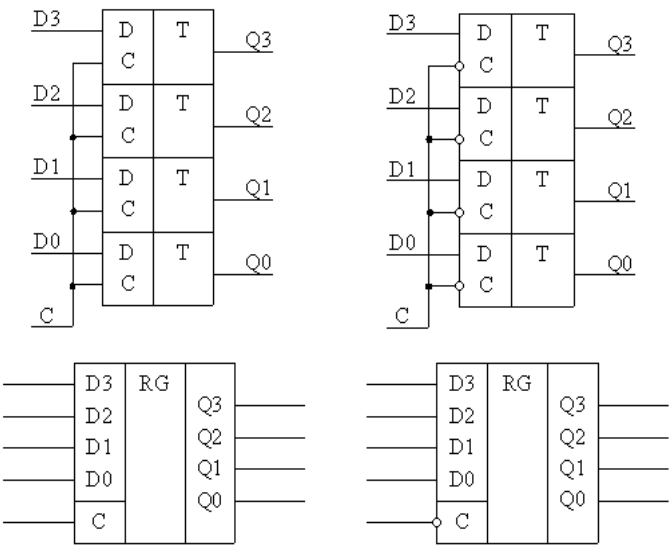
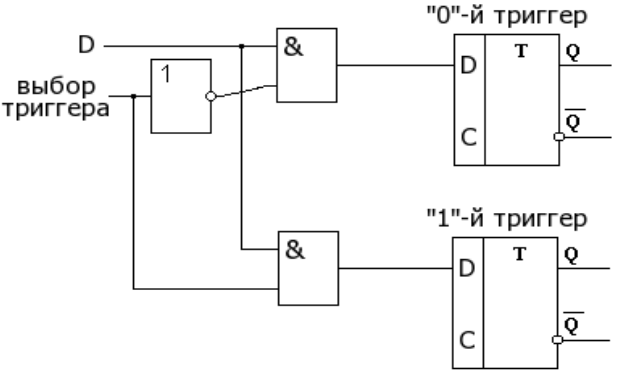


Рисунок 4.27. Объединение нескольких триггеров в регистр

[121] Если раньше инженеры делали это с помощью паяльника, то сегодня в вашем распоряжении есть СПО программы ktechlab и qucs позволяющие на базе ОС Linux моделировать работу электронных схем. Существуют также и аналогичные коммерческие продукты, например для ОС Windows: Proteus, Electronics WorkBench, Spicy для iOS и др.

Полагаем, что к данному моменту у читателей должны были сформироваться логичные вопросы «*А зачем это объединение схем нужно?*» и «*Как от простого объединения логических элементов получить качественно новый уровень или какую-то пользу?*» Для получения качественно нового уровня от комбинации элементов блок-схем необходимо придумать «управление». Ситуация жизненная, а именно когда интенсивность движения на небольшом нерегулируемом перекрёстке сильно возрастает, появляется новое качество [122] перекрёстка, его делают регулируемым либо превращают в развязку. Так и здесь нам потребуется придумать «регулировщика». Чтобы было более понятно – давайте рассмотрим (придумаем вместе) устройство памяти на 2 бита. Чуть ранее мы изучили память на 1 бит (триггер), и логично было бы поместить два триггера рядом. Например, два *D*-триггера, объединив следующим образом: (см. справа)

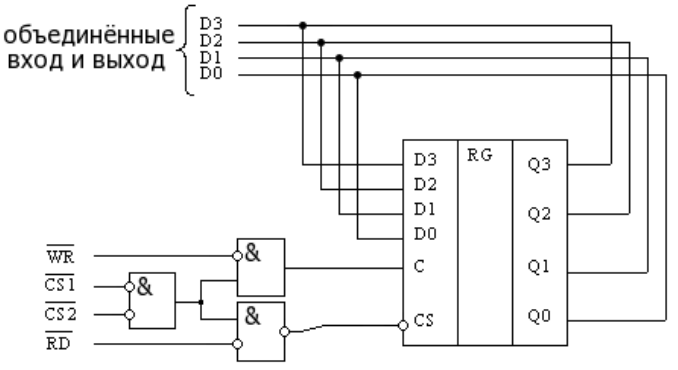


А если потребуется хранить 1 байт? – объединим восемь и т. д. Всё логично, пока мы не захотим хранить одновременно два, три, четыре или более байтов...

Если при хранении одного байта у схемы из расположенных рядом восьми *D*-триггеров получается 8 входов, 8 выходов и ещё один для тактовых импульсов, то сколько выводов должно быть у схемы, хранящей 2 байта? А сколько выводов потребуется для памяти объёмом в 16 ГБ?

Логично предположить, что создавать 16 входов для двух байтов неэффективно [123], разумнее оставить 8 входов и через ещё один вход управлять выбором схемы для хранения того или иного бита, назовём его адресом. Скажем, «0» – если нужно работать со схемой первого бита, и «1», если со схемой второго, наподобие того, как это реализовано в схеме выше. При таком подходе те же 16 входов можно использовать не для двух байтов, а для 256. По первым 8 передавать значение, а по вторым 8 передавать адрес нужного байта.

В целях экономии можно объединить и входы с выходами, например добавив ещё один контакт для выбора чтения или записи (Read/Write, $\overline{RD}/\overline{WR}$). Естественно, для хранения более чем двух бит таким образом потребуется более сложная схема, чем один элемент «НЕ».

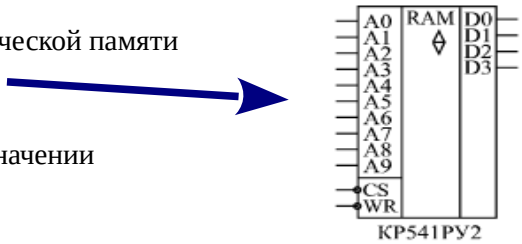


[122] Переход количества в качество.

[123] Мы сейчас не рассматриваем 16-, 32-, 64-, 128-битные и т. д. архитектуры, где указанное число бит передаётся параллельно.

Фактически мы получили ячейку памяти статического ОЗУ [124] с тремя состояниями выхода[125], выполненную из регистра памяти, дополненного простейшей логикой управления. На схеме полученная «ячейка памяти» имеет двунаправленные выводы *D*0…*D*3, по которым она либо принимает записываемое слово (в нашем случае 4-битовое для простоты) в режиме записи, либо выдаёт записанный код на выход в режиме чтения. Ячейка запоминает входной код в регистре, если сигналы («$\overline{WR}$», «ЗАПИСЬ») и («$\overline{RD}$», «ЧТЕНИЕ») равны, соответственно, «0» и «1». Если же $\overline{WR}$ = 1, а $\overline{RD}$ = 0, ячейка выдаёт на выводы *D*0…*D*3 слово, хранящееся в регистре. Как при записи, так и при чтении ячейка должна быть выбрана сигналами $\overline{CS1}$ = $\overline{CS2}$ = 0. В противном случае эти процессы будут блокированы.

В более общем случае схема статической памяти может выглядеть, например, так:



КР541РУ2

Буквы «РУ» в отечественном обозначении традиционно обозначают память.

Несмотря на то что указанного выше материала должно быть достаточно для первичного понимания того, как устроены современные ЭВМ, огорчим читателей, потому как указанная память по большей части не имеет ничего общего с той памятью, что стоит у них в персональных компьютерах и различных переносных устройствах. И дело вот в чём. Существуют два класса ОЗУ: статические и динамические. Конечно, статическая память используется, например, в регистрах процессора, частично в кэш-памяти, но привычная пользователям по прайс-листам оперативная память, которую мы измеряем на сегодня гигабайтами, является динамической. Для большинства пользователей и даже для процессора чаще всего всё выглядит прозрачно, потому как специальная схема контроллера памяти незаметно для нас регулярно регенерирует значения ячеек. Например, как жена, дожидающаяся мужа с работы и не знающая, когда он придёт, подогревает ему ужин по нескольку раз, чтобы по приходу он был максимально быстро подан тёплым. Естественно, мужу невдомёк, какие усилия затрачиваются в его отсутствие, как и процессору – чем там занят контроллер памяти.

Но обратная связь всё же есть... и в технике тоже, косвенно указанный нюанс регенерации может сказываться на том, что разная память имеет разные «тайминги», а это уже влияет на производительность и её цену в прайс-листе.

4.5.4. Другие схемы

Для полноценного построения небольшой завершённой вычислительной схемы потребуются и другие «цифровые» схемы, как то-то сумматор (полусумматор), регистр сдвига, шифратор, дешифратор, мультиплексор, тактовый генератор и др. Эти схемы не очень сложные и могут быть без труда найдены и изучены читателями самостоятельно.

[124] Оперативное запоминающее устройство. При отключении напряжения электропитания сохранённая информация теряется.

[125] Рассмотрение третьего высокоимпедансного Z-состояния выходит за рамки книги. Подробнее см. стр 511–512 в [1].

4.5.5. Алгоритмизация

Если схема простая, то обычно не составит труда одному инженеру на пальцах убедить другого что она работает. Например, что можно из базовых логических элементов собрать одно или восьми разрядный двоичный сумматор.

Если схема сложнее, то наверно убедить другого или убедиться самому, что схема будет работать, также можно. Для этого существует метод декомпозиции, заключающийся в том, что сложная схема делится на части (в идеале независимые) и далее работа каждой части рассматривается отдельно. Например, если планируется сделать калькулятор, то наверно такими частями будут блоки: сложения, вычитания, умножения, деления, памяти, ввода и вывода (индикации). Если необходимо считать всякие синусы, косинусы и другие функции, то потребуются дополнительные блоки, а чтобы понять как они работают придётся вспомнить курс математики, что любая (или не любая?) функция с оговорённой точностью может быть разложена в ряд.

В ряде случаев, для получения итогового математического результата достаточно будет просуммировать первые n членов этого ряда, а в каких-то случаях суммируемых членов может потребоваться k штук, причём $k>n$. По мере усложнения задачи предугадать успешность её решения, а тем более точное ожидаемое время решения ставится сложнее.

Для решения задачи необходимо создавать более точное описание её решения или алгоритм.

Алгоритм – описание последовательности действий для решения поставленной задачи.

Исторически термин «алгоритм» произошёл от имени средневекового математика, которого в Европе называли Мухаммед ибн Муса ал-Хорезми (787 – ок. 850 г. н. э., справа изображён на советской марке.)

Слово ал-Хорезми в имени этого знаменитого человека означает, что он является уроженцем Хорезма, местности в нынешнем Узбекистане. Одна из его работ была посвящена описанию индийской системы записи чисел. Арабский оригинал этой работы потерян, но имеется латинский перевод XII столетия. Эта книга была одним из источников, с помощью которых Западная Европа познакомилась с десятичной позиционной системой. Заглавие перевода: «*Об индийском числе, сочинение Алгоризми*» (*Algorizmi de numero Indozum*). В других рукописях автор именовался Algorismus и Algorithmic, что ввело в математический язык термин «*алгоритм*» – латинизированное имя автора.

Долгое время этот термин включал в себя понятие о четырёх типах арифметических действий, или, более широко, о процессе вычислений и не привлекал к себе особого внимания. Каждому оригинальному процессу вычислений присваивалось имя (алгоритм Евклида, алгоритм деления с остатком, алгоритм нахождения простых чисел и т. п.), и обсуждением общих свойств этих алгоритмов не занимались.

Потребность уточнения понятия алгоритма сформировалась в 30-х годах XX столетия, что было обусловлено, по-видимому, осознанием математиками реальных возможностей выполнения сложных вычислений с помощью машин. Это привело к разделению понятия алгоритм на интуитивное (нестрогое) и точное (математические модели).

4.5.5.1. Интуитивное определение алгоритма

Самое простое определение алгоритма можно встретить у Ю.И. Манина в [40, стр.5]:

Алгоритм – это текст, который в определённых обстоятельствах может привести к однозначному развитию событий – процессу выполнения алгоритма. Фермент, катализирующий специфическую реакцию, устав караульной службы или программа ЭВМ – примеры алгоритмов в этом широком смысле слова.

Более формальное определение: «*Алгоритм – точное предписание, которое задаёт вычислительный процесс (называемый в этом случае алгоритмическим), начинающийся с произвольного исходного данного (из некоторой совокупности возможных для данного алгоритма исходных данных) и направленный на получение полностью определяемого этим исходным данным результата...*» [126].

А. И. Мальцев [127] отмечает следующие характерные свойства алгоритмов:

1. Алгоритм – это процесс последовательного построения величин, идущий в дискретном времени таким образом, что в начальный момент задаётся исходная конечная система величин, а в каждый следующий момент система величин получается по определённому закону (программе) из системы величин, имевшихся в предыдущий момент времени (дискретность алгоритма).
2. Система величин, получаемых в какой-то (не начальный) момент времени, однозначно определяется системой величин, полученных в предшествующие моменты времени (детерминированность алгоритма).
3. Закон получения последующей системы величин из предшествующей должен быть простым и локальным (элементарность шагов алгоритма).
4. Если способ получения последующей величины из какой-нибудь заданной величины не даёт результата, то должно быть указано, что надо считать результатом алгоритма (направленность алгоритма).
5. Начальная система величин может выбираться из некоторого потенциально бесконечного множества (массовость алгоритма).

Обычно в литературе в качестве примеров приводятся два типа инструкций (правил), которые называют алгоритмами: первый тип – это стандартные математические операции (см. пример в следующем параграфе); ко второму относятся простые житейские ситуации. В качестве примеров из жизни часто приводят рецепт алгоритм приготовления какого-нибудь блюда, правила перехода улицы и т. п. С учётом рассмотренных выше свойств алгоритмов подобные примеры представляются неправильными, так как уровень формализации в них, как правило, недостаточен для получения результата, полностью определяемого исходными данными. При формализации человек абстрагируется от содержательной стороны дела, а в обыденной жизни только она его и интересует. При передаче рецепта люди «подстраиваются» друг к другу, непонятные элементы рецепта разъясняются с использованием нестрого определяемых понятий (посолить по вкусу или когда тесто поднимается), и всё же хороший торт или пирог, как правило, с первого раза не получается [128].

[126] Математический энциклопедический словарь. – М.: Советская энциклопедия, 1988.
[127] Мальцев Анатолий Иванович (1909–1967) – математик, академик АН СССР.
[128] Вспомним русскую пословицу «Первый блин комом» (Или «Первый блин комам, второй – знакомым,

Дональд Эрвин Кнут (англ. Donald Ervin Knuth, родился 10 января 1938 г.) – американский учёный. Автор всемирно известной серии книг, посвящённой основным алгоритмам и методам вычислительной математики. Удостоен многочисленных премий и наград в области программирования и вычислительной математики, среди которых премия Тьюринга (1974).

Толщина книг (объёмность, впрочем, как и фундаментальность) серии «Искусство программирования» Д. Э. Кнута зачастую является основой добрых шуток среди студентов и преподавателей вроде «Метод Кнута (кнута) и пряника», по аналогии с отсылкой ленивых к документации (RTFM).

По мнению крупнейшего специалиста по компьютерной математике Д. Кнута, современное значение слова «*algorithm*» во многом аналогично таким понятиям, как рецепт, процесс, метод, способ, процедура, программа, но всё-таки слово это имеет дополнительный смысловой оттенок. Алгоритм – это не просто набор конечного числа правил, задающих последовательность выполнения операций для решения задачи определённого типа. Помимо этого, он имеет пять важных свойств:

1. Конечность (результативность). Алгоритм всегда должен заканчиваться после выполнения конечного числа шагов [129]. Как следствие это определяет дискретность его шагов или его самого – алгоритм должен представлять процесс решения задачи как последовательное выполнение простых операций.
2. Определённость (однозначность). Каждый шаг алгоритма должен быть точно определён. Если алгоритм описывается обычным языком, то существует возможность неточного понимания, для устранения неточности понимания существуют языки программирования. Плавила и порядок выполнения операций (шагов) алгоритма не допускают неоднозначного толкования и тем самым определяют его однозначность. Как следствие, повторение выполнения алгоритма даёт те же самые результаты при неизменности входных значений.
3. Ввод. Алгоритм имеет некоторое (возможно, равное нулю) число входных данных, то есть величин, задаваемых до начала работы алгоритма или определяемых во время его работы. Как следствие, это определяет его универсальность (массовость). Алгоритм решения задачи разрабатывается в общем виде и оказывается применим для некоторого класса задач, различающихся лишь исходными данными. При этом исходные данные могут выбираться из некоторой области, которая называется областью применимости алгоритма.

третий – родне, а четвёртый – мне!»?).

[129] Процедура, обладающая всеми характеристиками алгоритма, за исключением конечности, называется методом вычислений. Алгоритм может содержать один или несколько методов вычислений, в этом случае он будет либо останавливаться из-за невозможности получить решение с выдачей соответствующего сообщения, либо неограниченно продолжаться в течение времени, отведённого для исполнения алгоритма, с выдачей промежуточных результатов, а по прошествии этого времени также завершаться.

4. Вывод. У алгоритма есть одно или несколько выходных данных, то есть величин, имеющих определённую связь с входными данными.
5. Эффективность. Алгоритм обычно считается эффективным, если все его операторы достаточно просты для того, чтобы их можно было точно выполнить в течение конечного промежутка времени с помощью компьютера.

Каждый алгоритм задаёт алгоритмический процесс, схема которого представлена на рис. 4.28.

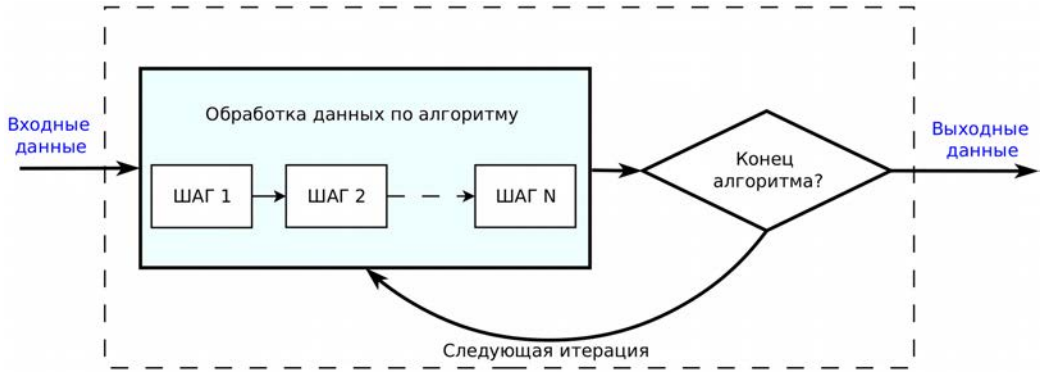


Рисунок 4.28. Схема алгоритмического процесса

4.5.5.2. Математические определения алгоритма

В середине 30-х годов XX века были разработаны три типа математических моделей алгоритма [2]. Первый тип моделей основан на понятии рекурсивной функции. Второй – на основе описания детерминированного устройства, работающего по шагам и выполняющего на каждом шаге заранее определённые операции с элементами устройства и данными (машины Поста, Тьюринга). Третий тип моделей связан с работой со словами в некотором фиксированном алфавите, которые с помощью подстановок переходят в другие слова (нормальный алгоритм Маркова).

Определение алгоритма с помощью рекурсивных функций было сделано английскими математиками А. Чёрчем [130] и С. Клини [131]. Оно достаточно сложно, связано с разделом математики – теорией вычислимых функций – и выходит за рамки курса информатики.

Пример: классический алгоритм Евклида (разработан в III-м веке до н. э.) для вычисления наибольшего общего делителя двух натуральных чисел, он приводится практически в каждой книге по программированию.

Дано два целых положительных числа. Требуется найти наибольшее положительное целое число, на которое нацело делятся заданные числа.

1. Сравнить два исходных числа, если они равны, то взять любое число в качестве

[130] Алонзо Чёрч (англ. Alonzo Church; 1903–1995) – выдающийся американский математик и логик, внёсший значительный вклад в основы информатики. Прославился разработкой теории лямбда-исчислений, последовавшей за его знаменитой статьёй 1936 года, в которой он показал существование т. н. «неразрешимых задач». Эта статья предшествовала знаменитому исследованию Алана Тьюринга на тему проблемы остановки, в котором также было продемонстрировано существование неразрешимых задач.

[131] Клини (Kleene) Стивен Коул (р. 1909), американский логик и математик. Основные работы посвящены теории алгоритмов и рекурсивных функций, а также проблемам интуиционистской логики и математики.

результата и закончить выполнение алгоритма; если числа не равны, то перейти к п. 2.
2. Определить большее из двух чисел.
3. Заменить большее число на разность большего и меньшего чисел.
4. Перейти к п. 1.

4.5.5.3. Основные свойства и формы описания алгоритмов

Описание алгоритма может быть представлено в виде:
1) текстовой инструкции;
2) графической схемы;
3) программы на одном из языков программирования.

Следуя из особенностей алгортима, введённых Д.Э.Крутом, каждый алгоритм должен иметь:
1) название, отражающее суть решаемой задачи,
2) описание исходной информации,
3) описание последовательности действий,
4) описание выходной информации.

Существуют три основных типа базовых структурных схем организации выполнения операторов в программах для реализации отдельных частей или всего алгоритма:
1) линейная – неизменная последовательность операций от начала до конца без повторов действий;
2) разветвляющаяся – последовательность выполняемых действий может изменяться в зависимости от каких-либо условий;
3) циклическая – группа операций может выполняться многократно с кратностью повтора определяемой некоторым условием.

Для графического представления сложных алгоритмов используются стандарты группы **IDEF**, языки **UML**, **BPML**, **UEML**.

Для элементарных алгоритмов можно использовать так называемые «структурные схемы», на которые ранее существовал стандарт ГОСТ 19.701–90 [132].

Структурные схемы типовых алгоритмов могут быть представлены следующим образом (см. Рис. 4.29, 4.30):

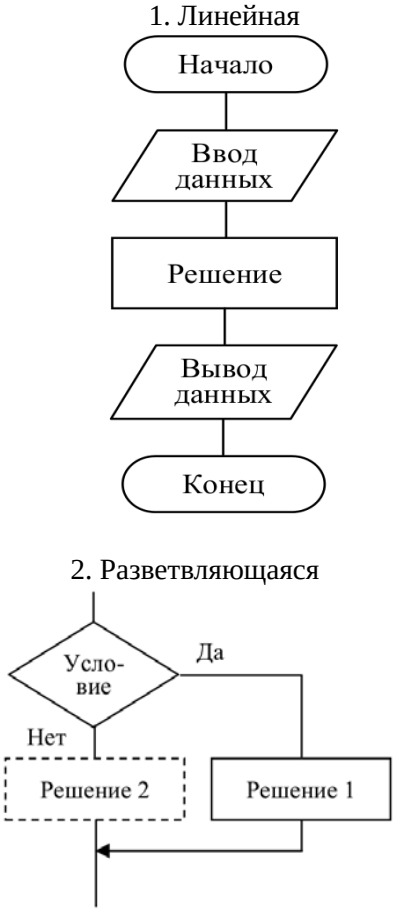


Рисунок 4.29. Типовые структурные схемы

[132] ГОСТ 19.701–90 *Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения.* Статус документа: действующий. Введён в действие с 01.01.1992.

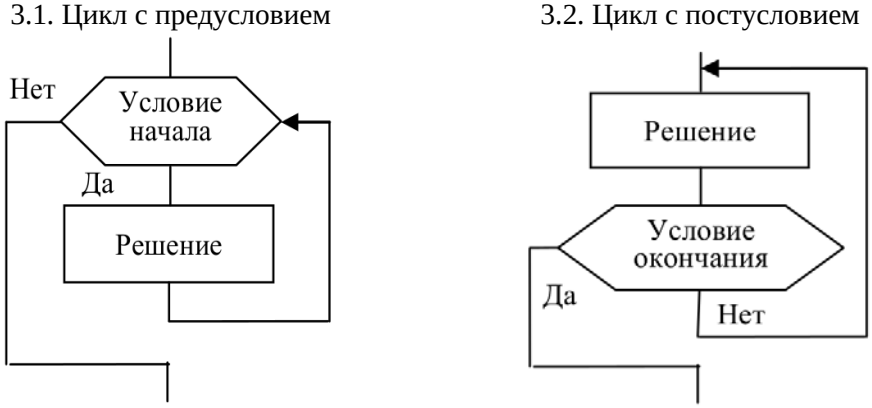
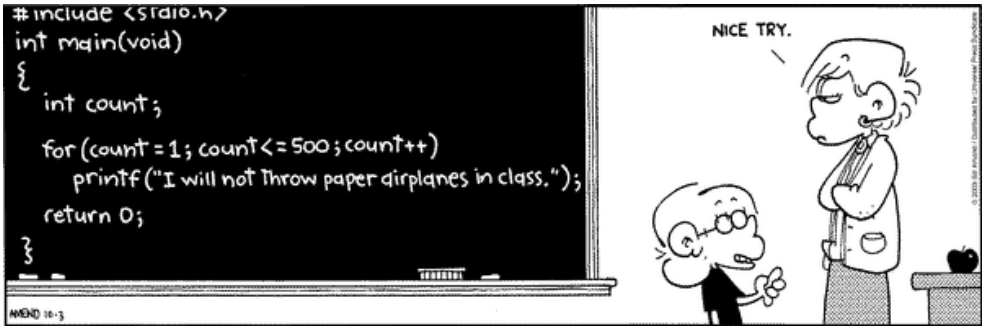


Рисунок 4.30. Типовые структурные схемы циклов

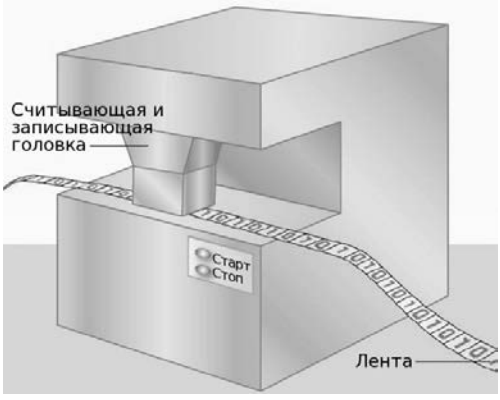
Замечание. Синтаксис некоторых языков программирования позволяет дополнительно реализовать более удобный варианта цикла с предусловием – «цикл с параметром».

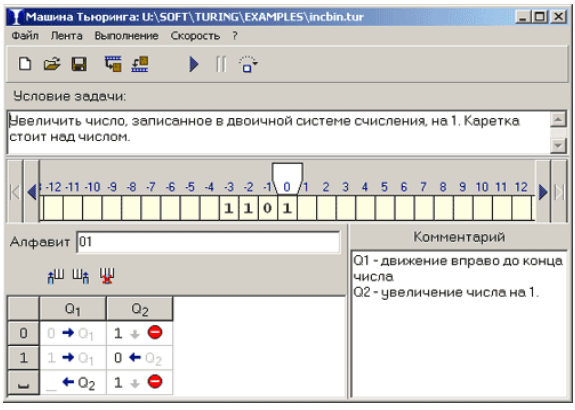


4.5.6. Машина Тьюринга, машина Поста

Любой алгоритм в конечном итоге придумывается не для красоты теории, а для того чтобы быть реализованным. Вот тут и пригодились труды Алана Тьюринга и придуманная им абстрактная машина Тьюринга.

Замечание. Бесплатную программу-эмулятор машины Тьюринга, с примерами программ для неё, можно найти на сайте учителя Полякова Константина Юрьевича kpolyakov.narod.ru:





Также см. [6] – учебно-методическое пособие по решению задач.

Математики Алан Тьюринг (Англия) и Эмиль Пост (США) независимо друг от друга в 1936 году попытались уточнить понятие алгоритма и определить точно класс вычислимых функций. Основная их идея заключалась в том, что алгоритмические процессы – это процессы, которые может совершать специально построенная «машина» (устройство, автомат). Они описали в точных математических терминах довольно узкие классы машин, но на этих машинах оказалось возможным реализовать все алгоритмические процессы, которые когда-либо описывались математиками. Модели, представленные этими машинами, было предложено рассматривать в качестве математических «представителей» вообще всех алгоритмов. Машинами эти математические построения называют потому, что при построении используются некоторые понятия реальных электронно-вычислительных машин – память, команда, программа.

А́лан Мэ́тисон Тью́ринг (англ. Alan Mathison Turing; 1912–1954) – английский математик, логик, криптограф, оказавший существенное влияние на развитие информатики.

Замечание. Кстати, высшей наградой за заслуги в области информатики является премия Тьюринга. Премия учреждена Ассоциацией вычислительной техники [133] в честь выдающегося английского учёного Алана Тьюринга, получившего первые глубокие результаты относительно вычислимости задолго до появления первых электронных вычислительных машин.

В сфере информационных технологий премия Тьюринга имеет статус, аналогичный Нобелевской премии в академических науках, она ежегодно вручается одному или нескольким специалистам в области информатики и вычислительной техники, чей вклад в этой области оказал сильное и продолжительное влияние на компьютерное сообщество.

Машина Поста состоит из не ограниченной в обе стороны ленты, разделённой на ячейки, которые последовательно пронумерованы целыми числами, как положительными, так и отрицательными. Лента играет роль памяти. В каждой ячейке ленты стоит либо признак того, что в ячейке записана метка, либо ячейка пустая. Состояние ленты – это данные о том, какие ячейки заняты, а какие пусты.

Кроме ленты, имеется головка чтения/записи, которая:

- умеет двигаться вперёд, назад и стоять на месте;
- умеет читать содержимое, стирать и записывать метку;
- управляется программой, в которую могут входить в любой комбинации и любом количестве шесть команд:
  1) вправо;
  2) влево;
  3) поставить метку;
  4) стереть метку;
  5) передача управления на один номер команды в программе, если в текущей ячейке есть метка; если метки нет, то передача управления на другой номер команды;
  6) прекращение работы.

[133] Ассоциация вычислительной техники (англ. Association for Computing Machinery, ACM) – старейшая и наиболее крупная международная организация в компьютерной области, основана в 1947 г. http://www.acm.org/.

Основные отличия машины Поста и машины Тьюринга чисто технические. Машина Тьюринга эквивалентна машине Поста по своим возможностям, что доказано математически.

Если для решения задачи можно построить машину Поста или машину Тьюринга, то она алгоритмически разрешима, то есть существует алгоритм её решения. Можно ли любой алгоритм представить в форме машины Поста или Тьюринга, то есть можно ли условие задачи записать в терминах соответствующей машины Поста и построить систему команд, приводящую к результату?

Существуют два утверждения:

1. Всякий алгоритм представим в форме машины Поста (тезис Поста).
2. Всякий алгоритм представим в форме машины Тьюринга (тезис Тьюринга).

Эти утверждения невозможно доказать, поэтому они называются «тезисами». В них фигурируют, с одной стороны, интуитивное понятие «всякий алгоритм», а с другой стороны – точные понятия «машина Поста» и «машина Тьюринга». Тезисы Поста и Тьюринга по своему характеру очень похожи на гипотезы физики об адекватности математических моделей и физических явлений. Эти гипотезы, как и тезисы, подтверждаются практикой.

Позднее, в 1946 году, Джоном фон Нейманом на летней сессии Пенсильванского университета был распространён отчёт, заложивший основы развития вычислительной техники на несколько десятилетий вперёд. Последующий опыт разработки ЭВМ показал правильность основных выводов Неймана, которые, естественно, в последующие годы развивались и уточнялись.

Пост, Эмиль Леон (1897–1954) – американский математик и логик; один из основателей многозначной логики (1921); основные труды по математической логике: алгебра Поста, классы Поста функций алгебры логики; предложил абстрактную вычислительную машину – машину Поста.

Машина Поста, несмотря на внешнюю простоту, может производить различные вычисления, для чего надо задать начальное состояние машины и программу, которая эти вычисления сделает. Состояние машины – это состояние ленты и положение головки чтения/записи.

Джон фон Не́йман
(англ. *John von Neumann*; нем. *Johann von Neumann*; при рождении Я́нош Ла́йош Не́йман, венг. *Neumann János Lajos*)

(1903–1957) – венгеро-американский математик еврейского происхождения, сделавший важный вклад в квантовую физику, квантовую логику, функциональный анализ, теорию множеств, информатику, экономику и другие отрасли науки.

Джон фон Нейман наиболее известен как праотец современной архитектуры компьютеров (так называемая архитектура фон Неймана), применением теории операторов к квантовой механике (алгебра фон Неймана), а также как участник Манхэттенского проекта и как создатель теории игр и концепции клеточных автоматов.

4.6. Структура классической ЭВМ

Старый компьютер лучше новых двух.

Основные рекомендации, предложенные фон Нейманом для разработчиков ЭВМ (принципы построения ЭВМ):

1. Машины на электронных элементах должны работать не в десятичной, а в двоичной системе счисления [134].

2. Программа должна размещаться в одном из блоков машины – в *запоминающем устройстве* (ЗУ), обладающем достаточной ёмкостью и соответствующими скоростями выборки и записи команд программы.

3. Программа, так же как и числа, с которыми оперирует машина, представляется в двоичном коде. Таким образом, по форме представления команды и числа однотипны. Это обстоятельство приводит к следующим важным последствиям:

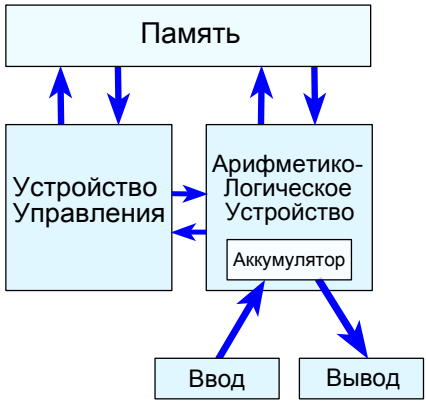


Рисунок 4.31. Структура классической ЭВМ, согласно архитектуре Джона фон Нейманаф

промежуточные результаты вычислений, константы и другие числа могут размещаться в том же ЗУ, что и программа;

числовая форма записи программы позволяет машине производить операции над величинами, которыми закодированы команды программы.

4. Трудности физической реализации ЗУ, быстродействие которого соответствовало бы скорости работы логических схем, требуют иерархической организации памяти.

5. Арифметические устройства машины конструируются на основе схем, выполняющих операцию сложения. Создание специальных устройств для вычисления других операций нецелесообразно.

6. В машине используется параллельный принцип организации вычислительного процесса (операции над словами производятся одновременно по всем разрядам).

ЭВМ, построенная по принципам, определённым фон Нейманом, состоит из следующих основных блоков (см. рис. 4.31): запоминающего устройства (память), арифметико-логического устройства (АЛУ) и устройства управления (УУ).

[134] По мнению авторов, требование об обязательности использования двоичной системы счисления на данном этапе излишне, так как существовали и существуют ЭВМ, работающие в троичной системе счисления, в том числе отечественного производства. Так, прообраз первой ЭВМ «Сетунь» был создан в ВЦ МГУ в 1958 году, а годом позже она стала серийной. Вопрос лишь в том, что схемотехническое решение оказывается более сложным, и поэтому указанные ЭВМ не получают широкого распространения. Также следует пересмотреть вопросы кодирования числовых, текстовых, звуковых, видео- и других данных в троичной системе счисления.

Запоминающее устройство, или память – это совокупность ячеек, предназначенных для хранения некоторого кода. Устройство последних было рассмотрено выше. Каждой из ячеек присвоен свой номер, называемый **адресом**. Информацией, записанной в ячейке, могут быть как команды в машинном виде, так и данные.

Машинная команда – это двоичный код, определяющий выполняемую операцию, адреса используемых операндов и адрес ячейки ЗУ, по которому должен быть записан результат выполненной операции. По сути, это и есть основа работы ЭВМ. Внутри памяти ЭВМ, кроме закодированных в двоичном виде данных, хранятся номера команд, которые процессор может считать и выполнить. Эти команды называются сегодня машинным кодом и очень сильно пересекаются с языком ассемблера под ту или иную архитектуру. При рассмотрении вопросов создания программного обеспечения мы вернёмся к этому вопросу.

Операции, определяемые кодом операции команды, выполняются в **арифметико-логическом устройстве** (АЛУ).

Все действия в ЭВМ выполняются под управлением сигналов, вырабатываемых **устройством управления** (УУ). Управляющие сигналы формируются на основе информации, содержащейся в выполняемой команде, и признаков результата, сформированных предыдущей командой (если выполняемая команда является, например, командой условного перехода). Устройство управления, помимо сигналов, определяющих те или иные действия в различных блоках ЭВМ (например, вид операции в АЛУ или сигнал считывания из ЗУ), формирует также адреса ячеек, по которым производится обращение к памяти для считывания команды и операндов и записи результата выполнения команды.

Устройство управления формирует адрес команды, которая должна быть выполнена в данном цикле, и выдаёт управляющий сигнал на чтение содержимого соответствующей ячейки запоминающего устройства. Считанная команда передается в УУ. По информации, содержащейся в адресных полях команды, УУ формирует адреса операндов и управляющие сигналы для их чтения из ЗУ и передачи в арифметико-логическое устройство. После считывания операндов устройство управления по коду операции, содержащемуся в команде, выдаёт в АЛУ сигналы на выполнение операции. Полученный результат записывается в ЗУ по адресу приёмника результата под управлением сигналов записи. Признаки результата (знак, наличие переполнения, признак нуля и т. д.) поступают в устройство управления, где записываются в специальный регистр признаков. Эта информация может использоваться при выполнении следующих команд программы, например команд условного перехода.

Полагаем, что изложенного выше материала достаточно для получения читателями общего представления о работе современных ЭВМ (в том числе и отдельных её узлов), а дальнейшее абстрактное теоретическое углубление в данном направлении нецелесообразно, поэтому перейдём к практике.

Простые пользователи обычно имеют дело с персональными компьютерами (**ПК**), поэтому в дальнейшем речь пойдёт именно о них.

Наиболее известны и распространены персональные компьютеры на базе процессоров фирм AMD, Intel и их аналогов (ранее их называли IBM-совместимые ПК) и ПК фирмы Apple – Macintosh.

В последние годы всё более заметна идея миниатюризации ЭВМ и её основных элементов во времени «из прошлого в будущее». Широкое распространение получают ноутбуки, нетбуки, ультрабуки, планшеты и «телефоны». Рассмотрим их подробнее.

Ноутбуки – переносные персональные компьютеры с размерами, например, 33×24×3,5 см, и массой порядка 2–2,4 кг (см. рис. 4.32).

Рисунок 4.32. Настольный ПК, ноутбук и планшет

Нетбуки – облегчённые (как в плане веса, так и в плане вычислительной мощности) версии переносных персональных компьютеров, ноутбуков, из которых исключены CD/DVD/Blue-Ray-приводы, а жёсткий диск предпочтительно заменён твердотельным носителем информации. Примерные размеры 29–33 × 20–22 × 1,4–2 см, вес 1,4–1,9 кг. Предполагаемая цель использования данных компьютеров – исключительно выход в интернет и запуск «лёгких» приложений.

Ультрабуки, по сути, ни чем не отличаются от нетбуков, за тем лишь исключением что при своих малых геометрических размерах (толщиной около 1,5–1,9 см.) и весе, меньшем 1,5 кг, они предоставляются пользователям вычислительную мощность сопоставимую с их настольными ПК.

Планшеты – нечто среднее между нетбуками и ультрабуками, когда из последних были убраны клавиатура и touchpad, которые заменил единый сенсорный дисплей. Как следствие компьютер стал «плоским» и нераскладываемым, потеряв тем самым из названия слово «book». Взаимодействие с планшетом производится путём нажатий и движений одним или несколькими пальцами по экрану планшета.

Отличительная особенность планшетов – это отличная от ПК, ноутбуков, нетбуков и ультрабуков операционная система, фактически привязанная к его «IBM не совместимой» аппаратной части. Чаще это одна из трёх: Android, iOS или WebOS, без возможности её замены/модификации пользователем. Упоминание о других ОС для планшетов можно найти в разделе 5.3. «Операционная система». Также в планшетах часто встречается GSM/CDMA-модуль для выхода в интернет через сотовые сети, модуль позиционирования GPS/ГЛОНАСС, несколько акселерометров и других датчиков.

В некоторых случаях можно встретить термин «планшетный ПК» и это оправданно, особенно при рассмотрении моделей с «пристёгиваемыми» (или «отстёгиваемыми», кому как больше нравится) клавиатурами, а также с x86 совместимыми процессорами.

Наблюдается интересная тенденция, что устройства с операционной системой iOS имеют лишь два разъёма: один – для наушников (minijack), а второй – для соединения с ПК/зарядки. Также они не имеют возможности прямого подключения внешних носи-

телей, таких как USB-флэшки или SD (microSD, SDHC, SDXC) карты памяти, в то время как устройства с ОС Android, наоборот, позволяют это делать, по крайней мере наблюдаемая тенденция такова.

Размеры планшетов сильно варьируются от больших, примерно 25 × 17 × 1 см, до маленьких 17 × 11 × 0,5–0,7 см. Вес первых, менее 1 кг, вторых – до 0,5–0,6 кг.

Современный телефон-компьютер – это фактически уменьшенный до размеров шоколадки планшет (хотя некоторые модели всё ещё имеют привычные кнопки), в котором обязательно присутствует функция телефона. Рынок операционных систем для телефонов несколько шире, чем для планшетов, например там можно уже встретить BlackBerry, Symbian, Windows Mobile/8/10, FirefoxOS и др. Но он также остаётся закрытым и привязанным к аппаратной части телефона.

Помимо встроенного микрофона и динамиков, практически все указанные портативные устройства сегодня оснащаются одной или двумя видео/фото/веб-камерами, разрешение которых может доходить до 12-16 мегапикселей.

Следующий этап миниатюризации – это встраивание компьютеров с привычным нам функционалом в дужки очков, наушники, часы, ручки, украшения и прочие носимые предметы обихода.

Работа любого вычислительного устройства зависит от двух составляющих: программной и аппаратной. Рассмотрим их подробнее, для этого введём определение:

Аппаратное обеспечение – все те компоненты, из которых состоит компьютер, а также оборудование для организации локальных и глобальных сетей – так называемое компьютерное «железо» (hardware).

Прежде всего это процессор, материнская плата и её главные микросхемы – чипсет, определяющий всю архитектуру компьютера, возможные типы основной оперативной памяти, видеокарт, дисковых устройств, мониторов, принтеров и других периферийных устройств.

Знание современных технических средств, а также истории их развития необходимо каждому пользователю ПК.

Замечание. Дополнительную информацию по истории развития средств вычислительной техники можно найти во многих учебниках по информатике, в интернете [5] и книгах [3], [4].

Обновление всех компонентов ПК идёт стремительными темпами. Не так давно появились многоядерные процессоры для настольных компьютеров и ноутбуков, а сейчас многоядерными выпускаются телефоны; новые шины для высокоскоростной работы с памятью, видеоподсистемой, жёсткими дисками; жёсткие диски всё чаще оказываются твердотельными, а не магнитными; интегрированные контроллеры для работы с гигабитными проводными сетями и с беспроводными сетями. Жидкокристаллические мониторы вытеснили с рынка мониторы на электронно-лучевых трубках. А теперь на их место покушаются мониторы с сенсорным дисплеем и повышенным разрешением. Разрешения экранов телефонов порой в несколько раз превышают последние у стационарных мониторов. Продвигается новая концепция домашнего компьютера – медиацентра, который не только работает с Blue-Ray и DVD-дисками и имеет качественную многоканальную аудиосистему, но и работает с радио- и телевизионными трансляциями интернета (в том числе платными сервисами, так как капиталисты и тут хотят по максимуму заработать на пользователях, подталкивая их к сверхпотребле-

нию), а также с FM и TV-тюнерами, которыми комплектуется ПК, с программируемым управлением этими средствами.

Умение подобрать необходимую конфигурацию ПК или осознанно выбрать ноутбук (или нетбук) позволит избежать покупки морально устаревшего оборудования, цена которого может ненамного отличаться от новой, более совершенной техники. Так, в компьютерных магазинах часто можно встретить типовые конфигурации ПК с морально устаревшими процессорами и с жёсткими дисками не самых оптимальных объёмов (часто новый диск с ёмкостью, в 2 раза большей, чем более старый, бывает дороже всего лишь на 10–20%). К сожалению, указанная тенденция определяется не здравым смыслом и техническим прогрессом, а желанием получения производителями сверхприбыли. Так, зачастую пользователи вынуждены выбрасывать работающие устройства лишь потому, что они не обеспечивают им нового имиджа или новых, разрекламированных свойств. Зачастую оказывается, что производители, как по сговору, отказались выпускать устройства для той или иной технологии, шины и прочего.

Запросы большинства пользователей год от года остаются постоянными и зачастую конечными... «ходить по сайтам», смотреть почту, фильмы, слушать музыку, печатать документы, позднее добавилось – общаться голосом с друзьями (взамен телефона), видеосвязь, игры. Вопрос лишь в том, что для обеспечения функции записи видео обоснованно использовать большой жёсткий диск... а вот для просмотра сайтов подойдёт любой. Вместо осознанного выбора по необходимости чаще выбор идёт по толщине кошелька. Не спешите покупать, проконсультируйтесь. Надеемся, что читатели не будут себе покупать то, что не очень им нужно, а мы поможем разобраться в большинстве устройств, предлагаемых сегодня на рынке. Лучше потратьте деньги на что-то более полезное (своё здоровье, детей, путешествия, образование и прочее), потому как «рынок постарается» и все ваши «железки» всё равно устареют, если не через год, так через два.

Народная мудрость *гласит, что для эффективного решения проблем «врачи, юристы и компьютерщики должны быть свои», или в крайнем случае человек сам должен немного разбираться в указанных областях. Так что, чтобы ваш ПК всегда быстро работал и оперативно выполнял все необходимые функции, необходимо самому пользователю разбираться в основном аппаратном обеспечении и соответствующем ему программном обеспечении.*

4.7. Процессор

Процессор [135] – специальная микросхема, которая выполняет операции по обработке информации в компьютере.

На небольшой кремниевой пластине размещены сотни миллионов транзисторов-переключателей и каналов передачи данных. Кроме центрального процессора (**CPU**), в современных компьютерах значительную роль играет процессор видеокарты (**GPU**), который занимается обработкой видеоинформации. В первом приближении алгоритмы их работы схожи и легки для понимания.

Имеющиеся транзисторы составляют отдельные небольшие функциональные блоки:

[135] На сленге называется «камнем».

- специальная память в виде регистров (размер зависит от разрядности);
- специальная память в виде флагов (ячейки хранящие 1 бит);
- блок выполняющий арифметические операции;
- блок выполняющий копирование данных из общей оперативной памяти в регистры и обратно;
- и другие.

Также в процессоре есть шина данных и шина команд. На первой можно выставить данные (значения чисел и др.), а на второй номер команды (инструкции), поясняющей процессору что следует делать с данными. Результат выполнения команды, если необходимо, может также выставляться на шину данных в виде логических «0» и «1», так что последующая команда сможет воспользоваться данными полученными на предыдущем шаге.

Есть модели процессоров, где шина данных и шина команд совмещены, большинство персональных компьютеров именно так и устроены. Разделение происходит «по времени». Вначале идут команды, потом данные.

Представьте, что процессор так сделан, что в него «зашита» таблица выполняемых им инструкций примерно следующего содержания.

Таблица 4.4. Таблица команд (инструкций) вымышленного процессора

Номер инструкции (команды)	Длина операндов в байтах	Описание
01	1	Сложение регистра AX с «операндом 1» команды и помещение результата сложения в регистр AX
02	1	Умножение регистра AX на «операнд 1» команды и помещение результата умножения в регистр AX
03	0	Поменять местами регистры AX и BX.
04	2	Поместить данные из ячейки адрес которой записан в «операнде 1» команды (размер 2 байта) в регистр AX.
...		и т.д.

Тогда последовательность байт «01 22 02 23» есть простая программа из двух команд, сообщающая процессору, сложить число «22» с тем, что имеется в регистре AX, а затем умножить имеющееся там значение на 23.

В первых реализациях все инструкции определялись исключительно исходя из внутренней аппаратной реализации процессора. В случае обнаружения ошибок в работе тех или иных команд их было невозможно никак исправить. Позднее появилось понятие микрокода (внутренней прошивки процессора). Обновлённый микрокод процессора не может исправить ситуацию с ошибками кардинально, но может подправить таблицу команд процессора до ближайшей холодной перезагрузки. Сделано это для того, чтобы через микрокод нельзя было в процессор записать вредоносное ПО (вирусы) и тем самым привести в систему в состояние невозможности включиться. Если известны

ошибки и производителем процессора выпущен микрокод, то система обычно обновляет его каждый раз при загрузке системы, причём на каждом ядре процессора.

```
$ dmesg|grep -i microcode
[0.000000] microcode: microcode updated early to revision 0x20, date = 2016-03-16
[2.558771] microcode: CPU0 sig=0x306c3, pf=0x2, revision=0x20
[2.558963] microcode: CPU1 sig=0x306c3, pf=0x2, revision=0x20
...
[2.560381] microcode: Microcode Update Driver: v2.01 <tigran@aivazian ...
```

Для простоты написания программ каждому процессору (точнее каждой строчке в таблице зашитых в него команд) можно сопоставить свой язык ассемблера, – набор осмысленных слов или словосочетаний, например «add», «mul», «exchange», «mov» и др. В результате программа «01 22 02 23» написанная на машинном коде конкретного процессора будет выглядеть более читаемо на языке ассемблера того же самого процессора как «add 22 mul 23», а если ещё добавить для удобства символы перевода строк, то получится что-то вида

```
add 22
mul 23
```

При рассмотрении заложенных в процессор команд и особенностей их выполнения всё многообразие процессоров, используемых компьютерной промышленностью массового сегмента, можно поделить на две основные ***архитектуры*** (по внутренней организации и наборам команд). Это архитектуры ***CISC*** и ***RISC***.

CISC	RISC
англ. *Complete Instruction Set Computer*, – компьютер со сложным (полным) набором команд	англ. *Restricted (Reduced) Instruction Set Computer,* – компьютер с сокращённым (более простым) набором команд

Кроме «ширпотреба» следует также выделить ***архитектуры: MISC, VLIW, EPIC*** и другие**.**

MISC	VLIW	EPIC
англ. *Minimal Instruction Set Computer* – компьютер, работающий с минимальным набором длинных команд. Увеличение разрядности процессоров привело к идее укладки нескольких команд в одно большое слово.	англ. *Very long instruction word* – «очень длинная машинная команда», архитектура процессоров с несколькими вычислительными устройствами. Характеризуется тем, что одна инструкция процессора содержит несколько операций, которые должны выполняться параллельно.	англ. *Explicitly Parallel Instruction Computing* – поддерживает явный параллелизм на уровне команд. EPIC представляет собой пример научно-исследовательской разработки, описывающий собой скорее принцип обработки информации, нежели архитектуру конкретного процессора.

Основоположником CISC-архитектуры можно считать компанию IBM с её базовой архитектурой /360, ядро которой используется с 1964 года и дошло до наших дней, например в таких современных мейнфреймах, как IBM ES/9000.

CISC – концепция проектирования процессоров, которая характеризуется следующим набором свойств:

- нефиксированное значение длины команды;
- арифметические действия кодируются в одной команде;
- небольшое число регистров, каждый из которых выполняет строго определённую функцию.

CISC-архитектура является практическим стандартом для рынка микрокомпьютеров. Для CISC-процессоров характерно: сравнительно небольшое число регистров общего назначения; большое количество машинных команд, некоторые из которых нагружены семантически аналогично операторам высокоуровневых языков программирования и выполняются за много тактов; большое количество методов адресации; большое количество форматов команд различной разрядности; преобладание двухадресного формата команд; наличие команд обработки типа «регистр – память».

До последних дней для производства ПК использовались только CISC-процессоры. Лидерами по производству указанных чипов считались фирмы AMD и Intel. Они бы ими и остались, если бы не их последние процессорные разработки, которые по формальным признакам стоит классифицировать уже как CISC-процессоры с RISC-ядром, что не позволяет явно и однозначно отнести новые процессоры (выше чем Pentium IV или Athlon) ни к одной из архитектур.

Использование процессоров не ограничивается их использованием в ПК, так, RISC-архитектура является основой многих устройств, современных рабочих станций и высокопроизводительных серверов (суперкомпьютеров). Зачатки этой архитектуры уходят своими корнями к компьютерам CDC6600, разработчики которых (Джим Торнтон, Сеймур Крэй и др.) осознали важность упрощения набора команд для построения быстрых вычислительных машин. Эту традицию упрощения архитектуры Сеймур Крэй с успехом применил при создании широко известной серии суперкомпьютеров компании Cray Research [136]. Однако окончательно понятие RISC в современном его понимании сформировалось на базе трёх исследовательских проектов компьютеров: процессора 801 компании IBM, процессора RISC университета Беркли и процессора MIPS Стенфордского университета.

Отдельно хочется выделить несколько фирм, слабо пересекающихся с рынком процессоров для ПК: ARM, Tilera, nVidia – но занимающих значительное место на рынке современных микропроцессорных технологий.

Так, на рынке процессоров для Wi-Fi-точек доступа, небольших компьютеров, тонких клиентов, бытовых устройств, электронных книг, некоторых КПК и прочего лидирует архитектура ARM (Advanced RISC Machine, Acorn RISC Machine, усовершенствованная RISC-машина) – семейство лицензируемых 32-битных и 64-битных микропроцессорных ядер разработки компании ARM Limited.

Созданная в 2004 году компания Tilera Corporation [137] – fabless-компания, разрабатывающая многоядерные процессоры общего назначения, с большим числом ядер (десятки и сотни). Её продукция используется в сетевом оборудовании, аппаратных меж-

[136] Любопытные факты: в 1975 году компания представила публике свой первый компьютер Cray-1; в ноябрьском рейтинге TOP500 (www.top500.org) 2012 года суперкомпьютер Cray Titan с ОС Linux оказался опять первым.

[137] http://www.tilera.com/.

сетевых экранах, аппаратных ускорителях, устанавливаемых в суперкомпьютерах, и в «облачных серверах».

В отличие от двух предыдущих, фирма nVidia должна быть известна большинству читателей по производимым ею видеокартам. Производительность последних выросла настолько, что позволила фирме создать специальную технологию CUDA (расширение команд для стандартного компилятора языка Си), позволяющую производить различные вычисления (помимо тех, что нужны для видеоигр) на процессорах, изначально «заточенных» для работы с графическими объектами. При этом фирмой выпускаются «графические» платы, не имеющие даже выдеовыхода, что побудило авторов поместить эту информацию в раздел о процессорах, нежели о видеокартах.

4.7.1. Отечественные разработки (процессор Эльбрус)

Рисунок 4.33. Процессор «Эльбрус»

С момента зарождения вычислительной техники в нашей стране проектирование её высокопроизводительных средств рассматривалось как одна из важнейших целей отечественной науки и технологии. В настоящее время всего в нескольких странах мира проектируют компьютеры на микропроцессорах собственной разработки – это США, Англия, Япония и Китай. С точки зрения национальных интересов очень важно, чтобы Россия была в их числе. [18]

В нашей стране выдающееся значение в становлении и развитии вычислительной техники имели работы академика Сергея Алексеевича Лебедева. В руководимом им Институте точной механики и вычислительной техники (ИТМ и ВТ) Академии наук СССР были созданы электронные вычислительные машины (ЭВМ) пятнадцати моделей – от первых, ламповых, до быстродействующих машин на интегральных схемах, проекты которых он развернул в последние годы жизни. [18]

Идея архитектурной линии «Эльбрус», заложенной С. А. Лебедевым, родилась в 1969 году в связи с необходимостью оснащения стратегических систем специального назначения высокопроизводительной вычислительной техникой. [18]

В 1979 году в ИТМ и ВТ был предъявлен государственной комиссии многопроцессорный вычислительные комплекс (МВК) «Эльбрус-1», спроектированный на базе TTL-логики. Он был введён в ряд стратегических систем. [18]

Через шесть лет успешно прошел государственные испытания МВК «Эльбрус-2», построенный уже на новой элементной базе отечественной разработки – быстродействующих интегральных схемах с эмиттерно-связанной логикой серии ИС-100. Его производительность в десятипроцессорной конфигурации составляла 125 млн операций в секунду. МВК строился по модульному принципу, с учётом обеспечения надёжности. Благодаря своему быстродействию и отказоустойчивости, он в течение многих лет использовался в центральных объектах стратегических систем страны. Уникальные для того времени характеристики комплекса, необходимые в этих применениях,

достигались внедрением, развитием и оптимальной реализацией ряда передовых идей в организации вычислительного процесса, которые не раз относились к числу значительных результатов мирового компьютеростроения 1970–80-х годов ведущими зарубежными и отечественными специалистами. [18]

Следующим этапом развития серии стал проект МВК «Эльбрус-3» реализующий концепцию широкого командного слова. Опытный образец машины изготовили в 1990 году, но её отладка не была завершена по причине прекращения финансирования проекта из-за экономических проблем того периода. [18]

Продолжение этой проектной линии связано с деятельностью закрытого акционерного общества «Московский центр Спарк-технологий», учреждённого ИТМ и ВТ АН СССР и НИИ СуперЭВМ и вскоре переименованного в ЗАО «МЦСТ». В его структуре ведущие разработчики серии «Эльбрус», сделав принципиальную ставку на использование микропроцессорных технологий, приступили к созданию двух серий микропроцессоров и вычислительных комплексов на их основе, по утвердившейся на практике традиции объединяемых понятием «семейства “Эльбрус”». Проектной основой первой серии стала открытая архитектура SPARC (Scalable Processor Architecture), специфицированная корпорацией Sun Microsystems, второй – оригинальная архитектура «Эльбрус», развивающая принципы, которые были апробированы и заложены в МВК «Эльбрус-3» (первоначально она именовалась «архитектура E2k») и предшествующих проектах. [18]

С 2006 года в разработках ЗАО «МЦСТ» непосредственно участвует коллектив открытого акционерного общества «Институт электронных управляющих машин имени И. С. Брука» (ОАО «ИНЭУМ им. И. С. Брука»). [18]

Одним из последних результатов (на момент написания 4-го издания учебника) является создание на основе технологии 28 нм российских высокопроизводительных 64-разрядных универсальных микропроцессоров «Эльбрус-8С» и «Эльбрус-8СВ». Микросхема последнего (1891ВМ12Я) – это вычислитель серверного класса с усовершенствованным набором векторных команд, содержащая 8 ядер архитектуры «Эльбрус» 5-го поколения с тактовой частотой до 1500 МГц. Наиболе массовым продуктом является «Эльбрус-4С», представляющий собой систему на кристалле, содержащую 4 вычислительных ядра, кэш-память 2-го уровня общим объёмом 8 Мегабайт, 3 контроллера памяти, 3 канала межпроцессорного обмена и канал ввода-вывода. Рабочая тактовая частота микросхемы составляет 800 МГц. Процессор произведён по технологии 65 нанометров, его среднее энергопотребление составляет 45 Ватт. [19]

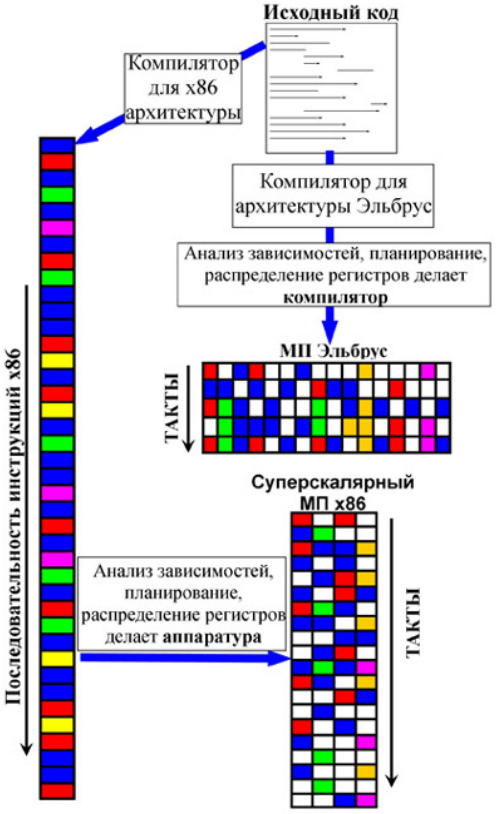
Замечание. Прямое сравнение тактовых частот процессоров VLIW-архитектур с традиционными RISC- и CISC-процессорами некорректно, как если сравнивать максимальную скорость гоночного автомобиля с аналогичной у грузовика или трактора в полевых условиях.

Базовой операционной системой для процессора является ОС «Эльбрус», построенная на основе ядра Linux версии 2.6.33. [19]

Особенности архитектуры

Архитектура «Эльбрус» – оригинальная российская разработка. Ключевые черты архитектуры «Эльбрус» – энергоэффективность и высокая производительность, достигаемые при помощи задания явного параллелизма операций. [19]

В традиционных архитектурах типа RISC или CISC (x86, PowerPC, SPARC, MIPS, ARM), на вход процессора поступает поток инструкций, которые рассчитаны на последовательное исполнение. Процессор может детектировать независимые операции и запускать их параллельно (суперскалярность) и даже менять их порядок (внеочередное исполнение). Однако динамический анализ зависимостей и поддержка внеочередного исполнения имеет свои ограничения: лучшие современные процессоры способны анализировать и запускать до 4-х команд за такт. Кроме того, соответствующие блоки внутри процессора потребляют заметное количество энергии. [19]



В архитектуре «Эльбрус» основную работу по анализу зависимостей и оптимизации порядка операций берёт на себя компилятор. Процессору на вход поступают так называемые «широкие команды», в каждой из которых закодированы инструкции для всех исполнительных устройств процессора, которые должны быть запущены на данном такте. От процессора не требуется анализировать зависимости между операндами или переставлять операции между широкими командами: всё это делает компилятор, исходя из анализа исходного кода и планирования ресурсов процессора. В результате аппаратура процессора может быть проще и экономичнее. [19]

Компилятор способен анализировать исходный код гораздо тщательнее, чем аппаратура RISC/CISC процессора, и находить больше независимых операций. Поэтому в архитектуре Эльбрус больше параллельно работающих исполнительных устройств, чем в традиционных архитектурах, и на многих алгоритмах она демонстрирует непревзойдённую архитектурную скорость. [19]

4.7.2. Процессор изнутри

Читатель, наверное, догадывается, что внутреннее устройство современных процессоров довольно сложное и выходит за рамки нашей книги, но в «двух словах» можно сказать, что процессоры, как и другие микросхемы, состоят из ключевых элементов логики, рассмотренных выше, а они, в свою очередь, выполнены интегральным способом из более простых элементов (транзисторов и прочего). Полагаем, что увидеть внутреннее устройство процессора будет интересно даже профессионалам. Так, шведский

учитель Кристиан Сторм (Kristian Storm) для наглядной демонстрации своим студентам устройства процессора взял и распилил последний, а фотографии по мере увеличения масштаба (разрешения) отображения образца разместил в своём блоге http://www.sciencystuff.com/?p=24 (на английском языке [138]). Ниже приведена одна из его фотографий, сделанная на электронном микроскопе с максимальным увеличением (см. рис. 4.34).

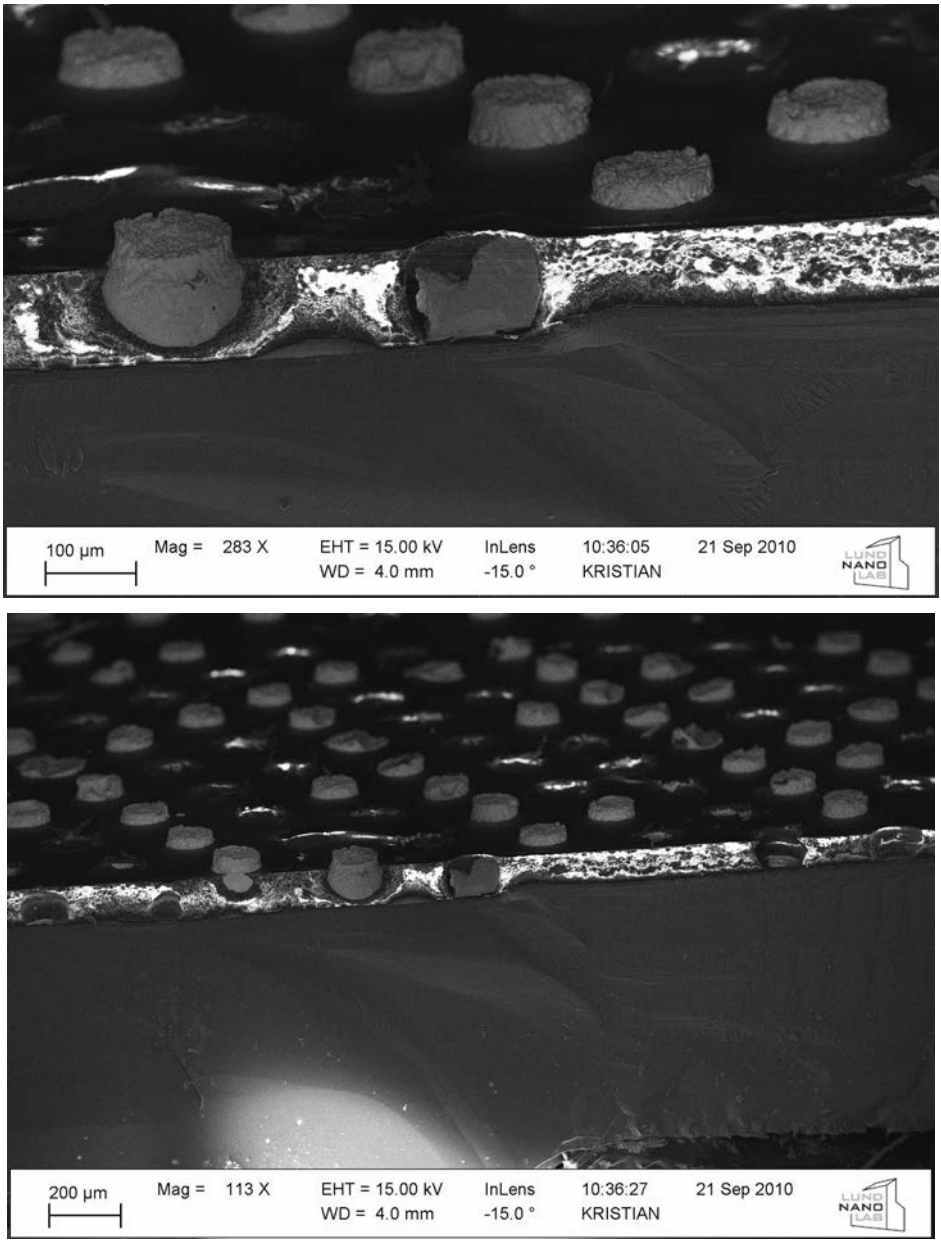


Рисунок 4.34. Срез процессора под электронным микроскопом

[138] Перевод на русский с комментариями http://habrahabr.ru/post/127205/.

4.7.3. Закон Мура

Говоря о процессорах, невозможно не упомянуть закон Му́ра [139]. 19 апреля 1965 года Го́рдон Эрл Му́р вывел и опубликовал своё наблюдение, так называемый «закон Мура», согласно которому количество транзисторов в кристалле микропроцессора удваивается каждый год. В 1975 году он изменил временную составляющую закона, и в современной формулировке закон звучит как «*количество транзисторов, размещаемых на кристалле интегральной схемы, удваивается каждые 24 месяца*».

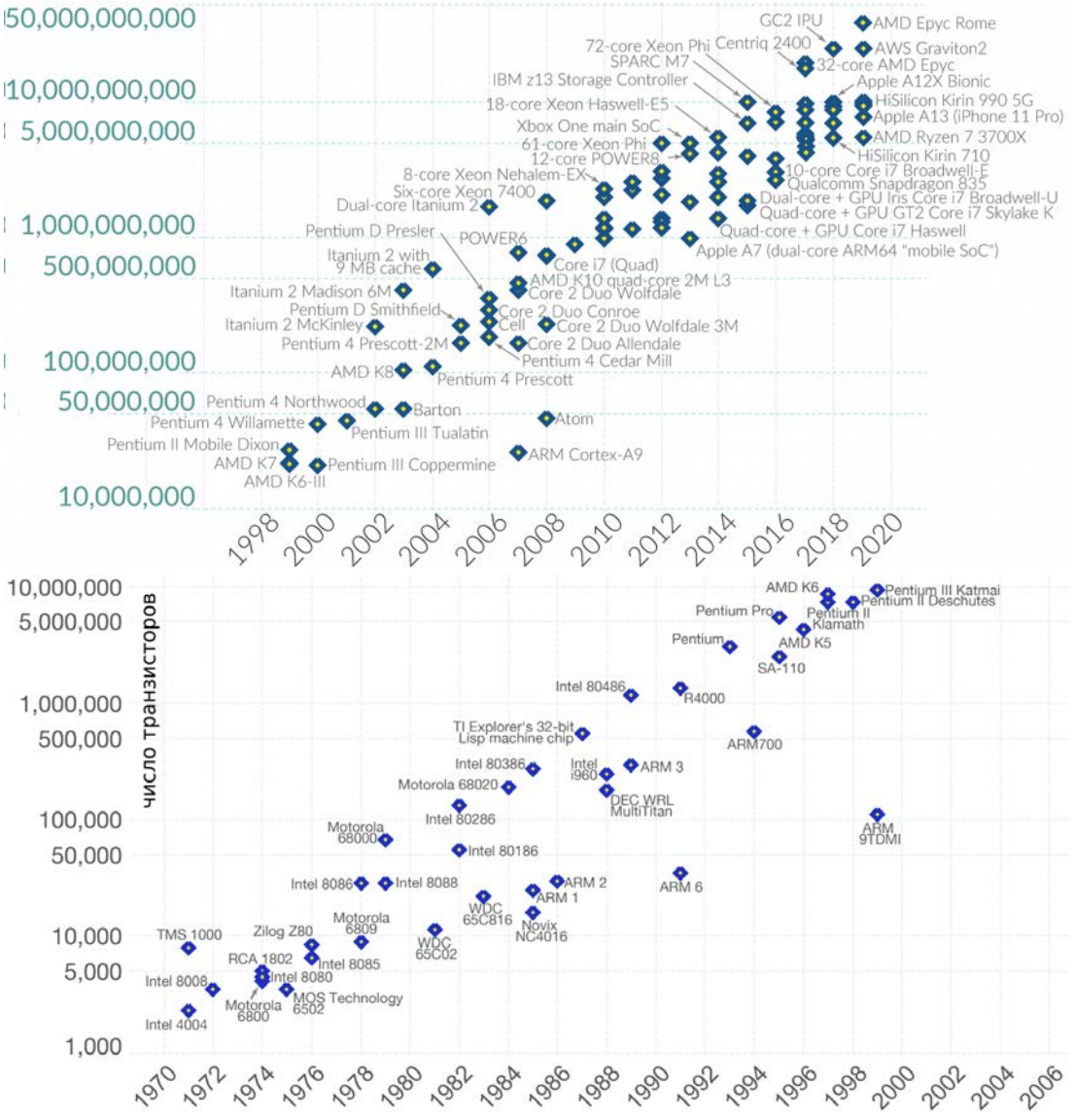


Рисунок 4.35. Наглядное отображение закона Го́рдона Эрла Мура [20] (диаграмма настолько большая, что её пришлось поделить на 2 части)

[139] Го́рдон Эрл Му́р (1929–н/вр.) – химик, почётный председатель совета директоров и основатель корпорации Intel.

Изначально выведенный как эмпирическое правило, закон Мура со временем стал основополагающим принципом полупроводниковой отрасли, определяющим создание всё более мощных полупроводниковых микросхем со всё более низкой себестоимостью. Наиболее наглядно его можно представить в виде графика [140], где по оси абсцисс откладывается год, а по оси ординат – число транзисторов на микросхему.

4.7.4. Что выбрать

К сожалению, случается так, что многие продавцы и сборщики ПК не имеют ни малейшего представления об информации, изложенной выше. Поэтому от теории перейдём к более необходимой практике по правильному выбору процессора для ПК. Чтобы покупатели не запутались в используемых технологиях (к сожалению, в рыночной экономике не нужно разбираться в товарах, главное – их покупать), фирмы Intel и AMD регулярно «делят рынок потребителей между собой», заманивая покупателей различными сравнительными характеристиками, выпускают различные рекламные проспекты, публикуют на своих сайтах наглядные схемы о том, как работают их процессоры. Попробуем во всём этом разобраться и мы вместе с читателями.

От процессора в значительной степени зависит скорость работы **ПК** (лимитирующим фактором также может быть объём оперативной памяти и другие характеристики). Процессор имеет сложную архитектуру, свою высокоскоростную буферную память (кэш), использует специальные технологии обработки информации.

Простейшая принципиальная схема микропроцессора Intel приведена на рис. 4.36 (по данным фирмы Intel [141]). Принцип работы центрального процессора можно представить следующим образом. Информация для обработки под управлением блока предварительной выборки поступает из системной памяти через блок шины в кэш данных процессора, команды обработки информации – в командную кэш-память. Блок декодирования команд преобразует их в двоичный код, который пересылается в управляющий блок и в кэш данных, давая им указание о том, как с полученной командой поступать дальше. Арифметическое логическое устройство выполняет готовые к исполнению команды и заносит результаты в блок регистров. Далее содержимое регистров передаётся в системную память или на внешние устройства. Более подробное описание выполнения процессором простой операции сложения «2 + 3» можно найти по адресу *http://www.intel.com/plt/cd/corporate/emea/rus/museum/mpuworks.swf*, там же, на сайте, в разделе глоссария, можно найти краткое описание функций всех устройств, показанных на рис. 4.36.

Если бы на земле существовал всего лишь один тип процессора, то пытливый ум читателя предложил бы его как-нибудь улучшить, и тем самым стало бы уже два типа процессоров (вариантов реализации). После этого вы, читатель, невольно постарались бы их сравнить... Вот так и сформировался сегодняшний рынок процессоров, который переполнен разными моделями, нуждающимися в сравнении.

Попробуем ответить на вопрос «Как сравнивать?» или «Какой процессор быстрее?». Ввести однозначный показатель скорости работы довольно сложно. Это как сравнивать поезд и велосипед для целей поездки в соседнюю булочную. Скорость ра-

[140] http://en.wikipedia.org/wiki/File:Transistor_Count_and_Moore's_Law_-_2011.svg, https://upload.wikimedia.org/wikipedia/en/9/9d/Moore%27s_Law_Transistor_Count_1971-2016.png
[141] http://www.intel.com/plt/cd/corporate/emea/rus/museum/mpuworks.swf.

боты процессора – это комплексная характеристика, зависящая в настоящее время прежде всего от типа и архитектуры процессора, а также от его тактовой частоты и объёма кэш-памяти. Поэтому правильнее говорить и оценивать комплексный параметр «производительность», так как это более практически применимая характеристика.

Процессоры Intel Pentium и Core используют технологию конвейерной обработки данных, в результате чего за один такт выполняется несколько машинных операций. Многоядерные процессоры позволяют увеличить производительность **ПК** за счёт одновременного выполнения нескольких программ пользователя на разных ядрах или выполнения одной программы на нескольких ядрах, если она предусматривает параллельную многопроцессорную обработку данных. Скорость обработки информации процессором может также лимитироваться скоростью поступления этой информации из оперативной памяти.

Процессоры выпускаются различными фирмами, для различных типов компьютеров и для другой электроники. Так, фирма Intel (www.intel.com) выпускает процессоры не только для настольных ПК, но и для ноутбуков, серверов, коммуникаторов и другого оборудования. Другой наиболее известный производитель того же сегмента – фирма AMD (www.amd.com).

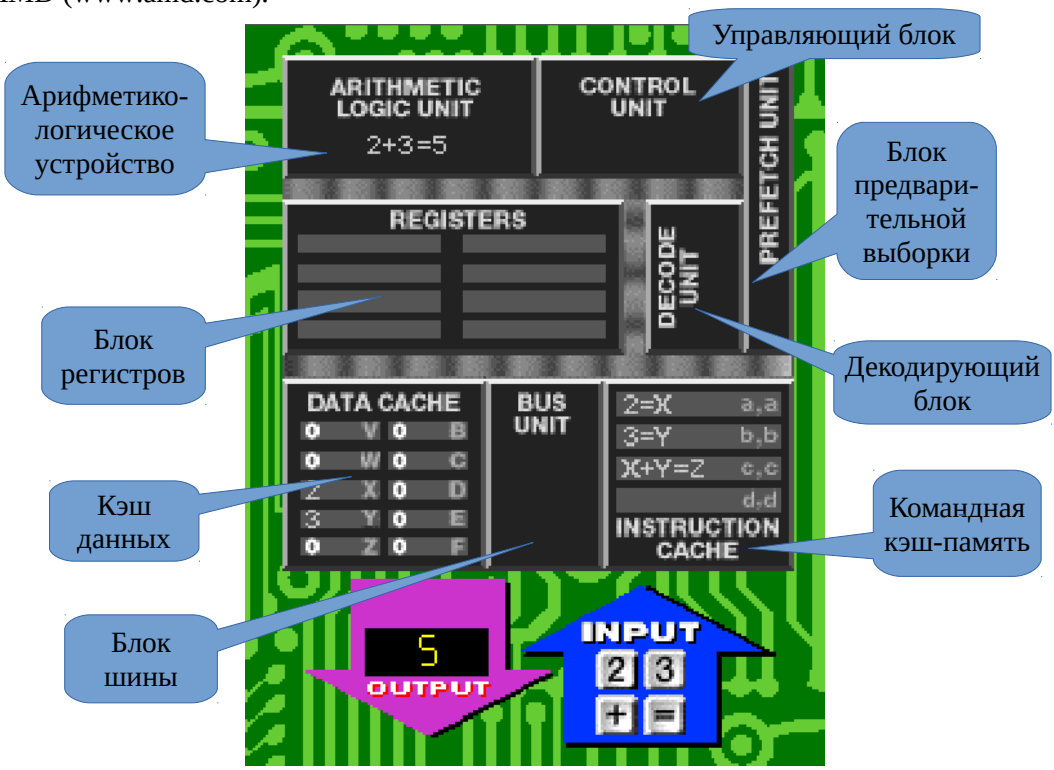


Рисунок 4.36. Принципиальная схема процессора (данные фирмы Intel)

В 1990 году прогнозировалось – тактовая частота процессора Intel возрастёт к 2000 году до 900 МГц, количество транзисторов в нём – до 40 млн штук, к 2005 году – 10 ГГц, 1 млрд транзисторов. Однако в 2005 г. стало ясно, что прогресс пошёл другим путём – развитием многоядерных процессоров без увеличения их рабочей частоты и ко-

личества транзисторов на чипе (но с увеличением частоты системной шины FSB до 1333 МГц и с поддержкой 64-разрядных вычислений и памяти).

Графики реального увеличения тактовой частоты процессора [142] и количества транзисторов в нём с момента их появления по сегодняшний день приведены на рис. 4.37.

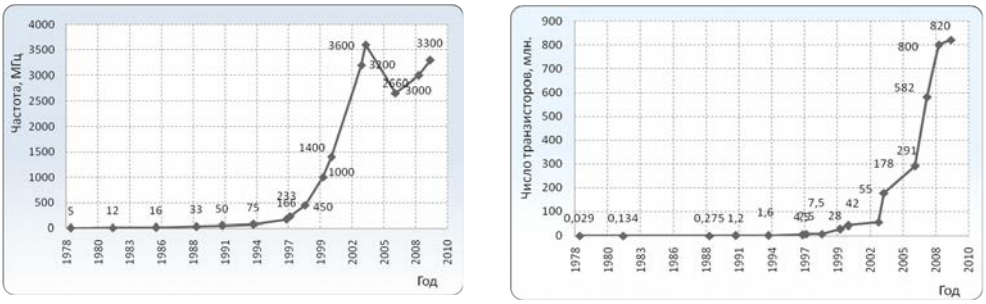


Рисунок 4.37. График изменения тактовой частоты и количества транзисторов в процессорах фирмы Intel

К сожалению, тенденции последних дней таковы, что информация о числе транзисторов в чипах не выкладывается в общий доступ производителями, потому как она мало информативна для покупателя, также не ясны подходы по оценке числа транзисторов, а именно стоит ли учитывать кэш-память, интегрированную в чип, размеры которой могут варьироваться? Что до тактовой частоты – информация по ней присутствует, но в разных моделях она варьируется, поэтому для сохранения наглядности мы не стали наносить на график информацию после 2010 года, потому как она не отображает никаких закономерностей. Мы предлагаем читателям больше ориентироваться на закон Мура, описанный ранее.

Некоторое снижение тактовой частоты и уменьшение числа транзисторов в 2005–2006 годах было вызвано появлением двухъядерных процессоров Core Duo и Core 2 Duo, а позднее 4-ядерного Core Quad, суммарная производительность двух или четырёх ядер которых выше, чем одноядерных процессоров с более высокой тактовой частотой.

Историю развития ПК можно проследить по основным этапам развития процессоров фирмы Intel [143]. Краткая справка по истории развития процессоров для настольных ПК приведена в табл. 4.5.

Таблица 4.5. История развития процессоров Intel (для настольных ПК)

Начало выпуска, год	Процессор	Тактовая частота	Технология; число транзисторов
2020	10-е поколение i9-10900K (10 ядер, 20 потоков)	3,7–5,1 ГГц	14-нм
2017-2019	7-9-е поколение i9-7980XE (18 ядер, 36 потоков)	2,6–4,2 ГГц	14-нм
2015-2016	5-6-е поколение Intel® Core™ i7-5650U-6700 (2-4 ядер, 4-8 потоков)	2,2–4,2 ГГц	14-нм

[142] См. http://en.wikipedia.org/wiki/Microprocessor_chronology.

[143] Полную таблицу выпускаемых фирмой Intel процессоров для настольных ПК, ноутбуков и серверов можно посмотреть на странице http://ark.intel.com/ru.

2013-2014	4-е поколение Intel® Core™ i7-4765–4790 (4 ядра, 8 потоков, HD Graphics 4600)	3,5–4,4 ГГц Turbo Boost	22-нм
2012	Intel® Core™ i7-3770 (4 ядра, 8 потоков, HT, HD Graphics 4000)	3,4–3,9 ГГц Turbo Boost	22-нм
2012	Intel® Core™ i5-3330, i5-3550 (4 ядра, 4 потока, HD Graphics 2500)	3,3–3,7 ГГц Turbo Boost	22-нм
2012	Intel® Core™ i3-3240, i3-3220 (2 ядра, 4 потока, HT, HD Graphics 2500)	3,2 ГГц	22-нм
2011	Intel® Core™ i7-2600 (4 ядра, 8 потоков, HT, HD Graphics 2000)	3,4–3,8 ГГц Turbo Boost	32-нм
2011	Intel® Core™ i5-2300, i5-2400, i5-2500 (4 ядра, 4 потока, HD Graphics 2000)	3,1–3,4 ГГц Turbo Boost	32-нм
2011	Intel® Core™ i3-2100, i3-2130 (2 ядра, 4 потока, HT, HD Graphics 2000)	3,4 ГГц	32-нм
2010	Intel® Core™ i7-970 (6 ядер, 12 потоков, HT)	3,2–3,46 ГГц Turbo Boost	32-нм
2010	Intel® Core™ i3-560, i3-530 (2 ядра, 4 потока, HT)	3,33; 2,93 ГГц	32-нм
2010	Intel® Core™ i5-680, ™ i5-670, i5-661, i5-660, i5-650 (2 ядра, 4 потока, HT)	от 3,46 до 3,73 ГГц Turbo Boost	32-нм
2010	Pentium® G6950 (2 ядра)	2,8 ГГц	32-нм
2009	Intel® Core™ i5-750 (4 ядра)	2,66 ГГц	45-нм
2008	Intel® Core™ i7-960 – i7-860 (4 ядра, 8 потоков, технология HT)	3,2–3,46 ГГц Turbo Boost	45-нм 731 млн
2008	Intel® Atom™ (для субноутбуков)	2,0–0,8 ГГц	45-нм; 47 млн
2007	Intel® Pentium® Dual-Core	2,8–1,60 ГГц	45 или 65-нм
2007	Intel® Core™ 2 Quad (4 ядра) Q9650 … Q6600	3,00–2,40 ГГц	45 или 65-нм 820 млн
2007	Intel® Core™ 2 Extreme QX9775, QX6700 (4 ядра)	3.20–2.66 ГГц	45 или 65-нм 410 млн
2006	Intel®Core™2 Duo (2 ядра) E6700, E6750 … E4300	2,66–1.80 ГГц	65-нм; 291 млн, 167 млн
2006	Core™ Duo (2 ядра)	1,66–2,33 ГГц	65-нм; 152 млн
2005	Intel® Pentium® Extreme Edition № 840 (2 ядра, технология HT)	3,20 ГГц	90-нм
2005	Intel® Pentium® D (2 ядра) №№ 960… 805	3,60–2,66 Гц	90-нм; 376 млн
2004	Intel® Celeron® D № 351, 346, … 320, 315	3,20–2,53ГГц	90-нм; 188 млн
2004	Intel® Pentium® IV Extreme Edition	3,73, 3,40 ГГц	90 нм, 0,13 мкм 169 млн
2003	Intel® Pentium® IV с технологией Intel® HT	3,60–2,40 ГГц	0,09 мкм, 0,13 мкм; 178 млн
2000	Intel® Pentium® IV	1,40 ГГц	0,18 мкм; 42 млн
1999	Intel® Pentium® III	1 ГГц–450 МГц	0,18 мкм; 28 млн 0,25 мкм; 9.5 млн
1998	Intel® Celeron®	2,80 ГГц, 266 МГц	0,13 мкм; 0,25 мкм; 7,5 млн
1997	Intel® Pentium® II	450–350 МГц 300–233 МГц	0,25 мкм; 0,35 мкм; 7,5 млн

1995	Intel® Pentium® Pro	200–166 МГц, 50 МГц	0,60–0,35 мкм; 5,5 млн
1997	Intel® Pentium® с технологией MMX™	233–166 МГц	0,35 мкм; 4,5 млн
1993	Intel® Pentium®	66–60 МГц	0,8 мкм; 3,1 млн
1992	Intel486™ SL	33–20 МГц	0,8 мкм;1,4 млн
1992–95	Intel486™ SX2/DX2-DX4	120,100,83,75,66,50 МГц	0,6 мкм; 1,6 млн
1989	Intel486™ DX	50, 33, 25 МГц	1–0,8 мкм; 1,2 млн
1992	Intel386™ SX	33, 25, 20, 16 МГц	1,5–1 мкм; 275 тыс.
1990	Intel386™ SL	20, 25 МГц	1,5–1 мкм; 855 тыс.
1985	Intel386™ DX	33, 25, 20, 16 МГц	1,5–1 мкм; 275 тыс.
1982	80286	12, 10, 6 МГц	1,5 мкм; 134 тыс.
1978	8086	10, 8, 5 МГц	3 мкм; 29 тыс.

В конце 2008 г. Intel представила четырёхъядерные восьмипоточные процессоры **Core i7** на микроархитектуре Nehalem со встроенным контроллером памяти (выпускается в корпусе LGA1366). Пропускная способность шины памяти компьютеров с процессорами Intel **Core i7** выросла более чем в два раза, по сравнению с процессорами Intel серии Extreme, благодаря новому интерфейсу **Intel QuickPath Interconnect** (Intel® QPI), заменившему привычную системную шину **FSB** (Front-Side Bus, см. далее в разделе 4.4 *«Чипсет»*).

Совершённый в 2010 году переход на 32-нм технологию для процессоров **Core i7**, **Core i5, Core i3** и **Pentium G6950** архитектуры Clarkdale (с интегрированными на одном кристалле контроллерами памяти и видеоподсистемой) продолжился в 2012 году переходом на 22-нм технологию, а в 2015 году на 14-нм технологию.

Блок-схема процессоров семейства Clarkdale (Core i5, i3, Pentium G6950) показана на рис. 4.38.

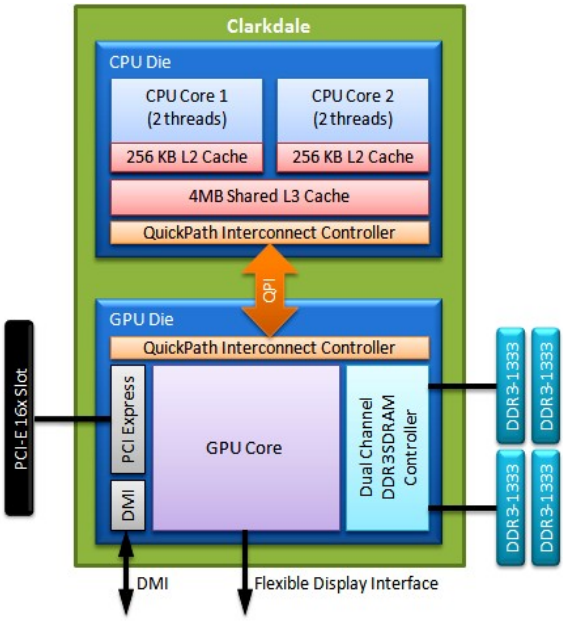


Рисунок 4.38. Блок-схема процессоров семейства Clarkdale

В двухъядерных и четырёхъядерных процессорах Intel используются различные технологии Intel, обеспечивающие повышение эффективности их использования, например для Core i5:

- Технология **Intel® Turbo Boost** – максимально повышает производительность ресурсоём-

ких приложений, динамически увеличивая производительность в соответствии с нагрузкой.

- Технология **Intel® Virtualization** (Intel® VT) – управление эффективной и масштабируемой виртуальной инфраструктурой с сохранением оптимальной производительности при низких затратах.
- Технология **Intel® Enhanced Speedstep** – управления энергопотреблением процессора в зависимости от нагрузки.
- Функция **Execute Disable Bit** позволяет процессору выделять области памяти, где допускается выполнение кода приложений и где оно не допускается. Когда вредоносная программа-червь пытается установить свой код в буфер памяти, процессор отключает выполнение кода.
- Архитектура **Intel 64** – обеспечивает поддержку 64-разрядных вычислений, предоставляет процессору доступ к большему объёму памяти.
- Технология **Intel® Hyper-Threading** – обеспечивает обработку 2 потоков команд каждым физическим ядром, повышая общую пропускную способность процессора до 4 потоков. Она позволяет каждому ядру процессора одновременно выполнять две задачи.
- Технология **Intel® Smart Cache** – использование общей кэш-памяти процессора 2-го уровня, которая динамически распределяется между ядрами в зависимости от нагрузки.
- **Интегрированный контроллер памяти** поддерживает 2 канала памяти DDR3 1333 МГц.
- Графическая система **Intel® Graphics Media Accelerator HD**, расположенная в одном корпусе с процессором, обеспечивает высокое качество воспроизведения видео HD и графики в трёхмерных играх без необходимости использования дополнительных видеокарт или декодеров. Графическое ядро HD Graphics оснащено 14 исполнительными блоками (14 EU, Execution Units) с потоковыми процессорами Intel 3-го поколения.

Фирма AMD в своих процессорах имеет не уступающий список новшеств. По сути, данные «красивые» названия носят больше рекламный характер для воздействия на умы покупателей и разделения их по критерию «толщины кошелька», так как большинство из них грамотно ответить на вопрос, как проверить, используется та или иная технология, не смогут, разве что в BIOS'e посмотреть скорость вращения вентилятора процессора. Скорее, эта информация полезна для программистов «аппаратного уровня» и разработчиков ОС. Если вы нам не верите, попробуйте доказать вашим коллегам, что у вас на компьютере используется технология **Execute Disable Bit** на практике и по сути, а не формально по вкладке «свойства процессора» какой-либо диагностической программы.

Блок-схема процессоров семейства Clarkdale (Core i5, i3, Pentium G6950) показана на рис. 4.38.

Начиная с **Core**, в модельном ряду **Intel** произошло изменение способа нумерации процессоров. Если раньше использовались трёхцифровые обозначения, то теперь модельный номер представляет собой комбинацию из пяти символов – одной буквы и четырёх цифр. Например: **Core 2 Duo E6700, Core 2 Quad Q6600** (у экстремальных процессоров добавляется ещё одна буква X: Core2 Extreme quad-core QX6800).

Первая буква символизирует энергопотребление/тепловыделение и, таким образом, тип процессора (например, **E –** более 50 Вт, процессор для настольных ПК). Четыре цифры после буквы показывают производительность в пределах одной линейки.

У процессоров **Core i7, Core i5** и **Core i3** первого поколения использовалось трёхцифровое обозначение, у 2-го и 3-го поколений добавились первые цифры 2 и 3 соответственно.

В рамках каждого класса или семейства процессоров больший номер соответствует большему количеству различных функций, в том числе: объём кэш-памяти, тактовая частота, частота системной шины, поддержка новых технологий, новых команд и прочего.

Рейтинг процессоров Intel можно без труда найти в интернете. Разброс цен между самой дешёвой и самой дорогой продаваемыми моделями процессоров (несмотря на инфляцию, скачки курсов валют и другие факторы) примерно одинаков год от года, производительность же постоянно увеличивается. В табл. 4.6 показаны некоторые процессоры Intel и их цены.

Таблица 4.6. Некоторые процессоры Intel и их цены

Модель (ядер/потоков), 14 нм	Тактовая частота, ГГц	Кэш L3, Мб	Цена, руб. (февраль 2018 г.)
Celeron G3900 (2/2)	2,8	2	2 370
Pentium G4600 (2/4)	3,6	3	4 470
Core i3-7100 (2/4)	3,9	3	6 910
Core i5-8400 (6/6)	2,8–4,0	9	12 720
Core i5-7600 (4/4)	3,5–4,1	6	15 280
Core i7-7700 (4/8)	3,6–4,2	8	19 930
Core i7-8700K (6/12)	3,7–4,7	12	24 780
Core i9-7940X (14/28)	3,1–4,4	19,25	99 880

Фирма **AMD** также выпускает многоядерные процессоры различных серий для настольных ПК – **FX**, **Ryzen**, **A-Series** (см. табл. 4.8) и более старые модели **Athlon™ II** и **Phenom™ II**.

Особенность процессоров AMD – контроллер памяти интегрирован в процессор. Каждое ядро имеет свой кэш второго уровня и работает со своей оперативной памятью, но при необходимости может обращаться и к памяти другого ядра (технология HyperTransport™). Некоторые процессоры имеют встроенный видеочип, например AMD A6-5400K Black Edition содержит Radeon HD 7540D. Удобную таблицу сравнения процессоров фирмы AMD по различным параметрам, в том числе и по потребляемой мощности, можно найти по адресу http://products.amd.com/en-us/comparison/DesktopCPU.aspx.

Таблица 4.7. Некоторые модели процессоров AMD и их цены

Модель (ядер/потоков)		Тактовая частота, ГГц	Кэш L2, всего, Мб	Цена, руб. (февраль 2018 г.)
AMD A4-5300 (2/2)	32 нм	3,4-3,6	1	1 570
AMD A6-7400K (2/2)	28 нм	3,5-3,9	1	2 730
AMD A10-7800 (4/4)	28 нм	3,5–3,9	4	6 810
AMD FX 4300 (4/4)	32 нм	3,8–4,0	4 + 4 (L3)	3 280
AMD FX 6300 (6/6)	32 нм	3,5–4,1	6 + 8 (L3)	4 680
AMD FX 8320 (8/8)	32 нм	3,5–4,0	8 + 8 (L3)	6 940
AMD Ryzen 3 1300X (4/4)	14 нм	3,5–3,7	2 + 8 (L3)	8 240
AMD Ryzen 5 1500X (4/8)	14 нм	3,5–3,7	2 + 16 (L3)	11 630
AMD Ryzen 5 1700X (8/16)	14 нм	3,4–3,8	4 + 16 (L3)	24 450

Как видно из таблиц, AMD не хочет отклоняться от взятого курса на предложение платформ по более выгодной, чем у конкурента, цене, поэтому старшие восьмиядерные модели с ядром Zambezi противопоставляются старшим же процессорам Intel Core i5. В целом AMD придерживается примерно следующей схемы позиционирования своих продуктов (см. рисунок 4.39).

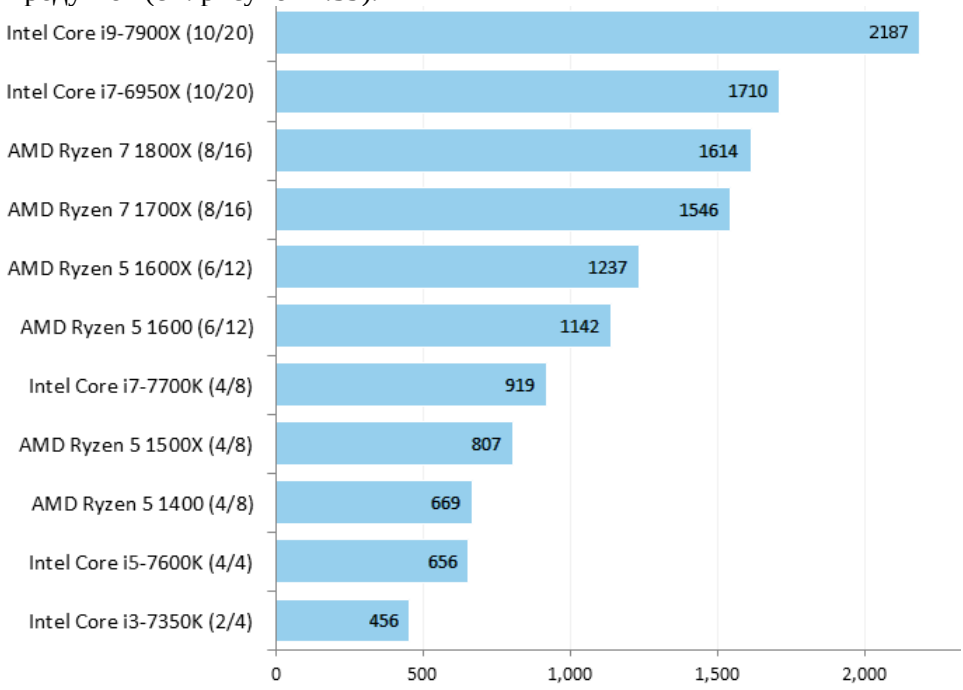


Рисунок 4.39. Соотнесение процессоров Intel и AMD по производительности

Иными словами, AMD вообще не намерена замахиваться на соперничество с восьмиядерными процессорами фирмы Intel, а хочет сосредоточиться на покорении среднего ценового сегмента.

Приятной новостью для энтузиастов является и тот факт, что во всех процессорах серии Ryzen не заблокированы и не будут блокироваться никакие множители. Их, как и всегда, можно легко разгонять простой сменой базового множителя. Также доступен разгон подсистемы памяти и частоты встроенного в процессор северного моста.

4.7.5. «Разгон» процессоров

Так сложилось, что мы живём в «аналоговом мире» и сталкиваемся с погрешностями... Проведите небольшой эксперимент, возьмите литр молока [144] и попробуйте быстро и на глаз разлить его на пять стаканов поровну. По завершении эксперимента, с большой вероятностью, где-то будет больше, где-то меньше. Или попробуйте 10 раз подряд выполнить какое-то непривычное для вас действие, неизбежно вы заметите, что делаете его с погрешностью. Естественно, имея регулярный опыт совершения чего-либо, со временем вы сможете довести процесс до автоматизма и погрешность (разброс) будет меньше. Аналогичная ситуация складывается и в производстве процессоров. Сначала они идут с большим разбросом, потом разброс уменьшается, а «относительное» внутреннее качество процессоров растёт. После производитель производит тестирование произведённых образцов, определяя «степень отклонения от желаемого образца», например по количеству совершённых ошибок на большом числе операций, и наносит различающие маркировки на корпус и во внутреннюю память процессора. Образцы с отсутствием ошибок получают высший рейтинг, те же, что содержат ошибки, переводятся на режим тестирования с меньшей тактовой частотой, и процесс повторяется. Например, если процессор с 4 ядрами дал сбой на каком-то из ядер, «сбойное» ядро можно программно отключить, а процессор маркировать как трёх- или двухъядерный и продать за меньшую сумму. К концу производственного цикла любой серии процессоров почти всегда оказывается, что спрос на процессоры с меньшими характеристиками есть, а сбойных образцов практически или вообще нет. В этом случае процессоры маркируют «искусственно» из маркетинговых соображений, а не из их потенциальных технических возможностей[145]. Данную ситуацию понимают не только пользователи, но и производители материнских плат, предлагающие различные опции для «разгона» искусственно «заниженных» процессоров. После чего в игру вступают энтузиасты со знаниями (может, это будете и вы, наш читатель), которые готовы немного рискнуть (а чем больше знания, тем меньше риск), потратить время и за меньшие деньги стать обладателем высокопроизводительного компьютера.

Мы постарались коротко и доступно описать технологию «разгона», более подробную информацию теперь вы можете найти самостоятельно на сайтах, специализирующихся на тематике разгона [146], где можно найти не только инструкции, как и что можно сделать, но и рекомендации по покупке той или иной серии процессоров и результаты экспериментов других пользователей. Рекомендуем также посетить страницу утилиты Hot CPU Tester Pro http://www.7byte.com/index.php?page=hotcpu.

В ОС Linux повысить производительность можно штатными средствами, например, следующая команда скажет процессору работать на максимальной производительности при электропитании от сети.

[144] Кефир или литр воды из-под крана, не важно, просто молочные продукты в силу своего белого окраса дают более наглядную картину сравнения.

[145] С помощью программы PassMark Burn-In Test (http://www.passmark.com) вы можете самостоятельно проверить свой процессор на «разгоняемость».

[146] Например http://www.overclockers.ru.

```
# cpuspeed -C
```

При «разгонах» может быть полезным диагностировать систему с помощью сенсоров

```
$ sensors
... пропущено...
Adapter: PCI adapter
temp1:    +24.0°C  (high = +70.0°C)
                   (crit = +71.0°C, hyst = +66.0°C)
in0:      +1.38 V  (min =  +0.00 V, max =  +4.08 V)
in1:      +1.52 V  (min =  +0.00 V, max =  +4.08 V)
fan2:    1341 RPM  (min =   10 RPM)
fan3:       0 RPM  (min =    0 RPM)
temp1:    +42.0°C  (low  = +127.0°C, high = +127.0°C)  sensor = thermistor
temp2:    +32.0°C  (low  = +127.0°C, high = +80.0°C)  sensor = thermal diode
cpu0_vid: +0.575 V
```

«Нагрузить» процессор можно командой stress, например

```
$ stress -cpu 16
```

Замечание. Указанные программы находятся в пакетах cpuspeed, lm_sensors и stress. Для получения информации от сенсоров их следует один раз предварительно настроить, запустив от администратора команду `sensors-detect`. Запустив программы одновременно, можно увидеть как при создании нагрузки растут: частоты, температура, обороты вентилятора. Частоту ядер процессора удобно отслеживать с помощью команды

```
$ watch "cat /proc/cpuinfo|grep MHz"
```

4.7.6. Резюме по процессорам

Сравнению процессоров AMD и Intel посвящено большое количество статей на сайтах http://*www.thg.ru*, http://*www.ixbt.com*, а также полезную информацию можно найти на сайтах http://www.infa.ru/process/, http://www.cpubenchmark.net и др.

О том, как производятся процессоры, читайте в [9].

В качестве резюме приведём ***краткий справочник терминов***.

Тактовая частота – это то количество элементарных единичных (однотактовых) операций, которые процессор может выполнить за единицу времени. В качестве мерки времени обычно выбирают одну секунду, при этом многие команды процессора могут выполняться за несколько тактов. У RISC-процессоров используются более короткие операции. У CISC-процессоров, наоборот, одна команда для выполнения может использовать десятки тактов.

Внутренняя тактовая частота процессоров обычно выше и определяется тактовой частотой шины подключения процессора на материнской плате умноженной на множитель (варьирующийся в районе 10–20).

Число тактов очень велико, увидеть единичный такт можно с помощью высокочастотного осциллографа, но «прочувствовать» его вряд ли получится.

Несмотря на то что единичные транзисторы могут работать на частотах порядка десятка ГГц, сегодняшний потолок для процессоров едва дотягивает до 5 ГГц. Впрочем, и эту мощность мы толком-то и освоить не можем...

С одной стороны, больше тактов – лучше производительность, с другой – на производительность большее влияние оказывает не тактовая частота, а число выполненных операций, число которых можно значительно повысить за счёт большего числа

ядер (распараллеливания). За одинаковое время бригада грузчиков явно перенесёт больше ящиков по весу, чем один спортсмен-штангист мирового класса.

Количество ядер. Когда стало ясно, что повышать тактовую частоту экономически не выгодно, фирмы перешли на выпуск многоядерных процессоров (самыми распространёнными сегодня являются четырёх- и шестиядерные). Реклама уверяет нас, что чем больше ядер в процессоре, тем он эффективнее... с одной стороны, это так, но с другой – попробуйте поручить 10 малярам одновременно выкрасить потолок маленькой ванной комнаты. Вряд ли они смогут это сделать быстрее, чем один или два маляра. Для эффективной загрузки многоядерных процессоров нужны большие объёмы выполняемых заданий и грамотное распределение задач между ядрами процессора.

Программ, умеющих совладать с большим числом ядер, – единицы, хотя их число постоянно растёт. Далеко не все программы умеют это делать, да и не все алгоритмы поддерживают распараллеливание.

Наиболее эффективным будет ПО, заранее разработанное и скомпилированное с поддержкой многоядерности, предпочтительно к таким программам, поддерживающим распараллеливание, можно отнести видеомонтажное ПО, архиваторы, программы обработки звука, компиляторы, системы управления базами данных (СУБД).

Для многих игр количество ядер в процессоре вообще не важно: основная нагрузка ложится на видеоплату.

Режим работы процессора. Для осуществления обратной совместимости все 32-разрядные и более поздние процессоры фирм Intel и AMD (начиная с 386-го) могут выполнять программы в нескольких режимах.

Процессоры могут работать в трёх режимах: реальном, защищённом и виртуальном реальном режиме (реальном внутри защищённого). Многие 64-битные процессоры дополнительно аппаратно поддерживают инструкции, упрощающие виртуализацию.

В разных режимах возможности процессора не одинаковы, потому что команды выполняются по-разному. Логично предположить, что в режиме эмуляции 32-битного процессора не будут выполняться 64-битные инструкции. В зависимости от режима процессора изменяется схема управления памятью системы и задачами.

Конечному пользователю, если только он не продвинутый «аппаратный» программист, не стоит расстраиваться и задумываться о режимах работы, так как за их выбор в большей степени отвечает операционная система, а многие пользователи даже и не догадываются об их существовании.

Кэш-память. Ещё один важный параметр, прямо связанный со скоростью работы процессора, – объём «кэша», быстрой буферной памяти, встроенной в кристалл процессора, – она работает значительно быстрее оперативной памяти, находящейся на материнской плате. Номинально чем больше быстродействующей кэш-памяти – тем лучше и тем дороже процессор. С целью не делать процессоры слишком дорогими «на борту» современных процессоров используется многоуровневое кэширование (L1, L2 и L3, «L» – от англ. level – уровень; соответственно, кэш 1-го, 2-го и 3-го уровней). Чем выше «уровень» – тем память медленнее, однако и объём её больше за счёт того, что она обходится дешевле в производстве. У дешёвых Atom, к примеру, кэш L2 составляет 512 КБ, а более мощные процессоры в дополнение к этому получают ещё и кэш L3 объёмом от 3 МБ (Core i3) до 8-20 МБ (Core i7, AMD FX). Больший кэш даёт некоторый выигрыш в производительности (до 15–20%) при работе с некоторыми про-

граммами – например, архиваторами, СУБД, математическим софтом – ну и отдельными трёхмерными играми.

При работе с воспроизведением видео, звуком (включая сжатие), графикой и в 9 из 10 игр выигрыш от увеличения кэша либо отсутствует вовсе, либо не слишком ощутим. Того минимального объёма, заложенного в самую младшую модель процессора, оказывается более чем достаточно.

Форм-фактор. Часто смена типа ядра и архитектуры процессора влечёт за собой изменения в числе используемых для его подключения контактов, а как следствие – форм-факторе корпуса. А следовательно, отличаются и «гнёзда» (сокеты), в которые эти корпуса устанавливаются. Вплоть до Pentium разные производители имели электрическую совместимость, но с выходом на рынок Pentium2 фирма Intel повернулась к потребителям так же, как в сказке «избушка на курьих ножках», и запатентовала свой разъём. (Как следствие стала требовать от других производителей не малых патентных отчислений.) С этого момента процессоры Intel и AMD обрели свои отличающиеся друг от друга разъёмы (гнёзда, сокеты). Выиграли от этого лишь производители материнских плат, которые стали выпускать в 2 раза больше модификаций последних.

Цифра в модели сокета чаще всего обозначает количество процессорных контактов-«ножек». Например, процессоры Intel 5 поколения образца 2015 года предназначены для разъёма FCBGA1168; 2011 года – LGA1366 (старшие модели семейства Core i7) и LGA1155 (семейства Core i5 и Core i7 2xxx), а старые модели тех же серий – для разъёма LGA 1156. А это значит, что новый процессор вы уже не сможете установить на старую системную плату – и наоборот. Что же выбрать? Если следовать рекомендациям Intel, то, конечно же, платы с разъёмом LGA1366 (и, следовательно, процессор Core i7): это отличная основа для самых мощных и быстрых систем.

А более дешёвые платы с разъёмом LGA 1155 – «фундамент» для домашних «середнячков». По мнению В. П Леоньтева [147]: «*Максималисты наверняка подвергнут такую точку зрения решительному осуждению, но ещё раз повторим: гоняться за "крутизной" в компьютерном мире – самый верный способ выкинуть деньги на ветер. В чём доводилось убеждаться лично неоднократно*».

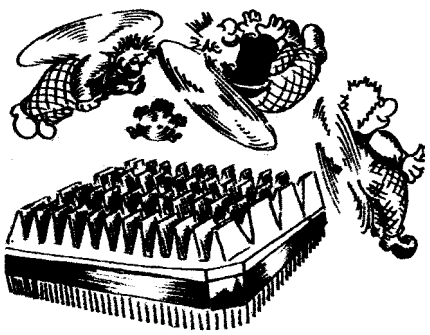
В лагере AMD ситуация более лояльна к потребителям: сегодня практически все процессоры этой компании рассчитаны на разъёмы AM3+ и AM4.

Узнать больше о всех технических характеристиках вашего процессора (внутренней тактовой частоте, частоте шины, размере кэша и т. д.) вы можете в ОС Windows с помощью сторонней бесплатной программы CPU-Z, доступной по адресу http://www.cpuid.com, а в ОС Linux это можно сделать штатными средствами от пользователя с помощью команд `lscpu`, `cpuid` и просматривая содержимое псевдофайла /proc/cpuinfo или от администратора командой `dmidecode -t 4`:

```
$ cat /proc/cpuinfo
processor  : 0
vendor_id  : GenuineIntel
cpu family : 6
model   : 37
model name : Intel(R) Core(TM) i5 CPU         650  @ 3.20GHz
stepping   : 5
cpu MHz    : 1200.000
cache size : 4096 KB
```

[147] Леонтьев В.П. Новейшая энциклопедия компьютера 2011. – М.: ОЛМА Медиа Групп, 2010. – 960 с.: ил. (Новейшая энциклопедия.) ISBN 978-5-373-03920-8.

```
physical id  : 0
... пропущено...
fpu     : yes
fpu_exception: yes
cpuid level  : 11
wp   : yes
flags   : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx rdtscp lm
constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm
sse4_1 sse4_2 popcnt aes lahf_lm ida arat tpr_shadow vnmi flexpriority ept
vpid
... пропущено...
# dmidecode -t 4
... пропущено...
Handle 0x0004, DMI type 4, 35 bytes
Processor Information
   Socket Designation: Socket M2
   Type: Central Processor
   Family: Athlon
   Manufacturer: AMD
   ID: A0 0F 10 00 FF FB 8B 17
   Signature: Family 16, Model 10, Stepping 0
   Flags:
      FPU (Floating-point unit on-chip)
      VME (Virtual mode extension)
      DE (Debugging extension)
... пропущено...
$ cpuid
CPU 0:
   vendor_id = "AuthenticAMD"
   version information (1/eax):
      processor type  = primary processor (0)
      family          = Intel Pentium 4/Pentium D/Pentium Extreme
Edition/Celeron/Xeon/Xeon MP/Itanium2, AMD Athlon 64/Athlon XP-
M/Opteron/Sempron/Turion (15)
      model           = 0xa (10)
... пропущено...
$ lscpu
Architecture:          x86_64
CPU op-mode(s):        64-bit
CPU(s):                6
Thread(s) per core:    1
Core(s) per socket:    6
CPU socket(s):         1
NUMA node(s):          1
Vendor ID:             AuthenticAMD
CPU family:            16
Model:                 10
Stepping:              0
CPU MHz:               3200.000
```



Чтобы процессор не перегревался, на него на термопасту (типа КПТ-8) крепят радиатор. К радиатору, для более сильного охлаждения, в свою очередь, может прикрепляться отдельный вентилятор, на жаргоне называемый «Карлсон».

4.8. Чипсет

Чипсет (chipset) – набор микросхем материнской платы для обеспечения работы процессора с памятью и внешними устройствами.

Раньше компьютер имел до двух сотен микросхем на материнской плате. Современные компьютеры содержат две основные большие микросхемы чипсета либо одну микросхему:

1. Трёхчиповая компоновка (третий чип – процессор):
   - северный мост (North Bridge) – один из двух вариантов:
     - ***контроллер-концентратор памяти*** (*MCH*), который обеспечивает работу процессора с памятью и с видеоподсистемой или
     - ***контроллер ввода-вывода*** (*IOH*) – для процессоров с интегрированным контроллером памяти;
   - южный мост (South Bridge) – ***контроллер-концентратор ввода-вывода*** (*ICH*), обеспечивающий работу с внешними устройствами.
2. Двухчиповая компоновка для некоторых процессоров Core i7 и Core i5 – кроме процессора, используется только одна большая микросхема чипсета (PCH – Platform Controller Hub), обеспечивающая работу с внешними устройствами.

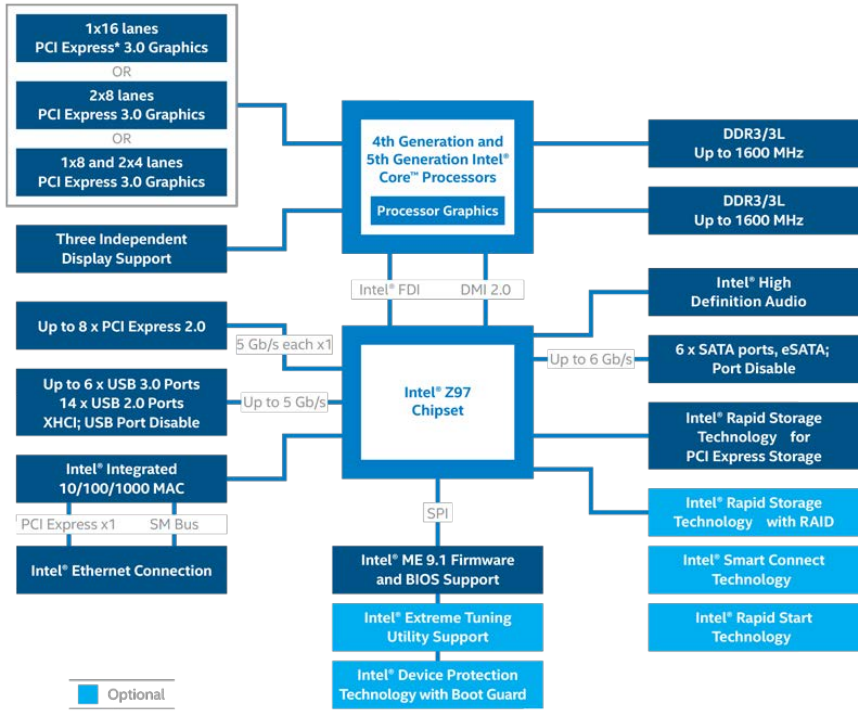


Рисунок 4.40. Принципиальная схема чипсета Intel® Z97

Разработкой чипсетов для настольных ПК занимаются 5 компаний: Intel, AMD, NVIDIA, VIA и SIS.

Выбор типа чипсета зависит от процессора, с которым он работает, и определяет вид других устройств ПК (оперативной памяти, видеокарты, винчестера и прочего). Поскольку для актуальных процессоров даже список чипсетов собственного произ-

водства [148] довольно велик: Z97, Z87, Z77, Z75, Z68, X99, X79, X58, X48, X38, Q965, Q963, Q87, Q77, Q67, Q57, Q45, Q43, Q35, P965, P67, P55, P45, P43, P35, H97, H87, H81, H77, H67, H61, H57, H55, G965, G45, G43, G41, G35, G33, G31, B85, B75, B65, B43, 975X, 946GZ, 945P, 945GC, 945G, 5400, не говоря о чипсетах производителей материнских плат с аналогичными свойствами, перед покупкой чего-либо высокотехнологичного неизбежны: поиск дополнительной информации в сети и изучение отзывов.

Принципиальная схема чипсета Intel® Z97 приведена на рис. 4.40 (информация с сайта фирмы Intel и страницы http://www.overclockers.ru/lab/60457/Znakomimsya_s_chipsetom_Intel_Z97_na_primere_materinskoj_platy_ASUS_Z97-DELUXE.html).

Самостоятельно изучая схему, приведённую на рисунке 4.40, вы можете без труда оценить скорости соединения компонентов, количество поддерживаемых портов и тому подобные параметры. Маркетинг прост: больше – не меньше.

Аналогичные сведения по чипсетам фирмы AMD можно найти на их сайте[149]. На рисунке 4.41 для сравнения представлена аналогичная принципиальная схема чипсета 9-го поколения.

Кстати следует заметить, что чипсет не всегда однозначно влияет на производительность, поскольку все существенно влияющие на скорость узлы находятся в самом процессоре. Поэтому от замены набора логики не стоит ждать того, что разгонный потенциал процессора значительно возрастет. Уже значительное время от чипсета материнской платы практически ничего не зависит, если на руках есть хороший экземпляр процессора и такие же хорошие модули памяти.

Однако не стоит думать, что производители компьютерных компонентов постоянно «клепают» одно и то же, и лишь успевают переклеивать наклейки. Напротив, рост технологий позволяет реализовывать новые функции, которые обеспечивают более комфортную работу для пользователя. Ключевым моментом для любых улучшений служит удобство использования. Однако не только этим путём идут производители, но стараются максимально расширить ассортимент продукции, чтобы охватить максимальную аудиторию покупателей.

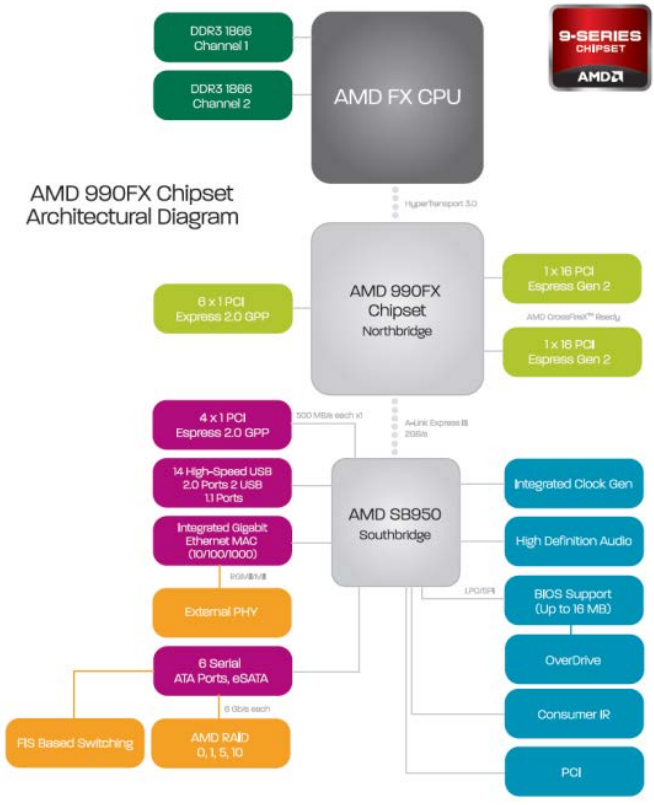


Рисунок 4.41. Архитектура чипсета 9-го поколения от AMD

[148] http://processormatch.intel.com/MotherBoards/.
[149] http://www.amd.com/en-us/products/chipsets

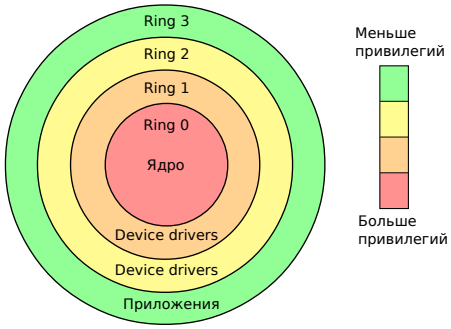
4.8.1. Intel Management Engine, AMD Secure Processor

Говоря о зарубежных чипсетах, стоит упомянуть о Intel Management Engine (ME) – подсистеме, которая встроена во все современные компьютерные платформы (десктопы, лэптопы, серверы, планшеты) с чипсетами компании Intel. Малое количество информации по этой системе в сети интернет объясняется введением NDA[150] производителем.

Эта технология многими воспринимается как аппаратная «закладка», и на то есть причины. Достаточно сказать, что Intel ME является единственной средой исполнения, которая[21]:

- работает даже тогда, когда компьютер выключен (но электропитание подаётся);
- имеет доступ ко всему содержимому оперативной памяти компьютера;
- имеет внеполосный доступ к сетевому интерфейсу.

Если посмотреть на традиционную схему колец защиты (колец привилегий) с нулевого по третье для архитектуры x86 процессоров, то в разговорном языке технологию ME считают кольцом уровня −3, лежащим ниже System Management Mode (кольца уровня −2) и гипервизора виртуализации (кольца уровня −1). Все отрицательные уровни имеют больше привилегий чем ядро операционной системы, работающее на уровне 0. [23]



Компания AMD предлагает на рынке схожую по возможностям технологию AMD Secure Processor, ранее известную как «процессор для обеспечения безопасности платформы», или «PSP». По сути это выделенным процессор с технологией ARM TrustZone®, а также программной защищённой средой Trusted Execution Environment (TEE), призванной обеспечить работу доверенных приложений сторонних разработчиков[151].

4.9. Материнская плата

Материнская плата – печатная плата, на которой осуществляется монтаж микросхем, разъёмов и других компонентов компьютерной системы.

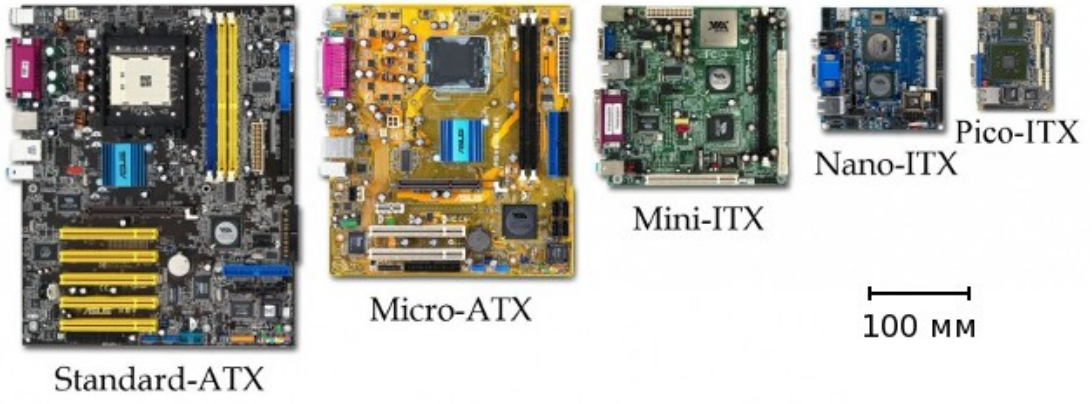
Название происходит от английского *motherboard* [152], иногда используется сокращение *MB* или слово *mainboard* – главная плата, в русских источниках используется также название *системная плата* или на сленге «*мама*», «*мать*», «*материнка*».

На материнской плате располагаются микросхемы чипсета, разъёмы для подключения центрального процессора, оперативной памяти, дисковых устройств, графической платы, звуковой платы и дополнительных внешних устройств.

[150] От англ. Not Disclosure Agreement, соглашение о неразглашении, – вид документа.

[151] См. http://www.amd.com/ru/technologies/security и «О безопасности UEFI, часть заключительная»,. http://habrahabr.ru/post/268423/.

[152] Существуют также и платы именуемые, fatherboard, используемые в основном в многопроцессорном серверном оборудовании высокой производительности. Одновременно может использоваться несколько таких плат. Часто сами платы называют лезвиями, а компьютеры на основе их – blade-системами. (В английском языке blade – лезвие).



Некоторые платы расширения с целью экономии места могут непосредственно интегрироваться в материнскую плату, без использования соединительных разъёмов между собой, например видеокарта, звуковая и сетевая карты.

Существуют различные спецификации (форм-факторы), описывающие конструктивные особенности материнских плат.

Форм-фактор – стандарт (спецификация), определяющий размеры материнской платы, расположение крепёжных отверстий, разъёма центрального процессора (сокета), слотов оперативной памяти, интерфейсов шин, портов ввода/вывода, разъёмов для подключения питания.

Наиболее известен форм-фактор ATX (Advanced Technology eXtended) [153] с достаточно большим размером, что позволяет производителям размещать на материнской плате наибольшее количество устройств. Самые распространённые спецификации форм-факторов показаны в табл. 4.8. Полные спецификации приведены на сайте *http://www.formfactors.org/formfactor.asp*, а для плат Raspberry Pi – *https://en.wikipedia.org/wiki/Raspberry_Pi*.

Таблица 4.8. Некоторые форм-факторы материнских плат

Форм-фактор	Размеры, мм	Примечание
ATX	305 × 244	для корпусов типов MiniTower, FullTower
MicroATX	244 × 244	меньше слотов, чем у ATX
MiniATX	284 × 208	для системных блоков типа Tower и Desktop
Mini-ITX	171,45 × 171,45	новый форм-фактор Intel и VIA (2007 г.)
Nano-ITX	120 × 120	анонсирован VIA Technologies
Pico-ITX	100 × 72	анонсирован VIA Technologies (2007 г.)
BTX	325 × 267	до 7 слотов и 10 отверстий для монтажа платы
MicroBTX	264 × 267	до 4 слотов и 7 отверстий для монтажа платы
PicoBTX	203 × 267	1 слот и 4 отверстия для монтажа платы
Raspberry Pi®	85,6 × 53,98	размером с банковскую карту; модель «1 A+» ещё меньше – 65 × 56,5
Raspberry Pi® Zero	67,6 × 31	модель «PCB ver 1.3» – 65 × 30

[153] До формата ATX широко использовался формат AT, существенным недостатком которого стал механический выключатель электропитания, который невозможно было ни программно выключить, ни включить. С одной стороны, вирусы не могли выключить компьютер программно, но и с другой включить компьютер по технологии Wake-On-Lan было невозможно.

Известны и другие форм-факторы, редко используемые в настоящее время (LPX, MiniLPX, NLX и прочие). Отдельного внимания заслуживает форм-фактор компьютеров «Raspberry Pi» и «Raspberry Pi Zero», о которых мы расскажем в конце раздела. Форм-фактор важен, потому как от него зависит тип системного блока ПК. В маленький корпус большая плата не поместится, может не хватить мощности блока питания или могут возникнуть проблемы с охлаждением из-за высокой плотности компоновки деталей. «Портативные» платы не столь требовательны к питанию, но размеры однозначно влияют на привлекательность создаваемых на основе их различых устройств.

Замечание. Raspberry Pi – это бренд задавший направление на миниатюризацию ЭВМ и открытые подходы. В погоне за совершенгством у него появилось множество разнообразных аналогов. Советуем просмотреть публикацию «Обзор 98 одноплатных компьютеров» состоящую из 4-х частей: https://geektimes.ru/post/290679/, https://geektimes.ru/post/290685/, https://geektimes.ru/post/290705/, https://geektimes.ru/post/290785/.

Формат MicroATX

Ниже приведён наиболее распространённый пример расположения портов ввода/вывода согласно спецификации ATX (см. рис. 4.42).

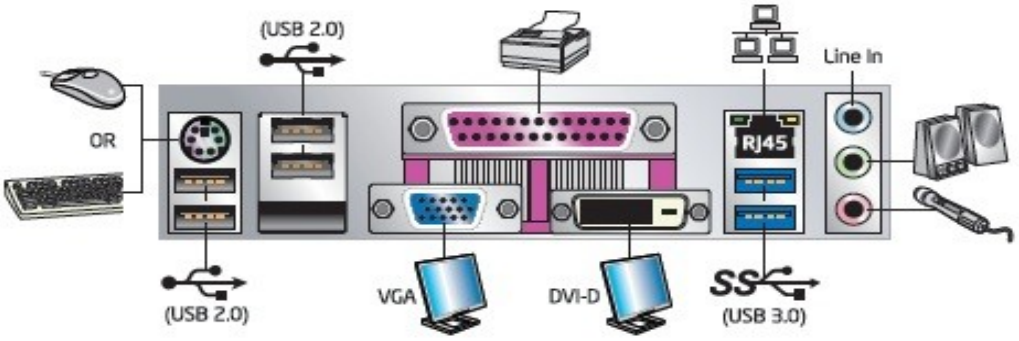


Рисунок 4.42. Пример портов внешних устройств на задней панели

Наиболее важными микросхемами материнской платы являются северный (при его наличии) и южный мосты чипсета. Именно чипсет определяет, в значительной степени, особенности материнской платы и то, какие устройства могут подключаться к ней.

Благодаря тому что на рынке производителей материнских плат для ПК не было и нет монополии, даже под один тип процессорного разъёма различные известные фирмы типа Asus, ASRock, Foxxcon, Gigabyte, MSI и др. производят десятки модификаций. Мы не будем вдаваться во все вариации конфигураций и не ставим своей целью всё рассмотреть. Для общего ознакомления читателей поверхностно рассмотрим лишь общие тенденции на примере одной из плат[154].

Таблица 4.9. Характеристики материнской платы Intel® Desktop Board DQ77CP

Форм-фактор	MicroATX
Процессор	Intel Core 2-го или 3-го поколения
Чипсет	Intel® Q77 Express

[154] http://www.intel.com/cd/products/services/emea/rus/motherboards/desktop/323255.htm.

Память	Четыре 240-контактных разъёма DIMM для двухканальных модулей DDR3 1600, 1333 или 1066 МГц Поддержка до 32 ГБ системной памяти
Аудио	Подсистема *Intel® High Definition Audio* в следующей конфигурации: 8-канальная (6.2) аудиоподсистема с поддержкой аудиокодека Realtek ALC892
Видео	Встроенная графическая подсистема процессоров
Сеть	Гигабитная (10/100/1000 Мбит/сек) сетевая подсистема на базе Intel® 82579LM Gigabit Ethernet Controller
Периферийный интерфейс	4 порта *USB 3.0;* 8 портов *USB 2.0* Один последовательный порт Пять портов *Serial ATA* Порт PS/2 для клавиатуры или мыши Один последовательный порт наушников Один разъём параллельного порта на заднюю панель
Возможности расширения	Один разъём для дополнительной платы *PCI Express* x16 Один разъём PCI Express 2.0 x1 Два разъёма PCI Conventional

Внешний вид материнской платы Intel DQ77CP показан на рис. 4.43, её схема – на рис. 4.44.

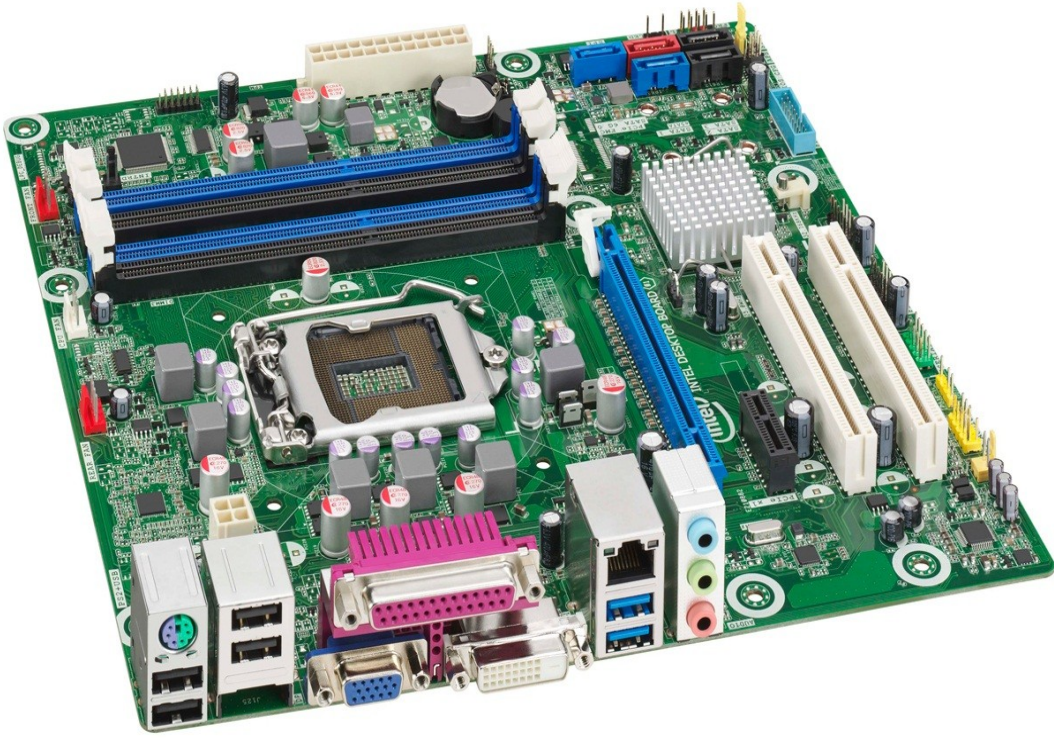


Рисунок 4.43. Материнская плата Intel DQ77CP

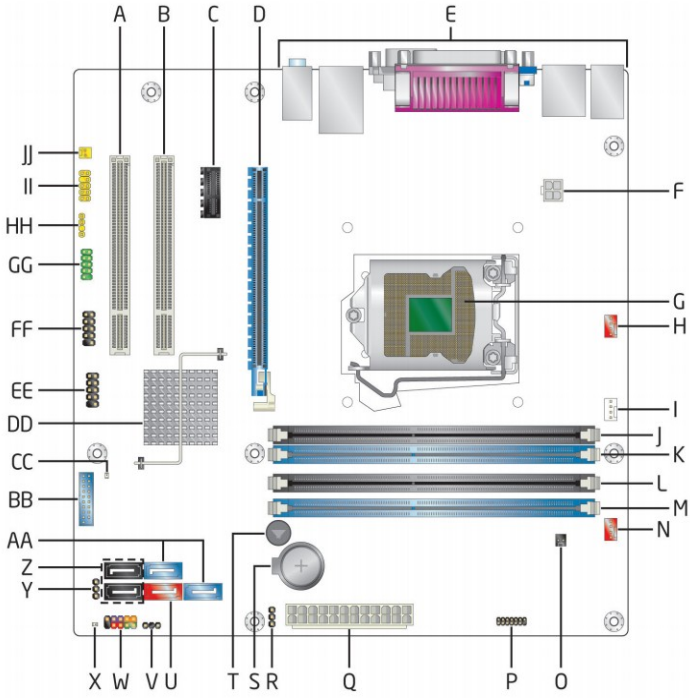


Рисунок 4.44. Схема материнской платы Intel DQ77CP

Таблица 4.10. Обозначения на схеме рисунка 4.44

A разъём шины PCI для установки плат расширения	**H** подключение блока задних вентиляторов
B разъём шины PCI для установки плат расширения	**I** разъём подключения вентилятора процессора
C разъём шины PCI Express x1 для установки плат расширения	**J** разъём для подключения памяти DIMM[155] 3 (канал A, DIMM 0)
D разъём шины PCI Express x16 для установки плат расширения	**K** разъём для подключения памяти DIMM 1 (канал A, DIMM 1)
E разъёмы с задней стороны платы	**L** разъём для подключения памяти DIMM 4 (канал B, DIMM 0)
F подключение 12 В питания для ядра процессора (разъём 2×2 контакта)	**M** разъём для подключения памяти DIMM 2 (канал B, DIMM 1)
G разъём LGA1155 для установки процессора	**N** разъём подключения вентиляторов системного блока
O датчик вскрытия корпуса	**Z** чёрные разъёмы SATA 3.0 Гб/с
P отладочный разъём Low Pin Count (LPC)	**AA** синие разъёмы SATA 6.0 Гб/с
Q Основной разъём электропитания (2 ⏤ 12 контактов)	**BB** контакты подключения разъёма USB 3.0 на переднюю панель корпуса (синий цвет)

[155] Аббревиатура DIMM – Dual Inline Memory Module – будет разъяснена в разделе 4.6 «Оперативная память».

R контакты сброса данных Intel® Management Engine BIOS Extension (Intel® MEBX)	**CC** светодиодный индикатор состояния Intel® Management Engine «M»
S Батарейка для часов и памяти BIOS (обычно литиевая CR2025 или CR2032)	**DD** чипсет Intel Q77 Express
T небольшой пьезоэлектрический динамик (PC-Speaker)	**EE** подключение разъёмов USB 2.0 (для передней панели корпуса)
U красный eSATA разъём до 3.0 Гб/с	**FF** подключение разъёмов USB 2.0 (для передней панели корпуса)
V разъём для подключения переднепанельного индикатора режимов «питание/сон»	**GG** разъём подключения планки последовательного COM-порта (RS-232)
W блок контактов для подключения кнопок питания, reset и прочих индикаторов передней панели	**HH** цифровой аудиовыход S/PDIF [156]
X разъём индикатора режима «Standby» (наготове)	**II** звуковые разъёмы передней панели корпуса
Y перемычка сброка конфигурации BIOS Setup	**JJ** разъём подключения внутреннего динамика (моно)

На материнской плате присутствует ещё одна[157] маленькая, но очень важная микросхема материнской платы – **BIOS** (Basic Input-Output System – базовая система ввода-вывода), а также устройства для её питания – микросхема BP_BIOS и аккумуляторная батарейка (BATTERY). Назначение этой микросхемы с постоянным питанием и наличием собственной энергонезависимой памяти (перепрограммируемая Flash EEPROM):

- конфигурирование компьютера (программа BIOS Setup) и сохранение параметров конфигурации;
- выполнение программы часов и календаря компьютера;
- при включении компьютера:
  - выполнение программы самотестирования (POST, power-on self-test);
  - поиск главной загрузочной записи (MBR) на дисках или иных устройствах для осуществления последующей загрузки ПК [158];
  - далее MBR (находящийся на диске или его эмуляции, если управление было передано ему) выполняет поиск активного (загрузочного) раздела, и если он найден, то запускает загрузчик ОС или тот код, что окажется вместо загрузчика.

В программе BIOS Setup (которая запускается нажатием клавиши Delete или другой в момент включения компьютера) есть возможность настроить порядок устройств для поиска загрузчика операционной системы – если необходимо, её можно загрузить

[156] S/PDIF, S/P-DIF – Sony/Philips Digital Interface (или Interconnect) Format. Разъём, протокол и формат для передачи «цифрового» звука, описывается в стандарте IEC-60958 как IEC 958 type II.

[157] На некоторых материнских платах может быть установлено две (основная и дублирующая) и даже три микросхемы BIOS'a. Ранее, предпочтительно до появления Flash EEPROM, микросхема BIOS'a не впаивалась, а устанавливалась через «разъём-панельку».

[158] Или на внешних устройствах: дисковод, CD/DVD/Blue-Ray-привод, внешний USB- носитель.

с CD-, DVD-диска или даже через USB-порт. Эта возможность необходима для установки новой операционной системы на компьютер (или её переустановки).

BIOS – это историческая прерогатива, если не традиция использования на всех IBM PC-совместимых ПК. Из-за наличия различных ограничений в первых версиях, например в адресуемости адресного пространства и прочего, на смену ему на серверном рынке пришёл EFI[159]. Первая спецификация EFI была разработана Intel совместно с HP в начале 2000-х годов, но позднее, в 2005 году (начиная с версии EFI 1.10), от первого названия отказались, и последняя версия стандарта носит название Unified Extensible Firmware Interface (UEFI) [160]. Последняя версия спецификации UEFI 2.4 Errata B была представлена в апреле 2014 года и может использоваться не только для серверных платформ, но и для ПК.

Назначение **слотов** (разъёмов) на материнской плате – подключение различных плат сопряжения с внешними устройствами и реализация дополнительных возможностей:

- **PCI Express x16** – только для соответствующих видеоплат;
- **PCI Express x1** – новые звуковые, сетевые платы;
- **PCI** – старые звуковые, сетевые платы, внутренние модемы, FM-, TV-тюнеры.

PCI (Peripheral Component Interconnect bus) – шина для подсоединения периферийных устройств. Впервые массово применяться начала для Pentium-систем.

Шина PCI Express x16 пришла на смену шине AGP, ранее использовавшейся для подключения видеокарт.

Внутренние порты, расположенные на материнской плате, служат для подключения устройств:

– **Serial ATA** – дисковые устройства (HD, DVD) соответствующего типа;
– **ATA IDE** – дисковые устройства (HD, CD, DVD) соответствующего типа.

ATA (Advanced Technology Attachment) – спецификация интерфейса для подключения дисковых устройств. **IDE** (Integrated Drive Electronics) – диск со встроенным контроллером управления для подключения с помощью параллельного интерфейса спецификации ATA. Для спецификации UDMA-33 используются 40-жильные шлейфы и 40-контактные разъёмы, начиная с UDMA-66 и выше, с целью минимизации влияния жил друг на друга вместе с 40-контактными разъёмами используются 80-жильные шлейфы, где через один подключена земля. Сейчас указанные разъёмы практически не используются – их полностью вытеснила технология Serial ATA.

Рисунок 4.45. Кабели подключения IDE и SATA

Serial ATA (SATA) – новый последовательный интерфейс подключения дисков. На рис. 4.45 показаны кабели подключения для дисков параллельного и последова-

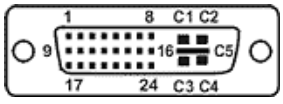
[159] Extensible Firmware Interface (EFI) (читается как «ифай») – интерфейс между операционной системой и микропрограммами, управляющими низкоуровневыми функциями оборудования, его основное предназначение: корректно инициализировать оборудование при включении системы и передать управление загрузчику операционной системы.
[160] В настоящее время разработкой UEFI занимается Unified EFI Forum http://www.uefi.org/.

тельного интерфейсов. Чаще всего цвет кабеля красный или синий, на характеристики работы он никак не влияет.

Для серверного оборудования и самых продвинутых пользовательских ПК для подключения жёстких дисков используются контроллер **Serial Attached SCSI (SAS)** и соответствующие ему разъёмы (SFF 8482, SFF 8484) по форм-фактору, совместимые с SATA-устройствами: они позволяют SATA-устройствам соединяться с SAS-контроллером или планкой SAS-разъёмов, что устраняет необходимость в дополнительном SATA-контроллере для подключения SATA-устройств типа DVD-приводов. Однако жёсткие диски с интерфейсом SAS не могут подключаться к шине SATA, потому что не поддерживаются контроллером SATA, а их физический разъём имеет «ключ», не позволяющий подключения к шине SATA.

Внешний порт – разъём на задней или передней панели системного блока, используется для подключения различных устройств:

- **USB 3.0/2.0** – принтеры, сканеры, внешние модемы, TV-тюнеры, клавиатура, мышь, внешние дисковые накопители, Flash Drive и многое другое; на относительно «старых» компьютерах можно ещё встретить электрически совместимый USB 1.1, имеющий меньшую скорость передачи;
- **IEEE-1394 (FireWire)** – цифровые видеокамеры, дисковые устройства;
- **PS/2** – клавиатура и мышь (см. рис. 4.46);
- разъём **DVI** (см. рис. 4.46) – цифровой выход на монитор (при наличии интегрированной видеоподсистемы);



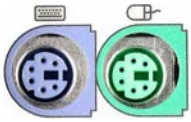


Рисунок 4.46 Разъёмы DVI для подключения монитора к системному блоку ПК и PS/2 – мыши и клавиатуры

- разъём **VGA (D-SUB 15 pin)** – аналоговый выход на монитор (при наличии интегрированной видеоподсистемы);
- в некоторых системах можно встретить разъём DisplayPort для подключения монитора, как правило, для систем с интегрированным видео не используется;
- при наличии интегрированной аудиоподсистемы разъёмы для подключения колонок и микрофона.

Кроме того, могут присутствовать «морально устаревшие» на сегодняшний день порты для подключения следующих устройств:

- **COM (RS-232)** – мышь, модем, ИК-приёмник, управление ИБП и прочее;
- **LPT** – принтер, сканер (старые модели), кард-ридер и прочее.

Новые интерфейсы подключения внешних устройств – USB, SATA, IEEE-1394 (FireWire) – это последовательные шины (Serial Bus) с небольшим количеством проводов (4–7) в кабеле.

Параллельные шины ATA IDE, LPT имеют значительно большее количество проводников в кабеле (IDE 40 или 80 проводов, LPT – 18) и, соответственно, контактов на разъёмах.

Скорость работы современных последовательных шин выше, чем параллельных.

Наиболее часто используется в настоящее время порт USB (Universal Serial Bus – универсальная последовательная шина), который имеет 3 типа: USB 1.0/1.1, USB 2.0 и немного отличающийся по разъёму USB 3.0.

Особенности этой шины:

- высокая скорость работы – до 480 Мбит/с для USB 2.0 и до 5 Гбит/с для USB 3.0;
- длина кабеля подключения – до 5 метров;
- количество подключаемых устройств (включая разветвители) – до 127;
- возможно подключение устройств с различными скоростями обмена;
- возможность подключать и отключать устройства при работающем компьютере;
- возможность подавать питание напряжением 5 В на маломощные периферийные устройства (ток до 500 мА, у USB 3.0 – 900 мА);
- кабель подключения имеет различные разъёмы для подключения к компьютеру (тип «А») и к внешнему устройству (тип «В»).

Шина USB предоставляет в полной мере возможность использования технологии Plug&Play для периферийных устройств, то есть при подключении нового устройства к работающему компьютеру (наиболее часто в настоящее время – флэш-накопителей, внешних жёстких дисков, «мышек» на ноутбуках, телефонов, GSM/WiMAX/LTE-модемов и т. п.) происходят его автоматическое распознавание и установка соответствующих драйверов (либо использование функций или модулей ядра), необходимых для работы с этим устройством. Для использования большого количества USB-портов применяются USB-концентраторы (хабы), как внешние, так и встроенные в монитор, клавиатуру и прочее.

В прайс-листах фирм по продаже компьютеров обычно приводится краткое обозначение материнской платы примерно следующего вида: **MB VPro Q77, LGA1155, 4*GDDR3(1600), PCI-E, mATX, GLan, 2*SATA/2*6G, 5.1CH, 2*USB 3.0, D-Sub/DVI, eSATA,** что означает: материнская плата **VPro Q77**, разъём процессора **LGA1155**, чипсет **Intel Q77**, память **DDR3** с частотой **1600** МГц, шина и слоты **PCI Express,** дисковый интерфейс 2 шт. **SATA2 6 ГБ/с,** звуковой адаптер 5.1, интегрированные сетевой адаптер Gigabit Ethernet (**GLan**), видеоадаптер с выходами **D-Sub** и **DVI**, форм-фактор **mini-ATX**, внешний порт **eSATA.**

Стоимость обычных материнских плат на чипсетах 8–9-го поколений от 2500 до 15000 руб.

Корпус системного блока ПК подбирается в зависимости от форм-фактора материнской платы и типа процессора. Бывает двух видов – TOWER (вертикальный) и DESKTOP (горизонтальный). Обычно в комплекте с корпусом идёт блок питания, мощность которого тоже может быть выбрана в соответствии с типом процессора, видеокарты и количеством периферийных устройств и лежит в настоящее время обычно в диапазоне 350–550 Ватт. Стоимость обычных корпусов 2500–8000 руб., однако существуют и корпуса стоимостью более 15 000 руб. (например, для домашних медиацентров).

Самые дешёвые корпуса брать не рекомендуем по той причине, что, скорее всего, их стенки будут сделаны из тонкой (и как следствие гремящей от вибрации) жести, а не более тяжёлой и тихой в использовании стали, незавальцованные края внутренних стенок могут повредить ваши руки, а также велика вероятность маломощного и некачественного блока питания.

Выпускаются также ПК-моноблоки (фирмы HP, Apple и др.) – в этом случае все устройства, размещаемые обычно в системном блоке, располагаются в одном корпусе с жидкокристаллическим монитором (может быть сенсорным). Толщина такого моноблока лишь немного больше толщины обычного монитора. В последнее время подобные компьютеры становятся более популярными.

4.9.1. Raspberry Pi®

Отдельного внимания заслуживает материнская плата компьютера Raspberry Pi[161] размером с пластиковую карту (85,6 × 53,98 × 17 мм, см. рис. 4.47). Выпускается в пяти комплектациях: модели «A» и «A+» без разъёма Ethernet и модели «B», «B+» и модель «B» 2-го поколения («2B») – с ним. Модель «A+» чуть меньше всех остальных – 65 мм. Все версии первого поколения оснащены ARM11 (ARMv6) процессором, работающим на частоте 700 МГц. Второе поколение содержит 900 МГц четырёх ядерный процессор Cortex-A7 (ARMv7). Оперативная память: «A», «A+» – 256 МБ; «B», «B+» – 512 МБ; «2B» – 1024 МБ. В среднем россиянам плата обходится в районе 3000–5000 руб. и поставляется в антистатическом пакете в маленькой белой картонной коробочке размером с колоду игральных карт. Корпус в комплект поставки не входит. На четырёхслойной печатной плате размещается не только процессор, но и большое количество периферийных разъёмов типа выхода HDMI, USB и прочих. В качестве жёсткого диска для загрузки ОС предполагается использование SD-карт памяти. Поддерживаются различные ОС, предпочтительно Linux различных версий, но есть также и FreeBSD, и Android.

В магазинах электроники (и в интернете) можно найти всевозможные платы расширения и аксессуары к указанным моделям: корпуса, отладочные платы, модуль видеокамеры, радио-модуль и другие.

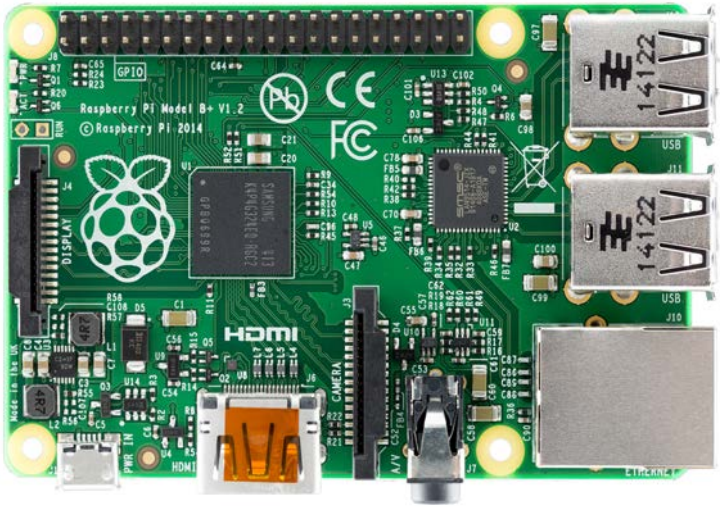


Рисунок 4.47. Материнская плата компьютера Raspberry Pi® (модель «B+»)

[161] http://www.raspberrypi.org/.

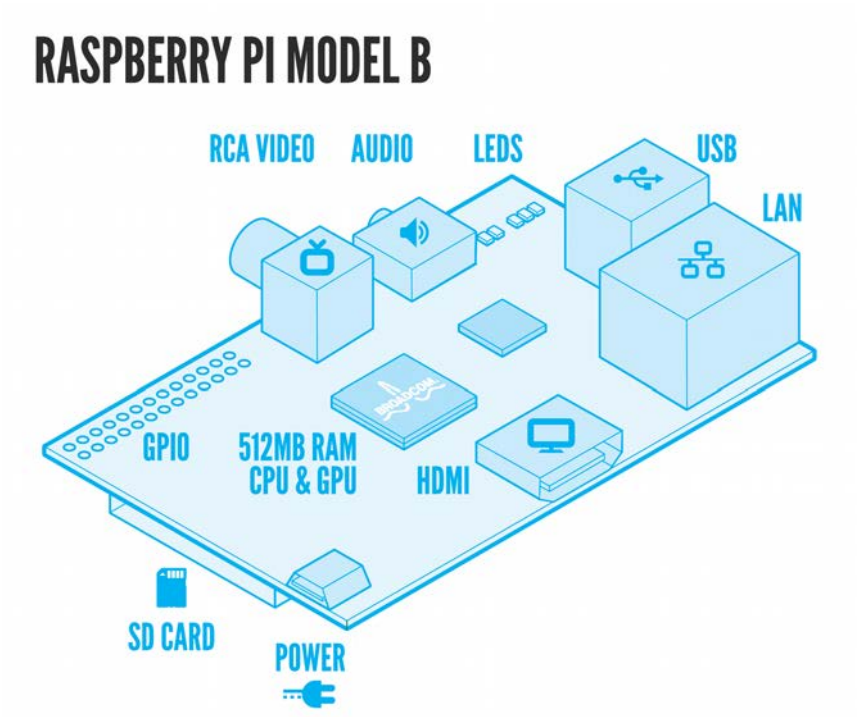


Рисунок 4.48. Схема расположения разъёмов и элементов на материнской плате компьютера Raspberry Pi® модель «B»

Если вас заинтересовали платы Raspberry Pi®, то также рекомендуем обратить внимание и на более малую по размерам модель – Raspberry Pi® Zero, а также книгу Эдварда Шнайдера с множеством примеров как с ней работать: *Edward Shajder Rapberry Pi Zero Cookbook, Packt Publishing, UK, 2017, ISBN 978-1-78646-385-2.*

4.10. Оперативная память

Оперативная память – один из основных компонентов компьютера, предназначенный для хранения информации (программ и обрабатываемых программами данных) во время исполнения программ на компьютере.

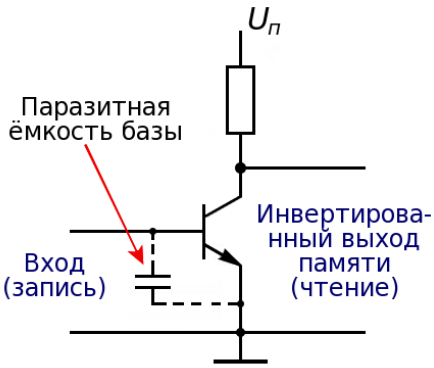
Всю оперативную память до начала её использования в каких-либо устройствах сегодня можно поделить по внутреннему устройству на два вида: динамическую (Dynamic RAM, DRAM) и статическую (Static RAM, SRAM) [162].

В подразделе «*4.5.3.4. От триггеров к статической памяти*» (см. стр. 217) мы показали пример построения ячейки памяти, выполненной на элементарных логических элементах (с использованием триггеров), так как это эволюционный, простой и максимально понятный для восприятия большинством читателей алгоритм работы памяти. Однако, как всегда бывает в жизни, находятся недостатки и оптимизаторы, эти недостатки устраняющие.

[162] Аббревиатура RAM – *Random Access Memory* – будет разъяснена позже.

Память, построенная на триггерах, является статической. Она быстрая и хорошая, но к её недостаткам можно отнести сложность, а именно большое число транзисторов, необходимых для реализации хранения одного бита, а как следствие получается высокая стоимость хранения единицы информации. Если для схемы хранения одного бита в лучшем случае потребуется порядка 5 транзисторов, рассчитайте, сколько ключевых элементов будет в компьютере с 32 Гбайтами оперативной памяти [163]. Сколько она будет потреблять электричества и производить тепло из-за невозможности достижения 100% КПД?

Чтобы снизить себестоимость, была придумана динамическая память, в идеале хранящая «1 бит на одном транзисторе».



Алгоритм работы динамической памяти следующий. В цепи управления ключевым элементом (обычно транзистором) используется небольшой накопитель – конденсатор (часто это паразитные ёмкости самого транзистора), собственно и «хранящий» внутреннее состояние этой ячейки памяти. Как мы помним из школьного курса физики, заряженный конденсатор, подключенный к высокоомной нагрузке, разряжается не сразу, а постепенно, что позволяет нам в пределах некоторого конечного промежутка времени использовать его для управления ключевым элементом [164], на выходе которого можно считывать значения напряжения. Если напряжение превышает некоторый порог, то можно читать, что ячейка хранит логическую «1».

Чтобы память хранила данные дольше, чем разряжаются её конденсаторы, необходимо организовать постоянную перезапись. Обычно этим занимается аппаратный контроллер памяти, а процесс называется регенерацией. В большинстве случаев контроллер работает прозрачно не только для пользователей, но и для программистов, поэтому о его существовании они также могут ничего не знать. Однако с увеличением скоростей работы время, отводимое на регенерацию памяти, даёт о себе знать. Так, в прайс-листах компьютерных фирм появились дополнительные характеристики, затрагивающие «карман пользователя» – тайминги, а в большинстве материнских плат можно встретить различные банки памяти, обычно это парные разъёмы одного цвета, запись в которые происходит с чередованием [165].

Так как в большинстве операций запись данных по адресам памяти (или чтение) проходит последовательно, можно использовать чередование адресов банков как полос в зебре, чем можно добиться эффекта отсутствия ожидания. Дословно: пока одна планка памяти занята регенерацией, мы работаем с другой.

***Резюмируем* коротко суть вышеизложенного:**

Современная оперативная память – динамическая (DRAM), содержимое её остаётся неизменным в течение очень короткого промежутка времени, поэтому память должна периодически обновляться (регенерироваться). Запоминающим элементом динамической памяти является конденсатор, который может находиться в заряженном

[163] 8 бит, умноженные на 1024, на 1024, на 1024 и на 5.

[164] Заряд конденсатора очень мал, чтобы использовать его вообще без усиливающего ключевого элемента – транзистора.

[165] Параметры записи с чередованием или без могут настраиваться в BIOS.

или разряженном состоянии. Если конденсатор заряжен, то в ячейку памяти записана логическая «1», если разряжен – логический «0». В идеальном конденсаторе заряд может сохраняться неограниченное время, в реальном конденсаторе существует ток утечки, поэтому информация, записанная в динамическую память, со временем будет утрачена в результате разрядки конденсаторов.

Единственным способом регенерации хранимой в памяти информации является выполнение операций чтения и повторной записи данных, выполняемых через определённые промежутки времени (например, каждые 2 мс) всех ячеек памяти. В эти моменты процессор находится в состоянии ожидания. Регенерация памяти происходит также при выполнении каждой операции чтения данных.

В описаниях термина «оперативная память» можно встретить как быструю и более дорогую статическую память, так и динамическую. Обычно под «оперативной памятью» ПК понимают RAM – Random Access Memory, память с произвольным доступом, ОЗУ – оперативное запоминающее устройство, память, включающую в себя и DRAM, и SRAM.

Современные ПК также имеют память видеокарты (с которой работает процессор видеокарты), а также «сверхоперативную» память центрального процессора (кэш [166] первого, второго и иногда третьего уровня), которая, как вы, наверное догадались и есть статическая память (SRAM).

Для долговременного хранения информации используются дисковые и прочие устройства, которые иногда называют внешней памятью (описаны далее в разделе 4.9 «*Устройства хранения информации*»).

Оперативная память, так же как и процессоры, имеет длительную историю своего развития – от ферритовых колец промышленных ЭВМ до современных сменных модулей, используемых в ПК ёмкостью в несколько Гбайт.

Говоря о современных модулях памяти, помимо объёма их делят по напряжению питания, используемым разъёмам [167], тактовой частоте и задержкам (таймингам). Тактовую частоту для прайс-листов часто пересчитывают в пропускную способность. А такой существенный для покупателей параметр, как надёжность, или наработка часов на отказ, в прайс-листах вообще не встречается, оказываясь условно «размытым» где-то в графах «срок гарантии», «производитель» и «цена».

Выбор памяти напрямую зависит от чипсета, используемого на компьютере, который и определяет тип оперативной памяти, который можно использовать.

4.10.1. Внутреннее устройство памяти

Ячейки памяти организованы в матрицу, состоящую из строк и столбцов. Полный адрес ячейки данных включает два компонента – адрес строки и адрес столбца. При чтении информации из памяти вначале считывается информация всех ячеек памяти для строки с заданным адресом и помещается в буфер ввода/вывода. Далее в соответствии с адресом столбца данные выбираются из буфера ввода/вывода и поступают на выход динамической памяти.

[166] В общем случае под кэшированием следует понимать использование небольшой и более быстрой памяти, по сравнению с большей по размеру и более медленной. Например, можно кэшировать данные, записываемые на жёсткий диск, в этом случае оперативная память будет сама выступать в роли кэша.
[167] Например, специальные уменьшенные разъёмы для ноутбуков.

Микросхемы памяти состоят из отдельных матриц, каждая из которых имеет собственную линию чтения/записи. В этом случае одновременно считывается или записывается несколько бит информации. Количество линий определяет разрядность шины ввода/вывода. Современные микросхемы памяти 64-разрядные.

Ёмкость матрицы памяти (глубиной адресного пространства, address depth) и их количество (разрядность) определяет общий объём микросхем памяти, которые размещаются на модулях памяти (DIMM-модули). DIMM-модуль (Dual In-Line Memory Module) имеет с двух сторон электрически независимые контакты для подключения к шине контроллера памяти через слот, в который они вставляются.

Наиболее распространёнными являются 240-контактные 64-разрядные модули DIMM, имеющие по 120 контактов с каждой стороны (см. рис. 4.49).

Рисунок 4.49. Модули памяти DDR3

Хотя число контактов у модулей DDR2 и DDR3 одинаково, слоты, предназначенные для памяти разного типа, отличаются расположением «ключа» – прорези между контактами. Поэтому установить память DDR3 SDRAM в разъём для памяти DDR2 DIMM и наоборот не удастся.

SDRAM – микросхема синхронной динамической памяти (Synchronous DRAM), в которой процессы записи и считывания информации синхронизированы с шиной памяти.

DDR SDRAM (Double Data Rate – двойная скорость передачи данных) – усовершенствованный стандарт SDRAM, при использовании которого скорость передачи данных удваивается. Это достигается не за счёт удвоения тактовой частоты, а за счёт передачи данных дважды за один цикл: первый раз в начале цикла – по фронту тактового импульса, а второй в конце – по его срезу.

DDR2, **DDR3** и **DDR4** – последующие модификации памяти DDR SDRAM, позволяющие за один такт записывать или считывать 4 (DDR2) или 8 (DDR3), или 16 (DDR4) блоков данных, имеют пониженное напряжение питания (DDR – 2,5 В, DDR2 – 1,8 В, DDR3 – 1,5 В, DDR4 – 1,2 В) и энергопотребление.

Основное отличие DDR4 заключается в повышенных частотных характеристиках, пониженном напряжением питания и в удвоенном до 16 числе банков. Максимально определённая спецификацией пропускная способность памяти – 34,1 ГБ/с (на частоте 4266 МГц). Кроме того, повышена надёжность работы за счёт введения механизма контроля чётности на шинах адреса и команд.

Память DDR2 рассчитана на работу на частоте от 533 МГц (обозначается как PC2-4200 по скорости передачи данных 4200 Мб/с) до 1066 МГц (PC2-8500), на данный момент считается устаревшей и в новых моделях компьютеров не используется.

Память DDR3 выпускается для работы с частотой от 1333 МГц до 2666 МГц, маркировка DDR3 в зависимости от частоты: PC-10600 … PC-21300 соответственно. Иногда пишут PC3-10600 … PC3-21300, чтобы подчеркнуть использование памяти DDR3.

В обозначениях модулей памяти встречаются также обозначения:

- **ECC** – память с коррекцией ошибок (Error Correct Code) – имеет дополнительную 8-разрядную микросхему памяти, в результате модули с ECC являются 72- разрядным (в отличие от стандартных 64-разрядных модулей);
- **Full Buffered (FB)** – память с буферизацией данных, имеет дополнительные микросхемы регистров для буферизации данных, наиболее часто используются в регистровых модулях (Registered, к спецификации которых предъявляются дополнительные требования), для серверов. Не совместима с не регистровой памятью.

Более ранние (старые) модули памяти были асинхронными и имели аббревиатуры: FPM RAM, EDO RAM. Отдельными линейками можно выделить модули памяти используемые для видеокарт: EDO, VRAM, WRAM, MDRAM, SGRAM, SDRAM, DDR, DDR2, DDR3, GDDR2, GDDR3, GDDR4, GDDR5 и специфичные типы памяти: RDRAM, XDR DRAM, XDR2 DRAM.

Нельзя сказать, что вся память DDR3 однозначно быстрее работает, чем DDR2, а DDR4 быстрее чем DDR3. Объективно сравнивайте память по характеристикам.

Стоимость памяти для серверов и ПК с высокими характеристиками достаточно высока, однако для бытового применения не нужны очень большие объёмы и скорости, не нужна и коррекция ошибок. Память с коррекцией ошибок вместо обычной памяти работать не будет. В обычных магазинах серверная память продаётся редко из-за малого спроса, так что шансов у вас её купить по ошибке очень мало.

Необходимый объём оперативной памяти для настольных персональных компьютеров зависит в первую очередь от операционной системы и от приложений пользователя, которые используются на ПК.

До недавнего времени наибольшее применение находили модули памяти DDR2 ёмкостью 2 Гбайт (2 модуля в комплекте для двухканальной памяти), для семейства процессоров Core i3/i5/i7 используют 2 модуля по 4–8 Гб DDR4.

4.11. Устройства хранения информации

Устройства хранения информации (внешняя память) – компоненты компьютера, позволяющие длительное время (сравнимое со временем жизни самого устройства) сохранять большие объёмы информации без потребления электроэнергии (энергонезависимые).

Если окунуться в школьный курс физики (раздел «Магнетизм»), вы обязательно вспомните, что все вещества по их магнитным свойствам можно поделить на три категории: диа-, пара- и ферро-магнетики. Именно последние и обеспечили возможность будущим пользователям ПК эффективно хранить свои данные. Если на небольшую основу нанести, а лучше приклеить, порошок с ферромагнитными свойствами, то, используя эффект сохранения им ориентации намагниченности после снятия внешнего магнитного поля, можно организовать хранение информации. Информацию можно закодировать направлением остаточной намагниченности небольших участков (доменов).

Первыми массовыми магнитными устройствами хранения информации для ПК были: для любителей – обычные магнитофонные аудио-кассеты, а для профессионалов – дисководы (Floppy-дисководы, FDD, Floppy Disk Drive) и их сменные носители – дискеты – вначале пятидюймовые (5,25″) ёмкостью 360 КБ, 720 КБ и 1,2 МБ, затем трёхдюймовые (3,5″) ёмкостью 1,44 МБ. Дискеты ёмкостью 2,88 МБ не получили широкого распространения. В настоящее время применяются редко в связи с широким распространением устройств флэш-памяти ёмкостью в десятки, а то и сотни гигабайт.

Примечание. На больших ЭВМ в эпоху до появления ПК для хранения информации использовались бумажные перфокарты, перфоленты, магнитные ленты и диски.

4.11.1. Винчестер

Объявление на форуме 1990-х:
«Куплю винчестер, оружие не предлагать!»

Хранение информации на гибких магнитных дисках обеспечивало удобную смену дисков, но плохую надёжность и довольно низкую скорость работы. С целью повышения «плохих показателей» было предложено повысить качество записи. Для этого была создана более точная механика перемещения головок, введены меньшие допуски, магнитное вещество стали наносить с большой плотностью и на более твёрдый носитель (алюминиевая, стеклянная или керамическая пластина, от чего диск стал называться жёстким), сам магнитный диск получил более мощный двигатель, защитный корпус и стал постоянно вращаться, обеспечивая максимальную готовность и минимальные задержки.

Так началась новая эра хранения информации для ПК – винчестеры (или жёсткие диски – Hard Disk Drive, HDD). До последнего времени использование жёсткого диска было обязательным компонентом каждого настольного персонального компьютера. Позже в офисных компьютерах их немного потеснили бездисковые сетевые рабочие станции с загрузкой операционной системы с сервера и работающие по сети с сетевыми хранилищами или сетевыми дисковыми устройствами сервера, а также терминалы (тонкие клиенты). В секторе домашнего использования ПК, ноутбуках и нетбуках жёсткие диски постепенно вытесняются твердотельными накопителями, речь о которых пойдёт в следующем разделе.

4.11.1.1. История названия

В 1895 году известной американской оружейной фирмой «Винчестер» был разработан и пущен в серийное производство патрон «.30-30» для применения в сконструированной за год до того винтовке с рычажным взводом Winchester Model 1894. Патрон

стал первым созданным в США боеприпасом для гражданского оружия «малого» калибра на бездымном порохе[168]. Новый боеприпас немедленно завоевал широчайшую популярность, поскольку превосходил патроны старых поколений по всем показателям.

Первая «тридцадка» в обозначении, идущая после точки, – это калибр в сотых долях дюйма. Ноль перед точкой, разделяющей целую и дробную части в числах по ту часть океана, не ставится. По-нашему, калибр патрона получается $0{,}30 \times 2{,}54 = 7{,}62$ мм, калибр пули же чуть больше – 7,82 мм. Понять, откуда разница, несложно. Калибром оружия принято считать диаметр канала его ствола. Если с гладкоствольным оружием вопросов по технологии измерения нет, то в нарезном диаметр может измеряться либо по дну нарезов, либо по полям, в зависимости от страны.

Вторая цифра в названии патрона (после чёрточки) появилась в тот момент, когда фирма Marlin Firearms начала выпускать крупные партии оружия под калибр .30. Её руководство принципиально не пожелало рекламировать фирму-конкурента, указывая её название «Winchester» в наименовании патрона, так появилось новое название со второй «тридцадкой», означавшей вес порохового заряда в гранах (1,94 грамма).

Что же до жёстких дисков, то первый из них был разработан ещё до появления персонального компьютера – 12 сентября 1956 года фирмой IBM[169]. Он состоял из 50 дисков (блинов) по 24″ каждый, имел объём в 5 МБ и стоил сумасшедших денег. Справа (от этого предложения, во врезке) вы можете увидеть фотографию (из [20]) сделанную во время его загрузки на борт самолёта. С того момента исследования по совершенствованию жёстких дисков продолжались и продолжаются по сей день. Наиболее интересной для истории является серия дисков с названием «3340», разработка которых началась в 1969-м, а 13 марта 1973 г. на суд общественности были представлены аж три модификации указанной серии. Указанные диски по праву можно считать жёсткими дисками ПК, так как они были разработаны для компьютеров IBM PC XT.

Далее мнения по возникновению названия «винчестер» расходятся, в [8, стр. 126] и ряде интернет-сайтов можно прочитать, что указанный диск имел 30 дорожек по 30 секторов в каждой. Оценим с точки зрения здравого смысла объём жёсткого диска с такими характеристиками. Для сравнения: обычная 1,44 МБ 3,5″ DS/HD-дискета имеет 80 дорожек по 18 секторов размера 512 байт на дорожку с обеих сторон дискеты. Размер секторов для жёстких дисков до 2009 года не менялся и составлял 512 байт. (Лишь с 2010–2011 гг. производители начали массово выводить на рынок диски с секторами в 4096 байт, который обычно называют размером 4 КБ и который теперь носит название Advanced Format (новый формат),

[168] До этого в «гражданском» оружии применялись исключительно патроны на дымном порохе, имевшие обычно калибр 10–12 мм, такие как .38-40 Winchester, .44-40 Winchester (11 мм), или патроны кольцевого воспламенения.

[169] http://www.research.ibm.com/research/gmr/history.htm.

присвоенное ассоциацией IDEMA (Международной ассоциацией производителей жестких дисков) [170].

Итак, если 30 дорожек по 30 секторов, в каждой дорожке по 512 байт, на 2 блинах с 2 сторон, получим: 30 × 30 × 512 × 2 × 2 = 1800 КБ – в целом неплохо для 1973 года, но нам кажется, что это мало, по сравнению с первым 5 МБ диском, пусть и «50-блинным». Так что указанная версия не очень правдоподобна.

Другая версия названия состоит в том, что при разработке 3340-й серии инженеры использовали краткое внутреннее название «30-30», означавшее два модуля (в максимальной компоновке) по 30 мегабайт каждый. Если предположить, что указанная на сайте IBM [171] информация немного преувеличена (там написано, что выпускались модификации в 35 и 70 МБ), то вполне вероятна ситуация, как указано на той же странице в сноске, что Кеннет Хотон, возглавлявший разработку указанной серии дисков, вполне мог оказаться автором фразы: «*Если 30-30, то это винчестер*», давшей указанному слову ещё один смысл.

Замечание. Приведём несколько интересных характеристик, чтобы читателям было над чем поразмышлять на досуге. Плотность записи первых винчестеров составляла 300 дорожек на дюйм (3,5″ дискета – 135). Зная объём дисков, попробуйте оценить их физический размер. Время доступа составляло 25 мс, а трансфер был в пределах 885 КБ/с. Попробуйте оценить скорость вращения вала диска. Полученный результат можете сравнить с дисководами. Скорость вращения дискеты 3,5″ – 300 об./мин (rpm [172]). Забавный факт: приводы дискет 5,25″, поддерживающих повышенную плотность записи (в пересчёте это 1,2 МБ), вращались быстрее – 360 оборотов в минуту. Основная масса современных жёстких дисков имеет скорость вращения 7200 rpm. Некоторые модели имеют заниженное число оборотов – 4200, 5200, 5400 и 5900, а большинство серверных версий – повышенное число 10000 и 15000 об./мин.

Развлечением программистов прошлого было написание программ управления индикатором дисковода и использование последнего как музыкального инструмента. Ролики с исполнением Гимна СССР и других мелодий без проблем можно найти на популярных видеосервисах. Стоит восхититься тем, насколько изобретательнее нас были наши родители, в сравнение, смеем предположить, что современная молодёжь, окружённая сабвуферами и звуком «5 вокруг» или «7 вокруг», вряд ли сможет без использования интерфейса операционной системы с мышкой каким-либо ещё программным способом отрегулировать громкость используемого ими «аудиохозяйства», не говоря уже о создании звуковых произведений.

Рисунок 4.50. Винчестер со снятой верхней крышкой (модель Samsung HD103UJ, 1000 ГБ, 7200 об./мин

4.11.1.2. Внутреннее устройство

Вернёмся от истории к текущему положению дел в отрасли. Современные винчестеры имеют ёмкость 500 Гбайт и более. Винчестер представляет собой плоскую, гер-

[170] http://www.seagate.com/ru/ru/tech-insights/advanced-format-4k-sector-hard-drives-master-ti/.
[171] http://www-03.ibm.com/ibm/history/exhibits/storage/storage_3340.html.
[172] RPM = *rotation per minute*, оборотов в минуту.

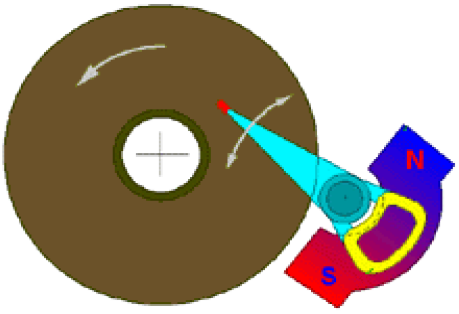
метически закрытую металлическую коробку, внутри которой на общей оси находятся несколько жёстких алюминиевых, стеклянных или кремниевых пластинок круглой формы (см. рис. 4.32).

Выпускаются винчестеры в основном двух размеров дисков (форм-фактор) – 3,5″ и 2,5″, реже 1,8″ (последние два преимущественно для использования в ноутбуках и нетбуках). Поверхность каждого диска покрыта тонким ферромагнитным слоем (веществом, способным длительное время сохранять состояние намагниченности). Для чтения и записи данных в винчестерах используются «плавающие» на воздушной подушке магнитные головки. Зазор между магнитным слоем и головкой составляет тысячные доли микрона (см. рис. 4.51).

Рисунок 4.51. Сравнение высоты «полёта» головки современного жёсткого диска с размерами отпечатка пальца, частичкой пыли и человеческим волосом [173]

Интересный факт. Воздушная подушка обычно рассчитывается для использования дисков при нормальном атмосферном давлении. Следует учитывать, что по мере возвышения над поверхностью земли атмосферное давление падает. Любители авиаперелётов и жители небоскрёбов могут расслабиться, потому как в самолётах давление поддерживается искусственно, а наличие даже 50-го этажа сказывается несущественно. А вот владельцам «реально» горных лабораторий не раз приходилось задумываться, почему их компьютеры и ноутбуки не загружаются. Для дисков с гермо-зоной (наполненной гелием), указанная проблема менее актуальна, а для твердотельных дисков и вовсе отсутствует.

В данный момент читателям будет в пору вспомнить и про левитатор Бернулли из школьного курса общей физики. Наглядно этот эффект можно увидеть самому и показать друзьям не только в различных музеях типа «Интеллектус» (г. Уфа) и «Экспериментариум» (гг. Москва, Белгород, Киев, Саратов), некоторых школьных физических кабинетах, где есть соответствующие экспонаты, торговых центрах и кинотеатрах с игровыми автоматами типа «аэрохоккей», но и у себя дома, поиграв немного с пылесосом.



Продолжим рассмотрение внутреннего устройства жёсткого диска. Для перемещения головок в первых моделях жёстких дисков использовался шаговый двигатель. В настоящее время для этой цели используются преимущественно линейные двигатели (типа voice coil или «звуковая катушка»), иначе называемые соленоидными.

Кроме того, при использовании соленоидных двигателей реализуется автоматическая парковка головок чтения-записи при отключении питания винчестера, чтобы, «перестав парить», они не царапали поверхность диска с данными при своём «приземле-

[173] Данные http://www.seagate.com/tech-insights/advanced-format-4k-sector-hard-drives-master-ti/.

нии». Для отслеживания местоположения головок применяется автоматическая система регулирования, которая по магнитным меткам на поверхности опорного диска позиционирует головки в нужном месте. (Так называемое заводское или низкоуровневое форматирование диска, недоступное пользователям без специальных программно-аппаратных комплексов, типа ставшего классикой у ремонтников – PC-3000.)

Пакет дисков вращается непрерывно с большой скоростью (7200 – 15 000 об./мин). Запись данных осуществляется следующим образом. При изменении силы тока, проходящего через записывающую головку, происходит изменение напряженности магнитного поля в щели между поверхностью и головкой, что приводит к изменению напряжённости магнитного поля на небольшом участке ферромагнитного покрытия диска.

Условно (для простоты понимания читателями) намагниченные участки и ненамагниченные соответствуют двоичным кодам «1» и «0». Для чтения данных используется магниторезистивный принцип, когда сопротивление полупроводниковой плёнки, из которой сделана головка чтения, меняется в зависимости от напряжённости магнитного поля на поверхности ферромагнитного диска. Электрический сигнал с головки усиливается и передаётся на обработку в контроллер жёсткого диска, расположенный в чипсете.

Вся электроника, предназначенная для управления двигателем привода дисков и электромагнитом головок, а также для чтения и записи данных, располагается на небольшой печатной плате, укреплённой снаружи корпуса.

Винчестер может иметь до десяти дисков (блинов) (ранее выпускались диски геометрических размеров 5,25″ и даже 10,5″ = 2×5,25″). На поверхности дисков размечаются траектории записи информации – концентрические окружности, которые называются дорожками, или треками (track), и секторы на каждой дорожке. Каждая дорожка имеет свой номер. Дорожки с одинаковыми номерами, расположенные одна над другой на разных дисках, образуют цилиндр. Каждый сектор на дорожке также имеет номер, нумерация начинается с единицы. Физически сектор на диске занимает 571 байт двоичной информации из которой для записи доступны лишь 512 байт, остальные байты отведены под низкоуровневую служебную информацию – заголовок (префикс), определяющий начало и номер секции, и окончание (суффикс), где записана контрольная сумма, нужная для проверки целостности хранимых данных. С 2009 года выпускаются диски, размеры секторов которых в 8 раз больше – 4096 байт (подробнее см. выше). Секторы и дорожки образуются во время низкоуровневого форматирования диска на заводе-изготовителе. Изначально число секторов на дорожку было постоянным, несмотря на то что физически ближе к краю диска дорожки длиннее. В новых винчестерах этот недостаток учтён и распределение числа секторов на дорожках неравномерное. Число секторов на внешних дорожках больше, чем на внутренних. Однако для пользователей это остаётся незамеченным. Для того чтобы они и программы, работающие с дисками, не сходили с ума от вычислений, где и сколько секторов добавить, в каких дисках нужен пересчёт, а в каких нет, сами контроллеры, встроенные в диск, занимаются этой черновой работой. Для пользователя и системе формат диска в головках, секторах и дорожках выдаётся «абстрактно», чтобы не надо было переписывать старые программы, в то время как внутри диска производятся постоянные пересчёты адресов «туда» и «обратно». В литературе это называется линейный способ адресации блоков LBA (Linear Block Addressing), когда все секторы нуме-

```
Размер сектора:
    512 байт
   4096 байт
```

руются последовательно от первого сектора нулевой дорожки нулевого цилиндра до последнего на крайней внутренней дорожке.

Исходя из этого, к современным винчестерам не применима операция низкоуровневого форматирования пользователем с помощью специальных программ (но возможна для гибких дисков и старых винчестеров).

Несколько секторов образуют ***кластер*** – наименьшую адресуемую единицу на диске. Большинство файловых систем для удобства оперируют кластерами. В ряде систем вместо кластера вводят понятие ***блок***а (обычно кратное целому числу секторов).

Форматирование диска в современном понимании – это есть создание ***таблицы разделов*** на нём (деление на логические диски) и создание ***файловой системы*** на одном или нескольких логических дисках.

Обычно эта операция выполняется с помощью специальной программы, входящей в состав операционной системы, и сулит потерю всех данных (иногда на стадии установки этой системы). Потерять данные с пустого диска – не страшно, так как там их ещё нет, а вот экспериментировать с форматированием имеющегося диска не стоит. Что такое файловая система и как она устроена подробнее, можно узнать из раздела 5.3.7. Пока же отметим, что для обычных пользователей неформатированный диск (то есть без таблицы разделов и файловой системы) не может быть использован для записи файлов.

До форматирования жёсткий диск можно разбить на логические диски (раздел винчестера или том). Это удобно, поскольку наличие нескольких логических дисков с некоторой стороны упрощает структуризацию данных, хранящихся на жёстком диске. По сути, разбиение есть выбор размеров разделов пользователем и запись этих значений в таблицу разделов, хранящуюся на том же диске (в первом секторе).

Существует огромное количество разных моделей жёстких дисков многих фирм, таких как Seagate, Maxtor, Western Digital, IBM, Fujitsu, Hitachi, Samsung, Toshiba и прочие. Для обеспечения совместимости винчестеров разработаны стандарты на их характеристики, определяющие номенклатуру соединительных проводников, их размещение в переходных разъёмах, электрические параметры сигналов.

4.11.1.3. Интерфейс жёсткого диска

Самый распространённый до недавнего времени тип винчестера по интерфейсу подключения был IDE (Integrated Drive Electronics). Винчестер типа IDE использует спецификацию параллельного интерфейса ATA (Advanced Technology Attachment). В настоящее время спецификация потеряла актуальность, так как новые чипсеты поддерживают новую спецификацию подключения HDD – последовательный интерфейс SerialATA и SerialATA II, а диски со старым интерфейсом практически не производятся. Соответственно, новые винчестеры имеют тип интерфейса SerialATA.

Особняком стоит **SCSI** (Small Computer System Interface – интерфейс малых компьютеров, читается «скази»). Существуют различные его модификации со словами Ultra, Wide и прочие, в том числе и последовательная SCSI – **SAS**. Так как этот интерфейс используется в основном в серверах и устройства SCSI довольно дороги, мы не будем их рассматривать.

ATA – один из старейших стандартов обмена данными, был разработан в 1989 году тремя компаниями: Imprimus – подразделением Control Data Corporation, Western

Digital и Compaq. Представляет собой параллельный шинный интерфейс. Первый стандарт ATA был утверждён в 1994 году национальным комитетом по стандартам информационных технологий (NCITS). ATA и его дальнейшие варианты, такие как ATA-2, ATA-3 и т. д., – официальные названия стандарта. IDE, EIDE, UltraATA и т. д. – маркетинговые термины, используемые производителями винчестеров. IDE-интерфейс в последней его модификации позволяет обмениваться данными со скоростью до 133 Мбайт/с, но на практике эта максимальная в теории скорость для конечных пользователей за счёт тех или иных нюансов оказывается раза в два меньше.

Маркетологи давно перевели взгляды покупателей на следующий шаг в развитии – замена параллельного интерфейса последовательным.

Serial ATA – последовательный шинный интерфейс для передачи данных, пришёл на замену ATA. Был разработан компанией Intel совместно с IBM, Dell, Seagate, Quantum, Maxtor, APT Technologies и др. в 2000 году. В первой ревизии интерфейса обеспечивал теоретическую пропускную способность до 1,5 Гбит/с. В ревизиях 2.0 и 3.0 максимальная теоретическая скорость передачи была повышена до 3 и 6 Гбит/с, соответственно.

Последняя ревизия SATA 3.1 учитывает уже некоторые особенности твердотельных дисков. (Подробнее см. раздел 4.7.4.4 «Как работает SSD-накопитель».)

Существуют две интересные модификации SATA Express и micro SSD:

SATA Express программно совместим с SATA, но в качестве несущего интерфейса используется PCI Express. Конструктивно представляет собой два рядом расположенных в длину SATA-порта, что позволяет использовать как накопители с интерфейсом SATA, так и непосредственно накопители, изначально поддерживающие SATA Express. Скорость передачи данных при этом достигает 8 Гбит/с в случае использования одного разъёма и 16 Гбит/с в случае, если задействованы оба разъёма SATA Express.

µSSD (micro SSD) – представляет из себя BGA-интерфейс для подключения миниатюрных, встроенных накопителей.

4.11.1.4. Параметры, влияющие на быстродействие

Среди других параметров, которые влияют на быстродействие жёсткого диска, следует отметить следующие:

- скорость вращения дисков – в наше время обычно 7200 об./мин, накопители SCSI (SAS) 10000–15000 об/мин;
- объём кэш-памяти – во всех современных дисковых накопителях устанавливается кэш-буфер, ускоряющий обмен данными; чем больше его ёмкость, тем выше вероятность того, что в кэш-памяти будет необходимая информация, которую не надо считывать с диска (этот процесс в тысячи раз медленней); ёмкость кэш-буфера в современных винчестерах 16–128 Мбайт (на пути увеличения кэша можно встретить гибридные диски, где вместо привычной оперативной кэш-памяти с той же целью используется память, организованная как хранение в SSD-дисках);
- среднее время доступа – время (в миллисекундах), на протяжении которого блок головок механически смещается с одного цилиндра на другой. У современных винчестеров составляет 5–10 миллисекунд;
- время задержки – это время от момента позиционирования блока головок на нуж-

ный цилиндр до позиционирования конкретной головки на конкретный сектор, другими словами, это время поиска нужного сектора (около 4 мс);

- скорость обмена – определяет объём данных, которые могут быть переданы из накопителя к микропроцессору и в обратном направлении за определённые промежутки времени; максимальное значение этого параметра равно пропускной способности дискового интерфейса и зависит от того, какой режим используется: PIO (Programmed I/O) или DMA; в режиме PIO обмен данными между диском и контроллером происходит при непосредственном участии центрального процессора, чем больше номер режима PIO, тем выше скорость обмена; работа в режиме DMA (Direct Memory Access) разрешает передавать данные непосредственно в оперативную память без участия процессора; скорость передачи данных в современных жёстких дисках до 300 Мбайт/с, при этом сохраняется тенденция к росту;
- место записи данных на диск – у края диска линейная скорость выше и, следовательно, скорость чтения-записи выше.

4.11.1.5. S.M.A.R.T., надёжность, RAID

Несмотря на внешнюю простоту и наличие всего двух разъёмов – информационного и питания, винчестеры являются очень сложными устройствами, которые требуют бережного и правильного обращения – следует избегать тряски и резкого перемещения работающего винчестера [174], это может привести к разрушению внутренних механических узлов и повреждению магнитных головок и поверхности дисков.

Современные винчестеры могут безотказно работать до миллиона часов, а у некоторых моделей и все два с половиной. Указанный параметр важен, но не всегда учитывается продавцами и пользователями. Его можно найти как MTBF [175] в описаниях к дискам, доступных на сайтах производителей.

Время наработки на отказ современных дисков высоко отчасти и потому, что они используют систему внутренней автодиагностики, именуемой как S.M.A.R.T. (Self-Monitoring, Analysis and Reporting Technology) вкупе с микропрограммой, устраняющей возможные недостатки дисковых накопителей незаметно для пользователей (разве что пользователь будет заглядывать в статистику по диску каждую минуту). Например, внутренняя программа диска, работающая по технологии S.M.A.R.T. (даже если соответствующая поддержка отключена в BIOS), может заменять на лету прозрачно для пользователя плохие кластеры на резервные. S.M.A.R.T. микропрограмма накопителя отвечает за:

- выдачу набора атрибутов, отражающих состояние отдельных параметров накопителя;
- внутренние тесты накопителя;
- ведение журналов и работу с ними.

[174] Читателям впору вспомнить из курса физики про гироскопы, прецессию, нутацию и силу Кориолиса. Если резко дёрнуть работающий жёсткий диск, находящийся в салазках, можно запросто если не сломать ось диска, то внести существенные искажения в геометрию вращения или испортить двигатель.

[175] Mean Time Between Failure – «Время между сбоями» или более привычное по ГОСТ 27.002–89 «наработка на отказ».

К сожалению, производители не публикуют о своих накопителях полные характеристики и поддерживаемые функции S.M.A.R.T., а будучи однажды определёнными опытным путём, их структура и назначение могут измениться, например при переходе к накопителю другой фирмы или даже по выходу более новой версии той же модели.

Из наиболее общих моментов следует отметить, что журналов 255, каждый журнал имеет свой порядковый номер, номера поделены на диапазоны, образуя несколько категорий: каталог журналов, журналы ошибок и их расширения, журнал самотестирования и его расширение, пользовательские журналы, журналы изготовителя накопителя и несколько зарезервированных. Журналы можно считывать и в них можно записывать. Со стороны операционной системы компьютера запись возможна не во все журналы. Использовать их для собственных нужд не предусмотрено, хотя в стандарте ATA/ATAPI-7, предназначение пользовательских журналов определяется как «to store any data desired».

По сути, многие графические утилиты, выводящие данные S.M.A.R.T., не делают ничего особенного, кроме как перерисовывают полученные выше данные журналов в цветные таблицы и диаграммы, радующие глаз пользователя. Информацию о назначении того или иного параметра и критичности его значения можно без туда найти в интернете. Потратьте полчаса на изучение вопроса, и вам не смогут продать битый или испорченный диск под видом нового.

В настоящее время на винчестеры производителями устанавливается 1–5-летний гарантийный срок. Если вам заведомо не попалась бракованная серия, то при соблюдении условий эксплуатации (температурный режим, влажность, стабильное электропитание, отсутствие постоянных ускорений) вероятность поломки накопителя мала. Если вы заметили, что на вашем винчестере появляются сбойные (плохие секторы) и их количество постоянно увеличивается, то замену такого винчестера на новый лучше не откладывать.

