ALT Linux 4.0 Desktop

Руководство пользователя

Дополнительная документация

составлено ООО «Альт Линукс » http://altlinux.ru

подготовлено к печати ООО «Дальком» г. Хабаровск http://dalcom.kha.ru

Руководство пользователя

Что такое Linux

Установка

Быстрый старт

Администрирование

Дополнительная документация

Linux для тех, кто хочет разобраться

Как это устроено

- Работа с оборудованием в Linux («Сага о Драйверах»)
- Из чего сделан «Рабочий стол»
- Права доступа в системе Linux
- Пользователи в Linux
- Что происходит в системе
- Настройка загрузки
- Прикладные программы для Linux
- WINE среда для запуска win-приложений на платформе Unix
- Подключение к Интернет через мобильный телефон

История и право

- Свободные программы и сообщество
- Программное обеспечение: право и свобода
- История Linux: от ядра к дистрибутивам

Зачем нужна командная строка

- Интерпретатор командной строки (shell)
- Удобства shell: экономия движений
- Ввод, вывод и конвейер
- Основные утилиты

Ресурсы в сети Интернет

- Сайт ALT Linux
- Поиск по ресурсам ALT Linux
- Вопросы и ответы
- Списки рассылки
- Система отслеживания ошибок
- Документация ALT Linux online

Лицензия

Свободные программы

Операционная система (далее — ОС) Linux — ядро и основные компоненты системы, а также большинство пользовательских приложений для Linux — свободные программы. Это означает, что их можно запускать на любом количестве компьютеров, без ограничений распространять за деньги или бесплатно, получать исходные тексты этих программ и вносить в них любые исправления.

Свобода программ обеспечила широкое их использование и интерес к ним со стороны тысяч разработчиков. Основные программы для Linux выходят под лицензией GNU General Public License (далее — GPL), которая не только гарантирует свободу, но и защищает её: она допускает дальнейшее распространение программ только под той же лицензией. Поэтому исходный код ядра Linux, компиляторов, библиотеки glibc, пользовательских оболочек KDE и GNOME не может быть использован для создания приложений с закрытым кодом. В этом принципиальное отличие Linux от свободных ОС BSD (FreeBSD, NetBSD, OpenBSD), фрагменты которых вошли в семейство Microsoft Windows и даже стали основой Mac OS X. Linux включает в себя многие разработки BSD, но её компиляторы и системные библиотеки разработаны в рамках проекта GNU.

Разработка Linux

В отличие от распространённых несвободных ОС, Linux не имеет географического центра разработки. Нет и фирмы, которая владела бы этой ОС; нет даже единого координационного центра. Программы для Linux — результат работы тысяч проектов. Некоторые из этих проектов централизованы, некоторые сосредоточены в фирмах, но большинство объединяют программистов со всего света, которые знакомы только по переписке. Создать свой проект или присоединиться к уже существующему может любой и, в случае успеха, результаты работы станут известны миллионам пользователей. Пользователи принимают участие в тестировании свободных программ, общаются с разработчиками напрямую, что позволяет быстро находить и исправлять ошибки и реализовывать новые возможности.

Именно такая гибкая и динамичная система разработки, невозможная для проектов с закрытым кодом, определяет исключительную экономическую эффективность Linux. Низкая стоимость свободных разработок, отлаженные механизмы тестирования и распространения, привлечение людей из разных стран, обладающих разным видением проблем, защита исходного текста программ лицензией GPL — всё это стало причиной успеха свободных программ.

Конечно, такая высокая эффективность разработки не могла не заинтересовать крупные фирмы, которые стали открывать свои свободные проекты. Так появились Mozilla (Netsape, AOL), OpenOffice.org (Sun), свободный клон Interbase (Borland), SAP DB (SAP). IBM способствовала переносу Linux на свои мейнфреймы.

С другой стороны, открытый код значительно снижает себестоимость разработки *закрытых систем* для Linux и позволяет снизить цену решения для пользователя. Вот почему Linux стала платформой, часто рекомендуемой для таких продуктов как Oracle, DB2, Informix, SyBase, SAP R3, Domino.

Защищённость

ОС Linux унаследовала от UNIX надёжность и отличную систему защиты. Система разграничения доступа к файлам позволяет не бояться вирусов. Тем не менее, программ без ошибок не бывает, и Linux исключением не является. Однако благодаря тому, что исходный код программ открыт, его аудит может осуществить любой специалист без подписок о неразглашении и без необходимости работать в стенах нанявшей его компании. Сообщества разработчиков и пользователей свободных программ создали множество механизмов оповещения об ошибках и их исправления. Благодаря доступности сети Интернет и открытости исходных текстов программ, сообщить об ошибке и принять участие в её исправлении независимому программисту или даже пользователю так же просто, как и специалисту

фирмы-разработчика или автору проекта. Именно поэтому ошибки защиты выявляются особенно эффективно и быстро исправляются.

Дистрибутивы Linux

Большинство пользователей для установки Linux используют дистрибутивы. Дистрибутив — это не просто набор программ, а ряд решений для разных задач пользователей, объединённых едиными системами установки, управления и обновления пакетов, настройки и поддержки.

Новичку

- Linux — самостоятельная операционная система. Здесь всё по-своему, а к новым правилам надо привыкнуть. То, что кажется поначалу странным и непривычным, завтра понравится. Все операционные системы разные: Linux — не Windows, не Mac OS и не FreeBSD. Терпение и настойчивость в изучении Linux будут вознаграждены значительным повышением эффективности и безопасности вашей работы.
- Не стесняйтесь задавать вопросы, ведь самый простой способ решить проблему и узнать новое — это общение. Взаимопомощь — хорошая традиция в мире Linux, поэтому всегда можно обратиться за помощью к сообществу пользователей и разработчиков Linux. Большинство вопросов повторяются, поэтому сначала стоит поискать ответ на ваш вопрос в документации, затем в Интернет. На сайте разработчиков вашего дистрибутива наверняка найдутся списки ответов на часто задаваемые вопросы (FAQ) и архивы списков рассылки. Если ответ всё-таки не нашёлся — не стесняйтесь писать в списки рассылки так, как писали бы своим друзьям — и вам наверняка помогут.

ALT Linux Team и проект ALT

Команда ALT Linux (ALT Linux Team) объединяет разработчиков свободных программ из России, Белоруссии, Украины, Казахстана, Эстонии и Израиля. Команда ALT — это сообщество, которое сейчас насчитывает более 150 программистов, большинство из которых не являются сотрудниками ООО «Альт Линукс». Альт Линукс координирует этот проект и осуществляет внедрение и поддержку решений.

Целью проекта ALT является разработка и поддержка широкого спектра решений на основе свободных программ, отличающихся высокой надёжностью и степенью защиты, простотой и доступностью обновления, простым и логичным интерфейсом, стандартной и качественной интернационализацией и локализацией. Все собственные разработки ALT Linux Team распространяются под свободными лицензиями. Проект ALT — часть движения по разработке и распространению свободных программ. Среди его участников есть и разработчики основных компонентов Linux. Разработки команды ALT входят во все дистрибутивы ALT Linux.

Сизиф

Sisyphus — наш ежедневно обновляемый репозиторий пакетов. На его основе создаются все дистрибутивы ALT Linux. Поддерживаемая ALT Linux Team целостность Sisyphus, оригинальная технология сборки пакетов, утилита apt-get и её оболочки alterator-packages, aptitude и synaptic позволяют пользователям легко обновлять свои системы и быть в курсе всех новостей мира свободных программ.

Вместе с тем, обратите внимание, что ежедневно изменяющийся репозиторий содержит самое новое программное обеспечение, со всеми его преимуществами и недостатками (иногда ещё не известными). Поэтому перед обновлением вашей системы из Sisyphus мы советуем взвесить преимущества от новых возможностей, реализованных в последних версиях программ, и вероятность возникновения неожиданностей в работе с ними.

Разработка Sisyphus полностью открыта. У нас нет секретных патчей и закрытого тестирования с подписками о неразглашении: то, что мы сделали сегодня, завтра вы найдёте в сети. По сравнению с другими аналогичными репозиториями (Debian unstable, Mandriva Cooker, PLD, Fedora), у нас есть много оригинального. Особое внимание уделяется защите системы, интернационализации, полноте и корректности зависимостей.

Sisyphus — не просто собрание программ, а в первую очередь лаборатория решений. Любое такое решение можно оформить в виде дистрибутива. Если вам это интересно, если вы хотите дополнить Sisyphus новыми решениями, если вы считаете, что можете собрать какой-то пакет лучше — присоединяйтесь к проекту ALT.

Sisyphus (Сизиф) — персонаж греческой мифологии. Миф о Сизифе[1], который непрерывно катил в гору камни, символизирует постоянный труд команды по усовершенствованию решений, заложенных в репозиторий. «Миф о Сизифе» — философское эссе Альбера Камю.

Дистрибутивы ALT Linux

Решение для тех пользователей, которым стабильность и предсказуемость работы системы важнее расширенной функциональности (а это в первую очередь начинающие и корпоративные пользователи) — стабильные дистрибутивы ALT Linux, выпускаемые на основе Sisyphus.

Дистрибутив Linux — это не просто собранные вместе операционная система и набор приложений, это интегрированная рабочая среда, предназначенная для решения тех или иных задач пользователей. ALT Linux выпускает дистрибутивы, ориентированные как на начинающих, так и на опытных пользователей, специализированные и универсальные. Более подробную информацию о дистрибутивах можно найти на сайте ALT Linux.

---

[1] Миф можно найти в любой соответствующей книжке, а для начинающих рекомендуем А. Куна.

Что такое ALT Linux 4.0 Desktop

ALT Linux 4.0 Desktop — это семейство простых в установке и удобных в работе дистрибутивов, дающих возможность пользователю решать обычные задачи, не опасаясь вирусов и не затрачивая время на поиск нужных прикладных программ в сети Интернет и на полках магазинов.

ALT Linux 4.0 Desktop — новый выпуск в линейке пользовательских дистрибутивов ALT Linux, в который включено всё необходимое для повседневной работы пользователя домашнего либо офисного компьютера:

- Для офисных работников:
  - полноценный офисный пакет OpenOffice.org 2.2;
  - средства просмотра и общения в Интернет: Mozilla Firefox, Mozilla Thunderbird, Konqueror, Kopete, KMail;
  - среда запуска Windows-приложений WINE.
- Для обладателей самого современного оборудования:
  - поддержка новейшего оборудования, в том числе видеокарт NVIDIA и ATI;
  - средства подключения к сети Интернет с помощью GPRS и WiFi.
- Для творчества:
  - средства создания и редактирования векторной и растровой графики: gimp, inkscape;
  - редакторы видео и звука;
  - средства обработки фотографий.
- Для обучения:
  - англо-русские словари: StarDict, Multitran;
  - словари в формате dict: WordNet, толковый словарь Даля, словарь компьютерных терминов engcom, толковый английский словарь FOLDOC;
  - обучающие программы по астрономии, географии, математике и иностранным языкам: KVocTRain, KWordQuiz, KLettres.

ALT Linux 4.0 Desktop прост в установке и настройке, а интерфейс и вводная документация — на русском языке.

Отличия

ALT Linux 4.0 Desktop отличается от других дистрибутивов ALT Linux:

Улучшенной программой установки,
: ориентированной на автоматическое определение и настройку оборудования. В большинстве случаев для успешной установки пользователю достаточно нажимать кнопку «Далее», разрешение технических вопросов можно оставить программе установки.

Оригинальной системой настройки,
: основанной на платформе ALTerator. С её помощью и опытные, и неподготовленные пользователи могут легко решать типичные задачи по настройке и администрированию системы.

Предусмотрено два дополняющих друг друга способа настройки ALT Linux 4.0 Desktop:

- Центр управления системы (www);
- Центр управления системы.

Обновления

В дистрибутив вошли последние стабильные версии основных программных пакетов:

- ядро Linux 2.6.18;

- графическая система XOrg;
- графическая пользовательская среда KDE 3.5, включающая веб-браузер Konqueror, почтовый клиент KMail, средства для работы с мультимедиа, игры и множество других приложений;
- интегрированный менеджер информации Kontact, совмещающий функциональность почтового клиента, электронной адресной книги, средства чтения новостных лент, ведения заметок, и планировки задач. Kontact также позволяет синхронизировать данные с карманными компьютерами на основе операционной системы Palm OS;
- Интернет-пейджер Kopete, поддерживающий множество протоколов передечи сообщений: jabber, msn, yahoo, aim, gadu-gadu, icq;
- веб-браузер Mozilla Firefox и почтовый клиент Mozilla Thunderbird;
- офисный пакет OpenOffice.org 2.2 с полной поддержкой русского языка, позволяющий создавать и редактировать текстовые документы, электронные таблицы, презентации, векторные изображения и даже базы данных. Возможность чтения и записи в форматы документов .doc, .xls, .ppt позволит беспрепятственно работать с накопленными ранее документами и обмениваться данными с коллегами и партнерами. Версия OpenOffice.org от ALT Linux содержит средства проверки орфографии и расстановки переносов для русского и украинского языков;
- Графический редактор GIMP 2.2 — профессиональная программа для обработки растровой графики, поддерживающая форматы PSD, GIF, JPG, PNG, BMP, TIFF, EPS и многие другие. GIMP включает множество профессиональных инструментов для обработки изображений, а также значительное число спецэффектов и дополнительных модулей. GIMP прекрасно подходит для редактирования фотографий и для создания графических элементов при оформлении веб-сайтов;
- Приложения для проигрывания мультимедийных файлов: Kaffeine, Amarok, Xine, Mplayer;
- K3B: средство записи CD и DVD.

Варианты

Издание ALT Linux 4.0 Desktop можно приобрести в розничной сети или у реселлеров ALT Linux.

Продукты из серии ALT Linux 4.0 Desktop:

ALT Linux 4.0 Desktop Personal
: выпускается на DVD. Может быть не только установлен на жёский диск, но и использован в режиме Live CD, а также как спасательная система.

ALT Linux 4.0 Desktop Small Business
: включает в себя WINE@Etersoft Local, что позволяет беспрепятственно запускать большое количество win-приложений. В остальном, эта версия идентична ALT Linux 4.0 Desktop Personal.

ALT Linux 4.0 Corporate Desktop
: предназначен для оборудования офисных рабочих мест и рабочих мест государственных служащих. ALT Linux 4.0 Corpotrate Desktop — это:

  - все возможности ALT Linux 4.0 Desktop;
  - расширенная поддержка на 5 рабочих станций;
  - WINE@Etersoft Local, то есть гарантированный запуск приложений 1С и других популярных windows-приложений. Купон на скидку при приобретении в ALT Linux WINE@Etersoft Network и SQL;
  - средства интеграции в гетерогенные сети;
  - средства коллективной работы;
  - расширенная печатная документация.

ALT Linux 4.0 Desktop Live CD
: позволяет воспользоваться всеми преимуществами ALT Linux 4.0 Desktop, не устанавливая операционную систему на жёсткий диск компьютера.

ALT Linux 4.0 Desktop Lite

адаптирован для компьютеров предыдущего поколения с небольшим объёмом оперативной памяти и медленным жёстким диском.

Все редакции ALT Linux 4.0 Desktop основаны на едином репозитории пакетов (срезе Sisyphus), поэтому устанавливать дополнительное программное обеспечение в ALT Linux 4.0 Desktop можно будет из любой редакции.

Редакции Personal, Live и Lite доступны для свободной загрузки с ftp://ftp.altlinux.org. Версии Business и Corporate доступны для загрузки только в их свободной части.

Если вы приобретаете нашу продукцию в магазинах, то тем самым вносите очевидный вклад в развитие свободных программ для Linux и совершенствование дистрибутивов ALT Linux.

Дистрибутив разработан в расчёте не только на розничную продажу, но и для OEM-поставок с новыми компьютерами.

Другие продукты ALT Linux серии 4.0

ALT Linux 4.0 Junior
: предназначен для использования в школьном образовательном процессе. Дистрибутив включает в себя все приложения, соответствующие утверждённой программе преподавания информатики в общеобразовательных учреждениях, и характеризуется исключительно простой процедурой установки.

ALT Linux 4.0 Server
: предназначен для создания серверов любых типов, обладает уникальной системой управления и настройки (ALTerator), включает в себя средства виртуализации и готовые профили для создания виртуальных серверов. Гарантированный срок поддержки продукта — три года. Высокое качество и оперативность услуг по техническому сопровождению обеспечивается тесным взаимодействием специалистов техподдержки и разработчиков системы.

ALT Linux 4.0 Office Server
: предназначен для организации эффективной коллективной работы на крупных и средних предприятиях и в государственных структурах. В состав дистрибутива входят средства автоматизации работы с клиентами (SugarCRM), предусмотрена возможность интеграции в инфраструктуру Active Directory. Сервер полностью настраивается и управляется через Web-интерфейс системы ALTerator, от администратора не требуется работать в командной строке.

Требования

Аппаратные требования:

- для DVD-редакции издания требуется привод DVD;
- процессор совместимой с Pentium III архитектуры (рекомендуется тактовая частота не ниже 1 ГГц);
- объём оперативной памяти от 128 Мб (рекомендуется от 256 Мб);
- свободное место на жёстком диске от 1 Гб (рекомендуется от 10 Гб);
- рекомендуется 3D-ускоритель (NVidia, ATI).

Установка

Перед установкой

Прежде чем приступить к установке, особенно если вы делаете это впервые, стоит потратить немного времени на подготовку.

Сделайте резервное копирование данных

Если установка производится на жёсткий диск, содержащий важные данные, прежде всего необходимо позаботиться об их сохранности. Если установка производится на чистый диск, данный раздел можно пропустить.

- Сохранение данных и меры предосторожности.

Решите, куда установить ALT Linux

Если вы хотите, чтобы ALT Linux использовал весь жесткий диск, либо если на диске имеется свободное пространство, не используемое другими операционными системами и достаточное для установки ALT Linux, не о чем беспокоиться — можете пропустить этот раздел.

В противном случае необходимо продумать стратегию разбиения диска. Будьте особо внимательны, если вы устанавливаете ALT Linux в добавок к уже установленным операционным системам и хотите иметь возможность попеременно загружаться в нужную вам ОС. Самостоятельное разбиение диска — очень ответственный этап установки системы, поэтому необходимо иметь представление о том, как данные размещаются на диске.

- Структура жёсткого диска
- Планирование диска
- Разбиение диска средствами программы установки
- Именование дисков и разделов в Linux
- Файловая система Linux
- Типы файловых систем

Сохранение данных и меры предосторожности

Если вы хотите установить ALT Linux и при этом сохранить уже установленную на вашем компьютере операционную систему (например, другую версию GNU/Linux или Microsoft Windows), вам нужно обязательно позаботиться о подготовке компьютера к установке второй системы и о сохранении ценных для вас данных.

- Если у вас нет загрузочного диска (дискеты) для уже установленной системы — создайте его. В случае прерванной установки ALT Linux или неправильной настройки загрузчика вы можете потерять возможность загрузиться в вашу предыдущую ОС, тут вам пригодится загрузочный диск.
- Если на диске, на который вы собираетесь установить ALT Linux, не осталось свободного раздела, то программа установки должна будет изменить размер существующего раздела. От этой операции могут пострадать ваши данные, поэтому предварительно надо выполнить следующие действия:
    - Выполнить проверку раздела, который вы собираетесь уменьшать. Для этого воспользуйтесь соответствующим ПО, входящим в состав уже установленной ОС. Программа установки ALT Linux может обнаружить некоторые очевидные ошибки при изменении размера раздела, но специализированое ПО предустановленной ОС справится с этой задачей лучше.
    - Для большей безопасности данных следует также выполнить для этого раздела дефрагментацию. Это действие уменьшит риск потери данных, оно не является обязательным, но мы настоятельно рекомендуем его произвести: изменение размера раздела пройдёт легче и быстрее.

Внимание

Не уменьшайте NTFS-раздел с установленной Microsoft Windows Vista средствами программы установки. В противном случае вы не сможете загрузить Microsoft Windows Vista после установки ALT Linux. Для выделения места под установку ALT Linux воспользуйтесь средствами, предоставляемыми самой Microsoft Windows Vista — «Управление дисками — Сжать»

Внимание

Полной гарантией от проблем с потерей данных является резервное копирование!

Сектора

Любой жёсткий диск можно представить как огромный «чистый лист», на который можно записывать данные и откуда потом их можно считать. Чтобы ориентироваться на диске, всё его пространство разбивают на небольшие «клеточки» — **сектора**. Сектор — это минимальная единица хранения данных на диске, обычно его размер составляет 512 байт. Все сектора на диске нумеруются: каждый из n секторов получает номер от 0 до n–1. Благодаря этому любая информация, записанная на диск, получает точный адрес — номера соответствующих секторов. Так что диск ещё можно представить как очень длинную строчку (ленточку) из секторов. Можете посчитать, сколько секторов на вашем диске размером в N гигабайт.

Разделы

Представлять жёсткий диск как единый «лист» не всегда бывает удобно: иногда полезно «разрезать» его на несколько независимых листов, на каждом из которых можно писать и стирать что угодно, не опасаясь повредить написанное на других листах. Логичнее всего записывать раздельно данные большей и меньшей важности или просто относящиеся к разным вещам.

Конечно, над жёстким диском следует производить не физическое, а логическое разрезание, для этого вводится понятие **раздел** (partition). Вся последовательность (очень длинная ленточка) секторов разрезается на несколько частей, каждая часть становится отдельным разделом. Фактически, нам не придётся ничего разрезать (да и вряд ли бы это удалось), достаточно объявить, после каких секторов на диске находятся границы разделов.

Таблица разделов

Технически разбиение диска на разделы организовано следующим образом: заранее определённая часть диска отводится под **таблицу разделов**, в которой и написано, как разбит диск. Стандартная таблица разделов для диска IBM-совместимого компьютера — HDPT (**H**ard **D**isk **P**artition **T**able) — располагается в конце самого первого сектора диска, после **предзагрузчика** (**M**aster **B**oot **R**ecord, MBR) и состоит из четырёх записей вида «*тип начало конец*», по одной на каждый раздел. *Начало* и *конец* — это номера тех секторов диска, где начинается и заканчивается раздел. С помощью такой таблицы диск можно поделить на четыре или меньше разделов: если раздела нет, *тип* устанавливается в 0.

Однако четырёх разделов редко когда бывает достаточно. Куда же помещать дополнительные поля таблицы разбиения? Создатели IBM PC предложили универсальный способ: один из четырёх основных разделов объявляется **расширенным** (extended partition); он, как правило, является последним и занимает *всё* оставшееся пространство диска.

Расширенный раздел можно разбить на подразделы тем же способом, что и весь диск: в самом начале — на этот раз не диска, а самого *раздела* — заводится **таблица разделов**, с записями для четырёх разделов, которые снова можно использовать, причём один из подразделов может быть, опять-таки, расширенным, со своими подразделами и т. д.

Разделы, упомянутые в таблице разделов *диска*, принято называть **основными** (primary partition), а все подразделы расширенных разделов — **дополнительными** (secondary partition). Так что основных разделов может быть не более четырёх, а дополнительных — сколько угодно.

Чтобы не усложнять эту схему, при разметке диска соблюдают два правила: во-первых, расширенных разделов в таблице разбиения *диска* может быть не более одного, а во-вторых, таблица разбиения *расширенного раздела* может содержать либо одну запись — описание дополнительного раздела, либо две — описание дополнительного раздела и описание вложенного расширенного раздела.

Тип раздела

В таблице разделов для каждого раздела указывается **тип**, который определяет **файловую систему**, которая будет содержаться в этом разделе. Каждая операционная система распознаёт определённые типы и не распознаёт другие, и, соответственно, откажется работать с разделом неизвестного типа.

Следует всегда следить за тем, чтобы тип раздела, установленный в таблице разделов, правильно указывал тип файловой системы, фактически содержащейся внутри раздела. На сведения, указанные в таблице разделов, может полагаться не только ядро операционной системы, но и любые утилиты, чьё поведение в случае неверно указанного типа может быть непредсказуемым и повредить данные на диске.

Подробнее о файловых системах см. раздел Типы файловых систем.

Логические тома (LVM)

Работая с разделами, нужно учитывать, что производимые над ними действия связаны непосредственно с разметкой жёсткого диска. С одной стороны, разбиение на разделы — это наиболее традиционный для PC способ логической организации дискового пространства. Однако если в процессе работы появится потребность изменить логику разбиения диска или размеры областей (т. е. когда возникает задача **масштабирования**), работа с разделами не очень эффективна.

Например, при необходимости создать новый раздел или увеличить размер существующего, можно столкнуться с рядом трудностей, связанных с ограничением количества дополнительных разделов или перераспределением данных. Избежать их очень просто: нужно лишь отказаться от «привязки» данных к определённой области жёсткого диска. В Linux эта возможность реализуется при помощи **менеджера логических томов** (LVM — **L**ogical **V**olume **M**anager). LVM организует дополнительный уровень абстракции между *разделами* с одной стороны и хранящимися на них *данными* с другой, выстраивая собственную иерархическую структуру.

Дисковые разделы (в терминологии LVM — **физические тома**) объединяются в **группу томов**, внутри которой создаются **логические тома**. Таким образом, группа томов выстраивает соответствие между физическим и логическим пространством диска.

Технологически это организуется следующим образом. Физические тома разбиваются на отдельные блоки — **физические экстенты**, которые объединяются в **группу томов**. Логические тома разбиваются на блоки такого же размера — **логические экстенты**. В разных группах томов размер экстента может быть различным.

Отношения между логическими и физическими томами представлены в виде **отображения** логических экстентов в физические. Возможны два способа отображения — **линейное** и **расслоенное** (striped). В первом случае логические экстенты располагаются последовательно соответственно физическим, во втором поочередно распределяются между несколькими физическими томами.

В свою очередь, между логическим томом и группой томов возникают отношения, аналогичные таковым между разделом и жёстким диском, с отличием в уровне абстракции и, соответственно, колоссальной разнице в гибкости манипуляции. Поскольку раздел — конкретная область физического диска между двумя определёнными секторами, а том — логическая категория, принимаемая для удобства использования дискового пространства, производить манипуляции со вторым значительно проще. Можно свободно перераспределять логические тома внутри группы, изменять их размер, увеличивать размер группы томов за счёт внесения в неё нового раздела (только при линейном отображении) и многое другое.

Дисковые массивы (RAID)

Иногда обычной производительности жёсткого диска может не хватать. В случаях, когда во главу угла ставится скорость работы с данными (скорость записи и чтения) или надёжность их хранения, используется технология **RAID** (Redundant array of independent disks — избыточный массив независимых дисков). Технология RAID позволяет объединять несколько физических дисковых устройств (жёстких дисков или разделов на них) в **дисковый массив**. Диски, входящие в массив, управляются централизованно и представлены в системе как одно логическое устройство, подходящее для организации на нем единой файловой системы.

Существует два способа реализации RAID: аппаратный и программный. Аппаратный дисковый массив состоит из нескольких жёстких дисков, управляемых при помощи специальной платы контроллера RAID-массива. **Программный RAID** в Linux-системах (Linux Software RAID) реализуется при помощи специального драйвера (**M**ultiple **D**evice driver — драйвер MD-устройства). В программный массив организуются дисковые разделы, которые могут занимать как весь диск, так и его часть, а управление осуществляется посредством специальных утилит (mdadm).

Программные RAID-массивы, как правило, менее надежны, чем аппаратные, но обеспечивают более высокую скорость работы с данными (производительность процессора и системной шины обычно намного выше, чем у любого дискового контроллера). Также их преимущество по сравнению с аппаратными массивами: независимость от форматов данных на диске и как следствие — большая совместимость с различными типами и размерами дисков и их разделов. Использование программного RAID также позволяет сэкономить на покупке дополнительного оборудования. Однако обратной стороной медали станет увеличение нагрузки на процессор и системную шину, это следует иметь в виду, принимая решение об использовании программного RAID.

Уровни RAID

Существует несколько разновидностей RAID-массивов, так называемых **уровней**. В Linux поддерживаются следующие уровни программных RAID-массивов.

RAID0

Для создания массива этого уровня понадобится как минимум два диска одинакового размера. Запись осуществляется по принципу **чередования**: данные делятся на **чанки** (chunk) — порции данных одинакового размера, и поочерёдно распределяются по всем дискам, входящим в массив. Поскольку запись ведётся на все диски, при отказе одного из них будут утрачены *все* хранившиеся на массиве данные. Это цена выбора в пользу увеличения скорости работы с данными: запись и чтение на разных дисках происходит параллельно и, соответственно, быстрее.

RAID1

Массивы этого уровня построены по принципу **зеркалирования**, при котором все данные, записанные на одном диске, дублируются на другом. Для создания такого массива потребуется два или более дисков одинакового размера. **Избыточность** обеспечивает отказоустойчивость массива: в случае выхода из строя одного из дисков, данные на другом остаются неповреждёнными. Расплата за надёжность — фактическое сокращение дискового пространства вдвое. Скорость чтения и записи остается на уровне обычного жесткого диска.

RAID4

В массивах RAID4 реализован принцип **чётности**, объединяющий технологии чередования и зеркалирования. Один из трёх (или из большего числа) дисков задействуется для хранения информации о чётности в виде суперблоков с контрольными суммами блоков данных, последовательно распределённых на остальных дисках (как в RAID0). Достоинства этого уровня — отказоустойчивость уровня RAID1 при меньшей избыточности (из скольких бы дисков не состоял массив, под контрольную информацию задействуется лишь один из них). При отказе одного из дисков утраченные данные можно будет восстановить из контрольных суперблоков, причем, если в составе массива есть резервный диск, реконстукция данных начнется автоматически. Очевидным недостатком, однако, является снижение скорости записи, поскольку информацию о чётности приходится высчитывать при каждой новой записи

на диск.

RAID5

Этот уровень аналогичен RAID4, за тем исключением, что суперблоки с информацией о чётности располагаются не на отдельном диске, а равномерно распределяются по всем дискам массива вместе с блоками данных. Как результат — повышение скорости работы с данными и высокая отказоустойчивость.

Массивы всех уровней помимо блоков данных и суперблоков с контрольными суммами могут также содержать специальный **суперблок** (persistent superblock), который располагается в начале всех дисков массива и содержит информацию о конфигурации MD-устройства. Наличие отдельного суперблока позволяет ядру операционной системы получать информацию о конфигурации устройства RAID прямо с дисков, а не из конфигурационного файла, что может быть полезным, если файл по каким-то причинам перестанет быть доступным. Кроме того, наличие отдельного суперблока — необходимое условие автоопределения RAID-устройств при загрузке системы.

Более подробная информация о RAID

Более подробную информацию можно найти в документации и статьях, посвящённых RAID:

- mdadm(8)
- http://opennet.ru/docs/HOWTO/Software-RAID-HOWTO.html

(русский перевод: http://www.opennet.ru/docs/HOWTO-RU/Software-RAID-HOWTO.html)

- http://freesource.info/wiki/HCL/XranenieDannyx/SoftwareRAID
- http://ferra.ru/online/storage/26107/
- http://citforum.ru/operating_systems/linux/raid_linux/
- http://nber.org/sys-admin/linux-nas-raid.html
- http://pythian.com/blogs/411/aligning-asm-disks-on-linux
    - http://linux-ata.org/faq-sata-raid.html

На жёстком диске любого компьютера хранятся данные, которые используются совершенно по-разному. Одни составляют операционную систему или нужны ей для работы, другие нужны пользователю, он их создаёт сам или откуда-то получает. Некоторые данные нужны временно, например, только на время работы программы, другие предназначены для «вечного» хранения. Есть такие данные, которые может изменить только человек, и такие, которые система сама создаёт или модифицирует в процессе работы. Наконец, есть такие данные, которые могут храниться на одном компьютере, а использоваться на нескольких (например, по локальной сети), и такие, которые предназначены только для данного компьютера.

Надёжность хранения данных и эффективность доступа к ним возрастает, если *разделять* данные разных типов (различающиеся по характеру использования). Для этого всё доступное пространство на жёстком диске (или дисках) разделяется на независимые области, каждая из которых предназначена для данных определённого типа. Для организации таких областей хорошо подходит технология разделения диска на **разделы**.

Поскольку разделы не зависят друг от друга, изменение содержимого одного раздела никак не сказывается на других. Одна из выгод такого подхода: в случае физического сбоя повреждения данных будут локализованы внутри того раздела, где произошёл сбой, и не затронут других разделов. Разделы открывают также путь для оптимизации скорости доступа: скорость чтения и записи для большинства дисков выше в середине и ниже к концу и началу диска. В самой быстрой области можно расположить раздел с данными, для которых важна скорость доступа.

Разделение диска на разделы *необходимо* в том случае, если на одном физическом устройстве должны быть установлены несколько операционных систем. Каждой операционной системе потребуется выделить не менее одного раздела.

Необходимые разделы

Минимальное количество разделов, которые необходимы Linux для работы — два. Первый — для **корневой файловой системы**, второй — для **области подкачки**.

Область подкачки (swap space) — это пространство на диске, используемое системой для организации **виртуальной памяти**. Если какая-то область оперативной памяти долгое время не используется, её содержимое записывается на диск, в область подкачки — тем самым освобождается место в физической памяти для других процессов. Когда же эта область памяти потребуется вновь, ядро *подкачает* её с диска и разместит в оперативной памяти.

Благодаря этому вполне может сложиться ситуация, когда используется больше оперативной памяти, чем её есть в действительности: если не вся заказанная память используется одновременно, что позволяет системе откачивать некоторые области. В Linux принято размещать область подкачки на отдельном разделе, что позволяет увеличить скорость доступа к данным и уменьшить риск повреждения ценных данных на основных разделах.

Для обеспечения максимальной скорости доступа к данным области подкачки ее раздел рекомендуется размещать в начале либо в середине диска. Данные, находящиеся в swap, являются временными и не представляют ценности после перезагрузки компьютера. Поэтому размещение swap-раздела, как и других разделов с неуникальными данными (например, /tmp и /usr — о них расказано ниже), в начале диска предпочтительнее: снижается риск потери важных данных.

Начало диска более подвержено повреждениям в том числе и из-за человеческого фактора. Печально известным примером опечатки, которая ведет к уничтожению данных в начале диска, может послужить dd of=/dev/hda вместо dd of=/dev/hda3. Однако если в начале диска хранятся неуникальные данные, то этот процесс, будучи вовремя остановленным, не успеет добраться до действительно ценной информации в разделах /etc, /home или /var, в то время как восстановление таблицы разделов — задача

несложная.

С корневой файловой системы начинает расти всё дерево файлов Linux. **Точкой монтирования** для корневой файловой системы служит “/” — корневой каталог. Можно поместить все данные (включая пользовательские файлы) на один раздел — это как раз тот случай, когда для Linux потребуется всего два раздела. Для повышения эффективности и надёжности некоторые ветви дерева файлов можно выносить на другие разделы. Поскольку для файловой системы Linux не важно, каким образом части дерева каталогов расположены на разделах дисков, у вас есть возможность использовать каждый из имеющихся разделов диска под любой каталог файловой системы. В этом случае раздел с корневым каталогом будет служить точкой монтирования для остальных файловых систем.

Смонтированной файловой системе важно указать правильные **параметры монтирования**. Задавая разделам с разными типами данных походящие параметры, можно добиться значительного повышения производительности и безопаснсти. Ниже перечислены наиболее часто используемые общие параметры монтирования.

noatime
: При каждом доступе к файлу, в том числе при чтении, обновляется время последнего доступа к нему. При использлвании этого параметра это обновление производиться не будет, что может быть полезно для ускорения работы (особенно актуально для серверов).

nodev
: Этот параметр не позволяет создавать на файловой системе файлы-устройства. Если точно известно, что на данной файовой системе файлы-устройства не нужны, можно использовать этот параметр для повышения безопасности.

nosuid
: Параметр запрещает исполнение SUID-программ.

noexec
: Запрещает запуск исполняемых программ из файлов на данной файловой системе.

ro
: Обеспечивает доступ к файловой системе только для чтения.

Дополнительно выделяемые разделы

/home
: Домашние каталоги пользователей. Здесь хранятся персональные каталоги всех пользователей машины. Размер каталога зависит от количества работающих пользователей и от их потребностей. Рекомендуемые файловые системы — Ext3 или XFS. Параметры — noexec (в случае невозможности применения — nosuid).

/usr
: Статические данные: большая часть пакетов устанавливает свои исполняемые файлы и данные в каталог /usr. Преимуществом размещения этого каталога в отдельном разделе является то, что при нормальной работе (кроме установки/удаления пакетов) не требуется в него записывать никаких данных, поэтому этот раздел можно монтировать в режиме «только чтение», в том числе по локальной сети. В этом случае несколько машин могут пользоваться одним сетевым разделом /usr. Размер этого раздела зависит от количества пакетов, которые будут установлены, он колеблется в пределах от 500 Мб для маленькой установки до нескольких гигабайт для полной установки. Вариант на 2–3 Гб (в зависимости от размера диска) скорее всего подойдёт.

/var
: Переменные данные, которые создаются системой в процессе работы. Запись в этот каталог осуществляется весьма часто, а количество данных в нём имеет тенденцию расти (здесь расположены все **системные журналы**). Требования к объёму очень сильно зависят от профиля машины. На пользовательских домашних станциях может быть достаточно и нескольких сотен мегабайт, однако лучше выделять не меньше свободного места, чем для раздела /usr. На серверах объем раздела с переменными данными, как правило, больше. К тому же, для повышения производительности и надежности хранения информации переменные данные разных типов рекомендуется располагать на разных разделах. Файловая система этого раздела должна

поддерживать журналирование (Ext3). При монтировании желательно задать параметры noexec и nosuid.

/var/log
: При установке как на сервер, так и на рабочие станции, лучшим решением будет вынести системные журналы на отдельный раздел. При сбоях или внешних атаках размер журналов может резко увеличиваться, заполняя все доступное на разделе пространство, что, в случае их хранения вместе с другими переменными системными данными, чревато сбоем в работе системы. Также, если сервер используется для выполнения узкого круга задач (например, как web-сервер), рекомендуется выносить на отдельный раздел журналы основной системной службы (например, /var/log/apache). Оптимальная файловая система — Ext3, параметры — noatime, noexec, nosuid.

/var/tmp
: Может быть полезным создать отдельную файловую систему для временных данных, которые нежелательно потерять в случае программного или аппаратного сбоя. Этот раздел должен обеспечивать высокую надежность хранения данных, поэтому оптимально создать в нем файловую систему с поддержкой журналирования (Ext3), указав параметры noexec и nodev.

/var/spool/mail
: Если на сервере хранится почта пользователей, каталог с ней необходимо выделить в отдельный раздел. Также обязательно устаналивать ограничения на использование дискового пространства для отдельных пользователей, чтобы избежать неожиданного переполнения раздела и проблем с работоспособностью сервера.

/var/www
: Раздел для сайтов пользователей.

/tmp
: Этот каталог предназначен для **временных файлов**: в таких файлах программы хранят промежуточные данные, необходимые для работы. После завершения работы программы временные файлы теряют смысл и должны быть удалены. Обычно каталог /tmp очищается при каждой загрузке системы. Поскольку запись в этот каталог осуществляется очень часто, а требования к надёжности очень низкие, то есть большой смысл выделить /tmp в отдельный раздел. В противном случае он окажется частью раздела “/”, требования к которому по записи и надёжности прямо противоположные (см. ниже). Размер раздела /tmp в обычном случае должен быть примерно равен размеру swap. В последнее время раздел /tmp зачастую размещают в виртуальной файловой системе tmpfs непосредственно в оперативной памяти.

/
: Корневой раздел — это самый важный раздел. Он не только содержит наиболее важные данные и программы системы, но будет также служить точкой монтирования для других разделов. Если /usr, /var и /home вынесены на отдельные разделы, то потребность в объёме корневого раздела небольшая, обычно достаточно 500 Мб. Требования: корневой раздел должен быть доступен в процессе загрузки, в процессе работы доступ на запись в этот раздел требуется нечасто, но весьма важна надёжность.

/boot
: Небольшой раздел (5–10 Мб), на котором хранятся исполняемые и failsafe ядра и данные используемого загрузчика. Обычно располагается в самом начале жесткого диска и всегда является первичным разделом (в отличие от логических томов в случае использования LVM). Оптимальная файловая система — Ext2, поскольку запись в раздел производится редко, а его объем мал.

Выделение вышеперечисленных разделов направлено прежде всего на повышение эффективности работы сервера. Для домашних рабочих станций чаще всего вполне достаточно, помимо необходимых разделов, выделить всго один — для хранения пользовательских данных (/home). Увеличения гибкости управления разделами, особенно при большом их количестве, можно добиться использованием технологии LVM, которая позволяет создавать, удалять и изменять размер логических устройств без риска потери данных.

Для администратора Linux важным моментом при установке системы является планирование и организация дискового пространства. Правильное планирование способствует успешному поддержанию работоспособности системы в далнейшем. Программа-установщик, кроме стандартных средств, поддерживает технологии, повышающие гибкость работы с жестким диском.

Рекомендации по разбиению диска

Доступное Linux дисковое пространство, как правило, разбивается на несколько логических областей, или **томов** в терминологии программы установки. **Том** — это дополнительный уровень между разделом и файловой системой, который создается для унифицированного представления в операционной системе различных типов устройств (аналогичен логическому тому LVM).

Разбивка на тома может быть организована с помощью разных технологий: самое простое — создавать тома, привязанные непосредственно к физическим дискам или областям дисков: т. е. занимать под том целиком жёсткий диск или раздел жёсткого диска. При использовании одного из стандартных профилей разбиения диска применяется именно эта схема: создаётся несколько разделов на свободном месте жёстких дисков.

Программа установки позволяет создавать на диске и более сложную разметку с использованием технологий LVM и Linux Software RAID. Технология LVM предоставляет возможность более гибко распределять логические тома по физическим устройствам. Интерфейс управления логическими томами доступен при выборе пункта LVM в дереве устройств.

Помимо этого, программа установки позволяет устанавливать ALT Linux на поддерживаемые аппаратные и программные RAID-массивы (в том числе создавать программные RAID уровней 0, 1, 4/5). Интерфейс для создания RAID доступен при выборе в дереве устройств пункта RAID.

Перед размещением данных на логическом томе в нём должна быть создана **файловая система** (т. е. произведено форматирование раздела). Далее каждому тому (точнее, файловой системе в нём) должна быть назначена **точка монтирования**, т. е. тот фрагмент единой файловой системы Linux, который следует разместить на этом томе.

Работа с диском

В дереве устройств представлены доступные жёсткие диски и разделы на них (в том числе здесь могут оказаться съёмные USB-носители, подключённые к компьютеру в момент установки), а также в дерево включены отдельные ветки для управления/отображения устройств LVM и RAID. Узнать, каким устройствам вашего компьютера соответствуют названия в списке, можно в разделе Именование дисков и разделов в Linux.

Если на жестком диске присутствует таблица разделов, в ветке дерева, начинающейся от этого диска, будет отображено текущее расположение разделов, кроме случаев, когда раздел входит в состав устройств LVM или RAID — такие разделы в составе диска не о отображаются. Для каждого раздела указаны его размер и тип файловой системы (в колонке «Файловая система»). Возможно удалить существующую таблицу разделов диска. Для этого понадобится поочередно удалить с него все разделы, после чего, выбрав диск в дереве устройств, нажать «Удалить таблицу разделов».

Для каждого вновь создаваемого раздела предлагается выполнить стандартную последовательность операций: от создания раздела до назначения точки монтирования.

- Создать раздел
- Создать том
- Создать файловую систему
- Назначить точку монтирования

Для **создания нового раздела** выберите свободное место на диске (выбрав в списке значок диска или свободную область на нем) и нажмите «Создать раздел». Если свободного места нет и оно не было освобождено заранее, это нужно сделать сейчас, удалив один или несколько из существующих разделов или, если есть возможность, уменьшив их размер.

При создании раздела прежде всего нужно указать его размер и определить его расположение на диске. Для этого используются регуляторы «Размер» и «Смещение». Можно **изменить размер** уже созданного раздела, для этого выберите раздел и нажмите кнопку «Увеличить» или «Уменьшить». При увеличении раздела пределом служит свободное место на диске, а при уменьшении — объём, фактически *занятый данными* на этом разделе.

Том с файловой системой, как правило, создается в разделе диска, однако может быть создан непосредственно на жестком диске, в случае если на нем ещё нет таблицы разделов (эту возмжность следует использовать с осторожностью, поскольку есть риск в будущем принять такой диск за неформатированный и потерять данные на нем). Для **создания тома** выделите нужный диск или раздел и нажмите «Создать том». Поскольку единственным параметром тома является тип создаваемой в нем файловой системы, вам будет сразу предложено выбрать в появившемся списке ее тип и перейти к следующей операции — назначению точки монтирования.

Можно отложить операцию **создания файловой системы** (сняв метку с пункта перехода к следующей операции), например, для того, чтобы изменить размер только что созданного тома. Вместе с размером тома изменится и размер раздела, в котором создан изменяемый том. Изменять размер только что созданного тома с файловой системой нельзя. Для изменения размера такого тома файловую систему с него необходимо предварительно удалить. Для создания файловой системы нажмите «Создать файловую систему».

Для тома с файловой системой могут быть доступны дополнительные настройки: проверка тома на наличие ошибок (сбойных участков) и присвоение ему метки тома. Для файловой системы Ext2/3 можно выбрать, использовать ли функцию поддержки журналирования.

Созданной файловой системе возможно сразу присвоить точку монтирования. причем наиболее подходящий вариант будет предложен по умолчанию. Есть возможность выбрать из списка наиболее часто используемых вариантов или вписать нужный самостоятельно. Выбор точки монтирования для файловой системы на уже существующем разделе осуществляется нажатием кнопки «Изменить точку монтирования».

Работа с LVM

Не размещайте корневую файловую систему «/» на LVM-томе. В противном случае вы не сможете загрузить систему.

Для создания группы томов и логических томов LVM необходимым условием является наличие на диске как минимум одного пустого раздела, т. е. такого, на котором нет тома с файловой системой. Необходимо создать такой раздел, не забыв правильно указать его тип — Linux LVM.

Выбрав в списке устройств LVM, нажмите кнопку «Создать группу томов». Сразу появится окно создания группы томов, в котором нужно определить основные параметры — дать новой группе имя и выбрать размер экстента.

Внутри группы томов создаются логические тома. Их может быть сколь угодно много в зависимости от требований пользователя. Как и при работе с разделами, можно сразу перейти к созданию логического тома, отметив пункт «Создать том». Если вы не хотите создавать том, например, если вы решили изменить размер группы томов, снимите выделение с этого пункта. Будет создана пустая группа томов, к созданию тома внутри неё можно вернуться, выделив ее и нажав «Создать том».

Каждому созданному в группе томов логическому тому нужно дать название и указать его размер. Имя тома может быть любым и, например, указывать на тип хранящихся на томе данных. Отметьте разделы

для размещения каждого тома и способ распределения данных по разделам. Выбор линейного или распределенного отображения логических экстентов в физические осуществляется при помощи движка «Число расслоений (stripes)». Перемещая его, можно изменять числовое значение от единицы (обозначающей линейное отображение) до числа, соответствующего количеству физических томов, по которым будут распределяться данные создаваемого логического тома. В новом томе нужно создать файловую систему.

В результате в дереве устройств LVM появится созданная группа томов с вложенными логическими томами. Одновременно с этим разделы, вошедшие в группу томов, перестанут отображаться среди разделов диска.

Над логическими томами LVM можно производить те же операции, что и над разделами с томами: изменять их размер или удалять. Для удаления группы томов необходмо сначала удалить все входящие в нее логические тома, в противном случае в качестве отказа выполнить операцию появится сообщение «Device or resourse busy».

Работа с RAID

Для RAID, так же как для LVM, необходим пустой раздел. Будьте внимательны: для того, чтобы при старте системы RAID-массив определялся корректно, необходимо указать тип раздела Linux RAID. Создав раздел нужного типа, в таблице устройств нужно выбрать RAID и нажать «Создать RAID».

В открывшемся списке выберите уровень RAID-массива, который вы хотите создать. Ниже перечислены уровни RAID, которые позволяет создавать программа установки.

RAID 0
: Для массива этого уровня нужно определить два параметра: определить размер чанка (минимум 4 кб, 32 кб по умолчанию) и выбрать, нужно ли создавать в нем отдельный суперблок.

RAID 1
: Для этого уровня, кроме вышеперечисленных параметров, можно определить количество резервных дисков. Есть пункт «Деградированный массив», выбрав который, можно создавать массив с неполным набором дисков. Это может быть полезно, если вы решили создать массив, но еще не установили второй диск.

RAID 4/5
: Для создания массивов этих уровней определяются те же параметры, что и для устройств уровня RAID1: размер чанка, наличие отдельного суперблока и поддержка возможности создания неполного (деградированного) массива. Также можно выбрать, какие диски или разделы войдут в массив, а какие будут использоваться в качестве резервных. Поскольку в массивах RAID4/5 используется чётность, помимо перечисленных выше параметров можно выбрать алгоритм проверки чётности, выбрав нужное значение из выпадающего меню рядом с соответствующим пунктом («Алгоритм RAID5»).

После создания массива в нем создается *один том* с файловой системой. Эта операция аналогична созданию тома в разделе диска. Том занимает весь объем массива, в него входят все разделы или диски, входящие в массив. Размер тома не может быть изменен, пользователю доступны операции удаления устройства целиком или содержащейся на нем файловой системы, изменение точки монтирования. Для их выполнения служат соответствующие кнопки, отображающиеся на экране при выборе устройства RAID.

Именование дисков и разделов в Linux

Файлы, соответствующие устройствам постоянного хранения информации, в том числе жёстким дискам, получают в Linux специальные наименования в зависимости от типа и способа подключения.

Файлы устройств

Многие устройства, в том числе жёсткие диски, лазерные приводы и разнообразные съёмные носители, представлены в системе Linux в виде *файлов* особого типа — файлов устройств (их ещё называют иногда файлами-дырками). Операции чтения и записи на диск система выполняет как чтение/запись файла, соответствующего данному устройству. Все файлы устройств располагаются в специально предназначенном для них каталоге: /dev. Полные имена файлов устройств, соответствующих дискам и разделам дисков, складываются из названия каталога /dev/ и обозначения соответствующего диска или раздела. Например, первому основному разделу первого диска IDE в Linux соответствует файл /dev/hda1.

Имена файлов, соответствующих жёстким дискам и разделам, довольно часто встречаются в конфигурационных файлах и в интерфейсе некоторых программ (особенно утилит, предназначенных для администрирования системы).

Устройства IDE

На сегодняшний день один из наиболее распространённых способов подключения жёстких дисков и лазерных (CD/DVD) приводов для IBM-совместимых персональных компьютеров — шина IDE. В Linux первый жёсткий диск на шине IDE обычно называется hda (**h**ard **d**isk «**a**»). Второй диск получает имя hdb, третий — hdc и так далее. Лазерные накопители по имени никак не отличаются от жёстких дисков. Часто бывает, что жёсткий диск — первый в системе (hda), а лазерный накопитель — *третий* (hdc), второго же вовсе нет. Обычно в персональном компьютере присутствует два канала IDE, на каждом из которых можно разместить до двух дисков.

hda
: Первый диск на первом канале IDE (Primary master);

hdb
: Второй диск на первом канале IDE (Primary slave);

hdc
: Первый диск на втором канале IDE (Secondary master);

hdd
: Второй диск на втором канале IDE (Secondary slave).

Устройства SCSI/SATA

Другой распространённый способ подключения жёстких дисков — интерфейс SCSI (по-русски произносится как «скази»). В Linux SCSI-диски нумеруются буквами латинского алфавита (так же, как и IDE-диски), в зависимости от порядкового номера диска на шине SCSI: первый SCSI-диск называется sda (**s**csi **d**isk «**a**»), второй sdb и т. д.

Диски SATA и съёмные USB-устройства (USB флэш-карты, цифровые камеры и т. п.) обычно распознаются системой как SCSI-диски и, соответственно, обозначаются также sda, sdb и т. д. Аналогично через эмуляцию SCSI в Linux могут работать записывающие лазерные приводы (CD- и DVD-RW), они также получают имена, соответствующие SCSI-дискам, даже если в действительности подключены к шине IDE.

Имена устройств в Linux никогда не дублируются, в том числе при эмуляции: если соответствующее имя (например, sda) уже занято каким-то устройством, для вновь подключаемого устройства будет

выбрано следующее (первое свободное) имя (например, sdb).

Нумерация разделов

Каждый **раздел** на жёстком диске также получает собственное обозначение в Linux. Обозначение раздела складывается из названия соответствующего диска и *номера* этого раздела на диске. Например, первый раздел на первом жёстком диске IDE обозначается hda1.

В Linux принята следующая схема нумерации разделов: **основные разделы**, которых на диске может быть не более 4-х (см. Структура жёсткого диска), получают номера от 1 до 4 соответственно. Если основных разделов на диске меньше четырёх, то и номера отсутствующих разделов остаются незанятыми.

Номера, начиная с 5 получают **дополнительные** разделы, вложенные в **расширенный**. Так, номер 5 получает дополнительный раздел в первом расширенном, далее нумерация идет подряд — *вложенные* расширенные разделы не нумеруются.

Файловая система Linux

Операционные системы хранят данные на диске при помощи **файловых систем**. Классическая файловая система представляет данные в виде вложенных друг в друга **каталогов** (их ещё называют папками), в которых содержатся **файлы**[1]. Один из каталогов является «вершиной» файловой системы (а выражаясь технически — «корнем»[2]), в нём содержатся (или, если угодно, из него растут) все остальные каталоги и файлы.

Если жёсткий диск разбит на разделы, то на *каждом* разделе организуется отдельная файловая система с собственным корнем и структурой каталогов (ведь разделы полностью изолированы друг от друга).

В Linux корневой каталог называется весьма лаконично — “/”. Полные имена (пути) всех остальных каталогов получаются из “/”, к которому дописываются справа имена последовательно вложенных друг в друга каталогов. Имена каталогов в пути также разделяются символом “/” («слэш»). Например, запись /home обозначает каталог “home” в корневом каталоге (“/”), а /home/user — каталог “user” в каталоге “home” (который, в свою очередь, в корневом каталоге)[3]. Перечисленные таким образом каталоги, завершающиеся именем файла составляют **полный путь** к файлу.

Относительный путь строится точно так же, как и полный — перечислением через “/” всех названий каталогов, встретившихся при движении к искомому каталогу или файлу. Между полным путём и относительным есть только одно существенное различие: относительный путь начинается *от текущего каталога*, в то время как полный путь всегда начинается *от корневого каталога*. Относительный путь любого файла или каталога в файловой системе может иметь любую конфигурацию: чтобы добраться до искомого файла можно двигаться как по направлению к корневому каталогу, так и от него. Linux различает полный и относительный пути очень просто: если имя объекта *начинается* на “/” — это полный путь, в любом другом случае — относительный.

Монтирование

Корневой каталог в Linux всегда только *один*, а все остальные каталоги в него вложены, т. е. для пользователя файловая система представляет собой единое целое[4]. В действительности, разные части файловой системы могут находиться на совершенно разных устройствах: разных разделах жёсткого диска, на разнообразных съёмных носителях (лазерных дисках, дискетах, флэш-картах), даже на других компьютерах (с доступом через сеть). Для того, чтобы соорудить из этого хозяйства единое дерево с одним корнем, используется процедура **монтирования**.

Монтирование — это подключение в один из каталогов целой файловой системы, находящейся где-то на другом устройстве. Эту операцию можно представить как «прививание» ветки к дереву. Для монтирования необходим пустой каталог — он называется **точкой монтирования**. Точкой монтирования может служить любой каталог, никаких ограничений на этот счёт в Linux нет. При помощи специальной команды (mount) мы объявляем, что в данном *каталоге* (пока пустом) нужно отображать файловую систему, доступную на таком-то *устройстве* или же по сети. После этой операции в каталоге (точке монтирования) появятся все те файлы и каталоги, которые находятся на соответствующем устройстве. В результате пользователь может даже и не знать, на каком устройстве какие файлы располагаются.

Подключённую таким образом («смонтированную») файловую систему можно в любой момент отключить — **размонтировать** (для этого имеется специальная команда umount), после чего тот каталог, куда она была смонтирована, снова окажется пустым.

Для Linux самой важной является **корневая файловая система** (root filesystem). Именно к ней затем будут подключаться (монтироваться) все остальные файловые системы на других устройствах. Обратите внимание, что корневая файловая система тоже монтируется, но только не к другой файловой системе, а к «самой Linux», причём точкой монтирования служит “/” (корневой каталог). Поэтому при загрузке системы прежде всего монтируется корневая файловая система, а при останове она размонтируется

(в последнюю очередь).

Пользователю обычно не требуется выполнять монтирование и размонтирование вручную: при загрузке системы будут смонтированы все устройства, на которых хранятся части файловой системы, а при останове (перед выключением) системы все они будут размонтированы. Файловые системы на съёмных носителях (лазерных дисках, дискетах и пр.) также монтируются и размонтируются автоматически — либо при подключении носителя, либо при обращении к соответствующему каталогу.

Стандартные каталоги

В корневом каталоге Linux-системы обычно находятся только подкаталоги со *стандартными* именами. Более того, не только имена, но и *тип данных*, которые могут попасть в тот или иной каталог, также регламентированы стандартом[5]. Этот стандарт довольно последовательно соблюдается во всех Linux-системах: так, в любой Linux вы всегда найдёте каталоги /etc, /home, /usr/bin и т. п. и сможете довольно точно предсказать, что именно в них находится.

Стандартное размещение файлов позволяет и человеку, и даже программе предсказать, где находится тот или иной компонент системы. Для человека это означает, что он сможет быстро сориентироваться в любой системе Linux (где файловая система организована в соответствии со стандартом) и найти то, что ему нужно. Для программ стандартное расположение файлов — это возможность организации автоматического взаимодействия между разными компонентами системы.

Параметры монтирования

При выполнении операции монтирования, в том числе при выборе точки монтирования во время установки Linux-системы, можно изменять свойства смонтированной файловой системы. Для этого нужно указать утилите mount один или несколько параметров. Существует ряд параметров монтирования, поддерживаемых всеми файловыми системами. Есть параметры, характерные для одной конкретной файловой системы. Подробно о параметрах монтирования можно прочитать в руководстве к утилите mount (mount(8)).

---

1 - Файл — область данных, имеющая собственное имя.

2 - Такой каталог называют **корневым каталогом**, поскольку он служит корнем дерева файловой системе (в математическом смысле слов «дерево» и «корень»).

3 - Весьма похожий способ записи полного пути используется в системах DOS и Windows, с той разницей, что корневой каталог обозначается литерой устройства с последующим двоеточием, а в качестве разделителя используется символ “\” («обратный слэш»).

4 - Это отличается от технологии, применяемой в Windows или Amiga, где для каждого устройства, на котором есть файловая система, используется свой корневой каталог, обозначенный литерой, например “a”, “c”, “d” и т. д.

5 - Этот стандарт называется **Filesystem Hierarchy Standard** («стандартная структура файловых систем»). Стандарт **FHS** регламентирует не только перечисленные каталоги, но и их подкаталоги, а иногда даже приводит список конкретных файлов, которые должны присутствовать в определённых каталогах. Краткое описание стандартной иерархии каталогов Linux можно получить, отдав команду man hier. Полный текст и последнюю редакцию стандарта **FHS** можно найти в пакете fhs или прочесть по адресу http://www.pathname.com/fhs/.

Типы файловых систем

Существует довольно много разных файловых систем, которые отличаются друг от друга внутренним устройством, однако пользователь везде найдёт привычную структуру из вложенных каталогов и файлов. Файловые системы различаются скоростью доступа, надёжностью хранения данных, степенью устойчивости при сбоях, некоторыми дополнительными возможностями. Современные операционные системы поддерживают по несколько типов файловых систем (помимо файловых систем, используемых для хранения данных на жёстком диске, также файловые системы CD и DVD и пр.). Хотя для каждой операционной системы обычно есть одна «традиционная» файловая система, которая предлагается по умолчанию, является универсальной и подходит абсолютному большинству пользователей.

Важное свойство файловых систем — поддержка журналирования. **Журналируемая файловая система** ведёт постоянный учёт всех операций записи на диск. Благодаря этому после сбоя электропитания файловая система *всегда* автоматически возвращается в рабочее состояние.

Существует несколько типов файловых систем, которые в полной мере поддерживают все возможности, необходимые для полноценной работы Linux (все необходимые типы и атрибуты файлов, в том числе права доступа).

Ext2/3
: Этот тип файловой системы разработан специально для Linux и традиционно используется на большинстве Linux-систем. Фактически в названии «Ext2/3» объединены названия двух вариантов этой файловой системы. Ext3 отличается от Ext2 только поддержкой **журналирования**, в остальном они одинаковы и легко могут быть преобразованы одна в другую в любой момент без потери данных. Обычно предпочтителен вариант с журналированием (Ext3) в силу его большей надёжности. При высокой параллельной дисковой загрузке производительность Ext3 снижается, что выражается в снижении скорости операций с диском и повышении значения нагрузки на систему (Load Average).

ReiserFS
: Файловая система этого типа похожа скорее на базу данных: внутри неё используется своя собственная система индексации и быстрого поиска данных, а представление в виде файлов и каталогов — только одна из возможностей использования такой файловой системы. Традиционно считается, что ReiserFS отлично подходит для хранения огромного числа маленьких файлов. Поддерживает журналирование.

XFS
: Файловая система, наиболее подходящая для хранения очень больших файлов, в которых постоянно что-нибудь дописывается или изменяется. Поддерживает журналирование. Лишена недостатков Ext3 по производительности, но при её использовании выше риск потерять данные при сбоях питания (в том числе и по причине принудительного обнуления повреждённых блоков в целях безопасности; при этом метаданные файла обычно сохраняются и он выглядит как корректный). Рекомендуется использовать эту файловую систему с проверенным аппаратным обеспечением, подключенным к управляемому источнику бесперебойного питания (UPS).

SWAPFS
: Этот тип файловой системы находится на особом положении — он используется для организации на диске **области подкачки** (swap). Область подкачки используется в Linux для организации виртуальной памяти: когда программам недостаточно имеющейся в наличии оперативной памяти, часть рабочей информации временно размещается на жёстком диске.

JFS
: Разработана IBM для файловых серверов с высокой нагрузкой: при разработке особый упор делался на производительность и надёжность, что и было достигнуто. Поддерживает журналирование.

В Linux поддерживается, кроме собственных, немало форматов файловых систем, используемых другими ОС. Если способ записи на эти файловые системы *известен* и не слишком замысловат, то работает и запись, и чтение, в противном случае — только чтение (чего нередко бывает достаточно). Файловые системы перечисленных ниже типов обычно присутствуют на разделах диска, принадлежащих другим операционным системам.

FAT12/FAT16/FAT32
: Эти файловые системы используются в MS-DOS и разных версиях Windows, а также на многих съёмных носителях (в частности, на дискетах и USB-flash). Linux поддерживает чтение и запись на эти файловые системы.

NTFS

Файловая система NTFS изначально появилась в системах Windows NT, но может использоваться и другими версиями Windows (например, Windows 2000). В Linux NTFS поддерживается только на чтение.

Установка ALT Linux Desktop 4.0

Начало установки: загрузка системы

Иллюстрация 1. Загрузка

Загрузка с установочного диска начинается с меню, в котором перечислено несколько вариантов загрузки, причём установка системы — это только одна из возможностей. Из этого же меню можно запустить программу для восстановления системы или проверки памяти. Мышь на этом этапе установки не поддерживается, поэтому для выбора различных вариантов и опций установки необходимо воспользоваться клавиатурой. Можно получить справку по любому пункту меню, выбрав этот пункт и нажав F1. Кроме установки с диска доступно несколько вариантов сетевой установки (об этом рассказано ниже).

Нажатием F2 осуществляется выбор страны. От выбора страны в загрузчике зависит, во-первых, язык интерфейса загрузчика и программы установки, и во-вторых, какие языки будут доступны в списке языков установки — кроме основного для выбранной страны языка, в список будут включены и другие языки данной территории. По умолчанию предлагается «Россия». Если выбрать вариант «Прочие», то в списке языков установки будут перечислены все возможные языки.

По нажатию F3 открывается меню доступных видеорежимов (разрешений экрана). Это разрешение будет использоваться во время установки и загрузки установленной системы.

Чтобы начать процесс установки, нужно клавишами перемещеня курсора «вверх», «вниз» выбрать

пункт меню «Установка» и нажать Enter. В начальном загрузчике установлено небольшое время ожидания: если в этот момент не предпринимать никаких действий, то будет загружена та система, которая уже установлена на жестком диске. Если вы пропустили нужный момент, перезагрузите компьютер и вовремя выберите пункт «Установка».

Начальный этап установки не требует вмешательства пользователя: происходит автоматическое определение оборудования и запуск компонентов программы установки. Сообщения о том, что происходит на этом этапе, можно просмотреть нажав клавишу ESC.

Сетевая установка

Установка ALT Linux возможна не только с лазерного диска, ее можно производить и по сети. Обязательное условие для этого — наличие на сервере дерева файлов, аналогичного содержимому установочного диска, и внешний носитель с начальным загрузчиком. Таким носителем может являться, например, flash-накопитель, который можно сделать загрузочным, воспользовавшись утилитой mkbootflash.

Кнопка F4 позволяет выбрать источник сетевой установки: FTP, HTTP или NFS-сервер. Нужно указать имя или IP-адрес сервера и каталог (начиная с /), в котором размещен дистрибутив ALT Linux. В случае установки по протоколу FTP может понадобиться также ввести имя пользователя и пароль.

При сетевой установке может понадобиться определить параметры соединения с сервером, в этом случае на экране будут запросы, например, с предложением выбрать сетевую карту (если их несколько) или указать тип IP-адреса: статический (потребуется вписать его самостоятельно) или динамический (DHCP).

После успешного соединения с сервером в память компьютера будет загружен образ установочного диска, после чего начнется установка системы так же, как и при установке с лазерного диска.

До того, как будет произведена установка базовой системы на жёсткий диск, программа установки работает с образом системы, загруженном в оперативной памяти компьютера.

Выбор языка

Если инициализация оборудования завершилась успешно, будет запущен графический интерфейс программы-установщика, и откроется меню выбора **основного языка** — языка интерфейса программы установки и устанавливаемой системы. В списке, помимо доступных языков региона (выбранного на этапе начальной загрузки), указан и английский язык. Выберите язык из списка и нажмите «Применить» для перехода к началу установки.

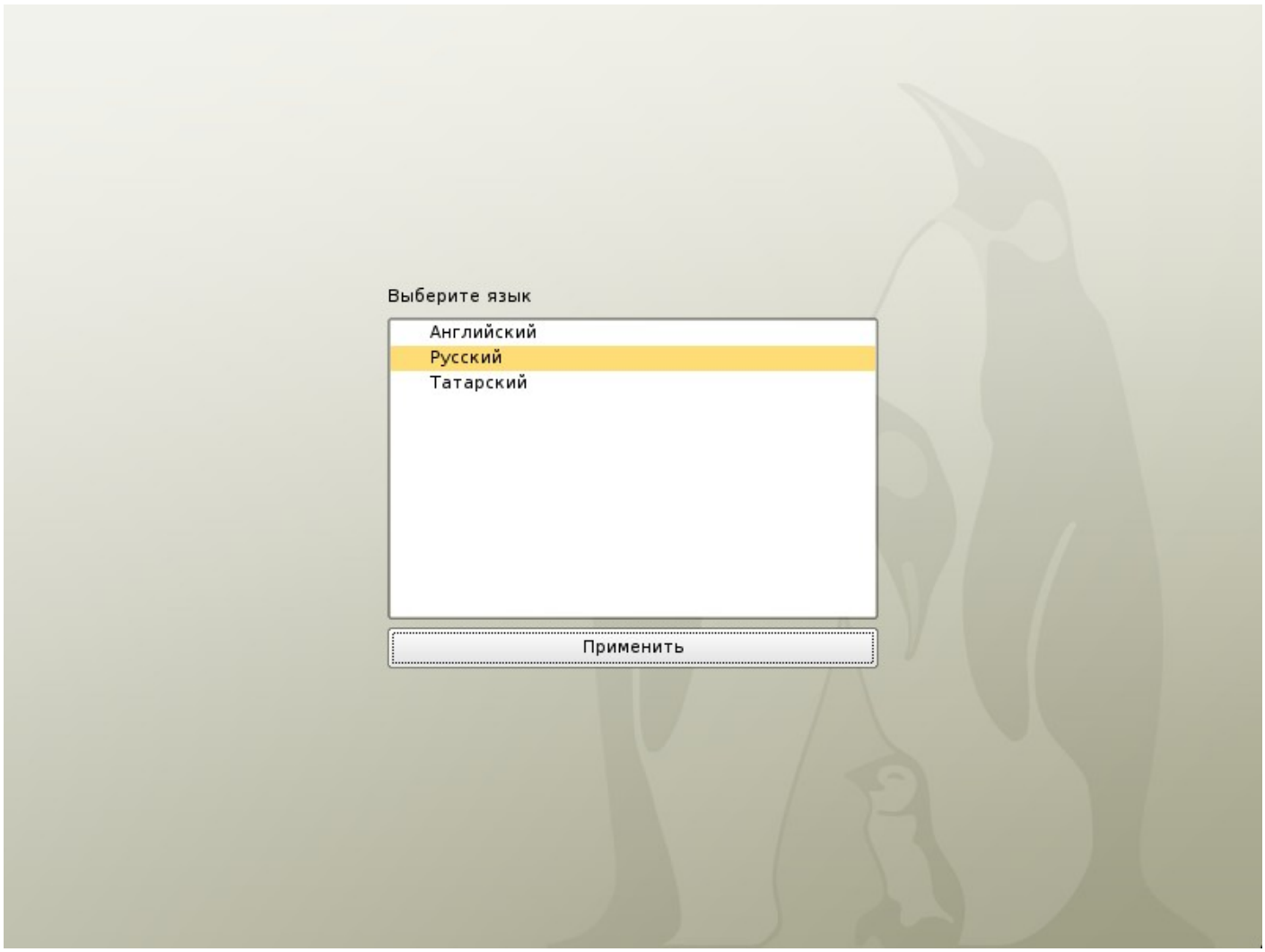


Иллюстрация 2. Выбор языка

Процесс установки разделен на шаги: каждый шаг посвящен настройке или установке определенного свойства системы. Шаги нужно проходить последовательно, переход к следующему шагу происходит по нажатию кнопки «Далее». При помощи кнопки «Назад» при необходимости можно вернуться к уже пройденному шагу и изменить настройки. Однако на этом этапе установки возможность перехода к предыдущему шагу ограничена теми шагами, где нет зависимости от данных, введённых ранее.

Если по каким-то причинам возникла необходимость прекратить установку, нажмите Reset на системном блоке компьютера. Помните, что совершенно *безопасно* прекращать установку только до шага «Подготовка диска», поскольку до этого момента не производится никаких изменений на жёстком диске. Если прервать установку между шагами «Подготовка диска» и «Установка загрузчика», вероятно,

что после этого с жёсткого диска не сможет загрузиться ни одна из установленных систем.

Технические сведения о ходе установки можно посмотреть, нажав *Ctrl+Alt+F1*, вернуться к программе установки — *Ctrl+Alt+F7*. По нажатию *Ctrl+Alt+F2* откроется отладочная виртуальная консоль.

Во время установки системы выполняются следующие шаги:

1. Лицензионное соглашение
2. Настройка клавиатуры
3. Часовой пояс
4. Дата/Время
5. Подготовка диска
6. Установка базовой системы
7. Установка загрузчика
8. Администратор системы
9. Пользователь
10. Дополнительные пакеты
11. Настройка сети
12. Настройка графической системы
13. Завершение установки

Лицензионное соглашение

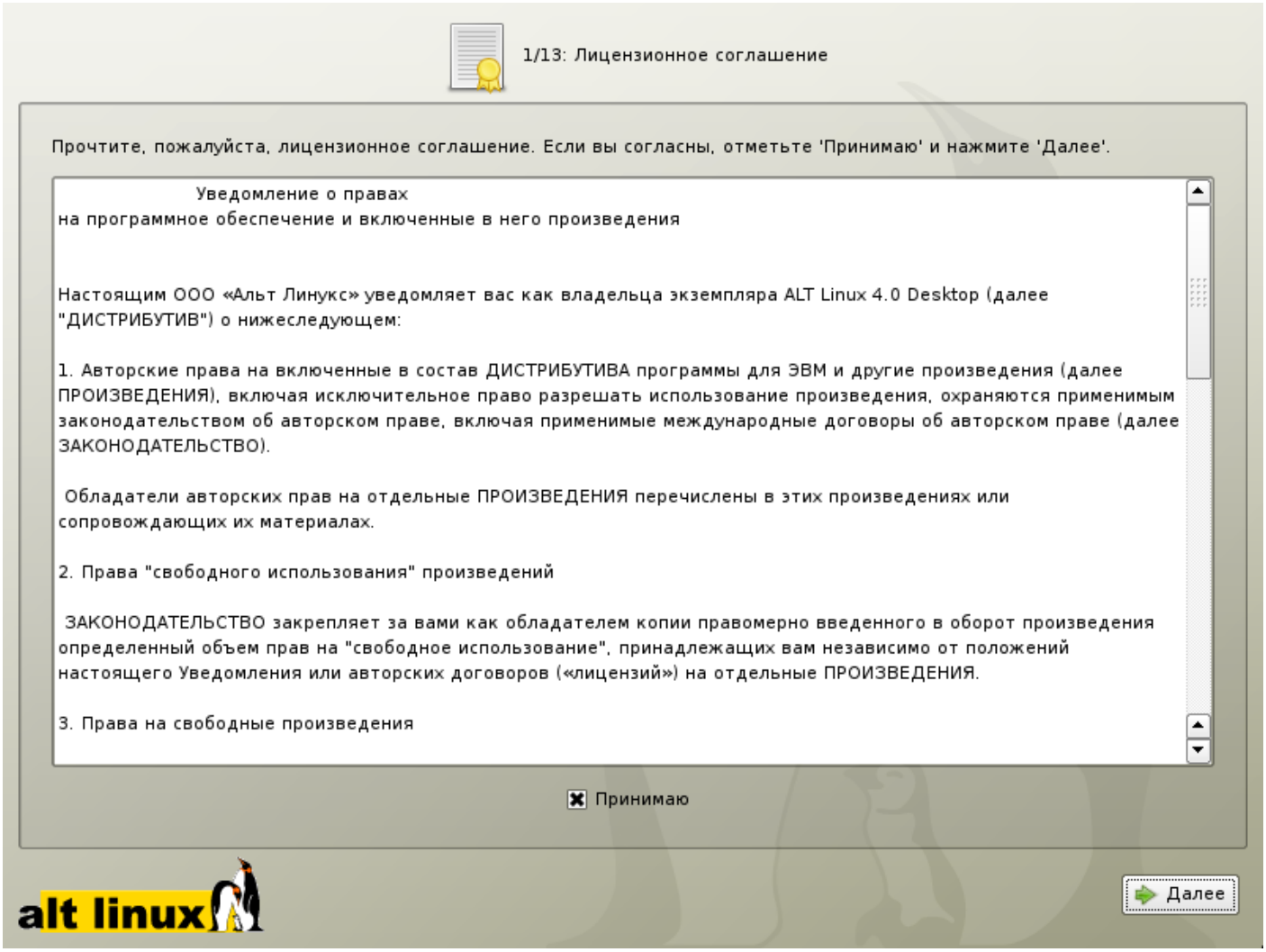


Иллюстрация 3. Лицензионное соглашение

Перед продолжением установки следует внимательно прочитать условия лицензии. В лицензии говорится о ваших правах. В частности, за вами закрепляются права на:

- эксплуатацию программ на любом количестве компьютеров и в любых целях;
- распостранение программ (сопровождая их копией авторского договора);
- получение исходных текстов программ.

Если вы прибрели дистрибутив, то данное лицензионное соглашение прилагается в печатном виде к вашей копии дистрибутива. Лицензия относится ко всему дистрибутиву ALT Linux. Если вы согласны с условиями лицензии, отметьте пункт «Принимаю» и нажмите «Далее».

Настройка клавиатуры

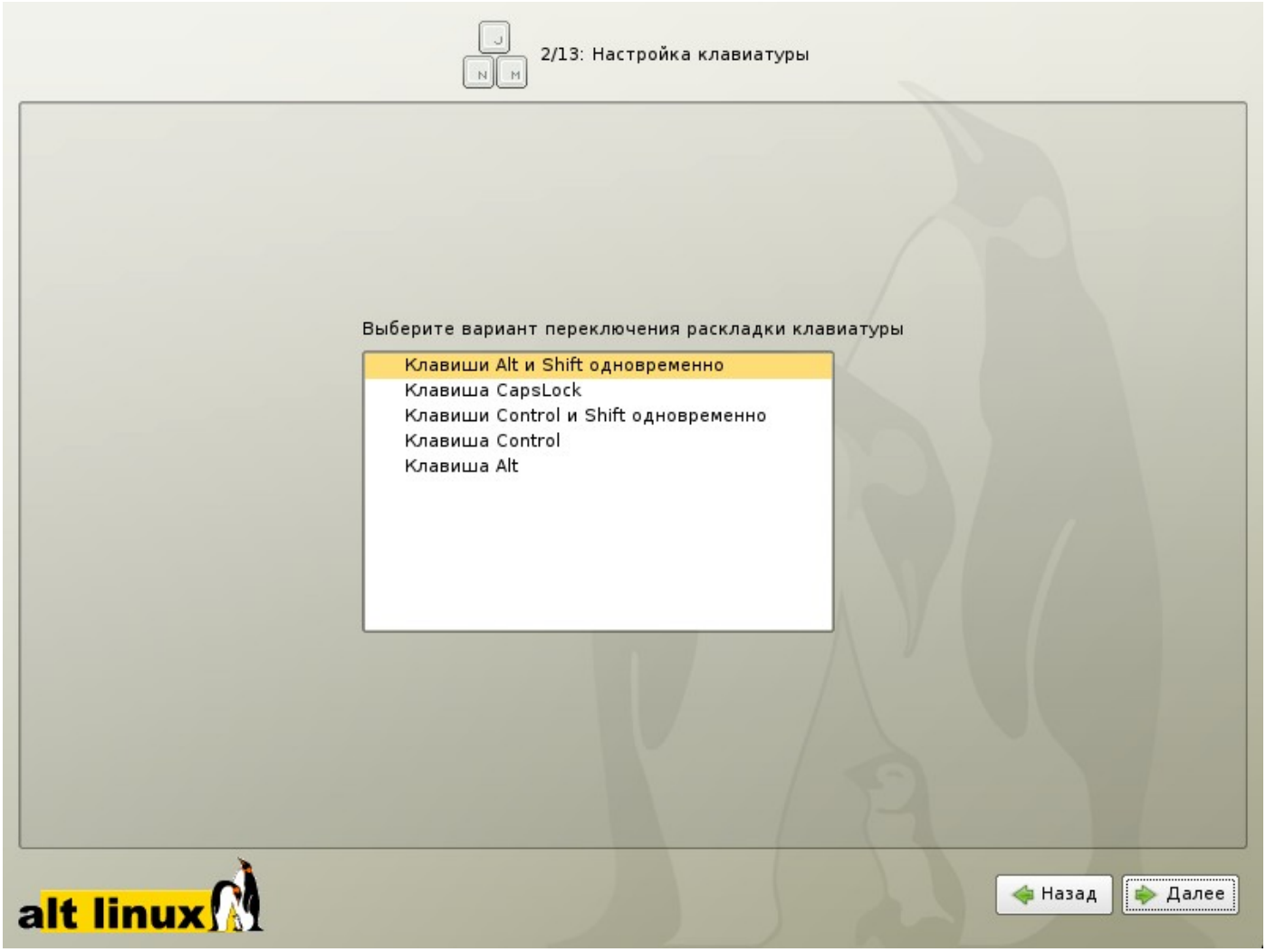


Иллюстрация 4. Настройка клавиатуры

Раскладка клавиатуры — это привязка букв, цифр и специальных символов к клавишам на клавиатуре. Помимо ввода символов на основном языке, в любой системе Linux необходимо иметь возможность вводить латинские символы (имена команд, файлов и т. п.), для чего обычно используется стандартная английская раскладка клавиатуры. Переключение между раскладками осуществляется при помощи специально зарезервированных для этого клавиш. Для русского языка доступны следующие варианты переключения раскладки:

- Клавиши *Alt* и *Shift* одновременно
- Клавиша *Capslock*
- Клавиши *Control* и *Shift* одновременно
- Клавиша *Control*
- Клавиша *Alt*

В случае, если выбранный основной язык имеет всего одну раскладку (например, при выборе английского языка в качестве основного), эта единственная раскладка будет принята автоматически, а сам шаг не будет отображен в интерфейсе.

Часовой пояс

Для корректной установки даты и времени достаточно правильно указать часовой пояс и выставить желаемые значения для даты и времени.

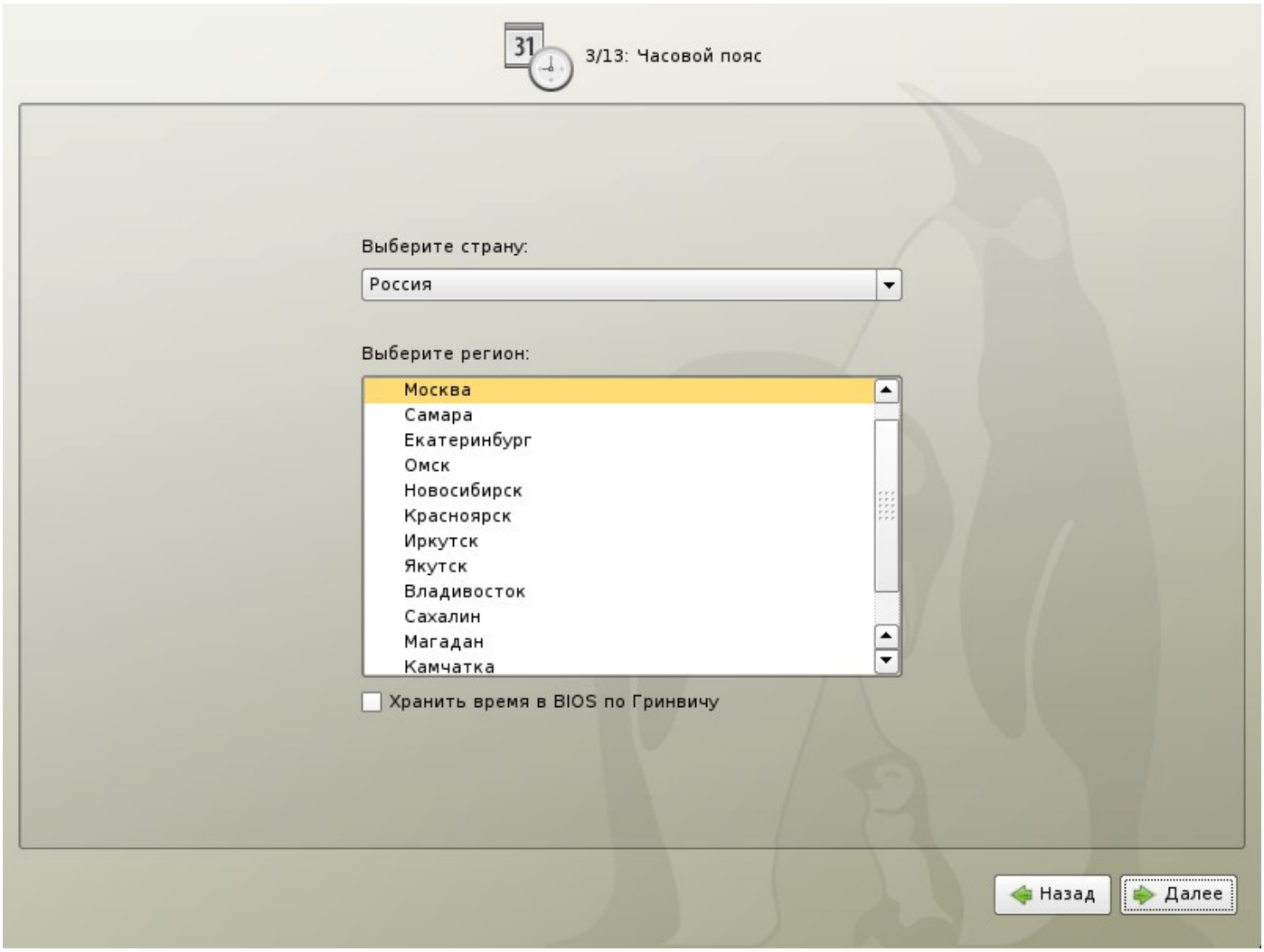


Иллюстрация 5. Часовой пояс

На этом шаге следует выбрать часовой пояс, по которому нужно установить часы. Для этого в соответствующих списках выберите страну, а затем регион. Поиск по списку можно ускорить, набирая на клавиатуре первые буквы искомого слова.

Обратите внимание на отметку «Хранить время в BIOS по Гринвичу». В системных часах BIOS желательно устанавливать не локальное, а универсальное время по Гринвичу (GMT). При этом программные часы будут показывать локальное время в соответствии с выбранным часовым поясом, и системе не потребуется изменять настройки BIOS при сезонном переводе часов и смене часового пояса. Однако если вы планируете на этом же компьютере использовать другие операционные системы, отметку нужно снять, иначе при загрузке в другую операционную систему время может сбиваться.

Дата/Время

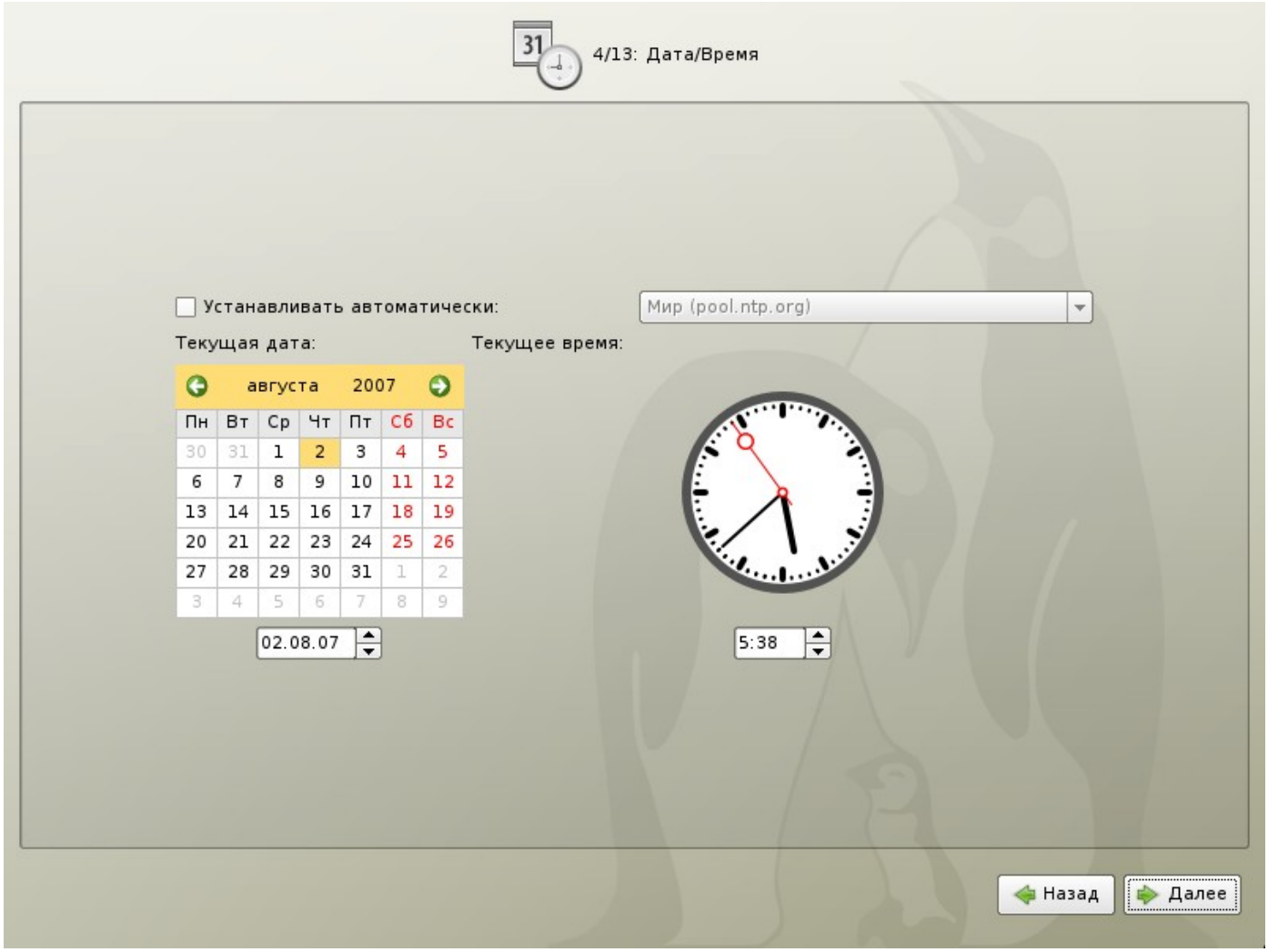


Иллюстрация 6. Дата/Время

Программа установки полагает, что системные часы (BIOS) отображают локальное время. Поэтому часы на этом шаге показывают либо время, соответствующее вашим системным часам, либо, если в предыдущем шаге была выставлена отметка «Хранить время в BIOS по Гринвичу», время, соответствующее, GMT с учётом вашего часового пояса. Это значит, что, если системные часы отображают локальное время, а вы всё же выставили отметку «Хранить время в BIOS по Гринвичу» в предыдущем шаге, то часы будут отображать неверное время.

Проверьте, верно ли отображаются дата и время, и, при необходимости, выставьте правильные значения.

Если ваш компьютер подключён к локальной сети или к Интернет, можно включить синхронизацию системных часов (NTP) с удалённым сервером, для этого достаточно отметить пункт «Устанавливать автоматически» и выбрать из списка NTP-сервер.

Подготовка диска

Переход к этому шагу может занять некоторое время. Время ожидания может быть разным и зависит от производительности компьютера, объема жесткого диска, количества разделов на нем и т. д.

На этом этапе подготавливается площадка для установки ALT Linux, в первую очередь — выделяется свободное место на диске. Для установки с выбором одного из автоматических профилей разметки потребуется не менее 8 Гб плюс удвоенный объём оперативной памяти на одном или нескольких жёстких дисках компьютера. К примеру, если ваш компьютер имеет 512 Мб оперативной памяти, то для применения профиля автоматической разметки потребуется около 9 Гб дискового пространства. При подготовке разделов вручную необходимо выделить как минимум 2 Гб.

Выбор профиля разбиения диска

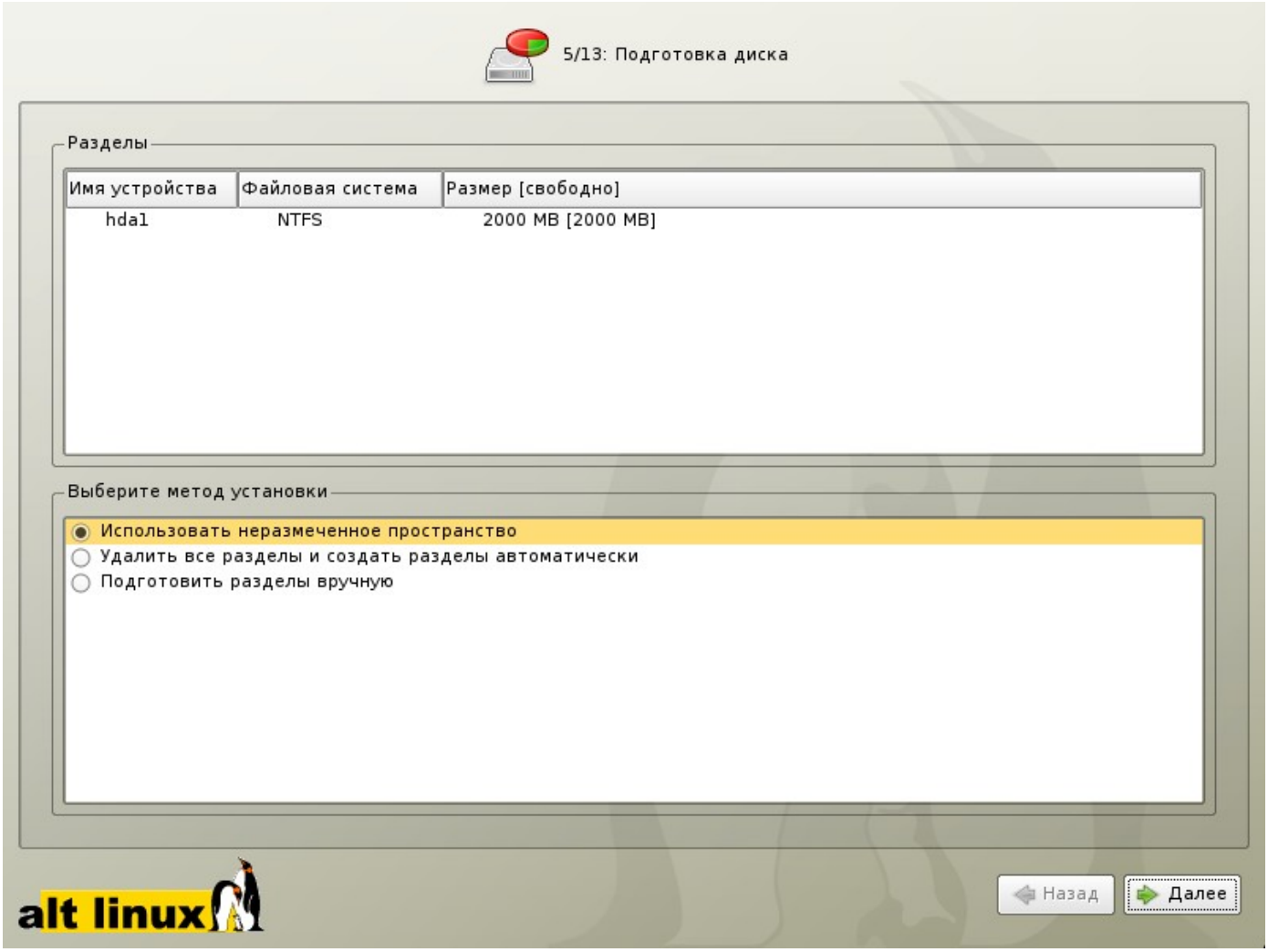


Иллюстрация 7. Выбор профиля разбиения диска

В списке разделов перечислены уже существующие на жёстких дисках разделы (в том числе здесь могут оказаться съёмные USB-носители, подключённые к компьютеру в момент установки). Узнать, каким устройствам вашего компьютера соответствуют названия в списке, можно в разделе Именование дисков и разделов в Linux. Ниже перечислены доступные профили разбиения диска. Профиль — это шаблон распределения места на диске для установки Linux. Можно выбрать один из трех профилей:

- Использовать неразмеченное пространство
- Удалить все разделы и создать разделы автоматически
- Подготовить разделы вручную

Первые два профиля предполагают автоматическое разбиение диска. Они ориентированы на среднестатистические рабочие станции и должны подойти для большинства пользователей.

Автоматические профили разбиения диска

Применение профилей автоматического разбиения происходит сразу по нажатию «Далее», после чего непосредственно начинается этап установки базовой системы

Если для применения одного из профилей автоматической разметки доступного места окажется недостаточно, будет выведено сообщение об ошибке: «Невозможно применить профиль, недостаточно места на диске».

Если данное сообщение появилось после попытки применить профиль «Использовать неразмеченное пространство», то вы можете очистить место, удалив данные, которые уже есть на диске. Выберите пункт «Удалить все разделы и создать разделы автоматически». При применении этого профиля сообщение о недостатке места связано с недостаточным объёмом всего жёсткого диска, на который производится установка. В этом случае необходимо воспользоваться режимом ручной разметки: профиль «Подготовить разделы вручную».

Будьте осторожны при применении профиля «Удалить все разделы и создать разделы автоматически»! В этом случае будут удалены **все** данные со **всех** дисков без возможности восстановления. Рекомендуется использовать эту возможность только в том случае, если вы уверены, что диски не содержат **никаких ценных данных**.

Ручной профиль разбиения диска

При необходимости освободить **часть** дискового пространства следует воспользоваться профилем разбиения вручную. Вы сможете удалить некоторые из существующих разделов или содержащиеся в них файловые системы. После этого можно создать необходимые разделы самостоятельно или вернуться к шагу выбора профиля и применить один из автоматических профилей. Выбор этой возможности требует знаний об устройстве диска и технологиях его разбиения, поэтому сначала рекомендуется внимательно прочитать главу Планирование диска данного руководства, там же разобрано несколько типичных способов разбиения диска.