Посмотреть параметры вашего жёсткого диска и некоторые настройки контроллера можно, запустив в ОС Linux с правами администратора следующую команду:

```
# hdparm /dev/sda
/dev/sda:
 multcount     = 16 (on)
 IO_support    =  1 (32-bit)
 readonly      =  0 (off)
 readahead     = 256 (on)
 geometry      = 19457/255/63, sectors = 312581808, start = 0
```

, где /dev/sda – блочное устройство, отвечающее за ваш жёсткий диск [176]. Указанная команда имеет возможность отображения (и изменения) многих параметров, подробнее см. «`man hdparm`». Например, на практике часто используется проверка скорости чтения с жёсткого диска (ключи -t и -T, без учёта кэша и с учётом соответственно):

```
# hdparm -t -T /dev/sda
/dev/sda:
 Timing cached reads:   11436 MB in  2.00 seconds = 5724.26 MB/sec
 Timing buffered disk reads:  414 MB in  3.01 seconds = 137.38 MB/sec
```

Данные S.M.A.R.T можно увидеть, запустив из консоли команду:

[176] Второй диск, скорее всего, будет /dev/sdb, старые ATA-диски и CD/DVD-приводы доступны как /dev/hda, /dev/hdb, /dev/hdc и т. д. Для ОС Linux (а также многих из семейства UNIX и Android) это стандартные названия, значения которых станет ясно после прочтения главы 5 «Программное обеспечение».

```
# smartctl -a /dev/sda
smartctl 5.40 2010-10-16 r3189 [x86_64-redhat-linux-gnu] (local build)
Copyright (C) 2002-10 by Bruce Allen, http://smartmontools.sourceforge.net
===== START OF INFORMATION SECTION === ==
Model Family:     Western Digital Caviar Blue Serial ATA family
Device Model:     WDC WD1600AAJS-00YZCA0
Serial Number:    WD-WCAYV10743337
Firmware Version: 01.03B01
User Capacity:    160 041 885 696 bytes
Device is:        In smartctl database [for details use: -P show]
ATA Version is:   8
ATA Standard is:  Exact ATA specification draft version not indicated
SMART support is: Available - device has SMART capability.
SMART support is: Enabled
```

. . . (часть вывода отсутствует)

```
SMART capabilities:  (0x0003) Saves SMART data before entering
                              power-saving mode.
                              Supports SMART auto save timer.
Error logging capability:   (0x01)  Error logging supported.
                                    General Purpose Logging supported.
Short self-test routine recommended polling time:        (   2) minutes.
Extended self-test routine recommended polling time:     (  29) minutes.
Conveyance self-test routine recommended polling time:   (   5) minutes.
SCT capabilities:     (0x3037)   SCT Status supported.
                                 SCT Feature Control supported.
                                 SCT Data Table supported.
SMART Attributes Data Structure revision number: 16
Vendor Specific SMART Attributes with Thresholds:
ID# ATTRIBUTE_NAME          FLAG  VALUE WORST THRESH TYPE  UPDATED
                                                        WHEN_FAILED RAW_VALUE
  1 Raw_Read_Error_Rate     0x002f  200  200  051  Pre-fail    Always  -     0
  3 Spin_Up_Time            0x0027  142  141  021  Pre-fail    Always  -     3866
  4 Start_Stop_Count        0x0032  100  100  000  Old_age     Always  -     204
  5 Reallocated_Sector_Ct   0x0033  200  200  140  Pre-fail    Always  -     0
  7 Seek_Error_Rate         0x002e  200  200  000  Old_age     Always  -     0
  9 Power_On_Hours          0x0032  099  099  000  Old_age     Always  -     866
 10 Spin_Retry_Count        0x0032  100  100  000  Old_age     Always  -     0
 11 Calibration_Retry_Count 0x0032  100  100  000  Old_age     Always  -     0
 12 Power_Cycle_Count       0x0032  100  100  000  Old_age     Always  -     195
192 Power-Off_Retract_Count 0x0032  200  200  000  Old_age     Always  -     21
193 Load_Cycle_Count        0x0032  200  200  000  Old_age     Always  -     182
194 Temperature_Celsius     0x0022  107  102  000  Old_age     Always  -     36
196 Reallocated_Event_Count 0x0032  200  200  000  Old_age     Always  -     0
197 Current_Pending_Sector  0x0032  200  200  000  Old_age     Always  -     0
198 Offline_Uncorrectable   0x0030  200  200  000  Old_age     Offline -     0
199 UDMA_CRC_Error_Count    0x0032  200  200  000  Old_age     Always  -     0
200 Multi_Zone_Error_Rate   0x0008  200  200  000  Old_age     Offline -     0
```

…

Замечание. Из вывода выше видно, что с помощью утилиты smartctl можно определить серийный номер диска, также это можно сделать с помощью утилит smartctl, hdparm и команды ls:

```
# smartctl -a /dev/sda|grep Serial
Model Family:     Western Digital Caviar Blue Serial ATA
Serial Number:    WD-WCAYV10743337
# hdparm -i /dev/sda|grep Serial
 Model=WDC WD1600AAJS-00YZCA0, FwRev=01.03B01, SerialNo=WD-WCAYV10743337
$ ls -l /dev/disk/by-id/*
lrwxrwxrwx 1 root root  9 окт 25 21:42 /dev/disk/by-id/ata-WD1600AAJS-
00YZCA0_WD-WCAYV10743337 -> ../../sda
lrwxrwxrwx 1 root root 10 окт 25 21:42 /dev/disk/by-id/ata-WD1600AAJS-
```

```
00YZCA0_WD-WCAYV10743337-part1 -> ../../sda1
lrwxrwxrwx 1 root root 10 окт 25 21:42 /dev/disk/by-id/ata-WD1600AAJS-
00YZCA0_WD-WCAYV10743337-part2 -> ../../sda2
...
```

Для повышения значений параметров надёжности работы, скорости чтения, скорости записи жёстких дисков (по одному или в различных комбинациях) диски могут объединяться в так называемые RAID[177] массивы. Поскольку одновременная работа с несколькими дисками будет отличаться от работы с одним диском, за управление последними обычно отвечает аппаратный контроллер, делающий работу с RAID-массивом прозрачной. Хорошие контроллеры могут стоить как сам жёсткий диск. С целью экономии, но в ущерб производительности, поддержка RAID-массива может быть программной (на уровне ядра или драйвера операционной системы), либо урезанно-аппаратной, за счёт BIOS. В зависимости от того, какие из двух параметров важнее: **надёжность**, **экономичность** или **скорость**, выбирают тип массива и как следствие число дисков входящих в него. Классическими являются RAID 0, RAID 1, RAID 2, RAID 3, RAID 4, RAID 5, RAID 6. Другие типы являются их модификациями и комбинациями.

4.11.1.6. Стоимость

Например, винчестеры фирм Western Digital (WD), Seagate и Hitachi объёмом 4000 ГБ состоят из 4 дисков (блинов) с плотностью записи 2×500 ГБ на один. Для меньших объёмов используются как пластины с меньшей плотностью записи, так и меньшее число самих пластин. На самых маленьких дисках может быть не только один блин, но и одна головка, то есть запись производится только с одной стороны диска.

Многие компании, такие как Seagate, Maxtor и WD, выпускающие диски класса 7200/3.5″, выпускают также **внешние** жёсткие диски, выполненные в отдельном корпусе с блоком питания, интерфейсом USB или IEEE1394 (FireWire).

Некоторые модели выпускаемых в настоящее время винчестеров показаны в табл. 4.12. Учитывая скорость вращения диска, его геометрические размеры и цену хранения единицы информации вы без труда найдёте оптимальный для себя вариант.

Таблица 4.12. Сравнение средней стоимости винчестеров (январь 2021 г.)

Марка жёсткого диска (интерфейс SATA-III)	Объём диска, Гб	Цена, руб.	Цена за 1 Гб, руб.
Toshiba MG08ACA16TE (7200 rpm 512 МБ)	16000	33000	2,06
Western Digital Ultrastar DC HC530 WUH721414ALE6L4 (7200rpm 512 МБ)	14000	31000	2,21
Toshiba N300 HDWG21EUZSVA (7200 rpm 256 МБ)	14000	37000	2,64
Toshiba X300 HDWR21EEZSTA (7200 rpm 256 МБ)	14000	29000	2,07
Western Digital Gold WD121KRYZ (7200 rpm 256 МБ)	12000	33000	2,75
Toshiba MG07ACA12TE (7200 rpm 256 МБ)	12000	24200	2,02
Toshiba X300 HDWR11AEZSTA (7200 rpm 128 МБ)	10000	20500	2,05

[177] RAID (*англ.* redundant array of independent disks – избыточный массив независимых дисков) – технология виртуализации данных, которая объединяет несколько дисков в логический элемент для избыточности и/или повышения производительности.

Western Digital Purple WD102PURZ (7200 rpm 256 МБ)	10000	28000	2,8
Western Digital Purple WD82PURZ (7200 rpm 256 МБ)	8000	20500	2,56
Seagate SkyHawk ST8000VX004 (7200 rpm 256 МБ)	8000	20500	2,56
Western Digital Purple WD60PURZ (5400 rpm 64 МБ)	6000	15000	2,5
Western Digital Black WD6003FZBX (7200 rpm 256 МБ)	6000	21000	3,5
Toshiba MG04ACA400E (7200 rpm 128 МБ)	4000	11500	2,88
Seagate IronWolf Pro ST4000NE001 (7200 rpm 128 МБ)	4000	14500	3,63

4.11.2. Внешние диски, DAS, СХД, SAN, NAS, iSCSI

Внимательный читатель заметит, что в последней сравнительной таблице (4.18) по ценам жёстких дисков мы привели для сравнения и пару внешних жёстких дисков. С одной стороны, суть работы и внутреннее устройство диска не зависят от того, где будет он использоваться: внутри системного блока компьютера или вне его. С другой стороны, разница должна быть.

Если отойти от параметров надёжности и стоимости, то для внутреннего диска большинству покупателей важны скорость (зависит от скорости вращения, объёма кэша, интерфейса, плотности записи, числа блинов) и объём. Для внешних дисков предпочитаемые параметры могут несколько измениться. Ситуация «1-в-1» со ставшим уже классикой в маркетинге случаем «Быстро, качественно, недорого – выберите два любых параметра».

Если важна надёжность – вырастет цена. Если важна цена – упадёт скорость. Если важны надёжность и скорость – вырастет цена. Если важна цена – не ждите скорости и надёжности.

Так, для покупателя может быть критично, что ему носить ежедневно в кармане 2,5″ или 3,5″ диски. Если же диск будет лежать дома, то геометрический размер не важен, как и повышенная устойчивость к сотрясениям.

Для каждого типа случая «внешнего подключения» рынок придумал что-то своё. Опишем коротко используемые для этого названия и технологии, чтобы читатель мог разбираться в современных терминах и технологиях.

DAS (от англ. Direct-attached storage – система хранения данных с прямым подключением) – запоминающее устройство, непосредственно подключенное к ПК или серверу без помощи сети хранения данных. Это ретроним, используемый в основном для отличия несетевых устройств хранения от SAN и NAS. Бывают как автономные, небольших размеров с питанием от разъёма подключения (например, USB, те самые коробочки, носимые пользователями), так и, исполнением размером с системный блок со своим отдельным подключением к сети 220 В. Внутри DAS-хранилища часто используется не один жёсткий диск, а несколько, так называемый массив жёстких дисков. В какой-то мере внутренний жёсткий диск компьютера – это тоже DAS, но так не говорят.

СХД, SAN (Сеть хранения данных, от англ. Storage Area Network) представляет собой архитектурное решение для подключения внешних устройств хранения данных, таких как дисковые массивы, ленточные библиотеки, оптические приводы к серверам таким образом, чтобы операционная система распознала подключённые ресурсы как локальные. (Доступ к данным SAN-устройств происходит поблочно, и, как видим, под-

ключаться могут не только жёсткие диски.) Несмотря на то что стоимость и сложность таких систем постоянно падают, по состоянию на 2015 год сети хранения данных остаются редкостью за пределами больших предприятий.

NAS (англ. Network Attached Storage) – сетевая система хранения данных, сетевое хранилище. Доступ к данным NAS-устройств происходит по сетевым протоколам на уровне файлов. По сути, NAS представляет собой ЭВМ [178] с некоторым дисковым массивом, подключенную к сети (обычно локальной) и поддерживающую работу по принятым в ней протоколам. Часто диски в NAS объединены в RAID-массив. Несколько таких компьютеров могут быть объединены в одну систему. Обеспечивает надёжность хранения данных, лёгкость доступа для многих пользователей, лёгкость администрирования, масштабируемость.

Различные сетевые маршрутизаторы и точки доступа домашнего использования со встроенными дисками по всем формальным признакам есть NAS, так как указанные устройства научились поддерживать и жёсткие диски, и сетевые файловые системы/протоколы. (Вплоть до протоколов торрент-клиентов.)

Пожалуй, единственное отличие промышленных и корпоративных NAS от домашних – это узкая направленность, нацеленная на повышенную надёжность хранения данных и скорость общения с носителем, предположительно в ущерб физическим размерам и высокой стоимости.

Принятое **отличие NAS и SAN**, состоящее в том, что «SAN – это только сетевые диски (доступ к блокам устройства), в то время как NAS – это только сетевая файловая система (доступ к файлам)», является скорее искусственным. С появлением iSCSI началось взаимное проникновение технологий с целью повышения гибкости и удобства их применения.

iSCSI (от англ. Internet Small Computer System Interface) – протокол, который базируется на TCP/IP и разработан для установления взаимодействия и управления системами хранения данных, серверами и клиентами. Буквы «SCSI» в названии появились от того, что по задумке данная технология позволяла получать доступ к более надёжным и быстрым (и дорогим) SCSI-дискам удалённо (по Сети).

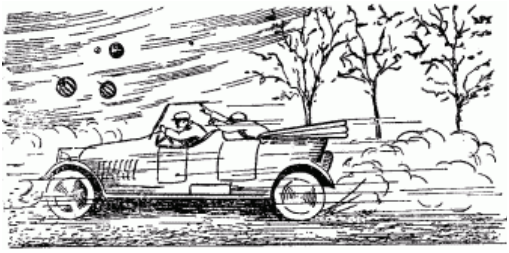
Про iSCSI должны знать сетевые администраторы (не путать с начинающими «anykey-шиками»), из которых использовать её (или его) будут единицы, обычным же пользователям эти знания будут, скорее, бесполезными. Использование iSCSI в домашних условиях равносильно поднятию вашей кровати 20-тонным домкратом, чтобы под ней было поудобнее пропылесосить или протереть пол влажной тряпкой.

4.11.3. Пути улучшения характеристик жёстких дисков

Скорость вращения шпинделя диска не может расти бесконечно. При скорости вращения в 15 000 оборотов в минуту на краю пластины начинают сказываться аэродинамические эффекты, воздух «цепляется за пластины», возникает эффект турбулентности. Попробуйте плавно выставить руку в окно автомобиля, движущегося на большой скорости – сразу почувствуете давление воздуха и эффект, сравнимый с тем, что будут

[178] Превратить в NAS любой ПК по силам даже вам, читатель, посетите сайт проекта FreeNAS – http://www.freenas.org/, и, возможно, вы найдёте применение вашему старому компьютеру, докупив к нему новый вместительный жёсткий диск.

испытывать головки на краю диска. А попробуйте выставить в руке небольшой листок бумаги, полагаем, он забавно захлопает. Сравните:



колёса автомобиля [179], двигающегося со скоростью 100 км/ч, будут вращаться со скоростью 700 оборотов в минуту против скоростей 7200 и 15 000 у современных жёстких дисков.

100 км/ч = 100000 м/ч = 100000 м/60 мин =
= 10000 м/6 мин ≈ 1666,7 м/мин

Колесо $R = 15''$ за один оборот проедет $l = 2\pi R$ метров.
$l \approx 2 \times 3{,}14 \times 15 \times 2{,}54 / 100 \approx 2{,}39$ м. Чтобы проехать 1666,7 м,
за минуту колесо должно обернуться 1666,7 / 2,39 ≈ 697 раз.

То есть если бы хватило мощности и колёса автомобиля приводились бы в движение именно двигателями от жёстких дисков, наподобие тех, что стоят в большинстве современных ПК, а не двигателями внутреннего сгорания, то предполагаемый автомобиль двигался бы со скоростью более 1000 км/ч! А его серверный 15-тысячный собрат давно бы перешёл через звуковой барьер и двигался бы на скорости, приближающейся к 2 Махам [180]. Что там, снять глушитель с автомобиля и поставить турбокит? Перейти звуковой барьер! Вот бы прибавилось веселья у автолюбителей, достаточно вспомнить, как громко хлопает кнут, маленький кончик которого движется быстрее скорости звука.

Шумность работы диска зависит от того, как он был сделан, и чаще всего этот параметр не поддаётся регулировке. Однако некоторые модели могут позволять выбирать режим более тихой работы в ущерб производительности, и наоборот.

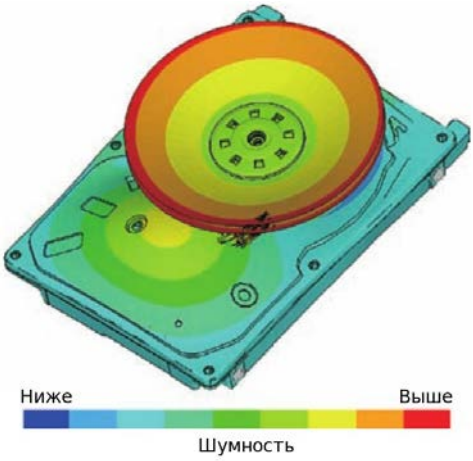


Рисунок 4.52. Распределение вибраций (шумности)

По данным TOSHIBA, распределение вибраций [181] блока головок от центра к краю диска выглядит следующим образом (см. рис. 4.52).

Температура. Чтобы жёсткий диск не перегревался, его важно охлаждать. В большинстве случаев ему хватает вентиляции, продуманной производителем корпуса, за счёт вентилятора блок питания охлаждается. Однако в серверах и в перспективе вам, читатель, лучше уделить этому вопросу чуть больше внимания.

[179] Идея с автомобилем, как и его изображение, позаимствованы из [12, стр. 38]. Мы не берём в расчёт мощность, аэродинамику, вопросы сцепления с дорогой и изнашивания покрышек.
[180] Упрощённо число Маха – это истинная скорость в потоке (то есть скорость, с которой воздух обтекает, например, самолёт), делённая на скорость звука в конкретной среде, поэтому зависимость является обратно пропорциональной. При давлении в 1 атм (у земли на уровне моря) скорость, соответствующая 1 Маху, будет равна приблизительно 330 м/с или 1100 км/ч, то есть скорости звука в воздухе.
[181] http://storage.toshiba.eu/export/sites/toshiba-sdd/whitepapers_casestudies/ToshibaSDDTechnicalPaper_MQ01ABD100.pdf.

Плотность записи увеличивать бесконечно тоже не получается. Выше мы коротко пояснили, как ведётся запись. При подаче переменного электрического тока (при записи) на катушку головки возникающее переменное магнитное поле из зазора головки воздействует на ферромагнетик поверхности диска и изменяет направление вектора намагниченности доменов в зависимости от величины сигнала. При считывании перемещение доменов у зазора головки приводит к изменению магнитного потока в магнитопроводе головки, что приводит к возникновению переменного электрического сигнала в катушке из-за эффекта электромагнитной индукции.

Вспомним из курса физики:

$$\varepsilon_i = -\frac{\Delta \Phi}{\Delta t}$$

В последнее время для считывания применяют магниторезистивный эффект и используют в дисках магниторезистивные головки. В них изменение магнитного поля приводит к изменению сопротивления, в зависимости от изменения напряжённости магнитного поля. Подобные головки позволяют увеличить вероятность достоверности считывания информации (особенно при больших плотностях записи информации).

Намагничивание, означающее «запись», может производиться двумя способами (направлениями): продольно и перпендикулярно. Изначально использовался продольный метод записи, при котором биты информации записываются с помощью маленькой головки, которая, проходя над поверхностью вращающегося диска, намагничивает миллиарды горизонтальных дискретных областей – доменов, «продольно», то есть вектор намагниченности домена оказывается расположен параллельно поверхности диска. Максимально достижимая при этом плотность записи составляет около 23 Гбит/см².

Примерно с 2005 года начал развиваться другой метод записи – «перпендикулярный» (см. рис. 4.53), в 2010 году он практически полностью вытеснил метод параллельной записи.

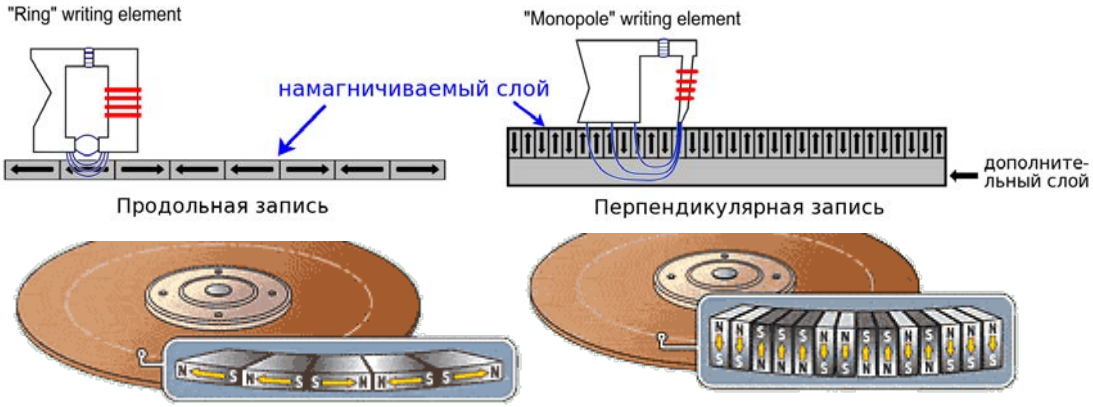


Рисунок 4.53. Механизмы продольной и поперечной записи

Вместо записывающей головки типа «кольцо» в нём используется тип «монополь». Как видно из рисунка, намагничиваемый слой при перпендикулярной записи толще, также под ним располагается дополнительный не рабочий слой. Метод перпендикулярной записи – это технология, при которой биты информации сохраняются в «вертикальных» доменах. То есть вектор намагниченности домена располагается перпендикулярно поверхности диска. Это позволяет использовать более сильные магнитные поля и уменьшить площадь диска, необходимую для записи 1 бита.

Таблица 4.13. Сравнение плотностей записи

Год	Плотность, Гбит/дюйм^2	Плотность, Гбит/см^2	Пластина от 3,5″ диска
2009	400	62	2×320 Гб
2011–2013	625	97	2×500 Гб
2014	1000	155	2×800 Гб
2015-2019	1300	202	2×1042 Гб
2020	2381[182]	370	2×1905 Гб

Материал пластины – не менее важный параметр, влияющий на запись. При повышении плотности записи начинают сказываться физические свойства материала, на который наносится ферромагнетик. Сравните неровности по фотографии, сделанной специалистами фирмы IBM: алюминиевая слева и стеклянная пластина справа [183] на рис. 4.54.

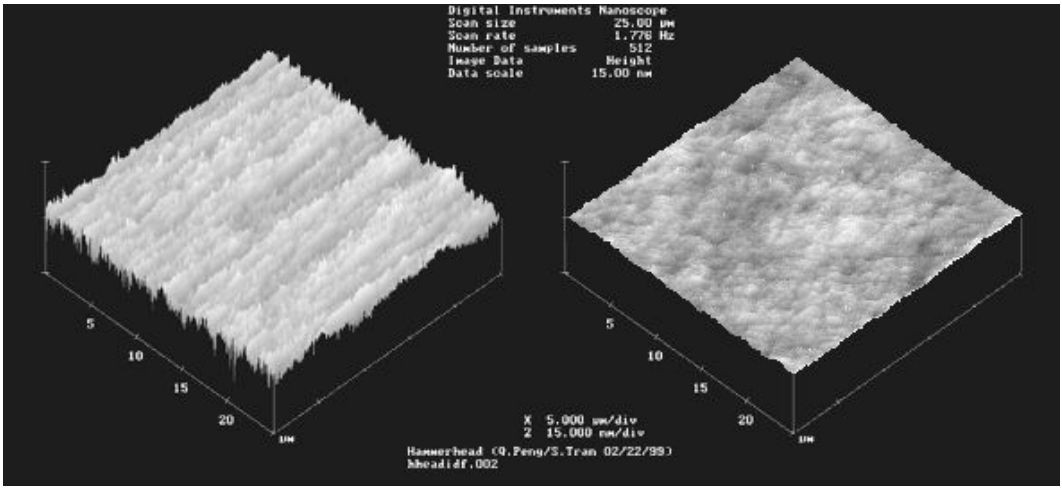


Рисунок 4.54. Неровности пластин под электронным микроскопом (алюминиевая – слева; стеклянная, более ровная – справа)

Термомагнитная запись. Наиболее вероятно, что будущее придётся на *термоассистируемую магнитную запись* (термомагнитная запись, тепловая магнитная запись, магнитная запись с подогревом, HAMR, от англ. Heat-assisted magnetic recording) – гибридную технологию записи информации, комбинирующую магнитное чтение и магнитооптическую запись. (см. рис. 4.55)

На данный момент это самый перспективный из существующих способов увеличения плотности записи, активно разрабатывается компаниями. При использовании этого метода применяется точечный подогрев диска лазером, который позволяет головке намагничивать очень мелкие области его поверхности. После того как диск охлаждается, намагниченность «закрепляется».

На 2009 год были доступны только экспериментальные образцы, плотность записи которых составляла 150 Гбит/см². Специалисты Hitachi называют предел для этой технологии в 2,3−3,1 Тбит/см², представители Seagate Technology – 7,75 Тбит/см².

[182] См. https://3dnews.ru/1000693/itogi-2019-goda-gestkie-diski.
[183] См. http://www.pcguide.com/ref/hdd/op/mediaMaterials-c.html.

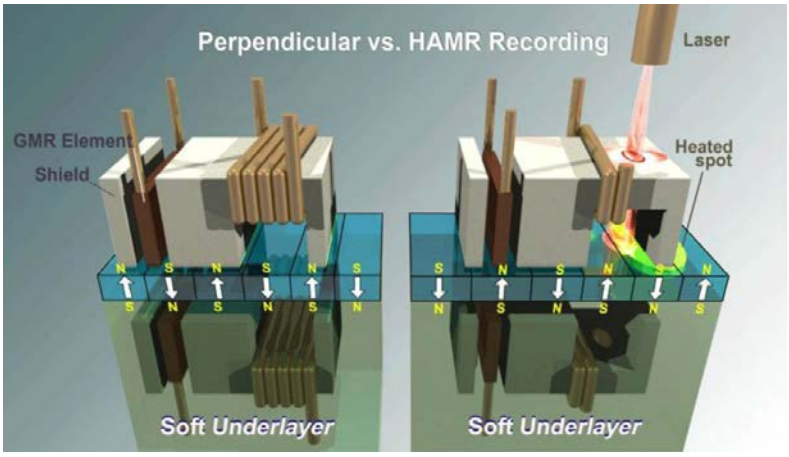


Рисунок 4.55. Сравнение обычной перпендикулярной записи слева и термомагнитной справа [184]

Герметичные диски заполненные гелием. Фирма HGST (Western Digital) в 2013 году начала выпуск жёстких дисков, где вращение дисков и «полёт» блока головок над дисками происходит в инертной среде. Гермозона диска в полном смысле этого слова теперь является герметичной и заполнена гелием (He) вместо воздуха. Поскольку вращение дисков в новой среде происходит легче, то это позволяет снизить нагрузку на мотор и тем самым, либо понизить потребление, либо позволяет использовать большее число блинов насаженных одновременно на один шпиндель. К тому же, вращаясь в гелии, диск меньше нагревается и меньше шумит. Используя эту технологию удалось выпустить на рынок модели объёмом 6, 8, 10, 12, 14, 16 и 18 ТБ. В некоторых моделях диски имеют до 9 пластин.

Не следует думать, что использование гелия обязательно для высокой плотности записи, фирма Seagate выпускает 6, 8 и 10 ТБ модели заполненные обычным воздухом.

4.11.4. Твердотельные накопители (SSD)

Чем-то схоже с зарядом конденсатора, сначала прирост напряжения идёт быстро, а потом он замедляется, идёт развитие винчестеров. За десятилетия такие характеристики (как скорость вращения, плотность записи, высота полёта головки, размер кэша) выросли на порядки и не могут далее улучшаться (расти или уменьшаться) с той же скоростью.

В предыдущем параграфе мы увидели, что старые пути повышения характеристик жёстких дисков сегодня подошли к «потолку теоретико-механических возможностей» и требуют изменения технологий хранения данных, то есть придумывания новых принципов. Объединение нескольких жёстких дисков в различно организованные массивы нельзя считать новшеством в той мере, что ни два, ни четыре диска не дадут прироста скорости на порядок.

Нужно новое решение, кардинальное! Именно таким решением и стали *твердоте́льные накопители* [185] – компьютерные немеханические запоминающие устройства

[184] http://diit.cz/clanek/seagate-dosahl-hustoty-zaznamu-1-tbitpalec2.
[185] англ. SSD, *solid-state drive*

на основе микросхем памяти. (Обычно к SSD-накопителю также относят и микросхемы контроллера памяти.) Различают *два вида твердотельных накопителей*:

SSD на основе памяти, подобной оперативной памяти компьютеров,	и	SSD на основе флэш-памяти.

Оперативная память была рассмотрена выше и требует постоянного электропитания (диск получается быстрый, но не долговременный, если электропитание отключить), чтобы понять, как работает флэш-память, не зависимая от электричества, кратко рассмотрим историю её появления.

4.11.4.1. Прародитель Flash – ПЗУ, история возникновения

В противовес ОЗУ существовало ПЗУ – постоянное запоминающее устройство (оно же ROM, *read only memory* – память только для чтения). Рассмотрим на пальцах работу ПЗУ. В первом приближении для организации такой памяти ничего сложного не надо, простые перемычки – и устройство готово.

При подаче напряжения на вход, например логической «1», на выходах, соединённых со входом, появится тоже «1». На неподключенных выходах (болтающихся в воздухе) будет «0».

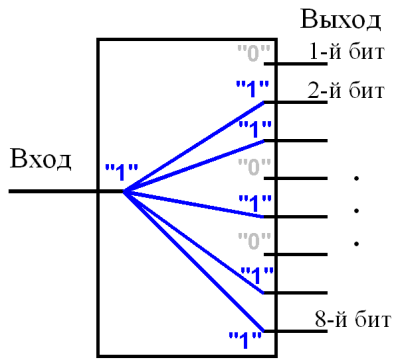


Рисунок 4.56. Внутреннее устройство одной ячейки памяти на 8 бит. Записанное число (01101011)

Если надо запомнить два байта, берём два таких устройства, если нужно запомнить *n* байт, берём *n* таких устройств. Ситуация – как с чёрными ящиками в разделе 4.5. Легко заметить, что подобная линейная организация неудобна и избыточна, поэтому на практике используется 2-мерная реализация, когда для снятия данных используются одни и те же выводы. Потеря пропускной способности (нельзя параллельно считывать все ячейки) компенсируется значительным упрощением схемы.

Для того чтобы сигналы с разных ячеек «не мешали» друг другу, используются диоды.

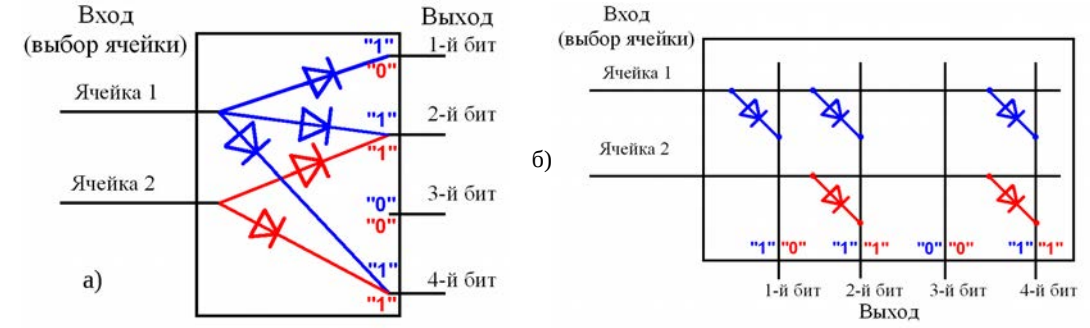


Рисунок 4.57. Внутреннее устройство 2 ячеек памяти (4 + 4 бита), записано: 1101 и 0101; а) неудобный способ изображения; б) удобный способ изображения

Использовать 1024 вывода для выбора нужной ячейки также не разумно. На практике для выбора нужной ячейки используется несложный декодер, встроенный в то же ПЗУ, преобразующий двоичный параллельный сигнал в выбор нужной линии (ячейки) ПЗУ. Дорисовав последовательно с каждым диодом плавкие предохранители, целые, если диод есть, и сгоревшие предохранители вместе с диодами, если их в схеме нет, мы получим схему, более-менее приближающуюся к реальности и встречающуюся в литературе.

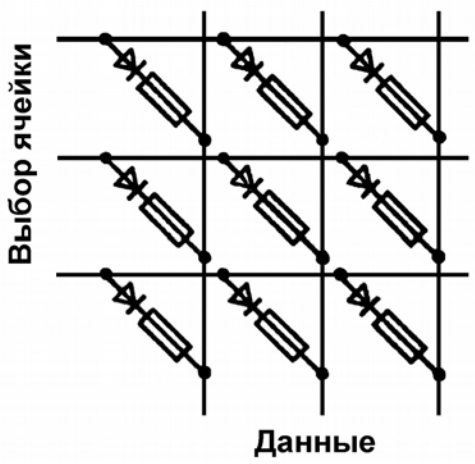


Рисунок 4.58. Схема ППЗУ, построенная на диодах

На практике организация записи в ПЗУ происходит по-разному. Если тираж очень большой, то дешевле получается штамповать уже зашитые ПЗУ как обычные алюминиевые CD/DVD-диски. Если тираж меньше и заранее неизвестно, что будет прошито, то дешевле обходится производить матрицы с диодами и предохранителями, после чего прожигать предохранители у ненужных ячеек, либо на заводе-изготовителе, либо продавать непрошитые микросхемы. В данные микросхемы возможно осуществить одноразовую запись. Что-то вроде обычного CD/DVD-R получается, правда, аналога мультисессионности (с дозаписью) тут пока не придумали. По аналогии с пишущими приводами для записи CD/DVD существуют программаторы, осуществляющие запись.

Если на диоды, используемые при выборе какой-то ячейки памяти, посмотреть внимательно, то можно заметить, что у всех них используется общий анод (или катод, если модифицировать схему). То же самое можно сказать про любую ячейку, у диодов, ответственных за её выбор, имеются общие выводы. Если взять половину транзистора, не обращая внимания на всё остальное, то это, по сути, тот же самый *p*-*n*-переход. На практике при реализации схемы в интегральном исполнении оказывается более выгодным вместо отдельных групп диодов создавать *n*-*p*-*n*-транзисторы с несколькими эмиттерами. Поэтому нередко ту же самую электрическую схему организации ПЗУ можно увидеть с транзисторами в следующем виде.

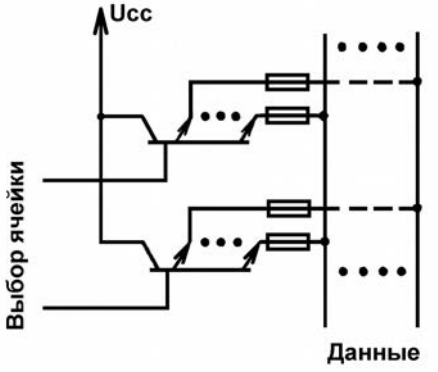


Рисунок 4.59. Схема ППЗУ, построенная на многоэмиттерных транзисторах

При занесении информации через диоды или эмиттеры шин с информационными нулями пропускаются большие токи, выжигающие плавкие перемычки. При подаче адресного импульса сигналы появляются только на тех выводах, где перемычки сохранены.

Так как прошивка данного вида ПЗУ возможна один раз в жизни, то в зарубежной литературе данный вид ПЗУ, помимо ROM, имеет ещё одно название OTP ROM (One-Time Programmable ROM), иногда One-Time опускают, и получается ещё одно более общее название PROM (Programmable ROM).

В случае если необходимо иметь возможность редкой перепрошивки записанных данных, следует использовать ППЗУ.

Замечание: современные ПЗУ могут иметь немного отличную и своеобразную организацию, будучи построенными на полевых транзисторах либо смешанным образом. В этом случае единичный элемент может выглядеть, как изображено на рис. 4.60.

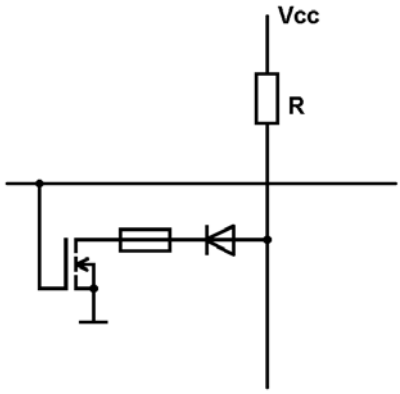


Рисунок 4.60. Один из вариантов организации элемента памяти с помощью полевого транзистора

ППЗУ – перепрограммируемое ПЗУ, ППЗУ с УФ-стиранием, EPROM (Erasable, Programmable, Read-Only Memory), UV-EPROM (UltraViolet EPROM). Несколько более дорогая технология, чем одноразовые ПЗУ, имеющая некоторые преимущества. В случае выполнения определённых действий позволяющая стирать старые и записывать новые данные. По внутреннему устройству данная память отличается от обычных ПЗУ. Вместо диодов и многоэмиттерных транзисторов используются специальные полевые МДП (металл–диэлектрик–проводник) транзисторы, в частности МОП (металл–оксид кремния (окисел)–проводник) с плавающим затвором (см. рис. 4.61).

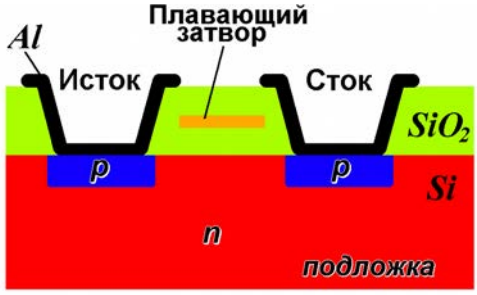


Рисунок 4.61. Полевой транзистор с плавающим затвором (ПЛМОП)

Из рис. 4.61 легко понять, почему затвор, состоящий из поликристаллического кремния, называется плавающим. Он со всех сторон окружён двуокисью кремния, не имеет выводов и соединений с остальной схемой. Такой транзистор представляет собой разомкнутую цепь до тех пор, пока на этот затвор не будет помещён заряд. Для этого между истоком и стоком прикладывается напряжение около 30 В, при этом происходит лавинный пробой, на затворе образуется остаточный заряд, достаточный для образования канала под затвором. Положительный заряд на затворе притягивает свободные электроны из подложки, а притянутые электроны создают канал, способный проводить электрический ток. Транзистор становится проводящей цепью. Если заряды на плавающем затворе не накоплены, то данный транзистор тока не проводит. Таким образом можно различать логические «1» и «0». Для стирания данных с ПЗУ необ-

ходимо удалить накопленный на плавающем затворе заряд. Для этого используют ультрафиолетовое излучение. Микросхемы данного вида памяти имеют прозрачные окошечки, выполненные из кварцевого стекла (см. рис. 4.62).

Обычно они заклеены чем-то непрозрачным. Ультрафиолетовое излучение (УФ) за счёт попадания на кристалл и его ионизации позволяет постепенно растечься накопленному заряду. В зависимости от интенсивности УФ-излучения процесс стирания может колебаться от 10 минут до одного часа.

Рисунок. 4.62. Фотография микросхем ПЗУ (http://www.155la3.ru/k573.htm)

Несмотря на то что для стирания рекомендуется использовать освещённость порядка 100 Вт/м^2 при времени экспозиции 1 час, редко кто оказывается в состоянии ждать такое количество времени.

Срок хранения информации в ПЛМОП ППЗУ практически бесконечен – постоянная времени утечки заряда составляет десятки лет. У данной технологии также есть и минусы: малое число возможных циклов перепрограммирования, примерно порядка 10, плюс для стирания необходимо наличие источника УФ-излучения, и другие связанные с этим неудобства. Так как у ПЛМОП-транзисторов нет затвора, то для выбора одного элемента памяти, как один из вариантов, последовательно с ними включаются обычные полевые транзисторы, что, несомненно, усложняет схему. Следующая технология лишена этих недостатков.

ЭС ППЗУ – электрически стираемые ППЗУ, EEPROM, E2PROM (Electrically Erasable PROM), как легко предположить из названия, вместо ультрафиолетового излучения для стирания используют электрический ток. В районе 1967 года для решения вышеописанных неудобств с УФ ППЗУ, в частности невозможности электрического стирания, были придуманы новые решения. В качестве ячейки памяти было предложено использовать транзисторы, выполненные по новой на то время МНОП (металл–нитрид кремния–оксид кремния (окисел)–полупроводник) или МАОП (металл–алунд–оксид кремния–полупроводник) технологии (см. рис. 4.63).

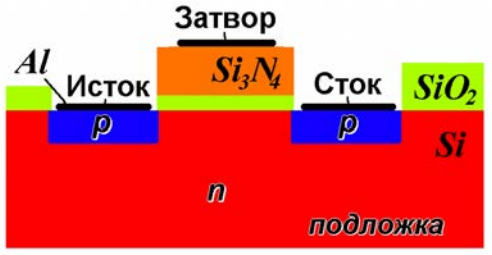


Рисунок 4.63. Транзистор, выполненный по МНОП (металл–нитрид кремния–оксид кремния–полупроводник) технологии

Транзистор, выполненный по МНОП технологии представляет из себя обычный МДП-транзистор, в котором диэлектрик состоит из двух слоёв. Поверх обычного диоксида кремния, используемого в МОП-транзисторах, нанесён ещё слой нитрида кремния. Нитрид кремния обладает очень высоким сопротивлением, значительно большим, чем диоксид кремния. Эффект памяти в МНОП-транзисторах основан на изменении порогового напряжения транзистора при наличии захваченного в подзатворном диэлектрике положительного или отрицательного заряда, который хранится на глубоких (1.3–1.5 эВ) ловушках, в нитриде кремния вблизи границы SiO_2–Si_3N_4 (в транзисторах памяти фирмы Hitachi толщина слоя окисла составляет порядка 20 Å, а нитрида кремния порядка 300–500 Å). [186]

Если на затвор подать напряжение порядка 30 В относительно подложки, то в течение 5 мс между тонким слоем окиси кремния и слоем нитрида кремния под затвором за счёт туннельного эффекта появятся неподвижные заряды. На образование токопроводящего канала между истоком и стоком будет влиять суперпозиция двух полей, от затвора и от накопленных в ловушках зарядах. Этим и объясняется следующая ВАХ (вольтамперная характеристика) МНОП-транзистора (см. рис. 4.64).

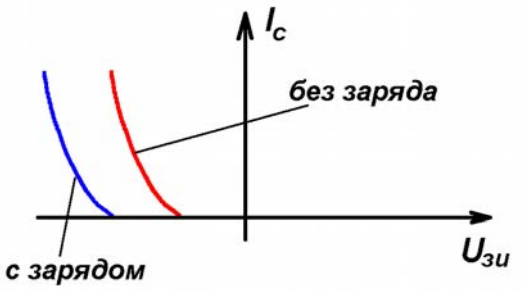


Рисунок 4.64. ВАХ МНОП-транзистора

Разность пороговых напряжений транзистора в ВАХ и фиксирует запись логических «0» или «1» в ЭС ППЗУ. Для того чтобы снять накопленные заряды (стереть записанную информацию), необходимо на затвор подать напряжение порядка 30 В обратной полярности.

В МНОП количество циклов перезаписи очень велико, однако обращения для чтения после записи постепенно уменьшают заряды, поэтому длительность сигналов чтения лимитирована. Как правило, ЭС ППЗУ допускает 10^{11}–10^{12} обращений, пока заряд начинает заметно снижаться.

В некоторых ЭС ППЗУ этого типа вместо нитрида кремния используется аналогичный по свойствам оксид алюминия (алунд), при этом технология называется МАОП и получаются не *p*-канальные, а *n*-канальные приборы, причём нормально открытые. При записи их пороговое напряжение не снижается, а повышается. Большим преимуществом МАОП и МНОП ЭС ППЗУ является наличие у запоминающих транзисторов внешнего управляющего затвора, с помощью которого и производится выборка ячейки.

4.11.4.2. Flash-память

То что, у каждой ячейки ЭС ППЗУ имеется возможность записи и удаления одного бита за одну операцию, это хорошо, но реально на практике это порождает много проблем и замедляет процесс перезаписи. Если ёмкость ПЗУ составляет 128 Кбайт, то это означает, что следует выполнить порядка одного миллиона стираний. Как вы понимаете, это не самый большой размер встречавшейся в прошлом и встречающейся на сегодня памяти. Для упрощения процесса стирания, чтобы это можно было сделать одной или несколькими командами, схема ЭС ППЗУ была немного внутренне усложнена,

[186] http://www.samag.ru/archive/article/219.

и появилась на свет так называемая flash-EEPROM-память или flash-память. В технологии непосредственного хранения и записи информации во flash-памяти ничего нового не появилось. Сейчас это наиболее ходовая и удобная память, используемая в компьютерной и бытовой технике.

Технология выпуска flash-памяти несколько дороже всех предыдущих, но получаемые преимущества с лихвой окупают повышение затрат.

Замечание 1: в некоторой литературе flash-EEPROM переводится как мигающий-EEPROM, наше мнение, что это не совсем правильный термин. Если дословно перевести слово flash, то получится вспышка. Скорее всего, этот смысл и закладывали разработчики данной технологии: насколько быстра вспышка, настолько быстро и осуществляется стирание данных из памяти. Помимо этого, если посмотреть на технологию стирания УФ ППЗУ, о которой написано чуть выше, то именно вспышка как раз лучше всего подходит по смыслу.

Замечание 2: в иностранной литературе наблюдается некоторая путаница, так у них имеется ещё одно сокращение для обозначения ЭС ППЗУ, а именно EAROM (Electrically Alterable ROM). По сути, ничего нового не придумано, однако имеются два различных способа обозначений, частично противоречащих друг другу: часто схемы запоминающих устройств с более высокой ёмкостью обозначают как EEPROM, а с меньшей – как EAPROM. Также через EEPROM иногда обозначают схемы запоминающих устройств, которые стираются целиком или поблочно, как flash-память, а под EAPROM понимают запоминаемые устройства, стираемые только по битам либо по байтам. В плане внесения путаницы в обозначения мы также не стоим на последнем месте, иногда под ПЗУ у нас понимается любое ПЗУ из вышеописанных, в том числе и flash-память.

Все описанные выше типы памяти имеют одинаковое обозначение на электрических схемах, за исключением случаев, если используется функция записи. Выпуск памяти как отечественными, так и импортными производителями производится в вариантах, сопрягаемых с ТТЛ, *n*-МОП и КМПОП-схемами. Это означает, что в пределах одной логики микросхемы памяти взаимозаменяемы. Сделано это целенамеренно с целью удешевления производства на этапах перехода к более массовому выпуску. Как выглядит микросхема flash-памяти, можно увидеть на рис. 4.65.

Рисунок 4.65. Пример микросхемы flash-памяти ёмкостью 128 Кбайт

Современные микросхемы flash-памяти несколько эволюционировали: они выполняются в других корпусах (меньше размер, иногда больше контактов) и имеют зна-

чительно больший внутренний объём. Следующим шагом и существенным изменением является изменение внутренней организации памяти (группировка ячеек).

4.11.4.3. NOR- и NAND-flash-память

Условно типы NOR и NAND обозначают логическую организацию элементов, используемую в данной единице flash-памяти, по аналогии с логическими элементами ИЛИ-НЕ и И-НЕ, рассмотренными в разделах 4.1.4 и 4.1.5. Различаются данные типы методом соединения ячеек в массив и алгоритмами чтения-записи.

Конструкция NOR использует классическую двумерную матрицу проводников, в которой на пересечении строк и столбцов установлено по одной ячейке. При этом проводник строк подключался к стоку транзистора, а столбцов – ко второму затвору. Исток подключался к общей для всех подложке. В такой конструкции было легко считать состояние конкретного транзистора, подав положительное напряжение на один столбец и одну строку. По сути, организацию NOR-памяти мы рассмотрели выше. Указанный тип организации хорош, пока размеры памяти не станут очень большими, настолько, что адресовать отдельно каждую ячейку памяти будет невыгодно.

Конструкция NAND – это трёхмерный массив. В основе него лежит та же самая матрица, что и в NOR, но вместо одного транзистора в каждом пересечении устанавливается столбец из последовательно включенных ячеек. В такой конструкции получается много затворных цепей в одном пересечении. Плотность компоновки можно резко увеличить (ведь к одной ячейке в столбце подходит только один проводник затвора), однако алгоритм доступа к ячейкам для чтения и записи заметно усложняется.

В сравнении получим следующее: технология NOR позволяет получить быстрый доступ индивидуально к каждой ячейке, однако площадь ячейки велика. Наоборот, NAND имеют малую площадь ячейки, но относительно длительный доступ сразу к большой группе ячеек. Соответственно, различается область применения:

NOR используется как непосредственная память программ микропроцессоров и для хранения небольших вспомогательных данных.	NAND используется для хранения значительно бóльших объёмов информации поблочно (например, во flash-накопителях и дисках).

Отсюда вытекает первое различие: достигнутые сегодня плотности записи для технологии NAND превосходят достигнутые в NOR, причём разница измеряется в порядках.

Существовали и другие варианты объединения ячеек в массив, но они не прижились.

Память для дисков делится на части, которые называются блоками. Обоснование деления памяти на блоки довольно просто. Это необходимо делать для преодоления некоторых физических ограничений и из ценовых соображений. Например, запись в обоих типах флэш-памяти определённой порции информации (блока) может быть произведена, только если место, куда она будет записываться (тоже блок), пусто или очищено. В большинстве случаев получается так, что перед операцией записи необходимо выполнить операцию стирания. И если в NAND-устройствах операция стирания блока может быть произведена сразу (вот оно, преимущество блочной записи), то в NOR-устройствах необходимо предварительно установить все байты блока в ноль.

Нужно также сказать, что типичное значение размера блока в NOR-устройствах составляет 64 или 128 КБ (8–32 КБ у NAND), что в сочетании с и так невысокими скоростями работы флэш приводит к тому, что операции записи со стиранием могут занимать до нескольких секунд. Это и является сдерживающим фактором применения NOR-флэша в качестве носителя данных. Применение же его для хранения программного кода возможно в том случае, если требования к скорости считывания и записи невысоки.

Время стирания памяти NAND измеряется в миллисекундах и имеет первый порядок. А малый размер блока в случае неблагоприятных внешних условий гарантирует потерю минимального объёма данных.

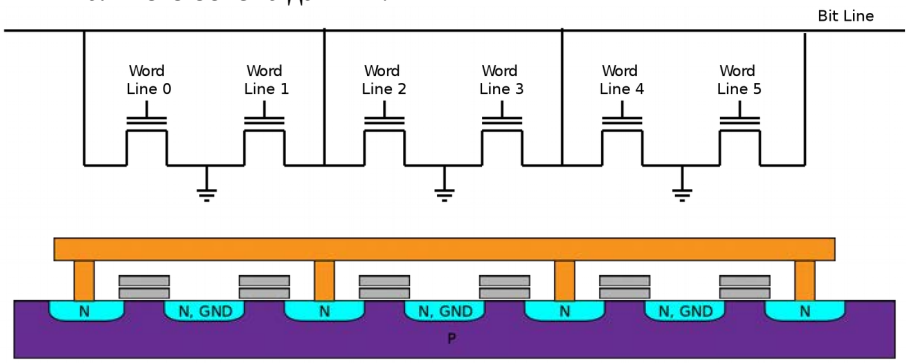


Рисунок 4.66. Компоновка шести ячеек NOR flash

Подведём итог: операции чтения NOR несколько быстрее NAND; операции же записи, наоборот, быстрее у NAND, причём значительно; благодаря малому размеру блока NAND в единицу времени нуждается в меньшем числе стираний (что, как мы увидим ниже, ещё и способно продлить срок её функционирования в устройстве), которые она проводит приблизительно на три порядка быстрее, чем NOR [187].

NOR-флэш является устройством памяти с произвольным доступом. Микросхемы NOR имеют интерфейс, позволяющий произвести адресацию и получить лёгкий доступ к каждому отдельному байту. Интерфейс ввода-вывода устройства памяти NAND значительно сложнее и меняется от устройства к устройству и от разработчика к разработчику. Одни и те же выводы (зачастую их 8) используются для передачи управляющих сигналов, адреса и данных.

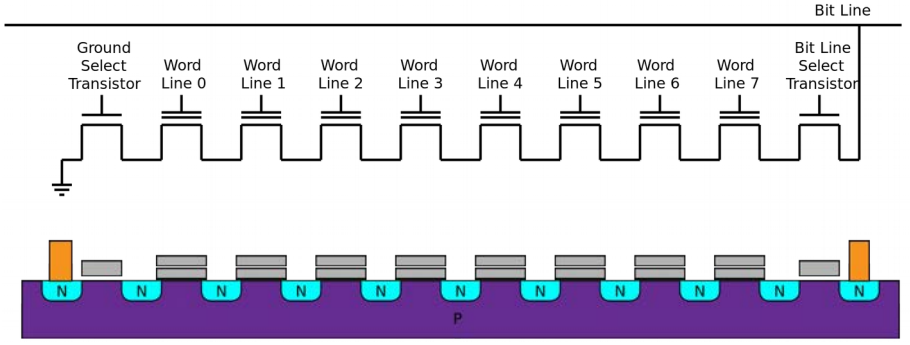


Рисунок 4.67. Структура одного столбца NAND flash

[187] http://www.nestor.minsk.by/kg/2004/35/kg43503.html.

Кроме того, в NAND-флэше доступ осуществляют блоками обычно в 512 байт, то есть за одно обращение считывается или записывается 512 байт. Доступ к каждому блоку произвольный, но, так как нет возможности обратиться к отдельному байту, память типа NAND не является в известном смысле памятью произвольного доступа. Выдача каждого байта из 512-байтного блока осуществляется на шину памяти последовательно, поэтому уместно говорить о последовательном доступе. Что и делают. Или о памяти со страничной организацией.

Схемотехнически ячейка памяти NAND организуется проще: она имеет меньший размер, по сравнению с NOR, и это соответственно приводит к повышению плотности записи, уменьшению энергопотребления и стоимости производства.

У любой технологии не могут быть только положительные стороны. В этом смысле NAND не исключение. Как и при эксплуатации любых накопителей, возможны случайные ошибки чтения и порча накопителя в целом. Для устройств flash-памяти актуально говорить о безошибочном чтении, обработке плохих блоков и числе циклов чтения/записи. Явление ошибочного вычитывания битов (называется bit-flipping) больше характерно для NAND-памяти, чем для NOR. Вред от одного ошибочного бита определяется типом данных, к которым он принадлежит. Так, для мультимедийных данных это окажется несущественным, но подобная ошибка в программном коде или критически важных данных может привести к весьма трагическим результатам. Как было отмечено ранее, для NOR-памяти это явление менее характерно, а память на технологиях NAND нуждается в использовании какого-то дополнительного механизма обнаружения и коррекции ошибок.

Технологии производства NAND-памяти пока несовершенны, и изначально память содержит какое-то число неработающих элементов. Так как в NAND группа запоминающих ячеек объединяется в блок, то испорченная ячейка в блоке приводит к неработоспособности блока в целом, то есть получается плохой блок. Поэтому появляется необходимость отслеживать состояние блоков и использовать только рабочие, что осуществить намного проще, чем произвести память, абсолютно не содержащую плохих страниц: такое производство оказывается очень дорогим (похожая ситуация была в свое время с битыми пикселями в LCD-панелях). По очевидным причинам этот вид дефектов не характерен для NOR-типа flash-памяти.

Рабочий ресурс микросхем флэш выражается в минимально и максимально возможном числе циклов стирания каждого отдельного блока (а мы уже знаем, что каждая запись блока обязательно сопровождается его предварительным стиранием). Для памяти на технологиях NOR оно составляет 10 000 и 100 000 циклов соответственно, для NAND – 100 000 и 1 000 000 циклов.

Использование NOR-памяти отличается сравнительной простотой. Она не нуждается в каких-либо дополнительных драйверах, а может быть просто установлена и использована. С NAND сложнее, так как разные производители используют разные интерфейсы, и для неё, скорее всего, понадобится драйвер. Впрочем, несмотря на то что у NAND-памяти много преимуществ, вы не должны думать, что NOR – это вчерашний день. NOR-память сегодня находит применение в многочисленных устройствах, не нуждающихся в больших объемах и некритичных к производительности. NAND находит применение в тех областях, где бóльшая сложность по применению оправдывается бо́льши́ми доступными объёмами и производительностью.

Увидеть данные микросхемы можно внутри обычных «USB-флэшек». Некоторые модели, обладающие полупрозрачным корпусом, даже не обязательно разбирать. Узнать подробнее, что находится внутри USB-flash-накопителя можно посмотрев на рис. 4.68.

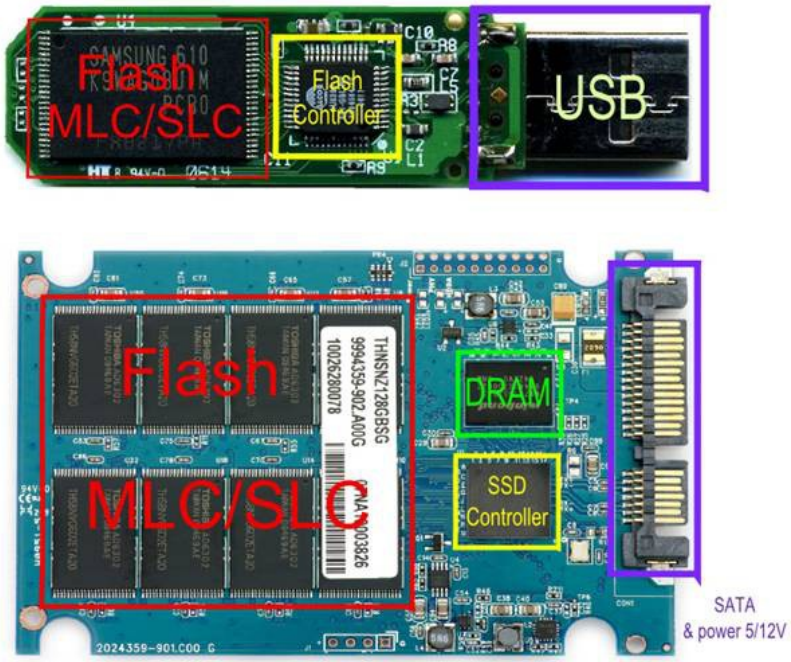


Рисунок 4.68. Внутренние платы USB-flash (сверху) и SSD (снизу) накопителей[188]

Теперь, имея представление о том, как работает flash-память, мы можем перейти к рассмотрению устройства существующих SSD-накопителей.

4.11.4.4. Разбираем SSD

Попробуем сравнить внутренности USB-flash и SSD, а также рассмотреть принцип работы последнего [189]. Как видно из рис. 4.68, SSD-накопитель состоит из печатной платы, на которой расположены flash-память, микросхема оперативной памяти – DRAM, контроллер и разъём SATA. Отличие от USB-flash накопителя [190] в наличии микросхемы оперативной памяти DDR DRAM, которая используется как кэш. Специфика появления кэша – это возросшая в несколько раз скорость обмена данными между контроллером и интерфейсом SATA. А из-за последнего мы поневоле относим SSD-накопители к жёстким дискам и наблюдаем их в прайс-листах рядом с классическими винчестерами.

В SSD, как и в USB-flash, используются flash память. Запоминание данных происходит в аналоговом виде. «Захваченные» ячейкой памяти электроны создают статическое электрическое поле влияющее на проводимость транзистора ячейки памяти. Первоначально различали два состояния «транзистор проводит электрический ток (низкое сопротивление току)» и «транзистор не проводит электрический ток (высокое сопротивление току)». Такие ячейки памяти назывались Single Level Cell. По мере совершенствования технологии в одном транзисторе удалось надёжно хранить и разделять четыре разных уровня проводимости, что позволило закодировать и хранить в такой ячейки сразу два бита. Такие ячейки назвали Multi Level Cell. Однако, позже оказа-

[188] http://www.usb-disk.ru/page22.php, http://www.storelab-rc.ru/ssd-review.htm.
[189] http://www.usb-disk.ru/page22.php, http://www.storelab-rc.ru/ssd-review.htm.
[190] См. раздел 4.7.6 «Флэш память».

лось, что это не предел совершенства технологии и стало возможным снизить шумы и выделить восемь различных аналоговых уровней записи. Это позволило в одной ячейки памяти хранить сразу три бита информации. Такие ячейки назвали Three Level Cell.

Описанные выше ячейки памяти относятся к планарному, то есть 2D-типу, поскольку их физическая организация на кремниевой пластине плоская или 2D. Увеличение плотности записи за счёт дальнейшего уменьшения размеров техпроцесса встретило трудности. Поэтому были разработаны 3D-ячейки памяти. Такая ячейка представляет собой цилиндр. Маркетологи их называют 3D V-NAND, 3D TLC. Увеличение размеров при переходе к 3D позволило довести количество разделяемых при записи уровней до 16 и тем самым появились QLC ячейки или 3D QLC (Quad Level Cell, – для хранения сразу 4-х бит информации).

SLC (Single Level Cell)	MLC (Multi Level Cell)	TLC (Three Level Cell)	QLC (Quad Level Cell)

Физика процессов такова, что надёжность памяти у ячеек с несколькими уровнями (TLC и QLC) ниже чем у SLC и MLC ячеек с меньшим их числом. Чтобы компенсировать и снизить вероятность ошибок чтения используется помехоустойчивое кодирование (ECC, Error Correction Code, код коррекции ошибок). Для разных типов памяти применяются коды с отличающимися параметрами.

В конечном итоге пользователю данные выдаются с гарантированно заданным уровнем надёжности. Так, благодаря маркетологам в плайс-листах можно встретить eMLC ячейки памяти (Enterprise Multi-Level Cell) – ячейки, аналогичные по структуре обычной MLC, но с увеличенным ресурсом по числу операций записи (стирания), как у SLC. Цена eMLC меньше чем у SLC. Этот хитрый ход можно применить и к TLC и к QLC памяти с коррекцией ошибок, назвав эти ячейки немного по другому.

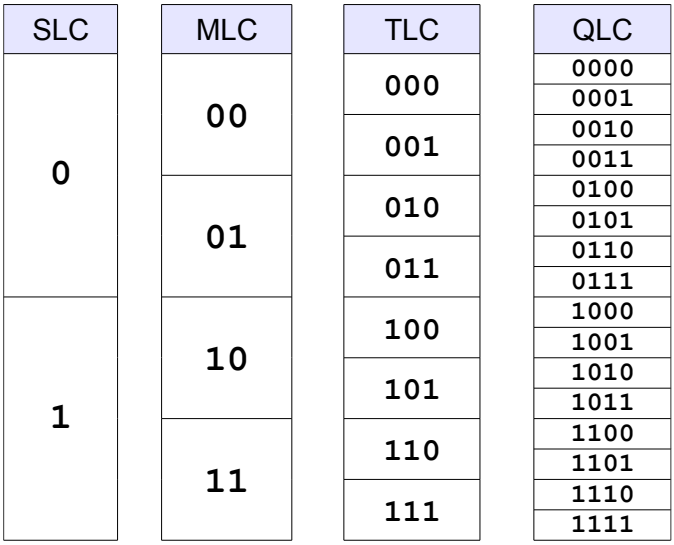


Рисунок 4.69. Варианты хранения данных в ячейках SLC, MLC и TLC

Естественно, память группируется так, чтобы предоставить доступ к ней блоками по 512 или 4096 байт, внутреннее физическое устройство одной ячейки памяти оказывается скрытым от пользователей.

Также память MLC/TLC/QLC обладает меньшим ресурсом (в среднем 100 000 циклов стирания у SLC, 10 000 – MLC, до 5000 – TLC и для QLC – ещё меньше) и худшим быстродействием. С каждым дополнительным уровнем усложняется задача распознавания уровня сигнала, увеличивается время поиска адреса ячейки, повышается вероятность ошибок. Так как SLC-чипы[191] намного дороже и объём их ниже, то для массовых решений применяют в основном MLC/TLC/QLC-чипы. На данный момент MLC/TLC/QLC-память активно развивается и по скоростным характеристикам приближается к SLC. Также низкую скорость MLC/TLC/QLC производители SSD-накопителей компенсируют алгоритмами чередования блоков данных между микросхемами памяти (одновременная запись/чтение в две или четыре микросхемы флэш-памяти, по байту в каждую) по аналогии с RAID 0, а низкий ресурс – перемешиванием и слежением за равномерным использованием ячеек. Плюс к этому в SSD резервируется часть объёма памяти (до 20%). Это недоступная память для стандартных операций записи/чтения. Она необходима как резерв в случае износа ячеек, по аналогии с магнитными накопителями HDD, который имеет резерв для замены bad-блоков. Дополнительный резерв ячеек используется динамически, и по мере физического изнашивания [192] основных ячеек предоставляется резервная ячейка на замену. Для конечного пользователя и операционной системы этот процесс происходит прозрачно и незаметно.

4.11.4.5. Как работает SSD-накопитель

Для чтения блока данных в винчестере сначала нужно вычислить, где он находится, потом переместить блок магнитных головок на нужную дорожку, подождать, пока нужный сектор окажется под головкой и произвести считывание. Причём хаотические запросы к разным областям жёсткого диска ещё больше сказываются на времени доступа. При таких запросах HDD вынуждены постоянно «гонять» головки по всей поверхности «блинов», и даже переупорядочивание очереди команд спасает не всегда. А в SSD всё просто – вычисляем адрес нужного блока и сразу же получаем к нему доступ на чтение/запись. Никаких механических операций – всё время уходит на трансляцию адреса и передачу блока. Чем быстрее флэш-память, контроллер и внешний интерфейс, тем быстрее доступ к данным.

А вот при изменении/стирании данных в SSD-накопителе не так всё просто. Микросхемы NAND флэш-памяти оптимизированы для секторного выполнения операций. Флэш-память пишется блоками по 4 КБ, а стирается по 512 КБ. При модификации нескольких байтов, внутри некоторого блока контроллер выполняет следующую последовательность действий:

- считывает блок, содержащий модифицируемый блок, во внутренний буфер/кэш;
- модифицирует необходимые байты;
- выполняет стирание блока в микросхеме флэш-памяти;
- вычисляет новое местоположение блока в соответствии с требованиями алгоритма перемешивания;

[191] Чип – от анг. *chip* – микросхема; интегральная схема.
[192] SSD-диски – принцип работы см. http://www.usb-disk.ru/page22.php.

- записывает блок на новое место.

Но как только вы записали информацию, она не может быть перезаписана до тех пор, пока не будет очищена. Проблема заключается в том, что минимальный размер записываемой информации не может быть меньше 4 КБ, а стереть данные можно минимум блоками по 512 КБ. Для этого контроллер группирует и переносит данные для освобождения целого блока.

Вот тут и сказывается оптимизация ОС [193] для работы с HDD. При удалении файлов операционная система не производит физическую очистку секторов на диске, а только помечает файлы как удалённые и знает, что занятое ими место можно заново использовать. Работе самого накопителя это никак не мешает, и разработчиков интерфейсов этот вопрос раньше не волновал. Если такой метод удаления помогает повысить производительность при работе с HDD, то при использовании SSD становится проблемой. В SSD, как и в традиционных жёстких дисках, данные всё ещё хранятся на диске после того, как они были удалены операционной системой. Но дело в том, что твердотельный накопитель не знает, какие из хранящихся данных являются полезными, а какие уже не нужны, и вынужден все занятые блоки обрабатывать по длинному алгоритму.

Прочитать, модифицировать и снова записать на место, после очистки затронутых операцией ячеек памяти, которые с точки зрения ОС уже удалены. Следовательно, чем больше блоков на SSD содержит полезные данные, тем чаще приходится прибегать к процедуре «*чтение–модификация–очистка–запись*» вместо прямой записи. Вот здесь пользователи SSD сталкиваются с тем, что быстродействие диска заметно снижается по мере их заполнения файлами. Накопителю просто не хватает заранее стёртых блоков. Максимум производительности демонстрируют чистые накопители, а вот в ходе их эксплуатации реальная скорость понемногу начинает снижаться.

Раньше в интерфейсе ATA просто не было команд для физической очистки блоков данных после удаления файлов на уровне ОС. Для HDD они просто не требовались, но появление SSD заставило пересмотреть отношение к данному вопросу. В результате в спецификации ATA появилась новая команда DATA SET MANAGEMENT, более известная как Trim. Она позволяет ОС на уровне драйвера собирать сведения об удалённых файлах и передавать их контроллеру накопителя.

В периоды простоя SSD самостоятельно осуществляет очистку и дефрагментацию блоков, отмеченных как удалённые в ОС. Контроллер перемещает данные так, чтобы получить больше предварительно стёртых ячеек памяти, освобождая место для последующей записи. Это даёт возможность сократить задержки, возникающие в ходе работы.

Но для реализации Trim необходима поддержка этой команды прошивкой накопителя и установленным в ОС драйвером. На данный момент только самые последние модели SSD «понимают» TRIM, а для старых накопителей нужно прошить контроллер для включения поддержки этой команды. Среди операционных систем команду Trim поддерживают: Linux 2.6.33, FreeBSD 9.0, Windows 7, Windows Server 2008 R2. Для остальных ОС необходимо инсталлировать дополнительные драйверы и утилиты либо перекомпилировать ядро или модули.

Например, для SSD от Intel существует специальная утилита SSD Toolbox, которая может выполнять процедуру синхронизации с ОС по расписанию. Кроме оптимизации, утилита позволяет выполнять диагностику SSD и просматривать S.M.A.R.T.-

[193] Об операционных системах (ОС) см. главу 5 «Программное обеспечение».

данные всех накопителей компьютера. С помощью S.M.A.R.T. можно оценить текущую степень износа SSD – параметр E9 (233_{10}) Media_Wearout_Indicator отражает оставшееся количество циклов очистки NAND-ячеек в процентах от нормативного значения. Когда величина, уменьшаясь от 100, дойдёт до 1, можно ожидать скорого появления «битых» блоков. Просмотреть имеющееся у вас значение можно с помощью ранее упомянутой утилиты smartctl:

```
# smartctl -a /dev/sda|grep Media_Wearout
233 Media_Wearout_Indicator 0x0000   000   000   000    Old_age   Offline
-       100
```

4.11.4.6. Контроллер SSD

Главными задачами контроллера является обеспечение операций чтения/записи и управление структурой размещения данных. Основываясь на матрице размещения блоков, в какие ячейки уже проводилась запись, а в какие ещё нет, контроллер должен оптимизировать скорость записи и обеспечить максимально длительный срок службы SSD-диска. Вследствие особенностей построения NAND-памяти работать с её каждой ячейкой отдельно нельзя. Ячейки объединены в страницы объёмом по 4 Кбайта, и записать информацию можно, только полностью заняв страницу. Стирать данные можно по блокам, которые равны 512 Кбайт. Все эти ограничения накладывают определённые обязанности на правильный интеллектуальный алгоритм работы контроллера. Поэтому правильно настроенные и оптимизированные алгоритмы контроллера могут существенно повысить производительность и долговечность работы SSD-диска.

В контроллер входят следующие основные элементы:

Процессор (Processor) – как правило, 16- или 32-разрядный микроконтроллер. Выполняет инструкции микропрограммы, отвечает за перемешивание и выравнивание данных на Flash, диагностику S.M.A.R.T., кэширование и прочее.

Error Correction (ECC) – блок контроля и коррекции ошибок ECC.

Flash Controller – включает адресацию, шину данных и контроль управления микросхемами flash-памяти.

DRAM Controller – адресация, шина данных и управление DDR/DDR2/SDRAM кэш-памятью.

I/O interface – отвечает за интерфейс передачи данных на внешние интерфейсы SATA, USB или SAS.

Controller Memory – состоит из ROM памяти и буфера. Память используется процессором для выполнения микропрограммы и как буфер для временного хранения данных. При отсутствии внешней микросхемы RAM памяти выступает в роли единственного буфера данных SSD.

На данный момент в SSD применяются следующие модели контроллеров [194]:

- Indilinx "Barefoot ECO" IDX110MO1;
- Indilinx "Barefoot" IDX110M00;
- Intel PC29AS21BA0;
- JMicron JMF602;
- JMicron JMF612;
- Marvel 88SS9174-BJP2;
- Samsung S3C29RBB01-YK40;
- SandForce SF-1200;
- SandForce SF-1500;
- Toshiba T6UG1XBG.

[194] http://www.storelab-rc.ru/ssd-review.htm.

4.11.4.7. Надёжность SSD

Казалось бы, нет движущихся частей – всё должно быть очень надёжно. Это не совсем так. Любая электроника может сломаться, не исключение и SSD. С низким ресурсом MLC-чипов ещё можно как-то бороться коррекцией ошибок ECC, резервированием, контролем за износом и перемешиванием блоков данных. Но самый большой источник проблем – контроллер и его прошивка. По причине того, что контроллер физически расположен между интерфейсом и микросхемами памяти, вероятность его повреждения в результате сбоя или проблем с питанием очень велика. При этом сами данные в большинстве случаев сохраняются. Помимо физических повреждений, при которых доступ к данным пользователя невозможен, существуют логические повреждения, при которых также нарушается доступ к содержимому микросхем памяти. Любая, даже незначительная ошибка, «баги» в прошивке, может привести к полной потере данных. Структуры данных очень сложные. Информация «размазывается» по нескольким чипам, плюс чередование, что делает восстановление данных довольно сложной задачей.

В таких случаях восстановить накопитель помогает прошивка контроллера с низкоуровневым форматированием, когда заново создаются служебные структуры данных. Производители стараются постоянно дорабатывать микропрограмму, исправлять ошибки, оптимизировать работу контроллера. Поэтому рекомендуется периодически обновлять прошивку накопителя для исключения возможных сбоев.

4.11.4.8. Особенности удаления файлов на SSD-носителях

В SSD-накопителе, как и в HDD, данные не удаляются сразу после того, как файл был стёрт из ОС. Даже если затереть файл нулями [195], физически данные ещё остаются, и если чипы флэш-памяти достать и считать на программаторе – можно найти 4 КБ фрагменты файлов. Полное стирание данных стоит ждать тогда, когда на диск будет записано число данных, равное количеству свободного места + объём резерва (примерно 4 ГБ для 60 ГБ SSD). Если файл попадёт на «изношенную» ячейку, контроллер ещё не скоро перезапишет её новыми данными.

Восстановление данных с SSD-накопителей – достаточно трудоёмкий и долгий процесс, по сравнению с портативными flash-накопителями. Процесс поиска правильного порядка, объединения результатов и выбора необходимого сборщика (алгоритм/программа, полностью эмулирующая работу контроллера SSD-накопителя) для создания образа диска – нелёгкая задача.

Связано это в первую очередь с увеличением числа микросхем в составе SSD-накопителя, что во много раз увеличивает число возможных вариантов действий на каждом этапе восстановления данных, каждое из которых требует проверки и специализированных знаний. Также в силу того, что к SSD предъявляются значительно более жёсткие требования по всем характеристикам (надёжность, быстродействие и т. д.), чем к мобильным флэш-накопителям, технологии и методики работы с данными, применяемые в них, достаточно сложны, что требует индивидуального подхода к каждому решению и наличия специализированных инструментов и знаний.

[195] Например, утилитой wipe в ОС Linux или с помощью команды:
«dd if=/dev/zero of=/путь/файл bs=размер_файла count=1».

4.11.4.9. Оптимизация SSD

1. Для того чтобы диск прослужил вам долго [196], нужно всё, что часто меняется (временные файлы, кэш браузера, индексирование), перенести на HDD (подразумевается, что у вас два диска), отключить обновление времени последнего доступа к папкам и каталогам (fsutil behavior set disablelastaccess 1). Отключить в ОС дефрагментацию файлов.

2. Перед установкой на SSD Windows XP, при форматировании диска, рекомендуется выполнить «выравнивание» разделов, кратных степени двойки (например, утилитой diskpart), иначе SSD придётся делать 2 чтения вместо одного. Кроме этого, у Windows XP есть некоторые проблемы с поддержкой секторов более 512 байт (в SSD по умолчанию используется 4 КБ) и вытекающие отсюда проблемы с производительностью. Windows Vista, 7 и 8, последние версии Mac OS и Linux выравнивают диски уже правильно.

3. Обновить прошивку контроллера, если старая версия не знает команду TRIM. Установить последние драйверы на SATA-контроллеры. Например, если у вас контроллер от Intel, вы можете на 10–20% увеличить производительность, включив режим ACHI и установив Intel Matrix Storage Driver в операционной системе.

4. Не следует использовать последние 10–20% свободного пространства от раздела, потому что это может отрицательно сказаться на производительности. Это особенно важно, когда работает TRIM, поскольку ему необходимо пространство для перегруппировки данных: для примера, похоже работают утилиты дефрагментации, ведь им тоже нужно не менее 10% от объёма диска. Поэтому очень важно следить за данным фактором, ведь из-за небольшого объёма SSD они очень быстро заполняются.

4.11.4.10. Преимущества SSD-накопителей

Собственно, давайте наконец перечислим все те плюсы, ради которых вам пришлось прочитать все разделы выше и, возможно, отказаться от использования привычных винчестеров:

- высокая скорость чтения любого блока данных независимо от физического расположения (более 300–400 МБ/с);
- низкое энергопотребление при чтении данных с накопителя (приблизительно на 1 Вт ниже, чем у HDD);
- пониженное тепловыделение (внутреннее тестирование в компании Intel показало, что ноутбуки с SSD нагреваются на 12,2 °C меньше, чем аналогичные с HDD, также тестированием установлено, что ноутбуки с SSD и 1 ГБ памяти в распространённых «бенчмарках» не уступают моделям с HDD и 4 ГБ памяти);
- бесшумность и высокая механическая надёжность.

4.11.4.11. Недостатки SSD-накопителей

- Высокое энергопотребление при записи блоков данных, энергопотребление растёт с ростом объёма накопителя и интенсивностью изменения данных;

[196] См. Оптимизируем Linux для SSD: http://www.openkazan.info/Linux-SSD.

- низкая ёмкость и высокая стоимость за гигабайт, по сравнению с HDD;
- ограниченное число циклов записи.

4.11.4.12. Заключение по SSD-накопителям

Первоначально из-за относительно высокой стоимостью хранения 1 байта информации и небольших объёмах носителей (доступных на рынке), использовать SSD-диски для долгого хранения больших объёмов данных было невыгодно. Их ставили в качестве второго (или первого) диска для системного раздела, на который устанавливалась ОС, а также их применяли на серверах для кэширования статичных данных. Сейчас ситуация меняется, цены падают, объёмы и надёжность хранения растут, поэтому всё чаще можно встретить компьютеры исключительно с твердотельными накопителем, а то и двумя.

При покупке диска обязательно обратите внимание на скорости чтения и записи. У дисков большего объёма они обычно больше (видимо, за счёт внутреннего, прозрачного для пользователя, чередования блоков), вплоть до 500 ГБ/с. Однако, достичь столь привлекательных характеристик в работе возможно лишь в случае использования SATA разъёмов 6.0 Gbps (SATA-3). При использовании SATA разъёмов 3.0 Gbps (SATA-2), а тем более SATA-1 – 1.5 Gbps, накопители будут работать, но их скорости чтения и записи могут упасть намного ниже предельной теоретической пропускной способности указанных для интерфейсов и оказаться в несколько раз ниже от заявленных производителем диска. Итоговые реальные скорости чтения и записи данных могут оказаться на уровне 30-40% от верхних ограничений пропускной способности интерфейса. В ряде случаев это становится неприятной неожиданностью, как если вам придётся ехать в хорошую погоду по свободной загородной трассе, имеющей ограничение скорости 90 км/ч, со скоростью 55-70 км/ч, выжав при этом ногой педаль акселератора до предела и смотря на спидометр, где раздражающее будет красоваться максимальное значение в 200 км/ч.

Также обратите внимание на срок гарантии для накопителя. Мы не рекомендуем покупать диск с гарантией менее 60 месяцев (5 лет). Во-первых, это добавит вам спокойствия, а во-вторых, будет стимулировать производителей и продавцов делать свою работу качественнее. Технический прогресс и так развивается очень быстро, чтобы его ещё ускорять из кармана потребителя.

Перечитайте рекомендации раздела 4.11.4.9. «*Оптимизация SSD*». Обновите прошивку диска сразу после покупки, скачав соответствующие программы на сайте производителя накопителя. Стоимость последствий сбоя будет минимальная для ваших ещё не записанных данных.

Не забывайте делать резервные копии всех ваших данных.

4.11.5. Дисководы оптических дисков

Приводы оптических дисков как нельзя кстати подходят для создания резервных копий. Узнать какой у вас стоит привод оптических дисков можно визуально, посмотрев на него, лучше конечно с открытой крышкой корпуса, чтобы можно было определить фирму производителя и модель, написанные на верхней части его корпуса, убе-

диться в правильности его подключения к разъёму питания и контроллеру для передачи данных.

В целом различные приводы хорошо унифицированы и не требуют какой-либо дополнительной настройки со стороны пользователя, легко определяются большинством операционных систем и в пределах гарантийного срока эксплуатации не капризны.

Программным способом увидеть какой у вас установлен привод можно во время загрузки ядра (естественно, если оно было сконфигурировано с поддержкой оптических приводов), либо позже с помощью команды dmesg.

```
$ dmesg
[3.080335] scsi 2:0:0:0: CD-ROM ФИРМА CDDVDW МОДЕЛЬ SB01 PQ: 0 ANSI: 5
...
[3.109197] sr 2:0:0:0: [sr0] scsi3-mmc drive: 50x/50x writer dvd-ram cd/rw
xa/form2 cdda tray
[3.109531] cdrom: Uniform CD-ROM driver Revision: 3.20
[3.109843] sr 2:0:0:0: Attached scsi CD-ROM sr0
```

Дайте команду

```
$ eject
```

или, если у вас несколько приводов, то укажите необходимый привод

```
$ eject /dev/sr0
```

и система среагирует на неё выдвижением трея загрузки дисков[197]. Если у вас не ноутбучный вариант исполнения привода, то та же самая команда, но с указанием ключа «`-t`», задвинет его обратно.

В целом принципиальной разницы между CD, DVD и Blue-Ray приводами и используемыми ими дисками нет, но поскольку исторически первым появились оптические диски называемые «компакт дисками» (по сравнению с размерами аналоговых аудио пластинок), то с них (и соответствующих им проводов) мы и начнём.

4.11.5.1. Дисковод CD-ROM

В 1980 году компаниями Sony и Philips был представлен стандарт CD-DA (он же «красная книга», по цвету обложки) для хранения звуковых записей. Через несколько лет, в 1984 году ими же был представлен стандарт для хранения данных **CD-ROM** (Compact Disk Read Only Memory, постоянное запоминающее устройство на базе компакт-диска), получивший название «жёлтая книга». По физическим размерам диски обоих форматов полностью совпадали. [4]

В 1995 году в базовой конфигурации ПК появился первый привод оптических дисков – **CD-ROM** drive. Устройство использовало компакт-диски диаметром 120 мм, толщиной 1,2 мм и ёмкостью 650–700 Мбайт.

Ставший стандартом, CD-диск состоит из 4-х слоёв (сверху вниз):

1) защитный слой из поликарбоната;
2) слой для записи информации;
3) отражающий слой;
4) основа из поликарбоната.

Дополнительно на защитный слой может быть на нанесён рисунок краской.

Процесс изготовления диска состоит из операций напыления на основу отражающего слоя серебра или золота, нанесения на него прозрачного слоя для записи инфор-

[197] Если только имеющийся в нём диск не заблокирован какой-либо программной и у вас не щелевой slice-привод без трея.

мации и выдавливанием на нём углублений, образующих спиральную дорожку, идущую от центра диска к его краю. Для штамповки диска используют матрицу-прототип (мастер-диск) будущего диска. После этого на поверхность диска наносят защитный слой из прозрачного пластика.

CD-ROM считывает информацию с диска с помощью лазерного луча с длиной волны 780 нм, который по-разному отражается от поверхности диска (land – ленд) и углублений на поверхности (pit – пит, см. рис. 4.70). Минимальный размер пита составляет 0,88 мкм, шаг дорожек – 1,5 мкм.

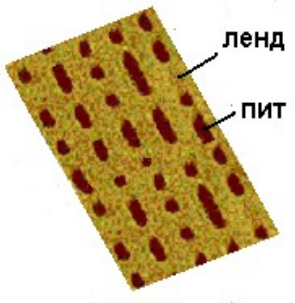


Рисунок 4.70. Вид поверхности оптического CD диска (под увеличением)

Информация при подготовке к записи на компакт-диске подвергается ряду сложных преобразований. Вначале используется ***помехоустойчивое кодирование***, затем осуществляется так называемое ***канальное кодирование***. В компакт-дисках применяется канальный код EFM (Eight to Fourteen Modulation), разработанный фирмой Philips для лазерной звукозаписи. Суть канального кодирования EFM заключается в том, что каждый байт информации заменяется 14-разрядным словом из специальной таблицы преобразования. К полученному таким образом 14-разрядному слову по определённому правилу добавляются ещё три так называемых соединительных разряда. В результате канального кодирования получается непрерывная последовательность бит, причём между двумя единицами никогда не может быть меньше двух или больше десяти нулей.

Именно эта последовательность бит и представлена на компакт-диске питами. Причём питы и ленды чередуются в моменты, соответствующие биту, равному 1 (см. рис. 4.71). Если до этого был, например, ленд, то становится пит, и наоборот. Таким образом, питы и ленды, с точки зрения представления информации, равнозначны и характеризуют только временные интервалы между двумя единицами в последовательности бит.

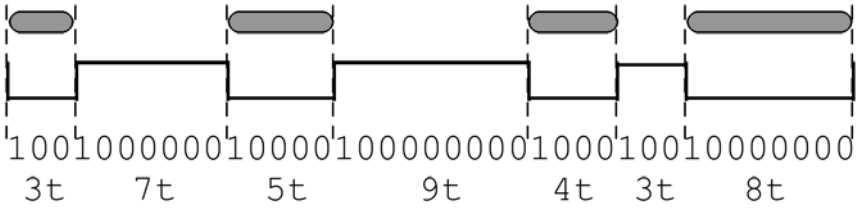


Рисунок 4.71. Схема преобразования при считывании информации с компакт-диска

При считывании информации луч лазера отражается от лендов и попадает в фотоприёмник, от питов свет тоже отражается, но из-за расфокусировки луча в фотоприёмник не попадает. Далее высокочастотный информационный сигнал преобразуется компаратором в последовательность прямоугольных импульсов. Длительность каждого из полученных таким способом прямоугольных импульсов преобразуется в двоичный код. Каждые 14 последовательных битов декодируются в соответствии с таблицей кодирования EFM, соединительные разряды при этом отбрасываются.

Система кодирования обеспечивает также работу системы автослежения (автотрекинга), которой оснащены все приводы CD-ROM. Размеры каждого пита очень малы – порядка 0,6 мкм, а шаг спирали – всего 1,6 мкм. Назначение системы автослежения – точное позиционирование считывающей головки. Слежение осуществляется за дорожкой с данными по расположенным на ней питам, для нормальной работы системы питы не должны отсутствовать более 10 периодов тактовой частоты, что и обеспечивается системой кодирования.

Вторая задача, которая возникает при считывании информации с компакт-диска, – точное выделение временных интервалов, соответствующих периоду следования символов последовательного кода. Поддерживать скорость вращения диска с требуемой точностью практически невозможно. Поэтому для решения этой задачи временные интервалы определяются путём подстройки частоты тактового генератора под реальную скорость поступления информации с помощью инерционной системы автоподстройки, что опять-таки возможно выполнить благодаря использованию канального кода EFM при записи.

Дисковод CD-ROM содержит:

- электродвигатель, который вращает диск;
- оптическую систему, состоящую из лазерного излучателя, оптических линз и датчиков и предназначенную для считывания информации с поверхности диска;
- микропроцессор, который руководит механикой привода, оптической системой и декодирует прочитанную информацию в двоичный код.

Основные характеристики CD-ROM:

- скорость передачи данных – измеряется в кратных долях скорости проигрывателя аудиокомпакт-дисков (150 Кбайт/с) и характеризует максимальную скорость, с которой накопитель пересылает данные в оперативную память компьютера, например, 2-скоростной CD-ROM (2× или 2x CD-ROM) будет считывать данные с скоростью 300 Кбайт/с, 50-скоростной (50× или 50x) – лишь до 7500 Кбайт/с;[198]
- время доступа – время, нужное для поиска информации на диске, измеряется в миллисекундах.

[198] Изначально приводы использовали моторы обеспечивающие фиксированную линейную скорость вращения дисков для обеспечения скорости чтения 150 Кбайт/с, то есть, при помещении считывающей головки в начало спирали (ближе к центру диска) скорость вращения увеличивалась, а по мере смещения её к краю диска скорость вращения привода уменьшалась. Вплоть до 8x-12x скоростных приводов удавалось поддерживать такой режим, после чего выяснилось, что на более быстрых моделях приводов 16x, 24x и выше добиться фиксированной линейной скорости вращения без значительного усложнения устройства мотора не получается. После этого стали производить комбинированные приводы, работающие на низких скоростях с фиксированной линейной скоростью вращения, а на больших – с фиксированной угловой скоростью вращения диска. Более поздние модели вообще отказались от режима фиксированной линейной скорости. В случае использования некачественных дисков (болванок), последние могли разорваться в моделях 52x и 54x, что было небезопасным. В сети без труда можно найти видеоролики на эту тему. На сегодняшний день практически все производители оптических приводов ограничивают скорость вращения дисков до условных 48x.

4.11.5.2. Дисковод CD-RW

Устройство используется для записи информации на диски CD-R (однократная запись) и CD-RW (CD-ReWritable – перезаписываемый диск).

Внешне похож на CD-ROM и совместим с ним по размерам дисков и форматам записи. Запись данных осуществляется с помощью специального программного обеспечения [199] или средств операционной системы. Скорость записи современных накопителей CD-R составляет 2x–48x.

Диск CD-RW имеет схожую с CD-R структуру их 4-х слоёв, отличие лишь в устройстве активного слоя для записи информации. Во время записи информации последний нагревается лазерным лучом (с длиной волны 780 нм) и в месте нагрева перестаёт пропускать свет. В результате отражающая способность на участках, подвергнутых действию луча лазера, уменьшается. Одним из типов активных слоёв CD-R-диска, широко используемых на сегодняшний день, является цианин (cyanine).

Активный слой CD-RW-диска под воздействием лазерного луча при записи информации изменяет своё агрегатное состояние с переходом из кристаллического состояния в аморфное. Участки диска с аморфным состоянием активного слоя имеют меньшую степенью отражения лазерного луча при считывании информации. Процесс стирания информации с CD-RW заключается в обработке поверхности диска лучом лазера меньшей интенсивности, чем при записи, в результате чего активный слой полностью возвращается в кристаллическое состояние.

Основа CD-R и CD-RW имеет негладкую поверхность: при изготовлении она получает разметку – сплошную спиральную канавку (Pregroove), необходимую для работы системы автослежения, заполненную органическим красителем, который немного хуже отражает свет, чем пластик основы. Для синхронизации частоты тактового генератора со скоростью вращения диска в CD-R канавка выполнена не в виде ровной спирали, а с микроскопическими отклонениями – вобуляцией (см. рис. 4.72).

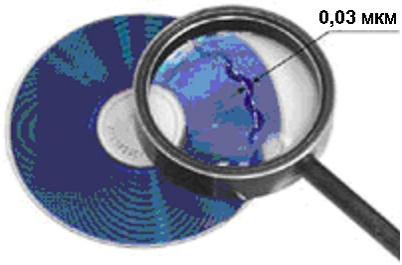


Рисунок 4.72. Разметка на слое-основе CD-R, видна вобуляция

Частота колебаний канавки относительно спиральной траектории составляет 22,05 кГц (для скорости вращения диска 1x). Соответственно, один период этих колебаний занимает 60 мкм спиральной траектории. Амплитуда колебаний – всего 0,03 мкм, значительно меньше ширины самой канавки, но этого достаточно, чтобы выделить колебания с частотой 22,05 кГц и синхронизировать этими колебаниями частоту своего тактового генератора. Кроме того, вся спиральная разметка разбивается на фреймы, каждый из которых по длительности соответствует одному кадру информации. Информация о номере фрейма (будущего кадра) представлена на разметке путём сдвига частоты вобуляции на 1 кГц от значения 22,05 кГц, то есть реально частота вобуляции принимает значения 21,05 кГц или 23,05 кГц.

Диск CD-RW (CD-ReWritable) используется для многоразовой записи данных, причём можно как просто дописать новую информацию на свободное пространство, так и полностью перезаписать диск новой информацией (после предварительной

[199] Например, с помощью свободного ПО k3b, brasero, dvdrecord для ОС Linux.

очистки всего диска). Работа с оптическими дисками в режиме записи отличается от аналогичных у дискет и USB-флэшек и обеспечивается поддержанием сессий на дисках, а также использованием формата UDF. Первые приводы дозапись не поддерживали. Скорость записи современных накопителей CD-RW составляет 2x–24x.

4.11.5.3. Дисководы DVD-ROM и DVD±RW

В 1996 году DVD Forum (Hitachi, Matsushita (Panasonic), Mitsubishi, Philips, Pioneer, Sony, Thomson, Time Warner, Toshiba и JVC) принял стандарт на диски DVD-R, предусматривающий два размера: «нормальный» (12 см) и уменьшенный (8 см)[200]. Расшифровывалась аббревиатура DVD как Digital Video Disk – «цифровой видеодиск», однако позже стандарт новых дисков переименовали в Digital Versatile Disk – «цифровой многофункциональный диск». При разработке DVD были учтены проблемы защиты информации на этих дисках (метка региональной зоны на диске, использование специальных методов защиты видео- и аудиоинформации).

Для работы с DVD-дисками используется лазер с длиной волны 635 нм. Минимальная длина пита уменьшилась в 2 раза по сравнению с CD – до 0,4 мкм, шаг дорожек – до 0,74 мкм.

Ёмкость диска DVD первого поколения составила 4,7 ГБ, и он получил официальное наименование **DVD-5**, стандарт **DVD-9** предусматривает использование двухслойных дисков. Первый слой может быть получен механической прессовкой, а дополнительный слой нанесён напылением. Дополнительный верхний слой является полупрозрачным и не препятствует чтению нижнего слоя. Диск стандарта **DVD-9** способен хранить до 8,54 ГБ данных. Дальнейшим развитием стандартов DVD-5 и DVD-9 стали стандарты на двухсторонние диски **DVD-10** (9,4 ГБ) и **DVD-18** (17,08 ГБ).

Позже стандарт DVD-дисков был дополнен спецификацией на записываемые и перезаписываемые диски **DVD-R** и **DVD-RW**.

Стандарт на **DVD-R** разбит на три так называемых «слоя», или «уровня».

Первый из них – «физический» («Physical layer»), определяет требования к физическим размерам информационных углублений (пит) и их расположению на диске, размерам самого диска и его основных зон, длине волны лазера в режимах записи и воспроизведения, мощность его излучения и т. д.

Второй, «системный» («File system layer»), описывает файловую систему данных.

Третий – пользовательский («Application layer»), определяет формат и параметры цифрового потока, включая управляющие команды переключения режимов кодера и декодера MPEG-2, данные о размещении файлов и доступе к ним и т. д.

Первые два уровня базируются на технологии записи/воспроизведения компьютерных **DVD-R/DVD-RW**, а третий – на стандарте DVD-Video.

Существуют также диски **DVD-RAM**, которые представляют собой одно- или двусторонний диск, помещённый в пластиковый картридж. Для работы с ними необходим специальный дисковод.

В 1999 году часть производителей «отделилась», и ими был разработан формат дисков **DVD+RW** (со знаком «+»). Существенной разницы в формате представления информации на **DVD+RW**, по сравнению с **DVD-RW**, для конечных пользователей –

[200] Диск с диаметром 8 см предназначался для использования в цифровых малогабаритных видеокамерах. Позднее в этом размере стали выпускаться и CD-R и CD-RW диски. Они оказались удобны, поскольку по размеру коробки сопоставимы с 3,5" дискетами.

нет. Если ранее приводы могли поддерживать только один формат «+» или «−», то на сегодня все выпускаемые приводы поддерживают оба стандарта. Особенность формата – более высокая точность позиционирования лазерного луча, позволяющая осуществлять коррекцию данных «на лету», в реальном времени переписывая отдельные сбойные секторы диска, то есть в **DVD+RW** реализован более совершенный алгоритм коррекции ошибок. В дисках **DVD+R** применяется специальный отражающий слой с повышенной отражающей способностью.

Устройства для работы с DVD-дисками по внешнему виду не отличаются от привода CD-ROM.

Принцип записи информации на **DVD-R** и **DVD-RW** тот же, что и для CD-R и CD-RW. Процесс воспроизведения ничем не отличается от штампованного DVD, с той лишь разницей, что у последнего от поверхности отражается 45–85% световой энергии, а у DVD-RW – всего 18–30%. Поэтому не все ранее выпущенные DVD-приводы могут надёжно их считывать.

Замечание. При использовании многосессионной записи на DVD и CD дисках, по умолчанию, системой монтируется последняя записанная сессия. Может оказаться, что она предоставляется доступ только к файлам этой сессии. Старые версии файлов (из более ранних сессий) и удалённые файлы будут недоступны. Получить к ним доступ можно указав системе монтировать с диска сессию с необходимым номером. Для ФС iso9660 и UDF доступен параметр session, например:

```
# mount -t iso9660 -o session=2 /dev/sr0 /mnt
```

4.11.5.4. Дисководы Blu-Ray и HD

В 2002 году представители девяти лидирующих высокотехнологических компаний Sony, Matsushita (Panasonic), Samsung, LG, Philips, Thomson, Hitachi, Sharp и Pioneer на совместной пресс-конференции объявили о создании и продвижении нового формата оптических дисков большой ёмкости под названием **Blu-Ray Disc**. Согласно объявленной спецификации, Blu-Ray Disc (или **BD-R** и **BD-RE**) – перезаписываемый диск следующего поколения со стандартным CD/DVD-размером 12 см с максимальной ёмкостью записи на один слой и одну сторону до 27 ГБ.

Рисунок 4.73. Привод BRD-SH6B и диски BD-RE и HD DVD-R

Формат **HD DVD** был предложен компаниями Toshiba и NEC на сессии DVD Forum в августе 2003 года. В феврале 2008 года стало известно о фактической победе **Blu-Ray** над **HD DVD**: компания Toshiba сообщила о полном сворачивании работ в этом направлении. Производство фильмов и других программ на HD DVD также прекращено.

Назвать Blu-Ray и HD принципиально новыми дисками для записи нельзя – это скорее эволюция формата DVD, «подогретая» маркетологами. В новых устройствах для записи и воспроизведения информации на диск вместо инфракрасного лазера (780 нм) у CD, красного лазера (635 нм) у DVD применён синий лазер (blue-violet laser) с длиной волны 405 нанометров. Меньшая длина волны – соответственно меньшая интерференция отражённого луча, что позволяет сделать толщину дорожки данных у Blu-Ray- и HD-дисков в два раза меньше, чем у DVD (см. рис. 4.74).

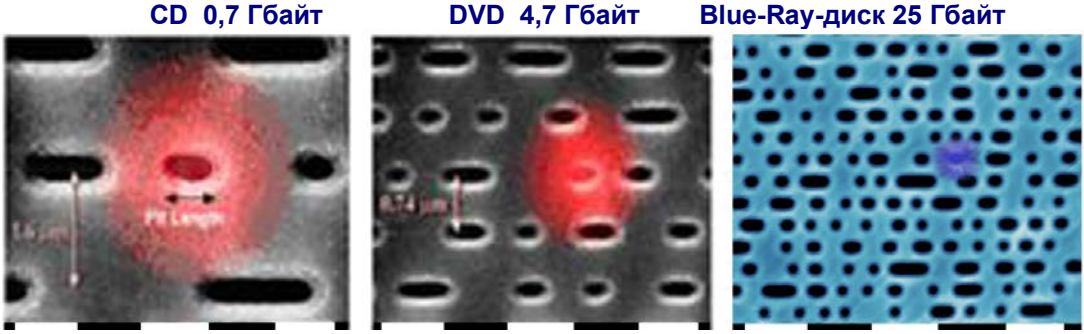


Рисунок 4.74. Поверхность CD-, DVD- и Blu-Ray-дисков

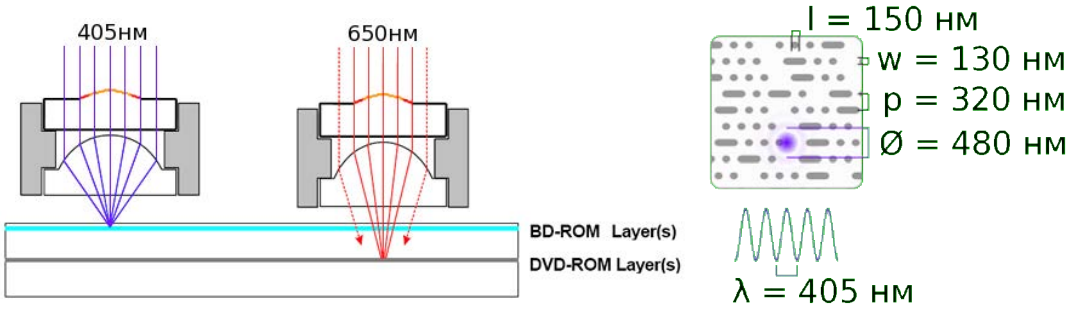


Рисунок 4.75. Характеристики записи

Покрытие дисков Blu-Ray, на которое записываются данные (optical transmittance protection layer), очень тонкое – 0,1 мм. Из этого факта следуют 3 вывода:

1) чем тоньше слой, тем меньше рассеяние отражённого луча и больше данных можно вместить на квадратный дюйм, то есть тонкий слой – это необходимость для достижения большой ёмкости диска;
2) тонкий слой позволяет без проблем сделать диск многослойным (по крайне мере двухслойным, как DVD), так как уменьшается рефракция луча отражённого от более глубокого слоя;
3) тонкий слой легко повредить, он требует защиты, поэтому у Blu-Ray-дисков над активным слоем присутствуют 3 слоя: защитный, покрывающий и особо прочный, как показано на рис. 4.76 (у HD-DVD такие же слои, как у обычного DVD).

Технологии Blu-Ray и HD создавались в первую очередь для записи, хранения и воспроизведения видео- и аудиоинформации, однако на эти диски можно записать и просто данные. Формат Blu-ray предполагает работу с видеопотоком разрешения до «1080p», звуком вплоть до «7.1» и поддержкой протокола защиты информации HDCP. Поддерживаются алгоритмы кодирования видео – MPEG-2 HD, VC1 (Video Codec 1, базируется на Windows Media Video 9) и H.264/MPEG-4 AVC, форматы звука – AC3, MPEG1, MPEG Layer2. Для цифровых видеоплееров формата Blu-Ray декодирование осуществляется аппаратно, для компьютерных приводов – программно.

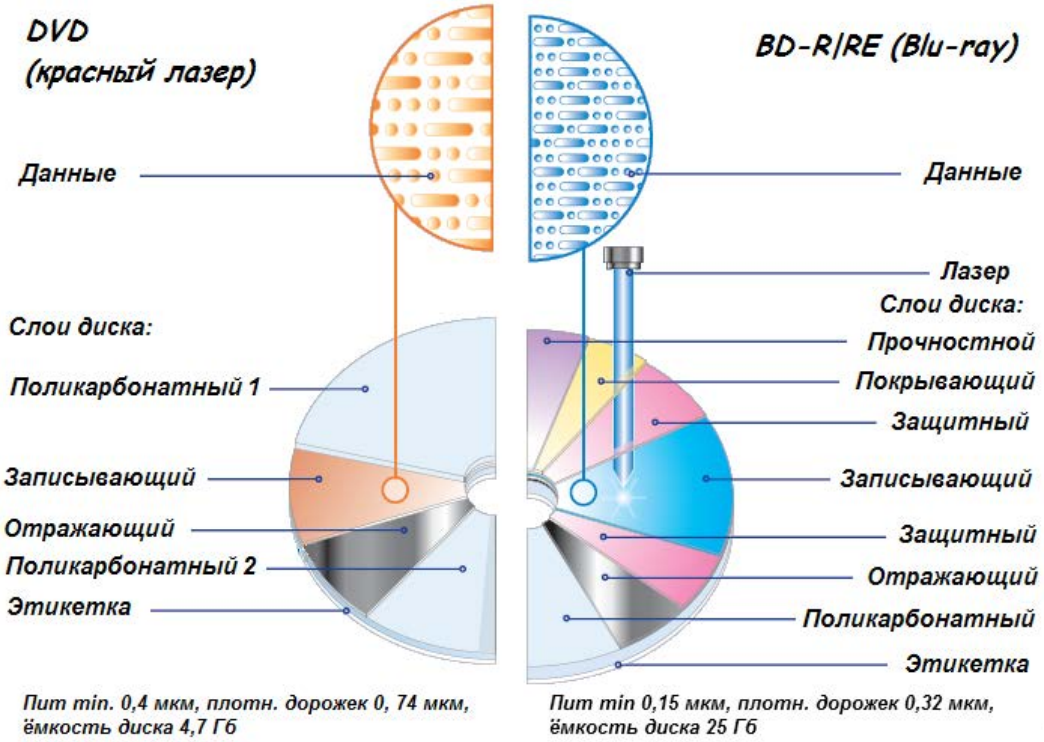


Рисунок 4.76. Диски DVD и BD-R/RE (www.verbatim.ru)

Blu-Ray-устройства имеют высокую скорость пересылки данных. Согласно спецификации, максимальная скорость пересылки данных между Blu-Ray-приводом и целевым устройством (MPEG-2-декодером или компьютером) может достигать 36 Мбит/с.

На рис. 4.73 показан внутренний дисковод с интерфейсом SATA, который читает и записывает BD-RE-, BD-R- и HD DVD-диски, а также показаны диски этих типов.

В настоящее время выпускаются BD-дисководы, которые читают и записывают также обычные DVD- и CD-диски, например **DVD±RW/CD±RW/BD-R/BD-RE/BD-R DL Pioneer BDR-209DBK**, CD: R40×/W40×/RW24×, DVD: R16×/W16×/RW6×(/RW8× двухслойный), BD: R12×/W16×(W14× двухслойный)/RW2×, SATA, стоимостью 4000 руб.

В табл. 4.14–4.17 приведены некоторые сравнительные данные по оптическим дискам и дисководам.

Таблица 4.14. Сравнительная характеристика оптических дисков

Показатель	Тип диска		
	CD	DVD	BD
Длина волны лазера, нм	780	650	405
Шаг дорожек, *p*, нм	1600	740	320
Минимальная длина пита, *l*, нм	800	400	150
Плотность записи, ГБ/дюйм^2	0,41	2,77	14,73
Объём, ГБ (один слой)	0,7	4,7	25

Таблица 4.15. Спецификации BD- и HD- дисков

Показатель	Тип диска							
	BD-R		BD-RE		HD DVD-R		HD DVD-RW	
Количество записываемых слоёв	1	2	1	2	1	2	1	2
Объём, Гб	25	50	25	50	15	30	15	30
Записываемый слой	Неорганическое вещество		Вещество, изменяющее агрегатное состояние		Неорганическое вещество		Вещество, изменяющее агрегатное состояние	
Длина волны лазера, нм	405		405		405		405	
Скорость передачи данных, Мбит/с	36		36		36,55		36,55	
Диаметр диска, мм	120		120		120		120	
Толщина диска, мм	1,2		1,2		1,2 (0,6 + 0,6)		1,2 (0,6 + 0,6)	
Покрывающий слой, мм	0,1		0,1		–		–	
Шаг дорожек, мкм	0,32		0,32		0,40		0,40	
Минимальный размер пита, мкм	0,149		0,149		0,204		0,204	

Таблица 4.16. Сравнительная характеристика стоимости оптических дисков разных типов (цены, февраль 2018 г.)

Марка	Цена, руб.
CD-R Verbatim 700Mb 52x 10 шт.	290
CD-RW CakeBox 700Mb 4–12x 10 шт.	310
DVD+R/DVD-R VERBATIM 4.7Gb 16x 10 шт.	300
DVD+R Bulk 4,7 ГБ 16x 50 шт.	750
DVD+RW/DVD-RW Verbatim 4.7 ГБ 4x 25 шт.	1290
8 см DVD+RW Verbatim Printable 1.4 ГБ 4x 10 шт.	750
DVD+R DL SlimCase 8,5 ГБ 8x 5 шт.	390
Blu-Ray 25 ГБ Verbatim BD-R 6x 10 шт.	950
Blu-Ray 25 ГБ Verbatim BD-RE, 2x 10 шт.	750
Blu-Ray 50 ГБ Verbatim BD-RE, 6x 5 шт.	2030

Таблица 4.17. Сравнительная характеристика стоимости DVD-дисководов на SATA (цены, февраль 2018 г.)

Тип привода	Марка	Цена, руб.
DVD-ROM, CD-ROM	LG DH18NS60	800
DVD±RW, CD±RW	ASUS DRW-24D5MT	1000
DVD±RW, CD±RW	Pioneer DVR-221LBK	1080
Blu-Ray (BD-ROM, DVD±RW, CD±RW)	Asus BC-12D2HT	3700
Blu-Ray (BD-RE, BD-R, DVD±RW, CD±RW)	LG BH16NS40	4050

4.11.6. Флэш-память (USB-flash и карты памяти)

О том, какая бывает flash-память, и как она работает, см. раздел 4.11.4.2. Flash-память (стр. 292). В данном разделе мы рассмотрим устройства, в которых, собственно, она применяется, за исключением SSD-накопителей.

Как мы узнали ранее, флэш-память появилась довольно давно, однако её массовое использование (в виде конечных устройств, а не отдельных микросхем) началось с широким распространением цифровых фотокамер. Сегодня производители выпускают флэш-память как устройства нескольких типов:

- **флэш-карты памяти** Compact Flash (CF), SmartMedia (SM), MultiMedia Card (MMC), SecureDigital (SD), Memory Stick PRO (MS PRO), Memory Stick (MS) и xD-Picture (xD) – для работы с ними необходимо устройство чтения флэш-карт;
- **USB-флэш-память** (так называемые «флэшки» или USB-накопители); последняя самодостаточна и не требует применения дополнительных устройств для записи и чтения информации, имеет разъём для подключения к USB-порту ПК (см. рис. 4.77).

Рисунок 4.77. USB-flash-накопитель Apacer AH324 ёмкостью 16 ГБ

Быстродействие флэш-памяти достаточно сильно варьируется в зависимости от характеристик микросхем, используемых производителем. Субъективно быстрая – USB-flash-память – скорость чтения и записи >20 МБ/с; медленная – меньше 10 МБ/с (чтение и запись). Кстати, существуют некоторые SSD-накопители с дополнительным USB-разъёмом – их, пожалуй, можно отнести к сверхбыстрой USB-flash памяти (чтение и запись от 50 МБ/с и выше).

Скорости различных флэш-карт рознятся: CF (чтение/запись до 160/120 МБ/с), SDXC (до 420/380 МБ/с), SM и xD (до 8/5 МБ/с), MMC (до 20 МБ/с), MS PRO DUO (более 100 МБ/с).

4.11.6.1. SD-карты памяти

Сегодня наиболее популярен формат карт памяти, специально разработанный для использования (в основном) в портативных устройствах: **Secure Digital Memory Card** (SD). Карты данного формата широко используются в цифровых фотоаппаратах, мобильных телефонах, КПК, MP3-плеерах, коммуникаторах и смартфонах, электронных книгах, GPS-навигаторах, автомобильных видеорегистраторах и в некоторых игровых приставках.

Существуют четыре поколения карт памяти данного формата:

SD 1.0	SD 1.1	SDHC	SDXC
от 8 МБ до 2 ГБ	до 4 ГБ	до 32 ГБ	до 2 ТБ

Совместимы слева направо, то есть старые кард-ридеры и устройства не смогут прочитать новые карты.

По физическому исполнению все поколения выпускаются в трёх типоразмерах (см. рис. 4.79). Электрически совместимы, поэтому многие microSD-карты продаются вместе с небольшими адаптерами (см. рис. 4.78).

Рисунок 4.78. Адаптер карты microSDXC слева в сравнении

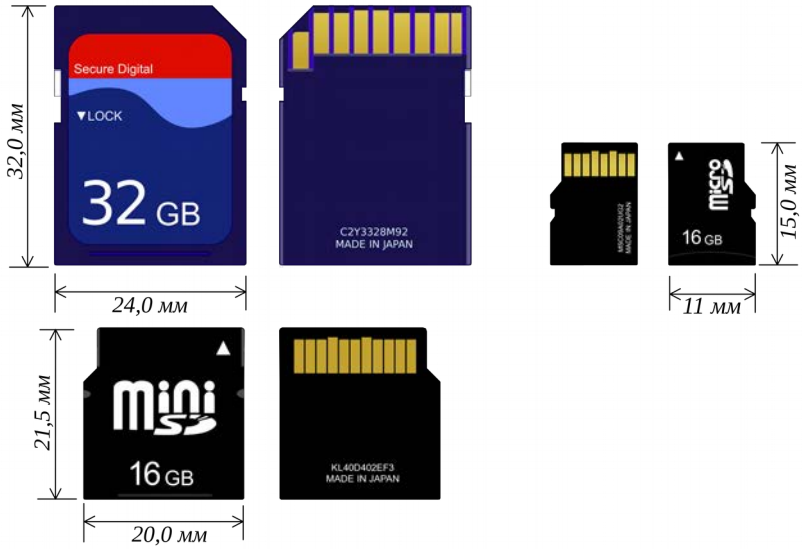


Рисунок 4.79. Три типоразмера SD-карт памяти (обычный, mini и micro)

Для удобства потребителей SD Card Association ввела классификацию скоростных характеристик карт и устройств и поделила их на классы **Class 2**, **Class 4**, **Class 6** и **Class 10**, указывающие на то, что данные устройства позволяют записывать данные со скоростью МБ/с не менее, чем номер класса. На практике пиковая скорость записи может быть до двух раз выше. Некоторые производители стали маркировать свои карты как несуществующий Class 16.

Более-менее ясная классификация появилась отчасти из-за потребностей потребителей в записи видеопотоков с разрешением 1920×1080, а сейчас и 4096×2160. При первом разрешении, если использовать алгоритмы компрессии, скорость видеопотока составляет – 24 Мбит/с (3 Мбайт/с), следовательно, требуется карта класса не ниже 4. В то же время реальные тесты не редко показывают, что скорость видеопотока в Full HD-качестве может достигать 50 Мбит/сек, а в 4K и того более. Поэтому для современных фотокамер и автомобильных видеорегистраторов рекомендуется приобретать карты «с запасом» по скорости, то есть не ниже Class 10.

Поскольку старая шина данных не позволяла более увеличивать скорость, была придумала новая – **UHS** (Ultra High Speed) Bus. Технические спецификации 3.01 и 4.00 определили два тип шин **UHS-I** и **UHS-II**, со скоростями обмена данными по интерфейсу (не путайте со скоростью записи) – не ниже 50 МБ/с (до 104 МБ/с) и не ниже 156 МБ/с (до 312 МБ/с), соответственно. Обозначение используемого типа шины на картах памяти производят знаками «**I**» или «**II**». Новая спецификация, определяет не только скорость ввода-вывода для карт памяти, но и протокол обмена данными, немного отличающийся от стандартного. Повышение скорости работы происходит посредством использования четырёхбитового режима шины SD (против однобитового) за счёт дополнительных контактов. Спецификация требует обратной совместимости карт и контроллеров UHS с более ранними стандартами, однако, не смотря на это использование новых шин **UHS-I** и **UHS-II** возможно лишь для карт SDHC и SHXC. Карты, работающие по шине **UHS-I** маркируются как UHS Speed Class 1 (U1) и UHS Speed Class 3 (U3), минимальная скорость записи 10 МБ/с и 30 МБ/с, соответственно.

Карты стандарта UHS-II содержат два ряда контактов – 17 для обычной карты и 16 для microSD. Например, карты SDHC SanDisk Extreme Pro с UHS-II U3 обеспечивают скорость чтения 420 МБ/с, а скорость записи 380 МБ/с.

Разброс ценовых характеристик на карты памяти и USB-носители можно оценить по табл. 4.18.

Таблица 4.18. Карты flash-памяти и USB-flash-память (цены на февраль 2018 г.)

Тип карты, разъём	Объём, Модель	Цена, руб.
Compact Flash (CF)	16 ГБ Transcend	1468
Compact Flash (CF)	32 ГБ Transcend	1900
Compact Flash (CF)	64 ГБ Transcend 1000x	5300
SDHC	8Gb Transcend Class 10 UHS-I	570
SDHC	16 ГБ Kingston Class 10 UHS-I	620
SDHC	32 ГБ Transcend Class 10 UHS-I	1380
SDHC	32 ГБ SanDisk Ultra Android Class 10 UHS-I 48 МБ/с	1070
SDXC	128 ГБ Transcend Class 10 UHS-I	5510

microSDHC	8 ГБ Apacer Class 2	370
microSDHC	16 ГБ TransFlash Class 10	640
microSDHC	32 ГБ TransFlash Class 10 UHS-I	1680
microSDXC	64 ГБ TransFlash Class 10 UHS-I	2750
microSDXC	128 ГБ TransFlash Class 10 UHS-I	5820
microSDXC	256 ГБ Samsung Class 10 UHS-I 80 МБ/с	8710
USB-flash	8 ГБ Transcend JetFlash 790	430
USB-flash	16 ГБ Transcend JetFlash 790	540
USB-flash	32 ГБ Transcend JetFlash 600	1400
USB-flash	64 ГБ Transcend JetFlash 350	1490
USB-flash, USB 3.0	128 ГБ Transcend JetFlash 760	3450
USB-flash, USB 3.1	256 ГБ SanDisk Extreme PRO 420/380 МБ/с	8 230

В качестве скоростного и универсального (не требующего применения дополнительных устройств) накопителя для переноса достаточно большого объёма данных (на сегодня до 256 Гбайт) удобно использовать USB-флэш-память.

В настоящее время некоторые ПК и ноутбуки комплектуются устройствами чтения флэш-карт – card reader. Можно также приобрести переносное устройство – внешний кард-ридер для всех типов карт (например, All-In-One REKi CF/MD/SM/MMC/SD/MS/SMC/TransFlash USB2.0, цена около четырёх сотен рублей), поэтому флэш-карты получают всё большее распространение.

Преимущества флэш-памяти, по сравнению с другими средствами переноса и хранения данных, очевидны – высокая надёжность и ударопрочность (в результате отсутствия движущихся компонентов и простоты механической конструкции носителей и накопителей), малое энергопотребление и компактность. Однако у неё есть и недостатки: ограниченное число циклов перезаписи (в настоящее время до 1 млн) и относительно медленная работа (у дешёвых моделей).

Замечание. Нередко можно слышать вопрос: Почему я не могу записать файл размером 4,7 ГБ на флэшку, если она объёмом 8 ГБ или больше? Ответ на данный вопрос кроется в том, какую файловую систему вы используете на вашем носителе. USB-flash, как и карты памяти, подобно жёстким дискам, содержат таблицы разделов, разбиваются и форматируются. С них даже можно загрузиться (главное, чтобы BIOS поддерживал такой фокус). По умолчанию используется файловая система FAT32 со всеми её ограничениями. Подробнее о файловых системах и их ограничениях см. в главе 5 *«Программное обеспечение»*.

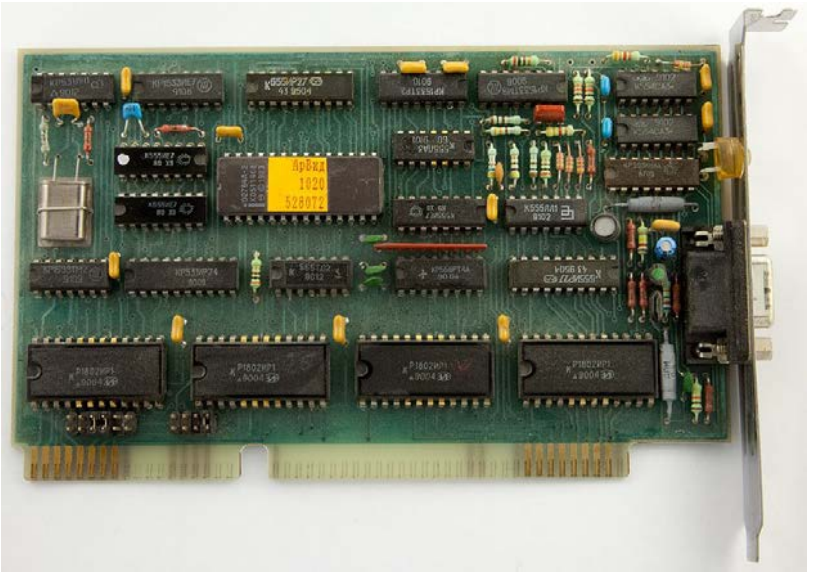
4.11.7. Стример

Стример – устройство для резервного копирования данных (backup) с жёстких дисков на магнитные ленты.

Считается, что жёсткие диски не обеспечивают должного уровня безопасности длительного хранения информации. Для более надёжного архивного хранения данных рекомендуется создавать резервные копии. Сегодня парк устройств, позволяющих делать резервные копии, разнообразен [201], а вот ранее, до появления CD и DVD, применя-

[201] Вплоть до их «отсутствия» – при выгрузке данных в «облако». Мы не берёмся обсуждать вопросы надёжности и конфиденциальности таких решений, но обращаем внимание читателей на то, что указанный сервис становится всё более популярным. Достаточно назвать Яндекс-диск, Drop-box, синхронизации контактов Android и iOS-устройств с различными облаками и другие сервисы.

лись лишь стримеры. В 90-е популярным для домашнего использования был стример «Арвид», позволявший записывать на 3-часовую VHS-видеокассету на обычном видеомагнитофоне до 4 ГБ информации, при среднем размере жёстких дисков менее сотни мегабайт на то время. Сейчас стримеры – пожалуй, удел больших корпораций и центров хранения данных.



Основной задачей резервного копирования является обеспечение сохранности оперативной информации, представляющей большое значение для любых предприятий. Потери организаций, работающих с банковскими счетами клиентов, от одного часа простоя, вызванного временным сбоем в работе компьютерного оборудования, могут исчисляться миллионами долларов. Потери, вызванные полной утратой коммерческой информации, могут привести к полному краху предприятия.

Ежедневное проведение резервного копирования (а лучше два, три раза в день), хоть и является наилучшим выходом с точки зрения безопасности данных, но для хранения нескольких копий требуется слишком много места на магнитных лентах. Поэтому, кроме полного резервного копирования, применяется инкрементальное и дифференциальное копирование. В этом случае после первоначального полного копирования всей необходимой информации выполняется копирование только новой и измененной информации. Для инкрементального метода полное копирование периодически повторяется.

Для небольших локальных сетей (до 20 компьютеров) возможно применение одного стримера (ленточного привода) со сменными ленточными картриджами. Такие устройства характеризуются относительно небольшими скоростными и ёмкостными характеристиками (вместительность одного картриджа от 40 до 400 ГБ в зависимости от длины ленты и степени сжатия информации при копировании) и невысокой стоимостью. Скорость чтения/записи таких устройств находится в пределах 6–12 МБ/сек. Недостатком подобных устройств является необходимость вручную заменять картриджи по мере их заполнения.

Для средних предприятий (до 100 компьютеров в сети) существуют автоматические ленточные библиотеки, способные вмещать до 10 картриджей одновременно. Такая библиотека самостоятельно выбирает необходимый картридж и вставляет его в ленточный привод для дальнейшей работы. Чтение/запись информации производится только с одного картриджа одновременно со скоростью от 21 до 43 ГБ/час. Общая ёмкость загруженной библиотеки составляет до 1,6 ТБ данных. Такие устройства могут

быть встроены в серверный шкаф, являются полностью автономными и не требуют внимания со стороны администратора.

В ленточные библиотеки более высокого уровня имеется возможность установить сразу несколько приводов для обеспечения необходимой скорости чтения/записи информации. Например, в ленточную библиотеку Scalar100 могут быть установлены 8 ленточных приводов для обеспечения теоретической скорости чтения/записи информации до 560 МБ/сек с общей ёмкостью хранимой информации до 28,8 ТБ. На практике скорость чтения/записи таких систем ограничивается только пропускной способностью существующих каналов связи.

Наиболее крупные распределённые системы (более 250 компьютеров) при объёмах информации, подлежащей резервному копированию, более 1 ТБ/сутки используют виртуальные ленточные библиотеки. В таких системах используется схема **D2D2T** (Disk To Disk To Tape), а не Диск – Лента (**D2T** – Disk To Tape). Такая система представляет собой дисковый массив, напрямую подключенный к собственной ленточной системе. И массив жёстких дисков, и ленточная система объединены в одном корпусе и соединяются между собой быстродействующими каналами связи (обычно SCSI).

В этой системе процесс резервного копирования происходит следующим образом: информация, подлежащая резервному копированию, передаётся с устройств своего непосредственного хранения на дисковую часть гибридной системы хранения, что происходит достаточно быстро, а следовательно, меньше загружает каналы передачи данных. Полученная информация далее переписывается на ленты, не загружая внешние дисковые системы и обеспечивая должный уровень надёжности хранения информации.

К наиболее распространённым форматам стримеров среднего класса относятся **DDS, DAT, DLT, SuperDLT, DLT VS80, LTO, IBM 3592, Ultrium** и прочие.

Технология DDS – в накопителях этой группы используется 4-миллиметровая лента формата DDS (Digital Data Storage), часто ошибочно называемого DAT (Digital Audio Tape). Термин «DDS» одобрен DDS Manufacturers Group и относится к исходному формату DAT, допускающему хранение до 4 Гбайт данных со сжатием или 2 Гбайт без сжатия. В новых четырёхмиллиметровых накопителях используется стандарт DDS-4, обеспечивающий ёмкость в 20 Гбайт без сжатия или около 40 Гбайт со сжатием и скорость передачи данных 40 Мбит/с.

Технология DAT развивается с 1989 г. Носитель – картридж величиной чуть меньше аудиокассеты (DAT-картриджи используются и для высококачественной записи музыки, что следует из их названия – Digital Audio Tape). Запись производится в одном направлении по всей ширине ленты. Современный стандарт DAT72 позволяет хранить на одном картридже DDS 36/72 [202] ГБ информации и производить чтение/запись со скоростью 20/40* МБ/с. Стримеры DAT поддерживают технологию One Button Disaster Recovery (восстановление состояния системы после полной потери содержимого дисков без необходимости форматирования новых дисков, установки ОС и т. п.). Технология DAT является абсолютным чемпионом: свыше половины используемых сейчас в мире накопителей на магнитной ленте – это стримеры того или иного формата DAT.

Технология DLT появилась в 1985 г. – корпорация DEC выпустила стример TK-50, в результате развития которого появился формат Digital Linear Tape. В настоящее

[202] Вторая цифра – объём сжатой информации.

время обладателем прав и основным разработчиком технологии DLT является компания Quantum. Носитель – квадратный картридж 10×10 см. Запись производится в двух направлениях последовательно челночным способом (сначала вперёд, затем назад, затем опять вперёд и т. д. до полного использования ширины ленты). Современный стандарт DLT8000 позволяет хранить на одном картридже DLT IV 40/80* ГБ информации и производить чтение/запись со скоростью 6/12* МБ/с. Стримеры DLT8000 могут также читать/писать картриджи предыдущих форматов DLT IIIXT и DLT III. Среднее время доступа к файлу – 50 с. Стримеры могут работать со 100% загрузкой (24 часа в сутки), что позволяет строить на их основе библиотеки магнитных лент.

Технология SuperDLT позволяет записывать на SDLT-картридж 160/320* ГБ при скорости обмена 42/85* МБ/с. Устройство может читать DLT IV картриджи, записанные на DLT8000. Среднее время доступа к файлу – 70 с. Стример может работать со 100%-ной загрузкой.

Технология DLT VS80 позволяет хранить на одном картридже 320/640* ГБ информации.

Технология Ultrium является первой реализацией стандарта **LTO** (Linear Tape Open), совместного проекта HP, IBM и Seagate. Картридж Ultrium Generation 1 по размеру почти не отличается от DLT. В картридж встроен микрочип, куда записывается информация о записанных файлах и их местонахождении, а также статистика использования картриджа. Ultrium 800 позволяет записывать на картридж 400/800* ГБ при скорости обмена 40/80* МБ/с. Стример может работать со 100% загрузкой.

Итак, максимальная ёмкость картриджей для разных технологий составляет, Гбайт:

- **DDS** – 40,
- **SuperDLT** – 320,
- **DLT** – 80,
- **DAT** – 72,
- **DLT VS80** – 640,
- **Ultrium** – 800.

Технология IBM 3592. Компания IBM в настоящее время поставляется на рынок, помимо оборудования LTO (Linear Tape-Open), стримеры собственного закрытого стандарта IBM 3592 (Jaguar), а также совместимые ленточные библиотеки. Это оборудование используется в серверах и мейнфреймах. Картриджи имеют физическую ёмкость до 4 Тбайт. В 2014 году фирмой IBM был представлен стример TS1150 с кассетами объёмом 10 ТБ.

Интересный факт: стандартная для ОС UNIX и Linux утилита упаковки файлов tar получила своё название от двух английских слов Tape и ARchive, потому как первоначально была придумана и использовалась для записи данных именно на стример. Она и сейчас не потеряла этой функции, но большинство знают команду tar как стандарт упаковки файлов, проверенный десятилетиями.

Замечание. Термин стример имеет ещё одно значение. Последнее время стало модно проводить онлайн-трансляции каких-либо мероприятий и событий. Стример – это человек, который ведёт онлайн-эфиры в интернете. Стримить можно всё: приготовление еды, рукоделие, путешествия, или процесс игры в компьютерные игры. Это могут быть даже родители наблюдающие за утренником в детском саду. Самые популярные платформы для стрима это: YouTube, Twitch, а так же Вконтакте, GoodGame и другие. Аудитория стримеров обычно состоит из молодых людей, которым интересно наблюдать за эмоциями ведущего, комментариями, общаться в чате. Обычно стримит один человек, но бывает и команда, состоящая из продюсера, ведущего, оператора.

4.12. Устройства ввода информации

К устройствам ввода информации персональных компьютеров относятся прежде всего клавиатура и «мышь», которыми любой пользователь постоянно пользуется. Реже используют сканер, дигитайзер, веб-камеру и музыкальные устройства ввода. Всякие световые перья давно уже ушли в историю, сейчас модно водить пальцем по экрану, а то и всеми пятью.

4.12.1. Клавиатура

Клавиатура – устройство с большим числом кнопок, позволяющее пользователю взаимодействовать с ПК путём их нажатия. Современные варианты имеют более сотни клавиш (102–105, в зависимости от модели). В России клавиши обычно подписаны изображениями латинского и кириллического алфавитов, цифрами и другими символами (для переключения раскладок по умолчанию используется наиболее удачное сочетание клавиш Левый Alt+Shift, реже два Shift'a вместе).

В других странах можно купить клавиатуры с нанесёнными местными алфавитами, где вместо кириллицы может быть даже арабская вязь. Изображение на клавиши обычно наносится заводским способом и не стирается при интенсивном нажатии клавиш несколько лет. Наиболее долговечной является лазерная гравировка [203] (типично для изделий фирмы Cherry), которая может служить десятилетиями. Приклеиваемые за 50–100 рублей наклейки обычно служат недолго, при интенсивном нажатии клавиш от недели до полугода.

Для подключения клавиатуры к ПК используется разъём PS/2 или USB-порт. Ранние версии клавиатур использовали электрически совместимый с PS/2 разъём большего размера (DIN 5-pin), его вы вряд ли увидите, разве что в музее. Также клавиатуры для IBM PC XT имели даже отличный от IBM PC AT протокол, для чего под клавиатурой можно было найти небольшой переключатель режима работы.

Первоначальной целью клавиатуры было – набирать тексты и команды, тем самым обеспечивать взаимодействие пользователя с ПК. С изменением интерфейса операционных систем эта функция отходит на второй план, особенно когда используются мониторы, по которым можно водить пальцами и нажимать прямо на изображение, когда вводятся голосовые интерфейсы.

Клавиатуру в карман не положишь – она большая. Если уже в 2013 году в некоторых «продвинутых» школах первоклашек учили не письму, а тому, как печатать 10 пальцами, то завтра, возможно сиё, не одобряемое авторами новшество (потому как считаем, что писать надо уметь всем) будет заменено обучением навыкам «разговора» с компьютером голосом, жестами и мимикой. Не секрет, что уже продаются мобильные телефоны с передними камерами, способные следить за глазами пользователя. Автоматическая прокрутка текстов и управление устройством движением глаз – уже не новшество.

Вторичная цель клавиатуры, как и всего окружающего нас нынче, – приносить эстетическое удовольствие пользователю от её использования, именно этим можно объяснить появление на рынке большого числа бесполезных, с нашей точки зрения,

[203] Нанесение лазерной гравировки на предметы сейчас не так дорого, вопрос лишь в том, чтобы найти фирму и специалиста, способного ровно положить образец перед лазером. Вкупе с полусотней клавиш указанное занятие может оказаться не сильно дешевле покупки новой клавиатуры заводского исполнения.

клавиатур. Что думает пользователь или вы, читатель: удобно и эргономично? Совсем не обязательно, что эти два параметра будут вместе.

Удобно – это то, к чему мы привыкли. К хорошему привыкаешь быстро, к плохому дольше. Эргономично – это когда с меньшими затратами совершается больше полезных действий.

Рисунок. 4.80. Беспроводной набор от Logitech
(Выглядит «вызывающе» но не очень удобен для быстрой печати 10-пальцевым методом из-за нестандартного расположения функциональных клавиш)

Клавиатура выше (см. рис. 4.80) есть пример человеческой глупости. Объясним почему. Скорее всего, вы не будете носить свою клавиатуру с собой постоянно, поэтому, привыкнув работать с ней дома, не сможете работать и печатать с той же скоростью на любой другой, в гостях, на работе, в школе, вузе и других местах, лишь потому, что ваши пальцы будут искать клавиши так, как они запомнили их в последний раз.

Также маловероятно, что на нестандартной клавиатуре вами будут набираться тексты со скоростью более 200 знаков в минуту. Залог высокой скорости печати (вслепую, естественно, и без ошибок) – это механическая память пальцев ваших рук, то есть то, куда какой палец должен нажимать.

4.12.1.1. 10-пальцевый метод быстрой печати

Эргономика (продуманность) здесь проявляется в том, что клавиши, предназначенные для правой руки, находятся ближе к ней, а для левой – рядом с ней. Перемещения пальцев к часто используемым клавишам минимизированы, а расположение кнопок оптимально. Классическая клавиатура не претерпевает существенных изменений десятилетиями лишь потому, что это действительно разумное решение.

Также обратите внимание на расположения клавиш и специальные выступы у клавиш «F, A», «J, O» и «5». Такие выступы есть практически на всех клавиатурах и служат маяками (ориентиром) правильной постановки рук, чтобы на ощупь пользователь мог понять, что его руки находятся на нужном месте, а не съехали вбок.

Проведите эксперимент: возьмите вашу клавиатуру (или ту, которой часто пользуетесь), и поставьте указательные пальцы на клавиши с маяками и постарайтесь понажимать различные клавиши, по возможности не отрывая рук, а точнее ваших указательных пальцев от маяков. Если на вашей клавиатуре какой-то «дурачок-дизайнер» укоротил часто используемые клавиши «Shift» справа или слева, либо «Backspace», то вы быстро заметите, что не можете дотянуться до них мизинцами рук. Либо в руках будут неприятные ощущения, либо рука естественным образом будет приподнята и оторвана. Если же получается дотянуться без неприятных ощущений – значит клавиатура вам подходит, если руку непременно надо снять – откажитесь от идеи печатать вслепую или замените клавиатуру на ту, до клавиш которой вы дотягиваетесь без труда.

Если вы читаете далее и не потеряли интереса, то мы объясним, как можно научиться быстро печатать без чьей-либо помощи. Поверьте, это возможно, так как не одна тысяча машинисток была обучена подобным образом в старые добрые времена. Следующим заданием для читателей будет – нарисовать вашу клавиатуру на бумаге желательно в естественный размер (в масштабе 1:1) и обозначить, какой палец какую клавишу должен нажимать (см. рис. 4.81).

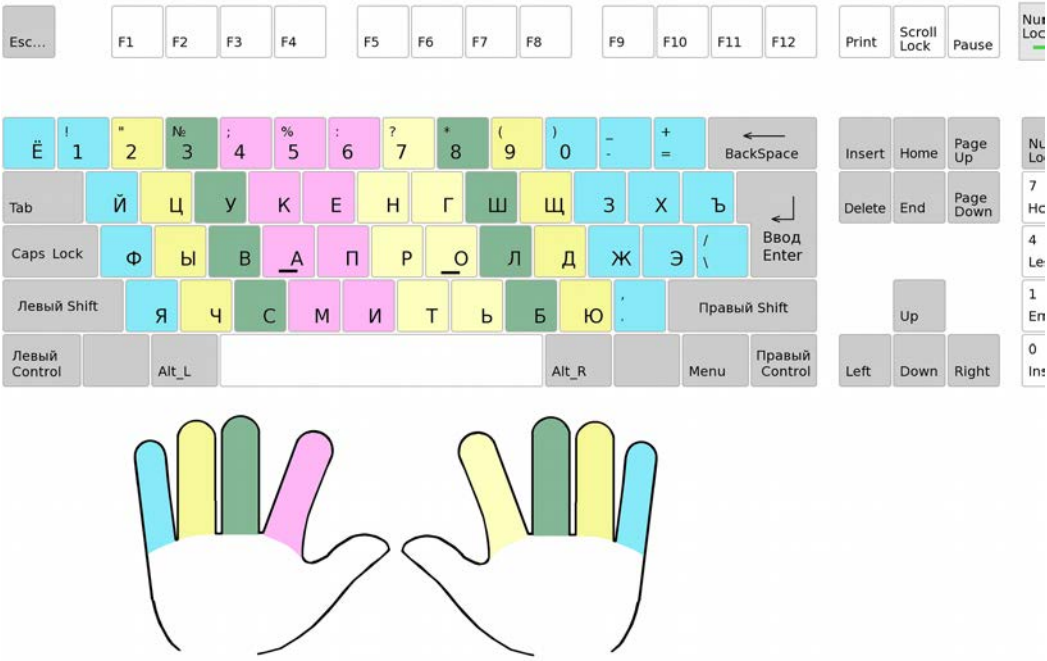


Рисунок. 4.81. Распределение клавиш по пальцам рук при 10-пальцевом методе печати вслепую

Практически любой учебник машинописи обяжет вас это сделать либо будет содержать в себе аналогичное рис. 4.81 изображение. Если нарисовали – вы готовы приступить к обучению. Главное – набраться терпения и соблюдать два правила:

1. Для каждого пальца своя клавиша.
2. На клавиатуру, где стоят ваши пальцы, смотреть запрещается [204].

[204] Сделанный вами рисунок можно (и даже нужно) использовать. Для удобства можете приклеить его к краю монитора – будет вам подсказкой. Если обратите внимание, то многие программы, обучающие печа-

Обещаем вам, что, честно соблюдая указанные два правила, через пару месяцев вы будете печатать вслепую. Возможно, первую неделю, особенно первый день, вы будете «плеваться», но в последующем, за месяц вы приучите себя печатать правильно и будете это делать быстро, не глядя на клавиши.

По мере того как вы научитесь быстро и без ошибок печатать на великом могучем русском языке, можете перейти к осваиванию других раскладок (английской, немецкой и др.), а также правой вспомогательной части клавиатуры (см. рис. 4.82). (Со стрелочками и функциональными клавишами обычно проблем не возникает.)

Если у вас недостаточно стеничности (упорства и усидчивости) в освоении нового метода печати на клавиатуре, можем посоветовать использовать различные обучающие программы, как бесплатные, так и коммерческие с демоверсиями [205]. Возможно, подстреливая или съедая падающие буквы, вам будет веселее обучаться.

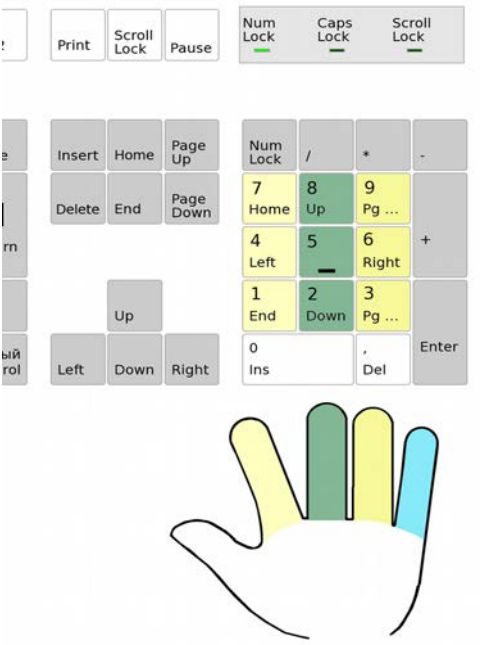


Рисунок. 4.82. Распределение пальцев правой руки по дополнительным клавишам. Клавиша с маяком – «5» – приходится на средний палец

Существенной разницы в выборе программы нет, так как большинство из них содержит примерно одни и те же упражнения по постановке рук, взятые из учебников по машинописи прошлого века. Купить программу, использовать бесплатную или прочитать книжку из библиотеки – решать вам.

4.12.1.2. Дополнительные клавиши

Существуют клавиатуры с дополнительными клавишами, например «запуск интернет-браузера», «запуск почты», «запуск калькулятора», ...универсального проигрывателя и других выбираемых пользователем приложений, клавиши регулирования громкости и прочее. Единого стандарта по дополнительным клавишам нет. В своё время разработчики ЭВМ из фирмы IBM и так придумали двенадцать функциональных клавиш F1–F12, посчитав, что этого будет более чем достаточно. Многие пользователи избегают их использовать, возможно, не зная их предназначения или как в той или иной программе переназначить их функционал. Вот и появляются дополнительные

ти 10-пальцевым методом, обязательно рисуют клавиатуру (обычно в нижней части экрана) при выполнении заданий.

[205] Наиболее популярная «СОЛО на клавиатуре»: http://ergosolo.ru/products/.

ряды специальных кнопок, являющиеся скорее данью моде нынешнего образа жизни. Единственное, где их использование дополнительных клавиш оправдано, – это склады, кассы магазинов, медицинское и другое специализированное оборудование. Обычно такие клавиатуры выглядят немного по-иному, могут иметь встроенный считыватель пластиковых карт и не продаются в обычных компьютерных магазинах.

Любители различных игр могут найти для себя так называемые «геймеркие» варианты клавиатур с улучшенными и программируемыми кнопками и клавишами, даже с небольшими жидкокристаллическими экранчиками и подсветкой.

Некоторые дополнительные клавиши могут определяться операционной системой автоматически, а для поддержки другой части клавиш придётся устанавливать программное обеспечение, идущее в комплекте или доступное на сайте производителя.

4.12.1.3. Ё – это тоже буква русского языка

Необоснованно часто буква «ё» в русском языке подвергается притеснениям, замене на букву «е», хотя использование её в учебниках, текстах для иностранцев и многих местах просто обязательно. Вместо того чтобы поддерживать свою культуру, мы зачем-то пытаемся ассимилироваться в европейскую... это неправильный путь. Возможно, с течением времени она, как и буква «ѣ» (ять), эволюционно исчезнет, но пока она есть в алфавите, а государственный язык, определяемый п. 1 ст. 68 Конституции РФ, – русский, её следует использовать.

В машинописных раскладках буква «ё» располагалась под правым мизинцем, но позднее её переместили в левый верхний угол клавиатуры. То, что она не поддерживается современными компьютерами и шрифтами, – миф и ложь чистой воды.

Следующий переезд клавиши произошёл с появлением экранных клавиатур на планшетах и телефонах, где для отображения её не нашлось места, но и там она есть, хотя и сразу не видна. Например, в операционной системе iOS для печати буквы «ё» следует нажать пальцем на клавишу «е» и подержать несколько секунд, после чего вы увидите дополнительные кнопки, доступные для набора (см. рис. 4.83). Попробуйте подержать другие кнопки, не у всех, но у части из них вы обнаружите для себя новые возможности.

Рисунок. 4.83. Как набрать клавишу «ё» на планшете или телефоне – нажмите и удерживайте клавишу «е»

4.12.2. Компьютерная мышь

Манипулятор мышь – инструмент для работы с объектами на рабочем столе операционной системы. Используется для выбора нужного окна, файла, позиции в меню, в тексте, поля на экранной форме и прочего, для открытия файлов (двойной или одиночный щелчок левой кнопки), вызова контекстного меню (правая кнопка), прокрутки списка или текста (колёсико мышки). Первоначально мышь была механической, содержала небольшой шарик. При движении мыши по столу шарик вращался, увлекая за собой во вращение датчики. Так компьютер понимал, что пользователь сдвинул мышь (и свою руку тоже) в ту или иную сторону. Периодически приходилось шарик и оси датчиков очищать от пыли, даже продавались специальные комплекты из ватных дисков и моющего средства. С развитием полупроводниковой оптики от механического варианта отказались в пользу светодиодов и лазеров. Они точнее, легче, и их не надо постоянно чистить. В настоящее время используются оптические мыши двух типов – светодиодные и лазерные.

Рисунок 4.84. Проводная мышь с колесом прокрутки

Несмотря на то что оба вида мышей оптические, оптическими обычно называют светодиодные, так как их светодиод светится в видимом для нас диапазоне – красный цвет, а вот лазерные по большей части используют невидимый для нас инфракрасный диапазон. Если вы учтёте, что в какой-то мере лазер и светодиод – это примерно одно и то же [206], то заметить разницу, кроме как в названии, вряд ли сможете, разве что лазерные мыши позволяют отслеживать передвижения более точно и могут работать на всевозможных поверхностях (стекло и прочее), на которых светодиодные собратья «могут отказаться» ездить [207].

Первые мыши использовали для подключения к компьютеру COM-порты. Сейчас купить такую мышку можно разве что б/у. На таких мышках не было колёсика прокрутки «scroll». Следующий эволюционный разъём для подключения мышек – это PS/2. В эпоху PS/2 появился scroll и всевозможные переходники PS/2-USB, и наоборот. Наличие переходника не гарантирует, что любая мышка с ним заработает. Последняя должна уметь работать с обоими видами портов на уровне протокола.

Все современные мыши (как светодиодные, так и лазерные) обычно подключаются к компьютеру через USB-порт. Существуют и беспроводные варианты, но тогда к

[206] Грубо полупроводниковый лазер – это есть полупроводниковый диод стенки которого хорошо отполированы и являются отражателями (резонаторами лазера).

[207] Возможно, отсюда и пошёл анекдот: бизнесмен производит проверку сметы нового офиса. Процессор такой-то... это хорошо, быстро.. память... о... это достаточно.... монитор цветной... а это что такое?! Коврик для мыши?! Вы бы ещё тапочки для тараканов заказали!.. (Впрочем, механические мыши тоже частенько предпочитали использование ковриков.)

USB подключается приёмопередатчик радиосигнала. В качестве протокола может использоваться стандартный bluetooth, а может быть «свой» закрытый протокол.

Главная техническая характеристика мыши – разрешение оптического сенсора, которое измеряется в точках на дюйм (dpi или cpi – count per inch) и может составлять 600–3200 dpi.

Существуют беспроводные комплекты клавиатура + мышь (см. рис. 4.80), такой набор комплектуется одним приёмником радиосигнала. Часто он же служит зарядным устройством для аккумуляторов, которые являются источниками питания для беспроводных устройств.

Замечание. Если к компьютеру подключить несколько мышек, то они будут работать параллельно. При одновременном противоречивом движении мышей курсор будет или оставаться на месте или непредсказуемо перемещаться. В ОС Linux обычные пользователи могут выделить каждой мыши по курсору[208]. На экране их будет несколько, все они будут рабочие и каждый будет управляться своим устройством. Сделать это можно в три команды:

Посмотрите группы и номера устройств

```
$ xinput list
```

Создайте ещё одну группу «mouse2» для новой мыши

```
$ xinput create-master mouse2
```

Переподключите мышь с id=11 к группе mouse2 (id=13)

```
$ xinput reattach 11 13
```

4.12.2.1. Touch pad

На некоторых ноутбуках можно встретить около клавиатуры небольшое окошко или область, двигая пальцем или несколькими по которой, можно полностью эмулировать движения мышью. Так называемый ноутбучный вариант мыши – touch pad. Подобно о том, как он работает, см. в разделе 4.12.5. Сенсорный монитор (стр. 331). По сути, область, где вы двигаете пальцами, есть небольшой сенсорный монитор.

4.12.3. Сканер

Сканер – устройство для автоматизированного считывания (сканирования) графической информации.

Принцип действия сканера заключается в следующем: установленный в нём источник света облучает[209] сканируемый объект, а оптическая система сканера воспринимает отраженный от объекта световой поток, который с помощью программы сканирования преобразуется в цифровую форму.

Сканеры разделяются на два основных типа:

- настольные,
- ручные.

[208] Linux :: два, три, пять… указателей мыши // http://habrahabr.ru/post/165385/.
[209] Существуют сканеры для сканирования слайдов и плёнок, работающие на просвет.

Существуют три разновидности настольных сканеров:

- планшетные,
- рулонные,
- проекционные.

Основной тип сканера для небольших форматов (А4) в настоящее время – настольный планшетный.

Рисунок 4.85. Сканер HP scanjet 4600

Характеристикой качества сканеров является разрешение, измеряемое числом точек на дюйм – dpi. Хорошие сканеры имеют разрешение 1200–4800 dpi. В основном современные планшетные сканеры цветные с 24–36-битным представлением цвета.

В **планшетном сканере** сканирующая каретка – блок, объединяющий в себе источник света и оптическую систему (ширина её обычно 210–220 мм) перемещается относительно бумаги с помощью шагового двигателя. Оптическая система состоит из одного или нескольких зеркал, линзы (либо призмы) и светочувствительной матрицы. Матрица содержит три линейки светоприёмных элементов (по одной линейке на каждый из базовых цветов – красный, зелёный, синий) и оптико-электронный преобразователь. За один такт работы сканера матрица «фотографирует» очередной горизонтальный фрагмент (линию, строку) оригинала. При этом три «разноцветных» элемента матрицы обеспечивают формирование одного пикселя будущего отсканированного изображения. Для сканирования изображения необходимо открыть крышку сканера, положить сканируемый лист на основание изображением вниз (или вверх для прозрачных сканеров HP scanjet 4600, см. рис. 4.85), после чего опустить крышку. Дальнейшее управление процессом сканирования осуществляется с клавиатуры компьютера – с использованием программы сканирования, поставляемой со сканером. В программе возможна настройка многочисленных параметров сканирования. Конструкция планшетного сканера позволяет сканировать не только отдельные листы, но и страницы толстого журнала или книги.

Для примера в табл. 4.19 приведена характеристики сканера HP scanjet 4600.

В **рулонных сканерах** отдельные листы документов протягиваются вращающимся валиком мимо закреплённой неподвижно сканирующей головки. К рулонному типу относятся широкоформатные сканеры с шириной области сканирования от 635 до 1372 мм, например Contex Cougar SX 25" или CalComp ScanPlus IV 954 C (1372 мм). Рулонный сканер также используется в большинстве компактных факсимильных аппаратов.

Проекционные сканеры напоминают собой своеобразный проекционный аппарат. Вводимый документ кладется на поверхность сканирования изображением вверх, блок сканирования (либо фотоаппарат высокого разрешения) находится при этом также наверху. Перемещается только сканирующее устройство. Основной особенностью данных сканеров является возможность сканирования проекций трёхмерных предметов. Частный случай проекционного сканера – это когда считывающая камера не перемещается и зафиксирована сверху подобно настольной лампе освещающий стол.

Для того чтобы ввести в компьютер какой-либо документ при помощи **ручного (handheld)** сканера, надо без резких движений провести его сканирующей головкой по изображению. Равномерность перемещения сканера существенно сказывается на качестве вводимого в компьютер изображения, которое обычно значительно ниже, чем у других типов сканеров.

Для превращения отсканированной текстовой картинки в текстовый документ, состоящий из отдельных символов, служат программы распознавания текста (OCR-системы, Optical Character Recognition) GOCR [210], OOCR [211] и др. Из русских коммерческих продуктов наиболее хорошо себя зарекомендовала (в том числе и на мировом рынке) фирма ABBYY с её продуктом FineReader.

Многие производители, в том числе и HP, сознательно не поддерживают некоторые свои устройства под ОС Linux, кроме того, они не дают это делать OpenSource сообществу, поэтому такие сканеры не поддерживаются данной ОС. Если вы планируете использовать сканер со свободным ПО, проверьте перед покупкой, поддерживается ли он на странице http://www.sane-project.org/lists/sane-mfgs-cvs.html.

Таблица 4.19. Некоторые технические характеристики сканера HP scanjet 4600

Наименование	Характеристика
Типа сканера	планшетный, до 216×297 мм
Скорость (время)	6 секунд (предварительный просмотр)
Разрядность	48 бит
Скорость сканирования	цветное фотоизображение 10 × 15 см: 27 секунд чёрно-белый рисунок: 22 секунды
Масштабирование	от 10 до 2000% с шагом 1%
Разрешение при сканировании	оптическое: 2400 т/д; аппаратное: 2400 × 2400 т/д;
Интерфейс соединения	1 высокоскоростной интерфейс USB 2.0
Совместимость с операционными системами	Microsoft® Windows® (98, 2000, Me, XP Home и Professional Edition), Mac OS X 9.1 или 10.1 и выше
Габариты, вес нетто	488 × 340 × 83 мм, сканер: 1,4 кг, держатель: 0,5 кг.
Форматы файлов	Bitmap (BMP), TIFF, GIF, PDF, HTML, JPEG, FlashPix (FPX), сжатый TIFF, PCX, PNG, RTF, TXT

4.12.3.1. Сканер штрихкодов

Указанный вид сканеров также относится к типу сканеров считывающих графическую информацию, однако у него есть своя специализация – штриховые коды. Технически это может быть как стационарное устройство, так и ручная модель. Последняя может быть переносной с автономным источником питания.

Сфера их использования – это магазины и различные виды складского учёта.

Внутреннее устройство системы сканирования различно, существуют модели и с полупроводниковыми лазерами и с ПЗС-матрицами как в цифровых фото и видео камерах.

В зависимости от модели считываться могут как одномерные штрих коды, так и двумерные.

Чаще всего указанные устройства представляются компьютеру как USB-HID[212] class, обычно компьютер их видит как дополнительную (дублирующую) клавиатуру. Сканирование штрих кода, можно представить следующим образом: он сканируется и распознаётся сканером, а затем

[210] Свободное ПО распознавания текстов http://jocr.sourceforge.net/.
[211] http://sourceforge.net/projects/oocr/.

распознанные цифры и буквы быстро набираются на клавиатуре. Отдельные, более старые модели, могут иметь отличный от USB интерфейс: PS/2, SCSI, LPT, COM-порт (RS-232) и другие.

Портативные переносные сканеры используют беспроводные протоколам: Wi-Fi, Bluetooth, GSM, LTE и др. Чаще всего они идут совместно с обслуживающим их небольшим компьютером, расположенным в одном корпусе со сканером. В этом случае протокол и интерфейс получения данных от сканера оказываются скрыты внутри устройства, а внешне пользователь видит лишь какой-либо из стандартных сетевых протоколов передачи данных от встроенной в сканер мини-ЭВМ.

4.12.3.2. Считыватели карт и меток и др.

В качестве USB-HID class устройств могут выступать не только сканеры штрих кодов, но и сканеры различных карт с магнитной полосой, Proximity карт и брелков, RFID-меток (меток-транспондеров) т. п.

В ряде случаев указанные устройства отнюдь не моделируют работу клавиатуры ЭВМ, а представляют законченные стационарные решения, например это может быть автоматический считыватель меток-транспондеров для автоматического открытия ворот для подъезжающих автомобилей и т.п.

К подобному классу странных устройств можно отнести и сканеры-счётчики учитывающие количество прошедших людей, например системы подсчёта посетителей на входе в магазин или системы учёта прошедших на конвейерной ленте товаров. Они могут быть как простыми в виде ИК-излучателя и счётчика, так и представлять специализированные двух-объективные камеры, обычно размещаемые на потолке.

Замечание. С развитием технологий распознавания изображений, в частности на базе свёрточных нейронных сетей, подсчёт числа прошедших людей без труда может производиться на основе видеоизображения полученного от видеокамеры. В этом случае, компьютер с программой распознавания объектов на видеоизображении никак не назовёшь сканером.

4.12.4. Дигитайзер, графический планшет

Дигитайзер – устройство, которое используется для оцифровки конструкторской документации (чертежей).

Графический планшет и интерактивный перьевой дисплей – устройства, которые используются в компьютерной графике и анимации и взаимодействует с графическими приложениями [213].

Дигитайзер может иметь размер рабочей области A1, рабочая область его выглядит как чертёжная доска. Применяется дигитайзер для поточечного координатного ввода графических изображений в системах автоматизированного проектирования (САПР).

Дигитайзер и графический планшет обычно содержат рабочую плоскость, рядом с которой находятся кнопки управления. Для ввода информации служит специальное перо или координатное устройство с «прицелом», подключенное кабелем к планшету

[212] HID – human interface device; USB HIB class – класс устройств USB для взаимодействия с человеком: клавиатуры, мыши и т. п.
[213] Типа GIMP, Adobe Photoshop и др.

или с беспроводной передачей данных. Сам дигитайзер подключается к компьютеру кабелем через USB-порт. Разрешающая способность таких графических планшетов – не менее 100 dpi (точек на дюйм).

В некоторых дигитайзерах ввод информации может происходить без специальных перьев или прицелов, так как рабочая поверхность планшета обладает чувствительностью к нажатию пером, основанной на использовании пьезоэлектрического эффекта. При нажатии на точку, расположенную в приделах рабочей поверхности планшета, под которой проложена сетка из тончайших проводников, на пластине пьезоэлектрика возникает разность потенциалов. Координаты этой точки считываются программой, сканирующей сетку проводников. Эта программа запоминает координаты точек и показывает графические объекты на мониторе.

В графических планшетах и интерактивных мониторах (см. рис. 4.86) для рисования используется устройство ввода (беспроводное перо), нажатие которого на поверхность рабочей области считывается с использованием технологии электромагнитного резонанса.

После появления на рынке планшетов и широкого распространения сенсорных мониторов с функциями multi-touch, определения силы нажатия и управления пальцами руки, данные устройства практически не используются.

Рисунок 4.86. Графический планшет и интерактивный перьевой дисплей

Основные задачи, решаемые с использованием этих устройств:

- рисование;
- редактирование фотографий и изображений;
- создание коллажей, открыток и календарей;
- компьютерная анимация и 3D-моделирование;
- работа в мультимедийных приложениях;
- аудио- и видеомонтаж;
- распознавание рукописного текста и прочее.

4.12.5. Сенсорный монитор

Сенсорные мониторы (обычный монитор + сенсорная накладка) отличаются от обычных мониторов тем, что чувствительная накладка интегрирована в экран монитора, она имеет на поверхности экрана специальное покрытие, которое чувствительно к

прикосновениям и передаёт координаты этого прикосновения определённым образом в операционную систему. Наиболее часто драйвер эмулирует манипулятор мышь компьютера либо взаимодействует со специальным программным обеспечением через порт USB. Сенсорные экраны используются также в мобильных телефонах, электронных книгах, планшетах, КПК, ПК-моноблоках, в интерактивных учебных досках и других устройствах.

Помимо мониторов, существуют отдельно продающиеся прозрачные сенсорные накладки, которые работают по тому же самому принципу, что и сенсорный монитор. Они могут быть применены совместно с любым несенсорным монитором подходящего размера.

Сенсорные мониторы в настоящее время наиболее часто можно встретить в терминалах по оплате различных услуг, информационных системах вокзалов, в ресторанах при оформлении заказов официантами и т. п. Они могут использоваться как мониторы обычных настольных ПК. Существуют ПК-моноблоки с сенсорными экранами (например, фирм HP, Sony и др.) с эффектным управлением папками и файлами с помощью руки (похоже на управление телефоном с сенсорным экраном – touch phone).

Существует несколько видов сенсорных технологий:

- резистивная,
- поверхностно-акустическая,
- ёмкостная,
- инфракрасная.

Резистивная сенсорная панель имеет многослойную структуру, состоящую из двух проводящих поверхностей, разделённых специальным изолирующим составом, распределённым по всей площади активной области экрана. При касании наружного слоя, выполненного из тонкого прозрачного пластика, его внутренняя проводящая поверхность совмещается с проводящим слоем основной стеклянной пластины, играющей роль каркаса конструкции, благодаря чему происходит изменение сопротивления всей системы. Это изменение фиксируется микропроцессором контроллера, передающим координаты точки касания управляющей программе компьютера.

Поверхностно-акустическая волновая технология ПАВ (SAW – Surface Acoustic Wave) основана на использовании сверхзвукового сигнала, вырабатываемого пьезоэлектрическими преобразователями для осей X и Y. Волны направлены перпендикулярно друг другу и образовывают координатную сеть всего поля панели. Прикосновение к экрану поглощает часть волн, проходящих в этот момент через место контакта. Полученные датчиками сигналы отличаются от эталонного, что приводит к автоматическому определению места касания по осям X и Y. Уникальной особенностью является возможность определять силу прикосновения – координату Z.

Ёмкостная технология основана на использовании сенсорных панелей, покрытых материалом, содержащим электрический заряд. Когда к панели прикасаются, в точке прикосновения заряд изменяется, что фиксируется измеряющей системой. При работе с ёмкостными панелями используется прикосновение пальцев, в отличие от резистивных и ПАВ-панелей, при работе с которыми могут быть использованы прикосновения пальцем или пером. Для ёмкостных экранов на рынке можно купить трикотажные перчатки со специально прорезиненными кончиками, позволяющие управлять тем же сенсорным телефоном на улице, не снимая перчаток.

Инфракрасная технология раньше использовалась в основном в экранах больших размеров (школьные интерактивные доски, рекламные демонстрации, большие информационные панели). Сейчас активно используется в электронных книгах.

Multi-touch – технология, позволяющая одновременно отслеживать две и более точек прикосновения. Использование этой технологии позволило пользователям применять ставшие уже привычными жесты: свести два пальца на экране – уменьшение изображения, развести два пальца – увеличение (приближение) изображения или объекта. Не ведитесь на рекламные уловки, практика показывает, что отслеживать более чем пять точек прикосновения не имеет смысла, хотя вполне возможно, что в будущем мы научимся использовать все 10 пальцев в жестах и, возможно, даже будем помогать им 11-й точкой прикосновения – носом.

Цена настольных 19″ сенсорных мониторов – от 16 000 руб. [214].

4.12.6. Платы видеозахвата, TV- и FM-приёмники

Существуют специальные платы для преобразования аналогового видеосигнала в цифровой вид. К ним относят платы видеоприёма. Видеосигнал может приходить непосредственно на один или несколько видеовходов (например, со старого видеомагнитофона) или с видеокамеры, либо в модулированно-закодированном виде (например, телевизионные передачи, принимаемые с эфира, в том числе со спутниковой антенны и со стереозвуком). В более простом варианте платы могут принимать радиопередачи (FM-tuner).

Примерами фирм, выпускающих как внутренние платы расширения, так внешне подключаемые USB-модули, могут быть AverMedia, Behold, Kworld и др.

У большинства внешних USB TV- и FM-tuner'ов разные названия и бренды, но объединяет их одно – практически все они построены на чипсете RTL2832. Это недорогая микросхема, содержащая два 8-битных АЦП с частотой дискретизации до 2,8-3,2 МГц. За счёт этого появилось целое направление «RTL-SDR'инга». Аббревиатура RTL появилась от названия чипсета фирмы Realtek, а SDR означает Software Defined Radio.

Так, RTL-SDR – это целое семейство дешёвых ТВ-тюнеров, способных выполнять функцию SDR-приёмника. Фактически они принимают сигнал, а его обработку производит компьютерная программа, которая моделирует внутри себя работу различных традиционно-аппаратных трактов и каскадов радиоприёма.

То есть, за счёт слова «Software» в SDR стало возможным не только написать свою программу для улучшения параметров приёма, но и принимать с эфира практически любые сигналы[215].

[214] По данным www.touchgames.ru.
[215] В зависимости от модели, 50-1700 МГц, а то и шире.

Так, скачав с https://github.com/keenerd/rtl-sdr исходные коды SDR, можно скомпилировать и установить под ОС Linux программу `rtl_fm`, после чего, используя утилиту play из пакета sox можно прослушивать радиопередачи, например радио 99,2 FM:

```
$ rtl_fm -M wbfm -f 99.2M |play -r 32k t raw -e s -b 16 -c 1 -V1 -
```

Выглядящие аналогично, USB DVB-T2 приёмники могут принимать телевизионные передачи цифрового вещания. Для поиска каналов и создания плейлистов используется утилита w_scan, а для просмотра – программы mplayer, vlc и др.

Для расширения кругозора в вопросе приёма радиосигналов на компьютере с ОС Linux рекомендуем познакомиться с программой gnuradio (пакет gnuradio из EPEL, http://www.gnuradio.org).

4.12.7. Музыкальные устройства ввода

Рисунок 4.87. MIDI-клавиатура

К музыкальным устройствам ввода прежде всего относится MIDI-клавиатура, подключаемая к ПК через его звуковую плату. Например, клавиатура M-Audio Pro Keys 88 (www.m-audio.com, см. рис. 4.87) – полноразмерное (88 клавиш с профессиональной рояльной механикой) цифровое пианино и MIDI-контроллер. Существуют и другие музыкальные устройства, подключаемые к ПК, например M-Audio Black Box – мощный эффект-процессор, способный моделировать 12 известных гитарных предусилителей со встроенной drum-машиной и гитарным тюнером.

4.12.8. Веб-камера

Современная веб-камера (рис. 4.88) представляет собой цифровое устройство, производящее видеосъёмку, оцифровку, сжатие, сохранение и передачу цифрового видео по компьютерной сети.

Рисунок 4.88. Веб-камера

В состав веб-камеры входят следующие компоненты: ПЗС-матрица, объектив, оптический фильтр, плата видеозахвата, блок компрессии (сжатия) видеоизображения, центральный процессор и встроенный веб-сервер, ОЗУ, флэш-память, сетевой интерфейс, последовательные порты, тревожные входы/выходы.

Некоторые модели камер имеют поворотные устройства и могут выполнять круговое видеонаблюдение, а также имеют варифокальный объектив, которым можно управлять по сети.

Подключаться камеры могут непосредственно к сетевому коммутатору витой парой (Ethernet) или по Wi-Fi, иметь встроенный веб-сервер для управления камерой и просмотра видео-изображения. Простейшие камеры подключаются к компьютеру по USB (см. рис. 4.89).

Замечание. С появлением систем видеонаблюдения появился и новый термин–аббревиатура CCTV, обозначающий c*losed-circuit television* – системы закрытой трансляции телевидения.

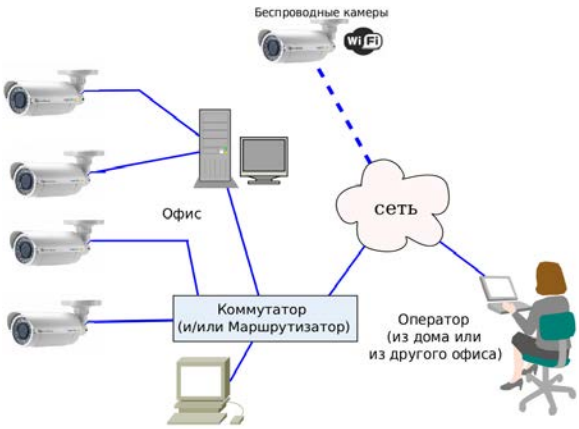


Рисунок 4.89. Варианты подключения веб-камер

4.13. Устройства вывода информации

Предназначены для представления различных видов информации, с которыми работает компьютер, в привычном для человека виде (изображения, текст, числа, звук). К устройствам вывода относятся:

- видеокарты и мониторы;
- принтеры;
- плоттеры;
- звуковые платы и колонки.

4.13.1. Видеоадаптер

Видеоадаптер – устройство для сопряжения ПК с монитором.

Может быть интегрирован с материнской платой или выполнен в виде отдельной платы (видеокарта), подключаемой к слоту PCI Express x16 материнской платы (в старых компьютерах для подключения использовался слот AGP). При наличии необходимых разъёмов возможно использование нескольких видеокарт. (Например, при подключении более одного монитора к ПК либо использовании видеокарт для вычислений. Подробнее см. «*Технология CUDA*» в главе 4.3. «*Процессор*»).

В ряде случаев функции видеокарты могут быть встроены в процессор, например решение *Intel® HD Graphics* для Core i3/i5/i7, при этом, с одной стороны, быстрее происходит обмен информацией, меньше суммарный нагрев, с другой, видеоподсистема для своей работы расходует основную оперативную память, к тому же, не такую быструю как видеопамять.

Однако наибольшие возможности для работы с графикой имеют дискретные графические адаптеры интерфейса PCI Express фирм NVIDIA (GeForce) и ATI (Radeon, см. рис. 4.90), которые имеют графический процессор, использующий специальные

технологии обработки графической информации, и собственную видеопамять большого объёма.

Современный видеопроцессор имеет более сложную архитектуру, чем центральный процессор. В нём реализованы потоковые процессоры с раздельными блоками для текстурных и растровых операций. Программы генерации видеоизображения (шейдеры) подразделяются на вершинные, пиксельные и геометрические шейдеры (последние только для DirectX 10 и выше). Вершинные программы-шейдеры отвечают за построение и изменение 3D-объектов, построенных из трёхмерных точек (вершин). Пиксельные программы-шейдеры позволяют менять цвета и интенсивность пикселей в результате каких-либо событий в программах. Геометрические шейдеры изменяют параметры трёхмерных изображений, позволяют ломать, модифицировать и уничтожать их. Новые стандарты графического интерфейса DirectX 10 и 11 опираются на архитектуру унифицированных (единых) шейдеров, то есть структура кода вершинных, геометрических и пиксельных программ единая, хотя шейдеры выполняют разную работу.

Рисунок 4.90. Видеокарта Radeon R7 260X

Основные характеристики некоторых видеокарт (с интерфейсом PCI Express) приведены в табл. 4.19.

Производительность видеокарты зависит как от частоты видеопроцессора, так и от количества потоковых процессоров, а также от объёма видеопамяти, используемой для хранения текстур, и от ширины шины памяти. Современные шины памяти имеют ширину от 128 до 448 бит.

Следует заметить, что стоимость видеокарт (на февраль 2018 г.) с памятью 1024 МБ находится в пределах 2000–5000 руб. в зависимости от модели, а цена самых новых видеокарт, на борту которых до 11 ГБ, доходит до 50 тысяч рублей.

Таблица 4.20. Технические характеристики некоторых видеокарт

Видеоадаптер	Частота процессоров, МГц	Память, частота МГц	Тип памяти, разрядность, бит	Число потоковых процессоров/унифицированных шейдеров	Объём памяти, МБ
GeForce GTX 730	1008	5012	GDDR5, 64	384	1024
GeForce GTX 750 Ti	1072	5400	GDDR5, 128	640	2048
GeForce GTX 980	1190	7012	GDDR5, 256	2048	4096
GeForce GTX 1080	1569	11010	GDDR5X, 352	3584	11 ГБ
Radeon R7 370	1015	5600	GDDR5, 256	1024	2048
Radeon R9 390X	1050	6000	GDDR5, 512	2816	8192
Radeon RX 550	1071	6000	GDDR5, 128	512	2048
Radeon RX 580	1430	8400	GDDR5, 256	2304	8192

Современные видеокарты имеют различные наборы видеовыходов: VGA (D-SUB), DVI, DisplayPort (DP), HDMI, некоторые порты могут идти по два и даже четыре раза. В зависимости от выбора порта и режима работы видеокарта может обеспечивать вывод изображения с различным максимальным разрешением. Обычно это до 2048×1536 для D-SUB, до 2560×1600 (на каждый выход) для DVI, до 3840×2400 для DP и до 1920×1080 для HDMI. Если требуется подключение монитора с большим разрешением или 4K(8K)-телевизора, чьи разрешения выше возможностей одного видеопорта видеокарты, подключение производится двумя кабелями с двух портов. Профессиональные видеокарты с разъёмом DisplayPort 1.2 могут обеспечить разрешения до 4096×2160 с глубиной цвета до 30-бит с одного порта.

Видеокарты могут потреблять до 300–400 Вт электроэнергии и выделять соответствующее количество тепла при этом. Подобное количество тепла, выделяемое с поверхности одного кремниевого чипа, может легко сжечь кристалл (тепловой пробой). Охлаждение процессора и памяти видеокарт обычно выполняется с помощью радиатора с вентилятором (или без). На жаргоне их часто называют «кулером» – от англ. cooler (охладитель, холодильник, радиатор, теплосъёмник), а вентилятор – «Карлсоном».

Высокая вычислительная мощность видеокарт может использоваться не только для построения видеоизображений в играх, но и в прикладных задачах, как то различные brute force атаки (вычислительные переборы), подсчёты хэшей, генерация радужных таблиц, майнинг криптовалют, создание, обучение и использование свёрточных нейронных сетей и др.[216]

4.13.2. Монитор

Монитор – устройство отображения графической информации, формируемой видеоадаптером (видеокартой).

В настоящее время основным типом мониторов для ПК является жидкокристаллический (ЖК, LCD – Liquid Crystal Display). До недавнего времени выпускались и использовались ЭЛТ (электронно-лучевая трубка, CRT, Cathode Ray tube) мониторы на основе электронно-лучевой трубки (см. рис. 4.91).

Рисунок 4.91. ЖК- и ЭЛТ-монитор

Кроме того, существуют плазменные панели большого размера (42″–70″), которые в основном используются в домашних кинотеатрах. Формально они могут иметь все те

[216] Также советуем осуществить поиск в интернете по ключевому слову CUDA.

же самые разъёмы для подключения к компьютеру, что и обычные мониторы, однако, относить их к мониторам не принято, скорее они воспринимаются людьми как большие телевизоры с возможность подключения к компьютеру.

Во всех типах мониторов для формирования цветного изображения используется цветовая схема **RGB** (см. раздел *2.4.1. Растровая графика*, стр. 125).

Сенсорные мониторы (см. ранее раздел 4.12.5. Сенсорный монитор, стр.331) могут быть одновременно устройствами ввода и вывода информации.

Основные характеристики современного монитора следующие.

Размер экрана – обычно измеряется по диагонали в дюймах и составляет у современных ЖК-мониторов 17″, 19″, 20″, 21″, 22″, 24″, 27″, 30″, 32″, 34″, 40″, 46″ (стоимость соответственно от 4000 до 150 000 руб.).

Разрешение монитора (разрешающая способность) характеризует способность отображать на экране определённое количество точек по горизонтали и вертикали. Для каждого монитора существует определённое максимально возможное разрешение, определяемое шириной полосы пропускания видеоусилителя для аналоговых мониторов и размерами матрицы для цифровых. Если на монитор подаётся изображение в другом режиме, то монитор может его интерполировать в своё внутреннее разрешение. Список подаваемых на вход монитора сигналов с теми или иными разрешениями должен согласовываться с возможным разрешением видеокарты. Обычное разрешение для 17″ – 1280×1024, для мониторов больших размеров может использоваться разрешение 1600×1200, 1920×1280, 2560×1600, 3840×2400, 4096×2160, 5120×2880. Для ЖК-мониторов существует штатное разрешение, которое соответствует количеству жидкокристаллических пикселей. У мониторов большого размера соотношение сторон может быть 4:3 или 16:10 (16:9) (см. табл. 4.21). На качество восприятия изображения влияет влияет размер одного элемента (точки, пикселя), для более удобной оценки этого параметра используют понятие плотности точек на дюйм – «ppi» (от англ. pixels per inch, количество пикселей, приходящихся на дюйм экрана). Иногда этот параметр ошибочно называют также разрешением. Чем выше плотность, тем более естественным изображение кажется глазу. Начиная с определённой плотности невооружённый глаз человека не может различить двух соседних точек. На расстоянии вытянутой руки это порядка 300 точек на дюйм.

Таблица 4.21. Характеристика ЖК-матриц большого размера

Диагональ, дюймы	Разрешение	Соотношение сторон	Шаг точки, мм	Плотность точек на дюйм, ppi
31,5	7680×4320	16:9	0,09	279,74
30	2560×1600	16:10	0,258	98,4
27	5120×2880	16:9	0,1169	217,58
24	1920×1200	16:10	0,27	94,0
23,1	1920×1200	16:10	0,258	98,4
22,2	1920×1200	16:10	0,249	102,0
22,2	3840×2400	16:10	0,1245	204,0
21,3	1600×1200	4:3	0,27	94,0
20,8	2048×1536	4:3	0,207	122,7
20,1	1600×1200	4:3	0,255	99,6
20	1680×1050	16:10	0,258	98,4

Частота кадровой развёртки, или частота смены кадров, выраженная в герцах (Гц), – сколько раз в секунду изображение обновляется. Показатель имел большое значение для ЭЛТ-мониторов, где изображение постоянно регенерировалось бегущим по строчкам лучом: чем выше у них была частота кадровой развёртки, тем меньше был уровень нежелательного мерцания изображения и, следовательно, меньше нагрузка на зрение. Поэтому стандарт определял обязательной поддержку 85 Гц, а некоторые модели поддерживали развёртку до 100–120 Гц. Для ЖК-мониторов используется другой принцип построения изображения, обновление изображения не вызывает мерцания как в ЭЛТ-трубках. Чем выше частота обновления, тем шире должна быть полоса видеоусилителя, поэтому для ЖК-мониторов оптимальной частотой обновления выбрали частоту в 60 Гц. Поддерживать на входе могут несколько больше.

Шаг точки – это расстояние по диагонали между двумя пикселями жидких кристаллов или точками люминофора одного цвета. Этот размер обычно выражается в миллиметрах (например, 0,24 мм).

Для ЖК-мониторов в паспортных данных указываются также следующие характеристики:

Яркость – обычно имеет значение 250–400 кд/м^2.

Контрастность – 400:1, 550:1, 600:1, 1000:1, 4000:1. Для плазменных панелей контрастность может достигать десятков тысяч.

Время отклика пикселя – 2–16 мс.

Углы обзора – по горизонтали (например, 178°) и по вертикали (178°).

Ориентация экрана у ЖК-монитора, в отличие от ЭЛТ-монитора, может быть как ландшафтная (горизонтальная), так и портретная (вертикальная). В то время как традиционные экраны ЭЛТ-мониторов и ЖК-экраны ноутбуков имеют только ландшафтную ориентацию, обусловленную тем, что поле зрения человека в горизонтальном направлении шире, чем в вертикальном, в ряде случаев (работа с текстовыми документами, веб-страницами) удобнее работать с экраном портретной ориентации. ЖК-мониторы некоторых моделей можно развернуть на 90° с сохранением правильной ориентации изображения.

Вогнутость. Благодаря компании Samsung к нам вернулись не плоские (изогнутые) дисплеи и телевизоры. Только если у первых ЭЛТ мониторов трубки были выпуклыми наружу, что мешало считыванию изображения и это был «минус», отчего их стремились сделать «плоскими», то сейчас изогнутое (не плоское) изображение следует воспринимать как «плюс», поскольку ЖК матрицы стали делать вогнутыми. Например модель Samsung S34E790C имеет не плоский экран.

Замечание. Некоторые мониторы имеют несколько видеовходов и позволяют подключать себя различными способами, либо использовать с двумя системными блоками. Так, если у вас есть выбор между аналоговым D-SUB и цифровым DVI, – выбирайте последний. Также, обратите внимание, что хотя DVI-D использует цифровую передачу данных, как и HDMI, и требования к соединительным кабелям стандартизированы, качество от выбора последних зависит напрямую. Не экономьте на видеокабеле, выбирайте модели, что имеют большее сечение проводников, а значит меньшее сопротивление и меньшие потери.

4.13.2.1. Принцип построения изображения

Первая монохромная ЖК-панель увидела свет в 1970 году. С тех пор технология непрерывно совершенствовалась. В современных профессиональных мониторах при-

меняются активные матрицы на тонкоплёночных транзисторах (AM TFT – active matrix thin film transistor) в различных модификациях. Таким образом, термин LCD (ЖК) более общий, а TFT – одна из разновидностей ЖК-мониторов.

Принцип работы ЖК-монитора следующий (см. рис. 4.92).

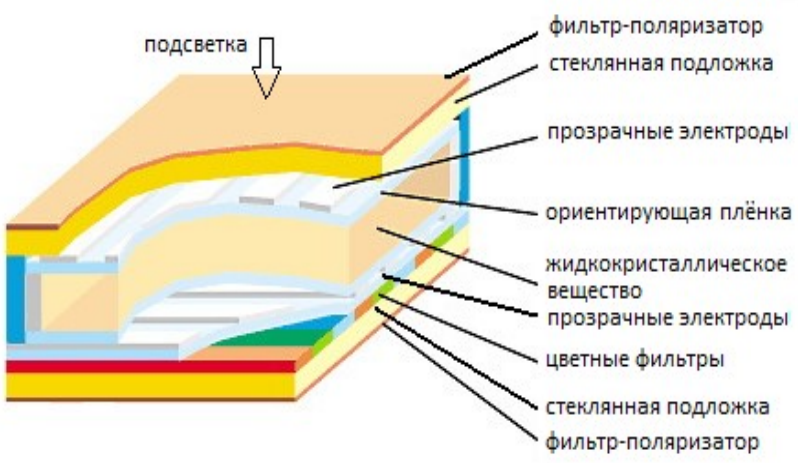


Рисунок 4.92. Устройство жидкокристаллической матрицы

Слой жидкокристаллического вещества (ЖКВ) располагается между двумя полированными прозрачными пластинами, сделанными из свободного от натрия и очень чистого стекла, на которые нанесены прозрачные электроды, управляющие каждой ячейкой структуры TFT. На стеклянных панелях имеются бороздки, которые ориентируют молекулы ЖКВ в определённом направлении. С двух сторон от стекол находятся поляризаторы, оси которых перпендикулярны. Это требуется потому, что плоскость поляризации светового луча поворачивается на 90° при прохождении ЖК-слоя. С внешней стороны экрана располагаются светофильтры, с помощью которых формируются пиксели из трёх ячеек основных цветов (RGB). ЖК-панель освещается источником света сзади, то есть работает на прохождение света. Свет от лампы подсветки, пройдя поляризатор и стеклянную пластину, попадает в слой ЖКВ. Если на элементарную ячейку не подано напряжение, то при прохождении света через слой жидкого кристалла его поляризация поворачивается в соответствии с поворотом оптической оси кристалла. В результате на выходе свет пройдет через второй поляризатор, и этот пиксель дисплея выглядит для наблюдателя окрашенным в один из трёх цветов RGB (в зависимости от фильтра), сумма максимальных яркостей трёх сабпикселей даёт белый цвет. Если же на электроды пиксельной ячейки подано напряжение, то свет пройдёт слой жидкого кристалла, не изменив своей поляризации, и поляризатор на выходе не пропустит его. Этот пиксель будет выглядеть чёрным на светлом фоне.

Управление каждым сабпикселем выполняется активно (активная матрица), с помощью двух транзисторов и прозрачного электрода. Транзисторы сформированы на прозрачной плёнке и выполнены либо из аморфного (a-Si), либо из поликристаллического кремния (p-Si). Поданный уровень напряжения удерживается и сохраняется до следующего сигнала. В принципе, поскольку каждым сабпикселем управляют индивидуально через адресные линии строк и столбцов, необходимости в регулярном обнов-

лении всех элементов изображения нет – достаточно скорректировать управляющие сигналы в тех областях, где изображение изменилось.

С типичными структурами матриц экранов можно познакомиться тут: http://www.ixbt.com/portopc/px-macro.shtml.

4.13.3. Принтер

Принтер – устройство вывода на бумагу (или иной носитель) текстовой и/или графической информации.

Принтеры бывают самые разные, например в кондитерском деле существуют пищевые принтеры и плоттеры фирмы Modecor [217] со съедобными чернилами, которые могут печатать тексты и фотографии на вафельных или сахарных пластинах. А плоттеры так вообще могут творить кондитерские произведения искусства (см. рис. 4.93).

Рисунок 4.93. Пищевой плоттер Modecor Decoplotty http://www.modecor.it/en/decoplotty/

Существуют принтеры, печатающие рекламные плакаты, наклейки в супермаркетах, заполняющие паспорта, билеты, платёжки и сберкнижки. В общем, «выбирайте на любой вкус и цвет». Чаще всего все эти принтеры подключаются к компьютеру, который подготавливает и отправляет информацию для печати. С этого момента и начинаются все тонкости.

Несмотря на то что за захват, перемещение носителя, нанесение изображения или перемещение печатающей головки отвечают различные механизмы, состоящие из роликов, моторчиков, шестерёнок и др., всеми этими частями управляет некоторое центральное устройство, которое мы назовём процессором или контроллером принтера. Пользователю всё равно, как эти части взаимодействуют между собой, ему важен результат. Поэтому с целью удобства и унификации принтеров их контроллеры должны понимать какой-либо язык описания команд, формирующих задание для печати. Подобных языков существует великое множество: ESC/P-последовательности, Postscript, PCL, HPGL, Lineprinter, Xerox XES/UDK и др. Наиболее распространёнными являются первые три. Они же практически всегда поддерживаются современными ОС. Большинство принтеров делают совместимыми с этими языками. Казалось бы, всё очень удобно и хорошо, но прогресс не стоит на месте.

В настоящее время широкое распространение получили 2 типа принтеров: лазерные – в организациях, для печати большого количества документов ежедневно, и

[217] Хотите порадовать и удивить любимых жену и дочку, мужа или начальника? Нанесите фотографию на торт, логотип фирмы на печенье.

струйные – в домашних условиях с возможностью печати цветных изображений. Ранее использовались также матричные принтеры. Лазерные принтеры обеспечивают более высокое качество печати, чем струйные.

4.13.3.1. Лазерный принтер

Лазерный принтер использует фотоэлектронный способ печати, основанный на формировании лазером изображения на заряженной светочувствительной поверхности промежуточного носителя (фотобарабана) в виде электростатического рельефа, притягивающего частицы красителя, которые далее переносятся на бумагу.

Наиболее известными фирмами-разработчиками лазерных принтеров являются Hewlett-Packard, Epson, Lexmark, Canon, Xerox, Kyocera, Toshiba.

Функциональная схема лазерного принтера приведена на рисунке 4.94.

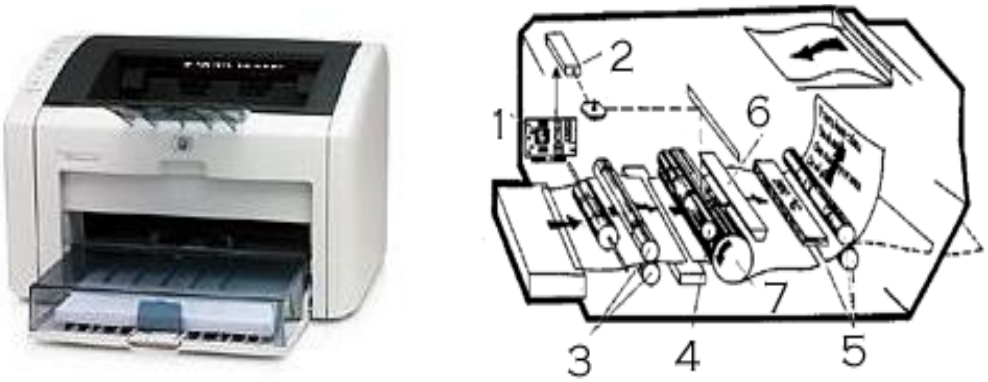


Рисунок 4.94. Лазерный принтер и его схема (1 – управляющая плата, 2 – лазер, 3 – механизм подачи бумаги, 4 – барабан-девелопер, 5 – механизм нагрева (припекания, сплавления порошка краски), 6 – лезвие, 7 – фотобарабан)

Основным элементом конструкции лазерного принтера является вращающийся фотобарабан (см. Рисунок 4.95), служащий промежуточным носителем, с помощью которого производится перенос изображения на бумагу. Барабан представляет собой цилиндр, покрытый тонкой плёнкой светочувствительного полупроводника. Обычно в качестве такого полупроводника используется оксид цинка или селен[218]. По поверхности барабана равномерно распределяется статический заряд. Это обеспечивается с помощью тонкой проволоки или сетки, называемой коронирующим проводом. На этот провод подаётся высокое напряже-

Рисунок 4.95. Фотобарабан лазерного принтера

[218] Изначально в качестве фотопроводника в фотобарабанах использовались соединения селена (34-й элемент в таблице Д. И. Менделеева). В настоящее время селеновые фотобарабаны практически полностью сняты с производства. Вместо селена используются различные органические соединения, состав которых является коммерческой тайной производителей. В последнее время получили широкое распространение фотобарабаны с покрытием из аморфного кремния.

ние, вызывающее возникновение вокруг него светящейся ионизированной области, называемой короной.

Лазер, управляемый процессором принтера, генерирует тонкий световой луч, направляемый с помощью зеркала на фотобарабан. Развёртка изображения происходит так же, как и в телевизионном кинескопе: есть движение луча по строке и кадру. С помощью поворачивающегося зеркала луч направляется на фотобарабан, его яркость меняется в соответствии с печатаемым изображением и в зависимости от интенсивности луча изменяет электрический заряд поверхности фотобарабана. Таким образом, на фотобарабане, промежуточном носителе, возникает скрытая копия изображения в виде электростатического рельефа. (см. Рисунок 4.96)

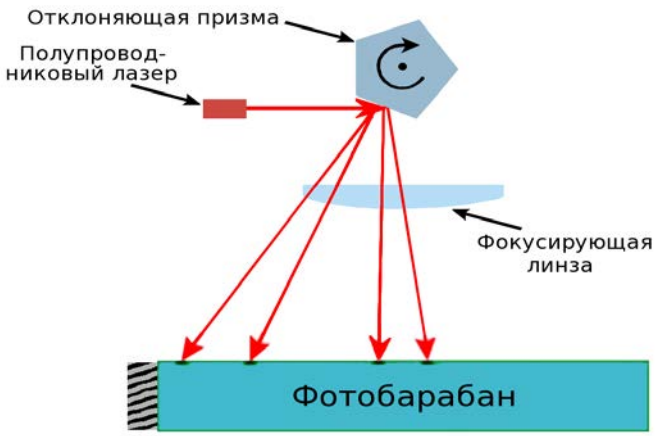


Рисунок 4.96. Формирование изображения

На следующем этапе на фотонаборный барабан наносится тонер – мелкодисперсная краска. Под действием статического заряда эти частицы легко притягиваются к поверхности барабана в точках, подвергшихся экспозиции, и формируют изображение уже в виде красителя.