Необходимую информацию о работе с диском и принципах ручного разбиения можно найти в разделе #Разбиение диска средствами программы установки.

По нажатию «Далее» будет произведена запись новой таблицы разделов на диск и форматирование разделов. Разделы, только что созданные на диске программой установки, пока не содержат данных и поэтому форматируются без предупреждения. Уже существовавшие, но измененные разделы, которые будут отформатированы, помечаются специальным значком в колонке «Файловая система» слева от названия. Если вы уверены в том, что подготовка диска завершена, подтвердите переход к следующему шагу нажатием кнопки «OK».

Не следует форматировать разделы с теми данными, которые вы хотите сохранить, например, с пользовательскими данными (/home) или с другими операционными системами. С другой стороны, отформатировать можно любой раздел, который вы хотите «очистить» (т. е. удалить все данные).

Установка базовой системы

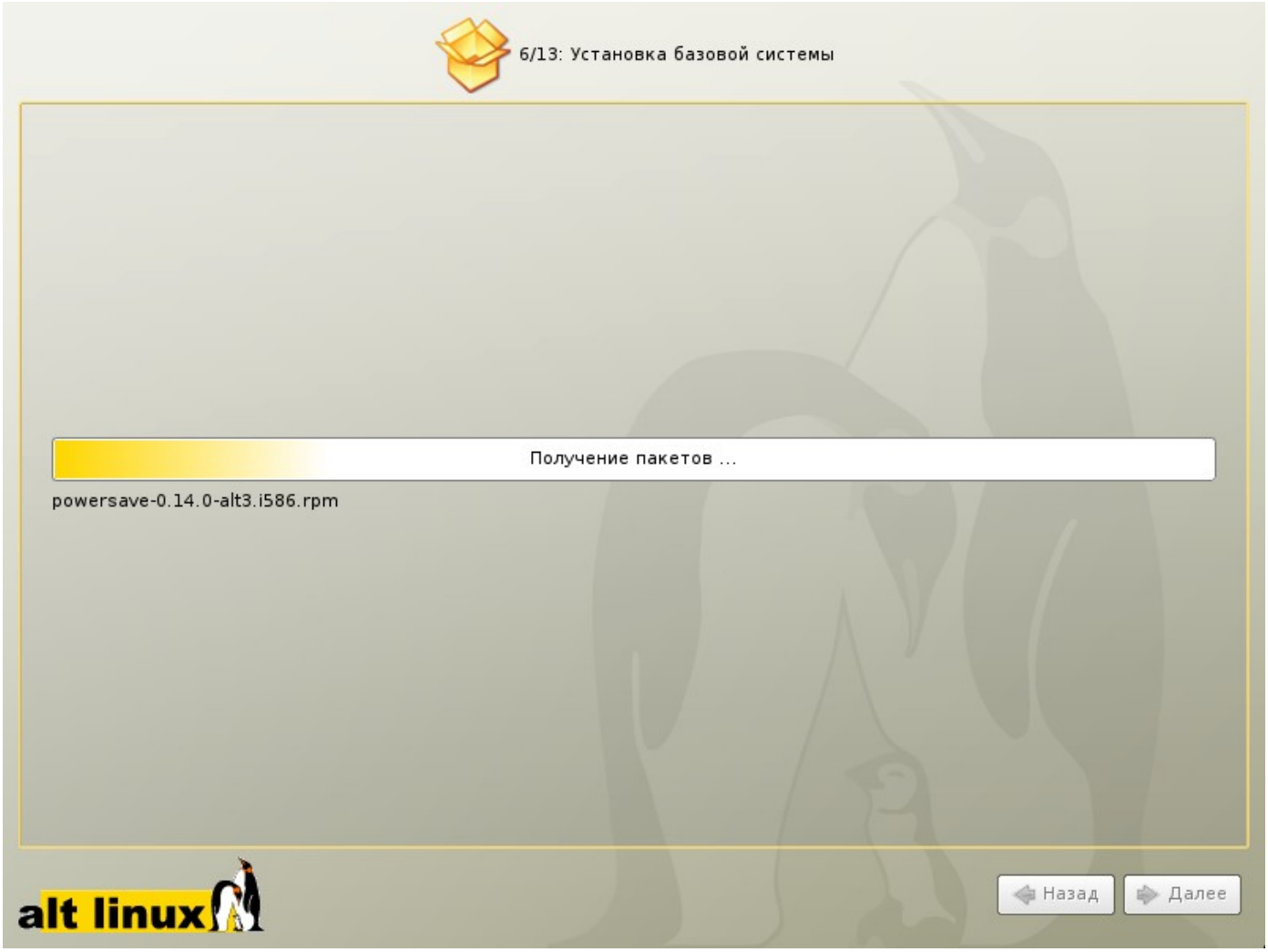


Иллюстрация 8. Установка базовой системы

На этом этапе происходит установка стартового набора программ, необходимых для запуска и первоначальной настройки Linux. Далее в процессе установки у вас будет возможность выбрать и установить все необходимые вам для работы приложения.

Установка происходит автоматически в два этапа:

- Получение пакетов
- Установка пакетов

Получение пакетов осуществляется с источника, выбранного на этапе начальной загрузки. При сетевой установке (по протоколу FTP или HTTP) время выполнения этого шага будет зависеть от скорости соединения и может быть значительно большим, чем при установке с лазерного диска.

Когда базовая система будет установлена, вы сможете произвести первичную настройку, в частности — настроить сетевое оборудование и сетевые подключения. Изменить свойства системы, которые были заданы при установке (например, язык системы), можно будет в любой момент как через веб-интерфейс, так и при помощи стандартных для Linux средств и специализированных модулей управления, включённых в дистрибутив.

Установка базовой системы может занять некоторое время, которое можно посвятить, например, чтению руководства.

Установка загрузчика

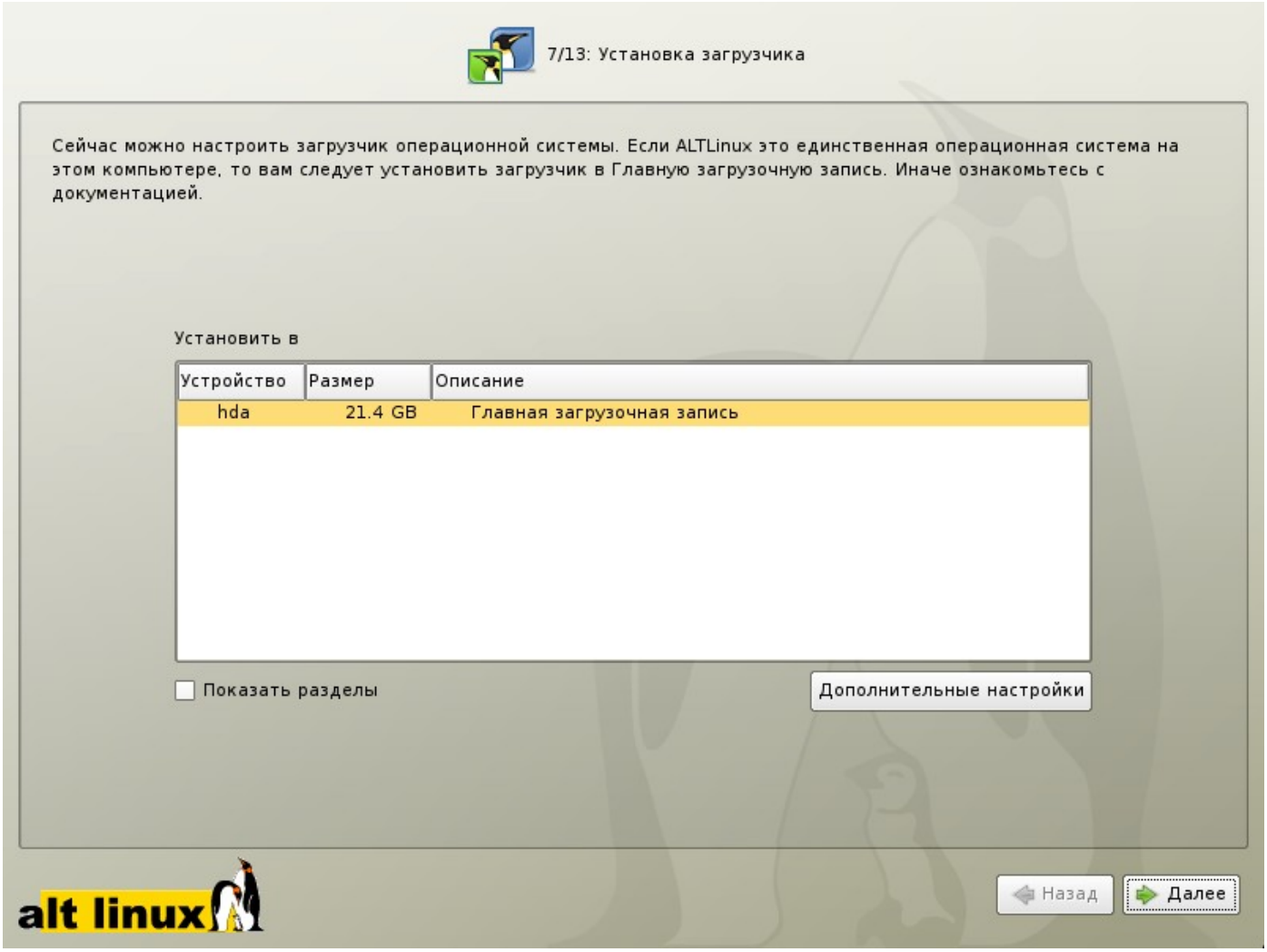


Иллюстрация 9. Установка загрузчика

Начиная с этого шага программа установки работает с файлами только что установленной базовой системы. Все последующие изменения можно будет совершить после завершения установки посредством редактирования соответствующих конфигурационных файлов.

Загрузчик Linux — программа, которая позволяет загружать Linux и другие операционные системы. Если на вашем компьютере будет установлен только Linux, то здесь не нужно ничего изменять, просто нажмите «Далее».

Если же вы планируете использовать и другие операционные системы, уже установленные на этом компьютере, тогда имеет значение, на каком жёстком диске или разделе будет расположен загрузчик. В большинстве случаев программа установки правильно подберёт расположение загрузчика, однако чтобы быть уверенным, что все операционные системы будут загружаться правильно, обратитесь к разделу "Настройка загрузки".

Опытным пользователям может пригодиться возможность тонкой настройки загрузчика (кнопка «Дополнительные настройки»). Параметры, которые можно здесь изменять, напрямую соотносятся с соответствующими параметрами конфигурационного файла загрузчика LILO (/etc/lilo.conf). Для простоты сохранены латинские названия параметров, об их значении можно справиться в документации по LILO (lilo.conf(5)).

Пользователи

Linux — это многопользовательская система. На практике это означает, что для работы в системе нужно в ней *зарегистрироваться*, т. е. дать понять системе, кто именно находится за монитором и клавиатурой. Наиболее распространенный способ регистрации на сегодняшний день — использование **системных имен** (login name) и паролей. Это надежное средство убедиться, что с системой работает тот, кто нужно, если пользователи хранят свои пароли в секрете и если пароль достаточно сложен и не слишком короток (иначе его легко угадать или подобрать).

Администратор системы

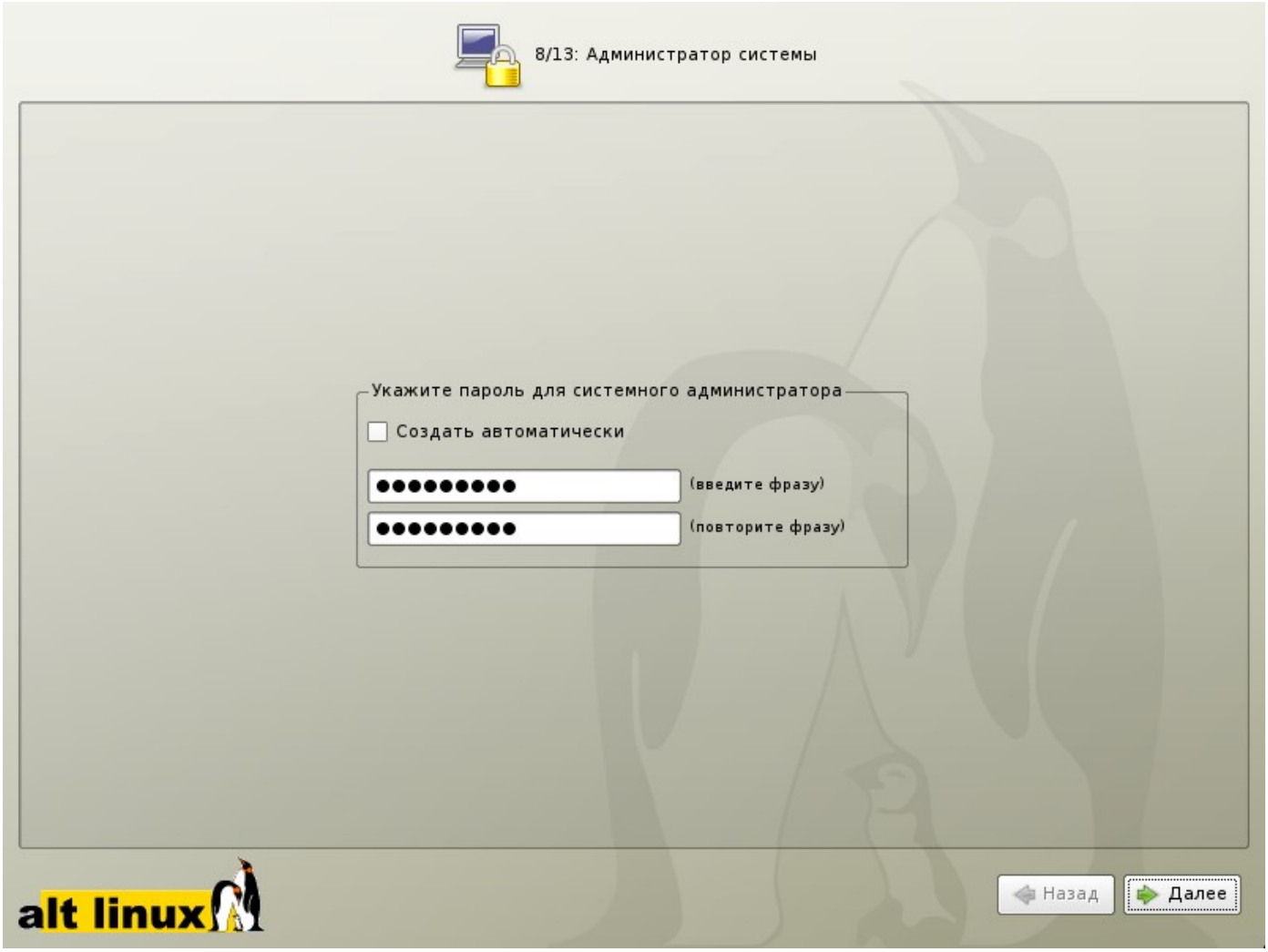


Иллюстрация 10. Администратор системы

В любой системе Linux всегда присутствует один специальный пользователь — администратор, он же **суперпользователь**, для него зарезервировано стандартное системное имя — root.

Стоит запомнить пароль root — его нужно будет вводить, чтобы получить право изменять настройки системы с помощью стандартных средств настройки ALT Linux.

При наборе пароля вместо символов на экране высвечиваются звёздочки. Чтобы избежать опечатки при вводе пароля, его предлагается ввести дважды. Можно воспользоваться автоматическим созданием пароля, выбрав «Создать автоматически». Вам будет предложен случайно сгенерированный и достаточно надёжный вариант пароля. Можно принять автоматически сгенерированный пароль, (не забудьте при этом запомнить пароль!), или запросить другой вариант пароля при помощи кнопки «Сгенерировать».

Администратор отличается от всех прочих пользователей тем, что ему позволено производить **любые**, в том числе самые разрушительные, изменения в системе. Поэтому выбор пароля администратора — очень важный момент для **безопасности**: любой, кто сможет ввести его правильно (узнать или подобрать), получит неограниченный доступ к системе. Даже ваши собственные неосторожные действия от имени root могут иметь катастрофические последствия для всей системы.

Пользователь

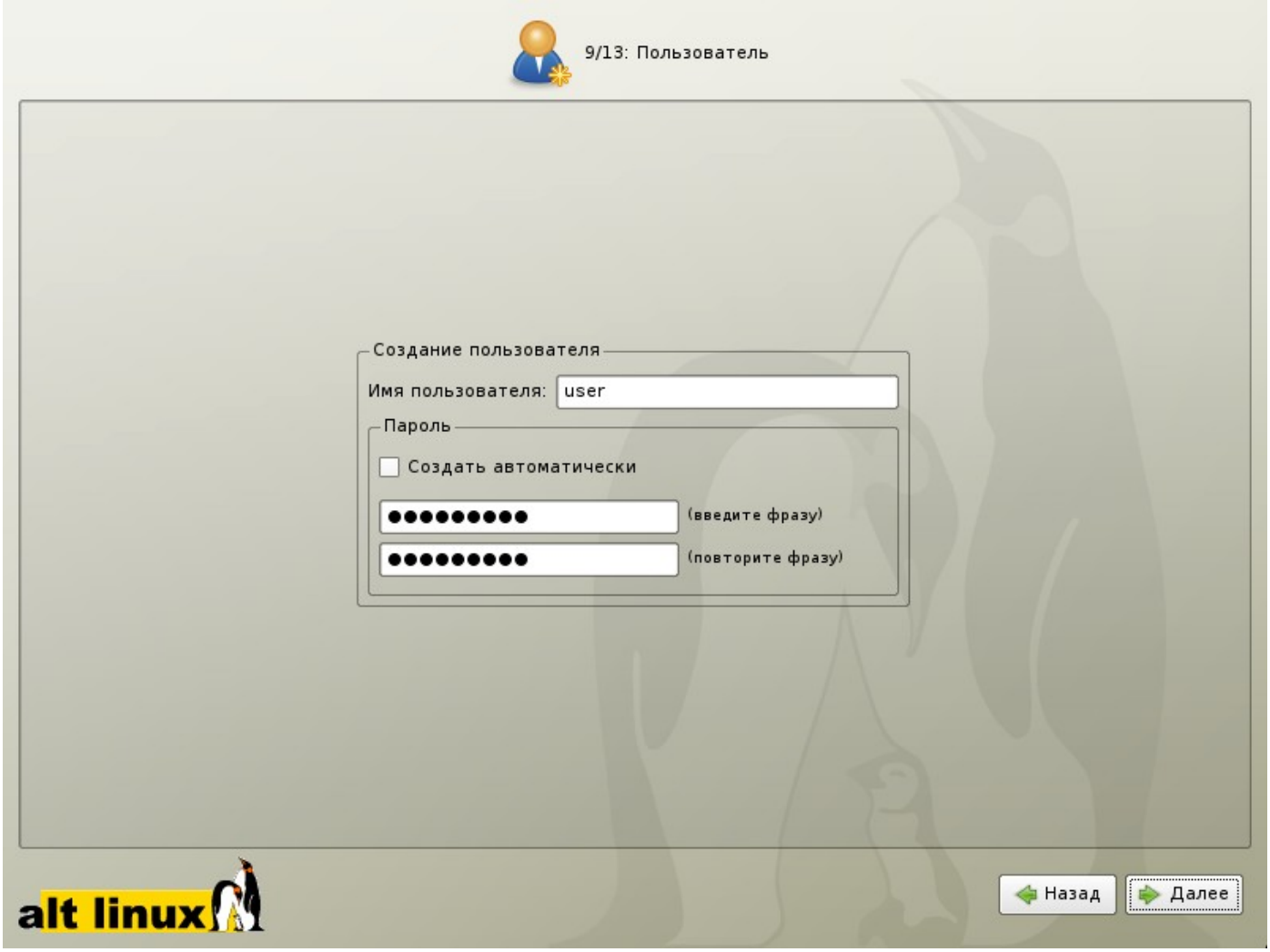


Иллюстрация 11. Пользователь

Помимо администратора (root) в систему необходимо добавить по меньшей мере одного **обычного пользователя**. Работа от имени администратора считается опасной (можно по неосторожности повредить систему), поэтому повседневную работу в Linux следует выполнять от имени обычного пользователя, полномочия которого ограничены.

При добавлении пользователя предлагается ввести имя учётной записи (login name) пользователя. Имя учётной записи всегда представляет собой одно слово, состоящее только из строчных латинских букв (заглавные запрещены), цифр и символа подчёркивания “_” (причём цифра и символ “_” не могут стоять в начале слова). Чтобы исключить опечатки, пароль пользователя вводится дважды. Так же, как при выборе пароля администратора (root), можно создать пароль автоматически.

В процессе установки предлагается создать только одну учётную запись обычного пользователя — чтобы от его имени системный администратор мог выполнять задачи, которые не требуют привилегий суперпользователя.

Учётные записи для всех прочих пользователей системы можно будет создать в любой момент после её установки.

Дополнительные пакеты

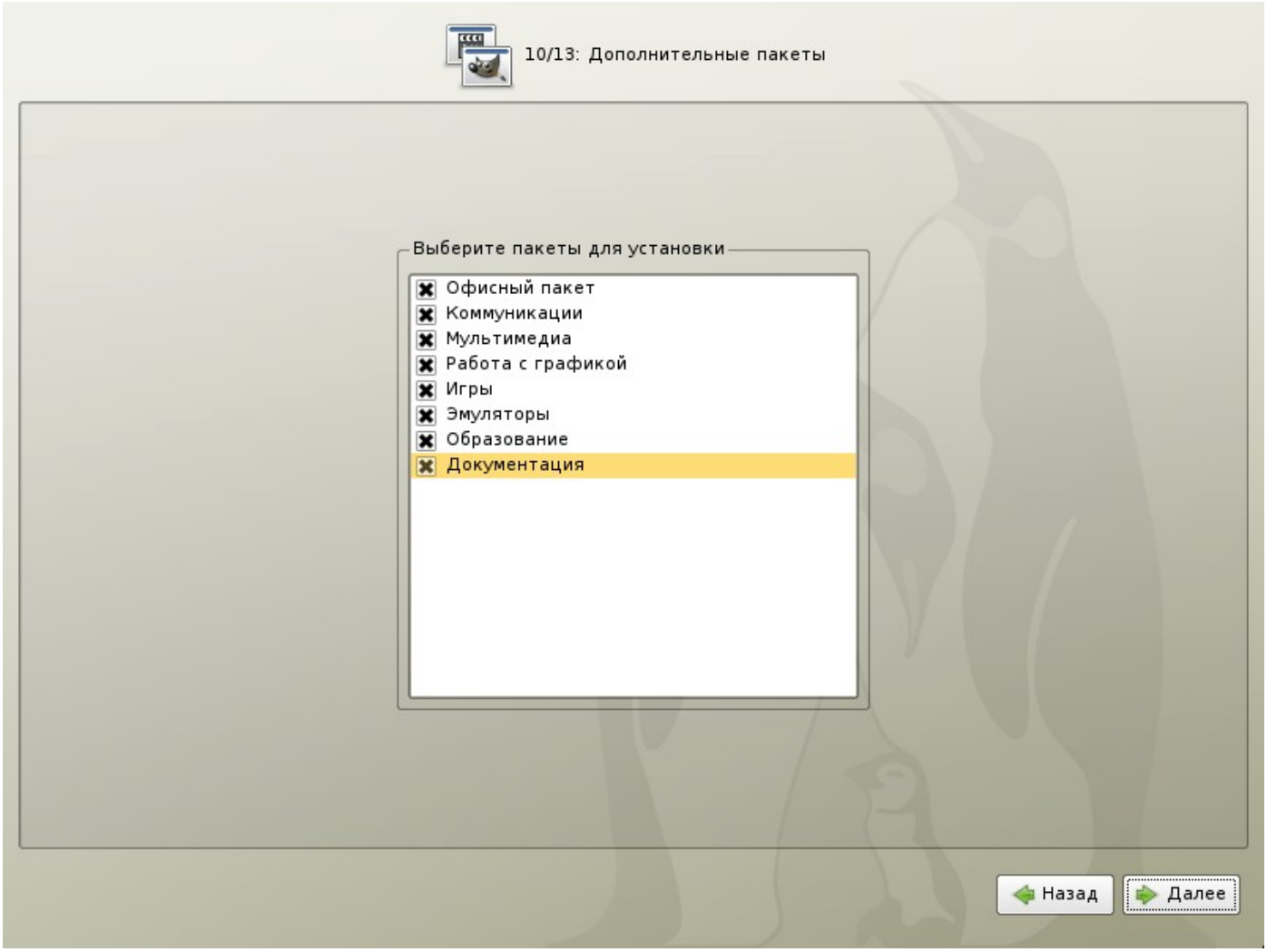


Иллюстрация 12. Дополнительные пакеты

В любом дистрибутиве ALT Linux доступно значительное количество программ (до нескольких тысяч), часть из которых составляет саму операционную систему, а все остальные — это прикладные программы и утилиты.

В операционной системе Linux все операции установки и удаления производятся над **пакетами** — отдельными компонентами системы. Пакет и программа соотносятся неоднозначно: иногда одна программа состоит из нескольких пакетов, иногда один пакет включает несколько программ.

В процессе установки системы обычно не требуется детализированный выбор компонентов на уровне пакетов — это требует слишком много времени и знаний от проводящего установку. Тем более, что комплектация дистрибутива подбирается таким образом, чтобы из имеющихся программ можно было составить полноценную рабочую среду для соответствующей аудитории пользователей. Поэтому в процессе установки системы пользователю предлагается выбрать из небольшого списка *групп пакетов*, объединяющих пакеты, необходимые для решения наиболее распространённых задач.

Выбрав необходимые группы, следует нажать «Далее». На экране появится информация о количестве устанавливаемых пакетов и объеме дискового пространства, который будет занят после их установки. По нажатию кнопки «Подробности...» откроется список, в котором перечислены конкретные пакеты, которые будут установлены в системе. Если вы готовы подтвердить изменения, нажмите «Да» — сразу после этого начнется установка пакетов. Чтобы вернуться к списку групп пакетов и отредактировать свой первоначальный выбор, нажмите «Нет».

Настройка сети

Существует ряд сетевых параметров, которые являются общими для всех подключений к сети и должны быть определены даже тогда, когда компьютер не подключён ни к какой сети. Для подключения к локальной сети необходимо к тому же настроить **сетевое подключение**, которое в Linux также принято называть **сетевой интерфейс**. Настройки этих двух типов производятся в закладках «Общие сетевые настройки» и «IP-интерфейсы».

IP-интерфейсы

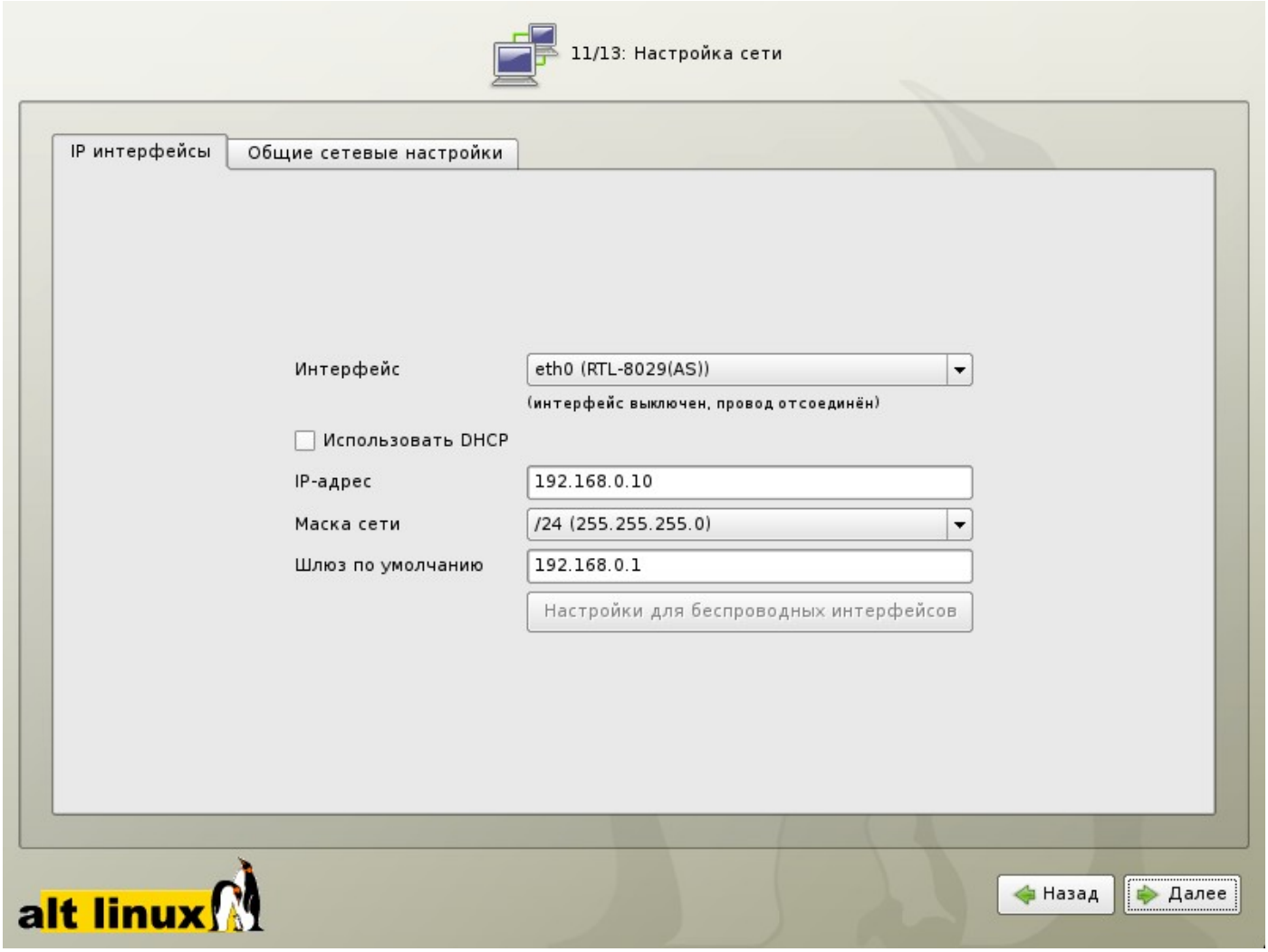


Иллюстрация 13. IP-интерфейсы

В случае локальной сети каждое подключение обычно привязывается к определённому физическому устройству — сетевой (Ethernet) карте. Чтобы настроить подключение, на закладке «Интерфейс» выберите из списка одно из обнаруженных сетевых устройств. При наличии беспроводных интерфейсов их настройки осуществляются по нажатию кнопки «Настройки для беспроводных интерфейсов».

В полях «IP-адрес» и «Маска сети» должны быть указаны обязательные параметры каждого узла IP-сети. Первый параметр — уникальный идентификатор машины, от второго напрямую зависит, к каким машинам локальной сети данная машина будет иметь доступ. Если требуется выход во внешнюю сеть, не забудьте про параметр «Шлюз по умолчанию».

При наличии DHCP-сервера все вышеперечисленные параметры можно получить автоматически, для этого отметьте пункт «Использовать DHCP». Если DHCP-сервера нет, потребуется указать IP-адрес и сетевую маску явно. Параметры настройки сетевого подключения можно узнать у администратора сети.

Общие настройки сети

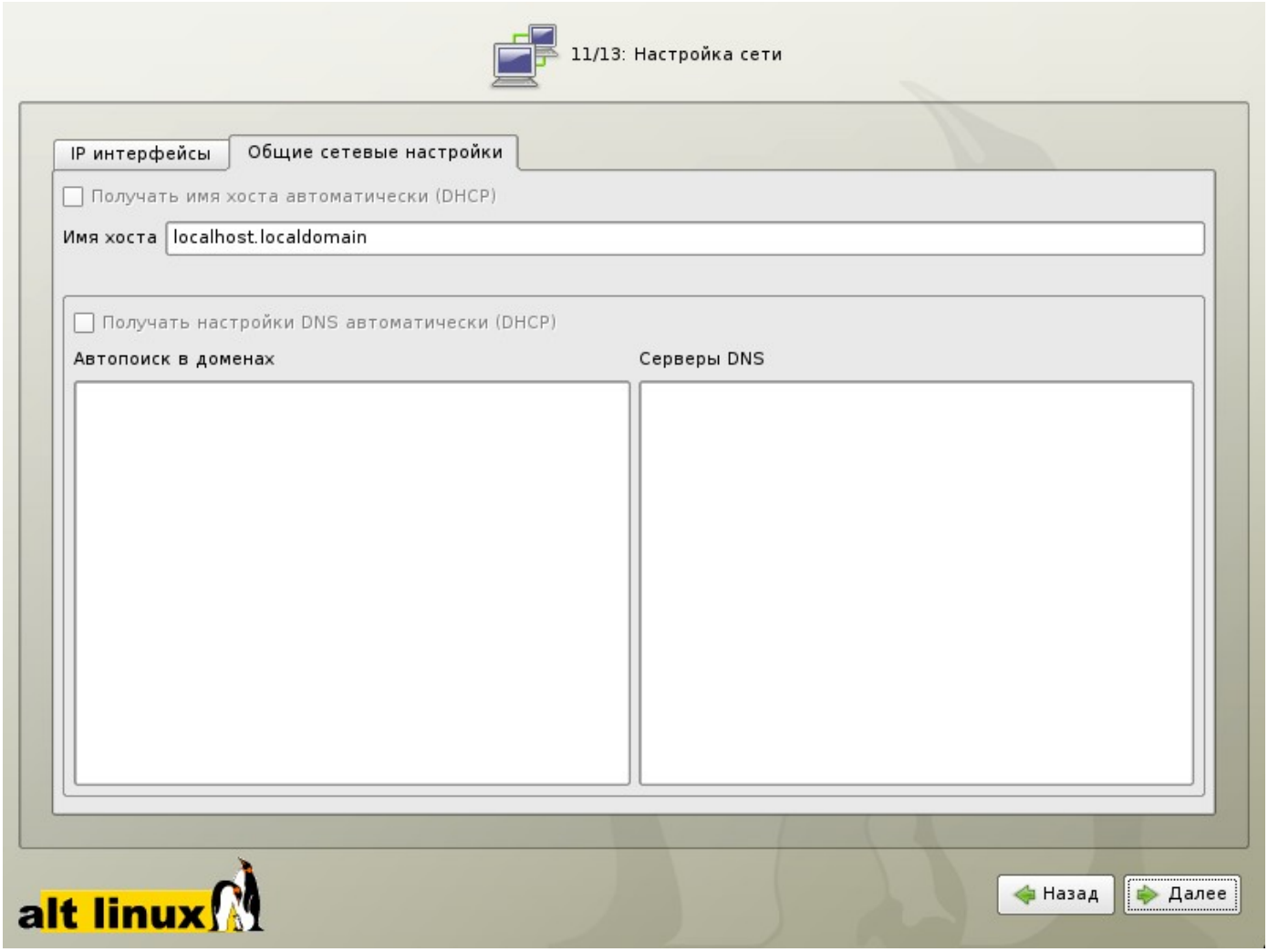


Иллюстрация 14. Общие настройки сети

Укажите **Имя хоста** (host name) — имя компьютера вида computer.domain. Несмотря на то, что этот параметр никому из соседних компьютеров в сети не передаётся (в отличие, скажем, от имени компьютера в Windows-сети), его используют многие сетевые службы, например, почтовый сервер. Если компьютер не подключён к локальной сети, имя хоста может выглядеть как угодно, можно оставить значение по умолчанию (localhost.localdomain).

При работе и настройке сетевых служб часто приходится использовать символьные имена других машин в сети. Чтобы система преобразовала их в IP-адреса, используются DNS-серверы. В случае локальной сети преобразования осуществляет локальный DNS-сервер. за пределами локальной сети используются DNS-серверы вышестоящих Интернет-провайдеров. Все необходимые серверы DNS перечисляются в соответствующем поле (по одному на строку или через запятую).

Если в поле «Автопоиск в доменах» перечислить наиболее часто используемые домены (например, domain), то можно пользоваться неполными именами машин (computer вместо computer.domain)

В случае наличия в сети DHCP-сервера общие настройки сети могут быть получены автоматически. Отметьте пункты «Получать имя хоста автоматически (DHCP)» и «Получать настройки DNS автоматически (DHCP)», если хотите воспользоваться этой возможностью. Обратите внимание, что для этого необходимо включить использование DHCP в настройках IP-интерфейса. Если сервера DHCP нет, при подключении к сети все необходимые параметры (имя хоста, шлюз по умолчанию, адреса серверов DNS) нужно выяснить у администратора сети или у Интернет-провайдера и вписать вручную.

Настройка графической системы

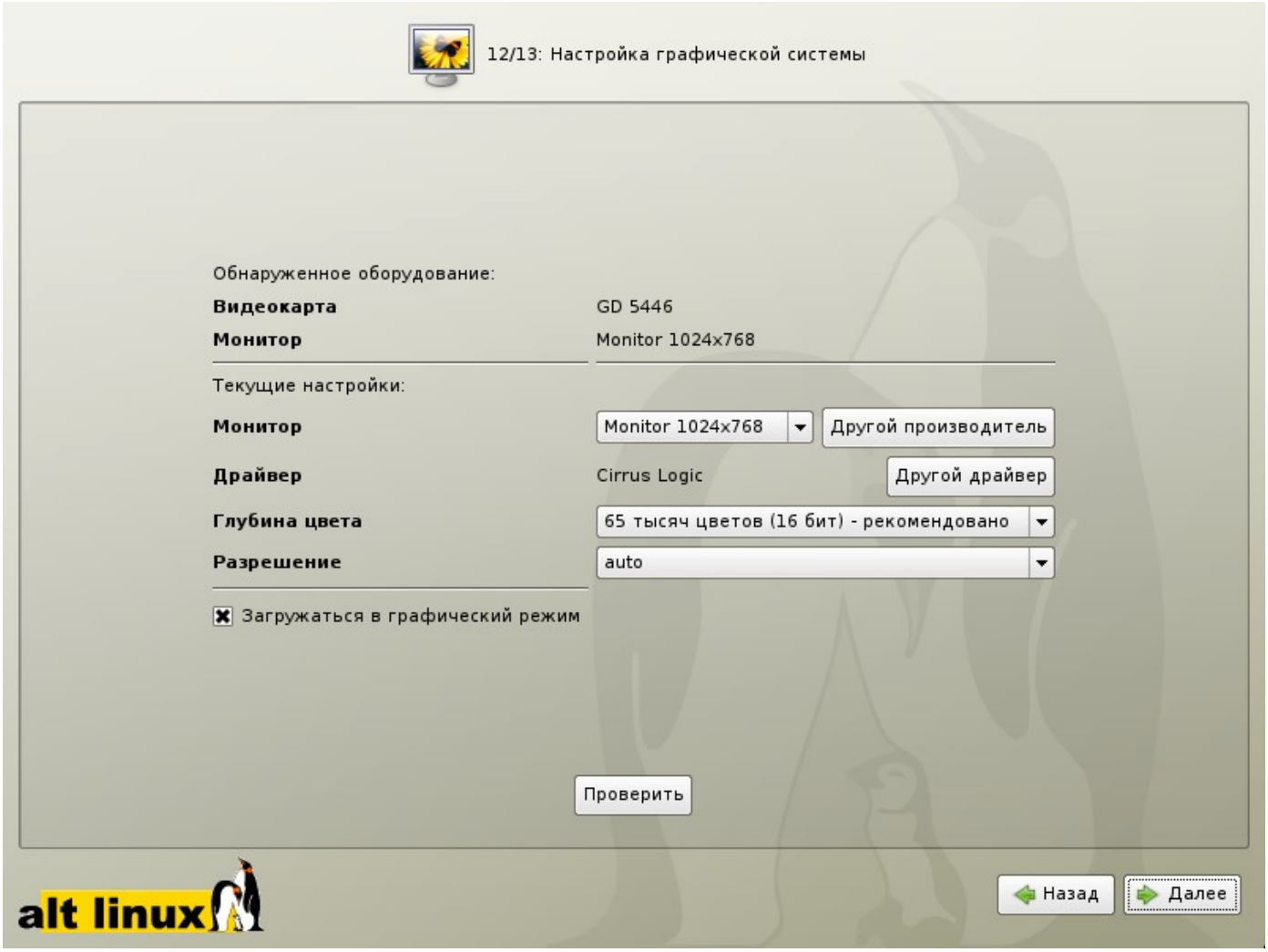


Иллюстрация 15. Настройка графической системы

Современное графическое оборудование в большинстве случаев поддаётся автоматическому определению, хотя некоторое очень новое или редкое оборудование может отсутствовать в базе данных. Автоматически определённые видеокарта и монитор будут указаны в разделе «Обнаруженное оборудование». В разделе «Текущие настройки» будут предложены наиболее подходящие настройки графического режима — их стоит испробовать в первую очередь. Довольно часто видеокарта может работать с несколькими разными драйверами. По умолчанию предлагается тот, который считается наилучшим для данной модели.

Нужно заметить, что оптимальные настройки — это не всегда максимальные значения из возможных (разрешение, глубина цвета и т. п.). При указании рекомендуемых значений учитываются свойства конкретного оборудования и драйвера, поэтому выбор более высоких значений не обязательно приведёт к улучшению качества изображения. Если оборудование автоматически не определилось, то драйвер для видеокарты и модель монитора придётся выбрать вручную.

Проверить работоспособность выбранных параметров можно, нажав на кнопку «Проверить». В случае успешной активации графического режима с новыми параметрами, вы увидите сообщение на чёрном экране, где можете либо подтветдить работоспособность графического режима нажатием кнопки «Да», либо отказаться от текущих настроек, нажав «Нет». Кнопка «Стоп» служит для приостановки счётчика времени задержки перед возвращением в диалог настройки графического режима. Если на нажимать никаких кнопок в окне тестирования видеорежима, к примеру, если из-за неверных настроек графического режима данное сообщение вообще не отобразилось на экране, то через несколько секунд будет возвращено исходное состояние, где вы можете выбрать более подходящие настройки.

Обратите внимание на отметку «Загружаться в графический режим»: новичку в Linux нужно проследить, чтобы она была установлена. В противном случае загрузка будет заканчиваться приглашением к регистрации в системе (login:) в текстовом режиме.

Смена драйвера видеокарты

При необходимости вы можете сменить драйвер видеокарты. В списке перечислены названия доступных драйверов с указанием через дефис производителя и, в некоторых случаях, моделей видеокарт. Вы можете выбрать тот их них, который считаете наиболее подходящим. Драйвер, рекомендуемый для использования помечен «рекомендовано».

Если в списке нет драйвера для вашей модели видеокарты, можно попробовать один из двух стандартных драйверов: «vga — Generic VGA Compatible» или «vesa — Generic VESA Compatible».

Выбор модели монитора

Модели мониторов можно выбирать по производителям: кнопка «Другой производитель». Ускорить передвижение по спискам можно, набирая первые буквы искомого слова. После выбора производителя в списке становятса доступны модели мониторов данного произаодителя. Не всегда обязательно подбирать монитор с точностью до номера модели: некоторые пункты в списке не содержат конкретного номера модели, а указывают на целый ряд устройств, например «Dell 1024x768 Laptop Display Panel».

Если в списке не нашлось производителя или близкой модели, то можно попробовать один из стандартных типов монитора. Для этого в списке производителей нужно выбрать «Generic CRT Display» (для электронно-лучевых мониторов) либо «Generic LCD Display» (для жидкокристаллических мониторов), а далее выбрать модель, руководствуясь желаемым разрешением.

Завершение установки

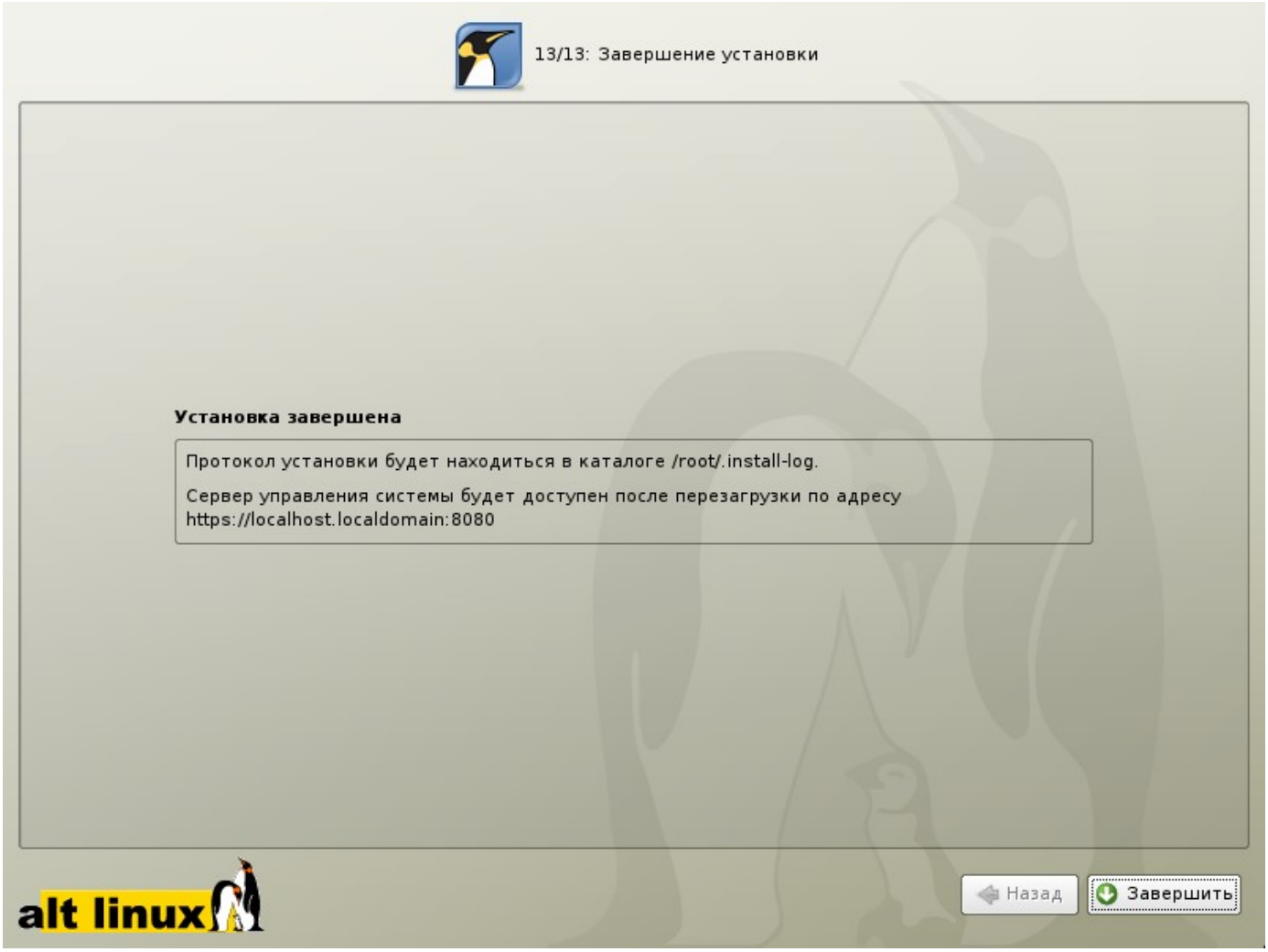


Иллюстрация 16. Завершение установки

На экране последнего шага установки отображается информация о местонахождении протокола установки (каталог /root/.install-log) и адрес веб-интерфейса управления системой вида https://localhost.localdomain:8080.

После нажатия кнопки «Завершить» и перезагрузки компьютера можно загрузить установленную систему в обычном режиме.

Удачной работы с ALT Linux!

Первая помощь

Главный совет: В случае возникновения каких-либо неприятностей не паникуйте, а не спеша разберитесь в сложившейся ситуации. Linux не так уж просто довести до полной неработоспособности и утраты ценных данных. Однако поспешные действия отчаявшегося пользователя могут привести к плачевным результатам. Помните, что решение есть, и оно обязательно найдётся!

Проблемы с загрузкой системы

Если не загружается ни одна из установленных операционных систем, значит проблема в **начальном загрузчике**. Такие проблемы могут возникнуть после установки системы, если загрузчик не установился или установился с ошибкой. При (пере) установке Windows на вашем компьютере загрузчик Linux будет перезаписан в принудительном порядке, и станет невозможно запускать Linux.

Повреждение или перезапись загрузчика никак не затрагивает остальные данные на жёстком диске, поэтому в такой ситуации очень легко вернуть работоспособность: для этого достаточно восстановить загрузчик.

Если у вас исчез загрузчик другой операционной системы или другого производителя, то внимательно почитайте соответствующее официальное руководство на предмет его восстановления. Но в большинстве случаев вам это не потребуется, так как загрузчик LILO, входящий в состав ALT Linux, поддерживает загрузку большинства известных операционных систем.

Для восстановления загрузчика LILO достаточно любым доступным способом загрузить Linux и получить доступ к тому жёсткому диску, на котором находится повреждённый загрузчик. Для этого проще всего воспользоваться **восстановительным режимом**, который предусмотрен на загрузочном диске дистрибутива. В ALT Linux к нему можно получить доступ, загрузившись с CD-ROM и выбрав в меню пункт «Спасательная система».

Загрузка восстановительного режима заканчивается приглашением клмандной строки: [root@localhost /]# Начиная с этого момента система готова к вводу команд.

Чтобы восстановить загрузчик, нужно выполнить следующие действия:

1. Смонтировать корневой раздел Linux (“/”) командой mount /dev/hda1 /mnt (На месте /dev/hda1 должен быть указан тот раздел диска, на котором у вас находится корневая файловая система Linux).[1]
2. Смонтировать необходимые для восстановления загрузчика фаловые системы:
   - mount /proc /mnt/hda1/proc -o bind
   - mount /dev /mnt/hda1/dev -o bind
3. Объявить файловую систему на этом разделе корневой:
   - chroot /mnt.
4. Если на диске ничего не менялось (не установлено новых систем), восстановить загрузчик можно одной командой — lilo. В результате загрузчик будет восстановлен в *той же конфигурации*, что и перед повреждением.

Если на диске произошли изменения, которые должны затронуть список загружаемых систем (добавлены/удалены ОС), перед выполнением команды lilo потребуется обновить конфигурацию загрузчика: с помощью любого текстового редактора (например, mcedit) отредактировать файл lilo.conf. Подробнее см. раздел "Настройка загрузки".

Проблемы при установке системы

Если в системе не произошла настройка какого-либо компонента после стадии установки пакетов, не отчаивайтесь — доведите установку до конца, загрузитесь в систему и попытайтесь теперь в спокойной обстановке повторить попытку. Если вы вообще не смогли установить систему (не произошла или не завершилась стадия установки пакетов), сначала попробуйте повторить попытку в режиме «Установка — безопасные параметры». Возможно также, что у вас какое-то новое или нестандартное оборудование, но может оказаться, что оно отлично настраивается со старыми драйверами. В любом случае, вы всегда можете сообщить о своих проблемах нам, написав в списки рассылки или обратившись в службу технической поддержки, и мы попытаемся вам помочь. Если вы хотите получить точный ответ, то сообщите, пожалуйста, подробный состав вашего оборудования и подробное описание возникшей проблемы.

1Для автоматизации монтирования уже установленной системы можно использовать скрипт mount-system. В этом случае обнаруженная корневая файловая система монтируется в каталог /mnt/system1 и все дальнейшие действия необходимо предпринимать с поправклй на использование этого каталога.

Быстрый старт

Что нужно знать о Linux пользователю

Вход в систему

Linux — это многопользовательская система. На практике это означает, что для работы в системе нужно в ней *зарегистрироваться*, т. е. дать понять системе, кто именно находится за монитором и клавиатурой. Вместо формального «зарегистрироваться в системе» обычно используют выражение «войти в систему». Операционная система представляется чем-то вроде замкнутого помещения, внутри которого можно оказаться, только успешно проникнув через «дверь» — пройдя процедуру регистрации. Наиболее распространённый способ регистрации на сегодняшний день — использование **системных имён** (login name) и паролей (password). Это надёжное средство убедиться, что с системой работает тот, кто нужно, если пользователи хранят свои пароли в секрете и если пароль достаточно сложен и не слишком короток (иначе его легко угадать или подобрать).

Загрузка заканчивается интерфейсом входа в систему: выводится приглашение ввести системное имя пользователя (login:) и пароль. Если вы выбрали загрузку в графический режим, то можно не вводить системное имя вручную, а нажать на кнопку с нужным именем, однако пароль при этом всё равно нужно ввести самостоятельно.

Пользователи обычно создаются непосредственно в процессе установки системы, однако всегда можно добавить новых пользователей или удалить существующих при помощи стандартных средств управления пользователями.

Не следует входить в систему в качестве пользователя root: он необходим для выполнения административных задач, поэтому *на него не распространяются ограничения доступа*. Для выполнения обычных администраторских задач (изменение настроек системы), не требуется *входить в систему* под именем root, обычный пользователь может временно получить права администратора (см. об этом далее).

Домашний каталог

В Linux у каждого пользователя обязательно есть свой домашний каталог, предназначенный для хранения *всех собственных данных* пользователя. Именно с этого каталога пользователь начинает работу после регистрации в системе. Домашние каталоги пользователей обычно собраны в каталоге /home, их название чаще всего совпадает с учётным именем пользователя в системе, например, для пользователя test домашним кататлогом будет /home/test.

Пользователь является полным хозяином внутри своего каталога, однако остальная часть **файловой системы** доступна ему только для чтения, но не для записи. Доступ других пользователей к чужому домашнему каталогу ограничен: наиболее типична ситуация, когда пользователи могут читать содержимое файлов друг друга, но не имеют права их изменять или удалять.

Графический и текстовый интерфейс

В операционной системе Linux пользователю доступны два режима работы: графический и текстовый. В текстовом режиме недоступны возможности графических интерфейсов: рисование окон произвольной формы и размера, поддержка миллионов цветов, отрисовка изображений. Все возможности текстового режима ограничены набором текстовых и псевдографических символов и несколькими десятками базовых цветов. Тем не менее в Linux в текстовом режиме можно выполнять практически любые действия в системе (кроме тех, которые требуют непосредственного *просмотра* изображений).

Текстовый режим в Linux — это полнофункциональный способ управления системой благодаря интерфейсу командной строки (см. об этом далее). В Linux существует огромное множество программ (включая даже игры), предназначенных для работы в текстовом режиме.

Бывают ситуации, когда графический режим недоступен или неработоспособен (удалённый доступ по сети, проблемы с поддержкой видеокарты, сбои системы и др.). В таких случаях всегда остаётся возможность работать в текстовом режиме, поскольку его возможности поддерживаются непосредственно графическим оборудованием и не требуют специальных драйверов или настройки.

В процессе работы Linux активно несколько **виртуальных консолей**. Каждая виртуальная консоль доступна по одновременному нажатию *Alt* и функциональной клавиши с номером этой консоли. На первых шести виртуальных консолях (*Alt+F1* — *Alt+F6*) пользователь может зарегистрироваться и работать в текстовом режиме. 12-ая виртуальная консоль (*Alt+F12*) выполняет функцию системной консоли — на неё выводятся сообщения о происходящих в системе событиях.

Если загрузка системы по каким-то причинам не дошла до графического режима и завершилась приглашением к регистрации (login:) на текстовой виртуальной консоли, то можно попробовать запустить графический режим вручную. Для этого следует войти в систему (ввести имя пользователя и пароль), и ввести команду startx. Эта команда запускает графическую подсистему X11, которая займёт седьмую виртуальную консоль. Можно запустить до трёх графических подсистем (интерфейсов) одновременно, они займут консоли с седьмой по девятую. Чтобы переключиться из графического режима на другую консоль, следует нажимать одновременно *Ctrl*, *Alt* и функциональную клавишу с номером нужной консоли.

Благодаря виртуальным консолям каждый компьютер, на котором работает Linux, предоставляет возможность зарегистрироваться и получить доступ к системе одновременно нескольким пользователям. Даже если в распоряжении всех пользователей есть только один монитор и одна системная клавиатура, эта возможность небесполезна: можно переключаться между виртуальными консолями так, как если бы вы переходили от одного монитора с клавиатурой к другому, подавая время от времени команды и следя за выполняющимися там программами. Более того, ничто не препятствует зарегистрироваться в системе несколько раз под одним и тем же **системным именем** — это один из способов организовать параллельную работу над несколькими задачами.

Завершение работы

Linux *нельзя* выключать, просто лишив компьютер электропитания. Множество информации, которая должна располагаться на диске, система держит в оперативной памяти для повышения быстродействия. Неожиданное выключение питания приводит к потере этой информации. Если вы работаете в графической среде, то для завершения работы нужно выбрать соответствующий пункт в главном меню. Если перед вами — графический интерфейс входа в систему, то там также есть кнопка меню, в котором имеется пункт «выключить компьютер». В командной строке (например, на текстовой виртуальной консоли), можно выполнить команду halt (требует привилегий администратора). Процедура выключения завершится автоматическим отключением питания компьютера, если это возможно. Если автоматическое отключение невозможно, на экран выведется соответствующее сообщение, и питание можно отключить кнопкой *Power*. Если компьютер поддерживает протокол работы ACPI, нажатие на кнопку *Power* приводит не к выключению электропитания, а к передаче системе ACPI-сообщения о том, что кнопка *Power* была нажата. При получении такого сообщения система выполняет ту же процедуру выключения. Только в этом случае *допустимо* выключение кнопкой *Power*.

Командная строка

Командная строка — это способ организации интерфейса, в котором каждая строка, введённая пользователем — это команда системе, которую та должна выполнить. Термин «командная строка» происходит от того, что команды вводятся обычно в одну строку, которая завершается нажатием клавиши «ввод» (Enter). В Linux этот вид интерфейса всегда был основным, а потому хорошо развитым.

Первое слово в такой строке — это, как правило, имя исполняемого файла — **программы**, все остальные слова — **параметры**. Программа выполняет нужные пользователю действия, но может

делать это по-разному в зависимости от полученных параметров. Параметры могут быть общими, например имя файла, который нужно обработать, или специфическими для этой программы модификаторами выполнения.

Чтобы получить командную строку, пользователь должен войти в систему и запустить программу, которая будет принимать его команды и передавать их на выполнение — командную оболочку (её ещё называют **интерпретатор командной строки**, просто **оболочка**, по-английски «shell»).

Получить командную строку можно многими способами. Самый простой и универсальный — зарегистрироваться на одной из первых шести виртуальных консолей: после входа в систему запустится командная оболочка и появится **приглашение командной строки**. Не выходя из графической среды можно получить командную строку при помощи любого **эмулятора терминала** — они перечислены в главном меню в разделе «Терминалы». Для пользователей графической среды KDE командная строка доступна также по нажатию *F2* (функция «ввести команду»).

Умение найти командную строку и выполнить в ней команду пригодится любому пользователю Linux, даже если он работает исключительно в графической оконной среде. Дело в том, что графические интерфейсы в Linux очень многообразны, кроме того, пользователь имеет возможность существенно поменять конкретный вид и расположение частей интерфейса по своему вкусу. Найти общие для всех и неизменные свойства графического интерфейса в Linux весьма непросто (если вообще возможно). В то же время командная строка доступна всегда и всюду выглядит практически одинаково. Поэтому очень часто в документации, рассчитанной на широкую аудиторию и общие случаи, в пример приводятся именно фрагменты командной строки. Нередко к командной строке апеллируют и люди, к которым обратились за советом по Linux. Оно и понятно: процитировать команду, которая даст нужный результат, гораздо проще и лаконичнее, чем словами описывать действия, которые нужно произвести для достижения того же эффекта в графической среде.

Когда упоминается команда, которую нужно выполнить в Linux, всегда имеется в виду команда, которую нужно ввести в командной строке.

Командная строка начинается **приглашением** — это подсказка, свидетельствующая о том, что система готова принимать команды пользователя. В процессе выполнения команды система может вывести те или иные сообщения, а когда выполнение завершается — вновь выводится приглашение командной строки. Приглашение может быть оформлено по-разному, но чаще всего оно заканчивается символом “$”. В примерах в документации этим символом условно обозначается командная строка: всё, что следует после него и до конца строки — это и есть команда, которую нужно ввести. Пока не нажат Enter, набранную команду можно редактировать.

```
$ date --universal
Чтв Окт 13 23:59:23 UTC 2005
$
```

Пример 1. Пример командной строки

В этом примере команда date --universal состоит из имени программы date и единственного параметра --universal, предписывающего ей выводить время по Гринвичу. Строка Чтв Окт 13 23:59:23 UTC 2005 — результат её выполнения, ответ системы. Если для выполнения команды требуются полномочия системного администратора, то в примерах для обозначения командной строки при такой команде ставится символ “#”.

О том, как узнать подробнее о разных командах, немного написано в разделе Документация, и много — во всевозможных учебниках и пособиях по Linux. Краткий рекомендательный список книг и сетевых ресурсов приведён в конце того же раздела «Документация».

Права доступа

Для каждого пользователя определена сфера его полномочий в системе: программы, которые он может запускать, файлы, которые он имеет право просматривать, изменять, удалять. При попытке сделать что-то, выходящее за рамки полномочий, пользователь получит сообщение об ошибке — Permission denied («в доступе отказано»). В полномочия обычного пользователя входит все необходимое для повседневной работы, однако ему запрещено выполнение действий, изменяющих саму систему. Это позволяет защитить систему от случайного или злонамеренного повреждения.

В Linux существует ровно один пользователь, права которого существенно выше прав остальных пользователей — это root (администратор). От имени этого пользователя можно выполнить любые административные (изменяющие систему) действия — *на него не распространяются ограничения доступа*.

Когда нужно сделать что-то, выходящее за рамки полномочий обычного пользователя, потребуется получить полномочия администратора. В большинстве случаев достаточно получить полномочия *временно*, для выполнения одного или нескольких конкретных действий. Некоторые программы (в том числе основное средство настройки системы — ALT Linux Control Center) при необходимости запрашивают пароль пользователя root. После того как пароль правильно введён, *эта программа* (и только она!) будет работать уже с правами администратора, поэтому следует внимательно относиться к совершаемым действиям.

Временно получить **командную оболочку** с правами администратора можно при помощи команды su -. Это операция доступна только тем пользователям, при добавлении которых был установлен флажок «Разрешить пользователю получать привилегии администратора (su)»[1]. По умолчанию этот флажок установлен только для первого из добавленных при установке пользователей, хотя впоследствии его можно установить или снять в любой момент для любого пользователя.

Как задавать вопросы?

Если в процессе работы возникнут сложности и сбои, очень важно по возможности конкретно сформулировать суть проблемы (вопрос). Поиски ответа стоит начать с документации (локальной и в Интернете), также можно спросить опытных пользователей и обратиться в службу поддержки. Ниже кратко описаны те шаги, которые стоит сделать для получения нужной информации.

Почитать документацию

Прежде всего следует обратиться к уже установленной документации. Основной массив документации на русском языке — это документация ALT Linux, к главной странице документации можно перейти в любом браузере со стартовой страницы дистрибутива. В документации ALT Linux содержатся вводные сведения о Linux, основные сведения по установке, настройке и использованию ALT Linux. Каждая программа также сопровождается собственной документацией, многие — и системой помощи, к сожалению, не везде эта документация переведена на русский язык. Подробнее о том, как найти документацию по конкретной программе, см. раздел Документация.

Поискать в Интернет

Если среди установленной документации не удалось найти ответа, стоит обратиться к поискам в сети Интернет. Самый первый адрес, куда следует отправиться — сборник наиболее ЧАсто задаваемых ВОпросов (FAQ) по использованию программных продуктов ALT Linux. Большинство затруднений при работе с Linux типичны, поэтому здесь с большой вероятностью найдётся ответ на ваш вопрос.

На сайте freesource.info есть довольно разнообразная информация, связанная со свободным ПО вообще, в частности, стоит обратить внимание на раздел, посвящённый ALT Linux. Здесь есть конкретные инструкции и примеры настроек, которые ещё не успели попасть в документацию, кроме того, предложения и планы разработчиков. Содержание сайта всё время уточняется и дополняется, поскольку он открыт для пополнения всем заинтересованным.

Кроме того, любому пользователю Internet доступен поисковый сайт http://google.com, наиболее подходящий для поиска чего бы то ни было. Если вы ищете причину конкретной ошибки и способ её устранить, стоит задать в качестве поискового выражения то **сообщение об ошибке**, которое было выдано системой. Программы с графическим интерфейсом обычно выводят такие сообщения в особых диалоговых окнах, которые появляются поверх основного окна программы и содержат текст сообщения об ошибке и как минимум одну кнопку — «OK». Если программа была запущена из командной строки, то сообщения о ходе её работы и об ошибках появятся там же. Сведения о событиях, происходящих в системе, всегда можно найти на 12-ой виртуальной консоли (*Ctrl+Alt+F12*), многие сообщения об ошибках тоже туда попадают.

Спросить в списке рассылки

ALT Linux Team поддерживает несколько списков рассылки, в которых обсуждаются вопросы использования и разработки дистрибутивов ALT Linux. Вы можете задать свой вопрос сообществу пользователей дистрибутивов ALT Linux, просто написав в один из списков рассылки ALT Linux Team. Основной список расылки сообщества пользователей ALT Linux — community@lists.altlinux.org. Возможно, в списке рассылки уже был дан ответ на ваш вопрос (большинство вопросов повторяются), поэтому прежде чем писать в список рассылки, стоит поискать ответ в архивах рассылки. Списки рассылки читают разработчики и активные пользователи ALT Linux, и обычно среди них находится тот, кто ответит на вопрос[2].

Обратиться в службу поддержки

Если вы — зарегистрированный пользователь дистрибутива, обращайтесь с вопросами в службу поддержки ALT Linux. Для регистрации потребуется серийный номер, который помещён на каждом продаваемом экземпляре дистрибутива.

Приобретая экземпляр дистрибутива, пользователь в том числе приобретает контракт, по условиям которого он получает право на объём услуг по информационной и технической поддержке, определённый в купоне технической поддержки дистрибутива.

Впрочем, для заключения контракта не обязательно покупать дистрибутив, просто обращайтесь к нам. Специалисты ALT Linux помогут всегда: от разового решения любых задач для пользователей *любых* Linux, до регулярного технического обслуживания и разработки технологических решений на базе Linux. Более подробную информацию об услугах ALT Linux по технической поддержке можно найти на сайте ALT Linux.

---

1 - Установка этого флажка означает, что пользователь будет включён в группу wheel.

2 - Задавший вопрос пользователь должен принимать в расчёт, что все подписчики списка рассылки участвуют в нём добровольно, и никто из них не обязан отвечать на какие-либо вопросы, поэтому предъявлять претензии на этот счёт бессмысленно и невежливо.

В дистрибутив ALT Linux входит комплект документации в печатном виде (если вы приобрели дистрибутив в розницу) и в электронном виде (в формате HTML). В нашей документации вы найдёте сведения и рекомендации по установке и настройке системы, а также обзор доступных прикладных программ и способов работы с ними. В случае установки по умолчанию, вся документация будет доступна через общесистемное меню Документация или по ссылке на рабочем столе KDE.

Не пренебрегайте чтением документации: она поможет вам избежать многих сложностей, сэкономить массу времени и усилий при установке, настройке и администрировании системы, поможет найти нужное для работы приложение и быстро разобраться в нём. Даже если вы — опытный пользователь Linux, в документации найдутся полезные для вас сведения об особенностях дистрибутива ALT Linux. Если же вы только начали знакомиться с ОС Linux и не имеете опыта работы в UNIX-подобных системах, вам необходимо обзавестись книгой по Linux. Список рекомендуемых нами книг вы найдёте в конце данного раздела.

Экранная документация

Помимо поставляемой ALT Linux документации и дополнительной литературы, всё программное обеспечение, входящее в дистрибутив, снабжается собственной документацией. Стандартный способ получить документацию по той или иной программе, функции или файлу, установленным в системе, унаследованный Linux от ОС UNIX, — это команда **man**, отображающая *экранную документацию*, иногда называемую «страницы руководства» (буквальный перевод англ. manual pages). Для того, чтобы прочесть экранную документацию по программе, достаточно в любой командной строке набрать **man** программа. Например, команда **man man** выдаёт справку по пользованию самой командой **man**. Если вы точно не знаете, как называется необходимая вам программа, может помочь поиск по ключевому слову при помощи команд **apropos** и **whatis**. Например, если вы введёте команду **apropos mail**, вы увидите список всех программ, в кратком описании которых упоминается слово mail. Разница между командами заключается в том, что **whatis** ищет только по названиям руководств, а **apropos** ещё и по кратким описаниям.

В технической документации по UNIX и Linux принят стандартный формат ссылки на экранную документацию, выводимую по команде **man**. Например, запись apt(8), отсылает к экранной документации по программе apt, вызываемой командой **man apt** (цифра в скобках обозначает раздел, к которому относится данная документация, её требуется вводить только в том случае, если есть несколько руководств с одним именем, но в разных разделах, например **man 8 apt**). К сожалению, большая часть экранной документации пока не переведена на русский язык. Переводы некоторых наиболее важных руководств есть в пакете man-pages-ru, если его установить, то при наличии перевода **man** будет отображать руководство по-русски.

Документация проекта GNU и многих других приложений существует в виде страниц info, просматривать которые можно при помощи команды **info**. Доступ к экранной документации возможен через интегрированные средства просмотра документации графической среды KDE — KDE Help Center. Это средство обладает собственными ресурсами помощи, которые легко вызываются с Рабочего стола или через общесистемное меню Документация.

Документация по пакетам

Основное место для хранения разнообразной документации, в основном на английском языке, — каталог /usr/share/doc. Особое внимание обратите на HOWTO (от англ. how to — «как сделать») — собрание практических рекомендаций по самым различным вопросам, связанным с использованием Linux.

Каждый пакет также содержит поставляемую вместе с включённым в него ПО документацию, располагающуюся обычно в каталоге /usr/share/doc/*имя_пакета*. Например, документация к пакету foo-1.0-alt1 находится в /usr/share/doc/foo-1.0-alt1. Для получения полного списка файлов документации,

относящихся к пакету, воспользуйтесь командой **rpm** -qld имя_пакета.

В документации к каждому пакету вы можете найти такие файлы как README, FAQ, TODO, ChangeLog и другие. В файле README содержится основная информация о программе — имя и контактные данные авторов, назначение, полезные советы и пр. FAQ содержит ответы на часто задаваемые вопросы; этот файл стоит прочитать в первую очередь, если у вас возникли проблемы или вопросы по использованию программы, поскольку большинство проблем и сложностей типичны, вполне вероятно, что в FAQ вы тут же найдёте готовое решение. В файле TODO записаны планы разработчиков на реализацию той или иной функциональности. В файле ChangeLog записана история изменений в программе от версии к версии.

Адреса сайтов в Интернет, посвящённых отдельным программным продуктам, указаны в информационных заголовках соответствующих пакетов, их можно получить с помощью команды **rpm** -qi имя_пакета.

Рекомендуемая литература

1. *Курячий Г. В.*, *Маслинский К. А.* Операционная система Linux: Курс лекций. Учебное пособие. — М.: Интернет-университет информационных технологий, 2005.

   http://www.intuit.ru/department/os/linux/

2. *Курячий Г. В.* Операционная система UNIX: Курс лекций. Учебное пособие. — М.: Интернет-университет информационных технологий, 2004.

   http://www.intuit.ru/department/os/osunix/

3. *Андреев С. В.*, *Роганова Н. А.* Практическая информатика. Ч. 1 — М.: МГИУ, 2001.

   http://www.ctc.msiu.ru/materials/Book1/index1.html

4. Библиотека LinuxCenter.

   http://linuxcenter.ru/lib/books/

5. Виртуальная энциклопедия «Linux по-русски».

   http://rus-linux.net/

6. *Угринович Н. Д.* Преподавание курса «Информатика и ИКТ» в основной и старшей школе: Методическое пособие + 2CD. М.: Бином, 2004.

В этом разделе содержится список задач, часто встающих перед пользователем ALT Linux после успешной установки системы.

Выяснить, какая программа в Linux мне нужна

Первое, что необходимо сделать, — это воспользоваться поиском по списку доступных для установки программ. Сделать это можно при помощи:

- программы synaptic. Она находится в меню «Система — Менеджер пакетов (Программа управления пакетами)». Краткий обзор по использованию этой программы содержится в Руководстве.
- команды apt-cache search ключевое_слово, которую можно выполнить в любом эмуляторе терминала либо в текстовом режиме работы при отсутствии запущенной графической оболочки.

При поиске нужной программы вам может помочь раздел "Прикладные программы для Linux" и таблица http://www.linuxrsp.ru/win-lin-soft/table-rus.html

Установить или удалить программу

Воспользуйтесь программой synaptic («Система — Менеджер пакетов (Программа управления пакетами)»). В меню программы «Помощь — Краткое описание» вы найдете подсказку по способам выбора пакетов и возможным операциям над выбранными пакетами.

В интерфейсе командной строки для этих задач существует утилита apt-get. Воспользоваться ей достаточно просто:

- su - (этот шаг необходим для получения привилегий суперпользователя root)
- apt-get install имя_пакета (для установки пакета)
- apt-get remove имя_пакета (для удаления пакета)

Более полная информация о работе с программными пакетами находится в разделе Система управления пакетами APT

Настроить выход в Интернет

Прежде всего, надо ответить на вопрос, каким образом вообще организован выход в Интернет.

Наиболее часто встречающиеся варианты — это:

- Dialup-соединение: выход в Интернет через обычный модем и телефонную сеть
  - для настройки такого рода соединения рекомендуется воспользоваться программой KPPP (пункт меню «Интернет — Подключение по диалапу (KPPP)»). При возникновении вопросов по её настройке следует обратиться к справке: кнопка «Справка» — «Руководство KPPP».
- через мобильный телефон — GRPS
  - этот случай отличается от предыдущего только тем, что надо обеспечить доступность встроенного в мобильный телефон модема. Подробности можно найти в разделе "Подключение к Интернет через мобильный телефон"
- через xDSL-модем. В случае, если ваш xDSL-модем является ethernet-модемом, все настройки, касающиеся непосредственно подключения, — имя и пароль пользователя и прочие данные, предоставляемые вашим провайдером, — вводятся в сам модем. За подробностями обратитесь к документации к вашему оборудованию. Настройки в Linux сводятся в этом случае к указанию правильных сетевых настроек, которые перечислены ниже.

Если ваш компьютер подключён к локальной сети, то всё, что вам потребуется — это указать верные

сетевые настройки:

- IP-адрес
- IP-адрес шлюза
- IP-адреса DNS-серверов
- возможно, прокси-сервер — его нужно будет указывать во всех программах, которым необходим доступ в Интернет.

Для указания сетевых настроек воспользуйтесь Центром управления системы либо Центром управления системы (www). Инструкцию можно найти в Документации.

Указанные выше параметры могут предоставляться автоматически, если в вашей сети настроен и работает DHCP-сервер. В этом случае всё, что требуется, — это поставить соответствующую отметку «Использовать DHCP» в сетевых настройках.

«Увидеть» другие компьютеры в сети

Для обзора доступных ресурсов сети воспользуйтесь программой konqueror (пункт меню «Личные файлы (Домой)»). После запуска программы перейдите в ней в меню «Перейти — Сетевые папки». Нажмите на значок «Ресурсы Samba». Вы должны увидеть список рабочих групп, доступных в вашей сети, а при нажатии на значок нужной рабочей группы — список компьютеров.

Альтернативные пути:

- Меню «Система — Сетевые ресурсы»
- «Рабочий стол — Система — Сетевые ресурсы»
- smb:/ в строке «Адрес» программы konqueror.

Печатать на принтер

Настройка принтера в Linux сводится к настройке служюы печати CUPS. Для этого есть несколько путей:

- Меню «Настройка — Настройка печати». Это стандартный web-интерфейс настройки CUPS
- Меню «Система печати — Добавить принтер».

Информацию о том, насколько хорошо поддерживается ваш принтер, можно найти по адресу http://linuxprinting.org.

Соединиться с мобильным телефоном или другим устройством по bluetooth

Для настройки соединения с bluetooth-устройством воспользуйтесь программой kbluetooth: меню «Настройка — Bluetooth Server (kbluetooth)». После её запуска в области уведомлений панели задач появится значок bluetooth. При нажатии на него открывается список обнаруженных bluetooth-устройств, с которыми можно связаться.

Отправить компьютер в «спящий» режим

Воспользуйтесь программой KPowersave. При запущенной программе в области уведомлений отображается значок (электрическая вилка). Нажатие правой кнопкой мыши открывает меню, в котором доступны различные функции, в том числе «Уснуть».

У меня есть устройство X. Поддерживается ли оно в Linux? Как это узнать? Где взять драйвер? На диске производителя устройства есть драйвер для Linux. Как его установить?

Не спешите искать драйверы на прилагаемых дисках или на сайте производителя конкретного оборудования. Ознакомьтесь с главой "Работа с оборудованием в Linux"

Интересной особенностью работы некоторых беспроводных карт (WLAN) является то, что для них могут использоваться win-драйвера. Для настройки таких карт нужно установить ndiswrapper и настроить его в соответствии с документацией.

Подходящей программы под Linux нет, а под Windows есть. Что делать?

Попробуйте запустить win-приложение, используя WINE. Подробности содержатся в разделе «WINE: среда для запуска win-приложений на платформе Unix»

Я ничего не понимаю

Не отчаивайтесь! Старайтесь действовать последовательно, и вы непременно решите вашу задачу. Прежде всего ознакомьтесь с разделами:

- Документация
- Что нужно знать о Linux пользователю

Администрирование

Настройка ALT Linux 4.0 Desktop

Центр управления системы (www)

Центр управления системы (www) — основной способ настройки ALT Linux. Он представляет из себя веб-интерфейс, который позволяет управлять большинством настроек системы. Центр управления системы (www) состоит из нескольких независимых диалогов-модулей. Каждый модуль отвечает за настройку определённой функции или свойства системы.

Способы запуска

Центр управления системы (www) можно запустить следующими способами:

- из меню в графической среде: «Настройка — Центр управления системы (www)»;
- из командной строки: командой configurator.

При запуске необходимо подтвердить (принять) сертификат, после чего ввести в соответсвующие поля имя пользователя (root) и пароль пользователя root.

Центр управления системы (www) содержит справочную информацию по всем включённым в него модулям. Об использовании самого интерфейса системы управления можно прочитать, перейдя по ссылке:

- /var/www/html/fbi/help/ru_RU/alterator.html

Центр управления системы

Для управления настройками ALT Linux вы можете воспользоваться Центром управления системы. Он позволяет в графическом интерфейсе управлять наиболее востребованными настройками системы: пользователями, сетевыми подключениями, и т. п. Центр управления системы состоит из нескольких независимых диалогов-модулей. Каждый модуль отечает за настройку определённой функции или свойства системы.

Способы запуска

Центр управления системы можно запустить следующими способами:

- из меню в графической среде: «Настройка — Центр управления системы»;
- из командной строки: командой alterator-standalone.

При запуске неоходимо ввести пароль суперпользователя (root).

Как это делают профессионалы

Разработчики ALT Linux приложили все усилия, чтобы сделать вашу работу с Linux максимально простой. Используя Центр управления системы (www) и Центр управления системы, вы можете производить необходимые при повседневной работе настройки. Однако важно понимать, что даже подобные средства конфигурирования не могут покрыть всю функциональность, доступную в ALT Linux. Более широкие возможности открывает умение натраивать систему «вручную», оно позволяет решать практически любые задачи, возникающие при работе с ALT Linux.

Как правило, настройка нужной вам программы либо свойства системы сводится к редактированию определённого конфигурационного файла. А так как конфигурационные файлы являются обыкновенными текстовыми файлами, то всё, что вам нужно — это выбрать любой текстовый редактор, к примеру, KWrite. При работе в режиме командной строки воспользуйтесь одним из консольных текстовых редакторов: mcedit, nano, joe, jed, vim и т. п. Как ни странно, именно простой текстовый редактор и является самым мощным средством конфигурирования любой Linux-системы.

Помимо обладания навыками работы с текстовым редактором, важно знать, *что именно* и *как* редактировать: какой файл и каков его синтаксис. Для ответа на эти вопросы необходимо обратиться к подсистеме помощи. Подробности о методах работы с документацией можно почерпнуть из раздела Документация.

Пример: настройка преобразования доменных имен в IP-адреса

Для преобразования символьнх имён в IP-адреса используется DNS-сервер. К примеру, первое, что происходит при запросе в адресной строке вашего web-браузера страницы http://www.google.com, это преобразование доменного имени (www.google.com) в IP-адрес. Его осуществляет специальный набор подпрограмм (resolver) путём обращения к DNS-серверу, укзанному в настройках вашей системы.

Для того, чтобы указать используемый DNS-сервер, необходимо решить несколько задач:

- Во-первых, выяснить, какой конфигурационный файл хранит нужные настройки. Выяснить это достаточно просто:
    - $ apropos resolver
- Во-вторых, выяснить, *что именно* нужно добавить либо отредактировать в этом файле:
    - $ man resolv.conf
    - /etc/resolv.conf — конфигурационный файл, в котором указываются используемые DNS-серверы. Из документации resolve.conf(5) становится понятен синатксис этого файла. А именно то, что DNS-серверы указываются за ключевым словом nameserver. Остаётся только внести в файл либо отредактировать в нём необходимые строки:

mcedit /etc/resolv.conf

В итоге, интересующая нас часть конфигурационного файла может выглядеть примерно так:

nameserver 192.168.0.1

nameserver 88.99.88.99

nameserver 77.88.77.88

Конечно, данный пример служит лишь демонстрацией принципов работы с конфигурационными файлами и ни в коем случае не претендует на полноту изложения.

Семь раз отмерь, один — отрежь

При редактировании конфигурационных файлов, в особенности если вы делаете это впервые, желательно не спешить и соблюсти простейшие меры предосторожности: создать резервные копии и не изменять более одного–двух параметров за раз, после каждого редактирования проверяя работоспособность системы. В противном случае найти ошибку будет сложно.

Помните, что глобальные конфигурационные файлы доступны для редактирования только администратору системы. Поэтому действуйте по принципу: «Семь раз отмерь, один — отрежь».

Установка и удаление программ (пакетов)

Система управления пакетами APT

Александр Боковой, Дмитрий Левин, Кирилл Маслинский

Введение: пакеты, зависимости и репозитории

В современных системах на базе Linux огромное число общих ресурсов, которыми пользуются сразу несколько программ: разделяемых библиотек, содержащих стандартные функции, исполняемых файлов, сценариев и стандартных утилит и т. д. Удаление или изменение версии одного из составляющих систему компонентов может повлечь неработоспособность других, связанных с ним компонентов, или даже вывести из строя всю систему. В контексте системного администрирования проблемы такого рода называют нарушением *целостности системы*. Задача администратора — обеспечить наличие в системе согласованных версий всех необходимых программных компонентов (обеспечение целостности системы).

Для установки, удаления и обновления программ и поддержания целостности системы в Linux в первую очередь стали использоваться *менеджеры пакетов* (такие, как rpm в дистрибутивах RedHat или dpkg в Debian GNU/Linux). С точки зрения менеджера пакетов программное обеспечение представляет собой набор компонентов — программных *пакетов*. Такие компоненты содержат в себе набор исполняемых программ и вспомогательных файлов, необходимых для корректной работы программного обеспечения. Менеджеры пакетов облегчают установку программ: они позволяют проверить наличие необходимых для работы устанавливаемой программы компонент подходящей версии непосредственно в момент установки, а также производят необходимые процедуры для регистрации программы во всех операционных средах пользователя: сразу после установки программа может быть доступна пользователю из командной строки и — если это педусмотрено — появляется в меню всех графических оболочек.

> **Важно**
>
> Благодаря менеджерам пакетов, пользователю Linux обычно не требуется непосредственно обращаться к установочным процедурам отдельных программ или непосредственно работать с каталогами, в которых установлены исполняемые файлы и компоненты программ (обычно это /usr/bin, /usr/share/*имя_пакета*) — всю работу делает менеджер пакетов. Поэтому установку, обновление и удаление программ в Linux обычно называют *управлением пакетами*.

Часто компоненты, используемые различными программами, выделяют в отдельные пакеты и помечают, что для работы ПО, предоставляемого пакетом A, необходимо установить пакет B. В таком случае говорят, что пакет A *зависит* от пакета B или что между пакетами A и B существует *зависимость*.

Отслеживание зависимостей между такими пакетами представляет собой серьёзную задачу для любого дистрибутива — некоторые компоненты могут быть взаимозаменяемыми: может обнаружиться несколько пакетов, предлагающих затребованный ресурс.

Задача контроля целостности и непротиворечивости установленного в системе ПО ещё сложнее. Представим, что некие программы A и B требуют наличия в системе компоненты C версии 1.0. Обновление версии пакета A, требующее обновления компоненты C до новой, использующей новый интерфейс доступа, версии (скажем, до версии 2.0), влечёт за собой обязательное обновление и программы B.

Однако менеджеры пакетов оказались неспособны предотвратить все возможные коллизии при

установке или удалении программ, а тем более эффективно устранить нарушения целостности системы. Особенно сильно этот недостаток сказывается при обновлении систем из централизованного репозитория пакетов, в котором последние могут непрерывно обновляться, дробиться на более мелкие и т. п. Этот недостаток и стимулировал создание систем управления программными пакетами и поддержания целостности системы.

Для автоматизации этого процесса и применяется Усовершенствованная система управления программными пакетами APT (от англ. Advanced Packaging Tool). Такая автоматизация достигается созданием одного или нескольких внешних *репозиториев*, в которых хранятся пакеты программ и относительно которых производится сверка пакетов, установленных в системе. Репозитории могут содержать как официальную версию дистрибутива, обновляемую его разработчиками по мере выхода новых версий программ, так и локальные наработки, например, пакеты, разработанные внутри компании.

Таким образом, в распоряжении APT находятся две базы данных: одна описывает установленные в системе пакеты, вторая — внешний репозиторий. APT отслеживает целостность установленной системы и, в случае обнаружения противоречий в зависимостях пакетов, руководствуется сведениями о внешнем репозитории для разрешения конфликтов и поиска корректного пути их устранения.

Первоначально APT был разработан для управления установкой и удалением программ в дистрибутиве Debian GNU/Linux. При разработке ставилась задача создать систему управления пакетами с простым пользовательским интерфейсом, позволяющую производить установку, обновление и повседневные «хозяйственные» работы с установленными на машине программами без необходимости изучения тонкостей используемого в дистрибутиве менеджера программных пакетов.

Эти привлекательные возможности долгое время были доступны только пользователям Debian, поскольку в APT поддерживался только один менеджер пакетов, а именно применяемый в Debian менеджер пакетов dpkg, несовместимый с используемым в ALT Linux RPM. Эта несовместимость заключается прежде всего в различии используемых форматов данных (хотя существуют программы-конвертеры), но имеются и другие различия, обсуждение которых выходит за рамки изложения.

APT, однако, изначально проектировался как не зависящий от конкретного метода работы с установленными в системе пакетами, и эта особенность позволила разработчикам из бразильской компании Conectiva реализовать в нём поддержку менеджера пакетов RPM. Таким образом, пользователи основанных на RPM дистрибутивов (дистрибутивы ALT Linux входят в их число) получили возможность использовать этот мощный инструмент.

Система APT состоит из нескольких утилит. Чаще всего используется утилита управления пакетами **apt-get**: она автоматически определяет зависимости между пакетами и строго следит за их соблюдением при выполнении любой из следующих операций: установка, удаление или обновление пакетов.

Источники программ (репозитории)

Репозитории, с которыми работает APT, отличаются от обычного набора пакетов наличием метаинформации — индексов пакетов, содержащихся в репозитории, и сведений о них. Поэтому, чтобы получить всю информацию о репозитории, APT достаточно получить его индексы.

APT может работать с любым количеством репозиториев одновременно, формируя единую информационную базу обо всех содержащихся в них пакетах. При установке пакетов APT обращает внимание только на название пакета, его версию и зависимости, а расположение в том или ином репозитории не имеет значения. Если потребуется, APT в рамках одной операции установки группы пакетов может пользоваться несколькими репозиториями.

> **Важно**
>
> Подключая одновременно несколько репозиториев, нужно следить за тем, чтобы они были совместимы друг с другом по пакетной базе, т. е. отражали один определённый этап разработки. Например, совместимыми являются основной репозиторий дистрибутива и репозиторий обновлений по безопасности *к данному дистрибутиву*. В то же время смешение среди источников APT репозиториев, относящихся к разным дистрибутивам, или смешение стабильного репозитория с нестабильной веткой разработки (Sisyphus) чревато различными неожиданными трудностями при обновлении пакетов.

APT позволяет взаимодействовать с репозиторием с помощью различных протоколов доступа. Наиболее популярные — HTTP и FTP, однако существуют и некоторые дополнительные методы.

Для того, чтобы APT мог использовать тот или иной репозиторий, информацию о нем необходимо поместить в файл /etc/apt/sources.list[1]. Описания репозиториев заносятся в этот файл в следующем виде:

```
rpm [подпись] метод:путь база название
rpm-src [подпись] метод:путь база название
```

rpm или rpm-src

: Тип репозитория (скомпилированные программы или исходные тексты).

[подпись]

: Необязательная строка-указатель на электронную подпись разработчиков. Наличие этого поля подразумевает, что каждый пакет из данного репозитория должен быть подписан соответствующей электронной подписью. Подписи описываются в файле /etc/apt/vendor.list.

метод

: Способ доступа к репозиторию: ftp, http, file, rsh, ssh, cdrom, copy.

путь

: Путь к репозиторию в терминах выбранного метода.

база

: Относительный путь к базе данных репозитория.

название

: Название репозитория.

Для добавления в sources.list репозитория на компакт-диске в APT даже предусмотрена специальная утилита — **apt-cdrom**. Чтобы добавить запись о репозитории на компакт-диске, достаточно вставить диск в привод и выполнить команду **apt-cdrom add**. После этого в sources.list появится запись о подключённом диске примерно такого вида:

```
rpm cdrom:[ALT Linux 4.0 Desktop DVD (Ajuga)]/ ALTLinux base contrib disk
```

После того как отредактирован список репозиториев в sources.list, необходимо обновить локальную базу данных APT о доступных пакетах. Это делается командой **apt-get update**.

Если в sources.list присутствует репозиторий, содержимое которого может изменяться (как происходит с любым постоянно разрабатываемым репозиторием, в частности, обновлений по безопасности (updates), backports[2]) или Sisyphus, то прежде чем работать с APT, необходимо синхронизировать локальную базу данных с удалённым сервером командой **apt-get update**. Локальная база данных создаётся заново каждый раз, когда в репозитории происходит изменение: добавление, удаление или переименование пакета. Для репозиториев, находящихся на компакт-дисках и подключённых командой **apt-cdrom add**, синхронизация производится единожды в момент подключения.

При выборе пакетов для установки, APT руководствуется *всеми* доступными репозиториями вне зависимости от способа доступа к ним. Так, если в репозитории, доступном по сети Интернет, обнаружена более новая версия программы, чем на компакт-диске, то APT начнёт загружать данный пакет из Интернет. Поэтому если подключение к Интернет отсутствует или ограничено низкой пропускной способностью канала или выской стоимостью, то следует закомментировать те строчки в /etc/apt/sources.list, в которых говорится о ресурсах, доступных по Интернет.

Репозитории ALT Linux

Все дистрибутивы ALT Linux выпускаются на основе репозитория Sisyphus команды ALT Linux Team. Следует иметь в виду, что Sisyphus не является самостоятельным дистрибутивом, а отражает текущее состояние разработки и может содержать нестабильные версии пакетов. Периодически на базе этого проекта выпускаются отдельные оттестированные «срезы» — дистрибутивы.

В отличие от Sisyphus, ежедневно обновляемого разработчиками, такие срезы являются «замороженными» — разработка в них не ведётся, и сами срезы сохраняются в целях обеспечения целостности среды дистрибутива, в которой уже не должны обновляться версии пакетов. Единственное исключение делается для обновлений, исправляющих проблемы в безопасности системы, однако такие обновления помещаются в отдельном репозитории для каждого дистрибутива. Срезы Sisyphus и репозитории обновлений также являеются полноценными репозиториями APT.

Пользователи стабильных дистрибутивов не всегда готовы переходить на нестабильную ветку разработки и в то же время заинтересованы в обновлении версий некоторых прикладных программ (в которых появляется новая функциональность и т. п.). Для этих целей заинтересованными пользователями для каждого дистрибутива ALT Linux создаются и поддерживаются специальные репозитории с обновлёнными версиями программ — backports[3]. Репозитории backports являются согласованными по пакетной базе с основным репозиторием соответствующего дистрибутива, и могут быть без опасений подключены параллельно с ним.

Непосредственно после установки дистрибутива ALT Linux в /etc/apt/sources.list обычно указывается несколько репозиториев:

- репозиторий обновлений в системе безопасности дистрибутива;
- полный срез репозитория Sisyphus, подмножеством которого является дистрибутив.

Поиск пакетов

Если вы не знаете точного названия пакета, для его поиска можно воспользоваться утилитой **apt-cache**, которая позволяет искать не только по имени пакета, но и по его описанию.

Команда **apt-cache search** подстрока позволяет найти все пакеты, в именах или описании которых присутствует указанная подстрока. Например:

```
$ apt-cache search dictionary
kdenetwork-kdict - A DICT (net dictionary) client for KDE
stardict-gtk - StarDict dictionary gtk version
stardict-mueller7 - V.K. Mueller English-Russian Dictionary, 7 Edition: stardict format
stardict-slovnyk_be-en - Dictionary: Slovnyk Belorussian-English
stardict-slovnyk_be-ru - Dictionary: Slovnyk Belorussian-Russian
stardict-slovnyk_be-uk - Dictionary: Slovnyk Belorussian-Ukranian
stardict-slovnyk_en-be - Dictionary: Slovnyk English-Belorussian
stardict-slovnyk_en-ru - Dictionary: Slovnyk English-Russian
stardict-slovnyk_en-uk - Dictionary: Slovnyk English-Ukranian
stardict-slovnyk_ru-be - Dictionary: Slovnyk Russian-Belorussian
stardict-slovnyk_ru-en - Dictionary: Slovnyk Russian-English
stardict-slovnyk_ru-uk - Dictionary: Slovnyk Russian-Ukranian
stardict-slovnyk_uk-be - Dictionary: Slovnyk Ukranian-Belorussian
stardict-slovnyk_uk-en - Dictionary: Slovnyk Ukranian-English
stardict-slovnyk_uk-ru - Dictionary: Slovnyk Ukranian-Russian
stardict-vera - V.E.R.A. -- Virtual Entity of Relevant Acronyms
stardict-wn - GCIDE - The Collaborative International Dictionary of English
```

Для того, чтобы подробнее узнать о каждом из найденных пакетов и прочитать его описание, можно воспользоваться командой **apt-cache show**, которая покажет информацию о пакете из репозитория:

```
$ apt-cache show stardict-mueller7
Package: stardict-mueller7
Section: Text tools
Installed Size: 3048495
Maintainer: Alex Murygin <murygin@altlinux.ru>
Version: 1.0-alt5
Pre-Depends: rpmlib(PayloadFilesHavePrefix) (<= 4.0-1), rpmlib(CompressedFileNames) (<= 3.0.4-1)
Depends: stardict (>= 2.4.2)
Provides: stardict-mueller7 (= 1.0-alt5)
Architecture: noarch
Size: 3052715
MD5Sum: 85778415d4c33513492d935ec03791ef
Filename: stardict-mueller7-1.0-alt5.noarch.rpm
Description: V.K. Mueller English-Russian Dictionary, 7 Edition: stardict format
 Electronic version of V.K. Mueller English-Russian Dictionary, 7 Edition
 in stardict format. You can use it with stardict client.
```

apt-cache позволяет осуществлять поиск и по русскому слову, однако в этом случае будут найдены только те пакеты, у которых помимо английского есть ещё и описание на русском языке. К сожалению, русское описание на настоящий момент есть не у всех пакетов, хотя описания наиболее актуальных для пользователя пакетов переведены.

Установка или обновление пакета

Установка пакета с помощью APT выполняется командой

```
# apt-get install имя_пакета
```

apt-get позволяет устанавливать в систему пакеты, требующие для работы другие, пока ещё не установленные. В этом случае он определяет, какие пакеты необходимо установить, и устанавливает их, пользуясь всеми доступными репозиториями.