Бумага втягивается из подающего лотка и с помощью системы валиков перемещается к барабану. Перед самым барабаном бумаге сообщается статический заряд. Затем бумага соприкасается с барабаном и притягивает, благодаря своему заряду, частички тонера, нанесённые ранее на барабан. Для фиксации тонера бумага вновь заряжается и пропускается между двумя роликами с температурой около 180 °C. После окончания процесса печати барабан полностью разряжается, очищается от прилипших лишних частиц, тем самым готовясь для нового процесса печати.

Цветные лазерные принтеры делятся на однопроходные и четырёхпроходные. В однопроходных все 4 картриджа (цветовая схема **CMYK** – Cyan, Magenta, Yellow, blacK) расположены последовательно по ходу движения бумаги, у каждого картриджа имеется свой фотобарабан и свой источник – лазер.

Основные характеристики лазерного принтера следующие.

Качество печати зависит от многих факторов. Основной параметр, по которому можно судить о качестве печати, – это разрешение принтера (измеряется в точках на дюйм – dpi). Современные лазерные принтеры печатают с разрешением от 600 dpi до 2400 dpi. Помимо физического разрешения, существуют технологии, улучшающие качество печати, например ImageRET у HP, Color RIT у Epson, FinePoint у XEROX и т. д. Рядом с названием технологии пишется число, показывающее, к какому разрешению пытаются приблизиться производители, например Epson описывает свой принтер, имеющий физическое разрешение 600×600 dpi, как печатающий с разрешением 2400 dpi (с использованием технологии RITech 2400). Конечно, качество изображения от примене-

ния таких технологий несколько улучшается, но качество печати всё же хуже, чем у принтеров с разрешением 1200 dpi.

Скорость печати лазерного принтера измеряется в страницах в минуту и для обычных принтеров находится в диапазоне от 10–14 страниц в минуту. При печати сложных графических изображений скорость печати лазерного принтера может снижаться. Высокопроизводительные сетевые принтеры обеспечивают скорость печати более 20 страниц в минуту. Скорость печати лазерного принтера зависит от скорости обработки данных, поступающих от ЭВМ, и формирования растровой страницы для печати. Как правило, лазерный принтер оснащен собственным процессором. Скорость печати определяется не только работой процессора, но и существенно зависит от объёма памяти, которую имеет принтер.

Память лазерного принтера, который обрабатывает информацию постранично, должна обеспечивать большое количество вычислений. Например, при разрешении 300×300 dpi на странице формата A4 насчитывается почти 9 млн точек, а при разрешении 1200×1200 – более 140 млн. Минимальной величиной памяти лазерного принтера считается 1 Мбайт, а в основном используют память от 2 до 4 Мбайт, причём цветные лазерные принтеры обладают ещё большей памятью. Сетевой лазерный принтер может иметь и внешнюю память (винчестер).

Интерфейс подключения самый распространённый – USB. И здесь есть несколько нюансов: существуют три версии – USB 1.1, USB 2.0 (HiSpeed USB) и USB 3.0. Интерфейсы USB 2.0 и USB 3.0 намного быстрее первой версии, и желательно, чтобы компьютер поддерживал их. Второй нюанс: для подключения по USB нужен качественный кабель, так как некоторые принтеры выдают при работе «со слабеньким» кабелем труднолокализуемые ошибки. Старые принтеры подключаются по параллельному интерфейсу (LPT). Многие принтеры (в основном для рабочих групп) поддерживают подключение по сетевому кабелю через Ethernet-карту – в названии моделей таких принтеров обычно присутствует буква «N». Принтеры, подключаемые через сеть, поставляются с развитыми средствами администрирования. В дополнение к этим распространённым интерфейсам подключения для многих принтеров опционально предлагаются средства беспроводной связи.

Формат бумаги – в основном лазерные принтеры используют для печати бумагу формата A4, и только некоторые модели обеспечивают печать на листах формата A3. Некоторые модели лазерных принтеров используют для работы бумагу в рулоне, выполняют двухстороннюю печать или имеют возможность выборки листов из нескольких лотков и раскладки напечатанных листов по нескольким приёмным карманам.

Языки управления принтером и драйверы. Принтер – устройство растровое, и чтобы превратить отправленный на печать документ в образ, понятный принтеру, нужна программа, которая выполнит все преобразования (масштабирование, растрирование и т. д.). Этим занимаются драйверы с помощью языка описания страниц. На вершине иерархии находится язык Postscript – аппаратно независимый язык, гарантирующий, что распечатки, полученные на разных принтерах от разных производителей, будут выглядеть одинаково. Посередине находится многочисленная группа принтеров, печатающих под управлением языка PCL, разработанного компанией HP. В отличие от Postscript’а, драйвер PCL загружает работой как компьютер пользователя, так и процессор принтера, разделяя нагрузку между ними. Ещё одно отличие от Postscript’а заключается в сильной оптимизации страниц, отправленных на печать, в результате

точность вывода снижается, но увеличивается скорость печати. Поэтому для точной печати (дизайн, вёрстка) нужно пользоваться Postscript'ом, а когда важна скорость, можно воспользоваться PCL-драйвером. Обычно к принтеру идёт несколько драйверов: Postscript, PCL-драйвер для цветных принтеров, цветные и монохромные версии Postscript- и PCL-драйверов.

4.13.3.2. Особенности печати на GDI-принтерах

Проблема перехода на новое всегда актуальна, вернёмся в 90-е годы прошлого века. У большинства были матричные принтеры, «знающие» только об ESC/P-последовательностях, а лазерные и струйные были уделом немногих в силу их высокой стоимости, но со временем ситуация начала меняться. Пользователи стали печатать всё больше графики. Разрешения, а значит, и качество документов росли. В переходный момент появления языков Postscript и PCL для поддержания обратной совместимости со старыми программами многие лазерные и струйные принтеры умели эмулировать матричные принтеры, а некоторые умеют делать это и сейчас. С одной стороны, это хорошо и удобно всем, но с другой – это означает дополнительные накладные расходы. Например, если на лазерный принтер, эмулирующий матричный, отправить на печать слово «мама», то он должен получить меньше десятка байт (4 байта текста + несколько служебных) и напечатать слово. При этом принтер, получив 4 байта текста, должен найти «у себя» таблицу шрифтов и понять, как та или иная буква выглядит. То есть это дополнительная память и вычислительные ресурсы. Когда же мы печатаем изображение, например того же слова «мама», то оно целиком формируется на компьютере и посылается на принтер. С точки зрения пользователя, разницы нет, картинка одинаковая в обоих случаях, но с точки зрения технологии и экономии, есть. Обратная совместимость сейчас не очень важна, а если изображение текста может формировать компьютер, то зачем держать аналогичный блок в принтере? Не проще ли сделать принтер без этого блока, будет наверняка надёжнее, так как меньше деталей и точно дешевле. Так и поступили производители, создавая технологию GDI-принтера. GDI – это Graphic Device Interface – не что иное, как библиотека функций ОС Windows для осуществления вывода информации на графические периферийные устройства, такие как дисплеи или принтеры. Вместе с этой библиотекой в связке работает драйвер принтера. Всё бы хорошо, только производители принтеров в погоне за дешевизной упустили из виду другие операционные системы, как ОС Linux, и забыли предоставить для них свои драйверы. Предполагаем, что тут не обошлось без влияния производителей Windows. Так как специфический внутренний формат данных GDI-принтеров не был стандартизирован, то у каждого производителя принтеров он свой, и каждому принтеру нужен свой драйвер. Даже в такой непростой ситуации драйверы под Linux не понадобились бы, если фирмы удосужились бы предоставить информацию о своих протоколах. Можно быть уверенными, что десяток-другой энтузиастов решили бы проблему менее чем за неделю. Но протокол каждого производителя – это «know-how» – запатентованная технология, поэтому они не спешат делиться информацией с open source сообществом. А в это время большинство пользователей покупающих принтеры, не только далеки от технологии печати, но и не задумываются о совместимости. Обидно, но именно они, массовые покупатели, не разбирающиеся в деталях, чаще всего определяют ход истории, развитие принтеров, операционных систем и многого другого, покупая то, что подешевле.

4.13.3.3. Струйный принтер

Струйные принтеры имеют печатающую головку, перемещающуюся поперёк листа бумаги, имеющую набор тонких сопел, через которые выбрасываются чернила. Диаметр сопел составляет десятые доли миллиметра. В основном число сопел в моделях различных изготовителей составляет от 16 до 64. Хранение чернил обеспечивается двумя конструктивными решениями. В одном из них головка принтера объединена с резервуаром (картриджем) для чернил, причём замена резервуара с чернилами одновременно связана с заменой головки. Другое предусматривает использование отдельного картриджа, который через систему капилляров обеспечивает чернилами головку принтера.

В струйных принтерах в основном используют следующие методы нанесения чернил: термическая струйная (пузырьковая) технология и пьезоэлектрический метод.

Термическая струйная печать разработана фирмами Hewlett-Packard и Canon. Успех принтеров этой технологии был обусловлен тем, что они обеспечивали качество печати, близкое к лазерным принтерам при значительно меньшей цене. Качество печати таких устройств зависит от размера точки от капельки чернил, а он очень маленький. Конструкция печатающей головки позволяет достичь разрешения до 9600×2400 точек на дюйм (Canon PIXMA iP4500).

Hewlett-Packard называет свою струйную технологию *drop-on-demand* (см. рис. 4.98), а Canon – пузырьковая технология (bubble-jet – рис. 4.97).

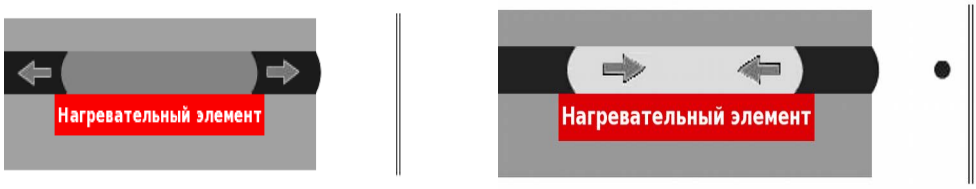


Рисунок 4.97. Принцип пузырьковой технологии (bubble-jet) фирмы Canon

В печатающих системах, использующих струйную пузырьковую технологию, текст и графика получаются при попадании на бумагу капелек чернил, вылетевших из очень тонких сопел.

Принцип работы термического струйного печатающего устройства заключается в следующем.

Рисунок 4.98. Принцип технологии drop-on-demand фирмы Hewlett-Packard

В стенку сопла печатающей головки встроен нагревательный элемент в виде тонкопленочного резистора, который при пропускании через него тока за 7–10 микросекунд нагревается до высокой температуры. Температура, необходимая для испарения чернил, например фирмы Hewlett-Packard, достигает 650 °С. Возникающий при резком нагревании чернильный паровой пузырь стремится вытолкнуть через выходное отверстие сопла каплю жидких чернил диаметром менее 0,16 мм, которая переносится на бумагу. При отключении тока тонкоплёночный резистор быстро остывает, паровой пузырь уменьшается в размерах, что приводит к разрежению в сопле, куда и поступает новая порция чернил.

Важной конструктивной особенностью такого печатающего устройства является простая конструкция сопел. Причём, кроме низкой стоимости изготовления, такая конструкция устройства имеет ряд других преимуществ:

- высокая надёжность каждого сопла;
- сопла можно располагать близко друг к другу, что позволяет получить высокое разрешение при печати;
- отсутствует шум от работы печатающей головки.

Пьезоэлектрический метод, разработанный фирмой Epson, основан на управлении выбрызгиванием чернил с использованием обратного пьезоэффекта, который заключается в деформации пьезокристалла под действием электрического поля. Для реализации этого метода каждое сопло имеет свой плоский пьезокристалл, который является одной из стенок камеры печатающей головки с чернилами.

Под действием электрического напряжения пьезокристалл выгибается и уменьшает объём камеры, под действием избыточного давления жидкие чернила вылетают из сопла в виде капли и образуют на бумаге точку (см. рис. 4.99).

Рисунок 4.99. Принцип пьезоэлектрической технологии фирмы Epson

Пионер пьезоэлектрической технологии – фирма Epson, не смогла успешно соревноваться в объёме продаж со своими конкурентами Canon и Hewlett-Packard из-за сравнительно высокой технологической стоимости пьезоэлектрических печатающих головок – они дороже и сложнее, чем пузырьковые печатающие головки.

4.13.3.4. Заправка картриджей и СНПЧ

Не секрет, что основную прибыль компании, производящие принтеры, делают на расходных материалах. Зачастую стоимость принтера сознательно занижается, так как с точки зрения компаний это вложения в их будущий бизнес – продажи расходных материалов. Производители не стесняются зарабатывать на покупателях, обещая им качество печати и удобство, умалчивая о будущих затратах. Если вы хоть раз покупали картриджи для принтеров, то наверняка задумывались, с точки зрения вашего кармана или семейного бюджета, почему оригинальные стоят дороже и не стоит ли купить совместимый. В некоторых случаях совместимые расходные материалы даже не хуже, но производители всячески не рекомендуют их.

Западный принцип из разряда «увидел – купил» (картридж имеется в виду) не очень применим к нашему менталитету. Наш народ думает, пытается экономить, проявляет чудеса изобретательности. Вот почему на компьютерных «развалах и рынках», а также в объявлениях можно увидеть предложения по скупке и заправке картриджей по куда более привлекательным ценам. Если нет денег, но есть время и знания, желание экспериментировать, то можно заняться экономией.

В интернете можно найти различные инструкции по разбору картриджей того или иного типа и их заправке. За небольшую цену можно купить «чипы-обманщики», сообщающие принтеру о том, что ваш «левый» картридж есть что ни есть самый оригинальный и заправлен по самое не балуй.

Несмотря на простоту процесса, мы не рекомендуем связываться с заправкой картриджей для лазерных принтеров в домашних условиях, так как используемый в них порошок вреден для человека, а без вытяжки и аккуратности велик шанс им перепачкаться и надышаться. Здоровье дороже.

А вот заправка чернилами картриджей струйных принтеров намного безопасней. Если вы много и регулярно печатаете, то наиболее вероятно, что, выполнив несколько заправок, вы придёте к мысли об использовании СНПЧ – станции (системы) непрерывной подачи чернил. На рынке можно найти комплекты СНПЧ для большинства популярных струйных принтеров. Обычно это несколько бутылочек объёмом до сотен миллилитров, соединённых с двигающейся печатающей головкой тонкими виниловыми трубочками.

Если вы аккуратный человек, прочитали разделы выше и поняли принципы работы струйных принтеров, риск перепачкать принтер, стол и всё в округе в вашем случае минимален. Если печатаете много – экономия очевидна, если меньше раза в месяц – скорее всего, овчинка не стоит выделки.

4.13.3.5. Жёлтые точки на борьбе с фальшивомонетничеством

Жёлтые точки – метки, ставящиеся многими цветными лазерными принтерами [219] на каждую печатаемую страницу. Приглядевшись, скопление точек можно увидеть по всей странице в местах расположения текста или изображений, на расстоянии нескольких миллиметров друг от друга.

Точки едва видны невооружённым глазом и содержат в себе информацию о серийном номере принтера, а также дате и времени печати. Они обычно наносятся краской жёлтого цвета, благодаря чему незаметны на белой бумаге. Размещение данных точек является видом стеганографии. Подтверждено использование данного метода в принтерах, выпускаемых под торговыми марками Brother, Canon, Dell, Epson, Hewlett-Packard, IBM, Konica, Kyocera, Lanier, Lexmark, NRG, Panasonic, Ricoh, Savin, Toshiba, Xerox.

Введение данной меры, согласно комментариям производителей, было направлено на борьбу с фальшивомонетничеством.

4.13.3.6. 3D-принтеры

В основе технологии трёхмерной печати лежит принцип послойной 2D-печати. Сам процесс печати изображения ещё называется процессом создания объекта печати

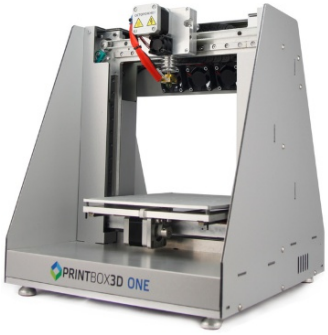
[219] https://www.eff.org/pages/list-printers-which-do-or-do-not-display-tracking-dots.

или «выращиванием». Для печати изображений сегодня используются две технологии: лазерная и струйная.

Лазерная технология, в основном промышленная:

- ***лазерная стереолитография*** – ультрафиолетовым источником света[220] точка за точкой, засвечивается жидкий фотополимер слой за слоем. После засвечивания жидкий полимер затвердевает, превращаясь в достаточно прочный пластик;
- ***лазерное сплавление*** (или склеивание, англ. melting) – при этой технологии лазер сплавляет порошок из металла или пластика, слой за слоем;
- ***ламинирование*** – печатаемая деталь создаётся из большого количества слоёв рабочего материала, которые постепенно накладываются друг на друга и склеиваются, при этом лазер используется для вырезания в каждом слое контура сечения будущей детали;

Важным элементом для струйной технологии является точная механика, позволяющая прецизионно позиционировать в пространстве печатающую головку, «выстреливающую печатаемый материал». Как следствие, «рама» хорошего принтера будет тяжёлой и заведомо больше печатаемого образца. Струйные технологии подразделяются на следующие:



- ***застывание материала при охлаждении*** – раздаточная головка выдавливает на охлаждаемую платформу-основу капли разогретого термопластика, либо это происходит непрерывный потоком, подобно выдавливанию зубной пасты из тюбика; Выдавленные капли (или масса) быстро застывают и слипаются друг с другом, формируя слои будущего объекта; В основном используется в бытовых 3D-принтерах.
- ***полимеризация*** фотополимерного жидкого пластика под действием ультрафиолетового излучения – способ похож на предыдущий, но пластик твердеет (полимеризуется) не за счёт его охлаждения до комнатной температуры, а под воздействием ультрафиолетовых лучей;
- ***склеивание*** (или спекание) ***порошкообразного материала*** – технология похожа на лазерное спекание, только порошковая основа (подчас на основе измельчённой бумаги или целлюлозы) склеивается жидким веществом (с клеевыми свойствами), поступающим из струйной головки. При этом можно воспроизвести окраску детали, используя вещества различных цветов.
- ***самозатвердеваемые керамические и бетонные смеси*** – аналогично технологии застывания пластика при охлаждении, за тем лишь различием что этой технологией печатаются крупные архитектурные модели (сооружения), например, печататься могут стены домов, при этом размеры принтера могут достигать десятков метров.
- ***биопринтеры*** – молодая технология, где будущий объект (орган для пересадки) печатается (производится) каплями, содержащими живые клетки. Последу-

[220] Обычно лазером, напрямую, либо иным источником, через шаблон.

ющее деление, рост и модификации клеток (уже без участия принтера) обеспечивают окончательное формирование объекта.

- ***пищевые принтеры*** – печатают образец подобно печати из пластика, но съедобными материалами (см. рисунок 4.79).

В первых моделях печать и движение головки производились в жидкости, предположительно из-за особенностей наносимого клеевого состава. Сегодня печать ведётся при обычных условиях – в воздухе. Разнообразие и доступность печатающих (расходных) материалов впечатляет: полноцветные гипсополимеры (CJP технология); фотополимеры (SLA и MJM технологии); ABS, PLA, HIPS пластики и резины (FDM технология), полиамиды (SLS технология).

Стоимость 3D-принтеров варьируется от 50 до 700 тыс. рублей. Цены на расходные материалы постоянно падают. Если есть желание сэкономить, то можно приобрести принтер в разобранном виде, как конструктор, и собрать его самостоятельно.

В интернете в последнее время появились десятки сайтов с базами 3D-моделей и их чисто продолжает расти. Многие такие сайты предоставляют доступ к скачиванию готовых файлов-моделей бесплатно и позволяют загружать свои.

Также хочется отметить, что в медицине могут использоваться не обязательно только биопринтеры. Например, 3D-принтеры могут печатать индивидуальные традиционные протезы и вживляемые фрагменты (имплантаты[221]) из пластика или металла. Подробнее см. http://habrahabr.ru /company/3dprintus/blog/244533/.

4.13.3.7. Другие принтеры

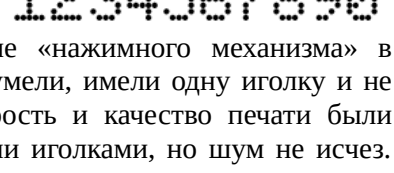
Помимо матричных, струйных, лазерных и пищевых принтеров, встречаются и другие принтеры. Обычно они не используются домашними пользователями ПК, но вы наверняка их видели. Например, ***термопринтеры*** кассовых аппаратов или весов в супермаркете, факс в офисе и др. Технология печати у термопринтеров довольно проста. Печать производится только на материалах, меняющих свой цвет от нагрева, а печатающая головка вместо распыления чернил нагревает поточечно «нужный рисунок» на

[221] Ответ справочной службы русского языка на вопрос № 247350: «*слова имплантант нет в русском языке, такое написание – орфографическая ошибка. Словари фиксируют только вариант имплантат; в последнее время в устной и письменной речи с ним конкурирует вариант имплант, но этот вариант в словарях русского языка пока не отмечен*», http://www.gramota.ru/spravka/buro/29_343947.

термоматериале, который во время печати протягивается через «проявляющий блок» небольшим моторчиком.

Матричные принтеры, пожалуй, были первыми придуманными принтерами. Принцип печати состоит в том, что небольшой ударный механизм осуществляет нажим на красящую ленту, которая, прижимаясь к листу бумаги, оставляет на нём отпечаток, некоторое подобие механической печатной машинки. Собственно, первые принтеры от неё и пошли, в них существовала так называемая головка «ромашка», на лепестках которой были рельефные начертания букв. Моторчик поворачивал нужную букву, а небольшой молоточек осуществлял нажим на красящую ленту и бумагу. Подробнее о матричной печати см. http://www.orgprint.com/wiki/tegi/Матричная_печать/.

Желая печатать графику, вместо ромашки придумали «иголочный» механизм. Теперь изображение формировалось и печаталось небольшими (различимыми на глаз) точками, а валик протяжки бумаги и каретка обеспечивали попадание «нажимного механизма» в нужные места листа. Первые принтеры характерно шумели, имели одну иголку и не отличались высокой скоростью печати. Позднее скорость и качество печати были повышены, появились принтеры с 9 и 24 печатающими иголками, но шум не исчез. Также в продаже недолгое время встречались и цветные матричные принтеры, но они довольно быстро были вытеснены струйными и лазерными.



Отличительная особенность матричных принтеров состоит в том, что за счёт нажимного механизма они могут печатать под копирку (или на самокопирующейся бумаге) сразу несколько копий. Благодаря этому качеству используются и по сей день. Их специфичный шум можно услышать в отделениях сбербанка, на кассах metro-cc и в других предприятиях.

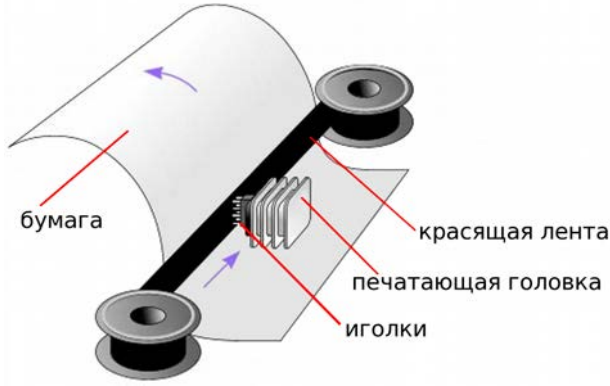


Рисунок 4.100. Матричная печать: иголки печатающей головки прижимают ленту с краской к бумаге

4.13.4. Плоттер

Плоттер (графопостроитель) – устройство вывода графической информации (чертежей, схем, карт, плакатов) на бумажные и другие листовые носители размером A1 (594×841 мм), A0 (841×1189 мм) и бо́льших форматов за счёт рулонной подачи бумаги.

До недавнего времени использовались различные типы перьевых плоттеров, однако в настоящее время в основном применяются широкоформатные ***струйные плоттеры*** (см. рис. 4.101) с термической и пьезоэлектрической технологиями печати, ***лазерные*** и ***LED-плоттеры***.

Рисунок 4.101. Плоттер

Принцип действия лазерных плоттеров аналогичен описанному для принтеров. LED [222]-плоттеры вместо лазера и системы призм используют линейку точечных светоизлучающих полупроводниковых светодиодов. Общий принцип создания изображения – тот же, что и в лазерных принтерах – изображение формируется на заряженном барабане с селеновым или органическим фоточувствительным покрытием, к которому притягивается сухой тонер, который затем переносится на проходящую под барабаном бумагу. LED-плоттеры относятся к классу растровых, каждой точке строки изображения соответствует свой светодиод (например, при разрешении 400 точек на дюйм линейка для формата A1 (24″) состоит из 9600 диодов).

К плоттерам для различных видов использования предъявляются следующие требования:

- САПР (системы автоматизированного проектирования) – вывод чёрно-белых и цветных чертежей; воспроизведение тонких линий и мелких объектов; достаточно высокая скорость вывода; возможность печати на бумаге низкого качества;
- ГИС (геоинформационные системы) – высокая точность – максимально допустимая ошибка вычерчивания карт равна 0,2 мм;
- полноцветная широкоформатная печать изображений – возможность печати как на бумаге, так и на синтетических материалах с использованием водных чернил или чернил на органических растворителях (сольвентная печать).

В свою очередь, третий тип плоттеров можно поделить по сферам использования:

- печать художественных изображений,
- профессиональная полиграфия,
- фотопечать,
- рекламная и оперативная печать.

Основные технические характеристики плоттеров:

- размер бумаги;
- разрешение при печати, точек на дюйм;
- скорость печати, м/ч или $м^2$/ч;
- память плоттера;
- наличие винчестера у плоттера и его объём;
- интерфейс подключения (USB, IEEE-1284, IEEE-1394a, Ethernet 100/1000).

В табл. 4.22 приведены некоторые модели плоттеров различного назначения и их краткая характеристика.

[222] Светодио́д или светоизлучающий диод (СД, СИД, LED – англ. *Light-emitting diode*) – полупроводниковый прибор с электронно-дырочным переходом, создающий оптическое излучение при пропускании через него электрического тока в прямом направлении.

Таблица 4.22. Характеристика различных типов плоттеров

Назначение	Модель (ширина бумаги)	Скорость печати, $м^2$/ч	Максимальное разрешение, dpi	Точность	Технология печати
САПР	HP DesignJet 800 (610 мм)	7,9	2400	±0,2% или 0,25 мм	Термоструйная
САПР	OCE 9555 (904 мм)	65	400		LED
ГИС	Mutoh Falcon RJ-800/801 (914 мм)	3	720	±0,1% или 0,25 мм	Пьезоэлектрическая
ГИС	Mutoh Falcon RJ-4100 (935 мм)	3	1440	±0,1% или 0,25 мм	Пьезоэлектрическая
Плакаты	HP DesignJet Z6100ps (107 см)	105	2400	±0,1%	Термоструйная
Наружная реклама	Mimaki JV3-160SP (160 см)	30	1440		Пьезоэлектрическая

4.13.5. Мультимедиапроектор

Мультимедиапроектор предназначен для вывода видеоинформации с компьютера на большой экран.

Размер изображения начинается со ста дюймов и практически не имеет предела. Разрешение от 800×600 у дешёвых моделей до 1920×1080 у дорогих и 4K у самых дорогих. Разрешение 1024×768 является de facto стандартным и идеально подходит как для компьютерного сигнала, так и для видео высокой чёткости; сами проекторы постоянно бьют рекорды по компактности и яркости (2000 ANSI-лм позволяют работать даже при дневном свете). Благодаря системам коррекции изображения проектор, в пределах разумного, можно установить под любым углом к экрану и настроить на отображение прямоугольного изображения.

Наиболее популярные технологии мультимедийных проекторов на сегодня – **LCD** и **DLP**.

Принципиальное различие технологий заключается в элементах, с помощью которых формируется изображение. В случае **LCD** это жидкокристаллическая матрица, через которую пропускается свет от источника, в случае **DLP** – матрица микроскопических подвижных зеркал.

4.13.5.1. DLP-проектор

DLP-технология расшифровывается как Digital Light Processing, то есть цифровая обработка света. Ключевым элементом служит DMD-матрица, уникальная разработка компании Texas Instruments. DMD – Digital Micromirror Device – цифровое микрозеркальное устройство. DMD-матрица состоит из тысяч микроскопических зеркал квадратной формы, с ребром 16 микрометров, которые, в зависимости от наличия или отсутствия сигнала, принимают одно из двух стационарных положений, тем самым либо направляя свет в оптическую систему, либо отклоняя его (см. рис. 4.102).

По количеству DMD-матриц DLP-проекторы бывают одноматричные и трёхматричные.

В первом случае между матрицей и источником света располагается трёхцветный светофильтр, своим вращением последовательно выделяющий цвета палитры RGB. На выходе оптической системы мы получаем последовательно сменяющиеся три составляющие изображения, которые чередуются настолько быстро, что человеческий глаз аппроксимирует их и получает полноцветное яркое изображение.

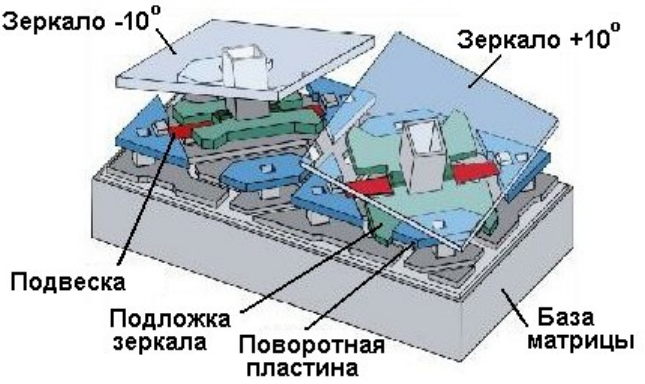


Рисунок 4.102. Зеркала DMD-матрицы

В случае с трёхкристальной системой свет от источника с помощью дихроичных призм расщепляется на три составляющие RGB, после чего каждый пучок попадает на свой кристалл, откуда отражается на оптическую систему синхронизированно с двумя другими. Таким образом, каждая составляющая изображения проецируется отдельной матрицей, что положительно сказывается на качестве получаемой картинки.

DLP-проекторы – пожалуй, самые миниатюрные представители своего класса аппаратуры: есть модели, которые без труда уместятся в дамской сумочке. К недостаткам можно отнести порой заметный «эффект радуги» – наличие радужного ореола вокруг объектов изображения. Также при длительном просмотре фильмов на однокристальном проекторе некоторые отмечают утомление глаз. По сравнению с аналогичными моделями LCD, этот тип проекторов обладает меньшей четкостью тонких деталей картинки. Однако при всём этом надо заметить, что более 70% всего рынка профессиональной проекционной техники принадлежит трёхкристальным DLP-проекторам.

Цена DLP-проекторов – от 14 тыс. руб. при яркости 2700 лм и контрастности 4000:1 (ViewSonic PJD5126 800×600) до полумиллиона рублей при яркости 6500 лм, контрастности 7500:1 и разрешении 1920×1080 (Projectiondesign F30). Последний запросто подойдёт для оборудования небольшого кинозала.

4.13.5.2. LCD-проектор

В LCD-технологии используется свойство жидкокристаллического вещества менять пространственную ориентацию молекул под воздействием электрического поля и оказывать поляризующий эффект на световые лучи. Упрощённая схема LCD-проектора показана на рис. 4.103. Свет от лампы проходит последовательно через три поляризатора, отсекающие последовательно из всего цветового спектра красный, зелёный и синий цвета. Таким образом воссоздаётся цветовая палитра RGB, необходимая при передаче изображения с компьютера или любого другого источника с RGB-выходом.

В современных проекторах используется полисиликоновая технология модуляции изображения. После разделения светового потока на три составляющие пучки цвета проходят каждый через свою полисиликоновую матрицу, каждая из которых формирует только одну цветовую составляющую изображения. Затем три цветовых компонента

изображения – красный, зелёный и синий – смешиваются системой призм (цветосмесительным призматическим блоком). Тот же блок передаёт уже полноцветное изображение на оптическую систему, которая формирует и фокусирует готовую картинку.

Рисунок 4.103. Принципиальная схема LCD-проектора

Цена LCD-проектора [223] зависит от яркости светового потока, для 2800 лм составляет около 22 000 руб., для 4500 лм – около 100 000 руб., для 7000 лм – до 450 000 руб. При этом показатели контрастности от 1000:1 до 40000:1.

Пример проектора наименьшей цены: Acer X113, LCD, 800×600, 2800 лм, 13000:1, 2,5 кг, 22 940 руб.; более мощный проектор: Sony VPL-FX500L, LCD, 1024×768, 7000 лм, 2500:1, 20 кг, 410 тыс. руб. Недавно в продаже появились проекторы разрешением 4K – цена от 600 тыс. рублей и выше.

4.13.5.3. Портативный (карманный) проектор

В результате развития технологии создания сверхярких светодиодов, последние всё чаще используются в бытовой технике, например, недавно появились портативные проекторы (с ценами в несколько десятков тысяч рублей). Модели с «плоским корпусом» по своим размерам сравнимы с карманной записной книжкой. «Объёмные» модели обычно идут в форме кубика или прямоугольника по размеру не более среднего грейпфрута. Они часто имеют встроенный аккумулятор, позволяющий работать автономно до нескольких часов, могут получать изображение как по USB/Wi-Fi/Bluetooth/HDMI, так и считывать его с MicroSD карты.

Внутри проектора имеется своя операционная система (Android или другая), позволяющая работать с файлами и видеопотоками разных форматов.

Создаваемое на белом фоне (экран, стена, потолок) изображение может иметь диагональ от 10 см до 3 метров. Разрешение, в зависимости от фирмы и модели, обычно варьируется в пределах от 640×480 до 1024×768.

[223] На апрель 2015 г.

Касаемо яркости, – чудес не бывает: для комфортного просмотра либо изображение должно быть по площади не более листа А4, либо его следует просматривать в слегка или полностью затемнённом помещении.

Похоже, фраза, что что-то «взялось с потолка» скоро утратит свой первоначальный смысл, ведь теперь без труда на потолке можно увидеть не только обучающее видео, но и любой иной контент.

4.13.6. Устройства вывода звука

Для обеспечения воспроизведения звука на ПК (аудио- и видеофайлов, радио и телевещания интернета и прочего) необходимо иметь звуковой адаптер и колонки, подключенные к нему (или наушники).

В связи с продвижением ведущими разработчиками аппаратных компонентов ПК (и прежде всего фирмой Intel) концепции ПК – домашнего медиацентра и технологии Intel Viiv (программно-аппаратная платформа, позволяющая создавать ПК с расширенными мультимедиавозможностями как компонента цифрового дома) – современные чипсеты и материнские платы имеют интегрированные аудиоподсистемы с достаточно хорошим качеством воспроизведения многоканального (7.1) звука (High Definition Audio, см. разделы 4.6 и 4.7).

Однако существуют и отдельные платы – аудиоадаптеры, обеспечивающие очень высокое качество воспроизведения звука, подключаемые к слоту PCI Express, например Auzen X-Fi Bravura 7.1, совместимая с операционными системами Windows 7/Vista/XP (см. рис. 4.104).

Рисунок 4.104. Звуковая карта Auzen X-Fi Bravura 7.1

Производители этой платы обещают обеспечить «захватывающее» качество музыки и звукового сопровождения в фильмах и играх. В основные возможности звуковой карты входит воспроизведение звука на 7.1-канальной системе с качеством до 24 бит/96 кГц и проигрывание стерео на наушниках с качеством до 24 бит/192 кГц с использованием ЦАП 120 дБ. Специальный усилитель позволяет использовать профессиональные Hi-Fi-наушники с разъёмом 1/4 дюйма (Jack). Разработанное заземление исключает появление помех от блока питания в аудио. Стоимость звуковой карты Auzen X-Fi Bravura 7.1 составляет около 5000 руб.

Стоимость качественных аудиокарт может доходить до 14 тыс. рублей, например для Creative Sound Blaster ZXR со звуком 5.1.

Количество ***акустических колонок***, подключаемых к аудиовыходам материнской платы или аудиокарты, может соответствовать количеству каналов аудиоподсистемы ПК или быть меньше. Например, при схеме звука аудиоподсистемы 7.1 можно подключить 7 колонок плюс 1 сабвуфер, но можно подключать и меньшее их количество (5.1, 4.1, 2.1). ***Сабвуфер*** – звуковая колонка для воспроизведения низкочастотных звуков, диапазон воспроизводимых частот от 25 до 150–200 Гц.

Первые звуковые карты ПК могли обеспечить вывод двухканального звука (стереозвука). Сейчас это соответствует звуковой схеме 2.0 (первая цифра – количество колонок, вторая – количество сабвуферов). Первые многоканальные акустические системы имели схему 4.0, в состав их входили 4 колонки – две фронтальные и две тыловые. В акустике 4.1 был добавлен сабвуфер. Правда, эти системы всё равно остались четырёхканальными – низкочастотные сигналы в них выделяются с помощью специального кроссовера.

Следующий тип акустических систем уже обладает полноценным 6-канальным звуком – это акустика 5.1 (см. рис. 4.105). В состав этих комплектов входят две фронтальные колонки, две тыловые, одна центральная и сабвуфер. То есть по сравнению с акустикой 4.1 появился центральный излучатель. Он нужен для соответствия формату Dolby Digital, часто используемому в фильмах, особенно на DVD-дисках. По центральной колонке передаются диалоги действующих лиц. Кроме того, акустика 5.1 может оснащаться декодерами DTS и Dolby Pro Logic. Таким образом, системы 5.1 – минимально необходимые для домашнего кинотеатра. На рис. 4.106 показана схема расположения колонок для звуковых схем 5.1 и 7.1.

Рисунок 4.105. Колонки звуковой схемы 5.1

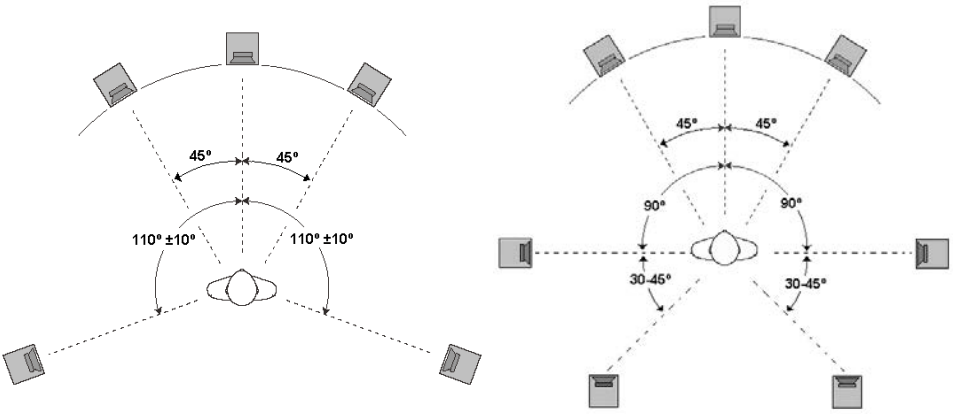


Рисунок 4.106. Расположение колонок звуковой схем 5.1 и 7.1

Последняя разработка в области звуковых схем – 8-канальные 7.1 и 7.2. В этой акустике добавились ещё два динамика – тыловые центральные. Кроме того, в системе 7.2 появился дополнительный сабвуфер, вот только «басовый» канал остался единым, так что особого эффекта эта прибавка не даёт. Если вы решитесь приобрести подобную акустику, то сможете дома насладиться звуком формата Dolby digital Surround EX или DTS Surround EX, который можно найти только в самых современных кинотеатрах.

Сокращения, используемые для обозначения различных колонок акустической системы:

- LF, RF – левая и правая фронтальные колонки;
- LS, RS – левая и правая тыловые колонки;
- C – центральная колонка (системы 5.1, 7.1 и 7.2);
- SW – сабвуфер;
- LRS, RRS – левая и правая тыловые центральные колонки (системы 7.1 и 7.2).

4.14. Оборудование компьютерных сетей

Локальная сеть (ЛС, LAN) обеспечивает передачу информации между ПК и другими сетевыми устройствами, обычно находящимися в одном или нескольких близко расположенных зданиях.

Территориально распределённая сеть (WAN) соединяет несколько локальных сетей, географически удалённых друг от друга.

Основные задачи, которые позволяет решать ЛС:

- использование автоматизированных систем управления предприятием и создание единой информационной среды предприятия;
- надёжное хранение больших объёмов информации на сервере с регулярным её резервным копированием;
- совместное использование в организациях общих ресурсов, таких как жёсткие диски, принтеры, накопители CD-ROM, серверные приложения (серверы баз данных, почтовые серверы, интернет-серверы и прочее).

Для организации территориально распределённых сетей используется коммутируемая телефонная сеть общего пользования (PSTN, Public Switched Telephone Network) с соединением через модем, линии высокоскоростной цифровой сети с предоставлением комплексных услуг (ISDN, Integrated Services Digital Network), оптоволоконная связь, спутниковые каналы связи.

При постоянном использовании организацией линии территориально распределённой сети (прежде всего, для связи с интернет-провайдером) используют выделенную телефонную линию с аналоговой или цифровой передачей данных. За выделенную линию вносится фиксированная плата. Коммутируемое соединение или ISDN предусматривают повременную оплату.

Стандарты в области локальных сетей объединяет семейство IEEE 802.xx, а также ISO 8802-1...5. Эти стандарты были созданы на основе распространённых фирменных стандартов сетей Ethernet, ArcNet и Token Ring. В частности, стандарт IEEE 802.3 описывает модификации сетей Ethernet – 10Base-5, 10Base-2, 10Base-T, 10Base-F, раздел стандарта IEEE 802.3u описывает технологию Fast Ethernet (100Base-T), IEEE 802.3ak стандартизует одну из технологий гигабитных сетей – 10GBase-CX4.

Для организации работы локальных сетей может использоваться следующее основное оборудование:

- интегрированные на материнской плате сетевые адаптеры или сетевые интерфейсные платы;

- концентраторы;
- коммутаторы;
- кабели;
- маршрутизаторы (территориально распределённые сети);
- модемы (территориально распределённые сети).

4.14.1. Сетевой адаптер (сетевая карта)

Назначение сетевого адаптера – обеспечение работы ПК в локальной сети. Для подключения к сетевому кабелю сетевая карта имеет соответствующий разъём.

Материнские платы современных ПК обычно имеют интегрированную гигабитную сетевую подсистему (10/100/1000 Мбит/с) технологии Ethernet (см. раздел 4.5 «*Материнская плата*»).

Однако выпускаются и отдельные сетевые интерфейсные платы (NIC, Network Interface Card), подключаемые к слоту PCI или USB.

4.14.2. Концентратор (hub)

В локальной сети технологии Ethernet все входящие в сеть ПК взаимодействуют с концентраторами (hub) или коммутаторами (switch). Соединённые с одним концентратором ПК образуют сегмент локальной сети. Концентраторы бывают разных видов и обычно обеспечивают соединение 12–24-х пользователей в пределах одного помещения или группы соседних помещений.

Разъём концентратора (и других сетевых устройств тоже) для подключения кабеля, приходящего от компьютера пользователя или другого сетевого устройства, называется ***портом*** или ***сетевым портом***.

При применении концентратора все пользователи делят между собой полосу пропускания сети. Пакет, принимаемый по одному из портов концентратора, рассылается во все другие порты, которые анализируют этот пакет (предназначен он для них или нет).

В настоящее время концентраторы практически не производятся, а вместо них используются простейшие неуправляемые коммутаторы (иногда их называют hub-switch, см. рис. 4.107).

Рисунок 4.107. 48-портовый неуправляемый коммутатор

4.14.3. Коммутатор (switch)

Коммутатор предоставляет каждому устройству (серверу, ПК или другому коммутатору/концентратору), подключенному к одному из его портов, всю полосу пропускания сети. Это повышает производительность и уменьшает время отклика сети за счёт сокращения числа пользователей на сегмент. Новые коммутаторы обычно поддерживают скорости передачи 10/100/1000 Мбит/с в зависимости от максимальной скорости подключаемого устройства и могут автоматически настраиваться на оптимальную скорость.

В отличие от концентраторов, осуществляющих широковещательную рассылку всех пакетов, принимаемых по любому из портов, коммутаторы передают пакеты только целевому устройству (адресату), так как знают MAC-адрес (Media Access Control) каждого подключенного устройства. Указанная функция называется «*локализацией сетевого трафика*». В результате уменьшается трафик и повышается общая пропускная способность, а эти два фактора являются критическими с учётом растущих требований к полосе пропускания сети современных сложных бизнес-приложений.

Пример описания неуправляемого коммутатора 8 D-Link DGS-1008D (стоимость около 1800 руб.): *коммутатор с 8 медными портами Gigabit Ethernet. Стандарты: IEEE 802.3 10BASE-T Ethernet, IEEE 802.3u 100BASE-TX Fast Ethernet, IEEE 802.3ab 1000BASE-T Gigabit Ethernet, ANSI/IEEE 802.3 NWay определение скорости и режима работы, IEEE 802.3x управление потоком. Количество портов: 8 портов 10BASE-T/100BASE-TX/1000BASE-T. Полный/полудуплекс для скоростей 10/100Мбит/с, полный дуплекс для скорости Gigabit Ethernet. Метод коммутации: Store-and-forward. Размер таблицы MAC-адресов: 8K записей на устройство. Изучение MAC-адресов: автоматическое обновление. Буфер RAM: 256 КБ на устройство. Размеры: 235×161,9×35,6 мм.*

Дорогие модели коммутаторов обычно имеют меньшую вероятность зависаний, повышенный размер внутренней памяти, поддерживают различные настройки, принудительное выставление режима порта, управляются по сети или с помощью специального консольного порта, могут соединяться с аналогичными им устройствами с помощью специального «stack»-порта, поддерживают различные технологии типа VLAN и др. Все расширенные функции коммутаторов обычно нужны администраторам и вряд ли заинтересуют большинство домашних пользователей.

4.14.4. Кабель

В наиболее распространённой в настоящее время сетевой технологии ***Ethernet*** сетевой кабель соединяет между собой сетевой адаптер ПК и концентратор или коммутатор (switch). Для подключения на концах кабеля обжимаются коннекторы (наиболее часто – восьмиконтактные **RJ-45**). Существует несколько типов кабелей, применяемых в различных сетевых технологиях.

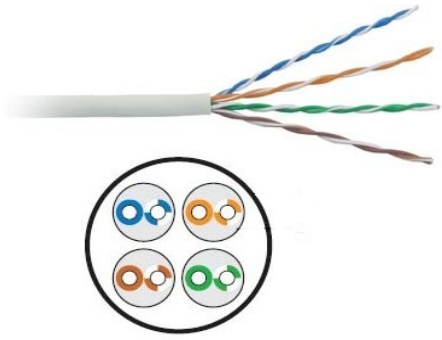


Рисунок 4.108. Кабель UTP, категория 5, 4 пары

Витая пара (TP, Twisted Pair) – основной тип кабеля в настоящее время. Бывает двух видов: неэкранированная витая пара (UTP, Unshielded Twisted Pair, см. рис. 4.108) и экранированная витая пара (STP, Shielded Twisted Pair). Оба типа кабеля состоят из нескольких пар скрученных изолированных медных проводов. Кабель типа неэкранированная витая пара стал наиболее популярным благодаря своей низкой стоимости, гибкости и простоте установки. Недостатком такого кабеля является уязвимость к электрическим помехам и шумам в линии. Кабели витая пара бывают разной категории (3, 4, 5, 5e, 6 или 7). Чем выше номер категории, тем большую скорость передачи поддерживает кабель.

Оптоволоконный кабель применяется в центральных магистральных сетях и связи между коммутаторами локальных сетей, находящимися на значительном расстоянии один от другого. Обеспечивает полную защиту от электрических помех, позволяет передавать информацию на большие расстояния. Поддерживает скорость передачи данных 10, 100, 1000 Мбит/с, 10 Гбит/с. Данные передаются с помощью световых импульсов, проходящих по оптическому волокну. Выделяют одномодовые и многомодовые оптические волокна. Недостатки – сложность монтажа и высокая стоимость.

Тонкий и толстый коаксиальный кабель – в настоящее время практически не используется. Конструкция его аналогична стандартному телевизионному кабелю.

Каждая технология ЛС (10-Мбит/с Ethernet, 100-Мбит/с Fast Ethernet или Gigabit Ethernet, 10-Gigabit Ethernet) предполагает использование одного или нескольких типов сетевого кабеля. Наиболее часто применяется кабель UTP категории 5e (см. табл. 4.23).

Таблица 4.23. Использование различных типов сетевого кабеля

Сетевой стандарт	Кабель	Число пар проводов
10BASE-T (Ethernet)	UTP, категории 3/4/5	2
100BASE-TX (Fast Ethernet)	UTP, категория 5	2
100BASE-T4 (Fast Ethernet)	UTP, категории 3/4/5	4
100BASE-FX (Fast Ethernet)	Оптоволокно	–
1000BASE-T (Gigabit Ethernet)	UTP, категория 5e	4
1000BASE-FX (Gigabit Ethernet)	Оптоволокно	–
10GBase-CX4 (10 Gigabit Ethernet)	CX4	8
10GBase-LX4 (10 Gigabit Ethernet)	Оптоволокно OM3	–

4.14.5. Маршрутизатор (router)

Основное отличие маршрутизаторов от коммутаторов и концентратов состоит в том, что они работают на «сетевом уровне»[224] и нужны для объединения сетей с адресами из разных подсетей друг с другом за счёт маршрутизации дейтаграмм и пакетов между ними. Кроме этого, в процессе своей работы они производят замену адресов отправителей и получателей в передаваемых через них на сетевом уровне пакетах и дейтаграммах, а на канальном – кадрах. (Это можно увидеть в Лабораторной работе 3.) Как следствие имеют минимум два IP-адреса. Обычно, один из адресов условно называется внешним, а другой – внутренним. «Внешних» или «внутренних» адресов может быть несколько. Количество физических сетевых портов на маршрутизаторе обычно два или более, но может быть и один.

Маршрутизаторы могут выполнять следующие функции:

- подключение локальных сетей (LAN) к территориально распределённым сетям (WAN);
- соединение нескольких локальных сетей.

Производительность маршрутизатора (объём передаваемых данных в секунду) обычно пропорциональна его стоимости. Поскольку маршрутизатор – более сложное устройство, то он может принимать решение о наилучшем маршруте доставки дан-

[224] Имеется в виду 5-уровневая сетевая модель интернета или 7-уровневая ЭМВОС (ISO/OSI-RM).

ных [225], руководствуясь такими факторами, как стоимость, скорость доставки и т. д. Кроме того, маршрутизаторы позволяют эффективно управлять трафиком широковещательной рассылки, обеспечивая передачу данных только в нужные порты.

4.14.6. Модем

Модемы позволяют пользователям ПК обмениваться информацией и подключаться к интернету по обычным телефонным линиям. Название «модем» означает «модулятор/демодулятор». Модем кодирует цифровые сигналы, поступающие от ПК, в аналоговые или цифровые сигналы[226], передаваемые по телефонной сети общего пользования, а модем на приёмном конце линии выполняет обратное преобразование (демодулирует эти сигналы), снова преобразуя их в первоначальную цифровую форму.

Модем поддерживает в каждый момент только одно соединение. Для ПК применяются встроенные и внешние модемы, а для портативных компьютеров обычно используются модемы формата PC Card.

Модемы используют специальные протоколы модуляции при передаче информации, коррекции ошибок и сжатии данных.

Протоколы передачи данных для модемов установлены Международным институтом телекоммуникаций ITU (International Telecommuni-cation Union).

Основными характеристиками протоколов передачи данных являются скорость передачи и режим: дуплекс или полудуплекс. В полнодуплексном режиме устройство может одновременно передавать и принимать данные.

Аналоговые модемы голосового диапазона используют протоколы V.42, V.42bis, V.92, позволяющие эффективно выполнять коррекцию ошибок и сжатие данных. Скорость передачи информации для протокола V.92 – 56 Кбит/с.

В большинстве мегаполисов модемы – это прошлый век, но в удалённых регионах до сих пор остаются актуальными, обеспечивая связь с миром, позволяя получать электронную почту и просматривать «неперегруженные» сайты. Модемы используют голосовой тракт и в принципе могут обеспечивать соединение точка-точка по всему миру, где есть стационарный телефон.

xDSL-модемы – это немного другая технология. Бывают двух видов ADSL и SDSL.

Технология **DSL** (Digital Subscriber Line – цифровая абонентская линия) позволяет организовать цифровую передачу данных также по телефонной линии, но только от абонента до ближайшей телефонной станции. Из-за того, что на указанном участке не используется уплотняющее оборудование, возможно использовать передачу в частотах, значительно превышающих 4 КГц, отводимые для передачи голосовых сообщений в аналоговом виде. Это позволяет одновременно использовать одну и ту же линию (медный провод) для передачи цифровых данных и обычных аналоговых телефонных разговоров.

В последнее время широкое распространение получило безлимитное подключение к интернету по технологии ADSL (Asymmetric Digital Subscriber Line – асимметричная цифровая абонентская линия, с увеличенным по скорости каналом в сторону абонента [227]) с использованием ADSL-модемов. Соединение является некоммутируе-

[225] Естественно, если есть из чего выбирать.

[226] Замечание. В сетевой терминологии есть различие между терминами «модуляция» и «манипуляция», большинству пользователей эта разница не важна.

[227] Меньшее распространение получила технология с симметричным по скорости туда и обратно каналом –

мым, то есть для подключения к провайдеру достаточно включить настроенный модем и компьютер. Технология ADSL обеспечивает скорость нисходящего потока данных (от абонента к провайдеру) в пределах от 1,5 Мбит/с до 8 Мбит/с и скорость восходящего потока от 640 Кбит/с до 1,5 Мбит/с. Технология ADSL 2+ поддерживает скорость к абоненту до 20 Мбит/с. Скорость обычно ограничивается провайдером в зависимости от суммы ежемесячной абонентской оплаты за соединение.

4.15. Оборудование беспроводных сетей

Беспроводное соединение компьютеров и периферийного оборудования получает всё большее распространение.

Беспроводные локальные сети (Wi-Fi) позволяют организовать связь между ПК там, где затруднено кабельное соединение или необходима полная мобильность. При этом они совместимы с проводными сетями. Необходимо учитывать возможность создания беспроводных соединений ПК при проектировании любых сетей – от малого офиса до предприятия.

В настоящее время на рынке сетевого оборудования Wi-Fi представляет четыре стандарта: **IEEE 802.11a, IEEE 802.11b, IEEE 802.11g** и **IEEE 802.11n**.

Стандарт IEEE 802.11a позволяет передавать данные на скорости до 108 Мбит/с. Эти беспроводные сети работают на частоте 5 ГГц и обеспечивают возможность шифрования с использованием WEP.

Стандарт IEEE 802.11b позволяет передавать данные на скорости до 11 Мбит/с и работает на частоте 2,4 ГГц по протоколу широкополосной передачи данных – DSSS.

Стандарт IEEE 802.11g использует скорость передачи до 54 Мбит/с и работает на частоте 2,4 ГГц – той же, что и устройства стандарта IEEE 802.11b, что обеспечивает совместимость этих стандартов.

Последний на сегодня стандарт **IEEE 802.11n** позволяет передавать данные на скорости до 600 Мбит/с. В России этот стандарт сертифицирован, а оборудование стандарта 802.11n разрешено к применению на территории России в диапазонах 2400–2483.5, 5150–5350 и 5650–5725 МГц [228].

Разработчики спецификации 802.11n позаботились о том, чтобы компоненты на её базе сохраняли совместимость с устройствами стандарта 802.11b или 802.11g в диапазоне 2,4 ГГц и с устройствами 802.11a – в диапазоне 5 ГГц. В новых сетях 802.11n ещё долгое время будет работать множество прежних беспроводных клиентов, так что при развёртывании беспроводных ЛВС администратору следует обязательно предусмотреть их поддержку.

Оборудование беспроводных сетей включает сетевые адаптеры, точки доступа (Access Point), беспроводные маршрутизаторы (см. рис. 4.109).

В настоящее время Wi-Fi-адаптерами оснащаются все ноутбуки и другие мобильные устройства (планшетные компьютеры, КПК, коммуникаторы).

SDSL.

[228] Приказ Министерства связи и массовых коммуникаций России от 14 сентября 2010 г. № 124 «Об утверждении Правил применения оборудования радиодоступа. Часть I. Правила применения оборудования радиодоступа для беспроводной передачи данных в диапазоне от 30 МГц до 66 ГГц».

Wi-Fi-адаптеры для настольных ПК выпускаются для подключения к PCI-слоту или USB-порту. Существуют также материнские платы настольных ПК с интегрированными Wi-Fi-адаптерами.

В разных странах существуют различные ограничения по использованию частотного спектра и различные разрешённые мощности сигналов. Выбирая в настройках Wi-Fi-устройств «страну использования», будьте внимательны, так как в некоторых случаях неправильного выбора вы можете выйти за предел разрешённой мощности или за пределы разрешённого частотного диапазона, что по этическим соображениям некорректно к другим участникам радиообмена, а во-вторых, может привести к наложению административного взыскания (штрафа). (См. подробнее раздел 4.11.1 «*Каналы Wi-Fi*».)

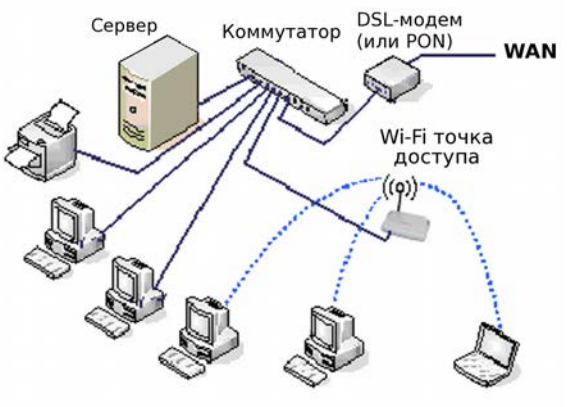


Рисунок 4.109. Пример локальной сети с использованием Wi-Fi

Стоимость Wi-Fi-адаптеров (как USB, так и PCI – от 400 до 2500 руб.) несколько выше, чем проводных адаптеров (250–500 руб.). Точка беспроводного доступа стоит 900–5000 руб.

Замечание. Хотите узнать, где находится ваша домашняя точка доступа, и увидеть её на карте мира? Если да, то введите её MAC-адрес (обычно написан на наклейке снизу точки доступа рядом с её серийным номером) по адресу http://samy.pl/mapxss.

4.15.1. Каналы Wi-Fi

В мегаполисах, особенно в кварталах с 22-этажными зданиями, можно запросто наблюдать ситуацию, что Wi-Fi-оборудование работает нестабильно. Причина этому явлению – большая плотность работающих wi-fi-точек доступа. Оборудование соседей справа, слева, сверху, снизу и от корпуса вашего дома сбоку может влиять на используемый вашими компьютером и точкой доступа частотный спектр. Чаще сбои и пропадания скорости наблюдаются в часы наибольшей нагрузки (обычно вечером), а бывает, что и всегда.

Чтобы решить проблему и избавиться от «провалов» в связи, давайте разберёмся с первопричинами.

Весь частотный спектр, отведённый для Wi-Fi, поделён на каналы подобно радио- и телеэфиру. Например, радиостанции FM-диапазона на одной местности обычно идут с шагом в 0,4 МГц или больше, чтобы не мешать друг другу. Так как мощности передатчиков для Wi-Fi значительно меньше эфирных, плюс желаемый результат иметь как можно больше непересекающихся каналов в теории, производители закладывают в оборудование поддержку большого их числа на всевозможные жизненные случаи. Впоследствии, выбрав в настройках оборудования страну, часть каналов можно заблокировать [229].

Подробнее разобраться с ситуацией вам поможет рис. 4.110.

[229] http://en.wikipedia.org/wiki/List_of_WLAN_channels.

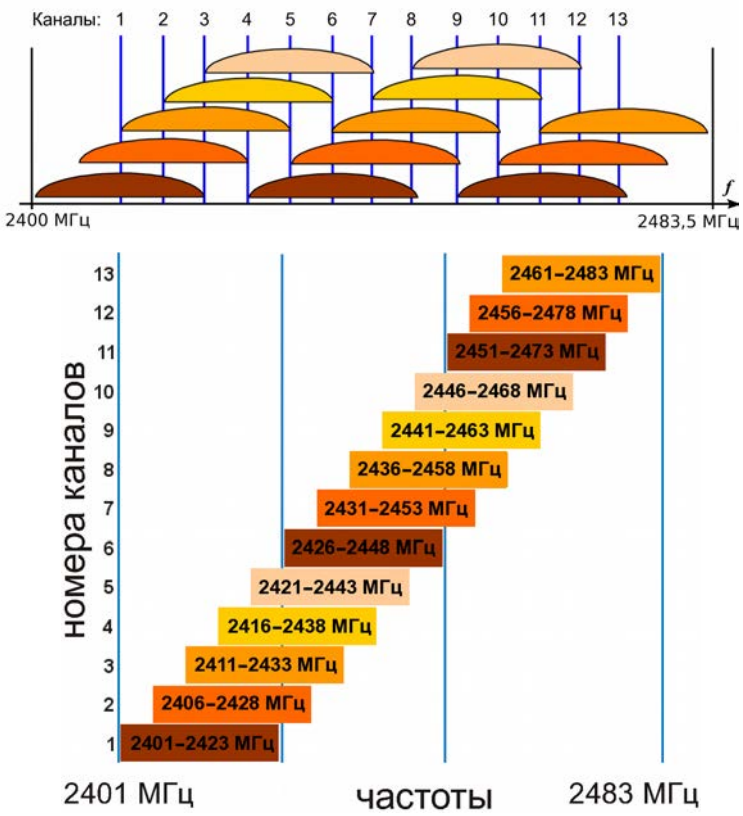


Рисунок 4.110. Распределение Wi-Fi-каналов по 22 МГц шириной (802.11 b/g и 802.11 n в режиме b/g)

Положение с 5 ГГц диапазоном «802.11 a» аналогично, с тем различием, что в разных странах разные каналы заблокированы, поэтому на рис. 4.111 нумерация каналов непоследовательная.

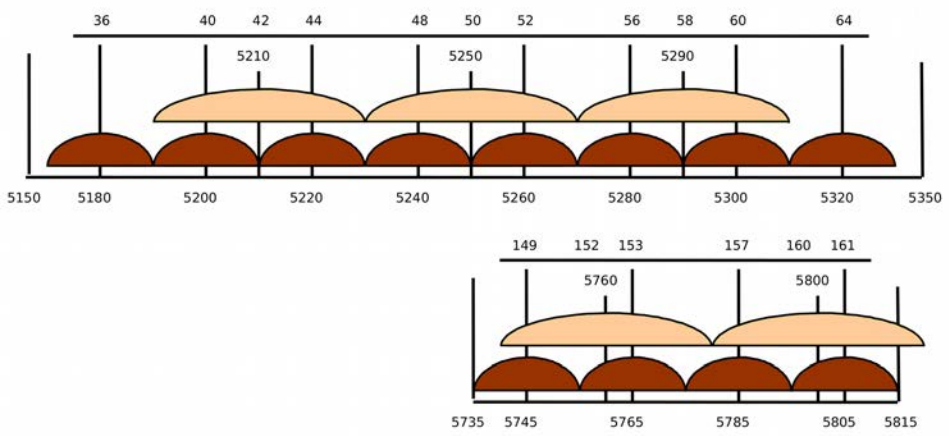


Рисунок 4.111. Распределение Wi-Fi-каналов по 20 и 40 МГц шириной (802.11 a и 802.11 n в режиме a)

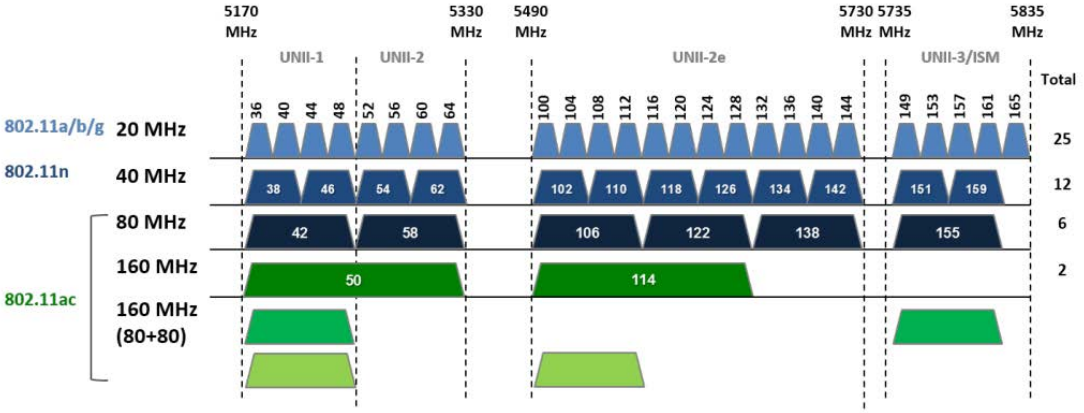


Рисунок 4.112. Распределение Wi-Fi-каналов (сравнение с режимом 802.11ac)

Даже не имея диагностического оборудования, в ситуации, описанной выше, можно найти выход, а именно – сменить рабочий канал. Также не забывайте физические законы, что мощность волны падает обратно пропорционально квадрату расстояния до излучателя. Если ваши соседи вменяемые, то можно и каналы полюбовно развести, и точки доступа разместить подальше друг от друга.

Не следует забывать, что радиоизлучение ВЧ диапазона особенно вредно детям и взрослым репродуктивного периода. Если есть возможность использовать проводной интернет – используйте и не включайте лишний раз точку доступа.

4.16. Дополнительное оборудование

Согласно ГОСТ 13109–97 (взамен ГОСТ 13109–87), на всей территории Российской Федерации определены следующие нормы для электропитающей сети общего пользования: напряжение 220 В ± 5% (предельные значения ± 10%); частота 50 Гц ± 0,2 Гц (предельные значения ± 0,4 Гц); коэффициент нелинейных искажений формы напряжения менее 8 % (длительно) и менее 12 % (кратковременно). Так как продаваемые ПК и даже ноутбуки (во время зарядки) являются потребителями электрического тока, то для продажи в розничной сети («вбелую», то есть официально) они выпускаются исходя из данных вышеуказанного ГОСТ 13109–97, иногда с небольшим запасом.

Следует понимать, что в жизни в случае форс-мажора возможны отклонения от требований ГОСТа и возникновение следующих формально классифицируемых неполадок:

- авария сетевого напряжения (напряжение в питающей сети полностью пропало);
- высоковольтные импульсные помехи (резкое увеличение напряжения до 6 кВ продолжительностью от 10 до 100 мс);
- долговременные и кратковременные подсадки и всплески напряжения;
- высокочастотный шум (высокочастотные помехи, передаваемые по электросети);
- побег частоты (отклонение частоты более чем на 3 Гц).

Далее будут рассмотрены устройства «электрической» защиты ПК от указанных неполадок.

4.16.1. Сетевой фильтр

Сетевой фильтр – это самое простое устройство «пассивной» защиты, спасающее от короткого замыкания на стороне потребителя (в зависимости от модели потребление тока более 10–20 А) и от импульсных помех со стороны электропитающей сети с энергией скачка обычно до 90 Дж.

Рисунок 4.113. Фильтр питания Power Cube

Внешне сетевой фильтр (см. рис. 4.113), выглядит как удлинитель с кнопкой или без, однако внутри он дополнительно имеет схему защиты, в которую входят автомат наподобие тех, что стоят в щитке перед вашей квартирой, и фильтр. В зависимости от модели в качестве фильтра может использоваться ферритовый сердечник или кольцо, LC-цепочка, варистр или всё вместе. Если вы обращали внимание, то некоторые кабели, например идущие к монитору (см. рис. 4.114), имеют бочкообразные утолщения на своих концах или одном конце – это и есть ферритовые фильтры.

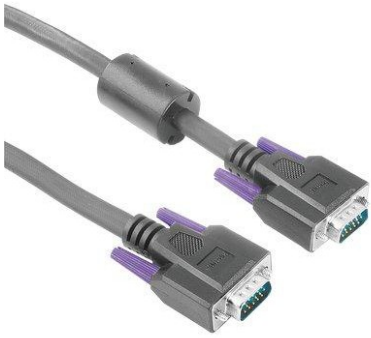


Рисунок 4.114. Ферритовые фильтры (слева несъёмный, справа продающиеся отдельно для самостоятельной установки)

Если вам доведётся как-нибудь разобрать сетевой фильтр, то внутри него вы можете увидеть целую схему, собранную на плате. Типичная схема защиты сетевого фильтра представлена на рис. 4.115, более сложная – на рис. 4.116.

Рисунок 4.115. Наиболее простая схема защиты сетевого фильтра

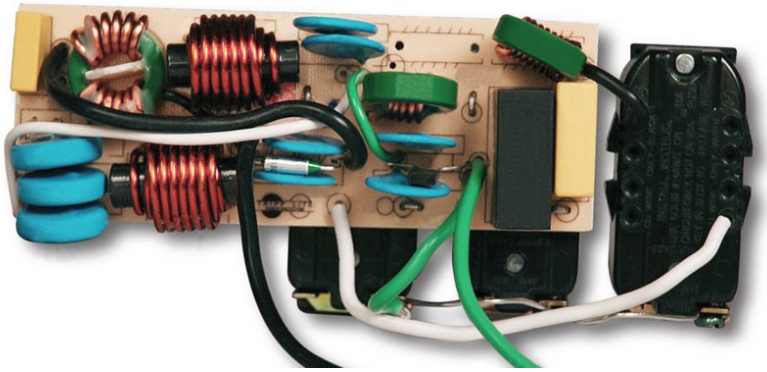


Рисунок 4.116. Более сложная схема защиты

Обычно стоимость сетевых фильтров немногим дороже продаваемых удлинителей. Наиболее популярными фильтрами на территории России являются фильтры, производимые российской компанией ZIS Company, основанной в 1992 году, под торговой маркой Pilot®.

Ознакомиться с результатами тестирования и сравнения различных сетевых фильтров можно по адресу http://www.defender.ru/press/publications/2008/59/ [230].

Средняя стоимость сетевых фильтров колеблется от 300 до 1900 рублей. Предполагаем, что значительную часть стоимости наиболее дорогих моделей составляет не качественная и продуманная схема защиты, а внешний «космический дизайн».

Замечание. Смеем предположить, что если в ваш компьютер будет прямое попадание молнии, то ничто ему уже не поможет. Обнадёживающим фактором является то, что данное событие маловероятно. Так, оценив риски, компания APC пошла на следующий рекламный трюк. В части стран (к сожалению, не в России [231]) она предложила выплачивать полную стоимость повреждённой электричеством техники пользователя, если техника была защищена с помощью оборудования фирмы и оно, будучи правильно установленным, «не помогло».

4.16.2. Стабилизатор напряжения

При долговременных «проседаниях» или постоянно повышенном напряжении сети электропитания используются сетевые стабилизаторы. Конечно, правильнее было бы разбираться с поставщиками электроэнергии и требовать от них стабильности, но в российских реалиях это может превратиться в ставшую уже анекдотической ситуацию: «*Вам шашечки или ехать?*»

Основная составляющая стабилилизатора – автотрансформатор. Определить его наличие можно по весу (от 5 кг и тяжелее). Для переключения нагрузки к нужному выходу трансформатора используется электронная схема, обычно имеющая индикацию на передней панели прибора. Обычно это размеченная линейка светодиодов, реже цифровое отображение входного напряжения с помощью светодиодной или ЖК-панели.

В зависимости от мощности к стабилизатору, помимо компьютера, могут подключаться и различные бытовые приборы, холодильники, насосы для полива огорода и

[230] Также см. видеролик с описанием этапов производства сетевых фильтров: http://www.youtube.com/watch?&v=1yWHBHXQZFE.

[231] Возможно, потому, что наш народ очень изобретательный.

прочее. Часто имеет встроенный автомат защитного отключения и пассивную схему фильтрации ВЧ-помех.

Отечественные производители с успехом выпускают данное оборудование в мощностях от 1 до 80 кВА, например ООО «Энергия» [232], г. Псков (см. рис. 4.117).

Рисунок 4.117. Отечественный сетевой стабилизатор Progress 2000T (эконом серия), рассчитанный на потребляемую мощность 2000 ВА

4.16.3. Источник бесперебойного питания (ИБП)

Сетевой стабилизатор – вещь хорошая, но не спасает от ситуации, когда напряжения вообще нет. В этом случае на серверах и на тех компьютерах, где предъявляются высокие требования к сохранности информации, используют источники бесперебойного питания (ИБП, UPS, Uninterruptible Power Supply).

Бесперебойность выдачи напряжения на выходе ИБП обеспечивается за счёт использования в его составе аккумуляторной батареи. Чем больше ёмкость батареи и меньше потребляемая нагрузка, тем выше время автономной работы ИБП.

Для домашнего использования мощность батареи (и ИБП в целом) выбирают из расчёта обеспечения 10–15 минут работы в режиме 70%-ной нагрузки. Подразумевается, что за это время можно успеть принять необходимые меры – сохранить документы, закрыть программы и выключить компьютер.

Для серверного оборудования, в зависимости от поставленных задач, может закладываться большая ёмкость, исходя из времени перехода на резервную схему, если такая есть, или запуска дизельного генератора и прочего.

В продвинутых моделях ИБП обычно имеется USB- или RS-232-порт для подключения к компьютеру и уведомления его о состоянии работы. Некоторые модели также могут подключаться к сети Ethernet и использовать протокол SNMP.

Несмотря на простоту задумки, внутреннее схемоустройство ИБП бывает разным. Важными показателями, обуславливающими выбор схемы построения ИБП, являются время переключения нагрузки на питание от аккумуляторных батарей и время работы от аккумуляторной батареи. Исходя из этих требований, используются два вида классификации: по мощности и типу.

[232] http://www.energostab.ru/.

4.16.3.1. Классификация ИБП по мощности

ИБП малой мощности подключаются непосредственно к защищаемому оборудованию и питаются от электрической сети через штепсельные розетки. Данные устройства изготавливаются в настольном (см. рис. 4.118), реже напольном исполнении, а также для монтажа в стойку и выпускаются в диапазоне мощностей от 250 до 3000 ВА[233].

Рисунок 4.118. ИБП в виде удлинителя фирмы APC, модель BE550G-RS. Максимальная выходная мощность: 330 Ватт (550 ВА)

Рисунок 4.119. Отечественный напольный ИБП «Штиль», модель HT1101S, максимальная выходная мощность: 0,8 кВт (1 кВА)

ИБП средней мощности имеют встроенный блок розеток для подключения защищаемого оборудования либо подключаются к групповой розеточной сети, выделенной для питания защищаемых электроприёмников. К питающей сети эти ИБП подсоединяются кабелем от распределительного щита через защитно-коммутационный аппарат. Они рассчитаны на установку как в специально приспособленных электромашинных помещениях, так и в технологических комнатах, где размещается инфокоммуникационное оборудование и допускается постоянное присутствие персонала. Типичный диапазон их мощностей – от 3 до 40 кВА.

[233] Полная мощность и активная мощность – разные физические величины, имеющие размерность мощности. Для того чтобы на маркировках электроприборов или в технической документации не требовалось лишний раз указывать, о какой мощности идёт речь, и при этом не спутать эти физические величины, в качестве единицы измерения полной мощности используют вольт-ампер вместо ватта. Форма записи единицы измерения «В·А» весьма удобна, поскольку отражает физический смысл величины полной мощности. В качестве единицы измерения активной мощности используется ватт.

ИБП большой мощности подключаются к питающей сети с помощью кабеля от распределительного щита через защитно-коммутационный аппарат, а к защищаемому оборудованию – через выделенную групповую розеточную сеть. Они имеют напольное исполнение для размещения в специально приспособленных электромашинных помещениях. Типичный диапазон мощностей охватывает значения от 10 кВА до нескольких сотен (на рынке имеются модели мощностью до 800 кВА, обычно совмещаемые с дизельными или бензиновыми генераторами). Параллельные системы ИБП и энергетические массивы могут иметь мощности до нескольких тысяч киловольт-ампер.

4.16.3.2. Классификация ИБП по типу

По принципу устройства ИБП можно отнести к трём основным типам:

Резервная схема (пассивного типа, англ. off-line, standby, также Back UPS). Принцип действия ИБП пассивного типа с режимом работы «вне линии» (off-line) заключается в питании нагрузки (ПК) от энергосети и быстром переключении на внутреннюю резервную схему, получающую электрическую энергию от собственных аккумуляторов с помощью простого инвертора, при отключении питания или выходе напряжения за пределы допустимого диапазона. При появлении напряжения в пределах нормы снова переключает нагрузку на питание от первичной сети (см. рис. 4.120).

При работе от первичной электрической сети ИБП (большую часть времени) работает как обычный сетевой фильтр.

Время переключения обычно составляет около 4–12 мс, что вполне достаточно для большинства технических средств инфокоммуникаций с импульсными блоками питания.

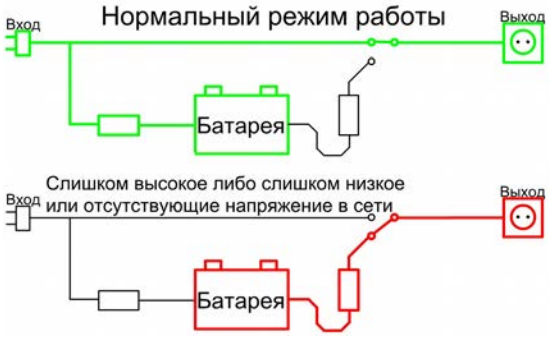


Рисунок 4.120. Принцип работы схемы резервного типа

Интерактивная схема (активного типа, англ. Line-Interactive). Источники бесперебойного питания активного типа с режимом работы «на линии» (on-line) – устройство аналогично предыдущей схеме; дополнительно на входе присутствует ступенчатый стабилизатор напряжения на основе автотрансформатора, позволяя получить регулируемое выходное напряжение (VI по классификации МЭК). При работе в нормальном режиме такие ИБП не корректируют частоту, пассивные фильтры фильтруют входящее переменное напряжение. При пропадании напряжения ИБП переходит на питание от инвертора, аналогично предыдущему (см. рис. 4.121).

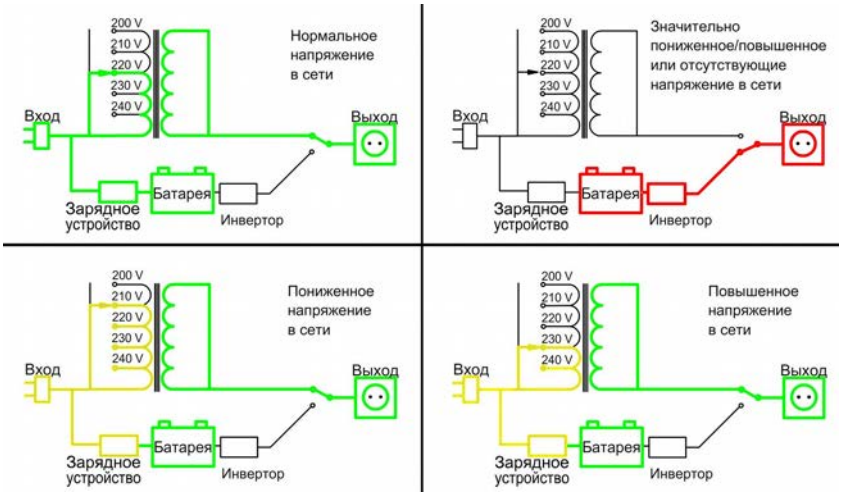


Рисунок 4.121. Принцип работы интерактивной схемы

Инверторы некоторых моделей линейно-интерактивных ИБП выдают напряжение как прямоугольной или трапецеидальной формы, аналогично предыдущему варианту, так и синусоидальной формы. Время переключения меньше, чем в предыдущем варианте, так как осуществляется синхронизация инвертора с входным напряжением. КПД ниже, чем у резервных.

Двойное преобразование (англ. *online* – «на линии») – используется для питания оборудования, предъявляющего повышенные требования к качеству сетевого электропитания. Принцип работы состоит в двойном преобразовании рода тока (переменный → постоянный → переменный). Сначала входное переменное напряжение преобразуется в постоянное, затем обратно в переменное напряжение с помощью обратного преобразователя (инвертора). При пропадании входного напряжения переключение нагрузки на питание от аккумуляторов не требуется, поскольку аккумуляторы включены в цепь постоянно, поэтому для этих ИБП параметр «время переключения» не имеет смысла. С другой стороны, когда батареи заряжены, данные ИБП могут пускать электричество в обход всей схемы, имеющей «нулевое» время переключения, в этом случае реальная задержка переключения равна скорости перещёлкивания контактов реле (см. рис. 4.122).

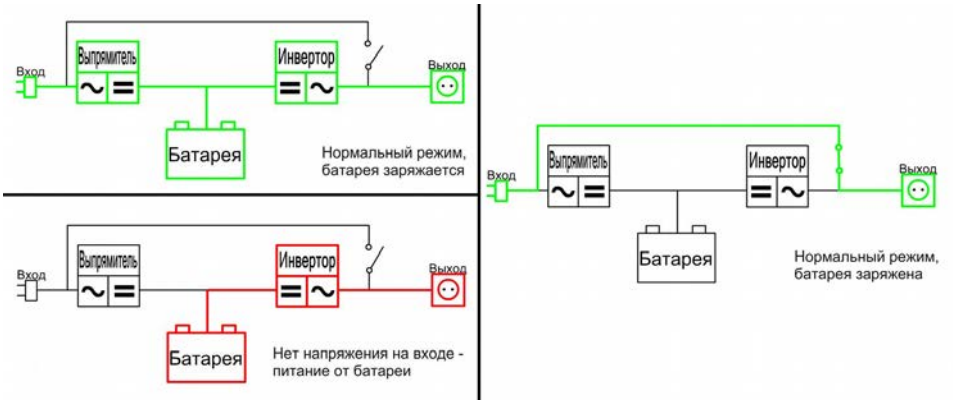


Рисунок 4.122. Принцип работы схемы с двойным преобразованием

ИБП двойного преобразования имеют невысокий КПД (от 80 до 96,5%) в режиме on-line, из-за чего отличаются повышенным тепловыделением и уровнем шума. Однако у современных ИБП средних и высоких мощностей ведущих производителей предусмотрены разнообразные интеллектуальные режимы, позволяющие автоматически подстраивать режим работы для повышения КПД вплоть до 99%. В отличие от двух предыдущих схем, способны корректировать не только напряжение, но и частоту (VFI по классификации МЭК).

Среди зарубежных компаний на 2014 год APC by Schneider Electric является основным производителем и уже долгое время с явным отрывом сохраняет лидерство на рынке ИБП.

4.16.4. «Грозозащита»

Глупо думать, что от прямого попадания молнии ваше оборудование может что-то спасти. Разве что уменьшить ущерб можно, сделав грозоотвод, резервную копию данных и оформив страховку на указанный случай. Даже использование оптики (диэлектрика) не поможет – расплавится, а что и говорить про проводники. Однако если молния ударит где-то рядом или случайно на проводе, приходящем в компьютерную систему, образуется наводка, то от неё защититься можно и даже нужно. Как это сделать, будет рассказано далее.

В реальных условиях эксплуатации электронного оборудования в его цепях могут возникать различные виды электрических перегрузок, наиболее опасными из которых являются перегрузки по напряжению (перенапряжения), создаваемые электромагнитными импульсами естественного происхождения (за счёт мощных грозовых разрядов), электромагнитными импульсами искусственного происхождения (за счёт излучений радиопередающих устройств, высоковольтных линий передачи, сетей электрифицированных железных дорог, метро и т. п.), а также за счёт внутренних переходных процессов в оборудовании при его функционировании (например, при переключениях индуктивных нагрузок) и электростатических разрядов (ЭСР).

Не следует забывать, что во время грозы пользоваться электрическими устройствами (в том числе сотовыми телефонами) опасно для собственной жизни, находиться с зонтиком на открытой местности, как и без, не менее опасно, но что и говорить, плотность населения в мегаполисах и их ритм жизни под страхом смерти не позволяют отключать большинство электронных систем. В результате воздействие электромагнитного импульса (ЭМИ) естественного и искусственного происхождения на электронные компоненты может привести к изменению их параметров за счёт как непосредственного поглощения ими энергии, так и воздействия на них наведённых в цепях импульсов токов и напряжений.

Рисунок 4.123. Оборудование защиты сетевых интерфейсов

Как защищать оборудование по сети электропитания, было рассмотрено в предыдущем разделе, но как быть, если к компьютеру подключены и другие провода, например сетевой кабель Ethernet – наиболее частый случай подключения к сети Интернет? Если это проводник, то по нему можно также передавать электричество «с избытком», а значит, сжечь сетевую карту или больше... На этот случай существует специальное оборудование для защиты, например APC ProtectNet (см. рис. 4.123).

Рассмотрим, как оно устроено внутри, ведь платим мы не только за дизайн и упаковку, но и за внутреннее устройство и функциональные возможности.

Прежде чем мы залезем внутрь устройства, показанного на рис. 4.123, рассмотрим, как в схемотехнике выполняется диодная защита.

Во-первых, диоды применяются для защиты устройств от неправильной полярности включения.

Во-вторых, для защиты входов схем от перегрузки, защиты ключей от пробоя ЭДС самоиндукции, возникающей при выключении индуктивной нагрузки (при подключении электромагнитных замков и реле параллельно им всегда вешают диод, иначе может сгореть не только блок питания) и т. п.

В-третьих, для защиты входов аналоговых и цифровых схем от перегрузки используется цепочка из двух диодов, подключенных к шинам питания в обратном направлении, защищаемый вход подключается к средней точке этой цепочки (см. рис. 4.123).

Вспомните из школьного курса физики, как устроена вольт-амперная характеристика (ВАХ) полупроводникового диода. При нормальной работе диоды закрыты и почти не оказывают влияния на работу схемы. При «уходе» потенциала входа за пределы питающего напряжения один из диодов открывается и «шунтирует» вход схемы, ограничивая таким образом допустимый потенциал входа диапазоном в пределах питающего напряжения плюс прямое падение напряжения на диоде [234]. Такие цепочки могут быть уже включены в состав ИС на этапе проектирования кристалла либо предусматриваться при разработке схем узлов, блоков, устройств. Выпускаются готовые защитные сборки из двух диодов в трёхвыводных «транзисторных» корпусах.

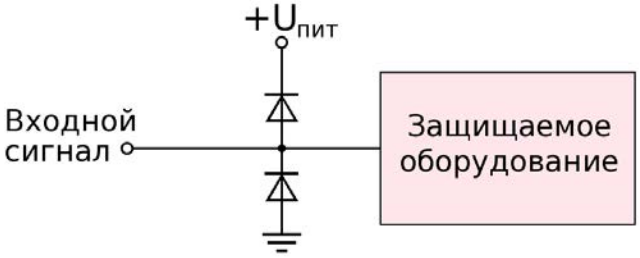


Рисунок 4.124. Защита входа от перенапряжения двумя диодами

Если быть точнее, в обратном включении используются TVS-диоды (от англ. transient voltage suppressors) (или так называемые супрессоры, защитные диоды, ограничители напряжения), схемотехнически обозначаются немного по-другому. Одним из основных параметров TVS-диодов является время реакции. Время реакции на обратной ветке ВАХ (ветка лавинного пробоя) составляет несколько пикосекунд.

[234] Всю нагрузку на себя берёт блок питания, пытаясь стабилизировать напряжение. Как вы понимаете, в этом случае «сжечь» более мощный блок питания будет сложнее.

Собственно, если входящих проводников 8, то схем, изображённых на рисунке 4.124 параллельно друг другу, будет 8, а диодов – соответственно 16 (см. рис. 4.125), плюс ещё один общий.

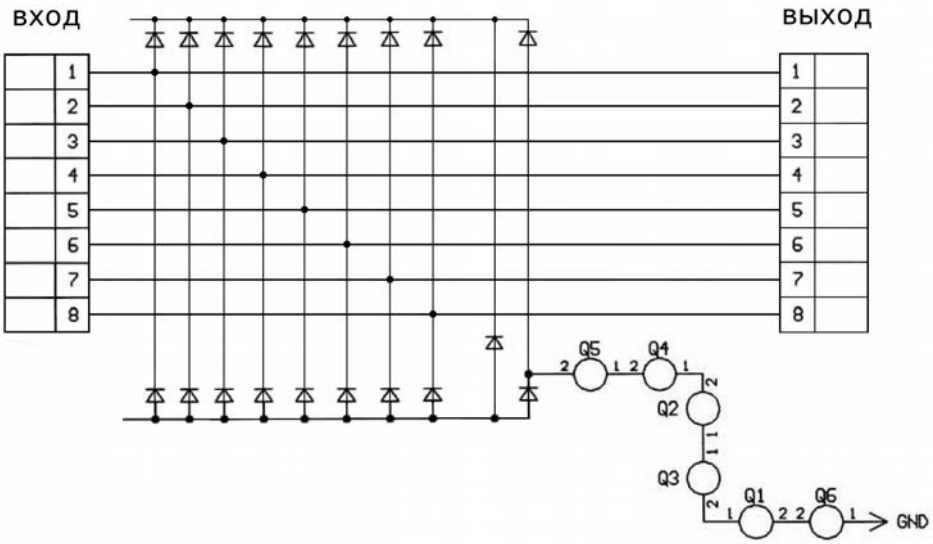


Рисунок 4.125. Схема сетевой защиты

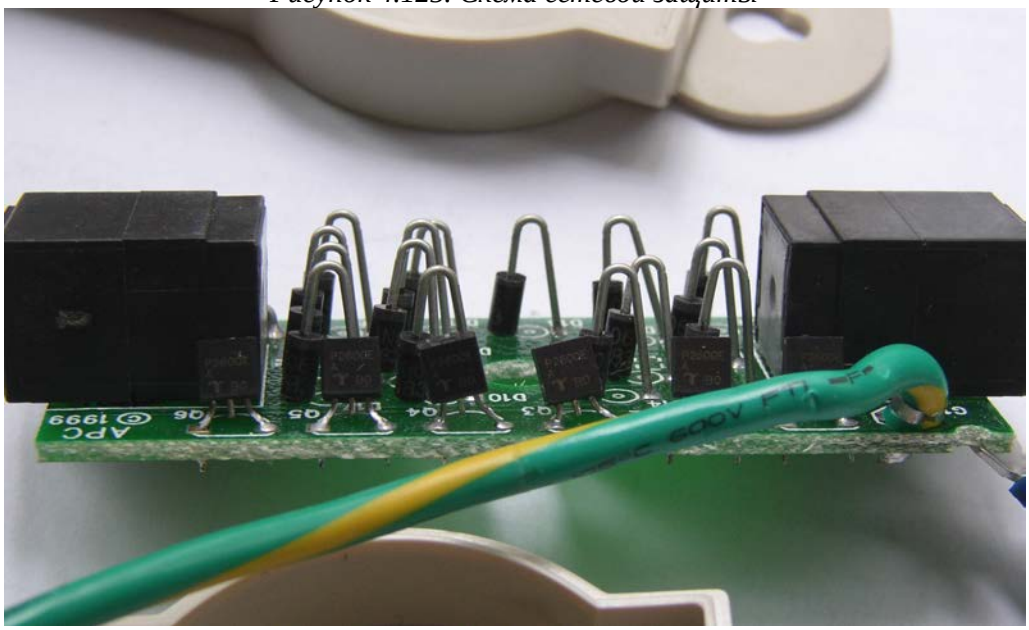


Рисунок 4.126. Внутренняя плата устройства APC ProtectNet

Дополнительно к диодам, для отвода излишнего напряжения на землю, используется несколько последовательно соединённых элементов Q1–Q6 – твердотельных схем автоматического шунтирования при превышении максимально допустимого уровня напряжения. Модель «P2600E» от фирмы Teccor Electronics. Как вариант в аналогичных схемах, вместо них может использоваться газонаполненный разрядник (GDT от англ. Gas Discharge Tube).

О том, как изготовить подобную схему защиты сетевого оборудования самостоятельно, см. http://greennet.ucoz.ru/publ/2-1-0-8.

4.17. Контрольные вопросы к главе 4

1. Назовите основные поколения ЭВМ и их свойства.
2. В чём состоит идея миниатюризации?
3. В каком веке появились работы по алгебре логики, или булевой алгебре?
4. В каком году была опубликована работе Джорджа Буля «Законы мышления»?
5. Сколько различных вариантов реализаций может существовать для чёрного ящика с 1 входом и 1 выходом в рамках булевой алгебры?
6. То же самое, но для чёрного ящика с *n* входами и 1 выходом?
7. Нарисуйте таблицу истинности для логического элемента «И».
8. Нарисуйте таблицу истинности для логического элемента «ИЛИ».
9. Нарисуйте таблицу истинности для логического элемента «НЕ».
10. Что такое триггер?
11. Нарисуйте схему соединения двух логических элементов «ИЛИ-НЕ», чтобы получился триггер.
12. Приведите таблицу истинности (переходов состояний) для RS-триггера.
13. Как из RS-триггера получить D-триггер?
14. Какое соединение (соединения) триггеров называется регистром?
15. Что такое машина Тьюринга?
16. Назовите примеры программного обеспечения для эмуляции работы электрических схем.
17. Что такое архитектура фон Неймана?
18. Перечислите основные рекомендации, предложенные Джоном фон Нейманом для разработчиков ЭВМ.
19. Расшифруйте аббревиатуру АЛУ.
20. Каково назначение персонального компьютера?
21. Произведите классификацию современных ПК, является ли ноутбук ПК?
22. Что такое аппаратное обеспечение ПК?
23. Какое устройство ПК выполняет обработку информации?
24. В чём отличие CISC- и RISC-архитектур?
25. Из чего сделан процессор, как вы понимаете закон Мура?
26. От чего зависит производительность процессора?
27. Как выглядит эволюционный ряд процессоров (например, Intel)?
28. Назовите марки наиболее производительных процессоров.
29. Что такое «разгон» процессора?
30. Что такое кэш-память процессора, зачем она нужна?
31. Сколько ядер содержат современные процессоры?
32. Назовите фирму, производящую процессоры с более чем 100 ядрами.
33. Совместимы ли между собой по разъёму современные процессоры фирм Intel и AMD?
34. Что такое режим работы процессора?
35. Зачем процессору термопаста, радиатор и вентилятор?
36. Что такое чипсет?
37. Чем отличаются чипсеты для процессоров Core i5, Core i7 и Core 2 Duo?
38. Какие интегрированные подсистемы может иметь чипсет?
39. Что такое материнская (или системная) плата?
40. Какие большие микросхемы размещены на материнской плате?

41. Какие задачи решает микросхема BIOS?
42. Перечислите слоты, внутренние и внешние порты современной материнской платы.
43. Каково назначение основной оперативной памяти?
44. В чём особенность работы динамической памяти, основное отличие от статической?
45. Какие типы устройств хранения информации вы знаете?
46. Что означает термин «винчестер», какая у него история возникновения?
47. Какой принцип записи информации используется в винчестерах?
48. Какие интерфейсы используются на современных жёстких дисках, используемых в ПК?
49. Какая скорость вращения шпинделя у современных жёстких дисков?
50. Что такое S.M.A.R.T.?
51. Поясните аббревиатуры DAS, СХД, SAN, NAS.
52. Какие ограничения встречаются на пути улучшения характеристик современных жёстких дисков?
53. Какой принцип записи информации использует flash-память?
54. В чём отличие NAND и NOR flash-памяти?
55. Как устроены и как работают твердотельные накопители (SSD)?
56. Назовите особенности хранения файлов на SSD-накопителях.
57. В чём особенность записи информации на оптические диски?
58. Зачем нужен лазер для работы с оптическими дисками?
59. Какой объём информации можно записать на CD-, DVD- и BD-диски?
60. Почему DVD-диск вмещает больше информации чем CD-диск?
61. Что такое стример?
62. Как можно самостоятельно научиться печатать слепым 10-пальцевым способом?
63. Как набрать букву «ё» на дисплейной клавиатуре планшета или телефона?
64. Чем лазерная мышка отличается от светодиодной?
65. Какой тип сканеров наиболее распространён?
66. Что такое дигитайзер?
67. Какие сенсорные технологии вы знаете, что такое multitouch?
68. Какие основные характеристики имеют современные видеокарты?
69. Опишите принцип работы жидкокристаллического монитора.
70. Какие основные характеристики имеют современные мониторы?
71. Какой принцип формирования изображения использует лазерный принтер?
72. Какой принцип формирования изображения использует струйный принтер?
73. Какой принцип печати у современных 3D-принтеров?
74. Какие ещё принтеры, кроме струйных, лазерных и 3D, вы знаете?
75. На каком принтере можно печатать под копирку?
76. Расшифруйте аббревиатуру СНПЧ.
77. Как работает плоттер?
78. В чём отличие LCD от DLP от проекторов?
79. Какие устройства для работы со звуком используются на ПК? Какие звуковые схемы используются?
80. Какое оборудование используется для работы локальных сетей по технологии Ethernet?
81. На каких скоростях позволяют работать Wi-Fi сети?
82. Что такое Wi-Fi-каналы?
83. Что делать, если две Wi-Fi-точки доступа, находящиеся на небольшом удалении, мешают друг другу передавать данные?

84. Как устроен сетевой фильтр?
85. Как работает сетевой стабилизатор?
86. Что такое ИБП (UPS)?
87. Какие типы ИБП вы знаете?
88. Возможно ли использовать полупроводники для защиты электронных схем?
89. Каково внутреннее устройство схемы защиты сетевого оборудования от перенапряжения?

4.18. Литература к главе 4

1. *Хоровиц П., Хилл У.* Искусство схемотехники/ пер. с англ. – Изд. 6-е. -М.: Мир, 2003. – 704 с., ил. ISBN 5-03-003395-5.
2. *Колин К. К.* Становление информатики как фундаментальной науки и комплексной научной проблемы: сб. научн. трудов // Системы и средства информатики. Спец. вып.: Научно-методологические проблемы информатики. / под ред. К. К. Колина. – М.: ИПИ РАН, 2006.
3. *Леонтьев В. П.* Новейшая энциклопедия компьютера 2011. – М.: ОЛМА Медиа Групп, 2010. – 960 с.: ил. ISBN 978-5-373-03920-8.
4. *Мюллер С.* Модернизация и ремонт ПК / пер. с англ. – 19-е изд. – М.: Вильямс, 2011. – 1072 с.: ил. ISBN: 978-5-8459-1668-6.
5. http://ru.wikipedia.org/wiki/Компьютер.
6. *Пильщиков В. Н., Абрамов В. Г., Вылиток А. А., Горячая И. В.* Машина Тьюринга и алгоритмы Маркова. Решение задач: учебно-метод. пособие – М.: МГУ, 2006. – 47 с.
7. *Старков В.* Архитектура персонального компьютера: организация, устройство, работа. – М.: Горячая Линия – Телеком, 2009. – 536 с.
8. *Борзенко А. Е.* IBM PC: устройство, ремонт, модернизация. – 2-е изд., перераб и доп. – М.: ТОО фирма «КомпьютерПресс», 1996. – 344 с.: ил. ISBN 5-89959-019-X.
9. Как рождается процессор // Computer Bild. – 2013. – № 9(188). – С. 20–27.
10. *Грошев А. С.* Информатика: учеб. для вузов / А. С. Грошев. – Архангельск: Арханг. гос. техн. ун-т, 2010. – 470 с. ISBN 978-5-261-00480-6.
11. *Смирнова Е. В.* Технологии современных сетей Ethernet. Методы коммутации и управления потоками данных.: учеб. пособие. /Е. В. Смирнова, П. В. Козик [под ред. Б. В. Кострова]. – СПб.: БХВ-Петербург, 2012. – 272 с.: ил. ISBN 978-5-9775-0831-5.
12. *Перельман Я. И.* Занимательная физика / по ред. А. Б. Млодзеевского. – 5-е изд. Кн. 1. – М., Л.: Гос.изд.технико-теоритич. литературы, 1949.
13. *Савельев И. В.* Курс общей физики: учеб. пос. для втузов: в 5 кн. -- Кн. 2: Электричество и магнетизм. – 4-е изд., перераб. – М.: Наука, Физматлит, 1998. – 336 с. – ISBN 5-02-015001-0.
14. *Кабардин О. Ф.* Физика: справ. материалы: учеб. пос. для учащихся. – 3-е изд.

– М.: Просвящение, 1991. – 367 с.: ил. – ISBN 5-09-003008-1.
15. *Макдона Р.* Основы микрокомпьютерных вычислений / пер. с англ. Т. Г. Никольского; под ред. В. Ф. Шаньгина. – М.: Высш. школа, 1989. – 272 с.: ил. ISBN 5-06-000318-3 (СССР).
16. *Борисов В. Г.* Юный радиолюбитель. – 7-е изд., перераб. и доп. – М.: Радио и связь, 1985. – 440 с., ил. (Массовая радиобиблиотека, Вып. 1101.)
17. Сворень Р. Электроника шаг за шагом. – М.: Детская литература, 1991. ISBN 5-08-001436-9.
18. *Ким А. К., Перекатов В. И., Ермаков С. Г.* Микропроцессоры и вычислительные комплексы семейства «Эльбрус». – СПб.:Питер, 2013. – 272 с.: ил. ISBN 978-5-459-01697-0.
19. Краткое описание архитектуры Эльбрус // http://elbrus.ru/arhitektura_elbrus.
20. *Roser M., Ritchie H.* Technological Progress, 2018, http://ourworldindata.org/technological-progress.
21. Intel ME. Как избежать восстания машин? 27 апреля 2016 http://habrahabr.ru/company/dsec/blog/282546/.
22. Безопасность прошивок на примере подсистемы Intel Management Engine, 4 марта 2016, http://habrahabr.ru/company/dsec/blog/278549/.
23. Intel Management Engine http://en.wikipedia.org/wiki/Intel_Management_Engine
24. *Галагузова М. А., Комский Д. М.* Первые шаги в электротехнику: Кн. для учащихся 4-7 кл. сред. шк. – 2-е изд., перераб. и доп. – М.:Просвещение, 1988. – 143 с.: ил. ISBN 5-09-000719-5.
25. *Акофф Р.* Искусство решения проблем: Пер. с англ. – М.:Мир, 1982. – 224 с.
26. *Романец Ю.В., Тимофеев П.А., Шаньгин В.Ф.* Защита информации в компьютерных системах и сетях /Под ред. В.Ф.Шаньгина. – 2-е изд., перераб. и доп. – М.:Радио и связь, 2001. – 376 с., ISBN 5-256-01518-4.
27. *Schneier B.* Applied Cryptography. – John Wiley & Sons. Inc., 1996. – 758 p.

Глава 5. Программное обеспечение

5.1. Введение

К данному моменту у читателей должно было сформироваться представление о том:

- что есть информация в окружающем нас мире, как её можно охарактеризовать и, возможно, измерить, если дело касается компьютера, как эта информация может быть представлена (закодирована) последовательностями нулей и единиц;
- какие юридические нормы, социальные и моральные ограничения налагает общество в отношении информации;
- а также как числовые, тестовые, аналоговые и визуальные данные (точнее, их цифровое представление) можно обрабатывать с помощью электрических схем, состоящих из деталей – «аппаратной части» ПК, в быту именуемой «железом» или компьютерными комплектующими.

Вопрос, которым должны были задаться читатели по завершении прочтения предыдущих глав: как заставить всё это полезное оборудование работать вместе максимально эффективно?

Парадокс сегодняшней ситуации заключается в том, что задай мы читателям цель, например снять видео о любом интересующем нас событии, предположительно мы получим много разнообразных ответов-решений, как это можно сделать... Например, можно взять телефон с видеокамерой, на месте телефона может быть планшет, фотоаппарат, MP3-плеер и вообще любой другой электронный прибор, разве что банковские карточки ещё не научились снимать видео и записывать звук. Ответы себя не заставляют долго ждать потому, что большинство читателей «варится» в мире компьютеров. Однако не все выберут для решения поставленной цели взять видеокамеру с видеовыходом либо веб-камеру, систему ввода видео или тот же TV-tuner, корпус, материнскую плату, процессор, оперативную память, жёсткий диск и привод для записи CD/DVD/BD-дисков, клавиатуру, мышку, монитор и после этого скажут, что они собрали видеокамеру. Ещё меньшее число придумает взять IP-камеру, патчкорд и хранилище данных (DAS, NAS, SAN или др.).

Предполагаем, что в ответ на последние решения от современной молодёжи, если читатели вообще поняли, что было спрошено и о чём шла речь, можно будет услышать «ну, вы и извращенцы».

Ещё меньшее число предложит взять видеокамеру с кассетами или плёнкой.

Что поделать, «общество потребления» отучает людей думать и лишь предлагает готовые решения, удобные рынку и производителям.

Цель данной главы – попытаться разобраться, как указанные решения строятся изнутри, с технической точки зрения.

Практически во всех решениях выше мы описали аппаратную часть решения, забыв о программной. Нам кажется естественным, что нажми мы на кнопочку «запись» пальцем или ткни по изображению кнопки на экране, и «оно» будет записывать.

Но как это работает?

В вопросе построения процессоров (по архитектуре Джона фон Неймана) как средства обработки информации частично мы коснулись ответа, а именно рассмотрели, что есть инструкция, а далее логично предположить, что напиши мы несколько инструкций подряд, если не сказать «много», – будет готова программа. Собственно, первые компьютеры и современная простая электроника так работали и работают.

По мере роста числа задач, поставленных для решения компьютеру, он из устройства для подсчёта (большого калькулятора) превратился в что-то многофункциональное, при этом неизбежно множество операций (подзадач) начало дублироваться и повторяться. То есть программный код, отвечающий за эти функции, стал неоптимальным.

Параллельно с этим вычислительная мощность росла, и компьютеры всё чаще оказывались незагруженными.

Вот тут на пути оптимизации и было предложено программистами на одном компьютере одновременно решать несколько задач с помощью специально написанного программного обеспечения, сначала в режиме последовательного разделения времени, а после и одновременно разные задачи.

5.2. Классификация программного обеспечения

Всё программное обеспечение (ПО, software) по назначению и использованию можно разделить на 2 большие группы – системное и прикладное.

Системное ПО – совокупность программ для обеспечения работы компьютера.

Прикладное ПО – комплекс программ для решения задач определённого класса конкретной предметной области.

В то же время в состав основных системных средств – операционных систем (ОС) – входят и компоненты прикладного ПО, например простейшие текстовые редакторы, программы для работы с мультимедиа, с интернетом и др. В составе прикладного ПО можно выделить особую группу – средства разработки ПО, назначение которых – разработка новых программ, в том числе и операционных систем. Так что эти две большие группы имеют некоторое пересечение по своим компонентам. Многие другие группы ПО также имеют общие компоненты, например система автоматизации управления предприятиями работает с системой управления базами данных (СУБД), имеет средства разработки, может обмениваться информацией с САПР и т. п. На рис. 5.1 показана классификация ПО, которая учитывает наличие общих компонентов в различных видах ПО и взаимодействие различных видов ПО.

Без системного ПО не может работать ни один компьютер. Наиболее распространёнными на сегодня операционными системами для настольных ПК являются коммерческая **Microsoft Windows** и **свободное ПО ОС Linux**. Первая предпочтительно продвигается за счёт маркетинговых усилий и затрат на рекламу, вторая держится на энтузиазме пользователей и за счёт корпоративных решений для больших корпораций. Назвать Red Hat Linux или Oracle Linux, используемые в промышленном масштабе, энтузиастскими проектами как-то язык не поворачивается.

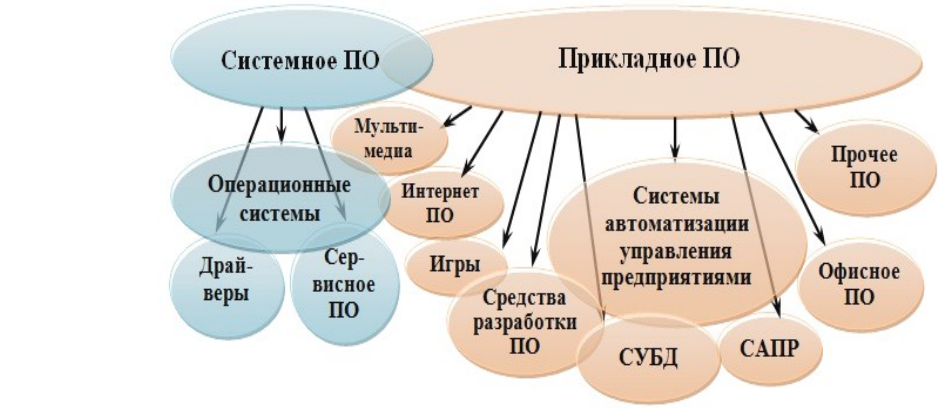


Рисунок. 5.1. Классификация программного обеспечения

Большинство пользователей не активно в выборе и в основном берёт то, что им предлагают продавцы (например, ненавязчиво-предустановленная ОС при покупке ноутбуков и системных блоков), поэтому более половины всех ПК всё ещё используют ОС Windows. В какой-то мере оно и проще – плыть по течению. Также и лозунги коммерсантов по борьбе с пиратством нацелены не на использование свободного ПО, а опять же призывают покупать коммерческое и зачастую ненужное ПО.

С выходом 17 декабря 2010 г. Распоряжения Правительства Российской Федерации № 2299-р [235] ситуация в нашей стране стала понемногу меняться. А именно в государственных и образовательных учреждениях начался переход на свободное ПО. Отчасти, возможно, вы и поэтому держите данную книгу у себя в руках как ответ авторов на запросы рынка.

Драйверы (программы, или компоненты ядра ОС, обеспечивающие работу ОС со всеми компонентам ПК) могут присутствовать в составе ОС, но для самых новых материнских плат и внешних устройств, не совместимых со старыми, поставляются отдельно. Касаемо ОС Linux, понятие драйвера в привычном смысле там отсутствует, поддержка нового железа сначала появляется в базовой ветке ядер на www.kernel.org, а оттуда перекочёвывает во все существующие дистрибутивы. Производители или отдают информацию разработчикам и забывают о проблемах совместимости, или регулярно выкладывают у себя на сайте готовые модули ядра, если не хотят делиться программным кодом.

В **сервисное ПО** входят программы проверки дисков и файловой системы, антивирусные программы, архиваторы, программы записи оптических дисков, программы обслуживания компьютерных сетей и прочее.

Среди **прикладного ПО** особое место по сложности разработки, внедрения и эксплуатации, по количеству компонентов системы, по объёму обрабатываемой информации занимают системы комплексной автоматизации управления предприятиями, на верхнем уровне которых – ERP-системы (ERP – Enterprise Resource Planning). В составе ERP-системы – несколько десятков подсистем, работающих с информацией корпорации (Планирование, Производство, Снабжение, Склады и сбыт, Финансы и прочее), вся информация хранится на сервере баз данных MySQL, Oracle, PostgreSQL или

[235] О плане перехода федеральных органов исполнительной власти и федеральных бюджетных учреждений на использование свободного программного обеспечения (2011–2015 годы).

Microsoft SQL Server, в систему входят также средства разработки модулей для работы с базой данных, средства для взаимодействия с системами автоматизации проектирования (САПР) и другими системами.

Среди основной массы пользователей ПК обязательным компонентом ПО, установленного на компьютере, являются **офисные системы**, в состав которых входят текстовый редактор, табличный процессор, СУБД, органайзер и средства для работы с электронной почтой, средства разработки презентаций и прочее. В нашей стране это обычно или свободное ПО LibreOffice, или его предшественник Apache OpenOffice, или коммерческий продукт Microsoft Office.

Прочее ПО в схеме рис. 5.1 – специальные математические программы, программы-переводчики, программы для распознавания текста и голоса, обучающие программы, экспертные, геоинформационные системы, программы для майнинга и многие другие.

Помимо классификации ПО по назначению, существует ещё одна классификация – рыночная или по типу лицензии на использование.

Лицензионные, или ***коммерческие, программы*** (*commercial ware*) распространяются на коммерческой основе. Также их ещё называют ***проприетарным ПО***. Для получения права на использование подобных программ необходимо купить экземпляр дистрибутива такой программы. Порядок использования программы оговаривается в лицензионном соглашении, подлинность ПО подтверждается сертификатом (**Certificate of Authenticity**), имеющим уникальный код каждой копии дистрибутива (product key). Свободное копирование и использование купленных коммерческих программ на числе компьютеров, превышающих число, указанное в лицензии, запрещается.

Свободное ПО, GPL, GNU Public License, Стандартная общественная лицензия GNU. GNU (рекурсивный акроним от GNU's Not UNIX – «GNU – не UNIX») – свободная Unix-подобная операционная система, а в дальнейшем ПО, не входящее в её состав, разрабатываемая Проектом GNU, который начался 27 сентября 1983 года, когда Ричард Столлман (Richard Stallman) опубликовал объявление о проекте в группах новостей net.unix-wizards и net.usoft.

Ричард Столлман – автор концепции «копилефта» – «🄯» (транслитерация copyleft), выражающей идею движения за свободное ПО; смысл которого воплощен в лицензии Открытое лицензионное соглашение GNU (GNU General Public License, GNU GPL) для ПО.

Текст лицензии доступен по адресу: http://www.gnu.org/copyleft/gpl.html.

GPL предоставляет получателям компьютерных программ следующие права, или «свободы»:

- свободу запуска программы с любой целью;
- свободу изучения того, как программа работает, и её модификации (предваритель-

ным условием для этого является доступ к исходному коду);
- свободу распространения копий как исходного, так и исполняемого кода;
- свободу улучшения программы и выпуска улучшений в публичный доступ (предварительным условием для этого является доступ к исходному коду).

Ядро Linux, используемое в операционной системе GNU/Linux, распространяется по лицензии GPL.

Бесплатные программы (*freeware*) – распространяются авторами бескорыстно. Чаще всего это небольшие программы, написанные одним или несколькими программистами для себя и предложенные ими для общего пользования. Такие программы можно найти в интернете. В общем случае бесплатные программы не являются свободными. Бесплатно распространять можно и несвободное ПО – программы с закрытым кодом.

Условно-бесплатные программы (*shareware*) занимают промежуточное положение между коммерческими и бесплатными программами. Самая массовая группа программ, в которую входят практически все утилиты для ОС Windows, а часто и весьма серьёзные, нужные многим программные пакеты. Эти программы предоставляются бесплатно, однако по истечении определённого срока необходимо заплатить их автору или распространителю небольшую сумму. В противном случае программа либо не загружается, либо начинает надоедать просьбами о её регистрации. «Попробуй и купи» *(try and buy)* – это более жёсткий вариант условно-бесплатного ПО, так как он не оставляет выбора пользователю, по истечении срока тестирования программу следует или купить, или удалить с компьютера, в противном случае вы будете нарушать лицензионное соглашение.

Пробные версии программ (*trialware*) – это полноценные версии коммерческих систем, которые можно использовать какое-то время бесплатно. По истечении этого времени программы прекращают работать.

Демоверсии (*demoware*) – демонстрационные версии коммерческих программ и игр с ограниченными возможностями. Например, с выключенной функцией сохранения документов в текстовом редакторе, с ограниченным количеством записей в базе данных информационной системы или с несколькими турами в играх.

5.3. Операционная система

Операционная система – комплекс программ, обеспечивающих работу компьютера.

Изначально появилась концепция некоторой программы менеджера, впоследствии оказавшейся операционной системой (ОС), – программы для решения следующих задач:
- разделения ресурсов компьютера между несколькими программами (пользователями);
- обеспечения «эффекта» одновременного выполнения нескольких программ (задач);
- оптимизации общего кода всех запускаемых программ (приложений) за счёт вынесения из их состава повторяющихся частей внутрь операционной системы;
- предоставления некоторого унифицированного интерфейса (оконного, консольного и прочего).

Впоследствии для современных ОС этот список был несколько расширен и уточнён за счёт того, что разные программы «захотели» взаимодействовать между собой.

Давайте разберёмся более подробно, из чего состоит ОС, изучим терминологию, познакомимся с большинством стандартных функций и программ, входящих в состав современных ОС, а после плавно перейдём к часто используемому офисному пакету LibreOffice (текстовому процессору Writer и электронным таблицам Calc).

Функции современной ОС для ПК:

1. Управление конфигурацией компьютера.
2. Управление процессами, потоками и заданиями.
3. Управление памятью.
4. Обеспечение информационной безопасности.
5. Управление подсистемой ввода-вывода.
6. Управление внешней памятью.
7. Управление файловой системой.
8. Поддержка сетей.

В управлении конфигурацией компьютера (в ОС Windows) ключевую роль играют данные, сохраняемые в реестре этой системы (база данных общесистемных и пользовательских параметров) и инструментарий управления Windows (WMI). В ОС Linux такого понятия, как реестр, нет, вместо него используются понятные текстовые конфигурационные файлы, располагающиеся в директории /etc.

Процесс в ОС – исполняемая программа и её данные в закрытом адресном пространстве, данные защиты и прочие объекты, принадлежащие процессу.

Поток – некоторый компонент внутри процесса, который получает процессорное время для выполнения.

Задание (job) – совокупность процессов, управляемых как единая группа. В ОС Linux как такового понятия «задание» нет. Для отложенного запуска программ используются демоны atd и crond.

ОС управляет виртуальной памятью, которую проецирует на физическую память компьютера.

Информационная безопасность предусматривает защиту любых разделяемых системных объектов (файлов, каталогов, процессов, потоков и т. д.), аутентификацию пользователей при входе в систему, предотвращение несанкционированного доступа, аудит событий в системе (ведение журналов событий).

Диспетчер ввода-вывода ОС подключает приложения и системные компоненты к виртуальным, логическим и физическим устройствам, а также определяет инфраструктуру, поддерживающую драйверы устройств. (В ОС Linux реализуется на уровне ядра ОС.)

Управление внешней памятью включает организацию работы с жёсткими и гибкими дисками, с оптическими дисками, магнитными лентами, флэш-памятью и другими устройствами. (В ОС Linux реализуется на уровне ядра ОС.)

Управление файловой системой, определяющей методы хранения информации на дисках, – важнейшая функция любой ОС. Многие ОС поддерживают работу с несколькими файловыми системами, например **Windows XP, 7** и более новые версии могут управлять файловыми системами **NTFS** (основная для магнитных дисков), **FAT12,**

FAT16, FAT32, ISO9660, CDFS и **UDF** (для лазерных дисков). При установке дополнительного драйвера [236] поддерживаются тома файловых систем **ext2** и **ext3**. Список файловых систем, поддерживаемых ядром ОС Linux, намного шире, к нему можно добавить такие ФС, как **UFS, XFS, HFS, ZFS, ReiserFS** и др.

Основные функции работы с информацией файловой системы – **CreateFile** (создать файл), **ReadFile** (читать файл), **WriteFile** (записать в файл), **ExecuteFile** (запустить файл).

Поддержка сетей присутствует во всех современных ОС.

Ядро́ – центральная часть ОС, обеспечивающая приложениям (запущенным программам) координированный доступ к ресурсам компьютера, таким как процессорное время, память, внешнее аппаратное обеспечение, внешние устройства ввода и вывода информации. Также обычно на уровне ядра реализуются сетевые сервисы (поддержка того или иного стека протоколов) и работа с файловыми системами. ОС, где ядро особо не выделено называют безъядерными. Более подробное рассмотрение структуры ядра ОС, а также их классификации, к сожалению, выходит за рамки данной книги.

5.3.1. Краткая история развития операционных систем для ПК

Наиболее полную историю развития операционных систем в виде двух больших графов можно найти на сайте Éric Lévénez-a (http://levenez.com/).

Первый граф http://levenez.com/windows/ на 26 страницах А4 изображает историю развития ОС семейства Windows начиная с 1980 года и по наши дни.

Второй граф http://levenez.com/unix/ на 30 страницах А4 изображает историю развития ОС семейства UNIX (включая Andoid, iOS и Linux) начиная с 1969 года и по наши дни.

Учитывая большой объём, рекомендуем читателям самостоятельно посетить указанные выше ссылки и ознакомиться с наглядными материалами.

Более развёрнуто развитие двух популярных ОС для ПК можно описать примерно так:

5.3.1.1. История Windows (зарубежная закрытая ОС)

Первый персональный компьютер PC появился в конце 1981 года (IBM 5150 PC: 4.77-MHz Intel 8088 CPU, 64KB RAM, 40KB ROM, 5.25-дюймовый флоппи-дисковод) с операционной системой PC-DOS[237] 1.0. Далее на рынке появились так называемые IBM-совместимые персональные компьютеры. Операционная система этих компьюте-

[236] http://www.chrysocome.net/explore2fs или http://www.ext2fsd.com/.

[237] Personal Computer Disk Operating System – операционная система для персональных компьютеров, загружаемая с дисков и обеспечивающая работу с прикладными программами, располагаемыми также на дисках. Дискеты формально относятся к дискам поскольку внутри них находится гибкий пластиковый диск с магнитным напылением. Аббревиатура DOS часто применялась как официальное название (торговый знак или его часть) многих операционных систем; например DR-DOR, TR-DOS, Альфа-DOS, для ОС ЭВМ серий IBM /360 и /370 и др., в связи с чем стала нарицательной. В какой-то мере DOS – это символ эпохи операционных систем и своя философия. В русскоязычной литературе того времени часто можно было встретить аналогичную русскую аббревиатуру ДОС, а в речи специалистов слышать диалоги вроде: «где? в досе», «чья? досовская» и т.п.

ров называлась MS DOS 1.0 – корпорация Microsoft предоставила в распоряжение фирм, производящих эти машины, точную копию операционной системы PC-DOS.

Всего было выпущено 12 версий MS DOS: 1.0, 1.1, 2.0, 3.0, 3.3, 4.0, 4.01, 5.0, 6.0, 6.2, 6.21, 6.22. Версия 7 входила в состав Windows 95 и отдельно не выпускалась.

При именовании директорий в файловой системе FAT пробелы не допускались в связи с чем появилось написание «MS-DOS» (и аналогичных названий) через дефис, в том числе и в литературе.

Замечание. Поскольку в данном параграфе речь идёт об истории Windows, то больше внимания уделено именно MS DOS, а не другим, не менее интересным, а порой и более эффективным, DOS-системам.

Файловые менеджеры под DOS

Все версии MS DOS были однозадачными ОС с текстовым интерфейсом и управлением из командной строки. Характерной сторонней программой того времени, облегчавшей работу с файловой системой, был файловый менеджер **Norton Commander**, имевший два окна для показа папок (директорий или каталогов, как их тогда называли) и файлов. Позднее появились его аналоги **Volcov Commander**, **DOS Navigator,** а также используемые и по сей день бесплатные **FAR Manager** (уже под Windows) и **midnight commander** (в виде порта под DOS и другие системы, а основная ветка под Linux).

Основой MS DOS был 1 файл – командный процессор **command.com**. Существовали ещё 2 скрытых файла – ibmbio.com и ibmdos.com (в других системах – их аналоги) – модуль расширения BIOS и модуль обработки прерываний, а также достаточно большое количество утилит (служебных программ) и драйверов внешних устройств (программ для работы с клавиатурой, мышкой, принтерами и прочей периферией).

Некоторые команды MS DOS могут использоваться и в настоящее время в командном интерпретаторе Windows (окно программы **cmd.exe**, похожее на сеанс MS DOS, но им не являющееся, см. рис. 5.2), обеспечивающем текстовый интерфейс между пользователем и этой ОС.

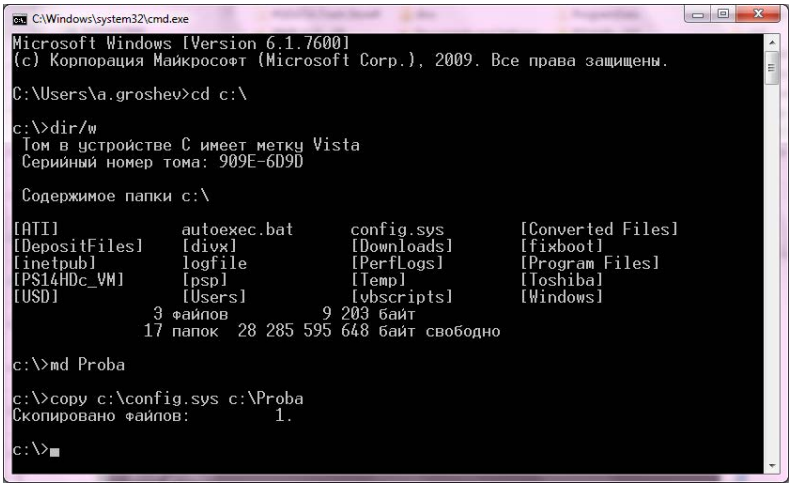


*Рисунок 5.2. Командный интерпретатор **Windows** (cmd.exe)*

Приведём далеко не полный и упрощённый список примеров команд вместе с их ана-

логами для командного интерпретатора BASH.

Команда MS-DOS / cmd.exe	Аналог в BASH	Описание
`dir`	`ls`	просмотреть список файлов текущего каталога
`ver`	`uname -a` `cat /etc/issue`	узнать версию операционной системы
`cd <имя>`	`cd <имя>`	сменить каталог
`md <имя>`	`mkdir <имя>`	создать подкаталог
`D:`	–	сменить рабочий диск
`copy <что> <куда>`	`cp <что> <куда>`	копирование файлов
`time, date`	`date`	настроить время и дату на системных часах

Полный список выводится по команде help и содержит 86 команд.

Для запуска программ MS-DOS в ОС Linux может использоваться программа-эмулятор DOSBox (см. рис. 5.3), впрочем «чистую» MS-DOS в настоящее время можно также установить в любом эмуляторе. Подробнее об эмуляторах см. раздел 5.4. Виртуализация и гипервизоры.

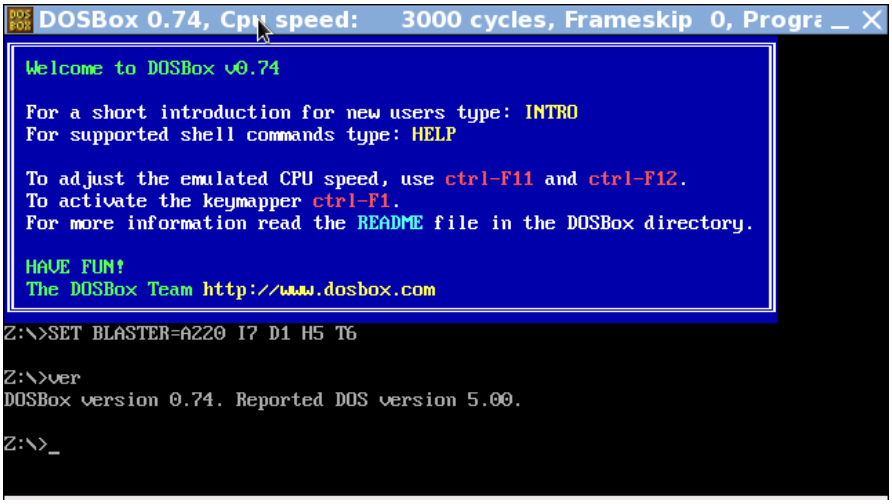


Рисунок. 5.3. Окно эмулятора DOSBox для запуска программ MS-DOS под ОС Linux

Интересный факт: первые версии Windows использовали MS-DOS для своей инсталляции (установки на компьютер) или запускались из MS-DOS. Современные версии Windows не используют для этих целей MS-DOS, но могут запускать некоторые из программ, написанных для MS-DOS в окне или в полноэкранном режиме (см. рис. 5.2). Однако сложные программы для MS-DOS (например, игры), не запускаются в последних версиях Windows. Для этой цели можно использовать свободное ПО – программу DOSBox [238] (Free Software Foundation, Inc), версии которой существуют для многих современных ОС: BeOS, Linux, Mac OS X, OS/2, и той же Windows!

[238] Свободное ПО – эмулятор DOS'а: http://www.dosbox.com/.

История развития ОС Microsoft Windows началась 20 ноября 1985 г. с выходом первой версии Windows 1.0. Несмотря на то, что это была кишащая ошибками и недочётами программа, в которой невозможно было полноценно что-либо сделать, это был настоящий прорыв, луч света в мир чёрных командных строк и белых (а то и зелёных или оранжевых на монохромных мониторах) безликих букв[239].

С появлением новых процессоров для ПК совершенствовалась и ОС Windows, расширялся объём памяти, с которым она могла работать. В 1987 году появились версии Windows 2.0 и 2.1, стали появляться первые программы (первая версия текстового редактора Word for Windows вышла в 1989 г.). В 1990 году – Windows 3.0. В 1994 г. – Windows for Workgroups 3.11 с поддержкой одноранговых сетей.

Чуть раньше, в августе 1993 г., появилась новая ОС – Windows NT 3.1 («Новая технология» – New Technology). Она действительно оправдывала своё название, так как была написана с нуля. Windows NT была первой ОС компании, которая была предназначена для организации выделенного файлового сервера в составе локальной сети и включала поддержку некоторых клиент-серверных бизнес-приложений. ОС имела ряд новых ключевых особенностей: приоритетный многозадачный планировщик для Windows-приложений, интегрированная работа с сетями, разграничение прав пользователей для сервера домена, поддержка ФС OS/2, поддержка многопроцессорной архитектуры, новая более надёжная файловая система NTFS.

Windows NT 3.1 была в двух версиях: для серверов и для рабочих станций (в составе последних отсутствовали серверные службы).

Реально ОС Windows начала широко использоваться в 1995 г. с выходом версии Windows 95, имевшей новый графический интерфейс. Основная масса прикладного программного обеспечения стала выпускаться для этой ОС.

Windows 95 – уже не графическая оболочка пользователя для MS DOS (как это было в предыдущих версиях), а полноценная псевдомногозадачная 32-разрядная операционная система. Хотя пользователи могли видеть окно MS DOS в процессе загрузки, система заменяла собой MS DOS 7.0 после того, как загружалась полностью. Дальнейшее развитие в Windows 98 и Windows Me оказалось тупиковым.

Ветка ОС на ядре NT развивалась параллельно: Windows NT 4.0 – 1996 г., Windows NT 5.0 (она же 2000) – 2000 г. Широкое распространение среди обычных пользователей она получила лишь с выходом версии Windows XP (NT 5.1) в 2001 г. Она имела изменённый графический интерфейс, использовала как основную файловую систему NTFS что дало возможность разграничения прав пользователей на локальном компьютере. Система Windows XP создавалась на основе усовершенствованного кода Windows 2000 (NT 5.0), причём были разработаны различные версии для пользователей домашних компьютеров и бизнес-пользователей: Windows XP Home Edition и Windows XP Professional (NT 5.1).

Далее хронология развития линейки NT была примерно следующей (в скобках указаны года выхода версий на рынок):

Windows XP 64-bit Edition – Windows NT 5.2 (2003)

Windows Server 2003 – Windows NT 5.2 (2003)

Windows XP Professional x64 Edition – Windows NT 5.2 (2005)

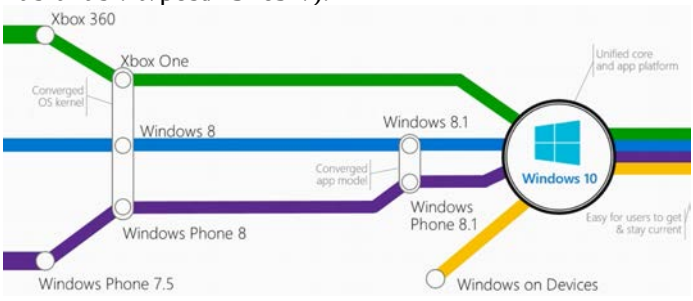
Windows Vista – Windows NT 6.0 (2006)

[239] Любопытства ради, образы виртуальных машин со старыми версиями ОС доступны по адресу http://old-dos.ru//index.php?page=files&mode=files&do=list&cat=43.

Windows Home Server – Windows NT 5.2 (2007)
Windows Server 2008 – Windows NT 6.0 (2008)
Windows Small Business Server – Windows NT 6.0 (2008)
Windows 7 – Windows NT 6.1 (2009)
Windows Server 2008 R2 – Windows NT 6.1 (2009)
Windows Home Server 2011 – Windows NT 6.1 (2011)
Windows 8 – Windows NT 6.2 (2012)
Windows Server 2012 – Windows NT 6.2 (2012)
Windows 8.1 – Windows NT 6.3 (2013)
Windows Server 2012 R2 – Windows NT 6.3 (2013)
Windows 10 – Windows NT 10.0 (2015)

Параллельно с этим, с 1990-х по 2015 г. было выпущено некоторое количество систем мобильных и встраиваемых версий «CE-» и «Mobile-» линий.

Планы компании – с 2015 года объединить ветки вместе и облегчить жизнь разработчикам (http://habrahabr.ru/post/254691/).



5.3.1.2. История Linux

В 1991 году **Линус Торвальдс**, финский студент, чрезвычайно увлёкся идеей написать совместимое с UNIX ядро операционной системы для своего персонального компьютера с процессором ставшей очень широко распространённой архитектуры Intel 80386. Именно поэтому на младших моделях процессоров ядро Linux не работает, зато из этого процессора ему удалось выжать максимум.

Совместимость с UNIX в тот момент означала, что операционная система должна поддерживать стандарт POSIX. POSIX – это функциональная модель совместимой с UNIX операционной системы, в которой описано, как должна вести себя система в той или иной ситуации, но не приводится никаких указаний, как это следует реализовать программными средствами. POSIX описывал те свойства UNIX-совместимых систем, которые были общими для разных реализаций UNIX на момент создания этого стандарта. В частности, в POSIX описаны системные вызовы, которые должна обрабатывать операционная система, совместимая с этим стандартом.

Важнейшую роль в развитии Linux сыграли глобальные компьютерные сети Usenet и Internet. На самых ранних стадиях Линус Торвальдс обсуждал свою работу и возникающие трудности с другими разработчиками в телеконференции comp.os.minix в сети Usenet, посвящённой операционной системе MINIX. Ключевым решением Линуса стала публикация исходных текстов ещё малоработоспособной первой версии ядра под свободной лицензией GNU GPL. Благодаря этому и получавшей всё большее распространение сети интернет очень многие получили возможность самостоятельно компилировать и тестировать это ядро, участвовать в обсуждении и исправлении ошибок, а также присылать исправления и дополнения к исходным текстам Линуса. Теперь над ядром работал уже не один человек, разработка пошла быстрее и эффективнее.

В 1992 году версия ядра Linux достигла 0.95, а в 1994 году вышла версия 1.0, что свидетельствовало о том, что разработчики наконец сочли, что ядро в целом закончено и все ошибки (теоретически) исправлены. В настоящее время разработка ядра Linux – дело уже гораздо большего сообщества, чем во времена до версии 1.0. Изменилась и роль самого Линуса Торвальдса: теперь он не главный разработчик, а наиболее авторитетный член сообщества, по традиции оценивающий качество исходных текстов, которые должны быть включены в ядро, и дающий своё добро на их включение. Тем не менее общая модель свободной разработки сообществом сохраняется.

Развитие дистрибутивов ОС Linux в мире можно описать графом http://commons.wikimedia.org/wiki/File:Linux_Distribution_Timeline.svg.

Если убрать из рассмотрения небольшое число дистрибутивов Linux, где основной способ установки программного обеспечения – это его компилирование из исходных кодов, то оставшуюся основную массу дистрибутивов можно поделить по способу установки программ на две группы. Фактически деление производится исходя из используемой системы пакетов:

- основанные на Debian или использующие формат пакетов Deb;
- на базе RedHat или использующие формат пакетов RPM.

Отдельно хочется выделить дистрибутивы, разрабатываемые российскими компаниями и сообществами:

Операционная система «Альт» (http://www.basealt.ru, ранее Альт Линукс). Технологическая программная платформа «Альт» предназначена для автоматизации деловых процессов организаций и управления ресурсами корпоративной ИТ-инфраструктуры. В составе платформы – семейство отечественных операционных систем «Альт» для серверов и рабочих станций (включая версии, сертифицированные ФСТЭК России, ФСБ России и Министерством обороны Российской Федерации), и большой набор прикладных программ различного назначения. Все операционные системы «Альт» совместимы с российским прикладным ПО различного назначения[240], внесённым в Единый реестр российских программ[241], а также с отечественными и зарубежными аппаратными платформами; служат полноценной альтернативой операционным системам семейства Windows и основанным на ней системообразующим инфраструктурным продуктам.

Разработка и обеспечение жизненного цикла операционных систем семейства «Альт» осуществляется компаниями ООО «Базальт СПО» и АО «ИВК» (Информаци-

[240] *https://www.basealt.ru/fileadmin/user_upload/compatibility/comptab2.html*, *https://www.basealt.ru/products/informacija/compatibility/*.
[241] *https://reestr.minsvyaz.ru/reestr*

онная внедренческая компания). Все работы контролируются из российской юрисдикции. Обеспечивается лицензионная и интеллектуальная защита создаваемых ОС, их производство в соответствии с требованиями отечественных регуляторов. ООО «Базальт СПО» имеет лицензии ФСТЭК на деятельность по разработке и производству средств защиты конфиденциальной информации, а также на деятельность по технической защите конфиденциальной информации (рег. №№ 1612 и 3025 от 16 сентября 2016 года).

В помощь пользователям, осваивающим ОС «Альт Образование» и прикладные программы из дистрибутива, создан специальный портал дистанционной методической поддержки https://kurs.basealt.ru/. На портале размещаются учебно-методические материалы, видеокурсы, работает форум. Бесплатные учебные материалы доступны в библиотеке https://www.altlinux.org/Books:Main_page.

Жизненный цикл операционных систем семейства «Альт» реализуется на основе российского репозитория Sisyphus (Сизиф) – одного из крупнейших в мире технологически независимых репозиториев. Он включает банк программных пакетов и инструменты собственной инфраструктуры разработки ООО «Базальт СПО». В создании ОС «Альт» участвуют более 200 разработчиков (>95% – граждане Российской Федерации).

На базе репозитория «Сизиф» создаются дистрибутивы отечественных ОС «Альт», межсетевой экран «ИВК Кольчуга», сертифицированный по требованиям Министерства обороны России и ФСТЭК России, другие программные продукты фирм-партнёров.

Актуальной на момент издания является девятая платформа ALT – новая стабильная ветка репозитория «Сизиф» (Sisyphus) – предназначенная для разработки, тестирования, распространения, обновления и поддержки комплексных решений всех уровней – от встроенных устройств до серверов предприятий и data-центров.

Девятая платформа ALT содержит хранилища пакетов и инфраструктуру для работы с девятью архитектурами:

- четырьмя основными (параллельная сборка, открытые репозитарии): x86_64, i586, aarch64 (ARMv8), ppc64le (Power8/9);
- двумя дополнительными (догоняющая сборка, открытые репозитарии): mipsel (MIPS 32 бита), armh (ARMv7), riscv64 (RISK-V);
- двумя закрытыми (отдельная сборка, репозитарии по запросу): e2k: e2kv3 (Эльбрус-4С) и e2kv5 (Эльбрус-8СВ); e2kv4 (Эльбрус-8С/1С+).

Сборка для всех архитектур производится нативно, без кросс-компиляции. В единой сборочной среде репозитория «Сизиф» каждый программный пакет собирается под все целевые аппаратные платформы одновременно, что обеспечивает эффективное портирование операционной системы «Альт» и прикладных приложений на новые платформы.

Девятая платформа предоставляет пользователям и разработчикам возможность использования российских систем «Эльбрус», «Таволга», Yadro, «Элвис» и совместимых, широкого спектра оборудования мировых производителей, в том числе мощных серверов ARMv8 Huawei и разнообразных одноплатных систем ARMv7 и ARMv8.

На январь 2021г. платформа протестирована на процессоре Байкал-М, производится установка порядка 14000 комплектов для поставки в РЖД.

Также на сегодняшний день потребителям продлена поддержка дистрибутивов операционных систем «Альт» из 8-й платформы:

- Альт 8 СП – (со встроенными программными средствами защиты информации). Сертифицирован для разных платформ (в том числе e2kv3, e2kv4 !)[242] ФСТЭК России по новым требованиям и Министерством обороны Российской Федерации
- Альт Сервер
- Альт Сервер Виртуализации
- Альт Рабочая станция
- Альт Образование

Russian Fedora (http://ru.fedoracommunity.org, ~~ранее http://rfremix.ru, бывшее название проекта Tedora.~~) – проект, направленный на улучшенную поддержку русскоязычных пользователей Fedora. Помимо всего прочего, включает в себя кодеки и драйверы «из коробки», которых нет в оригинальном дистрибутиве из-за патентных ограничений США.

ОС GosLinux (ОС «Гослинукс») фирмы ООО «Корпорация «Ред Софт» (http://goslinux.ru, http://goslinux.fssp.gov.ru, http://www.red-soft.biz/ru/node/293, ~~ранее http://goslinux.fssprus.ru~~). 24 марта 2014 года на дистрибутив был получен сертификат ФСТЭК России (со сроком действия после продления до 24 марта 2020 г.[243]), последние несколько лет ОС применялась в отделах судебных приставов. За основу взят дистрибутив CentOS версии 6.4.

Astra Linux (http://astra-linux.com) – серия дистрибутивов для платформы x86_64, выпускаемых компанией ОАО «НПО РусБИТех», некоторые из которых имеют сертификаты соответствия от Минобороны России, ФСТЭК России, ФСБ России. Создан на базе Debian.

«СинтезМ» (http://sintezos.ru) – Российская цифровая платформа нового поколения, результат развития (модернизации) ЗОС «Синтез» (заключение в/ч 43753 № 149/3/4/1–1534, 2015 г., Акционерное общество «Финансы, Информация, Технология», АО «Финтех», http://fintech.ru). Представляется собой комплекс программ, включающий отечественную сертифицированную операционную систему нового поколения со встроенной виртуализацией и защитой информации в базовой комплектации. Для выполнения всевозможных задач существует более десятка специализированных модификаций и компонентов на все случаи жизни: «СинтезМ-Сервер», «СинтезМ-Клиент», «СинтезМ-Сервер безопасности», «СинтезМ-Агент безопасности», «СинтезМ-Идентификатор», «СинтезМ-База данных SQL», «СинтезМ-База данных XL», «СинтезМ-Сервер приложений J», «СинтезМ-Офис», «СинтезМ-Контроль. С», «СинтезМ-Хранилище данных. 1», «СинтезМ-Резервное копирование ВМ», «СинтезМ-Почта. С», «СинтезМ- Видеоконференция. С». И это ещё не всё... В основе разработки частично использовались исходные коды версий Linux RedHat 6 и 7.

[242] Важность сертификации для отечественных платформ объясняется тем, что в соответствии с Проектом Указа Президента РФ «О мерах по обеспечению информационной безопасности в экономической сфере при использовании программного обеспечения и оборудования на объектах критической информационной инфраструктуры (КИИ)» постановлено для объектов КИИ до 1 января 2022 г. осуществить переход на преимущественное использование российского оборудования. Увы, другие дистрибутивы ОС Linux не могут обеспечить поддержку отечественных процессоров, а значит и не смогут использовать российское оборудование. Сертификация под наши процессоры – это не будущая отвёрточная сборка из завезённых комплектующих и перекомпиляция программ, а полностью отечественное решение! Существующие системы хранения данных от компаний bitblaze (*https://bitblaze.ru/*) и Аэродиск (ООО «Аеро Диск», ООО НИЦ «АЭРОДИСК», *https://aerodisk.ru/*) уже работают с процессорами Эльбрус на ОС Альт.

[243] На момент подготовки учебника в конце 2020 года информация на сайте о сертификате не обновлена.

Янукс – свободный универсальный дистрибутив Linux производства ФГУП "НИИ НПО "ЛУЧ" (http://yanux.ru) для процессоров x86/x86-64 для использования в информационных системах с повышенными требованиями к безопасности обрабатываемых данных. Текущая версия ЯНУКС 4.0 сертифицирована ФСТЭК России по безопасности информации на классы СВТ 5 и НДВ 4. Сертификат № 2973 от 25 сентября 2013 года (был продлён до 25 сентября 2019 года[244]). Построение идентично CentOS или Scientific Linux.

Прекратившие существование проекты: ASP Linux – с 2008 года, Linux XP – с 2010 года, возможно этот список пополнится, поскольку адаптация зарубежных репозиториев и выпуск на базе них отечественных ОС (адаптированных клонов) – ненадёжный путь развития в долгой перспективе. Отечественный репозитарий «Сизиф», в декабре 2020 года отметивший своё 20-летие, яркий пример полного отрыва от иностранных компаний и сообществ – вот причина и рецепт долголетия!

Типичный вопрос: Какую версию ОС Linux установить?

Типичный ответ: Ту, которая стоит у вашего друга.

Ответ авторов: Если Вы – патриот, то ставьте ОС «Альт 8 СП», надёжный дистрибутив включённый в Единый реестр российских программ и регулярно подстраиваемый разработчиками под нужны и требования российских реалий. Если же вы больше ориентируетесь на зарубежный опыт, – CentOS 7, дистрибутив с долгой поддержкой[245], обладающий возможностью выбора нескольких видов графического оконного менеджера, в том числе классического и дружественного MATE (на базе Gnome 2.x). В случае необходимости, Вы всегда сможете отказаться от бесплатной CentOS и безболезненно перейти на RedHat 7 с коммерческой поддержкой. К системе можно подключить дополнительные репозитарии, самый крупный и известный из них – EPEL (Extra Packages for Enterprise Linux), получив при этом возможность использования нескольких десятков гигабайт полезных и свободных программ.

Если же Вы не хотите ни от кого зависеть, один раз поставить систему, выбрать пакеты и минимально перенастраивать её ближайшее десятилетие – выбирайте Debian.

Совет геймерам и любителям экспериментировать – ставьте всё подряд, и вы найдёте своё счастье сами.

5.3.1.3. Операционные системы для мобильных устройств

С точки зрения ПО, операционные системы для мобильных устройств типа «умных» телефонов и планшетов ничем не отличаются по функциональному предназначению от других ОС, хотя, несомненно, у них есть своя специфика. Собственно, и названия у них немного другие. По данным IDC, в IV квартале 2012 года с долей более 70% на мировом рынке доминировали аппараты с ОС Android, на iOS-устройства приходился лишь 21%, а оставшиеся 9% поделили между собой Blackberry, Windows Phone и др. [28]. По мнению Андрея Борзенко [28], сенсацию готовит та группа ОС, которая скрывается за словом «другие».

Знаковым событием Всемирного мобильного конгресса (MWC 2013), проходившего в Барселоне с 25 по 28 февраля, стало известие о выпуске компанией Mozilla, из-

[244] Информация о продлении сертификата или выпуске новых версий в конце декабря 2020 года на официальном сайте отсутствовала.

[245] До 30 июня 2024 года (http://wiki.centos.org/Download.)

вестной в первую очередь своим браузером Firefox, мобильной операционной системы Firefox OS с открытым исходным кодом [28]. В отличие от главных конкурентов, например iOS и Android, она построена на технологии HTML 5, что позволяет, как утверждают в Mozilla, смартфонам работать очень быстро. Есть у неё ещё одна революционная особенность – Firefox OS настолько упрощает процесс создания приложений, что каждый пользователь, знакомый с HTML 5, сможет создавать свои вариации. В Барселоне Mozilla подписала контракт сразу с 18 операторами связи, принявшими решение о выпуске брендированных смартфонов на базе FirefoxOS, включая Deutsche Telekom, Telefonika, а также «ВымпелКом» и «МегаФон» [28].

Существуют и более экзотические ОС на рынке мобильников, основанные в основном на Linux: Sailfish OS, Tizen, Ubuntu Touch и др. Мобильная версия последней ОС была анонсирована фирмой Canonical лишь в начале года. Отличительные особенности данной ОС: она будет одинаковой для всех устройств – смартфонов, ПК и планшетов; пользователям будет доступен «весь парк наработанных приложений» от Canonical. В фирме рассчитывают, что новая система придётся «по вкусу» не только поклонникам Linux, но и более широкой публике. Сейчас уже Ubuntu Touch можно установить на «железе» для Android – Samsung Galaxy Nexus и LG Nexus 4 [28], либо купить готовый телефон – Bq Aquaris E4.5 Ubuntu Edition.

Интересные факты, или **«Сыр в мышеловке»**

Сама по себе операционная система Android действительно бесплатна. Но весь фокус заключается в том, что за использование магазина приложений и других сервисов «корпорации добра» надо платить. Этого можно избежать, если, например, вместо Google Play воспользоваться альтернативными каталогами. Так, для россиян уже доступны Яндекс.Store и GetUpps! (совместный проект «МегаФона» и «Яндекса»). Правда, ассортимент этих магазинов на порядок беднее, чем Google Play [28].

Но это не всё. С каждой проданной копии операционной системы производителю смартфонов приходится платить ещё и в кассу Microsoft. Дело в том, что Android включает в себя компоненты и технологии, которые являются объектом лицензионных отчислений. По оценкам аналитиков Trefis, с этого набегает порядка 300–400 рублей. Кстати, только за счёт роялти в 2012 году Microsoft ухитрилась заработать на Android больше, чем на собственной Windows Phone [28].

Несмотря на бесплатность и открытый исходный код системы, разработка технологий ядра Android полностью контролируется Google. Этот факт недавно стал неприятным открытием для китайских товарищей. В Поднебесной ведь популярность Android-гаджетов необычайно велика – в III квартале 2012 года она занимала 90% рынка смартфонов. И вот оказалось, что глубокая зависимость от операционной системы Android и работающих на ней устройств – одна из главных угроз конкурентоспособности китайских компаний. Об этом сообщило министерство промышленности и информационных технологий КНР в специальном докладе. Как сообщает ресурс Digitimes, китайские производители стремятся уменьшить свою зависимость от «корпорации добра», пытаясь переключиться на другие мобильные платформы. Естественным образом взгляды разработчиков устремляются в сторону открытых ОС [28].

5.3.2. Процесс

Выполняющуюся программу будем называть ***процессом***.

Каждый процесс в системе уникален и имеет свой идентификатор – идентификатор процесса, или pid (от process id).

Наличие идентификатора позволяет не только осуществлять целенаправленное межпроцессное взаимодействие, но и отличать две копии одной и той же программы, например дважды запущенный текстовый редактор [246]. Из наличия идентификатора и точки зрения общей классификации операционных систем их можно поделить на однозадачные и многозадачные. Сегодня большинство пользовательских ОС являются многозадачными.

Часть операционной системы, отвечающая за непосредственное взаимодействие пользователя с процессами, обычно называется диспетчером задач.

Так, в ОС Windows он доступен пользователям по нажатии клавиш Control+Alt+Delete.

В UNIX-системах, MacOS X и Linux для просмотра списка выполняемых процессов используется команда ps, например

```
$ ps
  PID TTY          TIME CMD
 5672 pts/0    00:00:00 bash
 5700 pts/0    00:00:00 ps
```

Наиболее часто эту команду используют с ключами «aux» как «ps aux», что позволяет увидеть все процессы в системе.

Под хранение номера процесса обычно отводится 2 байта, по крайней мере большинство пользовательских дистрибутивов скомпилированы именно так.

Процессы, кроме самого первого – init, имеют в качестве родителя другой процесс – их запустивший – и носят об этом информацию, подобно отчеству, в переменной ppid (от parent pid), также доступной для просмотра через команду ps.

Команда ps позволяет пользователю с помощью ключей (параметров командной строки) самому варьировать отображаемую информацию по процессам, например вот так:

```
$ ps axo stat,euid,ruid,tty,tpgid,sess,pgrp,ppid,pid,pcpu,comm
STAT  EUID  RUID TT       TPGID  SESS  PGRP  PPID   PID %CPU COMMAND
Ss       0     0 ?           -1     1     1     0     1  0.0 init
S        0     0 ?           -1     0     0     0     2  0.0 kthreadd
S        0     0 ?           -1     0     0     2     3  0.0 ksoftirqd/0
S        0     0 ?           -1     0     0     2     4  0.0 migration/0
S        0     0 ?           -1     0     0     2     5  0.0 watchdog/0
...
Sl     500   500 ?           -1  2023  2023     1  5600  0.0 gedit
Rl     500   500 ?           -1  2023  2023     1  5668  0.2 gnome-terminal
Ss     500   500 pts/0     5838  5672  5672  5668  5672  0.0 bash
S        0     0 ?           -1     0     0     2  5774  0.0 flush-8:16
R+     500   500 pts/0     5838  5672  5838  5672  5838  0.0 ps
```

Подробнее о параметрах см. из консоли «`man ps`» (выход клавишей q).

[246] Следует отметить, что не все программы позволяют запускать себя в двух экземплярах, это зависит от того, как программа была задумана и написана при разработке. Второй экземпляр может завершиться и передать фокус управления первому. Также может быть создано «второе окно» в рамках первой выполняющейся программы.

Так как процессы при выполнении могут по-разному загружать систему (кто-то больше, кто-то меньше), имеется возможность просмотра и этой информации. Так, команда «top» в реальном времени показывает загрузку системы.

Разовый срез из несколько максимально загружающих систему приложений можно увидеть по команде

```
$ top -n1
top - 02:31:37 up  3:23,  2 users,  load average: 0.12, 0.10, 0.13
Tasks: 231 total,   1 running, 230 sleeping,   0 stopped,   0 zombie
Cpu(s): 1.6%us, 0.4%sy, 0.0%ni, 97.7%id, 0.2%wa, 0.0%hi, 0.0%si, 0.0%st
Mem:  16466044k total, 5903700k used, 10562344k free,  87732k buffers
Swap: 16014576k total,        0k used, 16014576k free,  3424420k cached
  PID USER    PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
4322 guest   20   0 6923m 1.4g 103m S  3.9  8.8   7:36.68 soffice.bin
1828 root    20   0  313m 199m  14m S  1.9  1.2   4:17.19 Xorg
   1 root    20   0 19536 1480 1188 S  0.0  0.0   0:00.80 init
   2 root    20   0     0    0    0 S  0.0  0.0   0:00.00 kthreadd
   3 root    20   0     0    0    0 S  0.0  0.0   0:04.07 ksoftirqd/0
   4 root    RT   0     0    0    0 S  0.0  0.0   0:00.06 migration/0
   5 root    RT   0     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
   6 root    RT   0     0    0    0 S  0.0  0.0   0:00.06 migration/1
...
```

Для просмотра процессов в ОС Windows рекомендуем установить программу Process Explorer [247], написанную Марком Руссиновичем, которая значительно превосходит по функционалу штатный (см. рис. 5.4).

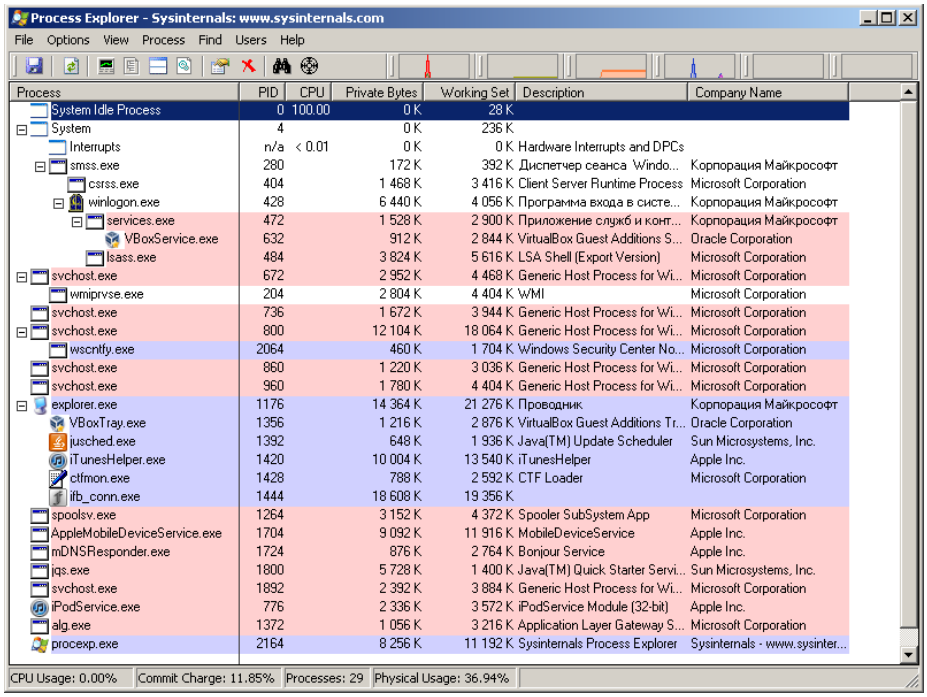


Рисунок. 5.4. Окно программы Process Explorer

[247] Входит в набор полезных программ серии «Windows Sysinternals», доступна для бесплатного скачивания по адресу http://technet.microsoft.com/ru-ru/sysinternals/bb896653.

Данная программа может быть весьма полезной при поиске запущенных вирусов и анализе, что компьютер делает в данный момент. Заметим, что полезными могут оказаться и другие утилиты из набора «Sysinternals», вот некоторые из них:

- FileMon – эта программа предназначена для отслеживания в режиме реального времени всей активности файловой системы.
- DiskMon – фиксирует все операции с жёстким диском; кроме того, она может исполнять роль индикатора активности диска на панели задач.
- Regmon – предназначена для отслеживания в режиме реального времени всей активности реестра.

Функционал большинства подобных программ продиктован здравым смыслом и многолетнесформированными потребностями при работе с ОС системными администраторами. Часть же функционала просто скопирована из UNIX-систем. Например, filemon есть смысловая копия консольной утилиты «lsof» (от list open files), позволяющей отслеживать все открытые файлы. Обе могут быть очень полезны, если вы хотите отмонтировать внешний накопитель (например, флэшку), а система «упорно» не хочет этого делать. Причиной тому могут быть открытые каким-либо процессом файлы на файловой системе подключённого устройства [248].

В различных планшетах и телефонах (точнее, используемых на них ОС) также существует диспетчер задач, с помощью которого можно посмотреть список ранее запущенных программ и завершить (выгрузить из памяти) ненужные. Например, для iOS диспетчер графический (см. рис. 5.5) и вызывается двойным нажатием на кнопку «домой». Далее путём удерживания пальца на иконке запущенного приложения в течение нескольких секунд можно заставить все приложения «дрожать» (и бояться), а пользователь через значок красного кружочка с минусом может завершать их. Понятно, что в таком упрощённом интерфейсе многие системные процессы будут скрыты от пользователя. В некоторых планшетах завершения процесса может выполняться отправлением его окна, как бросок пальцем по экрану, вверх.

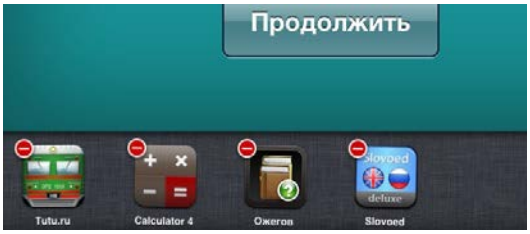


Рисунок. 5.5. Диспетчер задач iOS (значки вибрируют)

Философский смысл процесса

По наблюдениям авторов, в данный момент (2021 год, подготовка 5-го издания) можно отметить что мобильные устройства инициируют начало революции меняющей внутреннее устройство современных ЭВМ. Несомненно она может привести к исчезновению или сильному изменению понятия процесса. Уже сейчас есть трудность в объяснении пользователям мобильных устройств зачем нужен процесс. Ответить на

[248] Отсоединять носитель информации (например флэшку) не отмонтировав перед этим его файловые системы, не стоит, поскольку могут потеряться данные. Если устройство не поддаётся программному отключению и размонтированию, то постарайтесь минимизировать свои потери, – закройте лишние окна и приложения (лучше все) и синхронизируйте данные кэша и флэшки, запустив консольную команду sync.

обозначенный вопрос будет проще, если углубиться в историю. Ранее программы в ПК перед их выполнением находились на внешней, относительно медленной, памяти. Нахождение их там было обусловлено следующими вещами. Внешняя память была энергонезависимой, а оперативная память сбрасывалась при выключении компьютера. Сам компьютер довольно часто выключался. Оперативной памяти (далее – ОЗУ, от *оперативное запоминающее устройство*) в ПК было мало, но она работала быстрее внешней памяти, поэтому требовалось все программы копировать в неё перед запуском. Конечно, пользователи программы в ОЗУ не копировали, копирование происходило для них прозрачно за счёт функционала самой операционной системы. За счёт динамического выделения памяти под данные процесса, последний в памяти занимает больше места нежели программа из которой он был «создан» занимала на диске.

По мере распространения твердотельных дисков (SSD) и увеличения скорости их работы грань между чисто оперативной памятью и памятью «на диске» будет стираться. Если процессор сможет выполнять программу прямо с диска и туда же записывать результаты работы, то, можно будет избежать этапа ненужного копирования. Неизбежно придумают новые правила адресации и использования такой «внешней» памяти. Возможно наступит тот день когда данные быстрее будут копироваться по 128 или 256 бит из сети (облака), чем будет скорость доступа процессора непосредственно к ячейкам ОЗУ.

5.3.3. Файл

Понятие файла (англ. file) в файловой системе является одним из ключевых для любой операционной системы. Для ОС UNIX понятие файла несколько шире, под файлом также рассматривается практически всё, что существует в системе, – исполняемые программы, созданные ими данные, дисковые и другие накопители, а также все прочие устройства. Это справедливо и для ОС Android, и для iOS.

Файл – поименованная область файловой системы, служащая для хранения упорядоченной последовательности байтов.
Файл – упорядоченная совокупность информации на диске, имеющая своё имя.
Файл – это поименованная совокупность семантически связанных данных.

Файл может быть размещён и в оперативной памяти, но чаще всего термин относится к данным, размещённым на устройствах внешней памяти, например на жёстком диске или USB-flash-накопителе, подразумевая тем самым, что посредством файла организованы данные долговременного хранения в энергонезависимой памяти.

Файловая система (ФС, file system) – способ (договорённость, формат) организации выделенного пространства памяти (на жёстком диске, flash-накопителе, дискете, компакт-диске и прочем) с целью обеспечения оптимального хранения в ней информации в виде файлов и получения доступа к ним.

Замечание. Пространством памяти обычно является внешний носитель: жёсткий диск, флэшка, дискета или их часть, однако файловую систему можно создать и внутри файла достаточного объёма. Такое может случится, если вы снимаете образ с жёсткого диска или флэшки в файл. В Linux для этого можно использовать штатную команду «dd».

Фактически имеем, что файлы организованы в виде файловых систем. Термин этот понимается в двух различных смыслах – во-первых, как физическая сущность, то

есть способ хранения данных на диске (или другом накопителе), во-вторых, как логическая структура, в которую они организованы.

Все эти аспекты будут последовательно рассмотрены в этой главе, начиная с постановки задачи управления файлами, классификации файлов, через логическую и физическую их организацию и заканчивая управлением файловыми системами и собственно файлами.

5.3.3.1. Задачи управления файлами

Файлы на носителе организуются в виде ФС. Поддержку файловой системы осуществляет компонент операционной системы – файловый процессор. Часто понятие структуры размещения файлов на носителе – собственно файловую систему – и файловый процессор для ОС Windows (или код ядра для ОС Linux) совмещают, рассматривая ФС одновременно и как структуру, и как программу, поддерживающую эту структуру.

Структура размещения файлов	+	Файловая система	+	Файловый процессор или код ядра ОС	? =	Файловая система

Файловая система позволяет программам обходиться набором достаточно простых операций для выполнения действий над довольно абстрактным объектом, которым является файл. Это позволяет при написании программы не иметь дело с деталями действительного расположения данных на носителе, буферизацией данных и другими низкоуровневыми действиями для передачи [249] данных между оперативной памятью и внешней памятью, которые реализует ФС. Кроме предоставления операций с файлами, ФС распределяет память устройства, поддерживает именование файлов, отображает имена файлов в соответствующие адреса во внешней памяти, обеспечивает доступ к данным, поддерживает разделение, защиту и восстановление файлов при их повреждении.

Таким образом, ФС является промежуточным слоем, инкапсулирующим все сложности физической организации долговременного хранилища данных и создающим для программ более простую логическую модель этого хранилища с набором удобных в использовании команд для манипулирования файлами. Управление файлами имеет несколько уровней организации.

Уровень организации	Описание	Пример	Кратко
логический	Файловые объекты: файлы, директории и т.д.	/dir1/file1.txt	Имена
↓			
внутренний	Номера inode'ов	1371	Номера
↓			
физический	Номера информационных блоков	122,123,124,125	Блоки

[249] Напомним, что процессор в машине, построенной в соответствии с архитектурой Джона фон Неймана, обрабатывает только данные, размещенные в оперативной памяти.

Задачи, решаемые ФС, зависят от способа организации вычислительного процесса в целом [25]. Так, в однопользовательской и однозадачной ОС ФС реализует:

- именование файлов;
- программный интерфейс для приложений;
- отображение логической модели файловой системы на физическую организацию хранилища данных;
- поддержку целостности и доступности данных, хранящихся в файлах.

Задачи ФС усложняются в однопользовательских многозадачных ОС, которые управляют файлом как разделяемым ресурсом, что предусматривает реализацию блокировки файла и его отдельных частей, согласование копий файла, предотвращение тупиковых ситуаций, когда, например, два процесса пытаются получить монопольный доступ к взаимно используемым файлам и т. д.

В многопользовательских системах появляется ещё задача защиты файлов одного пользователя от несанкционированного доступа другого пользователя.

Современные файловые системы, построенные в целях обслуживания пользователей гетерогенных сетей, объединяют несколько файловых систем общим интерфейсом и общим набором правил, предоставляя пользователю возможность не только использовать привычный интерфейс, но и освобождая его от необходимости преобразования форматов и кодировки двоичной и символьной информации.

5.3.3.2. Именование файлов

Любой объект в ОС должен иметь уникальное имя или идентификатор. Имя файла – не исключение. Конечно, можно было бы пересчитать все файлы по порядку, впрочем, как и директории (каталоги, папки), и обращаться к ним по номеру, но помимо функции идентификации имя должно обеспечивать пользователю удобство работы с файлами.

Файлы и каталоги файловой системы именуются в соответствии с некоторыми правилами, для описания синтаксиса которых удобно использовать какую-либо формальную систему. Возможно, программистам со стажем было бы понятнее, если бы мы привели регулярное выражение, но чтобы не объяснять, что такое «регулярные выражения», мы откажемся от этой затеи и рассмотрим предложенную Джоном Бэкусом (John Backus) и усовершенствованную Питером Науром (Peter Naur) и благодаря этому получившую название *Бэкуса–Наура форму* (БНФ).[стр. 353, 30] Не вдаваясь в подробности, приведём коротко суть предложенного подхода на примере расширенной с помощью некоторых метасимволов БНФ (РБНФ).

Могут применяться следующие метасимволы:

Символы	Пример	Пояснение
`|`	`a|b`	*a* или *b*
`[]`	`[.txt]`	квадратные скобки [ ] заключают опциональные (факультативные) элементы
`()`	`(.txt|.mp3)`	круглые скобки ( ) заключают выбор из, возможно, более чем двух элементов
`{}`	`{/директория}`	фигурные скобки { } заключают рекурсивное повторение, возможное и ноль раз

С учётом введённых обозначений определим правило вывода для полного (абсолютного) пути к файлу [250].

В Linux (UNIX):	В Windows (DOS) [251]:
`full-path → /{name/}name`	`full-path → letter:\{name\}name`

В этом правиле `letter:` – обозначение имени логического диска, `\` – обозначение корневой директории, `{name\}` – рекурсивное и опциональное обозначение цепи имён директорий и следующее `name` – обозначение короткого имени файла.

Обозначение «`letter:`», в свою очередь, может быть представлено следующим образом.

```
letter: → (upcase-letter|locase-letter):
```

где `upcase-letter`, `locase-letter` – это латинские буквы в верхнем (заглавные) и нижнем (строчные) регистрах соответственно.

```
name → name-symbol{name-symbol}
```

Имена директорий или файлов представляют собой набор из одного или большего числа символов. Конкретная ОС (файловая система в ОС) может накладывать ограничения на число и порядок символов в именах директорий/файлов. Причём ограничения на имя файла (директории) и на длину полного имени могут быть различны. Так, в ФС FAT16 имена имеют формат 8.3, где 8 – это сообщение того, что собственное *имя файла* (file name) не может превышать 8 символов, за ним следует «.» и далее длиной не более 3 символов *расширение* (extension), обозначающее тип файла. Например, «log.txt», «convert.c», «find.exe». Количество символов расширения в разных системах разное. Так, например, FAT16 допускает расширение не более трёх символов. Файловые системы UNIX не ограничивают использования символа «точка» в имени файла. Имя файла (или директории) даже может начинаться с точки[252]. Например, допустимы имена файлов «Marina.letter», «.themes», «Scan.Document.13.Part.2». Расширения являются удобными, но не обязательными, хотя многие программные продукты подставляют расширение к имени файла автоматически, при их сохранении в том или ином формате. ОС может использовать *регистр-независимое* (case insensitive) именование объектов файловой системы или, наоборот, оперировать *зависимыми от регистра именами* (case sensitive). В первом случае это системы от Microsoft, для которых имена «File.ExT» и «file.eXt» будут именовать один и тот же объект. Во втором случае это системы, поддерживающие стандарт POSIX, в которых такие имена будут соответствовать разным объектам. Формально запись будет выглядеть так:

```
name-symbol → letter|any_allowed_char
```

, где `any_allowed_char` – любой иной символ, допускаемый в именах данной ОС (файловой системой в ОС).

[250] Если «полный путь к файлу» содержит имя файла, то он может также называться как «полное имя файла». В общем случае понятие «путь» не однозначно и не включает имена файлов.

[251] Здесь и далее будут приведены примеры для символьного уровня файловых систем, характерные для ОС Linux и UNIX (на языке оболочки Bourne shell, /bin/bash) и для ОС Windows (интерфейс командной строки DOS-интерпретатора cmd.exe).

[252] Кроме зарезервированных «.» и «..», означающих, соответственно – текущая директория и директория на уровень выше. Для корневой директории директория на уровень выше есть эта же. См. подробнее ниже про относительный путь именования файлов и директорий.

5.3.3.2.1. Максимальная длина имени файла

Какие существуют ограничения на длину имени файла в современных ОС?

В Windows имена томов NTFS могут содержать не более чем 255 символов, а из командной строки можно задать имя файла длиной не более 253 символов [253]. С другой стороны, по данным MSDN [254], полное имя файла (включая расширение) в Windows может содержать до 260 символов, данное значение определено константой MAX_PATH в Windows API; например, максимально допустимое полное имя файла на диске C будет таким «C:\строка_*256_символов*<NUL>».

Для Linux всё проще, смотрим требования POSIX [255], затем заходим на www.kernel.org, скачиваем исходники ядра и смотрим код:
/usr/src/linux-2.6.32.65/include/linux/limits.h
содержит строки

```
#define NAME_MAX  255 /* # chars in a file name */
#define PATH_MAX 4096 /* # chars in a path name including nul */
```

Замечание. В конечном итоге, если вы сомневаетесь в правильности приведённых выше значений, никто не мешает провести эксперимент на реальном компьютере или заглянуть в исходники.

5.3.3.2.2. Используемые символы (в именах)

В ОС для обозначения имён в файловой системе могут использоваться символы поддерживаемых письменностей, например буквы кириллицы или иероглифы в зависимости от того, что поддерживает интерфейс ОС. Однако не все символы могут использоваться для создания имён в файловой системе. Так, в DOS не могут использоваться символы **" * + , ; : < = > ? | /** (**** уже используется). В файловых системах, работающих под управлением Windows, множество запрещённых символов имеет отличный вид: **" * / : < > ? |** (**** также занят разделением каталогов и файлов), а правило вывода для полного пути независимо от ОС имеет вид:

```
full-path → [letter:](\|/){name(\|/)}name
```

В Linux и UNIX-подобных ОС запрещён только слэш (/) – разделитель подкаталогов – и символ конца строки (\0) [256]. Перечисленные выше символы (кроме слэша) использовать можно, но из соображений совместимости их лучше избегать.

Для обозначения объектов файловой системы может использоваться относительный (относительно некоторой, чаще всего текущей, ещё её называют рабочей, директории) путь:

```
relative-path → (..|.|(\|/){name(\|/)}name
```

. – текущая директория, **..** – переход в родительскую директорию из текущей директории, символ **** или **/** – если идёт вначале – обозначение корневой директории, если в середине – разделитель имени двух директорий или директории и файла.

В ОС Linux ~ – домашняя директория.

[253] http://support.microsoft.com/kb/100108.
[254] http://msdn.microsoft.com/en-us/library/windows/desktop/aa365247%28v=vs.85%29.aspx.
[255] http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap03.html.
[256] http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap03.html.

Если файловая система логического диска, с которым работает пользователь, с точки зрения ОС Windows, имеет вид, как показано на рис. 5.6, и текущей директорией является директория `D`, то относительный путь `\F.ext` эквивалентен абсолютному пути `C:\A\D\F.ext`, относительный путь `..\..\B` эквивалентен абсолютному пути `C:\B`, относительный путь `.\` эквивалентен абсолютному `C:\A\D`.

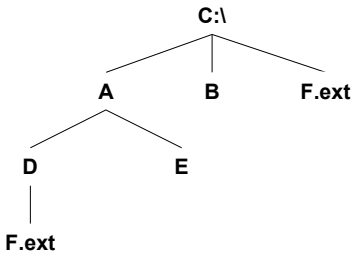


Рисунок 5.6. Древовидное представление объектов файловой системы

Файловые системы могут поддерживать несколько имён для своих объектов. Например, в ФС ext2-4 можно сделать две разные жёсткие ссылки из разных директорий на один inode; в ФС NTFS один и тот же файл в одной директории может иметь имя «*long file name.txt*» (длинное имя) и «*LONGFI~1.TXT*» (короткое имя). Обращение к файлу по обоим именам будет эквивалентным.

Допустимость имён директорий/файлов, содержащих пробелы, ведёт к необходимости использования синтаксического механизма языка командной строки, позволяющего указанное имя рассматривать как целостный элемент программирования. Это достигается через заключение такого имени в двойные кавычки.

Например, в Windows (DOS):

```
C:>cd "C:\Documents and Settings\user"
```

в Linux (UNIX):

```
$ cd "/home/user/Рабочий стол/dir1"
```

Также в Linux (UNIX) возможен и второй вариант записи – через экранирование пробелов знаком "\" (обратный слэш, back slash).

```
$ cd /home/user/Рабочий\ стол/dir1
```

5.3.3.2.3. Подстановка имён файлов

При работе в режиме командной строки довольно много времени уходит на поиск необходимых файлов. Интерпретатор shell предлагает набор метасимволов, позволяющих находить файлы, имена которых соответствуют предложенному шаблону. Иногда данные *метасимволы* называют *символами-джокерами* (wildcard characters), фактически они используются для замены других символов или их последовательностей. С помощью джокеров записываются символьные шаблоны.

Вот список основных метасимволов [257][2]:

`*`	Соответствует произвольной строке, содержащей ноль и более символов
`?`	Соответствует любому символу
`[ ... ]`	Соответствует любому символу из числа заключённых в квадратные скобки
`[! ... ]`	Соответствует любому символу, за исключением тех, которые указаны в квадратных скобках

[257] Когда интерпретатор shell встречает указанные символы в командной строке, он обрабатывает их особым образом, если только вы не защитили их с помощью кавычек.

Таким образом можно работать (выводить список, удалять, копировать и т. д.) сразу с группами файлов. Например, «*.txt» обозначает все файлы с расширением «.txt», а «lect.*» означает все файлы с именем, начинающимся на «lect.». В ОС Windows «*.*» исторически означает все файлы, а вот в Linux и UNIX для этой цели используется одна звёздочка «*». «?a.jpg» обозначает все файлы с именами из двух символов, второй из которых «a»: «aa.jpg», «ba.jpg» и т. д.

В bash на ряду с символами джокерами могут использовать конструкции с фигурными скобками осуществляющими перебор «с» и «по», например команда: «mkdir ab{01..31}» создаст директории с именами: ab01, ab02, ab03, … , ab31.

5.3.3.2.4. Дескриптор файла (индексный дескриптор)

Для управления файлом в файловой системе используются его описатель. Поскольку все файловые объекты проиндексированы, то он часто называется индексным дескриптором. Ещё его называют дескриптором файла или файловой записью (file record). В нём содержится вся необходимая информация о файле как об источнике данных: тип файла; права доступа; дата создания; даты последнего обращения и последней модификации; размер файла; текущее состояние файла (открыт/закрыт, количество одновременных пользователей, модифицирован или нет и т. п.); логическая структура доступа (в случае простой структуры) или указатель на неё, указатели на предка и потомков в дереве каталога. В рамках VFS дескрипторы файловых объектов соотносятся с дескрипторами тех файловых систем которые монтируются. Фактически сегодняшнее содержание дескриптора полностью повторяет структуру inode ФС ext (см. табл 5.1).

Таблица 5.1. Типичное содержание дескриптора файлового объекта

Поле структуры дескриптора	Примечание
Допуск/права доступа	Информация о правах на файл отдельных пользователей или групп пользователей
Тип файла	Данные, директория, метаданные [258], устройство [259]
Длина	В байтах или машинных словах
Время создания	год-месяц-день-час-минута-секунда
Время последнего изменения	год-месяц-день-час-минута-секунда
Время последнего доступа	год-месяц-день-час-минута-секунда
Счётчик ссылок	Число псевдонимов имени файла (жестких/символьных ссылок)
Размещение	Массив экстентов с указанием, какие физические блоки связаны с соответствующими логическими блоками

[258] Сами метаданные на логическом уровне ФС могут быть организованы в файлы для единообразного управления ими. Так, в ФС NTFS массив файловых записей тома является файлом.

[259] Прикладной интерфейс ФС в ряде случаев удобно использовать для обращения к устройствам. В этом случае устройство логически рассматривается как файл, с которым производятся все те же операции чтения/записи, что и с обычным файлом. На физическом уровне эти операции, конечно, различаются.

Дескриптор – это пример *метаданных* [260] *ФС*, то есть данных, которые не содержатся в файлах, но используются для поддержания самой системы хранения файлов на носителе внешней памяти.

Структура дескриптора файла в существенной степени зависит от конкретных ОС и ФС. Как правило, дескрипторы файлов хранятся во внешней памяти перед описываемыми файлами и передаются в основную память в момент их открытия.

Если ФС не поддерживает какие-то из параметров дескриптора, то в оперативной памяти они дополняются виртуальными значениями, необходимыми для успешной работы ОС.

Дескриптор файла – это структура, на которую ссылается символьное имя, используемое прикладной программой для доступа к файлу.

В случае использования нескольких псевдонимов для файла все они ссылаются на один дескриптор файла. Такой подход очень гибок и позволяет манипулировать внешним (символьным) представлением иерархии файлов, не затрагивая самих файлов.

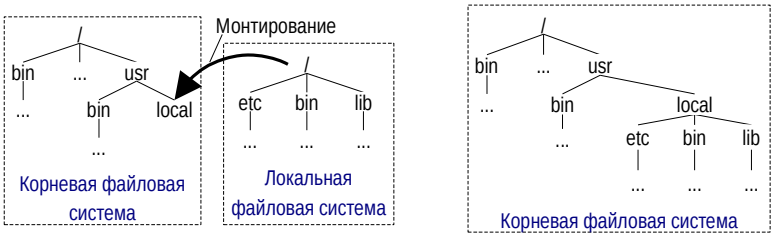
Каталоги файлов используются для логического управления файлами и дают возможность пользователям систематизировать их по предметным областям информации, хранящейся в файлах, и по пользователям, имеющим доступ к файлам. Имя файла удобно хранить в каталоге, чтобы обеспечить доступ к файлу как к конкретной совокупности данных по различным символьным путям. Записи в каталоге могут быть очень простыми: содержать имя файла и ссылку на его дескриптор.

В случае использования ФС ext2/3/4, дескриптором файла будет inode, подробнее см. параграф 5.3.7.5. Про организацию каталогов см. разделы 5.3.3.2.5 «*Стратегии адресного пространства*» и 5.3.4 «*Логическая структура файловой системы. Стандарт FHS*».

5.3.3.2.5. Стратегии адресного пространства

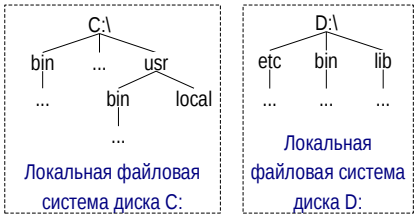
Адресное пространство внешних запоминающих устройств может формироваться с использованием глобальной или локальной стратегии. В случае глобальной стратегии операционная система считает, что все запоминающие устройства вычислительной системы составляют единое адресное пространство и существует в рамках единого каталога файлов (как виртуальная файловая система VFS в ОС UNIX/Linux). При локальной стратегии на каждом томе находится своё адресное пространство (как в ОС Windows).

Интересно отметить, что в ОС Linux даже корневая файловая система первоначально монтируется к виртуальной (VFS) в момент загрузки системы по адресу «/».



а) *б*)

[260] Метаданные – данные о данных (источник данных, когда, какой формат данных используется и т. п.).



в)

Рисунок 5.7. Примеры формирования адресного пространства внешней памяти: б – глобальное именование файлов, создаваемое с помощью монтирования (а) локальной файловой системы в корневую; в – локальное именование файловых систем

Подключение в коневую файловую систему происходит через *монтирование* (mount) в один из её каталогов локальной файловой системы некоторого тома. Операция монтирования включает несколько этапов:

1) определение типа монтируемой системы;
2) проверка целостности монтируемой системы;
3) считывание системных структур данных и инициализация соответствующего модуля файлового процессора в ОС;
4) установка флага, что система смонтирована;
5) включение пространства имён новой файловой системы в общее пространство имён корневой системы.

Для операции монтирования используются *метаданные монтирования* локального тома, обычно располагающиеся в самом его начале. Каждый том имеет свой главный каталог файлов, ссылка на который содержится в метаданных монтирования тома.

5.3.3.3. Типы файлов

Поскольку понятие файла применяется к достаточно разнородным вещам (файл как таковой, каталог, физические устройства и т. п.), возникает разделение файлов на типы. В UNIX существует шесть (семь) типов файлов:

- обычный файл (regular file);
- директория, каталог (directory);
- специальный файл устройства (special device file);
- именованный канал (named pipe, FIFO);
- символическая ссылка (symbolic link);
- доменное гнездо, или сокет (socket);
- жёсткая ссылка (hard link).

Хотя мы сказали о шести типах, выше приведено семь, так как жёсткая ссылка (hard link), по сути, файлом не является – это псевдоним имени файла, фактически связывающей символьное имя с номером дескриптора (Например связь «abc» – 1234.) Если псевдоним один – он и есть имя файла. Если их несколько – это разные и равносильные имена одного файла. Более ранняя жёсткая ссылка не имеет никаких преимуществ при работе с файлом в отношении более поздних, и наоборот.

5.3.3.3.1. Обычный файл

Это именованная совокупность данных. Для операционной системы обычный файл представляет собой просто последовательность байтов. Интерпретация содержимого файла осуществляется прикладной программой.

5.3.3.3.2. Каталог (директория)

Каталоги являются элементами иерархического дерева. Любой каталог может содержать файлы и подкаталоги. Каталог – это обычный файл, содержащий список записей с именами файлов и указателями на метаданные этих файлов. Отличие от обычного файла заключается в порядке работы с каталогами. Все операции по модификации каталогов выполняются системой, а не прикладными программами. Это позволяет поддерживать целостность хранящейся там информации.

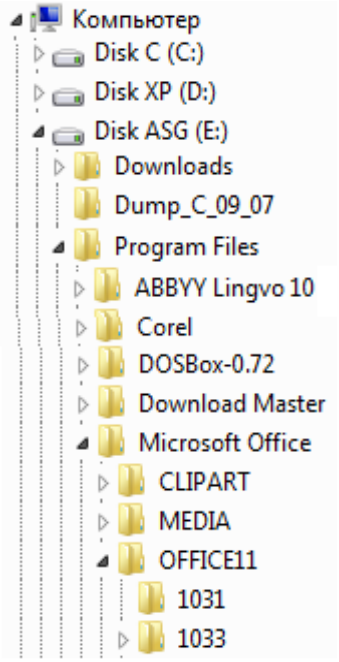


Рисунок 5.8. Дерево каталогов (папок) ОС Windows

5.3.3.3.3. Файл устройства (блочный и символьный)

В операционной системе Linux доступ к устройствам осуществляется через специальные файлы. Такой файл является точкой доступа к драйверу устройства. Существует два типа файлов устройств: символьные и блочные. **Символьный файл устройства** используется для небуферизированного обмена данными с устройством – байт за байтом. **Блочный файл устройства** используется для обмена с устройством блоками данных. Некоторые устройства имеют как символьный, так и блочный интерфейс. Номенклатура специальных файлов зависит от системы, и одинаковым физическим устройствам в разных версиях UNIX или Linux могут соответствовать файлы с разными именами. Специальные файлы устройств могут соответствовать как физически существующему оборудованию – дисковым накопителям, терминалам и т. п., так и виртуальным устройствам – программному датчику случайных чисел /dev/random и др. Хранятся файлы устройств на отдельной виртуальной файловой системе, расположенной на пути /dev.

5.3.3.3.4. Именованный канал

Файлы этого типа используются для связи между процессами для передачи данных. Название файлов данного типа произошло от дисциплины передачи данных – First In First Out (первый вошёл – первый вышел). Именованные каналы являются од-

нонаправленным средством передачи данных, причём чтение данных происходит строго в порядке их записи. Именованный канал может быть создан из командной строки

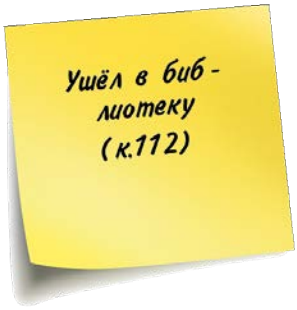
```
$ mkfifo name
```

либо программным путём посредством соответствующего системного вызова. После создания канал может быть открыт на запись и чтение, причём запись и чтение могут происходить в разных независимых процессах.

Каналы работают по следующим правилам:

1. При чтении меньшего числа байтов, чем находится в канале, возвращается требуемое число байтов, остаток сохраняется для последующих чтений.
2. При чтении большего числа байтов, чем находится в канале или FIFO, возвращается доступное число байтов. Процесс, читающий из канала, должен соответствующим образом обработать ситуацию, когда прочитано меньше, чем заказано.
3. Если канал пуст и ни один процесс не открыл его на запись, при чтении из канала будет получено 0 байтов. Если один или более процессов открыли канал для записи, операция чтения будет заблокирована до появления данных (если для канала не установлен флаг отсутствия блокирования).
4. Запись числа байтов, меньшего ёмкости канала или FIFO гарантирована атомарно. Это означает, что в случае, когда несколько процессов одновременно записывают в канал, порции данных от этих процессов не перемешиваются.
5. При записи большего числа байтов, чем это позволяет канал, операция записи блокируется до освобождения требуемого места. При этом атомарность операции не гарантируется.

5.3.3.3.5. Символическая ссылка (мягкая ссылка)



Файлы данного типа являются прямым аналогом ярлыков (shortcut) в Windows. Символическая ссылка содержит только имя файла. Возможно создание ссылок на файлы, расположенные в другой файловой системе. Эти ссылки могут указывать на файл любого типа, даже на несуществующий. В UNIX следует различать символические ссылки («мягкие») и так называемые «жёсткие» ссылки, которые представляют собой, по сути, альтернативные имена для файлов. Если жёсткая ссылка одна, то она и есть файл. Если ссылок несколько, то все жёсткие ссылки равноправны, и для удаления файла необходимо удалить все существующие на него жёсткие ссылки. По сути, жёсткая ссылка – это элемент каталога, тогда как символическая ссылка – отдельный файл.

Создать мягкую ссылку в ОС Linux можно командой:

```
$ ln -s file.txt name2.txt
```

где name2.txt будет мягкой ссылкой на файл (файловый объект) file.txt. Вместо коротких имён может использоваться абсолютная адресация. Мягкие ссылки могут быть созданы не обязательно в пределах одного физического тома.

5.3.3.3.6. Доменное гнездо (сокет)

Механизм доменных гнёзд (Sockets) впервые был реализован в 1982 году в UNIX BSD 4.1 в качестве развитого средства межпроцессных взаимодействий. Это средство, вообще говоря, позволяет любому процессу обмениваться сообщениями с любым другим процессом, независимо от того, выполняются они на одном компьютере или на разных, соединённых сетью. Функционально механизм программных гнёзд близок к возможностям TLI (Transport Level Interface – интерфейс транспортного уровня, пятый уровень в соответствии с моделью ЭМВОС (ISO/OSI-RM)). Программные гнёзда входят в число обязательных компонентов стандартной среды ОС UNIX, однако реализуются в разных системах по-разному. В BSD-ориентированных системах Sockets исторически реализуются в ядре ОС, и пользователям предоставляются пять специальных системных вызовов: socket, bind, listen, connect и accept. В UNIX System V Release 4 тоже поддерживается механизм программных гнёзд, однако он реализован не внутри ядра системы, а в виде набора библиотечных функций (библиотеки /usr/lib/libsocket.a), которые написаны с использованием механизма TLI. Заметим, что это в очередной раз демонстрирует преимущества подхода открытых систем, который всегда поддерживался в мире ОС UNIX: при наличии чётко определённых интерфейсов и развитых базовых средств прикладной программист и разработанные им программы не должны зависеть от конкретной реализации. Типичный пример использования доменных гнёзд – доступ к сети посредством протоколов TCP/IP.

5.3.3.3.7. Жёсткая ссылка (hard link)

Жёсткая ссылка, по сути, файлом не является – это псевдоним имени файла. Если же псевдоним один – он и есть имя файла. То есть, в общем случае, одна и та же совокупность данных может иметь несколько имён. Каждое имя может отражать смысл использования файла в разных обстоятельствах. При создании файла для него указывается имя. Если позднее на этот же файл будет создана жёсткая ссылка, то есть этот же файл будет назван иначе, то оба имени будут равноправными. Файл существует до тех пор, пока существует хотя бы одно из имён. Жёсткие ссылки возможны только на одном томе [261].

Создать жёсткую ссылку в ОС Linux можно командами:

```
$ echo "тест">1.txt
$ ls -li
100007963 -rw-rw-r-- 1 user user 9 мая   24 19:17 1.txt
$ ln 1.txt 2.txt
$ ls -li
100007963 -rw-rw-r-- 2 user user 9 мая   24 19:17 1.txt
100007963 -rw-rw-r-- 2 user user 9 мая   24 19:17 2.txt
```

Командой «*ln источник новая_ссылка*» делается ссылка, после чего мы видим, что номера inode, на которые ссылаются имена файлов 1.txt и 2.txt, одинаковые.