Установка пакета stardict-mueller7 командой **apt-get install stardict-mueller7** приведёт к следующему диалогу с APT:

```
# apt-get install stardict-mueller7
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  stardict-mueller7
0 upgraded, 1 newly installed, 0 removed and 0 not upgraded.
Need to get 0B/3053kB of archives.
After unpacking 3048kB of additional disk space will be used.
Committing changes...
Preparing...            ########################################################### [100%]
1: stardict-mueller7    ########################################################### [100%]
Done.
```

Команда **apt-get install** имя_пакета используется и для обновления уже установленного пакета или группы пакетов. В этом случае **apt-get** дополнительно проверяет, не обновилась ли версия пакета в репозитории по сравнению с установленным в системе.

При помощи APT можно установить и отдельный бинарный rpm-пакет, не входящий ни в один из репозиториев (например, полученный из Интернет). Для этого достаточно выполнить команду **apt-get install** путь_к_файлу.rpm. При этом APT проведёт стандартную процедуру проверки зависимостей и конфликтов с уже установленными пакетами.

Иногда, в результате операций с пакетами без использования APT, целостность системы нарушается, и **apt-get** отказывается выполнять операции установки, удаления или обновления. В этом случае необходимо повторить операцию, задав опцию -f, заставляющую **apt-get** исправить нарушенные зависимости, удалить или заменить конфликтующие пакеты. В этом случае необходимо внимательно следить за сообщениями, выдаваемыми **apt-get**. Любые действия в этом режиме обязательно требуют подтверждения со стороны пользователя.

Удаление установленного пакета

Для удаления пакета используется команда **apt-get remove имя_пакета**. Для того, чтобы не нарушать целостность системы, будут удалены и все пакеты, зависящие от удаляемого: если отсутствует необходимый для работы приложения компонент (например, библиотека), то само приложение становится бесполезным. В случае удаления пакета, который относится к базовым компонентам системы, **apt-get** потребует дополнительного подтверждения производимой операции с целью предотвратить возможную случайную ошибку.

Если вы попробуете при помощи **apt-get** удалить базовый компонент системы, вы увидите такой запрос на подтверждение операции:

```
# apt-get remove filesystem
Обработка файловых зависимостей... Завершено
Чтение списков пакетов... Завершено
Построение дерева зависимостей... Завершено
Следующие пакеты будут УДАЛЕНЫ:
basesystem filesystem ppp sudo
Внимание: следующие базовые пакеты будут удалены:
В обычных условиях этого не должно было произойти, надеемся, вы точно
представляете, чего требуете!
basesystem filesystem (по причине basesystem)
0 пакетов будет обновлено, 0 будет добавлено новых, 4 будет
удалено(заменено) и 0 не будет обновлено.
Необходимо получить 0B архивов. После распаковки 588kБ будет
освобождено.
Вы собираетесь совершить потенциально вредоносное действие
Для продолжения, наберите по-английски 'Yes, I understand this may be
bad'
(Да, я понимаю, что это может быть плохо).
```

Каждую ситуацию, в которой APT выдаёт такое сообщение, необходимо рассматривать отдельно. Однако, вероятность того, что после выполнения этой команды система окажется неработоспособной, очень велика.

Обновление всех установленных пакетов

Для обновления всех установленных пакетов используется команда **apt-get upgrade**. Она позволяет обновить те и только те установленные пакеты, для которых в репозиториях, перечисленных в /etc/apt/sources.list, имеются новые версии; при этом из системы не будут удалены никакие другие пакеты. Этот способ полезен при работе со стабильными пакетами приложений, относительно которых известно, что они при смене версии изменяются несущественно.

Иногда, однако, происходит изменение в именовании пакетов или изменение их зависимостей. Такие ситуации не обрабатываются командой **apt-get upgrade**, в результате чего происходит нарушение целостности системы: появляются неудовлетворённые зависимости. Например, переименование пакета MySQL-shared, содержащего динамически загружаемые библиотеки для работы с СУБД MySQL, в libMySQL (отражающая общую тенденцию к наименованию библиотек в дистрибутиве) не приводит к тому, что установка обновлённой версии libMySQL требует удаления старой версии MySQL-shared. Для разрешения этой проблемы существует режим обновления в масштабе дистрибутива — **apt-get dist-upgrade**.

В случае обновления всего дистрибутива APT проведёт сравнение системы с репозиторием и удалит устаревшие пакеты, установит новые версии присутствующих в системе пакетов, а также отследит ситуации с переименованиями пакетов или изменения зависимостей между старыми и новыми версиями программ. Всё, что потребуется поставить (или удалить) дополнительно к уже имеющемуся в системе, будет указано в отчёте **apt-get**, которым APT предварит само обновление.

При работе с Sisyphus для обновления системы рекомендуется использовать команду **apt-get dist-upgrade**.

---

[1] Если быть точным, этот файл может называться и иначе, другое имя файла со списком источников программ можно указать в конфигурационном файле /etc/apt/apt.conf. Подробнее о формате файла apt.conf можно узнать из руководства apt.conf(5).

[2] *http://backports.altlinux.ru*

synaptic — это графическая оболочка для APT. Она значительно проще в использовании, чем её аналоги. Вместо использования дерева для отображения пакетов synaptic основан на мощной системе фильтрации пакетов. Это значительно упрощает интерфейс и вместе с тем предоставляет гораздо больше гибкости при навигации по очень длинным спискам пакетов. Если вы хотите устанавливать и удалять пакеты и предпочитаете работать с программами с графическим интерфейсом, прежде всего попробуйте synaptic.

Краткий обзор программы

synaptic запускается из меню «Система — Менеджер пакетов (Программа управления пакетами)». При запуске программы запрашивается пароль суперпользователя (root).

Установка пакетов

Если вы хотите установить пакет, выполните следущие шаги:

- Обновите информацию о пакетах, чтобы узнать о последних доступных версиях, выбрав в меню «Редактирование — Получить сведения о пакетах» или нажав «Получить сведения» на панели инструментов.
- Отметьте пакет для установки:
    - Двойным щелчком мыши на названии пакета в списке пакетов.
    - Нажав правой кнопкой мыши на пакете и выбрав «Отметить для установки» в контекстном меню.
    - Выделив пакет и выбрав в меню «Пакет — Отметить для установки».

Если установка пакетов требует дополнительных изменений, вас спросят о подтверждении. Чтобы отметить дополнительные изменения нажмите «Применить».

- Примените отмеченные изменения, чтобы установить пакеты, нажав «Применить» на панели инструментов или выбрав в меню «Редактирование — Внести отмеченные изменения».
- Вас спросят о подтверждении. Проверьте итоговые изменения, которые будут применены. Чтобы продолжить установку подтвердите изменения, нажав «Применить».
- Во время проведения изменений вы увидите строку состояния. Подождите, пока изменения будут применены. Это может занять некоторое время в зависимости от количества изменений. После этого вы вернетесь в основное меню.

Удаление пакетов

Чтобы удалить пакет, выполните следущие шаги:

- Отметьте пакет для удаления:
    - Двойным щелчком мыши на названии установленного пакета в списке пакетов.
    - Нажав правой кнопкой мыши на пакете и выбрав «Отметить для удаления» в контекстном меню.
    - Выделив пакет и выбрав в меню «Пакет — Отметить для удаления».
- Примените отмеченные изменения, чтобы удалить пакеты (так же, как при установке пакетов).
- Вас спросят о подтверждении. Проверьте итоговые изменения, которые будут применены, после чего подтвердите изменения.
- Во время проведения изменений вы увидите строку состояния. Когда изменения будут применены, вы вернетесь в основное меню.

Установка доступных обновлений

Чтобы установить доступные обновления, выполните следущие шаги:

- Обновите информацию о пакетах, чтобы узнать о последних доступных версиях, как в случае установки пакетов.
- Отметьте пакет для обновления:
  - Двойным щелчком мыши на названии пакета с последней доступной версией в списке пакетов.
  - Нажав правой кнопкой мышки на пакете и выбрав «Отметить для обновления» в контекстном меню.
  - Выделив пакет и выбрав в меню «Пакет — Отметить для обновления».

Как и в описанных выше случаях, при необходимости дополнительных изменений вас спросят о подтверждении.

- Примените отмеченные изменения, чтобы обновить пакеты.
- Вас спросят о подтверждении. Проверьте итоговые изменения, которые будут применены. Чтобы продолжить обновление, подтвердите изменения, нажав «Применить».
- Во время проведения изменений вы увидите строку состояния. Когда изменения будут применены, вы вернетесь в основное меню.

Обновление всей системы

Менеджер пакетов Synaptic предусматривает два метода выделения пакетов для обновления:

Обновление по умолчанию
: По умолчанию обновляются только установленные пакеты. Если более свежая версия пакета зависит от неустановленных пакетов или конфликтует с уже установленным пакетом, обновление не будет отмечено.

«Умное» обновление (Dist-Upgrade)
: Метод «умного» обновления старается разрешить конфликты между пакетами интеллектуально. Это включает установку дополнительных требуемых пакетов и предпочтение пакетов с более высоким приоритетом.

«Умное» обновление также известно как dist-upgrade при использовании apt-get в интерфейсе командной строки.

Замечание: обновления до более свежих выпусков операционной системы должны производиться с помощью метода «умного» обновления.

Чтобы обновить вашу систему до самой последней версии выполните следущие шаги:

- Обновите информацию о пакетах, чтобы узнать о последних доступных версиях.
- Отметьте все возможные изменения, нажав «Отметить для обновления» на панели инструментов или выбрав в меню «Редактирование — Отметить для обновления».
- Выберите метод обновления («умное» обновление).
- Примените отмеченные изменения, чтобы обновить пакеты.
- Проверьте итоговые изменения. Чтобы продолжить обновление подтвердите изменения, нажав «Применить».
- Во время проведения изменений вы увидите строку состояния. После применения изменений вы вернетесь в основное меню.

Совет: вы можете изменить метод обновления по умолчанию в настройках synaptic (меню «Настройки — Параметры»).

synaptic — это очень мощная программа, обладающая множеством функций. Для более детального ознакомления со всеми её возможностями необходимо ознакомиться с документацией. Она доступна в html формате: /usr/share/synaptic/html/index.html.

Дополнительная документация

Linux для тех, кто хочет разобраться

Как это устроено

Работа с оборудованием в Linux («Сага о Драйверах»)

Когда компьютеры назывались «электронно-вычислительными машинами», они были размерами в среднем с кухонный гарнитур и занимались почти исключительно вычислениями. Ввод и вывод данных воспринимался пользователями ЭВМ — учёными-математиками — как нечто необходимое, но к работе ЭВМ имеющее лишь косвенное отношение. Учёного было довольно просто обучить, чтобы он составлял программы и оформлял входные данные для расчётов одним каким-нибудь способом, например, при помощи перфокарт. Подключение к компьютеру какого-нибудь другого устройства было делом трудоёмким, так как требовало усилий и электронщика, и программиста. Да и нужно это было нечасто.

Нынешний компьютер — игрушка не учёного, а любого рядового обывателя. Это бытовой прибор. Мало того, компьютер — это «самый умный» бытовой прибор: если имеется какой-нибудь другой бытовой прибор (скажем, кофеварка), прогрессивный обыватель тут же задумывается, нельзя ли обучить компьютер *управлять* этим прибором (скажем, варить кофе за минуту до приезда хозяина). В идеале получается «электронный дом», в котором работу любого оборудования можно контролировать, не вставая из-за рабочего места, или даже не садясь за него — посредством сети Интернет.

Самое поверхностное суждение об оборудовании и компьютере — что для подключения прибора нужна волшебная субстанция по имени «драйвер». Есть драйвер — компьютер оборудование «видит», нет драйвера — «не видит».

Это суждение во многом неверно.

Что такое «оборудование»?

Что и как можно подключить к компьютеру? Во-первых, на поверхности его корпуса обычно наблюдается множество разнообразных отверстий и разъёмов, очевидно предназначенных для того, чтобы туда что-то подключали. Уже подключены: клавиатура, мышь, монитор, возможно — принтер, наушники или колонки. Много отверстий остаётся неиспользованными, но и аппаратуры в «электронном доме» ещё много — от КПК до той же кофеварки (если на ней есть соответствующий разъём).

Во-вторых, внутри компьютера имеются специальные разъёмы для подключения к ним **плат расширения**: устройств, выглядящих не как бытовой прибор, а скорее как деталь самого компьютера. Таковы видеоадаптеры, сетевые адаптеры, «внутренние» модемы и т. п. Эти устройства — главный источник «Саги о Драйверах», потому что их много, и создатели каждого такого устройства желают сохранить его *устройство* втайне от конкурентов, прилагая к ним вместо документации ту самую волшебную субстанцию с пометкой «нажмите кнопку “Пуск” и попытайтесь расслабиться: от вас уже ничего не зависит».

В-третьих, *ещё более внутри* компьютера есть какие-то устройства, которые нельзя ни отключить, ни подключить, однако они используются при работе, имеют какое-то название и на разных компьютерах могут весьма отличаться. Например, звуковые подсистемы могут быть интегрированными, а могут быть выполненными в виде платы расширения, отличаясь редкостным разнообразием моделей и однообразием функций (разъём для микрофона, разъём (ы) для колонок, линейный вход... что-то ещё?).

Или устройство, *к которому* подключаются жёсткие диски: оно может быть рассчитано на 1 диск, 2, 4, иногда — более, иметь разные дополнительные свойства... и тоже требовать «драйвера» — по крайней мере, поддержки со стороны системы.

Что точно отличает один прибор от другого — это внешний вид разъёма, с помощью которого они подключаются к компьютеру. Очевидно, приборами, подключаемыми к разъёмам разного типа, машина управляет существенно по-разному. Более того, разъёмы настолько различны, что соединительный кабель одного типа просто не влезет в разъём другого[1]. Но всё равно, это не решает проблемы идентификации: например, мышь, подключённая к разъёму (**порту**) USB, отлично работает, а с цифровой фотокамерой как-то спроста не получается. Опять «драйвер» нужен?

Можно добавить, что некоторое оборудование вообще не нуждается в том, чтобы машине объявляли о его существовании: так, что бы ни подключалось к аналоговому звуковому входу, работать оно будет одинаково, компьютер не отличит колонки от наушников, да и отсутствия их не заметит. Словом, наружное наблюдение не даёт достаточно информации о том, как работать с оборудованием. На помощь должна прийти документация, но если в ней опять встретится слово «драйвер», оно может означать что угодно: слишком оно неопределённое.

Как распознаётся оборудование?

Попробуем внести определённость. Какую информацию относительно подключаемого прибора получает компьютер, и как он её получает?

Очевидно, «с той стороны», каждого разъёма, (допустим, USB, в который мы воткнули flash-диск), имеется какое-то оборудование, которое позволяет им пользоваться (как и USB-мышкой, USB-принтером и т. п). Это оборудование:

- определяет тип подключённого устройства
- управляет им (может, например, выключить или включить)
- передаёт на это устройство данные и/или принимает их оттуда

Такое оборудование называется **шиной** (bus). Этимология этого слова — что русского, что английского — загадочна и восходит к доисторическим временам, когда компьютеры назывались «ЭВМ».

Шин в компьютере несколько (грубо говоря — по количеству различных типов разъёмов). Есть совсем «глупые» шины — например, порт последовательного ввода-вывода (к нему подключаются мыши и прочая аппаратура «старого образца»). Глупость их в том, что информацию о типе подключённого оборудования приходится задавать вручную — либо заранее, либо с помощью наводящих вопросов пользователю.

Есть шины весьма умные, способные опросить и понять множество характеристик подключённого устройства. Такова, например, шина PCI — наиболее распространённое на сегодня оборудование для подключения плат расширения. Любопытный пользователь может посмотреть список устройств, подключённых к шине PCI с помощью команды lspci (от «list PCI», команда из пакета pciutils):

```
[tmpuser@arnor tmpuser]$ lspci

0000:00:00.0 Host bridge: VIA Technologies, Inc. VT8377 [KT400/KT600 AGP] Host Bridge (rev 80)
0000:00:01.0 PCI bridge: VIA Technologies, Inc. VT8237 PCI Bridge
0000:00:0f.0 RAID bus controller: VIA Technologies, Inc. VIA VT6420 SATA RAID Controller (rev 80)
0000:00:0f.1 IDE interface: VIA Technologies, Inc. VT82C586A/B/VT82C686/A/B/VT823x/A/C PIPC Bus Master IDE (rev 06)
0000:00:10.0 USB Controller: VIA Technologies, Inc. VT82xxxxx UHCI USB 1.1 Controller (rev 81)
0000:00:10.1 USB Controller: VIA Technologies, Inc. VT82xxxxx UHCI USB 1.1 Controller (rev 81)
0000:00:10.2 USB Controller: VIA Technologies, Inc. VT82xxxxx UHCI USB 1.1 Controller (rev 81)
0000:00:10.3 USB Controller: VIA Technologies, Inc. VT82xxxxx UHCI USB 1.1 Controller (rev 81)
0000:00:10.4 USB Controller: VIA Technologies, Inc. USB 2.0 (rev 86)
0000:00:11.0 ISA bridge: VIA Technologies, Inc. VT8237 ISA bridge [KT600/K8T800/K8T890 South]
0000:00:11.5 Multimedia audio controller: VIA Technologies, Inc. VT8233/A/8235/8237 AC97 Audio Controller (rev 60)
0000:00:12.0 Ethernet controller: VIA Technologies, Inc. VT6102 [Rhine-II] (rev 78)
```

0000:01:00.0 VGA compatible controller: ATI Technologies Inc Radeon R100 QD [Radeon 7200]

Пример 1. Команда lspci

Из устройств на иллюстрации только одно — видеокарта Radeon 7200[2] — в действительности является платой расширения, все остальные интегрированы в системную плату (бывает и по-другому). Тип устройства — «Multimedia audio controller», «Ethernet controller», «VGA compatible controller» и т. п. — лишь небольшая часть информации, которую шине рассказали о себе подключённые к ней устройства.

К шине PCI в качестве устройства подключена другая шина — USB, служащая для подсоединения внешних устройств. Она тоже довольно умная, а ещё отличается тем, что устройства подключаются к ней и отключаются от неё довольно часто. Существует команда lsusb (из пакета, естественно, usbutils), но ей, как и lspci приходится пользоваться нечасто (она даже убрана в каталог /usr/sbin, с глаз пользовательских долой):

```
[tmpuser@arnor tmpuser]$ /usr/sbin/lsusb
Bus 005 Device 001: ID 0000:0000
Bus 004 Device 001: ID 0000:0000
Bus 003 Device 001: ID 0000:0000
Bus 002 Device 002: ID 046d:c00c Logitech, Inc. Optical Wheel Mouse
Bus 002 Device 001: ID 0000:0000
Bus 001 Device 003: ID 08ec:0012 M-Systems Flash Disk Pioneers
Bus 001 Device 001: ID 0000:0000
```

Пример 2. Команда lsusb

Пример показывает пять USB-шин (это совпадает с данными lspci), к первой из которых подключён flash-диск, а ко второй — мышь[3]. Как правило, устройство определяется шиной, после чего специально обученная системная программа производит все действия, необходимые для того, чтобы этим устройством можно было воспользоваться. Например, для flash-диска потребовалось дополнительно загрузить **модуль ядра** usb_storage, да вдобавок **смонтировать** содержимое диска в каталог /media/usbdisk.

Специальный каталог /sys отражает представление системы о присоединённых к ней устройствах. В частности, все найденные на шинах устройства перечислены в виде подкаталогов /sys/bus/*шина*/devices. Если устройство установлено, а умная шина, наподобие PCI или USB, его не заметила — скорее всего неполадка аппаратная (несовместимое или неисправное устройство, таракан в разъёме и т. п.).

Увы. Бывает и так: устройство (видеокарта, модем, кофеварка) на шине появилось, а воспользоваться им не удаётся. Видимо, чего-то не хватает... драйвера?

Что такое «драйвер» и где он находится?

А в самом деле, чего может не хватать, если устройство распозналось, марка устройства — известна и как передавать данные по шине — тоже известно? Не хватает главного: сведений о том, *какие* данные надо передавать, чтобы добиться от устройства желаемого эффекта. Что передать по шине USB, чтобы кофеварка выключилась? Какие байты записать в последовательный порт модема, чтобы он повесил трубку? Что сделать с видеокартой, чтобы... всё было быстро и непременно 3d!?

Это вот «какие данные» — и есть «драйвер». Драйвер может быть где угодно, на любом уровне системы: от модуля ядра до куска пользовательской программы и даже её конфигурационного файла.

Типичные варианты:

- Драйвер — **модуль ядра**, подсказывающий шине, как правильно обращаться с устройством. Это,

как правило, относится к PCI-устройствам и стандартным USB-устройствам. Подключается к ядру командой modprobe *имя_модуля* (или insmod). Распознанные и классифицированные устройства (те, для которых есть драйвер-модуль ядра) отображаются в виде подкаталогов /sys/class/*класс_устройства*/.

- Драйвер видеокарты — модуль графической подсистемы X11 (X.Org). Подгружается при старте графической оболочки, достаточно лишь указать его в настройках X.Org (с помощью конфигуратора или вручную, в файле /etc/X11/xorg.conf). Часто требуется и специальный модуль ядра (возможно, несколько), организующий доступ к видеопамяти.
- Драйверы принтера и модема — описание характеристик для, соответственно, подсистемы печати и программы-«звонилки». Что с ними делать дальше, расскажет документация.
- Драйвер — прикладная программа или дополнение (plug-in) к ней (например, драйвер сканера — дополнение утилиты sane, а с некоторыми цифровыми проигрывателями звука «iRiver» можно взаимодействовать с помощью утилиты ifp из пакета ifp-line). Здесь главное — название программы, а драйвер, скорее всего, уже включён в дистрибутив.

В последнем случае нет никакой зримой информации о том, что устройством действительно можно пользоваться — до тех пор, пока не запущена соответствующая прикладная программа с соответствующими настройками (особенно это касается «глупых» шин наподобие последовательного порта). И в любом случае самостоятельная установка «драйвера» должна сопровождаться вдумчивым чтением документации к нему.

Опять «устройство»?

В документации Linux термин «**устройство**» (device) часто используется не в значении «прибор», а в значении «элемент каталога /dev». Что это такое?

Прибор подключается к машине, как правило, для того, чтобы передавать на него какие-то данные и/или получать их оттуда. Если задача компьютера — *управлять* внешним устройством, это всё равно можно рассматривать как передачу управляющих данных и приём диагностических. Во многих случаях передачу данных проще всего вести в синхронном (поточном) режиме, точно так же, как это делается при работе с файлом: открыть файл — записать данные — закрыть файл или открыть — прочитать — закрыть. Если бы можно было представить внутренность прибора в виде файла, работа с ним пошла бы легче: это означало бы, что система знает, как и куда передавать данные, а дело пользовательской программы — эти данные понимать[4].

В большинстве случаев именно так и устроено в Linux. После того, как система распознала внешнее устройство, а служба hotplug, при необходимости, загрузила соответствующий модуль ядра, в каталоге /dev заводится новый «файл», содержимое которого отражает содержимое подключённого устройства, не занимая при этом места на жёстком диске. Такой файл называется **файлом-дыркой**, его можно представить как отверстие в файловой системе, через которое видно не содержимое жёсткого диска, а данные, попадающие туда с «другой стороны» — со стороны подключённого внешнего устройства. Например, гибкий диск в дисководе представляется в виде файла-дырки /dev/fd0, (от **f**loppy **d**isk 0), а мышь — в виде /dev/mouse (строго говоря /dev/mouse — этот обычно **символьная ссылка** на актуальный файл-дырку — скажем /dev/psaux, порт PS/2).

В документации вместо «файл-дырка» чаще всего пишут просто «устройство» (device), а устройство-прибор — «внешним устройством». Если соответствующего устройства в каталоге /dev/ нет — значит, в цепочке его распознавания есть слабое звено.

Стоит напомнить, что файл-дырка, однако, не обязан существовать и непременно соответствовать одному внешнему устройству. Устройства, подключаемые ко второму последовательному порту, например, всегда видны как /dev/ttyS1 (а к первому — как ttyS0). Фактически, ttyS — это файл-дырка шины, настолько простой, что дальнейшее выяснение типа устройства перекладывается на программу пользователя.

Другой пример — это работа с аппаратурой по шине USB. USB-шин в системе зарегистрировано несколько, и к каждому можно подключить одно или несколько устройств. Для них независимо от типа устройства заводятся файлы-дырки вида usbdev.*шина*.*номер*, где каждому новому подключённому

устройству просто выдаётся очередной *номер*. Некоторые звуковые проигрыватели и цифровые фотокамеры распознаются как flash-диски; тогда в дополнение к нетипизированному файлу-дырке создаётся одна или даже несколько дисковых (допустим, само дисковое устройство /dev/sda и единственный раздел с файловой системой на нём /dev/sda1, который и монтируется в /media/usbdisk). Другие фотокамеры умеют больше, чем обычный диск: например, транслировать изображение и/или звук, и для них существуют специальные утилиты, например, gphoto2. В этом случае никакого дополнительного файла-дырки, за исключением /dev/usbdev.*шина*.*номер*, не создаётся, и о типе подключённого аппарата догадывается сама gphoto2.

Кто виноват и что делать?

Итак, свежеподключённый к компьютеру прибор не распознался «сам собой», и программы, которые должны были с ним работать, не работают.

1. Для начала стоит посмотреть на **системную консоль** (клавиши Ctrl + Alt + F12) и в файл /var/log/messages, возможно, системная диагностика подскажет, в чём дело
2. Неполадки могут быть аппаратными (проверяется в /sys/bus или с помощью lspci, как сказано выше).
3. Hotplug или другая программа авторaспознавания может не знать про конкретный подключённый прибор (придётся в режиме суперпользователя вручную загрузить модуль с помощью modprobe, а чтобы не делать этого каждый раз — отредактировать /etc/modules.conf).
4. Возможно, внешнее устройство распозналось и модуль для него есть, но служба udevd, которая заводит файлы-дырки в /dev, выбрала другое название или вовсе не завела нужного устройства (надо проанализировать содержимое /dev и, возможно, настроить udev или саму прикладную программу).
5. Ваше устройство может быть слишком новым, а дистрибутив Linux — оказаться слишком старым. В этом случае рекомендуется обновить части системы, содержащие «драйвер» (в зависимости от ситуации — модуль ядра, само ядро, графическую оболочку или её библиотеку, прикладную программу, и т. п.).
6. Стоит проверить, что сказано о вашем устройстве в сетевых информационных ресурсах (здесь поможет lspci или подобные ей утилиты, а также http://www.google.com). Вполне вероятно, там посоветуют загрузить некий заранее собранный модуль ядра (назовут его, конечно, «драйвером») либо подскажут, какую именно программу следует использовать.
7. Наконец, ваше устройство может просто не поддерживаться. Печально, но факт: некоторые производители аппаратуры настолько дорожат своими мелкими секретами, что не только не документируют устройство своих устройств, но тщательно скрывают его. Как следствие, Linux-сообщество не в состоянии быстро обеспечить поддержку таинственного прибора. Производители предпочитают писать «драйверы» — хорошие ли, плохие — за свои деньги, а особо жадные ограничиваются только одной, самой распространённой на сегодня пользовательской программной платформой. И это пока, к сожалению, не Linux.

Сказанное выше означает, что после каждого обновления системы устройство, ранее распознававшееся с трудом или вовсе не распознававшееся, может преспокойно заработать, особенно если это устройство относительно новое. Кроме того, стоит со всем вниманием относиться к ситуации, когда производитель прибора не просто анонсирует совместимость с Linux, а предлагает «драйверы» собственного изготовления. И последнее: если вы не в силах справиться с «драйвером» в одиночку — обращайтесь к Linux-сообществу! Вы или получите решение задачи, или в очередной раз подтвердите, что её стоит решать — и тем самым приблизите решение.

1Однако можно, например, подключить наушники вместо микрофона, причём они, скорее всего, *будут* работать микрофоном... правда, очень тихо.
2Она подключена к шине AGP, которая архитектурно похожа на PCI, поэтому система различия не делает.
3Некоторая путаница может возникнуть из-за того, что строгого соответствия между разъёмами на корпусе и номерами шин нет: «кто первый встал, того и тапки».
4Ну и где тогда находится «драйвер»? да какая разница...

Графический интерфейс в Linux

Кирилл Маслинский, Максим Отставнов

Оконная система X и её реализации

Графический интерфейс не является неотъемлемой частью Linux — это просто одна из её компонент, такая же необязательная с точки зрения архитектуры системы, как, например, программа для рисования изображений. Но для тех программ, которые используют графические ресурсы, эта компонента предоставляет возможность работать с графическими объектами (линиями, прямоугольниками, цветами), ничего не зная о деталях работы конкретных устройств графического вывода (видеокарты и монитора). Это похоже на то, как ядро скрывает от программ детали работы с конкретным оборудованием, например, жёстким диском, предоставляя им работать с файлами. Поэтому комплекс программ, предоставляющий доступ к графическим ресурсам, называют *графической подсистемой*. В Linux функции графической подсистемы выполняет оконная система «Икс».

Графическая подсистема с точки зрения операционной системы представляет собой группу обычных процессов, управление которыми производится общесистемными средствами. Точно так же, общесистемными средствами производится и управление процессами, запускаемыми «из-под» этой графической среды. Графическая подсистема отнюдь не монополизирует использование компьютера; параллельно с её работой продолжает исполняться множество служебных системных процессов; с других терминалов (если система многотерминальная) могут запускаться другие программы или даже другие графические подсистемы.

Оконная система Икс (от англ. X window system, далее — просто X) — один из самых больших и успешных проектов в истории компьютерной техники — восходит к 1984 г., когда разработчики двух систем компьютерной графики, претендующих на универсальность — проектов Athena (Массачусетский технологический институт) и W Windowing (Стэнфордский университет) — решили объединить свои усилия. Подробнее об истории этого проекта можно узнать, например, из статьи в Wikipedia.

Тогда перед ними стояла задача создать систему компьютерной графики, позволяющую совместно использовать самые разные компьютерные платформы. Решением стало создание специального протокола X, который позволял разделить программы-клиенты и сервер, предоставляющий графические ресурсы, отсюда и возможность исполнять программу-клиент на одном компьютере, сервер на другом, а данные между ними передавать по сети.

Проект этот был настолько наукоёмок и настолько полно охватывал тогдашнюю область задач, связанных с графикой, что серьёзных альтернатив ему так и не возникло. С тех пор X прошла через одиннадцать основных релизов (отсюда другое название — X11, представляющее собой название и текущую версию протокола) и множество версий. И возникновение, и вся история развития X тесно связаны с ОС UNIX, а теперь, естественно, и Linux. Тем не менее, реализации X доступны и для нескольких альтернативных архитектур ОС, включая Windows NT.

Существует несколько реализаций X, дальнейшее изложение будет ориентировано на две широко распространённые свободные реализации: XFree86 и XOrg. XFree86 изначально создавалась для семейства процессоров Intel 386, и вплоть до 2004 года была самой популярной реализацией X. XFree86 поддерживает беспрецедентно широкий спектр оборудования (оно и понятно, учитывая существующий «зоопарк» видеокарт и устройств ввода для платформы PC). Благодаря доступности исходных текстов и пользовательской аудитории в десятки миллионов человек, XFree86 весьма устойчива и хорошо оттестирована, по крайней мере, насколько это возможно для такого разнообразия поддерживаемого оборудования.

В последние годы параллельно с XFree86 развивается основанная на тех же исходных текстах X Window System графическая подсистема XOrg. До недавнего времени по спектру поддерживаемого оборудования, архитектур и функциональности XOrg мало чем отличалась от XFree86, и сейчас они

тоже примерно эквивалентны с точки зрения пользователя. Однако направление развития этих двух проектов, состав их разработчиков и лицензионная политика несхожи. В ближайшем будущем вполне вероятно, что XOrg обгонит XFree86 и по возможностям, и по частоте использования.

Большинство из того, о чем будет говориться в последующих разделах, справедливо для любой реализации X на любом оборудовании и под любой ОС, список которых можно найти на http://www.X.org.

Цветной бутерброд

Большинство пользователей, установив систему, получают в своё распоряжение готовую графическую среду. Однако то, что сидящему за монитором представляется сплошной графической операционной средой, реализовано как многослойный бутерброд технологий. Попробуем разобраться в его устройстве «по слоям».

«Чистая» X

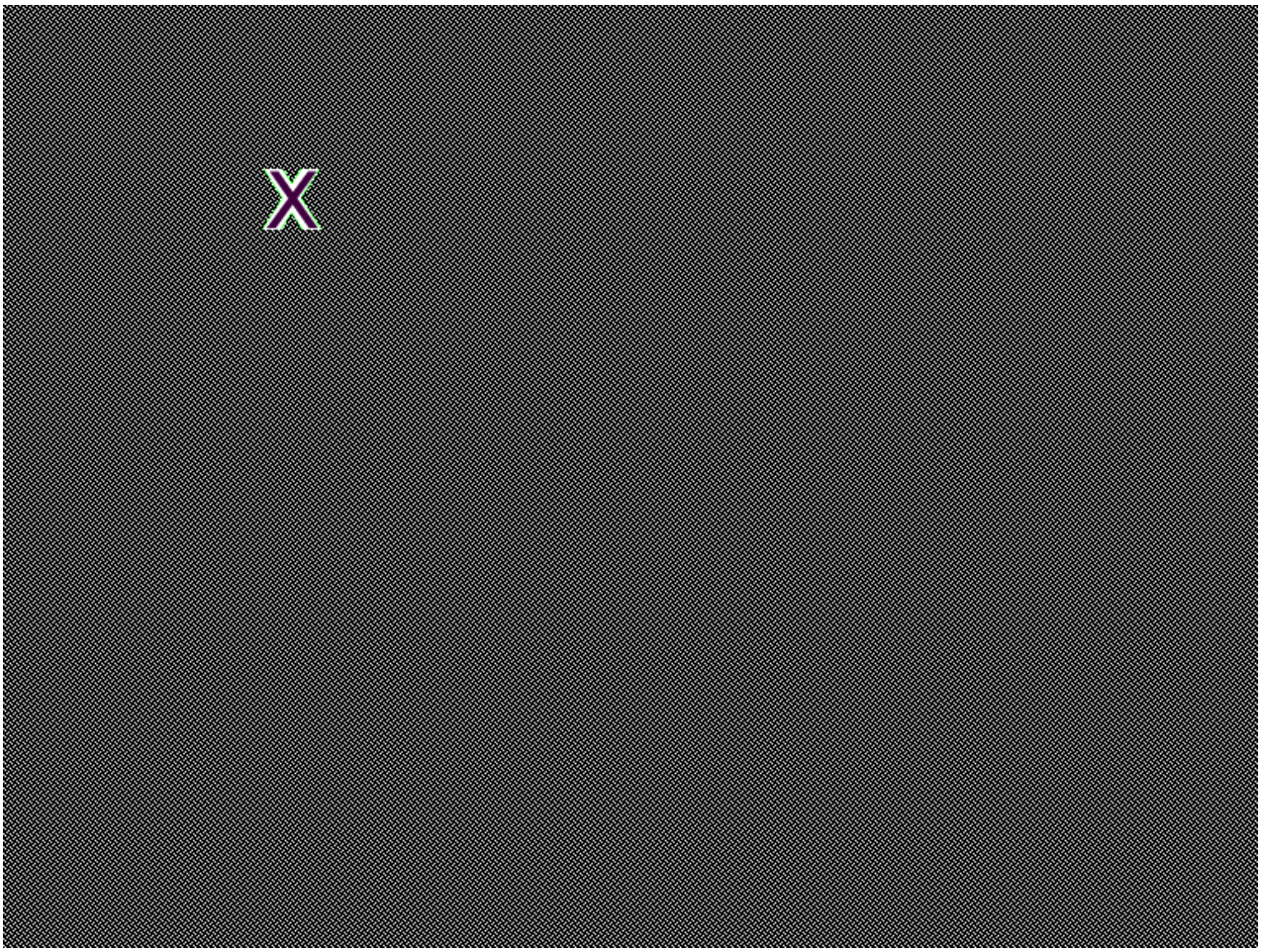
Непосредственно с оборудованием (видеосистемой, устройствами ввода и динамиком) работает *X-сервер*. Перечисленное оборудование в совокупности называется *X-терминалом* (аппаратным X-терминалом называется и специализированный компьютер, на котором исполняется исключительно X-сервер). X-сервер захватывает оборудование и предоставляет возможность выводить на эти устройства и получать от них ввод другим программам — X-клиентам — по особому протоколу, который так и называется, X-протокол.

Здесь сразу видно важное технологическое достоинство X: взаимодействие X-сервера с X-клиентами происходит по специфицированному протоколу, который может туннелироваться через TCP/IP и, соответственно, клиенты и сервер могут исполняться на разных узлах сети. Это означает, что одни и те же программы могут эксплуатироваться в разных топологиях, включая совокупность автономных рабочих станций (персональных компьютеров), совокупность рабочих станций без данных или бездисковых рабочих станций (локальная сеть), многопользовательскую систему с X-терминалами (или какую-либо смешанную топологию).

Ещё одним ресурсом, который предоставляет X-сервер, являются шрифты. Оперировать шрифтами он может самостоятельно, либо с помощью другой программы, которая обеспечивает масштабирование шрифтов — *фонт-сервера* (от англ. font — шрифт).

На Рисунок 1, «Чистая X» показана «чистая» оконная система X — когда запущен только X-сервер и никаких приложений — то, с чем большинство пользователей никогда не сталкивается. Запустить её обычно можно, подав команду: **X &**.

Рисунок 1. Чистая X

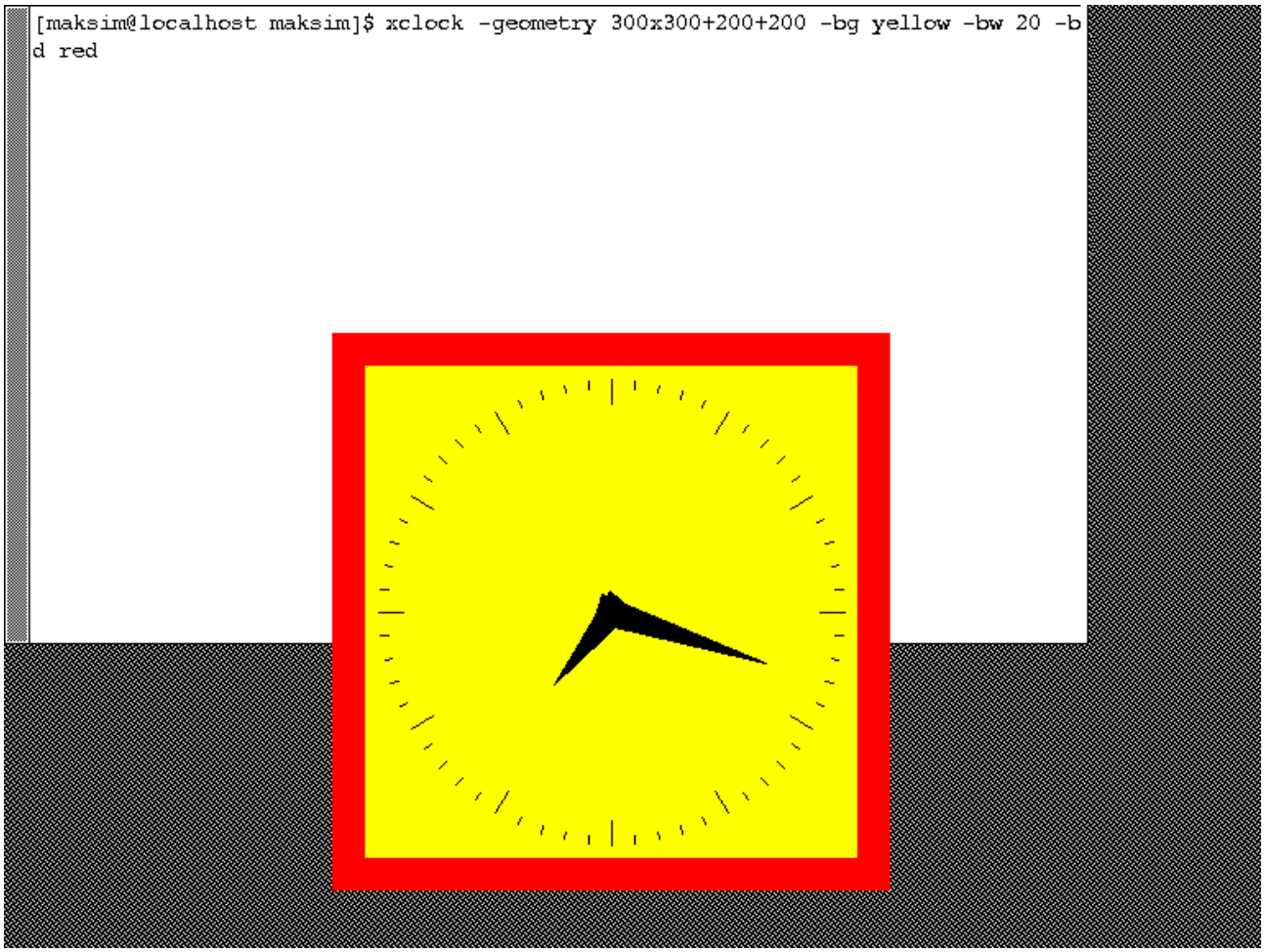


Мы видим традиционный серый экран с не менее традиционным курсором в виде буквы «x». Используя мышь или другое координатное устройство, курсор можно перемещать по экрану. На нажатие кнопок мыши и клавиш никакой видимой реакции не следует. И невидимой тоже — сервер готов передавать эти сигналы своим клиентам, а клиенты пока не запущены. Хотя на самом деле некоторые комбинации клавиш X перехватывает и обрабатывает. Это Zap (**Ctrl-Alt-Backspace**) — завершение работы сервера (если эта возможность не запрещена при конфигурации), Zoom (**Ctrl-Alt-+** и **Ctrl-Alt--**) — «горячее» переключение доступных видеорежимов. В некоторых ОС (Например, GNU/Linux) **Ctrl-Alt** в сочетании с функциональной клавишей освобождает оборудование и передаёт его на время соответствующей *виртуальной консоли* — запущенной в неграфическом режиме (обычно, vga-режиме) программе, позволяющей пользователю зарегистрироваться в системе и получить доступ к командной строке оболочки.

Рисунок 2. xterm в среде X

Воспользуемся последней возможностью, перейдём на консоль и запустим первое клиентское приложение: программу xterm (Рисунок 2, «xterm в среде X»). На экране X появилось окно, а в окне можно видеть интерфейс клиентского приложения. В данном случае интерфейс текстовый, а приложение представляет собой эмулятор терминала, на котором запущена командная оболочка системы по умолчанию. С эмулятором можно делать все то же, что и с обычным терминалом: издавать команды, получать результат и запускать другие программы. Если программы текстовые (строчные или оконные), исполняться они будут в том же окне, а если графические (как и сам xterm) — в отдельных окнах.

Рисунок 3. xclock в среде X



Запустим программу xclock (Рисунок 3, «xclock в среде X»). При её запуске мы использовали несколько параметров, задающих геометрию (местоположение и размер) вновь порождаемого окна, цвет его фона и шрифта по умолчанию, толщину и цвет рамки. Эти (и некоторые другие) параметры типичны для программ, построенных на основе графической библиотеки X Toolkit. Все значения параметров, заданные при вызове программы, кроме геометрии окна (местоположения на экране и размера), могут быть перекрыты самим запускающимся приложением. Дело в том, что окно выделяется клиентскому приложению при запуске, и все доступные ему ресурсы этим окном и ограничены — это свойство X-протокола. Переместить окно или изменить его размер средствами «чистой» X невозможно.

Запустив несколько экземпляров того же xterm (и почитав документацию) можно обнаружить, что и «чистая» X умеет не так мало. Например, оперирует буфером обмена текстом между приложениями и предоставляет текстовым приложениям такой ресурс, как полосу прокрутки (полоса сбоку окна, при помощи которой можно листать текст вверх или вниз, щёлкая по ней левой и правой кнопками мыши соответственно, — это наследие проекта Athena).

Итак, ключевой компонент графической платформы — X-сервер:

- захватывает оборудование;
- создаёт по запросу других программ (которые в этой терминологии называются *X-клиентами*) окна;
- предоставляет другим программам возможность работы в окнах, т. е. вывода информации в эти окна и обработки сигналов от устройств ввода (клавиатуры и мыши или другого координатного устройства), когда окно, назначенное программе, является активным. Предоставление ресурсов возможно в том числе и через сеть, когда клиент и сервер работают на разных компьютерах (узлах).

В среде, образуемой X-сервером, окно, выделяемое клиенту, является фиксированным: его геометрия задаётся при запуске клиента и сохраняется в течение всего сеанса работы с этим клиентом. Есть ли польза от системы, работающей с фиксированными окнами? Да, это вполне соответствует цели создания специализированных систем с графическим интерфейсом пользователя. Примером такой системы может служить терминал, позволяющий получать справки о расписании поездов и стоимости проезда, установленный на вокзале. Однако этих возможностей совершенно недостаточно для универсального «настольного» применения компьютера.

При универсальном применении компьютера характерна поочерёдная работа с различными программами (иногда достаточно большим их количеством), причём пользователь может отрываться, допустим, от редактирования текста, чтобы поработать с иллюстрацией при помощи другой программы, прочитать почту или заглянуть на интернет-страницу, затем возвращаться к редактированию текста и т. д. Эти возможности обеспечивает другая программа — *менеджер окон*, представляющая собой следующий «слой» в графической среде пользователя.

Менеджеры окон

Для одновременной и поочерёдной работы с разными программами, требуется возможность управлять окнами (с помощью клавиатуры или мыши), т. е. возможность изменять «на лету» их геометрию (положение и размеры), а также (обычно не относимое к геометрии) положение в воображаемой «стопке» окон — от этого зависит, какое из окон будет «верхним» (видимым полностью), если окна перекрывают друг друга на плоскости экрана.

Управление окнами и составляет основную функцию *оконного менеджера*. Устоявшийся англоязычный термин window manager, относящийся к этому классу программ, мы будем передавать далее словосочетанием-калькой «оконный менеджер», которое, впрочем, не представляется особенно удачным, так же, как и встречающиеся в литературе «менеджер окон», «администратор окон» и «диспетчер окон».

Технически ограничение на изменение геометрии однажды выделенного X-сервером окна преодолевается оконным менеджером за счёт того, что ему в качестве окна выделяется весь экран. Окно во весь экран может быть названо *корневым*, по аналогии с корневым каталогом в файловой системе. На самом деле, менеджер окон — не единственная программа, способная работать с корневым окном; например, входящая в комплект поставки X утилита xsetroot позволяет установить цвет фона или поместить на него рисунок.

Прикладным программам, таким образом, выделяются далее уже не окна собственно X, а окна оконного менеджера. Изменение положения окна в подавляющем большинстве случаев ничего не требует от программы-клиента, однако желательно, чтобы программа была достаточно «сообразительной», чтобы изменить своё поведение при изменении размеров выделенного ей окна «на лету». Это справедливо для большинства, но не для всех программ (в частности, этого «не умеют» многие старые программы и некоторые компьютерные игры).