[261] Интересный факт: жёсткие ссылки, а точнее псевдонимы имён файлов, поддерживаются и на компакт-дисках, например в файловой системе iso9660. В ОС Linux создание таких дисков проходит штатным образом. Если при переносе данных в iso-образ, например с помощью утилиты mkisofs или growisofs, файлы с разными именами оказываются одним и тем же файлом, то и внутри образа этот файл также записывается один раз.

Если жёсткие ссылки в ОС UNIX и Linux были с рождения, то в ОС Windows они пришли вместе с файловой системой NTFS (начиная с Windows NT4). В этой ФС каждый файл также можно считать жёсткой ссылкой на самого себя. На файл может ссылаться сколько угодно жёстких ссылок (точнее, не больше 1023). Они могут запускать его из разных расположений и при этом не отличимы друг от друга в проводнике и результатах команды dir.

Команда для создания жёсткой ссылки в Windows:

```
fsutil hardlink create новая_ссылка источник
```

Обратите внимание, тут у Windows тоже не так, как в Linux и UNIX, существующий файл идёт вторым! А создаваемый – первым.

Например:

```
:: Создаём произвольный текстовый файл
echo "тест" > 1.txt
:: Создаём жесткую ссылку на файл
fsutil hardlink create 2.txt 1.txt
:: Создаём жёсткую ссылку на жёсткую ссылку
fsutil hardlink create 3.txt 2.txt
```

Дополнительно отметим, что в Windows Vista для создания жёстких ссылок появилась новая команда mklink, делающая оба вида ссылок.

```
:: Символическая ссылка на файл
mklink %windir%\pe.exe C:\Sysinternals\procexp.exe
:: Жёсткая ссылка на файл
mklink /h %windir%\pe1.exe C:\Sysinternals\procexp.exe
```

В Windows XP нет возможности создать жёсткую ссылку на директорию. В последующих версиях проблема решена, но не полностью, Link Shell Extension не создаёт жёстких ссылок на каталоги в Windows 7.

Если в Linux жёсткие ссылки – давно работающая штатная функция команды ln и никому не приходит в голову её переписывать, то под Windows программы, работающие с жёсткими ссылками, появляются с завидной регулярностью, например вот наиболее известные:

NTFS-Link – дополнение к оболочке Windows, позволяющее создавать жёсткие и символьные ссылки на томах NTFS. Существующие символьные ссылки помечаются в Проводнике маленькой дополнительной иконкой.

Link Shell Extension – ещё одно дополнение к оболочке Windows. Последняя версия проверена и работает в Windows Vista/7. Существующие жёсткие и символьные ссылки помечаются в Проводнике маленькой дополнительной иконкой. Файлы и каталоги с существующими ссылками имеют дополнительную закладку в «Свойствах» со списком всех жёстких ссылок и целевым местом символической ссылки.

NTFS Links – плагин для файлового менеджера Total Commander, позволяющий создавать жёсткие и символьные ссылки на томах NTFS из Total Commander. FAR Manager – аналогичный файловый менеджер, также позволяет работать ссылками на NTFS-разделах (создавать, просматривать, удалять жёсткие и символьные ссылки).

5.3.4. Логическая структура файловой системы. Стандарт FHS

Структура каталогов – понятие чисто логическое и к реальным механизмам работы с файлами отношения не имеет. Каждая конкретная операционная система могла бы строить её по-своему, что привело бы к несовместимости и непереносимости программ.

Группой энтузиастов из сообщества разработчиков программ с открытым кодом была предложена спецификация структуры каталогов для UNIX-подобных систем, которая была оформлена в стандарт иерархии файловых систем (Filesystem Hierarchy Standard, FHS).

Работа над FHS началась в августе 1993 года с попытки упорядочить структуру файлов и каталогов Linux. Сначала его называли проектом стандартов файловой системы Filesystem Standards Project (FSSTND), первая версия была выпущена 14 февраля 1994 года. В начале 1995 года была поставлена задача по созданию более общей версии FSSTND, предназначенной не только для Linux, но и для других Unix-подобных систем, в первую очередь BSD 4.4. Учитывая расширение сферы действия стандарта, его переименовали в FHS. Стандарт вобрал положительные качества, присущие BSD и другим системам в части поддержки различных архитектур и учёта требований работы в гетерогенных сетях [262].

При разработке этого стандарта, во-первых, учитывалось, что, хотя в Unix-подобных системах структура каталогов представлена в виде единого дерева, отдельные его «ветви» могут располагаться на разных носителях или в разных файловых системах. Размещение файлов на разных носителях позволяет оптимизировать процессы загрузки, последующего функционирования и возможного обновления системы. При этом файловые системы могут физически располагаться на разных компьютерах и быть различными по своей внутренней организации (ext4fs, vfat и т. д.). Во-вторых, любая Unix-система – система сетевая. Поэтому при размещении отдельных файлов в различных частях файловой структуры учитывалось, что некоторые файлы должны быть доступны с других компьютеров в сети, а к иным файлам доступ по сети необходимо ограничить. Группа неразделяемых файлов вычленяется как по соображениям безопасности, так и просто потому, что эти файлы определяют локальную конфигурацию системы и поэтому нужны только на данном компьютере. Выделение группы разделяемых файлов позволяет также экономить общее дисковое пространство. В-третьих, существуют файлы, изменять которые может только администратор, и те, которые любой пользователь может менять самостоятельно. К числу статических относятся исполняемые файлы, библиотеки, документация и др. Для рядовых пользователей эти файлы должны быть доступны только для чтения. Знание этих предпосылок помогает понять

[262] На время написания настоящего учебника действовала версия стандарта 2.3 (www.pathname.com/fhs), анонсированная 29 января 2004 года.

логику размещения отдельных файлов и каталогов в структуре каталогов, предлагаемой стандартом FHS.

Поскольку структура каталогов к реальным механизмам работы с файлами отношения не имеет, изначально никаких особых требований к структуре логического дерева каталогов со стороны операционной системы не предъявляется, поэтому каждая ОС могла бы строить это дерево по-своему. Легко понять, что это привело бы к несовместимости. Стандарт FHS гласит, что содержание корневого каталога должно быть адекватным для загрузки, восстановления и/или устранения ошибок в системе. Согласно стандарту, в корневом каталоге требуется наличие следующих каталогов или символических ссылок на них:

- bin – файлы основных команд (утилит), которые необходимы, когда никакая другая файловая система ещё не смонтирована (например, в однопользовательском режиме);
- boot – неизменяемые файлы, необходимые для загрузки системы;
- dev – файлы устройств;
- etc– файлы конфигурации системы на данном компьютере;
- home – домашние каталоги пользователей (необязательно);
- lib – основные разделяемые библиотеки и модули ядра;
- lib<qual> – основные разделяемые библиотеки для альтернативных форматов (необязательно);
- media – точка монтирования для подключаемых файловых систем на сменных носителях (необязательно);
- mnt – точка монтирования для временно подключаемых файловых систем;
- root – домашний каталог пользователя root (необязательно);
- opt – дополнительные пакеты программного обеспечения;
- sbin – основные системные исполняемые файлы;
- srv – данные услуг, предоставляемых этой системой;
- tmp – временные файлы;
- usr – иерархия второго уровня;
- var – переменные данные.

В соответствии с требованиями стандарта приложения не должны создавать файлы и каталоги или требовать наличия каких-то специальных файлов и каталогов (помимо перечисленных) в корневом каталоге. Во-первых, размер корневой файловой системы желательно сохранять по возможности малым, а во-вторых, стандарт FHS обеспечивает достаточную гибкость и удобство размещения файлов, не попавших в корневую систему, в других файловых системах и подкаталогах. Некоторые подкаталоги корневого каталога необязательны. Но уж если они существуют, то должны размещаться в корневом каталоге, но не обязательно в корневой файловой системе.

Каталог **/bin** содержит команды, которые могут использоваться как администратором, так и рядовыми пользователями, причём только те команды, которые необходимы, когда никакая другая файловая система, кроме корневой, ещё не смонтирована (например, в однопользовательском режиме). Те утилиты, которые не так важны, чтобы размещаться в корневой файловой системе, должны размещаться в каталоге /usr/bin. В /bin обязательно должны иметься следующие команды (или символические ссылки на них): *cat, chgrp, chmod, chown, cp, date, dd, df, dmesg, echo, false, hostname, kill, ln, login,*

ls, mkdir, mknod, more, mount, mv, ps, pwd, rm, rmdir, sed, sh, stty, su, sync, true, umount, uname, csh, ed, tar, cpio, gzip, gunzip, zcat, netstat, ping. В каталоге /bin не должно быть подкаталогов.

Каталог **/boot** содержит всё, что необходимо в процессе загрузки, исключая конфигурационные файлы установщика и карты загрузки. Ядро операционной системы должно располагаться либо в корневом каталоге /, либо в /boot; программы, необходимые загрузчику для организации загрузки файлов, должны размещаться в /sbin, а конфигурационные файлы загрузчика – в /etc.

/dev – место расположения специальных файлов устройств. На случай, если потребуется создавать файлы устройств вручную, каталог /dev должен содержать команду MAKEDEV, которая может создать файл устройства в случае необходимости.

В прошлом /dev была частью нормальной файловой системы и состояла из специальных файлов, созданных при установке системы и хранящихся на жёстком диске. Обычно /dev занимала очень много места, чтобы поддерживать множество жёстких дисков, консолей и т. д. Если файл для устройства не присутствовал, его нужно было создавать специальной программой mknod или MAKEDEV. Хотя старая модель работала, она была сложной и неудобной. В ядрах версии 2.4 появилась альтернатива под названием DevFS. Принцип – файловая система /dev создаётся ядром при каждой загрузке и хранится в оперативной памяти. Если добавляются новые устройства, ядро просто добавляет запись, соответствующую им, в /dev. Если устройство требует специальной конфигурации для корректной работы с DevFS, то существует конфигурационный файл (обычно /etc/devfsd.conf). (В некоторых версиях Linux вместо /dev используется /udev.)

Каталог **/etc** содержит конфигурационные файлы и каталоги, специфичные для данной конкретной системы, но в нём не должно быть двоичных файлов. В соответствии со стандартом FHS каталог в обязательном порядке должен содержать подкаталог /opt, в котором должны размещаться подкаталоги с конфигурационными файлами отдельных пакетов и приложений. Для каждого установленного пакета <package> должен создаваться конфигурационный каталог /etc/opt/package. В каталоге /etc должны содержаться следующие каталоги и файлы:

/X11	конфигурационные файлы X Window
/sgml	конфигурационные файлы для SGML и XML
csh.login	общесистемный инициализационный файл для csh
exports	список контроля доступа для сетевой файловой системы NFS
fstab	постоянная информация для монтирования файловых систем
ftpusers	список контроля доступа для демона FTP
gateways	список шлюзов для демона routed
gettydefs	установки терминала, используемые демоном getty
group	список групп пользователей в системе
host.conf	файл конфигурации для системы разрешения имён
hosts	постоянная информация об именах компьютеров
hosts.allow	список компьютеров, с которых разрешён доступ в систему
hosts.deny	список хостов, с которых запрещён доступ в систему
hosts.equiv	список доверенных компьютеров для rlogin, rsh, rcp

hosts.lpd	список доверенных компьютеров для процесса печати lpd
inetd.conf	конфигурационный файл для процесса inetd
inittab	конфигурационный файл для демона init
issue	сообщение, выдаваемое системой до регистрации пользователя
ld.so.conf	список каталогов для поиска разделяемых библиотек
motd	сообщение, выдаваемое системой после регистрации пользователя
mtab	динамически изменяющаяся информация о смонтированных файловых системах
mtools.conf	конфигурационный файл для mtools
networks	статическая информация о сетевых именах
passwd	файл для хранения учётных записей пользователей
printcap	база данных с настройками принтеров для демона lpd
profile	общесистемный файл инициализации для оболочки, запускаемой при входе пользователя в систему
protocols	перечень IP-протоколов
resolv.conf	конфигурационный файл для системы разрешения имён
rpc	перечень протоколов удалённого вызова процедур
securetty	файл со списком устройств, с которых может заходить пользователь root
services	имена портов для сетевых служб
shells	список имеющихся в системе оболочек
syslog.conf	конфигурационный файл для процесса syslogd

Файл mtab не соответствует неизменяемой природе файлов, размещённых в /etc, и помещён в данный каталог в виде исключения, по историческим причинам.

В небольших системах каждый домашний каталог пользователя является одним из непосредственных подкаталогов каталога **/home**, таких как /home/smith, /home/operator и т. д. В больших системах (особенно когда каталоги /home являются разделяемыми между многими машинами) полезно объединить домашний каталог в группы, введя подкаталоги групп, такие как /home/staff, /home/students. Поскольку структура домашних каталогов различается от машины к машине, никаких требований на неё не налагается.

/lib содержит разделяемые библиотеки, необходимые для загрузки системы и запуска команд из каталогов /bin и /sbin. По крайней мере, один из файлов, соответствующих каждому из следующих шаблонов, должен найтись в данном каталоге (это могут быть либо реальные файлы, либо символические ссылки): libc.so.* – динамически подсоединяемые библиотеки Си; ld* – загрузчик времени выполнения. Не должны располагаться в /lib разделяемые библиотеки, которые необходимы только исполняемым файлам, расположенным в /usr (таким как двоичные файлы X Window). В частности, библиотека libm.so.* может быть расположена в /usr/lib, если она не требуется никаким программам из /bin и /sbin. Может существовать более одного варианта каталога /lib в системах, поддерживающих более одного формата исполняемых файлов (например, 32- и 64-разрядные форматы), при этом для каждого формата требуется свой отдельный вариант разделяемых библиотек (которые могут называться **/lib32** и **/lib64**).

/media предназначен для подключения файловых систем, размещённых на сменных носителях. В этом каталоге размещаются подкаталоги, соответствующие устрой-

ствам со сменными носителями, например cdrom и флэшка (название по тому носителя, задаваемому при форматировании).

/mnt предназначен для временного монтирования файловых систем по мере необходимости. Содержимое этого каталога индивидуально для каждой системы и не должно никоим образом влиять на работу запускаемых программ.

/opt резервируется стандартом FHS для установки дополнительных программных пакетов. Предполагается, что любой такой пакет должен размещать свои статические файлы в отдельной структуре /opt/<package>, где /<package> – название пакета.

Исполняемые программы располагаются в каталоге /opt/<package>/bin, а в /opt/<package>/man размещаются страницы обычного для UNIX интерактивного руководства man. Файлы пакета, которые являются переменными, должны устанавливаться в /var/opt, а специфичные для компьютера конфигурационные данные должны устанавливаться в /etc/opt. Никакие файлы пакета не должны размещаться вне каталогов /opt, /var/opt и /etc/opt, кроме тех файлов, которые должны оказаться в других местах по той причине, что иначе пакет не сможет функционировать нормально.

Например, файлы блокирования устройств должны располагаться в /var/lock, а файлы устройств – в /dev.

/root – домашний каталог суперпользователя. Рекомендуемое место расположения – корневая файловая система. В FHS подчеркивается, что учётная запись суперпользователя должна использоваться исключительно для системного администрирования и её не рекомендуется использовать для выполнения задач, которые могут быть выполнены непривилегированными пользователями [263].

/sbin содержит утилиты для выполнения задач системного администрирования (и другие команды, используемые только пользователем root). Этот каталог включает исполняемые файлы, необходимые для загрузки системы и её восстановления в различных ситуациях (restoring, recovering and/or repairing the system), не попавшие в каталог /bin.

Единственная команда, которая обязательно должна присутствовать в /sbin, – shutdown. К примеру, команда ping часто используется простыми пользователями и по этой причине должна размещаться в /bin. Авторы стандарта рекомендуют предоставить всем пользователям право на чтение и выполнение для всех файлов этого каталога, кроме, может быть, программ с установленными битами setuid и setgid.

/srv содержит данные об услугах, предоставляемых данной системой, таких, например, как ftp, cvs, www. Каких-либо устоявшихся соглашений в отношении структурирования этой информации в каталоге в настоящее время нет.

Каталог **/tmp** предназначен для хранения временных файлов, создаваемых в процессе работы различных программ. Рекомендуется удалять все файлы и каталоги в /tmp при каждой загрузке системы.

Второй по важности раздел файловой системы – каталог **/usr** – содержит разделяемые данные, предназначенные только для чтения. Это означает, что он может быть доступен с различных FHS-совместимых машин без права записи в него. Любая информация, которая специфична для конкретной машины или может изменяться со вре-

[263] Заметим, что данное правило очень хорошо реализуется в ОС Android и iOS. Получение прав суперпользователя обычными пользователями в данных ОС не предусмотрено разработчиками указанных ОС, а процедуры их получения стали называться «root-нуть» и «jailbreak-нуть». Заметим, что выполнение указанных процедур, если вы их выполняете на своих устройствах, не может быть ограничено, так как указанное право на территории РФ её гражданам предоставлено ст. 1280 ГК РФ, подробнее см. главу 3.

менем, должна записываться в другое место. Программные пакеты не должны создавать подкаталоги непосредственно в каталоге /usr. Исключение сделано для X Window в силу сложившихся традиций. В /usr содержатся следующие каталоги и символические ссылки:

- bin – основное место для размещения исполняемых файлов системы. /usr/bin/X11 должен быть символической ссылкой на /usr/X11R6/bin, если последний существует. Интерпретаторы языков Perl, Python и Tcl должны вызываться из /usr/bin.
- include – место, где должны быть размещены все системные подключаемые файлы общего пользования для языка C, в частности файлы заголовков. Символическая ссылка /usr/include/X11 должна указывать на /usr/X11R6/include/X11, если таковая существует;
- lib – содержит объектные файлы, библиотеки и внутренние исполняемые файлы, которые не могут вызываться непосредственно пользователями из командной строки или сценариев оболочки;
- lib<qual> - библиотеки для альтернативных форматов;
- local – каталоговая структура для локально устанавливаемых программ;
- sbin – системные команды, не являющиеся жизненно важными для системы и поэтому не попавшие в /sbin. Локально устанавливаемые системные программы должны размещаться в /usr/local/sbin;
- share – архитектурно-независимые данные;
- X11R6 – структура для X Window;
- games – игры и обучающие программы;
- lib<qual> – библиотеки для альтернативных форматов;
- src – исходные тексты программ.

Каталог **/usr/local** используется для установки программ, которые будут применяться локально в рамках данного компьютера. Он может использоваться для программ, не попавших в каталог /usr, доступ к которым разрешён с других компьютеров.

Этот каталог не должен перезаписываться при обновлениях системного программного обеспечения. Поскольку в этот каталог устанавливаются программные пакеты, в нём создаётся структура подкаталогов, аналогичная структуре корневого каталога и каталога /usr.

/usr/share содержит все файлы, которые предназначены только для чтения и не зависят от архитектуры. Примерами файлов, которые размещаются в этом каталоге, могут служить файлы документации (man, doc) или базы данных (dict, terminfo, zoneinfo). Любая программа или пакет, которые содержат или требуют данных, не подлежащих модификации, должны хранить эти данные в каталоге /usr/share (или /usr/local/share, если пакет установлен локально). В каталоге /usr/share создаются следующие подкаталоги или символические ссылки:

- man – интерактивные руководства;
- misc – различные архитектурно-назависимые данные, для которых не требуется отдельный подкаталог в /usr/share;
- dict – словари (факультативно), обычно здесь находится только файл words для английского языка, который используется утилитой look и различными программами проверки правописания; списки слов для других языков могут быть

добавлены использованием английского названия соответствующего языка, например, /usr/share/dict/french, /usr/share/dict/danish и т. д.;
- doc – различная документация (факультативно);
- games – файлы статических данных для /usr/games (факультативно);
- info – основной каталог для системы GNU Info (факультативно);
- locale – локальная информация (факультативно);
- nls – каталоги сообщений для поддержки языков (факультативно);
- sgml – данные для SGML и XML (факультативно);
- terminfo – каталог базы данных для terminfo (факультативно);
- tmac – макросы для troff (факультативно);
- zoneinfo – конфигурационные файлы и информация о временной зоне (факультативно).

Данные игровых программ, сохраняемые в /usr/share/games, должны быть статическими. Любые модифицируемые файлы, такие как файлы с протоколами и результатами игр, должны размещаться в каталоге /var/games.

Как известно, страницы интерактивного руководства man традиционно разбиты на секции. Для каждой секции создаётся отдельный каталог с именем <mandir>/<locale>/man*N*/<arch>, где <arch> – указание на архитектуру (например, i386), а строка <locale> определяет язык, страну и кодировку и имеет следующий формат: <language>[_<territory>][.<character-set>][.<version>].

Подробнее о локализации см. 2.2.7.1. Локализация в ОС Linux (стр. 107).

Каталог **/var** содержит файлы с изменяющимися данными: каталоги и файлы очередей, данные об администрировании, временные файлы. Некоторые части каталоговой структуры /var не являются разделяемыми между разными системами. К ним относятся /var/log, /var/lock и /var/run. Другие части могут быть разделяемыми, например /var/mail, /var/cache/man, /var/cache/fonts и /var/spool/news. Структура каталогов /var определяется в стандарте FHS с той целью, чтобы сделать возможным монтирование каталога /usr в режиме только для чтения. Всё, что записывается на диск в процессе выполнения системных операций (в противоположность процессам установки и поддержки программ), должно размещаться в каталоге /var. Несколько подкаталогов «зарезервированы» – они не должны использоваться произвольным образом, поскольку это противоречит сложившейся практике: /var/backups, /var/cron, /var/msgs, /var/preserve. Приложения в общем случае не должны добавлять каталоги непосредственно в /var.

Такие каталоги должны создаваться в соответствующих подкаталогах. Каталог **/var/cache** предназначен для кэширования данных приложениями. В отличие от /var/spool, кэшированные файлы могут быть удалены без потери данных. Но эти данные должны сохраняться между сеансами работы приложения и при перезагрузках системы. Приложение должно всегда иметь возможность продолжить работу, даже после удаления этих файлов администратором (например, при нехватке дискового пространства). Существование отдельного каталога для кэшируемых данных позволяет системным администраторам устанавливать для этого каталога правила использования и резервного копирования, отличающиеся от правил, устанавливаемых для других каталогов в /var.

Обычно в этом каталоге создаются подкаталоги fonts (локально сгенерированные шрифты), man (локально отформатированные страницы руководства), www (кэш дан-

ных для WWW-прокси), <package> (кэшируемые данные пакета <package>). /var/cache/man предусмотрен для сайтов, в которых файловая система /usr монтируется только на чтение, но в них допускается создание страниц руководства, отформатированных локально. Сайты, в которых /usr монтируется с правом записи (например, когда у системы всего один пользователь), могут не создавать каталога /var/cache/man, а использовать вместо него каталоги cat<section> непосредственно в /usr/share/man.

Файлы блокирования устройств и других ресурсов, используемые многими приложениями, такие как файлы блокирования последовательных портов, должны храниться в каталоге /var/lock. Названия этих файлов должны формироваться в соответствии с соглашением, согласно которому используется префикс «LCK.», за которым следует базовое имя устройства. Файлы блокирования в /var/lock должны быть всем доступны для чтения. В некоторых версиях Linux директория /var/lock может содержать поддиректории, например /var/lock/subsys, внутри которой создаются файлы блокировок уже без префикса «LCK.».

Каталог **/var/log** содержит разнообразные файлы протоколов: lastlog (запись о последнем входе в систему каждого пользователя); messages (системные сообщения от syslogd); wtmp (записи о всех входах и выходах пользователей в систему).

Область спулинга для почты должна размещаться в каталоге /var/mail, а имена файлов с сообщениями должны иметь вид <username>. Файлы почтовых ящиков в этих каталогах должны храниться в формате стандартных почтовых ящиков UNIX (формат UNIX mailbox). В случае использования формата Maildir почтовая директория создаётся в домашней директории соответствующего получателя почты.

Переменные данные для пакетов, установленных в /opt, должны размещаться в /var/opt/<package>, где <package> – название структуры каталогов в /opt, в которой хранятся статические данные дополнительного пакета ПО, исключая те случаи, когда размещение явно указано в каком-либо файле из /etc. На внутреннюю структуру каталога /var/opt/<package> никаких ограничений не накладывается.

Каталог **/var/run** содержит данные, описывающие состояние системы с момента её загрузки. Программы могут иметь подкаталоги в каталоге /var/run, тем более если они используют более одного файла времени выполнения. В этом каталоге должны быть, в частности, размещены файлы с идентификаторами запущенных процессов (PID). Попробуйте запустить команду

```
$ ls -l /var/run
```

Соглашение об именах этих файлов следующее: <*имя_программы*>.pid. Содержимое PID-файла представляет собой идентификатор процесса в коде ASCII, записанный в десятичной нотации, за которым следует символ конца строки. Например, если crond запущен как процесс с номером 1789, /var/run/crond.pid будет содержать пять символов: один, семь, восемь, девять и символ новой строки. Проверьте у себя командой

```
$ cat /var/run/crond.pid
1789
$ od -tx1 /var/run/crond.pid
0000000 31 37 38 39 0a
0000005
```

В **/var/run** расположен также файл utmp, в котором хранится информация о том, кто в данный момент использует систему. Непривилегированные пользователи должны быть лишены права записи в каталог /var/run.

Каталог **/var/spool** содержит данные, которые ожидают какой-то последующей обработки: подкаталоги lpd (спулинг для принтера), mqueue (очередь исходящей почты), news (очередь новостей), uucp (очередь для UUCP) и т. п.

Каталог **/var/tmp** используется программами, которым требуются временные файлы или каталоги для хранения данных, сохраняемых между перезагрузками системы.

Требования FHS к ОС Linux

Отдельный раздел стандарта FHS содержит требования и рекомендации, которые относятся только к Linux:

- В Linux-системах для ядра рекомендуется использовать названия vmlinux или vmlinuz (для ядер, сжатых алгоритмом gzip), которые используются в последних версиях исходных кодов ядра Linux.
- Файл setserial должен размещаться в каталоге /bin.
- Все устройства и специальные файлы в /dev должны соответствовать документу Linux Allocated Devices, который поставляется в составе исходных кодов ядра.
- Файл grub.conf (или lilo.conf) должен размещаться в каталоге /etc. Вместо файла может быть символическая ссылка.
- Поскольку файловая система proc является фактически стандартным для Linux методом обработки информации о системе и процессах, настоятельно рекомендуется использовать proc для хранения и получения информации о процессах, а также информации о ядре и памяти. Например, выполните команду

```
$ cat /proc/meminfo
MemTotal:       16466044 kB
MemFree:        10412716 kB
Buffers:          137124 kB
Cached:          2016804 kB
SwapCached:            0 kB
Active:          4214824 kB
Inactive:        1267576 kB
Active(anon):    3361564 kB
Inactive(anon):    27452 kB
Active(file):     853260 kB
Inactive(file):  1240124 kB
Unevictable:           0 kB
Mlocked:               0 kB

SwapTotal:      16014576 kB
SwapFree:       16014576 kB
Dirty:                 0 kB
Writeback:             0 kB
AnonPages:       3328500 kB
Mapped:           256696 kB
Shmem:             60556 kB
Slab:             138664 kB
SReclaimable:     107696 kB
SUnreclaim:        30968 kB
KernelStack:        3344 kB
PageTables:        42336 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:    24247596 kB
Committed_AS:    5635912 kB
VmallocTotal: 34359738367 kB
VmallocUsed:      365784 kB
VmallocChunk: 34359352316 kB
HardwareCorrupted:     0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
DirectMap4k:      126912 kB
DirectMap2M:     7211008 kB
DirectMap1G:     9437184 kB
```

- В Linux-системах следующие дополнительные файлы размещаются в /sbin – команды для управления файловой системой ext2fs (ext3,4) (badblocks, dumpe2fs, e2fsck, mke2fs, mklost+found, tune2fs); программа установки загрузчика системы grub (или lilo); неизменяемые исполняемые файлы ldconfig, sln, ssync; программы ctrlaltdel, kbdrate.

5.3.5. Пользователи и разграничение доступа

После того как операционная система научилась «одновременно» выполнять несколько программ, а мы ввели понятие файла, логично задать вопрос: а почему с операционной системой не могут одновременно взаимодействовать два или большее число пользователей? Например, первый с первой программой, а второй со второй и т. д. Для этого не обязательно иметь две клавиатуры или две мышки, взаимодействовать можно удалённо. (Подробнее см. главу 6 «*Объединение компьютеров в сети*») Хотим мы или нет, но нескольких пользователей надо как-то разделять и контролировать. В случае даже если работа ими ведётся последовательно, а не одновременно. Представьте ситуацию, что они оба случайно захотят сделать одно и то же, допустим создать один и тот же файл (с одинаковым именем), например «стихи.txt». (Что такое файл, см. в разделе 5.2.2 «*Файл*».) В этом случае они будут либо редактировать один и тот же файл, либо система должна иметь разные файлы пользователей и хранить их по-разному. В итоге исторически было решено, что имена у файлов должны быть уникальными, а пользователи должны иметь свои уникальные идентификаторы uid (от user id).

Операционных систем много, но исторически так сложилось, что именно «*операционная система UNIX широко используется во всех областях, в том числе и в организациях, где безопасности информации уделяется особое внимание*» [23, стр. 288]. В то время как при разработке системы более 40 лет назад ставились другие задачи и данная операционная система изначально не предназначалась для защиты информации, её базовые принципы разграничения доступа до сих пор остаются неизменными, дополняются и улучшаются. Рассмотрим их и мы.

При изучении существовавших проблем и принципов, заложенных более четырёх десятилетий назад, следует понимать, что существующего на сегодняшний день и привычного читателям инструментария могло не существовать! (Как и самих читателей.) Поэтому, читая текст ниже, делайте корректировку на время. Оценивайте адекватно решения, которые появятся у вас в голове в процессе чтения текста ниже.

Для простоты изложения назовём файлы пользователей объектами [264], а пользователей – субъектами [265]. Фактически этого уже достаточно, чтобы появилась наиболее простая и понятная, если не сказать первая приходящая на ум, дискреционная политика безопасности (Дискреционная ПБ; Discretionary Access Control; DAC), то, что сегодня реализуется практически в любой ОС, являясь её частью.

5.3.5.1. Дискреционная политика безопасности в ОС Linux

Рассмотрим её реализацию. Следуя теории, для осуществления указанной политики должны быть:

1. введены понятия объекта, субъекта, доступа;
2. все объекты и субъекты однозначно идентифицированы;
3. задана матрица доступов;
4. доступ субъектов к объектам осуществляется на основании матрицы досту-

[264] В общем случае под объектом понимается пассивная сущность.

[265] В общем случае под субъектом понимается активная сущность. Активной сущностью может быть и программа, запущенная пользователем и унаследовавшая его права.

па.

Указанные требования могут незначительно расширяться и по-разному дополняться, в зависимости от автора, перевода, реализации, но все эти небольшие изменения можно считать незначительными и уточняющими основные позиции.

Перейдём от теории к практике. Первое и второе требования: сопоставить субъекты и объекты с реальными файлами и процессами, идентифицировать те и другие – удовлетворяются просто. Например, для пользователей вводится UID (User ID), для процессов – PID (Process ID), а для файлов – имя файла в файловой системе. Однако третье и четвёртое требования по вопросам реализации матрицы доступов и фактического разграничения остаются открытыми и не всегда понятными.

Попытаемся ввести дискреционную ПБ в том виде, как она была описана нами выше, и оценим её недостатки.

Допустим, что в системе существуют объекты и субъекты. Обозначим их через *O* и *S* соответственно.

Тогда под объектами могут быть представлены файлы, а под субъектами – пользователи, а точнее процессы, выполняющиеся с их правами.

Изобразим примерную матрицу доступа (см. табл. 5.2).

Таблица 5.2. Пример матрицы доступа ("+" – есть доступ, "−" – нет доступа)

	Объект 1	Объект 2	Объект 3	...	Объект *m*
Субъект 1	+	−	−		−
Субъект 2	−	−	+		+
Субъект 3	−	+	−		−
...				...	
Субъект *n*	+	−	−		−

Если через «+» обозначить наличие доступа, а через «−» его отсутствие, то можно заметить, что при преобразовании этих обозначений в «1» и «0» можно получать в реальных системах сильно разряженную матрицу.

Если в среднестатистической системе, в которой число файлов измеряется тысячами (если не десятками тысяч), потребуется добавить нового пользователя, то есть создать новый субъект в нашей терминологии, то для него в матрице доступа потребуется добавить соответствующую строку, а при создании этим (или другим, уже существующим) пользователем новых файлов придётся добавлять соответствующие столбцы. А при удалении файлов или учётных записей пользователей – удалять соответствующие им столбцы и строки.

Указанная схема насколько хороша в плане наглядности, настолько и плоха в плане практичности. Учитывая, что доступ субъектов к объектам может быть в разных режимах (например, только для чтения, для изменения и др.), обозначения «+» и «−» придётся заменить большим количеством знаков либо хранить несколько матриц доступов. Например, одна матрица будет отвечать за операцию «чтения» субъектами объектов, а другая – за операцию «записи».

Указанная схема хороша в теории, но неудобна на практике, поэтому её несколько преобразовали, «выделив общее», что позволило сократить место для хранения матрицы доступа. После выделения «общего списка субъектов» появилась возможность сделать хранение данных из матрицы доступа распределённым.

Например, в UNIX-системах общим может быть то, что по отношению к конкретному файлу, независимо от его типа (директория – это тоже файл типа «директория»), все пользователи делятся на три категории:

1. user (владелец файла);
2. group (члены группы пользователя);
3. other (прочие пользователи).

Установить права доступа для этих категорий в одном месте проще, чем работать с большой по размерам и сильно разряженной матрицей выше.
Для указанных категорий пользователей (субъектов) исторически были выделены следующие возможные права доступа к файлам (объектам): **r**ead (r), **w**rite (w), e**x**ecute(x), чтение, запись и исполнение (выполнение), соответственно.

Таблица 5.3. Фрагмент реальной матрицы доступа ("+" – есть доступ, "−" – нет доступа; заполнение в данном случае произвольное и лишено смысла)

		read	**w**rite	e**x**ecute
all (все)	**u**ser	+	+	−
	group	+	−	−
	other	−	−	−

9 прав доступа (3 * 3) таблице 5.3 для одного объекта (не считая не вошедших в таблицу дополнительных и расширенных атрибутов, отвечающих за «модификацию условий» применения основных прав доступа.) намного лучше сильно разрежённой матрицы таблицы 5.2.

Так как одним из главных принципов UNIX-систем является доступ ко всем объектам системы через файловую систему («*...всё, что существует в POSIX [266]-системе в статическом виде, являет собою файлы. Собственно, использование файловой системы как универсального интерфейса доступа ко всему, чему угодно, – от устройств, физически подсоединённых к машине, до процессов, в системе протекающих – и есть один их критериев UNIX-подобия ОС (и её соответствия стандартам POSIX)*», стр. 143 [4]), то указанные права могут быть распространены не только на обычные файлы, но и на псевдофайлы [267] [15] (символические ссылки, сокеты, специальные блочные устройства, специальные символьные устройства, именованные каналы) и директории. Естественно, для последних смысл указанных прав несколько изменяется. Подробнее см. ниже или в [2, стр. 18].

[266] POSIX® (аббревиатура от англ. **P**ortable **O**perating **S**ystem **I**nterface for UNI**X** – Переносимый интерфейс операционных систем UNIX) – набор стандартов, описывающих интерфейсы между операционной системой и прикладной программой. Стандарт создан для обеспечения совместимости различных UNIX-подобных операционных систем и переносимости прикладных программ на уровне исходного кода, но может быть использован и для не-Unix-систем.

[267] Термин «псевдофайл» практически не используется в литературе, но в данном случае он просто необходим в том виде, как он сформулирован в [15]. Желательность его использования связана с тем, что пользователи, малознакомые с «UNIX-миром», могут впасть в глубокую задумчивость, узнав, что привычные для них понятия, такие как термин «файл», могут иметь совсем иной, чем они привыкли в него вкладывать, смысл. Так, например, директория с точки зрения UNIX/Linux-систем является тоже файлом с соответствующим атрибутом «директория» и может содержать ссылки на другие файлы. Для того чтобы избежать подобных неловкостей, при восприятии материала под псевдофайлом в ОС UNIX/Linux будем понимать любой файл, не являющийся обычным файлом или директорией.

Указанные права хранятся в элементах inode (индексных дескрипторах, i-узлах) файловой системы [6, стр. 297–299; 11] подробнее о правах и содержимом i-node написано ниже. (См. разделы 5.3.5.1.2 «*Стандартные и расширенные атрибуты файлов*» и 5.3.7.4 «*ext2, ext3, ext4*».)

5.3.5.1.1. Проблема хранения учётных записей (и/или паролей пользователей)

На практике указанных выше девяти атрибутов оказывается недостаточно, так как в рамках классической дискреционной модели один субъект не может передать другому субъекту прав больше, чем имеет сам. При этом задача хранения учётных записей пользователей (и паролей) становится нерешаемой, так как не ясно, кому должен принадлежать файл, в котором будет находиться указанная информация, и какие у данного файла должны быть выставлены права доступа. Выход из данной ситуации может быть найден либо созданием третьего процесса, с правами, большими, чем у всех пользователей системы, но тогда этот процесс должен быть запущен постоянно, либо должен быть создан механизм повышения привилегий.

По жизни UNIX-системы используют второй подход. А именно для повышения привилегий файлам выставляются дополнительные атрибуты. Например, идентификатор SetUID может использоваться для того, чтобы указать системе на необходимость замены, при проведении проверок согласно матрице доступа, одного параметра другим.

Подобная система позволяет уйти от принципа «*исполняемый код, запущенный на выполнение от имени пользователя, наследует права доступа этого пользователя*».

Фактически это означает увеличение числа строк (списка объектов) матрицы доступа пропорционально количеству субъектов в системе. При этом полученные записи также характеризуются большой разряженностью.

Предполагаем, что если вы честно дочитали от начала параграфа до этой строчки, то наверняка хотите уже отойти от теории и сказать: Давайте ближе к жизни! Как же всё-таки организовано хранение учётных записей пользователей (и паролей) в реальных системах? Или как правильно организовать подобную схему, если вы планируете (создаёте) многопользовательскую операционную систему с нуля?

Один из вариантов – это создать доверенную программу, осуществляющую взаимодействие с системой аутентификации и пользователями. При этом указанная программа будет единственной обращающейся к файлу, хранящему учётные записи. Доступ к этому файлу на чтение или запись пользователи иметь не будут. Доверенная программа (назовём её «программа контроля») будет проверять входящих в систему пользователей, а также менять пароли пользователям по их требованию.

Тогда логично спросить: а как пользователи будут взаимодействовать с этой программой? Если внешне проблем взаимодействия нет – для пользователей существуют и графические, и текстовые интерфейсы, то с внутренней стороны получается, что необходимо создать пользовательские приложения, которые будут выполняться в системе с правами пользователей и будут каким-то образом взаимодействовать с программой контроля.

В этом случае предположим, что межпроцессное взаимодействие может быть организовано через обычные файлы, именованные каналы, локальные или интернет-сокеты, при этом следует заметить, что именованные каналы и локальные сокеты, по

сути, являются псевдофайлами с такими же правами доступа, и в случае их использования возникает не только проблема монопольного использования данных ресурсов, но и протокола обмена через эти ресурсы. Так как возможность взаимодействия с «программой контроля» должна быть у всех пользователей, то, соответственно, все пользователи будут иметь доступ на чтение и запись в указанный «файл общения с программой». Для того чтобы один пользователь не мог поменять учётные записи другого (в том числе и пароль), системе придётся, перед внесением изменений, дополнительно удостовериться, что задания на выполнение тех или иных действий приходят санкционированно, то есть именно от владельца учётной записи. Данная проверка потребует аутентификации, при этом следует заметить, что аутентификационные данные, передаваемые между «программой контроля» и пользовательской частью, не должны оказаться доступными всем пользователям, что потребует усложнения алгоритма обмена информацией или алгоритма аутентификации с целью защиты данных, передаваемых в процессе аутентификации, от прослушивания и модификации. Использование сетевых сокетов (Unix-сокетов) позволит защититься от прослушивания, но заставит систему применять сетевые протоколы для осуществления тех действий, где наличие стека сетевых протоколов не требуется.

Сегодня сложно представить сетевую ОС без loopback-интерфейса, но, наверное в далёкие 70-е разработчики могли предположить наличие многопользовательской ОС без сетевых возможностей, поэтому вносить зависимость функционирования ОС от состояния её сетевых интерфейсов и прочего не стали. Обратите внимание, что RFC 791, описывающий привычный нам сегодня протокол IP, «по которому работает» большая часть интернета, датирован сентябрём 1981 года.

В целом можно заключить, что подходы к использованию стека сетевых протоколов, созданию вышеуказанных сложных схем решения поставленной ранее проблемы, как и наличие постоянно находящегося в оперативной памяти компьютера процесса, ожидающего обращений (оперативная память в начале 70-х стоила недёшево!), содержат недостатки и не являются оправданными для использования. Поставленная в начале абзаца проблема хранения учётных записей была решена путём создания в теории тогда, а точнее использования сейчас на практике дополнительных атрибутов у файлов, таких как SUID, SGID и др. [2, стр. 11–24; 8, стр. 88–97; 9, стр. 85–95].

5.3.5.1.2. Стандартные и расширенные атрибуты файлов

В основе практической реализации рассмотренной выше модели дискреционного разграничения доступа лежит проверка идентификаторов пользователей и групп, а также прав доступа (атрибутов) ядром [268] ОС. Будем предполагать, что ядро ОС реализовано и работает корректно. Информация «о доступах», по которой ядром осуществляется проверка прав и принятие решения, не имеет централизованного хранения. Данные распределены между индексными дескрипторами (i-node) всех файлов в системе, в которых содержится не только физическое расположение частей файлов на диске, но и данные двух категорий пользователей, к которым относятся права, хранимые в i-node: имя так называемого владельца файла (точнее UID, а соответствие имени и UID хранится в файле /etc/passwd) и GID группы, и сами права доступа (они же: атрибуты, «права», «доступы», биты доступа, биты прав, флаги доступа). Обычно какое-то право

[268] Определение понятия «ядро ОС» см. на стр. 386.

или есть, или его нет, поэтому одного бита достаточно для хранения информации об одном праве доступа. С целью экономии места физическое хранение разных прав доступа осуществляется совместно, группами битов, обычно кратными целому числу байтов. (Организуется структура.) Рядом с правами доступа в i-node хранится и тип файла. Фактически для хранения прав и типа файла отводятся два байта (16 бит). Рассмотрим, какому полю структуры сколько отводится места, подробнее см. табл. 5.4 или «man 2 stat».

Таблица 5.4. Описание части битов, хранимых в i-node

биты	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
описание	тип файла (-, d, l, s, b, c ,p)				особые свойства исполняемых файлов			права доступа (владелец)			права доступа (группа)			права доступа (остальные)		

Биты 8–16 – это те 9 прав, о которых шла речь выше, они и определяют права доступа (для выбранных нами трёх категорий пользователей) по три бита на каждую. Биты 8–10 задают права пользователя, биты 11–13– права группы пользователя, биты 14–16 определяют права всех остальных пользователей (то есть всех пользователей, за исключением владельца файла и группы файла).

Если соответствующий какому-либо праву бит имеет значение «1», то право предоставляется, а если он равен «0», то право не предоставляется. Точь-в-точь как субъекты получают доступ к объектам в табл. 5.2. В символьной форме записи прав для отображения пользователю единица заменяется соответствующим символом (r, w или x, от **r**ead, **w**rite, e**x**ecute см. выше), а 0 представляется прочерком (-).

Право на чтение «r» файла означает, что пользователь (или процесс, запущенный пользователем) может считывать содержимое файла. Право на запись «w» означает возможность записи в этот файл. Право на выполнение «x» означает, что файл может быть запущен как исполняемая программа. Конечно, если в действительности файл не является программой (или скриптом), то запустить этот файл на выполнение и получить полезный результат вряд ли удастся, но, с другой стороны, даже если файл действительно является программой, а право на выполнение для него не установлено, то он тоже «не запустится».

Красиво и понятно описывает права В. Костромин [12]:

По отношению к каталогам трактовка понятия «право на выполнение» несколько изменяется (естественно, ведь «выполнение» каталога бессмысленно). Оно в данном случае означает право переходить в этот каталог. Если вы как владелец хотите дать доступ другим пользователям на просмотр какого-то файла в своём каталоге, вы должны дать им право доступа в каталог, то есть дать им «право на выполнение каталога». Более того, надо дать пользователю право на выполнение для всех каталогов, стоящих в дереве выше данного каталога. Поэтому, в принципе, для всех каталогов по умолчанию устанавливается право на выполнение как для владельца и группы, так и для всех остальных пользователей. Если предоставить право на выполнение, но не предоставлять право на чтение каталога, то пользователь сможет войти в каталог и обратиться к файлу, находящемуся в этом каталоге, но только в том случае, если он знает точное имя этого файла. При отсутствии права на чтение каталога просмотр содержимого невозможен. Использование данного факта как элемента защиты возможно, но не является надёжным.

Алгоритм проверки прав пользователя при обращении к файлу можно описать следующим образом. Система вначале проверяет, совпадает ли имя пользователя с именем владельца файла. Если эти имена совпадают, то есть владелец обращается к своему файлу, то проверяется, имеет ли владелец соответствующее право доступа: на чтение, на запись или на выполнение (не удивляйтесь, суперпользователь может лишить некоторых прав и владельца файла). Если право такое есть, то соответствующая операция разрешается. Если же нужного права владелец не имеет, то проверка прав, предоставляемых через группу или через группу атрибутов доступа для остальных пользователей, уже даже не проверяется, а пользователю выдаётся сообщение о невозможности выполнения затребованного действия.

Если имя пользователя, обращающегося к файлу, не совпадает с именем владельца, то система проверяет, принадлежит ли владелец к группе, которая сопоставлена данному файлу (далее будем просто называть её группой файла). Если принадлежит, то для определения возможности доступа к файлу используются атрибуты, относящиеся к группе, а на атрибуты для «владельца» и «всех остальных пользователей» внимание не обращается. Если же пользователь не является владельцем файла и не входит в группу файла, то его права определяются атрибутами для остальных пользователей. Таким образом, третья группа атрибутов, определяющих права доступа к файлу, относится ко всем пользователям, кроме владельца файла и пользователей, входящих в группу файла.

Фрагмент Таблицы 5.4. Описание части битов, хранимых в i-node

биты	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
описание					особые свойства исполняемых файлов											

Три бита (5–7, табл. 5.4) интерпретируются следующим образом. Первый из этих атрибутов (бит 5, табл. 5.4) – так называемый «бит смены идентификатора пользователя (владельца)» (Set User ID, SetUID, SUID, suid). Смысл этого бита состоит в следующем: обычно, когда пользователь запускает некоторую программу на выполнение, эта программа получает те же права доступа к файлам и каталогам, которые имеет пользователь, запустивший программу. Если же установлен «бит смены идентификатора пользователя», то программа получит права доступа к файлам и каталогам, которые имеет владелец файла программы.

Это позволяет решать некоторые задачи, которые иначе было бы трудно выполнить. Самый характерный пример, рассмотренный выше, правда, без упоминания деталей – команда смены пароля passwd. Вся информация о паролях пользователей хранится в файле /etc/shadow (реже в /etc/passwd), владельцем которого является суперпользователь. Поэтому программы, запущенные обычными пользователями, в том числе команда passwd, не могут производить запись в этот файл. А значит, пользователь не может менять свой собственный пароль. Но для файла /usr/bin/passwd (см. `ls -l /usr/bin/passwd`) установлен «бит смены идентификатора владельца», каковым является суперпользователь. Следовательно, программа смены пароля запускается с правами суперпользователя (так как он владелец файла и на файле стоит атрибут SetUID) и получает право записи в файл /etc/shadow (а то, чтобы пользователь мог изменить только «свою» строку в этом файле, обеспечивается уже средствами самой про-

граммы). Девид Тейнсли в качестве примера в [2, стр. 19] пишет: «*Я отвечаю за администрирование нескольких больших баз данных. Чтобы выполнить операцию по их резервированию, требуется специальный профильный файл администратора. Я создал несколько сценариев и установил для них бит SGID, благодаря чему пользователи, которым разрешён запуск этих сценариев, не обязаны регистрироваться в качестве администраторов баз данных. А это, в свою очередь, уменьшает риск повреждения информации на сервере. При запуске указанных сценариев пользователи получают разрешение на выполнение операций по выгрузке базы данных, хотя обычно такое право предоставляется только административному персоналу. После завершения сценариев восстанавливаются изначальные права пользователей*».

Следующий атрибут (бит 6, табл. 5.4) – «бит смены идентификатора группы пользователя» (Set Group ID, SetGID, SGID, sgid) – аналогичен атрибуту SUID, но относится к группе владельца файла, а не к самому владельцу.

И наконец, третий атрибут (бит 7, табл. 5.4) – это «бит сохранения задачи», или «sticky bit» (дословно – «липкий бит»). Этот бит указывает системе, что после завершения программы надо сохранить её код в оперативной памяти. Этот атрибут был необходим на старых моделях компьютеров с малым быстродействием жёстких дисков и малым объёмом оперативной памяти. В современных системах с быстрыми дисками и объёмами памяти, исчисляемыми гигабайтами, данный бит потерял какой-либо практический смысл и часто просто игнорируется.

По отношению к директориям указанный бит (бит 7, табл. 5.4) меняет другие свойства. Например, его можно использовать в отношении общих разделяемых каталогов (/tmp и др.). Действует он следующим образом. Если предоставить всем полные права на доступ к какому-либо каталогу, то любой пользователь сможет записывать и удалять любые файлы в этом каталоге, вне зависимости от их владельцев. Если же установить «sticky bit», то пользователь сможет удалять только те файлы, владельцем которых он является. Таким образом, программы, запущенные от одного пользователя, могут создавать свои временные файлы в директории /tmp без боязни того, что эти файлы будут стёрты другими пользователями или их программами.

Общие файлы в таких директориях могут использоваться для обеспечения информационных потоков между различными процессами, поэтому данный факт должен учитываться администратором при реализации разграничения доступа в конкретных условиях.

Дополнительно из man chmod можно узнать, что «sticky bit» не описывается в POSIX, а из [22] – что впервые «sticky bit» появился в пятой редакции UNIX в 1974 году для использования в исполняемых файлах.

5.3.5.1.3. Изменение атрибутов (прав доступа)

Для изменения стандартных атрибутов и прав доступа к файлу используется команда chmod со следующим синтаксисом запуска.

В режиме относительного задания атрибутов (относительно тех атрибутов, что уже есть):

```
chmod [опции] режим[,режим]... файл ...
```

В режиме абсолютного задания атрибутов:

```
chmod [опции] OCTAL-MODE файл ...
В режиме копирования прав:
chmod [опции] --reference=filename1 filename2
```

`filename1` – имя файла с которого копируютсяправа;
`filename2` – файл которому ставятся права.

Замечание. Подробную информацию о способе запуска и описание команды можно вывести на экран, набрав в командной строке «`man chmod`» (выход нажатием «q»).

В первом варианте вы должны явно указать, кому какое право даёте или кого этого права лишаете. Каждая запись комбинации прав доступа «режим» имеет вид [ugoa] [+–=] [rwxXstugo]. Комбинация букв [ugoa] задаёт тех, для кого меняются права доступа: владельца (user), группы владельца (group), остальных (other) или всех остальных (all) соответственно. Символ «+» означает добавление права, «–» – удаление права, а «=» – установление права вместо имеющегося.

Например, добавить право «на чтение» группе для файла file1:
```
chmod g+r file1
```
или установить для владельца файла права на чтение и запись, сняв право исполнения, если оное было установлено:
```
chmod u=rw file1
```
Символы (буквы) 'rwxXstugo', те, что идут до знаков «+», «−» или «=», выбирают новые права доступа для категорий пользователей, заданных одной или несколькими буквами из ‘ugoa’, идущими после этих знаков: чтение (r); запись (w); выполнение (или доступ к каталогу) (x); выполнение, если файл является каталогом или уже имеет право на выполнение для какого-нибудь пользователя (X); setuid- или setgid-биты (s); sticky-бит (t); установка для остальных таких же прав доступа, которые имеет пользователь, владеющий этим файлом (u); установка для остальных таких же прав доступа, которые имеет группа файла (g); установка для остальных таких же прав доступа, которые имеют остальные пользователи (не входящие в группу файла) (o).

Например, «`chmod g-s file`» снимает бит set-group-ID (sgid), «`chmod ug+s file`» устанавливает биты suid и sgid, в то время как «`chmod o+s file`» ничего не делает.

Вышеуказанный режим задания прав называется относительным, так как выставляет права относительно тех, которые у файла имелись до запуска команды chmod.

Дополнительные примеры запуска команды chmod смотрите в «Кратком справочнике команд» на стр. 541.

При абсолютном формате назначения права биты атрибутов и права доступа задаются «в восьмеричном виде» одной до четырёх восьмеричных цифр, получаемых суммированием битов, соответствующих значениям 4, 2 и 1 (по степеням двойки: $2^2 = 4$, $2^1 = 2$, $2^0 = 1$). 4 соответствует праву на чтение или биту SUID, 2 – праву на запись или биту SGID, а 1 – праву на выполнение или биту «sticky bit». Первая цифра задаёт атрибуты, вторая – права доступа владельца, третья – права доступа группы владельца, четвёртая – права доступа остальных пользователей. Если одна или несколько цифр опущены, то считается, что это первые цифры, и они полагаются равными нулю. Запись «`chmod 0755 file2`» идентична записи «`chmod 755 file2`».

Существует и третий вариант задания прав, когда атрибуты устанавливаются такими же, как и у файла, указанного в качестве эталона. Например, установить файлу `file1` такие же права, как и у файла `/etc/passwd`:
```
chmod --reference=/etc/passwd file1
```

Также в командной строке могут указываться некоторые дополнительные параметры, среди которых отметим параметр «-R», при задании которого команда chmod будет выполнена рекурсивно для всех указанных файлов и подкаталогов.

Дополнительную информацию по команде chmod можно получить командой

```
$ info chmod
```

Как было сказано в начале, несмотря на то что для проведения работы используется ОС Linux, многие теоретические вопросы, в том числе и некоторые команды, хотелось бы рассмотреть шире, захватив в рассмотрение ещё и ОС FreeBSD.

Замечание 1. К сожалению, современная бумажная литература не отличается правдивостью, так, в [20, стр. 196] можно НЕПРАВИЛЬНО прочитать: «*Теоретически члены группы могут иметь меньше прав, чем остальные пользователи. Например, режим доступа rw----r-- (604) формально означает, что остальные пользователи имеют право чтения, а члены группы – нет. Впрочем, на практике этого ограничения не существует, поскольку категория "остальные пользователи" охватывает и членов группы тоже, то есть указанный выше режим доступа эквивалентен режиму rw-r--r-- (644)*». Что это не так легко проверить на практике, как в Linux:

```
$ uname -a
Linux linux 2.6.34.7-61.fc13.x86_64 #1 SMP Tue Oct 19 04:06:30 UTC 2010
x86_64 x86_64 x86_64 GNU/Linux
$ id
uid=500(pasha) gid=501(aaa) группы=500(pasha),501(aaa) ...
$ ls -l file*
-rw----r--. 1 www aaa 5 Окт 26 22:10 file1
-rw-r-----. 1 www aaa 5 Окт 26 22:12 file2
$ cat file1
cat: file1: Отказано в доступе
$ cat file2
test
```

Так и во FreeBSD.

```
$ uname -a
FreeBSD sys 6.0-RELEASE FreeBSD 6.0-RELEASE #0: Thu Nov  3 09:36:13 UTC 2005
root@x64.samsco.home:/usr/obj/usr/src/sys/GENERIC  i386
$ id
uid=1002(pasha) gid=0(wheel) groups=0(wheel)
$ ls -l file*
-rw-r-----  1 www  wheel  5 Oct 26 22:43 file1
-rw----r--  1 www  wheel  5 Oct 26 22:43 file2
$ cat file1
test
$ cat file2
cat: file2: Permission denied
```

Замечание 2. Используйте атрибут «`-R`» осторожно. Например, если вы захотите снять со всех файлов (в том числе и в поддиректориях) директории `dir1` атрибуты запуска, то не выполняйте команду

```
$ chmod -R a-x dir1
```

Скорее всего, вы получите ошибку при попытке доступа к файлам директории `dir1`, так как к моменту доступа к ним директория `dir1` уже не будет иметь атрибута «x», разрешающего поиск файлов в ней. Команда `chmod` для файлов и директорий одна, а наличие и отсутствие одних и тех же атрибутов даёт разные результаты.

В данном случае лучше воспользоваться рецептом, который предлагает Карла Шрёдер [16, стр. 144]:

«*Команда chmod поддерживает операции со списками файлов. Для построения списков можно воспользоваться командой find или метасимволами командного процессора*».

Чтобы снять атрибут «x», но «без изменения разрешений самого каталога воспользуйтесь командой»:

```
$ find /home/guest/dir1 -type f -exec chmod a-x {} \;
```

Команда find найдёт все файлы в директории /home/guest/dir1 (в том числе и в поддиректориях) и передаст их имена команде chmod, которая и снимет атрибут «x» у файлов.

В частных вопросах и ответах по ОС FreeBSD в [17] написано:

Опция -R выполняет команду chmod(1) РЕКУРСИВНО. Будьте осторожны, задавая каталоги или символические ссылки на каталоги в параметрах chmod(1). Если вы хотите изменить права на каталог, на который указывает символическая ссылка, используйте chmod(1) без опций и следуйте символической ссылке с помощью лидирующего слэша (/). Например, если foo является символической ссылкой на каталог bar, а вы хотите изменить права на foo (на самом деле bar), вы должны выполнить команду типа следующей:

```
% chmod 555 foo/
```

Если задан лидирующий слэш, то chmod(1) будет следовать символической ссылке, foo, меняя права на каталог bar.

5.3.5.1.4. Смена владельца (группы)

Смена владельца или группы выполняется от имени суперпользователя командами chown и chgrp. Синтаксис команд следующий.

```
chown [опции]... OWNER[:[GROUP]] файл...
chown [опции]... :GROUP файл...
chown [опции]... --reference=файл_образец файл2...
chgrp [опции]... GROUP файл...
```

, где *файл_образец* – это имя файла с которого берутся владелец и группа.

Команда chown позволяет сменить как владельца, так и группу владельца файла, поэтому команда chgrp является в некотором роде излишней. Обе команды также поддерживают рекурсивную обработку каталогов при указании соответствующего параметра `-R`, аналогично команде chmod. В командах могут использоваться как символические, так и цифровые идентификаторы пользователей и групп. Например:

«`chown root program1`»,
«`chown vasya:mail /var/spool/mail/vasya`»
или «`chgrp 12 /var/spool/mail/vasya`».

Замечание (спасибо техническому редактору журнала «Системный администратор [269]» за совет). Обращаем внимание, что суперпользователь и только он может сменить владельца файла. Многие бумажные издания, в частности [2, стр. 20; 9, стр. 83; 19, стр. 718], а также различные сайты в интернете (наберите в Яндексе: «*суперпользователь или владелец файла может сменить владельца*», чтобы найти их) пишут, что сменить владельца может суперпользователь или владелец файла. Это утверждение не верно, так как если бы это было так, то любой пользователь операционной системы мог бы написать свою программу, а после её выполнить с правами root. Для этого перед запуском программы ему понадобилось бы выполнить всего две команды: «chmod +s program1» и «chown root program1». Со сменой группы дела обстоят примерно аналогично – менять её может только суперпользователь, за тем лишь исключением, когда владелец входит в новую группу. Наиболее хорошо и понятно об этом написано в [21, стр. 196]: «*...Владелец файла может легко сменить все права на доступ файла для самого себя,*

[269] Сайт журнала «Системный администратор»: www.samag.ru.

группы и прочих. Может он и назначить принадлежность файла другой группе, хотя и не любой, а только той, членом которой он сам является. Однако изменить владельца файла (то есть назначить владельцем другого пользователя) он не имеет права. Это – прерогатива исключительно администратора, который располагает полномочиями изменить для файла все атрибуты доступа и принадлежности (как, впрочем, и почти все прочие атрибуты)».

В ОС FreeBSD, в отличие от ОС Linux, в man chown есть замечание по поводу рассуждений выше: *The ownership of a file may only be altered by a super-user for obvious security reasons.*

5.3.5.1.5. Изменение и просмотр расширенных атрибутов

Файловые системы ext2, ext3, ext4, используемые, предпочтительно ОС Linux, помимо стандартных, поддерживают набор расширенных атрибутов [8, стр. 95–96; 10, стр. 43–45], с помощью которых можно управлять режимами работы с файлами. Устанавливать расширенные атрибуты файлов можно с помощью команды chattr, имеющей следующий синтаксис.

```
chattr [-RV ] [ attribute... ] файл...
```

Формат символьного режима задания атрибутов `+-=[acdeijstuADST]`. Оператор «+» обозначает добавление указанных атрибутов к существующим, «-» обозначает их снятие, «=» обозначает установку только этих атрибутов файлам.

Символы ‘acdeijstuADST’ указывают на атрибуты файлов: только добавление (дозапись) к файлу (a), сжатый (c), не архивировать (d), для отображения используемых блоков на диске используются экстенты (e), неизменяемый (i), журналирование данных (j), безопасное удаление (s), запрет слияния в конце файла (t), неудаляемый (u), не обновлять время последнего доступа к файлу atime (A), синхронное обновление каталогов (D), синхронное обновление (S), вершина иерархического дерева (T).

Атрибуты: огромный файл (huge file) (h), ошибка сжатия (E), индексированная директория (I), прямой доступ к сжатым данным (X) и сжатый «грязный файл» (compressed dirty file) (Z) – скорее, необходимы для отладочных целей. Их можно только просматривать с помощью команды lsattr, изменять данные атрибуты с помощью chattr нельзя.

Рассмотрим атрибуты, предположительно которые могут быть наиболее востребованными.

Атрибут «A» позволяет не обновлять у файлов время последнего доступа к ним (atime). С одной стороны, наличие атрибута не позволит администратору узнать, когда к файлу обращались в последний раз, с другой стороны, позволит снизить задержки при дисковых операциях ввода/вывода на «медленных» компьютерах.

Файл с атрибутом «a» можно открыть для записи только в режиме добавления. Учитывая, что только суперпользователь может устанавливать и снимать этот атрибут, он очень удобен для ведения лог-файлов. Например, процесс httpd, выполняющий функции веб-сервера от пользователя www, или любой другой сервер должен вести лог-файл (журнал) обращений к нему, но в случае сбоя в программном обеспечении может возникнуть ситуация, когда анонимные пользователи, обращающиеся к веб-серверу, смогут получить права, с которыми выполняется сервер, и тогда они смогут удалить записи из log-файла и затруднить обнаружения факта атаки на сервер администратором. При наличии установленного атрибута «a» произвести указанную атаку не получится, пока он не будет снят. Данный атрибут напоминает ситуацию, когда в далёкие времена log-файлы сразу печатались принтером на бумажную ленту, так как

это было дешевле, чем запись данных в электронное хранение. А дальше действовала русская поговорка «что написано пером – не вырубишь топором».

Файл с атрибутом «c» автоматически сжимается на диске ядром. Чтение из такого файла возвращает обычные (несжатые) данные. При записи в такой файл данные перед записью на диск сжимаются. На данный момент не поддерживается.

Если изменяется каталог с атрибутом «D», то изменения сразу синхронно записываются на диск; это эквивалентно использованию опции «dirsync» при использовании команды mount.

Файл с атрибутом «i» не может быть изменён: он не может быть удалён или переименован, никакие ссылки к этому файлу не могут быть созданы, и никакие данные не могут быть записаны в этот файл. Только суперпользователь может устанавливать или снимать этот атрибут.

При записи в файл с атрибутом «j» все данные, записываемые в такой файл, записываются в журнале ext3 (ext4), прежде чем они будут записаны непосредственно в файл, если файловая система смонтирована с опциями «data=ordered» или «data=writeback». Если файловая система смонтирована с опцией «data=journalled», то все данные журналируются и этот атрибут не даёт никакого эффекта.

Когда удаляется файл с атрибутом «s», все его блоки заполняются нулями. Установка этого бита может быть важной, если мы хотим затруднить восстановление удалённых данных. Для параноиков отметим, что одноразовое заполнение места на диске, ранее занятого удалённым файлом, нулями – недостаточная мера для по-настоящему безопасного удаления данных. Обычным пользователям следует учитывать, что удаление больших файлов с указанным атрибутом может занять длительное время, напрямую связанное с производительностью вашего диска.

Когда изменяется файл с атрибутом «S», все изменения синхронно записываются на диск; это эквивалентно опции монтирования «sync», примененной ко множеству файлов, либо запуску /bin/sync по отношению к файлу с атрибутом «S» по завершению операции записи в файл.

Когда удаляется файл с атрибутом «u», его содержимое сохраняется. В будущем это позволит пользователю восстановить файл. Сейчас, к сожалению, данная опция не поддерживается.

Остальные опции не рассмотрены в силу их редкого использования. Желающие могут более подробно ознакомиться с командой изменения расширенных атрибутов, запустив man chattr.

Операции изменения атрибутов и просмотра могут быть совмещены, для этого допишите ключ -V к команде chattr [8, стр. 96]. Использование указанной опции позволит вам не только поменять атрибуты, но и увидеть их после установки на файл, как если бы вы запустили команду lsattr.

Замечание. Если обратиться к команде man chattr, а также к литературе, использующей в качестве первоисточника указанную документацию, то можно легко прийти к заблуждению. Например, если в ОС Linux CentOS 6.9 запустить команду

```
$ man chattr
```

то можно прочитать следующее:

BUGS AND LIMITATIONS

The ‘c’, ’s’, and ‘u’ attributes are not honored by the ext2 and ext3 filesystems as implemented in the current mainline Linux kernels. These attributes may be implemented in future versions of the ext2 and ext3 filesystems.

Дословно это можно понять как: атрибуты «c», «s» и «u» в основных версиях ядер ОС Linux не используются и созданы для будущих версий.

А так ли это на самом деле? К счастью, нет. По нашей практике можем сказать, что атрибут «s» поддерживается ядрами Linux не менее восьми лет (когда это случайно было обнаружено). Вывод, не верьте написанному, всё перепроверяйте.

Если вы во время установки атрибутов командой chattr не используете опцию -V либо отдельно хотите просмотреть установленные расширенные атрибуты – используйте команду lsattr. Синтаксис команды следующий:

```
lsattr [ -RVadv ] [ файл ]...
```

Основные параметры команды следующие:

`-R` – рекурсивно выводить список атрибутов каталогов и их содержимого.

`-a` – просматривать все файлы в каталоге, включая файлы, чьи имена начинаются с точки «.».

`-d` – отображать каталоги как обычные файлы вместо просмотра их содержимого.

Замечание. В ряде случаев к расширенным атрибутам относят и контексты SElinux'a, списки контроля доступа (ACL, см. «man acl», «man getfacl», [39]) и способности (см. «man 7 capabilities»), несколько больше о них можно узнать в *Лабораторной работе № 2. Исследуем inode*. см. стр. 548)

5.3.5.1.6. Списки управления доступом (ACL)

Стандартные средства разграничения доступа на основе таблицы Таблицы 5.3 и расширенных атрибутов во многих случаях недостаточны, например, при необходимости разрешить какому-либо отдельному пользователю доступ к некоторому файлу, не открывая этот доступ всем остальным и не включая этого пользователя в группу. В ФС ext2/3/4 каждому файлу может ставиться в соответствие список управления доступом (Access Control List – ACL), который также реализует дискреционный доступ к этому файлу. Кроме того, с каждым каталогом может быть связан дополнительный ACL, который будет определять начальные ACL для объектов, создаваемых внутри этого каталога. В современных системах поддержка ACL включена изначально, ранее же для этого необходимо было включить поддержку в ядре и производить монтирование файловой системы указывая дополнительный параметр «`-o acl`».

Список управления доступа состоит из записей, каждая из которых описывает права доступа отдельного пользователя или группы пользователей в виде комбинации прав на чтение, запись и поиск/выполнение.

Запись ACL содержит метку типа записи, необязательный квалификатор и набор разрешений. Квалификатор обозначает пользователя или группу для записей типа ACL_USER или ACL_GROUP, соответственно. Записи других типов не содержат квалификаторов.

Предусматриваются следующие типы записей:

- ACL_USER_OBJ – запись этого типа определяет права доступа владельца файла;
- ACL_USER – определяет права доступа для пользователя, заданного квалификатором;
- ACL_GROUP_OBJ – определяет права доступа для группы владельца файла.

- ACL_GROUP – определяет права доступа для группы, определенной квалификатором;
- ACL_MASK – запись этого типа определяет максимальный набор прав доступа, который может быть предоставлен записями ACL_USER, ACL_GROUP_OBJ или ACL_GROUP;
- ACL_OTHER – определяет права доступа для процессов, которые не соответствуют никаким другим записям в списке.

Правильный список контроля доступа содержит в точности по одной записи типа ACL_USER_OBJ, ACL_GROUP_OBJ и ACL_OTHER. Записи типа ACL_USER и ACL_GROUP могут присутствовать ноль или большее число раз. Список, который содержит записи типа ACL_USER или ACL_GROUP должен содержать в точности одну запись типа ACL_MASK. Если ACL не содержит записей этих типов, то запись типа ACL_MASK необязательна.

Квалификаторы в записях типа ACL_USER и ACL_GROUP не должны повторяться.

Биты прав доступа, установленные для владельца файла, соответствуют разрешениям в записи типа ACL_USER_OBJ. Биты, установленные для группы владельца, соответствуют разрешениям в записи ACL_GROUP_OBJ, если в ACL отсутствует запись ACL_MASK. В противном случае биты разрешения, определенные для группы, соответствуют разрешениям записи типа ACL_MASK. Биты, установленные для остальных пользователей, соответствуют разрешениям в записи типа ACL_OTHER_OBJ.

Изменение битов прав доступа к файлу приводит к изменению соответствующих записей в ACL, и наоборот.

Обработка списков управления доступом происходит по следующему алгоритму:

1. Если эффективный UID процесса совпадает с идентификатором владельца файла, то

1.1. если запись ACL_USER_OBJ содержит запрашиваемые разрешения, то доступ предоставляется, иначе доступ запрещается;

2. иначе если эффективный UID процесса совпадает с квалификатором какой-либо записи типа ACL_USER, то

2.1. если эта запись и запись ACL_MASK содержат требуемые разрешения, то доступ предоставляется, иначе доступ запрещается;

3. иначе, если эффективный GID процесса или любой дополнительный GID процесса совпадает с группой владельца файла или с квалификатором любой записи типа ACL_GROUP, то

3.1. если ACL содержит запись ACL_MASK, то

3.1.1. если запись ACL_MASK и любая подходящая запись типа ACL_GROUP_OBJ или ACL_GROUP содержит требуемые разрешения, то доступ разрешается, иначе доступ запрещается;

иначе[270], если запись ACL_GROUP_OBJ содержит требуемые разрешения, то доступ разрешается, иначе доступ запрещается;

4. иначе, если запись ACL_OTHER содержит требуемые разрешения, то доступ разрешается;

[270] Заметим, что не может быть записей типа ACL_GROUP без записи ACL_MASK.

5. иначе доступ запрещается.

Для задания списков контроля доступа предусмотрены длинная и короткая текстовые формы. В обеих формах запись ACL представляет собой три поля, разделённых двоеточиями:

метка_типа_записи : *квалификатор* : *набор_прав_доступа*

Первое поле может содержать одно из следующих значений:

- user – для задания прав доступа владельца файла (тип ACL_USER_OBJ) или определённого пользователя (тип ACL_USER);
- group – для задания прав доступа членам группы владельца файла (тип ACL_GROUP_OBJ) или определённой группы (тип ACL_GROUP);
- mask – задаёт максимальный набор прав доступа, которые может быть предоставлен какой-либо записью, за исключением записи владельца файла и записи для остальных (ACL_OTHER).
- other – задаёт права доступа для пользователей, которые не соответствуют никакой другой записи ACL.

Второе поле содержит идентификатор пользователя или группы для записей типа ACL_USER или ACL_GROUP, и остаётся пустым для записей других типов. Можно использовать как имена пользователей и групп, так и их числовые идентификаторы в десятичном виде.

Третье поле содержит права доступа, представленные буквами r, w и x для чтения, записи и исполнения соответственно.

Длинная текстовая форма содержит одну запись на строку. Например:

```
user:alex:r--          #какой-то комментарий
group::r--
group:toolies:rw-      #добавлено 01.01.2018
mask::rw-
other::r--
```

Короткая текстовая форма представляет собой последовательность записей ACL, разделённых запятыми, при этом комментарии не поддерживаются, метки типа записи могут быть указаны в полной форме или в сокращённой однобуквенной форме. Разрешения могут содержать максимум по одному из символов r, w, x в произвольном порядке. Например:

```
u::rw-,u:alex:r--,g::r--,g:toolies:rw-,m::rw-,o::r--
g:toolies:rw,u:alex:rw,u::wr,g::r,o::r,m::r
```

Для работы со списками управления доступом используются команды `getfacl` и `setfacl`. Увидеть, что у файла есть расширенные записи по правам доступа можно по наличию знака «+» после отображения традиционных прав доступа утилитой `ls` с ключом `-l`. Создадим файл и добавим право «r» для пользователя user2.

```
$ touch 1.txt
$ ls -l 1.txt
-rw-rw-r-- 1 user user 0 ноя 30 17:04 1.txt
$ setfacl -m u:user2:r 1.txt
$ ls -l 1.txt
-rw-rw-r--+ 1 user user 0 ноя 30 17:04 1.txt
$ getfacl 1.txt
# file: 1.txt
# owner: user
```

```
# group: user
user::rw-
user:user2:r--
group::rw-
mask::rw-
other::r--
```

Подробнее см. man к указанным командам, а также [39].

5.3.5.1.7. Пользователи ОС Linux

С точки зрения ОС Linux все пользователи (и запущенные от их имени процессы, помимо имеющихся у них pid) различаются друг от друга по наличию идентификатора пользователя uid (user id). При разрешении вопросов доступа система ориентируется исключительно по uid номерам. В традиционных системах для хранения uid отведено 2 байта и обычно этого (2^{16}=65536 пользователей) хватает. Увидеть свой uid-номер можно командой

```
$ id
uid=1000(user) gid=1000(user) группы=1000(user),983(vboxusers)
```

также его можно узнать из файла /etc/passwd, сопоставляющего имена и номера пользователей

```
$ cat /etc/passwd|grep user
user:x:1000:1000:user:/home/user:/bin/bash
```

У суперпользователя (обычно это root) uid=0. Пользователи с номерами uid до n=1000 называются системными. Обычные пользователи имеют номера $\geq n$. В некоторых системах граница разделения обычных пользователей и системных проходит по n=500, в каких-то по n=1000.

Индексные дескрипторы файловых систем и файловые дескрипторы операционных систем хранят исключительно uid номера, а не имена пользователей.

Однопользовательские файловые системы не хранят никаких uid владельцев файловых объектов.

Добавить нового пользователя в ОС Linux можно командой

```
# adduser имя
```

5.3.5.2. Реализация дискреционной политики в Windows

Управление доступом и реализация дискреционной политики безопасности в ОС Windows, по аналогии с ОС Linux, заключается в предоставлении пользователям, группам и компьютерам (субъектам) определённых прав на доступ к объектам на локальном компьютере или по сети.

Ключевыми понятиями управления доступом являются разрешения, владение объектами, наследование разрешений, права пользователя и аудит объектов.

Если, по аналогии с ОС Linux, отобразить права доступа таблицей для файлов, то они будут немного отличаться набором разрешений (и тем, что непосредственная реализация выполнена списками доступа).

Таблица 5.5. Матрица доступа для ОС Windows (права: a – дополнять (Append); d – удалять (Delete); x – исполнять (eXecute); r – читать (Read); w – писать (Write); «–» – нет доступа (No Access))

		Объекты						
		Файл 1	Файл 2	Файл 3	Файл 4	Файл 5	Файл 6	...
Субъекты	User1	rwd	–	r	–	x	x	...
	User2	r	r	r	–	x	rwx	...
	User3	x	rwd	r	–	rx	r	...
	User4	–	r	rw	rwa	x	rw	...
	...	...	...	...	...	...	...	...

Матрица доступа (табл. 5.5) указывает права доступа отдельных субъектов к различным объектам. Каждый элемент данной матрицы указывает право доступа пользователя к данному файлу.

ФС сопоставляет запросы пользователей на открытие файла с установленными правами доступа. Если они согласуются, то доступ разрешается, иначе выдаётся ошибка доступа. Как было отмечено выше, способ управления доступом с помощью матрицы доступа очень прост, однако в больших системах, обслуживающих большое число пользователей, с матрицей возникают проблемы, связанные с её размерами и большой разряженностью. Также в ОС Windows появляется и другая проблема, проявляющаяся в том, что матрица доступа полностью хранится во внешней памяти; в оперативной памяти, согласно принципу кэширования, находятся те её части, к которым были недавние обращения.

Для упрощения жизни матрица доступа может быть разделена двумя способами:

по строкам и по столбцам.

- *По строкам → список возможностей.*

При делении матрицы по строкам формируется список, который называется списком возможностей. Идея деления матрицы по строкам не очень удачна. Во-первых, каждый список возможностей, содержащий запись на каждый файл, хранящийся в системе, будет иметь много записей «нет доступа». Этого можно избежать, если договориться, что матрица доступа будет составляться не на все файлы, то есть не будет включать «разряженные по правам» файлы в список. Во-вторых, для некоторых пользователей список доступа может быть очень большим. Например, список доступа персонала, выполняющего инсталляцию ОС. В-третьих, когда пользователь получает доступ в систему, должен быть создан новый лист возможностей. Причём этот список уже должен содержать права доступа к имеющимся в системе файлам. Наконец, объекты, в частности файлы в системе, – это гораздо более динамичные сущности, которые создаются и удаляются значительно чаще, чем пользователи, что привело бы к частому изменению списка возможностей даже в пределах одного сеанса работы пользователя с ОС. Идея деления матрицы доступа по строкам, так же как и в ОС Linux, не нашла применения.

- *По столбцам → Access Control List.*

В ОС Windows (как и в некоторых других ОС) используется список контроля доступа (Access Control List, ACL) – способ создания списков доступа для каждого файла по столбцам матрицы доступа (см. рис. 5.9).

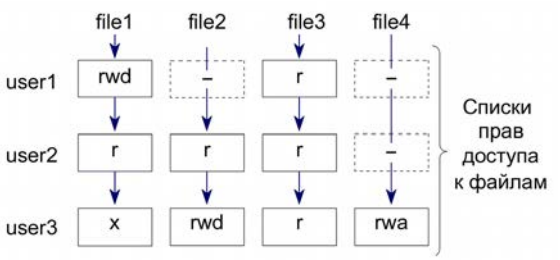


Рисунок 5.9. Разделение матрицы доступа «по столбцам»

Список доступа к файлу может храниться как часть файла, указатель на которую входит в состав дескриптора файла. Но и в этом случае список всех пользователей файла может быть очень длинным. Большинство пользователей разрешает совместное использование файлов в очень ограниченном виде, поэтому пользователей можно классифицировать следующим образом: владелец файла, группы, к которым принадлежит владелец файла, другие пользователи, утилиты архивации файла. Всей группе пользователей устанавливаются одинаковые права доступа. Права доступа могут быть изменены в командной строке посредством служебной программы *cacls*:

```
C:\>cacls c:\1.txt /G BUILTIN\Администраторы:W BUILTIN\Пользователи:R /C
Продолжить? (Y/N)y
обработан файл: c:\a.txt
```

Командой выше были устанавливаются права доступа к файлу c:\1.txt для встроенных групп пользователей BUILTIN\Администраторы – доступ «запись» – W, для BUILTIN\Пользователи – «чтение» – R.

В файловой системе NTFS файлам (см. табл. 5.6) и директориям (папкам, каталогам) (см. табл. 5.7) могут быть назначены следующие стандартные разрешения (permissions).

Таблица 5.6. Стандартные разрешения для файлов NTFS

Разрешение NTFS		**Пользователь**
No Access	None	… не может получить доступ к файлу
Read	RX	… может прочитать содержимое файла и запустить его на выполнение, если это файл программы
Change	RWXD	… вдобавок к разрешению Read может изменять и удалять файл
Full Control	All	…может читать, выполнять, изменять, удалять файл, устанавливать разрешения на доступ к файлу, становиться его владельцем

Таблица 5.7. Стандартные разрешения для папок NTFS

Разрешение NTFS [271]			**Пользователь**
No Access	None	None	…не может получить доступ ни к самой папке, ни к её содержимому
List	RX	Not Specified	…может только просмотреть содержимое папки (список файлов и вложенных папок) и перейти во вложенную папку; не может получить доступ к новым файлам, создаваемым в этой папке

[271] Во второй колонке указаны индивидуальные разрешения на доступ к самой папке, в третьей – на доступ к файлам, создаваемым в папке.

Разрешение NTFS			Пользователь
Read	RX	RX	…может прочитать содержимое файлов и запустить приложение в этой папке
Add	WX	Not Specified	…может создать в папке новые файлы и вложенные папки, но не может просмотреть её текущее содержание
Add & Read	RWX	RX	…может создавать в папке новые файлы или вложенные папки, прочитать содержимое самой папки и содержащихся в ней файлов и вложенных папок, а также запустить приложения, которые находятся в этой папке, но не может изменить содержимое файлов в этой папке
Change	RWXD	RWXD	…может прочитать, создать и удалить файлы и вложенные папки и запустить находящиеся в этой папке приложения
Full Control	All	All	…может прочитать, создать и изменить файлы и вложенные папки, изменить разрешения на папку и файлы внутри неё, а также стать владельцем папки и содержащихся в ней файлов

В скобках для стандартных разрешений показаны составляющие их индивидуальные разрешения (см. табл. 5.8) [4]. Подробнее о файловых системах см. раздел 5.3.7 «*Файловая система (уровень организации)*».

Таблица 5.8. Индивидуальные разрешения NTFS

Разрешение		Даёт пользователю возможность	
		для папки	для файла
Read	R	…считать список имён файлов и вложенных папок	…считать данные, содержащиеся в файле, и информацию о его атрибутах, разрешениях и владельце
Write	W	…создать в папке файлы и вложенные папки, изменить атрибуты папки, считать информацию о разрешениях и владельце; прочитать содержимое папки при этом нельзя, то есть разрешение Write не даёт разрешения Read	…создать или изменить данные, содержащиеся в файле, изменить атрибуты файла, прочитать информацию о разрешениях и владельце
Execute	X	…сделать папку текущей, например, командой «`cd имя`» в интерпретаторе командной строки	…считать информацию об атрибутах, разрешениях, владельце, запустить файл программы на исполнение (разрешение Read при этом не требуется)
Delete	D	…удалить папку	…удалить файл
Change Permission	P	…изменить разрешения доступа к папке	…изменить полномочия доступа к файлу
Take Ownership	O	…стать владельцем папки	…стать владельцем файла

Замечание. Для реализации в ОС матрицы доступа, списков и аналогичных средств требуется больше оперативной памяти по сравнению с однопользовательскими ОС или когда они не используются. Ведение таких средств доступа требует специальных методов обработки и приводит к дополнительным накладным расходам в системе. Например, чтобы оптимизировать

работу со списками прав доступа на файлы в томе, NTFS применяет индексный метод доступа к записям в файле метаданных $Secure, где хранятся все списки прав доступа к файлу на томе. Поскольку на томе персонального компьютера часто ограниченное число пользователей производит операции с файлами, то и списки прав доступа, назначаемые вновь создаваемым файлам, как правило, одинаковые. Поэтому ФС кэширует один экземпляр списка прав доступа в указанном файле, а в каждом дескрипторе файла размещает ссылку на этот экземпляр.

5.3.5.2.1. Пользователи ОС Windows

Поскольку файловые системы являются общим хранилищем файлов, принадлежащих разным пользователям (в многопользовательских ОС), системы управления файлами должны обеспечивать авторизацию доступа к файлам. В общем виде подход состоит в том, что по отношению к каждому зарегистрированному пользователю данной операционной системы для каждого существующего файла указываются действия, которые разрешены или запрещены данному пользователю или группе пользователей. В современных ОС для ПК предопределены три основные группы с разными правами: **администраторы, опытные пользователи, пользователи**.

При необходимости можно добавлять своих пользователей и создавать группы.

5.3.5.2.2. Разрешения (права доступа)

Конечным пользователям не обязательно работать с командной строкой и знать так глубоко все нюансы разграничения доступа на уровне файловой системы, им достаточно знать, что:

разрешения определяют тип доступа к объектам Windows или их свойствам, предоставляемый пользователю или группе (вкладка *Безопасность* окна свойств папки или файла).

Чтобы просмотреть или изменить разрешения на доступ к файлу или папке, достаточно в Проводнике щёлкнуть правой кнопкой мыши на имени файла или папке и выбрать пункт ***Свойства***. На вкладке ***Безопасность*** показаны текущие разрешения, которые можно изменить, если Обычный пользователь системы обладает необходимыми для этого правами, можете проверить, если у вас под рукой имеется компьютер.

Например, студентам группы 555 могут быть предоставлены разрешения на *создание, удаление, чтение* и *запись файлов* в папке «C:\Мои документы\xt-5» и запрещён доступ в другие вложенные папки для «C:\Мои документы».

С помощью управления доступом можно установить разрешения NTFS[272] для таких объектов, как файлы, объекты Active Directory, объекты реестра или системные объекты, например процессы. Разрешения могут быть предоставлены любому пользователю, группе или компьютеру. Целесообразно установить разрешения для групп, поскольку это увеличивает быстродействие системы при проверке прав доступа к объекту.

Для любого из этих объектов можно предоставить разрешения:

1. локальным группам и пользователям на том компьютере, на котором находится объект;
2. группам и пользователям данного или любого доверенного домена;

[272] Подробнее, что такое NTFS, см. раздел 5.3.6 «Файловая система».

3. группам, пользователям и другим объектам с идентификаторами безопасности домена.

Разрешения, назначаемые объекту, зависят от его типа. Например, разрешения, которые могут быть назначены для файла, отличаются от разрешений, допустимых для раздела реестра. Однако некоторые разрешения являются общими для большинства типов объектов. Общие разрешения перечислены ниже.

- Чтение
- Изменение
- Смена владельца
- Удаление

При установке разрешений определяется уровень доступа для групп и пользователей. Например, одному пользователю можно разрешить читать содержимое файла, другому – вносить изменения в файл, а всем остальным пользователям вообще запретить доступ к этому файлу. Также можно устанавливать разрешения на доступ к принтерам, чтобы одни пользователи могли настраивать принтер, а другие – только печатать.

5.3.5.2.3. Владение объектами

При создании объекта ему назначается владелец. По умолчанию владельцем объекта назначается тот, кто его создал. Какие бы разрешения не были установлены для объекта, владелец объекта всегда может изменить эти разрешения.

5.3.5.2.4. Наследование разрешений

Механизм наследования облегчает администраторам задачи назначения разрешений и управления ими. Благодаря этому механизму разрешения, установленные для контейнера, автоматически распространяются на все объекты этого контейнера. Например, файлы, создаваемые в папке, наследуют разрешения этой папки. Наследуются только отмеченные для наследования разрешения.

5.3.5.2.5. Права пользователя

Права пользователей дают возможность пользователям и группам в компьютерной среде пользоваться особыми привилегиями и правами на вход в систему. Администраторы могут назначать права учётным записям групп или отдельных пользователей. Эти права позволяют пользователям выполнять конкретные действия, такие как интерактивный вход в систему или архивирование файлов и каталогов.

Права пользователей отличаются от разрешений тем, что применяются к учётным записям пользователей, а не к объектам. Хотя права пользователей и могут применяться к учётным записям отдельных пользователей, удобнее администрировать права пользователей на основе групп. Пользовательский интерфейс управления доступом не поддерживает прав пользователей, однако их предоставлением можно управлять с помощью оснастки «Локальные параметры безопасности».

5.3.5.2.6. Аудит объектов

Обладая правами администратора, можно контролировать успешные и неудавшиеся попытки доступов пользователей к объектам. Можно выбрать объект, аудит доступа к которому выполняется с помощью пользовательского интерфейса управления доступом, но сначала необходимо включить политику аудита, выбрав доступ к объекту аудита в элементе «Локальные политики» в оснастке «Локальные параметры безопасности». Впоследствии события, связанные с безопасностью, можно просмотреть в журнале безопасности в окне программы «Просмотр событий».

5.3.5.2.7. Общий доступ к папкам и файлам

В **Windows 7/8** реализованы два способа предоставления общего доступа к файлам: можно предоставить общий доступ к файлам, находящимся в группе папок *Общие* (*Public*), или к файлам любой папки на компьютере, задавая общий доступ к папке в её свойствах (рис. 5.10).

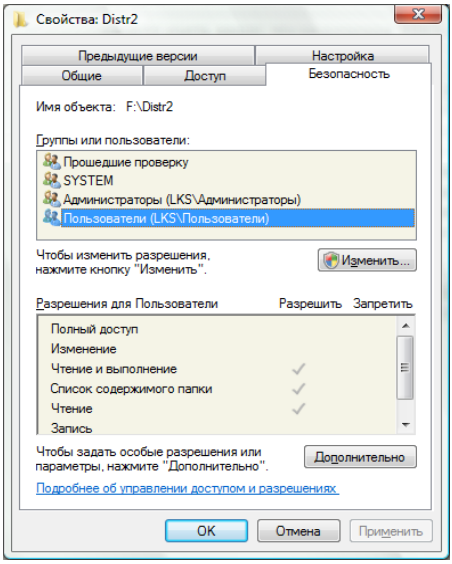


Рисунок 5.10. Окно свойств папки

Папки *Общая музыка*, *Общие видео*, *Общие документы*, *Общие изображения* в папке *Общие* являются удобным средством для обмена файлами, через них можно обмениваться файлами с другими пользователями компьютера или же с пользователями других компьютеров, находящихся в одной сети [273]. Любой файл или папка, помещённые в эти папки, автоматически становятся доступными для других локальных пользователей.

Папку *Общие* следует использовать, если вы хотите хранить все файлы, к которым необходимо предоставить совместный доступ, в одном месте на компьютере и нет необходимости настраивать разрешения для отдельных пользователей сети.

Второй способ следует использовать, если вы хотите предоставить доступ к файлам другим пользователям прямо из места их размещения, при этом необходимо настроить разрешения на доступ к файлам для отдельных пользователей компьютера – вкладка *Безопасность* окна свойств или сети – вкладка *Доступ* окна свойств (см. рис. 5.10). Окончательные разрешения на общий доступ к папке определяются с учётом как набора разрешений NTFS (*Безопасность)*, которые можно также установить для файлов, так и для общего ресурса (*Доступ*).

Панель управления **Windows Vista** содержит новый компонент, называемый **Центром управления сетями и общим доступом**. С помощью этого центра можно включить или отключить сетевое обнаружение и общий сетевой доступ к файлам и папкам компьютера.

[273] Доступ к папкам и файлам по сети происходит по протоколу CIFS (или SMB) после аутентификации в домене или на компьютере. Предполагается, что сетевое соединение установлено (присвоен IP-адрес и настроены другие сетевые параметры).

5.3.5.3. Другие политики безопасности

Прежде чем говорить о политиках безопасности, в общем случае следует выделить три основных вида угроз, которые существуют в компьютерных и информационных системах[274]:

угроза целостности	,	угроза конфиденциальности	и	угроза доступности.

Все другие существующие угрозы есть комбинации этих трёх.

Пример угрозы целостности: мошенники на вашем счету в банке взяли и вместо 100 рублей каким-то образом записали 0. Нарушена целостность данных, хранящихся в банковском компьютере.

Пример угрозы конфиденциальности: вы приготовили сюрприз жене и сделали заказ в службе доставки подарков. База заказов стала доступной через интернет, и ваша супруга узнала о готовящемся сюрпризе.

Пример угрозы доступности: перед Новым годом, 31 декабря, вы решили заказать букет цветов по телефону в интернет-магазине, но не смогли туда дозвониться по причине большого числа желающих и занятости телефонных линий. Необходимый вам сервис оказался недоступен.

Для обеспечения защиты от отказанных угроз в теории, помимо дискреционной политики безопасности, принимаются различные меры, а также существуют другие политики безопасности, например мандатная, типизированная и др.

Политика безопасности – это совокупность норм, правил и практических рекомендаций, регламентирующих работу средств защиты компьютерной системы от заданного множества угроз. [23, стр. 21]

Основная из политик в применении к ЭВМ – обеспечение эффективного разграничения информационных потоков между пользователями и процессами одной системы (или совокупности систем). Например, могут разграничиваться процессы одного и того же текстового редактора, но работающего с информацией разного грифа (напр., «конфиденциально» и «общедоступно»). Наиболее наглядно суть подобных политик можно пояснить следующим примером: вы готовите годовой отчёт фирмы (не обязательно текстовый, может быть таблица или презентация) и хотите вставить туда цифры о ваших конкурентах для сравнения. Для этого вы заходите в интернет, находите там интересующие вас данные о конкурентах, опубликованные публично, копируете их в свой отчёт (с помощью мышки copy-paste, копировать+вставить, или нажатием на клавиатуре клавиш «Ctrl+Ins»+«Shift+Ins» или «Ctrl+c»+«Ctrl+v»). Как вы понимаете, это разрешённая операция, но где гарантия, что вы не ошибётесь и не скопируете данные в

[274] В четвёртом издании мы пришли к выводу, что следует отдельно выделить угрозу «авторства» (они же репутационные риски). Это довольно новый тип угроз серьёзно не воспринимаемый большинством специалистов, поскольку реализуется вне компьютера жертвы. Например, в сети выкладывается какой-либо документ, якобы от автора X, при этом подписанный его электронной подписью. В случае нарушения авторских прав X будет тратить силы и время на доказательство своей непричастности. Также данный вид угроз легко применим к различным цифровым деньгам с распределённым механизмом подтверждения транзакций, например на основе block chain. Естественно, что операционные системы защищать пользователей от угрозы авторства пока не могут, но с течением времени вопрос будет становиться более актуальным.

обратном направлении, то есть из вашего отчёта в интернет? Хорошая система не должна дать пользователю возможность скопировать данные не в том направлении, выдав предупреждение либо запретив операцию целиком. В теории реализовать подобное можно, например, с помощью мандатного разграничения доступа (мандатной политики безопасности). На практике же (на сегодняшний день) это наиболее сложная задача – сделать систему удобной и «безопасной» для обрабатываемых данных, и решается она далеко не на бытовом уровне, потому как кроме передачи данных через буфер обмена могут существовать как вирусы (вредоносное ПО), так и штатные программы, работающие с файлами, и другие возможности для нежелательного распространения информации.

В ОС Linux попытки сдвинуться в этом направлении давно приняты – в частности, в коде ядра (начиная с версии 2.6.x) имеется специальное расширение SELinux, затрагивающее и файловую систему, и пользователей. В ОС Solaris используются зоны и прочее. Технология SELinux ввела такие понятия как ***типизированная политика безопасности*** и ***контекст безопасности***. Ряд штатных команд были дополнены новым ключом «`-Z`», например его можно использовать для таких команд как `ls`, `ps`:

```
$ ls -Z /|head -n 6
dr-xr-xr-x. root root system_u:object_r:bin_t:s0       bin
dr-xr-xr-x. root root system_u:object_r:boot_t:s0      boot
drwxr-xr-x. root root system_u:object_r:device_t:s0    dev
drwxr-xr-x. root root system_u:object_r:etc_t:s0       etc
drwxr-xr-x. root root system_u:object_r:home_root_t:s0 home
$ ps -Z
LABEL                                             PID TTY    TIME  CMD
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 7536 pts/0 00:00:00 bash
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 7577 pts/0 00:00:00 ps
```

Механизмов много, но их описание выходит за рамки данной книги. Можем констатировать, что самое эффективное, но не самое удобное на сегодня средство разграничения (защиты от нежелательного копирования данных) – это иметь два отдельных компьютера, не связанных друг с другом. Причём это можно распространить и на телефоны. Даже самый изощрённый вирус не сможет спрятать от пользователя SMS с кодом подтверждения (какой-либо online операции), если она придёт на другой телефонный аппарат.

Замечание. Даже идеальная программная реализация политики безопасности не поможет уберечь ваши данные, будучи запущенной на не доверенной аппаратной платформе. Вместе со «своей» операционной системой надо иметь «свой» процессор. Помните: идеальных средств защиты не существует, поэтому если не хотите, чтобы ваши данные были доступны всему миру, – не подключайте ваш компьютер к интернету. К сожалению, не подключать телефоны или планшеты к интернету невозможно. К счастью, не все страдают паранойей и, наоборот, только рады размещать свои фотографии и файлы на общее обозрение.

5.3.6. Основа безопасного разграничения – файловая система

Схожесть базовых принципов ОС UNIX и ОС GNU/Linux, в сочетании с большой гибкостью последней и наличием исходных кодов, доступных для свободной модификации согласно лицензии GPL, приводит к более быстрому росту Linux и постоянному увеличению степени её проникновения во все технические сферы, прямо или косвенно связанные с защитой информации. Не отстаёт от неё и Windows своей рекламой при-

влекать новых пользователей, однако первоначально заложенные в ОС UNIX решения по защите информации, такие как идентификация пользователей и разграничение доступа, с некоторыми дополнениями успешно используются и сегодня при построении защищённых систем.

Большинство систем UNIX по уровню безопасности удовлетворяют требованиям класса C2, сформулированным в Оранжевой книге «Критерии безопасности компьютерных систем» Министерства обороны США, однако сегодня эти критерии заменили новые документы – международный стандарт ISO 15408 «Общие критерии безопасности информационных технологий» и отечественный стандарт ГОСТ Р ИСО/МЭК 15408. Вместо критериев C2 используется набор критериев Controlled Access Protection Profile [6, стр. 289].

Изначально ОС UNIX разрабатывалась как открытая и удобная для использования система. К сожалению, открытость и удобство использования могут приводить к нарушению механизмов безопасности, определённых позднее, в связи с чем для обеспечения безопасности ОС UNIX принципиально важны её правильная конфигурация и настройка. С это целью, например в ОС OpenBSD, принципиально не содержится часть готовых к использованию примеров конфигурационных файлов, которые можно встретить в других системах UNIX или Linux.

Чтобы корректно настроить систему, требуется не только правильно и полно определить потенциальные угрозы, но и знать, как функционируют те или иные средства для защиты в ОС. Во время настройки следует учитывать «*вычислительные и материальные затраты на обеспечение безопасности и строить систему защиты в соответствии с принципом разумной достаточности*» [6, стр. 289].

Во время осуществления мер по защите любых систем следует понимать, что для успешного запуска стороннего кода (который потенциально может быть опасен), независимо от используемой операционной системы, необходимо соблюдение двух условий:

наличие самого кода и передача ему соответствующих прав на управление.

Отсутствие одного из двух компонентов полностью нейтрализует опасность. Понимая этот принцип, администраторы систем превращают последние в закрытые программные среды, а также выполняют разные вытекающие из этих принципов действия.

Так как любые программы (процессы, потоки), запущенные в системе, чаще всего изначально являются файлами файловой системы, наибольшее внимание в вопросах безопасности операционных систем, как и функционирующих под их управлением сервисов, необходимо уделять защите файловой системы. Случай, когда будущий исполняемый код получается системой не из файлов, например по сети, с клавиатуры и другими способами – редкий, но и в этом случае можно прибегнуть к понятию «псевдофайлов» [15] устройств или сокетов и оперировать с их правами. На основании этого принципа все пользователи ОС UNIX и ОС Linux (в том числе и идентифицируемые системой как анонимные: пользователь nobody) явно или неявно работают с файлами. («*...UNIX сформировалась как концептуальная целостность: понятие файла как универсального интерфейса доступа ко всему на свете*» [4, стр. 62].)

Файловая система UNIX-подобных операционных систем имеет как общее, например древовидную структуру, состоящую из директорий и файлов, обычно согласно стандарту (рекомендации) FHS [7] (см. раздел 3.5.4), так и различное, в зависимости от типа операционной системы и потребностей пользователей. Стандартов по вы-

бору «различного», то есть той или иной файловой системы (ФС), не существует и не может существовать, поэтому в качестве базовой ФС, то есть той, с которой может происходить загрузка ОС и на которой будут храниться файлы ОС, могут использоваться как ext2, ext3, ext4, reiserfs, hfr, ufs, xfs, zfs и др., в зависимости от выбранной вами ОС. Следует отметить, что не все ФС могут быть базовыми, например ядро ОС Linux хоть и может примонтировать ФС: zfs, а также семейства FAT и NTFS, но оно не может их использовать для полноценной работы.

Базовой системой для ОС Windows является NTFS.

«*В разных диалектах UNIX поддерживает множество разных типов файловых систем. Однако все они имеют схожую структуру. Файловая система UNIX, независимо от типа, содержит четыре основных компонента с управляющей информацией: загрузочный блок, суперблок, таблицу индексных дескрипторов (i-node table) и каталоги*» [14, стр. 49].

При защите данных важно понимать особенности хранения данных в той или иной файловой системе, наличие и предназначение дополнительных атрибутов, умело использовать их.

Переходя плавно к обсуждению вопросов физической реализации разграничения доступа в файловых системах, логично, что было необходимо рассмотреть изначальную предпосылку и желание реализовать ту или иную политику безопасности в операционных системах.

5.3.7. Файловая система (уровень организации)

Файловая система – способ (договорённость, формат) организации выделенного пространства памяти (на жёстком диске, flash-накопителе, дискете, компакт-диске и прочем) с целью обеспечения оптимального хранения в ней информации в виде файлов и получения доступа к ним.

Каждая операционная система использует одну основную файловую систему, а также дополнительно может работать с некоторыми другими файловыми системами. Напомним, что в рамках файловой системы под файлом понимается упорядоченная совокупность информации на диске, имеющая своё имя.

Существуют разные файловые системы, из разумных соображений их можно поделить на те, которые с большой вероятностью могут встретиться читателю: **FAT, NTFS, ext2/3/4, CDFS, UDF**, – и те, с которыми он, оставаясь в роли пользователя, скорее всего, никогда не столкнётся: *Btrfs, EFS, exFAT, ext3cow, FATX, Next3, ReFS, ReiserFS, Reiser4, Squashfs, ZFS* и др. Рассмотрим ФС первой группы.

Эксперимент. Уважаемые читатели, попробуйте ответить на вопрос: *сколько места на жёстком диске будет занимать файл размера 123 байта?* Если вам наш вопрос кажется слишком простым, а ответ «сто двадцать три байта» кажется правильным, то, забегая вперёд, скажем, что он не верен, а разъяснение «почему?» даётся в последующих разделах, описывающих устройство файловых систем.

Дело в том, что для организации записи конкретных 123 байт на диске используется некоторое количество служебной информации. Содержимое файлов и их хранение подобно тому, как вы не можете купить в магазине только 1 литр молока и ничего более[275], а вынуждены вме-

[275] Если только не в розлив. И то, посмотрев на рекламу холодильника – объём камеры охлаждения 200 литров (или автомобиля: объёмом багажника 400 литров), вы вряд ли сможете выделенное пространство

сте с молоком покупать и его упаковку – пакет или бутылку. Упаковка упаковке рознь, например мягкий полиэтиленовый пакет не хрупкий, зато он не может стоять самостоятельно вертикально. Стеклянная бутылка может стоять в открытом виде вертикально, зато она не только хрупкая, но и тяжёлая и т. д. Одна хозяйка может удобно упаковать и расставить продукты в холодильнике, другая ещё лучше и компактней... Собственно, файловая система есть описание того, как физически размещаются файлы на диске (продукты на полках холодильника или склада и в какой таре).

5.3.7.1. FAT

В файловой системе FAT той самой хозяйкой, распределяющей продукты по холодильнику, является таблица размещения файлов, по-английски File Allocation Table, откуда, собственно, и происходит название ФС. Таблица размещения файлов, а также её копия содержатся на каждом диске с ФС FAT.

Обращение к дискам на физическом уровне обычно производится посекторно, и весь диск можно поделить на маленькие кусочки по 512 байт, которые по одному или группируясь будут составлять бóльшие логические блоки – кластеры.

Группы кластеров можно объединить по своему назначению в функциональные блоки. Структура разбиения диска на функциональные блоки и место расположения таблицы FAT, как и её копии, представлены на рис. 5.11.

Рисунок 5.11. Структура тома ФС FAT. Нумерация секторов диска (и кластеров) слева направо

Таблица размещения файлов довольно условно называется таблицей и выполняет две функции: содержит информацию распределения данных каждого файла по кластерам диска в форме списка связей, занятых кластеров (цепочек) и указывает, какие кластеры свободны (см. рис. 5.12.).

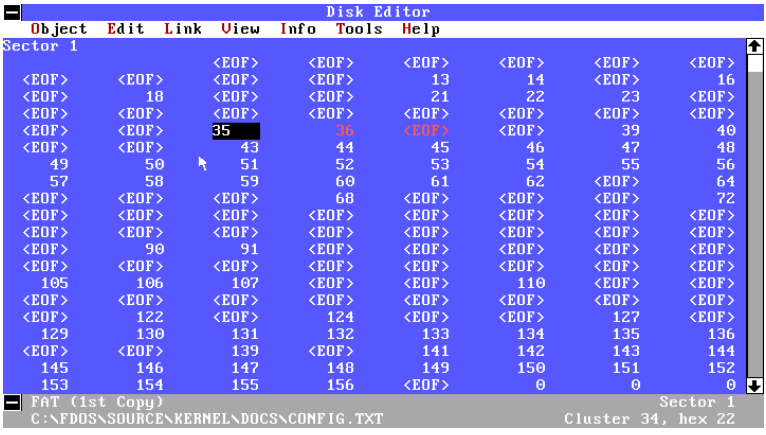


Рисунок 5.12. Вид FAT-таблицы

полностью заполнить молоком.

На Рисунке 5.12 представлен вид 1-й копии таблицы FAT в режиме редактирования в программе diskedit.exe из набора Norton Utilities. Три кластера, принадлежащие цепочке некоторого файла, помечены цветом. Остальные кластеры также заняты. Не занятые содержат 0.

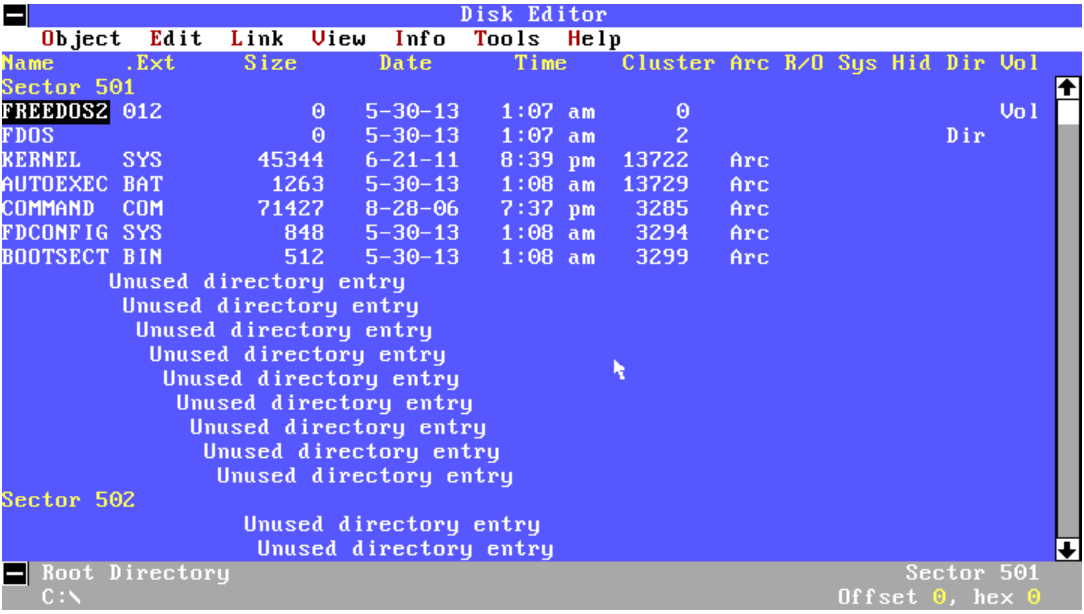


Рисунок 5.13. Внутреннее устройство корневой директории диска C: в отображении программой diskedit.exe из набора Norton Utilities. В колонке cluster указывается номер начального кластера хранения для файла или директории

Том, отформатированный для использования файловой системы **FAT**, размечается по кластерам (минимальная адресуемая единица в файловой системе). Размер кластера по умолчанию определяется, точнее зависит от размеров тома.

Существует несколько модификаций ФС **FAT**. Так, после названия ФС может идти число, уточняющее разрядность, применяемую для идентификации кластеров на диске[276]. 12-разрядный идентификатор кластеров в **FAT12** ограничивает размер дискового раздела 2^{12} (4096) кластерами. **FAT16** – за счёт 16-разрядных идентификаторов кластеров – может адресовать до 2^{16} (65 536) кластеров. В **Windows 2000** размер кластера **FAT16** варьируется от 512 байт до 64 КБ (размер 64 КБ не совместим с DOS и некоторыми другими ОС), поэтому размер FAT16-тома ограничен 4 ГБ (или в DOS – 2 ГБ). По задумке разработчиков, по мере роста размеров диска предполагался и рост размера кластера[277]. Зависимость между размером диска и размером кластера по умолчанию представлена в табл. 5.9.

[276] Интересный факт: ФС FAT 32 первые 4 бита для адресации зарезервированы, поэтому логичнее её было бы называть FAT 28. Вполне вероятно, это было сделано по маркетинговым соображениям, чтобы мотивировать пользователей использовать на NTFS. Различное ПО из-за своей внутренней непродуманности также не может работать с дисками больше того или иного размера, далеко не доходя до теоретически максимально возможного. Так, штатная утилита format из Windows 2000 не форматирует диски ФС FAT 32 более чем в 32 ГБ.

[277] Позже выяснилось, что обычно в ОС используется большое число мелких файлов, использовать для которых большой размер кластера не выгодно.

Пользователь имеет возможность указать другой размер кластера, однако устанавливаемый размер не может быть меньше размера по умолчанию. Из табл. 5.8. видно, что если вы не планируете хранить большие файлы, использовать файловую систему **FAT 16** на дисках более 511 Мбайт не очень выгодно из-за возрастающих накладных расходов (файл занимает всегда целое число кластеров, и основная их масса будет меньше 16 КБ, см. вопрос во врезке «Эксперимент» (стр. 447), про файл размером 123 байта). Это сейчас размеры дисков измеряются терабайтами плюс-минус километр, а во времена начала 90-х (расцвета использования DOS) стоимость хранения 1 МБ была высокой (пожалуй, сопоставимая с хранением 1 ГБ сейчас), поэтому даже существовали утилиты в виде драйвера ОС (Drvspace.sys и Dblspace.sys), предлагавшие увеличить эффективное использование жёсткого диска до 2 раз.

*Таблица 5.9. Размеры кластеров в **FAT 16** по умолчанию*

Размер раздела, Мбайт	Количество секторов в кластере	Размер кластера, КБ
0–32	1	0,5
33–64	2	1
65–128	4	2
129–256	8	4
257–511	16	8
512–1023	33	16
1024–2047	64	32
2048–4096	128	64

Файловая система **FAT 32** впервые появилась в **Windows 95 OSR2**, затем использовалась в **Windows 98** и **Windows ME**. **FAT 32** использует 32-разрядные идентификаторы кластеров, но при этом резервирует старшие 4 бита, так что эффективный размер идентификатора кластера составляет 28 бит. Поскольку максимальный размер кластеров FAT 32 равен 32 КБ, теоретически **FAT 32** может работать с 8-терабайтными томами (табл. 5.10).

Помимо большего предельного числа кластеров, преимуществом **FAT 32** перед **FAT 12** и **FAT 16** является тот факт, что место хранения корневого каталога **FAT 32** не ограничено предопределённой областью тома, поэтому его размер не ограничен. Кроме того, для большей надёжности **FAT 32** хранит вторую копию загрузочной записи. В **FAT 32**, как и в **FAT 16**, максимальный размер файла равен 4 ГБ, поскольку длина файла в каталоге описывается 32-битным числом.

*Таблица 5.10. Размер кластера на томах **FAT 32** по умолчанию*

Размер раздела	Размер кластера, КБ
от 32МБ до 8 ГБ	4
8–16 ГБ	8
16–32 ГБ	16
более 32 ГБ	32

В конце данного раздела логично спросить:

Как вы думаете, почему случается так, что на USB-флэшку или карту памяти, например для фотоаппарата или видеорегистратора, нельзя записать файл размером более 4 ГБ?

5.3.7.2. VFAT

VFAT – это расширение ФС **FAT**, появившееся в **Windows 95**. В **FAT** имена файлов имеют формат 8.3 и состоят только из символов кодировки **ASCII**. В **VFAT** была добавлена поддержка длинных (до 255 символов) имён файлов (англ. Long File

Name, LFN) в кодировке **UTF-16LE**, при этом LFN хранятся одновременно с именами в формате 8.3, ретроспективно называемыми SFN (англ. Short File Name). LFN нечувствительны к регистру при поиске, однако, в отличие от SFN, которые хранятся в верхнем регистре, LFN сохраняют регистр символов, указанный при создании файла.

Обратная совместимость с DOS, естественно без поддержки длинных имён, была сохранена за счёт того, что вносимые записи длинных имён отмечались как «Vol» (атрибутом метки тома) и поэтому в DOS не отображались.

5.3.7.3. NTFS

С целью преодоления имевшихся в **FAT** ограничений для операционной системы **OS/2** специалистами IBM и Microsoft была придумана и представлена в ноябре 1989 года новая файловая система **HPFS** (High Performance Filesystem – высокопроизводительная файловая система). Данная система до сих поддерживается в таких ОС, как **OS/2**, **Windows NT, Linux** и **FreeBSD**.

Позднее фирма Microsoft решила, как всегда, повести пользователей своим путём, выпустив на базе **HPFS** не совсем совместимую свою файловую систему **NTFS** (от англ. New Technology File System) и брать отчисления с других фирм, желающих её использовать. А чтобы энтузиасты не разрабатывали свободное ПО, работающее с томами NTFS, полная спецификация по ФС была закрыта. С связи с этим в «мире UNIX» долгое время запись в данную ФС не поддерживалась в полной мере. На сегодня поддержка томов **NTFS** в Linux давно реализована несколькими способами, но использование данной ФС (на уровне администраторов систем) не актуально, потому как в реальных системах используются другие ФС, обходящие NTFS по тем или иным параметрам, в том числе и по производительности. На действующих серверах, с целью экономии места и оптимизации производительности, даже может и не быть скомпилированного модуля для данной ФС.

В «мире Windows» ФС **NTFS** обеспечивает производительность, надёжность и совместимость, которые не в состоянии предоставить файловая система **FAT**. **NTFS** разрабатывалась с целью обеспечения высокой скорости и надёжности выполнения операций с файлами (включая чтение, запись, поиск) при использовании дисков больших объёмов. **NTFS** использует 64-разрядные индексы кластеров. Это позволяет в теории **NTFS** адресовать тома размером до 16 ЭБ ($16 \cdot 2^{60}$ байт ≈ 16 миллиардов ГБ).

NTFS поддерживает контроль доступа к папкам и файлам как для локальных, так и для сетевых ресурсов.

Как и другие файловые системы, **NTFS** делит всё полезное место диска на кластеры – адресуемые блоки данных, размер которых может быть от 512 байт до 64 Кбайт (см. табл. 5.11), стандартом считается кластер размером 4 Кбайт.

NTFS поддерживает ряд дополнительных возможностей – защиту файлов и каталогов, дисковые квоты, сжатие файлов, символьные ссылки на основе каталогов и шифрование.

*Таблица 5.11. Размеры кластеров в **NTFS** по умолчанию*

Размер раздела, МБ	Размер кластера, КБ
512 и менее	0,512
513–1024	1
1025–2048	2
более 2048	4

Одно из важнейших свойств **NTFS** – ***восстанавливаемость***. При неожиданной остановке системы [278] целостность метаданных тома **FAT** может быть утрачена, что

[278] До недавнего времени практически все ОС не отличались стабильностью работы даже при полностью

вызовет повреждение структуры каталогов и некоторого объёма данных. При больших размерах полная проверка диска и устранение неисправностей могут занять несколько часов. Чтобы сократить область проверки диска и не проверять всё, **NTFS** ведёт журнал изменений метаданных путём протоколирования транзакций, поэтому целостность структур файловой системы может быть восстановлена быстрее. Заметим, что восстановление целостности ФС не обязательно подразумевает восстановление утерянных данных из файлов. Транзакция [279] – операция, выполняющаяся по принципу «всё или ничего». При неудачном выполнении операции в файловой системе происходит ***откат*** (roll back), после которого система возвращается в исходное состояние, в котором она была до начала транзакции.

Помимо восстанавливаемости, ФС поддерживает ряд дополнительных возможностей:

- файлы и каталоги могут содержать несколько потоков данных;
- имена файлов, каталогов и томов имеют кодировку Unicode;
- универсальный механизм индексации файлов для ускорения их поиска;
- динамическое переназначение плохих кластеров;
- работа со сжатыми файлами;
- протоколирование изменений;
- квоты томов, индивидуальные для каждого пользователя;
- отслеживание ссылок (правильность ссылок ярлыков и OLE-связей при перемещении источников);
- шифрование (механизм Encrypting File System – EFS);
- частичная поддержка POSIX (чувствительность к регистру в именах, цепочечные разрешения доступа, метки времени изменения файлов).

В **NTFS** все служебные данные, хранящиеся на томе, содержатся в обычных файлах со специальными именами. Это относится и к структурам данных, используемым для поиска и выборки файлов, к начальному загрузочному коду и к битовой карте, в которой регистрируется состояние пространства всего тома (метаданные **NTFS**). Хранение всех видов данных в файлах позволяет файловой системе легко находить и поддерживать данные, а каждый файл может быть защищён дескриптором защиты. Кроме того, при появлении плохих секторов на диске **NTFS** может переместить файлы метаданных.

Диск **NTFS** делится на две логические части. При форматировании первые 12% диска отводятся под **MFT** зону (некий аналог таблицы размещения файлов FAT) – пространство, в котором растёт файл **MFT** (Master File Table, главная таблица файлов, см. рис. 5.14).

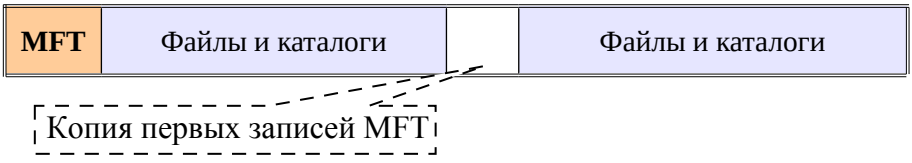


Рисунок 5.14. Структура тома в NTFS

исправном аппаратном обеспечении (железе). По поводу экранов BSOD (Blue Screen Of Death, жарг. «звёздное небо») в Windows в фольклоре системных администраторов сложено немало анекдотов.

[279] Термин пришёл в ФС из баз данных.

Это сделано для того, чтобы файл **MFT** не был фрагментирован. Но когда всё остальное место на диске заполняется – зона **MFT** может несколько раз уменьшаться в два раза, пока это возможно (на больших дисках эта зона – десятки ГБ и полностью под файл **MFT** не используется).

MFT реализована как массив записей о файлах и папках (номер в **MFT**, имя, размер, положение на диске отдельных фрагментов и т. д.). Размер каждой записи о файле фиксирован и равен 1 КБ. Кроме **MFT**, в корневом каталоге каждого тома **NTFS** имеется набор файлов метаданных с информацией, необходимой для реализации структуры файловой системы. Имена всех файлов метаданных **NTFS** начинаются со знака доллара ($), хотя эти знаки скрыты. Так, имя файла **MFT** – $**MFT** (см. табл. 5.12). Остальные файлы **NTFS**-тома являются обычными файлами и каталогами.

Таблица 5.12. Метафайлы NTFS (пользователь не видит их)

Имя файла	Назначение
$MFT	файл **MFT**
$MFTmirr	копия первых 16 записей **MFT**, размещённая посередине диска
$LogFile	файл поддержки журналирования (лог-файл)
$Volume	служебная информация о томе – метка тома, версия файловой системы и т. д.
$AttrDef	список стандартных атрибутов файлов на томе
$.	корневой каталог
$Bitmap	карта свободного места тома
$Boot	загрузочный сектор (если раздел загрузочный)
$Quota	файл, в котором записаны права пользователей на использование дискового пространства (начиная с NTFS версии 5)
$Upcase	таблица соответствия заглавных и прописных букв в именах файлов на текущем томе

Для конечных пользователей будет всё равно, но для администраторов существуют различия NTFS по версиям, например в **NTFS 5.0**, по сравнению с версией **NTFS 4.0**, добавлена такая функция, как точки монтирования, или, по-другому, точки соединения (junction point), появились квотирование и возможность поиска файла по имени его владельца.

Если вы используете операционную систему на ядре NT (**Windows XP, Vista, 7, 8**), то применять какую-либо файловую систему, отличную от **NTFS**, – значит существенно ограничивать своё удобство и гибкость работы самой операционной системы. Множество полезных возможностей NT напрямую завязано на физическую и логическую структуру этой файловой системы. При этом форматировать USB-флэшки в ФС NTFS не имеет смысла, так как большой процент места для записи займёт служебная информация. Оно может быть оправдано лишь как временный выход, если надо перенести файл более чем 4 ГБ на другой компьютер, а разбить его на части и потом склеить обратно нет никакой возможности.

С другой стороны, использование чего-то с большими возможностями, о которых вы не знаете и которые вам заведомо не нужны, вступает в противоречие с требованиями безопасности «*о минимизации возможностей и привилегий в системе*», что влечёт в себе опасность, что указанные «дополнительные» возможности могут быть исполь-

зованы тем же вредоносным ПО (компьютерными вирусами). Так, файловая система NTFS позволяет от пользователя (в силу его незнания) прятать информацию в альтернативных файловых потоках, что не позволяет делать ФС FAT.

5.3.7.3.1. Альтернативные файловые потоки в ФС NTFS

Возможность заключается в том, что у одного файла может быть несколько потоков, содержащих данные, причём по умолчанию пользователю доступен лишь главный поток, в котором хранится содержимое файла.

Поток можно прикрепить к любому файлу (при этом его размер не меняется и данные остаются нетронутыми, а значит, утилиты, проверяющие контрольные суммы файлов, не заметят изменений) или к каталогу. С другой стороны, тот же Антивирус Касперского также умеет использовать потоки, но для своих благих целей, чтобы не проверять одни и те же файлы по нескольку раз, в поток помещается отметка о моменте последней проверки файла. Архиватор WinRAR умеет сохранять потоки.

В Windows потоки обычно используются для хранения какой-либо дополнительной информации о файле. Например, в потоке может содержаться сводка документа. Если система стоит на диске с NTFS, то файл explorer.exe наверняка содержит сводку. В зависимости от содержимого сводки к файлу могут прикрепляться потоки с именами SummaryInformation, DocumentSummaryInformation и некоторые другие.

Приведём несколько консольных команд для работы с потоками:

```
type nul > file.txt:Stream            # Создание файла с потоком
echo "Тест" >> file.txt:Stream        # Запись в поток
more < file.txt:Stream                # Чтение из потока
type file1.txt >> file.txt:Stream     # Копирование содержимого
                                        существующего файла в поток
more < file.txt:Stream >> file2.txt   # Копирование содержимого потока в файл
```

Прикрепив поток с информацией к чему-нибудь, до его содержимого трудно добраться, не зная его имени. Начиная с Windows Vista найти альтернативные потоки у файла или директории возможно командой dir /R, которая показывает все потоки данных, в том числе и альтернативные, а вот, в более ранних версиях нет стандартно установленных по умолчанию средств обнаружения альтернативных потоков. Дополнительная трудность состоит в том, что в имени потока могут содержаться символы, недопустимые в именах обычных файлов (возврат каретки, перевод строки и прочие, которые просто невозможно набрать в консоли), что и создаёт дополнительные трудности при попытке узнать содержимое потока, пользуясь командной строкой. Помимо стандартно запрещённых для имён файлов символов: `*`, `?`, `<`, `>`, `|` и `"`, – в именах потоков могут использоваться и служебные символы с кодами 0x01–0x20.

Решение проблемы – написать свою собственную программу, используя в ней стандартные WinAPI-функции CreateFile, DeleteFile, ReadFile и WriteFile по работе с файлами, – не по силам большинству пользователей, зато о прикреплённых к файлу потоках они могут узнать и случайно, например в некоторых случаях копирования файлов с прикреплёнными потоками на диск, где они не поддерживаются. Так, при копировании с NTFS на какую-либо из систем FAT-система может выдать запрос на подтверждение потери информации в потоках, указав их названия.

Замечание 1. Нечто похожее есть в файловой системе HFS на MacOS. Там потоки (streams) называются разветвлениями (forks) и до появления MacOS X использовались как хранилище ресурсов файла или содержали информацию о типе файла (например, лицензия к звуковому файлу и прочее). Сейчас Apple рекомендует помещать ресурсы в отдельные файлы, а типы файлов определять по расширениям, но возможность использования разветвлений всё равно остаётся.

Замечание 2. Программа для отслеживания обращений к файловой системе – FileMonitor, упомянутая нами ранее, – не делает различий между обращениями к файлам или потокам.

Замечание 3. В альтернативных потоках могут размещаться любые файлы, в том числе и исполняемые. Например возможна запись файла приложения «Калькулятор» (calc.exe) в альтернативный поток «aaa.exe» файла test.txt:

```
type calc.exe>test.txt:aaa.exe
```

Запуск исполняемого файла из альтернативного потока возможен командой:

```
start .\test.txt:aaa.exe
```

Замечание 4. PowerShell умеет работать с альтернативными потоками – создавать, обнаруживать, выводить их содержимое и даже удалять.

5.3.7.3.2. Резюме по ФС NTFS

Для файловой системы **NTFS** теоретически файл – единица информации, имеющая номер в Master File Table (**MFT**, общей таблице файлов), однако на практике всё же основной атрибут файла – его имя.

Сохранение информации о файлах обеспечивает файловая система. Правила именования файлов, способы доступа к данным файла, структура его данных зависят как от типа используемой файловой системы, так и от типа файла. Файловая система берёт на себя размещение информации файла на диске, сохранение информации об этом в таблице файлов диска и обеспечение доступа к этим данным.

Кроме имени, каждый файл может иметь и другие свойства:

- расширение имени файла, характеризующее его тип [280] (символы после последней точки);
- атрибуты (архивный, только чтение, скрытый, шифрованный и прочие);
- время создания, изменения, последнего открытия;
- права доступа (безопасность);
- метаданные (название, тема, ключевые слова, авторы и прочие).

Каждый логический раздел жёсткого диска может быть отформатирован в той или иной файловой системе. Например, основной файловой системой ОС **Windows Vista, 7, 8** является **NTFS**. **Windows XP** может также не только работать с информацией, хранящейся на разделах диска с файловой системой **FAT** и **FAT 32**, но и загружаться с них.

Работа с ФС NTFS возможна даже в ОС DOS, для этого надо воспользоваться драйвером от Марка Руссиновича NTFSDOS.

[280] Справочник типов см. на http://open-file.ru/, также в ОС Linux есть командная утилита file, которая имеет сигнатуры большинства типа файлов и может определить тип файла по содержимому, а не по расширению. Например:

```
$ file picture_5.15.jpg
picture_5.15.jpg: JPEG image data, JFIF standard 1.01
```

5.3.7.4. exFAT

По мере роста популярности относительно быстрых, объёмных и недорогих флэш-накопителей, последние стали повсеместно использоваться совместно с вышеописанными ФС FAT и NTFS. Далее, возникла проблема многократных перезаписей одного и того же сектора. Неизбежно часто перезаписываемые области ФС, например таблица FAT или её копия, приходятся на некоторую фиксированную область накопителя расположение которой остаётся неизменной со временем. Как следствие, ячейки памяти неизбежно и необратимо «изнашиваются» после каждой операции записи, а после определённого количества раз и вовсе становятся непригодными для дальнейшего использования. Новая файловая система, кроме прочих приносимых её улучшений, должна была оказаться лишённой последнего недостатка, либо значительно снизить его эффект.

Так, в ноябре 2006 года появилась ФС exFAT (*от англ. Extended FAT – «расширенная FAT»*), иногда её ещё называют FAT64. К сожалению, данная файловая система не получила широкого распространения среди сторонников свободного ПО, поскольку она проприетарная. То есть, владелец патента на данную ФС в ряде случаев вправе требовать денежных отчислений. Ни о каком свободном использовании не может быть и речи.

Однако, существует свободный драйвер ФС exFAT в виде патча для ядра Linux, поддерживающий только чтение. Для чтения и записи существует другой драйвер, работающий через FUSE (Filesystem in Userspace). Кроме того, в августе 2013 года Samsung опубликовала драйвер для ядра Linux под лицензией GPL.

Для работы с ФС exFAT в Windows XP требуется Service Pack 2 (или 3) с обновлением KB955704. Windows Vista с Service Pack 1 и более поздние версии поддерживают exFAT, более ранее – нет.

5.3.7.5. ext2, ext3, ext4

Предложенная в январе 1993 года Реми Кардом (англ. Rémy Card) «вторая расширенная файловая система (ФС)» (Second Extended File System, сокращённо ext2 или ext2fs) взамен существовавшей тогда ФС ext, а сейчас de facto ушедшей в историю, отметила четверть вековой юбилей своего использования и сегодня фактически является не уступающим по производительности эталоном, с которым в «мире UNIX и Linux» часто сравнивают все другие файловые системы.

Первое русскоязычное описание ФС ext2 с примерами кодов под авторством Владимира Мешкова можно найти в журнале «Системный администратор» за 2003 год [11]. Эта система и её последующие версии доступны по умолчанию большинству пользователей в набирающей популярность ОС Linux.

Вполне возможно, что файловые системы семейства ext покажутся вам относительно простыми, продуманными и по сему привлекательными.

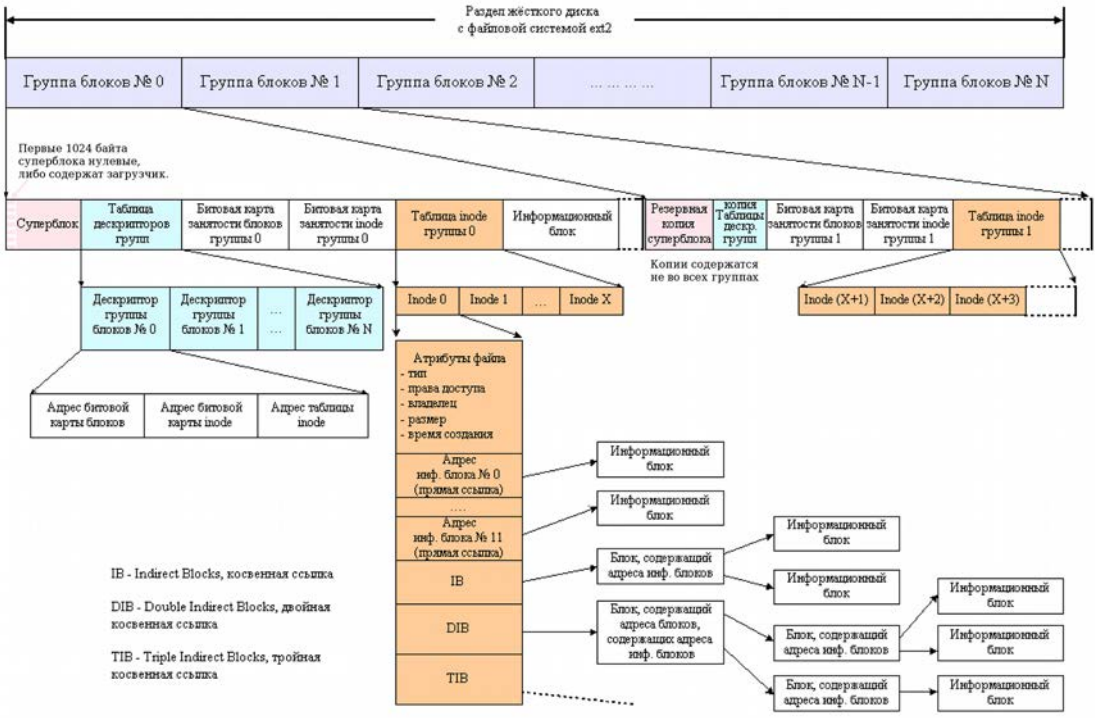


Рисунок 5.15. Структура файловой системы ext2

Как и в любой файловой системе UNIX, в составе файловой системы ext2 можно выделить следующие составляющие [11, 31, 35, 36]:

- блок и группа блоков;
- информационный узел (information node, inode);
- суперблок (superblock) и его резервные копии;
- директория.

Для эффективного доступа к которым в ФС ext2-4 дополнительно создаются и используются:

- таблица дескрипторов групп (и её копии);
- битовая карта занятости inode'ов в каждой группе блоков;
- битовая карта занятости блоков в каждой группе блоков;
- зарезервированные блоки (в группах, где имеются копии суперблока).

Блоки и группы блоков

Всё пространство раздела диска разбивается на блоки фиксированного размера: 1024, 2048 или 4096 байт[281]. По задумке все три числа были взяты кратными размеру сектора – 512 байт. В начале 1990-х такой размер сектора использовался как на дискетах, так и на жёстких дисках. Сектора по 4096 байт (используемые абсолютно всеми жёсткими дисками большого объёма, начиная примерно с 2009 года) не только не существовали, но и не предвиделись. В наши дни такие большие сектора нисколько не

[281]Предполагается, что в ext4 не включена поддержка 64-битных значений для ряда параметров ФС.

усложняют жизнь, при создании ФС размер одного блока выбирается равным размеру одного сектора. Также, среднестатистический размер мелких файлов типичной системы обычно оказывается близок к 4 КБ, что позволяет более оптимально использовать дисковое пространство за счёт минимизации неиспользуемого пространства в конце завершающих блоков с данными.

Размер логического блока указывается при создании файловой системы (с помощью ключа -b), либо определяется эвристически от размера файловой системы и ожидаемого предназначения ФС (типа использования).

Все блоки имеют порядковые номера. Поскольку ФС разрабатывалась предпочтительно для использования на традиционных жёстких дисках, то с целью уменьшения фрагментации и количества перемещений считывающих головок при чтении больших массивов данных блоки объединяются в группы.

Группа блоков № 0	Группа блоков № 1	Группа блоков № 2	...	Группа блоков № $N-1$	Группа блоков № N

Рисунок 5.16. Раздел жёсткого диска с ФС ext2. Нумерация секторов слева направо

Начиная с ФС ext4, появилась новая возможность группировки нескольких соседних групп блоков в укрупнённые виртуальные группы блоков (flexible block groups), данная возможность включается опцией «flex_bg» при создании ФС, а число группирующихся блоков задаётся опцией -G и должно равняться 2^n.

Практически все группы блоков имеют одинаковый формат. В каждой группе, помимо информационных блоков, хранится информация о занятости блоков и inode группы в виде битовой карты (*Block Bitmap* и *Inode Bitmap*). В состав 0-й группы блоков и некоторых других входят также суперблок, таблица дескрипторов групп и резервные блоки.

Суперблок

Суперблок – это основной элемент файловой системы ext2-4, представляющих из себя некоторую таблицу различных значений, определяющих параметры ФС и местоположение её составных частей. Если не считать смещение в 1024 байта от начала файловой системы, то он идёт самым первым по порядку (см. рис. 5.15). Смещение необходимо для возможности размещения в самом начале кода загрузчика ОС. Если размер логического блока ФС равен 1024, то суперблок фактически начинается и оказывается во втором блоке (с порядковым номером 1, так как блоки нумеруются начиная с нуля).

В зависимости от версии ФС суперблоки несколько различаются. Первые поля структуры суперблока (назовём их базовыми) одинаковы для разных версий ФС, их общий суммарный размер меньше 1024 байт, поэтому блок с суперблоком в конце заполняется нулями. Если же используется дополнение-модификация суперблока EXT4_DYNAMIC_REV, то тогда после базовой вместо нулей располагается таблица расширенных параметров, но в любом случае весь суперблок не превышает по размеру 1024 байта.

Если размер логического блока ФС оказывается таковым, что после размещения у нём суперблока остаётся свободное место, то оно заполняется нулями.

В базовой части суперблока содержится следующая информацию о файловой системе (список неполный):

- общее число блоков и inode в файловой системе;
- число свободных блоков и inode в файловой системе;
- размер логического блока файловой системы;
- количество блоков и inode в группе;
- размер inode;
- идентификатор файловой системы;
- номер первого блока данных;
- как часто (при каких условиях) необходимо делать проверку ФС;
- время создания ФС и т.д.

На практике вывести всю информацию, содержащуюся в суперблоке (например для раздела /dev/sda1), можно с помощью команды:

```
# dumpe2fs /dev/sda1 -h
dumpe2fs 1.41.12 (17-May-2010)
Filesystem volume name:   /
Last mounted on:          /
Filesystem UUID:          ff317125-1094-b60a-6c29-5d311b20e813
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index file-
type needs_recovery extent flex_bg sparse_super large_file huge_file
uninit_bg dir_nlink extra_isize
Filesystem flags:         signed_directory_hash
Default mount options:    (none)        Filesystem state:         clean
Errors behavior:          Continue      Filesystem OS type:       Linux
Inode count:              3276800
Block count:              13107200
Reserved block count:     655360
Free blocks:              11012226      Free inodes:              3051679
First block:              0             Block size:               4096
Fragment size:            4096          Reserved GDT blocks:      1020
Blocks per group:         32768         Fragments per group:      32768
Inodes per group:         8192          Inode blocks per group:   512
Flex block group size:    16
Filesystem created:       Sun May  1 13:55:30 2013
Last mount time:          Fri May 31 09:57:54 2013
Last write time:          Sun May  5 10:02:16 2013
Mount count:              20            Maximum mount count:      28
Last checked:             Sun May  5 10:02:16 2013
Check interval:           15552000 (6 months)
Next check after:         Fri Nov  1 10:02:16 2013
Lifetime writes:          57 GB
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11          Inode size:               256
Required extra isize:     28          Desired extra isize:      28
Journal inode:            8
First orphan inode:       1193396
Default directory hash:   half_md4
Directory Hash Seed:      eb7e52af-a644-430a-9f96-10913a257c34
Journal backup:           inode blocks
Journal features:         journal_incompat_revoke
```

```
Journal size:             128M
Journal length:           32768
Journal sequence:         0x0001610d
Journal start:            1
```

Несомненно, в суперблоке содержится довольно важная информация по файловой системе. От целостности суперблока напрямую зависит работоспособность файловой системы. Чтобы эта информация не потерялась и не произошло ситуации, что из-за сбоя одного блока стал недоступен весь файловый раздел, она дублируются в нескольких местах на этапе создания (формирования) файловой системы утилитой mkfs.

```
...
Superblock backups stored on blocks:
   8193, 24577, 40961, 57345, 73729, 204801, 221185, ... и т. д.
...
```

Поскольку информация содержащаяся в «дескрипторе группы блоков», следующего за суперблоком также важна, то она также дублируется, следуя за копиями суперблока.

Копии суперблока

В ранних (под)версиях ФС ext2 копия суперблока (и дескриптор группы блоков) содержались в каждой группе блоков, но после выяснилось, что указанная избыточность дорого обходится, поэтому от столь большого числа копий отказались в пользу хранения копий лишь в некоторых группах блоков. Для этого была введена функция (флаг) sparce_super (для ФС ext2 – EXT2_FEATURE_RO_COMPAT_SPARCE_SUPER, для ФС ext3/4 – EXT3_FEATURE_RO_COMPAT_SPARCE_SUPER), включённая по умолчанию при создании ФС. Алгоритм расчёта, в каких именно группах размещать копию (при наличии флага), а в каких нет, довольно прост. Суперблок (и дескриптор группы блоков) находится в 0-й группе блоков, а его копии в группах блоков, чьи номера равны степеням чисел 1, 3, 5 и 7, то есть, 1^1, 3^1, 5^1 ,7^1, 3^2=9, 5^2=25, 3^3=27 и 7^2=49, 5^3=125 и т. д. Если флаг не установлен (функция не включена), то копии делаются во всех группах. [33]

Для того чтобы в дальнейшем использовать копии, например при восстановлении ФС в случае потери данных из нулевой группы блоков, важный подготовительный шаг – это понять как пересчитать номера групп блоков содержащих копии суперблока в номера блоков (чья нумерация сквозная по всему диску), поскольку различные программы требуют указания именно блока, где находится копия суперблока. Не имея в своём распоряжение суперблока или его копии сделать это непросто, так как в суперблоке (и его копиях) содержится необходимая для расчёта информация – размер блока и число блоков в группе. Чем-то похоже на ситуацию «ключ от сейфа в сейфе». От полного перебора всех возможных вариантов обычно спасает то, что многие администраторы при создании ФС пользуются параметрами, предлагаемыми по умолчанию, что значительно уменьшает число возможных мест для поиска копий суперблока на диске. Если предположить, что блоков в группе n=8192, то копии суперблока будут в блоках с номерами, рассчитываемыми по формуле $m*n+1$, где m – номер группы блоков с копией (см. алгоритм получения m выше), т. е.: 1*8192+1=8193, 3*8192+1=24577, 5*8192+1=40961, 7*8192+1=57345, 3^2*8192+1=73729, 5^2*8192+1=204801 и т. д.

На практике вычисленных значений номеров блоков недостаточно, поскольку для чтения копий суперблока с диска дополнительно потребуется знать размер блока для

того чтобы правильно вычислить смещение в байтах от начала диска. В реальности, количество различных вариантов для размера блока конечно, что делает оправданным их перебор при поиске копий.

Совет. После создания ФС лучше всего выполнить команду

```
# dumpe2fs /dev/sdaX
...
Group 9: (Blocks 73729-81920)
  Backup superblock at 73729, Group descriptors at 73730-73730
  Reserved GDT blocks at 73731-73986
  Block bitmap at 74279 (+550), Inode bitmap at 74280 (+551)
  Inode table at 73987-74242 (+258)
  0 free blocks, 2048 free inodes, 0 directories
  Free blocks:
  Free inodes: 18433-20480
...
```

а результат её вывода сохранить на другом компьютере или распечатать на бумаге и хранить в отдельной папочке на случай последующего ручного восстановления данных.

Таблица дескрипторов групп (и её копии), резервные блоки, битовые карты

Каждая группа блоков описывается в таблице дескрипторов групп (дескрипторе группы блоков), – структуре, которая содержит информацию по каждой группе блоков: адреса битовых карт занятости блоков и inode'ов, таблицы inode'ов. Для хранения структуры (вместе с её копиями), а также следующими за ней резервными блоками используются те же группы блоков, что используются для хранения суперблока или его копий.

Поскольку эти все элементы ФС не будут использоваться в данной работе, то подробно они не рассматриваются. Для простоты изображения на рисунке 1 резервные блоки даже не отмечены.

В каждой группе блоков имеются свои битовая карта занятости inode'ов и битовая карта занятости блоков. При использовании опций «flex_bg» и «meta_bg» расположение битовых карт внутри укрупнённых групп однозначно не фиксировано.

По умолчанию размер битовой карты – один блок. Каждый бит карты обозначает состояние блока (или inode'а). Если бит установлен (1), то блок (inode) занят, если сброшен (0) – блок (inode) свободен. Первому блоку группы (inode'у) соответствует нулевой бит карты, второму – первый бит и т. д. Отличие inode'ов от блоков заключается в том, что они меньшего размера, поэтому inode'ы, находящиеся в пределах одной группы, собраны в таблицу. В битовой карте занятости inode'ов группы каждый бит аналогично характеризует состояние элемента в таблице inode группы.

Так, если размер блока выбран 1024, то в группе блоков будет содержаться 1024*8=8192 блоков. Поскольку inode'ов чаще требуется меньше числа блоков, их битовая карта оказывается меньше, а блок содержащий карту до конца заполняется нулями.

Информационный узел (inode)

Базовым понятием файловой системы является информационный узел, information node, inode, i-узел или i-node. Каждый inode, как и блок, имеет порядковый

номер, уникальный в пределах файловой системы, и включает информацию только об одном файле.

Следует отметить, что термин «файл» в контексте ФС может пониматься многозначно:

- на физическом уровне, как описание расположения данных внутри ФС (номера занимаемых блоков, смещения отдельных частей и пр.);
- на логическом уровне организации ФС, как его полное имя состоящие из иерархии вложенных директорий и имени файла;
- с точки зрения ОС как псевдофайл (сокет, канал, ссылка, блочное устройство, символьное устройств, директория);
- на пользовательском уровне, как содержимое файла.

inode содержит информацию о «физическом» расположении данных. Связь логического имени файла с inode'ом производится с помощью жёстких ссылок. Жёстких ссылок (как и логических имён) у одного файла в ФС ext2-4 может быть несколько. Типичный пример этому есть особый тип файла – директория, в которую можно попасть как сверху, например по имени «dir1» (`cd dir1`), так и снизу, например из низлежащих поддиректорий, переходом в директорию с именем «`..`» (`cd ..`).

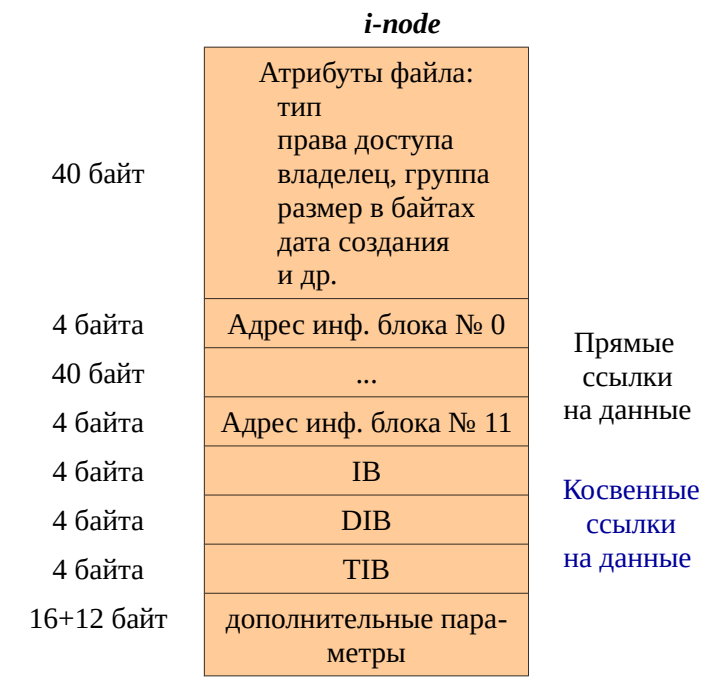


Рисунок 5.17. Приблизительная структура индексного дескриптора (первые 128 байт, общие для ФС ext2-4)

Посмотреть номера inode'ов у логических имён файлов (и директорий) можно с помощью команды «ls -i», либо с помощью утилиты «stat».

Внутреннее устройства *inode* следующее, это специальная структура (таблица), которая содержит информацию об атрибутах и физическом расположении файла или, одним словом, метаданные. (См. рис. 5.17) Её размер для ext2 – 128 байт, для ext4 – 256 байт. Причём увеличение размера происходит за счёт использования тех же структур, что и для inode ФС ext2, в первых 128 байтах, а всё дополнительное добавляется в конец. Фактически используется 156 байт, а оставшиеся 100 байт пустые и могут использоваться для хранения расширенных атрибутов, также, во время создания ФС размер inode может быть задан и большего размера, – по размеру блока ФС, хотя это не очень эффективно.

Таблица 5.13. Уточнённая структура inode, записанная в виде справочной таблицы

Смещение 16сс	Начало 10сс	Размер (бит)	Конец (последний байт)	Название поля	Описание поля
0x0	0	__le16	1	i_mode	Права доступа изменяемые командой chmod и тип файлового объекта
0x2	2	__le16	3	i_uid	Младшие 16 разряда поля Owner UID
0x4	4	__le32	7	i_size_lo	Младшие 32 разряда размера в байтах
0x8	8	__le32	11	i_atime	Последнее время доступа (access time), в секундах с 1 января 1970. (см. ls -lu) Однако, если установлен флаг EA_INODE, то в данном inode хранится значение расширенных атрибутов (extended attribute value), а это поле содержит контрольную сумму этого значения.
0xC	12	__le32	15	i_ctime	Последнее время изменения данных inode (change time), в секундах с 1 января 1970. (см. ls -lc, легко запомнить как change = chmod, chown, ch...) Однако, если установлен флаг EA_INODE, то в данном inode хранятся значения расширенных атрибутов (extended attribute value), а данное поле содержит младшие 32 разряда счётчика количества ссылок (attribute value's reference count).
0x10	16	__le32	19	i_mtime	Время последнего изменения содержимого файла (modification time), в секундах с 1 января 1970. (см. ls -l) Однако, если установлен флаг EA_INODE, то данный inode хранит расширенные атрибуты (extended attribute value) и это поле содержит значение счётчика количества inode'ов которые владеют данным inode с расширенным атрибутом.
0x14	20	__le32	23	i_dtime	Время удаления (Deletion Time) в секундах с 1 января 1970 или см.**
0x18	24	__le16	25	i_gid	Младшие 16 разрядов поля GID
0x1A	26	__le16	27	i_links_count	Счётчик количества жёстких ссылок ведущих на данный inode. В обычных условиях, ФС ext4 не позволяет записывать в это поле значение более чем 65 000 жёстких ссылок. Это справедливо как для обычных файлов, так и для директорий, что означает, что в последней не может быть более чем 64 998 поддиректорий (каждая запись «..» в каждой поддиректории есть жёсткая ссылка и она считается, также считается и запись «.»

					самой директории. (команда ls -a позволяет увидеть записи, чьи имена начинаются с точки) Если включена поддержка опции DIR_NLINK, то ФС ext4 поддерживает более чем 64 998 поддиректорий за счёт установки данного поля в 1, означающее, что число жёстких ссылок ведущих на данный inode неизвестно.
0x1C	28	__le32	31	i_blocks_lo	Младшие 32 разряда счётчика количества блоков. Если флаг huge_file не установлен в ФС, то файл занимает указанное в данном поле число 512-байтных блоков. Если же флаг huge_file установлен, а флаг EXT4_HUGE_FILE_FL внутри поля i_flags данного inode не установлен, то количество 512-байтовых блоков занятых файлом высчитывается по формуле: (i_blocks_hi << 32) + i_blocks_lo. Если оба флага huge_file и EXT4_HUGE_FILE_FL IS установлены, то число 512-байтных блоков, занимаемых файлом высчитывается по формуле (i_blocks_hi << 32) + i_blocks_lo.
0x20	32	__le32	35	i_flags	Флаги и расширенные атрибуты
0x24	36	__le32*	39	l_i_version*	Версия inode.* Однако, если установлен флаг EA_INODE, то в inode хранятся расширенные атрибуты, а данное поле хранит старшие 32 разряда счётчика ссылок на него.
0x28	40	__le32	43	i_block	Прямая ссылка на информационный блок № 0
0x2C	44	11 * __le32	87		Прямые ссылки на инф. блоки №№ 1-11.
0x58	88	__le32	91		Косвенная ссылка IB
0x5C	92	__le32	95		Двойная косвенная ссылка DIB
0x60	96	__le32	99		Тройная косвенная ссылка TIB
0x64	100	__le32	103	i_generation	Версия файла (для NFS)
0x68	104	__le32	107	i_file_acl_lo	Младшие 32 разряда номера информационного блока, хранящего расширенные атрибуты (extended attribute block). Буквы «acl» закрепились в названии лишь потому, что ACL- списки были первыми из всевозможных расширенных атрибутов для которых использовался этот дополнительный блок.
0x6C	108	__le32	111	i_size_high / i_dir_acl	Старшие 32 разряда для размера файла или директории. В ФС ext2/3 это поле имеет другое имя «i_dir_acl», обычно равно нулю и никогда не использовалось.

0x70	112	__le32	115	i_obso_faddr	(Устаревшее поле) адрес фрагмента.
0x74	116	__le16*	117	l_i_blocks_high*	*Старшие 16 разрядов счётчика числа блоков. (Формулу объединения см. в поле i_blocks_lo.)
0x76	118	__le16*	119	l_i_file_acl_high*	*Старшие 16 разрядов блока с расширенными атрибутами (обычно в нём хранятся ACL списки). Есть разные исключения, подробнее см. параграф Extended Attributes в [4]
0x78	120	__le16*	121	l_i_uid_high*	*Старшие 16 разрядов UID.
0x7A	122	__le16*	123	l_i_gid_high*	*Старшие 16 разрядов GID.
0x7C	124	__le16*	125	l_i_checksum_lo*	*Младшие 16 разрядов контрольной суммы inode.
0x7E	126	__le16*	127	l_i_reserved*	*Поле не используется.
------ невидимая граница разделения базовой и расширенной частей inode ------					
0x80	128	__le16	129	i_extra_isize	Начиная с этого поля начинается расширенная часть inode. В данном поле хранится её размер. Обычно он равен 128 байт.
0x82	130	__le16	131	i_checksum_hi	Старшие 16 разрядов контрольной суммы для inode.
0x84	132	__le32	135	i_ctime_extra	Дополнительные биты для хранения уточнённого значения поля «change time», с точностью до долей секунды.
0x88	136	__le32	139	i_mtime_extra	Дополнительные биты для хранения уточнённого значения поля «modification time», с точностью до долей секунды.
0x8C	140	__le32	143	i_atime_extra	Дополнительные биты для хранения уточнённого значения поля «access time», с точностью до долей секунды.
0x90	144	__le32	147	i_crtime	Время создания файла (creation time) в секундах с 1 января 1970.
0x94	148	__le32	151	i_crtime_extra	Дополнительные биты для хранения уточнённого значения поля «creation time», с точностью до долей секунды.
0x98	152	__le32	155	i_version_hi	Старшие 32 разряда поля «Версия».
0x9C	156	__le32	159	i_projid	Идентификатор проекта (Project ID).
0xA0	160	?	?	?	Размер указанного поля и его конец определяется размером расширенной части (см. поле i_extra_isize). Формат хранимой здесь информации оговаривается отдельно. Может использоваться для хранения расширенных атрибутов.

* Данное поле ФС зависит от ОС, в данном варианте представлено заполнение для ОС Linux.

Атрибутами файла являются его тип (обычный файл, каталог и т. д.), права доступа к нему (меняемые командой chmod), идентификаторы владельца и группы, размер,

время создания. Информация о физическом расположении зависит от размеров файла и представляет собой последовательность абсолютных номеров блоков, содержащих данные файла. Эта информация представляет собой последовательность 32-битных номеров блоков, содержащих данные файла. Первые 12 номеров – это прямые ссылки на информационные блоки (direct blocks number). 13-й номер является косвенной ссылкой (indirect blocks number). В нём находится адрес блока, в котором хранятся адреса информационных блоков. 14-й номер – двойная косвенная ссылка (double blocks number), 15-й номер – тройная косвенная ссылка (triple blocks number). (см. рис. 5.18)

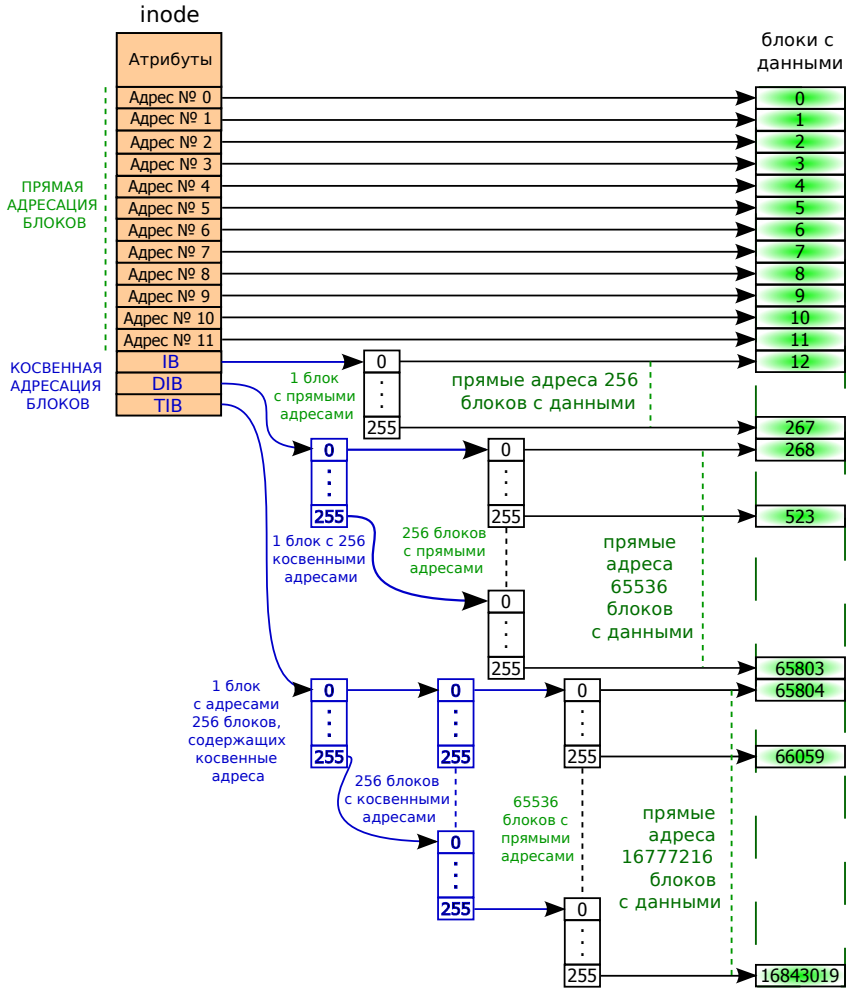


Рисунок 5.18. Структура ссылок на номера блоков в inode

Замечание. Поскольку адресация блоков 32-битная, на хранение адреса отводится 4 байта. В приведённом выше примере используются блоки размером 1024 байта, в которые помещается 1024 / 4 = 256 адресов. При использовании блоков большего размера нумерация на рисунке 4 изменится, поскольку при косвенной адресации один блок будет содержать больше 256 адресов. Соответственно, изменится и максимально адресуемое по такой схеме пространство внутри файла, то есть, максимальный теоретический размер файла.

Как было отмечено выше, «имя файла» в состав данных, хранимых в inode не входит, установление соответствия между именами файлов и порядковыми номерами inode выполняется с помощью жёстких ссылок, которые сами «хранятся» в директориях (каталогах).

Содержимое inode выглядит примерно так:

```
$ stat /etc/passwd
  Файл: «/etc/passwd»
  Размер: 2523       Блоков: 8          Блок В/В: 4096   обычный файл
Устройство: 813h/2065d  Inode: 1351224    Ссылки: 1
Доступ: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Контекст: system_u:object_r:etc_t:s0
Доступ: 2020-11-19 12:10:05.412397175 +0300
Модифицирован: 2020-11-19 12:10:05.412397175 +0300
Изменён: 2020-11-19 12:10:29.769023774 +0300
 Создан: -
# lde -i 910462 /dev/sda2
User requested autodetect filesystem. Checking device . . .
Found ext2fs on device.
------------------------------------------------------------------------
INODE: 910462 (0x000DE47E)
lrwxrwxrwx  root     root            19 Wed Jun 29 01:33:04 2011
TYPE:                  symbolic link
LINKS:                 1
MODEFLAGS.MODE:        012.0777
SIZE:                  19
BLOCK COUNT:           0
UID:                   00000 (root)
GID:                   00000 (root)
ACCESS TIME:           Sat May 12 23:42:40 2012
CREATION TIME:         Sun May 27 18:02:27 2012
MODIFICATION TIME:     Wed Jun 29 01:33:04 2011
DELETION TIME:         Thu Jan  1 03:00:00 1970
DIRECT BLOCKS:        0x692F2E2E 0x2E74696E 0x69662F64 0x62747372 0x00746F6F
INDIRECT BLOCK:
DOUBLE INDIRECT BLOCK:
TRIPLE INDIRECT BLOCK:
```

Подробнее о структуре inode см. [33], а также Лабораторную работу № 2 (см. стр. 548).

Метки времени в inode

В базовой части индексного дескриптора хранится 4 метки времени: *i_atime*, *i_ctime*, *i_mtime*, *i_dtime*. Префикс «*i_*» (от inode) для краткости и удобства может опускаться. Формат хранения – секунды с начала эпохи Unix в формате 32 разрядное знаковое целое. В современных ПК этому формату обычно соответствует тип int языка Си. Первые три из них можно просмотреть от обычного пользователя командами ls -lu, ls -lc, ls -l, соответственно, или все три сразу – через stat (в выводе выше они помечены розовым оттенком). Отображаемые значения пересчитываются и отображаются для текущей временной зоны.

Точность хранения времени в ФС ext2/3 (при размере inode = 128 байт) – до секунды.

Команда ls (с соответствующими ключами) отображает его урезанно: если файл этого года, – без секунд, а если он из прошлых лет, то и без часов и минут.

```
$ ls -lu /etc/passwd
-rw-r--r--. 1 root root 2523 ноя 19 12:50 /etc/passwd
```

Если в дополнение к базовой части inode используется расширенная, то за счёт неё можно увидеть дробную часть секунд, отображаемую после запятой (точки) с точностью до 1 наносекунды (10^{-9}). Если уточняющей информации нет, то в соответствующих полях вывода команды stat вы увидите нули.

```
...
Доступ: 2020-12-20 12:10:05.000000000 +0300
...
```

Временная метка dtime для просмотра стандартными утилитами ни пользователю, ни администратору недоступна. Метка crtime из расширенной части пока ещё также в большинстве дистрибутивов ОС Linux недоступна для просмотра. Метку crtime (creation time) иногда называют как в некоторых других ФС – btime (birth time), от этого её суть не меняется, но так меньше вероятность перепутать внешне схожие ctime и crtime. Расширенная часть повышенной точности для метки deletion time не предусмотрена.

Естественно, с правами администратора, все 5 меток (9 полей) можно увидеть через утилиту debugfs и при «сыром» чтении диска, чему собственно и посвящено большинство практических заданий данной работы. Для удобства выпишем отдельно все 9 полей индексного дескриптора, определяющие значения временных меток, и их смещения:

0x08	*i_atime*
0x0C	*i_ctime*
0x10	*i_mtime*
0x14	*i_dtime*

0x84	*i_ctime_extra*
0x88	*i_mtime_extra*
0x8C	*i_atime_extra*
0x90	*i_crtime*
0x94	*i_crtime_extra*

Особенности формата временных меток

Метки atime, ctime, mtime, dtime и crtime – 32 разрядные целые, хранящие секунды с начала эпохи Unix (т. е. с 1970-01-01 00:00:00 GMT). Из этого вытекает первая особенность, известная как возможность переполнения этих полей (так называемая «проблема 2038 года», «Unix Millennium bug», «Y2K38», или как было бы правильнее её назвать – «1901/2038», поскольку при уменьшении значений времени 1901 года также наступит схожая ситуация). Математика у данной особенности следующая: в формате дополнительного кода поле памяти размером в 32 двоичных разряда позволяет записать числа от −2147483648 (-2^{31}) до +2147483647 ($2^{31}-1$). Если это будут секунды, то не углубляясь в особенности формата UTC, временных зон и других систем измерения времени (поскольку это сейчас не существенно), определим в более привычном формате некоторые моменты времени для интересующего нас числа секунд:

```
$ TZ="GMT" date --date='@-2147483648'
Пт дек 13 20:45:52 GMT 1901
$ TZ="GMT" date --date='@0'
Чт янв  1 00:00:00 GMT 1970
$ TZ="GMT" date --date='@2147483647'
Вт янв 19 03:14:07 GMT 2038
$ TZ="GMT" date --date='@2147483648'
Вт янв 19 03:14:08 GMT 2038
```

Поскольку программа date и система в которой осуществлялся её запуск являются 64-битными

```
$ file `which date`
/usr/bin/date: ELF 64-bit LSB executable, x86-64 ...
$ uname -r -m -p
3.10.0-1062.el7.x86_64 x86_64 x86_64
```

проблемы «1901/2038» нет и время «идёт» как и ожидается большинством читателей, т.е. «последовательно», но в 32-битных системах, а также в указанных полях ФС ext2/3/4 будет большой «скачок» во времени или ситуация «переполнения» в двоичных разрядах. А именно, если увеличить на единицу, комбинацию двоичных разрядов для целого числа 2147483647 (в 2038 году)

```
01111111 11111111 11111111 11111111
```

как если бы это было беззнаковое целое (так как процессор работает именно так), то получится

```
10000000 00000000 00000000 0000000
```

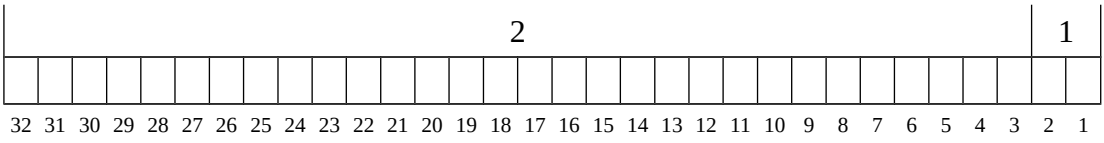
Замечание. В памяти в формате LitteEndian запись будет в ином порядке, но это не существенно.

которому соответствует −2147483648 секунд и дата в 1901 году, которую более точно вы можете увидеть выше. Во что подобное «скачкообразное» изменение метки времени выльется на практике зависит от того, насколько дата и время важны в каждой конкретной программе.

Рассмотрим подробнее, что же изменилось в трактовании трёх временных меток atime, ctime, mtime индексного дескриптора с появлением у него расширенной части, кроме появления ещё одной – пятой метки «crtime».

За счёт трёх дополнительных 32 битных полей *i_atime_extra*, *i_ctime_extra*, *i_mtime_extra* под соответствующие метки теперь стало отводиться 64 бита данных.[4] Поле метки dtime не получило соответствующего расширения, оставшись 32-битным, зато оно получило «новые дополнительные функциональные обязанности» которые будут рассмотрены подробнее ниже.

Дополнительные 32 разряда (в их двоичном представлении) поделили на две части



MSB (старшие разряды) (младшие разряды) LSB

и стали использовать следующим образом: два младших разряда расширенной записи отвели под расширение числа секунд (обозначим их через *i_Xtime_extra_2_bits*, где *X* – зависит от метки), а оставшиеся старшие 30 разрядов отвели для хранения наносекунд, тем самым обеспечив точность времени в секундах до 9 знаков после запятой. Таким образом, это позволило хранить в указанных трёх метках времени даты, вплоть до мая 2446 года.[33]

Замечание 1. Русская страница wikipedia (https://ru.wikipedia.org/wiki/Ext4, как было проверено в апреле 2020 года, ошибочно пишет, что временной диапазон дат у ФС ext4 увеличился до 25 апреля 2514 года. Обидно, что многие русскоязычные сайты копируют и множат эту глупость. И правда, читатель, не верь написанному, проверяй! В теоретической части работы разберёмся в деталях расширенного формата временной метки, убедимся, что это не так и посчитаем истинное теоретическое значение конца временного диапазона. После чего подготовим лабораторный стенд и при проведении работы проверим на практике теоретическую часть.

Грубо оценим, что даёт расширяющее поле. Касаемо наносекунд: 30 двоичных полей это 2^{30} комбинаций или $2^{10}*2^{10}*2^{10}$. Помним, что $2^{10} = 1024 \approx 1000 = 10^3$, то есть милли-, микро- и нано- точность. Точная проверка: 30 разрядов => $2^{30} = 1\,073\,741\,824$ комбинаций позволит осуществить беззнаковое хранение чисел от 000 000 000 до 999 999 999 и трактовать их как наносекунды после запятой (со смещением или нет – не важно). Лишние 73 741 824 комбинации – неизбежные траты.

Два младших бита дают $2^2 = 4$ комбинации. Текущий адресуемый диапазон дат с использованием 2^{32} секунд оказывается грубо с 1901 по 2038 годы = 136 лет будет увеличен в 4 раза. Таким образом, можно дополнительно получить ещё три диапазона по 136 лет, и, если добавить эти 136*3=408 лет к верхней границе, то получится 2446 год.

Появившиеся в расширенной части два поля crtime и crtime_extra позволяют аналогичным образом совместно получить новую, пятую метку времени – «creation time».

Замечание 2. Относительно временной метки «creation time» интересно сообщить следующие факты, чтобы читателям было о чём подумать на досуге. Поддержка ФС ext4 в ядрах GPL/Linux для широких масс (не считая патчей для разработчиков) появилась между 20 сентября 2006 года (выход версии 2.6.18 на сайте kernel.org – в которой поддержки не было) и 29 ноября 2006 года – датой выхода первого ядра с поддержкой (версии 2.6.19). Далее происходило усовершенствование кода отвечающего за взаимодействие с ФС ext4 с целью его улучшения и последующего отказа от приклеенного к нему изначально ярлыка «экспериментальный, развивающийся» (development). Проследим за развитием. Поиск среди исходных кодов ядра в директориях fs/ext4 по вхождению ключевого слова «crtime» даёт результат лишь в версии 2.6.23 от октября 2007 года. В более ранней версии 2.6.22 от июля 2007 года он ничего не находит. Примерно через год, 10 октября 2008 года ожидаемое случилось, поддержка ФС ext4 потеряла суффикс «dev» в названии части кода https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=03010a3350301baac2154fa66de925ae2981b7e3 и через некоторое время, (по информации https://ext4.wiki.kernel.org/index.php/Ext4_Howto) стала «стабильной» начиная с версии 2.6.28 (конец 2008 – начало 2009 г.г.).

Годом позже, в январе 2010 компания Google заявила о переводе своих старых систем на ФС ext4, а в конце 2010, то есть примерно ещё через год, сообщила о планируемом использовании данной ФС для ОС Android 2.3. В мире началось активное использование ФС ext4.

5 лет спустя на одном из форумов[282] был задан вопрос из которого обнаружилось, что в одном из базовых дистрибутивов Linux, – Debian, невозможно с помощью штатной утилиты stat посмотреть время crtime (в то время как при прямом доступе к ФС с правами администратора, через debugfs – возможно). Проблема быстро не решилась, через пару лет, в 2017 году появилась информация о том, что в ядрах, начиная с версии 4.11, это возможно с помощью нового системного вызова statx(), но совместно с библиотекой glibc версии >= 2.28 (которая вышла позже, в августе 2018). Причём обёртка для запуска statx() была добавлена в версии GNU coreutils 8.31 (выпущенной в марте 2019), а чтобы использовать и просматривать это временно поле штатными утилитами (cp, stat, rsync и др., а не программой на языке C), – их все тоже придётся подправить (обновить). Таким образом, в 2020 году мы имеем, что большинство стабильных (поддерживаемых на долгосрочную перспективу) дистрибутивов ОС Linux не позволяют работать с полем crtime: Debian 10.3 на ядре 4.19, Ubuntu 18.04 на ядре 5.3.x, Альт Рабочая станция 9 (basealt) на ядре 4.19.79 и CentOS. Актуальные и передовые CentOS 7 и 8, основанные на ядрах 3.10.x и 4.18.x, соответственно, тоже не позволяют работать с полем crtime. Так что же, верные стабильным операционным системам пользователи не обзаведутся такой возможностью вплоть до конца жизненного цикла упомянутых систем, то есть до 30 июня 2024 года и до 31 мая 2029 года?

[282] Debian Bug report logs - #793831 coreutils: stat does not report file birth timestamp, even when available https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=793831.

На самом деле в нашем динамично меняющемся мире, для тех кто любит экспериментировать в угоду стабильности, не всё так плохо. Жизнь неизбежно будет меняться к лучшему. Уже сейчас можно найти дистрибутивы «с поддержкой crtime» – Fedora 31 на ядре 5.3.7 и др. Также, продвинутым читателям, развлечения ради, посоветую самостоятельно проверить свою версию ядра и скомпилировать программу из [9], а также почитать [7] о возможностях использования расширенных полей inode в ФС ext4.

Поле dtime. Первоначально его предполагалось использовать как элемент контроля целостности ФС при проверках и для возможности восстановления хронологии удалений. У всех пустых и используемых inode'ов в dtime записаны нули (побитно). Во время обычного удаления файла (когда не установлен расширенный атрибут «secure deletion» (chattr +s), удалена последняя жёсткая ссылка ведущая на данный inode, а счётчик ссылок стал равен нулю) в битовых картах занятости inode'ов и блоков, проставляются нули, а в поле dtime записывается текущее время.

Замечание 3. Касаемо восстановления содержимого удалённых файлов (в случае отсутствия последующей повторной физической записи на носитель) и носителей, то это не тема данной работы, вот коротко несколько полезных программам и ключевых слов: extundelete, scalpel, testdisk, qphotorec – все есть в EPEL, первая и на sourceforge.net, а также rstudio, ufs explorer standard recovery, Active@ Disk Editor (disk-editor.org) и др.

Как легко заметить, большую часть времени поле dtime не используется, но место под него отведено в каждом индексном дескрипторе. При создании ФС ext4 подумали почему бы не использовать его для учёта «индексных дескрипторов – сирот», которые могут образоваться в следующей ситуации, например, файл открыт некоторой программой, а его или директорию в которой он находился удалили целиком. Тут или надо блокировать операцию удаления до закрытия файла, что происходит в других типах файловых систем и негативно сказывается на общей производительности. Ведь надо процесс, где было инициировано удаление притормозить и держать в ожидании в памяти до разрешения ситуации блокировки. Либо констатировать что работа с файлом одним процессом не мешает его удалению другим. Этот вариант по ресурсам менее затратный, но при этом в файловой системе образуются «полуудалённые файлы» или «индексные дескрипторы сироты» (orphan inodes). По закрытию файла блокировка с него (т. е. с inode) будет снята и система завершит удаление – проставит в битовой карте занятости inode'ов в соответствующем месте 0 и тоже самое сделает в битовой карте занятости информационных блоков, если такие были. Но пока это не произошло надо где-то хранить информацию об этих «сиротах», например списком. Создавать под него отдельную структуру как в памяти, так и внутри ФС нет смыла, вот и придумали воспользоваться полем dtime, коли оно всё равно есть, организовав эти поля в виде односторонне связанного списка. В суперблоке ФС в поле s_last_orphan записывается первый такой «сирота» из списка (точнее его номер), а в его поле dtime записывается номер следующего «сироты» и т.д. Если в поле адреса (номера следующего inode-сироты) встречается 0 – то это означает что список окончился (возможно и не начавшись). Таким образом теоретически в любой момент можно удалить хоть все файлы, а система продолжит работать, жаль, что штатного обратного механизма «усыновить» inode, то есть где-то в ФС создать жёсткую ссылку на его номер нет.

Замечание 4. На свой страх и риск (поскольку придётся править примонтированную ФС) можно использовать debugfs для создания жёсткой ссылки и увеличения числа ссылок в соот-

ветствующем поле inode. Также, будучи администратором системы, inode-сироты можно увидеть, например с помощью lsof, в выводимом списке после имени будет написано (deleted):

```
# lsof|grep "(deleted)"
mate-term  6837 user 17u  REG  0,38  24641536  180295 /tmp/#180295 (deleted)
mate-term  6837 user 18u  REG  0,38   5046272  175928 /tmp/#175928 (deleted)
…
```

и даже по ссылке через ФС procfs, зная номер pid (например 6837) удерживающего их процесса возможно к ним получится обратиться,

```
# ls -l /proc/6837/fd
итого 0
lrwx------. 1 vika vika 64 апр 16 19:29 17 -> /tmp/#180295 (deleted)
lrwx------. 1 vika vika 64 апр 16 19:29 18 -> /tmp/#175928 (deleted)
…
# cat 18 /tmp/file_copy
... тут будет вывод содержимого файла ...
```

но скопировать не получится.

```
# cp 18 /tmp/file_copy1
# cat 18 >/tmp/file_copy2
# ls -l /tmp/file_copy*
-rw-------. 1 root root 0 апр 16 19:42 /tmp/file_copy1
-rw-------. 1 root root 0 апр 16 19:42 /tmp/file_copy2
```

Новый формат секунд

Вернёмся к формату даты, хранимой в секундах. Посмотрим как обстоят дела у расширенных полей и как формируется конкретное число на примере поля «access time». Новый формат – это старый формат 32-х разрядов знакового целого (*i_atime*) + двух младших разрядов из поля расширенной части (*i_atime_extra*).

Объединение двух этих полей под эгидой формата дополнительного кода не лучшая затея (см. рис. 5.19),

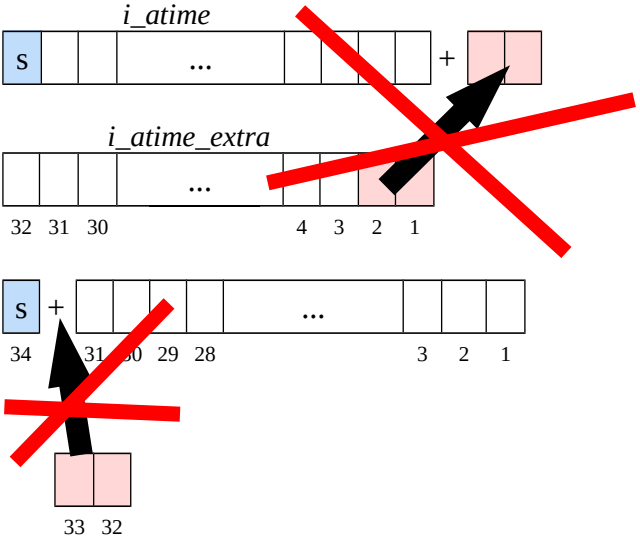


Рисунок 5.19. Приходящие на ум варианты как использовать дополнительные два бита

потому как при создании нового 34-разрядного двоичного формата, куда бы мы не поместили два лишних бита (будь то в конец, со стороны младших битов, в середину, или

в начало, сразу после бита знака – s, то есть со стороны старших битов) это приведёт к расширению диапазона представимых новым форматом чисел (секунд) с обеих краёв: как с как «положительный стороны», так и с «отрицательной», а это не совсем то, что нужно. Во-первых, вряд ли кто-то будет в будущем давать файловым объектам временные метки до 1901 года (т. е. ранее уже существующей нижней границы у «старого формата»), а во-вторых, хотелось бы чтобы земное время начала эпохи Unix осталось на прежнем месте, то есть – на нуле компьютерных секунд.

Поскольку операции с полями времени будут производятся не так часто и на высоком уровне (язык Си явно выше ассемблера), то выигрыша в производительности от операций с двоичными данными всё равно не получить, а также с учётом того, что поле времени хранит секунды, придумали объединение полей произвести не в двоичном виде (или как они хранятся в памяти), а в десятичном, для чего к получаемому из *i_atime* значению секунд в десятичном виде (обозначим через i_atime_{10}), добавлять сдвиг, рассчитываемый на основе нового двухбитового поля по следующей формуле

$$расширенные\_секунды = (i\_atime)_{10} + (i\_atime\_extra\_2\_bits)_{10} * 2^{32}.$$

То есть, фактически к старому значению надо прибавлять 232 = 4294967296 один раз, два или три, – в зависимости от значений дополнительных двух бит: 01, 10 и 11, соответственно. При наличии двух нулей второе слагаемое формулы просто обнулится. (Если под «Unix-эпохой» понимать не только время её начала и конца, как это часто можно встретить, а посмотреть более широко на адресуемое 32-битным целым числом общее количество секунд (≈136 лет), то данные дополнительные два бита как раз позволяют прожить ещё три эпохи. Как вы увидите ниже, в Замечании 5, этим битам в коде ядра не зря дали название битов дополнительных Unix-эпох (extra epoch bits).)

Дробная часть (количество наносекунд) легко получается побитным сдвигом поля i_atime_extra вправо на 2 разряда и последующим преобразованием получившегося числа в десятичный вид

$$число\ наносекунд = (i\_atime\_extra\_2\_bits >> 2)_{10}.$$

Единственное неудобство – при отображении получившегося значения пользователю может потребоваться дописать слева недостающее до 9 десятичных разрядов количество нулей (т. е. не «12345», а «,000012345»).

Замечание 5. Если говорить про внутреннюю кухню ядра и программ при работе с таким форматом, оптимальным решением будет не заниматься каждый раз расчётами, а создать класс и методы, либо новую структуру для хранения сразу двух полей *i_atime* и *i_atime_extra* и сделать для неё перегрузку часто используемых функций и операторов. Собственно в попытках подсмотреть а как же оно сделано в ядре при просмотре файла linux-3.16.82/fs/ext4/ext4.h был обнаружен следующий интересный код, в том числе и за счёт комментариев:

```
/*
 * Мы специально кодируем время так, чтобы была обратная совместимость,
 * т.е. в "00" эпохе оно осталось в старом формате без изменений.
 *
 * extra  msb of                       adjust for signed
 * epoch  32-bit                       32-bit tv_sec to       верный
 * bits   time    decoded 64-bit tv_sec  64-bit tv_sec   временной диапазон
 * 0 0    1    -0x80000000..-0x00000001  0x000000000 1901-12-13..1969-12-31
```

```
 * 0 0    0    0x000000000..0x07fffffff  0x000000000 1970-01-01..2038-01-19
 * 0 1    1    0x080000000..0x0ffffffff  0x100000000 2038-01-19..2106-02-07
 * 0 1    0    0x100000000..0x17fffffff  0x100000000 2106-02-07..2174-02-25
 * 1 0    1    0x180000000..0x1ffffffff  0x200000000 2174-02-25..2242-03-16
 * 1 0    0    0x200000000..0x27fffffff  0x200000000 2242-03-16..2310-04-04
 * 1 1    1    0x280000000..0x2ffffffff  0x300000000 2310-04-04..2378-04-22
 * 1 1    0    0x300000000..0x37fffffff  0x300000000 2378-04-22..2446-05-10
 *
 * Обратите внимание, что предыдущие версии ядер на 64-битных системах для
 * ситуации extra epoch bits = 1,1 производят некорректный пересчёт времени
 * для отрицательных значений секунд (т.е. между 1901 и 1970 гг.) Утилита
 * e2fsck при её запуске исправит значения, полагая, что она запускается
 * до 2242 года на системах, где есть описанная проблема расчёта. (или до
 * 2310 по данным из [33] - прим. авт., в общем, дожить бы, а там увидим)
 */

static inline __le32 ext4_encode_extra_time(struct timespec *time)
{
    u32 extra = sizeof(time->tv_sec) > 4 ?
         ((time->tv_sec - (s32)time->tv_sec) >> 32) & EXT4_EPOCH_MASK : 0;
    return cpu_to_le32(extra | (time->tv_nsec << EXT4_EPOCH_BITS));
}

static inline void ext4_decode_extra_time(struct timespec *time, __le32 ex-
tra)
{
       if (unlikely(sizeof(time->tv_sec) > 4 &&
                        (extra & cpu_to_le32(EXT4_EPOCH_MASK)))) {
#if LINUX_VERSION_CODE < KERNEL_VERSION(4,20,0)
    /* Поддерживаем старый (неправильный вариант) кодирования дат до 1970 г.
     * в ситуации с epoch bits=1,1, поскольку  мы предполагаем, что
     * большинство использует ядра не превышающие 4.20 и, соответственно,
     * запускает fsck на подверженных проблеме файловых системах чтобы
     * исправить ошибки. (По усмотрению ext4 разработчиков данный фрагмент
     * кода, обеспечивающий совместимость со старыми версиями ядра, в
     * будущем может быть удалён.)
                */
       u64 extra_bits = le32_to_cpu(extra) & EXT4_EPOCH_MASK;
       if (extra_bits == 3 && ((time->tv_sec) & 0x80000000) != 0)
               extra_bits = 0;
       time->tv_sec += extra_bits << 32;
#else
       time->tv_sec += (u64)(le32_to_cpu(extra) & EXT4_EPOCH_MASK) << 32;
#endif
       }
  time->tv_nsec = (le32_to_cpu(extra) & EXT4_NSEC_MASK) >> EXT4_EPOCH_BITS;
}
```

То есть, коротко суть проблемы: в ядрах до 3.12 существовала ошибка по работе с датами в ФС ext4 после 2038 года. Затем в ядре это исправили и внесли изменения в алгоритм работы утилиты проверки дисков e2fsprogs, начиная с версии 1.42.8. В 64-битных ядрах появилась вторая ошибка – они некорректно работают с epoch bits =1,1 и датами с «отрицательными» секундами (т. е. 1901–1970 проецируемый на 2310–2378). Исправили это в ядрах 4.20, а что делать сейчас и в будущем см. комментарии в ядре.

Директория (каталог)

Директория, так же как и файл, описывается при помощи inode. Наглядности ради её можно считать особым псевдофайлом, у которого в ФС установлен тип «d». а его содержимое представляет из себя массив записей, содержащих информацию о файлах, псевдофайлах или поддиректориях, которые находится «внутри» неё. Используется два не сильно отличающихся формата этих записей (ext4_dir_entry и ext4_dir_entry_2). Поскольку исследование директорий не входит в цели данной работы, коротко скажем, что первая из вышеупомянутых структур имеет следующий вид:

- порядковый номер inode файла;
- длина записи в байтах;
- длина имени файла;
- имя файла.

Формат группы блоков

Обобщённая структурная схема файловой системы ext2 представлена на рис. 5.15. Практически все группы блоков имеют одинаковый формат. В каждой группе, помимо информационных блоков, хранится информация о занятости блоков и inode группы в виде битовой карты (*Block Bitmap* и *Inode Bitmap*). В состав группы блоков 0 входят также суперблок и таблица дескрипторов групп.

Битовая карта занятости блоков (Block Bitmap) обычно расположена в первом блоке группы. Если в группе присутствует резервная копия суперблока, битовая карта располагается во втором блоке группы. Размер битовой карты – один блок. Каждый бит этой карты обозначает состояние блока. Если бит установлен (1), то блок занят, если сброшен (0) – блок свободен. Первому блоку группы соответствует нулевой бит карты, второму блоку – первый бит и т. д. Inode, находящиеся в пределах одной группы, собраны в таблицу. В битовой карте занятости inode группы каждый бит характеризует состояние элемента в таблице inode группы.

Каждая группа блоков описывается при помощи ***дескриптора группы блоков (Block Group Descriptor)***. Дескриптор группы – это структура, которая содержит информацию об адресах битовой карты занятости блоков, битовой карты занятости inode и таблицы inode соответствующей группы. Все дескрипторы групп собраны в таблицу дескрипторов групп, которая хранится в группе блоков 0. Вместе с каждой копией суперблока операционная система создаёт и резервные копии таблицы дескрипторов групп.

5.3.7.5.1. Работа с ФС ext2 в Windows

Многие пользователи имеют две операционные системы на своих компьютерах, например ОС Windows и Linux. Если какой-нибудь дистрибутив Ubuntu из коробки понимает NTFS, которую использует Windows, то, в обратную сторону, Windows никак не хочет понимать ext2 (а также в дальнейшем ext3 или ext4).

Указанная проблема решаема с помощью установки специфического драйвера для ФС ext2.

Для просмотра ext2/ext3/ext4, а также и raserFS, разделов в ОС Windows есть 4 основных бесплатных проекта:

- Ext2fsd (поддержка ext2, ext3 и raserFS)[283];
- Ext2IFS (поддержка ext2, ext3 и raserFS)[284];
- DiskInternal Linux Reader (поддержка ext2, ext3 и raserFS)[285];
- ext2explore (поддержка ext2, ext3, ext4 и raserFS)[286].

5.3.7.5.2. ext3

В ноябре 2001 года была представлена третья версия расширенной файловой системы (Third Extended File System), сокращённо ext3 или ext3fs – журналируемая файловая система. До выхода ФС ext4 использовалась по умолчанию во многих дистрибутивах Linux. Основана на ФС ext2.

Основное отличие от ext2 состоит в том, что ext3 журналируема, что позволяет значительно сократить время проверки и восстановления целостности ФС при сбоях в работе компьютера, по аналогии с NTFS.

Стандартом предусмотрены три режима журналирования:

- writeback: в журнал записываются только метаданные файловой системы, то есть информация о её изменении. Не может гарантировать целостности данных, но уже заметно сокращает время проверки, по сравнению с ext2;
- ordered: то же, что и writeback, но запись данных в файл производится гарантированно до записи информации об изменении этого файла. Немного снижает производительность, также не может гарантировать целостности данных (хотя и увеличивает вероятность их сохранности при дописывании в конец существующего файла);
- journal: полное журналирование как метаданных ФС, так и пользовательских данных. Самый медленный, но и самый безопасный режим; может гарантировать целостность данных при хранении журнала на отдельном разделе (а лучше на отдельном жёстком диске).

Технически хранение журнала выполнено в специальном файле, поэтому ext3 одновременно является полноценным ext2-разделом, просто с дополнительно установленным флагом и с файлом журнала, либо указателем на внешний журнал. Естественно, при монтировании ФС ext3 как ext2 функции журналирования не поддерживаются, в остальном пользователь не ощутит разницу.

Преобразовать ФС ext2 в ext3 (по сути, создать журнал) можно без переформатирования, то есть без потери пользовательских данных.

Файловая система ext3 может поддерживать файлы размером до 1 ТБ. Размер же тома ограничен версией ядра и разрядностью процессора.

Фрагментация в ext3

Большинство пользователей Linux, в отличие от Windows-пользователей, не знает про утилиты «дефрагментации диска», потому как они, наверное в 99% процентов слу-

[283] http://www.ext2fsd.com.
[284] http://www.fs-driver.org.
[285] http://www.diskinternals.com/linux-reader/.
[286] http://ext2read.blogspot.com.

чаев не нужны в силу оптимальности самой ФС, однако следует понимать, что фрагментация всё же затрагивает в той или иной степени все файловые системы.

Рассматривая вопрос о фрагментации в ФС ext3, необходимо различать два весьма различных понятия: внутренняя фрагментация [287] и внешняя фрагментация. Последняя относится к файлам, чьи блоки данных не следуют непрерывной чередой друг за другом, но разбросаны по всему диску. В результате считывающие головки тратят больше времени на доступ для чтения и записи. Внутренняя фрагментация «только» транжирит дисковое пространство, тогда как внешняя снижает быстродействие системы.

Ext3 весьма успешно справляется с внешней фрагментацией и минимизирует перемещения считывающих головок (для классического жёсткого диска), например пытается заготовить сектор из восьми последовательно идущих блоков для вновь создаваемых файлов. Кэширование записи также обеспечивает запись на диск «в один проход» тех данных, которые приложение записывало кусками, что гарантирует непрерывную область на диске. Конечно, кэширование записи не помогает, когда файлы растут медленно, например директории, которые заполняются от случая к случаю. Фрагментация также неизбежна в файлах различного размера, постоянно создаваемых и удаляемых, что приводит к появлению брешей из пустых блоков.

Борьба с фрагментацией накладывается на ещё одну оптимизирующую стратегию − локализацию данных и метаданных, которую пытаются достичь при помощи групп блоков. Из-за того, что ФС ext3 пытается сохранить файлы одной директории в одной группе блоков, в ней возникает фрагментация, хотя рядом на диске может находиться большая непрерывная свободная область. Утверждение, что ФС ext3 начинает фрагментировать только при заполнении файловой системы на 80−90%, не всегда верно. В зависимости от стиля работы файловая система может быть фрагментирована и при изобилии свободного места.

Как следствие борьбы с фрагментацией в ФС ext3 появилась ФС ext4, заодно в ней были улучшены и учтены другие требования времени.