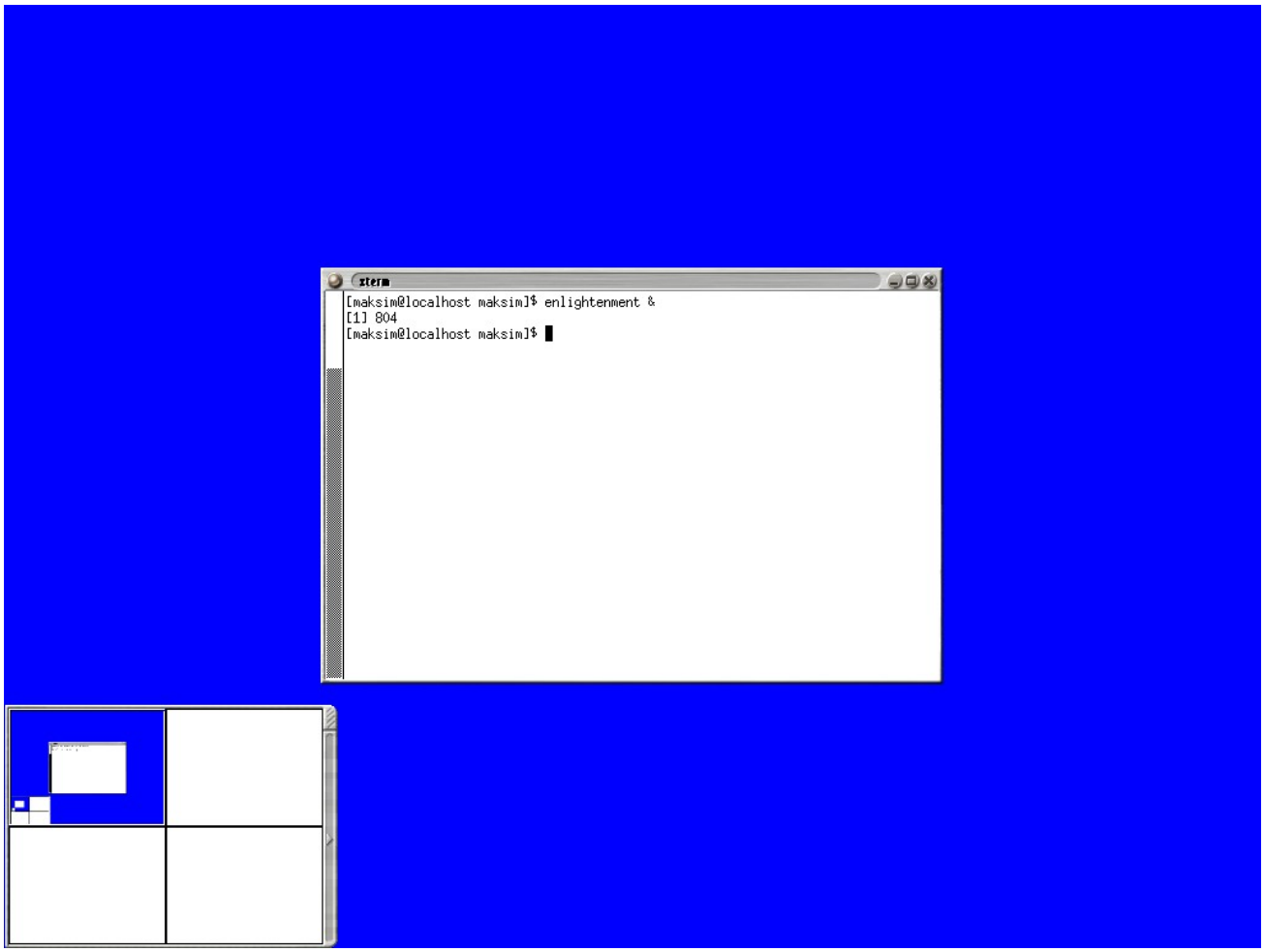
В свою очередь, и оконный менеджер может быть достаточно «умен», чтобы понять, что программа не реагирует на изменение геометрии окна, и запретить пользователю изменять размеры окна для данной программы (чтобы он не оказался в ситуации, когда ему видна лишь часть области вывода программы или наоборот, часть окна прикладной программы пуста). Однако такое решение может привести к весьма дискомфортным ситуациям (например, если при запуске программы её окно оказывается больше экрана)[1].

Менеджеров окон существует превеликое множество — под любой набор задач, которые может решать графическая многооконная система. Их настолько много, что выбрать какой-нибудь в качестве «типичного представителя семейства» затруднительно. Поэтому выберем один из самых развитых — Enlightenment.

«Просвещение» (англ. enlightenment) создано Карстеном Хайцлером и Джеффом Харрисоном (Carsten Haitzler, Geoff Harrison). До 2000 г. он был основным менеджером окон в популярной среде GNOME, затем уступив это место менее функциональной, но более быстрой «Рыбе-пиле» (Sawfish). Он

продолжает оставаться GNOME-совместимым, и многие пользователи этой популярной среды предпочитают его, хотя и без GNOME у Enlightenment поклонников хватает.

Рисунок 4. Оконный менеджер Enlightenment



Запустим Enlightenment (Рисунок 4, «Оконный менеджер Enlightenment»), набрав в командной строке xterm **enlightenment &**. Первое, что мы видим — это появившиеся вокруг окна нашего xterm элементы оформления: рамка и строка заголовка с кнопками. Окно теперь можно перемещать по экрану, «ухватив» за заголовок, масштабировать (изменять размер), «взяв» за бок или за угол, максимизировать, минимизировать или закрыть, нажав соответствующую кнопку. Спрашивается, что ещё можно делать с окном?

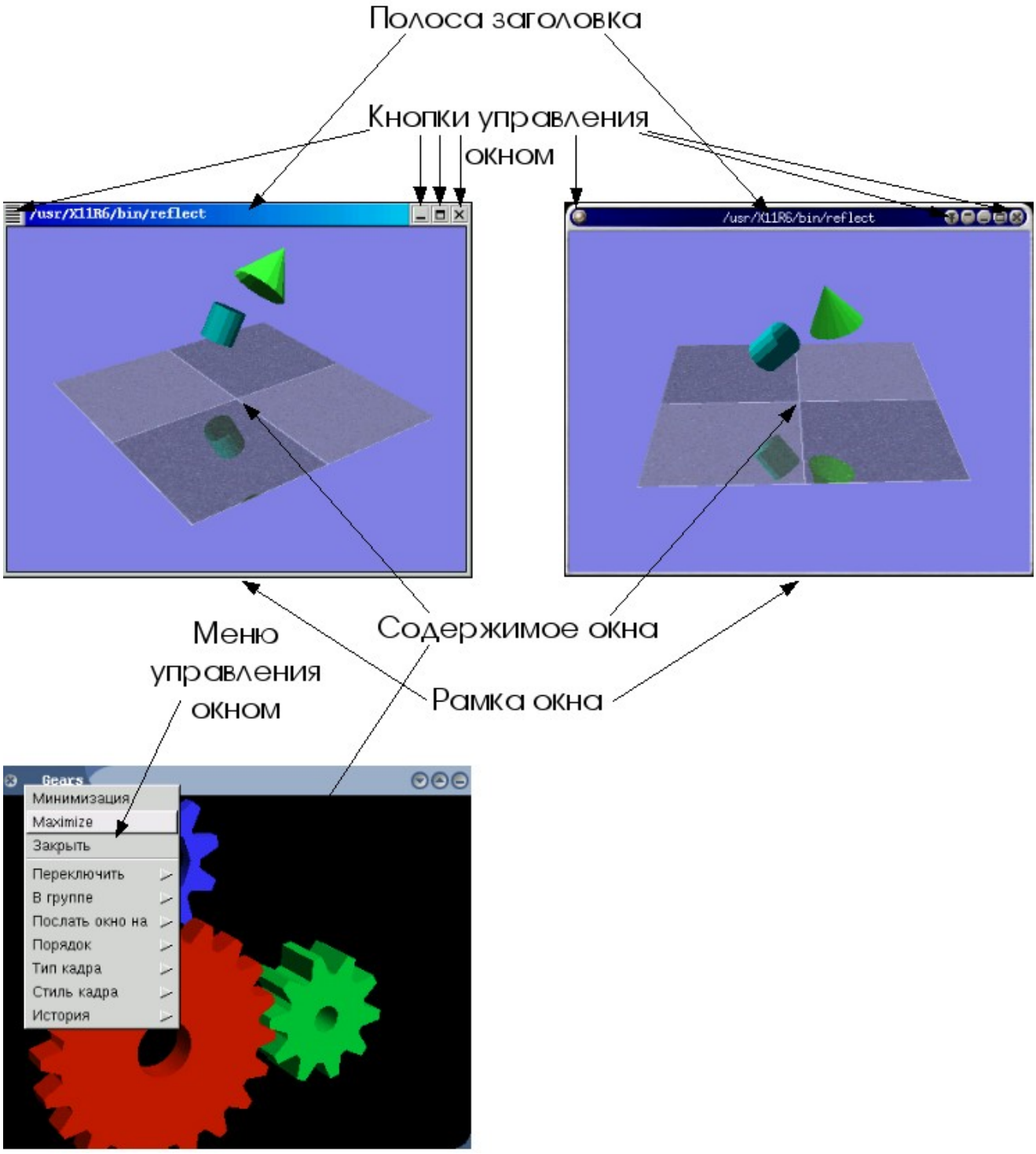
Рисунок 5. Оконное меню Enlightenment

Вопрос не праздный. Нажав на левую кнопку в заголовке, получаем неожиданно разнообразное меню таких действий (Рисунок 5, «Оконное меню Enlightenment»). Оказывается, его можно ещё уничтожить

(Annihilate), поднять/опустить (Raise/Lower), скрутить/раскрутить (Shade/Unshade) приклеить/отклеить (Stick/Unstick) и выполнить ещё массу действий, для которых потребовались отдельные меню. Набор этих действий зависит от конкретного менеджера окон (и Enlightenment — один из самых богатых возможностями), а то, какие из них выведены в строку заголовка отдельными кнопками, — от того, как он настроен (пользователем или по умолчанию).

Базовая (а также расширенная) функциональность оконных менеджеров доступна пользователю прежде всего за счёт введения в интерфейс так называемых *виджетов* (от англ. widgets, сокращение от window gadgets, «оконные приспособления»). Виджеты — это рамки, кнопки, меню и пр., которые служат «органами управления» окна. Технически (в терминах оконной системы X) виджеты представляют собой отдельные окна, примыкающие к окну прикладной программы и, как правило, перемещающиеся вместе с ним.

Рисунок 6. Виджеты



Обрамление окна обычно составляют следующие элементы:

Рамка, окружающая окно

> При «буксировке» рамки мышью окно изменяет свой размер. Иногда для изменения размера окна предназначены только выделенные «уголки» рамки, представляющие собой отдельные виджеты.

Полоса заголовка

> Часто совпадает с одной из (обычно, верхней) сторон рамки. В полосе заголовка может содержаться название программы или запустившая программу команда, а также другая информация, специфичная для окна. При «буксировке» полосы заголовка перемещается все окно. Со «щелчками» различными кнопками мыши на полосе заголовка также могут быть связаны различные действия по управлению окнами.

Кнопки управления окном

> Часто вынесенные на полосу заголовка или в другое место рамки кнопки позволяют выполнить с ним такие действия, как закрытие (часто сопровождающееся выходом из программы, открывшей окно), максимизация (разворачивание окна на весь экран), минимизация/сворачивание, вызов меню управления окном, которое может содержать весьма обширный репертуар других действий.

Детали реализации обрамления окна могут быть весьма различными в зависимости от конкретного оконного менеджера и его настроек.

Откуда берутся такие ресурсы, как виджеты, их декор и способ поведения? Конечно, менеджер окон может содержать их в себе. Но такой подход не очень характерен для открытых систем, одним из принципов разработки которых является компонентность. Поэтому виджеты и их функции обычно объединяются в стандартные библиотеки (toolkits), которые могут быть использованы множеством различных программ. Большинство развитых менеджеров окон, менеджеров рабочего стола и разработанных специально для них приложений можно сгруппировать по библиотекам виджетов, с опорой на которые они разработаны.

С точки зрения пользователя виджеты, составляющие обрамление окна, часто воспринимаются как его часть. Однако не следует забывать, что внутри окна (содержимым которого управляет уже прикладная программа, а не менеджер окон) зачастую тоже есть свои виджеты: кнопки, полосы прокрутки, переключатели, меню и т. п. В общем случае, используемые оконным менеджером и прикладной программой библиотеки виджетов могут и не совпадать.

Собственно, управление окнами — основная функция оконного менеджера, и на этом его функциональность может и заканчиваться. Однако большинство из них выполняют по крайней мере ещё одну функцию.

Рисунок 7. Пейджер

Вы уже обратили внимание на то, что при запуске Enlightenment на экране появилось ещё одно окно.

Это так называемый *пейджер* (pager), на Рисунок 7, «Пейджер» он изображён крупным планом. На пейджере представлена миниатюрная копия экрана, обновляющаяся в режиме реального времени, причём, если подвести курсор к изображению отдельного окна, оно увеличивается и рядом высвечивается название приложения, запущенного в нем. Но почему экран занимает только четверть окна пейджера? Потому что оконный менеджер позволяет оперировать «виртуальным» столом (от англ. virtual desktop, также *рабочим столом*), по размеру превышающим физический экран, а пейджер — одно из средств перемещения физического экрана по рабочему столу. Enlightenment позволяет создавать до 64 экранов на рабочем столе.

Менеджер окон, который помимо управления окнами обладает рядом дополнительных функций, может использоваться в качестве операционной графической среды пользователя, предоставляющей полный спектр возможностей для параллельной работы с несколькими задачами. Наиболее часто такими дополнительными функциями являются следующие:

Минимизация/сворачивание окон и управление свёрнутыми окнами

Работа на экране, «захламлённом» десятком различных окон, может быть дискомфортной, и крайне полезна возможность *свернуть* (минимизировать) окно со временно неиспользуемой программой. Для того, чтобы средствами графической среды можно было окно затем развернуть, оно и в свёрнутом состоянии должно каким-то образом отображаться. Существует несколько относительно распространённых способов отображения свёрнутых окон. Например, «на столе» может оставаться полоса заголовка свёрнутого окна, по щелчку на которой оно вновь разворачивается. Свёрнутым окнам могут соответствовать *пиктограммы* (иконки, значки) на поверхности рабочего стола или в специально отведённом для этого окне (*панели управления*). Свёрнутые окна могут отображаться как пункты общего или специального меню (см. ниже).

Управление несколькими рабочими столами

Практика показывает, что для многих продвинутых пользователей, которые осваивают открытые системы, уже имея опыт работы в характерных для ПК альтернативных системах, именно возможность работать на нескольких рабочих столах оказывается решающим плюсом оконной системы X. Действительно, переключение между виртуальными рабочими столами позволяет организовать комфортную работу со множеством программ даже на мониторах с относительно низким разрешением (1024x728, 800x600) и физическими размерами (17, 15-дюймовыми). В иных условиях комфортность работы существенно снизилась бы, или настоятельной необходимостью стало бы приобретение более крупного и ёмкого монитора (что зачастую влечёт за собой необходимость смены графической карты и прочих недешёвых мероприятий). Все современные оконные менеджеры поддерживают виртуальные рабочие столы, правда называются они везде по-разному: столы, рабочие области или экраны. До предела (чтобы не сказать, до абсурда) эта функциональность развита в оконном менеджере Enlightenment, который позволяет организовать до 64 экранов на рабочем столе, при этом рабочих столов также может быть более одного (точнее, до 32). Трудно представить, зачем может понадобиться две тысячи с лишним отдельных экранов (как правило, четырёх экранов хватает с избытком для любых практических задач), однако возможности приёма демонстрируются этим в полный рост.

Быстрый запуск команд

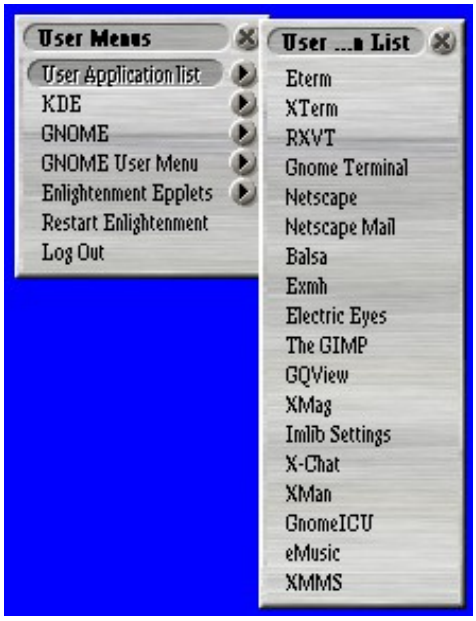
Возможность быстрого запуска предуготовленных команд обычно ассоциируется с *общим меню* (главным меню), вызываемым щелчком мыши на особом виджете, не связанном с прикладными окнами, или в свободной от прикладных окон области экрана.

Настройка внешнего вида и поведения среды

Поведение — реакция отдельных виджетов на операции с ними, модель фокусировки (способ переключения *активного* в данный момент окна, с которым связан ввод с клавиатуры и мыши)

и т. п. Поведение и внешний вид оформления окон, а также наличие на экране общих виджетов, не связанных с конкретными прикладными окнами, *обои* (цвет фона или изображение в корневом окне) и т. п. могут варьировать в очень широких пределах. Иногда возможности такой настройки считают некими «архитектурными излишествами», однако более взвешенной является точка зрения, согласно которой в хорошем визуальном дизайне (так же, как и в хорошей архитектуре) ничто не является излишеством.

Рисунок 8. Меню настройки Enlightenment



Пример возможностей настройки менеджера окон даёт тот же Enlightenment. Его меню настройки можно увидеть, щёлкнув правой кнопкой мыши на фоне экрана (Рисунок 8, «Меню настройки Enlightenment»). Исследовав возможности настройки, можно обнаружить, что сказанное выше о способах работы с этим менеджером окон весьма условно, потому что поменять можно буквально все, от декора виджетов до количества и функций элементов оформления окон и их реакции на различные действия.

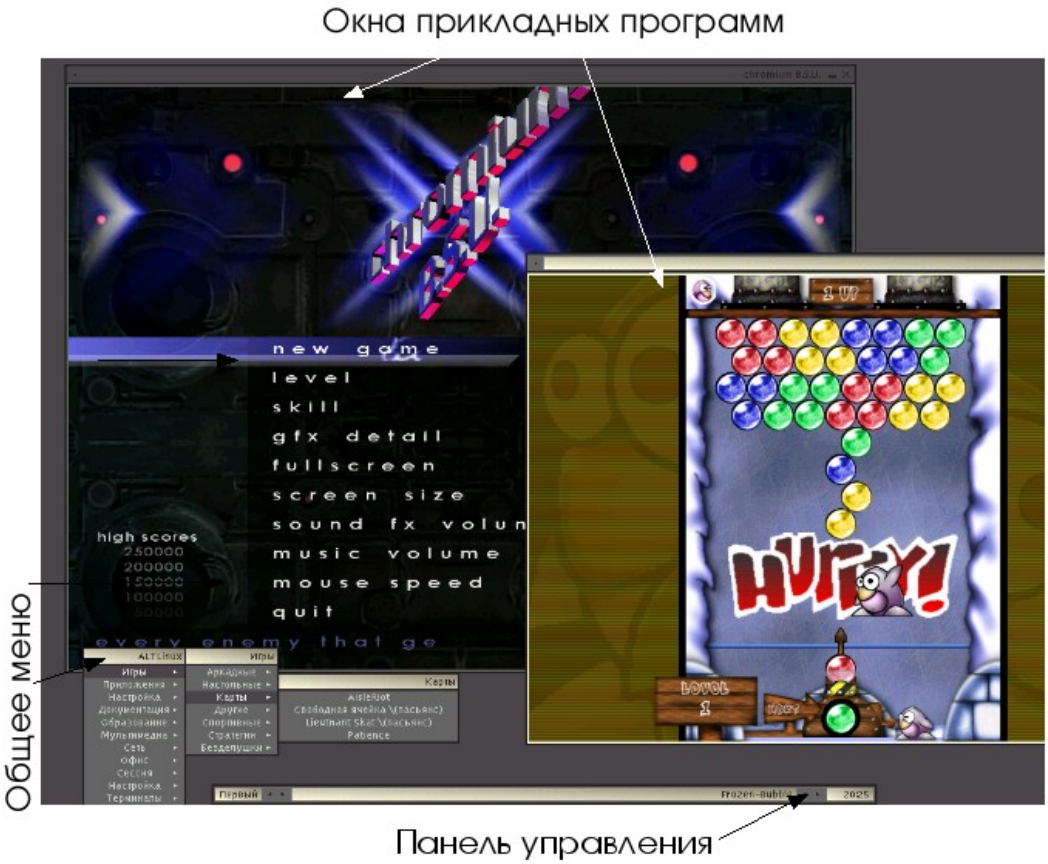
Лишь один пример: сколько способов визуализировать перемещение окна вы знаете? Разработчики Enlightenment придумали целых шесть, включая фантастический «полупрозрачный». Настройки и расширения Enlightenment можно объединять в *темы* (англ. themes) и обмениваться ими.

Выше речь шла о таких свойствах Enlightenment, которые предположительно являются общими для всех оконных менеджеров. Но при этом чаще обычного употреблялись слова «обычно», «как правило», «может» и т. п. Это связано с чрезвычайным разнообразием решений на базе распространённых оконных менеджеров. Ниже более подробно и определённо речь пойдёт ещё о нескольких оконных менеджерах.

Оконные менеджеры BlackBox и FluxBox

BlackBox — один из самых компактных и быстродействующих оконных менеджеров. Он позволяет эффективно организовать работу на рабочем столе, не «захламляя» его ненужными ссылками и не расходуя экранное пространство на отображение громоздких элементов оформления.

Рисунок 9. Оконный менеджер BlackBox



Наряду с базовой функциональностью, BlackBox предоставляет (факультативно) панель, содержащую кнопки переключения между рабочими столами (по умолчанию их четыре) и заголовки открытых окон. Общее меню вызывается щелчком правой кнопкой мыши на свободном от окон месте рабочего стола. Меню (или любое из вложенных в него меню) щелчком по заголовку может быть превращено в окно, остающееся на экране до явного его закрытия щелчком на соответствующей кнопке.

По умолчанию на полосе заголовка каждого окна присутствуют кнопки сворачивания (сворачивание можно выполнить также двойным щелчком на самом заголовке), максимизации и закрытия окна. Свёрнутое окно присутствует на экране в виде полосы заголовка, развернуть его можно повторным двойным щелчком на полосе заголовка или из меню Workspaces (рабочие области), доступного по щелчку средней кнопкой мыши на свободном от окон месте рабочего стола. Это же меню позволяет перейти на другой стол, добавить или удалить стол из рабочего пространства.

BlackBox поддерживает различные модели фокусировки ввода. Фокусировка по щелчку мыши (от англ. click to focus) позволяет реализовать стиль работы, привычный для пользователей KDE: окно становится активным (принимающим текущий ввод с клавиатуры и от мыши) после щелчка на нем. Активное окно автоматически становится верхним (видимым полностью, даже если оно частично перекрывается с другими окнами). Небрежная фокусировка (от англ. sloppy focus) предполагает активизацию окна при попадании на него курсора мыши (окно при этом не «всплывает» автоматически наверх).

Наряду с панелью и конвертируемыми в дополнительные окна-панели меню, BlackBox реализует ещё один автономный виджет — так называемую *щель* (англ. slit). Щель располагается на краю видимого экрана и может содержать маленькие (без обрамления) окна специализированных программ (их существует около десяти), индицирующих какие-либо состояния среды или позволяющих быстро выполнить часто исполняемые действия.

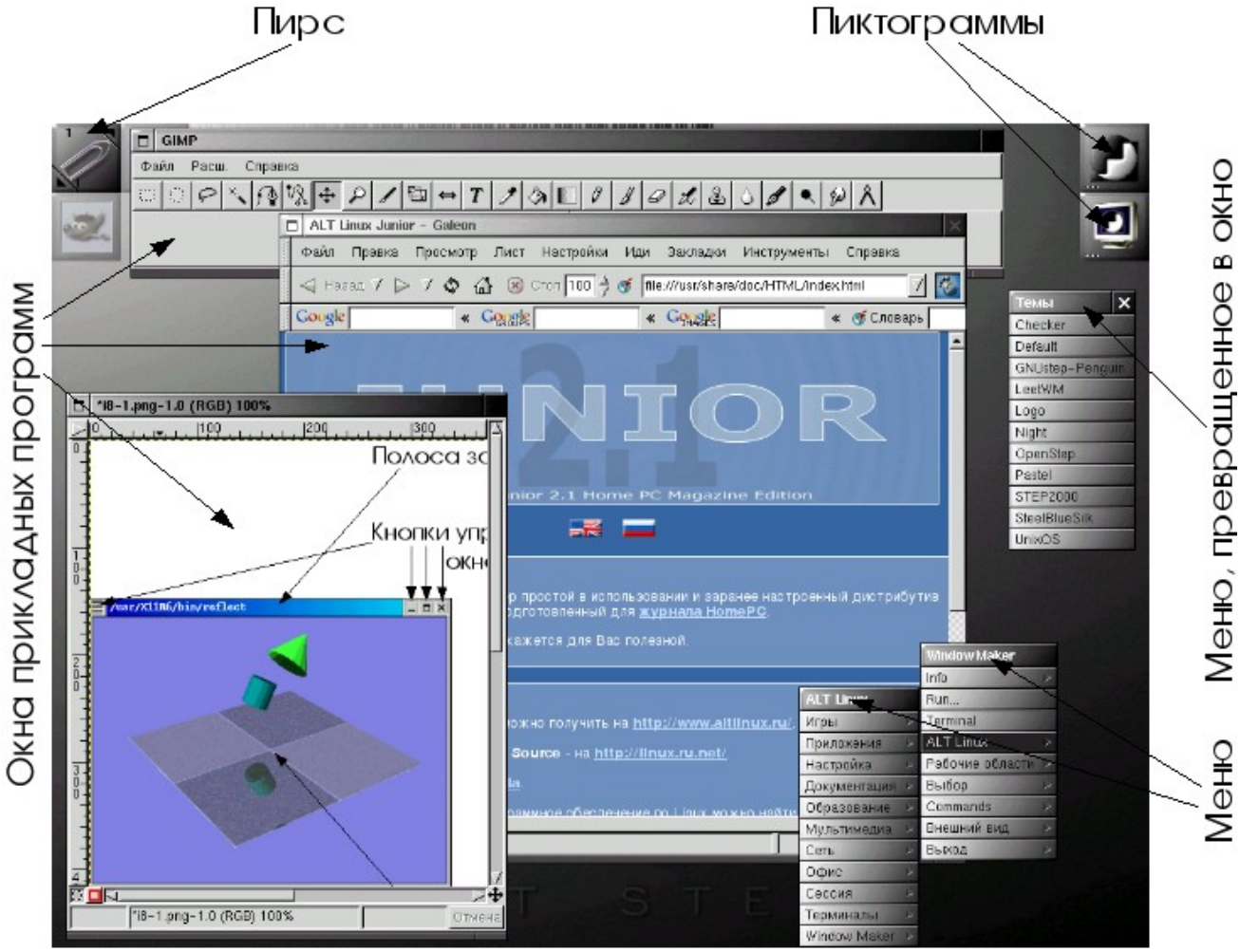
На основе BlackBox созданы два более развитых оконных менеджера — OpenBox и более популярный FluxBox.

Внешний вид BlackBox, FluxBox и OpenBox легко настраивается с помощью механизма тем рабочих столов.

Оконный менеджер WindowMaker

WindowMaker (WM) — это свободная реализация (в рамках проекта GNUStep) концепций NextSTEP — первой получившей более или менее широкую известность универсальной графической среды пользователя. За недоступностью оригинальной NextSTEP для современных платформ, познакомиться с WM полезно и поучительно вне зависимости от того, собираетесь ли вы с ним работать. Это позволит увидеть исходную точку развития графических сред и оценить продуктивность (или непродуктивность) того, с чем эти идеи стали ассоциироваться со временем.

Рисунок 10. Оконный менеджер WindowMaker



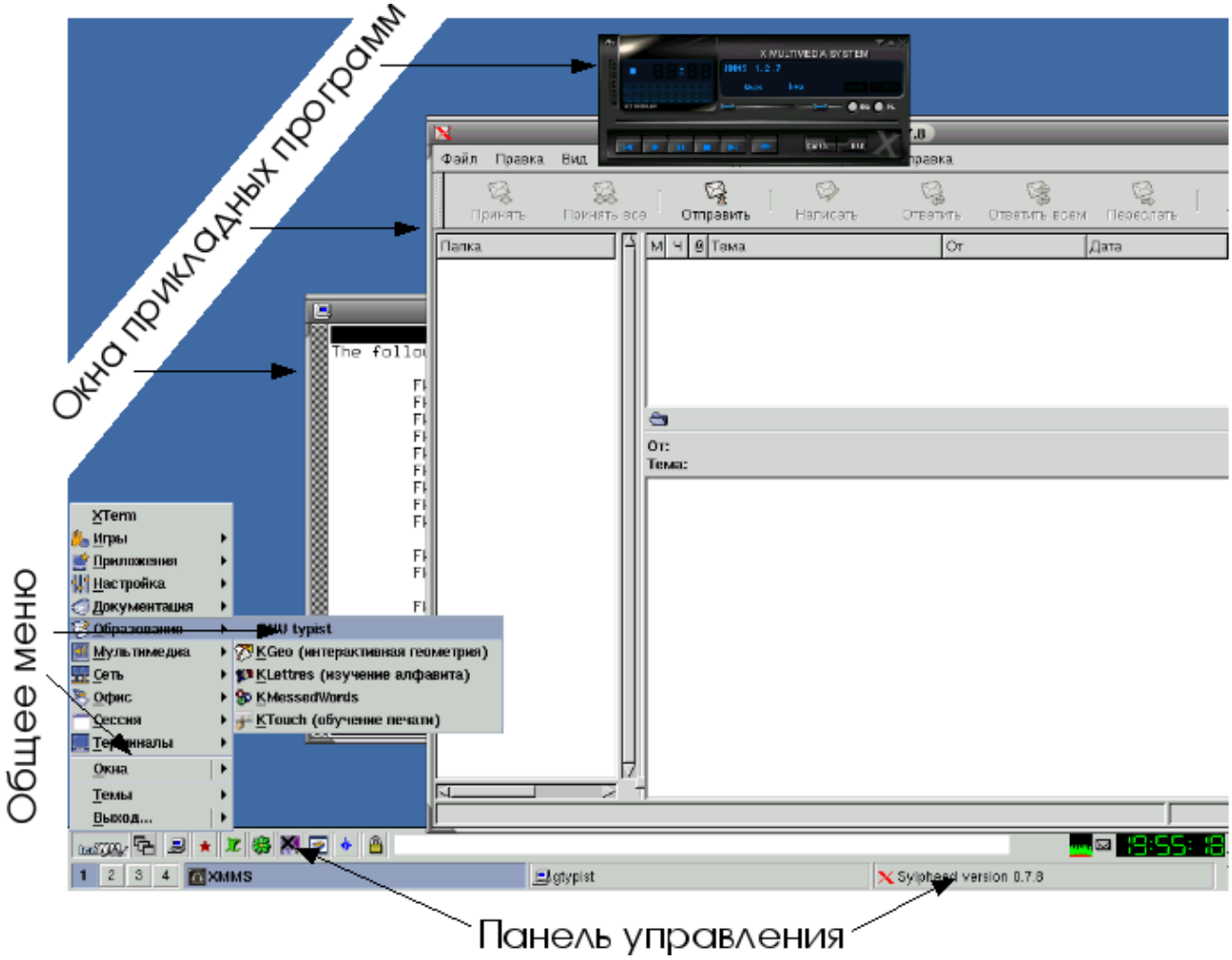
Основным автономным виджетом WM, как и NextSTEP, является *пирс* прикладных программ, представленный при запуске пиктограммой со скрепкой. При запуске любой корректной (с точки зрения WM), а также некоторых некорректных программ, кроме её окна на экране появляется её пиктограмма. Если «пришвартовать» эту пиктограмму к пирсу, она там и останется, позволяя запускать эту программу вновь и вновь простым щелчком по ней — это разработанный для NextSTEP интегрирующий интерфейс.

WM позволяет работать с несколькими рабочими столами (переключение по умолчанию при помощи клавиш **Alt-n** или через меню, доступное по щелчку правой кнопкой мыши на свободном месте рабочего стола). WM очень гибко настраивается, как в отношении внешнего вида, так и в отношении поведения, причём большая часть настроек доступна из специальной графической утилиты, запускаемой по щелчку на пиктограмме с изображением ступеньки.

Оконный менеджер IceWM

IceWM — простой оконный менеджер, его очень часто выбирают пользователи, переходящие с Microsoft Windows или OS/2, поскольку он достаточно точно повторяет основные черты привычной для них графической рабочей среды.

Рисунок 11. Оконный менеджер IceWM



Из автономных виджетов прежде всего стоит отметить панель с кнопкой, вызывающей главное меню (подобно тому, как это делает кнопка в Microsoft Windows, GNOME или KDE). С помощью панели можно также управлять текущим сеансом и настраивать IceWM. Впрочем, основное меню также доступно и по щелчку правой кнопкой мыши на свободном месте рабочего стола, что более привычно для пользователей WindowMaker, Sawfish, BlackBox или Enlightenment.

Панель также содержит список запущенных программ (включая те, окна которых минимизированы), на неё можно вывести и *мини-терминал*, позволяющий оперировать командной строкой. Для выполнения любого действия может быть назначена специальная клавиатурная комбинация.

IceWM также позволяет работать с множеством рабочих столов (рабочих мест), которые нумеруются или именуются пользователем.

Интегрированные графические среды

Существует два подхода к тому, как можно достроить оконную систему до полнофункциональной среды, позволяющей пользователю решать все (или почти все) его практические задачи. Во-первых, можно расширить функциональность менеджера окон, добавив в него недостающие возможности. Чего не хватает в оконном менеджере до полнофункциональной среды? Возможности запускать программы и утилиты. Достигается это обычно при помощи организации специального меню. С примером этого

подхода мы уже познакомились в предыдущем разделе: этим путём пошли разработчики Enlightenment и ряд других проектов. Во-вторых, можно добавить в «графический бутерброд» ещё один слой — *менеджер рабочего стола* — работающий «поверх» менеджера окон и использующий функциональность последнего. Этим путам идут команды разработчиков GNOME и KDE, и для пользователей этих графических сред «бутерброд» становится уже трёхслойным: оконная среда X, оконный менеджер и менеджер рабочего стола.

С точки зрения пользователя нет чёткой границы между менеджерами окон с расширенной функциональностью и менеджерами рабочего стола, работающими «поверх» менеджера окон, поскольку они обеспечивают одну и ту же функциональность и нередко даже графически организованы сходным образом. Оба варианта предоставляют пользователю возможность работать в *графической среде* (desktop environment).

Интегрированная графическая среда предполагает не только единство оформления, но и трактовку объектов в рабочем пространстве (окон, файлов, пунктов меню и т. п.) как физических объектов, которые можно перемещать, выбрасывать в «корзину» и т. д. Однако сколько-нибудь последовательной теории интегрированных графических сред не существует. Изучая отдельные среды в динамике их развития, можно, тем не менее, выделить несколько общих черт.

- Они опираются на определённый интерфейс разработчика (API), состоящий из библиотек, доступных также разработчикам прикладных программ (будь то MS Windows API для Microsoft Windows, Motif для CDE, Qt для KDE или GTK+ для GNOME);
- Они реализуют элементы объектной метафоры: файлы, процессы (их потоки ввода-вывода) изображаются как отдельные объекты, на них можно фокусироваться и выполнять с ними различные действия, их состояния и изменения этих состояний также могут визуализироваться или озвучиваться. Целостная объектная метафора своей реализации не нашла (и, видимо, последовательно объектная среда была бы крайне неудобной в использовании).
- Они реализуют единообразные элементы управления (виджеты), зачастую не только в оформлении отдельных окон, но и в их содержимом.
- Они содержат те или иные элементы управления, не привязанные к отдельным окнам прикладных программ (общие меню, панели управления, поверхность стола и т. п.).
- Они позволяют согласованно изменять свойства интерфейса образующих среду программ (менеджера окон, менеджера рабочего стола, приложений, разработанных специально для данной среды).
- Они реализуют *буфер обмена*, позволяющий передавать типизованные данные от программы программе (оконная система X содержит буфер, позволяющий передавать данные лишь простого текстового типа).
- Они реализуют возможность «перетаскивания» при помощи мыши (drag'n'drop) объектов или данных между окнами одной программы или разных программ.

Однородность опыта при работе в интегрированных средах и связанная с нею привычность (иногда ошибочно называемая «интуитивностью», хотя она не имеет отношения к философскому и психологическому понятиям интуиции) позволяют при освоении нового инструмента-программы сосредоточиться на её прикладной логике, не задумываясь и специально не фокусируя внимания на приёмах работы, общих для всех инструментов. Это позволяет новому пользователю гораздо быстрее осваивать прикладные программы (делает более «крутой» пресловутую кривую обучения)[2].

Как ни парадоксально, основной недостаток работы в интегрированной среде является оборотной стороной основного достоинства: жёстко закреплённые навыки мешают при выходе за её пределы. Конечному пользователю, ограниченному опытом работы в одной среде, недостаёт «стереоскопичности» видения, глубины понимания; элементы эргономической логики могут напрямую ассоциироваться с определёнными визуальными элементами и «жестами», с помощью которых подаются команды.

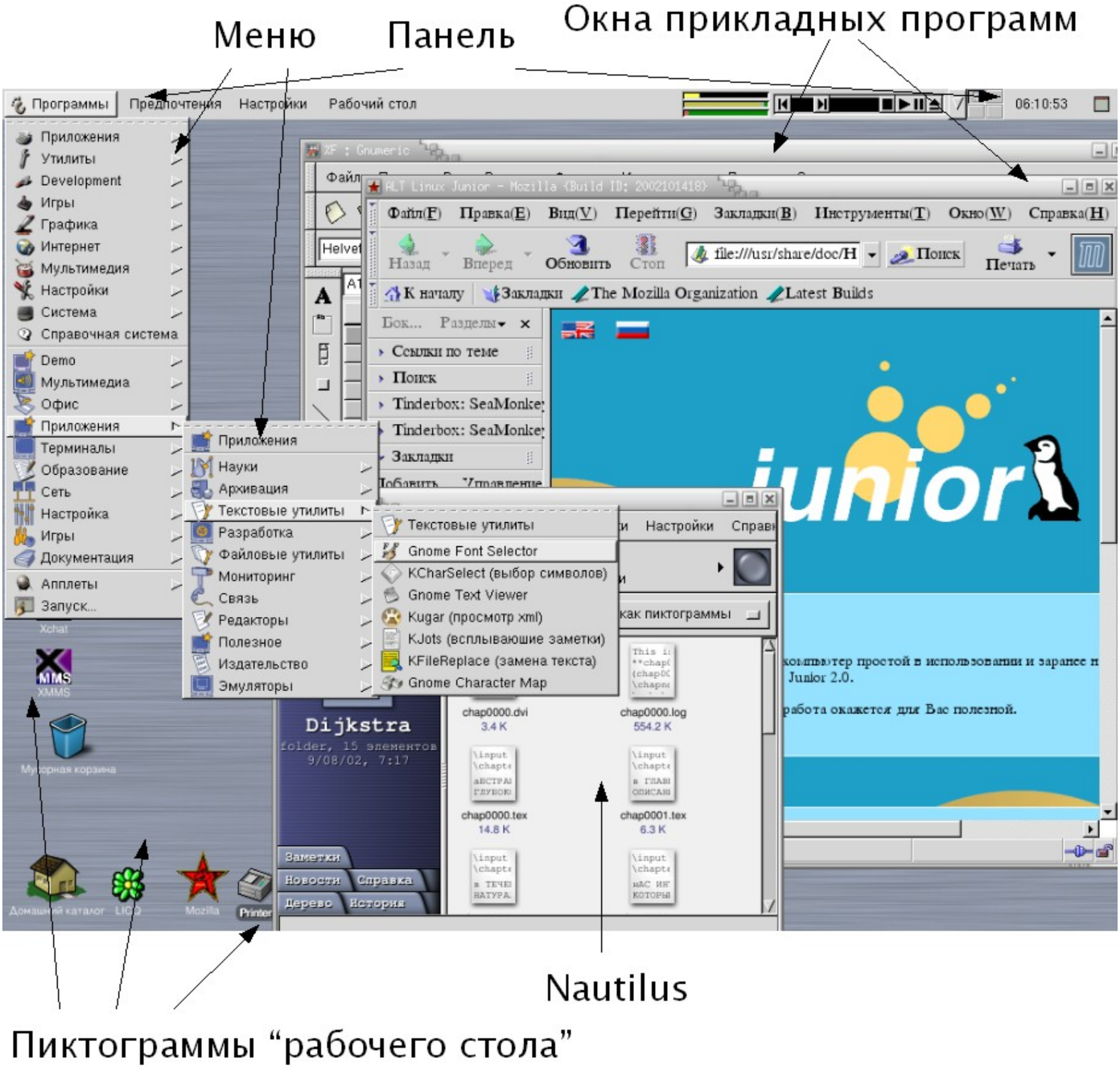
Общеизвестны сложности, с которыми сталкиваются люди, долгое время работавшие в одной

графической среде, при необходимости поработать в другой (пусть даже и весьма схожей). Для преодоления таких сложностей крайне полезным представляется знакомство с *разными средами* уже на начальном этапе освоения графических интерфейсов. Это не обязательно должны быть разные интегрированные среды, но само представление о том, что один и тот же результат может достигаться с помощью разных интерфейсных средств весьма важно. В общем случае это возможно и в рамках одной интегрированной среды из числа рассматриваемых ниже — и KDE, и GNOME в высшей степени гибки в отношении настройки внешнего вида и поведения.

На сегодня существуют и развиваются две свободные интегрированные графические среды общего назначения: KDE и GNOME. Они входят в поставку большинства стандартных (открытых) ОС, как свободных, так и несвободных. Хотя GNOME на полгода моложе KDE, мы начнём обсуждение именно с GNOME.

GNOME

Рисунок 12. Интегрированная среда GNOME



GNOME (GNOME, GNU Network Object Model Environment — «Среда ГНУ, основанная на модели сетевых объектов», но также и «Образцовая среда для сетевых объектов ГНУ») — один из самых амбициозных и масштабных проектов в программистском сообществе.

Кроме реализации функционально полной графической среды (возможно, уместнее говорить о сенсуальных средах, учитывая то, что звук стал их полноправной частью), GNOME претендует на то, чтобы полностью реализовать спецификации промышленной платформы сетевого взаимодействия CORBA и полностью абстрагировать слой менеджера рабочего стола (или графической среды) от низлежащего слоя управления окнами (менеджера окон).

GNOME поддерживает ряд оконных менеджеров, среди которых: Sawfish (оконный менеджер по умолчанию), Enlightenment, IceWM, WindowMaker, AfterStep и FVWM2, совместимые с GNOME, впрочем, в разной степени.

Сегодняшняя версия GNOME — полноценная интегрированная среда, включающая реализацию повседневно необходимых функций и позволяющая использовать сторонние решения для реализации функциональности, которая в ней отсутствует.

GNOME использует один из самых развитых интерфейсных пакетов GTK+, реализованный для разных платформ. Над ним надстраивается масса компонентов и библиотек, обеспечивающих сетевую функциональность, интерфейсы к различным языкам программирования, работу со звуком через механизмы ОС и пр. Сам GNOME стремится оставаться мобильным и доступным во всех открытых системах. Он стабильно работает в GNU/Linux, BSD, AIX и Solaris; последнее обстоятельство способствовало поддержке разработки GNOME, которую оказывает Sun Microsystems через созданный в 2001 г. году «Фонд GNOME», среди учредителей которого также крупнейшие поставщики свободных ОС.

С пользовательской точки зрения GNOME предстаёт как набор базовых компонентов интерфейса и *аплетов*, утилит и прикладных программ. К базовым компонентам относятся менеджер файлов и поверхности стола Наутилус (Nautilus), панели управления и меню GNOME Panel и центр управления (Gnome Control Center).

Менеджер файлов Nautilus позволяет отображать содержимое файлов и каталогов в окнах и выполнять над файлами обычные действия (удаление, переименование, копирование и перемещение и т. п.), а также осуществлять предварительный просмотр многих типов данных. Nautilus эффектен, но работа с ним не более эффективна, чем с прочими броузерами файлов, включаемыми обычно в графические среды (менеджер файлов CDE или Windows Explorer).

Помимо отображения содержимого каталогов в окнах, Nautilus также может отображать один из каталогов на поверхности рабочего стола: размещённые на нем иконки как бы приклеены к монитору, и при смене текущего экрана остаются на том же месте относительно наблюдателя (так же, кстати, ведут себя и открытые окна, если их «приклеить»).

Поддерживается широкий спектр операций переноса мышью (drag'n'drop), причём «перетаскивать» можно не только объекты (файлы, пункты меню и т. п.), но и некоторые их свойства: так, можно «взять цвет» в окне выбора цвета и перенести его на панель, которая воспримет его. Есть даже операции, позволяющие назначить один объект свойством другого: например, если на панель «перетащить» не цвет, а файл с картинкой, последняя станет её фоном. Переносить файлы между окнами Nautilus, на рабочий стол и панели можно практически без ограничений.

Уже упомянутые панели являются, наряду с менеджером файлов, важнейшей составной частью интерфейса GNOME. Панелей может быть неограниченное количество. Панель может быть двух типов: панель-меню (menu panel) и объектная панель (object panel). Первая из них содержит пункты меню и может содержать пиктограммы, а вторая — только пиктограммы. Последняя может быть краевой (edge), выровненной (aligned), скользящей (sliding) или плавающей (floating), но это скорее свойство панели (которое можно менять «на ходу»), определяющее особенности её поведения, чем тип.

Внешний вид и поведение панелей является в высшей степени конфигурируемым. Пользователь может задавать как глобальные предпочтения (анимация движения панелей, отображение панельных объектов и пр.), так и индивидуальные предпочтения для каждой из них (её тип и положение на экране, ширина,

возможность автосокрытия и принудительной минимизации, цвет и фоновое изображение и т. п.). Разумеется, пользователь может наполнять панели теми объектами, которые ему нужны.

На панелях могут присутствовать объекты пяти типов:

- *Аплет* (applet, «приложеньице») — интересный тип панельного объекта, демонстрирующий то, что он не обязан быть представлен статической картинкой. Это программа, места в панели которой достаточно, чтобы отображать какую-нибудь полезную (или забавную) информацию или даже принимать клавиатурный и/или координатный ввод. С GNOME поставляется масса аплетов, отображающих всякую полезную информацию (состояние ресурсов и статус сети, например) или позволяющих осуществить нетривиальные действия (например, mini commander, позволяющий набрать команду, не запуская терминала). Важными аплетами являются путеводитель по столу (Desktop Guide) и список задач (Task List), позволяющие переключаться между виртуальными экранами и активизировать окна запущенных программ соответственно.
- *Пускатель* (launcher) ассоциирован с приложением или командой, которые исполняются по щелчку на его пиктограмме в панели.
- *Выдвижной ящик* (drawer) — это кнопка, открывающая другую панель, перпендикулярно первой — некий аналог подменю в меню, который можно наполнить всевозможными аплетами.
- *Специальные объекты* — это те же аплеты, но выполняющие функции, которые другими средствами «достать» почему-либо нельзя (запереть экран, выйти из GNOME или запустить программу «вручную»). В качестве специальных объектов исполняются и программы, которые не были написаны специально для GNOME, но могут, тем не менее, осуществлять вывод в панель — *поглощённые программы* (swallowed applications).
- Наконец, *объект-меню* раскрывает меню.

За работу системы меню, как и за работу панелей, отвечает компонент GNOME Panel, и это не случайно: разница между панелью и меню в большей степени декоративная, чем сущностная. Любое меню можно зафиксировать на экране, и оно превратится в подобие панели-меню (только вертикальное, а не горизонтальное, и с меньшими возможностями настройки).

У GNOME нет единой иерархии меню: кроме главного, вызываемого объектом-меню с гномьей лапой (оно же, когда вызывается щелчком правой кнопки на фоне или нажатием клавиши, почему-то называется *глобальным* (global)), пользователь может создавать *обычные* (normal) меню, связанные с объектами-меню на панелях.

Меню настраиваются примерно так же, как и панели: пользователь может добавлять, менять и удалять пункты, создавать подменю и т. п. При этом создаваемые обычные меню изначально пусты, а главное/глобальное заполняется при установке всем, что GNOME найдёт в системе, и пользователю остаётся только убрать лишнее и переставить пункты в соответствии со своими предпочтениями.

Для настройки различных аспектов функционирования системы предназначен Центр управления, представляющий собой набор *управляющих аплетов* (каплетов, от англ. capplets, control applets), связанных с разными компонентами и прикладными программами.

Одни из них позволяют менять параметры рабочего стола и облик приложений (включая использование тем), другие — настраивать мультимедиа, третьи — управлять свойствами клавиатуры и мыши, и т. д.

Важным «каплетом» является менеджер т. н. *обработчиков документов* (Document Handlers), устанавливающий соответствие между типом файла или протокола и программой, выполняющей различные операции с ними. Набор каплетов является расширяемым, их можно разрабатывать не только для программ, написанных специально для GNOME, но и для внешних программ.

Также постоянно расширяется набор утилит, прикладных программ и аплетов, поставляемых с GNOME — вместе с программами, входящими в большинство дистрибутивов ОС, о которых GNOME «в курсе», их число превышает сотню. Перечислить их здесь нет никакой возможности, но среди них есть интерфейсы для администрирования системы, средства звукозаписи и воспроизведения, сетевые утилиты, игры и многое другое.