5.3.7.5.3. ext4

21 октября 2008 г. была представлена стабильная (тестовая версия была доступна двумя годами ранее) четвёртая версия расширенной файловой системы (Fourth Extended File System), сокращённо ext4, или ext4fs – журналируемая файловая система. Сегодня используется по умолчанию во многих дистрибутивах Linux. Основана на ФС ext3.

Основной особенностью стало увеличение максимального объёма одного раздела диска до 1 эксбибайта (260 байт) при размере блока 4 КБ и увеличение размера одного файла до 16 тебибайт. Кроме того, в ext4 представлен механизм пространственной (extent) записи файлов (новая информация добавляется в конец заранее выделенной по соседству области файла), уменьшающий фрагментацию и повышающий производительность.

[287] Термин «*внутренняя фрагментация*» взят из статьи Oliver Diedrich (в переводе Алексея Дмитриева за 2008 г.) и означает тот эффект, что файл всегда занимает целое число блоков. То есть размер файла на диске кратен размеру блока. В среднем каждый файл использует только половину последнего блока − чем крупнее блоки и меньше файлы, тем ощутимее потери.

Экстенты

Во многих файловых системах в общем случае файл хранится в виде «заголовка», то есть некой относительно небольшой структуры данных (например, inode и косвенных блоков в ext3 или строки Master File Table в NTFS), который содержит указатели на участки носителя информации, где по кускам хранится содержимое файла. В традиционных файловых системах это указатели на отдельные блоки (минимальные участки носителя, которые можно прочесть или записать за раз). В ряде современных файловых систем используются указатели не на блоки, а на экстенты [288].

Использование указателей на экстенты имеет ряд преимуществ над схемой с указателями на отдельные блоки. Поскольку все данные в одном экстенте расположены на диске подряд, повышается скорость чтения и записи файла и понижается степень фрагментации дискового пространства. При одинаковом размере и организации структуры данных «заголовка» файла файловая система с поддержкой экстентов будет иметь больший максимальный размер файлов.

Главный недостаток экстентов – повышенная сложность реализации файловой системы.

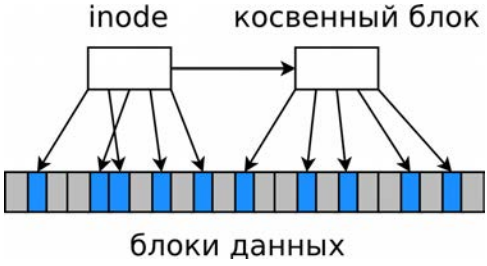


Рисунок 5.20. Файл в традиционной файловой системе

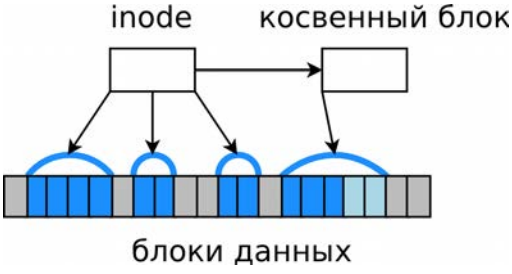


Рисунок 5.21. Файл в файловой системе с поддержкой экстентов

В файловой системе ext3 адресация данных выполнялась традиционным образом, поблочно. Такой способ адресации становится менее эффективным с ростом размера файлов. Экстенты позволяют адресовать большое количество (до 128 МБ) последовательно идущих блоков одним дескриптором. До 4 указателей на экстенты может размещаться непосредственно в inode, что достаточно для файлов маленького и среднего размера.

[288] **Экстент** (англ. *extent* − протяжённость) – в файловых системах непрерывная область носителя информации. Как правило, в файловых системах с поддержкой экстентов большие файлы состоят из нескольких экстентов, не связанных друг с другом напрямую.

Другие особенности ФС ext4

- 48-битные номера блоков. При размере блока 4 КБ это позволяет адресовать до одного эксабайта ($2^{48} \cdot 4$ КБ = $2^{50} \cdot 1$ КБ = 2^{60} байт = 1 ЭБ).
- Выделение блоков группами (multiblock allocation). Файловая система хранит не только информацию о местоположении свободных блоков, но и количество свободных блоков, идущих друг за другом. При выделении места файловая система находит такой фрагмент, в который данные могут быть записаны без фрагментации. Это снижает уровень фрагментации файловой системы в целом.
- Отложенное выделение блоков (delayed allocation). Выделение блоков для хранения данных файла происходит непосредственно перед физической записью на диск (например, при вызове sync), а не при вызове write. В результате операции выделения блоков можно делать не по одной, а группами, что, в свою очередь, минимизирует фрагментацию и ускоряет процесс выделения блоков. С другой стороны, увеличивает риск потери данных в случае внезапного пропадания электропитания.
- Преодолено прежнее ограничение в 32 000 каталогов в ext3. (Если изменять константы ядра и перекомпилировать его в части, поддерживающего указанную ФС было до 65 535).
- Резервирование inode'ов при создании каталога (directory inodes reservation). При создании каталога резервируется несколько inode'ов. Впоследствии при создании файлов в этом каталоге сначала используются зарезервированные inode'ы, и если таких не осталось, выполняется обычная процедура.
- Размер inode увеличен с 128 (по умолчанию) до 256 байт. Это дало возможность реализовать последующие три дополнительные возможности.
- Временные метки с наносекундной точностью (nanosecond timestamps). Более высокая точность времён, хранящихся в inode. Диапазон хранящихся времён тоже расширен: если раньше верхней границей хранимого времени было 18 января 2038 года, то теперь это 25 апреля 2514 года.
- Версия inode. В inode появился номер, который увеличивается при каждом изменении inode файла. Это будет использоваться, например, в NFSv4, для того чтобы узнавать, изменился ли файл.
- Хранение расширенных атрибутов в inode (EA in inode). Хранение расширенных атрибутов, таких как ACL, атрибутов SELinux и прочих, позволяет повысить производительность. Атрибуты, для которых недостаточно места в inode, хранятся в отдельном блоке размером 4 КБ. Предполагается снять это ограничение в будущем.
- Контрольное суммирование в журнале (Journal checksumming). Контрольные суммы журнальных транзакций. Позволяют лучше найти и (иногда) исправить ошибки при проверке целостности системы после сбоя.
- Предварительное выделение (persistent preallocation). Сейчас для того, чтобы приложению гарантированно занять место в файловой системе, оно заполняет его нулями. В ext4 появилась возможность зарезервировать множество блоков для записи и не тратить на инициализацию лишнее время. В стандартном коде, если приложение попробует прочитать данные, оно получит сообщение о том, что они не проинициализированы. Таким образом, несанкционированно прочитать удалённые данные не получится.
- Дефрагментация без размонтирования (online Defragmentation). Реализовано в

самой последней версии e2fsprogs (~/misc/e4defrag).

- Неинициализированные блоки (uninitialised groups). Пока не реализовано. Позволяет ускорить проверку файловой системы с помощью fsck. Блоки, отмеченные как неиспользуемые, проверяются группами, и детальная проверка производится, только если проверка группы показала, что внутри есть повреждения. Предполагается, что эта возможность может очень сильно ускорить процесс проверки целостности файловой системы; в зависимости от способа размещения данных время проверки будет составлять от 1/2 до 1/10 от нынешнего.

Несмотря на все улучшения в ФС ext4, ещё предшественница ФС ext2 по-прежнему используется на флэш-картах и твердотельных накопителях (SSD), так как отсутствие журналирования является преимуществом при работе с накопителями, имеющими ограничение на количество циклов записи.

Разрежённые файлы

Увеличенный размер inode (в т.ч. если он будет принудительно создан в ФС ext2 и ext3) позволяет хранить метаданные файлов, которые позволяют создавать разрежённые файлы.

Разрежённый файл (англ. sparse file) − файл, в котором, в какой-то мере по аналогии с адресацией экстентами, последовательности нулевых байтов в содержании файла заменены на информацию о количестве этих нулей. Фактически, вместо нулевых данных хранится список дыр в файле. В данном случае под дырой в файле понимается последовательность нулевых байт в файле не записанных на диск. Информация о дырах (смещение от начала файла в байтах и количество байт) хранится в метаданных ФС, а точнее в расширении inode.

VirtualBOX, различные торрент-качалки и другое ПО умеют работать с такими файлами. В ОС Linux правильное копирование подобных файлов выполняется командой cp с ключом − sparse. Приведём несколько примеров полезных команд.

Создание разрежённого файла размером 100 Гб:

```
$ dd if=/dev/zero of=./sparse-file bs=1 count=0 seek=100G
```

или

```
$ truncate -s200G ./sparse-file
```

Преобразование (копированием) обычного файла в разрежённый (выполнение поиска дыр и записи их расположения (смещений и длин) в метаданные файла):

```
$ cp --sparse=always ./simple-file ./sparse-file
```

Сохранение копии диска в разрежённый файл:

```
# ddrescue --sparse /dev/sda ./sparse-file ./history.log
```

Замечание. В ОС Windows ФС NTFS и некоторые программы для этой ОС, например Far manager и др. также поддерживают разрежённые файлы. См. утилиту fsutil.

5.3.7.6. CDFS, UDF, ISO 9660

Файловые системы, используемые для оптических носителей информации, таких как CD- и DVD-, BD-дисков, отличаются от тех, что используются для хранения информации на жёстких дисках и flash-носителях. Наиболее популярные сегодня форматы – **CDFS** и **UDF**.

CDFS, или файловая система CD-ROM, имеет относительно простой формат, определённый в 1988 году применительно к носителям CD-ROM, предназначенных только для чтения. В ОС **Windows** и **Linux** реализована **CDFS**, совместимая со стандартом ISO 9660, которая поддерживает длинные имена файлов в соответствии с ISO 9660 Level 2. Формат **CDFS** очень прост, и поэтому ему присущ ряд ограничений:

- длина имён файлов и каталогов не должна превышать 32 символов;
- глубина вложения каталогов не может превышать 8 уровней.

CDFS считается устаревшим форматом, поскольку принят новый стандарт для компакт-дисков – Universal Disk Format (**UDF**). **Windows** и **Linux** поддерживает файловую систему **UDF**, совместимую со стандартом ISO 13346. Ассоциация **OSTA** (Optical Storage Technology Association) определила **UDF** в 1995 году как формат магнитооптических носителей, главным образом DVD-ROM, предназначенный для замены **CDFS**.

Файловая система **UDF** обладает следующими преимуществами:

- длина имени файла – до 255 символов;
- максимальная длина пути – 1023 символа;
- имена файлов могут включать буквы как верхнего, так и нижнего регистра.

Windows Vista, 7, 8, 10, Linux и **MacOS X** поддерживают работу с файловыми системами **ISO9660**, **CDFS** и **UDF**.

ФС UDF имеет несколько версий, а их поддержка зависит от используемой ОС см. Таблицу 5.13. Для максимальной совместимости по чтению и записи лучше использовать версию 2.01.

Таблица 5.13. Поддержка разных версий ФС UDF[289]

ОС	версия				
	1.02	1.50	2.0x	2.50	2.60
Linux (ядра от 2.4.x до 2.6.25 включительно)	да	да	да	нет	нет
Linux (>=2.6.26)	да	да	да	только чтение*	нет
Mac OS X 10.4	да	да	да	нет*	нет*
Mac OS X (>10.5)	да	да	да	да*	только чтение*
Windows XP/Server 2003	только чтение*	только чтение	только чтение	нет	нет
Windows Vista/7/8/10	да*	да	да	да	только чтение*

* С ограничениями, либо требуется стороннее ПО.

С использованием ФС UDF появляется возможность работы с перезаписываемыми оптическими DVD и CD дисками как с флеш-накопителями. Для реализации этой возможности в ОС Linux необходимо предварительно отформатировать диск командой

```
$ mkudffs /dev/sr0
start=0, blocks=16, type=RESERVED
start=16, blocks=3, type=VRS
start=19, blocks=237, type=USPACE
start=256, blocks=1, type=ANCHOR
```

[289] Более полную таблицу совместимости см. на http://en.wikipedia.org/wiki/Universal_Disk_Format.

```
start=257, blocks=16, type=PVDS
start=273, blocks=1, type=LVID
start=274, blocks=2294573, type=PSPACE
start=2294847, blocks=1, type=ANCHOR
start=2294848, blocks=239, type=USPACE
start=2295087, blocks=16, type=RVDS
start=2295103, blocks=1, type=ANCHOR
```

входящей в пакет udftools.

Замечание. Опытным путём установлено, что лучше использовать болванки DVD+RW, поскольку некоторые DVD-RW, несмотря на успешное форматирование, использовать в режиме записи «простым копированием» не удалось.

О программах записи оптических дисков см. стр. 527.

5.3.8. Краткое сравнение файловых систем

Приведём краткую таблицу сравнения ФС (см. табл. 5.14). Следует отметить, что многие ограничения возникают не на уровне ФС, а на уровне ОС. Так, 64 КБ кластеры в ФС FAT16 поддерживаются не всеми ОС. Часть параметров, например max размер тома, зависит от размера кластера. Ограничение по максимальному числу файлов, например, для ФС ext может формироваться в момент создания ФС, размером получившейся битовой карте inode'ов. И прочие нюансы...

Таблица 5.14. Сравнение файловых систем по некоторым параметрам

Параметр	FAT12	FAT16	FAT32	exFAT	NTFS	ext2	ext3	ext4
max размер файла	32 МБ	2 ГБ	$2^{32}-1$	~16 ЭБ	~16 ЭБ	2 ТБ	2 ТБ	16 ТБ
max размер тома	32 МБ	2(4) ГБ	~8 ТБ	~16 ЭБ	~16 ЭБ	~32 ТБ	~32 ТБ	1 ЭБ
max файлов	4068	65460	~268 млн	-	$2^{32}-1$	10^{18}	~4 млрд	~4 млрд
журналируемая	нет	нет	нет	нет	да	нет	да	да
владельцы файлов	нет	нет	нет	нет	да	да	да	да

5.3.9. Таблица разделов, MBR, GPT

Использование одной ФС для одного жёсткого диска может быть неоправданной роскошью, поэтому во всех ОС изначально предусмотрена возможность размещения на одном физическом жёстком диске нескольких разделов, в каждом из которых может иметься своя файловая система.

Если мы говорим о жёстких дисках, то даже если ФС одна, таблица разделов всё равно нужна. На USB-дисках возможно использование ФС без таблицы разделов, но по умолчанию они форматируются с созданием таблицы разделов, в которую помещается один раздел размером на всё свободное пространство.

Физическое размещение таблицы разделов, если говорить о жёстких дисках, – первый сектор диска (цилиндр 0, головка 0, сектор 1). В этот сектор заносится главная загрузочная запись – **Master Boot Record** (**MBR**), которая содержит:

1) исполняемый код – программу начальной загрузки (внесистемный загрузчик – non-system bootstrap);
2) таблицу разделов – таблицу разбиения диска (Disk Partition Table).

В таблице разбиения диска содержатся сведения о том, с каких цилиндров, головок и секторов начинаются и какими заканчиваются имеющиеся на диске разделы (если головки, цилиндры и сектора виртуальные, как для SSD-накопителей следует понимать, что их всегда можно пересчитать в нужный сектор от начала диска). В этой таблице также содержатся указания для системной **BIOS**, какой из разделов является загрузочным, то есть где следует искать основные файлы операционной системы.

Для работы с таблицей разделов в ОС Linux существуют как консольные утилиты: fdisk, sfdisk, gpart и другие, так и с графическим интерфейсом, например GParted[290] и другие.

В ОС Windows используется в основном fdsik (более скудный по своим возможностям чем в Linux), либо графическая утилита «Управление дисками» из «Панели управления».

Отдельно следует отметить, что существуют и другие разноплатформенные коммерческие и бесплатные продукты, в том числе «Live» версии автономно загружающиеся с оптических дисков, по сети или с USB-накопителей. Среди наиболее известных сторонних программ для работы с таблицами разделов можно назвать Norton Partiton Magic, Acronis Disk Director, Paragon Partition Manager, EaseUS Partition Manager и другие.

Если ранее BIOS использовал таблицу разделов получаемую из MBR, то с подачи фирмы Intel в 2000-м году наблюдается постепенный переход на EFI и GPT, соответственно[291]. GPT – это GUID Partition Table новый стандарт формата размещения таблиц разделов на физическом жестком диске, где GUID (Globally Unique Identifier) – статистически уникальный 128-битный идентификатор. GUID-идентификаторы являются подмножеством более общих UUID-идентификаторов. Для удобства чтения человеком, они обычно записываются в виде текстовой строки следующего вида **`94618ade-887f-41d5-80bc-d7a3d984bf2a`**, получаемой из оригинального 128-битного числа перестановкой части его битов местами. Подробнее см. RFC4122 и [29].

Для просмотра таблиц разделов всех доступных «MBR-дисков» используется команда:

```
# fdisk -l
Диск /dev/sda: 2000.4 ГБ, 2000398934016 байт
81 heads, 63 sectors/track, 765633 cylinders, всего 3907029168 секторов
Units = секторы of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
Disk identifier: 0xXXXXXXXX

Устр-во Загр   Начало   Конец    Блоки   Id  Система
/dev/sda1 *        1    1402  11261533+   7  HPFS/NTFS
/dev/sda2       1403    1415    104422+  83  Linux
/dev/sda3       1416    1546   1052257+  82  Linux swap / Solaris
/dev/sda4       1547    9729  65729947+   5  Extended
/dev/sda5       1547    7920  51199123+  8e  Linux LVM
```

[290] GNOME Partition Editor http://gparted.sourceforge.net/.

[291] GPT является частью «Расширяемого микропрограммного интерфейса» (англ. Extensible Firmware Interface, EFI, ранее упомянутого на стр. 262).

Однако, для просмотра разделов на «GPT-диске», команда «`fdisk -l`» уже не подойдёт, следует использовать команды:

```
# lshw -C disk
```

или

```
# parted -l
```

или с указанием конкретного диска

```
# gdisk -l /dev/sda
```

которые входят в состав одноимённых пакетов.

Названия программ легко запомнить, поскольку они есть сокращения от «list hardware», «partition editor» и «GPT fdisk», соответственно.

5.3.9.1. Полезные советы про таблицу разделов

Когда функционала вышеописанных утилит для работы с таблицей разделов недостаточно, возможно обратиться к ней напрямую посредством утилиты lde (linux disk editor, входит в состав репозитария EPEL), а также программы debugfs.

Для снятия копии таблицы разделов (точнее MBR) возможно использовать утилиту dd, например для копирования MBR в файл «/путь/файл_копия_mbr»:

```
# dd if=/dev/sda of=/путь/файл_копия_mbr bs=512 count=1
```

Для восстановления MBR из копии достаточно поменять значения параметров if и of (от англ. input file и output file, соответственно) местами. Если опустить параметр count, то таким образом можно скопировать весь диск. Если необходимо «копирование» со смещением, то оно задаётся ключами `skip=` и `seek=`.

Полезный совет. Во время копирования, до тех пор пока оно не завершено утилита dd ничего не выводит. Если копируется большой жёсткий диск целиком, то процесс может занять несколько часов. Через час-два любой администратор начнёт если не нервничать, то интересоваться тем как идёт процесс копирования. В этом случае можно послать сигнал 1 (SIGHUP) процессу dd. Например «`kill -1 1234`», где 1234 – pid процесса dd, который можно узнать из вывода команд «`top`» или «`ps aux`».

Если у вас имеется диск или его посекторная dd-копия в файле, то выяснить где на нём (с каким смещением) находятся разделы возможно командой

```
# sfdisk -l -uS image_file.dd
```

Смещение даётся в секторах (размером 512 байт). В последующем полученное число может быть умножено на 512 для получения смещения в байтах и подставлено в команду

```
# mount -oloop,offset=смещение_в_байтах image_file.dd /куда/монтировать
```

для монтирования искомого раздела. Монтирование работает только для смещений менее чем 2 ГБ, для больших следует убедиться, что в ядре имеется поддержка «NASA loopback driver».щ

Восстановление MBR может также потребоваться в случае если находящийся там загрузчик или его часть были повреждены. Для ОС Linux загрузчиком, скорее всего, будет GRUB, для его восстановления достаточно ознакомиться с «info grub», где вы найдёте пару способов для восстановления и необходимые команды. В случае использования устаревшего загрузчика Lilo, придётся полистать документацию 10-20 летней давности. Для ряда версий ОС Windows загрузчиком является NT loader. Восстановить

его возможно командами fixboot и fixmbr, загрузившись с диска восстановления. Для Windows 7 используется команда «bootrec.exe /fixmbr».

Если вы установили Linux «поверх» Windows и загрузчик GRUB затёр загрузчик Windows, команды выше вернут первоначальный NT Loader.

Если вы хотите поставить Linux, а раздел с Windows занимает весь объём жёсткого диска, то начиная с Windows 7 ситуацию легко исправить штатными средствами. Для этого надо зайти в «Управление дисками» (из «Панели управления», либо через «Пуск» ► «Компьютер» - правой кнопкой мыши выбрать «Управление») и правой кнопкой мыши на укорачиваемом разделе диска выбрать пункт «Сжать том...».

ОС Linux возможно загружать из под загрузчика NT Loader. Для этого при установке ОС Linux загрузчик GRUB необходимо установить не в MBR, где находится NT Loader, а в первый сектор выделенного под Linux раздела. Далее следует сделать копию первых 512 байт этого раздела в файл (например linux.bs). Файл следует переписать на диск с Windows (например в C:\) и внести правку в файл boot.ini, лежащий в корне диска, дописав туда:

```
C:\linux.bs="OS Linux"
```

5.3.10. Полезная информация про ОС Windows

Рассмотрим некоторые интересные моменты касающиеся элементов интерфейса ОС Windows и её основных компонентов.

5.3.10.1. Меню Пуск

В **Windows 8** разработчики полностью отказались от ***Меню Пуск***, однако это решение было встречено негативно конечными пользователями и программистами. Как следствие, появилось много альтернативных решений по возврату меню и кнопки «Пуск» («Start» в англ. версии) на прежнее место. Например, несколько решений, как коммерческих, так и бесплатных, представлены по адресу http://www.x-drivers.ru/articles/software/178/full.html. Позднее оно вернулось.

5.3.10.2. Архивация и восстановление данных

Также см. разделы 2.5 «*Сжатие (архивация) различных видов информации*» и 5.6.3 «*Архивация файлов*».

Несмотря на то что архивация основных файлов и системной информации является чрезвычайно важной для обеспечения безопасности, не все пользователи находят время для выполнения этой операции. Как показали исследования, пользователям нужны более удобные способы выбора файлов и папок для архивации, а также встроенная поддержка распространённых резервных носителей, таких как CD-, DVD- и BD-диски.

Средство архивации и восстановления данных **Windows** решает эти задачи и позволяет пользователям выбирать файлы для архивации. Пользователи могут восстанавливать файлы с одного CD- или DVD-диска или набора дисков, даже если доступны не все входящие в архив диски.

Все функции архивации восстановления объединены в одной унифицированной панели архивации и восстановления, называемой центром архивации и восстановления.

Теневое копирование. Случайное удаление или изменение файла является распространённой причиной потери данных. В состав **Windows** входит новая полезная функция теневого копирования, которая помогает избежать потери данных. Эта функция, доступная в выпусках Business, Enterprise и Ultimate, автоматически создаёт текущие копии файлов, с которыми работает пользователь, что позволяет быстро и просто восстанавливать предыдущие версии случайно удалённого документа.

Замечание. Не защищайте с помощью теневого копирования ваши документы, потому как, исправив и сохранив документ сегодня, завтра вы можете обнаружить его более раннюю версию, ту, что без ваших последних исправлений.

Функция теневого копирования **Windows** включается автоматически и периодически создаёт копии изменённых файлов. Для хранения теневых копий используется минимум места на жёстком диске, поскольку сохраняются только новые изменения. Чтобы получить доступ к этой функции, достаточно щёлкнуть правой кнопкой мыши файл или папку и выбрать команду **Восстановить прежнюю версию**. Файлы можно просмотреть в режиме «только для чтения», чтобы решить, какую версию файла следует восстанавливать. Чтобы полностью восстановить файл, достаточно перетащить его в любую папку или выделить его и выбрать команду **Восстановить** для восстановления файла в исходное местоположение.

Эта функция позволяет восстанавливать предыдущие версии, как отдельных файлов, так и целых папок. При восстановлении файла отображаются все предыдущие версии, отличные от активной версии на жёстком диске. При доступе к предыдущей версии папки пользователи могут просматривать иерархию папок, существовавшую на момент сохранения теневой копии папки.

Восстановление системы

Для функции восстановления системы используется теневое копирование (Shadow Copy). Это позволяет ускорить процесс восстановления и сделать его более надёжным. Если компьютер находится в аварийном состоянии и на нём невозможно запустить функцию восстановления системы, можно воспользоваться средой восстановления **Windows** для запуска этой функции с установочного диска **Windows**. Теперь в функции восстановления системы предусмотрена возможность отмены – перед изменением системы автоматически создаётся точка восстановления на тот случай, если после восстановления системы потребуется отменить это действие.

5.3.10.3. Методы повышения производительности

Рассмотрим некоторые секреты повышения производительности, а также полезные, но малоизвестные функции.

В **Windows** входит компонент панели управления **Счётчики и средства производительности**, позволяющий оценивать показатели производительности и устранять возникшие трудности. Некоторые компоненты **Windows** и сторонние приложения могут работать только на компьютерах, оборудование которых соответствует определённым требованиям. Новый **индекс производительности Windows** позволяет опреде-

лить степень этого соответствия и выяснить, будут ли функционировать такие компоненты и приложения.

Технология **Windows ReadyBoost** (прежнее название – **EMD**) обеспечивает значительный рост производительности за счёт использования флэш-накопителей без необходимости расширения ОЗУ. Поддерживается и технология гибридных жёстких дисков **Windows ReadyDrive** (прежнее название – гибридный жёсткий диск), которая предназначена для повышения надёжности и производительности, а также увеличения срока работы от батареи.

Пользоваться функцией **Windows ReadyBoost** очень просто. При первой установке съёмного запоминающего устройства, например флэш-накопителя USB или карты памяти SD, в соответствующий порт операционная система **Windows** проверяет, обладает ли это устройство достаточным быстродействием для поддержки технологии **Windows ReadyBoost**. Если это так, появляется запрос на использование устройства для повышения производительности системы. Для этой цели можно выделить часть памяти устройства USB, а в другой части, как обычно, хранить файлы. Запоминающее устройство, используемое функцией **Windows ReadyBoost**, можно в любое время отключить – это не приведёт к потере данных и не окажет негативного воздействия на систему. Данные на запоминающем устройстве хранятся в зашифрованном виде для предотвращения несанкционированного доступа.

С точки зрения таблицы разделов, данная функция выглядит следующим образом:

```
$ fdisk -l /dev/sdg
Диск /dev/sdg: 32.5 ГБ, 32505856000 байт
64 heads, 32 sectors/track, 31000 cylinders, всего 63488000 секторов, Units
= секторы of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x6eXXXXXX

Устр-во Загр Начало    Конец    Блоки   Id Система
/dev/sdg1 ? 1936269394 3772285809 918008208 4f QNX4.x 3-я часть
/dev/sdg2 ? 1917848077 2462285169 272218546+ 73 Неизвестный
/dev/sdg3 ? 1818575915 2362751050 272087568 2b Неизвестный
/dev/sdg4 ? 2844524554 2844579527 27487 61 SpeedStor
```

Для использования функции **Windows ReadyBoost** объём съёмной флэш-памяти должен быть не менее 256 МБ и она должна иметь достаточное быстродействие. Скорость прямого чтения блоков размером 4 КБ должна составлять не менее 2,5 МБ/с, скорость прямой записи блоков размером 512 КБ – не менее 1,75 МБ/с.

Автоматическое управление памятью при переходе в режим сна осуществляется с помощью нового диспетчера памяти **Windows SuperFetch**. Эта новая технология управления памятью позволяет добиться высокой скорости реагирования на запросы пользовательских приложений за счёт управления памятью с использованием нового алгоритма, который назначает пользовательским приложениям более высокий приоритет, чем фоновым процессам, и адаптируется к стилю работы каждого пользователя. Фоновые процессы продолжают выполняться, пока компьютер бездействует. Однако по завершении фонового процесса **SuperFetch** восстанавливает в памяти данные, с которыми пользователь работал до запуска этого процесса.

Кроме того, для повышения быстродействия ПК функция **SuperFetch** определяет наиболее часто используемые приложения и предварительно загружает их в память. За

счёт этого увеличивается скорость работы системы при запуске или переключении на другой профиль пользователя.

Низкоприоритетный ввод-вывод при работе с диском для фоновых процессов – в соответствии с ним фоновые процессы имеют более низкий приоритет при доступе к жёсткому диску, чем пользовательские приложения. Если программа поддерживает низкоприоритетный ввод-вывод, то она может выполняться одновременно с пользовательским приложением, не замедляя его работы. Именно так функционирует ряд служб **Windows**, включая индексирование поиска, автоматическую дефрагментацию диска и ежедневную проверку системы Защитником Windows.

Автоматическая дефрагментация жёсткого диска может выполняться в фоновом режиме по расписанию без снижения быстродействия приложений.

5.3.11. Сбои системы: синий экран «BSOD» и kernel panic

При работе ОС возможны случаи сбоев, вплоть до невозможности продолжать работу. Иногда их даже называют катастрофическими сбоями. Пользователи редко их видят, но стоит быть готовыми. Если не вдаваться в различные нюансы, то имеются только две первопричины их возникновения – аппаратные и программные. Аппаратные – это сбои железа, перегрев и прочее. Обычно это гарантийные случаи. Программные – это различные ситуации в работе ОС, установки новых драйверов, не проверенных на совместимость, и прочее.

В зависимости от ОС данная ситуация называется по-разному. В ОС Windows указанная ситуация называется «синим экраном смерти», «звёздным небом», BSOD, blue screen of death. В ОС Linux ситуация сбоя на уровне ядра называется «kernel panic» (или «Ooops» – более лёгкий случай, когда работа может быть продолжена) (см. рис. 5.21).

Рисунок 5.21. Ситуация компьютерного сбоя. На жаргоне «сервер упал». Слева «синий экран смерти» ОС Windows, справа kernel panic в ОС Linux

Если вы обнаружили у себя на экране что-то, схожее с рис. 5.21, то ваши действия будут следующими: сфотографируйте экран цифровым фотоаппаратом или на телефон. В крайнем случае перепишите на листочек всё, что удастся. Это вам пригодится, если обратитесь за помощью к друзьям, интернету или куда-то ещё.

Для ОС Windows важны так называемые STOP-коды. Информацию по ним можно найти, например, на сайте http://bsodstop.ru/kod-bsod, также, введя информацию по но-

меру стопа, можно через поиск попасть на сайт разработчиков ОС, например для кода «STOP 0x0000007B» – http://support.microsoft.com/kb/316401. Часто проблемой бывает последняя установленная программа.

Если вы используете ОС Linux, вам поможет утилита ksymoops, которая с помощью файла System.map (обычно находится в /boot) преобразует различные цифры из Call trace вида [<c022bd67>] [<c021ade0>] … [<c021a93c>] в более понятные разработчикам инструкции ядра. В каждом исходном файле есть в заголовках e-mail, куда можно написать (естественно, на английском) о возникшей у вас проблеме с данным кодом. Если вы будете вежливы и доступно объясните проблему с разумной детализацией, вам постараются помочь.

Если вы думаете, что какая-нибудь набирающая популярность и дорогая MacOS не подвержена сбоям, – ошибаетесь, «падает» всё, и с ней тоже бывают подобные случаи, но они обычно решаются через обращение в службу технической поддержки.

По жизни, за исключением системных администраторов, редко кто занимается тщательным выяснением причин сбоев, чаще пользователи не имеют вообще каких-то важных файлов, а значит, каких-либо веских причин для восстановления системы, поэтому систему довольно быстро переставляют на новую и успокаиваются. Если же ошибка повторяется на «чистой», то есть только что установленной и проверенной ОС, – делают вывод об аппаратных проблемах и запускают тесты или сразу несут железо в гарантию. Если гарантия закончилась – покупают новую деталь вместо сбойной, а последнюю через некоторое время выбрасывают на помойку.

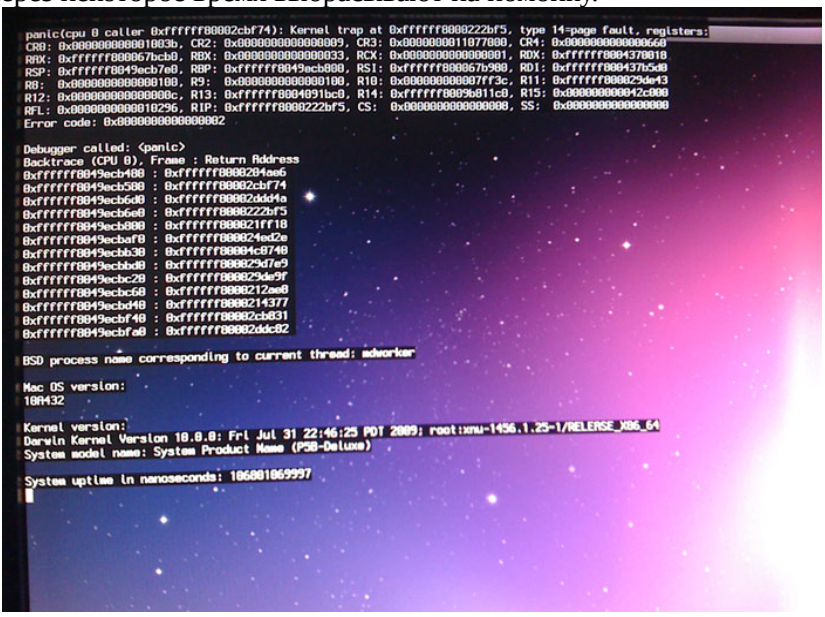


Рисунок 5.22. Сбой ОС MacOS

5.4. Виртуализация, гипервизоры

В желании, чтобы любая программа работала на любом компьютере, нет ничего противоестественного, но чем-то это похоже на борьбу за мир во всём мире. Она идёт постоянно, но счастливое и светлое будущее так и не наступает.

Добиться того, чтобы все программы работали везде, в первом приближении, означает сделать все компьютеры одинаковыми, а это противоречит человеческой природе стремиться к разнообразию и тем самым увеличивать шансы к выживанию.

Хотя компьютер – не биологическое существо, смеем предположить, что будь даже все компьютеры с одним типом процессора, в одних оказалось бы больше памяти, в других меньше (кто-то купил бы с перебором, а кто-то сэкономил бы), как следствие ресурсотребовательные приложения не пошли бы на одном из них. Догадайтесь – каком. Попробуйте запустить какую-нибудь игру, требующую 1 ГБ оперативной памяти и 2 ГБ на жёстком диске на компьютере, где оба параметра в два раза меньше требуемых!

Конечно, когда требуется один компьютер, а у вас их «ноль», виртуализация вам никак не поможет, но если вспомнить, что компьютер есть простой вычислитель, работающий с логическими «0» и «1» внутри себя, а в классической модели Джона фон Неймана ничего не сказано про время этих вычислений (впрочем, и машина Тьюринга «не задумывается» о наличии времени тоже), то, с этой точки зрения, все компьютеры – «братья» («сёстры»), а это означает, что за счёт временных затрат можно обойти практически любые аппаратные требования, фактически «обманув» программное обеспечение, предоставив ему другую вычислительную среду.

Первой универсальной «прослойкой», изолирующей программное обеспечение от аппаратной составляющей компьютера, была и есть операционная система.

На следующем этапе развития виртуализации со стороны пользователя возник вопрос о получении универсальности и совместимости между различными операционными системами, то есть хорошо бы иметь одну систему, где бы работали все программы.

Достижение этой цели было обеспечено двумя решениями:

аппаратная виртуализация	и	программная виртуализация.

Первоначально для этой цели употреблялись термины «эмуляция», «эмуляция ПО», «эмуляция аппаратной части» или «эмуляция железа», а программы назывались эмуляторами. Эмуляторы обычно запускались на более производительных компьютерах и эмулировали работу более медленных. Например, эмулятор компьютера ZX Spectrum для IBM PC [292].

Программы-эмуляторы используются и по сей день. Вычислительная мощность настольных ПК значительно выше мощности большинства планшетов и телефонов, поэтому можно встретить эмуляторы таких ОС, как Android [293] и iOS, например iPadian [294] и др. Ограничения на пути распространения и использования таких программ больше не технические, а юридические и административные из разряда лицензионных, а также нежелание производителей раскрывать какие-либо подробности программной или аппаратной реализации их устройств.

[292] Более сотни различных эмуляторов компьютера ZX Spectrum: http://ru.wikipedia.org/wiki/Список_эмуляторов_ZX_Spectrum.
[293] Эмулятор ОС Android: http://developer.android.com/tools/devices/emulator.html.
[294] Эмулятор iPad для ОС Windows: http://www.xpadian.com/, http://code.google.com/p/ipadian/.

5.4.1. Проблемы

Среди производителей закрытого коммерческого ПО возникла тенденция к намеренному сокрытию алгоритмов работы (как для защиты от копирования, так и из-за возможного наличия различных недокументированных возможностей) для чего активно используются антиотладочные приёмы и шифрование кода. С целью защиты от «виртуализации» закрытое ПО специально «привязывается» к серийным номерам (идентификаторам) и другим характеристикам аппаратной части ЭВМ. Программы отказываются работать при отсутствии постоянного подключения к сети интернет.

Зачастую ПО специально привязывается к серийным номерам и другим характеристикам аппаратной части, даже могут использоваться криптографические алгоритмы, проверяющие целостность ПО на аппаратном уровне. Попробуйте с не сохранённым вовремя SHSH-блоком установить на iOS-устройство предыдущую версию операционной системы. Если на ПК можно очистить жёсткий диск и начать «с нуля», то с большинством устройств (например, использующих ОС iOS) такая история «не прокатит». Также, на мобильном рынке в ряде моделей зарубежных телефонов и планшетов просматриваются случаи использования прозрачного шифрования оперативной памяти в устройствах (во избежание «горячего» подключения к работающим микросхемам памяти с целью восстановления алгоритмов функционирования, например для совершения того же jailbreak'a).

В защите программ от исследования (на факт восстановления алгоритма их работы) на сегодня можно выделить по крайней мере два направления:

противодействие дизассемблированию и противодействие отладчикам.

Рассмотрим проблемы связанные с противодействием отладчикам, поскольку виртуализация является технологией двойного назначения и, как следствие, может использоваться для отладки программ.

Аналогично предложенной выше классификации виртуализации на аппаратную и программную, все существующие отладчики можно разделить на две категории – первые используют отладочные средства процессора, а вторые самостоятельно эмулируют процессор, полностью контролируя выполнение «подопытной» программы.

Качественный эмулирующий отладчик отлаживаемому коду ни обнаружить, ни обойти невозможно, но полноценных эмуляторов на сегодняшний день нет и вряд ли они появятся в обозримом будущем. Данное направление скорее является перспективным направлением научных исследований. С точки зрения рядового программиста можно задаться вопросом, а есть ли смысл их создавать? Ведь, современные микропроцессоры (например фирмы Intel) предоставляют разработчику богатейшие отладочные возможности, позволяющие контролировать даже привилегированный код! Они поддерживают пошаговое исполнение программы, отслеживают выполнение инструкции по заданному адресу, контролируют обращения к заданным ячейкам памяти (или портам ввода-вывода), сигнализируют о переключениях задач и т. д.

Однако не всё так просто, если бит трассировки регистра флагов установлен, то после выполнения каждой машинной инструкции автоматически генерируется отладочное исключение INT1, передающее управление отладчику. Отлаживаемый код, конечно, может обнаружить трассировку, проверив регистр флагов, поэтому для соб-

ственной невидимости отладчик должен распознавать команды чтения регистра флагов и эмулировать их выполнение, возвращая нулевое значение флага трассировки.

Так что, смысл исследований в данном направлении (создание универсального не отслеживаемого отладчика/среды виртуализации) есть. Это вопрос престижа страны, сродни тем, «А зачем нам атомная бомба?» или «Почему стоит побеждать в олимпиадах, чемпионатах мира по футболу и другим видам спорта?».

Рассмотрим используемую терминологию и возможности ряда современных программ виртуализации.

5.4.2. Эмуляция

В отличие от виртуализации, эмуляция – это более широкий термин подразумевающий следующее определение:

Эмуля́ция (от англ. emulation) – комплекс программных и/или аппаратных средств, предназначенный для копирования (или эмулирования) функций одной вычислительной системы на другой.

В ряде случаев, реализация эмуляции может быть аппаратной. Например, когда один принтер эмулирует возможности другого. Список различных существующих эмуляторов довольно широк, поэтому нет смысла даже пытаться приводить его целиком, получится ещё одна книжка. Приведём лишь один любопытный пример. Так, для iOS существует программа iDOS (доступна через AppStore) – эмулятор ОС DOS, основанный на Dosbox, упомянутом в разделе 5.3.1.1 «*История Windows*». Благодаря нему владельцы планшетов могут запускать различные игры, рассчитанные под MS-DOS.

Виртуальная машина – это попытка создать некоторую виртуальную аппаратную среду (чаще x86 совместимую), для того чтобы (в ней) можно было установить ещё одну операционную систему и в ней работать.

Новые термины:

Хостовая ОС

Гипервизор

Гостевая ОС

Программное обеспечение называется виртуализирующим ПО. В процессе работы его часто называют ***гипервизор***ом (hypervisor) или **монитором виртуальных машин** (virtual machine monitor, VMM). Компьютер, на который оно ставится, называется хостовым, а система – хостовой ОС. В некоторых случаях хостовая ОС может не использоваться [295], так как её функционал входит в состав виртуализирующего ПО. ОС, запускаемая внутри виртуальной машины, называется виртуальной ОС. В общем случае, если ресурсов хватает, одновременно (либо в режиме разделения времени, в случае одного процессора с одним потоком) может работать несколько виртуальных машин, не мешая друг другу.

Примерами таких программ для конечных пользователей можно назвать свободное ПО Oracle VirtualBox (ранее Sun Microsystems VirtualBox) и коммерческую программу для домашнего использования VmWare Workstation, а также бесплатный VmWare Player. Предпоследняя, заметим, существует на рынке не первое десятилетие.

Все три программы имеют удобный графический интерфейс. В качестве хостовой ОС могут использоваться ОС Linux, Windows и др., поддерживаются как 32-, так и 64-битные процессоры и ОС.

[295] Например, ПО VmWare ESXi Server ставится непосредственно на компьютер, без использования ОС.

Существуют и другие программные продукты типа bochs [296], qemu [297], Virtual PC [298], Parallels [299] и прочие.

VirtualBox, bochs и qemu – свободное ПО. Эти пакеты наиболее перспективны для различных модернизаций и верификации их исходного кода. Подробнее две из них будут рассмотрены ниже.

Замечание. Для пользователя виртуализация может быть полезна как для проверки работы сомнительного ПО, так и для посещения сомнительных сайтов в интернете без ущерба основной ОС, так, в случае заражения вирусом виртуальной ОС её всегда можно довольно быстро «откатить» к ранее сделанному снимку.

Гостевая ОС – ОС, запущенная под виртуализирующим ПО.

Иногда говорят, что запуск виртуальной ОС происходит в песочнице (sandbox), хотя в большой мере это относится к возможностям некоторых ОС по ограничению функционала запускаемого в этих ОС ПО. Сходным по смыслу термином является «jail» (наверняка вы слышали производные этого термина, такие как jailbreak). С некоторой натяжкой под написанный выше смысл подойдёт и программа chroot, используемая в ОС Linux.

Наиболее активно виртуализация начала развиваться с появлением аппаратной поддержки в виде дополнительных команд на уровне процессоров AMD и Intel. После этого она начала широко использоваться в промышленности. Например, если ранее хостинг-провайдеры предлагали лишь пользовательские аккаунты на какой-нибудь ОС FreeBSD, где был установлен веб-сервер apache, сервисы FTP и SSH, СУБД MySQL или PostgreSQL, то сейчас всё чаще предлагаются готовые виртуальные машины, иногда даже с предустановленным ПО. Это удобно, так как пользователь может настраивать свою ОС как хочет, компилировать модули ядра, оптимизировать работу, даже использовать собственные программные разработки.

Замечание 1. Узнать запущена ли ваша ОС как гостевая в ОС Linux или нет можно с помощью консольной утилиты virt-what.

Замечание 2. С юридической точки зрения установка ещё одной ОС упирается в лицензионное соглашение этой ОС. Если виртуальная ОС коммерческая, то наряду с выделением виртуальных ресурсов с предустановленными ОС пользователю могут предлагаться для покупки и лицензии, чтобы «de jure» не было вопросов по защите авторских прав на используемое ПО. В этом плане свободное ПО намного лучше проприетарного, так как не требует лишних лицензионных затрат. Так, лицензия GNU Public License вообще не ограничивает число установок, а если вы купили техническую поддержку для свободного ПО, то без разницы, где и как вы будете устанавливать это ПО.

Использовать виртуальные ресурсы выгодно обеим сторонам. Провайдеры могут за счёт статистического уплотнения экономить средства. Этот принцип из теории массового обслуживания давно известен и регулярно используется. Так, из 1000 людей, купивших проездной билет на автобус, вряд ли все сразу поедут в одно время.

Чем крупнее провайдер, тем больше прибыль он может получить и тем больше у него преимущества. Несмотря на то что ночью люди спят (нагрузка должна быть меньше), сказывается тот факт, что земля круглая и имеется много часовых поясов. Так,

[296] Bochs IA-32 Emulator Project http://bochs.sourceforge.net/.
[297] Open source machine emulator and virtualizer http://qemu.org.
[298] Эмулятор http://www.microsoft.com/rus/windows/virtual-pc/.
[299] Эмулятор для MacOS http://www.parallels.com/ru/.

крупные дата-центры оказываются загружены практически равномерно за счёт того, что днём одни обслуживают одни страны (и даже континенты, где в это время день), а ночью другие (где в это время тоже день).

Пользователю удобно то, что ему не надо заниматься обслуживанием аппаратуры (электропитание, вентиляция, регламентные работы и прочее). При этом он может получать на необходимое ему время достаточно производительные ресурсы. (Иногда это называют – вынести на out source.) Подоспели за «своим куском пирога» и производители коммерческого ПО, они зачастую предлагают лицензии на ПО не навсегда, как ранее (то есть один раз купил – и пользуйся программой всю жизнь), а на месяц, год и прочие периоды. Появилась даже группа новых терминов, означающих подробное временное предоставление услуг: SAAS (от software as a service), PaaS (от program as a service) и др.

С технической точки зрения прогресс по виртуализации остановить невозможно. С моральной – вопрос коммерциализации и защиты персональных данных нуждается в дополнительной проработке.

5.4.3. Облака

Количество разнообразных гипервизоров за последние годы выросло в разы: OpenVZ (http://openvz.org/), Hyper-V, Xen (http://xenproject.org/), oVirt (http://www.ovirt.org/), K virtual machine (KVM, http://java.sun.com/products/cldc/wp/) и т. д., а их функционал сильно расширился. Например, новым стало то, что некоторые гипервизоры научились работать не только на одном компьютере, но и с целыми сетями распределённых ЭВМ. Так, работая с системой компьютеров, конечный пользователь может и не знать, что в процессе работы какой-либо обслуживающий его запросы виртуальный компьютер может быть прозрачно для него и этого виртуализированного компьютера перемещён в рабочем состоянии с одного гипервизора (в одном дата-центре) на другой (в другом) без потери данных и даже каких-либо временных состояний типа сессий и прочего. После внедрения таких «фокусов» в жизнь появились термины, их обозначающие: «облако», «облачный сервис», «облачное хранение», которые в полной мере отображают суть происходящего. Дословно, если ранее на картинках и блок-схемах, когда не требуется выборочная детализация, интернет изображался в виде некого серого облака, то сейчас и весь обслуживающий запросы клиентов серверный функционал сам превратился в облако на схеме, так как заведомо точно сказать, какая именно из множества аппаратных составляющих будет отвечать на (обслуживать) запросы клиента в данный момент времени, в большинстве случаев определить невозможно и не требуется.

Вслед за распределением вычислительных ресурсов в «облаке» стало возможно хранить данные в виде файлов. Появились отдельные файловые сервисы типа «Яндекс.Диск», «DropBox» и др. В ряде программных коммерческих продуктов появилась возможность сохранять файлы сразу в облако, в том числе хранить в облаке контакты записной книжки, а точнее e-mail-адреса и телефоны.

5.4.4. Недостатки облачных сервисов

У облачных сервисов есть и оборотная сторона: во-первых, они подразумевают, что у пользователя будет постоянное высокоскоростное соединение с интернетом как со средой, обеспечивающей доступ к облаку из любой точки в любое время. Как вы понимаете, если даже в мегаполисах покрытие не 100%, достаточно спуститься в какой-нибудь подземный гараж или попытаться воспользоваться сервисом передачи данных в местах большого скопления людей, под Новый год, после салюта в праздничный день – сеть не выдерживает нагрузки, не говоря уже о том, что по протяжённости наша страна сравнима с днём полёта на самолёте и другой такой страны авторы не знают. Не то чтобы интернета или сотовых вышек – дорог и электричества у нас местами нет.

Во-вторых, не ясно, как быть с хранением конфиденциальных данных. Нет никакой гарантии, что ваши коммерческие секреты завтра не окажутся у конкурентов просто потому, что фирма, поддерживающая облачный сервис, решит продать контрольный пакет своих акций, чтобы не разориться. А их запросто купит ваш конкурент. (Простите, но в бизнесе нет морали, каждый «берёт от жизни всё» и «крутится как может».) Не зря многие коммерческие фирмы проявляют скепсис в отношении предлагаемых облачных сервисов и предпочитают иметь свои центры обработки данных, тратя на это миллиарды рублей. Как мы понимаем, о хранении государственных секретов в облаках и говорить не стоит.

В-третьих, где гарантии? Прочитайте большинство соглашений, и вы откроете для себя, что вы пользуетесь всеми этими сервисами на свой страх и риск и никто конкретно за ваши данные не отвечает. Если пропадёт переписка подростков – Бог с ней, скорее всего, она не настолько важна, а если это SMS клиенту о переносе рейса или задержке поставок? В современном мире вероятность форс-мажора очень велика, в том числе и чисто по политическим соображениям. Хотите гарантий – страхуйте свой риск отдельно либо имейте свой собственный сервис. Облако хорошо использовать лишь в том случае, если оно целиком расположено на территории вашей страны, политический режим стабилен и вы владелец оборудования, а также вам принадлежат линии связи и электростанции, обслуживающие центры обработки данных, где «расположено» облако.

5.4.5. QEMU

Поскольку одной из сильных сторон ОС Linux является её кросс-платформенная природа: вы можете запускать её на различных аппаратных платформах x86, x86_64, SPARC, PowerPC и на многих платформах с другими процессорами, наибольший интерес для использования представляет эмулятор qemu. Он может работать как обычный отдельный процесс ОС (то есть без взаимодействия с ядром ОС), так и с использованием ускоряющих модулей ядра. Возможна эмуляция следующих платформ (процессор + железо):

- PC (x86 or x86_64 processor)
- ISA PC (old style PC without PCI bus)
- PREP (PowerPC processor)
- G3 Beige PowerMac (PowerPC processor)
- Mac99 PowerMac (PowerPC processor, in progress)

- Sun4m/Sun4c/Sun4d (32-bit Sparc processor)
- Sun4u/Sun4v (64-bit Sparc processor, in progress)
- Malta board (32-bit and 64-bit MIPS processors)
- MIPS Magnum (64-bit MIPS processor)
- ARM Integrator/CP (ARM)
- ARM Versatile baseboard (ARM)
- ARM RealView Emulation/Platform baseboard (ARM)
- Spitz, Akita, Borzoi, Terrier and Tosa PDAs (PXA270 processor)
- Luminary Micro LM3S811EVB (ARM Cortex-M3)
- Luminary Micro LM3S6965EVB (ARM Cortex-M3)
- Freescale MCF5208EVB (ColdFire V2)
- Arnewsh MCF5206 evaluation board (ColdFire V2)
- Palm Tungsten|E PDA (OMAP310 processor)
- N800 and N810 tablets (OMAP2420 processor)
- MusicPal (MV88W8618 ARM processor)
- Gumstix "Connex" and "Verdex" motherboards (PXA255/270)
- Siemens SX1 smartphone (OMAP310 processor)
- AXIS-Devboard88 (CRISv32 ETRAX-FS)
- Petalogix Spartan 3aDSP1800 MMU ref design (MicroBlaze)
- Avnet LX60/LX110/LX200 boards (Xtensa)

а также эмуляция лишь одних инструкций процессоров следующих семейств: x86 (32 и 64 бита), PowerPC (32 и 64 бита), ARM, MIPS (32 бита только), Sparc (32 и 64 бита), Alpha, ColdFire(m68k), CRISv32 и MicroBlaze.

Для специалиста эмулятор qemu довольно не сложно собрать из исходных кодов (доступны ОС Linux, Windows и Mac OS X) и запустить из консоли. При использовании возможностей кросс-платформенной библиотеки SDL (Simple DirectMedia Layer) отображение информации от виртуализируемого ПО или аппаратуры может производиться в графическое окно (с рамкой или без), целиком на X-Window сервер (на весь экран), так и никуда. В последнем случае доступ к изображению возможно получить удалённо посредством протокола VNC (qemu выступает в роли VNC-сервера).

Не специалисты могут воспользоваться менее продвинутой средой виртуализации – VirtualBox, зато позволяющей её настраивать посредством графического интерфейса.

5.4.6. VirtualBox

VirtualBox – свободный и доступный гипервизор. Для большинства пользователей это программный продукт, обеспечивающий виртуализацию для ОС Windows, Linux, FreeBSD, Mac OS X и др. После покупки Sun Microsystems фирмой Oracle полное название стало «Oracle VM VirtualBox».

Скачать последнюю версию можно по адресу https://www.virtualbox.org/wiki/Downloads.

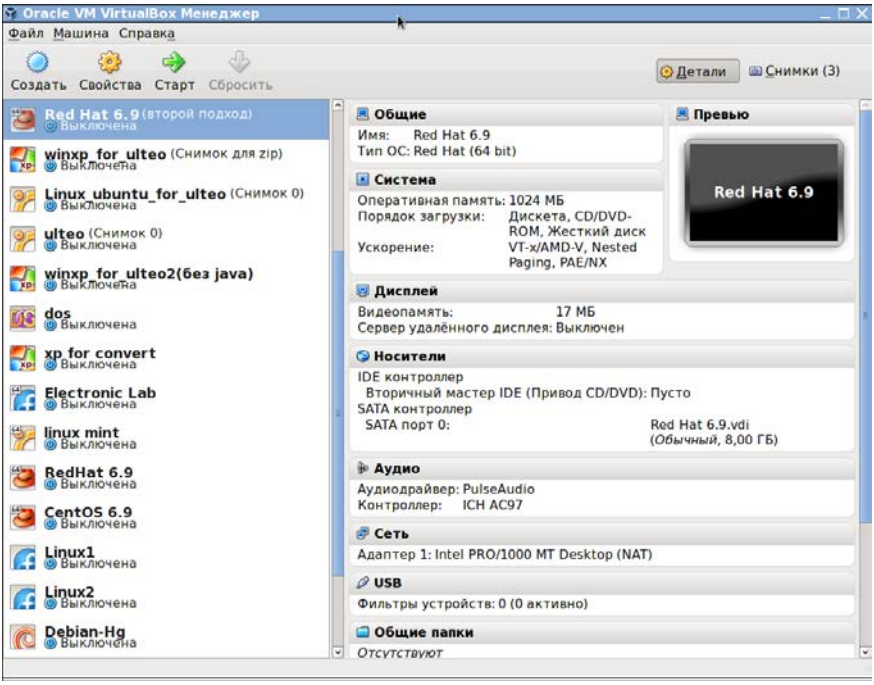


Рисунок 5.23. Менеджер виртуальных машин VirtualBox

После установки программы вы сможете создать «виртуальную машину», выбрав её кофигурацию несколькими щелчками мыши. Образ жёсткого диска будет храниться в виде .vdi-файла на хостовой ОС.

Включаем виртуальное питание, и гостевой компьютер начнёт загружаться. Виртуальный привод оптических дисков можно связать как с iso-образом диска, хранящимся в файле, так и сделать связь с физическим приводом хостовой машины.

Виртуальный жёсткий диск можно разбить, отформатировать, установить операционную систему и другие программы уже поверх системы. Внутрь виртуального компьютера можно пробросить подключение реальных USB-устройств, а также организовать виртуальную сеть между несколькими виртуальными ЭВМ. Начать работать с программой не так сложно. В интернете можно найти русскоязычную документацию и статьи, полагаем, что большинство читателей с этим справится самостоятельно.

Обратите внимание на следующие моменты:

По завершении установки программы скачайте и установите VirtualBox 5.x Oracle VM VirtualBox Extension Pack от вашей версии. Делается это через меню **Файл ► Свойства ► Плагины** и значок добавления. После установки вы сможете использовать в виртуальных машинах реальные порты USB 2.0, а также будут доступны виртуальные альтернативные сетевые карты с возможностью удалённой сетевой загрузки.

После установки гостевой ОС установите в неё пакет гостевых дополнений. Делается это через меню виртуальной машины **Устройства ► Установить Дополнения гостевой ОС...** на запущенной виртуальной ОС. Это позволит пользоваться общими сетевыми папками, в графическом режиме растягивать окно виртуальной машины мышкой до нужного размера и прочее.

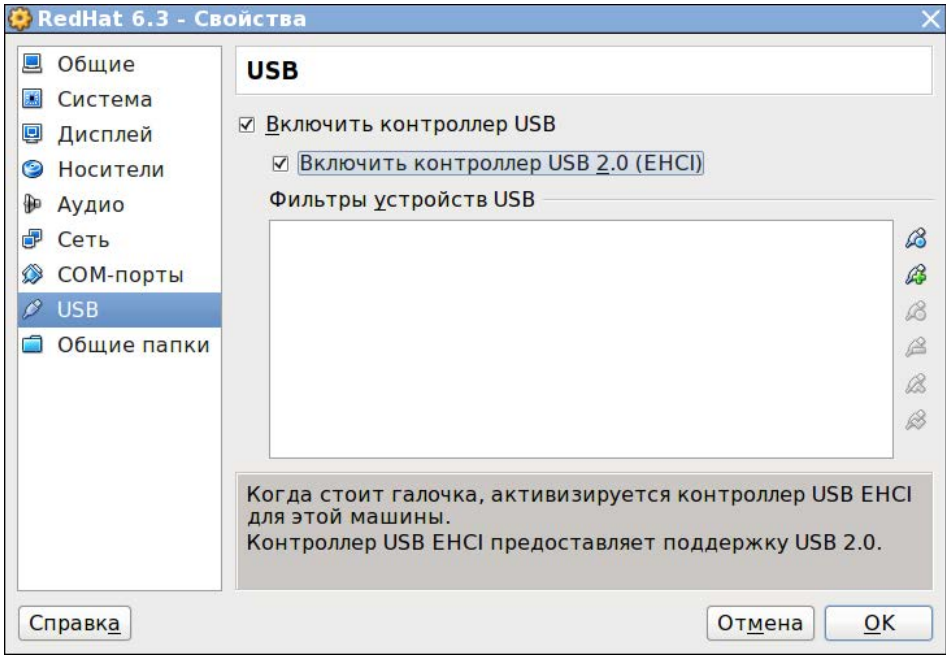
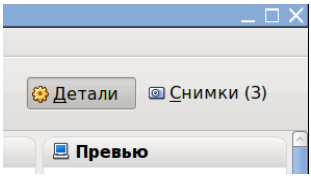


Рисунок 5.24. Включение поддержки USB 2.0

5.4.6.1. Снимки



Полезное свойство любой виртуальной машины – это возможность «откатить» систему к предыдущему, ранее сохранённому состоянию. Для этого имеется возможность создавать сохранённые состояния виртуальной машины называемые «снимками». По завершении работы гостевой ОС обратите внимание на наличие в менеджере виртуальных машин в правом верхнем углу кнопки-значка «***Снимки***», рядом с «Детали». Обязательно воспользуйтесь указанной функцией, для этого зайдите во вкладку снимков на интересующей вас виртуальной машине и используйте значок фотоаппарата. Тем самым вы сможете сделать снимок текущего состояния виртуальной машины. Впоследствии, в случае если внутри виртуальной машины произойдёт сбой, запустится вирус или что-то удалится, у вас всегда будет возможно вернуться к ранее сделанному снимку. По сути, это создание резервной копии виртуальной машины, но работает гораздо быстрее, чем реальное резервное копирование.

Замечание 1. Изменения в файл-образе жёсткого диска виртуальной машины после создания снимка делаются в отдельном файле, таким образом, возможно на базе одного образа создать несколько вариаций, например с разными версиями одной и той же программы, которые не могут быть установлены одновременно вместе. Как вы понимаете, откат к ранее созданному снимку проходит как удаление файла с изменениями, что работает довольно быстро.

Замечание 2. Файл, описывающий конфигурацию виртуальной машины (с расширением .vbox), есть текстовый файл в формате XML. При желании его можно открыть «Блокнотом» и подправить руками. Созданные образы виртуальных машин (файлы .vbox и .vdi) можно скопировать (при выключенных машинах), а после использовать на другом компьютере, где также будет установлена программа VirtualBox. При наличии быстрого USB 3.0 внешнего

носителя данная функция даст вам уникальную возможность иметь свой «виртуальный компьютер» с необходимым и заранее настроенным программным обеспечением всегда с собой. При этом другой компьютер может быть с другой хостовой ОС.

5.4.6.2. Дополнительные настройки

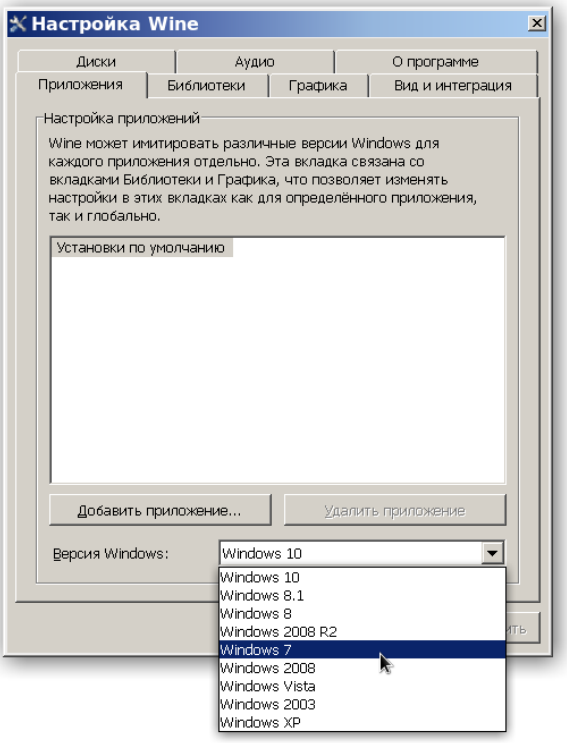
У программы виртуализации VirtualBox существует множество полезных настроек которые не доступны через пользовательское меню, но доступны через консольную утилиту `VBoxManage`. Подробнее о всевозможных настройках можно прочитать в устанавливаемом с программой пользовательском руководстве[300].

5.4.7. Wine и DOSBox

Другим примером, на сей раз программной эмуляции, могут выступать популярные и свободные программы wine[301] и DOSBox[302].

Wine – это свободное ПО, позволяющее пользователям UNIX-подобных систем архитектуры x86 (и других архитектур, при наличии совместимости, например, AMD64) исполнять 16-, 32- и 64-битные приложения ОС Windows. Wine также предоставляет программистам библиотеку программ Winelib, при помощи которой они могут компилировать Windows-приложения для портирования их в UNIX-подобные системы. Название Wine является рекурсивным акронимом и расшифровывается «Wine Is Not an Emulator» – «Wine – не эмулятор» (имеется в виду, что Wine не является эмулятором компьютера, как, например, qemu или VirtualBox, Wine – это альтернативная реализация Windows API).

Для запуска программ MS-DOS, использующих интерфейс командной строки для взаимодействия с пользователем, в ОС Linux может использоваться утилита wineconsole (из комплекта Wine), либо отдельная программа-эмулятор DOSBox (см. рис. 5.3).



[300] /usr/share/doc/VirtualBox-5.xxx/UserManual.pdf, http://download.virtualbox.org/virtualbox/UserManual.pdf
[301] http://www.winehq.org
[302] http://www.dosbox.com

5.5. Офисный пакет LibreOffice

Офисный пакет LibreOffice в последнее время набирает популярность и используется всё чаще.

Во-первых, это связано с тем, что это действительно мощное программное средство, позволяющее создавать документы различной сложности (тексты, таблицы, диаграммы, презентации, формы по работе с базами данных, формулы).

Во-вторых, формат «**Open Document**» (расширение «**.odt**»), используемый по умолчанию для сохранения документов в LibreOffice, является утверждённым и действующим национальным стандартом для офисных приложений на территории Российской Федерации: ГОСТ Р ИСО/МЭК 26300–2010 [303].


ФЕДЕРАЛЬНОЕ АГЕНТСТВО
ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСТ Р ИСО/МЭК 26300— 2010

В-третьих, это продукт, основанный на пакете OpenOffice и во многом с ним совместимый. Функционал, содержащийся в пакете, проверен годами, и в него вошли только те функции, которые реально востребованы большинством. Пакет постоянно обновляется, появляются новые востребованные функции, регулярно исправляются найденные ошибки. Не скроем нашу симпатию к данному продукту хотя бы потому, что данная книга полностью свёрстана в пакете LibreOffice Writer.

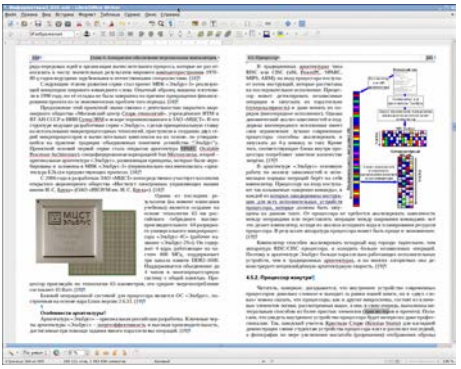
В-четвёртых, данный продукт существует для многих систем, будь то ОС Linux, Windows или MacOS X. Файлы, создаваемые в разных ОС, полностью между собой совместимы. Исходя из этого, данный продукт идеален для школ, потому как в последнее время в них можно встретить все три операционные системы, указанные выше.

В-пятых, он бесплатен и имеет исходные коды. Не нужно за него платить ни разово, ни раз в 365 дней. Бесплатно скачать LibreOffice можно по адресу www.libreoffice.org.

5.5.1. Подготовка текстовых документов в LibreOffice Writer

Текстовый процессор LibreOffice Writer может быть использован для подготовки документов сложной структуры, состоящих из большого количества глав, разделов, подразделов, имеющих таблицы, сложные математические, химические и прочие формулы, рисунки и диаграммы, которые могут создаваться в данной системе или добавляться как объект другой системы с помощью технологии OLE.

[303] ГОСТ Р ИСО/МЭК 26300-2010
http://protect.gost.ru/document.aspx?control=7&id=177075.



Эти документы должны соответствовать определённым стандартам, обязательным при оформлении деловых документов и научных отчётов, или строгим правилам оформления статей для различных журналов, текстов научных монографий для издательств, стилю оформления вебстраниц различных сайтов. Требования стандартов на научные отчёты и требования различных издательств могут сильно отличаться друг от друга. В связи с этим высокую актуальность имеет правильная технология подготовки документов с использованием стилей оформления и шаблонов документов.

Поясним подробнее о преимуществах оформления стилями. В целом начать работать с редактором несложно. Если вы немного сообразительны и имеете вкус, то вы быстро разберётесь и отформатируете текст довольно красиво, что-то выделите жирным, что-то отодвинете путём вставки пустых строк, где-то установите курсив. Когда вы делаете один небольшой документ в месяц, это быстро и просто, и так делает большинство.

Проблемы начинаются тогда, когда вы осознаёте, что LibreOffice Writer является редактором WYSIWYG [304], а это значит, что для правки текста необходимо установить курсор в то место документа, куда вы хотите внести изменения, а далее нажимайте клавиши и редактируйте – доступно даже первокласснику. Это может быть удаление, добавление или замена символов, перемещение или удаление части текста либо изменение элементов форматирования.

Перемещение курсора по тексту осуществляется или щелчком мыши в нужное место, или многократным нажатием «стрелочек» на клавиатуре. Удаление символов происходит с помощью клавиш «**Delete**» и «**Backspace**» (стрелка влево), причём первая удаляет один символ справа от курсора, а вторая – слева. Если при этом удерживать нажатой клавишу **Ctrl**, то произойдёт удаление целого слова, соответственно справа или слева от курсора. Во время редактирования текстов, будь то набор новых символов или удаление существующих, происходит запись изменений в журнал. Выделенная для хранения документа и его частей память, определяется через меню **Сервис ► Параметры ► Память**, см. рис. 5.25.

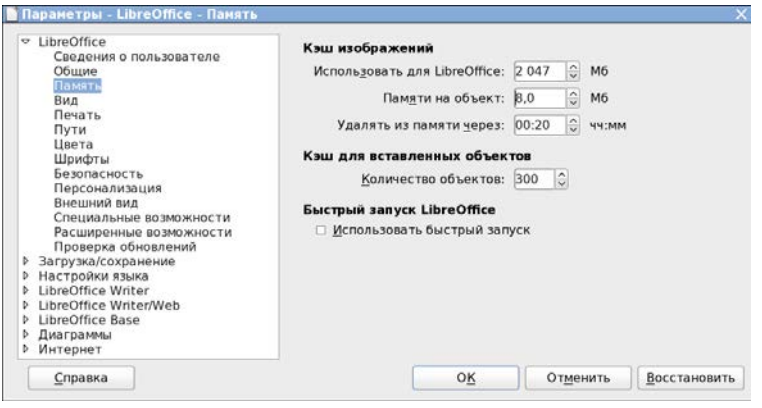


Рисунок 5.25. Окно изменения размеров памяти, выделяемой для работы с документом

[304] От англ. *What You See Is What You Get* – класс программ, в которых сразу создаётся изображение в том виде, как оно получится при печати на бумаге.

Если ваша система позволяет (в большинстве случаев это именно так), то рекомендуем увеличить используемые по умолчанию значения. В LibreOffice версий 4.x в этом же окне можно было указать количество запоминаемых шагов, для возможной отмены выполненных действий. В версиях 5.x и выше данный параметр убран, изменить его можно только через параметр «org.openoffice.Office.Common/Undo Steps» «экспертных настроек» куда можно попасть через меню **Сервис ► Параметры ► LibreOffice ► Расширенные возможности ► Открыть экспертные настройки** (Tools ► Options ► LibreOffice ► Advanced ► Expert Configuration, см. рис. 5.26).

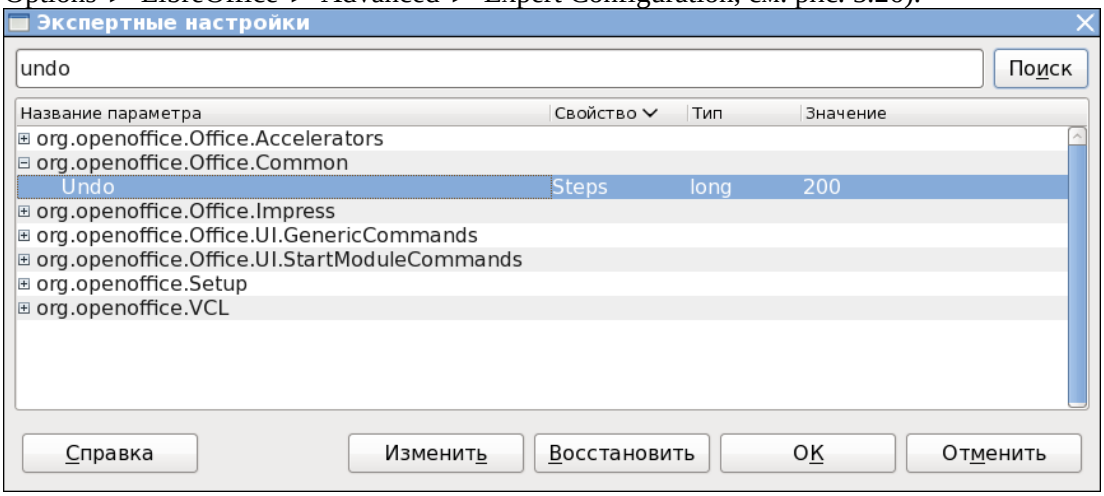
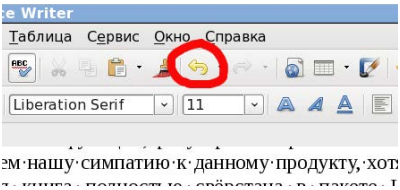


Рисунок 5.26. Окно экспертных настроек (строка поиска ограничивает вывод списка всех параметров)

Соответственно, чтобы вернуть удалённый по ошибке текст или «откатить» последние изменения, выполните **Правка ► Отменить**, или нажмите соответствующую кнопку **Отменить** на панели инструментов, или сочетание клавиш **Ctrl**+**Z**.



Представьте ситуацию: вы закончили с набором и приносите ваш красивый документ кому-либо «на критику», в ответ слышите замечания, что в этом году требования по оформлению опять поменялись и поля у курсовой работы должны быть чуть больше, после рисунков увеличился отступ, и теперь он должен быть не 5 мм, а 10, вместо курсива в заголовках следует использовать обычный шрифт, но на 2 кегля больше. Да и «вот эти три главы» надо поменять местами, после чего надо будет переделать и содержание со страницами, так как нумерация «поедет».

Сразу отметим, что далее большинство тех, кто не знаком со стилями оформления, идёт по самому длинному пути. А именно они начинают править все 20 или 30 заголовков, сначала выделяют мышью, потом в меню (хорошо, если знают о горячих клавишах) снимают курсив и меняют размер, затем после каждого из примерно сорока рисунков в документе жмётся несколько раз клавиша {**Enter**}, чтобы отступ был больше и т. п. Потом меняются главы местами, кропотливо исправляются заголовки и переставляются страницы в содержании. Документ на просмотр и в печать... Один раз такое пережить можно, но если вы будете что-то править постоянно, то эффективность работы значительно уменьшится. Чудное решение – это продуманность документа.

Пускай в нём будут два типа теста – «обычный текст» и «заголовки». С такими же именами можно создать и два стиля. В этом случае вы каждый абзац текста (или несколько абзацев) отмечаете своим стилем «обычный текст» или «заголовок». Может показаться, что это сложнее, зато после, если надо исправить заголовки, то вы не исправляете каждый из 30, а идёте в редактор стилей и там один раз правите стиль. После этого редактор сам находит все заголовки и исправляет их за вас. Аналогично если рисунки отмечены стилем «рисунок» и нужно поменять отступ снизу, то делается это в оформлении стиля, а не жмётся много раз {**Enter**} после каждого рисунка. Причём, выискивая глазами, где ещё вы не добавили пробелы, обязательно где-нибудь пропустите, а незамыленный взгляд проверяющего обязательно это место заметит и укажет исправить.

Если мы вас убедили в том, что ваша жизнь может упроститься с использованием стилей при оформлении документов, а вы захотели продолжить чтение данного раздела, то давайте изучим эти вопросы подробнее.

5.5.1.1. Стили оформления

Стиль – именованный набор параметров оформления абзаца, знака или таблицы.

Стиль определяет вид шрифта, параметры абзаца, табуляции, границ; язык, текст без рамки или в рамке, абзац без нумерации или нумерованный и горячие клавиши задания данного стиля (см. рис. 5.27).

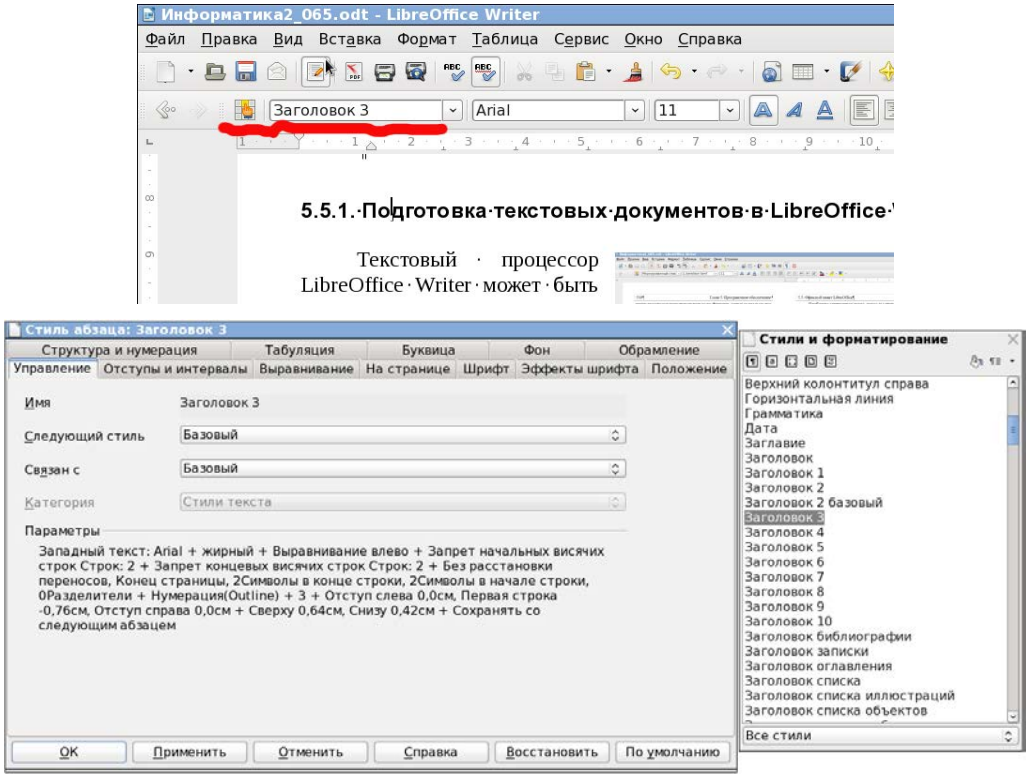


Рисунок 5.27. Окно настройки стилей

Следует знать и правильно применять основные параметры абзацев в стилях: выравнивание (по левому краю, по центру, по правому краю, по двум сторонам), отступы (слева, справа, первой строки), интервалы (перед абзацем, после него, межстрочный), а также положение на странице – запрет отрывать первую или последнюю строки, разрывать абзац между страницами, переносить слова по слогам, нумеровать строки; начинать абзац с новой страницы (см. рис. 5.28).

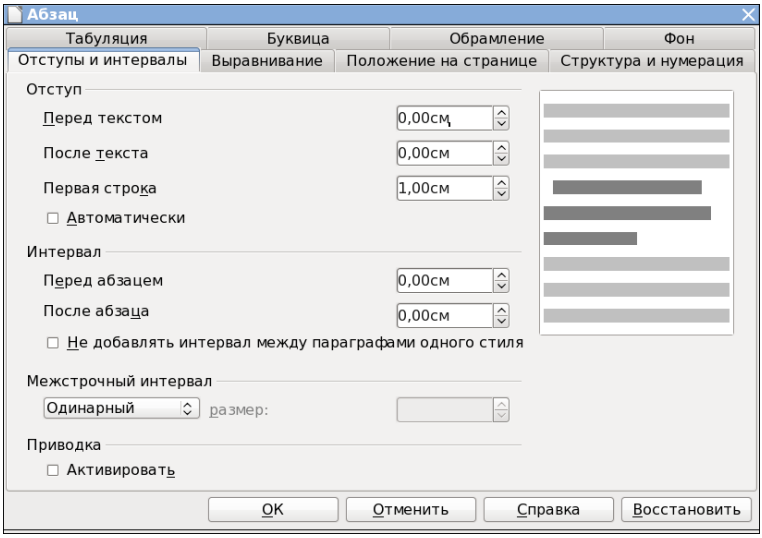


Рисунок 5.28. Окно настройки параметров абзаца

5.5.1.2. Создание документа

Чтобы не наступать на грабли, рассмотрим методику правильного создания документа, оформленного стилями.

При создании нового документа в приложении LibreOffice Writer команда **Файл ► Создать** открывает диалоговое окно с вариантами:

- Тестовый документ;
- Электронную таблицу;
- Презентацию;
- Рисунок;
- Базу данных;
- Документ HTML;
- Документ XML form;
- Составной документ;
- Формулу;
- Этикетки;
- Визитные карточки;
- Шаблоны.

Выбираем «Текстовый документ». Далее необходимо:

1. Задать параметры страницы

В меню **Формат ► Страница**… следует задать для страницы размер, ориентацию и поля – левое, правое, верхнее и нижнее. При необходимости можно задать количество колонок на листе, цвет страницы и текст подложки, колонтитулы и прочее.

2. Определить параметры необходимых стилей

Делается это клавишей **F11** или через меню **Формат ► Стили.** Вид окна стилей, как и кнопка меню, через которую его также можно вызывать, показаны на рис. 5.27.

Следует учитывать, что список стилей может быть неполный. В выпадающем меню снизу можно выбрать **Все стили**, **Условные**, **Используемые** и др. Полные возможности по настройке стилей даёт команда «**Изменить...**» из диалогового окна, вызываемого щелчком правой кнопкой мыши по имени стиля, см. рис. 5.27, после чего открывается левое нижнее окно.

Пункт **Создать...** (стиль) (правой клавишей мыши по окну) позволяет определить параметры стиля с новым названием на основе текущего.

Для основного текста следует определить параметры основного абзаца. При этом нужно учитывать, что традиционный русский стиль – абзац с красной строкой (первая строка с отступом), без интервалов до и после абзаца, без отступов слева и справа. Стиль некоторых новых журналов навязывает нам западный стиль – первая строка абзаца без отступа, но добавлен интервал после абзаца.

Не следует изменять стиль абзаца по умолчанию «Обычный», который является основой стиля «Обычная таблица», иначе в таблицах могут появиться нежелательные отступы первой строки, слева и справа; интервалы сверху и снизу в ячейках, увеличенный межстрочный интервал и прочее.

Если удалить какой-либо стиль абзаца, то ко всем абзацам, которые были оформлены удалённым стилем, будет применён стиль **Обычный**. Аналогично действует команда **Очистить формат**.

Если вы хотите заменить один стиль на другой, то в меню **Правка ► Найти и заменить** (также вызывается по нажатии **Ctrl**+**H**) следует выбрать «Детали» и поставить галочку «Искать по стилям». В этом случае в первом выпадающем списке выбирается, какой стиль следует заменить, а во втором – на какой.

Если в документе используется много таблиц, можно определить стиль таблицы, который будет использоваться по умолчанию (в меню **Вставка ► Таблицу… ► Автоформат** или в области задач **Стили и форматирование**). После этого при создании таблицы с помощью кнопки **Вставить таблицу** панели инструментов **Стандартная** она будет иметь стиль таблицы по умолчанию.

3. Задать колонтитулы

Если необходимо, можно определить текст верхнего и нижнего колонтитулов. Для этого следует воспользоваться командой меню **Вставка ► Верхний (нижний) колонтитул.** Кроме постоянного текста, колонтитулы могут содержать поля, показывающие информацию текущего файла: номер страницы, название раздела, автора, дату создания, имя файла, всего страниц и прочее. Подробнее можно узнать в справке, вызываемой по клавише **F1**.

Колонтитулы первой страницы, чётных и нечётных страниц могут различаться, что определяется галочкой «*Одинаковое содержимое справа/слева*» во вкладке «**Верхний/Нижний колонтитул**» меню **Формат ► Страница**.

Каждый раздел документа может иметь собственный колонтитул. Это можно использовать для нумерации страниц альбомной ориентации, оформленных как отдельные разделы (например, для широких таблиц или рисунков) при книжной ориентации основных страниц.

4. Приступить к набору текста

Перед началом набора текста можно определить, будет система автоматически переносить слова по слогам при достижении границы строки в абзаце или нет (по

умолчанию переносы отключены). Делается это на вкладке **Сервис ► Язык ► Расстановка переносов**.

Если документ начинается с титульного листа или обложки, на них не следует использовать стили заголовков 1–3 уровня, которые обычно входят в оглавление при его автоматическом формировании.

Основной текст документа обычно начинается с заголовка 1-го уровня, стиль его лучше задать перед началом написания этого абзаца. В конце заголовков точек не ставят, в стилях заголовков следует задать запрет автоматического переноса слов. Если заголовок должен быть написан прописными буквами, это следует задать в настройках стиля для раздела **Сервис ► Параметры ► LibreOffice ► Шрифты**. Часто для заголовков задают отступ слева и справа 1–1,5 см, обычно интервал до абзаца и после, выравнивание может быть по центру или по левому краю. Стиль заголовков 2-го и 3-го уровней может быть основан на стиле **Заголовок 1**, но несколько модифицирован.

Для абзацев основного текста следует задать стиль с отступом первой строки или без него, но с интервалом после абзаца. В тексте могут присутствовать абзацы иного стиля, чем основной текст (например, аннотация в начале статьи или выделенный текст).

При необходимости можно использовать оформление текста в несколько колонок для всего документа или для одного раздела.

При вводе текста параметры абзаца могут быть изменены с использованием меню **Формат ► Абзац**.

5.5.1.3. Формулы

Входящий в состав LibreOffice редактор формул LibreOffice Math позволяет быстро, легко и бесплатно вводить различные математические формулы. Также можно вводить простые химические формулы, не осложнённые бензольными кольцами и хитрыми связями. Редактор формул вызывается непосредственно из меню текстового редактора LibreOffice Writer через меню **Вставка ► Объект ► Формула**. При запуске редактора формул появляется окно набора формулы и панель «**Элементы**» с различными значками. Если она отсутствует в строке меню, выберите команду **Вид → Элементы** (см. рис. 5.29).

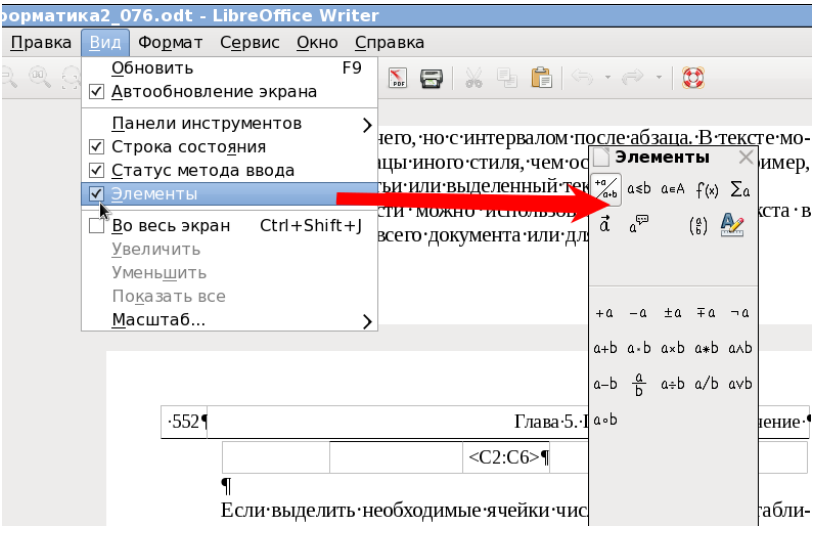


*Рисунок 5.29. Окно **Элементы** для «рисования формул мышкой»*

Как набирать формулы?

Профессионалы не первое десятилетие набирают формулы в LaTeX [305] с клавиатуры, все остальные используют различные распространённые офисные пакеты, где есть «мышка» в помощь начинающим. Ей можно щёлкать по панели со значками, как на рис. 5.28, после чего в окне набора формулы будут появляться символы и их конструкции – простой текст с относительно понятным языком. По мере получения опыта вы сможете сразу набирать нужные формулы с клавиатуры. Несомненно, если использовать все 10 пальцев, процесс будет проходить очень быстро.

Для того чтобы убедиться в простоте текстового формата, приведём несколько соответствий графического вида формул их коду (см. табл. 5.15).

Таблица 5.15. Соответствие формул их «Math-коду»

$x_{1,2}=\frac{-b\pm\sqrt{(b^2-4ac)}}{2a}$	`x_{1,2}={-b +- sqrt(b^2-4 a c)} over {2 a}`
$\mathrm{F}(\alpha,\beta)=\frac{\alpha\sin\alpha-\mathrm{tg}\beta}{\cos\beta}$	`func F(%alpha "," %beta)= {%alpha sin %alpha - "tg" %beta} over {cos %beta}`
$f_{a_i}(x)=f_{a_{i-1}}(x)+\sum_{i=1}^{\infty}g_i(x)$	`f_{a_i}(x)=f_{a_{i-1}}(x)+ sum from{i=1} to{infinity} g_i(x)`
$\boldsymbol{A}=\begin{bmatrix}A_{11} & A_{12} & \cdots & A_{1n}\\ A_{21} & & & A_{2n}\\ \vdots & & & \vdots\\ A_{m1} & A_{m2} & \cdots & A_{mn}\end{bmatrix}$	`font sans bold size *2 A =left[ matrix{A_11#A_12#dotsaxis#A_{1n}##A_21#{} #{}#A_{2n}##dotsvert#{}#{}#dotsvert##A_{m1 }#A_{m2}#dotsaxis#A_mn} right]`
$\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n$	`lim from {n toward infinity} left (1+1 over n right)^n`
$(a+b)^2=a^2+2ab+b^2$	`(a+b)^2=a^2+2ab+b^2`

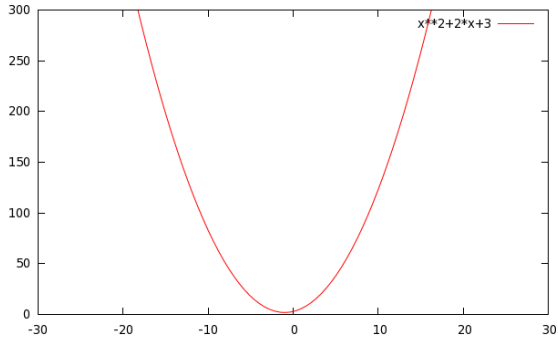
[305] LaTeX (по-русски латéх) – наиболее популярный набор макрорасширений (или макропакет) системы компьютерной вёрстки TeX, который облегчает набор сложных документов. Документ LaTeX – это текстовый файл, содержащий специальные команды языка разметки. LaTeX не так прост и интуитивно понятен, как, например, Microsoft Word или LibreOffice Writer, но, потратив один раз время на его изучение, вы будете настолько удивлены результатом, что перестанете использовать «ширпотреб» при создании текстовых документов.

LaTeX - это стандарт de facto при наборе научных статей, курсовых и дипломных работ, технических спецификаций, учебников и т. д. Статьи ко многим конференциям по математике не принимают ни в .doc, ни в .odt, ни в .pdf, предпочитая .tex. Главным преимуществом латеха является абсолютно одинаковый внешний вид готовых страниц во всех операционных системах и непревзойдённое до сих пор качество полиграфических текстов и математических формул. Кроме этого, скриптовый язык латеха – это универсальный язык для обмена формулами. Математики из разных стран легко понимают друг друга, если пишут формулы на этом языке. Во многих математических пакетах, например Maple, Mathematica, Maxima, возможен экспорт документов в формат *.tex. Интересный факт: для представления формул в Википедии также используется TeX-нотация.

Для того чтобы начать работать с LaTeX лучше всего поставить бесплатную программу TeXstudio (http://texstudio.org) доступную как для Linux, так и для Windows.

5.5.1.4. Математические графики в тексте

Хотя в текстовом редакторе и можно набирать математические формулы, он остаётся офисным продуктом и не является повседневной средой работы математиков. Поэтому для построения математических графиков лучше всего использовать сторонние программы, а в текстовом редакторе использовать функцию импорта изображения (см.п. 5.5.1.5). Также для построений можно использовать гистограммы построенные на основе таблиц (см. п. 5.5.1.6).



Примером наиболее простой сторонней программы для построения графиков может быть популярная программа gnuplot, доступная в большинстве ОС Linux.

Например, построение параболы с помощью gnuplot производится двумя командами, одна из которых, собственно, запуск программы в консоли:

```
$ gnuplot
gnuplot> plot [-30:30] [0:300] x**2+2*x+3
```

А эллиптический параболоид можно построить не менее просто (см. рис. 5.30):

```
$ gnuplot
gnuplot> set ztics (50,200,350)
gnuplot> p=2
gnuplot> q=4
gnuplot> splot x**2/(2*p)+y**2/(2*q)
```

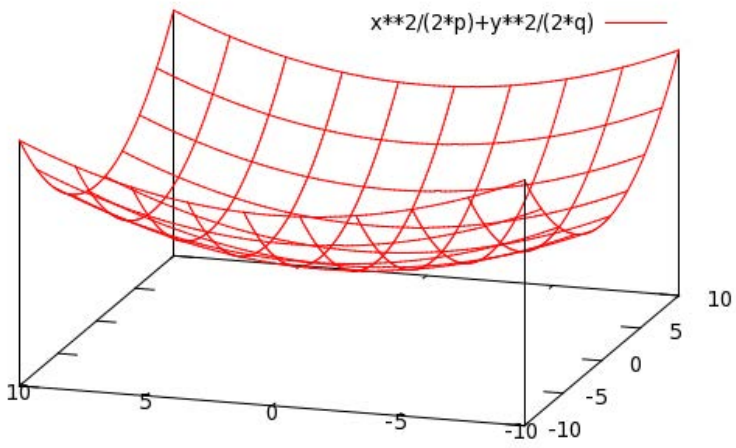


Рисунок 5.30. Эллиптический параболоид

Управляя стрелками на клавиатуре указанные объёмные графики можно «вращать» для того чтобы получить необходимый вид.

Примеры построения других (не менее красивых) графиков можно найти на странице http://www.cs.karelia.ru/studies/gnuplot/Examples.html.

5.5.1.5. Рисунки

Текстовый редактор LibreOffice Writer позволяет включать в документ различные графические объекты – рисунки. При этом простые графические объекты (стрелки, прямоугольники и прочее) можно нарисовать как в самом редакторе, так и используя LibreOffice Draw. Естественно, можно встраивать ранее подготовленные картинки и фотографии. Поддерживается множество растровых и векторных форматов.

Вставка готовых изображений делается либо из буфера (с помощью стандартных комбинаций клавиш Shift+Insert или Ctrl+V, либо правой кнопкой мыши через меню «Вставить»), либо из файла двумя шагами: через меню **Вставка ► Изображение...** (в версиях до LibreOffice 4.4.x требовался ещё и третий шаг: ► **Из файла...**).

Для работы с графическими объектами следует воспользоваться панелью инструментов **Рисование**. (Включается через меню **Вид ► Панели инструментов ► Рисование**.) На панели инструментов **Рисование** (см. рис. 5.31) выбираются объекты, которые необходимо нарисовать.

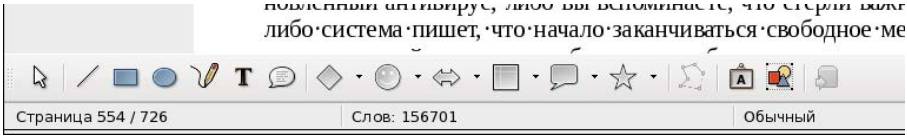


*Рисунок 5.31. Панель «**Рисование**»*

На панели **Свойства рисунка** находятся команды для редактирования графических объектов: изменения цвета контура и заливки, изменения толщины и типа линий, придания автофигуре объёма и т. д.

5.5.1.6. Таблицы в тексте

Таблицы в тексте могут занимать всю ширину страницы или часть её с обтеканием текстом или без обтекания. Ширина таблицы может быть задана в процентах от ширины страницы (без полей) или в сантиметрах (или других единицах, определённых в параметрах LibreOffice). Следует помнить, что в свойствах таблицы можно задать поля для её ячеек (верхнее, нижнее, левое и правое), выравнивание, эффекты шрифта и параметры абзацев в ячейках таблицы. Для многих операций при работе с таблицей удобно использовать контекстное меню, появляющееся снизу при наведении курсора на область таблицы, однако полный набор операций присутствует в пункте меню **Таблица**.

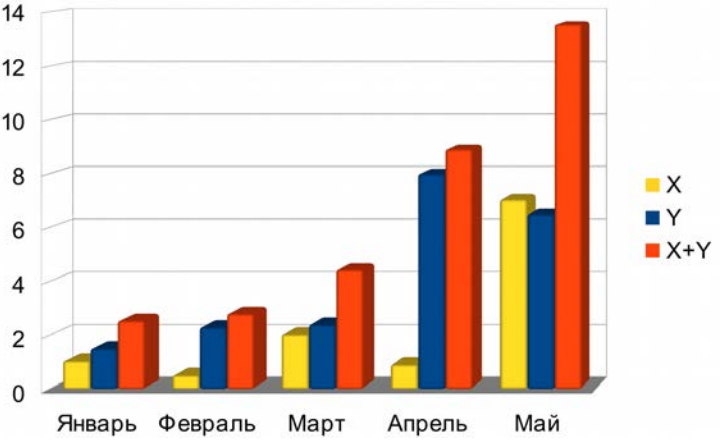
Таблица 5.16. Пример таблицы с формулами созданной в LibreOffice Writer

Месяц	X	Y	X+Y	
Январь	1,04	1,5	2,54	=sum<A2:B2>
Февраль	0,5	2,3	2,8	=sum<A3:B3>
Март	2,03	2,4	4,43	=sum<A4:B4>
Апрель	0,9	8	8,9	=sum<A5:B5>
Май	7,05	6,5	13,55	=sum<A6:B6>
	0,5	4,14	13,55	
	=min<B2:B6>	=mean <C2:C6>	=max<D2:D6>	

В ячейки таблицы можно вставлять поля с формулами для простейших математических вычислений по данным таблицы (для этого надо набрать первым знаком в ячейки символ «=», либо через меню выбрать **Вид ► Панели инструментов ► Формула**). Не путайте эти простейшие формулы со вставкой объекта «Формула...» (LibreOffice Math).

В табл. 5.16 показано использование функций вычисления минимального (min), среднего (mean), максимального (max) значений по столбцам и суммы по строкам (sum).

Если выделить необходимые ячейки числовых значений в таблице, то по ним можно построить диаграмму (график) через команды меню **Вставка ► Объект ► Диаграмма...**. После чего появляется окно мастера выбора диаграмм, где, выбрав соответствующие параметры, можно определить требуемый вид диаграммы (графика). В случае изменения данных в таблице 5.15, на основе которой построена диаграмма, последняя автоматически будет перерисовываться.



Для более сложных табличных зависимостей и диаграмм следует использовать электронные таблицы LibreOffice Calc.

5.5.1.7. Создание оглавления

В рассмотренной утрированной ситуации выше вы «подвигали» несколько глав по тексту и что же теперь: менять руками номера страниц в оглавлении? Конечно нет!

Если вы использовали стили для оформления заголовков в документе, как было описано выше, то для автоматического формирования оглавления сложного документа практически более ничего и не нужно.

Нажмите в меню **Вставка ► Оглавление и указатели ► Оглавление и указатели...**, выберите вид **Оглавление** и нажмите кнопку **Ok**. В месте, где находился курсор, будет автоматически сформировано оглавление из заголовков документа.

5.5.1.8. Проверка правописания

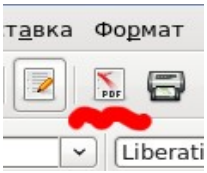
Если орфографическая проверка в LibreOffice встроена (предполагается, что вы установили русский helppack), то для проверки грамматики и стиля текстов, написанных на великом и могучем, рекомендуем установить свободное ПО LanguageTool, доступное для бесплатного скачивания по адресу: http://www.languagetool.org/ru/. По указанной ссылке имеется не только плагин (расширение) для пакета LibreOffice [306], но и

[306] Так как пакет проверки грамматики кросс-платформенный, то есть один и тот же для разных ОС, то для его работы требуется Java-виртуальная машина. Проверить, установлена ли у вас Java, а также скачать последнюю версию можно на сайте www.java.com.

аналогичное расширение для браузера Firefox и вообще независимая версия. Предлагаемая на момент публикации учебника версия 4.0 занимала 91 МБ и содержала в себе несколько сотен различных правил. В случае если программа считает, что вы забыли запятую или что-то не согласовали в тексте, сомнительный участок подчёркивается синим (см. рис. 5.32).

Маша ела кашу а саша суп.

Рисунок 5.32. Автоматическая проверка орфографии и грамматики (синим подчёркнуто из-за того, что в предложении забыта запятая после слова «каша». Красным подчёркнуто, потому как Саша – имя собственное и должно писаться с прописной буквы)



5.5.1.9. Экспорт в PDF

Имеется прямой экспорт созданных документов в формат PDF[307] нажатием одной кнопки.

5.5.1.10. Ленточный интерфейс в LibreOffice

Начиная с версии LibreOffice 5.3 существует возможность использования «ленточного» интерфейса. Для его включения необходимо в меню «Сервис» выбрать пункт «Параметры», в появившемся окне настроек (см. рис. 5.33) перейти в раздел «Расширенные возможности» и отметить флажком пункт «Включить экспериментальные возможности».

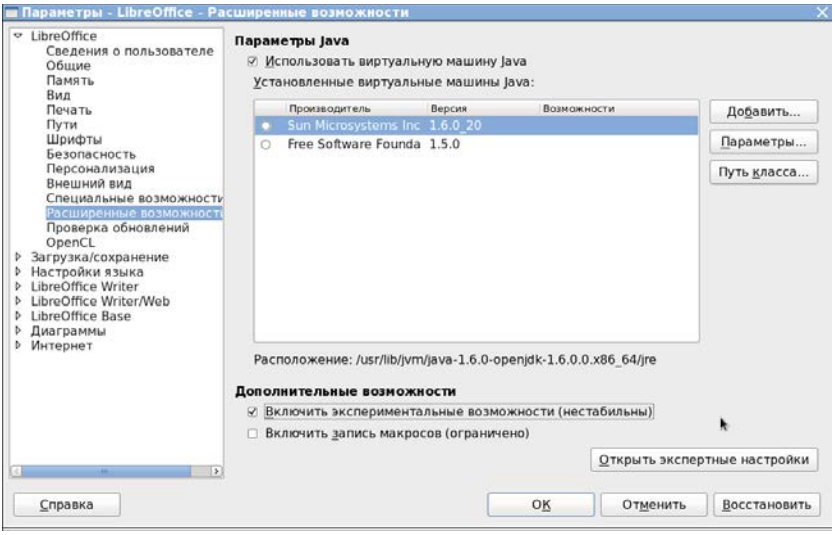


Рисунок 5.33. Включение экспериментальных возможностей

После этого в меню «Вид» в подменю «Разметка панели» станет возможным выбор пункта «Лента».

[307] Для других приложений рекомендуем читателям бесплатную утилиту doPDF для Windows (http://www.dopdf.com/), которая ставится в систему «как принтер».

5.5.1.11. Сохранение в облако

Также, начиная с версии LibreOffice 5.3 были выпущены исходники облачной версии – LibreOffice Online. По сути это позволяет компаниям создать на своём сервере собственную облачную структуру для работы с офисными документами. Это потребует соответствующей технической подготовки, зато обещает даже более широкие возможности, чем облачные решения от Microsoft или Google.

О том как настроить в ОС Windows прямой доступ к Google Drive из LibreOffice (начиная с версии 4.2) см. https://habrahabr.ru/sandbox/88871/. Поддержка SharePoint и OneDrive заявлена начиная с версии LibreOffice 4.4.

Работа с Яндекс Диска организована намного проще и поддерживается во всех ОС, поскольку используется протокол WebDAV.

5.5.2. Другие офисные программы, входящие в состав LibreOffice

Кроме текстового процессора, в LibreOffice входит ещё несколько полезных пакетов:

LibreOffice Calc – мощные электронные таблицы. Вероятнее всего, вы слышали про коммерческий продукт Excel. LibreOffice – это то же самое, но с лицензией свободного ПО и открытыми исходными кодами. Следует отметить, что Calc умеет делать всё то же самое и даже открывать файлы в форматах .xls и .xlsx. Многие коммерческие фирмы с целью экономии полностью перешли на свободное ПО LibreOffice Calc. Различные новые программы поддерживают экспорт в LibreOffice/OpenOffice Calc. А для относительно старых программ Санкт-Петербургская фирма «Этерсофт» разработала программу-переходник UniOffice [308]. Написав в техническую поддержку указанной компании, вы можете запросить для тестирования последнюю версию. Что же касается юридических лиц – стоимость указанного продукта смешна, так как сравнима со стоимостью расходных материалов – нескольких пачек бумаги А4.

LibreOffice Impress – средство создания презентаций (аналог коммерческого и закрытого продукта PowerPoint).

LibreOffice Draw – однозначного аналога нет – средство создания рисунков. Кстати, LibreOffice умеет открывать файлы пакета Visio.

LibreOffice Math – написание формул, вы уже с ним познакомились выше.

LibreOffice Base – средство для создания небольших баз данных, а также интерфейсов к уже существующим. В отличие от коммерческого продукта Access, LibreOffice Base может подключаться к различным многопользовательским базам, например к MySQL/MariaDB, и тем самым обеспечивать одновременную работу с одной базой нескольким пользователям, что в случае использования Access невозможно.

Так как всё вышеуказанное программное обеспечение является свободным, а информация по нему по большей части свободно распространяется в сети Интернет в виде различных справочников, документаций и публикаций, не видим смысла в её копировании. Приведём для читателей список полезных ссылок:

http://www.libreoffice.org/get-help/documentation/
http://wiki.documentfoundation.org/images/1/13/GS40-GettingStartedLO.pdf
http://wiki.documentfoundation.org/images/a/af/GS40-GettingStartedLO.odt

[308] http://wiki.etersoft.ru/UniOffice – UniOffice@Etersoft позволяет использовать OpenOffice.org вместо MS Office, выполняя трансляцию COM-запросов к MS Office в вызовы OpenOffice.org.

http://www.ict.edu.ru/ft/005682/Impress.pdf
http://linux.armd.ru/common/img/uploaded/files/Writer.pdf
http://kargasok.tomsk.ru/wp-content/uploads/2011/03/Calc.pdf

5.5.3. Альтернативные офисные пакеты

Пакет LibreOffice – не единственный офисный пакет для создания текстовых документов, кто-то может с иронией заметить, что и не лучший, поэтому хотелось бы представить на суд читателей несколько альтернатив.

Среди коммерческих продуктов чаще всего используется Microsoft Office различных версий, отчасти потому, что не все, но многие используют его пиратскую версию. А также на многих ноутбуках установлены пробные trial-версии, которые примерно через месяц требуют покупки полной версии, но до этого успешно работают.

Если же поставить целью не нарушать авторское право, то обратите внимание на достойный пакет AbiWord [309] – доступны версии для ОС Windows, ОС Linux и исходные коды. Проекту более 11 лет (с апреля 2002 года).

Пакет Apache OpenOffice практически является предком пакета LibreOffice. Бесплатно можно скачать[310] исходные коды, набор для разработчиков (SDK), а также готовые и протестированные версии для ОС Linux, Windows и MacOS.

Для ОС Windows можно скачать бесплатный китайский текстовый редактор Kingsoft Writer Free (доступны 32- и 64-битные версии [311]). Визуально редактор, впрочем, как и LibreOffice, сильно напоминает интерфейс Word 2003. За дополнительную плату доступны расширенные версии Standard и Professional. Все версии полностью совместимы с Microsoft Word (97/2000/2003/2007/2010). То есть можно успешно работать с .doc- и .docx-файлами, а также осуществлять экспорт в .pdf.

Аналогично редактору Kingsoft Writer существуют Corel Office [312] и Ashampoo Office [313], доступные также в пробной и коммерческой версиях.

Фирма IBM выпускает бесплатный набор (freeware) офисных приложений для создания, редактирования и коллективного использования текстов, электронных таблиц, презентаций и других документов с закрытым кодом – IBM Lotus Symphony [314]. Пакет использует технологии OpenOffice.org и как следствие доступен для ОС Linux, Windows и MacOS, поддерживается стандарт ODF 1.2. В январе 2012 года представители компании IBM заявили, что версия 3.0.1, скорее всего, будет последней, компания полностью переключилась на развитие OpenOffice и отказалась от развития собственного офисного пакета.

Для владельцев MacOC, помимо упомянутых выше бесплатных офисных пакетов, доступен iWork. iWork дешевле коммерческого пакета Microsoft Office for Mac, но не содержит эквивалентов ряда программ, входящих в некоторые версии Microsoft Office. Пробная 30-дневная версия iWork предлагается с каждым новым Mac, а также входит в состав приобретаемого пакета программ iLife и доступна для скачивания на официальном сайте [315].

[309] Свободный текстовый редактор AbiWord http://abiword.org/download/.
[310] http://www.openoffice.org/download/.
[311] http://www.kingsoftstore.co.uk/writer-standard.html.
[312] http://www.corel.com/corel/product/index.jsp?pid=prod3430104.
[313] http://www.ashampoo.com/ru/usd/pin/0538/Office_Software/Ashampoo-Office-2012.
[314] http://symphony.lotus.com/.
[315] http://www.apple.com/ru/iwork/download-trial/.

Если вы не хотите устанавливать ни один из офисных пакетов на ваш компьютер, то у вас есть два выхода: первый – использовать так называемые «portable»-версии, когда любая программа (например, текстовый редактор в нашем случае) запускается непосредственно с USB-флэш накопителя, не требует установки и ничего (кроме временных файлов) не записывает в вашу систему. Portable (от англ., означает переносимые) версии можно либо найти на сайте разработчиков, если они их изготовили, либо изготовить самостоятельно с помощью сторонних коммерческих приложений, например ThinApp [316] от VmWare.

Второй выход, также набирающий популярность в последнее время, – использовать возможности различных интернет-сервисов через браузер, ничего при этом не устанавливая на компьютер. Современные веб-технологии, в частности AJAX, позволяют реализовать в окне браузера функционал, сравнимый с классическими локальными версиями. Обработка и хранение документов при этом происходят на стороне сервера, расположенного «где-то там, в интернете». Наверняка вы слышали краем уха про «Google Documents» и аналогичные сервисы. Недостатков у данной технологии два: необходимо постоянно иметь «быстрое» и надёжное соединение с интернетом, а также следует понимать, что все создаваемые и хранимые данные будут доступны неопределённому кругу техники и людей, занятых в обслуживании ваших запросов. Это не только конечный центр обработки данных, которому вы можете доверять, но и все промежуточные интернет-узлы, работающие с вашим трафиком. Для больших фирм, обеспокоенных указанными выше вопросами, существуют различные intranet-аналоги, например Zimbra [317].

Для владельцев телефонов и планшетов с ОС Android также существует большое число офисных пакетов. Подробный обзор часто используемых представлен в [27].

5.6. Сервисные программы

Сервисные программы – для обычного пользователя это часто те программы, которые ему на первый взгляд кажутся ненужными и, возможно, о существовании которых он и не знает. Ситуация в чём-то аналогична с автомобилем: сначала вы его покупаете, а лишь потом вы узнаёте о различных дополнительных сервисах, ближайших автомойках, пунктах шиномонтажа, заправках, магазинах запчастей и прочем. Так и с компьютером – он включается, операционная система загружается, можно просматривать веб-страницы в интернете, получать почту, создавать офисные документы. Казалось бы, что ещё нужно для счастья пользователю?

Практика показывает, что «безоблачная жизнь» – скорее редкость, и через некоторое время использования компьютера обязательно случается что-то непредвиденное: или заканчивается лицензия на предустановленный антивирус, или вы вспоминаете, что стёрли важный файл, либо система пишет, что начало заканчиваться свободное место, либо компьютер поймал вирус, либо сам по себе стал медленнее работать. Вот тогда у большинства пользователей и случается то самое первое знакомство с различным сервисным ПО.

[316] http://www.vmware.com/products/thinapp/overview.html.
[317] http://www.zimbra.com/.

Немного усилий по самостоятельному изучению и решению возникших проблем – и вас уже сложно назвать новичками или неопытными пользователями.

Рассмотрим подробнее, какие типы сервисного ПО могут вам встретиться и пригодиться. По большей части будет рассмотрено сервисное ПО для ОС Windows, потому как пользователи Linux, подключенные к сети интернет, обычно таких проблем не имеют и могут бесплатно устанавливать и использовать все программы из репозиториев (например, на ftp.yandex.ru лежат для свободного скачивания, по протоколам FTP и HTTP, репозитарии большинства современных Linux-дистрибутивов), а это обычно более полусотни гигабайтов софта на всевозможные случаи жизни под каждую версию.

Назначение сервисных программ – обеспечение надёжной работы компьютеров и их сетей: тестирование и оптимизация, обеспечение резервного копирования и восстановления, защита от вредоносных, шпионских и мошеннических вторжений.

В последнее время наиболее крупные фирмы – поставщики подобного программного обеспечения (**Symantec, Vcom, Acronis, Aladdin Systems, PGSoft, PowerQuest** и прочие), выпускают комплексы сервисных пакетов, предназначенных для домашних компьютеров, малого и среднего бизнеса, крупных предприятий.

В первую очередь востребованными оказываются программы антивирусы.

5.6.1. Защита от вирусов

Компьютерные вирусы

В статье № 273 УК РФ (см. в главе 3) говорится про *создание, распространение или использование компьютерных программ либо иной компьютерной информации, заведомо предназначенных для несанкционированного уничтожения, блокирования, модификации, копирования компьютерной информации или нейтрализации средств защиты компьютерной информации*, но ничего не говорится про компьютерные вирусы.

То есть, с юридической точки зрения, понятие компьютерного вируса не определено. Оно и понятно, законы призваны защищать права граждан, и если созданный вами файл с информацией был уничтожен, то, наверное уже всё равно, сделал это «вирус» или программа, названная другим словом. А тем более был ли это «вирус-троян» или «вирус-червь». Классификация вирусов появилась позднее, с целью упрощения описания свойств тех самых вредоносных программ в быту, и в какой-то мере условна.

Официально считается, что термин «компьютерный вирус» впервые употребил сотрудник Лехайского университета (США) Ф. Коэн в 1984 г. на 7-й конференции по безопасности информации, проходившей в США.

Так что же подразумевает этот термин?

Компьютерный вирус – вредоносная программа[318], способная к саморазмножению[319] и распространению внутри одного компьютера или при передаче информации с одного компьютера на другой.

[318] Даже если вирус вам кажется безобидным, потому как не выполняет явно деструктивных действий по отношению к вашим файлами и системе, то он явно занимает какое-то место на жёстком диске, в памяти и расходует вычислительные ресурсы процессора, которые могли бы быть использованы вами или операционной системой в других целях.

[319] В случае передаче ей управления.

Алгоритм работы классического вируса

Классическим вирусом принято считать программу, «прикрепляющуюся» к любому исполняемому файлу так, чтобы это не нарушало его работы. То, что прикрепляется, называют телом вируса. Файл с прикреплённым телом вируса принято считать заражённым или инфицированным. При запуске файла (передаче ему управления операционной системой) до выполнения основного кода, после или в середине управление передаётся вирусу, который, в свою очередь, осуществляет поиск других исполняемых файлов и встраивает в них свой код. Данное действие называется размножением вируса. Попутно с размножением вирус может выполнять те или иные побочные действия – от безобидного «помигивания» индикаторами клавиатуры до более деструктивных действий.

Замечание. Количество вирусов, написанных для UNIX-систем (в том числе и под Linux), заведомо меньше и по той причине, что из-за существования изначально разграничения доступа в данных ОС с использованием дискреционной политики безопасности, когда пользователи работают от своих учётных записей, вирус, даже в случае получения управления, не может расширить возможности учётной записи, от которой он выполняется. Фактически это выражается в том, что он не может заразить существующие в системе, но недоступные на запись ему файлы. Операционная система DOS была однопользовательской и не имела разграничения доступа, что и способствовало бурному размножению вирусов.

Классификация вирусов

Первые вирусы существовали исключительно в рамках одного компьютера и распространялись вместе с копированием заражённых исполняемых файлов через дискеты.

Позднее программисты-вирусописатели освоили ставшие доступными повсеместно сетевые технологии. Таким образом, в настоящее время, кроме классических компьютерных вирусов, существуют также сетевые вирусы (черви и троянские программы), а также прочие вредоносные программы, которые сложно однозначно классифицировать по отношению к тому или иному виду.

Условно вирусы можно разделить на следующие типы:

- файловые;
- загрузочные;
- макровирусы;
- скриптовые;
- сетевые;
- майнеры.

Файловые вирусы различными способами внедряются в исполняемые файлы либо создают файлы-двойники. Классический вирус является файловым.

Пример, показывающий, насколько сложными являются подобные программы, – вирус WinNT.RemEx (Remote Explorer) под ОС Windows. При запуске файла вируса его код копируется в системный каталог System32\Drivers под именем IE403R.SYS и запускается как системный сервис, который один раз в 10 минут запускает одну из

своих подпрограмм – либо заражения, либо «заметания следов». Процедура заражения сканирует случайно выбранные локальные и доступные сетевые папки, ищет EXE-файлы и заражает их. При заражении вирус компрессирует файл-жертву, записывает в него свой код и добавляет компрессированный первоначальный код файла в его конец. При запуске заражённого EXE-файла вирус копирует его во временный файл, распаковывает и выполняет исходный файл программы. Вирус компрессирует файлы методом «deflate» и использует для этого встроенную в свой код библиотеку GZIP. В зависимости от случайного счётчика вирус также портит случайно выбранные файлы: компрессирует их тем же методом и затем зашифровывает криптоалгоритмом средней сложности. Процедура «заметания следов» запускается сразу за заражением и уничтожает следы жизнедеятельности вируса. Вирус закрывает сообщения об ошибках, произошедших при заражении файлов, и выполняет другие действия (по данным Лаборатории Касперского *http://www.kav.ru*).

Загрузочные вирусы записывают себя в загрузочный сектор диска (boot-сектор), либо в сектор, содержащий системный загрузчик винчестера (Master Boot Record), либо меняют указатель на активный boot-сектор. Данный тип вирусов был достаточно распространён в 1990-х годах, но практически исчез с переходом на 32-разрядные операционные системы и отказом от использования дискет как основного способа обмена информацией. Теоретически возможно появление загрузочных вирусов, заражающих оптические диски и USB-flash-накопители, то есть всё то, с чего можно произвести загрузку компьютера (операционной системы), также это могут быть различные USB-устройства и даже USB-удлинители и USB-хабы сомнительного производства.

Макровирусы заражают файлы данных различных популярных программ. Многие текстовые процессоры [320], табличные и графические редакторы, системы проектирования, имеют свои языки программирования для работы с различными объектами, присутствующими в файлах данных этих систем (например, Visual Basic for Application (VBA) в системе Microsoft Office). Текст программ на языке VBA сохраняется в модулях процедур (макросах), хранящихся в файлах документов. Макровирусы являются программами на языках, встроенных в такие системы обработки данных. Например, при открытии документа Word проверяет его на наличие процедуры с именем AutoOpen, если она присутствует, то Word выполняет её. Для своего размножения вирусы этого класса используют возможности языков этих систем и при их помощи переносят себя из одного заражённого файла (документа, таблицы или другого файла данных) в другие. По данным Лаборатории Касперского, количество макровирусов – несколько сотен.

Пример опасного макровируса: Macro.Word.KillDll. Содержит единственный макрос: AutoOpen. При открытии заражённого файла вирус заражает область глобальных макросов системы Word. После этого при открытии в системе Word незаражённого файла заражает его и уничтожает все файлы по маске *.D?? в папке C:\WINDOWS \SYSTEM\ (то есть файлы динамических библиотек и драйверов Windows).

[320]Текстовый файл заразить нельзя по определению, так как он содержит лишь читаемые символы (что исключает попадание в него нечитаемых машинных кодов, для ASCII это символы с кодами 0–31), а также ему никогда не передаётся управление. Путаница возникает из-за того, что более сложные по своей структуре файлы текстовых процессоров часто в быту (возможно, по ошибке или от незнания) называются текстами.

Скриптовые вирусы написаны на различных скрипт-языках (VBS – Visual Basic Script, JS – Java Script, .bat-файлы, bash-файлы, PHP и прочие), которые широко используются в ОС и веб-программировании. Они либо заражают другие скрипт-программы, присутствующие в виде программных файлов или фрагментов текста html-страниц, либо являются частями многокомпонентных вирусов.

Пример: Virus.VBS.Redlof написан на языке VBS и зашифрован. При первом запуске создаёт файл со своим исполняемым кодом в системном каталоге Windows с именем Kernel.dll. Кроме того, копирует себя во все каталоги на других дисках заражённого компьютера в виде файла настройки отображения файлов и папок – folder.htt. Заражённый файл folder.htt получает управление и копируется вирусом во все папки при их открытии при помощи Проводника. Вирус дописывает себя во все HTM-файлы [321], находящиеся в каталоге Windows\web, и таким образом он также получает управление при открытии данных файлов.

Сетевые вирусы используют для своего распространения протоколы и команды компьютерных сетей и электронную почту.

Вирусы-майнеры используют вычислительные ресурсы компьютера (процессора, видеокарты) пользователя для майнинга криптовалют (цифровых денег). Получили широкое распространение начиная с 2017 года. Заразиться таким вирусом можно даже через браузер. Существуют версии вирусов без размножения (формально это уже не вирус), когда использование ресурсов компьютера происходит во время просмотра web-страниц за счёт выполнения javascript-программы в браузере. Например это могут быть незаметные для пользователя прозрачный фрейм или отдельное окно.

Помимо вышеописанных типов можно выделить:

Троянская программа – любая программа, которая втайне от пользователя выполняет на компьютере какие-либо нежелательные для него действия. К троянским закладкам можно отнести и недокументированные возможности многих известных продуктов, часто именуемых «пасхальными яйцами». Например, внешне безобидные «пасхальные яйца», но никак не относящиеся к основному функционалу программы, были найдены во многих версиях электронных таблиц Excel. Воспользуйтесь поиском в интернете, и вы без труда найдёте список подобных закладок с описаниями, как их активировать.

По своим функциям троянские программы подразделяются на:

- бэкдор (Backdoor) – предоставляет несанкционированный доступ к удалённому администрированию компьютера;
- PSW-трояны – похищает всевозможные пароли;
- Trojan Clicker – открывает на компьютере пользователя различные веб-ссылки (URL) без ведома пользователя;
- Trojan Downloader – загружает из интернета какие-либо файлы без ведома пользователя;
- Trojan-Dropper – инсталлирует без ведома пользователя другие вредоносные программы на заражённый компьютер и запускает их;
- Trojan Proxy – запускает без ведома пользователя прокси-сервер на заражённом компьютере;
- прочие троянцы.

[321] Написание верное – прим. авт.

Сетевые черви подразделяются на почтовые (Email-Worm), использующие интернет-пейджеры типа ICQ (IM-Worm), использующие IRC-средства интернет-общения (IRC-Worm), заражающие интернет-серверы и прочие. Почтовые черви после инсталляции в системе могут рассылать себя по всем адресам, обнаруженным в адресной книге Windows и в файлах электронной почты. Другие виды сетевых червей могут рассылать URL-ссылку на себя или иные сайты по адресам, обнаруженным в базах интернет-пейджеров или IRC-клиентов, открывать полный сетевой доступ на диски компьютера и прочее.

К вредоносным программам относятся также **руткиты** (от англ. root kit, дословно «набор для получения прав суперпользователя в системе») – утилиты, используемые компьютерными преступниками для сокрытия своей вредоносной деятельности на заражённом компьютере. Для этого они модифицируют один или несколько файлов в ОС компьютера. Многие **руткиты** маскируют своё собственное присутствие в системе (то есть их бывает нельзя увидеть в списке процессов или как файл на жёстком диске из-за подмены выдаваемых пользователю или администратору данных на лету), как и действия, выполняемые с помощью них удалённо. **Руткит** может скрывать присутствие других вредоносных программ на компьютере, изменяя файловые данные, ключи реестра или активные процессы.

Каждый файловый или сетевой вирус существует в какой-либо одной или нескольких ОС – **DOS**, **Windows 9x**, **Windows NT/200x/XP/Vista/7/8**, **OS/2**, **Linux**, **MacOS**, **Android**, **Unix** и т. д.

Многие опасные вирусы, написанные для одной ОС, не представляют никакой опасности для другой. Причём это не обязательно из-за разных форматов исполняемых файлов. Даже если опасный вирус, изменяющий важные ключи реестра или стирающий файлы в папке «Windows», скомпилировать в формате ELF для ОС Linux и запустить, его работа будет бессмысленной из-за отсутствия указанных объектов.

APT-атаки[322]

APT (от англ. advanced persistent threat, дословно «развитая устойчивая угроза»). Своеобразный вид атак на компьютерные и технические системы с трудом относимый к традиционной вирусной активности. С другой стороны, для достижения поставленных целей, осуществляющие APT-атаки могут маскировать свои действия под обычные вирусы, либо использовать другие вирусы как части более сложных схем. Обычно APT добивается своих целей неоднократно в течение длительного времени, адаптируется (в том числе и людьми) к усилиям защищающихся оказать угрозе сопротивление.

Термин APT изначально использовался для описания кибернападений на военные организации, но более не ограничен военной сферой. Атака APT превосходит обычные киберугрозы, так как ориентируется на взлом конкретной цели и готовится на основании информации о ней, собираемой в течение длительного времени. Довольно часто APT осуществляет взлом целевой инфраструктуры посредством эксплуатации программных и аппаратных уязвимостей вместе с методами «социальной инженерии».

[322] См. *Яремучук С.* APT: реальность или паранойя? // Системный администратор № 7-8 (116-117), 2012, с. 52-56.

Особенности алгоритмов работы вирусов:

- резидентность;
- использование стелс-алгоритмов;
- самошифрование и полиморфичность;
- использование нестандартных приёмов.

Резидентный вирус при инфицировании компьютера оставляет в оперативной памяти свою резидентную часть, которая затем перехватывает обращения операционной системы к объектам заражения и внедряется в них. Резидентные вирусы находятся в памяти и являются активными вплоть до выключения компьютера или перезагрузки операционной системы.

Резидентными можно считать макровирусы, поскольку они постоянно присутствуют в памяти компьютера на всё время работы заражённого редактора.

Нерезидентные вирусы не заражают память компьютера и сохраняют активность ограниченное время. Некоторые вирусы оставляют в оперативной памяти небольшие резидентные программы, которые не распространяют вирус. Такие вирусы считаются нерезидентными.

Использование стелс-алгоритмов позволяет вирусам полностью или частично скрыть себя в системе. Наиболее распространённым стелс-алгоритмом является перехват запросов ОС на чтение/запись заражённых объектов. Стелс-вирусы при этом либо временно лечат их, либо «подставляют» вместо себя незараженные участки информации. В случае макровирусов наиболее популярный способ – запрет вызовов меню просмотра макросов.

Самошифрование и полиморфичность используются практически всеми типами вирусов, для того чтобы максимально усложнить процедуру детектирования вируса. Полиморфные вирусы (polymorphic) – это достаточно трудно обнаруживаемые вирусы, не имеющие сигнатур [323], то есть не содержащие ни одного постоянного участка кода. В большинстве случаев два образца одного и того же полиморфного вируса не будут иметь ни одного совпадения. Это достигается шифрованием основного тела вируса и модификациями программы-расшифровщика.

По деструктивным возможностям вирусы подразделяются на:

- безвредные, то есть никак не влияющие на работу компьютера (кроме уменьшения свободного места на диске в результате своего размножения);
- неопасные, влияние которых ограничивается уменьшением свободного места на диске и графическими, звуковыми и прочими эффектами;
- опасные вирусы, которые могут привести к серьезным сбоям в работе компьютера;
- очень опасные, в алгоритм работы которых заведомо заложены процедуры, которые приводят к порче программ, уничтожению файлов.

Полную информацию о многочисленных известных вирусах можно найти на сайтах фирм-производителей антивирусных программ, например на сайте Лаборатории Касперского *http://www.kav.ru,* и *http://www.securelist.com/ru/.*

[323] Сигнатура вируса – это описание характерной для данного вируса последовательности байтов (в какой-то мере являющихся его программой), используемое антивирусными программами для поиска вирусов. Сигнатурный метод поиска наиболее простой и быстрый. Это как если груши по форме отличать от яблок, но стоит сделать яблочное или грушевое пюре или сок, и данный сигнатурный метод «по форме плода» уже не сработает.

Антивирусные программы

Для борьбы с компьютерными вирусами используются антивирусные программы. Так как количество компьютеров простых пользователей, не подключенных к интернету, стремится к нулю, первой программой, устанавливаемой на ПК после установки операционной системы, должна быть антивирусная система. Интернет в настоящее время – источник повышенной вирусной опасности, особенно при посещении сайтов со «взломанными» программами и полезными утилитами, развлекательных, в том числе содержащих порнографические материалы!

В состав современных антивирусных систем входит, как правило, несколько программных компонентов:

- **сканер** – программа проверки заданных дисков, папок или файлов по указанию пользователя, по расписанию или по команде монитора, программы проверки электронной почты или скриптов;
- **монитор**, запускающий программу-сканер для проверки всех запускаемых на ПК программ и открываемых файлов данных, начиная с момента запуска ОС;
- программы проверки файлов электронной почты и скриптов в файлах интернета, поступающих на компьютер (**интернет модуль**);
- **программа обновления антивирусных баз**;
- **центр управления**, позволяющий задать расписание сканирования папок, обновления баз и прочие параметры работы;
- отдельно или совместно поставляемые загрузочные диски или флэшки (точнее, их программные образы) для проверки компьютера.

В антивирусных сканерах для поиска известных вирусов используются два основных метода: ***сигнатурный поиск*** и ***эвристический анализ***.

Сигнатурой, или маской вируса, является некоторая постоянная последовательность кода, специфичная для этого конкретного вируса. Все сигнатуры размещены в антивирусной базе – специальном хранилище, в котором программа-антивирус хранит характерные коды вредоносных программ.

Метод эвристического анализа проверяет не код подозрительного файла, а его действия. Для того чтобы размножаться, вирус должен копировать своё тело в память, открывать другие исполняемые файлы и записывать туда своё тело, записывать данные в секторы жёсткого диска и т. д. Есть характерные действия и у сетевого червя – доступ к адресной книге и сканирование жёсткого диска на предмет обнаружения адресов электронной почты. Благодаря этому эвристический анализатор способен обнаружить даже те вредоносные коды, сигнатуры которых ещё неизвестны.

Антивирусные сканеры можно разделить на две категории – универсальные и специализированные. Универсальные сканеры рассчитаны на поиск и обезвреживание всех типов вирусов вне зависимости от операционной системы, на работу в которой рассчитан сканер. Специализированные сканеры предназначены для обезвреживания ограниченного числа вирусов или только одного их класса, например макровирусов. Специализированные сканеры, рассчитанные только на макровирусы, часто оказываются наиболее удобным и надёжным решением для защиты систем документооборота в средах Microsoft Word и Excel.

К достоинствам универсальных сканеров относится большое количество вирусов, которые они знают и могут обнаруживать и обезвреживать. Недостатки – относительно невысокая скорость работы и большой размер антивирусной базы, которую необходимо систематически обновлять (желательно каждый день или чаще).

Системы контроля целостности

Не все вирусы можно обнаружить, зато можно отслеживать ситуацию, что какой-то системный файл изменился. Обычно исполняемые файлы, как компоненты ОС, так и вспомогательные программы, имеют фиксированный размер и фиксированное содержание в течение всего периода своей «жизни», за исключением обновлений и очень редких случаев хранения каких-то данных внутри исполняемого файла.

Если размер исполняемого файла, дата, как и его содержимое, поменялись – повод разобраться, что произошло. Для этого как раз и существуют программы ***CRC-сканеры*** (или в общем случае ***программы контроля целостности файлов***), принцип работы которых основан на подсчёте контрольных сумм файлов и системных секторов. Подобные CRC-суммы затем сохраняются в базе данных этой программы, как, впрочем, и некоторая другая информация: длины файлов, даты их последней модификации и т. д. При последующем запуске CRC-сканеры сверяют данные, содержащиеся в базе данных, с реально подсчитанными значениями. Если информация о файле, записанная в базе данных, не совпадает с реальными значениями, то CRC-сканеры сигнализируют о том, что файл был изменён или заражён вирусом.

Примером такой программы для ОС Linux можно назвать программу tripwire.

В 90-е годы прошлого века существовали программный комплекс Adinf и аппаратный Sheriff, подключаемый к шине ISA для обеспечения контроля целостности файлов. Сегодня подобные решения также существуют в виде небольших PCI-плат вставляемых в слоты расширения компьютера, и называются они АПМДЗ – аппаратно-программные модули доверенной загрузки. В пользовательских компьютерах данные решения не практикуются по той причине, что они не обеспечивают лечения заражённых файлов и неудобны в работе. Сами подумайте, какой нормальный пользователь будет ждать больше получаса каждый раз при загрузке, чтобы тщательно проверились все его системные файлы?

Найден вирус, что делать?

Если вы в офисе, то, скорее всего, появившийся вирус – это проблема системного администратора... а вот что делать, если на вашем домашнем компьютере найден вирус, где администратором и пользователем являетесь вы в одном лице? Главное – не паниковать, попробуйте сделать следующее:

1. **Отключиться от сети.** Если компьютер подключен к сети (локальной или интернет), необходимо отключить его от сети и проинформировать других участников (если такие в сети имеются). Отключение вашего и их компьютеров может снизить масштабы бедствия, хотя может и ни на что не повлиять. Обратное подключение рекомендуется выполнять после того, как проблема с вирусом будет устранена. Это означает, что, во-первых, будет вылечен и защищён ваш компьютер и также будут вылечены все соседние компьютеры (серверы и рабочие станции). Во-вторых, будет установлена и убрана причина проникновения вируса. (Например, будут установлены соответствующие обновления для ОС.) Конечно, всё написанное выше – это в идеале, бывают и отклонения от этих правил.

2. **Сделать копию данных из другой ОС.** Загружаемся с CD и снимаем копию жёсткого диска. Это на случай, если вы сделаете что-то хуже, чем было, тогда можно

будет «вернуться» и обратиться к профессионалам (чаще всего этот пункт в домашних условиях пропускают, так как ценных данных обычно нет, а переставить систему быстрее и проще).

3. **Проверить и вылечить заражённые файлы.** При помощи антивирусной программы нужно вылечить заражённые файлы и затем проверить их работоспособность. Если возможно, следует восстановить заражённые файлы из архивной копии, предварительно проверив архив на отсутствие вирусов. После лечения от вируса желательно выполнить сканирование всех дисков компьютера. Лучше всего это сделать, загрузившись с антивирусной флэшки или антивирусного CD/DVD.

Антивирусный диск или флэшку лучше подготовить заранее [324], но если их нет, то – выключаете компьютер и идёте к соседям, друзьям, в интернет-клуб и создаёте инструмент для лечения. Для этого вам надо зайти на сайт http://www.kaspersky.ru/virusscanner и скачать образ Kaspersky Rescue Disk, после чего записать его на CD-болванку. Аналогично идём на http://www.freedrweb.com/livecd/, скачиваем Dr.Web® LiveCD и записываем на диск (или Dr.Web LiveUSB на флэшку).

Один антивирус хорошо, а два последовательно лучше!

4. **Изучить особенности найденных вирусов.** В процессе «лечения» бывает так, что используется установленный антивирус вместо загрузки с CD/DVD/флэшки. (То есть и пункту 3 в домашних условиях никто не следует.) В этом случае важно понять, что вирус не блокирует работу антивируса. Если обнаружен резидентный файловый вирус, следует убедиться в том, что программе-сканеру данный вирус известен и она обезвреживает его резидентную часть. Полезно ознакомиться и с другими характеристиками вируса: типами заражаемых вирусом файлов, действиями вируса и прочим. Источником подробной информации может служить «Вирусная энциклопедия» Лаборатории Касперского (*www.securelist.com/ru/viruses/encyclopedia*).

После запуска одного антивируса попробуйте запустить другой. Записывайте на бумагу имена заражённых файлов и названия вирусов – это поможет вам в том случае, если в процессе проверки компьютер случайно перезагрузится. Также помните, при обращении к специалисту важно сообщить все детали, а тут «*тупой карандаш лучше вашей острой памяти*».

5. **Принять профилактические меры.** По завершении лечения важно выполнить профилактические меры, а именно, изучив особенности вируса, предупредить повторное заражение им вашего компьютера. Выполните меры, рекомендуемые вирусной энциклопедией. Обычно это установка какого-нибудь обновления или патча. Конечно, если вы сами щёлкнули на какой-то .exe-файл по глупости – ничего не поделаешь, но и в этом случае можно обновить базу антивируса, и, возможно, в будущем он вас спасёт. Может, стоит и установить антивирус, если он у вас не был установлен. Если вы разорились и купили операционную систему, то, постоянно используя интернет, грех не потратиться и на антивирус. Обычно антивирус скачивается бесплатно, а покупается к нему лицензия (ключ активации) на год, два или больше.

Если вирус не найден, а компьютер продолжает виснуть или перезагружаться – не паникуйте, вполне возможно, что ваш компьютер здоров программно, но сломался аппаратно, например перегрев летом и прочее. В этом случае вместо антивирусной

[324] Не забывайте обновлять подготовленные заранее диски и флэшки. Прошлогодний загрузочный, скорее всего, будет иметь меньше эффективности в поиске вирусов, чем недавно созданный.

программы подойдут различные тесты: памяти. процессора, материнской платы. Загрузка с чистой ОС обычно однозначно даёт понять, в чём проблема – в железе или программах.

В локальных сетях предприятий, школах и вузах могут использоваться специальные меры борьбы с проникновением вирусов: запрет пользователям, не обладающим административными правами, использовать устройства для работы с оптическими дисками, флэш-картами, USB-флэш-памятью, дискетами; защита компьютеров межсетевым экраном (брандмауэром); настройка прокси-сервера с доступом пользователей только к разрешённым ресурсам локальной сети и интернета.

Стратегическим компонентом защиты данных от вирусов является **систематическое резервное копирование информации**.

Сертификацией антивирусных программ занимается **ICSA** – International Computer Security Association[325] – Международная ассоциация по компьютерной безопасности (основана в 1992 году). В тестированиях, проводимых **ICSA Labs**, используется вредоносный код как из собственной коллекции, так и из списка **WildList** (список вирусов неформальной международной организации *WildList Organization International*). По результатам исследований продуктам выдаётся сертификат **ICSA** – его удостаиваются те антивирусы, которые способны обнаружить 100% вирусов из списка **WildList**, выпущенного за месяц до испытаний, и не менее 90% вирусов из собственной коллекции **ICSA**.

Сранение антивирусных программ можно найти на сайте https://www.av-comparatives.org/

Список сертифицированных **ICSA** антивирусов для разных ОС, в том числе и для Mac OS и Linux, можно найти по адресу https://www.icsalabs.com/products. Большинство пользователей верят не сертификатам, а друзьям и рекламе, поэтому чаще всего можно встретить десяток программ, характерных для России: Антивирус Касперского. Dr.Web, ESET NOD32, Avast antivirus, Panda Antivirus Pro, McAfee Antivirus Plus, Microsoft Security Essentials и Symantec Antivirus.

Сравнение различных антивирусных программ можно найти, например, на сайте *http://www.av-comparatives.org*. Конечно, может оказаться что один антивирус будет лучше другого и работает быстрее и база обнаруживаемых им вирусов больше чем у других, но, если он пропустит один новый вирус, который уничтожит все ваши данные, а другой антивирус, попроще, поймает такой вирус, какой из двух окажется лучше?

В настоящее время наиболее крупные фирмы, производители антивирусных программ, выпускают их в составе комплексных систем обеспечения компьютерной безопасности.

Двумя лидирующими компаниями в этой области в нашей стране являются Лаборатория Касперского (*http://www.kaspersky.ru*) и «Доктор Веб» (*http://www.drweb.com*). Поскольку продукты компании Евгения Касперского имеют больше присутствия за рубежом, нежели «Доктор Веб» Игоря Данилова, рассмотрим их подробнее.

Некоторые фирмы выпускают бесплатные антивирусные программы для домашнего использования или пробные версии с ограниченной функциональностью. Например, немецкая фирма AVIRA предлагает скачать бесплатную версию – программу

[325] https://www.icsalabs.com/.

AntiVir Personal Edition Classic (*http://www.free-av.com*), эта программа защищает пользователя не только от вирусов, червей и троянов, но также от программ-шуток, руткитов и фишинга. Шпионские и рекламные модули эта версия обнаруживать не умеет, подобная защита имеется только в платной версии AntiVir Personal Edition Premium и версиях продукта для рабочих станций и серверов. Эти версии также можно бесплатно скачать для предварительного ознакомления с ними в течение 30 дней.

Подобная ситуация с антивирусными программами Avast! – версию для домашнего использования *Free Antivirus* можно загрузить через интернет (*https://www.avast.ru/download-thank-you.php*) и использовать бесплатно после регистрации пользователя и получения ключа для использования через электронную почту. Другие версии: Avast Internet Security (Комплексная) и Avast Premier (Премиум) – являются платными, но их также можно бесплатно скачать для 30-дневного ознакомления.

5.6.2. Архивация файлов

Любая информационная система не может быть на 100% защищена от ошибок и сбоев, так как её проектируют, создают и реализуют люди, которым свойственно ошибаться. Даже если предположить, что в будущем удастся придумать и воплотить в жизнь идеальную систему, избежать возможных потерь данных всё равно не удастся, так как в ситуацию могут вмешаться сторонние факторы. В юридической практике давно используется понятие «форс-мажорных обстоятельств», или непреодолимой третьей силы. Защищаться от землетрясений, наводнений, ураганов, космического излучения и прочего мы пока не научились. В связи с этим единственным максимально эффективным способом защиты информации от потери является её размножение и копирование.

Копирование данных можно осуществлять вручную, а можно данный процесс автоматизировать. Также следует понимать, что какие-то данные являются более важными, а какие-то менее, и в случаях когда ресурсы системы ограничены, копирование может быть выборочным, а также могут использоваться различные алгоритмы сжатия. Подробнее см. раздел 2.5 «*Сжатие (архивация) различных видов информации*».

«Сжатие» – процесс уменьшения объёма, занимаемого какой-либо информацией. Бывает двух видов: «сжатие с потерей информации» (например, сохранение какого-либо изображения в формате .jpg) и «сжатие без потери информации» (например, использование модификаций архиваторов rar или zip). Если объём архива после сжатия оказывается больше, чем до, то сжимать файл нецелесообразно. В этом случае можно говорить об «отрицательном сжатии». Последняя фраза взята в кавычки, так как для увеличения объёма указанное словосочетание не является устоявшимся.

«Упаковка» - запись содержимого одного или нескольких файлов в один или несколько исходящих файлов. Упаковка файлов может производится как вместе с иерархией директорий, атрибутами файлов, времени создания и прочим, так и без оных. «Упаковка» может производиться как одновременно со сжатием, так и без него. Исходный файл может называться архивом, а программа, осуществляющая упаковку, – упаковщиком или архиватором. Если не используется сжатие, размер архива может быть больше суммы размеров файлов, входящих в него.

Процессы «копирования», «сжатия» и «упаковки» чаще всего нацелены на создание резервной копии, поэтому данные процессы вместе называют «архивацией».

В большинстве современных ОС имеются собственные средства архивации файлов, так, в ОС Linux имеются утилиты tar, gzip, bzip2 и др., в том числе и графические. ОС Windows, начиная с версии XP, умеет создавать ZIP-архивы. Делается это командой **Переместить** – **Папка архива** в контекстном меню или пункте **Файл** главного меню для группы выбранных файлов или папок.

Если большинство UNIX-пользователей не первое десятилетие используют следующий краткий справочник команд (см. табл. 5.17),

Таблица 5.17. Команды архивирования для ОС Linux

$ **tar -cf archive1.tar dir1/** Упаковать файлы директории dir1 в архив с именем archive1.tar без сжатия
$ **tar -czf archive2.tar.gz dir2/** Упаковать файлы директории dir2 в архив с именем archive2.tar.gz с сжатием алгоритмом gzip
$ **tar -cjf archive3.tar.bz2 dir3/** Упаковать файлы директории dir3 в архив с именем archive3.tar.bz с сжатием алгоритмом bzip2
$ **tar -xvf foo.tar** Распаковать файлы из архива «foo.tar»
$ **tar -xzf foo.tar.gz** Распаковать файлы из сжатого архива «foo.tar.gz»
$ **tar -xjf foo.tar.bz2 -C bar/** Распаковать файлы из сжатого архива foo.tar.bz2 в директорию «bar»
$ **tar -xzf foo.tar.gz aaaa1.txt** Извлечь из архива «foo.tar.gz» файл «aaaa1.txt»
$ **gzip file1** Сжать (заархивировать) файл «file1». После выполнения команды файл «file1» будет переименован в «file1.gz»
$ **gzip -c file1 > file2.gz** Создать заархивированный (сжатый) файл file2.gz из file1. После выполнения команды file1 останется. Ключ -c эквивалентен --stdout --to-stdout, знак «>» осуществляет перенаправление вывода из stdout в файл «file2.gz»
$ **gzip -d file1.gz** Расжать (разархивировать) файл «file1.gz». Ключ -d эквивалентен --decompress, --uncompress. По завершении работы команды файл «file1.gz» будет переименован в «file1»

то пользователи ОС Windows более приучены использовать сторонние коммерческие продукты. Наиболее известные программы-архиваторы **WinRar** [326] и **WinZip**, которые предоставляют пользователю дополнительные возможности, например, для WinRar:

– создание архивов в форматах RAR и ZIP;
– разархивация для архивов форматов RAR, ZIP, ACE, ARJ, CAB, ISO, JAR, LZH, TAR и др.;

[326] Для ОС Linux существует бесплатная версия консольной утилиты unrar.

- поддержка технологии перетаскивания при работе с архивами (drag & drop);
- возможность использования интерфейса командной строки;
- задание степени сжатия (без сжатия, скоростной, быстрый, обычный, хороший, максимальный);
- присутствие в контекстном меню работы с файлами и папками команд программы WinRar (добавить в архив, добавить в архив <имя>, добавить в архив и отправить по e-mail, добавить в архив <имя> и отправить по e-mail);
- поддержка кодировки Unicode в именах файлов;
- тестирование архивов после упаковки;
- кроме того, возможности только для формата RAR:
  - занесение в архив только файлов для заданных условий их изменения (с любым временем; старше, чем; новее, чем; измененные до; измененные после);
  - создание многотомных архивов, состоящих из нескольких файлов заданного размера для одного архива (имена файлов имеют вид «имя_тома.partNNN.rar», где NNN – номер тома);
  - создание самораспаковывающихся обычных и многотомных SFX-архивов (exe-файлов);
  - создание непрерывных (solid) архивов – архив RAR, упакованный специальным способом, при котором все сжимаемые файлы рассматриваются как один последовательный поток данных;
  - выбор – архивировать или нет открытые пользователями файлы;
  - возможность сохранения данных о правах доступа, в том числе данные из альтернативных потоков файла (для NTFS);
  - возможность восстановления физически повреждённых архивов;
  - шифрование данных и имён файлов при задании пароля архива;
  - задание параметров сжатия текста, аудио, графики, исполняемых файлов и структурированных таблиц;
  - добавление комментариев в архив, ведение протокола ошибок и пр.

5.6.3. Работа с оптическими дисками

Про форматы см. стр. 480.

Программы записи дисков

В ОС Linux для создания и записи дисков существует набор консольных утилит, например dvd+rw-tools. Исторически процесс записи состоял из двух частей, которые могли быть разнесены на разные компьютеры. Первая – создание образа диска с помощью утилиты mkisofs или growisofs, вторая – непосредственный прожиг iso-образа на диск. По мере развития аппаратных возможностей обе части стало возможно объединить в одну и сделать так называемую «запись или прожиг на лету», например командой

```
$ growisofs -Z /dev/dvd -R -J /файлы/для/записи
```

Так как многие серверы не имеют графической оболочки, то умение системными администраторами записывать диски через интерфейс командной строки (CLI) остаётся актуальным.

Начинающие пользователи предпочитают пользоваться удобным графическим интерфейсом (GUI) для записи дисков. С этой целью часто используются программы k3b и brasero. Первая по своему интерфейсу сильно напоминает Nero Burning ROM.

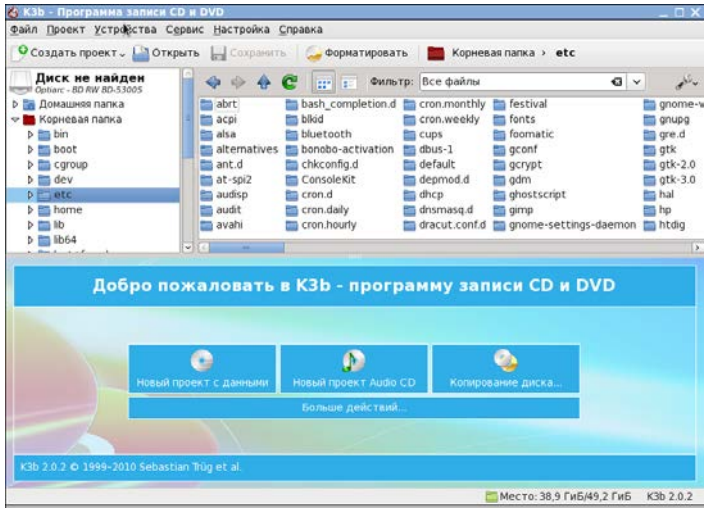


Рисунок 5.34. Вид окна программы K3B для записи CD/DVD/BD-дисков в ОС Linux

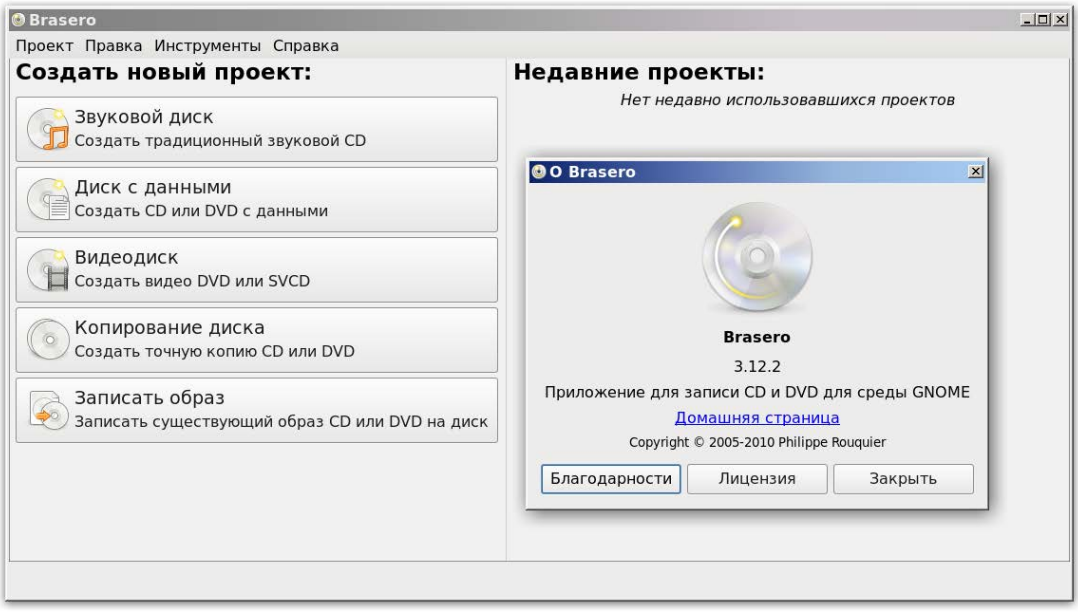


Рисунок 5.35. Вид окна программы barasero для записи CD/DVD/BD-дисков в ОС Linux

В операционной системе **Windows XP** и более старших версиях имеются собственные средства записи оптических дисков. **Windows Vista/7/8/8.1/10** имеют поддержку файловой системы **UDF** для работы с CD- и DVD-дисками.

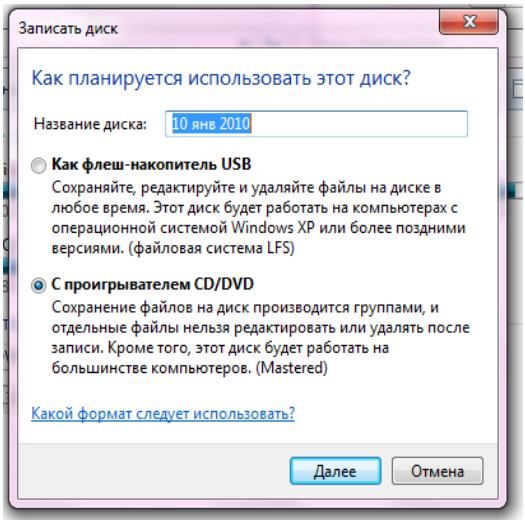


Рисунок 5.36. Подготовка диска к записи в Windows 7

На рис. 5.36 показано окно подготовки оптического диска к записи, которое появляется после двойного щелчка на иконке DVD-привода с чистым диском, в котором присутствуют 2 режима записи: **LFS** – Live File System (формат **UDF)** и **Mastered** (формат **ISO 9660**). Похоже, что новые названия (режимы) существовавшим форматам были придуманы специально «для упрощения» жизни пользователей.

чтение и запись как на флешку или диск	сессионная запись/дозапись
Формат UDF (разные версии)	Формат ISO9660 (с расширениями)

Напомним:

UDF имеет несколько версий: 1.02, 1.50, 2.00, 2.01, 2.50 и 2.60. Последняя версия – не значит лучшая, для совместимости лучше использовать версию 2.01.

Формат ISO9660 не позволяет записывать на диск файлы с размером более 2 ГБ.

В выпусках **Windows Home Premium** и **Ultimate** присутствует компонент ***Windows Media Center*** с большими возможностями по записи радио и телепередач, создания из аудио- и видеофайлов большого размера DVD-дисков для обычных бытовых DVD-плееров (с разбивкой на необходимое количество файлов *.vob с размером 1,048 ГБ и созданием необходимых информационных файлов).

Специализированные программы, предназначенные для записи и копирования CD- и DVD-дисков, имеют множество дополнительных возможностей. Наиболее распространённой программой этого типа на текущий момент можно считать коммерческую программу **Nero** фирмы **Ahead** [327]. В основном эта программа рассчитана на работу с ОС Windows, однако имеется и версия Nero для Linux. Последние версии программы (начиная с 8-й), кроме режимов записи дисков (Nero Burning ROM, Nero Express, Nero Start-Smart), имеют в своём составе ряд дополнительных программ для: управления домашним медиацентром (запись теле- и радиовещания с тюнера и из интернета); резервного копирования и восстановления (Backup) файлов, папок, логических дисков

[327] http://www.nero.com.

(на винчестер или оптический диск); записи и редактирования видео, фото и звука; просмотра DVD- и Blue-Ray-фильмов; тестирования и настройки приводов оптических дисков и прочего. (см. рис. 5.37).

При записи дисков возможны режимы:

- CD
  - ISO-диск (стандарт ISO 9660);
  - Аудио-CD
  - Mixed-CD (файлы и аудио-записи)
  - Копирование CD и создание образа диска
  - Видео-CD
  - Загрузочный CD
  - UDF CD; UDF/ISO CD
- DVD
  - DVD-ROM (ISO)
  - Копирование DVD
  - DVD-видео
  - DVD-ROM загрузочный
  - DVD-ROM (UDF)
  - DVD-ROM (UDF/ISO)
- HD DVD
  - HD DVD-ROM (UDF, форматы 1.02, 1.50, 2.00, 2.01, 2.50, 2.60)
- Blu-ray DVD
  - Blu-ray Disc (UDF, форматы 1.02, 1.50, 2.00, 2.01, 2.50, 2.60)

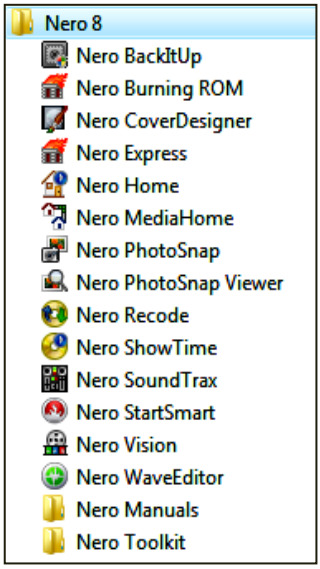


Рисунок 5.37. Меню системы Nero 8

Ещё одна распространённая программа в тему – **Alcohol 120%**. Её используются для создания образов CD-, DVD- и Blu-ray-дисков, создания виртуальных приводов CD и DVD (максимум 31 привод), а также для записи CD- и DVD-дисков, в том числе и копирования (с учётом различных нюансов). Поддерживаются образы в форматах MDF/MDS, CCD, BIN/CUE, ISO, CDI, BWT, B5T, B6T, BWI, BWS, BWA, ISZ. Кроме проприетарной версии имеется бесплатная полностью функциональная версия для некоммерческого использования Alcohol 120% Free Edition.[328]

5.6.4. Программы воспроизведения DVD-фильмов и видеофайлов, кодеки

Капиталисты везде хотят получать прибыль, поэтому простая функция воспроизведения DVD-фильмов в ОС Windows появилась лишь в **Vista Home Premium** и **Ultimate**. В предыдущих версиях стандартный *Универсальный проигрыватель* (*Windows Media Player*) мог проигрывать DVD-диски и различные видеофайлы только после установки дополнительных кодеков.

Кодеки – это условно небольшая инструкция для программы, воспроизводящей видео о том, как ей следует получать изображение из цифрового видеопотока того или

[328] Дополнительно рекомендуем посмотреть программы: UltraISO – для конвертирования образов и DAEMONTools – для эмулирования оптических приводов.

иного формата, считываемого с DVD, привода, из файла или получаемого по сети широковещанием. Поскольку существует много различных видеоформатов, логично, что для каждого формата должна быть своя «инструкция по воспроизведению». Чтобы не озадачивать пользователей установками кодеков, их распространяют в виде исполняемых файлов, которые можно запустить, а кодеки установить как программу. По сути, они и есть программы, но специфические. А раз это программы, то они защищаются законом об авторском праве, из чего следует, что не все из них, с юридической точки зрения, можно без денежных отчислений встроить в коммерческую операционную систему. С одной стороны, каждый хочет на своём формате заработать, с другой – понимает, что без широкого распространения формата (а это предположительно бесплатно) много не заработаешь. Посмотрев на вопрос с высоты философских размышлений, можно с досадой заметить, что даже в XXI веке человечество, с присущей ему разумностью, не научилось эффективно использовать все свои технические изобретения. Но, так как прогресс остановить невозможно, в интернете можно найти те или иные альтернативные кодеки. Наиболее популярным и полным набором кодеков для ОС Windows является пакет K-Lite Codec Pack [329].

Для пользователей MacOS, вместо K-Lite Codec Pack следует использовать пакет кодеков Perian [330]. Пользователи ОС Linux находятся в лучшем положении, так как они могут установить кодеки из основных или альтернативных репозитариев своего дистрибутива стандартными командами. Например, при установке бесплатного и кроссплатформенного медиапроигрывателя VLC (командами «yum install vlc» или «aptitude install vlc») кодеки будут установлены автоматически как зависимости, необходимые для полнофункциональной работы основного пакета.

Для воспроизведения DVD в ОС Windows наиболее широко используется коммерческая программа **Power DVD Player** корпорации **CyberLink**. В ряде случаев она бесплатно прилагается к DVD-приводам и некоторым видеокартам. Программа предназначена для воспроизведения мультимедиафайлов с CD-, DVD-дисков и с винчестера. Последние версии поддерживают воспроизведение не только старых форматов видео с разрешением 720×480 для NTSC и 720×576 для PAL, 8-канальный звук, технологию 24/96 LPCM вывода 24-битного звука с частотой выборки 96 кГц, но и большие разрешения и другие форматы.

Самой удачной, многофункциональной и рекомендуемой авторами для установки программой для воспроизведения видео вот уже много лет является простой, быстрый и мощный мультимедиапроигрыватель VLC.

«Проигрывает всё»: файлы, диски, веб-камеры, устройства и сетевые видеопотоки (например, IGMP). Воспроизводит большинство форматов без необходимости установки дополнительных кодек-паков: MPEG-2, DivX, H.264, MKV, WebM, WMV, MP3...

Программа является полностью свободной и кросс-платформенной. Работает на платформах Windows, Linux, Mac OS X, Unix... Не содержит рекламы и не следит за пользователями. Может конвертировать и передавать медиаданные по сети. На сайте http://videolan.org доступны исходные коды и исполняемые файлы.

[329] Доступен для бесплатного скачивания в версиях Bacis, Standard, Full, Mega по адресу http://www.codecguide.com/download_kl.htm.

[330] Доступны для бесплатного скачивания на http://www.perian.org.

В ОС Linux, если программа просмотра видео (mplayer, smplayer, vlc, xine и др.) ставится из пакетов, то установщик обычно разрешает зависимости и устанавливает всё необходимое. Конечно, после можно доставить дополнительные пакеты для поддержки старых и нестандартных форматов. Наиболее популярной является программа ffmpeg[331]. С помощью несложных консольных команд можно выполнить большинство часто используемых операций с видео, например

Извлечь звук из видеофайла:

```
$ ffmpeg -i video.MOV -vn audio.ogg
```

Обрезать видео по времени:

```
$ ffmpeg -i i.mp4 -ss 00:01:00 -t 00:02:00 -c copy o.mp4
```

Создать видео файл из фотографий:

```
$ ffmpeg -framerate 10 -pattern_type glob -i '*.jpg' -c:v libx264 o.mp4
```

Можно и наоборот, наделать фотографии из видео (конечно PrintScreen или Alr+PrintScreen никто ещё не отменял, как и стандартное графическое приложение «Сделать снимок экрана» делающее снимки с задержкой, если надо чтобы в снимке была какая-нибудь подсказка, всплывающая только при наведённии курсора):

```
$ ffmpeg -i o.mp4 -r 1 -q:v 2 -f image2 img-3%d.jpeg
```

Можно также открыть видео просмотр в графической среде и сделать фотографии экрана:

```
$ /usr/bin/xwd -display :0.0 -screen -root>screen.xwd
```

А если нужны jpg, то и сразу сконвертировать с помощью convert:

```
$ /usr/bin/xwd -display :0.0 -screen -root|/usr/bin/convert xwd:- -type
TrueColor -quality 85 screen.jpg
```

Повенуть видео:

```
$ ffmpeg -i i.mp4 -vf "transpose=0" o.mp4
```

Поместить 2 видео в одно (горизонтальное размещение, например как у видео с ШАД'а или малого ШАД'а яндекса, слева видео докладчика, справа видео от презентации):

```
$ ffmpeg -i i0.mp4 -i i1.mp4 -filter_complex hstack=inputs=2 o.mp4
```

Тоже самое, но вертикальное размещение:

```
$ ffmpeg -i i0.mp4 -i i1.mp4 -filter_complex vstack=inputs=2 o.mp4
```

Сконвертировать видео например от цифрового фотоаппарата в формат с лучшем сжатием:

```
$ ffmpeg -i MVI_4703.MOV MVI_4703.avi
```

5.6.5. Создание и просмотр специальных форматов документов

К подобным форматам относятся широко используемые **PDF**-документы, а также файлы форматов **DjVu**, **FB2** и, возможно, XPS (XML Paper Specification). Эти форматы создаются путём преобразования документов, подготовленных в текстовых редакторах и других программах, в новый формат, предназначенный в основном для чтения этих файлов (без возможности правки). Чтение файлов данных форматов (*.pdf, *.djvu, *.fb2, *.xps) может выполняться в специальных программах или в браузерах (интернет-обозревателях, при установленных в них для этого дополнительных программных компонентах).

[331] См. «19 команд ffmpeg для любых нужд» https://habr.com/ru/post/171213/, а также «FFmpeg. Трюки и хитрости» https://habr.com/ru/post/536170/ и «Укрощаем мультимедиа с помощью ffmpeg» https://habr.com/ru/post/333664/.

PDF-формат (Portable Document Format, формат документов корпорации Adobe Systems, международный стандарт ISO 32000) – платформонезависимый формат, предназначенный для представления в электронном виде полиграфической продукции. Позволяет сохранять точный внешний вид исходного документа за счёт внедрения необходимых шрифтов для текста, векторных и растровых изображений, форм и мультимедиа объектов. Поддерживает цветовые схемы RGB и CMYK, несколько типов сжатия встроенных растровых изображений.

Документы могут быть зашифрованы с заданием пароля на открытие документа или на его изменение, печать, копирование фрагментов и прочее (см. рис. 5.37). Однако в интернете легко можно найти программы, снимающие парольную защиту. При использовании некоторых новых возможностей последней версии формата 1.7 могут встречаться проблемы с открытием указанных документов старыми (более 5 лет) программами для просмотра. Решение проблемы – или пересохранить pdf-файл в более старой версии, или обновить просмоторщик.

Для просмотра **PDF**-документов можно использовать бесплатную программу **Adobe Reader** или программы других разработчиков (например, **Foxit Reader**). Чтобы создавать, добавлять, пересматривать, редактировать и коллективно использовать информацию, содержащуюся в PDF-файлах, следует использовать программы **Adobe Acrobat Professional** и **Adobe Acrobat Standard**.

DjVu (от фр. déjà vu – «уже виденное») – технология сжатия изображений с потерями, разработанная специально для хранения сканированных документов – книг, журналов, рукописей и прочего, где обилие формул, схем, рисунков и рукописных символов делает чрезвычайно трудоёмким их полноценное распознавание и перевод в другой формат. Также формат эффективен, если необходимо передать все нюансы оформления, например, исторических документов, где важное значение имеет не только содержание, но и цвет и фактура бумаги; дефекты пергамента: трещинки, следы от складывания; исправления, кляксы, отпечатки пальцев; следы, оставленные другими предметами, и т. д.

Существуют компьютерные программы просмотра файлов формата **DjVu** для различных платформ, в том числе в виде плагинов к браузерам, программных приложений для iOS- и Android-устройств, электронных книг, многие из них являются бесплатными. Наиболее популярная программа для просмотра – DjVuLibre [332], доступная для ОС Linux, Windows и MacOS.

В противовес хранению «тяжёлой» растровой графики используемые технологии сжатия предоставляют повышенную компрессию и высокое качество изображения. Степень сжатия изображений в **DjVu** значительно выше, чем в JPEG. Так же как и **PDF**, **DjVu**-документ может содержать текстовый слой, что позволяет производить такие привычные операции, как поиск по странице, копирование и вставка фрагментов текста и прочее.

В настоящее время документы многих онлайн-библиотек, архивов технической документации и любительских интернет-сайтов представлены в **DjVu**-формате [333].

Форматы PDF и DjVu изначально не планировались для использования в чёрно-белых электронных читающих устройствах с небольшим экраном («электронных книгах» или «читалках» в быту). Несмотря на то что данные файлы без проблем открыва-

[332] http://djvu.sourceforge.net/.
[333] Технически более подробное описание формата доступно по адресу http://djvu.sourceforge.net/abstract.html.

ются большинством таких устройств, в новых условиях у них проявились следующие недостатки, которые и создали спрос на появление ещё одного стандарта – FB2. Наличие графического слоя значительно увеличивает размер файлов. Даже если это не проблема при современных объёмах носителей, переформатирование документов для комфортного чтения под размер 6–7″ экрана без текстового слоя невозможно в принципе, да и при его наличии с этим справляются не все устройства.

Вместо простого чтения и перехода на следующую страницу пользователь вынужден видеть небольшой фрагмент А4-форматной PDF- или DjVu-страницы и много раз перемещаться вправо, влево, вверх и вниз для изучения её содержимого.

Новый формат FB2 лишён этих недостатков, предпочтительно хранит в себе только текст и позволяет его переформатировать подобно html-страницам в документе. Рассмотрим его подробнее.

FB2 (FictionBook) – формат представления электронных версий книг в виде XML-документов, где каждый элемент книги описывается своими тегами. Стандарт призван обеспечить совместимость с любыми устройствами и форматами. XML позволяет легко создавать документы, готовые к непосредственному использованию и программной обработке (конвертированию, хранению, управлению) в любой операционной среде. Документы, обычно имеющие расширение .fb2, могут содержать структурную разметку основных элементов текста, некоторое количество информации о книге, а также вложения с двоичными файлами, в которых могут храниться иллюстрации или обложка.

FB2-документ является файлом в формате XML

XML (англ. Extensible Markup Language – расширяемый язык разметки; рекомендованный Консорциумом W3C язык разметки, фактически представляющий из себя свод общих синтаксических правил. XML предназначен для хранения структурированных данных, для обмена информацией между программами, а также для создания на его основе более специализированных языков разметки (например, XHTML), иногда называемых словарями.

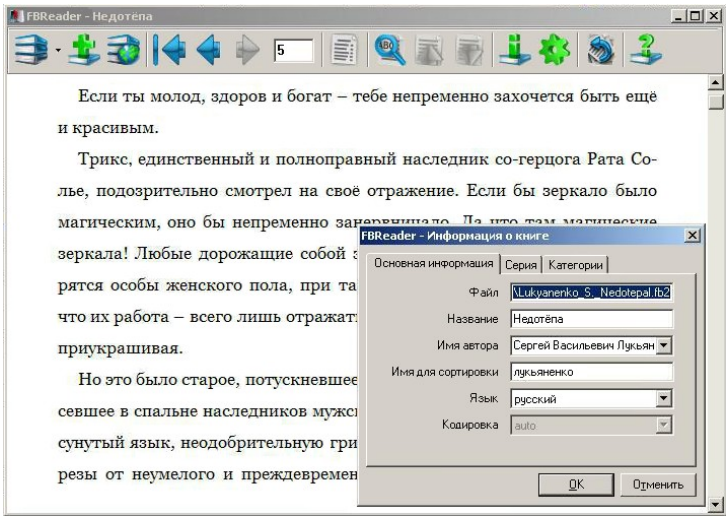


Рисунок 5.37. Программа FBReader для чтения файлов в формате TXT, HTML, CHM, RTF, OEB, FB2

Правильно подготовленный электронный текст в формате FictionBook содержит в себе всю необходимую информацию о книге – структурированный текст, иллюстрации, информацию об авторе и издании, но не содержит информацию о внешнем виде документа. Как будет выглядеть текст, полученный из формата .fb2, зависит либо от настроек программы-просмоторщика этого формата, либо от параметров, заданных при конвертировании файла в другой формат.

fb2 прекрасно позволяет организовать структуру книги (главы, подзаголовки, стихи, цитаты и т. д.). Он позволяет вставлять в текст иллюстрации.

Формат «.fb2» без проблем читается на «электронных книгах», ПК, планшетах и телефонах. Его понимает достаточное количество программного обеспечения. Для ОС Linux и Windows существует бесплатная программа FBReader.

5.7. Контрольные вопросы к главе 5

1. Как классифицируется ПО?
2. Зачем необходимо системное ПО?
3. Что такое прикладное ПО?
4. Приведите названия СУБД.
5. Назовите основные преимущества свободного ПО.
6. Как расшифровывается аббревиатура GPL?
7. Чем известен Ричард Столлман в компьютерном мире?
8. Что значит условно-бесплатные программы?
9. Где можно ознакомиться с текстом открытой общественной лицензии GNU General Public License?
10. По какой лицензии людям доступно использование ядра ОС Linux?
11. Что такое операционная система?
12. Какие задачи решает ОС?
13. В чём заключается задача разделения ресурсов компьютера между несколькими программами (пользователями)?
14. Назовите функции современной ОС для ПК.
15. Где можно ознакомиться с историей развития ОС для ПК (в виде графа)?
16. Какова история возникновения ОС Windows?
17. Какова история возникновения ОС Linux?
18. Для каких ОС доступен эмулятор DOSBox?
19. Чем известен миру Линус Торвальдс, каков его вклад в компьютерные науки?
20. Что такое POSIX?
21. Назовите основные русскоязычные дистрибутивы ОС Linux.
22. Какие отечественные дистрибутивы ОС Linux имеют те или иные сертификаты?
23. Какие форматы пакетов для распространения программ поддерживаются в ОС Linux?
24. Какие дистрибутивы ОС Linux имеют сертификацию ФСТЭК?
25. Что такое процесс в ОС?
26. Как ОС различает один процесс от другого?
27. Как посмотреть список выполняемых процессов в ОС?
28. В чём преимущества программы Process Explorer от Марка Руссиновича?
29. Какие полезные программы из набора «Sysinternals» вы знаете?
30. Дайте определение понятию «файл».
31. В чём состоит задача управления файлами?

32. Как производится именование файлов?
33. Какая максимальная длина имени файла поддерживается в современных ОС?
34. Какие символы запрещены для использования в именах файлов и директорий?
35. Какие файловые системы вы знаете?
36. Что значит древовидное представление объектов файловой системы?
37. Что есть метасимволы (символы-джокеры, wildcard characters), где они используются?
38. Что такое дескриптор файла?
39. Что такое разрежённый файл?
40. Какие стратегии организации адресного пространства внешних запоминающих устройств вы знаете?
41. Что делает операция монтирования?
42. Какие типы файлов вы знаете?
43. Назовите коротко основные положения стандарта иерархии файловых систем (Filesystem Hierarchy Standard, FHS).
44. Какие директории (и их предназначение) из стандарта FHS вам запомнились?
45. В чём суть задачи разграничения доступа между пользователями ОС?
46. Как реализуется дискреционная политика безопасности в ОС Linux?
47. Как реализуется дискреционная политика безопасности в ОС Windows?
48. В чём суть проблемы хранения учётных записей пользователей в ОС?
49. Какие стандартные и расширенные атрибуты файлов вы можете назвать и как они влияют на операции с указанным файлом?
50. Как поменять атрибуты (права доступа) у файла?
51. В чём различие абсолютного режима назначения прав и относительного?
52. Зачем нужен SetUID-бит?
53. Может ли обычный пользователь установить SetUID-бит на свои файлы?
54. Зачем нужен sticky-бит?
55. Кто может сменить владельца файла?
56. Какие стандартные разрешения использует ФС NTFS?
57. Что значит общий доступ к файлам и папкам?
58. Что такое политика безопасности (ПБ) и какие ПБ вы знаете?
59. Как устроена ФС FAT?
60. Как устроена ФС NTFS?
61. Как устроена ФС ext2?
62. Что такое альтернативные файловые потоки в ФС NTFS?
63. Какая информация хранится в индексном дескрипторе inode?
64. Зачем нужен суперблок и где хранится его копия?
65. Что такое фрагментация файла и на что она влияет?
66. Какие файловые системы используются при записи данных на оптические носители?
67. Какие программные компоненты ОС Windows вы знаете?
68. Назовите общее в ОС Windows и ОС Linux.
69. Как «вернуть» меню и кнопку «Пуск» в Windows 8 «обратно»?
70. Что значит термины синий экран «BSOD» и kernel panic?
71. Что делать в случае сбоя ОС?
72. Что значит «виртуализация» и какие возможности она предоставляет на сегодняшний день?
73. Что такое гипервизор и каково его функциональное предназначение.
74. Какие гипервизоры вы знаете?
75. Чем «гостевая ОС» отличается от «хостовой ОС»?
76. В каких ОС работает менеджер виртуальных машин Oracle VirtualBox?

77. Как вы понимаете фразу «сделать снимок виртуальной машины»?
78. Возможно ли две или три виртуальные машины объединить между собой виртуальной сетью?
79. В чём преимущества офисного пакета LibreOffice?
80. Под какими ОС можно использовать офисный пакет LibreOffice?
81. Из каких компонентов состоит офисный пакет LibreOffice?
82. Какие программные продукты поддерживают действующий национальный стандарт для офисных приложений на территории Российской Федерации ГОСТ Р ИСО/МЭК 26300–2010?
83. Работает ли LibreOffice под MacOS?
84. Что значит документ, оформленный с использованием стилей?
85. Как вставить таблицы, диаграммы и формулы в текст?
86. Как создаётся автоматическое оглавление?
87. Какие пакеты необходимо установить для поддержки проверки орфографии?
88. Какие ещё офисные пакеты вы знаете?
89. Зачем нужно сервисное ПО? Назовите примеры программ.
90. Что есть компьютерный вирус? Классификация и алгоритм работы.
91. Какие антивирусные программы вы знаете? Как они работают?
92. Что делать, если на компьютере найден вирус?
93. Откуда взять загрузочный CD (или загрузочную флэшку) с антивирусом?
94. Что значит архивация файлов?
95. Какие программы для записи компакт-дисков вы знаете?
96. Какие программы используются для воспроизведения DVD-дисков и видео?
97. Что такое кодек, зачем он нужен и где его (их) взять?
98. Как просмотреть PDF-файлы?
99. Как просмотреть DjVu-файлы?
100. Как просмотреть FB2-файлы?

5.8. Литература к главе 5

1. *Таненбаум Э.* Современные операционные системы. – 3-е изд. – СПб.: Питер, 2010. – 1120 с.: ил. ISBN 978-5-49807-306-4.
2. *Тейнсли Д.* Linux и Unix: программирование в shell: Руководство разработчика / пер. с англ. – Издательская группа BHV, 2001. – ISBN 966-522-085-7, ISBN 5-7315-0114-9.
3. *Митчел М., Оулдем Д., Самьюэл А.* Программирование для Linux. Профессиональный подход / пер. с англ. – М.: Издательский дом «Вильямс», 2002. – ISBN 5-8459-0243-6.
4. Доступный UNIX: Linux, FreeBSD, DragonFlyBSD, NetBSD, OpenBSD. – СПб.: БХВ-Петербург, 2006. – ISBN 5-94157-876-8.
5. *Соломон Д., Руссинович М.* Внутреннее устройство Microsoft Windows 2000. Мастер-класс / пер с. англ. – СПб.: Питер; М.: Издательско-тороговый дом «Русская редакция», 2001. – 752 с.: ил. ISBN 5-318-00545-4, ISBN 5-7502-0136-4.

6. *Таненбаум Э., Уэзеролл Д.* Компьютеные сети. – 5-е изд. – СПб.: Питер, 2012. – 960 с.: ил. ISBN 978-5-459-00342-0.
7. Filesystem Hierarchy Standard – домашняя страница проекта по иерархии файловых систем http://www.pathname.com/fhs/, последний стандарт 2.3 http://www.pathname.com/fhs/pub/fhs-2.3.pdf.
8. *Манн С., Митчелл Э. Л., Крелл М.* Безопасность Linux. – 2-е изд. / пер. с англ. – М.: Издательский дом "Вильямс", 2003. – ISBN 5-8459-0485-4.
9. *Бэндл Д.* Защита и безопасность в сетях Linux: Для профессионалов (+CD). – СПб.: Питер, 2002. – ISBN 5-318-00057-6.
10. *Фликенгер Р.* Взломы и настройка LINUX. 100 профессиональных советов и инструментов: практ. пособие / Р. Фликенгер, пер. с англ. – М.: Издательство ЭКОМ, 2006. – ISBN 5-7163-0121-5.
11. *Мешков В.* Архитектура файловой системы ext2 // Системный администратор. – 2003. – № 11 (12). – С. 26–32. URL: http://www.samag.ru/archive/article/203.
12. *Костромин В.* Права доступа к файлам и каталогам // http://www.linuxcenter.ru/lib/books/kostromin/gl_04_05.phtml. – 2010. – Сент.
13. *Frank Mayer, Karl MacMillan, David Caplan*, SELinux by Example: Using Security Enhanced Linux – Publisher: Prentice Hall; Pub Date: July 27, 2006; ISBN-13: 978-0-13-196369-6.
14. *Торчинский Ф.* UNIX. Практическое пособие администратора. – СПб: Символ-Плюс, 2003. – ISBN 5-93286058-8.
15. In UNIX Everything is a File // http://ph7spot.com/musings/in-unix-everything-is-a-file. – 2010. – Окт.
16. *Шрёдер К.* Linux. Сборник рецептов. – СПб.: Питер, 2006. – ISBN 5-46901188-7.
17. FAQ по FreeBSD 4.X, 5.X и 6.X // http://www.freebsd.org/doc/ru/books/faq/misc.html. – 2010. – Окт.
18. *Колисниченко Д. Н.* Серверное применение Linux. – 3-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2011. – 528 с.: ил. ISBN 978-5-9775-0652-6.
19. *Эбен М., Таймэн Б.* FreeBSD. Platinum Edition / пер. с англ. М. Эбен, Б. Таймэн – СПб.: ООО "ДиаСофтЮП", 2003. – ISBN 5-93772-107-1.
20. *Смит Р.* Полный справочник по FreeBSD / пер. с англ. – М.: Издательский дом «Вильямс», 2005. – ISBN 5-8459-0576-1.
21. *Федорчкук А. В., Торн А. В.* FreeBSD: установка, настройка, использование. – СПБ.: БХВ-Петербург, 2003. – ISBN 5-94157-200-X.
22. *Робачевский А. Н., Немнюгин С. А., Стесик О. Л.* Глава 1. Работа в операционной системе UNIX // Операционная система UNIX. – 2-е изд. – СПб.: БХВ-Петербург, 2008. – С. 40–43 – 656 с. – ISBN 978-5-94157-538-1.
23. *Шаньгин В. Ф.* Комплексная защита информации в корпоративных системах: учеб. пособие / В. Ф. Шаньгин. – М.: ИД «Форум», ИНФРА-М, 2010. ISBN 978-5-8199-0411-4, ISBN 978-5-16-003746-2.
24. *Закляков П.* Скупой платит дважды, а умный использует GNU Public Licence // Системный администратор. – 2003. – № 9 (10). – С. 80–86.
25. *Олифер В. Г., Олифер Н. А.* Сетевые операционные системы. – СПб.: Питер, 2001. – 544 с.:ил. ISBN: 5-272-00120-6.
26. *Касперски К.* Записки исследователя компьютерных вирусов. – СПб.: Питер, 2005. – 316 с.: ил. ISBN 5-469-00331-0.

27. *Бирюков А.* Обзор офисных приложений под Android // Системный администратор. – 2013. – № 5 (126). – С. 12–15.
28. *Борзенко А.* Кто был ничем (На рынке мобильников «для богатых» готовится пролетарская революция) // Итоги. – 2013. – № 12 (876). – С. 42–44. http://www.itogi.ru/hitech-business/2013/12/188273.html.
29. *Денищенко Н.* Устройство и криптоанализ UUID-генератора в ОС Windows // RSDN Magazine №2-2008, http://rsdn.ru/article/Crypto/UuidCrypto.xml.
30. *Гордеев А.В., Молчанов А.Ю.* Системное программное обеспечение – СПб.: Питер, 2001. – ISBN 5-272-00341-1.
31. Design and Implementation of the Second Extended Filesystem http://e2fsprogs.sourceforge.net/ext2intro.html
32. Analyzing a filesystem http://homepage.smc.edu/morgan_david/cs40/analyze-ext2.htm
33. Ext4 Disk Layout https://ext4.wiki.kernel.org/index.php/Ext4_Disk_Layout
34. Документация по ФС ext2-4 из исходников ядра (v.3.16.44 на kernel.org) файлы Documentation/filesystems/ext2.txt (ext3.txt, ext4.txt), либо из пакета kernel-doc-2.6.32-642.15.1.el6.noarch файлы /usr/share/doc/kernel-doc-2.6.32/Documentation/filesystems/ext2.txt, ext3.txt, ext4.txt.
35. A Non-Technical Look Inside the EXT2 File System (архивная версия страницы http://web.archive.org/web/20040301054851/http://linuxgazette.net/issue21/ext2.html)
36. The Second Extended File System http://www.nongnu.org/ext2-doc/ext2.html
37. debugfs Command Examples https://www.cs.montana.edu/courses/309/topics/4-disks/debugfs_example.html
38. Linux Ext2fs Undeletion mini-HOWTO http://www.faqs.org/docs/Linux-mini/Ext2fs-Undeletion.html
39. *Яремчук С.* Расширяем права доступа в Linux с помощью ACL // Системный администратор. – 2005. – № 11 (36). – С. 54–59. URL: http://www.samag.ru/archive/article/582.
40. *Манин Ю.И.* Вычислимое и невычислимое – М.: Сов. радио, 1980 – 128 с.

Приложение к главе 5. Примеры лабораторных работ

Лабораторная работа № 1. Разграничение доступа

Цель работы: получение навыков работы с консолью, проверка работы системы разграничения доступа и опытная оценка влияния установленных атрибутов на взаимодействие пользователя с файловыми объектами.

Организация и описание лабораторного стенда

Для выполнения работы потребуется компьютер с установленной ОС Linux, использующий одну из файловых систем ext2–ext4 для хранения файлов пользователей (раздел / или /home). Небольшое количество свободного места на диске (менее 1 МБ) для создания временных файлов и директорий. Права администратора для пунктов 27–35 (задание 3). Наличие графического интерфейса X-Window не требуется.

Специальных требований к дистрибутиву ОС Linux нет, все вышеуказанные требования могут быть с лёгкостью выполнены практически на любом дистрибутиве ОС Linux, выпущенном в течение последнего десятилетия (CentOS 6, 7, 8, Debian 9, 10, Альт Рабочая станция 8, 9 и др.).

Лабораторная работа выполняется одним обучаемым за одним компьютером. При использовании компьютерных классов возможно одновременное выполнение работы несколькими обучающимися, каждый за своим рабочим местом. По мнению авторов, проведение работы в группах 7–11 человек является наиболее оптимальным. [334]

Подготовка лабораторного стенда

Установите операционную систему Linux. Для того чтобы обучаемые не отвлекались на различные возможности графического интерфейса системы (GUI), работу рекомендуется проводить через интерфейс командной строки (CLI, в текстовом режиме). Для этого либо установите систему без поддержки сервера X-Window, либо осуществите загрузку в 3-м уровне (например, в /etc/inittab можно прописать id:3:initdefault:, либо выполните команду init 3), либо переключитесь в одну из текстовых консолей нажатием клавиш **Ctrl**+**Alt**+**F*n***, где *n* = 1...6;

[334] При необходимости возможно проведение работы удалённо, в том числе и из-под других операционных систем, например с использованием утилиты PuTTY или любого другого SSH-клиента, так и в среде виртуальных машин. Особого внимания заслуживает красивое решение: возможность использования полностью бездискового дисплейного класса (на базе проекта k12linux.org), когда один из компьютеров загружается с flash-накопителя и выполняет функции «раздающего» сервера (PXE, DHCP, NFS, X-Server и прочего), а все остальные компьютеры либо полностью загружаются с него по сети, либо как терминальные клиенты. Указанная возможность не связана с данной работой, поэтому она здесь рассматриваться не будет. Указанные способы усложнения лабораторного стенда рассматриваться ниже не будут и не рекомендуются, так как это потребует больших знаний со стороны отвечающего за проведение работы персонала, а с педагогической точки зрения, при возникновении каких-либо неполадок, отсутствие должной помощи обучаемым, может снизить эффективность выполнения всей работы.

Предпочтение к использованию интерфейса CLI определяется отчасти ещё и тем, что большинство бытовых сетевых устройств (маршрутизаторы, Wi-Fi, DSL и другие модемы, совмещённые с маршрутизаторами, HDD-видеоплееры, спутниковые декодеры и прочее), использующих ОС Linux в качестве своей «бортовой» операционной системы, не имеют в своём составе графического сервера. Веб-интерфейс либо отсутствует, либо не предоставляет полных возможностей по настройке. В жизни можно привести много случаев, когда навыки, полученные при работе через интерфейс CLI, не были лишними.

Создайте учётную запись пользователя guest, для дальнейшей передачи её параметров обучаемым

```
# useradd guest
```

и задайте пароль для пользователя guest.

```
# passwd guest
```

Для выполнения второй части задания потребуется аналогично создать второго пользователя в системе, например guest2.

Также добавьте пользователя guest2 в группу guest.

```
# gpasswd -a guest2 guest
```

Для выполнения третьей части задания обучающимся потребуются права администратора либо доступ к команде sudo.

Замечание. В случае повторного использования одного и того же лабораторного стенда требуется и повторная подготовка, которая сводится к удалению всех созданных при проведении предыдущей работы файлов и поддиректорий в домашней директории пользователя и в /tmp. В том числе не забудьте и про .bash_history.

Для выполнения первой части задания (не требующей прав администратора) стенд готов.

Краткий справочник команд

Операция	Примеры команд

Создание файла или изменение атрибутов времени; Создание файла; Дозапись в файл	`touch file`
	`echo "hello">file1`
	`echo "test">>file2`

Удаление файла	`rm file1` `rm /home/guest/file1`
Чтение файла	`cat /home/guest/file1` `cat ~/file1.txt`

Запись в файл
Дозапись в файл
Запись в файл копированием
	`echo "hello">file1`
	`echo "test">>file2`
	`cp source_file file1`

Переименование файла	`mv file2 file3` `mv /home/guest/file1 /home/guest/file2`
Создание директории (поддиректории)	`mkdir aaa` `mkdir –p /home/guest/aaa`
Удаление директории (поддиректории) и файлов в ней	`rm -r -f /home/guest/aaa`

Операция	Примеры команд

Изменение прав файлового объекта (файла, директории)	`chmod 000 file1.txt`
	`chmod u+r file2`
	`chmod -x dir1`
	`chmod g+s file3`
	`chmod 4755 file4`

Выполнение файла	`./file1` `/home/guest/bbb`
Изменение расширенных атрибутов	`chattr +a file1` `chattr -V +a /var/log/file2`

Просмотр списка файлов в директории	`ls -l`
	`ls -1`
	`ls`
	`ls  -l /home/guest/dir2`

Смена директории	`cd /home/guest/dir2`
	`cd ..`
	`cd ../dir1`
	`cd`
	`cd ~/dir1`

Замечание. Обратите внимание, что не все команды на 100% подходят под указанную операцию и только под неё. Например, одна и та же команда может выполнять одновременно несколько действий. Допустим, команда «`echo "test">>file2`» если файла file2 нет, создаёт его, а потом выполняет запись, а если есть – делает сразу «дозапись». Для правильного использования команд при выполнении работы используйте литературу, помощь к командам или «man».

Задание 1 на лабораторную работу (основные атрибуты)

1. Получите у руководителя занятия информацию об учётных записях пользователей и их паролях. Постарайтесь последовательно выполнить все последующие пункты, заносяся ваши ответы на поставленные вопросы и замечания в отчёт.

2. Войдите в систему от имени полученного пользователя (в нашем случае guest).

3. Определите директорию, в которой вы находитесь, командой pwd. Сравните её с приглашением командной строки. Определите, является ли она вашей домашней директорией. Если нет – зайдите в домашнюю директорию.

4. Уточните имя вашего пользователя командой whoami.

5. Уточните имя вашего пользователя, его группу, группы, куда входит пользователь, командой id. Выведенные значения uid, gid и другие запомните. Сравните вывод id с выводом команды groups.

6. Сравните полученную информацию об имени пользователя с данными, выводимыми в приглашении командной строки.

7. Просмотрите файл /etc/passwd командой
```
$ cat /etc/passwd
```

Найдите в нём свою учётную запись. Определите uid пользователя. Определите gid пользователя. Сравните найденные значения с полученными в п. 5.

Замечание. В случае, когда вывод команды не умещается на одном экране монитора, используйте прокрутку вверх-вниз (удерживая клавишу **shift**, нажимайте **page up** и **page down**) либо программу grep в качестве фильтра для вывода только строк, содержащих определённые буквенные сочетания

```
$ cat /etc/passwd|grep "guest"
```

8. Определите, какие пользовательские директории существуют в системе, командой

```
$ ls -l /home
```

Удалось ли вам получить список поддиректорий директории /home? Какие права установлены на директориях, в том числе на вашей?

9. Проверьте, какие расширенные атрибуты установлены на поддиректориях, находящихся в директории /home, командой

```
$ lsattr /home
```

Удалось ли вам увидеть расширенные атрибуты вашей домашней директории?
Удалось ли вам увидеть расширенные атрибуты директорий других пользователей?

10. Создайте в домашней директории поддиректорию dir1 командой

```
$ mkdir dir1
```

11. Определите, какие права доступа и расширенные атрибуты были выставлены на директорию dir1, командами «`ls -l`» и «`lsattr`».

12. Снимите с директории все атрибуты командой

```
$ chmod 000 dir1
```

и проверьте с помощью команды правильность выполнения команды

```
$ ls -l
итого 4
d---------. 2 guest guest 4096 Сен 27 14:04 dir1
```

13. Попытайтесь создать в директории dir1 файл file1 командой

```
$ echo "test">/home/guest/dir1/file1
-bash: /home/guest/dir1/file1: Отказано в доступе
```

Объясните, почему вы получили отказ в выполнении операции по созданию файла.

14. Оцените, как сообщение об ошибке отразилось на создании файла. Проверьте, действительно ли файла file1 нет внутри директории dir1, командой

```
$ ls -l /home/guest/dir1
```

Удалось ли вам получить ответ на ваш вопрос?

15. Заполните таблицу «Установленные права и разрешённые действия» (см. табл. ЛР1.1 *а*,*б*), выполняя действия от имени владельца директории (файлов), определив опытным путём, какие операции разрешены, а какие нет. Если операция разрешена, занесите в таблицу знак «+», если не разрешена – знак «−». Вдумчивое выполнение команд, осмысление полученного результата и заполнение таблицы займёт у вас ориентировочно около получаса.

Таблица ЛР1.1 а. Установленные права и разрешённые действия

Установленные права		Разрешённые действия						
директории*	файла*	Удаление файла	Запись в файл	Чтение файла	Просм. файлов в дир.	Переименование файла	Смена атрибутов файла	
							права	время
0000/ `d---------`	0000/ `----------`	–	–	–	–	–	–	
0100/ `d--x------`	0000/ `----------`	–	–	–	–			
0200/ `d-w-------`	0000/ `----------`							
...	...							
0700/ `drwx------`	0700/ `-rwx------`	+	+	+	+	+	+	+

|* Права файлов и директорий в этой последующих таблицах отображаются как вывод команды «stat». Поля таблицы могут не применяться, например, для «создания файла» не применимы «права файла», так как файл ещё не создан.

Таблица ЛР1.1 б. Установленные права и разрешённые действия

Права директории	Создание в директории			Смена директории (заход в неё)
	файла	поддиректории	мягкой ссылки	
0000/ `d---------`	–	–	–	–
0100/ `d--x------`				+
0200/ `d-w-------`				
...				
0700/ `drwx------`	+	+	+	+

Замечание 1. При заполнении табл. ЛР1.1 (*а*, *б*) рассматриваются не все атрибуты файлов и директорий, а лишь «первые три»: **r**, **w**, **x**, для «владельца». Естественно, остальные атрибуты также важны, особенно при использовании доступа от имени разных пользователей, входящих в те или иные группы. Проверка всех атрибутов при всех условиях значительно увеличило бы таблицу: так, 9 атрибутов на директорию + 9 атрибутов на файл = 18 атрибутов, дают 2^{18} строк (это не учитывая дополнительных атрибутов), плюс таблица была бы расширена по количеству столбцов, так как все приведённые операции необходимо было бы повторить ещё как минимум для двух пользователей: входящего в группу владельца файла и не входящего. После полного заполнения табл. ЛР1.1 и анализа полученных данных нам удалось бы выяснить, что заполнение её в таком виде излишне, можно разделить большую таблицу на несколько малых независимых. Поэтому в данном задании мы рассмотрели малую таблицу, «сняв» для наглядности лишние неиспользуемые атрибуты, так как они не влияют на заполнение таблицы. В данном примере предлагается рассмотреть 3 + 3 атрибута, то есть $2^6 = 64$ варианта. На первый взгляд, это число может также показаться большим, но обдуманный подход к выполнению задания позволит в процессе работы не только быстро заполнить таблицу, исходя из теоретических знаний, но и проверить её заполнение на практике, уложившись при этом в отведённые полчаса.

Замечание 2. В ряде действий при выполнении команды удаления файла вы можете столкнуться с вопросом «`rm: удалить защищённый от записи пустой обычный файл `dir1/file1'?`» Обратите внимание, что наличие этого вопроса не позволяет сделать правильный вывод о том, что файл можно удалить. В ряде случаев при ответе «y» (да) на указанный вопрос возможно получить в ответ другое сообщение «`rm: невозможно удалить `dir1/file1': Отказано в доступе`».

16. На основании заполненной таблицы из п. 15 определите те или иные минимально необходимые права для выполнения операций внутри директории dir1 и заполните нижеследующую табл. ЛР1.2.

Таблица ЛР1.2. Минимальные права для совершения операций

Операция	Минимальные права на директорию	Минимальные права на файл (поддиректорию)
Создание файла		—
Удаление файла		
Чтение файла		
Запись в файл		
Переименование файла		
Создание поддиректории		—
Удаление поддиректории		

Задание 2 на лабораторную работу (два пользователя)

17. По полученным в п. 1 (части 1) данным осуществите вход в систему от двух пользователей на двух разных консолях. Например, guest на первую консоль, а guest2 – на вторую. Переключение между консолями: **Alt**+**F1**, **Alt**+**F2**, **Alt**+**F3**...

18. Для обоих пользователей: определите директорию, в которой вы находитесь, командой pwd. Сравните её с приглашениями командной строки.

19. Уточните имя вашего пользователя, его группу, кто входит в неё и к каким группам принадлежит он сам. Определите, в какие группы входят пользователи guest и guest2, командами «`groups guest`» и «`groups guest2`». Сравните вывод команды `groups` с выводом команд «`id -Gn`» и «`id -G`».

20. Сравните полученную информацию в п. 19 с содержимым файла /etc/group. Просмотрите файл командой «`cat /etc/group`».

21. От имени пользователя guest2 выполните регистрацию пользователя guest2 в группе guest командой[335].

```
$ newgrp guest
```

22. От имени пользователя guest измените права директории /home/guest, разрешив все действия для пользователей группы.

```
$ chmod g+rwx /home/guest
```

[335] Представьте аналогию: можно записаться в разные спортивные секции: плавания, хоккея и футбола, но это не тоже самое, что придти по этим секциям на занятие. Без предварительной записи в группу тренер не пускает и присутствовать одновременно можно только или в бассейне, или на катке, или футбольном поле. Запись в секции, то есть создание групп, осуществляется на этапе подготовки лабораторного стенда.

23. От имени пользователя guest снимите с директории /home/guest/dir1 все атрибуты командой

```
$ chmod 000 dir1
```

и проверьте правильность снятия атрибутов.

24. Меняя атрибуты у директории dir1 и файла file1 от имени пользователя guest и делая проверку от пользователя guest2, заполните таблицу «*Установленные права и разрешённые действия для члена группы*» (см. табл. ЛР1.3), определив опытным путём, какие операции разрешены, а какие нет. Если операция разрешена, занесите в таблицу знак «+», если не разрешена – знак «−».[336]

Таблица ЛР1.3 а. Установленные права и разрешённые действия для члена группы

Установленные права		Разрешённые действия						
директории	файла	Удаление файла	Запись в файл	Чтение файла	Просм. файлов в дир.	Переименование файла	Смена атрибутов файла	
							права	время
0000/ d---------	0000/ ----------	–	–	–	–	–	–	
0010/ d-----x---	0000/ ----------	–	–	–	–			
0020/ d----w----	0000/ ----------							
0030/ d----wx---								
…	…							
0070/ d---rwx---	0700/ -rwx------	+	+	+				

25. Сравните табл. ЛР1.1 *а* из первой части задания и табл. ЛР1.3 *а* из второй.

26. На основании заполненной таблицы из п. 24 определите те или иные минимально необходимые права для выполнения пользователем guest2 операций внутри директории dir1 и заполните нижеследующую табл. ЛР1.4.

Таблица ЛР1.4. Минимальные права для совершения операций от имени пользователей входящих в группу

Операция	Минимальные права на директорию	Минимальные права на файл
Создание файла		—
Удаление файла		
Чтение файла		
Запись в файл		
Переименование файла		
Создание поддиректории		—
Удаление поддиректории		

[336] Таблица носит общий вид, поэтому какие-то её части могут быть излишними, например, для колонки «создание файла» не применимы «права файла»; без разницы какие они будут, так как файл ещё не создан.

Задание 3 на лабораторную работу (расширенные атрибуты)

(необходимы права администратора)

27. От имени пользователя guest определите расширенные атрибуты файла /home/guest/dir1/file1 командой

```
$ lsattr /home/guest/dir1/file1
```

28. Установите на файл file1 права разрешающие чтение и запись для владельца файла командой

```
$ chmod 600 /home/guest/dir1/file1
```

29. Попробуйте установить расширенный атрибут «a» на файл /home/guest/dir1/file1 от имени пользователя guest:

```
$ chattr +a /home/guest/dir1/file1
```

в ответ вы должны получить отказ от выполнения указанной операции

```
chattr: Операция не позволяется while setting flags on
/home/guest/dir1/file1
```

30. Зайдите на третью консоль с правами администратора либо повысьте свои права с помощью команды «`su -`». Попробуйте установить расширенный атрибут «a» на файл /home/guest/dir1/file1 от имени суперпользователя.

```
# chattr +a /home/guest/dir1/file1
```

31. От пользователя guest проверьте правильность установления атрибута.

```
$ lsattr /home/guest/dir1/file1
-----a-------e- /home/guest/dir1/file1
```

(***Замечание.*** Шага 31 можно избежать, если на шаге 30 к команде `chattr` добавить ключ `-V`.)

32. Выполните дозапись в файл file1 слова «test» командой

```
$ echo "test">>/home/guest/dir1/file1
```

после чего выполните чтение файла file1 командой

```
$ cat /home/guest/dir1/file1
```

и убедитесь, что слово test было успешно записано в file1.

33. Попробуйте удалить файл file1 либо стереть имеющуюся в нём информацию командой

```
$ echo "abcd">/home/guest/dir1/file1
```

Попробуйте переименовать файл.

Попробуйте установить на файл file1 права, например, запрещающие чтение и запись для владельца файла командой

```
$ chmod 000 file1
```

Удалось ли вам успешно выполнить указанные команды?

34. Снимите расширенный атрибут «a» с файла /home/guest/dir1/file1 от имени суперпользователя командой

```
# chattr -a /home/guest/dir1/file1
```

Повторите операции, которые вам ранее (на предыдущем шаге) не удавалось выполнить. Ваши наблюдения занесите в отчёт.

35. Повторите ваши действия по шагам 29–34, заменив атрибут «a» атрибутом «i». Удалось ли вам дозаписать информацию в файл? Ваши наблюдения занесите в отчёт.

Заключение

В результате выполнения работы вы повысили свои навыки использования интерфейса командой строки (CLI), познакомились на примерах с тем, как используются основные и расширенные атрибуты при разграничении доступа. Имели возможность связать теорию дискреционного разграничения доступа (дискреционная политика безопасности) с её реализацией на практике в ОС Linux. Составили наглядные таблицы, поясняющие, какие операции возможны при тех или иных установленных правах. Опробовали действие на практике расширенных атрибутов «a» и «i». К сожалению, в процессе работы часть существующих атрибутов рассмотрены не были: SUID-, SGID- и sticky-биты, расширенный атрибут «s» и другие.

Лабораторная работа № 2. Исследуем inode

Цель работы: получение навыков работы с консольными утилитами, в частности debugfs и od, исследование индексного дескриптора ФС семейства «ext».

Подготовка лабораторного стенда

Для проведения работы подойдёт любая установленная 64-битная система из CentOS 6, 7, 8, Debian 9, 10, Альт Рабочая станция 8, 9. С минимальными коррективами данная работа может быть проведена с использованием других дистрибутивов ОС Linux (в том числе 32-битных), при условии, что в них будут установлены нижеупомянутые основные и дополнительные программы. Все задания выполняются из под консоли от обычного пользователя. Графический интерфейс не требуется, но его наличие не возбраняется. Наличие прав администратора для выполнения заданий не требуется, за исключением этапа подготовки лабораторного стенда (установка дополнительных пакетов, внесение изменений в файл /etc/sudoers).

Этапы настройки лабораторного стенда следующие:

1. Произвести базовую (минимальную) установку вышеупомянутой системы (требуется первый установочный диск, CentOS-7-x86_64.iso).

2. Создать учётную запись пользователя, от которого будет выполняться работа.

```
# adduser user
# passwd user
```

3. Произвести дополнительную установку пакетов с установочного диска или с серверов обновления: man, tree, man-pages-ru (не обязательно), gpm-libs (необходим для lde).

```
# yum -y install tree man man-pages-ru gpm-libs
```

Вместе с man будут установлены зависимые пакеты: xz, xz-lzma-compat.

4. Произвести дополнительную установку пакетов из репозитария EPEL: lde.

```
# yum -y install lde
```

или

```
# rpm -ihv lde-2.6.1-7.el7.x86_64.rpm
```

5. Внести изменения в файл /etc/sudoers (дописать нижеследующие строчки), чтобы указанные в них команды можно было запускать через механизм «sudo» от обычного пользователя.

```
guest ALL=(ALL) NOPASSWD: /bin/mount -o loop disk100ext2 /mnt/ext2
guest ALL=(ALL) NOPASSWD: /bin/mount -o loop disk100ext4 /mnt/ext4
guest ALL=(ALL) NOPASSWD: /bin/umount /mnt/ext2
guest ALL=(ALL) NOPASSWD: /bin/umount /mnt/ext4
guest ALL=(ALL) NOPASSWD: /bin/mount -o loop,user_xattr,acl disk100ext2 /mnt/ext2
guest ALL=(ALL) NOPASSWD: /bin/mount -o loop,user_xattr,acl disk100ext4 /mnt/ext4
guest ALL=(ALL) NOPASSWD: /usr/sbin/setcap cap_sys_admin+ep /mnt/ext2/file1
guest ALL=(ALL) NOPASSWD: /usr/sbin/setcap cap_sys_admin+ep /mnt/ext4/file1
```

6. Создать директории для последующего монтирования в них исследуемых ФС.

```
# mkdir /mnt/ext2
# mkdir /mnt/ext4
```

7. Проверить наличие программ debugfs (пакет e2fsprogs) и od (пакет coreutils), в случае их отсутствии – установить.

Задания для самостоятельного выполнения

Часть 1. Взаимодействие с inode через стандартные программы на пользовательском уровне

1. Первое место, откуда обычный пользователь может узнать о существовании inode, даже ни имея ни капли теоретических знаний, – это из вывода ряда консольных программ при их запуске с соответствующими ключами. При запуске команд «`df -i`», «`ls -i`», «`tree -a -L 1 --inodes /`» стандартный вывод утилит дополняется информацией по inode'ам. Первая программа показывает свободное место (от англ. disk free) на примонтированных разделах жёсткого диска, вторая – список (от англ. list) файлов в рабочей директории, третья – содержимое корневой директории. Запустите первые две команды с ключом «`-i`» и без него, а третью – с и без «`--inodes`», проанализируйте их вывод. (В помощь вам существуют команды «`man ls`», «`man df`», «`man tree`».)

2. Права объекта файловой системы хранятся в его inode. Увидеть их можно командами «ls -li имя_объекта» или «stat имя_объекта». Используя их:

2.1. Просмотрите информацию для корня (/) файловой системы и директорий: /etc, /home, /tmp, причём, сделайте это как из корня, так и из самих директорий. Запомните номера inode'ов в привязке к именам директорий. (Смена директории производится командой cd с указанием её названия. Запуск cd без параметров помещает вас в домашнюю директорию. Посмотреть текущую (рабочую) директорию можно командой pwd.) Почему номер inode'а корневой директории ФС («/» и «.» в ней) имеет номер 2, а не 1? Сразу отметим, что это не журнал, поскольку в ФС ext2 он не предусмотрен, а номер будет также 2, а не 1. Для журнала ФС забронирован номер 8. Внимательно просматривая результат выполнения команды «ls -li /», можно заметить что традиционно первым свободным для использования номером inode для файлов и директорий является номер 11, он же прописан в параметре s_first_ino in суперблока, но чаще всего по техническим моментам он оказывается занят директорией «lost+found», поскольку именно она создаётся ОС сразу после создания ФС. При необходимости использования именно этого номера директорию можно переименовать или удалить.

Замечание *или вопрос «Как «захватить» «нужный» inode номер?»*

Попытаемся ответить: из пользовательского пространства создать файла с нужным номером inode нельзя, поскольку не существует команды, в которой был бы предусмотрен выбор номера. Но, косвенно («по блату»), можно попытаться. Поскольку в вопросе не оговорено как будет получен результат и какие права у пользователя (обычные или административные), то рассмотрим оба возможных варианта.

Естественно, от администратора ситуация выглядит легче, поскольку, последний может получить прямой доступ к файловой системе, через доступ к разделу (носителю) на котором она расположена. С другой стороны, существенен момент, в какой ФС будет создаваться файл и какой именно номер необходимо «заполучить»? Это важно, поскольку у двух разных ФС номера inode'ов неизбежно полностью или частично совпадут, и одно дело оперировать с ФС на отчуждаемом носителе, а другое – дело корневой ФС.

Номер inode также имеет значение, поскольку номера с 0 по 11 зарезервированы системой для разных целей [7], конкретно номер 1 используется (зарезервирован) для хранения списка дефектных (сбойных) блоков. Именно из-за этого корневая директория ФС (/) имеет номер 2. Без вмешательства в ФС через debugfs [8, 9] или напрямую в «сыром» режиме не обойтись.

Если же нужен конкретный номер, скажем 112233, то ситуации возможны следующие. Такого номера нет в принципе, поскольку ФС мала. Такой номер есть и занят файлом «X»/директорией «Y». Такой номер есть, но пока не занят. Во-втором случае, как найти объекты ФС по номеру inode'а в работе – рассмотрено, однако, многое ещё зависит от того, кто владелец файла/директории и какие есть права у пользователя. Если это файл и прав у пользователя достаточно, то можно создать жёсткую ссылку на него. А чтобы полноценно и единолично пользоваться содержимым, то ещё потребуется скопировать содержимое в новый файл, удалить другие ссылки на требуемый inode и создать их заново, но уже на inode нового файла-копии. С директорией сложнее, поскольку на неё создавать ссылки нельзя, поэтому придётся её удалить (предварительно переписав содержимое в созданную для этого новую). После этого будет третья ситуация, когда требуемый inode не занят.

Несмотря на то, что система многопользовательская, предположим, что мы единственные пользователи ФС и конкуренции по «захвату» нужного номера у нас не будет.

Единственный вариант решения от уровня пользователя – это создавать файлы, пока не будет занят нужный номер, например, банально циклом «for i in $(seq 1 *xxx*); do touch $i; done», указав вместо xxx число сравнимое с искомым «112233». Поскольку часть номеров в ФС уже занято, то необходимое количество создаваемых файлов будет меньше на число уже занятых и зарезервированных inode'ов. Создавать же файлов больше, чем требуется нет смысла. Ненужные файлы впоследствии можно стереть, а как найти среди них файл с нужным номером будет показано в работе.

Если требуемую ФС можно отмонтировать, то «захватить» требуемый inode намного проще с использованием утилиты debugfs. С помощью неё можно попытаться использовать и некоторые зарезервированные номера из первого десятка, но выйдет ли это «боком» предугадать сложно. Краткая последовательность действий, для номера *x* следующая: 0) отмонтировать раздел 1) открыть его в debugfs, указав опцию -w, чтобы можно было вносить изменения 2) clri <*x*> 3) seti <*x*> 4) sif <*x*> mode 0x81FF 5) ln <*x*> NewFileName 6) sif <*x*> links_count 1[337] 7) quit. После монтирования ФС файл NewFileName будет иметь номер inode'а равный *x* и его можно использовать как обычный файл. Номер *x* в командах выше (для debugfs) вводится в угловых скобках (знаках меньше и больше).

[337] Команда ln внутри debugfs не меняет это значение.

2.2. Обратите внимание на то, что есть такие имена директорий как «.» и «..», а команда ls по умолчанию не отображает имена, начинающиеся с «.». Для того чтобы их увидеть, используйте дополнительно ключ «-a».

2.3. Ответьте на вопросы: Совпадает ли номер inode у корневой директории (/) и у директории «..» внутри директории /etc? Совпадает ли номер inode у директории /etc при просмотре списка директорий внутри корня («сверху») и изнутри её самой («снизу»), где она представлена именем «.»?

3. Взаимодействие с объектами ФС по их номерам inode'ов довольно типично при решении ряда задач администрирования. Смоделируем эти ситуации.

3.1. Решим задачу поиска объекта(ов) ФС по номеру его inode'а. Для этого, исходя из предположения, что пользователь находится в своей домашней директории (~), создадим иерархию директорий, файл и несколько жёстких ссылок на него. Это будет файл, который в дальнейшем будет искаться.

```
~/dir1/
~/dir1/file1
~/dir1/hard_link_to_file1
~/dir1/dir2/
~/dir1/dir2/hard_link_to_file1
```

Используя команды «ls -li» и «stat *имя_объекта_ФС*» посмотрите информацию об объектах ФС, каждый раз до создания жёстких ссылок и после. В выводе обеих команд обратите внимание на параметр «число ссылок».

```
$ mkdir dir1
$ cd dir1
$ touch file1
$ mkdir dir2
```

посмотрите, что получилось

```
$ ln file1 hard_link_to_file1
```

посмотрите, что получилось

```
$ ln file1 dir2/hard_link_to_file1
```

посмотрите, что получилось

Имена file1 и hard_link_to_file1 равносильны, то есть «идентичны» с точки зрения доступа к содержимому и параметрам файла, заметили ли вы, что они ссылаются на один и тот же inode?

Допустим, он был «123456», осуществим поиск по нему. Для ускорения поиска выполним его внутри директории «~/dir1».

```
$ find ~/dir1 -inum 123456 -print
$ find ~/dir1 -inum 123456 -exec ls -l {} +
```

Команда find позволяет выполнять разные действия в отношении найденных объектов. С параметром «-print» осуществляется только вывод, с параметром «-exec» возможен запуск любой команды, которой вместо фигурных скобок «{}» передаётся найденное имя, знак «+» является неотъемлемой частью синтаксиса команды find.

3.2. Решим задачу удаления/переименования объектов ФС со «странными именами».

Как легко догадаться, выделенную выше команду «ls» с ключами легко заменить на любую, в том числе и на «rm» (или «rmdir»), в случае если требуется удаление объектов ФС.

Также вместо параметра «-print» у команды «find» возможно задать параметр «-delete» для удаления всех найденных объектов. Естественно, результат её выполнения будет зависеть от полномочий пользователя и установленных прав у найденных объектов.

Скорее вопрос заключается в том, откуда файлы (или директории) со странными именами могут появиться в ФС? При нормальных условиях работы ОС их быть не должно, но в ряде случаев, причиной этого могут быть заражённые различными вирусами внешние флеш-накопители, так и «слишком умные» пользователи системы, которые знают, что в большинстве ФС, используемых в ОС UNIX и Linux, в качестве имён объектов ФС разрешено использовать любые символы, кроме символа с кодом «0» (NULL) и разделителя «/».

То, что в именах могут встречаться символы имеющие одинаковое или сильно схожее начертание в латинице и кириллице, такие как «a», «o», «e», различные знаки: минус, дефис, тире и т. д. давно известно, но куда более интересно то, что в именах могут быть и более трудно отличимые при просмотре символы, например «пробел» (код 32 = 0x20) в конце имени, символы с кодами ASCII 1-31, скажем, тот же «backspace» (код 8 = 0x08, \b) и др., которые использовать при желании, набрав на клавиатуре, без ухищрений не получится. Также, поле для деятельности сильно расширилось с введением поддержки unicode в форматах UTF-8 и UTF-16.

Наиболее известен коррекционный символ «Right-To-Left Override» (*RLO*, UTF-16BE 0x202E, UTF-8 E2 80 AE, позволяющий в ряде программ с графическим интерфейсом (например, почтовых) видеть имя «pic*RLO*gpj.exe», где *RLO* – тот самый символ, как «picexe.jpg» [1, 2].

Также, достаточно много удивления (особенно после утомительного дня или бессонной ночи, когда сознание притуплено) может вызвать символ «/» (косая черта на право, UTF-8 E2 81 84) в имени, который сильно похож, а в ряде систем не отличим по начертанию от традиционного символа «/» («слэш»).

Рассмотрим варианты появления объектов ФС с нестандартными символами в имени и подходы для их удаления/переименования.

3.2.1. Выполните команды для создания файлов и директории.

```
$ mkdir aa$'\xe2\x81\x84'bb
$ touch pic$'\xe2\x80\xae'gpj.exe
$ touch "aa?b"
$ touch `printf "aa\bb"`
$ touch $(printf "aa\bb")
```

Обратите внимание на три вида разных кавычек в командах: одинарные «'», двойные «"» и обратные «`». Последняя команда идентична предпоследней. Она на 1 символ длиннее, но новичками воспринимается легче из-за другого синтаксиса (без обратных кавычек).

3.2.2. Запустите команду «`ls -l`» для просмотра содержимого текущей (рабочей) директории.

```
$ ls -l
-rw-rw-r--.  1 guest guest         0 Май 1 10:17 aa?b
-rw-rw-r--.  1 guest guest         0 Май 1 10:17 aa?b
drwxrwxr-x.  2 guest guest      4096 Май 1 10:17 aa⁄bb
-rw-rw-r--.  1 guest guest         0 Май 1 10:17 picgpj.exe
```

Как видите, отличить имена «aa?b» от «aa?b» не так просто, особенно если права доступа, владельцы, группы и времена создания совпадают. Попробуем удалить этот файл (эти файлы), для этого запустите команды

```
$ rm aa?b
$ ls -l
```

По незнанию можно подумать, что команда ls их тоже не различила, поскольку результат её работы выглядит именно так, однако, это утверждение не верно, поскольку знающие пользователи отметят, что в bash символ «?», без экранирования его знаком «\» (\?) или кавычками, является символом-джокером. Использование знака «?» в общем случае в команде rm без знания такой особенности, как и в других командах, может иметь свои последствия. Убедиться, что имена разные можно создав файлы заново (повторив команды 3 и 4 п.3.2.1), и просмотрев на номера их inode'ов. Они будут различны.

Сравните выводы команд «`ls -l`», «`ls -1`», «`ls -1|grep aa`».

Визуально вывод двух последних команд отличается, поскольку у команды ls есть опция «-q» позволяющая вместо непечатаемых символов в имени файла при выводе размещать знаки вопроса. Эта опция включена по умолчанию при выводе на терминал. При перенаправлении стандартного вывода команды ls на стандартный ввод команды grep ключ «-q» не добавляется, поскольку вывод команды ls не связан с терминалом. Если ключ добавить, то разница в отображении между двумя командами исчезнет.

```
$ ls -1 -q|grep aa
```

Однако, лучше всего увидеть разницу в выводе можно просматривая его как потока двоичных данных с помощью команды od.

```
$ ls -1|od -A n -t ax1
   a   a   ?   b  nl   a   a  bs   b  nl
  61  61  3f  62  0a  61  61  08  62  0a
$ ls -1|grep aa|od -A n -t ax1
   a   a   ?   b  nl   a   a  bs   b  nl
  61  61  3f  62  0a  61  61  08  62  0a
```

3.2.3. Для обращения к объектам ФС со странными именами можно использовать символ табуляции для дописывания конца имён. Рассмотрим как это работает. Создайте файл «pic*RLO*gpj.exe» из п.3.2.1, убедитесь что он создался, просмотром содержимого директории, удалите файл, набрав его имя побуквенно с клавиатуры.

```
$ touch pic$'\xe2\x80\xae'gpj.exe
$ ls -1
picexe.jpg
$ rm picgpj.exe
rm: невозможно удалить «picgpj.exe»: Нет такого файла или каталога
```

Вы получите ошибку, поскольку не смогли набрать коррекционный символ в имени. С точки зрения ОС и ФС получаются разные имена. Однако, если вы наберёте имя не до конца

```
$ rm pic
```

и после этого нажмёте клавишу «tab», то на экране вы увидите также

```
$ rm picgpj.exe
```

но команда отработает без ошибки.(– клавиатурный курсор)

3.2.4. Ещё одним способом набора «сложного» имени может быть выделение и копирование мышью. То есть, опять создаём файл, в предположении что он появился не по нашей воле, просматриваем результат и набираем на клавиатуре команду «rm ».

```
$ touch pic$'\xe2\x80\xae'gpj.exe
$ ls -1
picexe.jpg
$ rm
```

после этого с помощью курсора мыши выделяем имя файла в выводе команды ls и жмём на колесо вертикальной прокрутки у мыши (scroll). В месте расположения клавиатурного курсора появится выделенный мышью текст. Если после нажать «ввод», то команда удаления выполнится без ошибки. Естественно, в графической среде с мышью можно использовать и стандартные команды «копировать» и «вставить» контекстного меню, вызываемого правой кнопкой.

3.2.5. Наиболее простой способ оперирования со странными именами в bash, если известны коды их символов, – это посимвольная подстановка в создаваемое имя объекта ФС (параметр команды) необходимого числа конструкций вида $'\x*XX*' , или $'\x*XX*\x*XX*', или $'\x*XX*\x*XX*\x*XX*', и т. д., где *XX* – это шестнадцатеричные значения первого, второго и т.д. кодов символа в случае использования одно-, двух-, трёх- и много-байтового представления символов. Вместо шестнадцатеричных значений можно использовать восьмеричные без предшествующего «x». Естественно XX могут подставляться через переменные, а вся команда находится в теле цикла.

Для выяснения какой код у встретившегося символа используется команда od, как было написано выше.

Непривычным может показаться лишь то, что некоторые форматы кодировок используют разные длины для кодирования отдельных групп символов (планов), так называемые кодировки переменной длинны. Например при использовании популярного формата для Unicode – UTF-8, ответ на вопрос: «А сколько байт будет занимать строка из 5 символов?» отнюдь не тривиален при вхождении в строку символов из разных планов.

Узнать какая кодовая таблица используется у вас в консоли можно с помощью команды «`locale`».

4. Нюансы сохранения данных и использования блоков можно увидеть с помощью команды stat, либо сравнивая информацию от вывода команды `df` (статистику использования блоков) до создания файла и после, также можно использовать ключ `-s` (или `--size`) у команды `ls`.

Замечание. При использовании ключа -s размер выдаётся в блоках по 1024 байта слева от имени файла. Если установлена переменная окружения POSIXLY_CORRECT и не задана опция -k, то применяется размер блока 512. Если используется опция -k, то размеры выдаются в килобайтах (1 килобайт = 1024 байта).

Проверьте на практике один теоретический момент. В [3] в разделе «"Advanced" Ext2fs features» сказано, что для ускорения работы с символьными (мягкими) ссылка-

ми, в случае, если размер последних составляет менее 60 байт, то они хранятся в inode, при этом используется 0 блоков с данными.

Замечание. Аналогичную схему можно было бы применить и к маленьким файлам. В 1999-м году на сайте redhat.com из раздела Linux Kernel Hackers' Guide была опубликована ссылка на статью по разработке и использованию ФС ext2 [3], где Реми Кард с сотоварищами сообщил, что реализовать аналогичную возможность в отношении маленьких файлов планируется в ближайшем будущем.

Поддержка «встраиваемого хранения» (функция inline data) была добавлена в марте 2014-го, но на момент подготовки данной публикации (2018 год) в большинстве ОС Linux не реализована. Для её использованию в ФС ext4 необходима «WIP» версия утилит e2fsprogs (клон репозитария git). ФС должна быть создана командой mke2fs с опцией «-O inline_data». [4]

Последняя версия загрузчика GRUB (2.02 на момент подготовки публикации) не поддерживает данный функционал, однако, существует патч [5].

Разберёмся, откуда взялась цифра 60. Поскольку при «встраиваемом хранении» ни один блок с данными не используется, то невостребованными окажутся те байты структуры inode, которые ответственны за прямые и косвенные адреса блоков с данными. Из теории следует, что под адреса выделено: (12 прямых + 3 косвенных ссылки) * 4 байта = 60 байт. То есть, граница по использованию и не использованию блоков с данными должна проходить на переходе «60-61». С другой стороны, разработчиками для ссылок заявлено «меньше 60» («<60»), то есть граница должна быть на переходе «59-60». Попробуем понять откуда расхождение в 1 байт?

4.1. Создайте файлы с размерами от 0 до 61 байта. Посмотрите вывод команд «stat» и «ls -s» и оцените сколько каждый из них занимает блоков с данными.

Создавать файлы можно разными способами, вот лишь некоторые полезные команды, которые могут вам пригодиться:

`touch file0`	создать файл с именем file0 нулевого размера (если такой файл не существует)
`echo -n "1">>fileX`	дописать в файл с именем fileX 1 байт
`echo -n "12">>fileY`	дописать в файл с именем fileY 2 байта
`echo "1">>fileY`	дописать в файл с именем fileY 2 байта (цифру 1 + символ перевода строки)
`echo -n "1234567890">>fileZ`	дописать в файл с именем fileZ 10 байт

Также можно использовать либо цикл, дописывая в файл по одному однобайтовому символу нужное число раз, либо команду dd.

```
$ i=1; while [ $i -le 60 ] ;do echo -n "1" >>file60; let i=i+1;  done
$ dd if=/dev/zero of=file60 bs=1 count=60
```

Замечание. Обратите внимание, что если у вас не установлена ни одна из переменных: COMMAND_BLOCK_SIZE, BLOCK_SIZE или BLOCKSIZE, то команда stat по умолчанию будет считать блок равным 512 байт. То есть, при размере истинного логического блока в вашей ФС 1024 байта (в 2 раза больше), команда stat будет выдавать в 2 раза большее значение числа блоков. При размере 2048 байт – в 4 раза, 4096 – в 8 раз и т. д. (подробнее см. info coreutils 'stat invocation')

Обратили ли вы внимание, что файл размером 0 байт не занимает ни одного блока, что выглядит вполне логичным?

4.2. Расширьте поле эксперимента, добавив к исследованию файлы с размерами: 511, 512, 513, 1023, 1024, 1025, 2047, 2048, 2049, 4095, 4096, 4097, 8191, 8192 и 8193 байта.

4.3. Напишите, где (на границе каких значений объёма файла) по вашим наблюдениям происходит скачкообразное изменение количества блоков, используемых файлом для хранения данных по информации выводимой утилитой stat?

4.4. Ответьте на вопросы:

На сколько блоков сразу происходило скачкообразное изменение в предыдущем пункте? Можно ли по этой информации косвенно определить размер логического блока в используемой ФС? Если да, то как? Какой размер логического блока в ФС на которой вы создавали файлы в п.4.1 и п.4.2? Есть ли связь размера блока с «переходами» полученными в п.4.3?

4.5. Создайте файлы с именами длинной в 1, 2, 59, 60 и 61 байт.

```
$ touch 1
$ touch 12
$ touch 12345678901234567890123456789012345678901234567890123456-59
$ touch 123456789012345678901234567890123456789012345678901234567-60
$ touch 1234567890123456789012345678901234567890123456789012345678-61
```

4.6. Создайте файлы с именами длинной в 1, 2, 29, 30, 31 символ, используя буквы русского языка.

```
$ touch ы
$ touch ыы
$ touch абвгдеёжзийклмнопрстуфхцчшщьы
$ touch абвгдеёжзийклмнопрстуфхцчшщьыъ
$ touch абвгдеёжзийклмнопрстуфхцчшщьыъэ
```

4.7. Создайте мягкие ссылки на только что созданные файлы в п.4.5. и п 4.6.

```
$ ln -s 1 link-1
$ ln -s 12 link-12
$ ln -s 12345678901234567890123456789012…123456-59 link-123…6-59
$ ln -s 12345678901234567890123456789012…1234567-60 link-123…67-60
$ ln -s 12345678901234567890123456789012…12345678-61 link-123…678-61
$ ln -s ы link-ы
$ ln -s ыы link-ыы
$ ln -s абвгдеёжзийклмнопрстуфхцчшщьы link-абвгдеёжзийклмнопрстуфхцчшщьы
$ ln -s абвгдеёжзийклмнопрстуфхцчшщьыъ link-абвгдеёжзийклмнопрстуфхцчшщьыъ
$ ln -s абвгдеёжзийклмнопрстуфхцчшщьыъэ link-абвгдеёжзийклмнопрстуфхцчшщьыъэ
```

(Обратите внимание на пропущенные символы, они заменены знаком «…».)

Посмотрите вывод команд «`stat`» и «`ls -s`» и оцените сколько каждая из созданных ссылок занимает блоков с данными. Уловили ли вы разницу между п. 4.5 и п. 4.6 (между размерами в символах и в байтах)? Разница заметна поскольку имена закодированы с помощью кодировки в формате UTF-8, у которой латиница с цифрами и кириллица принадлежат к разным планам, использующим 1 и 2 байта для кодирования символов, соответственно.

Часть 2. Взаимодействие через утилиты ФС

5. По сравнению со стандартными командами, более информативный вывод по inode'ам можно получить используя утилиты, осуществляющие прямой доступ к ФС: debugfs, lde (linux disk editor), dumpe2fs и др.

Для экспериментов создадим новую ФС внутри файла и посмотрим как выглядят inode'ы в ней с точки зрения разных команд. Для того чтобы быть ближе к реальности будем сравнивать то, что мы видим с выводом стандартных команд взаимодействия с ФС, рассмотренных выше. Поскольку файловые системы ext2 и ext4 несколько отличаются, то файлов будет два: «disk100ext2» и «disk100ext4».

5.1. Создайте пустые файлы размером по 102 400 000 байт каждый и файловые системы внутри них:

```
$ dd if=/dev/zero of=disk100ext2 bs=1024 count=100000
100000+0 записей считано
100000+0 записей написано
 скопировано 102400000 байт (102 MB), 0,202756 с, 505 MB/с
$ dd if=/dev/zero of=disk100ext4 bs=1024 count=100000
...
$ mkfs -t ext2 -I 128 disk100ext2
...
$ mkfs -t ext4 -I 256 disk100ext4
...
```

Ключ «`-I`» задаёт размеры inode'ов 128 и 256 байт, поскольку, если размер не задать вручную, то для обеих ФС из файла mke2fs.conf будут взяты значения по умолчанию, как для ФС типа «small» – 128 байт, и, в этом случае, будет продемонстрировать улучшения, появившиеся в ext4 в сравнении с ext2.

Замечание. У ФС ext2-ext4 существует большое число всевозможных функциональных опций [6], влияющих на их работу и внутреннее устройство. Большая часть из них задаётся в момент создания ФС. Некоторые опции по-умолчанию оказываются включенными, другие, наоборот, выключенными. Вручную задать или отключить опцию можно с помощью ключа -O команды «mkfs», указав в качестве его параметра через запятую нужные значения. В случае если опция (возможность) должна быть отключена перед ней ставится знак «^», например:

```
$ mkfs -t ext2 -O ^ext_attr disk100ext2
```

До версии 1.43 дополнительные ключи «-O» игнорируются. Опция ext_attr указывает хранить расширенные атрибуты в inode. Таким образом, место хранения расширенных атрибутов в ФС, таких как ACL и SELinux (устанавливаемых через команды setfacl и chcon, соответственно), напрямую зависит от её состояния.

5.2. Оцените параметры созданных ФС командами

```
$ dumpe2fs disk100ext2
...
$ dumpe2fs disk100ext4
```

5.3. Ответьте на вопросы:

Какой размер блока оказался установлен по умолчанию?
В каких блоках размещена копия суперблока?
Сколько inode'ов было создано?
Сколько блоков входит в группу блоков?

5.4. Примонтируйте ФС, созданные внутри файлов disk100ext2 и disk100ext4, в директории /mnt/ext2 и /mnt/ext4, соответственно.

```
$ sudo mount -o loop disk100ext2 /mnt/ext2
$ sudo mount -o loop disk100ext4 /mnt/ext4
```

Оцените получившийся результат из ОС, определите число занятых inode'ов в ФС командами:

```
$ mount
...
/home/guest/disk100ext2 on /mnt/ext2 type ext2 (rw,loop=/dev/loop0)
/home/guest/disk100ext4 on /mnt/ext4 type ext4 (rw,loop=/dev/loop1)
$ df -i
Filesystem              Inodes IUsed  IFree IUse% Mounted on
…
/home/guest/disk100ext2 25064    11  25053    1% /mnt/ext2
/home/guest/disk100ext4 24960    11  24949    1% /mnt/ext4
```

5.5. Создайте файлы размером в 0 байт и несколько байт в обеих ФС.

```
$ touch /mnt/ext2/file0
$ touch /mnt/ext4/file0
$ echo hello > /mnt/ext2/file1
$ echo hello > /mnt/ext4/file1
```

Замечание. Создание файлов, как и внесение изменений в них, при отсутствии нагрузки в системе, должно происходить сразу. Однако, если при запуске последующих команд, осуществляющих прямой доступ к файлам disk100ext2 и disk100ext4 с ФС внутри, запись в ФС почему-то «не прошла» (команды «не видят» последних изменений, которые должны быть), следует выполнить сброс буферов командой /bin/sync, либо произвести отмонтирование ФС командами:

```
$ sudo umount /mnt/ext2
$ sudo umount /mnt/ext4
```

при выполнение которых сброс произойдёт автоматически.

5.6. Исследуйте особенности хранения файлов в ФС с помощью команд ls и stat:

```
$ ls -lisa /mnt/ext2
$ ls -lisa /mnt/ext4
$ stat /mnt/ext2/file0
$ stat /mnt/ext2/file1
$ stat /mnt/ext4/file0
$ stat /mnt/ext4/file1
```

Подумайте, почему по результатам вывода команды ls файл file0 нулевого размера в ФС ext2 занимает 1 блок с данными, а file1 целых два ? (Ответ на данный вопрос, с пояснениями, будет дан в третьей части.)

Заметили ли вы, что точность хранимых меток времени отличается от типа ФС? В случае с ФС ext2 точность указывается до секунды, а далее пишутся нули «для совместимости» формата вывода, ФС ext4 хранит значение секунд с точностью до 9 знаков после запятой.

```
Access: 2017-05-04 13:44:04.000000000 +0300
Access: 2017-05-04 13:44:08.024475329 +0300
```

Также заметьте, что в inode хранятся три метки времени, доступные пользовательским программам (на самом деле всего их четыре, подробнее об этом см. ниже):

- «Access» (atime, access time) – время последнего доступа к объекту ФС. Эта метка также есть и у директорий. Как следствие, ОС меняет её метки каждый раз как пользователь просматривает список файлов в ней. Таким образом, в ОС с большим числом операций с ФС, относительно быстрые операции чтения начинают смешиваться с более медленными операциями записи, от чего падает

производительность. С целью повышения производительности рекомендуется монтировать ФС с опцией «noatime» («nodiratime»).

- «Modify» (mtime, modification time) – время последней модификации содержимого файла (директории).
- «Change» (ctime, change time) – время последнего изменения inode (метаданных, например командой chmod) объекта ФС.

Определите номера inode'ов у исследуемых файлов. Напомним, что поскольку номера с 0 по 11 зарезервированы ФС для разных целей [7], созданные файлы будут иметь номера 12 и 13.

Также, объекты разных ФС могут иметь одинаковые номера, что вы могли успешно только что наблюдать, однако ОС не испытывает затруднений, работая с разными файлами в едином дереве VFS (Virtual File System), поскольку параметр «Device», выводимый командой «stat» у двух разных файлов с одинаковым именем «file0» отличается.

5.7. Исследуйте те же самые файлы с помощью программы linux disk editor:

```
$ lde -i 12 disk100ext2
Device "disk100ext2" is mounted, be careful
User requested autodetect filesystem. Checking device . . .
Found ext2fs on device.
------------------------------------------------------------------------------
INODE: 12     (0x0000000C)
-rw-rw-r--  user     user             0 Thu May  4 13:44:04 2017
TYPE:                   regular file
LINKS:                  1
MODEFLAGS.MODE:         010.0664
SIZE:                   0
BLOCK COUNT:            2
UID:                    00500 (user)
GID:                    00500 (user)
ACCESS TIME:            Thu May  4 13:44:04 2017
CREATION TIME:          Thu May  4 13:44:04 2017
MODIFICATION TIME:      Thu May  4 13:44:04 2017
DELETION TIME:          Thu Jan  1 03:00:00 1970
DIRECT BLOCKS:
INDIRECT BLOCK:
DOUBLE INDIRECT BLOCK:
TRIPLE INDIRECT BLOCK:
$ lde -i 13 disk100ext2
...
$ lde -i 12 disk100ext4
$ lde -i 13 disk100ext4
```

Обратите внимание, что при выводе содержимого 12-го и 13-го inode'ов ФС ext2, нигде не указаны имена файлов (file0 и file1, соответственно), то есть, эта информация в них действительно не хранится. Также, можно заметить и ранее упомянутую четвёртую метку времени – «deletion time» (время удаления). Отметьте время, установленное в ней: «начало эпохи UNIX» + «смещение часового пояса». Исследование inode'а после удаления файла мы произведём чуть позже.

Посмотрите сколько блоков занимает каждый из файлов.

Обратите внимание, что у файла с именем file1 (inode=12 ФС=ext2) в параметре «DIRECT BLOCKS» содержится один номер блока, а в параметре «BLOCK COUNT» указано значение 4. Можете объяснить это расхождение?

Замечание. Несмотря на то, что первые 128 байт inode'ов любых размеров одинаковы, можно отметить, что linux disk editor корректно не работает, когда размер inode'ов в ФС больше 128 байт (в нашем случае для ФС ext4 он равен 256). Как следствие, вывод последних двух команд содержит «мусор». Программа lde давно не обновлялась и неправильно определяет смещение для считывания нужных 128 байт, если в ФС используется размер inode'ов больше 128 байт. Если же создать ФС ext2-4 с размером inode равным 128 байт, то команда работает корректно. Помимо этого, у программы также возникают проблемы при отображении занимаемых блоков в ФС ext2. Для получения истинных результатов лучше использовать debugfs или «сырое» чтение с диска.

5.7. Исследуйте также с помощью команды lde содержимое соседних, заведомо не занятых inode'ов на ФС ext2, например 14 и 15.

```
$ lde -i 14 disk100ext2
```

5.8. Поскольку программа lde является отдельной программой (не входящей в основной репозитарий ОС), да и к тому же не всегда корректно показывает реальное положение дел, исследуйте те же самые файлы (созданные в п.5.5) с помощью программы debugfs, её внутренними командами «stat *имя_файла*» и «stat <X>», где X – номер inode'a. Выход осуществляется командой «quit».

```
$ debugfs disk100ext2
debugfs 1.41.12 (17-May-2010)
debugfs:  stat file0
Inode: 12   Type: regular    Mode:  0664   Flags: 0x0
Generation: 703348796    Version: 0x00000000  User:   500   Group:   500
Size: 0   File ACL: 516    Directory ACL: 0   Links: 1   Blockcount: 2
Fragment:  Address: 0    Number: 0    Size: 0
ctime: 0x5933e474 -- Thu May  4 13:44:04 2017
atime: 0x5933e474 -- Thu May  4 13:44:04 2017
mtime: 0x5933e474 -- Thu May  4 13:44:04 2017
BLOCKS:
debugfs:  stat <12>
...
```

Как видно из параметра «File ACL: 516», «нашлись ранее потерянные» блоки. Блок № 516 используется как расширение inode'a для хранения расширенных атрибутов (метаданных). Выполнив команду «imap file0» или «imap <12>», можно узнать в каком блоке находится inode, данная информация может быть полезна при «сыром» чтении данных из ФС, если вы не планируете самостоятельно выполнить вычисление смещений.

```
debugfs: imap file0
Inode 12 is part of block group 0
   located at block 292, offset 0x0180
```

Замечание. debugfs – это замечательная утилита, на изучение которой стоит потратить время. С одной стороны, она имеет функционал «редактора диска», где требуется понимание строения ФС, с другой, для выполнения ручного восстановления данных (чтение содержимого удалённых файлов, исправление ошибок возникших в результате сбоев) у неё есть много других полезных команд-аналогов, пришедших из bash'a: pwd, cd, rm, stat, cat. Схожесть имеется не только в названиях, но и в синтаксисе запуска, что сильно упрощает работу.

Например, с помощью команды «cat» можно просматривать содержимое файлов по имени или по номеру inode'a.

```
debugfs:  cat <13>
hello
```

Повторите это действие в отношении других файлов.

Выводить подобным образом двоичные файлы на консоль можно, но неудобно (из-за наличия в них непечатаемых и управляющих символов). При необходимости, содержимое файла можно вывести не на консоль, а сохранить во внешний файл в реальной ФС. Выполните сохранение файла file1 (inode № 13) во внешний файл file1-out и просмотрите его.

```
debugfs:  dump <13> file1-out
debugfs:  quit
$ cat file1-out
hello
```

Часть 3. «Сырое» чтение данных из ФС

Если задаться вопросами: А что содержится в блоке данных № 516, указанном в параметре File ACL на предыдущем шаге? Да и, вообще, можно ли «посмотреть» на метаданные любого файла побайтно в подробностях? То, без сырого чтения данных из ФС не обойтись, поскольку, если ставший в своё время классикой коммерческий Norton Disk Editor позволял делать это в отношении ФС FAT, то на сегодня найти аналогичную по удобности и функционалу GNU'шную программу для ФС ext2-4 проблематично.

При наличии у пользователя соответствующих прав доступа на помощь приходят команды:

`od` – для просмотра содержимого файлов и дисков (блочных устройств);

`dd` – для их поблочного (побайтного) копирования.

С таким набором фактически можно прочитать (просмотреть) и скопировать что угодно, откуда угодно и куда угодно, если речь идёт о файлах, ФС и физических дисках, доступных в системе как псевдофайлы блочных устройств. Зная из теории необходимое смещение [7], и обладая достаточным количеством времени, можно работать с блоками размером в 1, 2 или *n* байт. Например, попытаемся просмотреть как хранятся расширенные атрибуты у файлов и выясним и что же находится в блоке данных № 516, указанном в inode файла file1.

6. Просмотрите содержимое файла file1 (в ФС ext2) с помощью команды od, выполнив нижеследующие шаги.

6.1. Попытаться определить смещение inode № 13, можно и глядя на рис.1., но задача эта непростая. Во-первых, нужно знать размеры отображённых там элементов. Во-вторых, резервные блоки были упомянуты вскользь, они не отображены на рисунке и ничего не было сказано про их количество. В третьих, задача расчёта смещения inode'а не является целью данной работы. Поэтому проще воспользоваться информацией из вывода команды dumpe2fs – для широкого взгляда на ФС и debugfs – для рассмотрения интересующего нас места в более мелком масштабе.

```
$ dumpe2fs disk100ext2
…
Group 0: (Blocks 1-8192)
  Primary superblock at 1, Group descriptors at 2-2
  Reserved GDT blocks at 3-258
  Block bitmap at 259 (+258), Inode bitmap at 260 (+259)
  Inode table at 261-501 (+260)
  7675 free blocks, 1915 free inodes, 2 directories
  Free blocks: 518-8192
  Free inodes: 14-1928
…
```

```
debugfs:  imap file1
Inode 13 is part of block group 0
   located at block 262, offset 0x0200
```

Считаем inode и определим какой дополнительный блок используется для хранения расширенных прав (списков доступа (ACL)), а также какие блоки храня данные файла.

Предварительно преобразуем $0x0200_{16} = 512_{10}$ и вычислим 1024 * 262 + 512 = 268800.

```
$ od -t x1 disk100ext2 -j 268800 -A d -N 128
0268800 b4 81 f4 01 06 00 00 00 87 03 35 59 87 03 35 59
0268816 87 03 35 59 00 00 00 00 f4 01 01 00 04 00 00 00
0268832 00 00 00 00 00 00 00 00 05 02 00 00 00 00 00 00
0268848 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*
0268896 00 00 00 00 17 5a 1e 2b 04 02 00 00 00 00 00 00
0268912 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

в находим таблицу, описывающую строение inode'a (в [7] или на стр. 463 см. «*Таблица 5.13. Уточнённая структура inode, записанная в виде справочной таблицы*»), откуда определяем смещение для получения байтов с номером блока, где хранятся расширенные атрибуты.

0x68	__le32	i_file_acl_lo	Lower 32-bits of extended attribute block… (Младшие 32 бита адреса блока, где хранятся расширенные атрибуты...)

$0x68_{16}=104_{10}$, данные типа «__le32» занимают 4 байта. Все адреса в нашем примере 32-битные, поэтому старших 32 битов от адреса просто нет (считаем равными нулю). Смещение будет 268800 + 104 = 268904. Находим эти байты в выводе выше и переводим из формата «little endian unsigned int» в десятичный вид, либо выведем только эти байты, а также изменим формат их отображения.

```
$ od -t x1 disk100ext2 -j 268904 -A d -N 4
0268904 04 02 00 00
$ od -t d disk100ext2 -j 268904 -A d -N 4
0268904         516
```

Итого, получился номер блока 516 – тот же самый номер, что выводила команда stat утилиты debugfs. Смещение блока будет 516 * 1024 = 528384. Читаем содержимое блока, пытаясь при этом отобразить данные в виде ASCII символов.

```
$ od -t ax1 disk100ext2 -j 528384 -A d -N 1024
0528384  00  00  02  ea  02  00  00  00  01  00  00  00  cf  47  b1  6d
0528400  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0528416 bel ack   `  etx nul nul nul nul  sp nul nul nul   O   G   1   m
         07  06  e0  03  00  00  00  00  20  00  00  00  cf  47  b1  6d
0528432   s   e   l   i   n   u   x nul nul nul nul nul nul nul nul nul
         73  65  6c  69  6e  75  78  00  00  00  00  00  00  00  00  00
0528448 nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul
         00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
*
0529376   u   n   c   o   n   f   i   n   e   d   _   u   :   o   b   j
         75  6e  63  6f  6e  66  69  6e  65  64  5f  75  3a  6f  62  6a
0529392   e   c   t   _   r   :   f   i   l   e   _   t   :   s   0 nul
         65  63  74  5f  72  3a  66  69  6c  65  5f  74  3a  73  30  00
```

В первых четырёх байтах видно «магическое число» 0xEA020000 [7], хранимое в формате «little endian», которое ставится в начале блока как отличительный идентификатор, сообщающий о том, что далее хранятся расширенные атрибуты. В последних строчках (выделено цветом) явно просматривается типичный контекст безопасности SELinux. В подтверждение догадки проверим какой контекст был выставлен у файла /mnt/ext2/file1.

```
$ ls -Z /mnt/ext2/file1
-rw-rw-r--. user user unconfined_u:object_r:file_t:s0  /mnt/ext2/file1
```

Замечание 1. Первоначально блок расширенных атрибутов использовался для хранения как контекстов SElinux'a, так и списков контроля доступа (ACL). Поэтому, если просмотреть содержимое блока, а затем у файла поменять записи списка контроля доступа (ACL-записи), то после, при повторном просмотре, можно будет заметить изменения. Например (про условии, что ACL-блок сохранит свой номер и, как следствие, смещение):

```
$ getfacl /mnt/ext2/file1
$ od -t ax1 disk100ext2 -j 528384 -A d -N 1024
…
$ setfacl -m u:root:r-- /mnt/ext2/file1
$ getfacl /mnt/ext2/file1
$ od -t ax1 disk100ext2 -j 528384 -A d -N 1024
…
```

Единственное, наглядности, как с контекстом SELinux, не получится, поскольку данные не хранятся в виде текста из ASCII символов. Также, во избежание ошибки

```
setfacl: /mnt/ext2/file1: Неподдерживаемая операция
```

ФС должна быть примонтирована с опциями user_xattr и acl, например:

```
$ sudo /bin/mount -o loop,user_xattr,acl disk100ext2 /mnt/ext2
```

По умолчанию и, соответственно, при монтировании ФС в п. 5.4, эти опции отключены.

При просмотре списка файлов в директории командой «ls -l», после обычных прав доступа у файлов, имеющих ACL-записи, будет виден знак «+».

```
$ ls -l /mnt/ext2/file1
-rw-rw-r--+ 1 user user 6 May  4 13:44 /mnt/ext2/file1
```

Замечание 2. Также к расширенным атрибутам относятся и способности (см. «man 7 capabilities»). Вот, пример установки и считывания способностей

```
$ sudo setcap cap_sys_admin+ep /mnt/ext2/file1
$ getcap /mnt/ext2/file1
```

(Данный пример «включает» способности, использующиеся для предоставления разрешения перехвата данных с сетевых интерфейсов программам, запускаемым не от суперпользователя. В данный момент они бессмысленны, поскольку file1 – не программа.)

После установки какой-либо способности, при просмотре содержимого блока расширенных атрибутов в виде ASCII текста вы сможете увидеть слово «capability», а разобрав соседние двоичные значения по формату можно выяснить детали.

6.2. Определим номер блока с данными у того же файла file1 и прочитаем его. Из Рис. 3 видно, что адрес 0-го блока с данными будет храниться по смещению 40 байт от начала inode'a, то есть смещение от начала диска составит значение полученное на шаге 6.1 – 268800 + 40 = 268840. Считываем 4 байта и переводим их должным образом в десятичный вид.

```
$ od -t x1 disk100ext2 -j 268840 -A d -N 4
0268840 05 02 00 00
$ od -t d disk100ext2 -j 268840 -A d -N 4
0268840         517
```

Получился номер блока с данными 517 – тот же самый номер, что выводила команда stat утилиты debugfs. Смещение блока будет 517 * 1024 = 529408. Читаем содержимое блока (например первые 10 символов), отображая данные в виде ASCII символов.

```
$ od -t a disk100ext2 -j 529408 -A d -N 1024
0529408   h   e   l   l   o  nl nul nul nul ...
```

6.3. Исследуйте, какое значение хранится в байтах отведённых для адреса 1-го блока с данными. Из рисунка на странице 462 видно, что это значение следует сразу же за байтами отведёнными для адреса 1-го блока с данными. То есть, к смещению можно просто добавить 4. Также, если интересно, то можно посчитать адреса смещения соседних блоков и «заглянуть» в них, задавшись вопросом: пустые они или нет? Конечно, наличие нулевых символов по всему блоку ни о чём не говорит и правильным будет определять занят блок или нет будет обратившись к битовой карте блоков, но исследование её выходит за рамки темы данной работы.

7. Вычислите момент (два соседние значения размера файла) когда будет переход к хранению информации с использованием косвенной адресации. Обратившись к рисунку с inode на стр.462 можно увидеть, что это произойдёт когда будет израсходован объём 12 блоков данных, адресуемых 12-ю адресами прямой адресации.

7.1. Используя команды и знания применявшиеся ранее, проверьте на практике полученное в п.7 значение.

8. Для ФС с блоком данных размером 1024 байта выведите формулу и произведите теоретический расчёт размера файла, начиная с которого будет использоваться двойная косвенная адресация.

8.1. Повторите расчёт для ФС с блоком данных размером 4096 байт.

9. Просмотрите дату и время, а затем удалите файл /mnt/ext2/file1.

```
$ date -R
Thu, 04 May 2017 14:51:32 +0300
$ rm /mnt/ext2/file1
```

Поскольку в обычных условиях данные inode'a после удаления файла не стираются, то по ранее определённым его номеру и вычисленному на его основе смещению, произведите просмотр содержимого inode'a в части deletion time. Время хранится в формате «__le32» в секундах с начала эпохи UNIX по смещению 0x14, подробнее об этом и других полях см. [7].

Соответствует время удаления файла тому, что было записано в его inode, когда файл существовал?

10. Повторите пункты заданий частей 2 и 3 для ФС ext4, оцените что изменилось.

11. Верните систему к первоначальному состоянию. Исходя из того, что по ходу выполнения заданий все файлы и директории создавались в директории ~/dir1 – удалите её со всем содержимым.

```
$ rm -rf ~/dir1
```

Если вы создавали объекты ФС в других местах – удалите их. Будет ещё один повод повторно пробежаться по выполненным заданиям и уяснить что же вы проделали и какие объекты создавали.

Стереть историю команд в bash'e можно командой

```
$ history -c
```

При этом будут удалены все команды из истории, в том числе и те, что ранее существовали в файле ~/bash_history до проведения данной работы.

Ответ на вопрос, заданный в п.5.6: пустой файл по результатам вывода программы stat занимает 2 блока на диске потому как данные отображаются порциями по 512, а при размере блока 1024, порций будет две и отобразиться число 2, а не 1. Вторая часть вопроса: не важно один блок или два, почему файл занимает не ноль? Дело в том, что в системе по умолчанию «включен SELinux», который автоматически добавляет файловым объектам контекст безопасности (вы его могли увидеть в п.6.1). Поскольку в ФС ext2 мы специально задали inode размером 128 байт при её создании, то хранить его негде, кроме как в отдельном блоке. Создай мы ФС с inode размером 256 или больше байт, данные, скорее всего, уместились бы в нём и дополнительный блок не потребовался бы. Можете это проверить. Также и при размере inode'а в 128 байт возможно не использовать дополнительный блок. Для этого надо в корне отключить selinux (чтобы команда getenforce выводила «Disabled»). Сделать это можно тремя путями: либо установить в файле /etc/selinux/config параметр SELINUX=disabled и перезагрузить систему, либо передать ядру через загрузчик GRUB параметр selinux=0, либо перекомпилировать ядро без поддержки SELinux и загрузиться с него. У старых файлов контекст безопасности никуда не исчезнет, а создаваемые новые будут без него и тогда файл размера 0 будет занимать 0 блоков на диске.

Часть 4. Исследуем временные метки индексного дескриптора

12. Войдите в систему (повторно), запустите терминал или переключитесь на работу в текстовой консоли и войдите в систему.

13. При необходимости смените директорию на домашнюю или /tmp, то есть такую, где будет разрешено создание новых файлов.

14. Проверьте текущую дату и время в системе

```
$ date
Ср мар 18 12:04:00 MSK 2020
$ date -R
Web, 18 Mar 2020 12:04:02 +0300
```

15. Создайте два пустых файла disk100ext2 и disk100ext4 размерами ≈ по 100 МБ.

```
$ dd if=/dev/zero of=disk100ext2 bs=1024 count=100000
...
$ dd if=/dev/zero of=disk100ext4 bs=1024 count=100000
...
```

(О значениях параметров команд см. man или комментарии в первой части работы.)

16. Попытайтесь просмотреть параметры ещё не созданных ФС внутри этих файлов командами

```
$ dumpe2fs disk100ext2
...
$ dumpe2fs disk100ext4
```

Что выводит dumpe2fs, если ей «подсунуть» файл без созданной в нём ФС? Сочетается ли ваш ответ с тем, что написано ниже?

Программа попытается считать суперблок исходя из определённого для него смещения от начала ФС (А вы помните какое оно?), а затем проверить его контрольную сумму и сигнатуру (0xEF53). Естественно что файлы, заполненный от начала до конца нулями, не проходят эти поверки.

Замечание. В некоторых дистрибутивах у обычного пользователя переменная окружения PATH не содержит директорию /sbin, где находится указанная программа. Результат очевиден – сообщение об ошибке: команда не найдена. Совсем не то, что следует получить на шаге 5 и других. Выход из ситуации: указывать абсолютное имя программы от корня (/) или изменить содержимое переменной окружения.

17. Определите номер uid (а за одно и gid, поскольку он нужен для соблюдения формата в последующих командах) пользователя от которого работаете в системе.

```
$ id
uid=500(altlinux) gid=500(altlinux)
группы=500(altlinux),10(wheel),14(uucp),19(proc),80(cdwriter),81(audio),83(r
adio),100(users),471(fuse),481(netadmin),498(xgrp),499(scanner)
```

Указанные номера нужны для параметра -E root_owner=X:Y команды создающей ФС, чтобы эти номера были прописаны в качестве владельцев корневой директории, чтобы в последующем, при монтировании ФС, пользователь с указанными идентификаторами мог сразу же с ней работать (создавать файлы и т.д.). Если этого не сделать, то владельцем будет root (uid=0) и дискреционное разграничение доступа в ОС не даст доступа к объектам внутри созданной вами же ФС без изменения прав или владельца у корневой директории примонтированной файловой системы.

18. Создайте внутри файлов две новые ФС с inode'ами размером в 128 и 256 байт.

```
$ mkfs -t ext2 -I 128 -O ^ext_attr -E root_owner=500:500 disk100ext2
...
$ mkfs -t ext4 -I 256 -E root_owner=500:500 disk100ext4
...
```

Обратите внимание, что в случае отсутствия путей к /sbin, поскольку команда mkfs есть обёртка для запуска нескольких программ, использование полного имени программы есть не полное решение. В этом случае лучше напрямую обращаться к программам mkfs.ext2 и mkfs.ext4 как

```
$ mkfs.ext2 -I 128 -O ^ext_attr -E root_owner=500:500 disk100ext2
...
$ mkfs.ext4 -I 256 -E root_owner=500:500 disk100ext4
...
```

19. Оцените параметры созданных ФС с помощью утилиты dumpe2fs и ответьте на вопросы относительно обеих ФС, некоторые из ответов вам в последующем пригодятся:

19.1. Какой размер блока оказался установлен по умолчанию у ФС?

19.2. Какого размера создались inode'ы?

19.3. Сколько групп блоков было создано?

19.4. Сколько inode'ов всего было создано во всей ФС?

19.5. Сколько inode'ов входит в группу блоков?

19.6. Сколько блоков в одной группе блоков отведено для хранения inode'ов?

19.7. Как отличаются списки поддерживаемых функций (features) у обеих ФС?

19.8. Определите и выпишите в каких блоках находятся битовые карты занятости inode'ов (для всей ФС)?

19.9. Определите и выпишите в каких блоках находятся таблицы inode'ов (для всей ФС)?

19.10. Определите номера inode'ов хранимых в первой группе блоков (№ первого, № последнего, общее число).

19.11. Определите номера inode'ов хранимых во второй группе блоков (№ первого, № последнего, общее число).

19.12. Какой номер inode указан как первый? Совпадает ли это значение с первым свободным номером inode в первой группе блоков? Если нет, то почему?

19.13. Постарайтесь вспомнить из описания ФС ext4 в первой части работы: куда были распределены первые по порядку номера inode'ов в первой группе блоков? Т.е. почему нумерация свободных inode'ов начинается не с нуля (или единицы)?

20. Примонтируйте ФС, созданные внутри файлов disk100ext2 и disk100ext4, в директории /mnt/ext2 и /mnt/ext4 (созданные при подготовке стенда), соответственно.

```
# mount -o loop disk100ext2 /mnt/ext2
# mount -o loop disk100ext4 /mnt/ext4
```

Оцените получившийся результат:

```
$ mount
...
/home/.../disk100ext2 on /mnt/ext2 type ext2 (rw,realtime,errors=continue,
user_xattr,acl,user=500)
/home/.../disk100ext4 on /mnt/ext4 type ext4 (rw,realtime,user=500)
```

21. Посмотрите в системе текущие дату и время и запишите их. Они послужат вам грубой ориентировочной оценкой. Созданные на следующем шаге файлы будут иметь относительно близкие к текущему моменту времени метки.

```
$ date -R
```

22. Создайте файлы со словами test1, test2, test3, test4 (или любым другим разнородным содержимым) в обеих ФС.

```
$ echo "test1">/mnt/ext2/file1.txt
$ echo "test2">/mnt/ext2/file2.txt
$ echo "test3">/mnt/ext4/file3.txt
$ echo "test4">/mnt/ext4/file4.txt
```

23. Проверьте получившийся результат с помощью команд ls и stat, выпишите все полученные с помощью этих команд метки времени, а также номера inode'ов. Эти значения вам пригодятся. У каких файлов вы увидели метки времени с точностью до наносекунд? По каким признакам вы это поняли?

Для систематизации информации по файлам, занесите получаемые от команд данные в таблицу ЛР2.1.

Таблица ЛР2.1. Сводная таблица с информацией о файлах

имя файла	№ inode	удалён	atime	ctime	mtime	dtime	crtime	ФС
file1.txt	12	нет	2020-03-18 12:14:12. 000000000 +0300	2020-03-18 12:14:12. 000000000 +0300	2020-03-18 12:14:12. 000000000 +0300	-	-	ext2
…								

24. Удалите файлы /mnt/ext2/file2.txt и /mnt/ext4/file4.txt, после чего просмотрите и запишите текущее время.

```
$ rm /mnt/ext2/file2.txt /mnt/ext4/file4.txt
…
$ date -R
```

25. Отмонтируйте ФС ext2 и ext4 во избежание образования системных ошибок и случайного изменения ФС со стороны операционной системы, а также чтобы вы могли беспрепятственно сами экспериментировать и вносить необходимые вам правки. (Напомним, если вы находитесь в какой-либо директории примонтированной ФС, то отмонтировать её нельзя.)

```
# umount /mnt/ext2
# umount /mnt/ext4
```

26. На всякий случай сделайте резервные копии ваших ФС, то есть файлов disk100ext2 и disk100ext4.

```
$ cp disk100ext2 disk100ext2.copy
$ cp disk100ext4 disk100ext4.copy
```

27. Перейдите к исследованию образов ФС с помощью debugfs сначала для ФС ext2, а затем для ФС ext4.

```
$ debugfs disk100ext2
debugfs 1.44.6 (5-Mar-2019)
debugfs: ...
debugfs: quit
$ debugfs disk100ext4
...
```

27.1. Оцените информацию, получаемую с помощью команд ls и lsdel для обеих ФС и сравните её с информацией полученной на шагах 12 и 13. К сожалению, команда lsdel работает лишь в ФС ext2 (см. man debugfs). Как вы думаете, почему так происходит? Могут ли влиять на работу lsdel размер и степень фрагментации удалённых файлов?

```
debugfs:  ls
 2  (12) .    2  (12) ..    11  (20) lost+found    12  (980) file1.txt
debugfs:  lsdel
 Inode  Owner  Mode    Size      Blocks   Time deleted
    13    500 100644      6      1/     1 Sat Apr 18 19:18:23 2020
1 deleted inodes found.
```

27.2. Просмотрите содержание индексных дескрипторов для всех четырёх файлов командами «stat *имя_файлового_объекта*» или «stat *<номер>*», а также просмотрите какой-нибудь не занятый inode.

```
debugfs:  stat <15>
Inode: 15   Type: bad type    Mode:  0000   Flags: 0x0
Generation: 0    Version: 0x00000000
User:     0   Group:     0   Size: 0  File ACL: 0    Directory ACL: 0
Links: 0   Blockcount: 0  Fragment:  Address: 0    Number: 0    Size: 0
ctime: 0x00000000 -- Thu Jan  1 03:00:00 1970
atime: 0x00000000 -- Thu Jan  1 03:00:00 1970
mtime: 0x00000000 -- Thu Jan  1 03:00:00 1970
Size of extra inode fields: 0
BLOCKS:
...
debugfs:  stat file1.txt
Inode: 12   Type: regular    Mode:  0644   Flags: 0x0
Generation: 3833897581    Version: 0x00000000
User:   500   Group:   500   Size: 6  File ACL: 0    Directory ACL: 0
Links: 1   Blockcount: 2  Fragment:  Address: 0    Number: 0    Size: 0
ctime: 0x5e9b2754 -- Sat Apr 18 19:14:12 2020
atime: 0x5e9b2754 -- Sat Apr 18 19:14:12 2020
mtime: 0x5e9b2754 -- Sat Apr 18 19:14:12 2020
BLOCKS: (0):4097
TOTAL: 1

debugfs:  stat <13>
Inode: 13   Type: regular    Mode:  0644   Flags: 0x0
Generation: 3833897582    Version: 0x00000000
User:   500   Group:   500   Size: 6  File ACL: 0    Directory ACL: 0
Links: 0   Blockcount: 2  Fragment:  Address: 0    Number: 0    Size: 0
ctime: 0x5e9b284f -- Sat Apr 18 19:18:23 2020
```

```
atime: 0x5e9b2758 -- Sat Apr 18 19:14:16 2020
mtime: 0x5e9b2758 -- Sat Apr 18 19:14:16 2020
dtime: 0x5e9b284f -- Sat Apr 18 19:18:23 2020
BLOCKS: (0):4098
TOTAL: 1
...
debugfs:  stat <13>
Inode: 13   Type: regular    Mode:  0644   Flags: 0x80000
Generation: 1013115637    Version: 0x00000000:00000001
User:   500   Group:   500   Size: 0  File ACL: 0    Directory ACL: 0
Links: 0   Blockcount: 0  Fragment:  Address: 0    Number: 0    Size: 0
 ctime: 0x5e9b2850:ed7e30d8 -- Sat Apr 18 19:18:24 2020
 atime: 0x5e9b275f:230d1a8c -- Sat Apr 18 19:14:23 2020
 mtime: 0x5e9b2850:ed7e30d8 -- Sat Apr 18 19:18:24 2020
crtime: 0x5e9b275f:230d1a8c -- Sat Apr 18 19:14:23 2020
dtime: 0x5e9b2850 -- Sat Apr 18 19:18:24 2020
Size of extra inode fields: 32
EXTENTS:
```

28. По информации из предыдущего пункта запишите и заполните таблицу, аналогичную таблице ЛР2.2.

Таблица ЛР2.2. Пример сводной таблицы с информацией о файлах, полученной с помощью debugfs

имя файла (при наличии)	№ inode	удалён	atime	ctime	mtime	dtime	crtime	ФС
file1.txt	12	нет	0x5e9b2754	…		?	–	ext2
нет	13	да				0x5e9b2850		ext2
file3.txt	12	нет	0x5e9b275f:230d1a8c			–		ext4
нет	13	да						ext4

29. Произведите перевод временных меток из таблицы 3 в «человекоудобный вид» по примеру таблиц 4 и 5. Для удобства анализа дат можно использовать команду cal для вывода календаря в консоли, например «cal 2020», подробнее см. «man cal».

Таблица ЛР2.3. Преобразование простых меток времени к удобному для восприятия человеком виду

Описание поля	**Значение**
исходная метка	0x5e9b2754
16 система счисления	5e9b2754
10 система счисления (число секунд)	1587226452
схема перевода секунд	/60 секунд /60 минут /24 часа /≈ 365 дней
число дней с начала эпохи	18370,6765...
число лет с начала эпохи (+1970)	1970 + 50,33… ≈ 2020 год
конечная дата	Сб апр 18 16:14:12 GMT 2020

Таблица ЛР2.4. Преобразование расширенных меток времени к удобному для восприятия человеком виду

Описание поля	Значение
исходная метка	0x5e9b275f:230d1a8c
поле 1 в 16cc поле 2 в 16cc	5e9b275f 230d1a8c
поле 1 в 10cc (число секунд)	1587226463
дата на основе поля 1	Сб апр 18 16:14:23 GMT 2020
поле 2 в 2cc (биты эпохи выделены)	0010 0011 0000 1101 0001 1010 1000 1100
биты эпохи	00
влияние битов эпохи / смещение	нет
число для расчёта наносекунд в 2cc	0010 0011 0000 1101 0001 1010 1000 11
число для расчёта наносекунд в 10cc	147015331
конечная дата	Сб апр 18 16:14:23.147015331 GMT 2020

30. Для номеров inode'ов, упомянутых в предыдущих нескольких шагах, определите их смещение и номера блоков, в которых они находятся, для чего заполните таблицу ЛР2.5.

Таблица ЛР2.5. Информация о расположении inode'ов

Описание поля	Значение поля для объектов ФС			
	file1.txt	~~file2.txt~~	file3.txt	~~file4.txt~~
Номер искомого inode				
Файловая система (ext2 или ext4)				
Размер блока ФС				
Размер inode				
Номер группы блоков содержащей искомый inode				
Номер блока содержащего таблицу дескрипторов групп блоков				
Размер одной записи в таблице дескрипторов групп в байтах (32 или 64)				
Поля в записи таблицы дескрипторов групп отвечающие за адрес блока содержащего начало таблицы inode				
Значение поля (полей) содержащих адрес блока в котором располагается начало таблицы inode				
Смещение inode от начала таблицы в размерах inode				
Смещение inode от начала таблицы в байтах				
Номер блока где находится смещённый inode				
Смещение inode от начала блока в котором он находится				
Смещение inode от начала группы блоков в байтах				
Смещение inode от начала ФС в байтах				

30.1. При заполнении воспользуйтесь следующей информацией (предположительно вам уже известной) о строении ФС. Напомним, что цель работы не найти номера inode'ов удалённых файлов и восстановить последние, а исследовать временные метки у заранее определённых индексных дескрипторов, хотя несомненно, понимание одного может помочь в другом и, наоборот.

Нумерация inode'ов начинается с 1. Наличие или отсутствие зарезервированных inode'ов и их предназначение никак не влияет на алгоритм поиска блока содержащего inode с заданным номером. Размер блока ФС (см. шаг 19.1), наоборот, существенен для алгоритма поиска. Размер искомого inode'а (см. шаг 19.2) также имеет значение. Разного размера inode'ы в рамках одной ФС не бывают. Все inode'ы расположены в таблицах inode'ов которые, в свою очередь, расположены и имеются в каждой группе блоков.

30.1.1. Таким образом, первый шаг поиска inode'а – это определение той группы блоков к которой он принадлежит (см. вывод dumpe2fs (шаг 19), а также подумайте насколько эта информация вам сейчас полезна, а также сколько потратиться сил на её реконструкцию в случае повреждения суперблока ФС и его копий).

Определение группы блоков для inode'а производится по формуле

$$номер\_группы\_блоков = взять\_целую\_часть\left(\frac{номер\_inode - 1}{inode'ов\_в\_группе}\right)$$

, где значение числа *inode'ов_в_группе* берётся из суперблока (см. шаг 19.5).

30.1.2. Второй шаг – определение расположения таблицы inode'ов в этой группе, а точнее – её начального блока. Для этого придётся взглянуть в таблицу дескрипторов групп блоков (далее, – таблицу BGD, от Block Group Descriptors) которая содержит описание каждой группы блоков файловой системы. Каждой группе блоков соответствует одна запись в таблице размером 32 или 64 байта. В каждой записи содержится информация о расположении внутри каждой группы блоков важных для неё частей, как адреса начальных блоков, статистическая информация, и др. Общее число записей в таблице равно количеству групп блоков ФС.

Таким образом, для таблицы BGD потребуется определить:
– (30.1.2.1.) адрес нахождения таблицы BGD (или любой из её копий) в ФС;
– (30.1.2.2.) размер одной записи в таблице BGD (32 или 64 бита);
– (30.1.2.3.) смещение искомой записи в таблице BGD (1 группа блоков = 1 запись);
– (30.1.2.4.) формат записи (расположения нужной информации внутри одной записи).

Ответ на первый вопрос прост: таблица BGD находится сразу за суперблоком. Важно лишь помнить, что суперблок имеет размер 1024 байта и располагается со смещением тоже 1024 байта (в отличие от своих копий), а если размер блока <=1024, то запросто может оказаться и не в нулевом блоке. Также важно помнить, что номера блоков начинаются с нуля, а также в реальной жизни (когда ФС не в файле, а на диске) может быть ещё таблица разделов (MBR или GPT) и номера блоков обычно не соответствуют физическим адресам блоков вашего носителя данных.

Размер записи в таблице BGD по умолчанию равен 32 байтам. Это стандартное, совместимое со всеми работающими с ФС программами значение, пришедшее из ext2. Однако, оно может быть другим, если в ФС установлен 64-битный флаг поддержки нестандартных (несовместимых с ext2) значений. В этом случае (т. е. скорее всего уже в ext4) следует руководствоваться значением поля суперблока с именем «Group descriptor size» или «s_desc_size». Обычно это поле указывает на размер записи в 64 (байта).

Смещение нужной записи в таблице BGD определяется номером_группы_блоков (см п. 30.1.1), умноженном на размер одной записи. Который описывался в предыдущем абзаце.

Например для записи группы № 0 размером 32 байта получим смещение 1024+1024+0*32=2048. Считаем значения записи № 0 и сохраним результат в файл tableBGD_1_record.bin.

```
$ dd if=disk100ext2 of=tableBGD_1_record.bin skip=2048 bs=1 count=32
...
```

Форматы записей в таблице BGD естественно для размеров 32 и 64 разные, но есть частичная обратная совместимость (см. Таблицы ЛР2.6 и ЛР2.7).

Таблица ЛР2.6. Фрагмент структуры одной записи таблицы дескрипторов групп размером 32 байта

Смещение	Начальный байт	Конечный байт	Формат, размер	Описание поля
0x0	0	3	__le32	Адрес (начального) блока с битовой картой занятости информационных блоков
0x4	4	7	__le32	Адрес (начального) блока с битовой картой занятости inode'ов
0x8	8	11	__le32	Адрес начального блока таблицы inode'ов
0xC	12	13	__le16	Число не занятых блоков в группе
0xE	14	15	__le16	Число не занятых inode'ов в группе
0x10	16	17	__le16	Число директорий в группе

Таблица ЛР2.7. Фрагмент структуры одной записи таблицы дескрипторов групп размером 64 байта

Смещение	Начальный байт	Конечный байт	Формат, размер	Описание поля
0x0	0	3	__le32	bg_block_bitmap_lo Младшие 32 бита адреса (начального) блока с битовой картой занятости информационных блоков
0x4	4	7	__le32	bg_inode_bitmap_lo Младшие 32 бита адреса (начального) блока содержащего битовую карту занятости inode'ов
0x8	8	11	__le32	bg_inode_table_lo Младшие 32 бита адреса блока содержащего начало таблицы inode'ов
0xC	12	13	__le16	bg_free_blocks_count_lo Младшие 16 бит числа не занятых блоков в группе
...				
0x24	36	39	__le32	bg_inode_bitmap_hi Старшие 32 бита адреса (начального) блока содержащего битовую карту занятости inode'ов
0x28	40	43	__le32	bg_inode_table_hi Старшие 32 бита адреса блока содержащего начало таблицы inode'ов
...				

Искомые нами поля выделены цветом. Для нашего размера берём данные таблицы ЛР2.6: смещение 0x8, размер 4 байта = sizeof(для __le32). Выводим результат командой od.

```
$ od -tx1 tableBGD_1_record.bin
0000000 03 01 00 00 04 01 00 00 05 01 00 00 fc 1d 7c 07
0000020 02 00 04 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Наше значение будет $(05\ 01\ 00\ 00)_{LE} = (00\ 00\ 01\ 05)_{16} = 0x00000105 = 261_{10}$. То есть получилось что начало таблицы inode'ов для 0 группы лежит в 261 блоке. Как видно, информация согласуется с тем, что выдаёт dumpe2fs:

```
$ dumpe2fs disk100ext2
...
Group 0: (Blocks 1-8192)
...
  Inode table at 261-501 (+260)
```

30.1.3. Третий шаг (зная размер блока ФС, номер блока с началом таблицы inode'ов в нужной группе блоков, номер искомого inode'а, и его размер, номер первого inode'а в данной группе блоков) – это вычисление смещения для искомого inode'а от начала таблицы inode'ов в определённой на первом шаге группе блоков. А также определение адреса блока на который приходится смещение (начало inode'а) и смещение в этом блоке (от его начала), чтобы дальше можно было высчитать смещение от начала всей ФС до нужного inode'а одним числом. На языке формул будет следующее:

смещение_inode_от_начала_ФС = смещение_до_нужной_группы_блоков +
+ смещение_в_группе_блоков_до_начала_расположения_таблицы_inode'ов +
+ смещение_от_начала_таблицы_inode'ов.

*смещение_до_нужной_группы_блоков = №_группы_блоков * размер_группы_блоков*
*размер_группы_блоков = размер блока * число_блоков_в_группе_блоков*

смещение_в_группе_блоков_до_начала_расположения_таблицы_inode'ов =
*=(см._адрес_начала_табл._inode'ов_в_соотв._записи_таблицы_BGD) * размер_блока*

смещение_от_начала_таблицы_inode'ов =
*= (№_искомого_inode − номер_первого_inode_в_данной_группе) * размер_inode'а*

$$№\_блока\_с\_inode = \left\lfloor \frac{смещение\_inode\_от\_начала\_ФС}{размер\_блока} \right\rfloor$$

смещение_inode_внутри_блока = смещение_inode_от_начала_ФС % размер_блока,
где % – операция получения остатка от деления, $\lfloor \ldots \rfloor$ – взятие целой части.

31. Используя утилиту dd и данные из таблицы 6, сохраните исследуемые inode'ы в файлы с именами file1_inode.dd, file2_inode.dd, file3_inode.dd и file4_inode.dd, соответственно.

```
$ dd if=disk100ext2 of=file1_inode.dd skip=смещение_inode'а bs=1
count=размер_inode'а
```

32. Если ваша версия debugfs поддерживает такую возможность, то используя данные о файлах и номерах inode'ов из таблицы 6, сохраните последние в файлы с именами file1_inode_dump, file2_inode_dump, file3_inode_dump и file4_inode_dump, соответственно.

```
$ /sbin/debugfs -R 'inode_dump file1.txt' disk100ext2 >file1_inode_dump
debugfs 1.44.6 (5-Mar-2019) – дистрибутив Альт Рабочая станция 9
$ /sbin/debugfs -R 'inode_dump <13>' disk100ext2 >file2_inode_dump
```

```
...
```

К сожалению просматривать или сохранять в «сыром двоичном виде» индексные дескрипторы ни одна версия debugfs не умеет. Кроме того, не все версии e2fsprogs имеют команду inode_dump (idump, id), которая позволяет просмотреть inode в так называемом «hex» режиме, в частности (версия 1.42.9-16 из CentOS 7) не имеет такого функционала.

32.1. Если вы счастливый обладатель «правильно» работающей версии debugfs, сравните полученные с помощью неё данные с данными сырого чтения ФС в п.31.

```
$ cat file1_inode_dump
0000  a481 f401 0600 0000 5427 9b5e 5427 9b5e  ........T'.^T'.^
0020  5427 9b5e 0000 0000 f401 0100 0200 0000  T'.^............
0040  0000 0000 0000 0000 0110 0000 0000 0000  ................
0060  0000 0000 0000 0000 0000 0000 0000 0000  ................
*
0140  0000 0000 6da2 84e4 0000 0000 0000 0000  ....m...........
0160  0000 0000 0000 0000 0000 0000 0000 0000  ................
```

Посчитайте смещение как 261*1024+(13−1)*128 = 268800 байт, подставьте в параметры команды, посмотрите сырые данные с диска, сравните.

```
$ dd if=disk100ext2 of=file1_inode.dd skip=268800 bs=1 count=128
...
$ od -tx1 file1_inode.dd
0000000 a4 81 f4 01 06 00 00 00 54 27 9b 5e 54 27 9b 5e
0000020 54 27 9b 5e 00 00 00 00 f4 01 01 00 02 00 00 00
0000040 00 00 00 00 00 00 00 00 01 10 00 00 00 00 00 00
0000060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*
0000140 00 00 00 00 6d a2 84 e4 00 00 00 00 00 00 00 00
0000160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000200
```

Если данные двух способов получения информации из inode побайтно совпали (что можно увидеть глазами и убедиться), значит, все расчёты были верными и программа debugfs работает в соответствии с теорией.

Замечание. Несмотря на вывод данных в немного отличающихся форматах, процесс сравнения без труда автоматизируется с помощью команд cmp, head и cut.

```
$ cat file1_inode_dump |cut -c1-5,7-45|head -n7>cmp_file1
$ od -tx1 file1_inode.dd|cut -c1,5-10,12-16,18-22,24-28,30-34,36-40,42-
46,48-52,54,55|head -n7>cmp_file2
$ cmp -l cmp_file1 cmp_file2
$ echo $?
0
```

Опытные читатели наверняка предложат ещё не один универсальный вариант сравнения с использованием sed, awk, perl и других программ.

33. Просмотрите файлы file1_inode.dd, file2_inode.dd, file3_inode.dd, file4_inode.dd

```
$ od -tx1 file1_inode.dd
0000000 a4 81 f4 01 06 00 00 00 54 27 9b 5e 54 27 9b 5e
0000020 54 27 9b 5e 00 00 00 00 f4 01 01 00 02 00 00 00
0000040 00 00 00 00 00 00 00 00 01 10 00 00 00 00 00 00
0000060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*
0000140 00 00 00 00 6d a2 84 e4 00 00 00 00 00 00 00 00
0000160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000200
```

Используя теоретические знания о формате меток времени и справочную таблицу 5.13 (на стр.463)определите смещения для полей содержащих информацию о метках времени и найдите значения этих полей. Произведите выборочный пересчёт двоичных значений в «человеческий формат» времени. Сравните полученный результаты с таблицами ЛР2.1 и ЛР2.2.

Дополнительные задания повышенной сложности

34*. Придумайте существующий момент времени, например 01 февраля 2000 года 04 часа 05 минут 06 секунд во временной зоне +0300.

34.1.* Переведите это время в секунды формата эпохи Unix (например 949367106), а затем в формат, подходящий для хранения в inode, например 0x38963142.

34.2.* Создайте двоичный файл patch1 с указанным значением (цифры хранятся не как текст, а как число секунд в формате __le32).

```
$ echo -n $'\x42\x31\x96\x38'>patch1
$ od -tx1 patch1
0000000 42 31 96 38
0000004
```

34.3.* Примонтируйте ФС disk100ext2 и создайте в ней файл file5.txt (см. п.9-п.11).

34.4.* Отмонтируйте ФС disk100ext2.

34.5.* Определите № inode файла file5.txt, а также смещение от начала файла disk100ext2 по которому хранится метка времени ictime.

34.6.* Используя команду dd и параметр seek произведите замену 4 байтов по найденному вами смещению в файле disk100ext2 на байты из файла patch1. Возможно придётся набраться терпения и использовать команду несколько раз, может пригодятся и резервные копии ФС, созданные на шаге 15.

34.7.* Примонтируйте ФС disk100ext2 и проверьте правильно ли поменялось время у файла file5.txt.

34.8.* Предложенный алгоритм и работа заведомо придуманы так чтобы они были выполнимыми, но не самым оптимальным образом, чтобы у выполняющего регулярно возникали мысли, как в анекдоте: «обработать напильником до придания нужной формы». В связи с чем вопрос, где вы видите «подводные камни» и как предлагаете исправить ситуацию? Если на каких-то шагах что-то не получалось, то почему и что надо изменить?

35.* Придумайте существующий момент времени из 2300 года с учётом точности до 1 наносекунды. Повторите по аналогии шаги из п.23, но в отношении ФС disk100ext4.

36.* Можете ли вы придумать несуществующее время, но которое можно записать во временные поля ФС? Как будет себя вести: ФС после монтирования, команды ls и stat, если их запустить? Найдёт ли ошибку утилита fsck.ext2 (или fsck.ext4) и сможет ли исправить?

Замечание 1. Иногда для задания необходимого значения отдельным временным меткам файлов и директорий (например при выпуске в релиз какой-то версии многофайловой программы в заданную дату) могут использоваться более простые способы – команды touch и tar с соот-

ветствующими ключами. Интереса ради просмотрите временные метки в архивах у исходных файлов ядер ОС Linux.

Замечание 2. С теперяшними вашими знаниями, возможно вам покажется полезной программа Active@ Disk Editor (Freeware версия под ОС Linux есть на [10]).

Заключение

В результате выполнения заданий работы и подготовки к ним вы освежили в памяти устройство отдельных частей ФС ext2-ext4 и имели возможность более близко познакомиться с такой её частью как inode, если не сказать иначе, – «взглянуть внутрь». В ходе выполнения практических занятий вы должны были глубже понять роль индексного дескриптора с точки зрения организации хранения файлов и связанной с ними информации (метаданных), также вы имели возможность оценить различные количественные параметры за счёт использования стандартных программ и «сырого чтения», научились взаимодействовать с файлами, имеющими «необычные» символы в имени. Вы должны были «увидеть» где и как хранятся расширенные атрибуты файлов, а также что значит хранение данных (например, коротких ссылок) внутри inode. Узнали о количестве временных меток у файловых объектов в ФС ext2/4, их форматах, диапазонах, точных местах хранения. Возможно, полученные знания окажутся полезными в будущем не только при восстановления информации с повреждённых ФС, но и в других областях.

Ссылки к лабораторной работе № 2

1. Можно ли верить коду в редакторе? bi-directional текст https://habrahabr.ru/post/252813/.

2. Можно ли верить своим глазам? (Unicode в именах файлов) https://habrahabr.ru/post/126198/.

3. Design and Implementation of the Second Extended Filesystem http://e2fsprogs.sourceforge.net/ext2intro.html.

4. How to use the new Ext4 Inline Data feature? (storing data directly in the inode) https://unix.stackexchange.com/questions/197633/how-to-use-the-new-ext4-inline-data-feature-storing-data-directly-in-the-inod/197806.

5. grub ext4 inline data support patch http://git.savannah.gnu.org/cgit/grub.git/commit/?h=andrey/ext4_inline_data&id=d4af83fc600d02dcfedae457353efbce8726b8d9.

6. Об опциях ФС ext2-4 http://man7.org/linux/man-pages/man5/ext4.5.html.

7. Ext4 Disk Layout https://ext4.wiki.kernel.org/index.php/Ext4_Disk_Layout.

8. debugfs Command Examples https://www.cs.montana.edu/courses/309/topics/4-disks/debugfs_example.html.

9. Linux Ext2fs Undeletion mini-HOWTO http://www.faqs.org/docs/Linux-mini/Ext2fs-Undeletion.html.

10. Программа правки диска Active@ Disk Editor – Freeware Hex Viewer & Hex Editor for raw sectors/clusters, http://www.disk-editor.org.

Глава 6. Объединение компьютеров в сéти

Помните притчу Л.Н.Толстова «Веник»?
или поговорку «Один в поле не воин»?
(*народная мудрость, появилась задолго до изобретения компьютера и сети интернет*)

6.1. Зачем объединяться?

Если постараться коротко ответить на этот вопрос с точки зрения фирм или организаций – то для совместного использования ресурсов, целью которого является предоставление доступа к программам, оборудованию и данным для любого пользователя сети независимо от физического расположения затребованного ресурса и пользователя.

Например, в небольшом офисе или компьютерном классе это может быть общий сетевой принтер, использование которого обойдётся дешевле. В автоматизированных терминалах по продаже билетов на поезда или в кино общей будет информация из базы данных по наличию мест.

Независимо от решаемых задач систему, объединяющую несколько компьютеров, проще всего представить как совокупность одной или более баз данных с информацией [338] и некоторого количества пользователей, которым удалённо предоставляется информация. Данные, как правило, хранятся на мощном компьютере, называемом «сервером» [339]. Довольно часто сервер располагается в отдельном помещении и обслуживается системным администратором. С другой стороны соединения компьютеры подключающихся пользователей могут быть менее мощными, они идентифицируются в сети как «клиенты», могут в большом количестве располагаться как в одном помещении (например, компьютерный класс), так и иметь удалённый доступ к информации и программам, хранящимся на сервере.

В литературе «клиентом» часто называют и пользователя клиентского компьютера. В большинстве случаев из контекста должно быть понятно, когда речь идёт о компьютере, а когда о человеке.

Пример объединения клиента с сервером в сеть можно увидеть в [5, стр. 19] или на рис. 6.1. Обратите внимание, часто сеть изображается в виде «серого облака», без уточнения деталей. Когда же необходимо увидеть нюансы, при обсуждении того или иного аспекта, внутри серого облака обычно появляются различные детали.

[338] Здесь имеются в виду не реляционные СУБД, которые встречаются повсеместно, а понимание баз данных в общем смысле этих слов. Так, под это понятие подойдёт и простой файловый сервер.

[339] В общем случае в роли сервера может выступать любой компьютер, даже ноутбук, планшет и телефон. Другой вопрос, насколько эти устройства будут эффективны в роли сервера, особенно при обработке большого числа поступающих запросов.

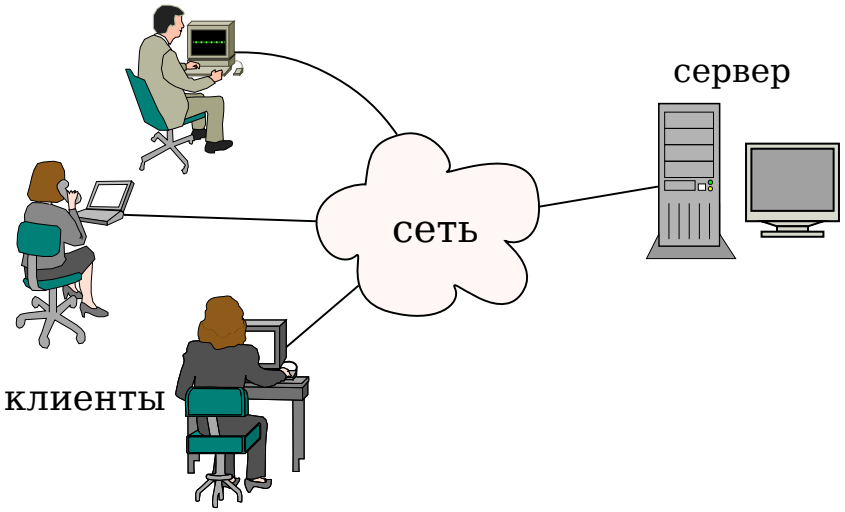


Рисунок 6.1. Сеть, состоящая из трёх клиентов и одного сервера

Такая система называется ***клиент-серверной моделью***. Она используется очень широко и зачастую является основой построения всех сетей. Самая популярная реализация – это любое веб-приложение, в котором сервер формирует и выдаёт клиенту веб-страницы, сформированные на основе данных сервера в ответ на запросы клиентов, которые, кстати, могут только одним своим существованием также обновлять базу данных сервера [340].

Между сервером и клиентом происходит межпроцессное взаимодействие через сокеты (интернет сокеты, UNIX-сокеты). По сути создаётся абстракция, когда один процесс совершает запись в сокет также как в локальный файл, а на другой стороне (на другом компьютере, а может и на этом же) другой процесс производит чтение из сокета, как из локального файла в файловой системе. Таким образом программисты не умеющие работать с сетью, но умеющие работать с файлами за счёт абстракции сокета быстро создают сетевые приложения не зацикливаясь на нюансах работы сети.

Использованию сокетов в учебнике специально посвящена лабораторная работа № 4 (см. стр.711).

Клиент-серверный подход в общем случае не учитывает расстояния, например клиент и сервер могут находиться в одном здании и принадлежать одному владельцу, а могут быть расположены далеко друг от друга, в разных странах [341]. Когда пользователь получает доступ к интернет-сайту, работает такая же модель: веб-сервер играет роль серверной машины, а пользовательский компьютер – клиентской. В большинстве случаев один сервер одновременно занимается обслуживанием большого числа клиентов [342].

[340] Например, счётчик числа посещений конкретной веб-страницы.

[341] Как следствие, находясь физически в разных странах, на отдельные части такой сети будут действовать разные местные законы. В такой сети, объединяющей компьютеры из разных стран, запросто может быть ситуация, что, например, получить доступ к какой-то информации из Германии можно, а из России нельзя, или наоборот.

[342] Когда клиентов, обращающихся к серверу, становится очень много, то чаще всего понятие «сервер» становится абстрактным, его функции «разделяют» на несколько физических компьютеров, а потоки обращений делятся между ними.

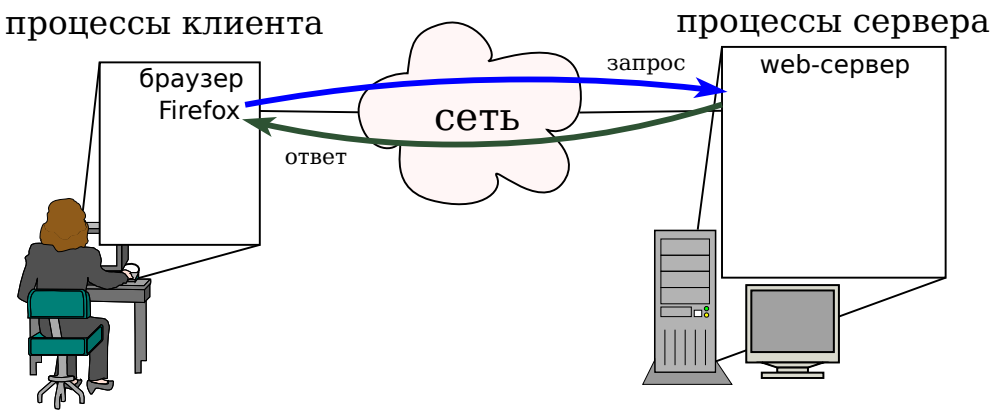


Рисунок 6.2. Процесс клиента шлёт запрос процессу сервера и получает ответ. Клиент-серверный подход: запрос – ответ

Вопрос: почему на рис. 6.2. изображён веб-сервер, а не какой-либо другой?

Ответ: потому, как это наиболее общий случай, удобный для рассмотрения. По протоколу HTTP методами GET и POST можно передавать любые данные (в параметрах запроса или как содержимое файла, аудио, видео, текст и прочее) на сервер, а также получать в обратном направлении от веб-сервера к клиенту ответы, например также двоичным файлом или html-страницей. Умение веб-браузерами поддерживать технологию AJAX позволяет принимать «ответы» не сразу, а «с задержкой», что лишь добавляет гибкости, хотя несомненно, что работа локальных сетей и интернета одним лишь протоколом HTTP не ограничивается.

Если мы посмотрим на «клиент-серверную» модель чуть пристальнее, то станет очевидно, что в работе сети можно всегда выделить два процесса (то есть работающие программы): серверный и клиентский. Обмен информацией чаще всего происходит в следующем порядке. Клиент посылает запрос серверу через сеть и начинает ожидать ответа. При принятии запроса сервер выполняет определённые действия или ищет запрашиваемые данные, затем отсылает ответ [5]. Всё это показано на рис. 6.2.

Вторым вариантом ответа, скорее, будет – для общения. Людям свойственно общаться, а компьютеры расширили разнообразие, добавив в этот процесс лишь новые способы. Если компьютеры между собой не связаны – общения не получится. Сеть – это замечательная коммуникационная среда как для частных лиц, так и для организаций различного вида собственности. Практически в любой фирме от десяти человек найдётся хотя бы один компьютер (мы не берём в расчёт личные устройства пользователей), умеющий принимать и отправлять электронную почту (e-mail, email). Электронная почта – довольно популярное средство общения уже как десятилетия. В одних случаях электронная почта не нужна, в других же фирмы ею активно пользуются. Порой руководители просто вынуждены рассылать своим подчинённым электронные послания, так как это удобно и просто, при этом может запрещаться личное общение и вестись контроль служебной переписки. На базе идеи электронной почты обмениваться текстовыми сообщениями существуют различные сервисы текстового обмена (сервисы мгновенного обмена сообщениями – Instant Messenger'ы и чаты, от англ. chat). А из веб-технологий и форумов родились различные социальные сети.

Телефонные звонки между служащими могут передаваться по компьютерной сети вместо телефонной, особенно если офисы расположены в разных городах, а тем более странах. Эту технологию называют IP-телефонией, или VoIP (Voice over IP). Гарнитура, состоящая из микрофона и динамика, может подключаться к компьютеру пользователя через звуковую карту либо иметь исполнение в виде привычной телефонной трубки. В целом это отличный способ экономить на телефонных счётах, подробнее об этом будет рассказано в разделе 6.4.4.4 «IP-телефония».

Понятно, что если можно передавать звук, хорошо бы ещё добавить видео... чтобы при общении собеседники могли видеть друг друга. Если рассказчик один, а слушателей много, то общение уже превращается в видеоконференцию или вебинар.

Передача видео (графической информации) может использоваться, не только чтобы видеть лица друг друга, но и, например, для совместного доступа к рабочему столу и окнам запускаемых программ. Например, так работают многие технические поддержки, используя программы RAdmin, PC Anywhere, TeamViewer и др., а также протокол VNC. Понемногу входят в жизнь амбициозные формы удалённой координации, такие как телемедицина, например осмотр удалённого пациента. Аналогично может осматриваться и различного рода техника, например авиа и прочая.

Третий вариант ответа – это электронная коммерция (e-commerce), которая быстро растёт в последние годы. Авиа- и ж/д компании, книжные магазины, гостиницы и другие продавцы обнаружили, что многим клиентам удобно совершать покупки удалённо, из дома или с работы. Появилась возможность оплачивать покупки, находясь в одном месте, а доставка будет производиться по другому адресу, например доставка новой плиты тёще или букета цветов жене домой. С появлением планшетов и «умных» телефонов география мест совершения удалённых покупок ещё расширилась. Много компаний перенесли каталоги своих товаров в сеть и принимают online-заказы. Наиболее продвинутые также создают приложения для «мобильных ОС» типа Android, iOS, BlackberryOS и др.

Производители автомобилей, самолётов, компьютеров и другой техники закупают комплектующие и детали у огромного числа поставщиков, а затем занимаются сборкой конечной продукции. С помощью компьютерных сетей процесс составления и отправки заказов можно автоматизировать. Более того, заказы могут формироваться строго в соответствии с производственными нуждами, что позволяет резко повысить эффективность [5]. (Подробнее см. раздел 6.6 «*Электронная коммерция*».)

6.1.1. Как и для чего используют сети частные лица?

Зачем люди устанавливают компьютеры у себя дома? Зачем покупают ноутбуки, планшеты и телефоны, стоимость которых порой превосходит их месячную, а то и квартальную зарплату? Постараемся ответить на эти вопросы больше с технической точки зрения, нежели психологической и социальной.

Смеем предположить, что несколько десятилетий назад для многих пользователей изначальной целью было редактирование текстов и электронные игры. Компьютер был лучше печатной машинки и позволял так или иначе развлечься. Старшее поколение наверняка помнит такой текстовый редактор, как Lexicon. Позднее к играм добавился факс-модем, позволявший отравлять и получать файлы без дискет, быть в курсе тем различных эхоконференций, получать и отправлять электронную почту и FidoNet-сообщения.

Позже, вероятнее всего, самой большой причиной покупки домашнего компьютера стала возможность получения доступа в интернет с помощью него.

Сейчас же многие бытовые электронные устройства, такие как электронные книги, различные приёмники (радио-, ТВ- и спутниковых передач), игровые приставки, mp3/mp4-плееры, принтеры, медиацентры и прочее, производятся с использование микропроцессоров, то есть на базе ЭВМ, что позволяет запросто встраивать в указанные устройства сетевые проводные и беспроводные интерфейсы.

Домашние сети широко используются для развлечения, включая слушание, просмотр и создание музыки, фотографий и видео [5].

Доступ к интернету предоставляет домашним пользователям связь с удалёнными компьютерами. Как и в случае с компаниями, домашние пользователи могут получить доступ к информации, общаться с другими людьми, купить продукты и услуги с помощью электронной коммерции. Боб Меткэйлф, изобретатель Ethernet, высказал гипотезу, что значение сети для социума пропорционально квадрату числа пользователей, потому что это – примерно число различных соединений, которые могут быть сделаны. Эта гипотеза известна как «закон Меткэйлфа». Она объясняет, что огромная популярность интернета обусловлена его размером [5].

Доступ к удалённой информации может осуществляться в различной форме. Можно «бродить по Сети» в поисках нужной или просто интересной информации. При этом практически невозможно найти такую область знаний, которая не была бы представлена в интернете. Там есть искусство, бизнес, кулинария, политика, здоровье, история, различные хобби, отдых, наука, спорт, путешествия и многое-многое другое, в том числе и запрещённое законодательством. Многие газеты стали доступны в электронном виде, их можно персонализировать. [5] Например, в рамках одного издания можно заказать себе лишь отдельные статьи на определённые темы, отказавшись от других.

Следующий шаг после создания электронных версий газет и журналов – это электронные библиотеки. Многие зарубежные профессиональные организации, такие как **Ассоциация вычислительной техники** ACM[343] и даже объединение IEEE Computer Society [344], занимаются этим. **Российская государственная библиотека** [345] также предоставляет электронный доступ к различным своим изданиям и ведёт работы по оцифровыванию имеющихся в их фонде бумажных носителей. Многие фирмы и частные лица выкладывают свои коллекции самых разнообразных материалов в интернете. Электронные книги и онлайн-библиотеки приводят к тому, что ежегодно число бумажных печатных изданий постепенно сокращается. Скорее, это связано не столько с удобствами, предоставляемыми электронными книгами, компьютерами-планшетами и телефонами, сколько со стоимостью бумаги и ценой печати. Молодое поколение не приучено покупать книги отчасти и потому, что для большинства из них бумажные издания не представляют никакой ценности, независимо от цены, а как следствие спрос падает ещё большими темпами.

Скептики уже сейчас сравнивают появление «электронных книг» с эффектом от появления в Средние века печатного станка, который заменил ручное письмо. Несомненно лишь одно, что независимо от формы издания наибольший интерес представляла и представляет новая (для читателей) информация, содержащаяся в книгах, которая всегда будет востребована. Вы же читаете это издание по какой-то причине?

[343] Association for Computing Machinery, http://acm.org.
[344] http://computer.org.
[345] ФГБУ «РГБ», ранее Государственная библиотека СССР имени В.И. Ленина, http://rsl.ru.

Собственно, благодаря компьютеру и сетям доступ к большей части информации осуществляется по модели «клиент–сервер», кратко рассмотренной в прошлом разделе, но есть и другой популярный тип сетевого общения – равноранговый. При такой форме связи ранее предложенное разделение на клиентскую и серверную части отсутствует. Каждый может связаться с каждым, выступая то клиентом, то сервером, в зависимости от ситуации. Основанные на таком принципе сети называются равноранговыми сетями (peer-to-peer), одноранговыми сетями или пиринговыми сетями, а информационные потоки называют пиринговым трафиком или пиринговым информационным обменом.

Как и у демократии, при всех её рекламируемых плюсах, у равнорангово построенных сетей существует небольшая проблема, а именно невозможно точно оценить число активных участников сети. Многие одноранговые сети, например BitTorrent, не имеют никакой центральной базы о том, что есть у её пользователей. (Имеется в виду, что сеть используется для обмена файлами, а под словами «что есть» понимается список файлов готовых для раздачи другим участникам сети.) Вместо этого каждый пользователь поддерживает свою собственную базу данных в местном масштабе и обеспечивает список других людей по соседству, которые являются членами системы. Новый пользователь может обратиться к любому существующему участнику, увидеть то, что он имеет, и получить имена других участников, таким образом получая доступ к большему количеству содержимого и большему количеству имён. Этот процесс поиска мог бы повторяться сколь угодно долго, чтобы в результате создать большую местную базу данных того, что имеется в наличии (по аналогии с библиотечным каталогом). Такая деятельность была бы утомительной для людей, ЭВМ же способна делать это в разы быстрее, однако компьютерно созданный каталог имеющихся файлов также не совершенен. В существующих сетях узлы могут отключаться, никого об этом не информируя, а состав имеющихся на них файлов – меняться. Поэтому процесс создания (поддержания) каталога в актуальном состоянии бесконечен, как и жизнь в широком смысле этого слова.

Равноранговые соединения узлов сети часто используются, чтобы совместно применять музыкальные, видео, программные и другие файлы. Как следствие эффективность таких сетей выше, когда у пользователей есть общие интересы. Либо интересов вовсе нет, а пользователи подвержены общим веяниям моды и рекламе. В этом случае найти и получить интересующий файл проще, так как он всем нужен (независимо от его истинной ценности). Если же ищется что-то редкое, то увы, начав поиски, они могут продолжаться до тех пор, пока разыскиваемые данные не перестанут быть актуальными для ищущего.

Если файлы большие, то эффективнее скачивать их частями. Учитывая, что многие интернет-соединения сейчас являются асимметричными[346], симметричный информационный обмен между двумя узлами в принципе не возможен. Загрузить полностью входной канал шириной в 20 Мбит/с (при исходящем 1 Мбит/с), скорее всего, будет возможно при информационном обмене не менее чем 20 аналогичными источниками. При этом одна часть файла будет получена от одного пользователя, другая – от второго, третья – от третьего и т. д. Всё это порождает сложности и приводит к созданию смешанных сетей, когда, помимо равноправного обмена, существует ещё какой-то узел

[346] Ширина канала (скорость) в сторону пользователя в несколько раз, а то и порядков, больше, чем от него. В большинстве случаев это оправдано. Отправка URL-адреса (например, страницы сайта) занимает не более 200–500 байт, а обратно может быть получено, с учётом картинок, 500 Кбайт или больше.

или несколько узлов, содержащих актуальную информацию о возможных владельцах интересующего вас файла. Например, «с актуальностью в 1 час такой файл есть на компьютерах с такими-то IP-адресами». Помимо информации о наличии, могут содержаться комментарии, оформленные в виде веб-форума. Например, «плохой сюжет», «хорошее качество» и прочее. В отношении указанных компьютеров можно услышать слово «трекер». Некоторые трекеры могут вести ваш рейтинг скачиваний и раздач, для того чтобы влиять на максимальную скорость скачивания, косвенно стимулируя вас поддерживать редкие файлы в раздаче.

Сама по себе информация, с которой работает трекер, не нарушает ничьих авторских прав, но часто является объектом нападков правообладающих компаний. В качестве примера можно привести службу Napster, когда с 2000 года такого рода коммуникации начали набирать популярность, произошёл самый большой скандал, связанный с нарушением закона об авторских правах за всю историю звукозаписи, когда компании-правообладатели смогли доказать в суде причастность «трекера» к нарушениям и закрыть проект решением суда. Истинные причины были не в нарушениях, а в том, что проект не приносил прибыль подававшим судебные иски. Экономически эффективнее было бы идти в ногу со временем, выкупить сервис и сделать его из бесплатного умеренно платным. Позднее эту коммерческую идею развил и воплотил в жизнь Стив Джобс, предложив покупать песни по одной, а не альбомами, сведя на нет первоначальную идею равноправного файлового обмена, про которую все забыли.

Между тем существуют и легальные равноранговые сети, та же технология электронной почты (e-mail) или обмен мгновенными сообщениями (Jabber) выросли именно из идеи равноранговых сетей. Электронная почта используется ежедневно миллионами людей во всём мире. Благодаря расширению MIME [347] в почте можно передавать не только текст, но и различные файлы: изображения, аудио, видео, трёхмерное видео и др. (передача запаха и тактильных ощущений – возможно, вопрос времени).

Практически все подростки увлекаются обменом мгновенными сообщениями (для них и электронная почта стала мгновенной, как SMS). Эта возможность, полученная из UNIX-программы talk, использовавшейся приблизительно с 1970 года, позволяет обмениваться сообщениями в режиме реального времени. Есть службы обмена сообщениями для нескольких человек, такие как Twitter, которая позволяет посылать короткие текстовые сообщения, названные «tweets», «чириканьем» (по аналогии с «сарафанным радио»), своему кругу друзей или тем, кто на это согласился. [5]

Интернет может использоваться для передачи потоков аудио (например, интернет-радиостанции) и видео (например, YouTube, RuTube, Яндекс.Видео, сайты телекомпаний). В конечном счёте именно он (по мнению авторов) объединит на базе коммутации IP-пакетов все другие современные средства информационного обмена.

Промежуточное положение между доступом к информации и коммуникациями от человека к человеку занимают социальные сети. Здесь поток информации управляется отношениями, которые люди устанавливают между собою внутри сети. Примеры социальных сетей – ВКонтакте, Одноклассники, Facebook и др. Они позволяют участникам сети обновлять свои файлы, «личные профили» и получать общий доступ к этим обновлениям с теми, кого они объявили своими друзьями. Если страница открыта – «другом» может являться кто угодно.

[347] От англ. Multipurpose Internet Mail Extensions – могоцелевые расширения электронной почты, подробнее см. RFC 822, RFC 2822, RFC 5322.

Цель таких сетей – реализовать свою возможность поделиться информацией о том, как, например, вы были на море с детьми, а младшая на видео научилась плавать, или с друзьями ходили на выставку и прочее. Другие сети могут больше иметь профессиональную направленность – собрать специалистов и вакансии в одном месте. В сети предпочтительно выставляются не ваши любимые музыкальные произведения и статусы вида «я принимаю водные процедуры» и «слушаю такую-то песню», а скорее все ваши служебные достижения: регалии, сертификаты, курсы, рекомендации и прочие заслуги, например LinkedIn [348].

Часть приложений социальных сетей может быть представлена через друзей друзей, посылать сообщения новостей друзьям, например упомянутый выше Twitter и многое другое.

Такие технологии позволяют создать подробное досье о каждом жителе планеты на основе результатов анализа его блогов и трафика используемых им электроприборов (различных гаджетов, компьютеров и прочего). Ежегодно доля трафика в сети, генерируемая именно устройствами (а не человеком), растёт. Данный феномен уже получил название «интернет вещей» (IoT, Internet of Things). [9]

Ещё более свободно группы людей могут сотрудничать, совместными усилиями создавая так называемый «контент» (содержимое). Технология wiki, например, – это веб-сайт, который совместно редактируют члены сообщества [5]. Самая известная wiki – Википедия, энциклопедия, которую любой может редактировать. Даже вы, читающий эти строки, можете это сделать. В случае если вы ошибаетесь, либо заведомо напишите глупость, вас исправят другие пользователи либо ваши творческие изложения будут с пометкой «непроверенные данные». Проверяют данные такие же пользователи, зарегистрированные и «с большим стажем». Недостаток подобного подхода – все дружно могут ошибаться в вопросе или мнении, склоняясь в какую-то одну неверную или субъективную сторону, не будучи специалистами [349]. Кроме Википедии, есть тысячи других сайтов wiki.

Ещё одна область использования сетей частными лицами – это та же «электронная коммерция», что и между юридическими лицами (подробнее см. раздел 6.6 «*Электронная коммерция*»). Закупка продуктов и вещей для дома через интернет уже давно стала привычным делом. Многие пользуются неравномерностью заполнения рынков товарами, в том числе и из-за законодательных ограничений, заказывая себе необходимые товары из-за рубежа. Электронная коммерция широко используется и в другом

[348] http://linkedin.com.

[349] В ряде случаев школьники больше верят Википедии, чем живым учителям. Плохие учителя, с низким уровнем знаний и слабым авторитетом, не учащие учеников думать, сомневаться в полученных знаниях, лишь утверждают это мнение в сознании подрастающего поколения, что в ней есть всё и всё написанное правильно. Увы, это заблуждение людей, не знакомых с интернетом подробно. При таком подходе, если в Википедии про Россию написать, что наш Дальний Восток принадлежит японцам или китайцам, а вовсе не наш, то мы это и получим, потому как сначала никто не сможет исправить ошибку, а потом никто и не будет возражать, что часть исконно нашей территории, богатств и достояний перестанут быть нашими. Увы, но против России ведётся целенаправленная информационная война, в том числе и через Википедию. В это сложно поверить, но задайтесь вопросом и проведите соцопрос своего юного окружения: «Кто выиграл Вторую мировую войну?» Не даром на Руси существовала поговорка «Что написано пером – не вырубить топором», так вот в случае ведения информационной войны корректировку содержимого интернет-сайтов (чьи хостинги находятся физически больше за рубежом, чем у нас) можно произвести менее чем за секунды. Уровень образованности среднестатистического пользователя интернета в России падает. Есть школьники и даже взрослые, которые не знают сегодня, как найти те или иные знания, кроме как набрав запросы в поисковой системе Яндекса, Рамблера, Google или др.

серьёзном направлении: доступ к финансовым учреждениям. Многие оплачивают свои счета и работают со вкладами с помощью соответствующих сетевых служб. Наряду с удалёнными поручениями по проведению операций с реальными банковскими счетами существуют и электронные деньги. Qiwi-кошельки, Яндекс-деньги, web-money (рублёвые и валютные) и др. По мере прогресса в области защиты передаваемой информации тенденция использования финансовых операций, безусловно, продолжится.

Ещё одна область, которую вряд ли кто-то мог предвидеть, – это электронные «блошиные» рынки. Онлайновые аукционы, на которых распродают подержанные вещи (б/у, бывшие в употреблении, second hand), стали довольно мощной сетевой индустрией. [5] Наиболее популярны avito.ru (общая база со slando.ru), ebay.com, komok.com, molotok.ru, irr.ru и др. В отличие от традиционных форм электронной коммерции, использующих модель «клиент–сервер» [350], такие разновидности бизнеса ближе к равноранговым сетям, в том смысле что пользователи могут действовать и как продавцы, и как покупатели.

Ещё одна область – развлечения. Она стремительно развивается в последние годы с распространением музыки, радио- и телевизионных программ и фильмов в интернете, начинающем конкурировать с традиционными механизмами. Пользователи могут бесплатно найти и загрузить (как и купить) песни в формате MP3 и фильмы качества HDTV (или выше) и добавить их в свою личную коллекцию, либо перекачать на планшет (mp3-плеер, телефон) для дальнейшего автономного просмотра или прослушивания. Многие телевизионные (и радио) передачи и сериалы теперь можно посмотреть по системе IPTV (IP-TeleVision), которая основана на кодировании видеоизображения и помещении его частей в IP-пакеты, которые в последующем передаются по сети Интернет пользователям вместо традиционного кабельного или радиовещания. Приложения, передающие потоковое мультимедиа, позволяют пользователям слушать интернет-радиостанции или смотреть новые либо пропущенные эпизоды их любимых сериалов. Есть и обратная сторона данного явления – растёт интернет-зависимость людей со слабой психикой, не имеющих других видов общественно полезных занятий: дети, подростки и пенсионеры.

Технически уже сейчас, с появлением различных интернет-подключаемых «TV-приставок» и телевизоров, в полной мере «видящих своих пользователей» со встроенной видеокамеры и позволяющих последним управлять техникой взмахом руки, можно организовать доступ к видеоархивам со всеми когда-либо снятыми видеосюжетами по запросу, а также создавать обратную связь со зрителями, например для выбора той или иной концовки фильма. Практическая реализация вопроса тормозится отсутствием спроса на подобные услуги, а также из-за проблем в юридической плоскости – отставания законодательной базы в вопросах защиты авторских прав и оплаты. Ни один капиталист не будет работать за идею, не приносящую прибыль.

К сожалению, «новые» технологии сегодня больше используются не на благо общества, а во вред. Например, заболевшему и оставшемуся дома подростку непросто найти по телевизору действительно информативную и полезную передачу. В интернете изобилует реклама и порнография, идёт постоянная пропаганда жизни в стиле обще-

[350] Электронные магазины вида ozon.ru, amazon.com и других, хотя и похожи на доски объявлений, но не подходят под данную категорию, потому как это взаимодействие бизнеса (магазина, юридического лица) с покупателем, нежели взаимодействие физических лиц друг с другом.

ства потребления. Многие СМИ далеки от объективности, постоянно производится навязывание той или иной точки зрения.

Другая форма развлечения – игры. Уже сейчас существуют игры-симуляторы в реальном времени с большим количеством участников, например прятки в виртуальном средневековом замке [5] или симуляторы космических полётов, с торговлей виртуальными товарами и виртуальными деньгами. Героев можно не только виртуально одевать в разные костюмы, но и обучать («прокачивать» теми или иными знаниями, повышать их «скил»: от англ. skill – мастерство, умение, искусство). Виртуальные миры обеспечивают постоянное место действия, где тысячи пользователей могут иметь совместный доступ к виртуальной реальности с трёхмерной графикой. Во многих подобных играх участие платное (около 300–600 рублей в месяц), и играют в них далеко не дети. Оплачивать участие можно как виртуальными деньгами, заработанными в игре, так и реальными. Также можно за счёт реальных денег пополнять свои виртуальные игровые счета и, как следствие, покупать различные космические корабли, двигатели, защиты и прочее. К сожалению, и игры, несмотря на свой высокий потенциал, несут больше вреда, чем пользы современным подросткам.

6.1.2. Социальные аспекты в развитии сетевого обмена информацией

Всё большее количество потребительских электронных устройств объединяется в сеть. Например, фотоаппараты научились присоединяться к беспроводной сети и используют её, чтобы послать фотографии «в облако». Даже если в вашем цифровом фотоаппарате не предусмотрен Wi-Fi/3G/LTE (4G) или иной модуль связи, вы уже сейчас можете купить карту памяти со встроенным Wi-Fi-адаптером, например Eye-Fi [351] и др. Устройства, например телевизоры, могут использовать электрическую сеть, чтобы посылать информацию «по дому», используя силовую проводку. Обмениваться информацией могут и объекты, о которых мы совсем не думаем как о компьютерах.

Технология под названием RFID (Radio Frequency Identification, радиочастотная идентификация) продвинет эту идею ещё далее в будущем. [5] Метки RFID пассивны (то есть не имеют источника питания), размером не более почтовой марки. Они могут быть прикреплены к книгам, домашним животным, кредитным картам и другим элементам дома и на улице. Это позволяет RFID-считывателям определять местонахождение и «общаться» на расстоянии до десятков метров в зависимости от вида RFID. Первоначально коммерческой целью RFID было заменить штрих- (а теперь и QR-) коды, пока этого не произошло, потому что штрихкоды бесплатны (не более себестоимости отпечатка), а метки RFID стоят небольших, но денег.

Логотип общества, выступающего против распространения RFID-меток

RFID (англ. Radio Frequency IDentification, радиочастотная идентификация) – метод автоматической идентификации объектов, в котором посредством радиосигналов считываются или записываются данные, хранящиеся в так называемых транспондерах, или RFID-метках.

[351] http://eye.fi/ Wi-Fi SD Card. SD-карта памяти со встроенным Wi-Fi-модулем.

Технические характеристики RFID

По дальности считывания RFID-системы можно подразделить на системы:
- ближней идентификации (считывание производится на расстоянии до 20 см);
- идентификации средней дальности (от 20 см до 5 м);
- дальней идентификации (от 5 м до 100 м).

Только оцените дальность – до 100 метров! – а также оцените, где и как часто сегодня используются RFID-технологии. Они применяются в самых разнообразных сферах человеческой деятельности:
- промышленность;
- транспортная и складская логистика;
- системы контроля и управления доступом;
- медицина – мониторинг состояния пациентов, наблюдение за перемещением по зданию больницы;
- библиотеки – станции автоматической книговыдачи, быстрая инвентаризация;
- паспорта;
- транспортные платежи;
- дистанционное управление;
- опознавание животных, электронное окольцевание птиц;
- сельское хозяйство;
- человеческие имплантаты;
- системы управления багажом;
- системы локализации объектов в реальном режиме времени
- и др.

В первую очередь сегодня используется функционал RFID, предоставляющий информацию об объекте, его свойствах, качествах и т. п., а также о его положении.

Информация для размышления: по данным IdTechEx 2011, в 2015 г. будет изготовлено 2 млрд активных RFID-меток и триллион пассивных. Крупнейшим потребителем RFID-меток является сеть супермаркетов Walmart. На втором месте идёт министерство обороны США. [9] Уже сейчас недорого стоят принтеры программирующие метки-наклейки с одновременной печатью на них же штрих кода.

Прогресс остановить невозможно, поэтому радиометки уже сегодня используются в различных документах, например в российских загранпаспортах, выдаваемых сроком на 10 лет. При этом вполне вероятно, что в будущем прохождение государственных границ будет возможным «на ходу» за время, пока вы двигаетесь от самолёта до выхода из здания аэропорта. Вопрос – лишь быстро считать данные из паспорта и сопоставить с изображением, полученным с видеокамеры для идентификации.

Если же ввести радиометки в водительские права, то, установив считыватели по всем автомагистралям, можно будет не только отслеживать перемещения указанных документов, но и штрафы, полученные от комплексов фото- и видеофиксации нарушений, выставлять непосредственно водителям-нарушителям, а не владельцам транспортных средств, как это происходит сейчас.

С другой стороны, вопрос появления удалённых (с широким радиусом действия) RFID-считывателей, «копировщиков» и прочей китайской технической утвари за небольшие деньги – вопрос лишь времени.

Мы же привыкли к крупным и заметным номерам на автомобилях, некоторые владельцы даже стремятся заполучить красивое сочетание букв и цифр, так почему же не продавать красивые номера для RFID-меток? Вам такое предложение кажется утопией, но в мире, где все ценности перевёрнуты «с ног на голову», возможны и не такие глупости.

Радиометки наряду со штрихкодами используются в системах складского учёта, например с помощью карманных компьютеров. Во многих крупных и загруженных аэропортах служащие, принимающие на стоянках арендованные машины, зачастую используют мобильные компьютеры и телефоны. Они фотографируют QR-код, сканируют штрихкод, или считывают RFID-метку машины, их мобильное устройство, имеющее встроенный принтер, связывается с главным компьютером, получает от него информацию и сразу же распечатывает счёт. Не думаем, что вы будете в восторге, если всем, в том числе и криминальным элементам, будет доступна информация о ваших перемещениях, а также, например, о марке, модели, цене и цвете нижнего белья, которое на вас в текущий момент.

Один из двигателей развития приложений беспроводной связи – подростки, активно отправляющие короткие текстовые сообщения с мобильных телефонов, являющиеся огромным источником дохода для телефонных компаний. Несмотря на падение цен за интернет-трафик, услуга по использованию коротких сообщений – SMS'ок «Short Message Service» – не перестаёт быть актуальной и очень выгодна для сотовых компаний, так как передача текстового сообщения стоит меньше копейки, а берут за это гораздо больше.

Долгожданная конвергенция телефонов и интернета наконец наступила, и это ускорит рост мобильных приложений. Умные телефоны (Smart Phones) объединяют в себе возможности мобильных телефонов и мобильных компьютеров. [5] Сотовая связь третьего и четвёртого поколений, с помощью которой они соединяются, может обеспечить быстрый доступ к различным интернет-сервисам, помимо привычной обработки телефонных звонков. Усовершенствованные телефоны соединяются с беспроводными точками доступа и автоматически переключаются между сетями, чтобы выбрать наилучший вариант для пользователя, например SIP-звонок может быть намного дешевле сотовой телефонии. Другие устройства бытовой электроники могут использовать сотовые сети и сети с точками доступа для соединения с удалёнными компьютерами. [5] Электронные устройства чтения книг и планшеты могут загрузить недавно купленную книгу или очередной выпуск журнала или сегодняшней газеты везде, где бы они ни находились. При этом имея доступ к вашему устройству, можно узнать о ваших предпочтениях.

Электронные фоторамки могут обновлять изображения на своих дисплеях. В процессе передачи данные, то есть ваши любимые фотографии, могут быть стать доступными третьим лицам, хотя многим это и надо, иначе зачем и без радиотехнологий выкладывать фотографии в сети на общий доступ с просьбами «like-нуть» их. Происходит подмена ценностей, вместо того чтобы наслаждаться как таковым отдыхом, видами и прочим, пользователям становится важнее, чтобы об этом узнали их друзья и коллеги, так называемый сопернический инстинкт и желание продемонстрировать свой ранговый потенциал.

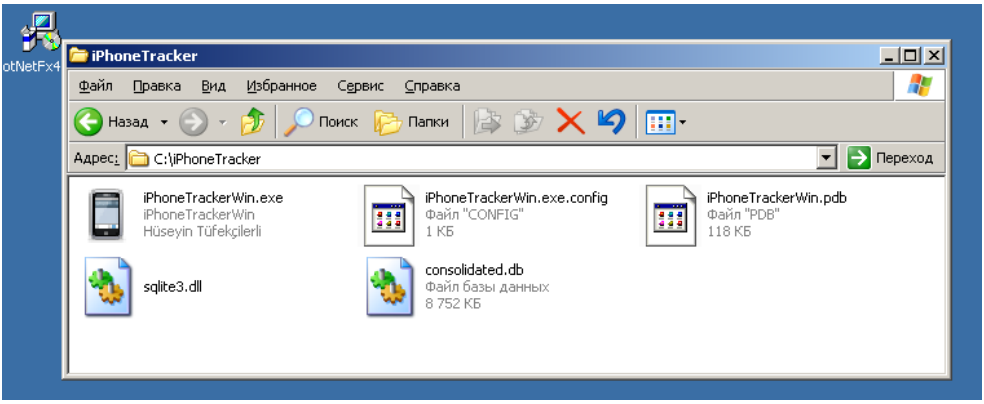
Мобильные телефоны «знают» своё местоположение, потому что они оборудованы системой глобального позиционирования GPS, или ГЛОНАСС, или могут получать свои координаты по пересчёту радиосигналов от ближайших базовых станций или Wi-Fi-точек доступа. Многие мобильные сервисы зависят от местоположения пользователя. Поисковые системы, мобильные карты и указатели – очевидные кандидаты. Фокус, но порой ваш телефон и автомобиль с навигатором, вероятно, лучше вас знают, где вы находитесь. Такая же ситуация с поиском ближайшего книжного магазина, или аптеки, или получением местного прогноза погоды. Другие службы могут записывать текущее местоположение, например указывать для фотографий и видео место, в котором они были сделаны. Такие указания называют «геотегирование». [5]

Зарубежные мобильные устройства

С распространением мобильных устройств возник вопрос конфиденциальности местоположения [352]. В процессе оказывания услуги вашему мобильному устройству сетевые операторы узнают, где вы находитесь в течение дня. Это позволяет им отслеживать ваши передвижения. Они могут знать, какой ночной клуб или медицинский центр вы посещаете, вплоть до этажа и расположения кабинета врача. [5][353]

Устройства iPhone, iPad, iPod и другие постоянно ведут базу своих перемещений, то есть и их владельцев. Данные о перемещениях складируются в файле /private/var/root/Library/Caches/locationd/consolidated.db. Для получения указанного файла совсем не обязательно иметь доступ к файловой системе устройства и делать jailbreak. При синхронизации устройства с программой iTunes указанный файл копируется на компьютер пользователя в директорию /Users/***/Library/Apllication Support/Mobile Sync/Backup, откуда может быт свободно скопирован.

Существуют программы вроде iPhoneTracker [354] под MacOS, а также её портированная версия под ОС Windows [355], которые позволяют визуализировать данные файла consolidated.db (см. рис. 6.3).



(продолжение рисунка 6.3 на следующей странице)

[352] Beresford A., Stajano F. Location Privacy in Pervasive Computing // IEEE Pervasive Computing, vol. 2, pp. 46-55. Jan. 2003, http://www.cl.cam.ac.uk/~fms27/papers/2003-BeresfordSta-location.pdf.

[353] Также посмотрите видео от 2018 года записи конференции TED https://www.youtube.com/watch?v=4tAOsN-ueR4 с данными в нём от 2009 года, а после подумайте что произошло за 12 лет!?

[354] http://petewarden.github.com/iPhoneTracker/, http://petewarden.github.io/iPhoneTracker/.

[355] http://huseyint.com/iPhoneTrackerWin/.

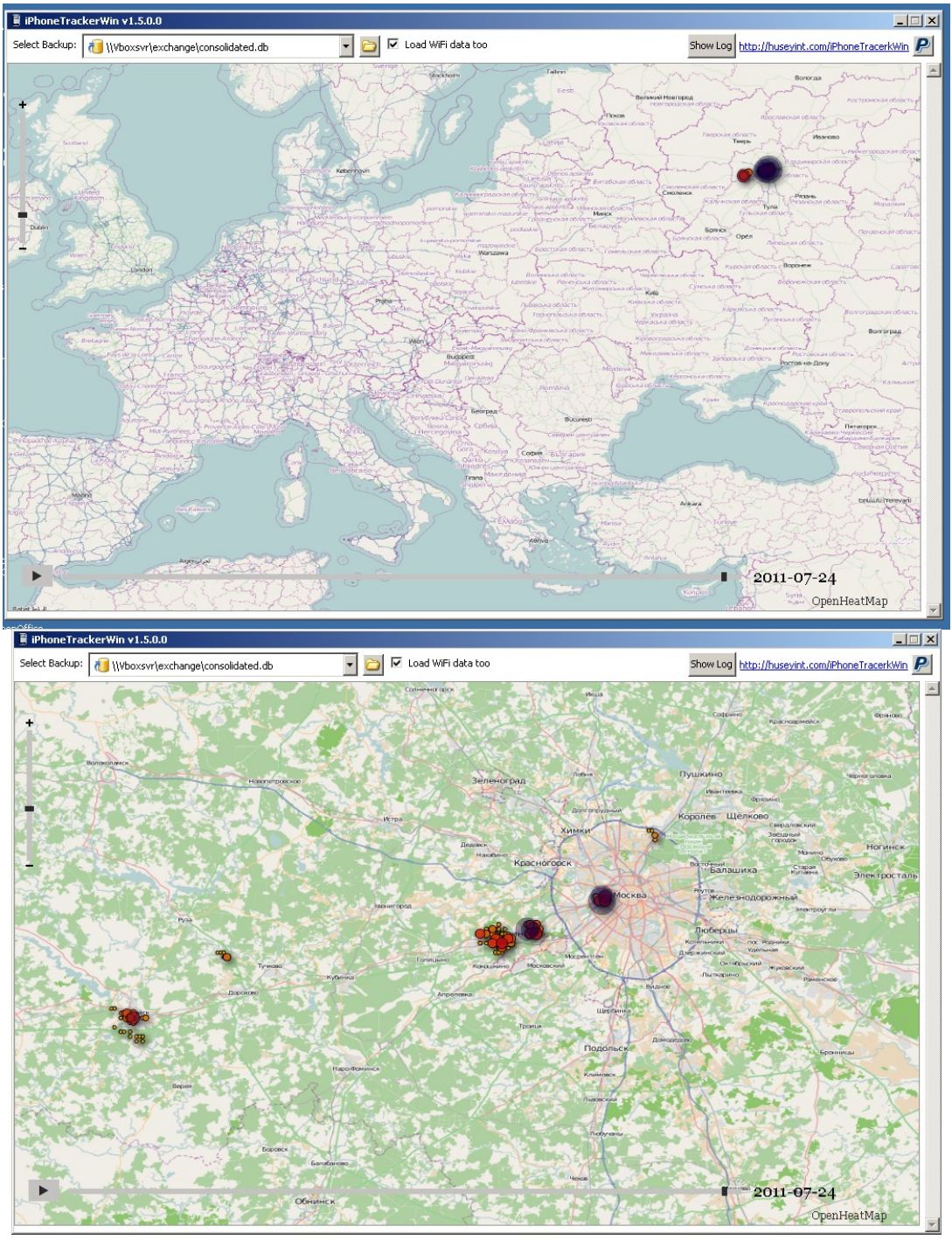


Рисунок 6.3. Программа iPhoneTrackerWin

В некоторых Android-устройствах ранее использовалась программа Carier IQ, постоянно записывающая различные параметры телефона, состояние памяти, запущенные приложения, местоположение, историю нажатий кнопок и прочее. По заявлениям

разработчиков, приложение используется для сбора информации, которая может пригодиться при отлаживании программного обеспечения телефона, ведь телефоны также на рынок выходят с «сырыми» и нередко зависающими операционными системами.

Официально мобильные устройства на базе Android также постоянно фиксируют местоположение пользователя и хранят эти данные. Координаты они определяют несколькими способами: через геолокацию, вышки операторов сотовой связи или точки Wi-Fi. Если у вас есть Android-устройство, то, интереса ради, вы можете зайти в свой аккаунт и посмотреть историю перемещений вашего устройства. Она хранится с самого первого дня использования устройства и не очищается автоматически. Для просмотра достаточно залогиниться зайти в Google Maps через аккаунт Google, где выбрать страницу истории местоположений. Там вы увидите карту с линиями перемещений в течение суток и календарь с историей за любой день. График под картой позволяет просмотреть историю перемещений в течение выбранного дня, при нажатии на точку можно увидеть, когда мобильное устройство было в отмеченном на карте месте.

Аналогичным образом работает функция «Find My Device» для ОС Windows 10, кроме этого «*20.07.2016 Национальная комиссия по информационным технологиям и свободе (CNIL) Франции предписала американской корпорации Microsoft привести свою операционную систему Windows 10 в соответствие с законом, ограничив сбор личной информации.*

«Комиссия предписывает Microsoft Corporation прекратить избыточный сбор данных и наблюдение за действиями пользователей без их согласия. Она требует также в удовлетворяющем порядке обеспечить конфиденциальность данных пользователей», – сказано в коммюнике CNIL.

Данные собственного расследования комиссии свидетельствуют о том, что Microsoft собирает слишком много данных. В частности, разработчики ОС получают информацию о загружаемых пользователем приложениях, о том, как и сколько времени он использует каждое из них.[356]»

Подобные функции могут оказаться полезными при контроле родителями своих несовершеннолетних детей, поиске недееспособных лиц, преступников, заблудившихся экспедиций, больных или страдающих потерей памяти, однако, следует помнить о невидимой тонкой грани законов Российской Федерации, ограждающих личное пространство каждого из нас, указанная информация может легко оказаться и в руках злоумышленников. Нередки случаи когда миллионы гаджетов оказываются беззащитны перед вредоносными программами. Например, новостное сообщение о программе FREAK гласит: «*...Ранее сообщалось, что партнёры АНБ из британского Центра правительственной связи (GCHQ) следят за владельцами iPhone. У каждого iPhone есть уникальный 40-значный идентификационный номер (UDID). Узнать номер определённого человека организация может разными способами, проще всего это можно сделать, если за телефон заплатили кредитной картой. Получив искомый UDID, сотрудники GCHQ могут выяснить, какие приложения установлены на конкретном аппарате, а далее, используя имеющиеся в них уязвимости, получить доступ к хранящейся в телефоне информации.*»[357]

[356] Оригинал публикации Власти Франции просят ограничить сбор личной информации в Windows 10 https://ria.ru/world/20160720/1472446763.html.

[357] Оригинал публикации Миллионы гаджетов Apple и устройств на Android беззащитны перед вредоносной программой FREAK http://russian.rt.com/article/77569.

Аспекты интернета вещей

Областью, в которой начинают использоваться мобильные телефоны, является m-commerce (мобильная торговля). Короткие текстовые сообщения с мобильного телефона используются вместо наличных денег и кредитных карт для оплаты парковки, продуктов в торговых автоматах, билетов в кино и других мелочах. Оплата будет включена в счёт мобильного телефона. Кстати, в центре Москвы (2013 г.) деньги за неиспользованные часы парковки, если вы уехали ранее, могут возвращать также по SMS, обратно на телефонный счёт. Мобильный телефон, снабжённый технологией NFC (Near Field Communication), может действовать как карта RFID и взаимодействовать для оплаты с ближайшим считывателем. В будущем, вероятно, будут развиваться [358] технологии, основанные на всеобщей тенденции миниатюризации вычислительной техники и повсеместного её встраивания (продолжения развития «интернета вещей»), а в указанной технике предположительно будет востребована низкоресурсная криптография [9] для защиты указанных устройств.

Итак, подобно печатному станку 500 лет назад, компьютерные сети предоставляют новые способы распространения гражданами их взглядов среди самой различной аудитории. Новая свобода распространения информации несёт с собой и новые нерешённые политические, социальные и этические проблемы. Упомянем кратко только некоторые из них; полное исследование потребовало бы, по крайней мере, книги. [5, стр. 29]

Социальные сети, доски объявлений, сайты для хранения контента и узлы с другими приложениями позволяют людям делиться своими взглядами с аналогично мыслящими людьми. [5, стр. 29]

До тех пор пока обсуждаемый предмет не выходил за рамки техники или увлечений вроде возделывания огородов, проблем, которые могут возникнуть, было не так уж много. [5, стр. 29]

Проблемы начались с возникновением конференций, посвящённых темам, по-настоящему волнующим людей, таким как политика, религия или секс. Взгляды, излагаемые одними людьми, могут оказаться оскорбительными для других. Они и в самом деле зачастую далеки от норм политкорректности. Кроме того, сетевые технологии, как известно, не ограничены только лишь передачей текста. Без особых проблем по Сети ходят фотографии с высоким разрешением и даже видеофрагменты. Некоторые люди придерживаются позиции «живи и дай жить другим», однако другие считают, что помещение в Сети некоторых материалов (словесные угрозы в адрес отдельных стран или религий, порнографии т. д.) просто недопустимо и такой контент должен подвергаться цензуре. Законодательства разных стран имеют различные взгляды на эту проблему; таким образом, страсти накаляются. [5, стр. 29] Например, по сообщениям РИА «Новости», 30 июля 2013 г. Глава комитета Госдумы по информационной политике Алексей Митрофанов предлагал рассмотреть вопрос о запрете на размещение в интернете записей видеорегистраторов, так как (по его мнению) это противоречило российской Конституции и нарушало право на частную жизнь. «Это проблема, которую мы должны рассмотреть отдельно», – заявил Митрофанов во время круглого стола в Госдуме, посвящённого защите детей от мата в интернете. Многие автолюбители поз-

[358] Из презентации NSA Trusted Systems Research Group проведённой в MIT Media Lab 30 января 2013 г.: «RFID technology adoption is accelerating rapidly. Merged into real-time location systems (RTLS) and converged with worldwide communications systems, by 2013 RFIDs will be a pervasive way to identify, locate, and track people and objects.»

волили себе с ним не согласиться. С другой стороны, на портале госуслуг [359], наоборот, ещё в 2014 году планировали принимать «видеожалобы» от граждан, чтобы навести порядок и не отставать от технического прогресса, используя его на благо общества в целом, то есть нас с вами.

Раньше люди подавали в суд на сетевых операторов, считая их ответственными за содержимое сайтов, подобно тому как газеты и журналы несут ответственность за содержимое своих страниц. В ответ же операторы сетей утверждают, что сеть подобна телефонной компании или почтовому отделению и они не могут отвечать за то, что говорят их клиенты, а тем более управлять содержанием этих разговоров [5, стр. 29].

Теперь немного удивительно узнавать, что некоторые сетевые операторы блокируют контент по своим собственным причинам. Некоторых пользователей приложений соединения одноранговых узлов отключали от сети, потому что сетевые операторы не считали выгодным передачу большого количества трафика, посылаемого этими приложениями. Те же самые операторы, вероятно, хотели бы по-разному обслуживать разные компании. Если вы – крупная компания и платите хорошо, тогда вы получаете хорошую услугу, но если вы – мелкий игрок, вы получаете услугу худшего качества. Противники этой практики утверждают, что соединение одноранговых узлов и другой контент должны быть обработаны одинаково, потому что все они – биты в сети. Такая позиция, выступающая за коммуникации, которые не дифференцированы их контентом, источником или тем, кто создал контент, известна как сетевой нейтралитет [360]. Вероятно, эти дебаты продолжатся ещё некоторое время [5, стр. 30].

В споре относительно контента есть и другие стороны. Например, пиратская музыка и фильмы питали массивный рост одноранговых сетей, что не нравилось правообладателям, которые угрожали судебными исками (и иногда так и делали). [5, стр. 30.] В ряде стран за рубежом появились автоматизированные системы, которые ищут одноранговые сети и посылают операторам предупреждения о пользователях, которые подозреваются в том, что посягают на авторское право. В США эти предупреждения известны как уведомления DMCA, появившиеся после принятия Закона о защите авторских прав в цифровую эпоху (Digital Millennium Copyright Act). Такой поиск – гонка вооружений, потому что достоверно поймать нарушение авторского права трудно. Даже Ваш принтер мог бы быть принят за преступника [361] [5, стр. 30].

Компьютерные сети делают общение очень лёгким. Они также упрощают возможности отслеживать трафик. Так областью конфликтов оказались права наёмных работников, вступившие в противоречие с правами нанимателей. Некоторые наниматели считают себя вправе читать и, возможно, подвергать цензуре сообщения своих работников, включая сообщения, посланные с домашних терминалов после работы. Не все с этим согласны [5, стр. 30].

Другой конфликт разворачивается вокруг проблемы взаимоотношений государства и граждан. Известно, что в поисках крупиц информации ФБР установило на серверах многих поставщиков услуг специальные системы, позволяющие просматривать

[359] Портал государственных услуг Российской Федерации http://gosuslugi.ru.

[360] Wu T. «Network Neutrality, Boardband Discrimination» // Journal on Telecom and Hi-Tech Law. – 2003. – Vol. 2. – Pp. 141–179.

[361] Piatek M., Kohno T., Krishnamurthy A. Challenges and Directions for Monitoring P2P File Sharing Networks or Why My Printer Received a DMCA Takedown Notice 3rd Workshop on Hot topics in Security, USENIX, July 2008 // http://dmca.cs.washington.edu/uwcse_dmca_tr.pdf, Савчук И. Принтер как источник угрозы. Антируководство по взлому локальных сетей. Ч. 1-2 // Системный администратор. -- 2012. – №1–2 (110–111). – С. 58–60; № 3 (112). – С. 64–68.

входящую и исходящую почту. [5, стр. 30.] Система изначально называлась Carnivore [362], однако такое зловещее название обращало на себя слишком много внимания общественности. Было решено переименовать систему и назвать ее невинным именем – DCS1000 [363]. Цель таких систем состоит в том, чтобы шпионить за миллионами людей в надежде на обнаружение информации о незаконной деятельности [5, стр. 30]. Даже в самой «демократичной» стране – США – de jure четвёртая поправка к их Конституции запрещает правительственные поиски без ордера на обыск, но de facto правительство часто игнорирует это. Достаточно посмотреть на последние информационные сообщения в отношении Эдварда Сноудена, Джулиана Ассанджа, Бредли Мейнинга и станет ясно чего стоит их разрекламированная демократия.

X-Keyscore – секретная программа компьютерного слежения, осуществляется совместно Агентством национальной безопасности США, Управлением радиотехнической обороны Австралии [364] и Службой безопасности правительственных коммуникаций Новой Зеландии. Предназначена для слежения за иностранными гражданами во всём мире (в том числе и за россиянами), деятельность осуществляет с помощью более чем 700 серверов, расположенных в США и на территории стран-союзников США, а также в посольствах и консульствах США в нескольких десятках стран [365]. Их партнёрами были такие крупные компании, как Google, Facebook, Microsoft и другие. [16] Существование программы было раскрыто в июле 2013 года.

PRISM – государственная компьютерная программа США, принятая американским Агентством национальной безопасности (АНБ) в 2007 году в качестве замены Terrorist Surveillance Program. Широкой общественности о существовании программы стало известно 6 июня 2013 года, когда отрывки из секретной презентации о PRISM были опубликованы в газетах «Вашингтон пост» и «Гардиан».

Если вы не верите иностранной прессе, то обратитесь к публикациям в Российской газете от 21 августа 2013: «*По данным издания (The Wall Street Journal), сотрудники ведомства могут контролировать как звонки через интернет, так и содержимое электронных писем. Это стало возможным благодаря сотрудничеству АНБ с интернет-провайдерами, которые оправляют агентству трафик, с высокой вероятностью содержащий те или иные разведданные. Полученная информация тщательно анализируется.*» [16], либо посмотрите на русском языке фильм «База» А. Мамонтова из серии «Специальный корреспондент», в том числе и про «Эшелон», снятый до вышеописанных информационных открытий летом 2013-го.

Конечно, не только правительства стран угрожают частной жизни, многие коммерческие фирмы также вносят свою лепту, создавая профили своих клиентов. И это не только истории заказов клиентов на серверной стороне, но и различные cookie-

[362] Англ. хищник.

[363] Blaze M. A., Bellovin S. M. Tapping on my network door // Commun. of the ACM. – 2000. – Oct. – Vol. 43. – p. 136. http://www.csl.sri.com/users/neumann/insiderisks.html, Sobel D. L. Will Carnivore Devour Online Privacy? // Computer. – 2001. – May. – Vol. 34. – № 5. – Pp. 87–88., ZACKS, M.: «Antiterrorist Legislation Expands Electronic Snooping» IEEE Internet Computing, vol. 5, pp. 8-9, Nov.-Dec. 2001.

[364] Snowden reveals Australia's links to US spy web: http://www.smh.com.au/world/snowden-reveals-australias-links-to-us-spy-web-20130708-2plyg.html.

[365] German Intelligence Used NSA Spy Program //
http://www.spiegel.de/international/germany/german-intelligence-agencies-used-nsa-spying-program-a-912173.html.

файлы, хранимые на стороне самих пользователей. Благодаря ним работают такие удобные вещи автозаполнения форм адресов, номеров карт и прочего, но также они позволяют нечистым на руку сайтам, имеющим в своём составе вредоносное ПО, за счёт программных ошибок, содержащихся в браузере клиента, узнавать различную конфиденциальную информацию.

Компании, которые оказывают услуги в Сети, могут хранить большое количество персональных данных об их пользователях, что позволяет им изучать пользовательские действия. Например, если вы используете электронную почту Gmail, то Google может читать вашу электронную почту [5, стр. 31] и показывать вам контекстные рекламные объявления, основанные на содержимом приходящих и отправляемых писем. Содержимое писем анализируется подобно поисковым запросам, в некоторых случаях это более чем удобно, например вы интересуетесь у друга в письме про установку кондиционеров, на что вам сбоку показываются рекламные предложения на эту тему.

Поисковые системы, в частности Google, желая быть впереди технического прогресса давно уже предложили пользователям функцию Safebrowseing практически во всех браузерах. Есть и обратная сторона вопроса: если Вы посещаете какой-то уникальный URL, известный только вам (например, нерекламируемый корпоративный раздел на сайте), через некоторое время туда придут поисковые системы, а за ними и новые посетители, причём для этого совсем не обязательно иметь Google Toolbar, который «передаёт персональную информацию узлам sb.google.com и www.google-analytics.com» [10], или использовать Chrome. Если вы используете браузер Firefox – введите в строке поиска «about:config» и покопайтесь во всех возможных настройках, вас поразит их количество. Все вводимые адреса в адресной строке отправляются в центр анализа, указанный в настройках, и, в случае если этот сайт (доменное имя) подозревается в мошенничестве, вы получите предупреждение. Кстати, в Google не ограничились указанной функцией и пошли дальше в погоне за информацией об интересах пользователей и техническим прогрессом, «заполучив» легко запоминающиеся и красивые IP-адреса 4.4.4.4 и 8.8.8.8 и открыв на них DNS-серверы для всех желающих. С одной стороны, вы получаете всегда один стабильно работающий DNS-сервер, с другой – вы делитесь всеми своими используемыми доменными именами. Если пользователь не знает, что такое DNS (подробнее см. раздел 6.3.4.1 «*Символьная адресация*».), то он и возражать не будет, доверясь своему системному администратору, главное, чтобы всё работало.

Компьютерные сети также предоставляют возможность и увеличить защиту частной жизни, например путём отправки анонимных сообщений. В некоторых ситуациях такая необходимость есть. Помимо защиты от изучения компаниями ваших личных привычек, студенты, солдаты, служащие и граждане могут, таким образом, пожаловаться на незаконные действия профессоров, офицеров, начальства и политиков, не опасаясь репрессий. С другой стороны, в США и многих других «демократических» странах законом особо предусмотрено право обвиняемой стороны на очную ставку со своим обвинителем в суде, а также на встречный иск. Поэтому анонимные обвинения не могут рассматриваться в качестве свидетельств в суде [5, стр. 31]. В Российской Федерации, согласно Закону от 2 мая 2006 года № 59-ФЗ «О порядке рассмотрения обращений граждан Российской Федерации», в обращении, согласно ст. 7, должны содержаться «*фамилия, имя, отчество (последнее – при наличии), почтовый адрес, по которому должны быть направлены ответ, уведомление о переадресации обращения, из-*

ложенная суть предложения, заявления или жалобы, личная подпись и дата», то есть анонимные обращения граждан не рассматриваются, безопасность обратившихся обеспечивается *ст. 6* «*Гарантии безопасности гражданина в связи с его обращением*», хотя, как на одной из горячих линий ответил В.В. Путин, с долей юмора, никто не мешает вам обратиться в правоохранительные органы анонимно, на то оно и анонимно.

С целью некоторой «анонимизации» своего местоположения и «отвязки» интернет-активности (сетевого трафика) от своего текущего IP-адреса пользователи могут использовать различные proxy-серверы, web- и cgi-proxy. В последнее время набирает активность сеть Tor [366] – свободное ПО для реализации второго поколения так называемой «луковой маршрутизации». Описание работы сети, как и ей подобных, можно встретить в отечественной литературе ещё с 2006 г. [17]

Интернет позволяет с огромной скоростью находить нужную информацию, однако кто проверит её качество и достоверность? Совет врача, которого вы ждёте по поводу боли в груди, может на самом деле исходить как от лауреата Нобелевской премии в области медицины, так и от разгильдяя, которого выгнали из школы и которому нечем заняться. Другая информация часто нежелательна. Электронная макулатура, «спам», стала, увы, частью нашей жизни, потому что есть способы собрать миллионы адресов e-mail и дёшево рассылать по ним всё, что угодно. Получающаяся волна спама конкурирует с потоком сообщений от настоящих людей. К счастью, фильтрующее программное обеспечение с меньшим или большим успехом в состоянии вычислить и отсечь спам, произведённый другими компьютерами [5, стр. 31].

Так как барузеры отображают не только html, но и выполняют небольшие программы на стороне клиента (тот же Javascript и др.), возникает возможность передавать «активный контент», в том числе и макросы, обсуждавшиеся в главе про антивирусы, появляется возможность «захватить» ваш компьютер. Например, с целью украсть ваши пароли и персональные данные или вовлечь ваш компьютер в рассылку спама как часть «ботнета» или группы заражённых машин, либо произвести DDoS-атаку по заданным узлам, либо использовать ресурсы вашего ПК для майнинга.

Фишинговые сообщения (phishing messages) внешне очень похожи на настоящие, произошедшие из тех источников, которым вы доверяете. Но на деле это подделки, и таким образом пытаются узнать от вас какую-либо информацию. Вкупе с социальными сетями хищение личных данных становится серьёзной проблемой, поскольку воры собирают достаточно информации о жертве, чтобы войти в доверие и получить кредитные карты и другие документы на имя жертвы. Если вы выложили в сети ваши недавние фотографии прямо из поездки в какую-то страну, то вот вам и повод построения легенды мошенниками. К сожалению, домушники также взяли в оборот социальные сети, благодаря чему оперативно знают, когда хозяев нет дома. Зачастую крайними в цепочке событий оказывают дети, которым родители позволяют бесконтрольно в Сети писать всё, что не придёт им в голову. Сопоставив сообщения, комментарии, статусы, выложенные фотографии, аватарки и прочую общедоступную информацию на некотором промежутке времени, можно запросто нарисовать полную картину и определить время для будущего преступления. Количество комбинированных преступлений и краж с использованием информационных технологий постоянно растёт.

[366] Сокр. от англ. The Onion Router, дословно «луковый маршрутизатор».

Чтобы увидеть «динамику» в сети, необходимо вести мониторинг интересующей вас информации постоянно, с целью отслеживания изменений. В ряде случаев можно подписаться на события – например, падение цен на билеты (акции и прочее) ниже определённой или просто появление оных в продаже, но часто такие уведомления не предоставляются. Попробуйте заставьте сделать это ваших конкурентов, в этом случае на помощь приходят специально написанные программы, избавляющие людей от рутинной работы. С другой стороны, идёт обратная реакция, чтобы оградиться от различных программ, сканирующих те или иные страницы, копирующих содержимое сайтов, производящих анализ цен конкурентов из online-магазинов, постоянно придумываются те или иные механизмы. Наиболее эффективными оказываются так называемые тесты CAPTCHA [367] (в русск. транслитерацией «капча») для различения живых людей от ЭВМ, в которых вас просят решить короткую задачу распознавания, например ввести буквы, показанные в искажённом изображении, или сложить два числа, или прослушать текст и набрать его. Чтобы работа миллионов пользователей не пропадала зря, в проекте reCAPTCHA придумали не только проверять «компьютерочеловечность», но и распознавать реальные тексты и обучать системы оптического распознавания OCR.

Рисунок 6.4 а. Примеры CAPTCHA. Слева reCAPTCHA

Бывают и другие варианты CAPTCHA, например на выполнение простых арифметических действий типа сложения двух небольших чисел, нарисованных также, искажённо, картинкой, либо построение (сбор из кусочков) картинки (см. рис. 6.4б).

Рисунок 6.4 б. CAPTCHA - puzzle, решаемый мышкой. Необходимо собрать картинку

Собственно, подобные задания есть некоторая вариация на тему знаменитого *теста Тьюринга*, в котором человеку предлагается задавать вопросы, например, человеку или компьютеру, находящемуся в соседней комнате, набирая их на клавиатуре и отправляя по сети, с получением ответов на экран, чтобы после судить, является ли отвечающий человеком или ЭВМ, только проверку здесь делает ЭВМ.

В случаях когда речь не требуется наличие интеллекта человека «на другой стороне», то может использоваться «Доказательство выполнения работы» (англ. *Proof-of-work, POW, PoW*) – принцип защиты сетевых систем от злоупотребления услугами (например, от DoS-атак или организации рассылок спама), основанный на необходимо-

[367]CAPTCHA – от англ. «Completely Automated Public Turing test to tell Computers and Humans Apart» – компьютерный тест, используемый для того, чтобы определить, кем является пользователь системы: человеком или компьютером.

сти выполнения на стороне клиента некоторой достаточно длительной работы (нахождение решения задачи), результат которой легко и быстро проверяется на стороне сервера. Для пользователей появляется опасность что ресурсы их компьютеров или мобильных устройств будут использованы, например для майнинга, а это приведёт к излишнему потреблению электричества, которое копеечку, но стоит.

Многие из описанных выше проблем могут быть решены, если компьютерная индустрия всерьёз займётся вопросами защиты информации [5, стр. 32]. Например, если бы все сообщения передавались в зашифрованном виде, это позволило бы избежать огромных убытков, понесённых как частными лицами, так и крупными компаниями. Более того, несмотря на повышение накладных расходов и доли служебного трафика, многие системы уже давно разработаны, но их изучение выходит за рамки данной книги. Проблема в том, что производители аппаратного и программного обеспечения прекрасно знают, каких денег стоит внедрение систем защиты, и понимают, что попытки продать такую дорогостоящую продукцию частным лицам обречены. Например, платы с модулями TPM[368] до сих пор не присутствуют в широкой продаже. Немало хлопот доставляют и «баги» (ошибки в программах), «дыры» в защите, китайские закладки [11] и т. д. Они возникают потому, что производители добавляют всё новые и новые функции, а это приводит к увеличению числа неполадок [5, стр. 32]. Доработки ведутся как программно, так и аппаратно.

Возможным выходом из такой ситуации является взимание платы за расширенные версии программ (то есть потенциально содержащие новые ошибки), однако поди заставь конторы, занимающиеся разработкой ПО, отказаться от такого хорошего рекламного хода, как бесплатные обновления. Можно, конечно, обязать фирмы возмещать убытки, нанесённые выпущенными ими дефектными программами, однако это приведёт к банкротству практически всей программной индустрии в первый же год [5, стр. 32].

Наличие большого числа уязвимого оборудования среди «интернета вещей» привело к появлению специализированных поисковых систем типа Shodan (см. рис. 6.5).

Рисунок 6.5. Поисковая система Shodan (http://shodan.io)

[368] От англ. Trusted Platform Module, дословно «доверенный платформенный модуль», существует одноимённая спецификация для указанных модулей.

В отличие от обычных поисковых систем, индексирующих доступные веб-страницы, Shodan работает с невидимыми для человека аппаратными ресурсами сети. Он позволяет искать различные серверы, веб-камеры, принтеры, маршрутизаторы и самую разную технику, которая подключена к интернету и составляет его часть. Искать можно по различным критериям, вплоть до указания подверсий используемого ПО, где это применимо.

Shodan работает 24 часа в сутки 7 дней в неделю, собирая информацию о 500 млн подключенных устройствах и услугах ежемесячно. Естественно, в его базу попадает большое число уязвимого оборудования, в том числе с несменёнными паролями установленными по умолчанию и др.

Просто невероятно, чтó можно найти в Shodan с помощью простого запроса. Бесчисленные светофоры, камеры безопасности, домашние системы автоматизации, системы отопления – всё это подключено к интернету и легко обнаруживается. [22, 23]

Всё это оборудование интернета вещей генерирует постоянно растущие объёмы трафика (в том числе и HTTPS). Одновременно с этим (по данным телеметрии, собираемой браузером Firefox), 29 января 2017 года количество страниц, загруженных по протоколу HTTPS (от общего числа вместе с протоколом HTTP) превысило 50%[369], что вместе с тенденцией на «выдавливание» из поисковых систем сайтов работающих по открытому протоколу HTTP за счёт их неиндексирования и латентно-агрессивному поведению браузеров (благодаря внедрению HSTS с 2012 года[370]) по навязыванию HTTPS[371], ведёт к ситуации когда, с одной стороны, мы защищаемся от атак класса man-in-themiddle, а с другой, различный «вредоносный» и «криминальный» трафик, создаваемый «интернетом вещей», будет маскироваться под обычные зашифрованные HTTPS запросы, а в силу его закрытости, его будет сложно обнаружить и блокировать даже легальным антивирусным средствам. Как бороться с ростом криминала?

Фактически компьютерные сети поднимают новые правовые проблемы, когда они взаимодействуют со старыми законами. Например, электронные азартные игры. Компьютеры в течение многих десятилетий моделировали различные процессы и позволяли работать удалённо, так почему бы не моделировать игровые автоматы, рулетку, дилеров блэк джека и другое игорное оборудование? Ну, потому что это во многих местах незаконно. Но в большом количестве других мест (Англия, например) азартная игра является законной, и владельцы казино там оценили потенциал для азартной игры через интернет. Что происходит, если игрок, казино и сервер – всё в разных странах, с противоречивыми законами? Хороший вопрос [5, стр. 32].

Неоднозначно можно прочитать и утверждённую приказом министра промышленности и энергетики Российской Федерации (бывш. Б. В. Христенко) от 7 августа 2007 г. № 311 **Стратегию развития электронной промышленности России на период до 2025 года** [372]:

«*Внедрение нанотехнологий должно ещё больше расширить глубину их проникновения в повседневную жизнь населения. Должна быть обеспечена постоянная связь каждого индивидуума с глобальными информационно-управляющими сетями типа Internet.*

[369] Доля HTTPS трафика в интернете превысила 50%, http://habrahabr.ru/post/321002/

[370] HTTP Strict Transport Secutiry или как обезопасить себя от атак «man-in-the-middle» и заставить браузер всегда использовать HTTPS, http://habrahabr.ru/post/216751/

[371] Удаление принудительного HTTPS запроса к сайту http://habrahabr.ru/sandbox/95217/

[372] http://www.minpromtorg.gov.ru/ministry/strategic/sectoral/11,
http://www.minpromtorg.gov.ru/ministry/strategic/sectoral/11/Strategiya_do_2025.doc,
http://base.consultant.ru/cons/cgi/online.cgi?req=doc;base=LAW;n=99457.

Наноэлектроника будет интегрироваться с биообъектами и обеспечивать непрерывный контроль за поддержанием их жизнедеятельности, улучшением качества жизни, и таким образом сокращать социальные расходы государства.

Широкое распространение получат встроенные беспроводные наноэлектронные устройства, обеспечивающие постоянный контакт человека с окружающей его интеллектуальной средой, получат распространение средства прямого беспроводного контакта мозга человека с окружающими его предметами, транспортными средствами и другими людьми. Тиражи такой продукции превысят миллиарды штук в год из-за её повсеместного распространения».

В заключение заметим, что остановить технический прогресс запретами невозможно. Юридическая наука и законодательство отставали, отстают и будут отставать от жизни.

Например, сейчас ведутся попытки защиты интеллектуальной собственности в интернете, «борьба с пиратством», защита детей от нежелательной информации и прочее (подробнее см. главу 3), и это правильно, но в большинстве случаев запрещать технологии – не эффективная мера, так как это или борьба со следствием, или паллиатив. Самая эффективная борьба – это просветительская работа, то есть образование. Науку и новые технологии следует использовать с умом, разъясняя школьникам и взрослым, что такое «хорошо» и что такое «плохо». Первичный тормоз «нельзя» или «не стоит» должен закладываться в голове у каждого в процессе воспитания и образования, нежели быть пустым законом на бумаге. Использование технологий для обеспечения жизни в стиле «сверхпотребления» – плохо, неразумно. Интернет – лишь наглядное техническое отображение процессов, происходящих в обществе, в голове каждого из нас.

Как вы думаете, что правильнее: провести во все школы бесплатный Wi-Fi интернет или поставить «глушилки» сотовой связи, запретив школьникам быть заложниками своих «гаджетов»?

ГОСТ Р12.4.026–2001 знак Р 18

В ответ на этот риторический вопрос, впервые заданный читателям во втором издании, хотелось бы добавить следующее по теме:

С 1 сентября 2018 года ученикам начальных и средних школ во Франции запрещено пользоваться мобильными телефонами. И по новому законопроекту гражданам до 16 лет также запрещено заводить аккаунты в социальных сетях без согласия родителей. Причём, разрешение родителей должно быть обязательно письменным! [19]

Вводимый возрастной ценз на заведение профиля в социальной сети преследует двоякую цель. С одной стороны, закон предполагает обязательства для соцсетей. При нарушении новых возрастных правил соцсети станут наказывать штрафом в 20 миллионов евро или на сумму, равную четырём процентам общего годового дохода интернет-корпорации. С другой – речь идёт о том, чтобы оградить подрастающее поколение

от негативных последствий «цифровой революции» – морального насилия, киберунижения и террористических угроз. [19]

В последнее время даже ряд экс-руководителей Facebook стали предупреждать мировую общественность о том, чтобы детям был ограничен доступ к социальным сетям. Бывший вице-президент компании Чамат Палихапития, выступая в Высшей бизнес-школе в Стэнфорде, заявил, что в основе соцсетей лежит принцип использования человеческой психологии и формирования зависимости. По его словам, социальные сети стали «инструментами, которые подрывают социальную структуру общества». Сам он сократил выход в соцсети, а своим детям вообще запретил их использование. Примерно в таком же критическом духе высказался один из основателей компании (вместе с Марком Цукербергом) Шон Паркер. «Только одному Богу известно, что Facebook выделывает с мозгом наших детей», – заявил он. (Социальные сети «ВКонтакте», «Одноклассники» и другие оказывают аналогичное влияние на свою аудиторию – *прим. авт. уч.*) [19]

Как недавно сообщила английская газета «Гардиан», действующая верхушка Facebook своих детишек обучает в элитных школах Силиконовой долины, «где айфоны, айпады и переносные компьютеры находятся под абсолютным запретом».[19]

Мнение министра образования и науки России Ольги Васильевой: «*Рассматривать ли айфон как помощь на уроке – вопрос сложный. Моё убеждение, что он мешает.*»[20]

Однако, не дожидаясь мнений и указания минобрнауки, школьные учебные заведения страны действуют самостоятельно. Формально в российских школах мобильные телефоны (и иные устройства – *прим авт. уч.*) принести можно, а вот пользоваться ими – открытый вопрос. Каждая школа может установить собственные правила, так во многих школах большинства регионов действует запрет на телефоны во время уроков. Есть регионы, где местные власти готовы и готовятся запретить мобильники, наример в Сибири и на Северном Кавказе. В Чеченской Республике запрет на пользование мобильными телефонами в школах уже действует. С 2017 года несколько школ Владимира, Ярославля, Тюмени и других городов ввели полный запрет на мобильные в школах во время уроков.[20]

6.2. Компьютерная сеть

Компьютерная сеть – система связи компьютеров или компьютерного оборудования (серверы, маршрутизаторы и другое оборудование).

«ЭВМ-ная сеть» – несуществующий термин, см. «Сеть ЭВМ» или «ЛВС».

Лока́льная вычисли́тельная сеть (ЛВС, локальная сеть; англ. Local Area Network, LAN, сеть ЭВМ) – компьютерная сеть, покрывающая обычно относительно небольшую территорию или небольшую группу зданий (дом, офис, фирму, институт) [373].

Вычислительная сеть – зачастую то же самое, что и компьютерная сеть.

Интранет [374] – локальная сеть, построенная с использованием Internet-технологий (протоколов из сетевой модели TCP/IP), при этом не имеющая подключения к сети Ин-

[373] Также существуют локальные сети, узлы которых разнесены географически на расстояния более десятка тысяч километров (космические станции и орбитальные центры). Несмотря на такие расстояния, подобные сети всё равно относят к локальным.

[374] От англ. Intranet, синоним «интрасеть», термин чаще используется в английском написании без перевода.

тернет. Например, внутренняя частная сеть организации. Как следствие все приложения в Intranet основаны на применении интернет-технологий и в особенности веб-технологии: гипертекст в формате HTML, протокол передачи гипертекста HTTP и интерфейс серверных приложений CGI. Составными частями Intranet являются веб-сéрверы для статической или динамической публикации информации и браузеры для просмотра и интерпретации гипертекста.

Изучение компьютерных сетей – это вопрос отдельной книги (области знаний), поэтому мы ограничимся лишь приведением классификаций существующих сетей для общего понимания вопроса, а также кратко приведём выборочные моменты, необходимые для понимания нижеизложенного материала касаемо сети Интернет.

6.2.1. Классификации компьютерных сетей

- По территориальной распространённости: PAN (Personal Area Network), LAN (Local Area Network), CAN (Campus Area Network), MAN (Metropolitan Area Network), WAN (Wide Area Network).
- По типу функционального взаимодействия: Клиент–сервер, Смешанная сеть, Одноранговая сеть, Многоранговые сети.
- По типу сетевой топологии: Шина, Кольцо, Двойное кольцо, Звезда, Ячеистая, Решётка, Дерево, Утолщённое дерево (Fat Tree).
- По типу среды передачи: проводные[375], беспроводные.
- По функциональному назначению: Сети хранения данных, Серверные фермы, Сети управления процессом, Сети SOHO, домовые сети.
- По скорости передач (на момент издания): низкоскоростные (до 10 Мбит/с), среднескоростные (до 100 Мбит/с), высокоскоростные (1 Гбит/с и выше).
- По используемым сетевым моделям.
- По другим критериям: используемым сетевым операционным системам на узлах, необходимости поддержания постоянного соединения и прочему.

6.2.2. Сетевые (эталонные) модели

Имея большое число компьютеров в сети, важно обеспечить не только совместимость компьютеров между собой, но и расширяемость сети с учётом её будущего развития. Задача непростая, так как жизнь не стоит на месте. Для этой цели создаются (придумываются) так называемые «сетевые модели», ещё их называют эталонными (или архитектурными, так как они описывают архитектуру сети). Имея описание того, к чему стоит стремиться, различные независимые производители получают возможность производить различные сетевые устройства, писать программное обеспечение.

[375] Правильнее говорить «кабельные», потому как диэлектрическую среду передачи оптических сигналов в волокно-оптических кабелях с трудом можно назвать «проводом».

Наиболее известными являются две эталонные модели ЭВМОС[376] и модель интернета (TCP/IP). Несмотря на то что протоколы, связанные с эталонной моделью OSI, сейчас не используются, сама модель до сих пор весьма актуальна, а свойства её уровней очень важны. В эталонной модели TCP/IP всё наоборот: сама модель сейчас почти не используется, а её протоколы являются едва ли не самыми распространёнными [5, стр. 57]. Исходя из этого, мы рассмотрим то, что используется по факту большинством администраторов – нечто среднее, помесь обеих моделей.

Существуют и другие менее известные стандартные модели протоколов, под которые разработано несколько стеков: IBM System Network Architecture (SNA), Digital DECnet™, Novell NetWare, Apple AppleTalk.

Замечание. Необходимость рассмотрения эталонных моделей вызвана желанием авторов попытаться структурировать все последующие знания, передаваемые читателям касаемо сетей и интернета в отношении указанных моделей, которые можно рассматривать как некоторый скелет или как базовые знания, необходимые для систем среднего и высшего образования. В последующем мы неоднократно будем ссылаться к различным сетевым уровням. В случае появления у вас желания к более глубокому изучению сетевых технологий вам будет проще перейти к самостоятельному изучению специализированной сетевой и компьютерной литературы.

Эталонная модель OSI считается классикой сетестроения и содержит 7 уровней. Отчего её часто называют семиуровневой. При создании такого числа уровней была идея разделить функционал между уровнями. Практически любой хороший сетевой специалист знает их названия и предназначение каждого уровня.

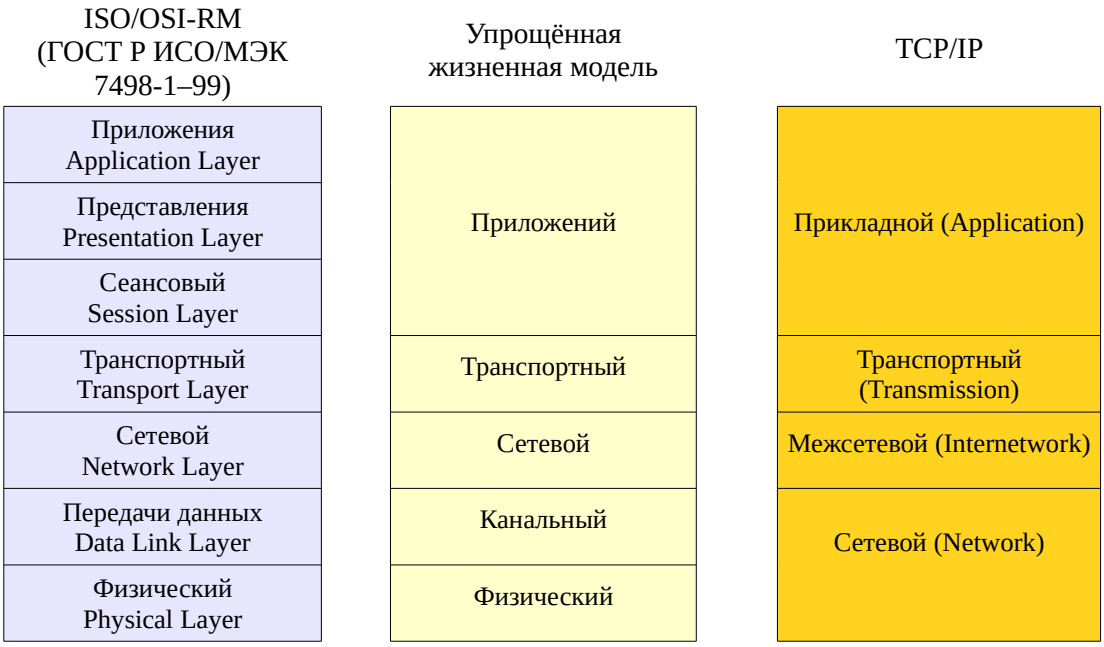


Рисунок 6.6. Соотнесение уровней сетевых моделей

[376] Эталонная модель взаимодействия открытых сетей, она же ISO/OSI-RM – от англ. Open Systems Interconnection Relationship Model, постфикс -RM нередко опускают, описывая то же самое как Open systems interconnection basic reference model. Она же описана в ГОСТ Р ИСО/МЭК 7498-1-99.

Сложного в этом ничего нет, при наличии опыта знания прочно оседают. Если сравнить сетевую модель с работой метрополитена, то выяснится, что для обеспечения перемещения пассажиров используется большое число различных специалистов и техники, начиная от ремонтников путей, которых мы практически никогда не видим (так как чаще всего они работают ночью, когда мы спим), и заканчивая контролёрами на проходах и у эскалаторов с машинистами электропоездов, которые у всех на виду. Это всё люди с разными специальностями, поэтому заменить одного работника другим не всегда предоставляется возможным, впрочем, как и работа одних без других. Также и в работе сети используются различные протоколы, каждый из которых можно отнести к тому или иному уровню сетевой модели, на плечи которого возложено выполнение тех или иных функций.

Сильные стороны моделей: OSI (пожалуй, за исключением уровней представления и сеансового) оказалась исключительно полезной для обсуждения компьютерных сетей, TCP/IP – протоколы, которые широко использовались много лет и продолжают использоваться. Как соединяются модели, показано на рис. 6.6. Чтобы совместить лучшие качества и упростить восприятие читателям, мы будем использовать в книге гибридную модель. Коротко уровни выполняют следующие задачи:

Задача физического уровня – определить выбор среды передачи, провод, оптику, радиосигнал и прочее и, как следствие, определить соответствующие способы кодирования сигналов («0» и «1»).

Задача канального уровня – повысить надёжность передачи данных, создав виртуальный канал с заданным уровнем ошибок между двумя компьютерами. Если данные «не проходят» (замирания радиоканала, электрические помехи в проводах), то прозрачно для пользователя, за счёт протокола этого уровня, происходит повторная отправка.

Задача сетевого уровня – объединить многочисленные каналы в сети и сети сетей, а также в объединённые сети, чтобы мы могли посылать пакеты между удалёнными компьютерами. На этом уровне появляется IP-адрес как идентификатор компьютера в сети. Также на сетевом уровне существует «разборка» и «сборка» пакетов из кадров канального уровня, размер поля данных которых меньше.

Транспортный уровень повышает надёжность доставки сообщений сетевого уровня, а также может гарантировать последовательность доставки, создавая сессии или сеансы связи между узлами. Протокол TCP – важный пример протокола транспортного уровня.

Наконец, уровень приложений, или прикладной уровень, содержит привычные нам программы, которые используют сеть: интернет-браузер, электронная почта, программа обмена мгновенными сообщениями и прочее.

6.2.3. Заключение, используемые термины

Современные сети построены по многоуровневому принципу. Чтобы организовать связь двух компьютеров, требуется сначала создать набор правил их взаимодействия, определить язык их общения, то есть определить, что означают посылаемые ими сигналы, и т. д. Эти правила и определения называются протоколом.

Протокол – набор правил и соглашений для передачи и приёма сообщений между одинаковыми уровнями разных сетевых узлов.

Любой протокол состоит из двух частей: логической и процедурной стороны. Логическая часть определяет форматы посылаемых пакетов (кадров, дейтаграмм). Например, это может быть стоимость билета в кассе и описание всех возможных билетов, что вы можете купить: на 1 поездку, 2, 5, 10 и на месяц. Процедурное описание определяет, какие пакеты будут первыми и что может ответить сервер в ответ на тот или иной запрос, ту или иную ситуацию. Например, при продаже билетов кассир здоровается с вами, выслушивает, какой билет вы хотите купить, объявляет его стоимость, получает деньги, возвращает билет со сдачей либо запрашивает дополнительную сумму, если переданных денег оказалось недостаточно. По завершении продажи желает вам удачной поездки или дня. Если вы ведёте себя неадекватно – связывается со стражами правопорядка, подав два красный свистка вверх. Как вы понимаете, при правильном описании это будет что-то вида «стулья вечером, деньги утром» или «стулья утром, деньги вечером», но никак не наоборот. Непродуманные протоколы чаще всего являются источниками сетевых атак и «дырками» для распространения компьютерных вирусов.

Вместо логической и процедурной частей протокола его можно более подробно делить на три части (см. рис. 6.7):

- синтаксическую (формат данных и программного кода);
- семантическую (управление информацией и обработка ошибок);
- временную (очерёдность или согласование по времени и последовательности).

Работу сети обеспечивает множество различных протоколов: например, протоколы управления физической связью, установления связи по сети, доступа к различным ресурсам и т. д.

Так как каждый уровень «живёт своей жизнью», для обозначения взаимодействия соседних сетевых уровней на одном компьютере между собой (направления вверх и вниз на рис. 6.6) вместо слова «протокол» используется термин «***интерфейс***». Интерфейсы существуют только к ближайшему верхнему и ближайшему нижнему уровням. «Перескакивание» через уровни не принято, за исключением инкапсулирующих и туннелирующих протоколов.

Рисунок 6.7. Три составные части протокола [377]

В заключение ответим на каверзный вопрос, который многих сетевых специалистов приводит в состояние лёгкой задумчивости, а именно ***в чём различие терминов: архитектура, структура и топология сети?***

Архитектура – это используемая модель, например ISO/OSI-RM.

Топология – способ соединения компьютеров между собой, как было предложено выше: Шина, Кольцо, Двойное кольцо, Звезда, Ячеистая, Решётка, Дерево, Утолщённое дерево (Fat Tree), смешанная и др.

Структура – это распределение функциональных обязанностей между компьютерами сети. Например, кто будет клиентом, а кто сервером. «Кто начальник, а кто под-

[377] http://dev.opera.com/articles/view/http-basic-introduction/.

чинённый». Понятно, что в случае объединения компьютеров по звёздной топологии и использовании архитектуры TCP/IP все компьютеры будут одинаковы, но если вы какой-то один из них выделите для хранения общих файлов, а к другому подключите принтер, то вот тут вы и увидите структуру, какой-то компьютер превратится в «файловый сервер», а какой-то в «сервер печати». При одинаковом или равноранговом подключении устройств друг к другу (сетевые коммутаторы не в счёт) на физическом уровне структура сети уже не будет равноранговой.

6.2.4. Документы RFC (Request For Comments), draft

Попытаемся ответить на вопрос: как получается создавать и скоординированно развивать различные вышеописанные модели, или кто есть кто в мире стандартов интернета?

Всемирная сеть Интернет имеет свой механизм стандартизации, значительно отличающийся от ITU-T и ISO. В двух словах основное отличие заключается в том, что сотрудники ITU и ISO носят деловые костюмы, тогда как стандарты интернета разрабатывают в основном люди в джинсах (ну, кроме тех, кто работает в Сан-Диего – на них надеты шорты и футболки с коротким рукавом) [5, стр.97].

На совещания ITU-T и ISO собираются администраторы корпораций и государственные гражданские служащие, для которых стандартизация является их работой. Они считают, что стандартизация – Очень Нужная Вещь, и посвящают ей свою жизнь. Для людей интернета, напротив, анархия является делом принципа, однако когда сотни миллионов делают какое-то общее дело, иногда им всё же приходится о чём-то договариваться, чтобы хоть что-то работало. Волей-неволей стандарты оказываются необходимыми [5, стр. 97]. В разработке стандартов интернета можете принять участие и вы, читатель. О том, как это сделать, расскажем подробнее.

Когда была запущена сеть ARPANET, Министерство обороны США создало неофициальный комитет для наблюдения за сетью. В 1983 году этот комитет был переименован в Совет по деятельности интернета (Internet Activities Board, IAB). Перед советом были поставлены несколько расширенные задачи, а именно удерживать исследователей, включённых в проекты ARPANET и интернет, в более-менее одном направлении, что напоминало попытку выпаса стада кошек. Значение сокращения IAB было затем изменено на Совет по архитектуре интернета (Internet Architecture Board) [5, стр.98].

Каждый из приблизительно десяти членов IAB возглавлял специальную комиссию по отдельному важному вопросу. Совет по архитектуре интернета собирался несколько раз в год для обсуждения результатов работы и предоставления отчёта Министерству обороны и NSF, которые в то время осуществляли основное финансирование в этой области. Когда требовался какой-либо стандарт (например, новый алгоритм маршрутизации), члены совета прорабатывали этот вопрос, после чего объявляли об изменениях аспирантам, занимавшимся реализацией программного обеспечения сетей. Стандарты оформлялись в виде набора технических отчётов, называемых RFC (Requests for Comments). RFC доступны в интернете для всех желающих [378]. Они пронумерованы в хронологическом порядке их создания. На

[378] http://ietf.org/rfc. Язык документов RFC английский, но при желании в Сети можно найти их многочисленные неофициальные переводы на русский язык.

сегодняшний день существует более 6000 этих документов [5, стр.98]. Хорошие сетевые администраторы знают про эти документы и обязательно читали хотя бы один из них.

К 1989 году интернет вырос настолько, что подобный неформальный подход к его стандартам перестал работать. К тому моменту многие производители предлагали продукцию на основе протокола TCP/IP и не хотели их менять просто потому, что десятку исследователей пришла в головы одна хорошая идея. Летом 1989 года IAB был снова реорганизован. Исследователи были переведены в группу исследования интернета (Internet Research Task Force, IRTF), подконтрольную IAB, и в группу проектирования интернета (Internet Engineering Task Force, IETF). В совете IAB появились люди, представляющие более широкий спектр организаций, чем исследовательское сообщество. Вначале это была группа, в которой члены работали в течение двух лет, после которых сами назначали своих преемников. Затем было создано «общество интернета» (Internet Society), в которое вошли люди, заинтересованные в интернете. Таким образом, интернет-сообщество в каком-то смысле сравнимо с Ассоциацией по вычислительной технике (ACM, Association for Computing Machinery) или IEEE. Оно управляется избираемыми доверенными лицами, которые утверждают состав IAB.[5, стр.98]

Идея этого разделения заключалась в том, чтобы сосредоточить IRTF на долгосрочных исследованиях, а IETF – на краткосрочных инженерных вопросах. Проблемная группа IETF была разделена на рабочие группы, каждая из которых решала свою задачу. Первое время председатели рабочих групп встречались друг с другом в составе руководящего комитета для координации совместных исследовательских усилий. Рабочие группы занимались такими вопросами, как новые приложения, информация для пользователей, OSI-интеграция, маршрутизация и адресация, безопасность, управление сетью и стандарты. В конце концов, было сформировано так много рабочих групп (более 70), что их сгруппировали по областям, после чего в руководящем комитете стали собираться председатели областей [5, стр.98].

Кроме того, был принят более формальный процесс стандартизации по аналогии: с процедурой, принятой в ISO. Чтобы стать предлагаемым стандартом, основная идея должна быть полностью изложена в RFC и представлять достаточный интерес, гарантирующий её рассмотрение. Затем, чтобы стать проектом стандарта, должна быть создана работающая реализация, которую нужно тщательно протестировать минимум двумя независимыми сайтами в течение 4 месяцев. Если IAB уверен, что идея здравая и программное обеспечение работает, он может объявить RFC стандартом интернета. Некоторые стандарты интернета стали стандартами Министерства обороны США (MIL-STD), что сделало их обязательными к применению поставщиками министерства [5, стр.99].

Консорциум World Wide Web (W3C) развивает протоколы и направляющие линии для веб-стандартов, чтобы облегчить долгосрочный рост Сети. Это промышленный консорциум во главе с Тимом Бернерсом-Ли, организованный в 1994 году, когда Web действительно начал стремительно расти. Теперь W3C имеет более 300 участников со всего мира и произвёл больше 100 Рекомендаций W3C, как называют его стандарты, затрагивая такие темы, как HTML и веб-безопасность [5, стр.99].

В настоящее время первичной публикацией документов RFC занимается IETF под эгидой открытой организации «Общество Интернета» (англ. Internet Society, ISOC). Правами на RFC обладает именно «Общество Интернета» [http://ru.wikipedia.org/wiki/RFC].

Тимоти Джон Бернерс-Ли (1955 г. по наст. вр.) – британский учёный, изобретатель URI, URL, HTTP, HTML, изобретатель Всемирной паутины (совместно с Робертом Кайо).

Формат RFC появился в 1969 году при обсуждении проекта ARPANET. RFC 1 был опубликован 7 апреля 1969 г. и назывался «Host Software». Первые RFC распространялись в печатном виде на бумаге в виде обычных писем, но уже с декабря 1969 г., когда заработали первые сегменты ARPANET, документы начали распространяться в электронном виде.

Большинство ранних RFC были созданы в Калифорнийском университете Лос-Анджелеса и Стэнфордском исследовательском институте.

С 1969 по 1998 г. бессменным и единственным редактором RFC был Джон Постел. После его смерти «Общество Интернета» (ISOC) поручило редактирование и публикацию RFC Институту информационных наук Университета Южной Калифорнии.

Очерк истории RFC за 30 лет с 1969 по 1999 г. представлен в RFC 2555.