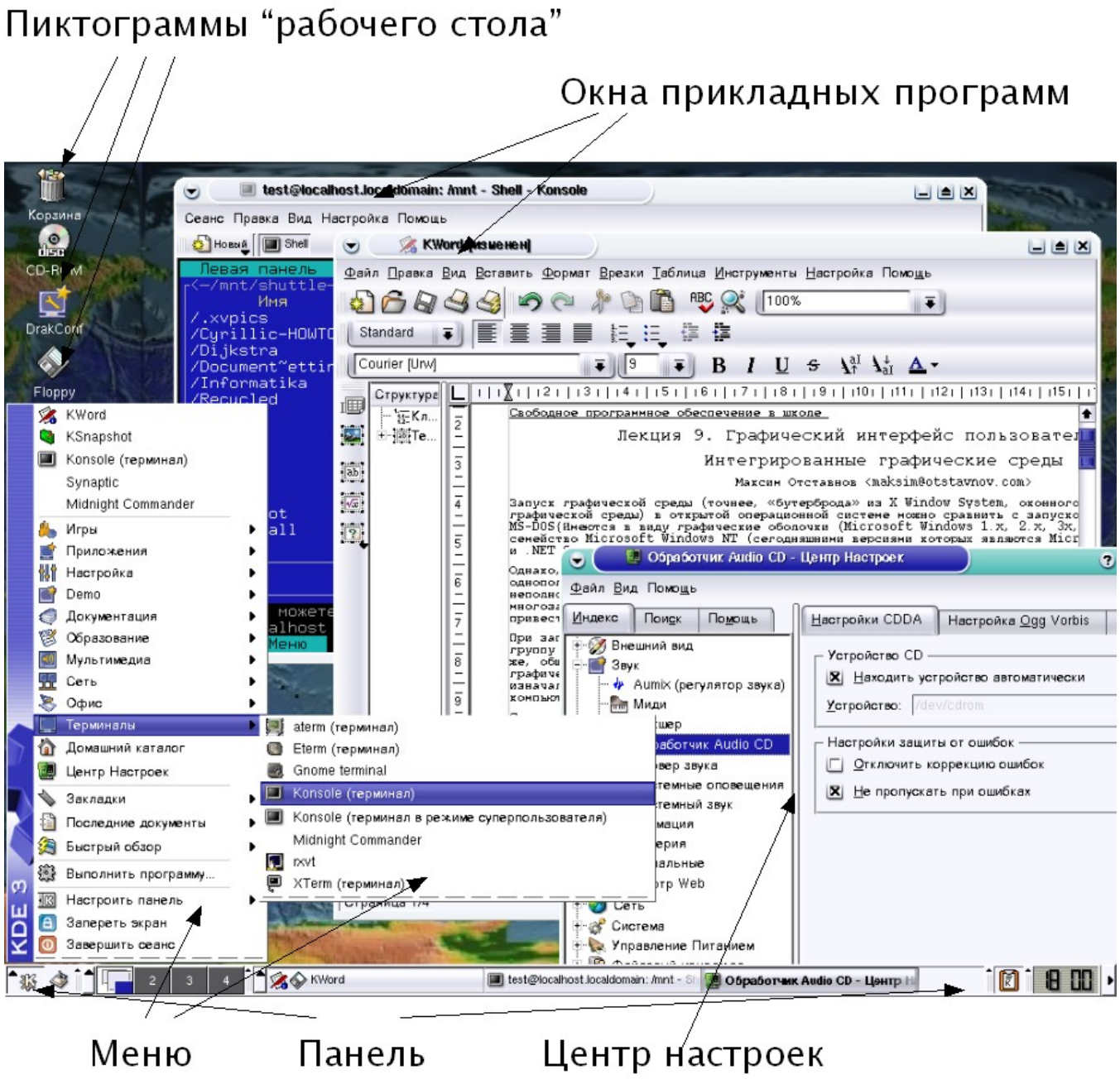
GNOME снабжён встроенной системой помощи; кроме того, его разработчиками совместно с Sun Microsystems подготовлено компактное руководство, доступное в разных форматах на сайте проекта. В его поставку входит система разработки графических приложений под GTK+, которая называется Glade и включает в себя специфические для GNOME элементы.

GNOME и большинство его компонентов соответствуют соглашениям об интернационализации и, соответственно, поддерживают работу с кириллицей, локализацию и перевод интерфейса.

KDE

Само название KDE (KDE, K Desktop Environment — «Графическая среда К») — явная пародия на CDE (Common Desktop Environment — «Общая настольная среда»). CDE была последней попыткой отрасли стандартизовать графическую среду на несвободной основе, предпринятой в конце девяностых годов. Буква «К» в KDE ничего не означает.

Рисунок 13. Интегрированная графическая среда KDE



Несмотря на явно игривый тон, начинающийся с названия среды и продолжающийся в названии

компонентов, KDE — очень серьёзный проект. В KDE любят играть со словами; например, универсальный броузер, входящий в среду, называется Konqueror (от англ. conqueror — «завоеватель», «покоритель»), терминал — Konsole (от console — «консоль»), а система помощи — вообще Kandalf (от имени Гэндальфа, мага из фантазийных произведений Дж. Р. Р. Толкиена).

Если единообразие и однородность графической среды считать достоинством, то KDE — несомненный лидер среди всех (как свободных, так и несвободных) интегрированных графических сред. Основное видимое средство интеграции — это универсальный броузер Konqueror. Функция Konqueror близка к той, которую приобрёл Windows Explorer — он совмещает функции гипермедийного броузера WWW и броузера локальных ресурсов.

Разработчики KDE пошли даже дальше своих коллег из Microsoft и определили ряд дополнительных протоколов, что позволило, в частности, просматривать с помощью броузера в единообразном формате все разнообразие справочной информации, представленное в сегодняшних открытых система (традиционные страницы руководства **man**, гипертекстовую систему **Info** из проекта GNU, разрозненные файлы документации в текстовом и гипертекстовом формате). В Konqueror интегрирована также возможность предварительного просмотра содержимого большого количества типов файлов.

KDE включает также настраиваемую систему панелей и меню и интегрированный *центр управления*, позволяющий согласованно изменять параметры среды. KDE менее гибка в настройке, чем GNOME, однако её гибкости вполне достаточно для решения любых практических задач (в том числе, имитации вида и поведения других сред). KDE работает только с собственным оконным менеджером KWin.

В поставку KDE входит множество «аксессуаров» и прикладных программ, к тому же рядом с проектом выросла целая группа сопутствующих, ориентированных на те или иные предметные приложения, из которых самым развитым является офисный пакет KOffice.

Зачем нужны «лёгкие» среды?

В то время, как сама оконная система X много лет является фактическим отраслевым стандартом, лежащие «над» нею слои графической среды не стандартизованы. Какую-либо классификацию графических сред дать затруднительно, однако самым грубым образом их можно разделить на интегрированные и лёгкие. В последние годы разработчики GNOME и KDE работают над формулировкой общего открытого стандарта на интегрированную графическую среду.

Оборотной стороной интегрированности является достаточно высокая их требовательность к ресурсам. Комфортная работа с KDE или GNOME последних версий начинается при производительности компьютера, примерно эквивалентной производительности 800 МГц процессора Celeron. Отказ от некоторых ресурсоёмких свойств (анимация изменений в среде и т. п.) позволяет снизить требования примерно до 500 МГц при объёме оперативной памяти от 128 МБ. Разумеется, эти цифры даже ниже характерных для компьютеров стартового уровня, поставляемых сегодня производителями, однако парк машин, находящихся в эксплуатации, как в офисах, так дома и в школе, включает и компьютеры с более низкими характеристиками.

Здесь помогут *лёгкие* графические среды, представляющие собой оконные менеджеры с расширенными возможностями. Описанные выше IceWM, BlackBox и FluxBox (а также чуть более требовательный к ресурсам WindowMaker)[3] позволяют достаточно комфортно работать с графикой на машинах производительностью (в эквиваленте Intel Pentium) примерно от 100 МГц и с памятью от 32 Мб.

Следует оговориться, что отказ от интегрированных графических сред не является панацеей: конкретные прикладные программы могут быть сами по себе достаточно требовательными к ресурсам. Кроме того, если прикладная программа изначально создана с ориентацией на определённую интегрированную среду, она может интенсивно использовать соответствующие библиотеки, даже если запускается в лёгкой среде. Например, запуск программ из пакета KOffice в лёгкой среде, на самом деле, даёт небольшой выигрыш по сравнению с его запуском из «родной» для него среды KDE.

Если необходимо задействовать имеющийся парк «слабой» техники для таких задач, а также, если необходимо сохранять в эксплуатации ещё менее производительные машины (например, старшие модели IBM PC-совместимых компьютеров на базе процессоров Intel 486 или AMD 586 или Макинтош

на процессорах Motorola 68K), следует подумать об использовании такой техники в режиме графических терминалов или, по крайней мере, варианте запуска наиболее требовательных к ресурсам прикладных программ на сервере.

Следует оговорить также, что ограниченность аппаратных ресурсов не является единственным мотивом применения лёгких графических сред. Каждая графическая среда, интегрированная или лёгкая, обладает собственными уникальными особенностями, собственным стилем. Уместность использования каждой конкретной среды в значительной степени зависит от набора задач, решаемых на компьютере конкретным пользователем, и от его личных предпочтений.

---

[1] Следует отметить, что большинство базовых функций оконных менеджеров при исполнении опирается на поддержку оконной системой X функций двумерной графической акселерации (ускорения), реализованных практически во всех современных графических адаптерах. В отличие от трёхмерной акселерации, полезной лишь для достаточно узкого круга приложений (программ трёхмерного моделирования, компьютерных игр), двумерная акселерация — действительно универсальна и полезна для графического пользовательского интерфейса. При использовании карты без двумерного ускорения или карты, чья акселераторная функциональность не поддерживается системой X, можно рекомендовать настроить среду таким образом, чтобы исключить ресурсоёмкие функции, например, визуализацию перемещения окна со всем его содержимым, дабы избежать неоправданного роста нагрузки на процессор и драматического падения производительности.

[2] Разумеется, это сильно идеализированная картина. Иногда прикладная логика диктует некоторые элементы эргономики. Например, интерфейсы большинства систем автоматизированного конструирования и проектирования (CAD, САПР) весьма сходны, вне зависимости от среды, в которой работают эти программы.

[3] Возможностью представить их обзор в компактном виде автор обязан прежде всего своим соавторам Егору Гребневу, Сергею Иванову, Михаилу Шигорину. — Прим. М. Отставнова.

Права доступа в системе Linux

Вадим Виниченко, Мэтт Уэлш

Пользователи и группы

Поскольку система Linux с самого начала разрабатывалась как многопользовательская, в ней предусмотрен такой механизм, как права доступа к файлам и каталогам. Он позволяет разграничить полномочия пользователей, работающих в системе. В частности, права доступа позволяют отдельным пользователям иметь “личные” файлы и каталоги. Например, если пользователь[1] ivanov создал в своём домашнем каталоге файлы, то он является владельцем этих файлов и может определить права доступа к ним для себя и остальных пользователей. Он может, например, полностью закрыть доступ к своим файлам для остальных пользователей, или разрешить им читать свои файлы, запретив изменять и исполнять их.

Правильная настройка прав доступа позволяет повысить надёжность системы, защитив от изменения или удаления важные системные файлы. Наконец, поскольку внешние устройства с точки зрения Linux также являются объектами файловой системы, механизм прав доступа можно применять и для управления доступом к устройствам.

“Пользователями” системы Linux, выполняющими различные действия с файлами и каталогами, являются на самом деле вовсе не люди, а программы, выполняемые в системе — *процессы*. Одна из таких программ — командная оболочка, которая считывает команды пользователя из командной строки и передаёт их системе на выполнение. Каждая программа (процесс) выполняется от имени определённого пользователя. Её возможности работы с файлами и каталогами определяются правами доступа, заданными для этого пользователя.

С целью оптимальной настройки прав доступа для ряда программ-серверов в системе созданы системные пользователи (учётные записи), от имени которых работают эти программы. Например, в системе ALT Linux веб-сервер (Apache) выполняется от имени пользователя apache, а ftp-сервер — от имени пользователя ftp. Такие учётные записи не предназначены для работы людей-пользователей.

У любого файла в системе есть *владелец* — один из пользователей. Однако каждый файл одновременно принадлежит и некоторой *группе* пользователей системы. Каждый пользователь может входить в любое количество групп, и в каждую группу может входить любое количество пользователей из числа определённых в системе.

Когда в системе создаётся новый пользователь, он добавляется по крайней мере в одну группу. В системе Linux при создании новой учётной записи создаётся специальная группа, имя которой совпадает с именем нового пользователя, и пользователь включается в эту группу. В дальнейшем администратор может добавить пользователя к другим группам.

Механизм групп может применяться для организации совместного доступа нескольких пользователей к определённым ресурсам. Например, на сервере организации для каждого проекта может быть создана отдельная группа, в которую войдут учётные записи (имена пользователей) сотрудников, работающих над этим проектом. При этом файлы, относящиеся к проекту, могут принадлежать этой группе и быть доступными для её членов. В системе также определено несколько групп (например, bin), которые используются для управления доступом системных программ к различным ресурсам. Как правило, членами этих групп являются системные пользователи, пользователи-люди не включаются в такие группы.

В некоторых дистрибутивах Linux (в т. ч. в дистрибутивах ALT Linux) с помощью групп могут быть предоставлены права, необходимые для выполнения определённых пользовательских задач. Например, чтобы пользователь получил возможность собирать пакеты RPM, его следует включить в группу rpm, чтобы предоставить возможность записи дисков CD-R/RW, пользователя нужно включить в группу cdwriter и т. д.

Виды прав доступа

Права доступа определяются по отношению к трём типам действий: чтение, запись и исполнение. Эти права доступа могут быть предоставлены трём классам пользователей: владельцу файла (пользователю), группе, которой принадлежит файл, а также всем остальным пользователям, не входящим в эту группу. Право на чтение даёт пользователю возможность читать содержимое файла или, если такой доступ разрешён к каталогам, просматривать содержимое каталога (используя команду **ls**). Право на запись даёт пользователю возможность записывать или изменять файл, а право на запись для каталога — возможность создавать новые файлы или удалять файлы из этого каталога. Наконец, право на исполнение позволяет пользователю запускать файл как программу или сценарий командной оболочки (разумеется, это действие имеет смысл лишь в том случае, если файл является программой или сценарием). Для каталогов право на исполнение имеет особый смысл — оно позволяет сделать данный каталог *текущим*, т. е. “перейти” в него, например, командой **cd**.

Чтобы получить информацию о правах доступа, используйте команду **ls** с ключом -l. При этом будет выведена подробная информация о файлах и каталогах, в которой будут, среди прочего, отражены права доступа. Рассмотрим следующий пример:

```
/home/ivanov/docs# ls -l report1303
-rw-r--r-- 1 ivanov users 505 Mar 13 19:05 report1303
```

Первое поле в этой строке (-rw-r--r--) отражает права доступа к файлу. Третье поле указывает на владельца файла (ivanov), четвёртое поле указывает на группу, которая владеет этим файлом (users). Последнее поле — это имя файла (report1303). Другие поля описаны в документации к команде **ls**.

Данный файл является собственностью пользователя ivanov и группы users. Последовательность -rw-r--r-- показывает права доступа для пользователя — владельца файла, пользователей — членов группы-владельца, а также для всех остальных пользователей.

Первый символ из этого ряда (-) обозначает тип файла. Символ - означает, что это — обычный файл, который не является каталогом (в этом случае первым символом было бы d) или псевдофайлом устройства (было бы c или b). Следующие три символа (rw-) представляют собой права доступа, предоставленные владельцу ivanov. Символ r — сокращение от read (англ. читать), а w — сокращение от write (англ. писать). Таким образом, ivanov имеет право на чтение и запись (изменение) файла report1303.

После символа w мог бы стоять символ x, означающий наличие прав на исполнение (англ. execute, исполнять) файла. Однако символ -, стоящий здесь вместо x, указывает, что ivanov не имеет права на исполнение этого файла. Это разумно, так как файл report1303 не является программой. В то же время, пользователь, зарегистрировавшийся в системе как ivanov, при желании может предоставить себе право на исполнение данного файла, поскольку является его владельцем. Для изменения прав доступа к файлу или каталогу используется команда **chmod**.

Следующие три символа (r--) отражают права доступа группы к файлу. Группой-собственником файла в нашем примере является группа users. Поскольку здесь присутствует только символ r, все пользователи из группы users могут читать этот файл, но не могут изменять или исполнять его.

Наконец, последние три символа (это опять r--) показывают права доступа к этому файлу всех других пользователей, помимо собственника файла и пользователей из группы users. Так как здесь указан только символ r, эти пользователи тоже могут читать файл

Вот ещё несколько примеров:

-rwxr-x--x

Пользователь-владелец файла может читать файл, изменять и исполнять его; пользователи, члены группы-владельца могут читать и исполнять файл, но не изменять его; все остальные пользователи могут лишь запускать файл на выполнение.

-rw-------

Только владелец файла может читать и изменять его.

-rwxrwxrwx

Все пользователи могут читать файл, изменять его и запускать на выполнение.

Никто, включая самого владельца файла, не имеет прав на его чтение, запись или выполнение[2]. Хотя такая ситуация вряд ли имеет практический смысл, с точки зрения системы она является вполне корректной. Разумеется, владелец файла может в любой момент изменить права доступа к нему.

Возможность доступа к файлу зависит также от прав доступа к каталогу, в котором находится файл. Например, даже если права доступа к файлу установлены как -rwxrwxrwx, другие пользователи не могут получить доступ к файлу, пока они не имеют прав на *исполнение* для каталога, в котором находится файл. Другими словами, чтобы воспользоваться имеющимися у вас правами доступа к файлу, вы должны иметь право на исполнение для всех каталогов вдоль пути к файлу.

Права доступа и администрирование системы

Установка и поддержание оптимальных прав доступа является одной из важнейших задач системного администратора. Права должны быть достаточными для нормальной работы пользователей и программ, но не большими, чем необходимо для такой работы. Дистрибутивы ALT Linux обладают продуманной системой прав (предопределённые группы, псевдопользователи для различных программ-серверов, права доступа для системных файлов и каталогов). Прежде чем вносить существенные изменения в эту систему, целесообразно понять её логику и выяснить, нет ли другого способа достичь нужной цели.

Поскольку программы, исполняемые от имени суперпользователя (root), могут совершать любые действия с любыми файлами и каталогами, их выполнение может нанести системе серьёзный ущерб. Это может быть как следствием уязвимостей или ошибок в программах, так и результатом ошибочных действий самого пользователя. Поэтому работа с правами суперпользователя требует особой осторожности. Чтобы уменьшить связанные с этим риски, разработчики дистрибутивов ALT Linux рекомендуют для выполнения задач, требующих таких прав, использовать утилиту **sudo**.

Основные команды

Ниже перечислены важнейшие команды для решения задач, связанных с правами доступа. Для получения более подробной информации об этих командах обращайтесь к руководствам по ним.

chmod

Изменение прав доступа к файлу или каталогу.

chown

Изменение владельца файла.

chgroup

Изменение группы, которой принадлежит файл.

umask

Определение прав доступа по умолчанию для файлов, создаваемых пользователем.

---

[1] В этом разделе, если специально не оговорено иное, под пользователем понимается пользователь с точки зрения системы, т. е. зарегистрировавшийся в определённой учётной записи (работающий под определённым именем пользователя). Права доступа определяются именно для пользователей в указанном смысле. При этом один человек, работающий в системе, может регистрироваться под различными именами пользователя для выполнения различных действий. Наоборот, несколько человек, использующих одну и ту же учётную запись, для системы являются одним и тем же пользователем.

[2] Строго говоря, за исключением суперпользователя (root), который может выполнять любые операции над любыми файлами в системе.

Учётные записи

Linux — система многопользовательская, а потому пользователь — ключевое понятие для организации всей системы доступа в Linux. Когда пользователь регистрируется в системе (проходит процедуру авторизации, например, вводя системное имя и пароль), он идентифицируется с **учётной записью**, в которой система хранит информацию о каждом пользователе: его системное имя и некоторые другие сведения, необходимые для работы с ним. Именно с учётными записями, а не с самими пользователями, и работает система. Ниже приведён список этих сведений.

Системное имя (user name)

Это то имя, которое вводит пользователь в ответ на приглашение login:. Оно может содержать только латинские буквы и знак «_». Это имя используется также в качестве имени учётной записи.

Идентификатор пользователя (UID)

Linux связывает **системное имя** с **идентификатором пользователя** в системе — **UID** (User ID). **UID** — это положительное целое число, по которому система и отслеживает пользователей[1]. Обычно это число выбирается автоматически при регистрации учётной записи, однако оно не может быть совершенно произвольным. В Linux есть некоторые соглашения относительно того, каким типам пользователей могут быть выданы идентификаторы из того или иного диапазона. В частности, **UID** от «0» до «100» зарезервированы для **псевдопользователей**[2].

Идентификатор группы (GID)

Кроме идентификационного номера пользователя с учётной записью связан **идентификатор группы**. **Группы пользователей** применяются для организации доступа нескольких пользователей к некоторым ресурсам. У группы, так же, как и у пользователя, есть имя и идентификационный номер — **GID** (Group ID). В Linux каждый пользователь должен принадлежать как минимум к одной группе — **группе по умолчанию**. При создании учётной записи пользователя обычно создаётся и группа, имя которой совпадает с **системным именем**[3], именно эта группа будет использоваться как группа по умолчанию для этого пользователя. Пользователь может входить более чем в одну группу, но в учётной записи указывается только номер группы по умолчанию. Группы позволяют регулировать доступ нескольких пользователей к различным ресурсам.

Полное имя (full name)

Помимо **системного имени** в учётной записи содержится и **полное имя** (имя и фамилия) использующего данную учётную запись человека. Конечно, пользователь может указать что угодно в качестве своего имени и фамилии. Полное имя необходимо не столько системе, сколько людям — чтобы иметь возможность определить, кому принадлежит учётная запись.

Домашний каталог (home directory)

Файлы всех пользователей в Linux хранятся раздельно, у каждого пользователя есть собственный **домашний каталог**, в котором он может хранить свои данные. Доступ других пользователей к домашнему каталогу пользователя может быть ограничен. Информация о домашнем каталоге обязательно должна присутствовать в учётной записи, потому что именно с него начинает работу пользователь, зарегистрировавшийся в системе.

Начальная оболочка (login shell)

Важнейший способ взаимодействовать с системой Linux — **командная строка**, которая позволяет пользователю вести «диалог» с системой: передавать ей команды и получать её ответы. Для этой цели служит специальная программа — **командная оболочка** (или **интерпретатор командной строки**), по-английски — shell. Начальная оболочка (login shell) запускается при входе пользователя в систему в текстовом режиме (например, на виртуальной консоли). Поскольку в Linux доступно несколько разных командных оболочек, в учётной записи указано, какую из командных оболочек нужно запустить для данного пользователя. Если специально не указывать начальную оболочку при создании учётной записи, она будет назначена по умолчанию, вероятнее всего это будет bash.

Все перечисленные данные об учётных записях хранятся в файле /etc/passwd. Сведения о конкретной учётной записи пользователя можно получить с помощью утилиты getent[4]:

```
[tester@tacit tester]$ getent passwd tester
tester:x:506:506:Test Testovitch:/home/tester:/bin/bash
[tester@tacit tester]$
```

Пример 1. Сведения об учётной записи

Первый параметр, passwd — это название базы, в которой нужно производить поиск, оно совпадает с именем соответствующего конфигурационного файла. Второй параметр, tester — это название учётной записи пользователя (системное имя). getent выводит ту строчку /etc/passwd, где описана искомая учётная запись: в ней через «:» указаны системное имя, пароль (тут стоит буква «x», потому что пароль спрятан в другом месте, об этом ниже), UID, GID, полное имя, домашний каталог и начальная оболочка.

В Linux пароль пользователя в явном виде не хранится нигде, но только в зашифрованном. В современных системах обычно применяются так называемые «теневые пароли» (shadow passwords), которые хранятся отдельно от остальных сведений об учётной записи, а также позволяют назначать дополнительные ограничения, в частности, «срок годности» пароля. В зависимости от строгости политики безопасности зашифрованные пароли пользователей могут храниться в общем файле /etc/shadow (менее строго) или в отдельном файле shadow для каждого пользователя. В ALT Linux по умолчанию используется схема tcb, реализующая более строгую политику. Для просмотра сведений из файла shadow требуются полномочия суперпользователя, это можно сделать с помощью команды getent passwd tester. Подробнее о возможностях теневых паролей можно прочитать в руководствах shadow(5) и tcb(5).

Также отдельно хранится информация обо всех группах пользователей в системе, для этого предназначен файл /etc/group. Информацию о конкретной группе можно получить с помощью той же утилиты getent:

```
[tester@tacit tester]$ getent group audio
audio:x:81:yura,fedya
[tester@tacit tester]$
```

Пример 2. Сведения о группе

Запись в файле /etc/group устроена очень просто: сначала идёт имя группы (как и имя учётной записи, потом поле для пароля (здесь опять «x», но пароли для группы используются очень редко), GID, список через запятую названий учётных записей (имён пользователей), входящих в данную группу. Любой пользователь может получить список названий групп, в которых он состоит командой groups, а более подробные сведения о своей или чужой учётной записи командой id *имя_пользователя*.
Принадлежность к группе существенна только в одном отношении — прав доступа, поскольку для каждого файла определён не только пользователь-владелец, но и группа-владелец.

Управление пользователями

Создание пользователей

Для создания полноценного пользователя Linux нужно выполнить несколько относительно независимых действий:

- создать запись в /etc/passwd, где присвоить учётной записи уникальное имя, UID и пр.;
- создать домашний каталог пользователя, обеспечить пользователю доступ к его домашнему каталогу (сделать его владельцем каталога);
- поместить в домашний каталог стандартное наполнение (обычно конфигурационные файлы), взятое из /etc/skel;
- модифицировать системные конфигурационные файлы, в частности, создать хранилище для приходящей почты для данного пользователя (/var/spool/mail/tester).

Все эти действия могут быть выполнены и вручную, однако это довольно неудобно и можно что-нибудь забыть. Для упрощения процесса используется утилита useradd (она же по традиции называется adduser), для выполнения которой, естественно, потребуются полномочия администратора. В простейшем случае достаточно будет двух шагов:

```
root@tacit ~# useradd test
root@tacit ~# passwd test
passwd: updating all authentication tokens for user test.

You can now choose the new password or passphrase.

. . .

Enter new password:
```

Пример 3. Создание пользователя

Сначала useradd добавляет учётную запись (имя пользователя — единственный параметр, в нашем примере — test), заполняя её значениями по умолчанию и проводя все необходимые изменения в системе. С помощью дополнительных параметров при вызове useradd можно явно указать значение для того или иного поля учётной записи, также эта утилита позволяет модифицировать параметры создания пользователей по умолчанию. Подробности можно найти в руководстве useradd(8). Утилита passwd, вызванная с правами суперпользователя, позволяет назначить данному пользователю любой пароль. При этом сведения о предшествующем пароле данного пользователя (если таковой был), будут полностью утрачены. passwd, вызванная обычным пользователем (без параметров), позволяет ему сменить свой собственный пароль, но для этого потребуется ввести текущий пароль пользователя.

Существуют аналогичные useradd утилиты для модификации параметров уже существующей учётной записи (usermod) и для удаления пользователей (userdel). Пользователь также может изменить некоторые некритичные сведения в своей учётной записи самостоятельно. В частности, для установки своего **полного имени** и некоторых других информационных полей учётной записи служит утилита chfn(1) из пакета shadow-change, сменить **начальную оболочку** поможет утилита chsh(1) (позволяет выбрать только одну из оболочек, перечисленных в /etc/shells) из того же пакета. Обратите внимание, что в ALT Linux пользователь имеет право редактировать собственную учётную запись только в том случае, если установлен соответствующий режим доступа: команда control chsh должна возвращать public, аналогично control chfn. Установить нужный доступ может суперпользователь командой control chsh public (аналогично для chfn).

Самую востребованную операцию по работе с группами — добавление пользователя в группу — проще всего выполнить простым редактированием файла /etc/group. Достаточно открыть этот файл в любом текстовом редакторе (естественно, с правами суперпользователя), найти строчку, начинающуюся с названия нужной группы, и добавить в конец этой строки имя нужного пользователя через запятую). Для управления группами существует комплект утилит groupadd(8), groupdel(8), groupmod(8), подробности о работе с ними можно найти в соответствующих руководствах.

1Это может оказаться важным, например, в такой ситуации: учётную запись пользователя с именем test удалили из системы, а потом добавили снова. Однако с точки зрения системы это уже другой пользователь, потому что у него другой UID.

2Обычно Linux выдаёт нормальным пользователям UID, начиная с «500» или «1000».

3Как правило, численное значение GID в этом случае совпадает со значением UID.

4Эта утилита также полезна для получения сведений о некоторых других системных ресурсах, см. getent –help.

Что происходит в системе?

Георгий Курячий, Дмитрий Левин

Человеку, отвечающему за работоспособность системы, очень важно всегда отчётливо представлять, что с нею творится. Теоретически, никакое происшествие не должно ускользнуть от его внимания. Однако компьютерные системы столь сложны, что отслеживать *все* события в них — выше человеческих возможностей. Для того, чтобы довести поток служебной информации до разумного объёма, её надо *просеять* (выкинуть незначащие данные), *классифицировать* (разделить на несколько групп сообразно тематике) и *журнализировать* (сохранить в доступном виде для дальнейшего анализа).

В Linux эта задача решается с помощью механизма *централизованной журнализации*, который реализован демоном (системной службой) syslogd. Все части системы (включая ядро и системные службы) рапортуют syslogd о происходящих в них событиях. В этот рапорт включается имя службы, *категория* (facility) и *важность* (priority) произошедшего события. Демон, сообразно настройкам, классифицирует все эти рапорты в несколько выходных потоков. Классификация и отсев данных всякого выходного потока происходит так: для каждой категории событий определяется *наименьшая* важность, которой событие должно обладать, чтобы попасть в этот выходной поток. Например, легко определить поток «ошибки», в который будут попадать только *важные* рапорты *любых* категорий, или поток «безопасность», в который будут попадать *все* рапорты категории «безопасность» и те рапорты других категорий, важность которых заставляет подозревать угрозу безопасности системы (например, рапорт категории «демон» об аварийном завершении работы системной службы).

Главное место хранения уже классифицированного syslogd потока событий — *системный журнал* (т. н. log-файл). Системный журнал — *текстовый* файл, содержащий рапорты *одного* потока. Обычно syslogd хранит системные журналы в каталоге /var/log и его подкаталогах. Именно в системные журналы, прежде всего в /var/log/messages, /var/log/maillog и /var/log/dmesg, необходимо заглядывать администратору, который хочет знать, что происходит в системе. Поток рапортов о *важных* событиях syslogd направляет и на *системную консоль* — выделенное терминальное устройство. В ALT Linux роль системной консоли выполняет 12-я *виртуальная консоль*, доступная по сочетанию клавиш **Alt-F12** или **Alt-Ctrl-F12**. Стоит заметить, что некоторые службы (например, WWW-сервер apache) *самостоятельно*, в обход syslogd, ведут журнализацию своих событий, поэтому информацию о количестве и местоположении их журналов можно почерпнуть из их файлов настроек (обычно, тем не менее, журналы хранятся в /var/log).

Новые рапорты, поступающие в системный журнал, наиболее актуальны, а предыдущие, по мере их устаревания, эту актуальность утрачивают. Если самые старые данные в журнале не удалять, файловая система, рано или поздно, окажется переполненной. В Linux организован механизм *устаревания журналов*, которым занимается служба logrotate. Запускаясь раз в день, logrotate проверяет, какие из файлов следует признать устаревшими. Файл объявляется устаревшим раз в определённый промежуток времени (например, раз в неделю), или если он достиг определённого размера.

Процедура устаревания такова. Для каждого журнала, как, например, для /var/log/syslog/alert, logrotate держит в том же каталоге *очередь устаревших копий* — файлы с именами от alert.0.bz2 (предыдущая копия) до alert.4.bz2 (самая старая копия). Очередь alert в нашем примере состоит из пяти упакованных с помощью bzip2 файлов. В момент устаревания alert.3.bz2 переименовывается в alert.4.bz2 (старые данные теряются), копия с номером 2 превращается в третью, первая — во вторую, нулевая в первую. Наконец, сам журнал упаковывается и переименовывается в alert.0.bz2, а на его месте заводится новый — пустой. Таким образом, администратор всегда имеет доступ к *свежему* журналу и к нескольким его копиям за определённое время.

Некоторые файлы в /var/log — нетекстовые, они не являются полноценными журналами, а представляются собой «свалку событий» для служб авторизации и учёта. Текстовую информацию о входе пользователей в систему и выходе оттуда можно получить по команде **last**, а узнать о тех, кто в данный момент пользуется системой, помогут команды **w** и **who**.

Множество важной информации может дать анализ *загруженности* системы — в плане процессорного времени и потребления оперативной памяти (**ps**, **top**, **vmstat**), в плане использования дискового пространства (**du**, **df**, **lsof**) и в плане работы сетевых устройств (**netstat**).

Настройка загрузки

Олег Власенко

Загрузка системы

Linux, установленный на жёстком диске, загружается при включении компьютера при помощи специальной программы — *загрузчика*. Программа-загрузчик исполняется при загрузке системы с жёсткого диска и загружает ядро ОС Linux, расположенное также на жёстком диске.

Загрузчики Linux можно также использовать для загрузки нескольких операционных систем, поскольку они позволяют выбирать при включении компьютера, какую систему нужно загрузить в этот раз. Если есть выбор из нескольких вариантов загрузки, то после некоторого времени ожидания будет загружена та система, которая выбрана по умолчанию: это не обязательно должен быть Linux, а может быть другая операционная система или специальный режим загрузки (например, восстановительный).

Например, при стандартной установке в начальном меню загрузчика ALT Linux доступны три альтернативы: ALT Linux, ALT Linux — Безопасные настройки (загрузка с минимальным количеством драйверов, что может оказаться необходимым в случае неполадок), Спасательная система. Если у вас есть установочный CD ALT Linux, вы также можете загрузиться с него: помимо установки новой системы можно загрузить уже установленный на жёстком диске Linux, который по тем или иным причинам невозможно загрузить прямо с жёсткого диска.

В нижней части экрана начального меню загрузчика располагается строка «Параметры». В этой строке можно указать параметры, которые будут переданы ядру Linux при загрузке.

Загрузка нескольких операционных систем

Прежде всего следует отметить, что ОС Linux может быть загружена с любого жёсткого диска системы и любого типа раздела — и *основного* (primary), и *дополнительного* (secondary), с различных типов файловых систем (например Ext2, Ext3, ReiserFS). При этом раздел, содержащий корневую файловую систему, не обязательно должен быть активным (иметь статус A в таблице разделов). Более того, вы можете использовать любой загрузчик, при условии, что он в состоянии передать управление на загрузочный сектор любого раздела (при этом несущественно, с какой операционной системой поставляется данный загрузчик). При наличии какого-либо стороннего загрузчика, загрузчик Linux следует устанавливать не в MBR первого жёсткого диска системы, а в загрузочный сектор корневого раздела Linux, на который впоследствии необходимо передать управление со стороны стороннего загрузчика. Подавляющее большинство UNIX-подобных систем не чувствительны к месту их размещения — главное, чтобы был способ передать управление на их программу начальной загрузки[1].

При использовании поставляемого с дистрибутивом загрузчика LILO передача управления на загрузочный сектор любого раздела, физически доступного в момент загрузки, не вызывает проблем. В то же время специфика архитектуры некоторых нестандартных операционных систем накладывает ряд ограничений на размещение этих систем на диске. Возможна, что такая система может загружаться только с активного *основного* (primary) раздела на первом жёстком диске системы, в противном случае возможны самые неожиданные проблемы с загрузкой. В такой ситуации лучше полностью сохранить статус загрузочного раздела этой операционной системы.

Настройка загрузчика

Первое решение, которое нужно принять — где расположить загрузчик. Программа установки предлагает на выбор несколько позиций, где может быть размещён загрузчик. Общее правило: если устройство указано как «полный» жёсткий диск (без указания номера раздела — например, /dev/hda), то загрузчик будет поставлен в MBR указанного диска; если устройство указано как раздел диска (в конце номер раздела), то загрузчик будет установлен в загрузочный сектор соответствующего раздела. Можно

переместить загрузчик и после установки, исправив соответствующим образом конфигурационные файлы и дав команду **lilo** (см. ниже).

Если для загрузки всех операционных систем предполагается использовать загрузчик Linux (LILO), то в качестве загрузочного устройства необходимо выбрать первый диск системы; обычно это /dev/hda или /dev/sda. При таком выборе загрузчик первым получит управление от BIOS. Чтобы загрузчик Linux мог загружать другие операционные системы, ему нужно сообщить об их существовании. Программа установки ALT Linux умеет делать это автоматически. Однако если вам нужна более тонкая настройка, или что-то изменилось уже после установки ALT Linux, то можно отредактировать конфигурационные файлы загрузчика самостоятельно.

Это делается следующим образом: в файле /etc/lilo.conf для каждой операционной системы, которую потребуется загружать, нужно добавить новый раздел по аналогии со следующей записью:

```
other=/dev/hda1
label=other
table=/dev/hda
```

Данная запись сообщает LILO о том, что на раздел /dev/hda1 установлена неизвестная ОС; в меню её надо отобразить под именем «other»; если пользователь выберет этот пункт меню — передать управление на загрузочный сектор /dev/hda1.

После сохранения данного файла конфигурации необходимо дать команду **lilo**, чтобы изменения вступили в силу.

Наоборот, если общим для всех ОС будет загрузчик другой операционной системы, то LILO необходимо установить на *корневой раздел* Linux (точка монтирования — /). После этого необходимо сообщить общему загрузчику всех ОС о том, как передавать управление на раздел Linux. Как это сделать — смотрите в документации к используемому вами программному обеспечению.

Восстановление загрузчика

Загрузка Linux может быть нарушена, если загрузчик Linux окажется по каким-то причинам повреждён или заменён другой программой. Последнее может произойти, например, в процессе установки другой ОС, если загрузчик был установлен в загрузочный сектор диска (MBR), содержание которого будет перезаписано и заменено загрузчиком другой ОС. В этой и подобных ситуациях необходимо восстановить загрузчик Linux, и возможно, изменить его размещение на диске.

Восстановить загрузчик в той же конфигурации, в которой он был до повреждения, несложно, для этого достаточно:

- любым способом загрузиться в Linux;
- смонтировать тот раздел жёсткого диска, на котором находится корневая файловая система Linux (выполнить **mount** раздел **/mnt**, где раздел — это имя соответствующего файла устройства, например, /dev/hda1);
- объявить раздел со смонтированной корневой файловой системой корневым (выполнить **chroot /mnt**);
- выполнить команду **lilo**.

В случае, если потребуется изменить конфигурацию загрузчика, например, переместить его на другой диск или раздел, перед выполнением **lilo** нужно будет соответствующим образом исправить конфигурационный файл /etc/lilo.conf.

---

[1] Для очень старых BIOS действует правило 1024-го циллиндра: загрузка невозможна, если раздел, с которого загружается система, будет расположен на диске далее 1024-го циллиндра. В случае Linux — это раздел, содержащий корневую файловую систему или /boot, если он выделен на отдельный раздел.

Прикладные программы для Linux

Основная особенность программного обеспечения Linux — многообразие продуктов, решающих сходные задачи, особенно если дело касается области, в которой существует *несколько* подходов к их решению. Открытая модель разработки программ, позволяет любому выбрать самый подходящий для него инструмент и развивать именно его. Поэтому список проектов, так или иначе связанных с Linux, насчитывает десятки (или даже сотни) тысяч наименований.

Конечно же, работа с самой операционной системой не может быть самоцелью. Все усилия по изучению операционной системы Linux и основных утилит нужны для того, чтобы впоследствии наилучшим образом решать в этой операционной системе любые из своих прикладных задач, разрешимых при помощи компьютера. Для очень многих задач достаточно стандартных инструментов Linux и текстового редактора, однако есть случаи, в которых всё-таки необходима специальная прикладная программа, именно для этого предназначенная, или в которых специальная программа удобнее комбинации стандартных утилит.

Этот раздел посвящён краткому обзору **прикладных программ** для Linux, специально предназначенных для решения самых разных пользовательских задач. Вошедший сюда материал нужно воспринимать только как пример, демонстрацию того, что и как можно делать в Linux, но вовсе не исчерпывающий список. В отличие от основных принципов устройства системы или стандартных утилит, которые не изменяются (почти) в течение десятилетий, прикладное программное обеспечение — это область, где всё меняется очень быстро. Технологии, сегодня считающиеся самыми передовыми, уже через несколько месяцев могут устареть. Вместе с ними могут устареть использующие их программы, а другие программы могут, наоборот, перейти в разряд наиболее современных и развитых. Поэтому перечисленные здесь прикладные программы — это не безусловная рекомендация, а довольно случайная выборка, отражающая текущее состояние дел в разработке приложений для Linux. Самый лучший способ найти и выбрать самые подходящие прикладные программы для своих задач — посоветоваться с людьми, которые решают подобные задачи в Linux в настоящее время — и попробовать.

Нужно отдавать себе отчёт в том, что прикладные программы для Linux не являются частью самой Linux, поэтому любой из названных ниже программ может не оказаться в каком-то из конкретных **дистрибутивов** Linux. Но почти наверняка в любом дистрибутиве найдётся не меньше одной или нескольких программ для решения каждой из перечисленных ниже прикладных задач. Чтобы не загромождать изложение, мы остановимся лишь на самых распространённых программных продуктах, входящих во многие дистрибутивы Linux.

Рабочий стол

Первое, что стоит сделать, начав постоянно использовать Linux — организовать для себя удобное «рабочее место»: подобрать и настроить программы, с которыми приходится работать каждый день. Рабочее место в Linux может выглядеть очень по-разному. Можно вовсе обойтись без графического интерфейса, используя только текстовый терминал для управления системой. Такой выбор будет правильным, если рабочее место находится на сервере, подключённом к сети Internet, доступ к которому осуществляется только при помощи ssh или аналогичных клиентов удалённого доступа. Впрочем, некоторые пользователи предпочитают работать в текстовом интерфейсе, возможно, по эргономическим причинам — ничто не отвлекает?

Если графический интерфейс используется, то и в этом случае есть огромный выбор, как его организовать. Прежде всего, решить: нужно ли устраивать «**рабочий стол**» (для этого подходят GNOME, KDE, XFCE) или можно обойтись возможностями одного из развитых диспетчеров окон (Enlightenment, FVWM2, WindowMaker и многие другие). Помимо функциональности, в выборе графической среды решающее значение могут сыграть и эстетические критерии. Дальше всех в этом направлении продвинулась среда Enlightenment, работа с которой в некоторых вариантах настройки количеством украшений и эффектов напоминает участие в компьютерной игре (скорее всего, сетевой).

Диспетчеры файлов

Многие пользователи привыкли оперировать с файлами и каталогами как с наглядными штучными объектами (папками и документами), они могут выбрать для себя программу, которая позволяет наглядно и поштучно работать с объектами файловой системы — **диспетчер файлов** (file manager). Поскольку представление файлов и каталогов как папок и документов нужно в первую очередь в рамках метафоры рабочего стола, то и диспетчеры файлов для Linux разрабатываются прежде всего как приложения той или иной среды рабочего стола. В частности, и в KDE, и в GNOME есть свои диспетчеры файлов — konqueror и nautilus соответственно, которые по совместительству служат www-браузерами. Такое совмещение функций вполне логично, поскольку в среде рабочего стола нужно представлять доступные локальные и удалённые ресурсы как единое пространство, наполненное объектами, которыми можно манипулировать, можно «открывать», т. е. запускать соответствующее приложение для просмотра и/или редактирования.

Для многих пользователей наиболее удобный способ работы с файловой системой — «классический» двухпанельный диспетчер файлов, работающий в текстовом режиме (в терминале) — Midnight Commander (название утилиты — mc)[1]. Его функциональность также шире просто операций с файлами — он позволяет открывать файлы для просмотра и редактирования, вызывать вспомогательные программы для работы с архивами (и даже «заходить» в архивы, как в каталоги), передавать данные по сети и т. п. Midnight Commander имеет также неплохой встроенный текстовый редактор, опять-таки «классического» стиля.

Далеко не всё, что нужно делать в Linux, в среде mc так же удобно, как и в полноценной командной строке. Кроме того, при работе с *графическими* файлами сильно не хватает представления этих файлов в виде миниатюр (thumbnails), чтобы выбирать среди них по содержимому, а не только по имени. Такими возможностями обладают многочисленные графические диспетчеры файлов; помимо тех, что включены в среды KDE и GNOMe, есть множество независимых: dfm (похожий на диспетчер файлов OS/2), emelFM2, EZFM и X Northern Captain (двухпанельные, причём автор последнего — наш человек из Дубны), gentoo и worker (двухпанельные, в стиле диспетчера файлов DirectoryOpus из AmigaOS), FSV и XCruiser (трёхмерные! причём последний похож скорее на космический симулятор). Среди них встречаются и ориентированные специально на просмотр изображений, такие как GQView, endeavour, gview, qiv, xzgv и некоторые другие, — с возможностями слайд-шоу, автоматического изменения размера, показа картинки на полный экран и т. п.

Эмулятор терминала

Даже для такой на первый взгляд тривиальной функции, как эмуляция терминала для X Window System, существует целый круг программ. Самая стандартная из них поставляется вместе с XOrg — xterm. Вариант xterm, поддерживающий отображение шрифтов в кодировке UNICODE, вызывается командой uxterm. Однако каждое приложение, организующее среду рабочего стола, включает собственный **эмулятор терминала**, внешний вид и поведение которого настраивается централизованно вместе со всеми остальными приложениями рабочего стола. Есть и другие эмуляторы терминала, не связанные с конкретным рабочим столом, к таким относится 9term, повторяющий возможности «окна» системы Plan9, mlterm, имеющий многоязыковую поддержку, rxvt — очень нетребовательный к ресурсам эмулятор терминала, или его потомки, наподобие aterm.

Большое количество терминальных окон на рабочем столе может образоваться, даже если раскладывать их по разным виртуальным экранам. Некоторые версии xterm (например, konsole), позволяют открывать окна «стопками», переключаясь между ними с помощью «закладок», как в записной книжке.

WWW-браузеры

WWW-браузер — программа для просмотра гипертекста, доступного через Internet — на сегодня чуть ли не самое важное приложение для персонального компьютера. Сегодняшний **www-браузер** должен «уметь» гораздо больше, чем просто отображать страницы HTML и переходить по гиперссылкам. Фактически, на него ложится задача работы данными Internet во всём их многообразии, сюда входит и поддержка постоянно развивающихся стандартов, и обеспечение безопасности, и многое другое.

В Linux есть довольно большой выбор www-браузеров, однако первым действительно современным свободным приложением для работы с Internet стала Mozilla, а затем её потомки, которые сегодня вполне успешно конкурируют с аналогичными коммерческими программами. Mozilla — это целый пакет приложений для работы с Интернетом: мощный, насыщенный функциями коммуникационный центр для персонального компьютера. В состав пакета входит браузер, программа для работы с электронной почтой и редактор www-страниц. История Mozilla началась в 1998 году, когда фирма Netscape опубликовала исходные тексты своего браузера Netscape Navigator. Одно из важных свойств пакета Mozilla — его принципиальная расширяемость. В Mozilla реализован язык XUL на основе XML, при помощи которого очень легко разрабатывать дополнительные компоненты Mozilla, ориентированные на выполнение специальных функций.

Более современен FireFox, разрабатываемый командой Mozilla на основе исходных кодов, соответствующих только WWW-браузеру. Остальная часть FireFox написана полностью на XUL, поэтому разработка этой молодой программы идёт существенно бодрее и проще, её настройка считается самой гибкой среди www-браузеров, а главное, любой желающий может написать на высокоуровневых языках программирования XUL/JavaScript и опубликовать свой модуль расширения (т. н. plugin; на сегодня таких модулей известно более полутораста).

О www-браузерах, разработанных специально для той или иной среды рабочего стола, уже шла речь выше (они превосходно справляются с ролью файловых диспетчеров). Важная разновидность www-браузеров — текстовые браузеры, т. е. те, которые могут быть запущены в любом текстовом терминале Linux. Самый старый и известный из них, один из прототипов современных www-браузеров — Lynx. Он не имеет возможности отображать графическую информацию, но отлично поддерживает HTML, формы и таблицы. Современные версии поддерживают также соединения, защищённые при помощи SSL. Links — это текстовый браузер, на первый взгляд очень похожий на Lynx, но все же несколько отличающийся от него:

- умением работать с таблицами и фреймами;
- отображением цветов, указанных в HTML-странице;
- использованием выпадающих меню (как в Midnight Commander);
- возможностью загрузки файлов в фоновом режиме.