Несмотря на название, запросы на отзывы RFC сейчас рассматриваются как стандарты интернета (а рабочие версии стандартов обычно называют драфтами, от англ. draft здесь – проект). Согласно RFC 2026, жизненный цикл стандарта выглядит следующим образом:

1. Выносится на всеобщее рассмотрение интернет-проект (Internet Draft). Проекты не имеют официального статуса и удаляются из базы через шесть месяцев после последнего изменения.
2. Если проект стандарта оказывается достаточно удачным и непротиворечивым, он получает статус предложенного стандарта (Proposed Standard) и свой номер RFC. Наличие программной реализации стандарта желательно, но не обязательно.
3. Следующая стадия – проект стандарта (Draft Standard) – означает, что предложенный стандарт принят сообществом, в частности существуют две независимые по коду совместимые реализации разных команд разработчиков. В проекты стандартов ещё могут вноситься мелкие правки, но они считаются достаточно стабильными и рекомендуются для реализации.
4. Высший уровень – стандарт интернета (Internet Standard). Это спецификации с большим успешным опытом применения и зрелой формулировкой. Параллельно с нумерацией RFC они имеют свою собственную нумерацию STD. Список стандартов имеется в документе STD 1 (сейчас это RFC 5000, но нумерация может измениться). Из более чем трёх тысяч RFC этого уровня достигли только несколько десятков.
5. Многие старые RFC замещены более новыми версиями под новыми номерами или вышли из употребления. Такие документы получают статус исторических (Historic).

Практически все стандарты Глобальной сети существуют в виде опубликованных заявок RFC. Но в виде документов RFC выходят не только стандарты, но также концепции, введения в новые направления в исследованиях, исторические справки, результаты экспериментов, руководства по внедрению технологий, предложения и рекомендации по развитию существующих Стандартов и другие новые идеи в информационных технологиях:

Экспериментальные (Experimental) спецификации содержат информацию об экспериментальных исследованиях, интересных для интернет-сообщества. Это могут быть, например, прототипы, реализующие новые концепции.

Информационные (Informational) RFC предназначены для ознакомления общественности, не являются стандартами и не являются результатом консенсуса или рекомендациями. Некоторые проекты, не получившие статуса Предложенного стандарта, но представляющие интерес, могут быть опубликованы как Информационные RFC.

Лучший современный опыт (Best Current Practice). Эта серия RFC содержит рекомендации по реализации стандартов, в том числе от сторонних организаций, а также внутренние документы о структуре и процедурах стандартизации.

Почти все стандарты разрабатываются под эгидой каких-либо научных или интернет-организаций (например W3C, IETF, консорциум Юникода, Интернет2).

Поскольку строгих требований к оформлению RFC нет, а пишут их разные живые люди, не лишённые чувства юмора, можно встретить как документы, написанные в строгом академическом стиле, так и иные – в дружеской неформальной манере. Существуют и первоапрельские шуточные RFC, например, RFC 1149 рассказывает о передаче IP-пакетов с помощью почтовых голубей.

6.3. Интернет

Интернет, **интернет**, **сеть Интернет**, **Сеть** [379] – глобальная компьютерная сеть, построенная на базе стека протоколов TCP/IP, объединяющая компьютеры тысяч региональных сетей и миллионов пользователей всего мира.

С функциональной точки зрения, интернет – новая коммуникационная платформа, соединяющая людей и организации в самых разных сферах жизни и деятельности. С другой стороны, не что иное, как обычная компьютерная сеть, только большая.

Интернет обеспечивает пользователям ПК доступ к информации, размещённой на различных интернет-серверах. В силу своего большого размера позволяет людям всего мира получать необходимую им информацию и общаться друг с другом.

При числе жителей Земли более 7,3 млрд в 2015 году, по данным опроса Всероссийского центра изучения общественного мнения (ВЦИОМ[380]), проведённого в октябре 2014 г., интернетом пользуются 66% граждан России от 18 лет и старше или 76,3 млн человек. С учётом детей и подростков эта цифра может оказаться выше. В мировом масштабе оценивается, что интернетом пользуются около 39% населения. Для оценки динамики роста, по данным ФОМ'а[381] на конец 2007 года, в России насчитывалось 35 млн активных пользователей интернета.

6.3.1. История интернета

История глобальных сетей началась после Второй мировой войны, однако первопричины, ускорившие происходившие события, появились несколько ранее. Страны, подговаривавшие Гитлера начать войну, хотели быстрого передела мира, но несколько просчитались. Несмотря на скрытую финансовую поддержку нашего противника, война затянулась. Наши деды сражались за наше будущее, за идею жить свободно, поэтому победить их было невозможно. Как говорилось тогда: за нами Москва, отступать некуда. Американский президент понимал безвыходность ситуации, складывающейся для США по итогам Второй мировой войны.[21] Не имело значения, кто победит, Гитлер или Сталин. Победитель оккупировал бы Европу, Китай, Индию и в итоге Евразию. Геополитическое правило Макиндера: «*Кто правит центральным регионом, тот правит островом мира (Евразией). Кто правит островом мира, тот правит и самим миром*». Власть над Евразией означает власть над мировыми торговыми путями, то есть, по сути, над планетой [12].

Рузвельт осознал угрозу, и начались поиски «штуковины», которая остановит победителя. Оной оказалась ядерная бомба. Ядерная атака на Японию в августе 1945-го

[379] Internet – от англ. Interconnected networks, связанные сети. Несмотря на то что во многих изданиях изначально слово «Интернет» писалось со строчной буквы, как у нарицательного имени, многие носители русского языка с этим были не согласны и продолжали писать слово «интернет» с прописной (маленькой) буквы. Обе стороны были примирены решением Орфографической комиссии РАН, предложившим, в соответствии с практикой письма, оба варианта написания – с прописной и со строчной буквы – считать правильными, что можно увидеть в 4-м издании «Русского орфографического словаря» РАН (2012).

[380] Официальный сайт «Всероссийского центра изучения общественного мнения» – http://wciom.ru.

[381] «Фонд общественного мнения», официальный сайт – http://runet.fom.ru/.

не имела военного смысла. Хиросима была рекламной акцией, рассчитанной на одного зрителя – Сталина. Имея под ружьём более чем десятимиллионную армию победителей-фронтовиков, он смёл бы любую силу на своём пути. Бомба, по словам Оппенгеймера, «вспыхнувшая ярче тысячи солнц», остановила СССР в границах завоёванного [12].

Сталин полагал: капиталисты, понимая ущербность своей системы, никогда не пойдут на честную конкуренцию, что мы сейчас и наблюдаем. Спасая своё положение, они будут искать войны с советской Россией. Их сдержит только одно: если война с СССР для Запада будет равносильна самоуничтожению. Логика генерального развития СССР – вооружаться. В навязанных условиях мощь оружия гарантирует мир, поэтому мы довольно скоро тоже обзавелись ядерным оружием, а потом, первые, и термоядерным.

Руководство США это не протрезвило, оно готовилось к войне, поэтому в конце 50-х годов, в самый разгар холодной войны, Министерство обороны США пожелало иметь сеть, которая могла бы пережить даже ядерную войну. В то время все военные телекоммуникации базировались на общественной телефонной сети, которая была сочтена слишком уязвимой. Графически эта уязвимость демонстрируется на рис. 6.8 *б*.

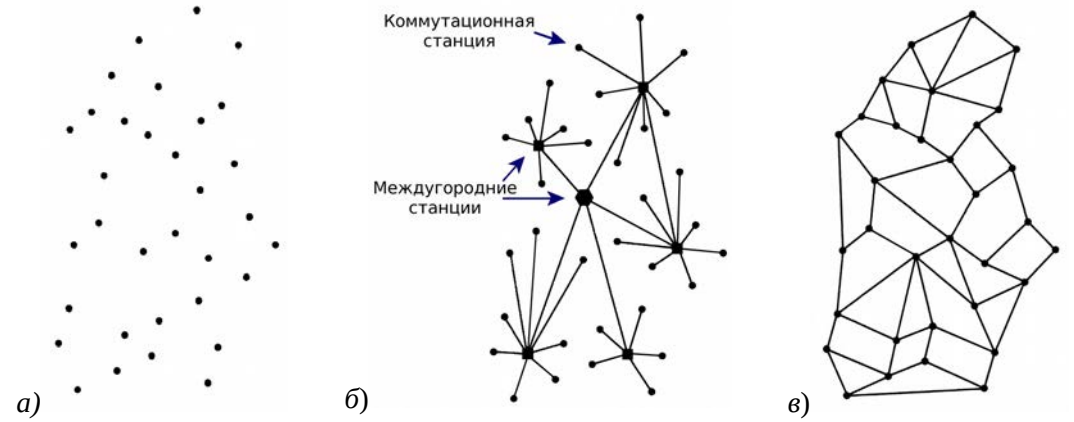


Рисунок 6.8. Расположение узлов (а); структура телефонной сети (б); предложенная Бэраном архитектура распределённой сети (в)

Здесь чёрными точками обозначены коммутационные станции, с каждой из которых были связаны тысячи абонентов. Эти коммутаторы, в свою очередь, являлись абонентами для станций более высокого уровня – междугородных. Междугородные станции формировали национальные сети. При этом степень резервной избыточности была минимальной. Уязвимость заключалась в том, что потеря всего одного ключевого коммутатора или междугородной станции разделила бы сеть на изолированные участки [5, стр. 70].

Для решения этой проблемы в 1960-х годах Министерство обороны США обратилось к корпорации RAND. Один из её работников, Пол Бэран (Paul Baran, 1926–2011), разработал проект высоконадёжной распределённой сети (см. рис. 6.8 *в*) [5, стр. 71].

Поскольку по линиям такой большой длины тяжело было бы передать аналоговый сигнал с допустимым уровнем искажений, Бэран предложил передавать цифровые данные и использовать технологию коммутации пакетов. Им было написано несколько отчётов для Министерства обороны, в которых описывались подробности реализации

его идей. Пентагону понравилась предложенная концепция, и компании AT&T (тогдашнему единственному в США монополисту в области телефонных сетей) было поручено разработать прототип. AT&T сразу же отклонила идеи Бэрана. Конечно, богатейшая и крупнейшая компания не могла позволить какому-то мальчишке указывать ей, как следует строить телефонные сети. Было заявлено, что бэрановскую сеть построить невозможно, и проект был на этом закрыт [5, стр. 71].

Прошло ещё несколько лет, но Министерству обороны США так и не было предложено никакой замены существующей оперативной системе управления. Чтобы понять, как развивались события дальше, мы вспомним октябрь 1957 года, когда в СССР был запущен первый в мире искусственный спутник Земли и тем самым мы получили преимущество над США в космосе. Тогда президент Эйзенхауэр задумался о том, кто же допустил такой прокол. И выяснилось, что армия, флот и ВВС США только зря проедают деньги, отпущенные Пентагоном на научные исследования. Было немедленно решено создать единую научную организацию под покровительством Министерства обороны, **ARPA** (Advanced Research Projects Agency, Управление перспективного планирования научно-исследовательских работ). У ARPA не было ни учёных, ни лабораторий. У неё вообще практически ничего не было, за исключением небольшого офиса и скромного (по меркам Пентагона) бюджета. ARPA занималась тем, что выделяла из множества предлагаемых университетами и компаниями проектов наиболее перспективные и организовывала получение грантов под эти проекты и заключение контрактов с этими организациями [5, стр. 71].

Первая исследовательская программа Агентства передовых исследовательских проектов в области обороны (DARPA) при Министерстве обороны США, посвящённая системе глобальной коммуникации, была начата 4 октября 1962 года. Возглавлявший программу Дж. Ликлайдер (J. C. R. Licklider) опубликовал работу «Galactic Network», в которой предсказывал возможность существования в будущем глобальной компьютерной связи между людьми, имеющими мгновенный доступ к программам и базам данных из любой точки земного шара.

В первые годы своего существования ARPA пыталась определиться с направлением своей деятельности. В 1967 году внимание Ларри Робертса, диспетчера программ в ARPA, который пытался выяснить, как обеспечить удалённый доступ к компьютерам, привлекли компьютерные сети. [5, стр. 72] Один из экспертов, Весли Кларк (Wesley Clark), предложил построить подсеть с коммутацией пакетов, идею которой Леонард Клейнрок (Leonard Kleinrock) впервые опубликовал в июле 1964 года. При коммутации пакетов передаваемая информация разбивается на части, к каждой части присоединяется заголовок, который содержит полную адресную информацию. При коммутации же каналов во время передачи информации пара компьютеров соединяется «один к одному».

В 1966 году DARPA приступило к разработке проекта компьютерной сети **ARPANET**. Проект имел сугубо военные цели, отвечающие задачам управления военными и гражданскими объектами в период войны. Принципиально важна была децентрализация, позволяющая сети функционировать даже при уничтожении нескольких узлов сети.

29 октября 1969 года, в 21:00 был успешно установлен сеанс связи. В процессе обмена данными создателям системы удалось успешно передать три символа – LOG, после чего случилась знакомая и сейчас многим из нас неприятность – связь оборвалась. Повторную связь удалось установить уже в 22:40. Оба компьютера,

использовавшиеся в системе при передаче данных, стали первыми узлами связи будущей сети **ARPANET**. [18] В декабре того же года была сдана в эксплуатацию экспериментальная сеть, объединявшая четыре узла со скоростью передачи данных 56 Кбит/с[382].

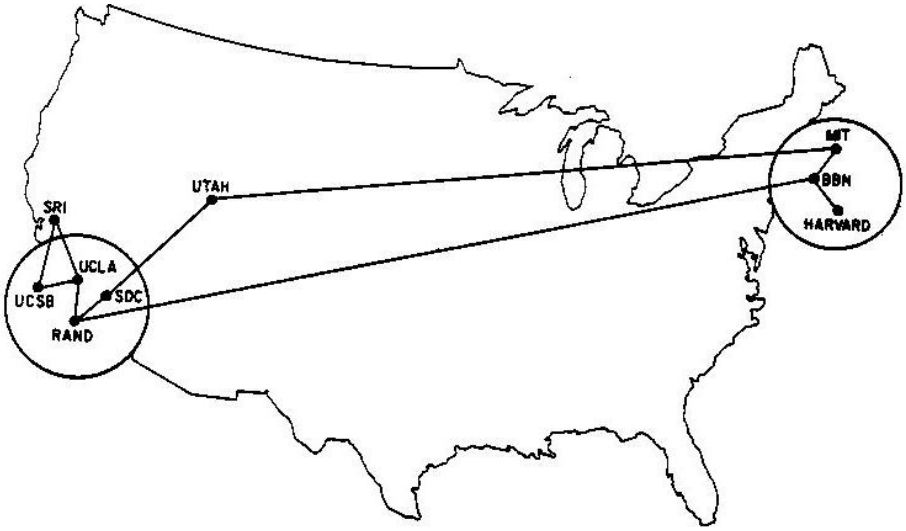


Рисунок 6.9 а Структура сети ARPANET (июнь 1970 г.)

В 1970-х годах сеть в основном использовалась для пересылки электронной почты, тогда же появились первые списки почтовой рассылки, новостные группы и доски объявлений.

В 1972 году – 37 узлов. К слову, в 1972 году появился и значок @, определив дальнейшую структуру e-mail адреса. Благодаря возможности удалённой передачи данных решалась сложная на тот момент задача переноса информации с одного ПК на другой. Это была сложная операция, в силу ряда причин, одной из которых являлась несовместимость вычислительных машин. [18]

Впоследствии, в 1973 году появились международные сетевые узлы, которые были расположены в Великобритании и Норвегии [18], позднее к сети **ARPANET** были подключены тысячи серверов и отдельных сетей (главным образом университетских и государственных), что привело к созданию так называемой «**ARPA Internet**» – прародительницы современной сети **Интернет** (см. рис. 6.9 *б*).

[382] Скорость канала в 56 Кбит/с определялась, исходя из использования уплотняющего оборудования на междугородних телефонных станциях. Под голос отводилась полоса канала тональной частоты – 0,3–3,4 кГц, а фильтры были настроены отрезать всё, что выше 4 кГц. Следуя теореме Котельникова (см. стр. 111), можно было указанный сигнал закодировать, взяв 8000 отсчётов в секунду, а каждый отсчёт квантовать 256 уровнями, что даёт 64 Кбит/с – максимальную теоретическую скорость передачи данных через телефонные линии. Учитывая, что телефонная связь не идеальная, скорость в 56 Кбит/с – очень хороший результат.

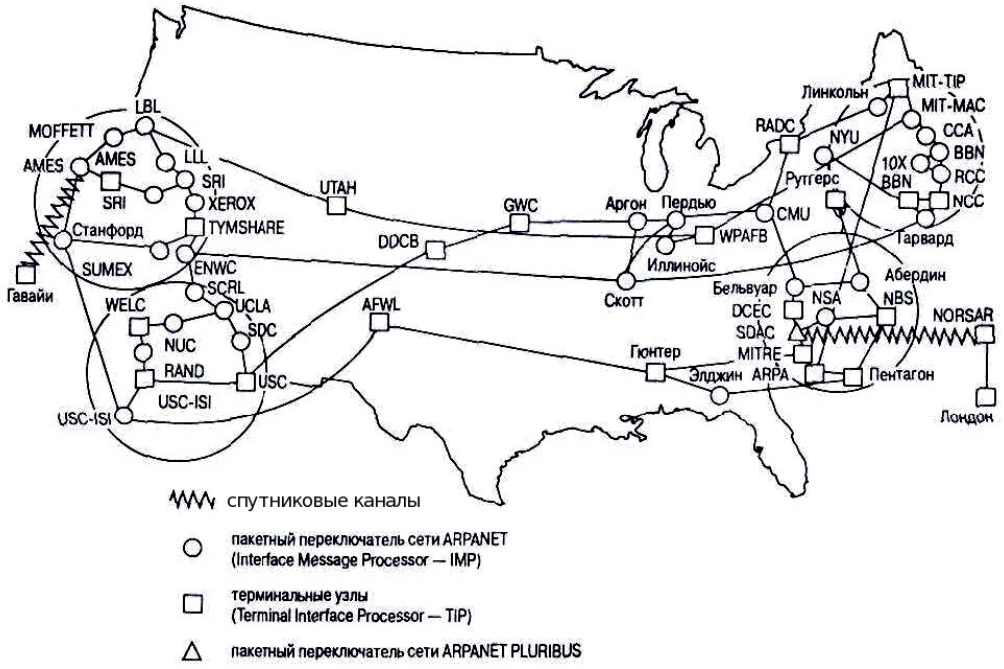


Рисунок 6.9 б. Структура сети ARPANET (1976 г.)

В 1983 году сеть **ARPANET** перешла с протокола NCP (Network Control Protocol) на TCP/IP (протокол управления передачей/инернет-протокол) как основу архитектуры. **ARPANET** сыграл важную роль в развитии интернета. Однако к тому времени стали возникать проблемы при расширении сетей, наиболее очевидными из которых являлись перегрузки на линиях связи. Помехой развития был также набор правил для пользователей сети, обязательный для всех (так называемая Acceptable Usage Policy – AUP). Согласно этому своду правил, запрещалось использование сети **ARPANET** в коммерческих целях.

В 1985 году Национальный научный фонд (National Science Foundation – **NSF**) приступил к созданию новой сети **NSFNET**. NSF финансировал создание сети, объединившей пять суперкомпьютерных центров, и предложил любым региональным и университетским компьютерным центрам, находившимся в пределах физической досягаемости от этой сети, подсоединиться к ней. Это был зародыш сети Интернет в нынешнем её виде. Целый ряд университетов подсоединились к сети **NSF**, чтобы получить доступ к суперкомпьютерам. Кроме исследовательских задач, сеть использовалась для передачи электронной почты, файлов и новостей.

Большое значение для развития интернета имела разработка международного стандарта локальных сетей Ethernet. Локальные сети произвели революцию, позволив существенно увеличить производительность труда работников управленческого аппарата, разработать и внедрить автоматизированные системы управления предприятиями. Но они были явно недостаточными для крупных корпораций, имеющих отделения в разных городах и даже странах. Поэтому вполне естественным стал их интерес к подключению своих локальных сетей к общенациональной сети **NSFNET**.

Всемирная паутина – World Wide Web (**WWW**, или просто Web) – появилась после разработки в 1990 году проекта гипертекстовой разметки документов. Автор разра-

ботки – Тим Бернер-Ли – программист Европейского центра ядерных исследований (CERN), расположенного в Женеве (см. стр. 608). В документе 1990 года рассматривались понятия, имеющие принципиальное значение для гипертекстовых документов: общий протокол обмена документами (протокол **HTTP**); общий документальный протокол для поставщиков информации и потребителей (язык разметки гипертекстов); поддержка поиска по индексу и возможность просмотра этих документов при помощи браузеров текста и графики.

Слово **Web** также используется для обозначения многих компонентов WWW: Web-браузер, веб-страница, Web-почта, Web-чат, Web-конференция и прочее. Всё чаще эти слова можно встретить в русской транскрипции: веб-браузер, веб-страница, веб-почта, веб-чат, веб-конференция и прочее.

В 1993 году **NSF** предложил радикальным образом изменить архитектуру сети:

- создать разветвлённую сеть точек доступа к сети (англ. Network Access Point, **NAP**), посредством которых провайдеры (поставщики интернет-услуг) могли бы осуществлять обмен трафиком;
- реализовать проект арбитража маршрутизации в сети (Routing Arbiter project – RA), что облегчило бы согласование правил работы в сети и управление адресами между несколькими провайдерами, подключенными к **NAP**;
- утвердить единого провайдера для службы обеспечения высокоскоростной магистральной сети (very high-speed Backbone Network Service – **vBNS**);
- обеспечить транзит трафика по существующим сетям в целях поддержки межрегиональных соединений путём подключения к провайдерам, которые подключены к **NAP** или напрямую к **NAP**. Любой избранный для этой цели провайдер должен иметь соединения, по крайней мере, с тремя **NAP**.

В 1994 году **NSF** объявил о строительстве четырёх **NAP** – в Сан-Франциско, Нью-Йорке, Чикаго и Вашингтоне.

Согласно терминологии NSF, точка доступа к сети NAP – это высокоскоростной коммутатор или сеть коммутаторов, к которой подключается определённое число маршрутизаторов для обмена трафиком. Любая NAP должна работать со скоростью не ниже 100 Мбит/с и иметь возможность повышения пропускной способности по запросу или в зависимости от нагрузки.

30 апреля 1995 года опорная сеть NSF была закрыта, а архитектура NAP превратилась в интернет.

6.3.2. Современная структура интернета

Современная сеть Интернет представляет собой объединение операторов интернет-услуг (провайдеров), у которых в нескольких регионах имеются узлы, называемые точками присутствия (Points Of Presence – **POP**). Все подключения клиентов к интернет-провайдерам осуществляются через серверы доступа или маршрутизаторы, расположенные на **POP** провайдера. Строгая иерархическая структура в интернете отсутствует.

Решением проблем функционирования интернета, разработкой новых технологий и стандартов Сети, распределением доменных имён занимается достаточно большой

круг общественных и научных организаций, история появления части которых была рассмотрена в разделе 6.2.4.

Ведущей среди них является специальная негосударственная некоммерческая организация, расположенная в США, – **Internet Society** (**ISOC**). Членами **ISOC** могут быть как частные лица, так и организации, но правом голоса на выборах в управляющие органы **ISOC** обладают только частные лица.

В рамках **ISOC** действуют несколько специальных групп:

1) **Совет по архитектуре интернета** – **IAB** (Internet Architecture Board) – занимается перспективами развития семейства протоколов TCP/IP: рассматривает различные идеи усовершенствования, принципы построения новых систем и прочее;
2) **Рабочая инженерная группа по интернету** – **IETF** (Internet Engineering Task Force) – занимается оперативным разрешением текущих проблем эксплуатации и развития, а также разработкой новых стандартов и протоколов;
3) **Центр сетевой информации – InterNIC** – занимается регулированием IP-адресов, доменных имён, справочных служб и хранением документов по стандартам интернета.

В 1998 г. функции **InterNIC** были переданы специально созданной организации **ICANN** (Internet Corporation for Assigned Names and Numbers), которая занимается вопросами стратегического регулирования назначения IP-адресов и доменных имён. Международной организацией оперативного управления распределением IP-адресов и доменных имён является **IANA** – Internet Assigned Numbers Authority (Уполномоченная организация по распределению нумерации в сети Интернет).

IANA определила Региональных регистраторов сети Интернет (Regional Internet Registry – **RIR**), обеспечивающих распределение пространства IP-адресов и назначения номеров автономным системам [383] в сети Интернет по всему земному шару:

- Американский реестр адресов сети Интернет (American Registry for the Internet Numbers – **ARIN**);
- Европейский сетевой координационный центр **RIPE NCC** [384];
- Азиатско-Тихоокеанский сетевой информационный центр (**APNIC**);
- Региональный центр Латинской Америки и Карибских островов (Regional Latin-American and Caribbean IP Address Registry – **LACNIC**).

Общее администрирование и техническое сопровождение российского национального домена (**.RU**) первоначально (в 1993 году) было поручено нейтральной организации – Российскому научно-исследовательскому Институту развития общественных сетей (**РосНИИРОС**, http://ripn.net).

7 апреля 1994 года **IANA** занесла в международную базу данных национальных доменов верхнего уровня российский национальный домен RU.

В 2001 году был учреждён российский Координационный центр национального домена сети Интернет.

В 2002 году Координационному центру переданы полномочия по выработке правил регистрации доменных имён в домене ***RU***, аккредитации регистраторов и разработке

[383] Грубо, между провайдерами интернета есть соединённые между собой автономные системы.
[384] Сокращение от фр. Réseaux IP Européens + англ. Network Coordination Centre.

перспективных проектов, связанных с развитием российского национального домена. В целях сохранения стабильности функционирования домена ***RU*** на **РосНИИРОС** возложены функции Технического центра домена ***RU***.

На прошедшей в Каире 2–7 ноября 2008 года 33-й конференции ICANN было принято Решение о выделении России кириллического домена верхнего уровня «.рф» (punycode: .xn--p1ai). 16 ноября 2009 года Координационный центр домена RU подал заявку на делегирование домена «.рф» в ICANN. Во избежание захвата популярных словосочетаний киберсквотерами, регистрация доменных имён открывалась в несколько этапов. Практическое использование зарегистрированных доменов началось с 12 мая 2010 года.

Список аккредитованных регистраторов доменов ***RU*** и ***РФ*** можно найти на сайте Координационного центра (http://*cctld.ru/ru/*), в апреле 2015-го было аккредитован 31 регистратор (http://cctld.ru/ru/registrators/, см. табл. 6.1), зарегистрировано доменов в зоне ru около ~4,9 млн, в зоне рф – 0,8 млн.

Структура российского сегмента интернета начала формироваться в 1991–92 годах (EUnet/Relcom, Sovam Teleport, Гласнет, FREEnet). В конце 90-х Ростелеком, почувствовав коммерческий интерес к интернет-рынку, создал первую магистральную IP-сеть РФ. До 2003 г. Ростелеком контролировал магистрального оператора интернет-доступа РТКомм.ру. В 2000–2001 году, на рынок магистрального провайдинга вышел ТрансТелеКом (**ТТК**) с молниеносно построенной более современной, более широкополосной и более разветвлённой сетью. Магистральных операторов стало два (см. рис. 6.10).

Таблица 6.1. Список аккредитованных регистраторов доменов ru и рф

Название	Адрес сайта	Город
ООО «101 домен Регистрация Доменов»	www.101domain.ru	Москва
ООО «АГАВА-хостинг»	www.hosting.agava.ru/domains/	Долгопрудный
ООО «АксельНейм»	http://axelname.ru	Санкт-Петербург
ООО «Ардис»	http://ardis.ru	Калининград (Калининградская обл.)
ОАО «ВымпелКом»	www.beeline.ru	Москва
ЗАО «Демос-Интернет»	https://reg.demos.ru	Москва
ООО «Имена Интернет»	http://webnames.ru	Самара
ЗАО «Караван-Телеком»	www.caravan.ru	Москва
ООО «Клевер Телеком»	www.regplanet.ru	Самара
ООО «Наунет СП»	www.naunet.ru	Москва
ООО «НЕТФОКС»	www.netfox.ru	Санкт-Петербург
ООО «Объединённые доменные имена»	http://uninic.ru	Москва
ООО «Персонал-Н»	http://personal-n.ru	Москва
ООО «Перспектива»	http://centralreg.ru	Наро-Фоминск
ООО «Регги Бизнес»	www.reggi.ru	Москва
ЗАО «Региональный сетевой информационный центр»	www.nic.ru	Москва
ООО «Регистратор доменных имён РЕГ.РУ»	www.reg.ru	Москва

ООО «Регистр 1»	http://registr1.ru/	Москва
ООО «Регистрант»	www.registrant.ru	Санкт-Петербург
ЗАО «Регистратор»	www.domenus.ru	Москва
ООО «Регистратор доменов»	www.mastername.ru	Москва
ЗАО «Регистратор Р01»	www.r01.ru	Москва
ООО «Регтайм»	www.webnames.ru	Самара
ООО «Рилтайм Реджистр Рус»	-	Москва
ОАО «РТКомм.РУ»	www.rtcomm.ru	Москва
ООО «РунетПроект»	runetproekt.ru	Наро-Фоминск
ООО «Степ Медиа»	www.sm-domains.ru	Москва
ООО «СэйлНэймс»	www.salenames.ru	Москва
ООО «ФОРМАТ»	www.regformat.ru	Москва
ООО «ЦЭТИС»	www.cetis-reg.ru	Москва
ЗАО «Элвис-Телеком»	www.getname.ru	Москва

Операторы интернет-доступа по функционально-географическому признаку разделяются на следующие типы:

- магистральные операторы (Internet Backbone Provider – **IBP**) – операторы межрегиональных IP-сетей, обеспечивающие перенос интернет-трафика между регионами страны и за рубеж;
- конечные провайдеры (интернет-сервис-провайдеры – **ISP**), обеспечивающие предоставление интернет-услуг конечным пользователям:
  - региональные **ISP –** действующие на региональном уровне;
  - локальные **ISP** – действующие на локальном уровне.

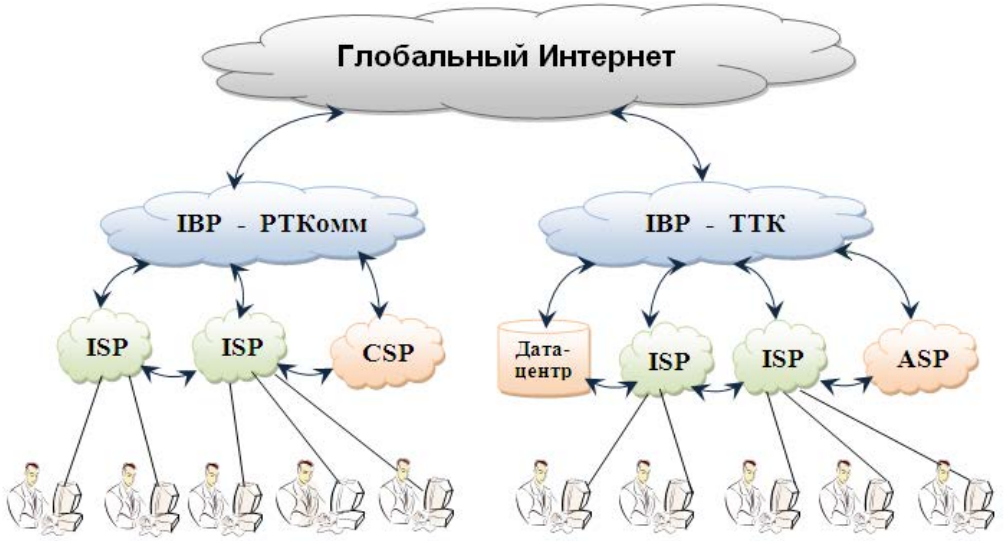


Рисунок 6.10. Структура интернет-провайдинга в России, 2001 г.

Обычному пользователю для получения доступа в сеть необходимо воспользоваться услугами локального интернет-провайдера.

Магистральные сети связи в России делятся на два сегмента:

- сегмент международной канальной ёмкости в направлении «Москва – Санкт-Петербург – Хельсинки – Стокгольм»; (по наравлению к матрице присутствия European Internet Exchange Association https://www.euro-ix.net/ixp-matrix)
- сегмент внутрироссийских каналов.

Кроме операторов интернет-доступа существуют также операторы контент-ресурсов – компании, деятельность которых направлена на информационное наполнение сети Интернет и предоставление конечным пользователям услуг информационного характера.

Различают следующие типы операторов контент-ресурсов:

- операторы дата-центров, предоставляющие возможность своим пользователям разместить их информацию на серверах и сделать её доступной в сети Интернет;
- провайдеры приложений (Application Service Provider – **ASP**) – компании, предоставляющие пользователям программные приложения, использующие интернет-технологии и доступные из сети Интернет;
- контент-провайдеры (Content Service Provider – **CSP**) – компании, предоставляющие пользователям интернета ту или иную информацию или службу (контент).

Классификация интернет-операторов показана на рис. 6.10.

Следующие 10 лет к построению магистральных сетей внутри России присоединились сотовые операторы и другие компании (не обязательно с российскими учредителями). На конец 2011-го наблюдался следующий список магистральных операторов связи. см. табл. 6.2.

Рисунок 6.11. Классификация операторов интернет-услуг

Таблица 6.2. Магистральные операторы связи в России

№	Оператор связи, ссылка на карту сети	Протяжённость, тыс. км
1	Ростелеком, http://www.rostelecom.ru/about/net/magistr/	500
2	Мегафон + Синтерра, http://www.synterra.ru/company/net/	118
3	МТС, http://www.corp.mts.ru/for_operators/map/	117
4	ВымпелКом	136,6 (2013 г.)
5	ТрансТелеКом, http://ttk.ru/rus/msk/business/62329/	75
6	Старт Телеком, http://www.starttelecom.ru/startmap/	16
7	Orange Business Services, http://www.orange-business.com/ru/content/about/about/infrastructure/	8,5
8	Раском, http://www.rascom.ru/network/	6
9	RetnNet, http://www.retn.net/network/map.html	5,7
10	«ТелиаСонера Интернешнл Кэрриер Раша», http://www.teliasoneraic.com/Our-network/Network-map.html	2

По данным исследований Фонда общественного мнения (http://*fom.ru*), на лето 2017 года показатель проникновения интернета среди взрослого населения России составил 70% (≈ 82 млн человек). Наибольший процент пользователей в Москве и Северо-Западном федеральном округе (см. рис. 6.12).

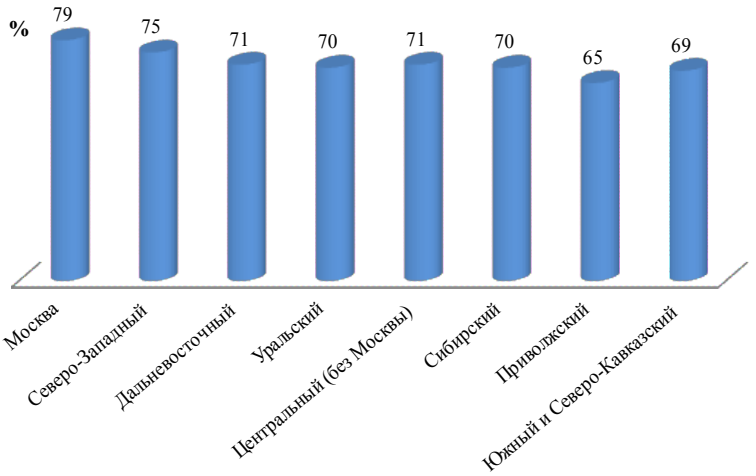
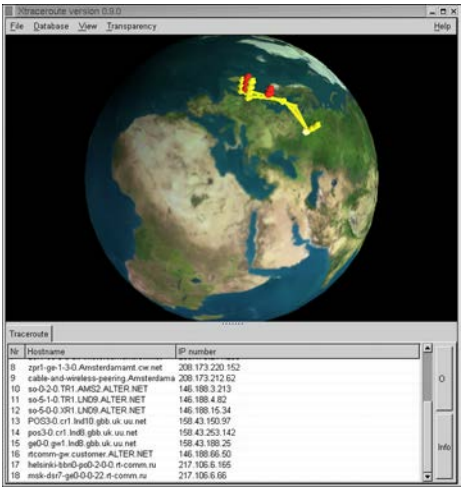


Рисунок 6.12. Доля пользователей интернета в регионах России (данные ФОМ, лето 2017 г.)



Одной из проблем российского интернета на начальной стадии его развития было практическое отсутствие межсетевого обмена IP-трафиком внутри страны (отсутствие пиринга [8]). Межсетевые связи реализовывались в глобальном интернете. Вследствие этого пользователи, имеющие подключение через одну национальную сеть, добирались до российского сервера, стоящего в другой сети (нередко в том же городе), проходя через океаны и геостационарные космические аппараты и перегружая не слишком широкие внешние магистральные каналы.

В середине 1996 года было подписано историческое соглашение по созданию точки взаимного обмена IP-трафиком (Internet eXchange) в Москве (M9-IX на автоматической междугородней телефонной станции АМТС-9). Участниками соглашения стали АО «Релком» (сеть EUnet/Relcom), компания «Демос» (сеть Demos/Internet), МГУ (MSUnet), НИИЯФ МГУ (сеть RUHEP/Radio-MSU), корпорация «УНИКОР» (сеть FREEnet), ассоциация RELARN (сеть RELARN-IP), АО «РОСПРИНТ».

Впоследствии к соглашению присоединились Институт космических исследований (сеть RSSI), компании «Совам Телепорт» (сеть Sovam), «Глас-Интернет» (сеть GlasNet), Институт «Открытое общество» (сеть IIP), Вузтелекомцентр (сеть RUNNet), ОАО «Российская телекоммуникационная сеть» (сеть Rosnet). Обмен IP-трафиком между участниками осуществляется на основе отдельных парных договоров. Аналогичная по своим задачам точка была создана в Санкт-Петербурге с участием сетей EUnet/Relcom, RUNNet, RELARN-IP и региональных сетей RUSnet N/W и «РОКСОН СЗ». Более подробную информацию о работе этих точек взаимного обмена можно получить на сервере Российского НИИ развития общественных сетей (*www.ripn.net*).

Организация обмена между операторами интернет-доступа интернет-трафиком своих клиентов разгрузила внешние каналы связи, но роль последних была весьма велика, так как большую часть информационных ресурсов пользователи 90-х потребляли из зарубежных сетей. В конце 1995 года суммарная пропускная способность каналов связи российских сетей с зарубежными провайдерами составляла чуть более 2 Мбит/с, а через год превысила 15 Мбит/с и ежемесячно продолжает возрастать как за счёт ввода новых каналов, так и за счёт повышения производительности существующих. В 2018 г., судя по графику загрузки каналов (http://www.msk-ix.ru/network/traffic.html), пиковые нагрузки доходят до 2,5 Тбит/с (для сравнения – в 2015 г. до 1,5 Тбит/с).

MSK-IX является одним из крупнейших мировых Internet Exchange по объёмам трафика и по числу участников и входит в международную ассоциацию European Internet Exchange Association (Euro-IX).

Крупнейшими магистральным оператором России остаётся Ростелеком.

Рисунок 6.13. Сеть ОАО «Ростелеком» (http://www.rostelecom.ru/about/net/magistr/)

Следующая компания, исключая сотовых операторов, – ЗАО «ТрансТелеКом».

Первоначально ЗАО «ТрансТелеКом» занималась строительством и эксплуатацией высокоскоростной телекоммуникационной сети вдоль российских железных дорог. Сейчас услугами ТТК охвачена территория, на которой проживает 90% населения России, ТТК имеет 50 000 км магистрально-цифровой сети связи. Сеть проходит через 11 часовых поясов (которые с 28 марта 2010 года объединены в 9 часовых зон), имеет более 950 точек доступа в 71 из 89 регионов России и выходит на границы с Финляндией, Польшей, Монголией, Китаем, странами СНГ и Балтии (см. рис. 6.14). По утверждению главы компании Сергея Липатова (когда он занимал эту должность, до августа 2010-го), ТТК занимал 44% рынка выделенных каналов и 45% рынка транзита интернет-трафика.

Рисунок 6.14. Сеть ЗАО «ТрансТелеКом»

Другие сети национального масштаба, представляющие российскую часть интернета:

- **Роснет** (*www.rosnet.ru*) – с 2013 года присоединилась к ОАО «Ростелеком»;
- **EUnet/Relcom** (*www.relcom.ru*) – базовые узлы в Москве и Санкт-Петербурге, имеет более 100 узлов в России и СНГ и два выхода в европейскую сеть Eunet;
- **Demos/Internet** (*www.demos.su*) – ядром сети являются несколько узлов в Москве, имеет магистральный выход в сеть MCI (США);
- **Sovam Teleport** – в 2000 году упразднена учредителями, её активы целиком переведены в компанию Golden Telecom;
- **Orange Business Services (ранее Global One Russia)** (*http://www.orange-business.com*) – эксплуатируют собственную однородную IP MPLS сеть с точками присутствия в крупнейших городах России и других странах СНГ; собственная магистральная сеть протяженностью 8 500 км от Москвы до Санкт-Петербурга, Приволжского, Уральского и Южного Федеральных округов;
- **RUNNet** (*www.runnet.ru*) – федеральная университетская сеть России, магистральная часть которой включает узлы более чем в 56 регионах России, спутниковые и наземные цифровые каналы и выход в глобальный интернет через скандинавскую сеть NORDUnet;
- **RUHEP/Radio-MSU** (*www.radio-msu.net*) – имеет узлы в ряде городов России и СНГ, основана на системе спутниковых каналов, в которой ключевую роль играют узлы в Москве и Гамбурге;
- **RSSI** (*www.rssi.ru*) – Russian Space Science Internet (Российская космическая научная сеть Интернет) – объединяет научно-исследовательские центры и институты, медицинские учреждения, учебные заведения и имеет выход в глобальный интернет через американскую сеть NASA Internet;
- **FREEnet** (*www.free.net*) – старейшая российская научно-образовательная сеть, имеющая более 15 региональных отделений, через которые работает большое число вузов и научных организаций РАН;
- **RELARN-IP** (*www.ripn.net/relarn-ip/*) – сеть, поддерживаемая Ассоциацией научных и учебных организаций – пользователей компьютерных сетей передачи данных (ассоциация RELARN);
- **MSUnet** (*www.msu.ru*) – сеть Московского университета, имеет выход в европейскую сеть Ebone и предоставляет свои услуги многим организациям, а также региональным сетям.

На странице https://www.telegeography.com/telecom-maps/global-internet-map/ доступна актуальная мировая карта отображающая связанность узлов сети.

6.3.3. Некоторые сетевые протоколы

Рассмотреть в полной мере сетевые технологии и все используемые на сегодня протоколы не получится, поэтому мы ограничимся приведением поверхностной картины, то есть рассмотрением лишь названий наиболее часто используемых протоколов и решений, раскинув их по уровням сетевой модели, чтобы читатели знали в каком направлении им осуществлять самостоятельные поиски в случае необходимости.

На физическом уровне технологии обеспечения доступа в сеть Интернет можно разделить на пять категорий, в зависимости от того, какой канал или среда передачи данных используется. К ним относятся:

- телефонный провод;
- сетевой кабель (предположительно, технологии Ethernet);
- оптико-волоконный кабель;
- беспроводные системы (сотовая, радиорелейная или спутниковая связь);
- другие, в том числе и смешанные системы.

Выбор протокола на канальном уровне обычно не зависит от пользователя и практически всегда однозначно определятся выбором физической среды передачи.

Почти все попадающие в поле зрения читателей бытовые компьютеры [385], участвующие в передаче данных по локальной сети, так и из/в интернет, используют на сетевом уровне протокол **IP** (с большой вероятностью 4-й версии). Подобно тому как слово «автомобиль», скажи мы его читателям, и воображение нарисует что-то с «колёсами», с протоколом сетевого уровня «IP» обычно используется протокол транспортного уровня «TCP». Вместе они образуют пару, которую ещё называют стеком[386], довольно часто приходится слышать «**стек TCP/IP»** (читается «ти-си-пи/ай-пи»). Словосочетание «~~протокол TCP/IP~~» некорректно, так как это два разных протокола, определяющих различные аспекты передачи данных в сети:

- **TCP** (Transmission Control Protocol) – протокол управления передачей данных, использующий автоматическую повторную передачу пакетов, принятых с ошибками; этот протокол отвечает за разбиение передаваемой информации на пакеты и правильное восстановление информации из пакетов получателя;
- **IP** (Internet Protocol) – протокол межсетевого взаимодействия, отвечающий за адресацию и позволяющий пакету на пути к конечному пункту назначения проходить по нескольким сетям.

Схема передачи информации по двум протоколам **TCP/IP** такова: протокол **TCP** разбивает информацию на пакеты и все ихнумерует; далее с помощью протокола **IP** все пакеты помещаются в IP-дейтаграммы и подобно пассажирам такси передаются получателю, где с помощью протокола **TCP** проверяется, все ли пакеты получены; после получения всех пакетов протокол **TCP** располагает их в нужном порядке и собирает в единое целое.

Кроме протокола TCP (с установлением соединения), на транспортом уровне может использоваться более простой протокол UDP, посылающий дейтаграммы без установления соединения. Выбор протокола транспортного уровня практически не определяется пользователем (прозрачен для него) и зависит лишь от используемых программ, а точнее протоколов более высоких уровней (уровня приложений).

На уровне приложения выбор протокола определяется приложением и его режимом работы: получение почты, отправка почты, просмотр веб-страниц браузером, просмотр страниц через защищённое соединение, обмен файлами, обмен мгновенными сообщениями и др.

Общий список используемых в интернете протоколов достаточно большой, вот некоторая часть:

[385] В том числе планшеты, телефоны и прочие устройства. Bluetooth-устройства и ИК-пульты управления бытовой техникой не в счёт.

[386] В общем случае в стеке может быть более двух протоколов, например HTTP/TCP/IP/Ethernet.

1) *на прикладном уровне используются протоколы*:

- *DNS*
- *FTP*
- *HTTP*
- *AOL/ICQ*
- *HTTPS*
- *IMAP*
- *LDAP*
- *SMB/CIFS*
- *POP3*
- *SMTP*
- *SSH*
- *NTP*
- *Telnet*
- *XMPP*
- *SNMP*
- и др.

2) *на сеансовом уровне/уровне представления используются:*

- *SSL*
- *TLS*

3) *на транспортном уровне*:

- *TCP*
- *UDP*

4) *на сетевом уровне*:

- *BGP*
- *ICMP*
- *IGMP*
- *IP*
- *OSPF*
- *RIP*
- *EIGRP*
- *IS-IS*

5) *на канальном уровне*:

- *Ethernet*
- *SLIP*
- *Frame relay*
- *PPP*
- *HDLC*

6.3.4. Адресация в интернете

Адресацию в интернете можно разделить на два вида: численную и символьную.

6.3.4.1. Символьная адресация

С символьными адресами вы, скорее всего, сталкивались. Например, символьным адресом является адрес электронной почты или адрес какого-либо сайта [387].

Исторически символьные адреса появились позднее, когда пользователи поняли, что им удобнее запомнить некоторую (особенно осознанно связанную) последовательность символов, такую как доменное имя, нежели набор ничего не значащих цифр, пускай и более коротких в записи.

Для работы компьютеров «между собой» на сетевом уровне система символьных имён скорее является дополнительной, чем обязательной, разве что кроме случаев доставки почты по доменному имени и других случаев обязательного использования доменов.

Все символьные имена принято записывать определённым образом. А именно в формате

```
[поддомен.][домен.]зона
```

где [поддомен.] может встречаться 0 или более раз, в случае если есть домен, а [домен.] может встречаться 0 или более раз. Зона присутствует всегда. Вся запись целиком также может называться доменом или доменным именем.

Принято называть «зону» в записи выше «корневым доменом». Примеры «.ru», «.org», «.com», «.рф» и др.

[387] В ряде протоколов (приложений) символьным может быть имя пользовательского аккаунта, например Skype и прочее. Использование символьного имени в данном случае к адресации никакого отношения не имеет.

Остальную запись также называют доменом *n*-го уровня по числу точек в записи. Вообще, от чего вести отсчёт, а именно вопрос, относить зону к домену первого уровня или нет, открыт и является условностью при исчислении. Так, домен «yandex.ru» является доменом первого уровня. А «zao.mos.ru» – домен второго уровня.

Разрешением доменных имён в IP-адрес занимается система с названием DNS (Domain Name System). Данная система состоит из большого числа серверов DNS (Domain Name Server), которые объединены между собой в иерархическую структуру. Есть корневые DNS-серверы, а есть обычные, то есть те, что идут ниже и к которым обращаются компьютеры пользователей (в настройках сетевых интерфейсов операционных систем есть соответствующие поля или файлы для записи DNS-серверов). Каждый провайдер поддерживает свой DNS-сервер для большей надёжности работы и минимизации трафика. При запросе пользователя к DNS-серверу провайдера о разрешении какого-либо доменного имени пришедший от него запрос анализируется, и по необходимости или даётся сразу ответ, или делается запрос к другому (вышестоящему) серверу. Между серверами существует такая же система запросов, как и от пользователя к DNS-серверу, за тем лишь дополнением, что сервер может отправить в ответ информацию о необходимости обращения к другому серверу, с указанием его адреса, чего между провайдерским DNS-сервером и клиентами не происходит.

Как следствие DNS-сервер знает все адреса, которые вы у него пытались разрешить. Обращение к DNS-серверу чужого провайдера считается дурным тоном (но не блокируется, хотя технически такая возможность есть). С другой стороны, как было отмечено на стр. 595, фирма Google, желая знать предпочтения пользователей (какие сайты они посещают), открыла два своих DNS-сервера по адресам 4.4.4.4 и 8.8.8.8 для всеобщего доступа. Надо сказать, что большинство администраторов на это предложение клюнуло, и теперь запросы многих пользовательских систем уходят на эти адреса, навечно оставаясь в истории обращений.

Рассмотрим пример доменного адреса: maestro.it3207.agtu.ru. Здесь maestro – имя реального компьютера в домене it3207 (третьего уровня), принадлежащем домену более высокого второго уровня agtu, ru – обозначение группы русскоязычных доменов первого уровня.

Доменов первого (высшего) уровня немного – около 250. Большая часть из них – так называемые географические домены. Например, .ru и .рф (Россия), .by (Белоруссия), .jp (Japan, Япония), .us (United States, США), .uk (United Kingdom, Великобритания), .ca (Canada, Канада), .au (Australia, Австралия) и т. д. Негеографические домены верхнего уровня – .com (для коммерческих компаний), .net (для сетевых ресурсов), .edu (образовательные учреждения), .mil (военные организации), .org (некоммерческие организации), .gov (правительственные ведомства), .int (интернациональные корпорации), .travel (фирмы, занимающиеся туризмом и гостиничным бизнесом), .aero (авиаперевозчики), .xxx, (сайты содержащие порнографию) и др. [388]

В процессе передачи данных доменный адрес преобразуется в **IP**-адрес. В Сети существуют специальные DNS-серверы, отвечающие за адресацию некоторой части Сети и сохраняющие информацию о соответствии **IP**-адресов и **DNS**-имён.

[388] Домен .onion, пришедший из «глубокого интернета», по формальным признакам выглядит как домен первого уровня и выполняет все его функции, но таковым не является, так как работа с ним не поддерживается традиционной системой DNS. Обратиться к сайтам, расположенным в этом домене, можно через Tor-браузер (см. www.torproject.org).

Для приобретения в собственность имени домена второго уровня (например, www.agtu.ru) необходимо обратиться к одному из регистраторов доменных имён (см. табл. на стр. 617). Имя может быть зарегистрировано на физическое или юридическое лицо. Все необходимые документы имеются на сайте регистраторов. В большинстве случаев регистрация платная (порядка 450–600 рублей). Регистрация действует 1 год. До истечения указанного срока домен можно делегировать на следующий учётный период, оплатив чуть меньшую сумму (также в пределах 400–600 руб./год). Стоимость доменов в других доменных зонах NET, COM, ORG, BIZ, INFO и прочих может быть как больше, так и меньше. В ряде случаев отечественные фирмы-регистраторы, в том числе и фирмы – хостинг-провайдеры, могут быть посредниками или агентами зарубежных регистраторов, для регистрации доменов у которых действуют свои, схожие правила.

Регистрация доменного имени не означает, что у вас сразу будет сайт или электронная почта с таким доменным именем. Для того чтобы «появился» сайт, кроме имени, необходимо иметь один, а лучше два DNS-сервера, поддерживающих ваш домен (часто регистратор предоставляет подобный сервис бесплатно), компьютер с запущенным и настроенным веб-сервером (хостинг) и собственно созданный сайт, размещённый на этом компьютере.

Заниматься поднятием собственного веб-сервера новичкам и несетевым специалистам мы не рекомендуем. Лучше обратиться к не один год существующим фирмам – хостерам. Например, http://zenon.net, http://demos.ru или др. За небольшую годовую плату они помогут и с регистрацией доменных имён, и с размещением файлов. А о том, как создать html-страницы для самого простого сайта, мы расскажем ниже в разделе 6.8. «*Основы создания Web-страниц* » (см. стр. 646).

URL, URI, URN

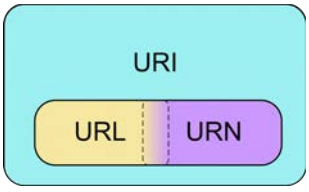


Диаграмма Венна, отображающая подмножества схемы URI: URL и URN

Universal Resource Identifier – универсальный идентификатор ресурса. URI является либо URL, либо URN, либо одновременно обоими.

URL – это URI, который, помимо идентификации ресурса, предоставляет ещё и информацию о местонахождении этого ресурса. А URN – это URI, который только идентифицирует ресурс в определённом пространстве имён (и, соответственно, в определённом контексте), но не указывает его местонахождения. Например, URN urn:ISBN:5-978-12345-1 – это URI, который указывает на ресурс (книгу) 5-978-12345-1 в пространстве имён ISBN, но, в отличие от URL, URN не указывает на местонахождение этого ресурса: в нём не сказано, в каком магазине её можно купить или на каком сайте скачать. Впрочем, в последнее время появилась тенденция говорить просто URI о любой строке-идентификаторе, без дальнейших уточнений. Так что, возможно, термины URL и URN скоро уйдут в прошлое.

Поскольку URI не всегда указывает на то, как получить ресурс, в отличие от URL, а только идентифицирует его, это даёт возможность описывать с помощью RDF (Resource Description Framework) ресурсы, которые не могут быть получены через интернет (например, личность, автомобиль, город и прочее).

Для поиска нужной информации в Сети чаще всего используется адрес ресурса (**URL**-адрес, англ. Uniform Resource Locator), содержащий имя протокола, по которому

нужно обращаться к требуемой информации, адрес сервера и имя файла на этом сервере. В общем виде формат URL выглядит так:

метод://host.domain[:port]/path/filename

где:

метод – одно из значений, перечисленных ниже:

- file – файл на локальной системе;
- http – файл на World Wide Web-сервере;
- gopher – файл на Gopher-сервере;
- wais – файл на WAIS (Wide Area Information Server);
- news – группа новостей телеконференции Usenet;
- telnet – выход на ресурсы сети Telnet;
- ftp – файл на FTP-сервере.

host.domain – доменное имя в сети Интернет.

port – номер UDP-порта (User Datagram Protocol – протокол пользовательских датаграмм).

Пример: ***http://www.narfu.ru/archive/index.html***

Префикс http:// указывает, что далее следует адрес веб-страницы, **www.narfu.ru** – зарегистрированное доменное имя и одновременно имя интернет-сервера, /archive описывает каталог с именем archive на сервере, а index.html – имя файла.

Часто имена файлов на интернет-серверах выглядят непривычно. Это связано с тем, что большинство серверов работают под управлением операционной системы UNIX, в которой имена файлов чаще всего не имеют расширения файла, идентифицирующего его тип. Основной кодировкой кириллицы в текстовых файлах на UNIX-серверах является UTF-8 (ранее была KOI8-R).

В интернете в настоящее время основным протоколом работы с информацией является гипертекстовый протокол **http** (hyper text transfer protocol) – основной режим работы современных интернет-обозревателей (браузеров). В состав большинства операционных систем входят браузер Forefox, Konqueror, Safari, Internet Explorer, Opera, Chrome, Lynx и прочие.

Организация информации в интернете в таком виде, когда файлы связаны между собой информационными связями в виде гипертекстовых ссылок, называется Всемирной паутиной и обозначается **www**. Под «паутиной» подразумеваются не линии связи, по которым идёт сигнал, а связи информационные.

Набор файлов на одном интернет-сервере, связанных между собой гипертекстовыми ссылками, называется **сайтом**. Например, сайт в интернете может содержать информацию о фирме. Он имеет свой «входной» файл – домашнюю страницу (home page), которая содержит в виде меню и названий разделов гипертекстовые ссылки на остальные части сайта. Обычно структура ссылок имеет иерархическую или более сложную структуру.

6.3.4.2. Численная адресация

Для правильной работы любой сети компьютерам на сетевом уровне следует присвоить сетевые адреса, так чтобы в одной подсети у любых двух компьютеров не было одинаковых адресов.

Обычно в небольших локальных сетях компьютерам присваиваются адреса вида 192.168.0.1, 192,168.0.2 ... 192.168.0.254 и на этом всё заканчивается, но откуда они берутся? В случае использования протокола IP выделение адресов происходит двумя способами: либо через вашего провайдера, либо самостоятельно. Провайдер может вам выделить как какой-то постоянный адрес, статически, например, прописав его в договоре или сообщив по телефону, так и выделять адрес динамически по каждому вашему запросу, например используя свой DHCP-сервер. В этом случае вопрос правильного выделения адреса лежит на плечах провайдера или вашего сетевого администратора. Но как получается, что разные провайдеры договариваются между собой и не используют одинаковых номеров?

Для того чтобы не было коллизий в глобальных сетях, то есть двух одинаковых адресов у разных хостов, в сети Интернет существует система регистрации. Если вы претендуете на получение уникального адреса, точнее подсети адресов, то вам необходимо будет подать заявки по формам RIPE-219, RIPE-220. Обычно этими заявками занимается ваш интернет-провайдер.

IP-адрес v.4 представляет собой уникальную четырёхоктетную (32-битовую) величину, выраженную в десятичных числах, разделённых точками, в форме W.X.Y.Z, где точки используются для разделения октетов (например, 213.107.22.1).

Согласно протоколу IP v.4, под адрес отведено 4 байта, причём для удобства его записывают в десятично-точечной записи, как четыре числа от 0 до 255, разделённых точками. То есть IP-адрес версии 4 – это что-то в диапазоне от 0.0.0.0 до 255.255.255.255. Не всякий адрес этого диапазона может быть присвоен компьютеру, так как среди этого диапазона есть зарезервированные по тем или иным причинам (широковещательные, для группового вещания, для интерфейса обратной петли и др.). Адреса, которые могут быть присвоены компьютеру (в общем случае хосту), назовём условно допустимыми или «разрешёнными». В свою очередь, допустимые IP-адреса делятся на две непересекающиеся группы:

реальные, или легитимные (внешние)	и	нелегитимные, или «для внутреннего использования» (внутренние).

Реальные используются в действующей сети, и по этим адресам можно обратиться к хосту от любого другого узла сети. Адреса «для внутреннего использования» предназначены для использования в различных локальных сетях, вне этих сетей пакеты с такими адресами не маршрутизируются и не передаются. На первом маршрутизаторе сети Интернет, если не задано иное, по умолчанию данные пакеты будет уничтожены. (Уничтожены – громкое слово, физического уничтожения не требуется, они будут просто проигнорированы.)

За использование адресов из обоих блоков легитимного и нелегитимного диапазонов отвечает администратор. Логично задаться вопросом: как администратор выделяет адреса, чем он руководствуется? Если для внешних адресов последние выделяются регистратором в ответ на поданную заявку и это проблема регистратора – следить за их расходованием, то со внутренними адресами ситуация несколько иная. Теоретически для изолированной сети можно выбрать любые адреса (так как она изолированная), но как быть, если вы не хотите, чтобы они пересекались с адресами каких-либо других сетей, например в будущем, когда ваша изолированная сеть получит внешнее подключение. Для того чтобы не было коллизий между внутренними адресами в ло-

кальных сетях и реальными адресами в глобальных сетях, следует руководствоваться документом RFC 1918 – Address Allocation for Private Internets при назначении адресов. Этот документ предлагает к использованию следующие диапазоны:

```
10.0.0.0      -  10.255.255.255  (10/8 prefix)
172.16.0.0    -  172.31.255.255  (172.16/12 prefix)
192.168.0.0   -  192.168.255.255 (192.168/16 prefix)
```

Именно их чаще всего и используют в локальных сетях администраторы.

Для того чтобы адресовать компьютеры из внешних сетей в таких подсетях, следует настроить маршрутизацию и должен существовать шлюз (в данном случае маршрутизатор, имеющий связи с другими сетями).

6.3.4.3. Сетевая маска

Сетевая маска – это важный параметр сети, позволяющий объединять адреса в группы (подсети). Посмотрев на номер какого-либо телефона с кодом, большинство без труда заметит в написании код своего города (области, района) либо скажет, что код (496) или (2816) им не знаком. Фактически телефонный код у вас и вашего соседа есть что-то общее между этими двумя номерами. Более чем уверены, что при наборе телефона соседа по проводной связи вы набираете меньше цифр, чем если бы пытались позвонить в другую страну. Скорее всего, вы опускаете код страны (и возможно, ещё и код города). Схожая ситуация наблюдается с IP-адресами, объединение их в группы позволяет уменьшить число записей в таблицах маршрутизаторов и межсетевых экранов [389]. Для того чтобы понять, как правильно делать это объединение, как вычислять широковещательные адреса на группу и прочее в 1981 году был официально опубликован документ RFC 791, в котором были описаны три подсети, классов A, B и C. Немногим позже некоторые зарезервированные и групповые адреса (видимо, по аналогии и для удобства) также были выделены в классы. В связи с этим многие учебники и книги до сих смело приводят на своих страницах таблицы, в которых делят IP-адресное пространство на классы адресов от «A» до «E». Однако неожиданный и драматический рост сети Интернет, при ограничениях, заложенных в определении указанных классов, потребовал пересмотра этих определений. Следствием стало создание более гибкой системы адресации [13]. (Скот Манн и Митчел Крелл рекомендуют по данному вопросу обратиться к ссылке [14], где выложены различные истории и факты времён становления интернета, правда на английском языке.)

Вместо фиксированных масок A (255.0.0.0), B (255.255.0.0) и C (255.255.255.0) и т. д. появилось понятие VLSM (Variable Length Subnet Mask, маска подсети переменной длины). Маски VLSM функционально ничем не отличаются от сетевых масок, состоящих из целого числа октетов. Эти маски уже приходится рассматривать не в десятичном, а в двоичном представлении. Изначально существовала десятично-точечная запись маски, например 255.255.0.0, в двоичном виде такая маска представляет

```
11111111 11111111 00000000 00000000
```

(Каждое десятичное число маски было переведено в двоичный вид.)

[389] «Глобальные» маршрутизаторы работают с автономными системами, за номерами которых закреплены те или иные подсети.

Если же необходимо сформировать маску

```
11111111 11111111 10000000 00000000
```

или

```
11111111 11111000 00000000 00000000
```

то гораздо проще записать число единиц в масках как «/17» или «/13», чем оперировать с записью 255.255.128.0 или 255.248.0.0. Конечно, для обратной совместимости никто не стал отказываться от старых возможностей записи маски, поэтому в «инертных» операционных системах мы всё ещё наблюдаем десятично-точечный формат.

Новая маска переменной длины разрушила понятие классовой адресации и сами классы. В связи с этим появилась бесклассовая адресация (англ. Classless InterDomain Routing, аббревиатура CIDR), а от неё – CIDR-нотация – формат записи маски в виде указанного через «/» числа единиц в двоичной записи. Очень удобно записывать эти маску сразу после IP-адреса, например 192.168.0.0/24 или 63.242.117.3/20.

Пример расшифровки маски сети в бесклассовой нотации:

Биты маски подсети	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0
Октеты маски подсети	255	255	248	0

С помощью специального механизма («маскирования» – не путать с «маскарадом») любая сеть может быть «разделена» или представлена набором более мелких сетей (подсетей).

Дополнительно про маски

В исходном документе, определяющем стандарты масок (RFC 922 [13]), рекомендуется использовать сплошные маски подсетей – то есть такие, в которых любой бит, установленный в 1, старше (имеет более высокий порядок), чем любой бит, установленный в 0. Скорее всего, вы скажете, что все рассматриваемые сегодня маски и так сплошные, но тем не менее RFC 922 не запрещает использования несплошных сетевых масок, подобных следующей:

```
11111111 11111111 11111111 10101010 = 255.255.255.170
```

Теперь вообразите, каково отслеживать принадлежность хостов к той или иной подсети в такой схеме! К счастью, в ряде документов RFC, последовавших за RFC 922, прозвучал запрет подобных масок. Соответственно, такие маски не допускаются ядром Linux современных версий.

6.3.4.4. Кому принадлежит «этот» IP-адрес?

Довольно часто приходится слышать подобные вопросы. Даже с некоторым гангстерским окрасом «пробить по IP-адресу». Для тех, кто знает, как устроена система интернет-адресации, указанный вопрос покажется более чем смешным. Следуя нижеописанным постулатам, вы сами поймёте, почему.

Дело в том, что число адресов конечно и адреса вне диапазона 0.0.0.0–255.255.255.255 не существуют по определению.

Для удобства адресное пространство можно делить (нарезать) на сети (подсети), как это делается, было пояснено выше.

Часть подсетей зарезервирована согласно документу IANA IPv4 Special-Purpose Address Registry [390] (см. табл. 6.3).

Таблица 6.3. Специальные IP-адреса

Блок адресов	Название	Документ RFC
0.0.0.0/8	«Этот узел в этой сети»	RFC 1122
10.0.0.0/8	Для частного использования	RFC 1918
100.64.0.0/10	Shared Address Space	RFC 6598
127.0.0.0/8	Сетевая заглушка (Loopback)	RFC 1122
169.254.0.0/16	Link Local	RFC 3927
172.16.0.0/12	Для частного использования	RFC 1918
192.0.0.0/24	IETF Protocol Assignments	RFC 6890
192.0.0.0/29	DS-Lite	RFC 6333
192.0.0.170/32, 192.0.0.171/32	NAT64/DNS64 Discovery	RFC-ietf-behave-nat64-discovery-heuristic-17
192.0.2.0/24	Documentation (TEST-NET-1)	RFC 5737
192.88.99.0/24	6to4 Relay Anycast	RFC 3068
192.168.0.0/16	Для частного использования	RFC 1918
198.18.0.0/15	Benchmarking	RFC 2544
198.51.100.0/24	Documentation (TEST-NET-2)	RFC 5737
203.0.113.0/24	Documentation (TEST-NET-3)	RFC 5737
240.0.0.0/4	Reserved	RFC 1112
255.255.255.255/32	Limited Broadcast	RFC 919

Всё оставшееся и распределяемое адресное пространство поделено между региональными регистраторами (о которых было рассказано на стр. 616) согласно документу IANA IPv4 Address Space Registry [391]. (Например, см. табл. 6.4.)

Таблица 6.4. Фрагмент таблицы распределения IP-адресного пространства по регистраторам (регионам)

Адресное пространство	Кому выдано
...	...
61.0.0.0–61.255.255.255	Тихоокеанский регион
62.0.0.0–62.255.255.255	Европа
193.0.0.0–195.255.255.255	Европа
199.0.0.0–199.255.255.255	Северная Америка
200.0.0.0–200.255.255.255	Центральная и Южная Америка
201.0.0.0–201.255.255.255	Центральная и Южная Африка
202.0.0.0–203.255.255.255	Тихоокеанский регион
204.0.0.0–205.255.255.255	Северная Америка
206.0.0.0–206.255.255.255	Северная Америка

[390] http://www.iana.org/assignments/iana-ipv4-special-registry/iana-ipv4-special-registry.xhtml.
[391] http://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.xhtml.

Адресное пространство	Кому выдано
207.0.0.0–207.255.255.255	Северная Америка
208.0.0.0–208.255.255.255	Северная Америка
209.0.0.0–209.255.255.255	Северная Америка
210.0.0.0–210.255.255.255	Тихоокеанский регион
211.0.0.0–211.255.255.255	Тихоокеанский регион
212.0.0.0–212.255.255.255	Европа
213.0.0.0–213.255.255.255	Европа
216.0.0.0–217.255.255.255	Северная Америка
...	...

А далее регистратор разделяет полученное во временное пользование «своё» адресное пространство согласно поданным ему заявкам. Информация о распределении заносится в базу. Соответственно, у каждого из регистраторов имеется своя база. Доступ к базе открыт с помощью whois-сервиса. В UNIX это штатная команда наряду с dig. Владельцы других ОС могут использовать общедоступные веб-интерфейсы для определения, какой подсети принадлежит интересующий их адрес и кому был выделен соответствующий сетевой блок.

Существуют пять WHOIS-баз, где можно поискать интересующую вас информацию о распределении IP-адресов: whois.apnic.net, whois.ripe.net, whois.arin.net, whois.afrinic.net, whois.lacnic.net. Для поиска по российским доменам рекомендуется использовать сайт Российского НИИ развития общественных сетей [392]. В отношении поиска по «российским» IP-адресам, ранее база РОСНИИРОСа также синхронизировалась с базой RIPE и в ней можно было посмотреть информацию о принадлежности IP-адресов см. на рис. 6.15., сейчас же за этой информацией следует напрямую обращаться к первоисточнику – базе данных RIPE по адресу https://apps.db.ripe.net/search/query.html.

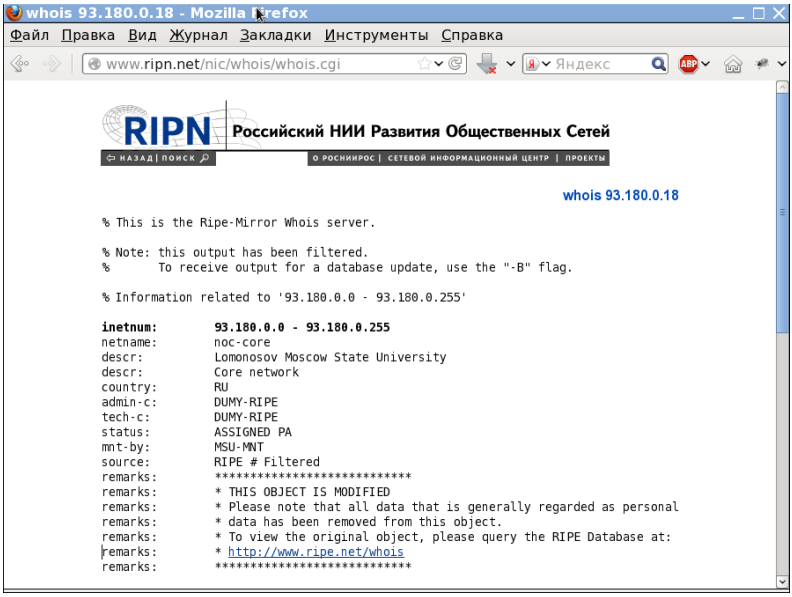


Рисунок 6.15. Пример определения принадлежности IP-адреса сервисом WHOIS РОСНИИРОСа http://www.ripn.net/nic/whois/

[392] http://www.ripn.net/nic/whois/.

6.3.4.5. Протокол IP версия 6 (RFC 2460)

В конце 1980-х стала очевидна необходимость разработки способов сохранения адресного пространства интернета. В начале 1990-х, несмотря на внедрение бесклассовой адресации, стало ясно, что этого недостаточно для предотвращения исчерпания адресов и необходимы дальнейшие изменения инфраструктуры интернета. К началу 1992 года появилось несколько предложений, и к концу 1992 года IETF объявила конкурс для рабочих групп на создание интернет-протокола следующего поколения (англ. IP Next Generation – IPng). 25 июля 1994 года IETF утвердила модель IPng, с образованием нескольких рабочих групп IPng. К 1996 году была выпущена серия RFC, определяющих интернет-протокол версии 6, начиная с RFC 1883.

14 июля 1999 года **IANA** объявила о введении версии **IPv6** во всемирном масштабе. 4 февраля 2008 г. **ICANN** начала добавлять в шесть из тринадцати корневых DNS-серверов записи, содержащие адреса в формате протокола **IPv6**.

Протокол **IPv6** выделяет на адрес 128 бит вместо 32 для IPv4. Новая версия IP-протокола **IPv6** даст интернету 1 млрд адресов в квадрате, которых должно хватить на много лет. С другой стороны, все понимают, что одновременно с переходом на новую версию вырастут накладные расходы за счёт повышения доли служебного трафика, ведь одни только служебные поля заголовка IP-дейтаграммы содержат 2 адреса (отправителя и получателя), и оба поля увеличатся в 4 раза, поэтому быстрое и прозрачное внедрение не ожидается.

Остальные изменения, коих немало, больше коснутся сетевых специалистов, поэтому мы их приводить не будем. Наиболее существенная и заметная для обычных пользователей сторона – это формат и семантика записи адресов **IPv6**, описанные в документе RFC 1884. Несомненно, что наши читатели рано или поздно увидят такие адреса, и чтобы они сразу их узнали, рассмотрим существующие три стандартные формы для представления **IPv6**-адресов в виде текстовых строк:

1. Основная форма имеет вид `x:x:x:x:x:x:x:x`, где «x» – шестнадцатеричные 16-битовые числа. Например:

   ```
   fedc:ba98:7654:3210:FEDC:BA98:7654:3210
   1080:0:0:0:8:800:200C:417A
   ```

   Начальные нули в каждом из конкретных полей не пишутся, но в каждом поле должна быть, по крайней мере, одна цифра (за исключением случая, описанного в пункте 2).

2. Из-за метода записи некоторых типов **IPv6**-адресов они часто содержат длинные последовательности нулевых бит. Для того чтобы сделать запись адресов, содержащих нулевые биты, более удобной, имеется специальный синтаксис для удаления лишних нулей. Использование записи «::» указывает на наличие групп из 16 нулевых бит. Комбинация «::» может появляться только при записи адреса. Последовательность «::» может также использоваться для удаления из записи начальных или завершающих нулей в адресе.

Например, нижеследующие адреса идентичны:

Полный формат адреса	Короткий формат адреса	Описание
`1080:0:0:0:8:800:200c:417a`	`1080::8:800:200c:417a`	уникаст-адрес
`ff01:0:0:0:0:0:0:43`	`ff01::43`	мультикаст-адрес
`0:0:0:0:0:0:0:1`	`::1`	адрес обратной связи

3. Альтернативной формой записи, которая более удобна при работе с **IPv4** и **IPv6**, является `x:x:x:x:x:x:d.d.d.d`, где «x» – шестнадцатеричные 16-битовые коды адреса, а «d» – десятичные 8-битовые, составляющие младшую часть адреса (стандартное IPv4-представление).

   Например: `0:0:0:0:0:0:13.1.68.3`
   `0:0:0:0:0:FFFF:129.144.52.38`

   или в сжатом виде: `::13.1.68.3`
   `::FFFF:129.144.52.38`

При использовании IPv6-адреса в URL необходимо заключать адрес в квадратные скобки:

```
http://[2001:0db8:11a3:09d7:1f34:8a2e:07a0:765d]/
```

Если необходимо указать порт, то он пишется после скобок:

```
http://[2001:0db8:11a3:09d7:1f34:8a2e:07a0:765d]:8080/
```

6.3.5. Способы подключения к интернету конечных пользователей

http://avdey.ru
Карикатуры от Авдея

В настоящее время существует достаточно большое количество способов простому пользователю подключиться к интернету и пользоваться всеми его большими возможностями.

Модем

Старый способ подключения – с использованием аналогового модема для коммутируемых линий (dial-up-подключение). При этом подключении заключается договор с провайдером [393], в котором указаны номер телефона, по которому должен звонить модем (телефон модемного пула провайдера), имя пользователя, его пароль для подключения и прочие данные для настройки подключения, на счёт договора вносится плата за время работы, либо время не ограничивается. Из-за низкой скорости подключения – максимум 56 Кбит/с – вести расчёты по скачанным мегабайтам экономически не эффективно.

Замечание. *Напомним, что линия телефонной связи за счёт использования уплотняющего оборудования ограничена полосой «голосового канала» в диапазоне 0,3-3,4 КГц, под которую модему приходится подстраиваться. Чтобы оценить пропускную способность такой полосы решим обратную задачу: оценим какая должна быть минимальная скорость цифрового потока для безошибочного восстановления сигналов в этой полосе. Если грубо взять аналоговый*

[393] Договор может быть заключён и в форме договора-оферты, например вы покупаете пластиковую карту с написанными на ней под защитным слоем логином и паролем. Вы читаете прилагаемую инструкцию (договор оферты). Если согласны – стираете защитный слой и начинаете пользоваться.

сигнал в полосе 0-4 КГц, и перевести его в цифровой вид, то отсчёты следует брать 8000 раз в секунду. При частоте дискретизации сигнала на 256 уровней (8 бит) потребуется передавать 64000 бит в секунду. Собственно получено значение близкое к 56 Кбит/с.

DSL-модем

За счёт обхода уплотняющего оборудования телефонной сети, путём размещения интернет-провайдером своего оборудования на каждой телефонной станции появилась возможность более эффективного использования медной пары на участке «АТС – пользователь». Для передачи данных интернета медная пара используется в частотном диапазоне по спектру значительно выше спектра работы обычного телефона в режиме передачи голоса. По приблизительным оценкам верхняя граница полосы пропускания медной пары составляет несколько МГц. Использование имеющихся телефонных линий позволяет сэкономить на прокладке новых линий связи, а широкая полоса пропускания позволяет значительно поднять скорость передачи. При использовании разделяющих фильтров (сплиттеров) возможно одновременное использование линии для передача данных вместе с аналоговым голосовым трафиком, как и набором телефонного номера или приёма сигнала вызова. Провод от телефонной розетки подключается к фильтру-разветвителю – маленькой коробочке с одним входом и двумя выходами к которым подключаются аналоговый телефон и DSL-модем.

Технология получила название xDSL [394], а модемы – приставку «DSL-». Вместо «x» могут подставляться разные буквы, означающие особенности реализации технологии, например: ADSL, HDSL, IDSL, MSDSL, PDSL, RADSL, SDSL, SHDSL, UADSL, VDSL. Наиболее распространённой оказалась ADSL (Asymmetric Digital Subscriber Line) – модемная технология, в которой доступная полоса пропускания канала распределена между исходящим и входящим трафиками асимметрично (к абоненту канал шире). В зависимости от выбранного тарифа скорость передачи данных к пользователю составляет от 64 Кбит/с до 24 Мбит/с, от пользователя – от 64 Кбит/с до 3,5 Мбит/с.

Для жителей Архангельска на начало 2015 г. скорость исходящего трафика для всех ADSL-тарифов (от абонента в сеть Интернет) составляла 512 Кбит/с при ограничении скорости входящего трафика до 2 Мбит/с при стоимости безлимитного подключения (без ограничения трафика) 395 руб./мес., при скорости до 7 Мбит/с – 499 руб./мес. На ряде АТС с модернизированным оборудованием (поддерживающим ADSL2+ технологию) примерно за те же деньги предлагается в 2 раза большая скорость в сторону абонента.

Подключение к компьютеру модема-маршрутизатора выполняется через сетевой разъём (Ethernet) или USB-порт.

[394] От англ. *digital subscriber line*, цифровая абонентская линия.

Прямое кабельное подключение

Третья технология, имеет большее распространение в мегаполисах с большой плотностью населения, когда до квартиры пользователя подводится кабель (витая пара технологии Ethernet), так называемый широкополосный доступ. В некоторых случаях возможно подключение даже оптикой. В этом направлении в столице ОАО «МГТС» активно (если не агрессивно) проводит работы по прокладке оптических кабелей взамен телефонных с целью перевода абонентов на GPON-технологию [395]. С небольшой задержкой стоит ждать повторения ситуации в регионах и меньших по числу населения городах.

Спутниковый интернет

Четвёртая технология, обеспечивающая приемлемые скорости передачи данных в сторону пользователя в отдалённых районах – асимметричный доступ в интернет через спутник.

Существуют две основные схемы работы через спутник – симметричная и асимметричная.

В первом случае клиент осуществляет и передачу запроса на спутник, и приём данных со спутника. Такое решение является достаточно дорогим, как по части клиентского оборудования, так и по стоимости обслуживания, и применяется в основном в тех случаях, когда его использование является либо единственно возможным, либо более дешёвым, чем использование проводных или радиоканалов (например, в труднодоступных или удалённых районах с неразвитой инфраструктурой связи).

Во втором случае со спутника осуществляется лишь приём информации, в то время как передача запросов осуществляется по наземным каналам связи через интернет-провайдера (см. рис. 6.16).

Рисунок 6.16. Схема асимметричного спутникового доступа в интернет

Стоимость такого решения более низкая – не требуется дорогостоящего спутникового оборудования передачи данных, для приёма можно использовать стандартную

[395] GPON (Gigabit-capable Passive Optical Network) – гигабитная пассивная оптическая сеть с гарантированным качеством обслуживания.

«тарелку» совместно с DVB-картой, для передачи исходящих запросов – обычный модем. DVB-карта (сокращение от Digital Video Broadcast – передача цифрового видео) представляет собой PCI-карту, которая производит обработку цифрового спутникового сигнала, поступающего с приёмной антенны.

Жители европейской части России могут воспользоваться услугами двух наиболее известных провайдеров – Europe OnLine (EOL) и NTV Internet (NTVi).

Алгоритм работы систем следующий.

Запрос от пользователя на требуемый ресурс уходит по наземным линиями связи в интернет, до сервера провайдера через VPN-соединение поверх обычного интернета.

Далее провайдер делает от себя запрос к запрошенному ресурсу и за короткое время получает ответ, после чего он отправляется на наземную станцию спутниковой связи. Станция спутниковой связи преобразует ответ клиенту в формат DVB и отправляет его на спутник.

Спутник транслирует ответ клиенту в эфир. Ответ приходит вместе с общим DVB-потоком на приёмную антенну клиента. Спутниковый сигнал принимается картой DVB-декодера, и из потока извлекаются IP-пакеты клиента. Клиент получает ответ от нужного ресурса.

Для того чтобы воспользоваться услугами системы, необходим стандартный набор оборудования, включающий в себя спутниковую антенну диаметром от 70–90 см, DVB-плату, а также комплект программного обеспечения.

Пользователю, помимо веб-сёрфинга (привычных HTTP-запросов), доступны все остальные сервисы и службы. Ограничение скорости передачи данных в системе составляет около 400 Кбит/с в сторону абонента.

Ежемесячная абонентская плата – около 500 руб.

Wi-Fi интернет

Wi-Fi – вид беспроводного подключения. Сейчас большое количество оборудования делается с поддержкой Wi-Fi (ноутбуки, планшеты, телефоны, электронные книги, игровые и видеоприставки и прочее). Объединять в одну сеть и раздавать интернет всем этим устройствам может или ваша домашняя точка доступа (или соседская), обычно имеющая DSL или проводное подключение к сети, реже 3G или LTE (4G). Помимо вашего Wi-Fi, вы можете получать интернет, находись в зоне покрытия местного провайдера. Покрытие может быть как у вас дома, так и в ближайшем аэропорту, офисном здании, ресторане, кафе и даже на улице. По сути, это единственный способ подключений к интернету планшетов, не оборудованных модулем связи через проводного или сотового оператора. Глобальный переход на этот тип подключения всех устройств не происходит из-за малого радиуса действия «точек доступа». Даже с направленными антеннами удаление от точки доступа не превышает сотен метров. Провайдеров для этой технологии мало. Довольно часто её используют сотовые операторы, предлагая комбинированный доступ с лучшим покрытием [396]. Многие любят Wi-Fi за то, что чаще всего эту технологию используют для

[396] Например, планшеты с SIM-картой от Билайн могут работать через сотовую сеть, а в случае обнаружения рядом Wi-Fi-покрытия своего оператора производят прозрачное переключение пользователей на более быстрый Wi-Fi-канал.

частной раздачи интернета в публичных местах, кафе, аэропортах как дополнительную бесплатную услугу. Сама технология довольно скоростная, но точки доступа часто бывают медленны, так как перегружены любителями бесплатного. Немало головной боли доставляют Wi-Fi-сети родителям, которые сначала покупают своим несовершеннолетним детям «гаджеты», а потом желают ограничить им доступ в сеть, но у них ничего не получается, так как их чада то и дело находят то незакрытую соседскую точку, то имеют легальный доступ в публичном месте.

Мобильный интернет

Сюда относятся подключения через мобильный телефон, либо беспроводной модем. Сейчас почти любое мобильное устройство имеет выход в интернет, и не одним способом. Android- и iOS-устройства при этом могут «раздавать» мобильный интернет, полученный от сотового оператора, всем другим домашним Wi-Fi устройствам, выступая в роли беспроводной точки доступа.

Также телефоны могут предоставлять интернет-доступ для отдельного компьютера, будучи соединённым с ним по Bluetooth или USB-кабелем, выступая в качестве сотового модема.

Относительно старые телефоны подключаются по медленным и дорогим технологиям WAP и GPRS. Дни жизни этих технологий сочтены. Новые телефоны дополнительно могут использовать более скоростные технологии: CDMA, UMTS, LTE, WiMAX. Для последних операторы почти всегда предлагают альтернативу: недорогой и маленький USB-модем[397], который можно подключить к компьютеру или ноутбуку и без проблем пользоваться интернетом везде, где есть сотовое покрытие. Многие так «берут интернет на дачу».

Скорость мобильного интернета чаще всего удовлетворительная, а качество сигнала сильно варьируется от покрытия, погодных условий и других факторов. Для большинства пользователей нет разницы в порядке использования устройств 3G и 4G, однако при полноценном использовании 4G-модемов сложно не почувствовать разницу.

4G (от англ. fourth generation – четвёртое поколение) – поколение мобильной связи с повышенными требованиями. К четвёртому поколению принято относить перспективные технологии, позволяющие осуществлять передачу данных со скоростью, превышающей 100 Мбит/с подвижным и 1 Гбит/с – стационарным абонентам.

При таких скоростях можно хоть online-видео просматривать в разрешении HD. Сети 4G сейчас активно развиваются.

Стоимость безлимитного мобильного интернета в зависимости от потребностей (скорости доступа и/или объёма включённого трафика) варьируется от 100 до 1500 руб./мес.

[397] По размерам не больше USB-флэшки, внешне отличается лишь тем, что имеет разъём для SIM-карты (или RUIM-карты) и опционально небольшую антенну.

6.3.6. Что надо «знать» компьютеру, чтобы «выходить» в интернет?

У пользователей сотового интернета телефон или модем обычно настраиваются в магазине при их покупке, в крайнем случае, телефон «сам» получает SMS-ку после включения с вопросом «*Принять сетевые настройки?*» – *Конечно, да!*, обычно отвечают пользователи, что может быть проще? Естественно, что при такой схеме практически никто не задумывается о сетевых настройках подключения к интернету.

Чуть больше думать приходится пользователям Wi-Fi. Ими или ищется сеть «без замочка», или «с замочком» и спрашивается у владельца сети пароль (например, в кафе). В общественных местах зачастую и думать не надо – бесплатное подключение после минутного просмотра рекламы, а адрес идентификатора сети написан крупными буквами на видном всем месте (например MT_FREE в московском транспорте).

Владельцы домашнего проводного интернета в наиболее выгодном положении – подключил сетевой кабель и работай, главное, чтобы на ноутбуке или стационарном компьютере был соответствующий разъём (сетевая карта).

Технический прогресс отучает людей думать, в результате появляются «далёкие от техники» пользователи, которые начинают считать интернет чем-то божественным... боятся его, трясутся над ним, чтобы не отключился и прочее. И всё потому, что «оно само»... а мы не знаем, как. В XXI веке для людей, умеющих читать и писать, это выглядит более чем смешно. Хотелось бы приподнять завесу происходящего и пояснить, что за внешней простотой стоят несложные сетевые процессы, в которые вмешаться по силам любому школьнику.

Для работы интернета необходимо, чтобы в устройстве (компьютере, телефоне или планшете) были настроены несколько сетевых параметров:

- связь на канальном уровне;
- настройки сети: IP-адрес и сетевая маска;
- внешний шлюз или proxy-сервер;
- DNS сервер;
- опционально: VPN-соединение (туннель) до сервера провайдера.

Все эти настройки или выставляются «вручную», или за пользователя работает протокол DHCP [398] (Dymanic Host Configuration Protocol, что часто выглядит как «автоматическая настройка»).

Рассмотрим настройки подробнее.

Проводное соединение: на канальном уровне, при правильно обжатых разъёмах сетевого кабеля и исправном сетевом оборудовании дополнительные настройки не требуются.

Беспроводное соединение: сотовый интернет – обычно вмешательство пользователя не требуется, но в ряде случаев необходимо прописать вручную три параметра: **APN точка доступа**, **логин** и **пароль**. Для МТС – internet.mts.ru, mts, mts или все три параметра mts. Для Билайна: internet.beeline.ru, beeline, beeline. Для Мегафона – internet, gdata, gdata, либо все три параметра megafon, а в ряде округов действует услуга «Любой APN». Для Tele2 – internet.tele2.ru, логин и пароль пустые.

Беспроводное соединение: Wi-Fi – требуется выбрать сеть и «подключиться» к ней. Все сети идентифицируются по SSID (Service Set Identifier). Данный параметр может широковещаться, и тогда ваше устройство покажет, мол, обнаружены такие-то сети. А затем предложит выбрать сеть для подключения. В случае если широковеща-

[398] Точнее, думает сетевой администратор, который настраивает DHCP-сервер.

ние SSID отключено, будет показано, что найдена «неизвестная сеть», или вы об этом не узнаете. Чтобы подключиться, надо знать SSID. Далее, беспроводные подключения бывают открытые – для подключения к ним ничего не требуется, либо защищённые (аутентификация и шифрование передаваемых данных) – аббревиатуры WEP, WPA или WPA2. Грубо говоря, для работы с последними требуется пароль [399].

На сетевом (межсетевом) уровне требуется указать IP-адрес и сетевую маску. Если Вы указали в настройках сетевого интерфейса «получить адрес автоматически», будет использован протокол DHCP. В этом случае адрес запрашивается у DHCP-сервера на некоторое время, значение которого определяется настройками сервера. Именно по этой причине многие DSL-модемы раз в сутки «рвут» соединение, так как вынуждены переполучать сетевой адрес. Посмотреть адреса на всех сетевых интерфейсах в ОС Windows можно, запустив команду «`ipconfig /all`» в консоли (Пуск, выполнить, cmd.exe), в ОС Linux – «`ip addr show`». Получить (обновить) адрес от DHCP сервера – «`ipconfig /renew`» и «`dhclient eth0`» соответственно. Существуют и другие способы/команды просмотра и изменения сетевых настроек, в том числе и через графический интерфейс.

Далее, пакетам из вашей локальной сети для путешествия «в глобальную» нужен или шлюз [400] – компьютер, который будет иметь два подключения, одно в локальную сеть (домашнюю) и второе во внешнюю сеть (по направлению к другому шлюзу или интернету), или локальный посредник в виде proxy-сервера (SOCKS или HTTP). Если прокси локальный – настройка шлюза не требуется. Самому прокси шлюз нужен. Прокси может быть как в локальной сети, так и в интернете. В ряде случаев один прокси может использовать другой, в этом случае говорят о «цепочке прокси-серверов» или «каскаде прокси». Не все прокси поддерживают каскадирование. Чаще всего пользователь прописывает всего один маршрут – «маршрут по умолчанию» (он же «шлюз по умолчанию») – либо получает эту информацию автоматически по протоколу DHCP одновременно с получением IP-адреса и шлюза.

Посмотреть таблицу маршрутизации в ОС Windows можно, запустив в консоли команду «`route print`», в ОС Linux – «`ip route show`». Существуют и другие способы/команды сделать то же самое.

Последняя важная настройка – это DNS-сервер вашего провайдера. Зачем он нужен, было сказано выше. Для надёжности DNS-серверов обычно указывают несколько. К сожалению, стандартный оконный интерфейс ОС Windows предоставляет только два поля для ввода. В ОС Linux информация о DNS хранится в файле /etc/resolv.conf.

6.3.7. Поиск информации в интернете

Для поиска информации в интернете созданы мощные средства: поисковые серверы (поисковики), каталоги (рубрикаторы), списки рейтингов, тематические списки ссылок, on-line-энциклопедии, справочники и прочее.

[399] С точки зрения конфиденциальности передаваемых данных использование WEP, WPA или WPA2 оправдано, но не следует забывать, что при низкой сложности пароля и нечастой его смене указанные технологии смогут защитить разве что от домохозяек, вопрос «взлома» вашей домашней точки доступа – лишь вопрос времени. Так что, уходя из дома или надолго уезжая, лучше её отключить, даже если у вас и безлимитный тариф. С одной стороны, интернет наводнён подобными документами – http://niochem.ru/wifi/wifi-security/vzlom-wpa-wpa2-wifi-v-windows/, с другой – УК РФ никто не отменял. Что потенциальные взломщики найдут раньше? – вопрос открытый.

[400] В общем случае шлюзов может быть несколько. Каким воспользоваться, решает пользователь, создавая правила-маршруты, по которым компьютер впоследствии и принимает каждый раз решения.

Поисковая система – комплекс программных и аппаратных средств для автоматического просмотра ресурсов интернета, индексации их содержания и предоставления услуг по поиску информации интернет-пользователям. Поисковые системы могут отличаться по эффективности поиска, по языку поиска (русский, английский и др.) и по некоторым другим возможностям. Например, одни поисковые системы находят информацию только в виде веб-страниц, другие могут просматривать и группы новостей, и файловые серверы.

Наиболее известны следующие российские поисковые системы:

Яндекс (http://www.yandex.ru/, http://ya.ru, http://yandex.com);

Рамблер (http://www.rambler.ru/, http://r0.ru);

Апорт (http://www.aport.ru/);

Поиск@mail.ru (http://go.mail.ru/).

К сожалению, с выпуском акций и выходом указанных компаний (дочерних, учредительных и прочих) на инвестиционные рынки со своими предложениями их принадлежность «к российским» или отечественным оказывается под вопросом. В каждом случае следует исследовать вопрос более тщательно и чётко определять критерии принадлежности к тому или иному лагерю.

К международным системам для поиска информации в информационных ресурсах можно отнести:

Google (*http://www.google.com/* или *http://www.google.ru/*);

Alta Vista (http://www.altavista.com/);

Yahoo! (http://www.yahoo.com/);

Infoseek (http://www.infoseek.com/);

Microsoft Live Search (*http://www.live.com/*);

Hot Bot (http://www.hotbot.com/).

Поисковые системы могут быть двух типов: универсальные и специализированные. Наиболее популярные современные поисковые системы сочетают в себе оба типа.

В универсальных системах используется обычный принцип поиска в неструктурированных документах – по заданной строке поиска.

На домашней странице поисковой системы обычно расположено поле для ввода строки поиска, могут также присутствовать меню и прочие элементы. Домашняя страница систем **Яндекс** и **Google** может настраиваться зарегистрированным в системе пользователем. Большинство поисковых серверов предоставляют пользователям также услуги интернет-почты (почтовые ящики), возможность создания личных веб-страниц, новости, гороскопы, курсы валют, прогноз погоды и прочие сервисы.

Довольно популярен сервис поиска адресов на картах. На сегодня проект **Яндекс.Карты** научился не только показывать карты городов, «пробки» в реальном масштабе времени, ссылки на работающие веб-камеры, частные фотографии, но и предлагает совершить виртуальные туры практически по всем российским городам от Калининграда до Владивостока и даже таким небольшим, как Павловский Посад и др. (см. рис. 6.17). В глобальном масштабе доступны снимки, полученные с российских спутников, в ряде городов, например для г. Санкт-Петербург, кроме виртуальных туров, доступны также виртуальные полёты над городом на воздушном шаре – аэрофотосъёмка.

Большинство имеющихся панорам были сделаны с помощью специального автомобиля на дорогах общего пользования, поэтому они больше полезны автолюбителям

поскольку содержат виды дорог и развязок с позиции водителя и пассажиров автотранспортных средств, но есть также панорамы сделанные и с водных транспортных средств и в пеших зонах и даже в музеях.

Последнее нововведение – это функция «машины времени», когда пользователь может выбирать просмотр одного и того же места в разные моменты, сохранённые в архиве, в том числе и в разные времена года. Сравнивая виды можно видеть динамику жизни, где-то появляются новые здания, где-то сносятся старые и т. п.

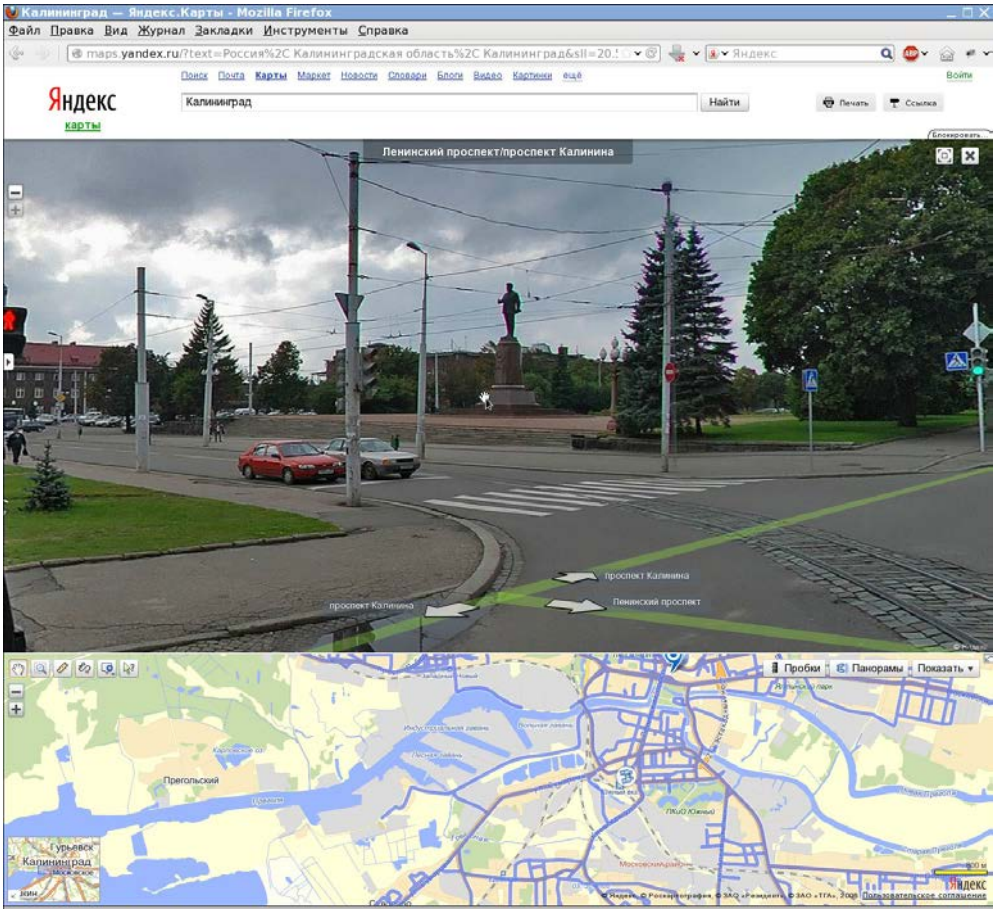


Рисунок 6.17. Виртуальная экскурсия по г. Калининград. Можно посмотреть «влево», «вправо», «вверх», «вниз», сделать шаг вперёд или назад, приблизить или удалить картинку. Внизу на карте биноклем показано отображаемое место

Аналогичный проект с виртуальными пешими экскурсиями по планете Земля есть и у Google (http://www.google.com/earth/).

Сложные поисковые запросы

Строка поиска может состоять из одного слова или группы слов. Если слова разделены пробелами, ищутся документы, в которых присутствует хотя бы одно слово из перечисленных. Для поиска словосочетания его следует заключить в кавычки. В некоторых системах можно осуществлять поиск по части слова, оставшаяся часть слова за-

меняется знаком «*», как в шаблоне имени файла. Знак «+» или «&» между словами требует обязательного присутствия всех слов в документе.

Часто существует также кнопка перехода к расширенному поиску. Главное отличие расширенного поиска – использование в запросе логических операторов и круглых скобок. Для построения сложного запроса используются логические операторы AND (И), OR (ИЛИ), NOT (НЕТ) и NEAR (около; не далее чем в 10 символах). Логические операторы ставятся между словами или словосочетаниями. Здесь могут использоваться даты документов, размер документов и другие критерии. Интерфейсы расширенного поиска у разных поисковых систем существенно отличаются.

Поисковые системы обычно состоят из трёх компонентов:

1) поисковый робот (агент, паук или crawler), который «перемещается» по сети и собирает информацию;
2) база данных, которая содержит всю информацию, собираемую роботом;
3) поисковый механизм, который используется как интерфейс для взаимодействия пользователей с базой данных.

Поисковые роботы – это специальные программы, которые занимаются поиском страниц в сети, сохраняют гипертекстовые ссылки на эти страницы и автоматически индексируют присутствующую на них информацию для построения базы данных поисковика. При построении индекса исходные данные преобразуются так, чтобы объём базы был минимальным, а поиск осуществлялся очень быстро и давал максимум полезной информации.

При запросе к поисковой системе она отыскивает в своей базе данных информацию, соответствующую запросу, и выводит список ссылок. В этом списке представлены ссылки на различные веб-страницы, причём ссылки располагаются по степени убывания встреченных на данных страницах слов, совпадающих с ключевыми словами, или по дате источника. При просмотре списка можно выбрать и просмотреть заинтересовавшие вас страницы.

Признанным лидером среди поисковых систем в мире является **Google**, который обеспечивает наиболее скоростной поиск с наибольшим количеством ссылок по многим тематикам. В распоряжении **Google** имеется более 15 000 серверов, разбросанных по дата-центрам всего мира, которые выполняют поиск информации.

В России более популярен **Яндекс**. Популярность отчасти объясняется высоким качеством сервиса. Илья Сегалович (1964–2013) – основатель и директор по технологиям Яндекса, создатель первой версии поисковика и автор слова Yandex – считал, что высокую планку можно держать лишь тогда, когда тебе дышат конкуренты в спину, поэтому на Яндексе вы всегда можете найти внизу, после результатов поиска, ссылки вида: Поискать «то же самое» на … (перечислены сайты конкурентов).

На рис. 6.18 показано поле поиска системы Яндекс с автоматически раскрывающимся контекстным списком ссылок после набора первых букв строки поиска.

Каталог в интернете – данные, структурированные по темам в виде иерархических структур. Ссылки в такие каталоги заносятся не автоматически, а с помощью администраторов. Как правило, хорошие каталоги интернета обеспечивают разнообразный дополнительный сервис: поиск по строке поиска в своей базе данных, списки последних поступлений, списки наиболее интересных из них, выдачу случайной ссылки, автоматическое оповещение по электронной почте о свежих поступлениях. Один из наиболее популярных каталогов в России – ***http://list.mail.ru/***.

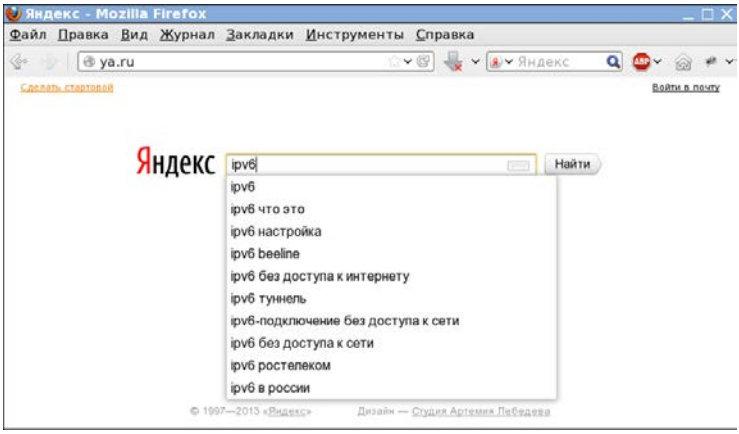


Рисунок 6.18. Поиск в системе Яндекс

Рейтинги сайтов содержат списки наиболее посещаемых мест в интернете. Одним из наиболее популярных рейтингов является Rambler's Top 100. (*http://top100.rambler.ru/top100/*). Популярность ресурса оценивается по ряду параметров, в том числе по количеству уникальных посетителей в единицу времени и хитам – количеству заходов на сайт за период времени. Большая посещаемость часто никак не связана с качеством посещаемых ресурсов. В нынешнем мире «раскрутить» можно любую глупость, были бы деньги и время.

Онлайновые энциклопедии и справочники представляют собой электронные версии многих известных соответствующих бумажных документов. Одной из крупнейших on-line-энциклопедий является ресурс «Яндекс.Словари» (ранее «Яндекс.Энциклопедии») (*http://slovari.yandex.ru/*), который содержит более полусотни энциклопедий и словарей, в том числе Большую Советскую Энциклопедию, Энциклопедию Брокгауза и Эфрона и прочие. К крупным относится и Мегаэнциклопедия Кирилла и Мефодия, которую можно найти по адресу http://megabook.ru/.

Помимо традиционного словаря, ресурс Webopedia (*www.pcWebopedia.com*) имеет массу специализированных сервисов, например: «Кто есть кто в компьютерных технологиях», «Сравнительная таблица микропроцессоров», «История развития компьютерных технологий» и др.

Интенсивно развивается Википедия – свободная энциклопедия в интернете (*http://ru.wikipedia.org*), в которой могут размещать свои статьи по любым вопросам, редактировать и дополнять существующие материалы любые желающие.

Для студентов и школьников полезным ресурсом является ***интернет-портал Минобразования РФ*** (*http://www.edu.ru*, см. рис. 6.19) и его **«Единое окно доступа к образовательным ресурсам»** *http://window.edu.ru.*

Рисунок 6.19. Федеральный портал Российское образование

6.3.8. Основы создания веб-страниц

Все сайты при обращении к ним браузером по протоколу HTTP (или HTTPS) выдают последнему HTML-документы или веб-страницы (либо ошибку, если что-то пошло не так). Эти страницы могут генерироваться на лету, а могут лежать на диске сервера как текстовые файлы в определённой директории. Второе – самый простой случай. Можно ли самому создать такую страницу? Конечно, да! Для этого есть множество программ и способов.

Разработка веб-страниц может вестись в различных программных средствах: начиная от простейшего текстового редактора Блокнота в ОС Windows или gedit (pluma) в ОС Linux до сохранения документов из офисного пакета LibreOffice в формате html, так и в специализированных средствах разработки, таких как BlueFish (см. рис. 6.20), Macromedia HomeSite, Adobe® Dreamweaver®, Microsoft FrontPage, Adobe Creative Suite®, Serif WebPlus, Eclipse, Zend Studio и многих других.

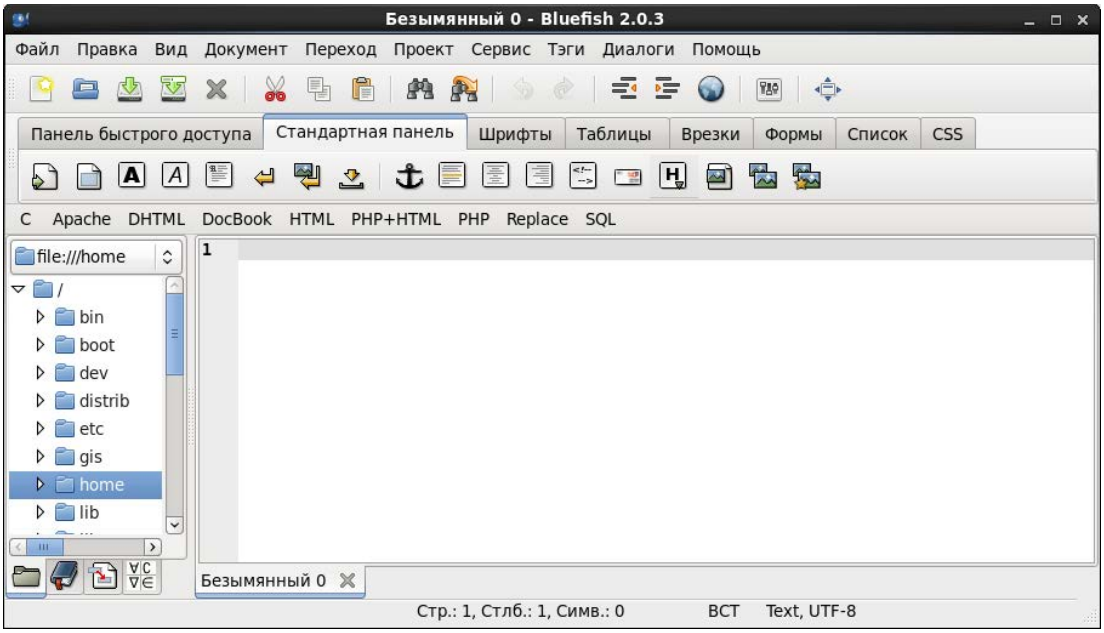


Рисунок 6.20. Свободный редактор BlueFish для создания HTML/PHP страниц

Для просмотра веб-страниц, как было сказано ранее, используются различные интернет-обозреватели (браузеры: Спутник, Яндекс.Браузер, Mozilla Firefox, Mozilla SeaMonkey, Google Chrome, Chromium, Safari, Konqueror, Lynx, Links, Internet Explorer, Opera и др.). Они отображают веб-страницы в соответствии с правилами их формирования, описанными в стандарте гипертекстовых документов HTML 4.01 (или HTML 5). Однако они могут показывать и простые текстовые файлы без специальной разметки, в чём нетрудно убедиться, открыв любой текстовый файл на просмотр в браузере, в ответ вы увидите простой текст, без эффектных стилей оформления, которые обычно присутствуют на веб-страницах. Для того чтобы научиться делать страницы, которые браузере будет отображать так, как вы захотите, следует изучить основы HTML-вёрстки.

Язык HTML (HyperText Markup Language – язык гипертекстовой разметки) позволяет сформировать структурированный текстовый файл, состоящий из отдельных фрагментов, начало и конец которых обозначается с использованием так называемых тегов. Теги применяются для задания форматирования или назначения тех или иных элементов веб-страницы. Особые теги используются для размещения на веб-страницах графических изображений, аудио- и видеоклипов и прочих так называемых внедренных объектов.

Теги пишутся в угловых скобках < > (то есть между знаками меньше и больше), они бывают парными и одиночными. Регистр в имени тегов не важен. Теги могут иметь атрибуты. Набор возможных тегов определён в стандарте HTML 4.01. В стандарте HTML 4.0 впервые появилась возможность использования таблиц стилей для конкретного элемента или для группы элементов. Стилевая информация может быть установлена в документе HTML или во внешней таблице стилей. В текст веб-страниц могут включаться сценарии (скрипты) на языках JavaScript, VBScript, TCL (Tool Command Language) для создания динамических страниц и использования HTML как средства создания сетевых приложений.

Приведём простейший пример HTML-документа, построенного с использованием тегов:

```
<html>
  <head>
    <title>Надпись на заголовке
окна</title>
  </head>
  <body>
    <h1>Заголовок 1</h1>
    <h2>Заголовок 2</h2>
    <p>Текст основной части
страницы</p>
    <p>Второй абзац основной части
страницы</p>
  </body>
</html>
```

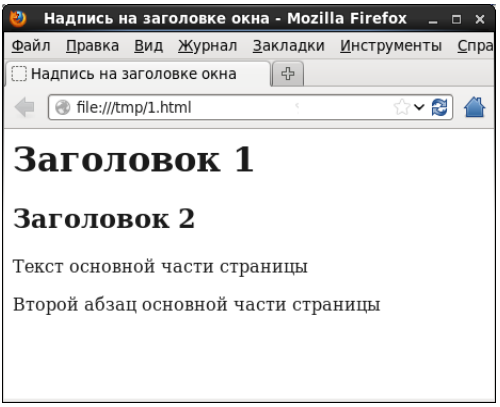


Рисунок 6.21. Пример простейшего HTML-документа

В данном примере документ состоит из раздела-заголовка между тегами **<head>** и **</head>** и тела – основной части веб-страницы между тегами **<body>** и **</body>**. Тело в этом примере состоит из фрагментов двух заголовков (теги **<h1>** и **</h1>**, **<h2>** и **</h2>**) и двух параграфов (теги ** **, ** **).

Секция **<head>...</head>** обычно содержит информацию о документе, такую как заголовок между тегами **<title>...</title>** (отображается в области заголовка окна), ключевые слова, которые могут оказаться полезными при использовании машин поиска, и другие данные, не являющиеся содержимым документа и не отображаемые на основной странице документа, например используемую кодировку и прочее:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

или

```
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
```

Если текст примера записать в файл с расширением *.html и открыть его, документ будет иметь вид, показанный на рис. 6.21.

Корректный документ HTML 4 состоит из трёх частей:

1) строка, содержащая информацию о версии HTML;
2) объявляющий раздел (тег **head**);
3) тело, содержащее собственно сам документ (теги **body** или **frameset**).

Таким образом, для полноценного функционирования предыдущего примера его нужно дополнить информацией о версии HTML, как это показано в следующем примере.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
   <TITLE>RU-CENTER Центр регистрации доменов</TITLE>
</head>
<body>
 <table>
  <tr>
   <td width="110">
    <P style="font-size: 18pt; color: Green;
        font-weight: bold;"
    <p><a href="http://www.nic.ru/">
       <img src="RU_center.png"
       alt="Эмблема RU-CENTER"></a> </p> </td>
   <td width="500">
    <td style="font-size: 18pt; color: Green;
        font-weight: bold;">
    <p>(размер 18, зелёный, жирный) </p>
    <p>RU-CENTER - аккредитованный Регистратор: </p>
   </td> </tr>
 </table>
 <ul>
  <li>
   <p style="font-size: 14pt;
      color: red;font-style: italic;">
   <p>в домене.RU –
     <a href="http://www.cctld.ru/ru/regru/"> Координационным центром
        национального домена Сети Интернет</a> </p>
   <LI>
     <p>в домене.SU –  <A href="http://www.fid.su/projects/Registrars/">
        Фондом Развития Интернет</A> </p>
   <LI>
      <p>в доменах NET, COM, ORG, BIZ, INFO -
          <a href=http://www.icann.org/registrars/>
         ICANN (Интернет-корпорация
  по распределению
  доменных имён и
  адресного
  пространства)</a>
      </p></li></ul>
  </body>
</html>
```

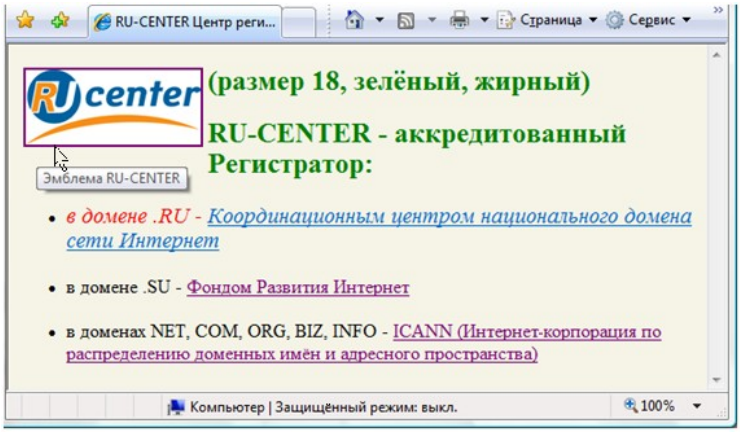


Рисунок 6.22. Пример 2, вид HTML-документа

Во втором примере информация размещена в таблице, состоящей из двух ячеек в одной строке. В первой ячейке находится ссылка на адрес `http://www.nic.ru/`, которая срабатывает при щелчке на рисунке из файла `RU_center.png`. Во второй ячейке таблицы присутствуют два абзаца, для которых заданы параметры шрифта с использованием внедрённых стилей. Ниже таблицы оформлен маркированный список со ссылками на веб-сайты.

Вид второго примера в веб-браузере показан на рис. 6.22.

Впрочем, знать в деталях стандарт HTML 4.01, его новую версию HTML 5 или новую спецификацию XHTML 2.0 веб-дизайнерам не обязательно – разработка html-страниц ведётся обычно в визуальном режиме. Простой пример – документ, подготовленный в LibreOffice Writer с использованием всех возможностей этой системы, сохранённый в формате «Веб-страница, *.html», реализует воспроизведение этого документа в веб-браузере с соблюдением всех требований стандарта HTML 4 с использованием внедрённых таблиц стилей CSS и xml-кода. Однако веб-страница, созданная таким образом, содержит очень большой объём описаний параметров шрифтов и стилей абзацев, предопределённых по умолчанию для стандартного шаблона документа. Поэтому для разработки веб-страниц лучше использовать специализированные средства, либо, если душа лежит к программированию, научиться писать чистый html-код в «Блокноте» или более удобных программах типа notepad ++.

Веб-страницы могут быть связаны с программами-скриптами, которые загружаются на клиентский компьютер вместе с документом HTML или присутствуют в нём. Программа выполняется на клиентской машине при загрузке документа или в другое время, когда, например, активируется гиперссылка.

Скрипты позволяют авторам расширить возможности документов HTML – создавать активные или интерактивные страницы. Скрипты могут выполняться по мере загрузки документа для динамической модификации содержимого этого документа, могут обрабатывать ввод данных пользователем, могут запускаться событиями, происходящими в документе: передача фокуса элементу, движение мыши, закрытие документа и т. п., могут быть связаны с элементами управления (например, кнопками, списками и прочим).

Созданные html-страницы, после их создания, загружают на веб-сервер, откуда они становятся доступными всем окружающим в сети.

HTML – это основы создания статических страниц, в реальности же почти всегда требуется чтобы страницы каждому пользователю генерировались индивидуально, поэтому для этой цели, преимущественно, используют язык PHP. Выполнение его сценариев происходит на стороне сервера, поэтому для создания сайтов и проверки их корректности работы одними блокнотом и браузером не обойтись, требуется «поднять» (установить, запустить и настроить) веб-сервер.

Денвер, WAMP, LAMP, XAMPP

Если вы пользователь ОС Windows, то перед переходом на более серьёзные ОС проще всего установить веб-сервер буедт используя Джентльменский набор Web-разработчика («Д.н.w.p», читается «Денвер», http://www.denwer.ru/) – проект Дмитрия Котерова, локальный сервер (Apache, PHP, MySQL, Perl, PHP, PostgreSQL и т. д.) и программная оболочка, используемые веб-разработчиками для разработки сайтов на «домашней» (локальной) Windows-машине без необходимости выхода в интернет.

Неофициально проект «Денвер» называют также WAMP от первых букв используемых в его составе программ (операционной системы **W**indows, веб-сервера **A**pache, СУБД MySQL и языка программирования **P**HP или Perl).

Идея создания аббревиатуры была взята с проекта LAMP – аббревиатуры, обозначающий набор широко используемого серверного программного обеспечения (см. рис. 6.23).

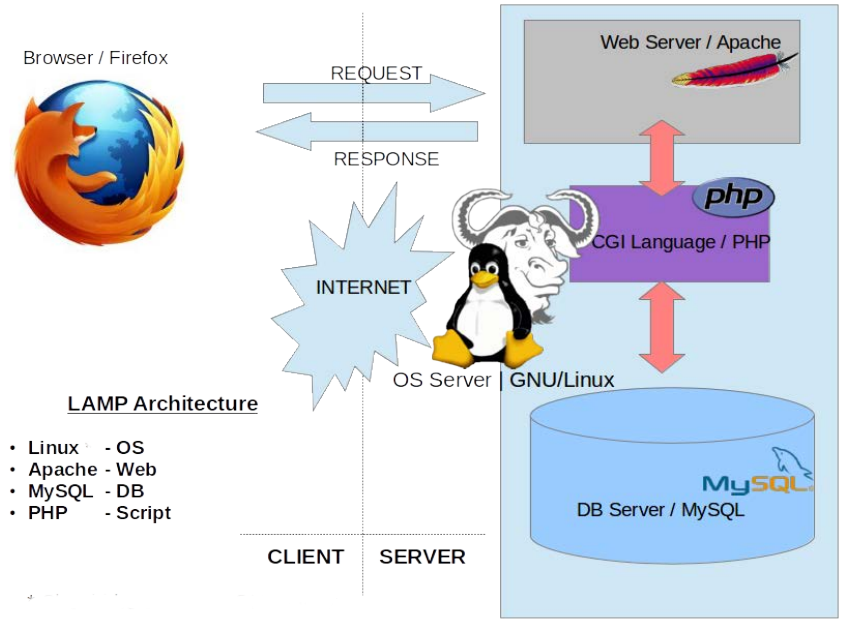


Рисунок 6.23. Типичное ПО веб-сервера и его взаимодействие

LAMP назван по первым буквам входящих в его состав компонентов:

- **L**inux – операционная система Linux;
- **A**pache – веб-сервер;
- **M**ariaDB / **M**ySQL – СУБД;
- **P**HP – язык программирования, используемый для создания веб-приложений (помимо PHP могут подразумеваться другие языки, такие как **P**erl и **P**ython).

По мере развития, функционал проекта LAMP был расширен, в него было добавлено большое количество дополнительных библиотек, позволяющих запустить полноценный веб-сервер. Название заменили на XAMPP, где было уже две буквы «P» от PHP и Perl, а X вначале символически указывал на кроссплатформенность проекта, что это «переменная», где вместо X может стоять любая операционная система из следующего набора: Windows 98/2000/XP/2003/Vista/7/8/8.1/10, Linux, Mac OS X или Solaris.

Кроме этого полный пакет содержит дополнительно FTP-клиент FileZilla, POP3/SMTP сервер, утилиту phpMyAdmin и др.

Указанный пакет программ свободно распространяется согласно лицензии GNU General Public License.

6.4. Общение и обмен информацией в интернете между пользователями

6.4.1. Электронная почта

E-mail (чаще email, «мыло»), электронная почта – один из самых популярных сервисов интернета. Она пришла как удобное дополнение к обычной почте, факсу и телеграфу. Конечно она не заменит запаха духов на открытке или письме, она не может передать вам мелкие вещи как посылку или бандероль с чем-нибудь новеньким, интересненьким или вкусненьким. У неё есть и конкуренты, пытающиеся её потеснить – SMS'ки, мессенджеры. Всё бы удалось, но формальное общение в офисной среде, а также внутренние (не зависящие от интернета) почтовые сервисы в различных фирмах и компаниях не дадут ей умереть ещё очень долго.

Первое электронное почтовое сообщение было отправлено в 1971 году, когда Рей Томлинсон создал программу, отправляющую текстовые сообщения с одного компьютера сети Arpanet на другой. По сути это было копирование текстовых файлов определённого формата из

одной папки одного компьютера в другую папку на другом компьютере через специально сделанный для этих целей интерфейс. И всего через год эти «копирования» сообщений электронной почты заняли 3/4 от всего объёма передаваемого сетевого трафика на тот момент.

Поскольку основными пользователями компьютеров были люди умеющие программировать, в 1975 году появилась первая электронная рассылка, то есть периодическая отправка на определённые адреса электронных писем. Вскоре в рассылках стали публиковаться письма подписчиков, на которые другие читатели отвечали также через рассылку. Так родилась UseNet – система электронных конференций.

В настоящее время для приёма и хранения писем используются почтовые серверы (MTA, mail transfer agent), часто имеющие ещё и функцию веб-сервиса.

Первоначально для работы с этими почтовыми серверами использовались клиентские программы подготовки, получения, передачи и хранения полученных сообщений, называемые MUA (mail user agent) или постовые клиенты. Сегодня эти программы заменил веб-интерфейс сайтов обеспечивающих доступ пользователям к их почтовым ящикам. При наличии постоянного высокоскоростного соединения с интернетом и постовым сервисом этот вариант оказывается удобнее. Если же интернет не постоянен, то, как ни крути, придётся воспользоваться постовым клиентом, сообщения получать и отправлять в определённые моменты времени и работать с ними уже в режиме «offline».

Основные функции почтовых клиентов – приём писем, обеспечение их просмотра, распределение писем по папкам (Входящие, Отправленные, Удалённые и прочие), автоматизация создания новых писем и поддержка адресной книги. При подготовке электронного письма пользователь готовит текст сообщения и заполняет ряд стандартных полей: «To» (Кому), «CC» (Копия), «BCC – Blind Carbon Copy» (Невидимая копия). Разница между полями CC и BCC заключается в том, что все те адреса, которые вы укажете в поле CC, будут видны всем участникам переписки, а те, что вы укажете в поле BCC, не будут видны всем адресатам (каждый будет видеть только свой адрес).

Пользователи электронной почты регистрируются на сервере, каждому из них даётся адрес (например, *a.ivanov@narfu.ru*, где a.ivanov – имя пользователя и его почтового ящика, narfu.ru – имя домена, в котором находится почтовая служба).

Для работы специальных почтовых серверов и клиентов используются специальные протоколы, основные из них – **POP3, IMAP, SMTP**.

«Классикой жанра» для почты считается протокол **POP3** (Post Office Protocol – почтовый протокол, версия 3). Его серверная поддержка в ОС Linux возможна программой Dovecot. Протокол **POP3** предусматривает обращение клиента с предложением забрать пришедшие письма, сообщение серверу имени ящика и пароля, загрузку писем на машину пользователя и удаление их из ящика.

Удобство такого способа получения почты ранее состояло в том, что почта получалась клиентом, после чего она не хранилась на сервере и доступ к интернету не требовался. Можно было получать и отправлять почту один или два раза в день, скажем утром и вечером, соединившись с интернетом или напрямую с почтовым сервером по модему.

Протокол POP3 актуален и сейчас в больших фирмах, и, например, если вы часто делаете перелёты на самолётах, в которых нет интернета, а почта нужна. В ряде случа-

ев вместо протокола POP3 может использоваться его защищённый собрат – протокол **POP3S**, в котором данные передаются в зашифрованном виде.

Для предоставления возможности выборочной работы с почтой есть вторая «классика жанра» – протокол **IMAP** (Internet Mail Access Protocol) или его защищённая версия **IMAPS**. В этом случае пользователь может просмотреть информацию о полученных сообщениях, об отправителях, узнать размеры, темы и выборочно загрузить необходимые письма с разных компьютером потому как письма после их получения продолжают хранится на сервере. На сервере они распределены по папкам, пользователь может создавать новые папки и перемещать письма из одной в другую. Поддержкой протокола **IMAP**, хотя и не в полном объёме, обладает большинство современных почтовых клиентов.

В принципе, для получения сообщений достаточно наличия в сети только POP3/IMAP-серверов. Почтовый отправитель переправит письмо серверу получателя, например копированием файла, однако, если почтовый сервер адресата по каким-то причинам окажется неработоспособным, то письмо отправлено не будет и придётся ждать, когда тот снова начнёт функционировать. Для решения этой проблемы был создан **SMTP**-протокол (Simple Mail Transfer Protocol – простой протокол пересылки почты), по которому работают все **SMTP**-серверы исходящей почты, занимающиеся накоплением отправляемых писем и обеспечением их доставки адресатам. После помещения пользователем на **SMTP**-сервер писем тот начинает запрашивать почтовые серверы, которым эти сообщения адресованы, на предмет готовности принять эти письма. Если сервер адресата не готов, то **SMTP**-сервер пробует соединиться с ним через некоторое время, а если он недоступен на длительный срок – возвращает письмо отправителю (на обратный адрес, указанный в письме). Кроме того, при загрузке сообщений на **SMTP**-сервер проводится анализ адресов получателей и выясняется, существуют ли такие адреса в принципе. Если нет, сервер возвращает письмо отправителю с информацией о том, что получатель не найден.

Вместе с усовершенствованием механизмов передачи электронных посланий менялась и структура самого письма. Первоначально письма представляли собой простые текстовые сообщения. Однако впоследствии был разработан так называемый стандарт **MIME** – Multipurpose Internet Mail Extensions (многоцелевые расширения почтового стандарта интернета). В соответствии с этим стандартом электронное письмо может состоять из нескольких частей, содержащих свой пакет данных в отдельном формате. Так, с помощью **MIME** можно одновременно поместить в письмо текст, вебстраницу, графический, звуковой либо исполняемый файл или архив. Каждая часть будет иметь свой заголовок, в котором обозначен формат этой части. Почтовый клиент способен анализировать заголовки и соответствующим образом воспроизводить фрагменты письма.

В состав большинства операционных систем входят стандартные для каждой системы почтовые клиенты, например в ОС Linux – Evolution, имеющие возможность работы с электронной почтой по протоколам **POP3**, **IMAP**. Существуют и альтернативные почтовые программы, например кросс-платформенная, свободная, бесплатная и расширяемая Thunderbird [401]. Есть и коммерческие программы для ОС Windows: Eudora, Outlook, The Bat! и др.

[401] http://www.mozilla.org/ru/thunderbird/.

При настройке почтовой программы необходимо указать имена почтовых серверов для входящей почты (напр., pop3.narfu.ru) и исходящей почты (напр., smtp.narfu.ru).

В состав почтовых систем может включаться функция шифрования. Самыми распространёнными системами шифрования являются **S/MIME**, **PEM** (privacy enhanced mail – почта с повышенной конфиденциальностью) и **PGP** (pretty good privacy – достаточно высокая степень конфиденциальности). **S/MIME** – стандартная система шифрования, которую поддерживают многие популярные программы.

У многих бесплатных и платных почтовых серверов существует такая услуга, как «**веб-почта**», точнее веб-интерфейс для работы с почтой, когда для доступа к почте вы используете обычный веб-обозреватель в качестве почтового клиента и программы создания почтовых сообщений. Удобство данной системы в том, что доступ к сообщениям в почтовом ящике можно получить с любого компьютера (планшета и прочего), подключенного к интернету без установки и настройки каких-либо дополнительных программ. Минус – требуется постоянная связь с интернетом.

В Сети достаточно много широко известных специализированных почтовых сайтов и поисковиков с почтовыми сервисами, предоставляющих бесплатные почтовые услуги для личного использования всем желающим. Например, Яндекс.Почта (mail.yandex.ru), *www.hotmail.com,* служба Gmail на Google, *www.mail.ru, www.zmail.ru* и др.

После регистрации на подобных сервисах вам предоставляется почтовый ящик некоторого конечного размера. Желая привлечь к себе больше клиентов, указанная цифра постоянно увеличивается, вплоть до нескольких десятков и даже сотен гигабайт. (Не забываем, бесплатный сыр бывает только мышеловке!) В некоторых случаях работает и автоматическое увеличение размера ящика, например если в процессе использования в почтовом ящике остаётся менее 100 МБ, то пользователь может увеличить его объём ещё на 2 ГБ. Если со временем и этого места не будет хватать, то возможно и дальнейшее поэтапное расширение. Максимальный размер письма варьируется от системы к системе. Лет десять назад дурным тоном считалось посылать письма размером более 20 МБ, двадцать – более 1-2 МБ. Сейчас у ряда сервисов стоят ограничения в районе 50-100 МБ, а у некоторых и вовсе нет.

Дешевизна отправки сообщений по сравнению с бумажной почтой привела к появлению такого явления как спам или нежелательные почтовые сообщения, обычно содержание рекламную информацию. Не смотря на совершенствование спам фильтров на серверной стороне подобные сообщения до сих пор регулярно приходят.

Веб-почта предоставляет все типичные функции электронной почты. На веб-страничке пользователю предлагается ввести имя и пароль, после чего он видит папки своих писем «Входящие», «Отправленные», «Черновики», «Удалённые» и др. и может читать выбранные письма, создавать и отправлять новые письма.

К недостаткам бесплатной веб-почты следует отнести то, что на почтовых сайтах всегда присутствует много рекламы, в том числе и контекстной, исходя из информации, полученной из ваших же сообщений.

Веб-почта может использовать для установления связи протоколы с шифрованием **SSL/TLS** (Secure Sockets Layer / Transport Layer Security) – **HTTPS**, что обеспечивает конфиденциальность сообщений.

Удобный сервис «почта на 10 минут». В интернете существуют сайты которые создают временные почтовые ящики и позволяют вам на них получать сообщения. Отправлять таким образом почту нельзя, а получать можно. Скажем, если вам нужно зарегистрироваться на каком-то форуме (форум высылает вам на почту пароль или ссылку-подтверждение регистрации), а свой основной ящик вы использовать не хотите или у вас его попросту нет, то вы можете зайти на сайт, например http://10minutemail.org, узнать свой временный email-адрес и получить на него письма. Если 10 минут мало, то существует кнопка продления времени пользования ящиком.

6.4.2. RSS-каналы

RSS-каналы – новый вид представления списков новостей в интернете. RSS включает в себя несколько стандартов, в том числе RSS 0.91 – **R**ich **S**ite **S**ummary (перечнь «богатых» сайтов), RSS 0.9, 1.0 и 1.1 – **R**DF **S**ite **S**ummary (библиотека описаний ресурсов перечня сайтов) и RSS 2.0 – **R**eally **S**imple **S**yndication (очень простое приобретение информации, или действительно простая раздача новостей). Atom – ещё одна XML-структура, используемая для распространения содержимого.

Основа версии RSS 2.0 – формат XML версии 1.0. Более старые версии RSS 0.9× и RSS 1.0 использовали форматы XML и RDF (см. *http://rdfabout.com*). Существует несовместимость форматов 2.0 и 0.9x, 1.0. Многие современные браузеры, почтовые клиенты и интернет-пейджеры умеют работать с RSS-каналами, среди них Microsoft Internet Explorer (начиная с версии 7), Microsoft Office Outlook, Opera, Mozilla Firefox, Miranda, Safari, Maxthon и др. Кроме того, существуют специализированные приложения (RSS-агрегаторы), собирающие и обрабатывающие информацию RSS-каналов. Также очень популярны веб-агрегаторы, представляющие собой сайты по сбору и отображению RSS-каналов, такие как Яндекс.Лента, Google Reader, Новотека, Bloglines и прочие.

RSS-канал отображается в браузере, почтовом клиенте или агрегаторе как список заголовков. Заголовки обновляются автоматически. Выбрав любой заголовок, можно прочитать информацию по этой теме.

Преимуществом RSS является сосредоточение всей информации по выбранной теме канала из нескольких веб-источников в одном месте. В ряде случаев можно не посещать информационные веб-узлы в поисках информации, с помощью RSS можно получить краткое изложение содержимого веб-узлов и решить, какие именно статьи следует прочитать, щёлкнув соответствующую ссылку.

RSS-каналы можно найти с помощью специализированных поисковых систем. В интернете существуют веб-узлы, которые ведут большие списки доступных RSS-каналов, например по адресу *http://subscribe.ru/catalog?rss* размещён RSS-каталог, содержащий около 179 тыс. ссылок на RSS-ленты, сгруппированные по следующим тематикам: *Открыто недавно*, *Автомобили*, *Бизнес и карьера*, *Дом и семья*, *Мир женщины*,

Hi-Tech, *Компьютеры и интернет*, *Культура, стиль жизни*, *Новости и СМИ*, *Общество*, *Прогноз погоды*, *Спорт*, *Страны и регионы*, *Туризм*, *Экономика и финансы*, *Email-маркетинг*, *Игры*.

Для просмотра RSS-каналов существует множество как платных, так и бесплатных приложений. Наиболее удобны кросс-платформенные решения от mozilla.org: так же как браузер Firefox, Thunderbird может использоваться для доступа к веб-сайтам, которые делают своё содержимое доступным через RSS-каналы. Однако, в отличие от Firefox, который даёт вам доступ к RSS-каналам через Закладки, Thunderbird позволяет вам просматривать содержимое RSS путём, подобным чтению электронной почты: RSS-канал, на который вы подписались, будет отображен в области окна папок, индивидуальные заголовки статей или «заголовки» будут отображены в области окна списка сообщения, и когда вы щёлкнете по заголовку, содержание статьи будет отображено в области окна предварительного просмотра сообщения. Подробнее см. http://mozilla-russia.org/products/thunderbird/support/ rss.html.

6.4.3. Twitter

Твиттер (от англ. Twitter – «чирикать», «щебетать», «болтать») – система, позволяющая пользователям отправлять короткие текстовые заметки (до 140 символов[402]), используя веб-интерфейс, SMS, средства мгновенного обмена сообщениями или сторонние программы-клиенты. Характерной особенностью Твиттера является публичная доступность размещённых сообщений; это называется микроблоггингом. Хотя услуга является бесплатной, доступ к ней через SMS может значительно увеличить телефонные счета, так как каждое посланное SMS так или иначе оплачивается по тарифам оператора. Владельцем системы Твиттер является зарубежная компания Twitter Inc.

Твиттер часто используется для передачи новостей как личного, так и общественного значения, в том числе и с целью рекламы политиками, актёрами.

Все сообщения хранятся в централизованной базе, доступ к которой осуществляется различными способами. Чаще всего twitter-клиенты в режиме online поддерживают связь с базой и отображают пользователю все последние изменения в ней. Поскольку изменений очень много, пользователи выбирают за какой именно информацией они хотели бы следить, тем самым становятся «фолловер'ами»[403], подписавшись на обновления аккаунтов тех или иных пользователей.

Возможность поиска по уже опубликованным сообщениям, беспрерывные копирования интересных сообщений (репосты) породили использование в сообщениях двух значков: «@» и «#», придав им следующий смысл.

@ – означает цитирование. Например, если в сообщении встречается комбинация символов «@vasya:», то это значит, что следующий за ним текст изначально принадлежал пользователю с аккаунтом vasya.

– используется для задания «хэштегов»[404] – слов или словосочетаний с решёткой перед ними. Служат для облегчения последующего поиска по сообщениям. Пользователи заранее договариваются, либо им через рекламу ~~навязы-~~

[402] 11 августа 2015 года ограничение в 140 символов было отменено.

[403] От англ. follower, следующий (за сообщениями), последователь, читатель, подписчик.

[404] От англ. hashtag: hash – символ решётки (#) + tag (тег). Синонимы: канал, тема, топик, ветка обсуждений, пометка.

ваются предлагаются варианты таких тегов, например #abc_conference2015. Пользователи размещают эти теги в любом месте своих сообщений, обычно в конце. После чего не составляет труда отобрать все сообщения всех пользователей теме «Конференции abc 2015 года». В одном сообщении может быть несколько, например, к тегу выше может быть добавлен ещё один – #Omsk, – по месту проведения конференции, и так далее.

Свобода выбора тегов (в рамках приличий) позволяет пользователям придумывать и использовать забавные, порой поднимающие всем настроение теги, без всякого на то смысла или связи с первоначальным сообщением, типа #покрась_волосы_в_красный_цвет. Что интересно, подобная «глупость» зачастую становится очень популярной, но по мере времени она забывается, заменяясь, новой, более радикальной.

Сайт твиттера, как и twitter-клиенты, показывают цитирования и теги ссылками, поэтому каждый без труда может перейти по ним и наблюдать, как другие пользователи оставляют (или оставляли) свои твиты с использованием этого хэштега. Так без особых усилий формируется (получается) тематический канал. Если вам надоело следить за каким-либо каналом или пользователем вы можете от него отписаться.

Многие социальные группы, например друзья по классу в школе или по группе в вузе, лояльные клиенты какой-либо фирмы и тому подобные, находят это средство удобным в своей повседневной жизни. Например, преподаватель узнал, что учебная часть перенесла занятия – скинул сообщение в группу студентов. Фирма сделала спецпредложение к празднику – уведомило своих клиентов.

Единственный недостаток такого общения, – не проработанные формальная и социальная стороны вопроса. Между людьми существует большой социальный разрыв: между теми кто пользуется этой технологией и теми кто нет.

Например, если преподаватель начнёт скидывать домашние задания по twitter'у студентам, то это может привести к дискриминации, так как ставит одних в более выгодное положение по отношению к другим. При таком подходе все человеческие качества уходят на второй план и важным становится то есть ли у тебя устройство получения сообщений или нет. К сожалению, в этом плане это плохая асоциальная тенденция.

6.4.4. Общение в реальном времени

По сравнению с RSS-рассылками, общение через twitter настолько быстрое, что его можно в полной мере считать общением в реальном времени. С момента появления компьютерных сетей и интернета и по настоящее время, ещё за долго до появления твиттера, множество людей как-то использовали эти средства коммуникации для обмена новостями, своими мнениями и знаниями, спрашивали совета у специалистов и просто разговаривали на всевозможные темы.

Первоначально главным средством для этого были электронная почта и серверы списков рассылки. Серверы рассылки формируют группы списков по различным тематикам, к этим спискам адресатов может присоединиться любой желающий, и он будет получать все сообщения членов этой группы и сам сможет отвечать в эту группу, его сообщения будут рассылаться по всем адресам в списке группы. В настоящее время такой способ общения известен как группы новостей (News), конференции или форумы на интернет-страницах.

Другие способы общения – службы мгновенных сообщений (мессенджеры, интернет-пейджеры) и чат (chat) двух видов: IRC (Internet Relay Chat) и веб-чат.

В ряде случаев сервисы используемые для общения могут обеспечивать своим пользователям связь с абонентами других сетей, например совершать звонки на обычные телефоны. В этом, а также в других случаях, они могут с правовой точки зрения рассматриваться как операторы связи[405] со всеми вытекающими из этого последствиями (необходимость получения лицензии и т.п.).

6.4.4.1. Службы мгновенных сообщений (ICQ и др.)

Программы обмена мгновенными сообщениями (Instant Messaging, IM, мессенджеры) позволяют пользователям в интернете устанавливать связь друг с другом и общаться в реальном времени или посылать сообщения неактивному пользователю, который получит их после подключения к системе. Исторически самой известной **IM**-программой (интернет-пейджером) является программа **ICQ**. Само название ICQ является английским омофоном фразы «I seek you» («я ищу тебя»), в РФ прижилось «аська» – сленговое название любого ICQ-клиента.

Систему **ICQ** разработали четыре молодых программиста из Тель-Авива (Израиль): Сефи Вигисер, Арик Варди, Яир Гольдфингер и Амнон Амир. Они основали свою компанию Mirabilis (лат. чудесный, удивительный), и в 1996 году система начала своё распространение в интернете. Программное обеспечение изначально распространялось бесплатно (в отличие от конкурентов). В 1997 году компания объявила о том, что количество пользователей программы достигло миллиона. В 1998 году компания Mirabilis сообщила о том, что сеть **ICQ** растёт со скоростью 1 млн пользователей за 23 дня. Количество одновременных подключений составляло ~ 500 тысяч. В 1998 году компания Mirabilis продала права на ICQ крупнейшему американскому провайдеру AOL (America Online) за 407 млн долларов. В результате компания Mirabilis была преобразована в ICQ Inc. и стала частью корпорации America Online. В итоге последовательной смены собственников в настоящее время принадлежит инвестиционному фонду Mail.ru Group (Россия).

Если в 90-х наблюдался рост числа пользователей и модно было спросить «А у тебя есть аська?», то в 2000-е наблюдался спад популярности. Сегодня многие интернет-пользователи даже и не знают, что это такое – «аська». По данным Mail.ru Group, ежемесячная аудитория ICQ во всем мире с декабря 2009 года по декабрь 2010 года сократилась на 35% (17,6 млн чел.) и составила 33,5 млн человек. В России к декабрю 2010 года аудитория составила 16,4 млн человек (сократилась за год на 9%). Согласно отчёту, опубликованному 5 сентября 2012 года Mail.ru Group (http://corp.mail.ru/en/IR/news/1482), общемировая месячная аудитория ICQ снизилась за период с июня 2011 по июнь 2012 года с 30,8 до 20,5 млн человек, в числе которых 10,8 млн российских пользователей, потеряв таким образом за год ещё треть своей аудитории.

Падение аудитории ICQ – часть общемировой тенденции: всё большее количество пользователей интернета предпочитают использовать в качестве средства коммуникации социальные сети. Отщепили свою долю пользователей и Твиттер, и Mail.ru-агент.

[405] См. закон «О связи» от 07 июля 2003 года № 126-ФЗ.

Непродуманная маркетинговая политика в борьбе со спамом привела к тому, что несколько раз были отключены от сети различные неоригинальные клиенты, что явно не добавило популярности и вызвало отток пользователей, в частности в Jabber, где отсутствуют централизованные серверы и такое в принципе не возможно.

Несмотря на падение популярности, данное средство до сих пор используется многими. Помимо основных функций обмена сообщениями и поиска зарегистрированных пользователей по задаваемым параметрам (фамилия, имя, прозвище (ник), город, пол, возраст и прочее – см. рис. 6.24),

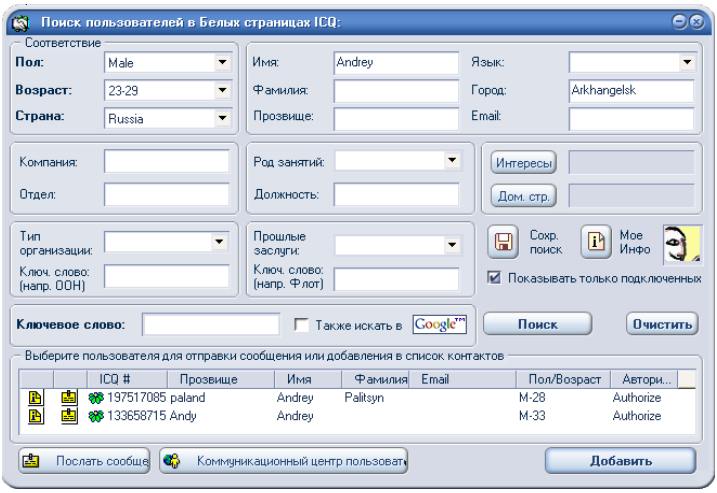


Рисунок 6.24. Поиск пользователей в базе ICQ

ICQ может также выполнять большое количество других задач, таких как ICQ-чат, организация видеоконференций, интернет-телефония, отправка SMS-сообщений, передача файлов и т. д. Для работы с программой новому пользователю необходимо установить приложение, а оно в свою очередь, установит online-связь с одним из серверов этой системы и зарегистрироваться в системе, после чего он получает уникальный 9-значный номер в системе (**UIN**) [406].

Программное обеспечение для работы с сервисом не входит в состав большинства ОС и может быть бесплатно загружено с сайта производителя (http://www.icq.com/ru) и бесплатно использоваться.

Кроме официального клиентского приложения существуют многофункциональные клиенты (для разных ОС, в том числе для iOS и Android) [407], поддерживающие несколько протоколов систем обмена мгновенными сообщениями, в том числе и ICQ:

- **Miranda IM** – протоколы ICQ, IRC, AIM, MSN, Jabber, Google Talk, Yahoo, Skype, Gadu-Gadu, Tlen, Netsend и прочие;
- **SIM IM** – протоколы ICQ, AIM, MSN, Jabber, Yahoo и прочие;

[406] По мере обращения к сервису регистрации, номера аккаунтам присваиваются последовательно. Ранее использовались более короткие номера. Какое-то время существовала мода: «чем меньше номер, тем круче», поэтому, в Сети и сейчас можно видеть объявления о продаже коротких и/или красивых ICQ-номеров от 30 до 1000 руб.

[407] Некоторой популярностью обладает объединённый клиент WhatsApp (http://www.whatsapp.com/) недоступный для стационарных ПК, но доступный для «мобильных» iOS, BlackBerry, Android, Windows Phone и Nokia (Symbian и S40).

- **Trillian** – протоколы ICQ, MSN, Jabber, Yahoo и прочие;
- **QIP** – поддерживает ICQ, Jabber и прочие;
- **Pidgin** – протоколы ICQ, IRC, AIM, MSN, Jabber, Yahoo, Gadu-Gadu и прочие;
- **Instantbird** – протоколы XMPP, Google Talk, ICQ, Yahoo!, AIM, MSN, Gadu-Gadu и прочие.

В 2021-м году сервис ICQ получил новый толчок в своём развитии, поскольку приложение ICQ оказалось включённым в Перечень российских программ для электронных вычислительных машин, которые должны быть предварительно установлены на отдельные виды технически сложных товаров, утверждённый распоряжением Правительства Российской Федерации от 31 декабря 2020 года № 3704-р. Так что, на новых устройствах вам ничего не придётся дополнительно скачивать и/или устанавливать.

Аналогичный ICQ функционал предоставляют и другие программы, работающие по своим протоколам, подробнее о них рассказано в следующем параграфе.

6.4.4.2. Альтернатива времени: Whatsapp, Viber, Telegram и др.

К сожалению, какой бы хорошей технология не была, её поддержка требует усилий, которые всё чаще становятся «коммерческими». Деньги «берут» прямо или косвенно. Либо сервис становится платным, либо «продают аккаунты пользователей» – появляется реклама. Появляются недовольные, возникают новые схожие сервисы и отток пользователей с одновременным появлением новых возможностей, соразмерных с запросами времени. Старые сервисы стагнируют, либо умирают совсем, уступая место новым.

Так, в последнее время появилось несколько популярных альтернатив ICQ – это сервисы Whatsapp, Viber, Telegram и др. делающие вид, что конкурируют между собой, как производители автомобилей двух схожих марок. Молодёжь массово пользуется новыми сервисами, не беря в голову, что их идея стара как мир, точнее как ICQ и интернет.

Технически обмен сообщениями происходит подобно ICQ, но современными средствами. Под современными понимается то, что ранее компьютеры были преимущественно стационарными и об их использовании при поездке на эскалаторе в метро или идя пешком по лестнице не было и речи. Сейчас же наличие планшета-телефона в кармане позволяет взаимодействовать с ЭВМ и её программами, как и интернет-сервисами в прямом смысле «на ходу». ICQ-клиент для мобильного телефона работает так же как Viber или Whatsapp.

Отличие сервисов состоит в различных подходах при регистрации и выборе номеров аккаунтов. Если в ICQ надо было регистрироваться самому на сайте или через клиента и помнить свой номер, чтобы после сообщить его друзьям, то в Whatsapp и Viber вам присваивается номер аналогичный номеру вашего телефона. После запуска программы автоматически делается запрос на сервер и вам приходит SMS-сообщение с кодом (со ссылкой) для активации. Достаточно «подтвердить» сообщение-запрос, ткнув по экрану пальцем и вы готовы к работе. Поскольку при общении с сервером на последнем формируется уникальный список номеров телефонов всех пользователей, то

это позволяет программе сразу после установки просмотреть всю вашу записную книжку и отправить её на сервер для поиска уже зарегистрированных номеров. Таким образом, сервер узнаёт все ваши контакты и, если ваш друг «Вася» пользуется сервисом Whatsapp или Viber, то напротив его записи появляется соответствующий значок. Естественно что из двух разных сервисов, работающих аналогично, каждый знает только о своих.

Поскольку SMS-сообщения обычно платные, а интернет у пользователей или безлимитный или они пользуются бесплатным wi-fi, посылка сообщений через Whatsapp, Viber или Telegram бесплатно пользуется спросом. К тому же, у этих сервисов нет ограничений ни в 70, ни в 140, ни в 160 символов. Ещё они поддерживают голосовые и видео-звонки, подобно Skype.

То есть, при наличии интернета получается следующая картина: пользуемся как SMS, но бесплатно, можем посылать больше знаков, можем прикреплять картинки, смайлики, видео и звуки; пользуется как телефоном и даже совершаем видеозвонки, но при этом не платим за телефонный трафик. «Сыр в мышеловке» подействовал. Число пользователей Whatsapp или Viber неуклонно растёт. Особенно молодёжь и дети, чей баланс редко превышает 0, при общении предпочитают меняться контактами Whatsapp'а или Viber'а, поскольку при наличии бесплатного интернета в общественных местах, городском транспорте, школе и прочих местах можно быть «всегда на связи».

В дополнение к функционалу традиционных сервисов: обмен текстовыми сообщениями, совершение голосового звонка, почти во всех мессенджерах появились дополнительные возможности в разных комбинациях или все сразу:

- оставление голосового сообщения (если у вас руки заняты или вам лень печатать);
- совершение видеозвонка;
- создание голосовых и видео конференций для общения трёх и более абонентов;'
- создание тематических групп (каналов);
- отправка друг другу неограниченного количества файлов (изображений, видео и аудио мультимедийных сообщений и т.п.);
- хранение истории сообщений на сервере;
- хранение списка абонентов на сервере;
- возможность использования одного аккаунта на нескольких устройствах;
- наличие web-версии мессенджера.

Техническое отличие систем между собой скорее внешнее, как то, наличие клиентов для той или иной системы, поддержка шифрования, различия в дизайне и прочие мелочи. В интернете присутствует много информации о сравнении популярных IM-сервисов и она легко находится. Эти сравнения редко имеют объективный научный вид и достаточную глубину, но при этом они содержат достаточно полезной информации о сервисах. Объясняется это тем, что с жизненной точки зрения важным оказывается не то, как была реализована база данных сервера, была ли она реляционной или нет, *n*-й версии, как у конкурентов, или (*n+1)*-й, или существует ли клиентское приложение для такой-то ОС,а то, где физически размещается сервер обслуживающий абонентов и где хранится их переписка.

Практически для всех стран вопрос использования тех или иных мессенджеров на своей территории коррелируется с вопросом их цифрового суверенитета. А именно,

будет ли гарантироваться гражданам этой страны соблюдение их прав и свобод. Например, то, что нам гарантировано нашей конституцией, что охраняется уголовным кодексом[408]. Добропорядочные организации не будут покушаться на ваши свободы, но ни смогут запустить алгоритмы анализа сообщений и создать на базе неё обезличенную карту интересов и характеристик клиентов, которую можно продавать рекламным компаниям для совершения таргетированной рекламы. Понятно, что реклама запчастей и расходников для мотоциклов будет иметь больший эффект среди владельцев двухколёсных агрегатов, состоящих в группах «мото», «мототюнинг» и т.п., чем среди тех, у кого нет ни прав необходимой категории, нет ни одной фотографии с мотоциклом и в аккаунтах которых слова вроде «ремонт», «мотосервис» не встречаются.

Как мы понимаем, слово «ремонт» не показательно, так как человек мог недавно обсуждать ремонт чего угодно, например, ботинок, поэтому совершенствовать эффективность работы подобных алгоритмов можно постоянно, что и и происходит в действительности. Тем, кому эта тема интересна рекомендуем заняться изучением публикаций в области Big Data и для начала посмотреть выступления специалиста из этой области – Артура Хачуяна (компания SocialDataHub)[409].

Поскольку клиенты ICQ изначально существовали для стационарных ПК и это многим казалось удобным, то с Whatsapp, Viber, Telegram и др., родившимся «в мобильных устройствах», наблюдается обратная миграция. Нередко тем или иным мессенджером невозможно воспользоваться на стационарном компьютере, другие же наоборот, например Viber, изначально выпустил клиенты для всех популярных стационарных и мобильных платформ, в том числе и 64-битных: Linux, Windows, Mac, iOS, Android, BlackBerry и Nokia. Неудобство лишь в первоначальной регистрации через получение SMS. WhatsApp пока отстаёт и таким широким охватом систем похвастаться не может. Запуск WhatsApp-клиента для стационарного компьютера решается следующим образом, для ОС Linux – запуском альтернативного клиента, через добавление плагина к программе Pidgin[410], для ОС Windows – с использованием эмуляторов ОС Android[411], например BlueStacks[412].

Что интересно, в начале 2014 года сервис WhatsApp был куплен[413] компанией Facebook, как перспективный, и в тоже время, WhatsApp начал проводить агрессивную политику в адрес открытых решений, реализующих его протокол[414].

Приведём названия некоторых популярных на сегодня мессенджеров (кроме трёх из названия параграфа), уверены, что некоторые из них вы даже не слышали: Briar, Dialog Messenger, Dicrod, Disa, Franz, ginlo, Facebook Messenger, FaceTime, Hangouts, iMessage, Imo, KakaoTalk, Kik, Line, Matrix.org, Mattermost, Messages, OK, Pinger, QIP Infium, Riot.im, Ring, Signal, Skype, Slack, Snapchat, TamTam, TeamTalk.io, textPlus, Threema, TikTok, Tinder, Tox, Trillian, WeChat, Wire, Zoom, Zulip и др. За отдельными из них закрепился определённый, выдающих их, функционал, который стоит прокоммен-

[408] Вспомните ст.138 из УК РФ «Нарушение тайны переписки, телефонных переговоров, почтовых, телеграфных или иных сообщений».

[409] Например, от 22 июня 2018 г. «Зачем за нами следят в соцсетях и кто продает наши данные» https://www.youtube.com/watch?v=LZIpsq1YyBg.

[410] Подробнее см. http://www.ubuntu.sumy.ua/2014/05/whatsapp-в-ubuntu-linux-регистрация-и-использование.html, http://dnevniki.ykt.ru/ИванПитерский/705288.

[411] Установка WhatsApp-клиента http://whatsapps-pc.com/10-whatsapp-dlya-kompyutera.html.

[412] Эмулятор ОС Android для ОС Windows http://www.bluestacks.com/.

[413] Facebook покупает WhatsApp за $16 млрд http://habrahabr.ru/post/213245/.

[414] WhatsApp вышел на тропу войны http://habrahabr.ru/post/212957/.

тировать: Matrix.org – открытый стандарт для децентрализованной постоянной связи по IP; TikTok – сервис для создания и просмотра коротких видео; Tinder – сервис для знакомств с людьми в соответствии с заданными параметрами и с учётом геолокации. Отличительная особенность интерфейса: основными действиями в приложении являются пролистывания – «свайпы»[415].

Все мессенджеры поддерживают одновременное общение более двух абонентов, однако, коллективное общение в интернете зародилось намного раньше и имеет свои особенностями. Подробнее об этом рассказано в следующем параграфе.

6.4.4.3. Коллективное виртуальное общение

Если **IM**-сети в основном предназначены для индивидуального общения, то программы организации **чатов, веб-форумов** и **веб-конференций**, **гостевых книг**, **блогов, твиттер** – для общения групп людей, имеющих какие-либо общие интересы. Рассмотрим подробнее предлагаемые возможности.

Чат (chat) – специальное программное средство для организации виртуального общения одновременно большого числа людей в интернете – существует в двух разных видах:

1. **IRC** (англ. *Internet Relay Chat*) – протокол прикладного уровня для обмена сообщениями в режиме реального времени [416]. IRC использует специальные серверы и клиенты для своей работы, подключенные к интернету;
2. **Веб-чат** – реализован как программное обеспечение, работающее на обычных веб-страницах.

У этих двух видов много общего: для участия в разговорах нужно зарегистрироваться и задать себе ник (псевдоним), разговоры ведутся в общих тематических «комнатах» – окнах системы (или каналах в IRC), можно уйти в «приват» – поговорить тет-а-тет в отдельной «комнате» и прочее.

IRC-сеть состоит из серверов, соединённых друг с другом. В мире существуют десятки IRC-сетей. Наиболее старые и известные – EFNet (существует с 1990 г.), DALnet (1994), IRCNet (1996) и RusNet (1997), в каждой из этих сетей десятки серверов и десятки каналов (комнат). Более полный список можно найти на *www.ircnet.ru*. Основные реализации (клиенты): irssi, KVIrc, mIRC, X-Chat, WeeChat, Miranda IM, x3irc, Pidgin. Самым известным из перечисленных клиентов является mIRC. После его установки и настройки подключения к интернету можно выбрать сеть и канал сети среди множества их в списке этого клиента. Список может редактироваться и обновляться.

Наиболее популярными программами – IRC-серверами являются: IRCPlus, Conference Room, WIRCsrv, IRCD.

Существенных преимуществ IRC перед веб-чатами, кроме хранения сообщений off-line, в настоящее время не существует. Недостаток IRC для простых пользователей – необходимость устанавливать специальную программу, настраивать её и разбираться с особенностями её использования. Для таких пользователей существуют веб-шлюзы для работы с IRC-серверами.

[415] От англ. swipe – проведение пальцем по экрану в горизонтальном направлении влево или вправо.
[416] См. RFC 1459, RFC 2810, RFC 2811, RFC 2812, RFC 2813.

Ранее **веб-чаты** существовали на многих известных и широко посещаемых сайтах, например *www.chat.ru, chat.mail.ru* и прочих. Если первый сейчас ещё работает, то второй адрес уже не существует. Это были места общения в реальном времени, подобно группам по интересам в «ВКонтакте» и других социальных сетях.

Сегодня веб-чаты переживают второе рождение, перестав быть многопользовательскими, они переродились в средство общения фирм с клиентами (см. рис. 6.25). За счёт поддержки плавающих фреймов браузером в правом нижнем углу любого сайта может появиться всплывающее окно, где с вами будет общаться реальный человек. Например, вам могут помочь с выбором курсов английского языка, так и со всем, чем угодно, от автомобильных шин до туристических продуктов. Использование этой технологии поставлено на коммерческие рельсы и стоит недорого, подробнее см. https://www.zopim.com/?lang=ru, http://www.jivosite.ru/ и др.

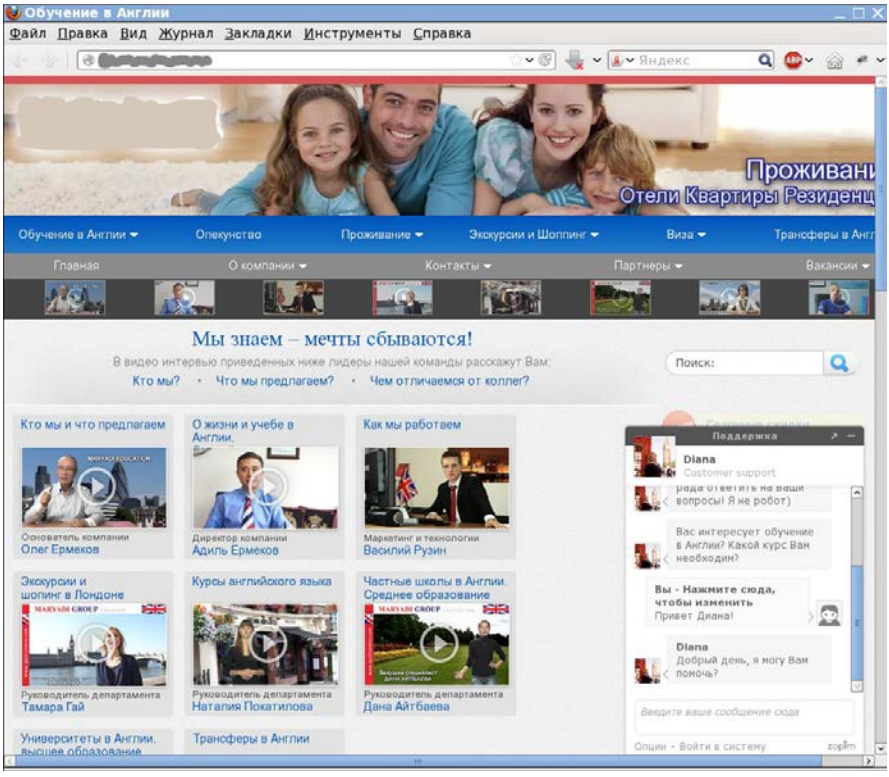


Рисунок 6.25. В правом нижнем углу сайта с вами в окошке общается реальный человек (обычно продавец-консультант)

Сетевые новости (Netnews) или группы новостей (Newsgroups) – это глобальная система интернет-конференций, которая позволяет организовать текстовые дискуссии в рамках тематических групп. Информация, которую можно прочитать в группе новостей, – это необязательно новости в привычном смысле слова. Это может быть любое сообщение, поданное для обсуждения в данную группу новостей, например предложение работы или её поиск, просьбы о консультациях по каким-либо вопросам и многое другое.

Интернет-конференции используют для передачи сообщений протокол NNTP. По строению этот протокол во многом сходен с протоколом приёма и передачи электронной почты SMTP. Протокол NNTP, как и SMTP, является текстовым, то есть все команды и ответы на них являются обычными текстовыми строками. Важной особенностью протокола NNTP является его эффективность в случае сложных графов связей между серверами новостей. Чтобы одно и то же сообщение не передавалось многократно, отправляющий сервер сначала сообщает идентификатор нового сообщения, а само сообщение отправляет только после подтверждения принимающей стороны о том, что этого сообщения там ещё нет.

Сообщение, посылаемое в Newsgroups, становится доступным для всех участников группы новостей на определённый срок. Существуют специальные клиентские программы для работы с группами новостей, например Newsreader, по аналогии с чтением RSS-лент.

Есть огромное количество новостных серверов, которые образуют целую инфраструктуру, обеспечивающую механизм передачи новостей по всему миру. Каждый узел сети, получивший сообщение, передаёт его всем узлам, с которыми он обменивается новостями.

Веб-форум – специальное программное обеспечение для организации общения посетителей веб-сайта. Сообщения в форум передаются по электронной почте или создаются прямо на сайте. Читать сообщения форума может любой желающий; чтобы отправить сообщение в форум, обычно нужно в нём зарегистрироваться, поэтому у каждого сообщения присутствует информация о его авторе и дате/времени отправления. В некоторых специализированных форумах регистрируют не всех желающих, а только тех, кто смог пройти профессиональный тест из 20–40 вопросов по теме форума.

Форум предлагает набор разделов для обсуждения. Работа форума заключается в создании пользователями тем в разделах и последующем обсуждении внутри этих тем. Отклонение от начальной темы обсуждения часто запрещено правилами поведения в форуме. За соблюдением правил следят модераторы и администраторы – участники, наделённые возможностью редактировать, перемещать и удалять чужие сообщения в определённом разделе или теме, а также контролировать к ним доступ отдельных участников. За несоблюдение правил следует предупреждение или «бан», отключение возможности создавать сообщения на какое-то время.

По методу формирования набора тем форумы бывают с динамическим списком тем и с постоянным списком тем. В форумах с динамическим списком тем простые участники могут создавать новую тему в рамках тематики форума.

Форум отличается от чата разделением обсуждаемых тем и возможностью общения не в реальном времени. Это располагает к более серьезным обсуждениям, поскольку предоставляет отвечающим больше времени на обдумывание ответа. Форумы часто используются для разного рода консультаций, в работе служб технической поддержки.

Например, среди системных администраторов и технических специалистов уже два десятилетия популярен форум http://sysadmins.ru/. Кстати, внешне с момента запуска он совсем не изменился. Компьютерный форум: http://forum.ru-board.com/. Форум – доска объявлений: http://www.komok.com/. Популярные форумы различных вузов (официальные и не официальные): http://2msu.ru/, http://forum.fizteh.ru/,

http://www.bmstu.ru/ps/forum/, http://forum.hse.ru/newforum/, https://forumbgz.ru/, http://mephi.ru/communication/forum/talk/forum9/ и другие.

В настоящее время веб-форумы почти полностью вытеснили новостные группы на базе NNTP и являются одним из наиболее популярных способов обсуждения вопросов в интернете. Многие форумы не отличимы от социальных сетей, поскольку ведутся на их платформе, а некоторые, хотя и сделаны на другом движке и располагаются на другом сайте, осуществляют аутентификацию пользователей через их учётную запись в социальной сети. На данный момент форумы сосуществуют наравне с блогами. Эти две формы общения в интернете практически не уступают друг другу по популярности.

Блоги – личные электронные дневники на веб-страницах.

В интернете имеются сайты, главное назначение которых – размещение блогов всех желающих, например *www.livejournal.com*. Пользователям веб-почты также часто предоставляется возможность вести свои дневники на страницах почтового сайта, например очень большое количество блогов можно найти на «@mail.ru». Особенность этого вида общения – любой желающий может читать блоги и писать свои комментарии к записям автора блога, каждая такая запись имеет дату и время создания. В разделе «Звёздные блоги на Mail.Ru» присутствуют дневники сотен российских знаменитостей. Иногда подобные блоги преследуют рекламные цели – сообщения о новых дисках, личных сайтах звёзд и т. п.

Сайты многих известных и крупных корпораций часто организуют на своих интернет-серверах страницы с блогами своих сотрудников. Предполагается, что они будут там рассказывать всем любопытствующим в основном о своей работе. Как бы и волки сыты, и овцы целы. Зачастую такие блоги превращаются в полноценные публикации, становясь кладезями информации по своей тематике.

Упрощение процесса публикации видеозаписей на таких страницах привело к появлению нового явления – «видео блогов» или «**влогов**». Люди пытаются делать интересные репортажи, рассказывают как готовить вкусную еду, чинить машины, делать ремонты, проходить игры и т.д. Если контент интересный и актуальный, то растёт число подписчиков. Если их становится много, то им показывается реклама, а блогер или влогер может получать с этого деньги. Существуют даже бесплатные сервисы оценивающие по URL страницы или блога сколько примерно может зарабатывать в месяц её владелец на рекламе.

6.4.4.4. Вебинары

On-line-семинар (веб-конференция, вебинар, англ. webinar) – разновидность веб-конференции, проведение on-line-встреч или презентаций через интернет в режиме реального времени. Во время веб-конференции каждый из участников находится у своего компьютера, а связь между ними поддерживается через интернет посредством загружаемого приложения, установленного на компьютере каждого участника, или через браузер. В последнем случае, чтобы присоединиться к конференции, нужно просто ввести URL (адрес сайта) в адресной строке браузера.

Вебинары могут быть совместными и включать в себя сеансы голосований и опросов, что обеспечивает полное взаимодействие между аудиторией и ведущим. В некоторых случаях ведущий может говорить через телефон, комментируя информацию,

отображаемую на экране, а слушатели могут ему отвечать тоже по телефону (создаётся параллельная телефонная конференция). На рынке также присутствуют и более дешёвые технологии, например VoIP, когда обеспечивается полноценная аудиосвязь через сеть. Вебинары (в зависимости от провайдера) могут обладать функцией анонимности или «невидимости» пользователей, благодаря чему участники одной и той же конференции могут не знать о присутствии друг друга.

В первые годы после появления интернета термином «веб-конференция» часто называли ветку форума или доски объявлений. Позже термин получил значение общения именно в режиме реального времени. В настоящее время вебинары часто используются в рамках систем дистанционного обучения. Собственный аналогичный функционал проведения вебинаров появился у многих мессенджеров, например многие вспомнят Zoom.

6.4.4.5. IP-телефония

IP-телефония (интернет-телефония) – технология, которая использует интернет для передачи речевых сигналов. Технология работы IP-телефона довольно проста. Как и в обычном телефоне, голосовые сигналы (слова, которые мы произносим) преобразуются в последовательности сжатия и разрежения воздуха, которые микрофоном преобразуются в электрический сигнал. В дальнейшем цифровой сигнал дискретизируется и квантуется (подробнее см. стр. 113), в результате чего получается некоторая цифровая последовательность. В дальнейшем она может быть преобразована (сжата до меньших размеров с помощью голосовых кодеков). Сжатые последовательности байтов упаковываются в IP-пакеты и по локальной сети или через интернет передаются другой стороне. Когда пакеты данных достигают адресата, они декодируются в исходные голосовые сигналы.

Поскольку в общем случае пакеты данных могут добираться до адресата по различным маршрутам, после получения они перегруппировываются, что вносит некоторую задержку. Суммарная задержка «доставки голоса» с учётом времени на передачу и получение пакетов, кодирование и декодирование не должна превышать 250 мс, в противном случае общение будет некомфортным.

Преимущества IP-телефонии перед обычной телефонной связью:

- не нужно платить за междугородние звонки (только за интернет-трафик);
- возможность выбора кодеков (качества сжатия звука), или лучшее качество, или меньший передаваемый объём.

Существуют 3 типа IP-телефонии:

1) с компьютера на компьютер (при этом к его звуковой плате нужно подключить микрофон);
2) с компьютера на телефон;
3) с телефона на телефон.

Несмотря на широкое распространение интернета и первоначальную простоту алгоритмов, наступлению светлого будущего мешают операторы связи, напрямую не заинтересованные в уменьшении своих прибылей от стационарной и сотовой телефонной связи.

Для передачи голосового трафика с компьютера на компьютер широкое распространение получили разрекламированные программы Skype, Google Talk и др., а также

протокол H.323. В 1996 году начались разработки нового протокола SIP – который сейчас очень популярен и фактически заменил H.323.

SIP [417] – протокол передачи данных, который описывает способ установления и завершения пользовательского интернет-сеанса, включающего обмен мультимедийным содержимым (видео- и аудиоконференция, мгновенные сообщения, онлайн-игры).

Сегодня на рынке, кроме различных платных и бесплатных программ SIP-клиентов для разных ОС, можно также купить и аппаратные решения: SIP-телефоны (см. рис. 6.26) или SIP-шлюзы, позволяющие подключать обычные телефоны к интернету.

В теории два SIP-телефона или две SIP-программы могут работать друг с другом напрямую, но на практике дальше локальной сети такие коммуникации проблематичны, потому как в общем случае SIP-телефоны или шлюзы не знают, где находятся их «собратья», особенно если IP-адрес, выделяемый вам вашим провайдером, постоянно меняется. Да и пользователи не утруждают себя запоминанием IP-адресов и сетевых настроек. Чтобы сделать жизнь пользователей проще, для связи SIP-устройств придумали использовать SIP-сервер. На сервере каждому устройству выделяется его номер – SIP ID.

В интернете доступно множество работающих и бесплатных SIP-серверов для связи SIP-устройств между собой. В России наиболее популярна сеть sipnet.ru. Помимо передачи голосового трафика, сервер может поддерживать различные дополнительные услуги, такие как почтовые (голосовые) ящики, конференции, отправку SMS и звонки из/в реальную телефонную сеть общего пользования.

Рисунок 6.26. SIP-телефон от D-Link (средняя цена 1500–3000 руб.)

Для привлечения пользователей обычно звонки с SIP на SIP бесплатны, а вот остальные услуги, таких как например, аренда обычного телефонного номера в коде того или иного города, «переадресация», «приземление» или «отправка» звонков в ре-

[417] От англ. Session Initiation Protocol – протокол установления сеанса.

альную сеть, стоят денег. Например, за сумму порядка 30–50 руб. в день можно получить «привязку» SIP ID к телефонному номеру в коде (+7495) – Москва или зарубежным телефонам. При этом все местные входящие и исходящие звонки будут бесплатны.

Это просто бесценное решение для поездок в командировки или при организации call-центров. Клиенты звонят по телефону с местным телефонным кодом, а звонки «падают» в call-центр, размещённый в регионе или даже другой стране, где затраты на его обслуживание ниже. Или во время отпуска вам звонят на ваш арендованный местный номер, а звонки падают на ваш телефон, планшет или компьютер в номере гостиницы по интернету. Так и в обратном направлении – вы просто набираете нужный вам телефонный номер в SIP-приложении и делаете телефонный звонок, не оплачивая при этом никаких междугородных счетов, так как обычно Wi-Fi-интернет в гостиницах бесплатный.

Чтобы начать использовать удобства SIP-решения, не обязательно покупать SIP-телефон или SIP-шлюз, существует множество программ SIP-клиентов как для ПК, так и для тех же смартфонов с ОС iOS и Android, многие современные телефоны Nokia имеют даже встроенный (штатный) SIP-клиент. Некоторые SIP-клиенты умеют соединяться друг с другом напрямую (без SIP-аккаунта на SIP-сервере) по IP-адресу. Для ОС Linux можно рекомендовать программу Twinkle из репозитария EPEL.

Если вы не хотите, чтобы ваш голосовой трафик шёл через сторонний коммерческий SIP-сервер, вы можете поднять собственный сервер для SIP-телефонии, например Asterisk, и подключаться к нему, используя технологию VPN. Впрочем, многие коммерческие фирмы так и поступают, снижая свои расходы на связь.

Asterisk – свободное решение компьютерной телефонии (в том числе VoIP) с открытым исходным кодом от компании Digium, первоначально разработанное Марком Спенсером. Приложение работает на операционных системах Linux, FreeBSD, OpenBSD и Solaris и др. Имя проекта произошло от названия символа «*» (англ. asterisk – «звёздочка»).

Asterisk в комплексе с необходимым оборудованием обладает всеми возможностями классической АТС, поддерживает множество VoIP-протоколов и предоставляет богатые функции управления звонками, среди них:

- голосовая почта;
- конференции;
- интерактивное голосовое меню (IVR);
- центр обработки вызовов (постановка звонков в очередь и распределение их по агентам с использованием различных алгоритмов);
- запись (Call Detail Record).

Список поддерживаемых Asterisk'ом протоколов довольно широк:

- SIP
- H.323
- IAX2
- MGCP
- SIMPLE
- Skinny/SCCP
- XMPP (Google Talk)
- Unistim
- Skype [418].

Для создания дополнительной функциональности можно воспользоваться собственным языком Asterisk для написания плана нумерации, написав модуль на языке Си, либо воспользовавшись AGI – гибким и универсальным интерфейсом для интегра-

[418] Через коммерческий канал.

ции с внешними системами обработки данных. Модули, выполняющиеся через AGI, могут быть написаны на любом языке программирования.

Существует и более простая программа SIP-сервера – kamailio.

Если вы не сильны в программировании и настройках, но понимаете, что IP-телефония нужна, возможно, вам подойдут различные уже существующие на рынке готовые решения. Например, от отечественной компании «АГАТ-РТ» (http://www.agatrt.ru), созданной и занимающейся вопросами передачи голосового трафика с 1994 года.

6.4.5. Обмен файлами

Нередко возникает ситуация, когда надо передать или получить какой-то файл. Хорошо, если его можно записать на дискету, флэшку, телефон (в его внутреннюю flash-память), на оптический или жёсткий диск и захватить носитель с собой, но часто бывает ситуация, что мы не можем физически взять носитель и встретиться с нужным нам человеком из-за географической удалённости или разных временных графиков, либо ещё по каким-то причинам. В этом случае мы обычно посылаем файл по почте получателю или самим себе, а потом забираем файл из почты в другом месте через веб-интерфейс или даём наш пароль от ящика другу. Если файл большой, скажем более 10–20 МБ, то его приходится архивировать томами или делить на части.

Для решения вышепоставленной проблемы файлового обмена можно использовать специально созданные для этой цели файлообменные веб-серверы: http://webfile.ru, ~~http://zalil.ru~~, ~~https://rapidshare.com~~, http://dfiles.ru (http://depositfiles.com), http://megashares.com/, http://rusfolder.com (http://ifolder.ru) и др.

Указанный сервис популярен, потому как файлы обычно загружаются бесплатно и без регистрации. При необходимости загружаемому файлу можно задать пароль. По нему файл можно досрочно удалить, посмотреть статистику скачиваний, либо сделать загрузку также по паролю. Ограничения на максимальный размер файла зависят от используемого сайта и варьируются в пределах 350–1000 МБ. После загрузки файла вы сразу получаете короткую http-ссылку, которую можете отправить по почте, SMS-кой или продиктовать голосом по телефону. Вашему оппоненту остаётся только ввести ссылку в браузере и скачать файл. Через некоторое заранее оговорённое время (обычно около 30 дней) файл автоматически удаляется с сервера.

Мотивация к созданию подобных сервисов – зарабатывание денег на рекламе, просматриваемой посетителями во время загрузки или скачивания файлов.

6.4.5.1. Торренты

У описанных в разделе выше файлообменных сервисов есть минус – они не могут одновременно обеспечить на должной скорости скачивание файлов несколькими участниками одновременно. Например, если вы выложите в Сеть какой-то популярный видеоролик минут на 30 в хорошем качестве (естественно, с соблюдением авторских прав, без нарушений законодательства) объёмом в 300–500 МБ, а скачивать его начнут сразу 100–1000 человек, получив от вас рассылку в том же Твиттере, то закончат они это делать не скоро. Для подобных целей лучше использовать программы, осуществляющие «равноранговый» информационный обмен. Чаще их называют собирательным термином «торренты» из-за используемых при работе файлов с метаданными,

имеющими расширение .torrent. Файлы «.torrent» являются словарём в bencode-формате, используются в p2p сети BitTorrent и содержат информацию о файлах, трекерах и др.

В случае использования торрентов разные части одного и того же файла прозрачно для пользователя могут скачиваться от других участников файлового обмена, например скачавших этот же файл с сервера ранее. Загрузка сети при использовании «торрентов» более равномерная. В общем случае сервер для первичной раздачи файлов не существует, по сути, им является первый раздающий.

Для распространения .torrent-файлов обычно используются форумы, структурированные по разделам, зачастую к ним можно оставлять свои комментарии. Указанные форумы называют торрент-трекерами или просто трекерами.

Через торренты можно обмениваться видеофайлами, музыкой, программами и прочим. Наиболее популярные сайты-трекеры в России и странах СНГ: http://rutracker.org, http://www.torrentino.com/, http://rutor.org, http://xrutor.org, http://multi-tor.org, http://nnm-club.ru и др.

Нередко трекеры (не только у нас, но и по всему миру) становятся причиной подачи судебных исков к их владельцам, потому как часто используются с нарушением законов об авторском праве. Зачастую это происходит из-за нерадивых модераторов, попустительствующих нарушениям. Так, по результатам судебных решений сайты могут быть закрыты, либо доступ к ним может быть ограничен, путём их внесения в единый реестр запрещённых сайтов, см. стр. 170. По мнению авторов это полезная мера, поскольку интернет временами напоминает большую и плохо пахнущую груду мусора. Нередко источниками этого мусора становятся зарубежные правительственные организации, которые будоражат умы наших сограждан, покушаются на человеческие ценности, распространяют дезинформацию, пытаются посеять панику.

Существуют торрент-клиенты для ОС Windows, Linux, iOS, Android и др.

Небольшой словарь: [419]

Трекер (от англ. tracker) – специализированный сервер, работающий по протоколу HTTP. Трекер нужен для того, чтобы клиенты могли найти друг друга. Фактически на трекере хранятся IP-адреса и входящие порты клиентов и хэш-суммы, уникальным образом идентифицирующие объекты, участвующие в закачках. По стандарту, имена файлов на трекере не хранятся, и узнать их по хэш-суммам нельзя. В практических реализациях, однако, трекер часто, помимо своей основной функции, выполняет и функцию небольшого веб-сервера. Такой сервер хранит файлы метаданных и описания распространяемых файлов, предоставляет статистику закачек по разным файлам, показывает текущее количество подключенных пиров и прочее.

Торрент (.torrent, торрент-файл) – это идентификатор. То есть файл, который содержит в себе информацию о запрошенных файлах, а именно:

- о размере и количестве фрагментов (файл разбивается на несколько тысяч частей) и контрольной сумме скачиваемого файла (файлов);
- о трекере, на котором можно получить информацию о сидах и личерах, распространяющих этот файл.

[419] Другие термины см. на http://wiki.rutracker.org/Bittorrent/Термины.

После того как пользователи скачивают этот файл себе, они уже знают, где брать файлы, из которых был создан торрент. По сути, торрент-файл – это ярлык, который однозначно идентифицирует содержимое передаваемой информации и адрес, где можно получить эту информацию. Обычно весит примерно 100 КБ, в дальнейшем открывается специальным торрент-клиентом. Торрент-файл плох тем, что он файл, а значит, его надо где-то хранить. Заблокируйте хранилище с .torrent-файлами (веб-сайт торрент трекера) и вы перекроете доступ новым пользователям к скачиванию файлов, потому как их программы не будут знать откуда что скачивать. Лишены этого недостатка магнитные ссылки (magnet-ссылки) используемые совместно с DHT.

Magnet-ссылка – открытый, находящийся в стадии рабочего черновика стандарт, определяющий URI-схему для ссылок, указывающих на ресурсы доступные к загрузке через пиринговые сети. Такие ссылки в основном идентифицируют файлы не по их расположению на том или ином компьютере или имени, а по хеш-коду высчитываемому на основе содержания этих файлов. Такая схема подходит только для файлов, которые нужны и есть у большого числа пользователей (например новогодний фильм перед новым годом), поскольку позволяет скачивать файл частями из разных источников не привязываюсь к ним. Для редковостребованных файлов, интересных только узкому кругу пользователей, такая схема ссылок оказывается неэффективной.

По формату magnet-ссылка – это текстовая строка длинной в десятки-сотни символов. Она может содержать один или несколько параметров, разделённых между собой знаком «&» (как в URL). Порядок следования параметров не документирован, но для отдельных программ он важен. Разные пиринговые сети (её клиенты) содержат в ссылках разные параметры, т.е. по сути получаются единоформатные, но не совместимые между различными сетями magnet-ссылки. К счастью, самые большие пиринговые сети BitTorrent (Azureus, µTorrent) используют формат BTIH (BitTorrent Info Hash), который является самым популярным и часто встречающимся форматом magnet-ссылок.

DHT (от англ. distributed hash table — «распределённая хеш-таблица») – это инфраструктура, которая может быть использована для построения многих сложных служб, таких как распределённые файловые системы, пиринговое распространение файлов, кооперативный web-кэш, многоадресное вещание (multicast), система мгновенных сообщений и др. Сегодня в основном DHT используют распределённые сети: I2P, BitTorrent, eDonkey network и др. Распределённое хранение не исключает возможности создания поисковых машин по сети DHT.

6.4.5.2. Хранение файлов в облаке

Если вы планируете хранить файлы в интернете постоянно, то рассмотренные в начале раздела файлообменные сервисы не очень удобны, так как не позволяют структурировать файлы, удаляют их через некоторое время. Для более постоянного хранения лучше воспользоваться специализированными сервисами, например ***Яндекс.Диск*** – облачный сервис, принадлежащий компании Яндекс, позволяющий пользователям хранить свои данные на серверах в облаке и передавать их другим пользователям в интернете. Работа построена на синхронизации данных между различными устройствами. Сервис полностью бесплатен. API для разработчиков доступен по адресу

http://api.yandex.ru/disk/. Синхронизация с ОС Linux[420] и другими ОС доступна за счёт поддержки протокола WebDAV по первому классу соответствия согласно RFC 4918.

По умолчанию предоставляются 3 ГБ пространства, которые впоследствии можно расширить до 10 ГБ и более. После установки программы в «компьютере» появляется «Яндекс.Диск», в который можно перетаскивать файлы мышкой (drag-and-drop) (см. рис. 6.27).

Через несколько секунд файл отображается в хранилище, как если это был файл на жёстком диске или внешнем USB-flash-накопителе. Большие файлы, в зависимости от скорости вашего подключения, могут копироваться десятки минут.

Сервис Dropbox аналогичен сервису «Яндекс.Диск», за тем лишь отличием, что это не российская компания. Первые несколько гигабайт, подобно сыру в мышеловке, предоставляются бесплатно, а вот большие объёмы: 100, 200 и 500 ГБ – предлагается арендовать за сумму порядка 300 руб. в месяц. Реализована поддержка разных версий ОС Linux.

Практически для всех подобных файловых сервисов существуют клиенты для ОС, используемых на планшетах и телефонах. Перенос файлов из или в такие устройства не сложнее переноса мышкой.

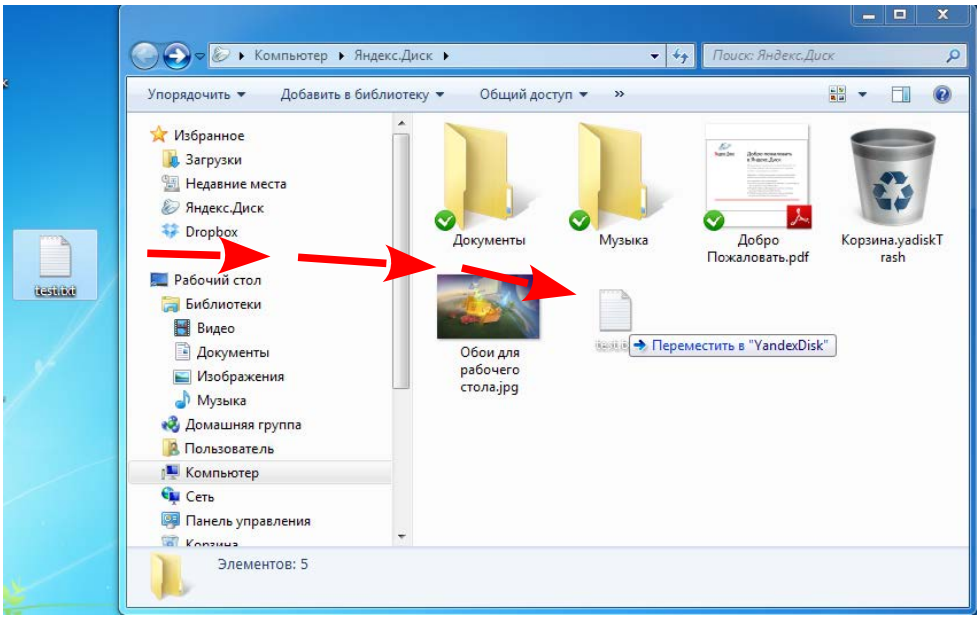


Рисунок 6.27. Работа с «Яндекс.Диск» очень проста. Перетаскивание файла мышью с рабочего стола в сетевое хранилище

Если вы не хотите устанавливать себе (или в другом месте) «клиентскую часть», то можете работать с хранилищем посредством браузера (см. рис. 6.28).

[420] Для конечных пользователей доступна небольшая программа на Perl по адресу https://github.com/kappa/yadisk-sync.

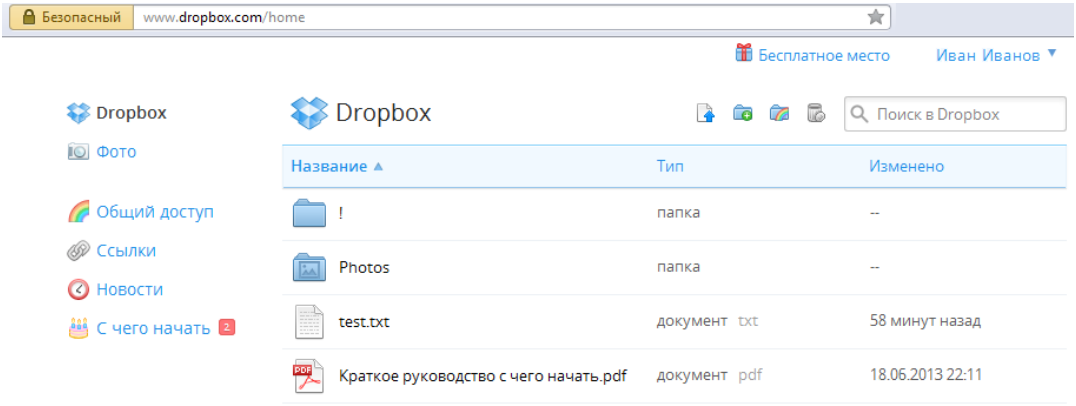


Рисунок 6.28. Работа с Dropbox через веб-интерфейс

Из недостатков можно отметить, что оба сервиса для регистрации требуют привязку к электронной почте. Также Dropbox протоколирует, с какого компьютера (или иного устройства) к нему подключались и что на него отправлялось (получалось), и, при желании, отправляет на почту уведомление, что кто-то подключился к аккаунту.

Ограничения на размеры одного и всех файлов различны, наиболее удачное сравнение данных сервисов с другими приведено в [6] (см. рис. 6.29).

Общее сравнение (1 из 4) BeamTeam.ru 20.02.2013	Я.Диск	Dropbox	G.Drive	SkyDrive	SugarSync	Box
Бесплатный объём стартовый, ГБ	**3-6**	2	5	**7**	5	5
Возможно бесплатное расширение до, ГБ	**20**	**20**	Нет	Нет	Любое	Нет
Максимальный объём для 1 человека	20	500 ГБ	16 ТБ	107 ГБ	Любой	1 ТБ
Максимальный размер файла, ГБ	3 ГБ	Любой	10 ГБ	2 ГБ	Любой	5 ГБ
Загрузка файлов через настольное приложение	10 ГБ	Любой	10 ГБ	2 ГБ	н/д	н/д
Загрузка файлов через веб-сайт	2 ГБ	300 МБ	10 ГБ	300 МБ	300 МБ	250 МБ
Потребление памяти, МБ (WinXP)	16-36	55-71	52-78	9-32	46-70	16-47
Время загрузки файла, % от Яндекса	100	132	100	104	134	н/д
Доступность за 30 дней, %	н/д	99,97	100	99,97	н/д	100
Использование существующего аккаунта почты без необходимости регистрации	Да	Нет	Да	Да	Нет	Нет
Тарифные планы, USD/год, округлённо	**Я.Диск**	**Dropbox**	**G.Drive**	**SkyDrive**	**SugarSync**	**Box**
25+ ГБ	Нет	Нет	30	**10 (27 ГБ)**	50 (30 ГБ)	100
50+ ГБ	Нет	Нет	Нет	**25 (57 ГБ)**	100 (60 ГБ)	Нет
100 ГБ	Нет	100	60	**50 (107 ГБ)**	150	Нет
200+ ГБ	Нет	200	**120**	Нет	250 (250 ГБ)	Нет
500 ГБ	Нет	500	240 (400 ГБ)	Нет	400	**180 (1 ТБ)**
Поддержка платформ	**Я.Диск**	**Dropbox**	**G.Drive**	**SkyDrive**	**SugarSync**	**Box**
Windows XP	Да	Да	Да	Нет	Да	Да
Windows Vista, 7	Да	Да	Да	Да	Да	Да
Windows 8	Нет	Да	Да	Да	Да	Да
Mac OS X	Да	Да	Да	Да	Да	Да
iOS	Да	Да	Да	Да	Да	Да
Android	Да	Да	Да	Да	Да	Да
Windows Phone	Нет	Нет	Нет	Да	Да	Да
Blackberry	Нет	Да	Нет	Нет	Да	Да
BeamTeam.ru 20.02.2013	Я.Диск	Dropbox	G.Drive	SkyDrive	SugarSync	Box

Рисунок 6.29. Сравнение различных файловых сервисов [6]

6.5. Интернет-радио и интернет-телевидение

Интернет-вещание – радиоканалы и телеканалы, которые можно слушать и смотреть на веб-страницах сайтов, посвящённых этому или через мультимедийные приложения – VLC Media Player, WinAmp, RealPlayer и прочие.

Например, проекты http://www.moskva.fm/ и http://online-television.net предоставляет всем посетителям возможность прослушивания интернет-радиостанций и просмотра интернет-телевидения со своих сайтов либо перенаправляют на сайт первоисточника с видео. Для просмотра телеканалов желательна скорость подключения к интернету не менее 1 Мбит/с.

На сайте *http://sradio.ru* можно найти список и слушать эфир около четырёх тысяч радиостанций из более чем ста разных стран мира. Значительная часть присутствующих радиостанций имеет каналы вещания через интернет. Сайт http://*E-Radio.Ru* посвящён интернет-радиостанциям России, стран СНГ, русскоязычным и ведущим музыкальным интернет-радиостанциям мира, вещающим в потоковых форматах для Windows Media Player и Flash Player. Имеется множество и других сайтов подобной тематики.

Интернет-телевидение предъявляет более высокие требования к скорости соединения пользователя с сетью. Имеется достаточно много сайтов, назначение которых – подключение пользователя к телеканалам, вещающим в интернете. Например, TV-portal.ru (*http://www.tv-portal.ru/*) имеет ссылки почти на 400 каналов, при поиске канала позволяет задать язык, страну и тематику вещания, а также интернет-TV-плеер, через который возможен просмотр – VLC Media Player, Real Player, RuTube и др. Фрагмент сайта с некоторыми кнопками выбора телеканалов показан на рис. 6.30.

Рисунок 6.30. «Кнопки» для выбора каналов на сайте http://tv-portal.ru/ (фрагмент)

Помимо «веб-ТВ-вещания», которое, по сути, работает «поверх протокола HTTP», существуют и более специализированные сайты и фирмы (например, http://www.n3.ru), использующие специализированные широковещательные протоколы типа IGMP для распространения видео по сети. В этом случае поддержка раздачи контента должна обеспечиваться на уровне вашего провайдера. Некоторые провайдеры не заинтересова-

ны в повышении трафика сторонними фирмами и предлагают свои ТВ-каналы либо бесплатно, либо за символическую плату. Например, многие тарифы при DSL-подключении к интернету имеют возможность получения «ТВ в придачу».

С увеличением скорости интернет-соединений набирают популярность различные ТВ-приставки: AppleTV, Google TV, менее известные вроде iconBIT Toucan W и другие с ОС Android. Некоторые современные телевизоры имеют собственное программное обеспечение для просмотра видео через интернет не хуже, чем в приставках выше.

6.6. Электронная коммерция

Электронная коммерция – форма взаимодействия продавцов и покупателей, при которой выбор и заказ товаров осуществляется через компьютерные сети, а расчёты между покупателем и поставщиком осуществляются с использованием электронных документов или средств платежа.

При классификации электронного бизнеса выделяют несколько категорий, которые со временем обрели сокращённые обозначения, наиболее популярные из которых приведены в табл. 6.5. (Английский предлог to, отвечающий на вопросы «кому», «чему», произносится так же, как 2 (two).)

Таблица 6.5. Некоторые формы электронной коммерции

Обозначение	Полное название	Пример
B2C	Коммерсант клиенту (Business-to-Consumer)	Заказ книг, цветов, еды и прочего в режиме on-line
B2B	Коммерсант коммерсанту (Business-to-Business)	Сотовая компания предоставляет корпоративную связь коммерческим фирмам
B2A	Коммерсант государству (Business-to-Administration)	Фирма, выполняющая государственный контракт по прокладке дороги или постройке моста.
G2C	Государство частному клиенту (Government-to- Client)	Распространение бланков квитанций через интернет, изображение с веб-камер на социально значимых объектах
C2A	Частное лицо государству (Consumer-to-Administration)	Электронная приёмная (форма на веб-сайте) или «горячая линия» губернатора области
C2B	Частное лицо коммерсанту (Consumer-to-Business)	Системы обработки ценовых заявок, по которым потребители хотели бы приобрести товары и услуги (в своём роде обратный аукцион)
C2C	Клиент клиенту (Client-to-Client)	Продажа подержанных вещей
P2P	Равноранговые сети (Peer-to-Peer)	Совместный доступ к музыкальным и видеофайлам

Глобальная сеть Интернет сделала электронную коммерцию доступной для фирм любого масштаба. Если раньше организация электронного обмена данными требовала заметных вложений в коммуникационную инфраструктуру и была по плечу лишь крупным компаниям, то использование интернета позволяет сегодня вступить в ряды «электронных торговцев» и небольшим фирмам. Веб-магазин даёт любой компании

возможность обслуживать покупателей всего мира. Подобный on-line-бизнес формирует новый канал для сбыта – «виртуальный», требующий небольших материальных вложений. Если информация, услуги или продукция (например, программное обеспечение, книги и музыка в компьютерных форматах) могут быть поставлены через веб, то весь процесс продажи (включая оплату) может происходить в on-line режиме.

На сегодняшний день доминирующим платёжным средством при on-line-покупках являются кредитные банковские карты, пришедшие к нам с Запада со всеми своими недостатками. Однако на сцену выходят и новые платёжные инструменты: смарт-карты, цифровые деньги, микроплатежи и электронные чеки.

Электронная коммерция включает в себя не только on-line транзакции. В область, охватываемую этим понятием, входят и такие виды деятельности, как проведение маркетинговых исследований, определение возможностей и партнёров, поддержка связей с поставщиками и потребителями, организация документооборота и прочее. Таким образом, электронная коммерция является комплексным понятием и включает в себя электронный обмен данными как одну из составляющих.

Для простых потребителей наиболее важной категорией электронного бизнеса является B2C. К этой категории бизнеса относится весьма значительный круг предприятий электронной коммерции: интернет-магазины, платные сервисы для физических лиц, электронные казино, многочисленные компании, продающие консультационные и информационные услуги. В структуру B2C-компании обычно входят следующие составляющие:

- интерактивный веб-сайт, содержащий информацию о продукции, товарах и услугах, прайс-листы и каталоги. Как правило, сайт предоставляет клиенту возможность оформить on-line-заказ и отслеживать стадии его исполнения;
- площадка хостинга сайта – место, где сайт физически размещён. Это может быть собственный сервер компании, расположенный в офисе фирмы или на территории (хостинг-площадке) интернет-провайдера. Такой вариант характерен для крупных сетевых проектов. Для небольшого проекта возможно размещение сайта на сервере провайдера. Это значительно дешевле и позволяет сократить количество персонала, отвечающего за функционирование сайта;
- back-офис – технический персонал и администрация компании, помещение, где она размещается и откуда осуществляется административное и техническое управление проектом;
- служба доставки. В B2C-компаниях служба доставки играет весьма значительную роль. Чем быстрее товар будет доставлен клиенту и чем меньше неудобств он при этом испытает, тем больше вероятность того, что он совершит повторную покупку. Для различных компаний служба доставки может иметь самые разные формы. Иногда её может вообще не быть как сколько-нибудь значимого подразделения (например, если компания оказывает консультационные или информационные услуги). В других случаях можно ограничиться курьерской службой. Встречаются ситуации, когда компания торгует товарами, имеющими значительные размеры и вес. Тогда в службе доставки может быть собственный или арендованный склад, парк автомобилей и т. п.;
- подразделение по работе с поставщиками. Как и в любой не электронной компании, хороший поставщик (надёжный, с приемлемыми ценами) является неотъемле-

мой частью бизнеса. Обойтись без этого подразделения могут только компании, оказывающие услуги целиком силами их собственного персонала;

- система расчётов за товары и услуги. Заказав товар, клиент должен его оплатить. Хорошо, когда можно совместить оплату с доставкой товара. Но это не всегда возможно (например, когда товар отправляется клиенту по почте или вообще в электронном виде, если это информация или программное обеспечение). В таком случае клиент должен оплатить товар, не вступая в непосредственный контакт с персоналом компании. Конечно, можно оплатить покупку обычным банковским или почтовым переводом, но это чревато большими неудобствами, связанными прежде всего со значительным временем прохождения платежей. Для решения этих проблем в интернете существуют системы электронных платежей. Формирование системы расчётов подразумевает заключение договоров с этими системами и установку на сайт программного обеспечения, позволяющего начать работу с ними;
- маркетинговая служба. В современных условиях любой интернет-проект не может выжить без продуманной системы маркетинга. Маркетинг традиционно является одной из самых сложных составляющих сетевого бизнеса, где ошибки обходятся особенно дорого. Именно на этом направлении можно легко потерять или весьма нерационально израсходовать значительные ресурсы, чему есть немало печальных примеров.

Для промышленных предприятий наибольшее значение имеют категории электронной коммерции B2B и B2C. К категории B2B могут относиться взаимоотношения между подразделениями корпорации и между предприятиями-поставщиками и потребителями, а также межкорпоративные интернет-биржи.

В настоящее время, говоря о B2B-рынке, часто делают основной упор на межкорпоративные торговые площадки в интернете. Действительно, в мире значительную часть оборотов B2B-рынка подают именно они. В России тоже появились свои отраслевые электронные биржи: MetalsRussia.com (металлы), ChemForum.ru (фармацевтика), eMatrix (компьютерная техника), Zerno OnLine (зерно, сахар, подсолнечник), Faktura (универсальная интернет-биржа), eMetex (трубы и комплектующие), Depo.ru (компьютерная техника).

Структура B2B-компании во многом схожа со структурой компании B2C.

6.6.1. Виды денег

Для осуществления платежей в электронной коммерции применяются различные платёжные системы использующие различные типы денег:

реальные, электронные, виртуальные, и цифровые.

Реальные системы расчётов работают с реальными (фиатными[421]) деньгами (расчёты наличными, банковскими и почтовыми переводами, платёжные системы, работающие с банковскими счетами клиентов).

Электронные системы расчётов, – использующие электронную валюту (электронные деньги), эмитируемую платёжными системами. К ним можно отнести платёжные

[421] Фиатные деньги (от лат. fiat – декрет, указание, «да будет так») – деньги, законные платёжные средства, номинальная стоимость которых устанавливается, обеспечивается и гарантируется государством посредством его авторитета и власти.

системы, обеспечивающие возможность расчётов в том числе и в интернете с помощью банковских пластиковых карт. Они делятся на два основных подтипа:

1. Платёжные системы, использующие для осуществления платежей только реквизиты кредитной/дебетовой карты плательщика, которые передаются через интернет (обычно с применением протокола SSL). Такие системы просты в использовании плательщиком, но имеют ряд недостатков. Процессинговый центр получает информацию о всех ваших платежах, а также если он находится на территории чужой страны, то вы в один миг можете оказаться без денег. Информация напечатанная на банковской карте (номер, защитный код, имя владельца, срок годности) может быть легко скопирована и использована любым лицом без ведома владельца карты для оплаты услуг в том же интернете.
2. Платёжные системы, использующие технологию SET или другие аналогичные технологии, специально созданные для защиты платежей в интернете. Эти технологии предусматривают выдачу каждому пользователю цифрового сертификата. Аутентификация пользователя с помощью цифровой подписи позволяет предотвратить мошеннические платежи и отказы от платежей. Кроме того, информация о платёжной карте не поступает продавцу, а находится только в банке-экваере (банке, который обслуживает платежи в интернете по кредитным картам).

Электронные деньги – это электронный эквивалент реальных, фиатных денег. Они эмитируются платёжной системой и зачисляются на электронные счета клиентов в обмен на деньги, которые клиент вводит в систему банковским переводом или путём внесения наличных.

По месту хранения электронных денег платёжные системы делятся на три основных типа:

1) системы, хранящие цифровую наличность в электронном «кошельке», расположенном на стороне клиента. В этом случае программное обеспечение платёжной системы обеспечивает клиенту совершение операций с его деньгами и невозможность несанкционированного доступа к ним (к электронному счёту) других лиц. Так как сами цифровые деньги хранятся на стороне клиента (на его компьютере, телефоне, планшете и тому подобных устройствах) ответственность за их сохранность лежит на нём (в случае потери данных на компьютере клиента его электронный «кошелёк» не может быть восстановлен). Для подключения к такой системе клиент должен скачать с сайта системы программное обеспечение, установить его на своём компьютере и настроить в соответствии с правилами системы;

2) системы, в которых информация о состоянии электронных счетов клиентов хранится на серверах платёжной системы. Для подключения к такой системе клиенту достаточно зарегистрироваться на сайте платёжной системы. Иногда системы этого типа требуют от клиента скачать и установить программное обеспечение, позволяющее работать со счётом. Преимуществом таких систем для клиента является то, что потеря информации на его компьютере никак не отразится на состоянии его электронных счетов;

3) системы, в которых хранение цифровых денег и/или информации о счетах клиента осуществляется на стороне клиента, но на смарт-картах (картах с чипом) или USB-токенах, – небольших недорогих устройствах с микросхемой внутри. Для осуществления платежей пользователю необходимо иметь соответствую-

щее периферийное оборудование, которое позволяет считывать в компьютер (планшет или телефон) информацию со смарт-карт и токенов контактным или бесконтактным способом. В случае использования разъёма USB дополнительные устройства не требуются. Ведутся также работы по внедрению смарт-карт, использующих технологию SET.

Виртуальные расчёты (виртуальные деньги) – обязательства о выполнении услуг или предоставлении товаров на какую-то ранее внесённую в систему сумму. Например скидочно-накопительные или бонусные системы, подарочные карты и сертификаты, билеты для прохода через турникеты (например для проезда в общественном транспорте). К виртуальным деньгам также относятся деньги в различного рода online играх.

Замечание. В ряде случаев значения терминов виртуальные и цифровые деньги ошибочно смешиваются, либо меняются смыслом друг по отношению к другу.

Цифровые деньги – отличное по своей сути от трёх предыдущих типов платёжное средство. В ряде стран получили широкое распространение начиная с 2009 года (когда был изобретён Биткойн или Биткоин[422] – пиринговая система цифровых денег, использующая одноимённую цифровую валюту). В ряде стран находятся под полным или частичным запретом. Поскольку цифровая валюта основана на крипрографических алгоритмах, не является фиатной и не привязана к реальным денежным средствам, её часто называют криптовалютой. Bitcoin была первой цифровой валютой, получившей широкую известность, однако, довольно быстро на рынке появилось около сотни аналогичных альтернативных цифровых валют. Наиболее известные среди них Litecoin, Megacoin, WorldCoin, CryptoCoins, NetCoin, Ripple, Dash, BitShares, Dogecoin и другие. В сети стали появляться сайты, где отслеживаются котировки цифровых валют по отношению к реальным, например - http://coinmarketcap.com/.

Финансовый рынок и учёное сообщество не однозначно, но с любопытством отнеслись к цифровым деньгам из-за ряда специфичных, присущий только им, свойств:

- наличие отрицательной инфляции на всём периоде существования денег;
- отсутствие централизованного управления;
- отсутствие возможности осуществления запланированной эмиссии;
- возможность совершения анонимных платежей;
- невозможность отмены ранее совершённых платежей;
- другие, менее значимые свойства.

Официальное отношение к цифровым деньгам в нашей стране было озвучено 27 января 2014 года Банком России, заявившим[423], что в последнее время в мире получили определённое распространение так называемые «виртуальные валюты», в частности, Биткойн. То есть, указанные валюты, несмотря на научный интерес, были отнесены к суррогатным, а это означает, что согласно статье 27 Федерального закона «О Центральном банке Российской Федерации (Банке России)» выпуск на территории Российской Федерации денежных суррогатов запрещается.

[422] От англ. *bitcoin*: *bit* – бит и *coin* – монета.

[423] Об использовании при совершении сделок «виртуальных валют», в частности, Биткойн, 27 января 2014 года, http://www.cbr.ru/press/PR.aspx?file=27012014_1825052.htm.

06 февраля 2014 года в Генеральной прокуратуре Российской Федерации состоялось совещание[424] по вопросу правомерности использования анонимных платёжных систем и криптовалют на котором было всесторонне изучено положение дел и подтверждено официальное мнение: «*Получившие определённое распространение анонимные платёжные системы и криптовалюты, в том числе наиболее известная из них – Биткойн, являются денежными суррогатами и не могут быть использованы гражданами и юридическими лицами.*» А использование указанных платёжных средств со стороны юридических лиц на территории России «*будет рассматриваться как потенциальная вовлечённость в осуществление сомнительных операций в соответствии с законодательством о противодействии легализации (отмыванию) доходов, полученных преступным путём, и финансированию терроризма.*»

Мировой опыт: с предостережениями использования биткоинов ранее выступали также ЦБ Франции, Индии и Эстонии. Тем не менее ни один из этих регуляторов не объявлял биткоины вне закона. Наряду с ЦБ РФ, запретительные меры в отношении криптовалюты ввёл Нацбанк КНР. Китайский регулятор разрешил гражданам пользоваться биткоинами на свой риск, но обязал банки отказаться от операций с цифровыми деньгами.

С точки зрения электронной коммерции используемые для оплаты товаров биткоин-кошельки можно назвать новой платёжной системой, по типу существующих цифровых денег «Яндекс.Деньги» или «WebMoney», но проблема в том, что законом для каждой платёжной системы предусмотрен оператор, определяющий правила платёжной системы, при этом данный оператор получает лицензию на осуществление банковских операций. В случае с биткоинами такого оператора в России на данный момент не существует и не может существовать. Основой этому лежит принципиальное различие в выпуске (способе генерации) «цифровых» денег.

Поскольку развитие науки и технологий невозможно остановить, логично задаться вопросом, а как сеть (платежей) работает без единого оператора? Сеть полностью децентрализована, не имеет центрального администратора или какого-либо его аналога. Базовым элементом этой платёжной системы является программа-клиент с открытым исходным кодом. С помощью сетевого протокола прикладного уровня запущенные на множестве компьютеров клиенты соединяются между собой в одноранговую сеть. Клиенты существуют для Мобильных устройств с ОС Android, iOS, Blackberry, стационарных ПК с ОС: Windows, Linux, Mac OS X и для специальных аппаратных устройств: TREZOR и HW.1. Также существуют web-клиенты, серверная часть которых располагается в разных странах мира. Для обеспечения функционирования и защиты системы используются криптографические методы. Кроме этого зачастую оплаты происходят поверх анонимных сетей, в частности Tor. Также интересно отметить, что все совершаемые платежи в системе bitcoin, как и обобщённая статистика, доступны для анализа в реальном времени на сайте https://blockchain.info/.

Майнинг – термин обозначающих добывание (создание, генерирование) новых цифровых денег. За счёт решения вычислительных математических задач регулярно создаются новые цифровые деньги. Для каждого отдельного вида цифровых денег решаются свои, но примерно схожие задачи. Для самого процесса майнинга существует специальное бесплатное программное обеспечение. Оно использует вычислительную мощность процессора, и, если возможно – видеокарты. В силу роста курса обмена

[424] http://www.genproc.gov.ru/smi/news/genproc/news-86432/.

цифровых денег на реальные отдельные граждане пытаются на этом заработать создавать специальные компьютеры для майнинга – фермы. Такие компьютеры оборудуются несколькими видеокартами и мощным блоком питания и вентиляторами. В процессе работы производится много тепла. Некоторые умудряются параллельно с процессом майнинга, нагревать воду для отапливания помещений. Существуют фирмы занимающиеся майнигом в промышленных масштабах.

6.6.2. Отличия в работе юридических и физических лиц

Обычно физические и юридические лица работают с платёжными системами по разным схемам.

Физические лица используют систему электронных «кошельков» или электронных счетов, с которых они могут оплачивать товары и услуги юридических лиц или совершать платежи друг другу.

Юридические лица, как правило, работают с платёжными системами по договору комиссии, поручения или по агентскому договору. В соответствии с таким договором платёжная система оказывает продавцу услуги по продаже его товаров и услуг и получает за это комиссию в виде фиксированного процента от выручки. В рамках указанного договора платёжная система размещает на своём сайте, в разделе «Каталог магазинов», информацию о продавце, его товарах и услугах. Продавец, в свою очередь, размещает у себя на сайте информацию о том, что его товары и услуги можно оплатить с использованием данной платёжной системы. Расчёты между платёжной системой и продавцом происходят обычно один или два раза в месяц.

Пример платёжной системы в интернете – CyberPlat (*www.cyberplat.ru*), предоставляющая услуги для ведения электронной коммерции, включая обработку (процессинг) платежей и закрытый документооборот в режиме on-line. Расчётным банком системы является банк «Платина». К системе CyberPlat подключено более двух сотен интернет-магазинов. Согласно отчёту самой системы, её оборот за 2011 год составил более 200 млрд руб. Компания продвигает на рынке интернет-платежей следующие технологии:

- CyberCheck – защищённый документооборот по совершению сделок и их оплате, обеспечивающий обработку платежей и совершение сделок в сегменте B2B в режиме on-line;
- CyberPOS – обслуживание (эквайринг) платёжных карт в интернете. Обработка платежей в сегменте B2C в режиме on-line;
- интернет-банкинг (банковское обслуживание клиентов через интернет). Система предоставляет клиентам возможность управления счетами в банках – участниках системы через интернет.

С подробным описанием и схемами этих видов услуг можно познакомиться на сайте компании.

Другие отечественные платёжные системы в интернете: ASSIST (*www.ASSIST.ru*), Рапида (*www.rapida.ru*), Instant! (*www.paybot.com/*), PayCash (*www.paycash.ru*), WebMoney Transfer (*www.webmoney.ru*), КредитПилот (*www.creditpilot.ru*) и др.

6.7. Обеспечение безопасности информации в интернете

Понятие безопасности очень ёмкое. Оно подразумевает, что каким-то образом создано и поддерживается состояние в котором нет опасности. Обеспечение этого состояние не всегда возможно, а когда возможно, – процесс сложен и многогранен.

Лучше всего для его описания подходит идиома «бочки Либиха», состоящей из досок, то есть тех или иных принимаемых мер. При этом максимальный возможный запас воды в бочке будет определяться самой маленькой дощечкой (или самым ненадёжным звеном системы).

Количество граней, даже у самой большой бочки, измеряется десятками и их можно рано или поздно подогнать друг к другу, в реальных же системах число входящих в них компонентов (или звеньев) может оказаться в разы, а то и на порядки больше[425]. В последующих параграфах дан очень поверхностный и грубый обзор лишь двух из них.

6.7.1. Обеспечение конфиденциальности информации в интернете

Конфиденциальность передаваемой информации обеспечивается её шифрованием. При помощи процедуры шифрования отправитель сообщения преобразует его из простого текста в набор символов, не поддающийся прочтению без применения специального ключа, известного получателю. Получатель сообщения, используя ключ, преобразует переданный ему набор символов обратно в текст.

Процесс преобразования с помощью ключа простого текста в зашифрованное сообщение и обратно называется алгоритмом шифрования. Обычно алгоритмы шифрования общеизвестны и не являются секретом. Конфиденциальность передачи и хранения зашифрованной информации обеспечивается за счёт конфиденциальности ключа.

Ключ к шифру – конкретный набор символов и процедур, применяемых при шифровании и расшифровывании сообщений. Обычно степень защищённости информации зависит не только от алгоритма шифрования, но и от длины ключа, измеряемой в битах. Чем длиннее ключ, тем лучше защита[426], но тем больше вычислительных процедур необходимо провести компьютеру для шифрования и расшифровывания передаваемой информации, что замедляет передачу данных.

Существуют два вида алгоритмов шифрования [15]:

1) **симметричные**. В алгоритмах этого вида ключ зашифрования может быть вычислен из ключа расшифрования, и наоборот. В большинстве симметричных алгоритмов ключи зашифрования и расшифрования одинаковы;

[425] Не зря у нас в стране специальности дающие образование по вопросам безопасности, как и врачебное дело (медицина, «безопасность здоровья»), изучаются в вузах только по дневной форме обучения.

[426] В ряде зарубежных алгоритмов существует «подвох»: длина ключа не всегда однозначно влияет на стойкость алгоритма, так как часто используются не все биты ключа.

2) **асимметричные**. (Они же алгоритмы с открытым ключом.) Алгоритмы этого вида используют два ключа: один – для шифрования, другой – для расшифрования сообщения. Один из таких ключей является закрытым (секретным), другой – открытым (общедоступным). Ключ расшифрования не может быть (по крайней мере, в течение разумного периода) вычислен из ключа зашифрования.

Некоторые наиболее известные алгоритмы симметричного шифрования:

- **ГОСТ 28147–89** (RFC 4357) – советский и российский стандарт симметричного шифрования, введённый в 1990 году, также является стандартом СНГ;
- **DES** (Data Encryption Standard). Разработан фирмой IBM и широко используется с 1977 года. В настоящее время несколько устарел, поскольку применяемая в нём длина ключа недостаточна для обеспечения устойчивости к вскрытию методом полного перебора всех возможных значений ключа;
- **Triple DES**. Это усовершенствованный вариант DES, применяющий для шифрования алгоритм DES три раза с разными ключами. Он значительно устойчивее к взлому, чем DES;
- **Advanced Encryption Standard (AES)**, также известный как **Rijndael**. Алгоритм разработан в Бельгии. Работает с ключами длиной 128, 192 и 256 бит. На данный момент к нему нет претензий у специалистов по криптографии;
- **Skipjack**. Алгоритм создан и используется Агентством национальной безопасности США. Длина ключа 80 бит. Шифрование и расшифрование информации производится циклически (32 цикла);
- **IDEA**. Алгоритм запатентован в США и ряде европейских стран. Держатель патента – компания Ascom-Tech. Алгоритм использует циклическую обработку информации (8 циклов) путём применения к ней ряда математических операций;
- **RC4**. Алгоритм специально разработан для быстрого шифрования больших объёмов информации. Он использует ключ переменной длины (в зависимости от необходимой степени защиты информации) и работает значительно быстрее других алгоритмов. RC4 относится к так называемым потоковым шифрам.

Наиболее распространённые алгоритмы асимметричного шифрования:

- **RCA**. Алгоритм разработан в 1977 году и использует открытые ключи, обеспечивающие преобразование информации «только в одну сторону» (шифрование) за счёт сложности решения задачи факторизации (разложения на множители) больших чисел;
- **ECC** (Elliptic Curve Cryptography). Однонаправленность преобразований (шифрования) обеспечивается в этом методе сложностью математических вычислений, связанных с эллиптическими кривыми.

6.7.2. Обеспечение целостности информации в интернете

В настоящее время в системах документооборота широко используется электронная цифровая подпись (**ЭЦП**), которая является электронным эквивалентом собственноручной подписи. **ЭЦП** служит не только для аутентификации отправителя сообщения, но и для проверки его целостности.

При использовании **ЭЦП** для аутентификации отправителя сообщения применяются открытый и закрытый ключи. Процедура похожа на осуществляемую в асиммет-

ричном шифровании, но в данном случае закрытый ключ служит для подписи сообщения, а открытый – для её проверки.

Алгоритм применения **ЭЦП** состоит из ряда операций:

1) генерируется пара ключей: открытый и закрытый;
2) открытый ключ передаётся заинтересованной стороне (получателю документов, подписанных стороной, сгенерировавшей ключи);
3) отправитель сообщения шифрует его своим закрытым ключом и передаёт получателю по каналам связи;
4) получатель расшифровывает сообщение открытым ключом отправителя.

Создать зашифрованное сообщение, при расшифровании которого открытым ключом получается исходный текст, может только обладатель закрытого ключа, то есть отправитель сообщения. Использовать для этого открытый ключ невозможно.

Для защиты данных, передаваемых через интернет, используется протокол **SSL** (Secure Socket Layer). Этот протокол основан на комбинации алгоритмов асимметричного и симметричного шифрования.

Протокол может работать в трёх режимах:

1) при взаимной аутентификации сторон;
2) при аутентификации сервера и анонимности клиента;
3) при взаимной анонимности сторон.

При установлении соединения по протоколу SSL для данной сессии связи генерируется разовый ключ, который служит для симметричного шифрования данных, передаваемых в течение данной сессии. Разовый ключ генерируется на этапе установления соединения. При этом используются асимметричные алгоритмы шифрования.

Технология SET (Secure Electronic Transactions), используемая в системах электронной коммерции, появилась в 1996 году. Её основными разработчиками были MasterCard International и Visa International.

SET предусматривает использование цифровых сертификатов всеми участниками сделки, что позволяет проводить их однозначную взаимную аутентификацию.

Технология **SET** направлена на организацию максимально защищённых транзакций с использованием кредитных карт.

Взаимная аутентификация сторон и использование **ЭЦП** позволяют избежать проблем с отказами сторон от обязательств по сделкам и полностью закрыть проблему необоснованного отзыва плательщиками своих платежей.

В основе процедур защиты информации, используемых **SET**, лежат технологии шифрования RSA и DES, что обеспечивает относительно высокий уровень безопасности.

В общем случае алгоритм взаимодействия участников сделки по технологии **SET** выглядит следующим образом:

1) прежде чем начать работу с использованием **SET**, все участники сделки получают цифровые сертификаты у сертифицирующей организации (например, VeriSign (*www.verisign.com*) или Thawte (*www.thawte.com*)). Таким образом устанавливается однозначное соответствие между участником и его ЭЦП;
2) посетив сайт продавца, покупатель оформляет заказ и указывает способ оплаты при помощи кредитной карты;
3) покупатель и продавец предъявляют друг другу свои сертификаты;

4) продавец инициирует проверку платёжной системой предоставленной клиентом информации. Платёжная система передаёт продавцу результаты проверки;
5) при положительных результатах проверки по запросу продавца совершается перечисление денег.

Замечание. В [15, стр. 9] Брюс Шнайер написал: *в мире различают два типа криптографии: криптография, которая помешает вашей младшей сестре читать ваши файлы, и криптография, которая помешает читать ваши файлы правительствам крупных стран.*

1. Если идти по второму пути, то доверенным и надёжным алгоритмом шифрования на территории Российской Федерации является ГОСТ 28147–89 (на момент издания).

2. Обработку защищаемых данных следует вести только на доверенных аппаратных решениях, имеющих необходимые сертификаты соответствия. Например, заслуживает внимания «рутокен» – устройство для авторизации в компьютерных системах и защиты персональных данных, в котором на аппаратном уровне реализован российский стандарт шифрования.

3. Самый безопасный способ защитить данные: не использовать компьютеры, не иметь дело с такими данными. Если у вас в кармане нет 100 рублей, то даже самый ловкий мошенник не сможет их украсть. Невозможно украсть или потерять то, чего у тебя нет[427].

4. Когда ты «неуловимый Джо», то ты никому не интересен. Поэтому большинство пользователей интернета, живут счастливо не имея ни капли представления о том, что вы прочитали в двух последних параграфах.

Многие знания лишь множат печали. Вы готовы отказаться от импортных и закрытых операционных систем? Перейти на GNU/Linux. Не посещать сайты по протоколу https://, чьи SSL-сертификаты не подписаны российскими удостоверяющими центрами? Отказаться от зарубежных аппаратных компонентов и перейти на отечественные, в том числе полюбить процессор Эльбрус? Если да, то вы выбрали правильный учебник для начала своего обучения и вы стоите в начале очень долгого пути, который несомненно будет интересным. Если нет, то мы вас тоже поздравляем, вы – большинство, вы нормальный человек, и вы уже прочитали последнее слово последнего предложения последнего абзаца последнего подраздела последнего раздела последней главы этого учебника.

[427] Посмотрите российский юмористический сериал «Адаптация», где главный герой все серии пытался добраться до интересующей его информации, в а последней серии оказалось, что её нет в природе.

6.8. Литература к главе 6

1. *Леонтьев В. П.* Интернет 2010: Универсальный справочник – М.: Олма медиа групп, 2010. – 800 с.
2. *Журавлев А. В.* Персональный компьютер. Просто как дважды два. – М.: Эксмо, 2008. – 288 с.
3. *Аксак В. А.* Интернет. Просто как дважды два. – М.: Эксмо, 2009. – 288 с.
4. *Борисенко А. А.* Локальная сеть. Просто как дважды два. – М.: Эксмо, 2009. – 192 с.
5. *Таненбаум Э., Уэзеролл Д.* Компьютерные сети. – 5-е изд. – СПб.: Питер, 2012. – 960 с.: ил. ISBN 978-5-459-00342-0.
6. Сравнение интернет-хранилищ файлов: Яндекс.Диск, Dropbox, Google.Drive, SkyDrive, SugarSync, Box. // http://beamteam.ru/2013/02/cloud-storage-comparison-yandex-disk-dropbox-google-drive-skydrive-sugarsync-box-part1of4/, май 2013.
7. Сеть ОАО «РТКомм.РУ» // http://www.rtcomm.ru/geo/net/pics/big_maprus2010.jpg.
8. *Закляков П.* Пиринг // Системный администратор. – 2004. – № 6 (19). – С. 82–87.
9. *Жуков А. Е.* Дайджест новостей мировой и российской криптографии // http://www.ruscrypto.ru/netcat_files/File/ruscrypto.2013.001.zip.
10. *Касперски К.* У Google под колпаком! // Хакер. – № 5/101/07. – С. 64–68.
11. Китайские закладки: Непридуманная история о виртуализации, безопасности и шпионах // Хакер – № 12/11 (155). – С. 30–35. http://www.xakep.ru/58104.
12. Третье тысячелетие. Проект Россия – Кн. 3. – М.: Эксмо, 2009. ISBN 978-5-699-34786-5.
13. *Крелл М., Манн С.* Linux. Администрирование сетей TCP/IP – 2- изд. / пер. с англ. – М.: ООО «Бином-Пресс», 2008. ISBN 978-5-9518-0230-9, ISBN 0-13-032220-2.
14. Internet Society (ISOC) All About The Internet: History of the Internet // http://www.isoc.org/internet/history.
15. *Шнайер Б.* Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. – М.: ТРИУМФ, 2002. – 816 с.: ил. ISBN 5-89392-055-4.
16. Российская газета от 21.08.2013: WSJ: АНБ следит за 75% интернет-трафика в США. http://www.rg.ru/2013/08/21/nsa-site.html
17. *Погребенник А.* Какова цена анонимности в Сети // Системный администратор. – 2006. – №2(39). – С. 74–81.
18. Глобальной Сети исполнилось 45 лет. Как всё начиналось. (30 октября 2014) // http://habrahabr.ru/post/241929/.
19. *Прокофьев В.* Гаджеты – под запрет // Российская газета от 20 декабря 2017. – №288 (7454).
20. *Емельяненко В., Ивойлова И.* Мобильные ждёт мягкое выключение // Российская газета от 20 декабря 2017. – №288 (7454).
21. *Стоун О.* Нерасказанная история США / Оливер Стоун и Питер Кузник; пер. с англ. А. Оржицкого, В. Полякова. – М.: КоЛибри, Азбука-Аттикус, 2014. – 928 с. ISBN 978-5-389-06146-0.
22. Shodan – самый страшный поисковик Интернета // http://habrahabr.ru/post/178501/.
23. Примеры поиска в Shodan // http://habrahabr.ru/post/237787/.

6.9. Контрольные вопросы к главе 6

1. Зачем объединять компьютеры в сеть?
2. В чём суть клиент-серверной модели?
3. Как сети используются в организациях?
4. Как сети используют частные лица?
5. Назовите несколько проблем, которые возникли в обществе с развитием сетевого информационного обмена.
6. Что такое RFID?
7. Какова дальность считывания RFID-меток?
8. Как современные портативные ЭВМ (планшеты, телефоны) узнают своё местоположение?
9. Что такое сеть Tor?
10. Что такое CAPTCHA?
11. Что проводится тест Тьюринга на определение, кто перед вами: компьютер или человек?
12. Приведите примеры «за» и «против» использования бесплатного свободного и неконтролируемого Wi-Fi в школах.
13. Каким образом классифицируются компьютерные сети?
14. Приведите примеры сетевых эталонных моделей.
15. Какие сетевые уровни вам запомнились и каково их предназначение?
16. Что такое протокол?
17. Что есть интерфейс в сетевой модели?
18. Что такое RFC?
19. Поясните термины: архитектура, структура и топология сети.
20. Какова история возникновения интернета?
21. Что такое Интернет и Web (WWW)?
22. Какие организации осуществляют перспективное и текущее планирование функционирования интернета?
23. Как подразделяются операторы интернет-доступа по функционально-географическому признаку?
24. Какую функцию выполняет координационный центр национального домена сети Интернет?
25. Как можно зарегистрировать доменное имя в зоне .ru или .рф, сколько это примерно будет стоить?
26. Как классифицируются между собой интернет-провайдеры?
27. Назовите запомнившиеся вам магистральные операторы на территории России.
28. Что такое M9 или MSK-IX?
29. Назовите известные вам протоколы и поясните на каком уровне эталонной сетевой модели они работают.
30. Какие способы используются для адресации компьютеров в интернете?
31. Как работает символьная адресация в сети Интернет?
32. Приведите пример IP-адреса версии 4.
33. Зачем нужна сетевая маска?
34. Антивирусная программа-монитор показала повышенную активность с некоторого IP-адреса, как узнать, какому регистратору он принадлежит?
35. Зачем надо переходить на IPv6?
36. Какие недостатки, мешающие быстрому и прозрачному переходу, есть у IPv6?

37. Назовите известные вам способы подключения к сети Интернет оборудования конечных пользователей.
38. Какие сетевые параметры должны быть настроены в компьютере, чтобы он мог «выходить» в интернет?
39. Какие поисковые системы интернета наиболее известны?
40. Назовите сервисы, работающие с картографической информацией.
41. На каком сайте можно совершить виртуальную пешую экскурсию по улицам большинства российских городов?
42. На каких протоколах основана работа электронной почты?
43. Зачем нужны RSS-каналы или RSS-ленты?
44. Что такое twitter?
45. Назовите программы для общения через Сеть в реальном времени.
46. Что такое вебинар?
47. Какие преимущества у IP-телефонии?
48. Приведите способы обмена файлами через интернет.
49. Что значит облачное хранение файлов?
50. Что такое интернет-радио и интернет-телевидение?
51. Как по интернету слушать радио и просматривать телепередачи?
52. Какие существуют категории электронной коммерции?
53. Какие методы используются для обеспечения конфиденциальности информации в интернете?
54. Что такое язык HTML?

Приложение к главе 6. Примеры лабораторных работ

Лабораторная работа № 3. Настройка и исследование работы сетевого шлюза

Цель работы: получение знаний об основных подходах по настройке сетевых интерфейсов и созданию сетевых шлюзов на базе ОС Linux, а также приобретение практических навыков сетевого администрирования через консоль, настройки сетевого шлюза, использования программ сетевого мониторинга.

Содержание работы: Объедините три компьютера в две подсети, настройте сетевой шлюз. Исследуйте с помощью сниффера изменяемые параметры кадров, дейтаграмм и пакетов, проходящих через шлюз.

Организация и описание лабораторного стенда

Для организации лабораторного стенда вам потребуется всего три компьютера: два с одной сетевой картой и один с двумя, два коммутатора и четыре патч-корда соединяющие оборудование, как показано на Рисунке ЛР3.1. Вместо коммутаторов (switch) возможно использовать концентраторы (hub).

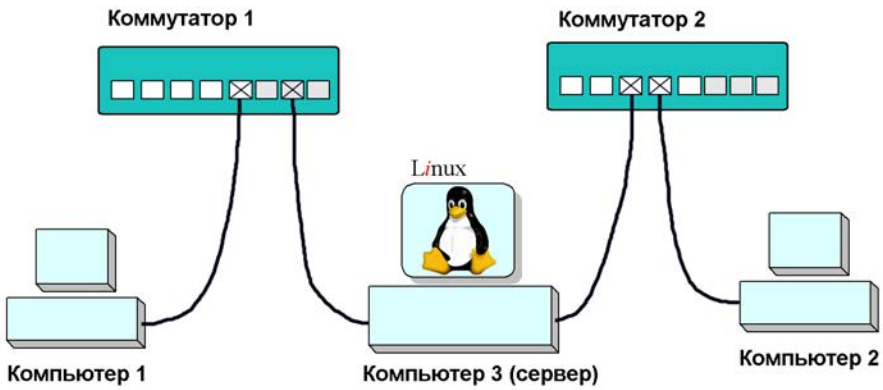


Рисунок ЛР3.1. Общая схема лабораторного стенда

Компьютеры соединены так, чтобы организовать две раздельные подсети:

Подсеть 1 (LAN1, Local Area Network 1) и Подсеть 2 (LAN2, Local Area Network 2).

Коммутатор 1 отвечает за подсеть 1, а Коммутатор 2 отвечает за подсесть 2. см. Рисунок ЛР3.2.

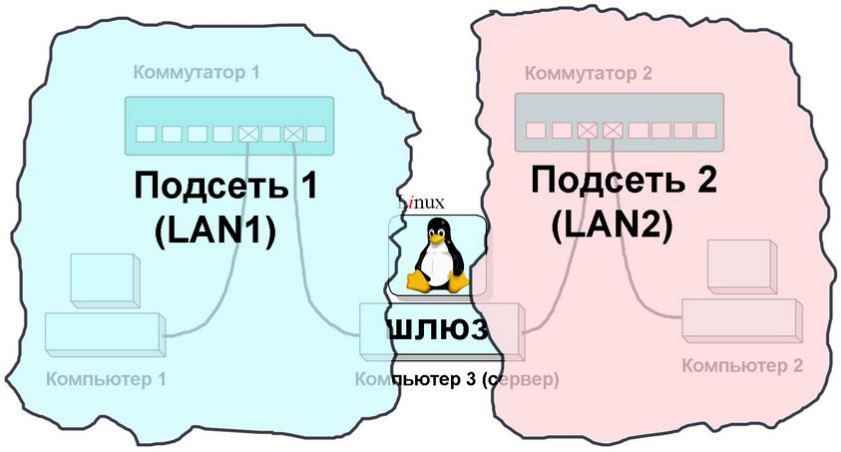


Рисунок ЛР3.2. Выделение подсетей LAN1 и LAN2

Использование одного коммутатора (с разбиением на подгруппы с помощью технологии VLAN) вместо двух допускается, но не рекомендуется, так как от этого теряется наглядность лабораторного стенда.

Компьютер 3 (сервер) с ОС Linux выполняет роль сетевого шлюза. Он осуществляет передачу данных из подсети 1 (192.168.1.0/24) в подсеть 2 (172.16.2.0/24) и обратно. см. Рисунок ЛР3.3.

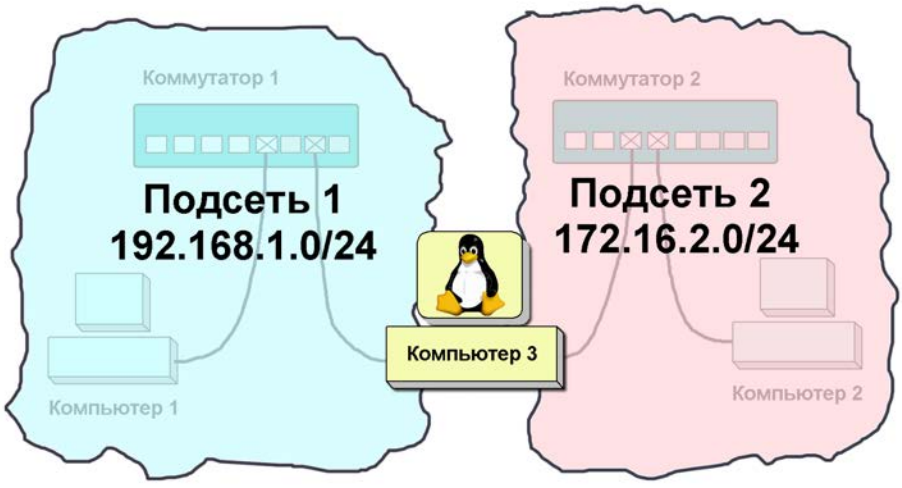


Рисунок ЛР3.3. Сетевой шлюз между двумя подсетями

Для организации обмена данными между подсетями 1 и 2 на Компьютере 3 установлены две сетевые карты (eth0 и eth1), которые подключены с помощью сетевых шнуров (патч-кордов) к коммутаторам 1 и 2, соответственно. См. Рисунок ЛР3.4.

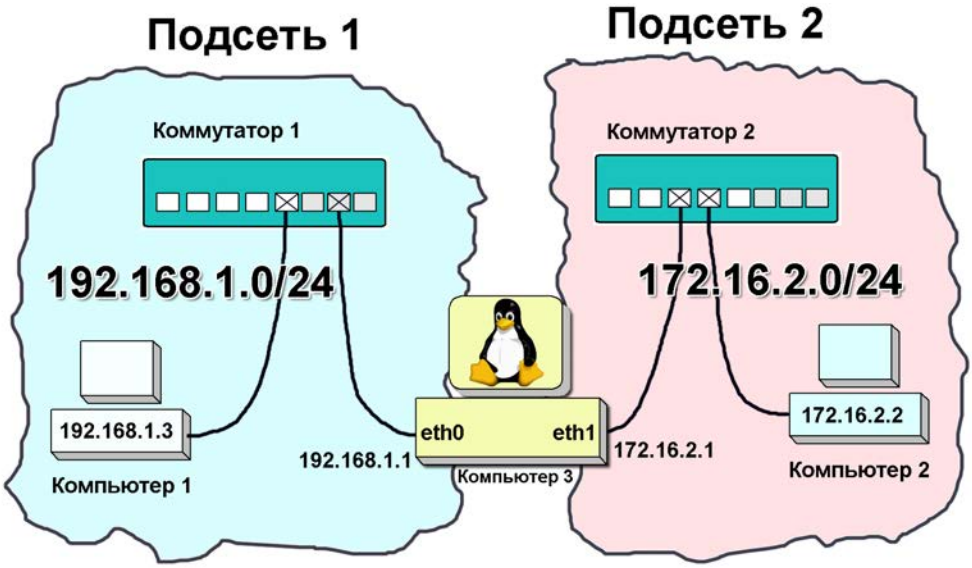


Рисунок ЛР3.4. Назначение IP-адресов компьютерам и сетевым интерфейсам

Компьютеру 1 присвоен IP-адрес 192.168.1.3.

Компьютеру 2 присвоен IP-адрес 172.16.2.2.

Компьютеру 3 присвоены два IP-адреса: 192.168.1.1 на интерфейсе eth0 и 172.16.2.1 на интерфейсе eth1.

Предполагается, что на компьютерах 1, 2 и 3 уже установлена операционная система Linux. При этом, в набор установленных программ входят программы снифферы – tcpdump и wireshark. Проведение работы возможно как на физическом оборудовании, так и в среде виртуализации Oracle VM VirtualBOX.

Основные теоретические сведения

Предполагаемые в данной работе исследования будут проводиться на стыке сетевого и канального уровней ЭМВОС (эталонной модели взаимодействия открытых систем, ISO/OSI-RM, International Organization for Standardization/Open Systems Interconnection-Reference Model) в применении к стеку протоколов TCP/IP.

5. Приложений
4. Транспортный
3. Сетевой
2. Канальный
1. Физический

Рисунок ЛР3.5. Уровни, на которых будет проводиться исследование

На физическом уровне

Для того чтобы обеспечить правильную работу на уровнях 2 и 3, необходимо, чтобы на нижележащем уровне, то есть на первом, всё было правильно организовано. Рассмотрим устройство лабораторного стенда начиная с этого (физического) уровня. Организация сети на физическом уровне не имеет принципиального значения для данной работы, поэтому мы предполагаем, что будет использоваться технология Fast Ethernet. Скорость передачи данных на канальном уровне (10, 11, 22, 54, 100, 108, 150, 300 или 1000 Мбит/с), выбор физической среды, режим передачи (полный дуплекс, полудуплекс или симплекс) также не играют роли. Так как нами была выбрана технология Fast Ethernet, предположим, что будет использоваться скорость передачи данных 100 Мбит/с в режиме полного дуплекса. Поскольку большинство читателей вряд ли имеют под рукой три лишних компьютера для экспериментов, скорее всего будет использоваться среда виртуализации, поэтому отметим, что гипервизор Oracle VM VirtualBOX предлагает на выбор несколько сетевых адаптеров. Если установлен VirtualBox Extension Pack, то по умолчанию моделируется работа Intel PRO/1000 MT Desktop (82540OEM) (См. Рисунок ЛР3.6.). Для данной работы выбор сетевой адаптера не принципиален. Обратите внимание, чтобы в поле «Неразборчивый режим» было выбрано «Разрешить всё», а MAC-адреса для всех сетевых интерфейсов и у всех виртуальных машин были разными. Справа от поля MAC-адреса в имеется кнопочка с закольцованными стрелочками, позволяющая сгенерировать псевдослучайный адрес.

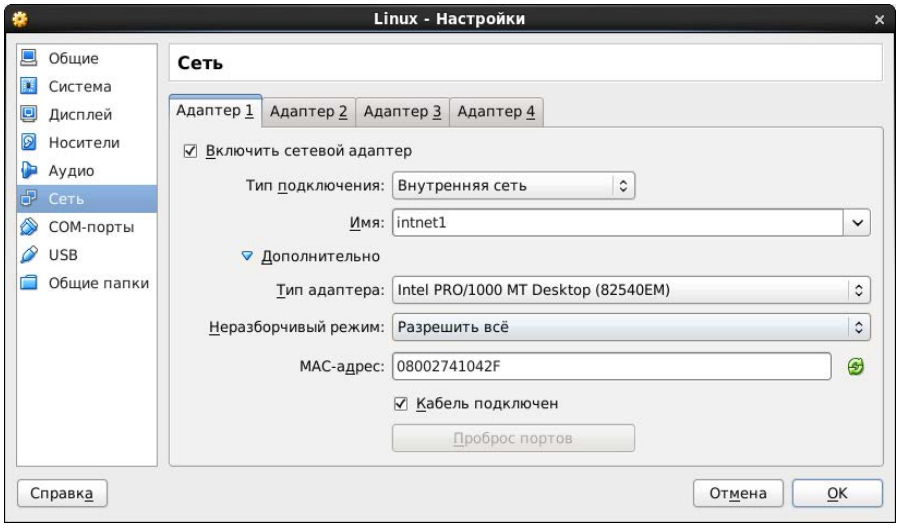


Рисунок ЛР3.6. Настройка сетевого адаптера в виртуальном компьютере

Топология подключения компьютеров к сети (то есть схема по которой Компьютеры 1, 2, 3 и Коммутаторы 1 и 2 соединены патч-кордами) тривиальна и смысла её обсуждать нет. Естественно, при использовании виртуализации патчкорды подключаются автоматически с обеих сторон (к коммутатору и к сетевому адаптеру) с помощью галочки «кабель подключен», а коммутаторы становятся разными за счёт внесения разных имён в поле «Имя» (см. Рисунки ЛР3.6 и ЛР3.7).

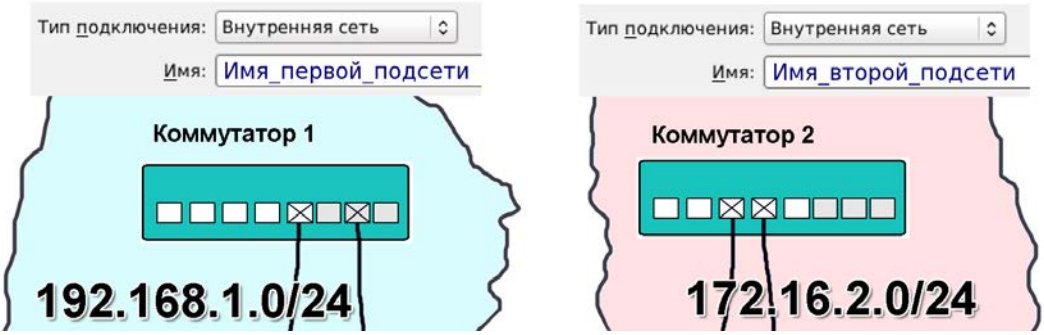


Рисунок ЛР3.7. Указание разных внутренних сетей с среде виртуализации

Замечание. Предполагается, что на физическом уровне сеть (длины кабелей, их категории, категории сетевых разъёмов, разводка контактов и пр.) выполнена с соблюдением стандартов сетей Ethernet. Данное требование при проведении работы особенно важно, так как в действительности часто можно встретить неправильно или плохо обжатые кабели, нарушение категорий, рекомендованных длин и пр. Для начинающих администраторов эти факторы являются ключевыми в закладывании знаний и становлении сетевой культуры. При использовании виртуализации подобных проблем не бывает как класса, зато возникают другие, более «глючные» проблемы. Например, часто виртуализируемые сетевые средства отказываются правильно (то есть так же как это бы делали физические их аналоги) работать при замене MAC-адреса на сетевом интерфейсе средствами виртуальной ОС, из-за чего виртуальный адрес выделенный сетевому адаптеру гипервизором при настройке виртуальной машины до её запуска и адрес установленный внутри виртуальной ОС в процессе её работы начинают отличаться.

На канальном уровне

Работа сети Ethernet на канальном уровне изначально не требует каких либо настроек.

В процессе работы будет предложено изменить MTU (Maximum Transer Unit) – максимальный размер пакета[428] в подсетях 1 и 2, чтобы они отличались.

На сетевом уровне

Для правильной работы сети компьютерам на сетевом уровне следует присвоить сетевые адреса так, чтобы в одной подсети у любых двух компьютеров не было одинаковых адресов. (Подробнее о выделении адресов см. параграф 6.3.4.2. *Численная адресация.*)

В данной работе адреса выделяются согласно RFC1918.

Для того чтобы адресовать компьютеры из других сетей следует настроить маршрутизацию. В данной работе Подсеть 1 является внешней для Подсети 2 и наоборот.

[428] Здесь и далее в работе под пакетом понимается порция данных. Это может быть как традиционный пакет протокола с установлением соединения, так и дейтаграмма сетевого уровня, так и кадр канального уровня. Объединение нескольких различных терминов в один – «пакет» не верный подход, но он упрощает жизнь тем, кто не имеет глубоких знаний по сетевой тематике и позволяет в работе сосредоточиться на других вещах.

Маршрутизация

В данной работе используется статическая маршрутизация. То есть, в работе прописывается в таблице маршрутизации компьютера 1, что компьютер 2 доступен через компьютер 3.

Для компьютера 2 прописывается, что компьютер 1 доступен через компьютер 3.

Компьютер 3 (сервер) мы настраивается так, чтобы он «видел» обе подсети и осуществлял перебрасывание транзитных пакетов из одной подсети в другую.

Краткий справочник команд

Для выполнения работы не помещает под рукой иметь небольшой справочник по примерам команд. Не они используются работе, но, возможно они вам понадобятся в жизни.

1. Посмотреть имеющиеся в системе сетевые интерфейсы и их параметры

```
$ ip link show
$ ip link sh
$ ip link
$ ip l
```

2. Посмотреть параметры только интерфейса eth0

```
$ ip link sh dev eth0
$ ip l sh dev eth0
```

3. Посмотреть присвоенные сетевым интерфейсам сетевые адреса

```
$ ip addr show
$ ip addr sh
$ ip addr
$ ip a
```

4. Тоже самое, но только для интерфейса eth0

```
$ ip address show dev eth0
$ ip addr show dev eth0
$ ip addr sh dev eth0
$ ip a sh dev eth0
$ ip a s dev eth0
```

5. Изменить состояние сетевого интерфейса eth0 (поднять/опустить)

```
# ip link set eth0 up
# ip link set eth0 down
# ip l s eth0 up
# ip l s eth0 down
```

6. Присвоить сетевой адрес 192.168.1.3 с маской /24 интерфейсу eth0

```
# ip addr add 192.168.1.3/24 dev eth0
```

7. Удалить сетевой адрес 192.168.1.3 с маской /24 у интерфейса eth0

```
# ip addr del 192.168.1.3/24 dev eth0
```

8. Присвоить сетевой адрес 192.168.1.3 с маской /24 и адресом широковещания 192.168.1.255 интерфейсу eth0

```
# ip addr add 192.168.1.3/24 broadcast 192.168.1.255 dev eth0
```

9. Присвоить сетевой адрес 192.168.1.3 с маской /24 физическому интерфейсу eth0 с указанием алиаса eth0:2 для имени сетевого интерфейса

```
# ip addr add 192.168.1.3/24 dev eth0 label eth0:2
```

10. Создать на базе интерфейса eth0 псевдоинтерфейс eth0.100 с VLAN ID=100

```
# ip link add link eth0 name eth0.100 type vlan id 100
```

11. Создать мостовой сетевой интерфейс br1 (аналог команды `brctl addbr br1`)

```
# ip link add br1 type bridge
```

12. Посмотреть записи в ARP-таблице

```
# ip neigh list
# ip neigh show
# ip neigh sh
# ip n s
# ip n
```

13. Создать статическую запись в ARP-таблице (аналог `arp -s _ip_ _mac_`)

```
# ip neigh add 10.0.0.1 lladdr 00:11:22:33:44:55 nud permanent dev eth0
```

14. Сбросить (удалить) все IP-адреса версии 4 у интерфейса eth0

```
# ip -4 address flush dev eth0
# ip -4 a f dev eth0
```

15. Посмотреть правила в таблице маршрутизации

```
# ip route list
# ip route list table all
# ip route show
# ip route show table all
# ip route sh
# ip ro sh
# ip r s
# ip r s table all
```

16. Добавить маршрут для сети 192.168.1.0/24 через интерфейс eth0

```
# ip route add 192.168.1.0/24 dev eth0
```

17. Удалить маршрут для сети 192.168.1.0/24 через интерфейс eth0

```
# ip route del 192.168.1.0/24 dev eth0
```

18. Добавить маршрут к адресу 192.168.5.6 через 192.168.1.1 (маршрут к адресу-посреднику должен просчитываться исходя из имеющихся записей в таблице маршрутизации)

```
# ip route add 192.168.5.6 via 192.168.1.1
```

19. Тоже самое, но для сети 192.168.6.0/24

```
# ip route add 192.168.6.0/24 via 192.168.1.1
```

20. Задать/удалить шлюз по умолчанию 192.168.1.1

```
# ip route add default via 192.168.1.1
# ip route add 0.0.0.0/0 via 192.168.1.1
# ip route del default via 192.168.1.1
# ip route del 0.0.0.0/0 via 192.168.1.1
```

21. Установить MAC адрес 00:11:22:33:44:55 интерфейсу eth0

```
# ip link set addr 00:11:22:33:44:55 dev eth0
```

Замечания. Некоторые сетевые адаптеры позволяют производить замену канального адреса интерфейса «на лету» (то есть, когда последний находится в состоянии «up»), для большинства остальных указанная операция разрешена к выполнению лишь при «опущенном» (находящемся в состоянии «down») сетевом интерфейсе. Также не всякий MAC-адрес может быть присвоен, существуют несколько десятков зарезервированных адресов (например ff:ff:ff:ff:ff:ff и др.). Если используется среда виртуализации, например Oracle VM VirtualBox, то для успешного сетевого взаимодействия в ней может потребоваться аналогичное изменение адреса в настройках сетевого интерфейса виртуальной конфигурации компьютера.

22. Сбросить (удалить) все IP-адреса версии 6 у интерфейса eth0

```
# ip -6 address flush dev eth0
# ip -6 a f dev eth0
```

Подготовка лабораторного стенда

Практическая часть подготовки лабораторного стенда является частью выполнения лабораторной работы и будет состоять из нескольких этапов:

1. Установка операционной системы Linux (например, CentOS 6.9) в среде виртуализации Oracle Vm VirtualBox.
2. Клонирование образа виртуальной машины 2 раза.
3. Проверка аппаратной конфигурации виртуальных машин: число доступных сетевых интерфейсов и их MAC-адреса. На шлюзе должно быть доступно 2 интерфейса, все MAC-адреса у виртуальных компьютеров должны быть разными..
4. Настройка подсети 1.
5. Настройка подсети 2.
6. Настройка компьютера 3 в качестве шлюза и его прописывание в подсетях 1 и 2.

Первые три этапа по установке операционной системы и проверке параметров читатели делают самостоятельно (или с помощью знакомых). В данном учебнике по установке ОС информации нет. Затем они приступают к настройке сетевых интерфейсов и маршрутизации (собственно подготовка стенда). По этому поводу ниже даются советы и комментарии.

После настройки всех трёх виртуальных компьютеров для единой работы в объединённой виртуальной сети, состоящей из двух подсетей, проводится проверка некоторых теоретических данных на практике – собственно само лабораторное исследование (задания по нему будут приведены ниже).

Настройка подсети 1

И так, перейдём к настройке компьютеров подсети 1.

Физические соединения

Для выполнения лабораторной работы необходимо, чтобы компьютеры 1, 2 и 3 были соединены, как указано на Рисунке 1. После соединения сетевых разъёмов[429] убедитесь, что у вас загорелись индикаторы подключения на физическом уровне, как на коммутаторах, так и на сетевых платах. Возле индикатора может быть написано слово «link», но это не означает, что происходит подключение именно на канальном уровне. Так как в сети Ethernet используется широковещательная среда для передачи данных, выявить факт того, что имеется соединение на канальном уровне сложно, поэтому чётко разграничить на каком уровне есть соединение (физическом или канальном) индикатор не может. Судить о наличии связи на канальном уровне можно косвенно, например по количеству успешно полученных и переданных пакетов.

Настройка сетевых интерфейсов

Типичная подача информации по настройке сетевых интерфейсов для тех, кто готовит лабораторный стенд будет выглядеть так: предполагается, что на компьютерах 1

[429] Конечно, в среде виртуализации вы этого сделать не сможете, и звука защёлкивания разъёма RJ-45 вы также не услышите, но, по крайней мере представьте как если бы это было в жизни на реальном оборудовании.

и 2 установлена ОС CentOS 6.x, а сетевой интерфейс на обоих называется eth0. Присвоим этим компьютерам IP-адреса 192.168.1.3 и 172.16.2.2 соответственно.

Для этого освежим в памяти теорию про сетевую маску (см. параграф 6.3.4.3. *Сетевая маска*), посмотрим на адреса выделенные для подсетей 1 и 2 и выделим сетевую маску «/24» или, что тоже самое в десятично-точечной записи «255.255.255.0».

Пояснение. Для изменения параметров сетевых интерфейсов исторически существуют две программы: ip (/sbin/ip) из набора программ iproute2 и старая утилита ifconfig (/sbin/ifconfig). Адрес интерфейсу может быть присвоен любой из них и мы приведём два примера. Однако, утилита ifconfig считается устаревшей и использовать её настоятельно не рекомендуется, с этой целью все устаревшие команды будут приведены зачёркнутыми ~~вот так~~.

```
# ip addr add 192.168.1.3/24 dev eth0
~~# ifconfig eth0 192.168.1.3/24~~
~~# ifconfig eth0 192.168.1.3 netmask 255.255.255.0~~
```

Замечание 1 + дополнительное задание. Для изменения сетевых параметров рекомендуется использовать утилиту ip, входящую в состав пакета iproute2 [7], так как она позволяет выполнять больше функций, чем ifconfig, route и arp вместе взятые. Большинство руководств, описаний и книжек до сих пор ссылаются на команды ifconfig и route как основные средства настройки сети, но обратите внимание на тот факт, что жизнь не стоит на месте, существуют случаи, когда `ifconfig` ведёт себя неадекватно при настройках современного сетевого оборудования. В документе [8] можно прочитать следующее: «The `ip link list` command provides the current true device state, whereas `ifconfig` shows the flag state which was set through ifconfig itself.» (Команда `ip link list` показывает текущее состояние настроек, в то время как команда `ifconfig` показывает те флаги, что были установлены ей самой.)

В качестве иллюстрации вышесказанного проведите простой эксперимент.

Выберите два адреса не использующиеся в вашей сети и проделайте следующие шаги:

1. Присвойте интерфейсу eth0 первый сетевой адрес командой ifconfig, например

```
# ifconfig eth0 192.168.1.3/24
```

или

```
# ifconfig eth0 192.168.1.3 netmask 255.255.255.0
```

2. Затем проверьте, что адрес именно такой

```
# ifconfig eth0
```

3. Присвойте второй адрес командой

```
# ip addr add 192.168.1.5/24 dev eth0
```

4. Запустите команду ifconfig и убедитесь, что по её мнению настройки сетевого интерфейса не изменились.

```
# ifconfig eth0
```

5. Запустите команду

```
# ip address show
```

и убедитесь, что сетевому интерфейсу присвоено два адреса.

6. Для проверки работает ли этот адрес выполните команду

```
# ping -c2 192.168.1.5
```

Убедитесь, что пакеты (ICMP Echo request) дошли до адресата и вам вернулся ответ (ICMP Echo reply).

Замечание. Запуск утилиты ping возможен и от пользователя.

```
$ ping -c2 192.168.1.5
```

Такой вариант более правильный с точки зрения безопасности: запускать команды с минимально возможными правами для них, но в данной работе здесь и далее он не используется поскольку может внести затруднения, обучающимся при выполнении заданий работы помимо внесения изменений в сетевые настройки и проверки правильности придётся дополнительно думать о том от какого пользователя запускать ту или иную команду.

7. Удалите второй адрес

```
# ip addr del 192.168.1.5/24 dev eth0
```

8. Повторите команду шага 6.

```
# ping -c2 192.168.1.5
```

На этот раз пакеты не будут доставлены.

Если вам понравился этот фокус, то занесите ваши наблюдения в отчёт по лабораторной работе.

Замечание 2. Вопросы присвоения нескольких сетевых адресов одному интерфейсу, использования сетевых алиасов и изменения адресов канального уровня в данной работе не рассматриваются.

Для проверки правильности присвоенных параметров используйте команды

```
# ip addr sh eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
UP qlen 1000
    link/ether 08:00:27:fd:d4:c2 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.3/24 brd 192.168.1.255 scope global eth0
    inet6 fe80::21f:c6ff:fe29:de7b/64 scope link
    valid_lft forever preferred_lft forever
```

или

```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 08:00:27:FD:D4:C2
          inet addr:192.168.1.3  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::21f:c6ff:fe29:de7b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1820 errors:0 dropped:0 overruns:0 frame:0
          TX packets:73 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:130686 (127.6 KiB)  TX bytes:11595 (11.3 KiB)
          Interrupt:20
```

(Примечание. В Windows для просмотра параметров интерфейсов используется команда ipconfig /all.)

Обратите внимание на параметр «MTU», по умолчанию, для интерфейсов ethernet он равен 1500 байт, а также что интерфейс поднят – «up».

Изменение состояния интерфейса может быть осуществлено командами

```
# ip link set eth0 down
# ip link set eth0 up
```

или

```
# ifconfig eth0 down
# ifconfig eth0 up
```

Если требуется автоматическая настройка сетевых интерфейсов во время загрузки операционной системы, указанные команды прописываются в файлы автозагрузки. Дополнительные программы, назовём их «скрипты-настройщики» считывают сетевые настройки из файлов и передают их вышеуказанным командам. В зависимости от операционной системы хранение сетевых настроек может отличаться. Обычно настройки находятся в каком-либо файле в директории /etc/sysconfig или в её поддиректориях. De facto в ОС Linux сетевые настройки применяются одним из двух способов – или из rc-сценариев (/etc/rc.d/init.d/network, /etc/rc.d/init.d/NetworkManager, /etc/rc.d/rc.network и др.), или при помощи программы NetworkManager. Указанные скрипты и программа демон NetworkManager считывают настройки из разных конфигурационных файлов (например, но не обязательно, из /etc/sysconfig/network-scripts/ifcfg-eth0) и настраивают интерфейсы. Вместе с настройками интерфейсов осуществляется запись данных в таблицу маршрутизации.

Обратите внимание, что существует несколько файлов NetworkManager, что вносит свою путаницу. Так, файл /etc/rc.d/init.d/NetworkManager является скриптом, а /usr/sbin/NetworkManager – двоичной программой. При разговоре, электронной переписке и в отчётах старайтесь уточнять о каком именно файле идёт речь.

Проверить выше сказанное можно командами find и file. Команда find найдёт все файлы (на тестовом компьютере их оказалось 8), а команда file подскажет вам их тип.

```
# find / -name NetworkManager
# file /usr/sbin/NetworkManager
# file /etc/rc.d/init.d/NetworkManager
```

В связи с тем, что алгоритм поведения программы NetworkManager сложно настраивается, а чаще не настраивается вовсе, данная программа может осложнить проведение лабораторной работы. Например, в процессе выполнения работы NetworkManager может обнаружить какую-либо Wi-Fi сеть и «подключиться» к ней (естественно, если в компьютере имеется беспроводной сетевой адаптер), изменив текущие сетевые настройки, либо при случайном отсоединении и повторном подсоединении сетевого разъёма, сбросить сетевые настройки интерфейса и начать получать адрес от DHCP сервера. Так как настройка NetworkManager не относится к данной работе мы его, как и пакетный фильтр, выключим. Для отключения демона NetworkManager выполните команду

```
# service NetworkManager stop
```

или в CentOS 7

```
# systemctl stop NetworkManager.service
```

Желательно выполнить эту команду до того, как вы будете производить изменение сетевых настроек, так как выключение может «сбить» уже имеющиеся настройки.

Настройка таблицы маршрутизации

Для того, чтобы лучше понять необходимость настройки маршрутизации в работе и почему будут прописываться в таблицу маршрутизации те или иные маршруты рассмотрим суть происходящих процессов в сети когда какое-либо приложение осуществляет подключение к другому приложению на другом компьютере. Программа будет запущена на уровне приложений (см. Рисунок ЛР3.8.) и, возможно, будет использовать один и протоколов данного уровня. (Если используется утилита ping, то выделить протокол уровня приложений невозможно, так как его нет.)

5. Приложений
4. Транспортный
3. Сетевой
2. Канальный
1.Физический

Рисунок. ЛР3.8. Программа запущена на уровне приложений

Когда программа соберётся передавать данные по сети, последние вынужденно попадут на уровень ниже (транспортный), где к ним будет дописан соответствующий заголовок. При переходе с уровня приложений на транспортный проблем не возникает, так как обмен данными в приложениях идёт через сокеты, а при создании подключения программа явно укажет какой из протоколов транспортного уровня следует использовать – TCP или UDP.

5. Приложений	socket(AF_INET, SOCK_STREAM,0)	socket(AF_INET, SOCK_DATAGRAM,0)
4. Транспортный	TCP	UDP
3. Сетевой	IP	IP
2. Канальный		
1.Физический		

При обращении к функции socket() одним из входных параметров явно прописывается тип подключения как SOCK_STREAM (для TCP) или SOCK_DATAGRAM (для UDP).

Следующий этап – это переход от транспортного уровня к сетевому. При попытке создания соединения с сервером при помощи функции connect(), ей передаётся структура sockaddr_in с sin_family с указанными PF_INET или PF_INET6, sin_port для указания порта прослушивания (в сетевом порядке), и sin_addr для указания IPv4 или IPv6 адреса прослушиваемого сервера (также в сетевом порядке). Эти данные определят протокол и адрес. До указанного момента, то есть до сетевого уровня, алгоритм прозрачен, а, вот, далее возникает вопрос, как быть при переходе от сетевого уровня на канальный?

У вас, как и у системы на этом этапе должно возникнуть два вопроса. Первый – это через какой интерфейс пакет выбросить в сеть и, второй, какой MAC-адрес получателя следует прописать в Ethernet кадре?

Получить ответы на эти вопросы как раз помогают таблица маршрутизации, arp-кэш и arp-таблица. Так как исследование протокола ARP и всё, что с ним связано, заслуживают отдельной лабораторной работы, а его работа обычно происходит прозрачно для пользователя, то в данный момент рассматривать ответ на второй вопрос мы не будем, а сразу перейдём к настройкам таблицы маршрутизации.

Следующим нашим этапом будет прописывание маршрутов.

Посмотреть имеющиеся маршруты можно с помощью любой из нижеследующих команд

```
# ip route show
# ip route
# ip r
# route -n
# netstat -r -n
```

(Примечание. В Windows для просмотра таблицы маршрутизации используется команда route print.)

Скорее всего, после присвоения IP-адреса сетевому интерфейсу (как описано выше) у вас будет выведено нечто похожее.

```
# ip route show
192.168.1.0/24 dev eth0  proto kernel  scope link  src 192.168.1.3  metric 1

# route -n
Kernel IP routing table
Destination   Gateway      Genmask          Flags Metric Ref    Use Iface
192.168.1.0   0.0.0.0      255.255.255.0    U     1      0        0   eth0

# netstat -r -n
Kernel IP routing table
Destination   Gateway      Genmask          Flags   MSS Window  irtt Iface
192.168.1.0   0.0.0.0      255.255.255.0    U         0   0        0   eth0
```

Замечание. «`metric 1`» может отсутствовать в выводе команды выше.

Фактически, программа присваивавшая IP-адрес интерфейсу прописала за вас эти маршруты. Видя их, ядро операционной системы понимает, что если вы адресуете пакет узлу с IP адресом 192.168.1.1, то его следует отправить на интерфейс eth0, потому как указанный адрес принадлежит подсети 192.168.1.0/24.

Если указанных выше маршрутов нет, а IP-адрес сетевому интерфейсу присвоен и он поднят, сетевой обмен осуществить не получится.

```
# ping -c2 192.168.1.1
connect: Сеть недоступна
```

Добавить маршрут в таблицу можно следующей командой

```
# ip route add 192.168.1.0/24 dev eth0
```

или

```
# route add -net 192.168.1.0/24 dev eth0
```

После указанных команд вы сможете осуществлять обмен пакетами с компьютера 1 в пределах подсети 1.

Для того, чтобы компьютер 3 мог также вести обмен данными в подсети 1, например отвечать на ваши запросы ICMP Echo request, которые формирует утилита ping с компьютера 1, необходимо на последнем произвести аналогичные действия: присвоить IP-адрес 192.168.1.1 интерфейсу eth0 и проверить наличие в таблице правильного маршрута. В случае отсутствия – прописать.

Настройка подсети 2

Аналогичным образом настраивают интерфейс eth0 компьютера 2 и сетевой интерфейс eth1 компьютера 3, входящих в подсеть 2. При выполнении этого этапа не должны встретиться сложности. Если вы будете испытывать затруднения, вернитесь к настройке подсети 1 и выполните её повторно.

В задании специально не было указано как сохранить настройки, поэтому, после выполнения перезагрузки компьютера через reboot или перезапуска сетевых сервисов через один из вариантов ниже

```
# service network restart
```

или

```
# service NetworkManager start
# service NetworkManager stop
```

вы сможете вернуться к первоначальной настройке системы и выполнить задание с нуля.

Последние две команды смотрятся не логично (вначале останавливаем, потом запускаем), но если вы внимательно читали выше, то было рекомендовано выключить NetworkManager, поэтому вместо команды

```
# service NetworkManager restart
```

мы вначале запускаем NetworkManager с параметром «start», чтобы он при запуске установил «свои» настройки и сбросил текущие, а после его выключаем через «stop», чтобы он не мешал выполнять работу.

Сервис network «менее навязчив», поэтому таких сложных операций для него не требуется.

Выбор команды зависит от того, какой сервис у вас был выбран в качестве основного при установке ОС. Даже если вы выполните оба варианта – ничего страшного не произойдёт.

После того как обе подсети будут настроены следует переходить к настройке шлюза между сетями.

Настройка компьютера 3 в качестве шлюза

На данный момент на компьютере 3 должна наблюдаться ситуация, что с него возможно осуществить обмен данными с компьютером 1 из подсети 1 и компьютером 2 из подсети 2.

Сначала проверим, что на компьютере 3 присвоены адреса интерфейсам, установлена правильная маска и интерфейс поднят. Фиолетовым цветом отмечены параметры на которые следует обратить внимание.

```
# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
UP qlen 1000
    link/ether 00:c0:26:a6:b1:72 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.1/24 brd 192.168.1.255 scope global eth0
    inet6 fe80::a00:27ff: ...
     valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
UP qlen 1000
    link/ether 00:c0:26:b3:18:7f brd ff:ff:ff:ff:ff:ff
    inet 172.16.2.1/24 brd 172.16.2.255 scope global eth1
    inet6 fe80::a00:27ff: ...
     valid_lft forever preferred_lft forever
```

```
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:C0:26:A6:B1:72
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1530919898 errors:0 dropped:2880 overruns:0 frame:0
          TX packets:1424546042 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1182864309 (1128.0 Mb)  TX bytes:3397533140 (3240.1 Mb)
          Base address:0xcc00 Memory:e6420000-e6440000

eth1      Link encap:Ethernet  HWaddr 00:C0:26:B3:18:7F
          inet addr:172.16.2.1  Bcast:172.16.2.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2713912538 errors:0 dropped:7127 overruns:0 frame:0
          TX packets:2669743822 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:707714899 (674.9 Mb)  TX bytes:2006908399 (1913.9 Mb)
          Base address:0xd000 Memory:e6440000-e6460000
```

Мы не акцентируем внимание на адресах широковещания (192.168.1.255 и 172.16.2.255) и других параметрах в выводе выше, потому как они должны быть правильно выставлены по умолчанию. На практике довольно часто возникают случаи неправильных настроек, когда сетевая маска и адрес широковещания могут противоречить друг другу.

В этом случае следует изменить неправильную настройку интерфейса через удаление неправильного присвоенного адреса (в синтаксисе команды на добавление вместо add пишется del) и добавление нового с правильным адресом широковещания. Например

```
# ip addr del 10.1.1.5/24 dev eth0
# ip addr add 10.1.1.5/24 broadcast 10.1.1.255 dev eth0
```

С утилитой ifconfig дело обстоит немного иначе

```
# ifconfig eth0 10.1.1.5/24 broadcast 10.1.1.255
# ifconfig eth0 10.1.1.5 netmask 255.255.255.0 broadcast 10.1.1.255
# ifconfig eth0 broadcast 10.1.1.255
```

Лучше все параметры задавать сразу правильно.

Проверим таблицу маршрутизации

```
# ip route sh
192.168.1.0/24 dev eth0  scope link
172.16.2.0/24 dev eth1  scope link
127.0.0.0/8 dev lo  scope link
```

```
# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
172.16.2.0      0.0.0.0         255.255.255.0   U     0      0        0 eth1
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0        0 lo
```

Замечание. В последних версиях утилиты ip маршрут для 127.0.0.1 не показывается, однако он есть, посмотреть вообще все маршруты, имеющиеся в системе, можно командами:

```
# ip route show table all
# ip route list table all
```

После того как вы убедились что таблицы маршрутизации не пустые и содержат правильные маршруты, проверим доступность компьютеров 1 и 2 от шлюза (компьютер 3).

```
# ping -c3 192.168.1.3
PING 192.168.1.3 (192.168.1.3) from 192.168.1.1 : 56(84) bytes of data.
64 bytes from 192.168.1.3: icmp_seq=1 ttl=128 time=0.149 ms
64 bytes from 192.168.1.3: icmp_seq=2 ttl=128 time=0.155 ms
64 bytes from 192.168.1.3: icmp_seq=3 ttl=128 time=0.129 ms
--- 192.168.1.3 ping statistics ---
3 packets transmitted, 3 received, 0% loss, time 1998ms
rtt min/avg/max/mdev = 0.129/0.144/0.155/0.014 ms
```

```
# ping -c3 172.16.2.2
PING 172.16.2.2 (172.16.2.2) from 172.16.2.1 : 56(84) bytes of data.
64 bytes from 172.16.2.2: icmp_seq=1 ttl=128 time=0.142 ms
64 bytes from 172.16.2.2: icmp_seq=2 ttl=128 time=0.121 ms
64 bytes from 172.16.2.2: icmp_seq=3 ttl=128 time=0.099 ms

--- 172.16.2.2 ping statistics ---
3 packets transmitted, 3 received, 0% loss, time 1998ms
rtt min/avg/max/mdev = 0.099/0.120/0.142/0.021 ms
```

Отметим, что компьютер 1 должен видеть адрес 192.168.1.1, но не будет видеть 172.16.2.2 (подсеть 2 – 172.16.2.0/24). А компьютер 2 должен видеть адрес 172.16.2.1, но не будет видеть подсеть 1 – 192.168.1.0/24. Проверить это можно также утилитой ping.

Замечание 3. В данной работе предполагается, что пакетные фильтры на компьютерах 1, 2 и 3 выключены. В случае если они включены (например сразу после установки системы), возможна ситуация, что пакеты могут не доставляться, так как они будут отфильтрованы. Для того чтобы стенд работал, правильнее прописать правила по принципу «запрещено всё то, что не разрешено», однако это сильно усложнит жизнь, так как правила для всех трёх хостов будут отличаться. Поэтому мы в данному случае выберем политику «всё разрешено» и отключим вообще пакетные фильтры, точнее, удалим все правила (нас больше интересуют запрещающие) и установим политику в режим «всё разрешено» (ACCEPT). Для этого на компьютерах 1, 2 и 3 следует выполнить команды

```
# iptables -P INPUT ACCEPT
# iptables -P OUTPUT ACCEPT
# iptables -F
```

А на шлюзе (компьютере 3), в дополнение к выше написанным командам, следует ещё выполнить

```
# iptables -P FORWARD ACCEPT
```

Замечание. Обратите внимание, что в дистрибутивах ОС Windows, например XP, также существует пакетные фильтры, которые по умолчанию могут не пропускать некоторые пакеты.

Так, по умолчанию «пропинговать» хост (Послать пакеты ICMP типа «Echo request» и получить в ответ пакеты ICMP типа «Echo reply».) у вас не получится, пока вы не перенастроите пакетный фильтр на сетевом интерфейсе вышеуказанной ОС.

Объединение

Для того чтобы появилась возможность обратиться с компьютера 1 на компьютер 2 следует выполнить следующие действия:

1. В добавление к уже имеющимся маршрутам (см. выше) прописать в таблице компьютера 1 маршрут на компьютер 2. В данном случае маршрут будет пролегать через компьютер 3 который будет шлюзом. Прописать можно маршрут как для всей подсети 2

```
# ip route add 172.16.2.0/24 via 192.168.1.1
```

или

```
# route add -net 172.16.2.0/24 gw 192.168.1.1
```

так и конкретно для IP-адреса компьютера 2

```
# ip route add 172.16.2.2 via 192.168.1.1
```

или

```
# route add -host 172.16.2.2 gw 192.168.1.1
```

2. Зеркально дописать маршруты на компьютере 2. Потому как даже если мы настроим шлюз и передадим пакеты с узла 192.168.1.3 на узел 172.16.2.2, мы не сможем получить ответы. Компьютер 2 не будет знать куда их отправить.

Пропишите маршруты по аналогии самостоятельно.

3. На компьютере 3 (шлюзе) разрешить переброс пакетов пришедших из одной подсети в другую, в некоторой литературе пишут «включить IP-продвижение данных».

Для этого нам надо присвоить «1» параметру ядра net.ipv4.ip_forward (/proc/sys/net/ipv4/ip_forward)

Например так:

```
# sysctl net.ipv4.ip_forward=1
```

или

```
# echo "1">/proc/sys/net/ipv4/ip_forward
```

Если "1" заменить на "0", то будет совершено обратное действие – переброс пакетов будет выключен.

Просмотреть включен переброс или выключен можно командами

```
# sysctl net.ipv4.ip_forward
```

или

```
# cat /proc/sys/net/ipv4/ip_forward
```

Для дистрибутивов Linux, отличных от CentOS (RedHat-совместимых), данные операции могут несколько отличаться. Фактически, после выполнения этой команды компьютер 3 можно назвать по выполняемой им работе маршрутизатором.

На данном этапе сеть из трёх компьютеров должна работать. Компьютер 1 и 2 должны видеть друг друга через сетевой шлюз – компьютер 3.

Задания на лабораторную работу

Опишем по шагам действия, которые необходимо выполнить в только что созданной лабораторной сети.

Выполнение работы разделено на несколько шагов. В процессе выполнения шагов делайте отметки в отчёте, куда заносите свои наблюдения, результаты вывода команд, значения выбранных параметров, если это требуется. Статус завершения выполнения команд: успешно или выдана ошибка.

1. Просмотрите таблицы маршрутизации всех трёх компьютеров 1,2,3. Перепишите или распечатайте маршруты к себе в отчёт.

Замечание 4. Записать данные выводимые любой консольной командой в файл можно с помощью перенаправления ввода вывода. Для этого, после любой команды напишите знак ">" и имя файла, куда хотите сохранить данные. Например

```
$ /sbin/ip addr show>/tmp/settings1.txt
```

Если файл /tmp/settings1.txt уже имеется, то всё его содержимое будет удалено перед записью. Если вы хотите дописать в конец файла, не стирая уже имеющуюся в нём информацию, – напишите подряд два знака больше ">>" вместо одного. Также, обратите внимание, что большинство команд будут выводить ошибки не в stdout, а в stderr. Для того чтобы осуществить запись данных и ошибок в один файл – используйте объединение потоков, например, вот так

```
$ /sbin/ip addr show>/tmp/settings1.txt 2>&1
```

Либо перенаправьте потоки в два разных файла, например, вот так

```
$ /sbin/ip addr show>/tmp/settings1.txt 2>/tmp/errors.txt
```

2. Просмотрите настройки всех сетевых интерфейсов у компьютеров 1, 2 и 3. Определите какие из интерфейсов активны в данный момент. Перенесите настройки в отчёт. (По идее все интерфейсы должны быть подняты, поскольку вы только что на-

строили и проверили работу сети перед началом выполнения этих заданий, однако, возможные ситуации что что-то уже перестало работать.)

3. В указанных настройках укажите, где имеется IP адрес, маска подсети, адрес широковещания. Найдите MAC-адрес каждого сетевого интерфейса и укажите их в отчёте. Увидеть MAC адрес с помощью команды ip можно следующим образом:

```
# ip link show
```

4. Обратите внимание на параметр MTU. Укажите его значение.

5. Проверьте прохождение пакетов от компьютера 1 до компьютера 2 и обратно на практике. Для этого, запустите на всех сетевых интерфейсах сниффер tcpdump и проанализируйте пакеты. Для выбора интерфейса – используйте ключ `-i`, например

```
# tcpdump -i eth1
```

Если во время запуска tcpdump интерфейс не был задан, то по умолчанию может выбраться совсем не тот, который вы планируете прослушивать.

Замечание 5. Если пакет tcpdump не установлен, как и любой другой, – найдите его на установочном диске или в репозитарии вашей ОС и установите командой

```
# rpm -ihv tcpdump-....rpm
```

или

```
# yum install tcpdump
```

для компьютеров подключенных к интернету.

Замечание 6. Для того чтобы запустить несколько копий tcpdump в текстовой консоли используйте многопользовательские возможности и многозадачность вашей ОС. Для переключения на вторую, третью и т.д. текстовые консоли – нажмите ALT+F2 или ALT+F3 и т.д., где цифра клавиши «F*» совпадает с номером консоли. По умолчанию текстовых консолей 6, их количество прописывается файлами /etc/events.d/tt1, /etc/events.d/tt2 и т.д. Ранее оно прописывалось в файле /etc/inittab. Из графической среды X-Window переключение в текстовую консоль осуществляется с донажатием клавиши CTRL в дополнение к указанным комбинациям, например переключение во вторую консоль можно осуществить с помощью нажатия CTRL+ALT+F2. Графической среде обычно отводится клавиша F6 или F7, но иногда может быть и F1. Если вы запустили «Linux в Linux'e» с помощью Oracle VM VirtualBox, и при нажатии выше указанных комбинаций клавиш оказываетесь в текстовой консоли хостового компьютера, а не виртуального, то обратите внимание на заданную вами «хостовую клавишу» в программе VirtualBox, по умолчанию это клавиша «правый Ctrl», в этом случае или переназначьте её или используйте для виртуальных машин следующие комбинации для переключания: правый_CTRL+F2, правый_CTRL+F3 и т. д.

Если на компьютере 1 выполнить команду

```
$ ping -c1 172.16.2.2
```

то вы должны увидеть прохождение одного пакета протокола ICMP типа Echo request и прохождение обратно одного ответного пакета протокола ICMP типа Echo reply.

Если маршрутизация настроена не верно и пакеты теряются, то, наблюдая прохождение пакетов с помощью сниффера, вы сможете определить участок на котором теряются пакеты.

Если вы видите все отправленные, так и полученные пакеты «дважды», то значит у вас есть ошибка в настройке среды виртуализации, чтобы её найти и понять вам понадобятся заголовки канального уровня.

6. Изучая документацию к программе tcpdump (например `man tcpdump` или справочник) определите:

6.1. каким ключом включается просмотр заголовков канального уровня (link-level header).

6.2. какими ключами осуществляется просмотр содержимого пакетов (дейтаграмм, кадров).

Обратите внимание, что в пакете просматривается только заданное число байт, начиная с начала пакета (snap length).

6.3. Определите каким ключом изменяется параметр snap length.

7. Просматривая сетевой трафик вместе с информацией о заголовках канального уровня обратите внимание на размер передаваемых пакетов (дейтаграмм, кадров), их содержимое и заголовки. Определите, какого размера пакеты по умолчанию посылает утилита ping. Данные наблюдения, как и наполнение пакетов ICMP занесите в отчёт.

8. Определите с какого и на какой меняются MAC адрес отправителя и MAC-адрес получателя в Ethernet кадрах на компьютере 3, наблюдая за передаваемыми данными через интерфейсы eth0 и eth1.

9. Найдите параметр утилиты ping, меняющий размер отсылаемых пакетов. (Обратите внимание, что для ОС Windows и ОС Linux данные ключи различны.) Определите минимальный и максимальный размеры пакетов, формируемые с помощью данного ключа. Проверьте правильность работы данного параметра с помощью снифферов.

10. Обратите внимание и ответьте на вопрос, что происходит с IP-дейтаграммой (при переходе от сетевого уровня к канальному), если её размер превышает 1500 байт?

11. Измените размер MTU в подсети 1 с 1500 байт на 1000 байт.

Произвести изменение можно командами

```
# ip link set eth0 mtu 1000
```

или

```
# ifconfig eth0 mtu 1000
```

Измените значение MTU как на интерфейсе eth0 компьютера 1, так и интерфейсе eth0 компьютера 3 (шлюза).

Проведите эксперименты, отправляя ICMP пакеты разных размеров с компьютера 1 на компьютер 2 и обратно. С помощью запущенных снифферов определите в каком случае происходит фрагментация пакетов, если они передаются из подсети с большим MTU в подсеть с меньшим MTU. Указанные наблюдения занесите в отчёт.

12. С помощью утилиты ping отправьте с компьютера 1 несколько пакетов на компьютер 2. Используя ранее запущенные снифферы, определите значения параметра TTL (Time To Live) в заголовках передаваемых пакетов. Так как данный параметр должен меняться, отследите изменения. Действительно ли при переходе через шлюз TTL уменьшается? Полученные наблюдения занесите в отчёт.

13. Используйте команду traceroute (аналог в Windows – tracert), для того чтобы определить прохождение пакетов из одной подсети в другую. Трассировку маршрута занесите в отчёт.

С помощью сниффера определите какой протокол использует программа traceroute.

14 (дополнительно). Если у вас под рукой имеются два компьютера с ОС Windows и ОС Linux – сравните реализацию команд ping и traceroute (tracert) на сетевом уровне используя снифферы. Обратите внимание, что их реализация отличается.

15 (дополнительно). Запустите сниффер wireshark (Wireshark Network Analyzer) из под графической среды X-Window (пакет wireshark-gnome) и проведите те же самые исследования, что вы провели со сниффером tcpdump.

Замечание 7. Для ОС Windows также имеются снифферы: Wireshark Network Analyzer (ранее назывался Ethereal) – он кроссплатформенный, а портированный tcpdump называется windump.

Просмотрите ещё раз отчёт по проделанной работе, запишите в него выводы.

Заключение

В результате выполнения заданий работы и подготовки лабораторного стенда вы должны были научиться настраивать сетевые шлюзы, получить навыки работы со сниффером и командной строкой.

Список литературы и ссылок к лабораторной работе № 3

1. Крелл М., Манн С. Linux. Администрирование сетей TCP/IP. 2-е изд, пер с англ. – М.: ООО "Бином-Пресс", 2008 (ISBN: 978-5-9518-0230-9, 0-13-032220-2).

2. Титтел Э., Чеппел Л. TCP/IP Учебный курс: Перс с англ. – СПб.: БХВ-Петербург, 2003.

3. RFC1918 Address Allocation for Private Internets (http://www.ietf.org/rfc/rfc1918.txt, русский перевод http://rfc2.ru/1918.rfc).

4. Internet Society (ISOC) All About The Internet: History of the Internet http://www.isoc.org/internet/history/.

5. RFC922 Broadcasting internet datagrams in the presence of subnets, http://www.ietf.org/rfc/rfc922.txt.

6. Полезные сетевые команды с примерами для разных ОС http://vds-admin.ru/content/view/19/.

7. Страница о наборе программ iproute2 http://www.linuxfoundation.org/collaborate/workgroups/networking/iproute2.

8. IPROUTE2 Utility Suite Howto http://www.policyrouting.org/iproute2.doc.html.

9. nettools, iproute2, полезные примеры // https://habrahabr.ru/post/320278/.

10. Ещё одна лабораторная работа по сетям // https://habrahabr.ru/post/320428/.

Лабораторная работа № 4. Исследуем сокеты

В работе исследуется взаимодействие процессов в ОС Linux через интернет сокеты созданные на базе протоколов TCP и UDP работающих поверх протокола IP версии 4.

Введение

В английском языке слово «socket» может употребляться как в обычной жизни, так и во многих отраслях науки: биологии, медицине, механике, электронике и компьютерный науках.

В русском языке термин «сокет» имеет более узкое смысловое наполнение, поскольку для большинства случаев у нас есть свои аналоги. Однако, аналоги не успели появиться, либо были вытеснены, в таких быстро развившихся областях науки и техники как электроника, вычислительная техника (компьютеры), компьютерные сети и интернет, где у сокета остаётся несколько смысловых значений, поэтому его употребление происходит либо в контексте, откуда становится ясно какое его смысловое наполнение используется, либо этот термин является составной частью другого многословного термина. Так, в литературе можно встретить без перевода: Berkeley sockets, Internet sockets, Network socket, Unix sockets, Unix domain sockets, raw socket, IC socket, CPU socket, ZIF Socket и др., а также их различные переводные и транслитерированные аналоги.

Таким образом, в отечественной литературе преимущественно «сокет» это:

- либо соединительный разъём или панелька (панель) для установки микросхемы, в том числе разнообразные исключительно для конкретной микросхемы процессора;
- либо название программного интерфейса для обеспечения обмена данными между процессами. Процессы при таком обмене могут исполняться как на одной ЭВМ, так и на различных ЭВМ, связанных между собой сетью.

В рамках данной работы будем использовать термин сокет для обозначения абстрактного объекта, представляющего конечную точку, используемую программой на языке Си, для соединения двух различных процессов поверх протокола IP версии 4. Или, по сути: сокет – это то, через чего две различные программы могут взаимодействовать друг с другом, обмениваться информацией.

Цели работы

Показать, что используя метод декомпозиции[430], возможно организовать силами начинающих программистов и/или системных администраторов относительно сложные процессы информационного обмена между двумя процессами в ОС Linux.

Показать, что при использовании абстракции «сокет» возможно понизить требования к знаниям программиста для решения задачи информационного обмена между процессами.

[430] Метод решения одной большой задачи путём её разделения и решения серии меньших задач.

Закрепить теоретические знания и дать (выполняющим данную работу) практические навыки использования сокетов.

Сформировать у выполняющих работу мысль о том, что практически все программы в интернете, независимо от того из под какой операционной системы они запущены, взаимодействуют между собой через сокеты, также как взаимодействуют программы в данной работы.

Усовершенствовать навыки работы через интерфейс командной строки.

Исследовать взаимодействие процессов в ОС Linux через интернет сокеты созданные на базе протоколов TCP и UDP работающих поверх протокола IP версии 4.

Описание лабораторного стенда

Для выполнения работы используется ОС Linux (например, CentOS 6). Поскольку понятие «сокета» изначально абстрактное, то тексты программ и задания разработаны таким образом, чтобы была возможность расширить (с минимальными изменениями или совсем без них) как перечень исследуемых вопросов, так и набор используемых для этого программных и аппаратных компонентов.

Для выполнения работы возможны различные сценарии (в зависимости от варианта установки), в каждом из которых указаны минимальные требования к организации лабораторного стенда, определены задания для последовательного самостоятельного выполнения.

Поскольку для выполнения некоторых заданий необходимо наличие прав root, самое простое решение – использовать виртуализацию, например на основе гипервизора VirtualBox (см. стр. 496), что позволит иметь полный доступ к возможностям операционной системы. С другой стороны, существует более сложный, но имеющий право на жизнь, вариант – пользователи остаются с обычными правами, но запуск нужных им команд разрешён через прописывание в /etc/sudoers всех необходимых комбинаций запуска программ через sudo.

В этом случае следует помнить, что:

- Для iptables придётся прописать разрешения для просмотра статистики с различными комбинациями ключей, а также для добавления и удаления правил без действия. Потенциальная опасность: пользователи увидят текущие настройки системы фильтрации и значении счётчиков не у своих правил.
- Для nc придётся прописать различные разрешающие шаблоны для создания серверных сокетов TCP и UDP для диапазона привилегированных портов (0–1023). Потенциальная опасность: пользователи могут «занять» все свободные привилегированные порты и тем самым осложнить возможный запуск других программ на этих портах.
- Проблема запуска веб сервера не актуальная если он всегда запущен.
- Для запуска снифферов придётся прописать различные комбинации в /etc/sudoers. Потенциальная опасность – получение пользователями доступа не к своему трафику, а также в случае использования ключа -w проблемы переполнения диска.

Примеры таких записей в файле /etc/sudoers:

```
user  ALL=(ALL)  NOPASSWD: /usr/sbin/tcpdump -i lo -n -nn -X
user  ALL=(ALL)  NOPASSWD: /usr/sbin/tshark -i lo -x -V
```

```
user  ALL=(ALL)  NOPASSWD: /sbin/iptables -L
user  ALL=(ALL)  NOPASSWD: /sbin/iptables -L -v -x -n
user  ALL=(ALL)  NOPASSWD: /sbin/iptables -L -v -x -n --line-numbers
user  ALL=(ALL)  NOPASSWD: /sbin/iptables -L INPUT -v -x -n --line-numbers
…
user  ALL=(ALL)  NOPASSWD: /usr/bin/nc -l ?
user  ALL=(ALL)  NOPASSWD: /usr/bin/nc -l ??
user  ALL=(ALL)  NOPASSWD: /usr/bin/nc -l ???
user  ALL=(ALL)  NOPASSWD: /usr/bin/nc -l ????
user  ALL=(ALL)  NOPASSWD: /usr/bin/nc -u -l ????
```

После этого эти команды можно запускать с правами пользователя, как

```
$ sudo /usr/sbin/tcpdump -i lo -n -nn -X
...
```

Какой путь выбрать – решать вам.

Возможны следующие варианты установок операционной системы:

Вариант 1. «Минимум»

Минимальная установка ОС (в неё уже включены программы netstat, ss, ps, iptables)+ дополнительная установка программ: apache+gcc+lsof+nc+tcpdump+telnet+tshark+wget.

Консольный интерфейс. Исследование взаимодействия двух процессов производится через интерфейс lo. Подключение во внешнюю сеть отсутствует.

Вариант 2. «Графический интерфейс»

Установка системы с графической средой в варианте «Desktop»+дополнительные программы как в Варианте 1+ дополнительно сниффер Wireshark Network Analyzer (пакет wireshark-gnome).

Сценарий проведения работы аналогичен Варианту 1, за тем лишь исключением, что дополнительно появляется возможность использования двух графических программ: браузер Firefox и сниффер Wireshark Network Analyzer.

Вариант 3. «Сетевой интерактив»

Установка программ производится как в Варианте 2 (или 1), но с поддержкой сети.

То есть работа проводится не одним человеком в одной операционной системе, а нескольких, например в компьютерном классе.

Появляется возможность создания сокетных соединений между физически разными компьютерами.

В случае наличия в локальной сети web-серверов или прямого доступа к сети интернет возможно расширение экспериментов заданиями по подключению к реальным web-серверам.

Несомненно то, что можно смоделировать работу любой сети и все задания выполнять с одного рабочего места на двух разных виртуальных машинах в пределах одного физического компьютера, однако у обучаемых остаётся больше впечатлений, если компьютеры с сокет-клиентом и компьютер с сокет-сервером оказываются физически

разными на небольшой удалении друг от друга, а сеть между ними – реальной. Таким образом, оптимальным вариантом для проведения лабораторной работы можно считать использование нескольких физических компьютеров объединённых в сеть, на каждом из которых, в независимости от используемой операционной системы, установлен гипервизор VirtualBox. Внутри гипервизора произведена установки одной виртуальная машины с ОС Linux согласно Варианту 3. Для обеспечения сетевого обмена между всеми виртуальными машинами, администратором оговорены правила использования локального IP-адресного пространства, а в виртуальные машины подключены к реальным сетевым интерфейсам хостовых машин в режиме «Сетевой мост».

Подготовка лабораторного стенда

Вариант 1

1. Установка операционной системы, добавление пользователя

По завершению минимальной установки, запустите в систему и добавьте пользователя «user» с паролем «user» (Там, где это возможно, старайтесь приучать себя работать с минимальными привилегиями, то есть с правами обычного пользователя.).

```
# adduser user
# passwd user
```

Поскольку не все необходимые пакеты имеются в минимальной установке, сразу подключаем установочный диск как репозитарий. В случае наличия соединения с интернет, шаги 1.1-1.2 можно пропустить, поскольку, по умолчанию, пакеты берутся из основного хранилища пакетов и/или его зеркал расположенных в интернете. Вопросы обновления пакетов (как и установки пакетов используя yum через прокси сервер) в данной работе не затрагиваются. Следует понимать, что на каждый пакет установленный с диска всегда могут иметься один или несколько более свежих с исправленными одними ошибками и внесёнными другими. Проведение работы предполагается в компьютерных классах без подключения к сети интернет.

1.1. **Подключение установочного диска** как репозитария для последующей удобной установки необходимых дополнительных пакетов с зависимостями.

Примонтируйте диск.

```
# mount /dev/sr0 /mnt
```

/dev/sr0 – привод для CD/DVD дисков, может отличаться в вашей системе
/mnt – куда монтировать диск

Совет. Если не знаете как в системе определился привод оптических дисков и определился ли, а также какой блочный псевдо-файл соответствует ему в /dev, то смотрите выводимые ядром сообщения во время загрузки системы, либо по её завершению воспользуйтесь командой dmesg.

1.2. **Определение источника получения пакетов.**

При установке пакетов через команду «rpm» можете столкнуться с проблемой разрешения зависимостей. Чтобы не тратить на это время, будет использоваться установщик пакетов yum, поскольку он умеет автоматически определять зависимости между пакетами и одновременно ставить недостающие. У yum имеется несколько настроеч-

ных конфигурационных файлов, а файлы с описаниями репозитариев находятся в директории /etc/yum.repos.d/.

Исправим источник получения пакетов в файле CentOS-Base.repo с помощью редактора vi, который изначально установлен в минимальной конфигурации.

```
# vi /etc/yum.repos.d/CentOS-Base.repo
```

Перейдите в режим правки со вставкой, для чего нажмите «i» и внесите следующие изменения.

```
$ diff -urN CentOS-Base.repo.old CentOS-Base.repo
--- CentOS-Base.repo.old  2015-08-03 19:13:00.000000000 +0400
+++ CentOS-Base.repo    2016-01-22 14:00:19.006142526 +0400
@@ -12,8 +12,9 @@

 [base]
 name=CentOS-$releasever - Base
-mirrorlist=http://mirrorlist.centos.org/?
release=$releasever&arch=$basearch&repo=os&infra=$infra
+#mirrorlist=http://mirrorlist.centos.org/?
release=$releasever&arch=$basearch&repo=os&infra=$infra
 #baseurl=http://mirror.centos.org/centos/$releasever/os/$basearch/
+baseurl=file:///mnt
 gpgcheck=1
 gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6

@@ -22,6 +23,7 @@
 name=CentOS-$releasever - Updates
 mirrorlist=http://mirrorlist.centos.org/?
release=$releasever&arch=$basearch&repo=updates&infra=$infra
 #baseurl=http://mirror.centos.org/centos/$releasever/updates/$basearch/
+enabled=0
 gpgcheck=1
 gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6

@@ -31,6 +33,7 @@
 mirrorlist=http://mirrorlist.centos.org/?
release=$releasever&arch=$basearch&repo=extras&infra=$infra
 #baseurl=http://mirror.centos.org/centos/$releasever/extras/$basearch/
 gpgcheck=1
+enabled=0
 gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6

 #additional packages that extend functionality of existing packages
```

Выход из текстового редактора vi (vim) через нажатие «ESC», «:», «w», «q», «Enter».

Совет. Если незнакомы с редактором vi, потратьте полчаса на его изучение. Затраченное время с лихвой окупится в будущем. Для этого поставьте и запустите программу vimtutor (пакет vim-enhanced). Если переменная окружения LANG=ru_RU.UTF-8, то уроки будут доступны на русском языке.

2. **Присвоение IP-адреса сетевому интерфейсу и настройка фильтрации с помощью iptables**.

IP-адрес присваивается серверу администратором статически (шаг 2.1.а), либо динамически по протоколу DHCP (шаг 2.1.б). Присвоение адреса используется для общения компьютеров друг с другом. Большая часть работы нацелена на выполнение иссле-

дований через интерфейс обратной петли (lo), поэтому если необходимо сократить время работы, то указанные этапы настройки можно пропустить.

2.1.а. **Присвоение IP-адреса** 192.168.1.4 сетевому интерфейсу eth0 вручную.

Присвоим адрес с сетевой маской /24 и адресом широковещания 192.168.1.255, после чего «поднимем» сетевой интерфейс.

```
# ip addr add 192.168.1.4/24 brd 192.168.1.255 dev eth0
# ip link set eth0 up
```

Посмотреть на результат, присвоенные IP-адреса и состояние сетевых интерфейсов можно командой

```
# ip addr show
```

или

```
# ip addr
```

Замечание. Для выхода за пределы локальной сети скорее всего вам понадобится прописать маршрут до вашего шлюза, например 192.168.1.1, а также указать способ разрешения доменных имён DNS.

Задать шлюз по умолчанию можно командой

```
# ip route add default via 192.168.1.1
```

Посмотреть таблицу маршрутизации

```
# ip route show
```

или, более коротко

```
# ip route
```

Для работы с DNS следует проверить в файле /etc/host.conf наличие строки

```
orderbind,hosts
```

а имено, наличие bind у параметра order, что должно направить систему в файл resolv.conf, где адреса DNS серверов прописываются отдельными строчками как

```
nameserver 192.168.1.1
```

2.1.б. **Присвоение IP-адреса** и других параметров «автоматически» по протоколу DHCP.

В этом случае предполагается, что у вас имеется работающий DHCP-сервер в сети, поскольку ему предстоит выделить адрес и другие сетевые параметры в ответ на ваш запрос по команде.

```
# dhclient eth0
```

Посмотреть на результат, присвоенные IP-адреса и состояние сетевых интерфейсов – аналогично тому как это описано в шаге 2.1а.

Замечание 1. В операционной системе, оговорённой вначале, по умолчанию правилами сетевой фильтрации разрешено прохождение исходящих DHCP-запросов и входящих DHCP-ответов. Таким образом, команда выше будет работать. Если же это не так, например в других системах, следует разрешить прохождение следующих IPv4 дейтаграмм:

Широковещательный запрос:

```
0.0.0.0.bootpc > 255.255.255.255.bootps: BOOTP/DHCP, Request
```

Широковещательный ответ DHCP-сервера:

```
192.168.1.1.bootps > 255.255.255.255.bootpc: BOOTP/DHCP, Reply
```

(192.168.1.1 – адрес отвечающего сервера)

Замечание 2. После перезагрузки сервера параметры присвоенные на шаге 2.1.а (или 2.1.б) пропадут. Хранение сетевых настроек при выключенном питании сервера для интерфейса eth0 осуществляется в файле /etc/sysconfig/network-scripts/ifcfg-eth0.

Определитесь с используемыми в вашей сети сетевыми адресами (либо возьмите сеть 192.168.0.0/24, либо уточните у администратора). Далее полагаем, что у первой виртуальной машины адрес 192.168.0.2, у соседней – 192.168.0.3, и т. д.

2.2. **Настройка средств фильтрации**

Отключите средства фильтрации, установлением разрешительной политики и удалением всех имеющихся правил. (После установки ОС, в ней по умолчанию запрещены входящие соединения, что делает невозможным подключение к прослушивающим сокетам (которые будут создаваться в процессе работы) извне системы.)

```
# iptables -P INPUT ACCEPT
# iptables -P OUTPUT ACCEPT
# iptables -F
```

3. **Запуск web-сервера**

```
# service httpd start
```

4. **Доустановка требуемых пакетов**

Доустановите (если вы предусмотрительно не сделали это в процессе установки) требуемые вашему варианту пакеты. Также добавьте компилятор g++ (пакет gcc-c++) и файловый менеджер mc. В работе они не используются, но могут пригодиться вам в дальнейшем.

```
# yum install mc httpd gcc gcc-c++ lsof lynx nc tcpdump telnet wireshark
wget
```

После разрешения зависимостей также будут установлены: apr, apr-util, apr-util-ldap, cloog-ppl, cpp, glibc-devel, glibc-headers, gnutls, gpm-libs, httpd-tools, kernel-headers, libgomp, libpcap, libsmi, libstdc++-devel, mailcap, mpfr, perl, perl-Module-Pluggable, perl-Pod-Escapes, perl-Pod-Simple, perl-libs, perl-version, ppl.

На этом настройка лабораторного стенда по Варианту 1 завершена. Варианты конфигураций 2 и 3 не сильно сложнее.

Вариант 2

Доустановите пакеты как для Варианта 1 + wireshark-gnome.

Вариант 3

Создайте в менеджере виртуальных машин VirtualBox новую виртуальная машину и выполните действия для Варианта 2. Выключите машину. Создайте клон получившейся виртуальной машины. Для этого используйте копирование внутри менеджера виртуальных машин, поскольку в этом случае будет произведена замена идентификаторов дисков и MAC-адресов сетевых интерфейсов, что позволит избежать проблем взаимодействия двух виртуальных машин между собой в будущем.

Перенесите клонированную виртуальную машину на другой компьютер. Убедитесь, что в настройках сетевых интерфейсов у виртуальных машин Тип подключения указан как «Сетевой мост», а MAC-адреса у всех используемых копий виртуальных машин разные.

Краткие теоретические сведения

Предполагается, что приступая к выполнению лабораторной работы, лица планирующие её выполнить уже имеют достаточный багаж теоретических знаний о сокетах, либо оперируют с ними как с абстрактными понятиями. В первом случае используется путь «от теории к практике». Во втором случае, глубокого понимания не требуется и изучение происходит «от практики к теории». Если ваш случай второй, либо хотите освежить в своей памяти теорию, то см. [1] и [2]. Общая схема по которой производится сокетное соединение для протоколов TCP (тип сокета SOCK_STREAM) и UDP (тип сокета SOCK_DGRAM) представлена на Рисунках 1 и 2.

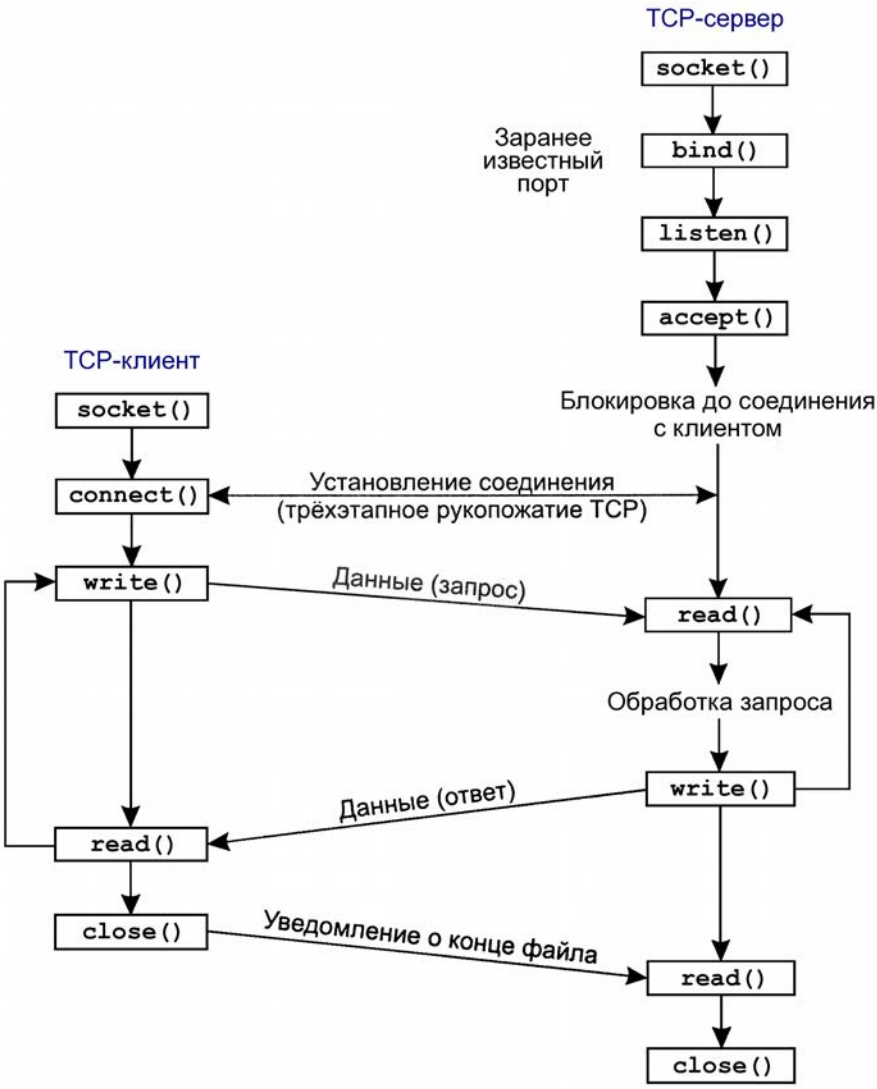


Рисунок ЛР4.1. Функции сокетов для элементарного клиент-серверного TCP соединения (схема взята из [2, стр.119])

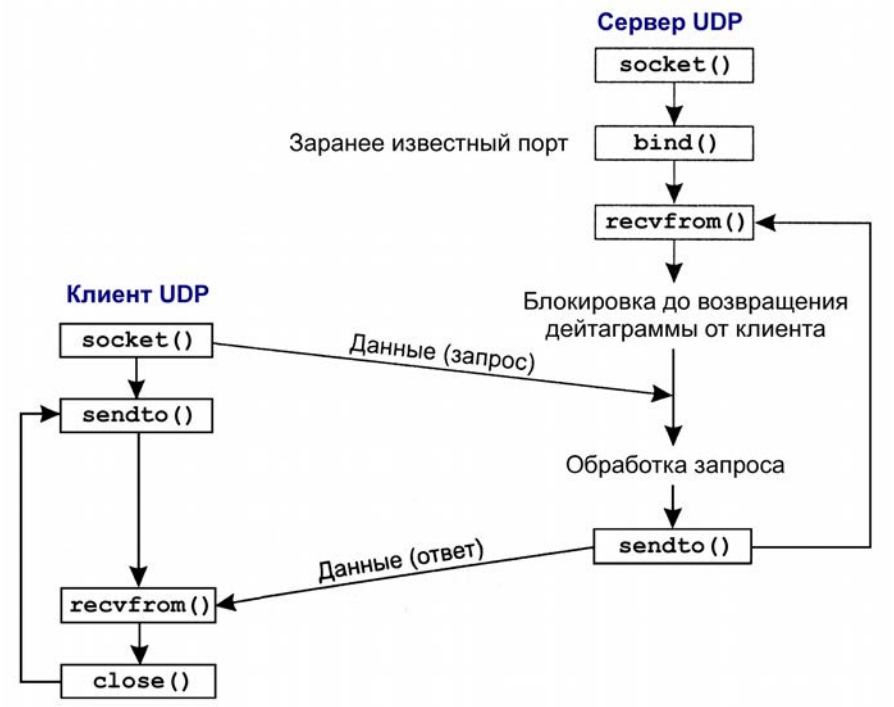


Рисунок ЛР4.2. Функции сокетов для элементарного клиент-серверного UDP соединения (схема взята из [2, стр.254])

Описание хода проведения работы и отдельных компонентов

Для исследования межпроцессного информационного обмена (обмена данными между двумя процессами), в работе предлагается произвести связь процесса клиента и процесса сервера, используя сокеты в режиме клиента и сервера, соответственно. Для этого в распоряжении обучаемых имеется (см. Таблицу ЛР4.1) 6 вариантов программ использующих сокет в режиме клиента, 3 варианта программ открывающих сокет на прослушивание на стороне сервера и 7 различных инструментов для исследования тех и иных аспектов связанных с работой указанных программ и передачей данных.

Таблица ЛР4.1. Схема исследований

клиент	инструмент исследования	сервер
telnet nc* программа на Си* браузер Firefox браузер lynx wget	tcpdump tshark Wireshark netstat ss lsof iptables ps	nc* программа на Си* web-сервер apache

* Все приведённые в таблице клиент-серверные средства поддерживают TCP сокеты, а поддерживать протокол UDP умеют лишь «nc» и «программа на Си».

Хотя, общее число комбинаций записей в таблице ЛР4.1 будет 6 × 3 × 8 = 144 и кажется большим, число принципиально исследуемых моментов на порядок, а то и два меньше. То, что в ряде комбинаций наблюдаемые результаты будут не так сильно отличаются друг от друга должно быть очевидным после ознакомления с теорией, а на проверку каждого такого случая потребуется не более 5-10 секунд.

При наличии технической возможности и достаточного опыта у обучаемых, число используемых программ и инструментов, как и число вариантов использования их друг по отношению к другу, в процессе исследования можно самостоятельно увеличить, а интересные наблюдения отразить в отчёте. Ниже даётся краткое описание всех компонентов из Таблицы ЛР4.1, типичные способы их использования, примеры запуска и ожидаемые результаты.

telnet

Изначально протокол TELNET (уровень приложения (прикладной), RFC 854) использовался для предоставления двухстороннего доступа терминальным устройствам к серверу. Таким образом пользователь мог через терминал взаимодействовать с различными процессами на сервере, в том числе и с командным интерпретатором. В протоколе TELNET все данные передаются по сети в открытом виде и могут быть легко перехвачены. По этой причине для удалённого управления не сегодня протокол telnet практически не используется (его заменил прокол SSH), но для целей данной работы он прекрасно походит. Программная реализация консольного клиента называется также telnet, но в нижнем регистре. На транспортном уровне она использует протокол TCP. Этой программой нам и предстоит воспользоваться в процессе работы.

Например, подключение к серверу 127.0.0.1 на порт 1234 протокола TCP:

```
$ telnet 127.0.0.1 1234
```

тоже самое, но на порт 80 протокола TCP:

```
$ telnet 127.0.0.1 80
```

Замечание 1. Аналогичная программа в ОС Windows называется «telnet.exe». Если в Windows XP она была установлена по умолчанию, то начиная с Windows Vista (в том числе и на серверных линейках) её убрали. Установить её можно следующим образом:

Панель управления -> Программы и компоненты -> Включение или отключение компонентов Windows; Клиент Telnet – поставить галочку. Либо можно произвести установку через консоль командой «pkgmgr /iu:"TelnetClient"».

Замечание 2. Вместо клиента telnet сегодня довольно часто используется бесплатная графическая утилита PuTTY имеющая не только ряд других полезных свойств (например, поддержка протокола SSH, работа с различными кодировками), но и имеющая реализацию как под Windows, так и под UNIX и Linux. [4]

Замечание 3. Программа telnet в ОС Linux не использует пользовательский сигнал SIGINT, возникающий при нажатии клавиш CTRL+c, для выхода из программы. Чтобы выйти из сеанса работы telnet-клиента в случае зависания сервера, нажмите ctrl-], после чего у вас появится приглашение «telnet>», где наберите quit для выхода из программы.

nc

nc – это программа для создания соединений с использованием сокетов протоколов TCP и UDP и последующей передачей данных по ним.[5] Она позволяет как осу-

ществлять клиентские подключения с использованием указанных протоколов, так и создавать на серверной стороне сокеты находящиеся в режиме ожидания входящих соединений от клиентов.

Примеры использования nc в режиме «сервера»:

Создание прослушивающего сокета протокола TCP на порту 1234 для всех имеющихся у хоста IP-адресов.

```
$ nc -l 1234
```

Заметим, что для обычного пользователя доступны для прослушивания (для создания входящих соединений) лишь непривилегированные номера портов, то есть с номерами из диапазона 1024–65535. Для портов с номерами 1023 и меньше потребуются права суперпользователя.

```
# nc -l 1023
```

По умолчанию, программа nc использует протокол TCP. Для использования протокола UDP необходимо дополнительно указывать ключ «-u». Например, открытие на прослушивание порта 1234 протокола UDP.

```
$ nc -l 1234 -u
```

Выдавать содержимое файла 1.txt всем присоединившимся к сокету на 1080-м порту протокола TCP.

```
$ nc -l 1080<1.txt
```

Совместно с циклом на bash возможно создание временной заглушки для замены web-сервера на 80 порту.

```
# while true; do nc -l 80<index.html; done
```

Работает это следующим образом. При обращении к сокету HTTP запросы клиента игнорируются, и сразу начинается передача данных из файла index.html клиенту. Большинство браузеров при получении такого потока данных игнорируют отсутствие заголовков от сервера и выводят полученное содержимое пользователю. Имя файла (index.html в нашем случае) может быть любым, поскольку клиент его не видит.

Примеры использования nc в режиме «клиента»:

Подключение к хосту с именем samag.ru на порт 80 по протоколу TCP.

```
$ nc samag.ru 80
```

Подключение к хосту с IP адресом 192.168.3.1 на порт 53 по протокола UDP.

```
$ nc -u 192.168.3.1 53
```

В man'e к nc можно найти много других полезных примеров по её использованию.

Обратите внимание, что программа nc в режиме клиента в случае отсутствия возможности установления соединения по разному себя ведёт для протоколов TCP и UDP.

Поскольку почтовые и многие другие сетевые службы используют текстовый режим работы с клиентом (либо всегда, либо только вначале соединения), то nc и telnet могут использоваться администраторами с целью «ручной» диагностики работы сетевых сервисов.

Замечание. Обратите внимание, что существуют и другие, аналогичные nc (netcat) программы: ncat, socat (Multipurpose relay (SOcket CAT)). Первая была специально написана для проекта Nmap. Она имеет несколько дополнительных возможностей, которые не потребуется в данной работе, но их возможно стоит знать, например, поддержка прокси, работа с SSL:

```
$ ncat -l -p 6500 --ssl --ssl-cert /path/host.crt --ssl-key /path/host.key > out.tgz
$ tar -zc ~ | ncat --ssl machineb 6500
```

Вторая программа умеет делать множественные перенаправления сокетов.

браузер Firefox

Популярный кроссплатформенный браузер с GUI интерфейсом. Доступен на официальном сайте [6]. При необходимости возможна его замена на любой другой браузер. Для обращения к какому-либо серверу, в адресной строке браузера следует указать его адрес и нажать {enter}. По умолчанию используется протокол TCP и обращения направляются на порт 80. В случае необходимости использования отличного порта, например 1180, его следует указать через знак «:» в адресной строке. Например, http://192.168.1.1:1180

Удобен тем, что через обращение к странице «about:config» позволяет редактировать многие параметры своей работы. Также имеется возможность добавления большого числа расширений (файлы с расширением .xpi), добавляющих различную полезную функциональность.

браузер lynx

Один из первых текстовых браузеров. Недостатки: не поддерживает отрисовку таблиц HTML, фреймы HTML и JavaScript. Плюсы: небольшой, консольный, то есть работающий только через CLI интерфейс, кроссплатформенный (Linux, Windows, UNIX, OS/2, DOS и др.), умеет работать с простыми формами.

Обратите внимание, что созвучно названию lynx также имеется совершенно другой программный продукт: Links. Это уже текстово-графический браузер с поддержкой фреймов, вкладок, таблиц и javascript. Также поддерживается широкая кроссплатформенность и даже имеются 64-битные сборки. В данный момент существуют две основные ветки разработки на базе кода links: ELinks – для повышения функциональности за счёт встроенного языка программирования Lua и Links2 – с поддержкой отображения графики.

Пример обращения к web-северу по адресу 192.168.1.1:

```
$ lynx 192.168.1.1
```

Использование текстовых браузеров может быть оправданно в случае если на сервере по каким-то причинам отсутствует графическая среда X-Window. Например таковым сервером запросто может оказаться домашний маршрутизатор или какое-либо иное бытовое устройство с сетевым интерфейсом и ОС Linux «на борту».

Замечание. Обратите внимание, что существует библиотека AAlib [7], с помощью которой возможно просматривание графических объектов посредством «ascii-графики» непосредственно на текстовой консоли. Если же «просмотр» скачиваемых с web-сервера файлов не требуется, а требуется лишь их сохранение «на диск», то удобнее для этих целей использовать утилиту wget.

wget

Свободная неинтерактивная консольная программа для загрузки файлов по сети. Поддерживает протоколы HTTP, FTP и HTTPS. Умеет работать через HTTP прокси. Имеются сборки под Linux, Windows, UNIX. Включена практически почти во все дистрибутивы GNU/Linux. Из полезного – умеет самостоятельно находить в html документах ссылки на другие документы, находящиеся на сервере и также скачивать их, что используется в функции «выкачивания сайтов», вплоть до указанного уровня вложенности. Многие сторонние коммерческие программы предпочитают не изобретать вело-

сипед и активно пользуются штатными функциями wget, например для получения файлов обновлений.

tcpdump

Консольный анализатор трафика (сниффер), позволяет мониторить и анализировать сетевой трафик, проходящий через сетевой интерфейс. Часто используется сетевыми администраторами для оценки сетевой активности и диагностики сетевых подключений.

Поскольку доступ к сетевым интерфейсам имеется лишь у администратора, то запуск программы в режиме подключения к сетевым интерфейсам системы (в том числе и lo) возможен лишь с его правами.

Среди полезных (для данной работы) опций у программы следует отметить следующие:

-e выводит заголовки канального уровня для каждого перехваченного кадра;

-i имя_интерфейса – задаёт сбор пакетов с указанного интерфейса. Параметр «any» соответствует всем сетевым интерфейсам. Если интерфейс не задан, tcpdump ищет в системе список доступных интерфейсов и выбирает первый (исключая loopback, в том числе даже если интерфейс находится в состоянии down). Также возможен анализ данных передаваемых по USB, например через обращение к интерфейсам usbmon1, usbmon2 и т.д.

-c <число пакетов> – указывает завершить программу по достижению заданного числа перехваченных кадров;

-n – отключает преобразование адресов в символьные имена;

-nn – отключает преобразование номеров портов в символьные имена;

-X – задаёт вывод дампа в шестнадцатеричном и ASCII-формате без заголовков канального уровня. Эта опция может быть очень удобна при анализе новых протоколов, а также для просмотра передаваемых данных в рамках данной работы;

-s xxx – указывает производить просмотр первых xxx байт от начала пакета (по умолчанию просматриваются лишь первые 68);

-w *имя_файла* – указывает записывать все перехваченные кадры в файл в сыром (raw) виде. Указанный файл впоследствии можно просмотреть с использованием ключа -r или передать сторонней программе, например Wireshark Network Analyzer.

Примеры запуска:

Просмотр всего трафика проходящего через интерфейс lo.

```
# tcpdump -i lo
```

Просмотр всего TCP и UDP трафика приходящего на сетевой интерфейс eth0 из сети 192.168.1.0/24, в котором порт получателя равен 80 или 1080.

```
# tcpdump 'src net 192.168.1.0/24 and (dst port 80 or 1080)'
```

Просматривать UDP трафик идущий на порты 1234 и 1235 от хоста 192.168.1.3 через интерфейс eth0.

```
# tcpdump -i eth0 'src 192.168.1.3 and udp and (dst port 1234 or 1235)'
```

Просматривать содержимое UDP дейтаграмм имеющих порт получателя 1234 на интерфейсе eth0.

```
# tcpdump -X -s 1000 -i eth0 'udp and (dst port 1234)'
```

Замечание. Для ОС Windows доступна бесплатная портированная версия tcpdump – WinDump [8].

TShark

Программа tcpdump имеет довольно простой синтаксис написания правил и фактически является своего рода «неустаревающей классикой» среди снифферов. В большинстве случаев она продолжает удовлетворять потребности администраторов, и в ближайшее десятилетие также будет это делать. Однако, если программа не умеет делать какие-то новые «полезные вещи», то не примянёт появиться другая программа. Для целей данной работы TShark не даст каких либо существенных преимуществ по сравнению с tcpdump, тем более обе программы используются одну и ту же библиотеку libpcap, реализующую API pcap (packet capture), однако иные преимущества стоит упомянуть, поскольку указанная информация может пригодиться вам в будущем. Среди преимуществ TShark можно отметить умение работать с протоколом https, а также значительно более информативный вывод информации о полученных данных. Кроме того, tshark поддерживает огромное количество и других сетевых протоколов (более трёх сотен, что охватывает практически все когда-либо изобретённые виды сетей).

Среди полезных ключей для запуска программы можно выделить следующие:

-a условие_останова_перехвата – указывает условие при котором программа завершить своё выполнение; Возможно задание разных условий, например по длительности работы, если в качестве параметра в ключе указать «duration:10», то это укажет программе завершить свою работу через 10 секунд после её запуска.

Например, просматривание информации о приходе UDP дейтаграмм имеющих порт получателя 1234 и адрес отправителя 192.168.1.1 на интерфейсе eth0 в течение 30 секунд с момента запуска можно осуществить командой.

```
# tshark -i eth0 src host 192.168.1.1 and dst port 1234 and udp -a
duration:30
```

-b опции_кольцевой_записи – позволяет производить запись в несколько файлов.

Представьте ситуацию, что вам необходимо сохранять статистику трафика в течение длительного периода времени. Сохранить весь вывод в одном файле в такой ситуации заведомо не получится. Использование сторонних механизмов ротации файлов журналов возможно, но не очень удобно. Самое простое что можно сделать – это производить сохранение информации о трафике в нескольких файлах, количество и размер которых указываются пользователем. Как только один файл будет заполнен, tshark продолжит запись в следующий, можно указать чтобы, скажем запись циклически велась в 15 файлов размером по 100 КБ каждый.

```
# tshark -b filesize:100 -a files:15 -w traffic_dump.pcap
```

запись будет производиться в файлы с именами вида

```
traffic_dump_00001_20160317185736.pcap
traffic_dump_00001_20160317185802.pcap
traffic_dump_00002_20160317185806.pcap
...
```

-R фильтр_отображения – позволяет производить фильтрацию полученных данных по различным критериям (в том числе и на основе данных с уровнях приложений) и выводить лишь ту информацию, что соответствует заданному условию. Например, фильтр http.request будет отображать лишь приходящие HTTP. Совместно с другими параметрами можно рассматривать лишь запросы приходящие на 80 порт протокола TCP на интерфейсе eth1.

```
# tshark tcp dst port 80 and tcp -R http.request -i eth1
```

-V – сообщает TShark выводить подробную информацию, а не отчёты в одну строку на каждый фрейм (пакет, дейтаграмму);

-x – сообщает TShark также печатать содержимое (поле данных) из полученных пакетов (дейтаграмм, кадров) в шестнадцатеричном формате и в виде ASCII.

Wireshark

Wireshark (Wireshark Network Analyzer) – это достаточно известный инструмент для захвата и анализа сетевого трафика, аналогично tcpdump и tshark, но с графическим интерфейсом. (Ранее программа называлась Ethereal.)

Wireshark работает с подавляющим большинством известных протоколов, имеет понятный и логичный графический интерфейс на основе GTK+ и мощнейшую систему фильтров.

Кроссплатформенный, работает в таких ОС как Linux, Solaris, FreeBSD, NetBSD, OpenBSD, Mac OS X, и Windows.

Использование графического интерфейса интуитивно понятно. Для начала захвата достаточно в окне «Capture» найти в списке Interface List нужный сетевой интерфейс, например eth0, и нажать на него.

После чего и начнётся процесс захвата, причём «прилетевшие» на сетевой интерфейс пакеты (кадры, дейтаграммы) будут появляться в реальном времени, подобно получению писем электронной почты. Имеется возможность «щёлкнуть» на любую из строчек и открыть просмотр содержимого, подобно письму электронной почты, после чего в нижней половине будут показана структура заголовков, представленная деревом и побайтовое наполнение пакета, как в hex-редакторах.

netstat, ss

netstat – консольная утилита для просмотра списка установленных сетевых соединений, таблиц маршрутизации, статистики сетевых интерфейсов и другой полезной сетевой информации. На сегодняшний день считается устаревшей, а вместо неё рекомендуется использовать утилиту ss (из комплекта утилит iproute2). ss умеет показывать всю ту же информацию, что и netstat, но при этом она может показывать больше информации о состоянии и о TCP сокетах, нежели какие-либо другие утилиты.

Среди полезных ключей для запуска обеих программ можно выделить следующие:

-n – не преобразовывать имена хостов и номера портов в соответствующим им доменные имена и названия сервисов. Отображение в цифровом виде производится значительно быстрее.

-t – выводить информацию лишь по TCP сокетам. Обратите внимание, что прослушивающие сокеты по умолчанию не отображаются, для получения полной картины используйте совместно с ключом «-a».

-u – выводить информацию лишь по UDP сокетам. Аналогично, см. замечание про ключ «-a» выше.

-a – выводить информацию по всем сокетам (то есть вне зависимости от состояния сокета).

Примеры использования:

Показать информацию обо всех TCP сокетах (в том числе и слушающих).

```
$ netstat -t -a -n
$ ss -t -a -n
```

Показать информацию обо всех UDP сокетах (в том числе и слушающих).

```
$ netstat -u -a -n
$ ss -u -a -n
```

Отобразить все установленные TCP-соединения с web-сервером. (Соответствие названий сервисов их номерам согласно рекомендациям IANA смотрите в файле в /etc/services.)

```
$ ss -o state established '( dport = :http or sport = :http )'
$ ss -o state established '( dport = :80 or sport = :80 )'
```

Отобразить все TCP-сокеты с адресом клиента из подсети 192.168.1.0/24

```
$ ss -t -a src 192.168.1.0/24
```

lsof

Отображает список открытых файлов и псевдофайлов, в том числе и сокетов, как локальных, так и протоколов TCP и UDP.

Примеры использования:

Показать все TCP и UDP сокеты

```
$ lsof -i
```

Показать все TCP и UDP сокеты, связанные с адресом 192.168.1.5.

```
$ lsof -i@192.168.1.5
```

Тоже самое, но при отображении не преобразовывать адреса хостов и номера портов в доменные имена и названия сервисов.

```
$ lsof -i@192.168.1.5 -n -P
```

Показать все TCP сокеты; при отображении не преобразовывать адреса хостов и номера портов.

```
$ lsof -i TCP -n -P
```

Показать все UDP сокеты, связанные с адресом 192.168.1.5; при отображении не преобразовывать адреса хостов и номера портов.

```
$ lsof -i UDP@192.168.1.5 -n -P
```

iptables

Программа iptables используется для взаимодействия суперпользователя со структурой и кодом ядра «netfilter». Фактически используется для управления правилами и цепочками правил, определяющими режимы фильтрации сетевого трафика. В рамках данной работы данная утилита интересна тем, что позволяет просматривать счётчики имеющиеся у всех правил и цепочек, то есть получать статистическую информацию о количестве переданных (принятых) байт и пакетов (дейтаграмм).

Для просмотра статистики используется следующий синтаксис вызова:

```
# iptables -L -v -x -n --line-numbers
```

Тоже самое, но только для цепочки INPUT:

```
# iptables -L INPUT -v -x -n --line-numbers
```

Назначение ключей:

-L – отобразить статистическую информацию;

-v – отобразить дополнительную информацию, в том числе статистику;

-x – отображать значения счётчиков с точностью до байта;

-n – не преобразовывать при отображении адреса и номера портов в доменные имена и названия сервисов;

--line-numbers – перед каждым правилом выводить его порядковый номер в цепочке;

Для обнуления значений счётчиков используется ключ -Z:

```
# iptables -Z
```

Возможно совмещение с ключом -L для просмотра статистики с последующим обнулением.

Также, из полезных функций iptables для исследования сокетов можно отметить возможность создания фильтрующих правил без указания действия (то есть без «-j действие»), такие правила могут использоваться для ведения статистического учёта.

Например, следующая команда добавит в начало цепочки INPUT правило для подсчёта пакетов протокола TCP во входном трафике, идущих с адреса 192.168.1.4.

```
# iptables -I INPUT -p tcp -s 192.168.1.4
```

Для удаления правил используется ключ -D:

```
# iptables -D INPUT -p tcp -s 192.168.1.4
```

или

```
# iptables -D INPUT номер_правила
```

При использовании модулей TCP и UDP (параметры «-p tcp» и «-p udp»), возможно дополнительно указание портов отправителя и получателя, через ключи (--source-port XX, --destination-port XX, либо --sport XX и --dport XX).

ps

Используется для просмотра списка запущенных процессов в системе. Для просмотра всех процессов в системе часто используется в следующих формах запуска:

```
$ ps aux
```

и

```
$ ps ax
```

Данная команда не отображает никакой информации о сокетах. В рамках данной работы приведена лишь с целью более полного представления происходящих в операционной системе процессов.

Может быть полезна для получения дополнительной информации. Заметим, что утилита lsof в столбце «PID» отображает идентификаторы процессов, работающих с файлами и псевдофайлами (в нашем случае с сокетами).

web-сервер apache

Веб сервер apache в данной работе нужен лишь для придания «веса» процессу исследований, как солидное программное средство, работающее с сокетами. Задумка заключается в том, чтобы пользователь смог понять и «прочувствовать», что только что им написанная и скомпилированная программа на языке Си способна выступать в роли клиента и получать данные от web-сервера. В случае наличия постоянного прямого подключения к интернету возможно использование любых известных пользователю и работающих web-сайтов с этой целью. («Прямое подключение» означает возможность обращения к сайтам на 80-й порт по протоколу TCP без использования прокси-серверов для этой цели. Использование нелегитимных адресов (согласно RFC 1918) и/или трансляции адресов для осуществления подключения не имеет значения.)

Запуск web-сервера apache в режиме сервиса:

```
# service httpd start
```

Для проверки работы исследования достаточно и тестовой страницы.

По умолчанию в качестве корневой директории web-сервера (то, куда помещать html, php и другие файлы) используется /var/www/html.

Замечание. При включённом SELinux и отсутствии разрешающих доменов обратите внимание на используемый для файлов и директорий контекст безопасности. (system_u:object_r:httpd_sys_content_t:s0)

программы на Си

Ниже приведены простые программы на Си реализующие клиентское и серверное использование TCP и UDP сокетов. В угоду краткости и понятности (важные требования чтобы не отпугнуть начинающих и сомневающихся в своих силах программистов) их простота заключается в том, что в них во многих привычных местах отсутствуют проверки получаемых статусов соединений, учёта переданных данных и проверки самого факта передачи данных. Несомненно, в реальных условиях эксплуатации подобных программ не избежать различных доработок, а там, глядишь и... «велика беда начало», и... «у совершенства нет предела» и... «лучшее враг хорошего». Чувствуете свои силы и имеет под рукой более сложный код? – Дерзайте! Всем же остальным, кто сомневается в своих силах ещё на старте, ниже даны примеры простого и работоспособного кода.

tcp_client1.c – простой TCP клиент

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void main(void)
{
  int s;
  char buf[256]="Привет! Hello!\n";
  struct sockaddr_in serv_addr;
  bzero(&serv_addr, sizeof(serv_addr));
  serv_addr.sin_family=AF_INET;
  serv_addr.sin_addr.s_addr=inet_addr("127.0.0.1");
  serv_addr.sin_port=htons((u_short) 1234);

  s=socket(PF_INET, SOCK_STREAM, 0);
  connect(s, (struct sockaddr *)&serv_addr, sizeof(serv_addr));
  send(s, buf, strlen(buf), 0);
  close(s);
}
```

tcp_client2.c – по сути, тот же клиент что tcp_client1.c, но после отправки своей порции данных на сервер он получает от него ответ и выводит его на консоль (не более 255 байт, большее будет утрачено), для этого в коде перед закрытием сокета (строчка close(s);) следует дописать:

```
  int nbytes;
  nbytes=recv(s, buf, sizeof(buf)-1, 0);
  if (nbytes>0) { buf[nbytes]='\0';  printf("%s",buf); }
```

tcp_client3.c – в коде tcp_client2.c самостоятельно поменяйте местами порядок получения и отправки данных. На практике существуют TCP сервера, выдающие приветственное приглашение, прежде чем что-то получать от клиента. Что-то вроде жизненного случая когда... вы определились с мыслями куда пойти («выбрали заведение», по аналогии с «создали сокет»), вошли через двери заведения (подключились к сокету), ещё не успели подумать, что сказать (отправить серверу), а вы уже слышите в свой адрес: «Свободная касса!» (получили приглашение от сервера), либо более грубое, но чёткое, «Чего надо?!».

Замечание. Посылаемые в программах сообщения специально выбраны содержащими как кириллицу, так и латиницу. Сделано это по причине наиболее вероятного использования кодировки Unicode и способе кодирования UTF-8 для записи текстового сообщения. Поскольку символы кириллицы в этом случае кодируется двумя байтами, а латиницы – одним (с использованием ASCII), то при просмотре сырых данных из сокета, полученных с помощью сниффера, требуется изрядное искусство, чтобы разглядеть великий и могучий русский язык в сообщениях. А вот текстовые строки закодированные с помощью ASCII увидят все и сразу. А потом, если захотят, смогут раскодировать и оставшуюся часть сообщения. Как это сделать см. 2.2.7. Работа с текстовыми данными на практике (стр. 105).

udp_client1.c – в коде tcp_client1.c самостоятельно замените в строке

```
  s=socket(PF_INET, SOCK_STREAM, 0);
```

тип сокета с SOCK_STREAM на SOCK_DGRAM.

udp_server1.c – простой UDP сервер

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <arpa/inet.h>
#include <sys/socket.h>

int main(void)
{
    struct sockaddr_in serv_addr, clnt_addr;
    int s, i, slen = sizeof(clnt_addr), nbytes;
    char buf[256];

    if ((s=socket(PF_INET, SOCK_DGRAM, 0)) == -1)
    {
      perror("Ошибка вызова socket()");
      exit(1);
    }

    /* Очистка памяти структуры serv_addr. */
    memset((char *) &serv_addr, 0, sizeof(serv_addr));

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(1234);
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    /* Привязка сокета к прослушиваемому порту и адресу. */
    if( bind(s , (struct sockaddr*)&serv_addr, sizeof(serv_addr) ) == -1)
    {
      perror("Ошибка вызова bind()");
      exit(2);
```

```
    }
    /* Принимаем входящие соединения. */
    while(1)
    {
        printf("Ждём подключения.\n");
        fflush(stdout);

        if ((nbytes = recvfrom(s, buf, sizeof(buf)-1, 0, (struct sockaddr *)
&clnt_addr, &slen)) == -1)
        {
          perror("Ошибка вызова recvfrom()");
          exit(3);
        }
        buf[nbytes]='\0';
        printf("Дейтаграмма от %s:%d\n", inet_ntoa(clnt_addr.sin_addr),
ntohs(clnt_addr.sin_port));
        printf("Её содержимое: %s\n", buf);
    }
    close(s);
    return 0;
}
```

tcp_server1.c – простой TCP сервер обрабатывающий одновременно одно соединение

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <netdb.h>
#include <string.h>

#define PORTNUM 1234 // номер порта для приёма входящих соединений сервером

int main(void)
{

  int listen_s, connected_s, nbytes;
  struct sockaddr_in serv_addr, clnt_addr;
  char buf[256]; // буффер для приёма сообщения
  unsigned long addr;

 if ( (listen_s=socket(PF_INET, SOCK_STREAM, 0) ) == -1 )
    {
      perror("Ошибка вызова socket()");
      exit(1);
    }

  /* Заполняем, предварительно обнулив, структуру serv_addr для описания
"своего" конца коммуникационного канала: указываем семейство адресов AF_INET
(то есть, IP-адреса); входящий адрес сокета INADDR_ANY (0.0.0.0) означает
принимать соединения на всех имеющихся у хоста адресах. */
  bzero(&serv_addr, sizeof(serv_addr));
  serv_addr.sin_family=AF_INET;
  serv_addr.sin_addr.s_addr=INADDR_ANY;
  serv_addr.sin_port=htons((u_short)PORTNUM);
```

```
  /* Связываем сокет listen_s с адресом и портом, содержащимися в serv_addr,
     определяя локальную часть коммуникационного канала. */
  if ( bind(listen_s, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) == -1 )
   {
      perror("Ошибка вызова bind()");
      exit(2);
    }
  /* Переводим сокет в режим ожидания соединений - фактически в модуль TCP
передаётся вызов PASSIVE_OPEN. Второй параметр задаёт максимальный размер,
до которого может расти очередь ожидающих соединений у listen_s. */
  if ( listen(listen_s,10)==-1 )
   {
      perror("Ошибка вызова listen()");
      exit(3);
    }

  printf("Сервер готов принимать соединения\n");

    /* Обнуляем структуру clnt_addr, в которую будут записаны адрес и порт
подсоединившегося клиента. */
    int addrlen;
    bzero(&clnt_addr, sizeof(clnt_addr));
    addrlen=sizeof(clnt_addr);

    /* Принимаем запрос. Возврат из функции accept() производится только
после поступления запроса и установления соединения с клиентом. При этом на
базе старого сокета listen_s создаётся новый сокет connected_s, у которого
определены уже обе части соединения - локальная и удалённая. Таким образом,
сокет connected_s уже может использоваться для передачи данных. Адрес и порт
подсоединившегося клиента возвращаются в структуре clnt_addr. */

    if ( (connected_s=accept(listen_s, (struct sockaddr *)&clnt_addr, &ad-
drlen) ) == -1 )
    {
   perror("Ошибка вызова accept()");
   exit(4);
    }

    printf("Соединение от %s\n", inet_ntoa(clnt_addr.sin_addr));

    nbytes=recv(connected_s, buf, sizeof(buf)-1, 0);
    if (nbytes>0)
    {
       buf[nbytes]='\0';
       printf("Получено сообщение: %s\n", buf);
    }

    printf("Завершаем работу, закрываем сокет.\n");
    close(connected_s);
    close(listen_s);
    exit(0);
}
```

Самостоятельная доработка программ

Программы выше заведомо упрощены. Кроме облегчения процесса написания работы это открывает широкие возможности для творчества обучаемых, формируя у них критический взгляд на исследуемые процесс. Возникает возможность самостоятельно-

го исследования, внесения изменений в представленный код, экспериментирования. Если вы не увидели то, что можно улучшить в программах, приведём некоторые очевидные варианты и заготовки в помощь.

Программа реализующая серверную часть TCP-сокета способна принимать лишь одно входное соединение одновременно. На практике же практикуются множественные клиентские подключения одновременно. Такую возможность программе можно добавить за счёт использования функции fork(). Она создаёт дубликат процесса; возвращая в родительский процесс идентификатор порождённого дочернего процесса, а в дочерний – ноль. Таким образом, её использование позволит одновременно делать две вещи: работать с установленным соединением через сокет connected_s, а дубликату процесса продолжать принимать новые соединения через старый сокет listen_s.

Также программа принимает лишь одно сообщение от клиента и завершает работу. На практике можно реализовать приём нескольких сообщений циклом.

На сервере или на клиенте может иметься возможность автоответа, то есть автоматическое отправление противоположной стороне сообщения в ответ на приём какого-то ключевого слова или команды.

Задания на лабораторную работу (для пошагового выполнения)

1. Войдите в систему.

2. Запустите три терминальных окна: для клиента, для сервера и для исследования.

3. Запустите прослушивать сокет 1234 протокола TCP на серверной стороне.

```
$ nc -l 1234
```

4. В окне для исследования просмотрите занятость портов для протокола TCP

```
$ netstat -t -a -n
...
tcp        0      0 0.0.0.0:1234            0.0.0.0:*               LISTEN
...
$ ss -t -a -n
..
LISTEN     0      1                         *:1234                  *:*
...
```

и найдите в выводе строку, отвечающую за ваш прослушивающий сокет.

5. Запустите последовательно команды

```
$ lsof
...
$ lsof -i
...
$ lsof -i TCP -n -P
COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
nc      2691 user    3u  IPv4  20425      0t0  TCP *:1234 (LISTEN)
```

Сравните вывод и найдите PID процесса, ожидающего подключения на порту 1234 протокола TCP.

6. Запустите команду

```
$ ps aux
```

Найдите среди списка всех процессов тот, что отвечает за работу серверного сокета.

7. Запустите снифер.

```
$ su -
```
(введите пароль для повышения своих прав, выход из этого режима – exit)
```
# tcpdump -i lo -n
```
Он начнёт перехватывать данный на интерфейсе lo.

8. Перейдите в окно клиента и подключитесь к серверу.
```
$ nc 127.0.0.1 1234
```
обратите внимание на то, что после подключения в окне диагностики сниффер покажет информацию о прохождении нескольких пакетов
```
09:51:01.490961 IP 127.0.0.1.54908 > 127.0.0.1.search-agent: Flags [S], seq
1468622890, win 65495, options [mss 65495,sackOK,TS val 1444852 ecr
0,nop,wscale 7], length 0
09:51:01.490982 IP 127.0.0.1.search-agent > 127.0.0.1.54908: Flags [S.], seq
2911204310, ack 1468622891, win 65483, options [mss 65495,sackOK,TS val
1444852 ecr 1444852,nop,wscale 7], length 0
09:51:01.490994 IP 127.0.0.1.54908 > 127.0.0.1.search-agent: Flags [.], ack
1, win 512, options [nop,nop,TS val 1444852 ecr 1444852], length 0
```
Это клиент и сервер установили соединение.

9. Наберите на стороне клиента «Привет сервер! Hello server!» и отправьте его на сервер, нажатием клавиши {enter}. Сниффер покажет перехват ещё одного пакета. В окне сервера появился отправленное сообщение.
```
09:53:24.397337 IP 127.0.0.1.54908 > 127.0.0.1.search-agent: Flags [P.], seq
1:42, ack 1, win 512, options [nop,nop,TS val 1587758 ecr 1444852], length
41
09:53:24.397564 IP 127.0.0.1.search-agent > 127.0.0.1.54908: Flags [.], ack
42, win 512, options [nop,nop,TS val 1587758 ecr 1587758], length 0
```

10. Выполните такую же операцию на стороне сервера, отправив клиенту сообщение «Привет клиент! Hello client!». Клиент получит сообщение, а сниффер зафиксирует факт передачи данных.
```
09:54:37.185271 IP 127.0.0.1.search-agent > 127.0.0.1.54908: Flags [P.], seq
1:42, ack 42, win 512, options [nop,nop,TS val 1660546 ecr 1587758], length
41
09:54:37.185317 IP 127.0.0.1.54908 > 127.0.0.1.search-agent: Flags [.], ack
42, win 512, options [nop,nop,TS val 1660546 ecr 1660546], length 0
```

11. Прервите работу сниффера нажатием CTRL+C.

Что означает "127.0.0.1.search-agent"?

Посмотрите файл /etc/services и вспомните что у tcpdump есть также ключ -nn, используйте его в следующий раз.

12. Ответьте на вопросы: Сколько пакетов было передано в результате информационного обмена клиента с сервером? Сколько пакетов было от клиента к серверу? Сколько пакетов было в обратном направлении? Какого размера (параметр length) были пакеты?

13. В окне для диагностики выполните запуск команд
```
$ lsof -i TCP -n -P
$ lsof -i TCP -n -P|grep 1234
$ ss -t -a -n
$ ss -t -a -n|grep 1234
$ netstat -t -a -n
$ netstat -t -a -n|grep 1234
```

Оцените, что изменилось, по сравнению с выводом при выполнении команд из шагов 4 и 5?

Поскольку вы видите статистику одновременно и для клиента и для сервера, то вы будете видеть сокет клиента в режиме установленного соединения (ESTABLISHED), сокет сервера в режиме установленного соединения (ESTABLISHED), а также сокет клиента в режиме прослушивания (LISTEN), поскольку команда nc может принять ещё одно соединение.

14. Если у серверной части nc имеется слушающий сокет, то сможет ли к нему ещё подключиться ещё один nc-клиент и передать сообщение? Запустите ещё одно окно терминала и выполните команду по запуску второго клиента.

```
$ nc 127.0.0.1 1234
```

Отправьте с обоих клиентов и с сервера сообщения и проследите кто их получит.

Может создаться впечатление, что соединение у второго клиента есть, но данные не передаются.

Посмотрите список открытых файлов и сравните его с п.13.

```
$ lsof -i TCP -n -P|grep 1234
```

Что поменялось? Обратите внимание на номера процессов (вторая колонка - PID) в выводе.

Ответьте, является ли соединение второго клиента полноценно установленным?

15. Прервите выполнение серверов и клиентов, нажимая CTRL+C.

16. Запустите сниффер и сервер повторно, но с другими ключами

```
# tcpdump -i lo -n -nn -X -s 200
$ nc -l 1234 -v
```

17. Подключитесь клиентом. Повторите посылку сообщений: "Привет сервер! Hello server!" и "Привет клиент! Hello client!". Видите ли вы среди перехваченных данных ваше сообщение? Видите ли вы байты, отвечающие за кириллицу?

```
10:33:08.363361 IP 127.0.0.1.54920 > 127.0.0.1.1234: Flags [P.], seq 1:42,
ack 1, win 512, options [nop,nop,TS val 3971724 ecr 3949456], length 41
   0x0000:  4500 005d a1ad 4000 4006 9aeb 7f00 0001  E..]..@.@.......
   0x0010:  7f00 0001 d688 04d2 1b2f e225 7605 3290  ........./.%v.2.
   0x0020:  8018 0200 fe51 0000 0101 080a 003c 9a8c  .....Q.......<..
   0x0030:  003c 4390 d09f d180 d0b8 d0b2 d0b5 d182  .<C.............
   0x0040:  20d1 81d0 b5d1 80d0 b2d0 b5d1 8021 2048  .............!.H
   0x0050:  656c 6c6f 2073 6572 7665 7221 0a         ello.server!.
```

18. Оставьте сниффер включённым. Прервите выполнение клиента и сервера. Запустите сервер повторно. Запустите браузер Firefox. Возможно, вы увидите с помощью сниффера, что браузер уже во время своего запуска сгенерировал достаточно трафика. Сервер будет в это время "молчать", ожидая входящие соединения.

19. Обратитесь к серверу через браузер Firefox, введя в адресной строке «http://127.0.0.1:1234».

После сообщения от nc о подключении

```
Connection from 127.0.0.1 port 1234 [tcp/search-agent] accepted
```

вы увидите непосредственно HTTP-запрос браузера.

```
GET / HTTP/1.1
Host: 127.0.0.1:1234
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:38.0) Gecko/20100101
Firefox/38.0
```

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Далее, думая, что он подключился к полноценному web-серверу, браузер будет ожидать ответ.

Вероятнее всего, он захочет получить в ответ что-то вида «Content-Type: text/html» и два символа перевода строки – типичный HTTP-заголовок ответа сервера, а затем и запрошенный корневой файл с сервера – «/». Вы можете игнорировать эти правила обмена, ввести любой текст и закрыть соединение на стороне сервера, нажатием CTRL+c.

После этого браузер отобразит полученный от сервера текст.

20. Запустите сервер заново и обратитесь к нему с помощью консольного браузера lynx.

```
$ lynx 127.0.0.1:1234
```

Обратите внимание, что запрос, передаваемый серверу несколько отличается.

```
GET / HTTP/1.0
Host: 127.0.0.1:1234
Accept: text/html, text/plain, text/css, text/sgml, */*;q=0.01
Accept-Encoding: gzip, bzip2
Accept-Language: en
User-Agent: Lynx/2.8.6rel.5 libwww-FM/2.14 SSL-MM/1.4.1 OpenSSL/1.0.0-fips
```

21. Используйте для обращения wget

```
$ wget 127.0.0.1:1234
--2016-03-20 10:54:28--  http://127.0.0.1:1234/
Устанавливается соединение с 127.0.0.1:1234... соединение установлено.
Запрос HTTP послан, ожидается ответ...
```

на сервер придёт, отобразившись в окне со сниффером, следующий запрос

```
GET / HTTP/1.0
User-Agent: Wget/1.12 (linux-gnu)
Accept: */*
Host: 127.0.0.1:1234
Connection: Keep-Alive
```

22. Завершите выполнение сервера nc. Вспомните, что у вас на компьютере запущен собственный web-сервер на 80 порту.

23. Обратитесь с помощью браузеров Firefox, lynx и программы wget на локальный сервер 127.0.0.1. Изучите информационный обмен клиента и с сервера в окне со сниффером для всех трёх случаев.

Если вы надумаете обратиться к какому-либо серверу в вашей локальной сети или в интернете, то обратите внимание, что сниффер у вас запущен на интерфейсе lo, а обращаться в сеть вы будете через интерфейс eth0.

24. Смоделируйте с помощью утилит nc и telnet работу браузера, для этого обратитесь к серверу с HTTP GET запросом, передав ему строку «GET /» или «GET / HTTP/1.0». В ответ вы получите тестовую страницу. Сниффер покажет прошедшие через интерфейс данные.

```
$ telnet 127.0.0.1 80
Connected to 127.0.0.1.
Escape character is '^]'.
```

```
GET /
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
   <head>
...
</html>
Connection closed by foreign host.
$ nc 127.0.0.1 80
GET /
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
...
```

25. Запустите nc как сервер для прослушивание порта 1234 протокола TCP.

```
$ nc 1234 -l
```

26. Скомпилируйте программу tcp_client1.c

```
$ gcc tcp_client1.c -o tcp_client1
```

27. Запустите клиент протокола TCP.

```
$ ./tcp_client1
```

Дошли ли данные до сервера? Увидели ли вы их с помощью сниффера?
Почему сервер закончил прослушивать соединение?

28. Запустите сниффер tshark и повторите шаги 25 и 27.

```
# tshark -i lo -n -x
```

Удалось ли вам увидеть переданное сообщение среди информации перехваченной сниффером?

```
0.001907785    127.0.0.1 -> 127.0.0.1    TCP 87 54953 > 1234 [PSH, ACK]
Seq=1 Ack=1 Win=65536 Len=21 TSval=6717342 TSecr=6717340

0000  00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00   ..............E.
0010  00 49 fe 1e 40 00 40 06 3e 8e 7f 00 00 01 7f 00   .I..@.@.>.......
0020  00 01 d6 a9 04 d2 b7 af 81 81 35 b3 5e 85 80 18   ..........5.^...
0030  02 00 fe 3d 00 00 01 01 08 0a 00 66 7f 9e 00 66   ...=.......f...f
0040  7f 9c d0 9f d1 80 d0 b8 d0 b2 d0 b5 d1 82 21 20   ..............!
0050  48 65 6c 6c 6f 21 0a                              Hello!.
```

29. Повторите действия шага 28, добавив в строке запуска tshark ключ -V. Насколько больше полезной информации выдал сниффер?

30. Скомпилируйте TCP сервер.

```
$ gcc tcp_server1.c -o tcp_server1
```

31. Запустите TCP-сервер, подключитесь к нему с помощью nc и пошлите сообщение «Привет!».

```
$ ./tcp_server1
Сервер готов принимать соединения
Соединение от 127.0.0.1
Получено сообщение: Привет

Завершаем работу, закрываем сокет.
$ nc 127.0.0.1 1234
Привет
```

32. Проверьте работу TCP клиента и TCP сервера вместе.

33. Повторите шаг 32 с минимальной задержкой.

Если между шагами 32 и 33 прошло менее минуты, то не получили ли вы при попытке запуска сервера сообщение об ошибке?

```
Ошибка вызова bind(): Address already in use
```

Если да, то попытайтесь устранить ошибку, воспользовавшись советом в замечании ниже по устранению этой проблемы.

Замечание. Можно заметить, что довольно часто при повторном запуске программы использующей TCP-серверные сокеты выдаётся сообщение вида «Ошибка вызова bind(): Address already in use». Однако, через какое-то время та же самая программа запускается и работает без этого сообщения.

Подробно эта ситуация описана в [3]. Коротко, ситуация происходит потому, что сокеты, установившие соединение, всё ещё «висят» в ядре ОС, и оно «держит» порт занятым. Решение либо подождать (минуту или около того), пока ядро не освободит порт, либо, что более правильно, добавить в программу вызов функции setsockopt(), что позволит повторно использовать уже свободный порт без ожиданий. Для программы tcp_server1.c строки избавляющие пользователя от указанной ошибки и ожидания будут выглядеть следующим образом:

```
// избавляемся от ошибки bind(): Address already in use
int opt = 1;
if (setsockopt(listen_s,SOL_SOCKET,SO_REUSEADDR,&opt,sizeof (opt)) == -1)
{
  perror("Ошибка вызова setsockopt()");
  exit(5);
}
```

Добавьте вышеприведённый код в файл tcp_server1.c после вызова функции socket() перед вызовом функции bind().

34. Перезапустите сниффер, с параметрами

```
# tshark -i lo -n -x
```

Создайте UDP слушающий сокет на порту 1234.

```
$ nc -l 1234 -u
```

35. Исследуйте серверный сокет командами lsof, netstat и ss, повторив шаги 4,5,6. Запомните вывод команды «lsof -i -n -P».

36. Поключитесь с помощью nc как клиент к UDP серверу.

```
$ nc 127.0.0.1 1234 -u
```

Показал ли сниффер какой-то трафик во время запуска клиента, как это было при установлении TCP соединения?

37. Исследуйте открытые сокеты системы командами lsof, netstat и ss, повторив шаги 4,5,6. Как изменился после подключения вывод команды «lsof -i -n -P»? Видите ли вы разницу по сравнению с использованием TCP-сокетов?

38а. Пошлите сообщения от клиента серверу и обратно. Проанализируйте трафик, перехваченный сниффером. Как изменился вывод команды «lsof -i -n -P»?

38б. Пошлите сообщения от сервера клиенту и обратно. Проанализируйте трафик, перехваченный сниффером. Как изменился вывод команды «lsof -i -n -P»?

38в. В чём разница при выполнении между ветками 34-35-36-37-38а и 34-35-36-37-38б?

Как изменился после подключения вывод команды «lsof -i -n -P»?

39. Завершите клиентский nc, нажатием CTRL+c.

40. Запустите клиента повторно и отправьте сообщение серверу.

```
$ nc 127.0.0.1 1234 -u
```

Получил ли сервер сообщение? Был ли сетевой трафик (что зафиксировал сниффер)? Не выдал ли клиент похожую ошибку?

```
nc: Write error: Connection refused
```

Почему так произошло?

41. Проанализируйте ситуацию возникшую на шаге 40 с помощью lsof, netstat и ss, повторив шаги 4,5,6.

Возможно ли из вывода этих программ понять почему нет обмена трафиком между клиентским и серверным приложениями?

Как решить проблему, чтобы была связь между сервером и клиентом?

42. Скомпилируйте UDP клиент.

```
$ gcc udp_client1.c -o udp_client1
```

43. Проверьте работу клиента с помощью сервера на nc. Для этого заново запустите UDP сервер

```
$ nc -l 1234 -u
```

а после этого запустите UDP клиент.

```
$ ./udp_client1
```

Была ли осуществлена передача данных клиентом? Увидел ли передачу данных сниффер? Получил и отобразил ли полученные данные сервер?

44. Обратите внимание, что сервер nc не завершился и продолжает ожидать следующие порции данных. Это явно можно увидеть используя lsof.

45. Запустите повторно UDP клиент.

```
$ ./udp_client1
```

Была ли осуществлена передача данных клиентом? Увидел ли передачу данных сниффер? Получил и отобразил ли полученные данные сервер? Почему данные не дошли до сервера?

46. Скомпилируйте UDP сервер.

```
$ gcc udp_server1.c -o udp_server1
```

Запустите UDP сервер и UDP клиент. Передал ли клиент серверу сообщение?

```
$ ./udp_server1
$ ./udp_client1
```

47. Обратите внимание, что сервер продолжает ждать подключения. Запустите клиент ещё раз.

Была ли осуществлена передача данных клиентом? Увидел ли передачу данных сниффер? Получил и отобразил ли полученные данные сервер? Почему данные дошли до сервера? Почему не повторяется ситуация возникшая на шаге 43?

48. Поэкспериментируйте с кодом программ на Си сделать доработки предложенные ранее.

При наличии возможности подключите к экспериментам соседние компьютеры и их пользователей. Организуйте обмен трафиком по локальной сети.

Может ли UDP клиент соединиться с TCP сервером и передать данные? А наоборот? А если проверить экспериментально?

Если увеличить значение размера массива buf в использовавшихся программах на Си и перекомпилировать их, то какого максимального размера можно осуществить передачу данных на практике? Какие будут ограничения и чем они будут вызваны?

49. Увеличиваются ли значения счётчиков iptables? Создайте правила и посмотрите, считается ли с помощью них проходящий между клиентом и сервером трафик?

```
# iptables -L -v -x -n
# iptables -I INPUT -p udp -s 127.0.0.1
```

50. Запустите сниффер WireShark Network Anlyzer и самостоятельно проведите несколько экспериментов.

Чем удобнее пользоваться графической средой или консолью?

Заключение

В результате выполнения заданий работы и подготовки лабораторного стенда вы должны были лучше понять принципы работы сокетов, повысить свои навыки работы с консольными приложениями.

Список литературы и ссылок к лабораторной работе № 4

1. *Стахов В.* Программирование сокетов // Системный администратор № 1, с.78-82, 2002. http://samag.ru/archive/article/33.

2. UNIX: разработка сетевых приложений /У. Стивенс. – СПб.:Птиер, 2003. – 1088 с. ISBN 5-318-00535-7.

3. Описание ошибки «Bind: Address Already in Use» http://hea-www.harvard.edu/~fine/Tech/addrinuse.html.

4. PuTTY: a free SSH and Telnet client http://www.chiark.greenend.org.uk/~sgtatham/putty/.

5. man для программы nc (русский язык) http://bsdadmin.ru/index.php/mans-freebsd/295-nc-netcat.

6. Сайт для проектов Mozilla http://www.mozilla.org.

7. Portable ascii art GFX library AAlib. http://aa-project.sourceforge.net/aalib/.

8. Сайт библиотеки WinPcap и программы WinDump http://www.winpcap.org/.

Алфавитный указатель

2-й закон Ньютона.....187
алгоритм….4, 14, 19, 24, 32, 35-41, 53- 55, 58, 60, 65, 67, 80, 95, 96, 97, 100, 113, 114, 121, 134, 136, 138, 150, 152, 153, 155, 156, 157, 158, 169, 174, 182, 200, 201, 203, 210, 220, 221-227, 232, 238, 251, 266, 267, 294, 299-302, 310, 312, 316, 378, 420, 425, 427, 435, 460, 474, 487, 491, 516, 517, 520, 525, 526, 537, 571, 575, 606, 638, 662, 667, 669, 680, 683-687, 700, 701
алгоритм Хаффмана.....153
АЛУ.....70, 229
архитектура фон Неймана.....228
ассемблер.....229
АЦП.....113, 115
блог.....666
булева алгебра.....205
веб-форум.....665
вебинар.....666
Винчестер.....271
влог.....666
геотегирование.....589
гипервизор.....492
ГЛОНАСС.....589
ГОСТ 8.417–2002.....13
гостевая ОС.....493
Джордж Буль.....204
дизъюнкция.....207
Дополнительный код.....36, 37, 38, 40, 158
ИБП.....263, 369-373
имплʲкацияʲ.....207, 208
инверсия.....41
интернет вещей.....584
интерфейс.....605
информация.….7-10, 14, 22, 23, 110, 113, 241, 255, 277, 647, 664, 665, 677
квантование.....113
Клод Шеннон.....14
количество ядер.....251
контейнер MKV.....137
конъюнкция.....207
кэш-память.....241, 243, 251, 301
лампа октальная.....187
лампа пальчиковая.....187
лампа электровакуумная.....189
логический элемент.....211, 212, 213
мантисса. 54, 55, 56, 62, 66, 70, 73, 78
Машина Поста.....226
машина Тьюринга.....225
маши́на Тью́ринга.....19, 225
машинное слово.....28, 49
мессенджер.....658
микрокод.....233, 234
микросхема.....196
модуль TPM.....598
Н. Винер.....3
ОЗУ.....219
пиксель.....125
пленоптика.....140, 158
показатель степени.....54, 61, 78
полупроводни́к.....191
прокси-сервер.....518, 524, 641
протокол.....604
процесс.....30, 45, 228, 407
прямой код.....35, 37
развёртка.....339
Разрежённый файл.....480
разрешение.....126
растр.....125
режим работы процессора.....251
реле.....186
руткит.....519
сабпиксель.....340
серийный номер.....280
Сетунь.....23, 27, 228
сложение по модулю 2.....36
СНПЧ.....348
сокет.....407, 410, 462, 578, 711, 713, 719, 729-732, 734, 736, 737
соленоид.....186
стрелка Пирса.....207
стример.....317
субтрактивная CMYK-модель.....129
тактовая частота.....250

термомагнитная запись.....286
тест Тьюринга.....597
тонер.....343
транзистор……3, 187, 189, 190, 191, 193, 194, 195, 196, 212, 213, 214, 217, 232, 238, 240, 241, 242, 243, 250, 267, 289, 290, 291, 292, 294, 340, 374
треугольник Серпинского.....134
трёхмерная графика.....132
триггер.188, 203, 211, 214, 215, 216, 217, 218, 266, 267, 376
троичная логика.....175
Тьюринг.....226
фон Нейман.....87, 227, 228
форма Бэкуса-Наура..... БНФ.....401
формат с плавающей запятой.....48
фотобарабан.....342
фрактал.....134
функция Пирса.....211
функция Шеффера.....211
хостовая ОС.....492
хэштег.....656
цветовая модель.....128
штрихкод.....145, 146, 147, 149, 150, 158, 329, 586, 588
экстент.....405
Эльбрус.....236, 237, 238
ЭМВОС.....410, 692
энтропия.....14, 18
ядро́ ОС.....386
ARM.....156, 235, 265
ASCII.....88, 108
Big Data.....662
binary32.....53, 55, 56, 61-65, 69, 72, 74, 75, 77
BIOS.....262
bitcoin.....681
BOM-символ.....104
BSOD.....488
Burrows-Wheeler transform.....155
BWT.....155
CAPTCHA.....597
Carier IQ.....590
CD-ROM.……305, 306, 307, 308, 310, 314, 358
CIDR.....631
CISC.....234, 235
CUDA.....236, 335
DHCP.....640
DLP-проектор.....354
DNS-сервер.....626
DOS.91, 109, 386, 387, 388, 389, 402, 403, 404, 449, 450, 451, 455, 492, 499, 516, 519
DSL-модем.....636
EFI.....262
EXIF.....127
FAT. .385, 386, 448, 449, 450, 451, 455
FAT 16.....450
FB2.....534
FictionBook.....534
file.....405
Firefox OS.....395
FLAC.....115
FreeBSD 265, 300, 430, 431, 432, 451, 493, 496, 669
gnuplot.....508
GPT.....482
HDD.....271
HPFS.....451
hub.....359
ICQ.....658
IEEE 754.…47, 49, 53, 56, 60, 74, 159, 160
IEEE 754-2008.....49, 55, 56
IEEE 854-1987.....49
iPhoneTracker.....589
iptables. .706, 712, 713, 715, 717, 719, 726, 727, 739
jailbreak.....493, 589
John Backus.....401
kernel panic.....488
KOI8-R.....91
LCD-проектор.....354
Linux.....91, 108, 159, 252, 265, 279
M9-IX.....621
MAC-адрес.....360, 364
mantissa.....54
MBR.....482, 484
microcode.....234
NaN.....49
NTFS.....451, 453
Peter Naur.....401

pixel.....125, 127
POP3.....652
POSIX.....390, 402, 403, 423, 452
QR-код.....5, 146, 147, 148
RAID.....278, 281, 283, 299
Raspberry Pi®.....265
resolution.....126
RFID.....148, 586
RGB-модель.....129
RISC.....234
router.....361
RSS-каналы.....655
S.M.A.R.T.....278
Serial ATA.....262
Serial Attached SCSI.....263
sign and magnitude.....36
significand.....54, 56
SIP.....668
SMTP.....653
socket.....253, 407, 410, 655, 685, 701, 711, 721, 728, 729, 730, 737
SPARC.....237
SSD.....287, 301, 302, 303
subnormal numbers.....54, 64, 67
switch.....359
Tor.....596
Twitter.....656
Ubuntu Touch.....395
UEFI.....262
UHD.....126
UNICODE.....108
UNIX.....91, 402
URL.....627
UTF-8.....108
Variable Length Subnet Mask.....630
Windows.....109, 128, 156, 345, 403
XOR.....24

Содержание

Введение....3
Глава 1. Общие сведения об информационных процессах....6
1.1. Понятие информации, её виды и свойства....6
1.2. Свойства информации....9
1.3. Характеристики информации....12
1.3.1. Формальные единицы измерения информации в технике....12
1.3.2. Обозначение одного байта по ГОСТ 8.417–2002....13
1.3.3. Методы и модели оценки количества информации....13
1.3.3.1. Объёмный метод....14
1.3.3.2. Энтропийный метод (информация как снятая неопределённость)....14
1.3.3.3. Алгоритмический метод....19
1.3.4. Семантическая (смысловая) мера информации....19
1.3.5. Прагматическая мера информации (мера полезности)....21
1.4. Заключение....22
1.5. Контрольные вопросы к главе 1....24
1.6. Литература к главе 1....25

Глава 2. Представление информации в компьютере....26
2.1. Основные теоретические сведения представления чисел в ЭВМ....27
2.1.1. Положительные целые числа (машинное представление без знаковых целых чисел)....28
2.1.1.1. Зачем столько двоичных форматов? 29
2.1.2. Основы кодирования, представления и записи целых чисел....31
2.1.2.1. Прямой код....34
2.1.3. Отрицательные целые числа (машинное представление любых целых, как со знаком, так и без)....35
2.1.3.1. Дополнительный код....36
2.1.3.2. Алгоритмы вычисления дополнительного кода....39
2.1.3.3. Ошибки «переполнения» целых....44
2.1.4. Целые и не целые (числа с плавающей точкой/запятой)....48
2.1.4.1. Кратко о стандартах IEEE 754-2008 и IEEE 854-1987....48
2.1.4.2. Типы данных в языке C (для хранения действительных чисел)....49
2.1.4.3. Пример «36,6» (один частный случай)....52
2.1.4.3.1. Терминология....54
2.1.4.3.2. Алгоритм преобразования для одного числа из примера....54
2.1.4.4. Общий случай для действительных чисел: 10 диапазонов, 5 алгоритмов преобразований, 2 формулы!....58
2.1.4.4.1. Диапазон нормализованных чисел....61
2.1.4.4.2. Ноль (нуль)....63
2.1.4.4.3. Диапазон денормализованных чисел....64
2.1.4.4.4. Бесконечность (∞)....71
2.1.4.4.5. Не числа (NaN, Not a Number)....72
2.1.4.5. Интересные наблюдения....73
2.1.5. Ошибки представления (точность)....79

2.2. Представление текстовой информации в ЭВМ....87
2.2.1. Кодовые таблицы....87
2.2.1.1. Русификация....89
2.2.1.2. Кодовая таблица Unicode....92
2.2.2. Однобайтное и многобайтное кодирование текстов, кодировки переменной длины....94
2.2.3. Недостатки многобайтовых кодировок....95
2.2.3.1. Краткое сравнение UCS-2 и UTF-16....96
2.2.4. Кодирование символов таблицы Unicode в формате UTF-8....97
2.2.5. Транслитерация, ASCII art....99
2.2.6. Специальные символы....100
2.2.6.1. Знаки валют....101
2.2.6.2. Смайлики (значки, маленькие цветные картинки, иконки, эмодзи)....101
2.2.6.3. Специальный символ «BOM»....104
2.2.6.4. Символ(ы) перевода (конца) строки....104
2.2.6.5. Другие специальные символы....105
2.2.7. Работа с текстовыми данными на практике....105
2.2.7.1. Локализация в ОС Linux....107
2.2.7.2. Перекодирование текстов, ошибки....107
2.3. Кодирование звуковой и аналоговой информации....110
2.3.1. Терминология....114
2.3.2. Формат кодирования FLAC....115
2.3.3. Основы звукообработки....115
2.3.3.1. Обратный путь: из цифры в аналог....121
2.4. Кодирование графической и видеоинформации....123
2.4.1. Растровая графика....125
2.4.1.1. Стандарт EXIF хранения метаданных об изображении....127
2.4.1.2. Цветовая модель....128
2.4.2. Векторная графика....130
2.4.3. Трёхмерная графика (3D-графика)....132
2.4.4. Фрактальная графика....133
2.4.5. Представление видеоинформации в ПК....135
2.4.5.1. Цифровое вещание....138
2.4.5.2. 3D-изображение, 3D-видео....139
2.4.5.3. Пленоптика (вычисляемое видео)....140
2.5. Некоторые виды кодов и кодирования....144
2.5.1. Штрихкоды....145
2.5.1.1. QR-коды....146
2.5.2. Шрифт Брайля....150
2.5.3. Сжатие (архивация) различных видов информации....151
2.6. Контрольные вопросы к главе 2....157
2.7. Литература к главе 2....158

Глава 3. Законодательство РФ о защите компьютерной информации....161
3.1. УК РФ о преступлениях в сфере компьютерной информации....162
Статья 272. Неправомерный доступ к компьютерной информации....162

Статья 273. Создание, использование и распространение вредоносных компьютерных программ....163
Статья 274. Нарушение правил эксплуатации средств хранения, обработки или передачи компьютерной информации и информационно-телекоммуникационных сетей....163
Статья 274[1] Неправомерное воздействие на критическую информационную инфраструктуру Российской Федерации....164
3.2. ГК РФ об информации и её взаимосвязях с гражданами и ЭВМ....165
Статья 1225. Охраняемые результаты интеллектуальной деятельности и средства индивидуализации....165
Статья 1253.1. Особенности ответственности информационного посредника....165
Статья 1256. Действие исключительного права на произведения науки, литературы и искусства на территории Российской Федерации....166
Статья 1261. Программы для ЭВМ....166
Статья 1262. Государственная регистрация программ для ЭВМ и баз данных....166
Статья 1280. Право пользователя программы для ЭВМ и базы данных....166
Статья 1286.1. Открытая лицензия на использование произведения науки, литературы или искусства....167
3.3. Другие федеральные законы и подзаконные акты....168
ФЗ от 27 июля 2006 года № 149-ФЗ «Об информации, информационных технологиях и о защите информации»....168
ФЗ от 27.07.2006 г. № 152-ФЗ «О персональных данных»....169
ФЗ от 29.12.2010 г. № 436-ФЗ «О защите детей от информации, причиняющей вред их здоровью и развитию»....170
ФЗ от 02.07.2013 г. № 187-ФЗ «О внесении изменений в отдельные законодательные акты Российской Федерации по вопросам защиты интеллектуальных прав в информационно-телекоммуникационных сетях»....170
Распоряжение Правительства РФ от 31 декабря 2020 года № 3704-р....170
3.4. Контрольные вопросы к главе 3....173

Глава 4. Аппаратное обеспечение компьютеров....174
4.1. Введение....174
4.2. История механических вычислителей....175
4.3. Аналоговые компьютеры (аналоговые вычислители)....181
4.3. Поколения цифровых ЭВМ....185
4.4. Другие компьютеры (вычислители)....198
4.4.1. Квантовые компьютеры (квантовые вычислители)....198
4.5. Научные основы работы современной ЭВМ....204
4.5.1. Математические основы работы «чёрного ящика»....204
4.5.1.1. Какие они, сколько их?....205
4.5.1.2. Основные положения алгебры логики и вытекающие из этого интересные свойства....209
4.5.1.3. Способы описания работы «чёрного ящика»....210
4.5.1.4. Базисы функций алгебры логики....211
4.5.2. Часто используемые базовые логические элементы....211
4.5.3. Бистабильные схемы, триггеры....214

4.5.4. Другие схемы....219
4.5.5. Алгоритмизация....220
4.5.6. Машина Тьюринга, машина Поста....225
4.6. Структура классической ЭВМ....228
4.7. Процессор....232
4.7.1. Отечественные разработки (процессор Эльбрус)....236
4.7.2. Процессор изнутри....238
4.7.3. Закон Мура....240
4.7.4. Что выбрать....241
4.7.5. «Разгон» процессоров....249
4.7.6. Резюме по процессорам....250
4.8. Чипсет....254
4.8.1. Intel Management Engine, AMD Secure Processor....256
4.9. Материнская плата....256
4.9.1. Raspberry Pi®....265
4.10. Оперативная память....266
4.10.1. Внутреннее устройство памяти....268
4.11. Устройства хранения информации....270
4.11.1. Винчестер....271
4.11.2. Внешние диски, DAS, СХД, SAN, NAS, iSCSI....282
4.11.3. Пути улучшения характеристик жёстких дисков....283
4.11.4. Твердотельные накопители (SSD)....287
4.11.5. Дисководы оптических дисков....304
4.11.6. Флэш-память (USB-flash и карты памяти)....314
4.11.6.1. SD-карты памяти....315
4.11.7. Стример....317
4.12. Устройства ввода информации....321
4.12.1. Клавиатура....321
4.12.2. Компьютерная мышь....326
4.12.3. Сканер....327
4.12.4. Дигитайзер, графический планшет....330
4.12.5. Сенсорный монитор....331
4.12.6. Платы видеозахвата, TV- и FM-приёмники....333
4.12.7. Музыкальные устройства ввода....334
4.12.8. Веб-камера....334
4.13. Устройства вывода информации....335
4.13.1. Видеоадаптер....335
4.13.2. Монитор....337
4.13.3. Принтер....341
4.13.4. Плоттер....351
4.13.5. Мультимедиапроектор....353
4.13.6. Устройства вывода звука....356
4.14. Оборудование компьютерных сетей....358
4.14.1. Сетевой адаптер (сетевая карта)....359
4.14.2. Концентратор (hub)....359
4.14.3. Коммутатор (switch)....359

4.14.4. Кабель....360
4.14.5. Маршрутизатор (router)....361
4.14.6. Модем....362
4.15. Оборудование беспроводных сетей....363
4.15.1. Каналы Wi-Fi....364
4.16. Дополнительное оборудование....366
4.16.1. Сетевой фильтр....367
4.16.2. Стабилизатор напряжения....368
4.16.3. Источник бесперебойного питания (ИБП)....369
4.16.4. «Грозозащита»....373
4.17. Контрольные вопросы к главе 4....376
4.18. Литература к главе 4....378

Глава 5. Программное обеспечение....380
5.1. Введение....380
5.2. Классификация программного обеспечения....381
5.3. Операционная система....384
5.3.1. Краткая история развития операционных систем для ПК....386
5.3.1.1. История Windows (зарубежная закрытая ОС)....386
5.3.1.2. История Linux....390
5.3.1.3. Операционные системы для мобильных устройств....394
5.3.2. Процесс....396
5.3.3. Файл....399
5.3.3.1. Задачи управления файлами....400
5.3.3.2. Именование файлов....401
5.3.3.3. Типы файлов....407
5.3.4. Логическая структура файловой системы. Стандарт FHS....412
5.3.5. Пользователи и разграничение доступа....421
5.3.5.1. Дискреционная политика безопасности в ОС Linux....421
5.3.5.1.1. Проблема хранения учётных записей (и/или паролей пользователей). 424
5.3.5.1.2. Стандартные и расширенные атрибуты файлов....425
5.3.5.1.3. Изменение атрибутов (прав доступа)....428
5.3.5.1.4. Смена владельца (группы)....431
5.3.5.1.5. Изменение и просмотр расширенных атрибутов....432
5.3.5.1.6. Списки управления доступом (ACL)....434
5.3.5.1.7. Пользователи ОС Linux....437
5.3.5.2. Реализация дискреционной политики в Windows....437
5.3.5.3. Другие политики безопасности....444
5.3.6. Основа безопасного разграничения – файловая система....445
5.3.7. Файловая система (уровень организации)....447
5.3.7.1. FAT....448
5.3.7.2. VFAT....450
5.3.7.3. NTFS....451
5.3.7.3.1. Альтернативные файловые потоки в ФС NTFS....454
5.3.7.3.2. Резюме по ФС NTFS....455
5.3.7.4. exFAT....456

5.3.7.5. ext2, ext3, ext4....456
5.3.7.6. CDFS, UDF, ISO 9660....480
5.3.8. Краткое сравнение файловых систем....482
5.3.9. Таблица разделов, MBR, GPT....482
5.3.9.1. Полезные советы про таблицу разделов....484
5.3.10. Полезная информация про ОС Windows....485
5.3.11. Сбои системы: синий экран «BSOD» и kernel panic....488
5.4. Виртуализация, гипервизоры....489
5.4.1. Проблемы....491
5.4.2. Эмуляция....492
5.4.3. Облака....494
5.4.4. Недостатки облачных сервисов....495
5.4.5. QEMU....495
5.4.6. VirtualBox....496
5.4.7. Wine и DOSBox....499
5.5. Офисный пакет LibreOffice....500
5.5.1. Подготовка текстовых документов в LibreOffice Writer....500
5.5.1.1. Стили оформления....503
5.5.1.2. Создание документа....504
5.5.1.3. Формулы....506
5.5.1.4. Математические графики в тексте....508
5.5.1.5. Рисунки....509
5.5.1.6. Таблицы в тексте....509
5.5.1.7. Создание оглавления....510
5.5.1.8. Проверка правописания....510
5.5.1.9. Экспорт в PDF....511
5.5.1.10. Ленточный интерфейс в LibreOffice....511
5.5.1.11. Сохранение в облако....512
5.5.2. Другие офисные программы, входящие в состав LibreOffice....512
5.5.3. Альтернативные офисные пакеты....513
5.6. Сервисные программы....514
5.6.1. Защита от вирусов....515
5.6.2. Архивация файлов....525
5.6.3. Работа с оптическими дисками....527
5.6.4. Программы воспроизведения DVD-фильмов и видеофайлов, кодеки....530
5.6.5. Создание и просмотр специальных форматов документов....532
5.7. Контрольные вопросы к главе 5....535
5.8. Литература к главе 5....537

Приложение к главе 5. Примеры лабораторных работ....540
Лабораторная работа № 1. Разграничение доступа....540
Лабораторная работа № 2. Исследуем inode....548

Глава 6. Объединение компьютеров в сéти....577
6.1. Зачем объединяться?....577

6.1.1. Как и для чего используют сети частные лица?....580
6.1.2. Социальные аспекты в развитии сетевого обмена информацией....586
6.2. Компьютерная сеть....601
6.2.1. Классификации компьютерных сетей....602
6.2.2. Сетевые (эталонные) модели....602
6.2.3. Заключение, используемые термины....604
6.2.4. Документы RFC (Request For Comments), draft....606
6.3. Интернет....610
6.3.1. История интернета....610
6.3.2. Современная структура интернета....615
6.3.3. Некоторые сетевые протоколы....623
6.3.4. Адресация в интернете....625
6.3.5. Способы подключения к интернету конечных пользователей....635
6.3.6. Что надо «знать» компьютеру, чтобы «выходить» в интернет?....640
6.3.7. Поиск информации в интернете....641
6.3.8. Основы создания веб-страниц....646
6.4. Общение и обмен информацией в интернете между пользователями....651
6.4.1. Электронная почта....651
6.4.2. RSS-каналы....655
6.4.3. Twitter....656
6.4.4. Общение в реальном времени....657
6.4.4.1. Службы мгновенных сообщений (ICQ и др.)....658
6.4.4.2. Альтернатива времени: Whatsapp, Viber, Telegram и др....660
6.4.4.3. Коллективное виртуальное общение....663
6.4.4.4. Вебинары....666
6.4.4.5. IP-телефония....667
6.4.5. Обмен файлами....670
6.4.5.1. Торренты....670
6.4.5.2. Хранение файлов в облаке....672
6.5. Интернет-радио и интернет-телевидение....675
6.6. Электронная коммерция....676
6.7. Обеспечение безопасности информации в интернете....683
6.8. Литература к главе 6....687
6.9. Контрольные вопросы к главе 6....688

Приложение к главе 6. Примеры лабораторных работ....690
Лабораторная работа № 3. Настройка и исследование работы сетевого шлюза....690
Лабораторная работа № 4. Исследуем сокеты....711

Алфавитный указатель....740
Содержание....743

Книги издательства «ДМК Пресс» можно заказать в торгово-издательском холдинге «АЛЬЯНС-КНИГА» по электронному адресу: **orders@alians-kiniga.ru**.

При оформлении заказа следует указать адрес (полностью), по которому должны быть высланы книги; фамилию, имя и отчество получателя. Желательно также указать свой телефон.

Эти книги вы можете заказать и в Интернет-магазине: **www.dmkpress.com**.

Оптовые закупки: электронный адрес **books@alians-kniga.ru**.

Закляков В.Ф.

Информатика

5-е издание, переработанное и дополненное

(учебник)

Главный редактор *Мовчан Д. А.*
dmkpress@gmail.com
Корректор
Вёрстка *Закляков В. Ф.*
Дизайн обложки *Закляков В. Ф.*

Подписано в печать 19.01.2021. Формат 70×100 $^{1}/_{16}$.
Гарнитура «Liberation Serif». Печать цифровая.
Усл. печ. л. 54,4. Тираж 100 экз.

Страница книги (отзывы, исходные коды, обратная связь):
http://dmkpress.com/catalog/computer/handbooks/978-5-97060-921-7/
http://www.дмк.рф/catalog/computer/handbooks/978-5-97060-921-7/

Web-сайт издательства: www.дмк.рф
Интернет магазин: www.dmpkress.com

Отпечатано в ООО «Издательство ДМК Пресс»