Помимо возможности *просмотра* WWW-страниц часто выпадает необходимость их «скачивания», т. е. записи в файл. Это же относится и к ресурсам, доступным по протоколу FTP. Все описанные выше браузеры способны записывать HTTP- и FTP-ресурс в файл, но для удобной работы из командной строки они, как правило, непригодны. Кроме старой и весьма простой утилиты ftp, имеются два её мощных расширения: lftp и wget. Обе утилиты поддерживают как FTP, так и HTTP, причём lftp может работать, как и ftp, в режиме «оболочки», а wget предназначена именно для работы из командной строки. Если при получении файлов с какого-то сервера или группы серверов необходимо описывать множество исключений (чтобы не скачать лишнего), выполнять какие-то действия (например, заполнять формы или выполнять java-сценарии), можно воспользоваться более сложными программами pavuk или httrack.

Почтовые программы

Подобно тому, как Firefox возрождает WWW-ипостась Mozilla, Thunderbird повторяет — и расширяет — почтовую составляющую Mozilla. Большинство сказанного о Thunderbird на сегодня справедливо и для

MozillaMail. Так же, как и в Firefox, в Thunderbird используется часть исходного кода Mozilla, которая работает с сетью (на этот раз — с отсылкой почты и доступом к почтовым ящикам), а интерфейс и архитектура приложения в целом — переделаны для того, чтобы избавиться от стародавних частей Netscape и облегчить дальнейшую разработку. Thunderbird (как и MozillaMail) обладает самым мощным на сегодняшний день встроенным антиспам-фильтром. Если непрошеная почта всё-таки попадает в ваш почтовый ящик, просто показывайте её Thunderbird со словами «это — спам!». Через некоторое время программа сама научится отличать непрошеную почту от полезной. Как и Firefox, Thunderbird легко расширять собственными модулями, написанными на высокоуровневых языках, и можно очень гибко настраивать.

Ещё один **почтовый клиент**, несколько уступающий Thunderbird по возможностям, но превосходящий его по быстродействию, называется Sylpheed. Интерфейс этой программы весьма похож на стандартную почтовую программу для Windows, Outlook Express, что может помочь избежать лишних хлопот при смене операционной системы. Автор этой программы, Хироюки Ямамото, человек аккуратный и пунктуальный, так что некоторый недостаток возможностей (эта программа умеет столько же, сколько и Outlook Express) компенсируется безотказной работой и гибкой системой интеграции с другими утилитами системы (антивирусом, антиспам-фильтром и т. п.). Кроме того, существует ветка Sylpheed, называемая Sylpheed-Claws, в которой проходят проверку все нововведения. Стабильная версия Sylpheed-Claws работает ничуть не хуже авторской Sylpheed, а возможностей у неё больше.

Поскольку управление электронной перепиской — одна из задач рабочего стола, в каждой среде рабочего стола есть свой собственный почтовый клиент. Почтовый клиент для KDE называется KMail, он поддерживает как локальную доставку почты, так и множество почтовых протоколов (POP3, IMAP, SMTP). Почтовый клиент для GNOME называется Evolution, он интегрирован с календарём, адресной книгой и претендует на функции индивидуальной «записной книжки».

Электронная переписка сама по себе не требует графического интерфейса, для чтения и написания электронных писем вполне достаточно возможностей терминала и текстового редактора. Среди текстовых почтовых клиентов для Linux наиболее известны Mutt и Pine, оба очень функциональны, поддерживают множество протоколов и форматов почтовых ящиков, хорошо настраиваются. Требовательным пользователям, которые хотят иметь возможность изменять внешний вид и способ работы почтового клиента, дополнять его сценариями и получать от почтовых служб всё, что те могут дать, рекомендуется Mutt. Тем же, кому главное — просто получать, читать и отправлять почту (со всеми полагающимися удобствами), стоит начать с Pine. Любители Emacs используют встроенный в него модуль GNUS, весьма богатый функциями.

Обмен сообщениями

Если компьютер подключён к Internet постоянно, бывает удобно пользоваться службами, передающими сообщения в реальном времени (instant messaging service). Таких служб довольно много, самая популярная из них — ICQ. Множественность объясняется тем, что в большинстве случаев этот сервис предоставляется *централизованно*, какой-нибудь крупной корпорацией. Во многих случаях *серверы* этих служб не доступны под свободной лицензией. Исключение в ряду «собственников» — служба Jabber, основанная на полностью открытом протоколе XMPP. Jabber позволяет любому сообществу создавать собственные сервера, управляемые собственными администраторами. Сам Jabber-сервер имеет возможность соединять своих клиентов не только с другими Jabber-серверами, но и со службами ICQ, MSN, Yahoo и AIM. В Linux есть несколько клиентских программ для обмена мгновенными сообщениями. Особняком стоят клиенты IRC (**I**nternet **R**elay **C**hat), службы с более долгой историей и сложным протоколом (имеется в виду и сетевой протокол, и протокол работы пользователя в IRC).

Psi — удобный графический клиент сети быстрого обмена сообщениями Jabber (а значит, по всем протоколам, которые поддерживает выбранный Jabber-сервер). Psi поддерживает такие возможности Jabber, как одновременная работа с несколькими серверами, конференции, криптозащиту передаваемой информации (через SSL и GnuPG), работу через HTTP (S) прокси-сервер и т. д. SIM — многопротокольный клиент обмена мгновенными сообщениями. Поддерживаются протоколы ICQ, Jabber, MSN, AIM, YIM, а также LiveJournal. Кроме того, имеется множество модулей, реализующих дополнительные возможности. Есть вариант SIM, ориентированный на среду KDE. «Прицельно» на

среду KDE ориентирован и другой мощный клиент, имеющий поддержку также и IRC, — Kopete. На среду Gnome ориентирован Gaim — наиболее мощный и наиболее гибко настраиваемый клиент. Имеет модули доступа почти ко всем мыслимым протоколам, позволяет писать сценарии на Perl и TCL. Для IRC есть и специальные клиенты: ChatZilla (как можно догадаться из названия, он «встроен» в Mozilla, но доступен и как дополнение к Firefox) или X-Chat — весьма мощная программа, ориентированная на «хитрости» IRC.

Предупреждение! Обмен информацией и бессмысленными текстами при помощи любой из перечисленных служб, а также телефона, *не заменяет* человеческого общения! Помните, что компьютер передаёт только данные, но не эмоции.

Не обойдён стороной и интерфейс текстовой консоли: CenterICQ, поддерживающий несколько протоколов (среди них Jabber и IRC); licq, обладающий как текстовым, так и графическим интерфейсами (следовательно, им можно пользоваться и находясь за рабочей станцией, и дистанционно); irssi, нацеленный на службы типа IRC (на сегодняшний день поддерживаются IRC, SILC и ICB), и т. д.

Офисные программы

Важной частью современной рабочей станции являются так называемые офисные средства обработки информации. Под **офисными приложениями** обычно понимают стандартный набор из текстового процессора, средства работы с электронными таблицами, средства создания презентаций, средства для работы с базами данных. Все перечисленные офисные приложения входят в пакет OpenOffice.org — это свободный набор офисных программ, не уступающий по возможностям несвободному Microsoft Office, а кое в чём даже превосходящий его. Например, частность, которая может иметь очень важное значение: компонент OpenOffice.org OpenWriter позволяет экспортировать документы непосредственно в формат PDF. Интерфейс OpenOffice.org устроен принципиально так же, как и у аналогичных продуктов Microsoft, так что пользователю, привыкшему к Microsoft Office, не составит большого труда перейти к работе в OpenOffice.org. Кроме того, OpenOffice.org позволяет работать со всеми форматами файлов Microsoft Office.

История OpenOffice.org напоминает историю Mozilla: поначалу проект (под именем StarOffice) развивался закрыто, без доступа мирового программистского сообщества к исходным текстам. Однако в 2000-м году компания Sun Microsystems открыла исходные тексты программного продукта, образовав OpenOffice.org. Так же, как и в случае Netscape/Mozilla, пара StarOffice/OpenOffice.org использует двойное лицензирование, дающее право как свободного доступа к исходным текстам, так и использования их в закрытых коммерческих продуктах.

По возможностям OpenOffice.org остаётся самым развитым и полным офисным пакетом для Linux, однако есть и другие офисные средства. В частности, офисный пакет Koffice, ориентированный на среду KDE, в котором есть примерно тот же набор офисных приложений, что и в OpenOffice.org. Кроме того, есть отдельные офисные приложения, не составляющие пакетов — словарный процессор Abiword и электронные таблицы GNUmeric.

Графика

Чем проще пользовательская задача, тем больше программ под Linux её решают. В частности, манипуляция геометрическими фигурами с возможностью изменения их параметров (цвета, размера и т. п.), хранением набора фигур в файле и преобразованием получившегося изображения в растровый формат — довольно простая задача, требующая аккуратной реализации основных функций какой-нибудь высокоуровневой библиотеки (или двух — интерфейсной и графической). Неудивительно, что редакторы с подобными возможностями есть и для каждого рабочего стола, и независимо от них. Это утверждение относится и к ещё более простым программам работы с растровой графикой. Ниже описаны только существенно более сложные программы.

Векторная графика

Векторной графикой называется способ работы с изображениями, при котором оно представлено в виде фигур, каждая из которых имеет собственное описание (тип, размеры, кривизну или иные параметры составных частей, их цвета, способ представления и т. п.). Некоторые графические устройства (например, распознающие формат PostScript) умеют сами интерпретировать описания фигур, для других необходимо заранее просчитать и сформировать картинку программным путём.

Работа с PostScript и PDF

Современная полиграфия уже не мыслится в отрыве от компьютеров, все допечатные материалы обычно существуют в электронной форме, и именно электронные документы подаются на печатающие устройства для вывода. Причём для современной полиграфии de facto стандартом является формат PostScript. PostScript — это язык описания страницы, позволяющий представить любые полиграфические материалы в векторном формате (однако он допускает и включение растровых фрагментов). Файл в формате PostScript фактически представляет собой программу, описывающую, какие действия нужно произвести, чтобы получить требуемый вывод. Профессиональные печатающие устройства умеют самостоятельно интерпретировать документы на языке PostScript.

PDF (**P**ortable **D**ocument **F**ormat, переносимый формат документов) создан на основе языка PostScript. Его основная задача — обеспечить одинаковый внешний вид документа в любой операционной системе. В PDF есть специальные возможности для публикации документов в Сети, в частности, поддержка гиперссылок, а некоторые возможности языка PostScript оттуда, наоборот, исключены.

Ghostscript — интерпретатор языка описания страниц PostScript и файлов в формате PDF (формат переносимых документов). Ghostscript преобразует PostScript во многие растровые форматы, подходящие для вывода данных на экран или на принтер, не поддерживающий PostScript. Обычно Ghostscript используется для просмотра файлов PostScript и для печати на принтерах, не поддерживающих язык PostScript, GhostScript используется множеством приложений для вывода данных на печать. Графический интерфейс для GhostScript предоставляет программа GhostView (команда gv), она позволяет отображать документы в форматах PostScript и PDF в графической среде X Window System. Для различных манипуляций с файлами в формате PostScript предназначен пакет утилит командной строки psutils, с их помощью можно выбрать, переупорядочивать, масштабировать страницы в PostScript-файлах, изменять параметры текста и делать многое другое.

Специально для просмотра PDF-файлов предназначена программа xpdf, она позволяет переходить по гиперссылкам в документе, просматривать структуру документа, производить поиск и поддерживает сглаживание шрифтов. На основе исходных текстов xpdf создана библиотека poppler, предназначенная для отрисовки PDF, которая стала основой других популярных приложений для просмотра PDF: kpdf (компонент графической среды KDE, отличается более богатыми интерфейсными возможностями) и Evince. Многие дистрибутивы Linux включают Acroread — версию известного приложения Adobe Acrobat для Linux, однако, в отличие от названных выше, оно является несвободным программным продуктом.

Диаграммы

Отдельно стоит упомянуть редакторы диаграмм и блок-схем, которые часто смешивают с обычными редакторами векторной (плакатной) графики. Между тем, задачи у них разные: если для плакатной графики главное — построение «картинки», соответствующей задумке автора по *внешнему виду*, то в диаграмме автора более беспокоит *логическое* соответствие изображения проекту и его *наглядность*. Поэтому при построении диаграммы много внимания уделяется «стрелочкам» и прочим соединительным линиям, оптимальному размещению объектов на странице, типизации объектов и т. п.

Самая старая из подобных утилит, xfig, и по сей день активно используется, формат её диаграмм распознают многие средства работы с векторной графикой. Более мощной является утилита Dia, возможности которой продолжают расти (среди проектов: перевод диаграмм, представленных в нотации языка моделирования программных продуктов UML, непосредственно в текст программ на C++ и

других языках). Аналогом Dia для KDE является встроенная в пакет KOffice утилита Kivio.

Плакатная графика

Что же касается собственно векторной (плакатной) графики, то и здесь есть из чего выбирать. Например, Inkscape — программа векторного рисования общего назначения. Она использует в качестве формата собственных файлов W3C SVG и обладает не только полным набором базовых функций работы с векторными объектами и слоями, но и рядом функций, аналогов которым нет и в крупных закрытых продуктах, таких как Adobe Illustrator или Corel DRAW. Кроме того, в ней реализован механизм расширений, благодаря которому к программе можно дописать новые функции на языках Perl, Python и Ruby. Непосредственный предок, от которого «отпочковался» Inkscape — программа Sodipodi, обладающая сходными возможностями и более продолжительной историей. Многообещающе выглядит проект Skencil, позволяющий редактировать некоторые виды PostScript-файлов. Среда KDE также имеет «свой» редактор векторной графики, Karbon14, входящий в состав семейства программа KOffice.

Растровая графика

Растровая графика означает работу с изображением, представленным в виде матрицы точек («пикселей»). Это значит, что при сильном увеличении границы любого объекта будут выглядеть «лесенкой» из точек (в отличие от векторного представления, где увеличение повышает качество изображения). С другой стороны, растр — удобный для компьютерной обработки формат представления фотографий, сделанных от руки рисунков и прочих изображений, которые нельзя расчленить на отдельные фигуры.

В GNU/Linux есть развитые средства для редактирования растровой графики. Самым мощным из них является GIMP (**G**NU **I**mage **M**anipulation **P**rogram). С её помощью пользователь сможет редактировать изображения, создавать логотипы и другие графические элементы, особенно полезные при создании Web-страниц. GIMP включает много инструментов и фильтров, аналогичных тем, которые можно найти в коммерческих графических редакторах, а также несколько возможностей, эксклюзивных для этой программы. GIMP предоставляет возможность работать с цветовыми каналами, уровнями изображения, накладывать эффекты, сглаживать шрифты и конвертировать изображения в разные форматы. В GIMP имеется собственный язык программирования сценариев (на основе Scheme), на котором можно создавать довольно замысловатые дополнения к основной программе. Такие дополнения можно писать также на Tiny-Fu (облегчённая версия Script-Fu), Python, Perl и C#. Недостаток GIMP — слабая поддержка цветовой модели CMYK, используемой в полиграфии, поэтому в электронной документации, редактировании изображений для www-страниц и прочих областях, не имеющих дела с бумагой, его применяют чаще.

Очень полезен набор утилит для обработки графики из командной строки — ImageMagick. В этот набор входят утилиты для отображения (display), преобразования (convert) изображений, захвата изображений с экрана (import) и даже собственный интерпретируемый язык программирования, Magick Scripting Language. Для полуавтоматического перевода из растрового представления в векторное существует несколько специальных утилит, например, autotrace/autofig или potrace.

Трёхмерная графика

Для Linux создано несколько программных пакетов, работающих с пространственными представлением объектов.

Исходные тексты одного из самых мощных пакетов трёхмерного моделирования, пересчёта (рендеринга) и анимации — Blender — в 2002 году были открыты и весь проект полностью переведён под свободную лицензию. Авторы Blender пришли к выводу, что открытая разработка *инструмента* более эффективна и прибыльна для тех, кто этим инструментом (а не его продажей) зарабатывает. Для этого пришлось выкупить находящиеся в собственности спонсоров части проекта у хозяев за сумму сто тысяч евро. Искомую сумму предоставило сообщество пользователей Blender, уже тогда немалое: каждый внёс сколько смог, и менее чем за два месяца денег на счёте оказалось достаточно. С тех пор круг пользователей и возможности Blender продолжают постоянно расти.

Для выполнения задач, совмещаемых Blender, есть и отдельные программные средства. Например, популярный пакет трассировки лучей (трёхмерного проектирования и сценографии) POV-Ray, с помощью которого создаются проекты удивительной сложности и красоты (например, перевод картины Уильяма Марлоу «Каприччо» в трёхмерное представление — с тем только, чтобы из определённой точки *повторить* её). Многие графические редакторы имеют встроенные средства анимации, а иные (как, например, CinePaint, называвшийся ранее FilmGimp) специально разрабатываются для покадровой обработки видео.

Не стоит забывать, что популярный нынче стандарт OpenGL — открытый; он разрабатывался для UNIX-подобных систем и используется большим числом программ для Linux (в том числе и Blender). К сожалению, производители *аппаратного* обеспечения (видеокарт), как правило, скрывают не только устройство своих карт, но даже и способ их низкоуровнего использования. Поэтому в открытом доступе оказываются лишь готовые драйверы (без исходных текстов) к *некоторым* версиям ядра Linux и определённым сборкам XOrg. Отображение трёхмерных объектов с пересчётом на программном уровне пока работает существенно медленнее, хотя ничуть не хуже, поэтому используя OpenGL для игр и прочих программ, требующих действительно быстрой работы графической подсистемы, нужно всегда помнить о необходимости получить — возможно, несвободный — драйвер.

Мультимедиа

Музыкальные шкатулки

Программ-проигрывателей звуковых файлов в Linux не перечесть. Очевидный лидер по популярности среди них — XMMS (**X** **M**ulti **M**edia **S**ystem). Помимо основной функции — играть музыку (поддерживается множество форматов) — в нём реализовано немало звуковых и визуальных эффектов благодаря большому количеству расширений. Интерфейс XMMS аналогичен интерфейсу не менее популярного в системах Windows приложения WinAMP (кстати, XMMS умеет использовать «шкурки» WinAMP2). Почти не уступает XMMS его «брат» BEEP, использующий графическую библиотеку GTK2, а не GTK. Есть и другие программы, которые ничуть не хуже этих играют музыку. Обычно каждая среда рабочего стола реализует собственный проигрыватель звуковых файлов, хотя бы для того, чтобы воспроизводить собственные звуковые эффекты, связанные с различными системными событиями, однако с их помощью прослушивать файлы может и пользователь.

Очевидно, что для прослушивания звука совсем не обязательно использовать графический интерфейс, поэтому в Linux есть большое количество терминальных утилит для воспроизведения звука. Некоторые из них, например, mpg123, mpg321, ogg123 или splay, предназначены для проигрывания оцифрованного звука (возможно, в сжатых форматах), другие, такие как lazy или cd-console, управляют музыкальными лазерными дисками, есть утилиты, играющие музыку в нотном (midi) и других форматах — timidity (она отличается тем, что преобразует ноты, записанные для инструментов в оцифрованное звучание этих инструментов, а значит, не требует MIDI-устройства), mikmod (распознаёт множество форматов: MOD, STM, S3M, XM и т.д.), sidplay и прочие. Чтобы пользователь не запутался, специальные оболочки, например mpfc или cplay, предоставляют общий интерфейс ко всем консольным проигрывателям.

Музыкальные редакторы

Часть профессиональных музыкантов предпочитает использовать для записи и сведения многоканального звука дорогие специализированные цифровые станции: в этом повинна и реклама, и низкое, с точки зрения профессионала, качество звука большинства звуковых карт в компьютерах общего назначения. Несмотря на это и для таких компьютеров существует немало программ, работающих со звуком на профессиональном уровне. Такие программы можно разделить на две категории: нотные редакторы, задача которых — создание, редактирование, запись и нотное представление *музыкальных композиций*, и звуковые редакторы для собственно звука, а также преобразования его, наложения эффектов и т. п.

Нотные редакторы

В операционных системах, основанных на GNU/Linux, также присутствуют мощные программы для редактирования музыки и звука. Пожалуй, самым известным из них является Rosegarden. Программа изначально разрабатывалась для профессиональных **мультимедиа**-станций от Silicon Graphics и работала на операционной системе IRIX, потом она была перенесена на Linux, а исходные тексты программы были открыты. Сегодня Rosegarden представляет из себя развитый MIDI- и аудиосеквенцер, нотный редактор, а также редактор общего назначения для сочинения и редактирования музыки. Он прост в изучении и идеально подходит для композиторов, музыкантов или студентов музыкальных специальностей, работающих в маленькой студии или записывающихся дома.

Noteedit — нотный редактор (редактор партитур), основанный на MIDI-библиотеке TSE3. Он может писать и читать MIDI-файлы и сигналы от внешней MIDI-клавиатуры. Системные MIDI-устройства используются для воспроизведения нотной записи. Имеется возможность сохранить партитуры в формате MusiXTeX или Lilypond для последующего вывода на печать.

MusE — это MIDI-секвенсер в стиле Cubase/Logic Audio, поддерживающий ввод MIDI-событий с клавиатуры и последующее их редактирование в нотном редакторе, матричном редакторе, редакторе списка событий и редакторе ударных инструментов.

Редакторы и фильтры оцифрованного звука

Популярный свободный редактор звука — Audacity. Он умеет записывать звук сразу в форматы WAV, AIFF, AU, IRCAM или MP3. В нём есть всевозможные инструменты для редактирования записанного звука, в том числе встроенный редактор амплитуды, настраиваемый режим отображения спектрограммы и средства частотного анализа звуковых дорожек. Встроенные простейшие аудиоэффекты включают усиление баса, WahWah, удаление шума и т. д. Audacity поддерживает модульные дополнения, в которых обычно поставляются более сложные аудиоэффекты. В список поддерживаемых форматов модулей входят VST, LADSPA и Nyquist.

Sweep — это многоканальный звуковой редактор, в котором реализованы все основные операции, такие как удаление, копирование, вставка и применение эффектов, оформленных в виде плагинов, к любой части звукового файла. Примерно теми же возможностями обладают и другие редакторы звука — Rezound, WaveSurfer и GNUSound.

Как и в случае с другими мультимедиа-форматами, в Linux существуют терминальные утилиты для обработки звука, не требующие графического интерфейса. Основной пакет терминальных утилит для работы со звуком называется SOX, в него входят утилиты для преобразования, записи и проигрывания звуковых файлов, поддерживается множество форматов.

При помощи консольных утилит можно также сжимать звуковые файлы в различные форматы с потерей качества. Содержимое файла, сжатого «с потерей качества», может быть неотличимо на слух от содержимого исходного файла: алгоритмы преобразования учитывают человеческую физиологию, например, формат MP3 не воспроизводит слишком высоких звуков, а слишком низкие не разделяет на два канала. Смысл термина «потеря качества» — в том, что из упакованного файла *исходный* восстановить уже нельзя. Сжатие с потерей качества можно настраивать на определённую мощность потока упакованных данных: чем больше данных можно передавать в единицу времени, тем чище звук, поэтому такие форматы подходят для передачи по сети (например, интернет-радио).

Основные форматы с потерей качества — это MP3 (с ним работают упаковщики lame/toolame, bladeenc) и OGG Vorbis (утилита oggenc). Эти форматы (особенно OGG) хорошо подходят для упаковки качественной музыки. Файлы в формате OGG, упакованные семикратно (192 kbp/s), почти неотличимы на слух от исходных. Если необходимо сжать звук с ощутимой потерей качества (но без потери членораздельности и раз в двадцать), можно использовать другие форматы — gsm, aiff, adpcm, speex (сжатие речи) или bonk. Последний формат — нестандартный, он поддерживается одноимённой утилитой и отличается большой гибкостью, так как может работать и в режиме «сжатие без потерь». Для сжатия без потерь разработан специальный формат — FLAC, его распознают многие программы (в том числе и утилиты от авторов этого формата — flac и metaflac) и даже аудиоустройства.

Видеопроигрыватели

Наиболее полнофункциональным и удобным «домашним кинотеатром» для Linux является программа xine. Xine поддерживает mpeg-2 и mpeg-1 (включая DVD) потоки, MPEG-4 и другие форматы. Альтернативный ему универсальный проигрыватель — MPlayer. Существует приложение для проигрывания видеопотока, получаемого по Сети — VideoLAN (vlc), которое работает с форматами MPEG1, MPEG2, MPEG4 (также известный как DivX) и DVD.

Xawtv — программа для просмотра и записи видеопотоков Video4Linux, то есть программа для просмотра ТВ. Xawtv использует набор графических элементов Athena. Может использоваться совместно с VDR для просмотра цифрового спутникового, кабельного и эфирного ТВ формата DVB.

Видеоредакторы и конвертеры

В Linux есть выбор средств для преобразования и обработки видео. LiVES (the Linux Video Editing System) претендует на звание простого, но мощного средства редактирования и эффект-обработки видео. Базируясь на GTK+, оно использует для работы такие широко распространённые средства, как MPlayer/mencoder и ImageMagick (в будущем, возможно, GStreamer и Xine). В настоящий момент рекомендуется использовать LiVES для работы с небольшими файлами.

GStreamer представляет собой библиотеку для обработки медиапотоков, основанное на идее объединённых в графы фильтров, обрабатывающих медиаданные. Приложения, использующие эту библиотеку, смогут производить любую обработку медиаданных от обработки звука до проигрывания видео. Модульная архитектура позволяет реализовать поддержку любого нового формата данных, просто установив соответствующее расширение.

Kino — это нелинейный редактор цифрового видео (DV) для GNU/Linux. Он хорошо интегрирован с IEEE 1394 и позволяет захватывать изображение, управлять VTR, и записывать на камеру. Этот редактор записывает видео на диск в формате AVI в кодировках type-1 DV и type-2 DV. Существуют терминальные утилиты для обработки видеопотока, например, пакет transcode. Кодирование и декодирование видеопотока осуществляется с помощью загружаемых модулей. Также поддерживается загрузка внешних фильтров. В число модулей входят: модули импортирования из DVD, элементарных MPEG (ES) и программных потоков (VOB), видео в формате MPEG, цифрового видео (DV), потоков YUV4MPEG, поддержка формата файлов NuppelVideo и необработанных потоков видео; модули для записи DivX, OpenDivX, DivX 4.xx или несжатых файлов AVI с MPEG, звука в форматах AC3 или PCM; дополнительные модули для записи отдельных кадров (PPM) или потоков YUV4MPEG. Пакет transcode содержит набор утилит для демультиплексирования (tcdemux), выделения (tcextract) и декодирования (tcdecode) видеопотока, исследования (tcprobe) и сканирования (tcscan) ваших файлов и пост-обработки файлов AVI, изменения заголовков файлов AVI (avifix), соединения нескольких файлов в один (avimerge) или разделения большого файла на несколько AVI-файлов меньшего размера (avisplit) для размещения на CD.

Ffmpeg — это «сверхзвуковой» кодировщик/декодировщик видео и звука, работающий в режиме реального времени, а также потоковый сервер и преобразователь различных звуковых и видеоформатов. Ffmpeg умеет захватывать видеосигнал из источника Video4Linux и преобразовывать его в файлы различных форматов на основе компенсирующего кодирования DCT/motion. Звук при этом сжимается по алгоритму MPEG-2 или алгоритму, совместимому с AC3.

Запись CD и DVD

Для записи дисков и сопровождающих запись задач в Linux есть как минимум два приложения с графическим интерфейсом: входящее в комплект приложений для KDE k3b и написанное на GTK xcdroast. Фактически, оба этих приложения — это графические оболочки над терминальными утилитами для записи CD и DVD, в первую очередь cdrecord и cdrdao, которыми можно пользоваться и непосредственно из командной оболочки. cdrecord — утилита для записи дисков с цифровыми данными, в нём реализована полная поддержка аудио-, смешанных, мультисессионных и CD+ дисков. cdrdao — программа записи аудиодисков в одну сессию позволяет управлять областями в начале дорожек данных

(длиной до 0, ненулевые аудиоданные) и, например, международными стандартными кодами записи. Все данные, которые будут записаны на диск, должны быть описаны в текстовом файле. Аудиоданные могут быть в форматах WAVE или raw.

Помимо того, для Linux есть множество программ, позволяющих производить обратную операцию: считывание данных с аудиодиска в файл, такие программы называются грабберами (grabber). Один из удобных грабберов с графическим интерфейсом — Grip.

Издательские системы

Подготовка печатных документов и оригинал-макетов изданий — хоть и не очень распространённое, но важное приложение компьютера. В Linux самой известной и системой подготовки качественных документов, пригодных к печати в типографии, является TeX. TeX — это фактически специализированный язык программирования, специально разработанный для описания типографского набора. Документ в TeX представляет собой текст, сопровождённый командами, указывающими, какое форматирование следует произвести. Возможности TeX очень широки, однако для того, чтобы их использовать в полной мере, требуются довольно серьёзные познания в нём. Чем шире познания — тем легче, быстрее и удобнее готовить документы в TeXe и тем лучше их качество.

Обычно TeX используется совместно с пакетами форматирования более высокого уровня, например, LaTeX. LaTeX — это комплекс написанных на языке TeX макропакетов, предоставляющих удобные средства для решения типичных задач оформления печатных изданий. В LaTeX определено оформление для нескольких стандартных классов документов.

LyX — это современных подход к написанию документов, разрывающий с устаревшей парадигмой использования компьютеров как пишущих машинок, применяемой в большинстве других систем подготовки документов. Он разработан для тех, кто хочет получить профессиональное качество документа при печати, не тратя при этом много времени и усилий, и не становясь специалистом по полиграфическому оформлению. Основное новшество в LyX — это WYSIWYM (What You See Is What You Mean — вы видите то, что вы имели в виду), которое означает, что автор сосредотачивается над своей работой, а не над деталями оформления документа. Это позволяет продуктивно работать, оставляя заключительное оформление специальному движку (такому как LaTeX), который специально разработан для подобных задач. С LyX автор может сконцентрироваться на содержании своей работы и позволить компьютеру взять большинство забот об оформлении на себя.

В Linux есть по крайней мере одна программа для визуальной подготовки оригинал-макетов, аналогичная **издательским системам** Adobe PageMaker, QuarkXPress и подобным — scribus. Возможности его могут быть более ограничены, чем у перечисленных коммерческих аналогов, однако он распространяется свободно и в настоящее время активно разрабатывается.

Нельзя объять необъятного

В этот краткий и фрагментарный обзор не вошли собственно инструменты для разработки программного обеспечения, которые развиты в Linux чуть ли не лучше, чем все остальные приложения, поэтому написать краткий обзор для них гораздо сложнее. Не сказано ничего о серверах баз данных (не не потому, что таких серверов нет!) и серверах приложений в составе сложных проектов. Обойдены вниманием и игры — любой читатель этого раздела сможет самостоятельно решить, сколько внимания и каким из них уделять.

Напоследок повторим: главной целью приведённого обзора приложений для Linux было показать, что приложения есть и их много, нужно только достаточно внимательно искать — и нужное обязательно найдётся. В ALT Linux для поиска доступен очень удобный инструмент — менеджер пакетов APT и команда apt-cache search. Поскольку в современные дистрибутивы Linux входят тысячи пакетов, почти наверняка среди них найдётся нужное приложение. Кроме того, любому пользователю Internet доступен поисковый сайт http://google.com, наиболее подходящий для поиска чего бы то ни было, а для поиска существующих приложений для Linux можно воспользоваться специализированными сайтами — http://rpmfind.net, http://freshmeat.net, http://rpm.pbone.net или сайтом, посвящённым выбранному дистрибутиву.

Обратите внимание, что все названные в этой лекции приложения — это свободно распространяемые и разрабатываемые программы, если не оговаривается обратное. Характерная черта свободного программного обеспечения состоит в том, что если для решения какой-то задачи есть одно свободное приложение, то всегда есть и несколько других, так что пользователь всегда может выбрать себе приложение по вкусу, а если подходящего не обнаружится — изменить для себя одно из уже существующих или даже написать новое. В конце концов, нет ничего дороже и милее сделанного собственными руками велосипеда.

1Пользователи, знакомые с MS-DOS, вспомнят Norton Commander, а пользователи помоложе — Far Manager.

Технология

Благодаря распространённости ОС Windows на сегодняшнем рынке очень многочисленны приложения, разработанные для этой платформы[1]. Однако зависимость коммерческого приложения от определённой платформы (ОС) может быть не всегда удобной или выгодной. На этот случай существуют средства, позволяющие программам, разработанным для ОС Windows, работать в другой операционной системе. Одним из наиболее развитых среди подобных средств является WINE.

WINE (**W**ine **I**s **N**ot **E**mulator) *не является* эмулятором операционной системы: то есть он не создаёт изолированной среды для выполнения и не обеспечивает доступ к низкоуровневым системным ресурсам, таким как непосредственный доступ к оборудованию. Функция WINE состоит в том, чтобы, с одной стороны, предоставить win-приложению Win API — стандартный системный интерфейс операционных систем Windows, а с другой стороны, транслировать запросы win-приложения в соответствующие системные вызовы (Unix API). WINE работает на различных Unix-системах, в том числе на Linux. Таким образом, WINE — это своеобразная «прослойка» совместимости между win-приложениями и host-системой[2].

Хотелось бы отметить, что процесс WINE всегда выполняется в непривилегированном режиме и не требует никакой модификации ядра операционной системы (в том числе динамически загружаемых модулей). Отсюда следует простой вывод относительно безопасности: любые проблемы, которые могут быть вызваны запуском win-приложений, будут ограничены правами доступа того пользователя, который запустил WINE. В результате win-приложения будут подчиняться политике доступа UNIX-системы и не смогут её нарушать.

У данного ограничения есть и другая практическая сторона: в WINE нет поддержки *низкоуровневого* обращения к оборудованию (драйверов оборудования, прямой работы с USB-устройствами). Всё периферийное оборудование следует подключать и настраивать в host-системе: для win-приложений эти устройства могут быть доступны стандартным способом через файловую систему или другие стандартные интерфейсы (например, TWAIN для сканеров, который реализован в WINE как обёртка над библиотекой SANE).

Наиболее распространённый способ применения WINE — запуск двоичных win-приложений в Unix-среде. Удобство заключается в том, что при этом не требуется никак изменять приложение — один и тот же вариант годится и для Windows, и для WINE.

Другое, на сегодняшний день пользующееся незаслуженно меньшей популярностью применение — с помощью WINE разработчики ПО могут компилировать свои win-приложения из исходных текстов непосредственно в двоичные исполняемые файлы для Unix. Опять-таки, это те же самые исходные тексты, из которых компилируются двоичные файлы для Windows.

Третий способ использования — WINE позволяет скомпилировать win-приложение из исходных текстов в исполняемый exe-файл, который будет работать на любой Windows-системе.

WINE состоит из нескольких компонент, которые условно можно поделить на три части:

libwine
: Библиотека, предоставляющая Win API для win-приложений. По количеству предоставляемых функций её можно сравнить с Qt — столь широк спектр предлагаемых вызовов: от операций с файлами до построения графического интерфейса и обращения к базам данных.

wine
: Среда для исполнения двоичных win-приложений, предоставляет программам окружение, неотличимое от Windows. Это окружение помимо Win API включает реестр, стандартные каталоги и файлы. Реестр является единственной изменяемой информацией, необходимой для работы WINE и win-приложений в нём.

утилиты

Утилиты, имитирующие некоторые стандартные win-приложения: текстовый редактор (блокнот), файловый браузер и т. п. Средства компиляции и отладки: имеются заголовочные файлы, которые описывают доступное API, компилятор winegcc, представляющий собой обёртку над gcc, отладчик winedbg и прочие вспомогательные утилиты.

Разработка

WINE — это свободный проект, который был начат в 1993 году. На тот момент распространённой платформой была Win16 (Windows 3.1), на неё и был ориентирован WINE, на сегодняшний день основным русло разработки — Win32. Исходные тексты WINE выпускаются под лицензией LGPL (Lesser GPL), никаких ограничений по доступу к исходным текстам и их модификации не имеется. WINE снабжён достаточно вразумительной документацией, имеется ряд списков рассылки (англоязычных), как для пользователей, так и для разработчиков, где оперативно решаются любые вопросы.

Процесс разработки WINE во многом похож на метод, применяемый при разработке ядра Linux. Все присылаемые (в специальную рассылку) патчи подвергаются рассмотрению разработчиков, которые могут высказывать свои соображения и добавления. Имеется один человек, Александр Джулиард, который принимает решение о том, включать ли патч в CVS, и при необходимости совершает в нём какие-то улучшающие изменения (например, исправляет ошибки в оформлении кода). Ведётся контроль и учёт всех отправляемых патчей и их авторства. Прежде чем патч будет принят, он проходит автоматическое тестирование — WINE компилируется с новым патчем, и выполняется регресс-тестирование: запускается тестовый код, написанный практически для каждого API, с помощью которого можно удостовериться, что добавление патча не нарушает совместимость.

Реализация

Успешность и корректность работы win-приложений в WINE естественно определяется тем, насколько среда WINE неотличима от Windows с точки зрения win-приложения. Иначе говоря, вопрос в том, насколько полно Win API и другие стандартные компоненты и процедуры Windows реализованы в WINE. Текущая оценка полноты реализации конкретных функций публикуется на сайте разработчиков WINE. В WINE реализованы функции практически всех динамических библиотек (DLL), входящих в Windows: начиная от 16-разрядных и заканчивая появляющейся поддержкой 64-битного режима. На хорошем уровне находится поддержка OLE, MSI и DirectX.

Если говорить об общей оценке полноты реализации, то на сегодняшний день разработчики называют цифру 90%. Однако относиться к этой цифре нужно не совсем так, как к обычному процентному соотношению. Дело в том, что с точки зрения успешной разработки WINE Win API должно быть таким, *каким его хотят видеть программы*. Полных и безошибочных спецификаций Win API в публичном доступе нет (и никогда не было), и это во многом определило характер разработки на платформе Win. Большинство разработчиков win-приложений используют только незначительную часть стандартных функций API, а остальные необходимые функции реализуют самостоятельно и поставляют вместе с программой. В последние годы подмножество широко используемых функций API в широкой массе win-приложений уже стабилизировалось и практически не меняется. Для WINE это означает, что цифра 90% означает стабильную работу большинства win-приложений в WINE даже больше, чем в 90% случаев.

По этой же причине не так существенна опасность отставания от изменений, вносимых в Win API в рамках операционной системы Windows. Единственная особенность — развивающаяся поддержка 64bit, при разработке архитектуры WINE принималась в расчёт возможность расширения в этом направлении. Поэтому как только функции Win64 API получат более широкое распространение, добавление их поддержки в WINE не заставит себя долго ждать.

Настройка локального win-окружения

Прежде чем начинать работать с WINE, *каждому пользователю*, от имени которого будут запускаться win-приложения, необходимо настроить локальное win-окружение. Настройка окружения выполняется автоматически при первом запуске WINE (достаточно ввести команду wine в командной строке и дождаться завершения её работы).

При первом запуске WINE создаёт необходимую инфраструктуру в домашнем каталоге данного пользователя, для чего выполняет следующие действия:

- создаёт начальную версию реестра;
- выстраивает соответствия каталогов host-системы и логических дисков WINE;
- создаёт каталог с программами, который будет служить основным диском (C:) для win-приложений, для удобства этот каталог доступен как wine_c в домашнем каталоге пользователя.

По умолчанию логические диски WINE будут расположены следующим образом:

```
C:      $HOME/wine_c
D:      $HOME/Documents
E:      /media/cdrom или /mnt/cdrom
```

Пример 1. Размещение логических дисков WINE по умолчанию

Если какого-то из нужных каталогов не окажется, то соответствующие ссылки просто не будут созданы. Как минимум один диск — C: будет создан в любом случае. Остальные диски необязательны, даже одного C: будет достаточно для работы в WINE. Правила создания ссылок по умолчанию описаны в файле /etc/wine/map_devices.sh, при необходимости их можно изменить. Изменения в этом файле затронут всех пользователей, которые будут затем выполнять первый запуск WINE.

Каждый пользователь может вручную изменить соответствия логических дисков WINE каталогам host-системы или создать любое количество допольнительных дисков. Все логические диски для WINE представлены обыкновенными символьными сылками на каталоги в каталоге $HOME/.wine/dosdevices:

```
[tester@tacit tester]$ ls -l $HOME/.wine/dosdevices
total 0
lrwxrwxrwx 1 tester tester 13 Nov 25 14:50 a: -> /media/floppy
lrwxrwxrwx 1 tester tester  8 Nov 25 14:50 a:: -> /dev/fd0
lrwxrwxrwx 1 tester tester 26 Nov 25 14:50 c: -> /home/tester/.wine/drive_c
lrwxrwxrwx 1 tester tester 10 Nov 25 14:50 com1 -> /dev/ttyS0
lrwxrwxrwx 1 tester tester 22 Nov 25 14:50 d: -> /home/tester/Documents
lrwxrwxrwx 1 tester tester 12 Nov 25 14:50 e: -> /media/cdrom
lrwxrwxrwx 1 tester tester 10 Nov 25 14:50 e:: -> /dev/cdrom
```

Пример 2. Логические диски и устройства WINE

Чтобы создать новый логический диск или изменить имеющийся, достаточно создать новую символьную ссылку с нужным именем.

```
[tester@tacit tester]$ ln -s /var/data/1c ~/.wine/dosdevices/f:
[tester@tacit tester]$ ls -l ~/.wine/dosdevices
total 0
lrwxrwxrwx 1 tester tester 13 Nov 25 14:50 a: -> /media/floppy
lrwxrwxrwx 1 tester tester  8 Nov 25 14:50 a:: -> /dev/fd0
```

```
lrwxrwxrwx 1 tester tester 26 Nov 25 14:50 c: -> /home/tester/.wine/drive_c
lrwxrwxrwx 1 tester tester 10 Nov 25 14:50 com1 -> /dev/ttyS0
lrwxrwxrwx 1 tester tester 22 Nov 25 14:50 d: -> /home/tester/Documents
lrwxrwxrwx 1 tester tester 12 Nov 25 14:50 e: -> /media/cdrom
lrwxrwxrwx 1 tester tester 10 Nov 25 14:50 e:: -> /dev/cdrom
lrwxrwxrwx 1 tester tester 12 Nov 25 14:54 f: -> /var/data/1c
```

Пример 3. Создание логического диска wine

Создавая логические диски WINE, нужно принимать в расчёт, что права доступа win-приложений к файлам на этих дисках будут определяться правами доступа данного пользователя к реальным файлам host-системы.

Запуск win-приложений

Общее правило для запуска всех win-приложений в WINE — запускаемые файлы должны находиться в **области видимости** WINE, то есть на одном из логических дисков WINE или в его подкаталогах. Если программа поставляется на компакт-диске, то не забудьте должным образом смонтировать диск[3], прежде чем обращаться к нему из WINE. Обратите внимание, что в этом случае у вас должен быть разрешён запуск приложений с компакт-диска. Если приложение распространяется не на диске — не забудьте сначала скопировать его в область видимости WINE.

Для запуска win-приложений проще всего воспользоваться файловым браузером winefile: его можно запустить из командной строки. Здесь достаточно перейти в необходимый каталог и запустить программу двойным щелчком мыши.

Можно запускать win-приложения как обыкновенные исполняемые файлы host-системы (например, из Midnight Commander или из командной строки), для этого должна быть запущена системная служба (service) wine.

Установка и удаление win-приложений

Как и в Windows, перед использованием большую часть приложений сначала потребуется установить. Установка производится обычным для Windows способом — с помощью поставляемой вместе с win-приложением программы установки. Разница в том, что в случае WINE программа будет установлена в локальном win-окружении пользователя.

Для установки win-приложения следует любым удобным способом запустить программу установки (чаще всего setup.exe). Дальше можно действовать по инструкции, предлагаемой поставщиком win-приложения.

Многие win-приложения запрашивают перезагрузку для завершения установки. Естественно, перезагружать host-систему при этом не следует. В локальном win-окружении процедуре загрузки Windows соответствует команда wineboot — её можно вызвать из любой командной строки. Если в этот момент в WINE выполняются другие приложения, то рекомендуется их завершать до перезагрузки.

Для удаления win-приложения, установленного в win-окружении, следует воспользоваться утилитой uninstaller. Эта утилита выводит список установленных в win-окружении приложений (если они зарегистрированы в реестре). Чтобы удалить приложение, выберите его из списка и нажмите кнопку «Uninstall». Если приложения, которое вы хотите удалить, нет в списке, то достаточно просто удалить каталог с приложением (можно воспользоваться для этого программой winefile, а можно — стандартными средствами host-системы).

Безопасность

Советы по соблюдению должного уровня безопасности в WINE могут быть сведены к двум простым соображениям:

- жертвой ошибки в программе или злонамеренных действий со стороны win-приложения (вируса) может стать только та часть файловой системы, которая входит в область видимости WINE;
- права доступа к данным определяются правами пользователя, запустившего WINE.

Поэтому следует *максимально ограничить* область видимости WINE, включив туда только те данные, доступ к которым *необходим* win-приложениям для работы. Можно сформулировать и несколько более конкретных рекомендаций:

- Никогда не запускайте WINE от имени пользователя root! Запущенное от имени root win-приложение получит привилегии этого пользователя. Для работы они ему никогда не потребуются, а во вред могут быть употреблены запросто.
- Не следует давать доступ win-приложениям к важным системным каталогам, и в особенности к корневому каталогу файловой системы (“/”). Даже целиком включать домашний каталог пользователя в зону видимости WINE почти наверняка не требуется.

Шрифты

Системе WINE доступны те же шрифты, что и другим приложениям в host-системе[4]. Соответственно, к этим шрифтам получают доступ и win-приложения.

Сделать определённые шрифты доступными win-приложению можно несколькими способами:

- Для всех пользователей — поместить шрифты в host-системе штатным для системы способом;
- Для конкретного пользователя — поместить эти шрифты в каталог шрифтов пользователя (~/.fonts);
- Только для win-приложений данного пользователя — поместить шрифты непосредственно в каталог шрифтов на логическом диске WINE (обычно ~/wine_c/windows/fonts).

Существует базовый набор шрифтов (Corefonts) для систем Windows — многие приложения рассчитывают на наличие в системе шрифтов со стандартными именами из данного набора. Для корректной работы таких приложений, возможно, потребуется установить этот набор шрифтов. Его можно скачать с сайта http://corefonts.sourceforge.net.

Дополнительная информация

Наиболее подробную документацию о WINE для пользователей и разработчиков можно найти на сайте разработчиков WINE. К сожалению, на сегодняшний день эта документация доступна только на английском языке.

На официальном сайте проекта WINE доступна самая свежая информация по WINE, сведения о разработке, включая дальнейшие планы, списки рассылки, исходные тексты WINE, списки работающих win-приложений, поддерживаемых фукнций WinAPI и множество другой информации. Из русскоязычных ресурсов можно обратиться к проекту «Русский WINE», который позиционируется как ресурс, объединяющий русскоязычных пользователей WINE. Здесь большое внимание уделяется проблемам локализации WINE и запуска специфических приложений, актуальных для русскоязычных пользователей. Многие пользователи могут найти для себя полезным русскоязычный форум, посвящённый WINE.

1Здесь и далее мы будем называть такие приложения win-приложениями.
2ОС Unix/Linux, в которой установлен и выполняется WINE.
3Нужно делать это вручную, или монтирование выполняется автоматически — зависит от вашего дистрибутива и стиля работы.
4Для получения списка доступных шрифтов WINE использует пакет fontconfig, а для отрисовки символов — библиотеку freetype2.

Настройка инфракрасного порта (IrDA)

Прежде всего необходимо установить пакет irda-utils и в файле /etc/modules.conf добавить следующие строки:

```
alias irda0 smc-ircc
# IrDA over a normal serial port, or a serial port compatible IrDA port
alias tty-ldisc-11 irtty

# IrCOMM (for printing, PPP, Minicom etc)
alias char-major-161 ircomm-tty
```

Пример 1. Загрузка модулей для IrDA

Проверьте настройки в файле /etc/sysconfig/irda:

```
IRDA=yes
DEVICE=irda0
DISCOVERY=yes
```

Пример 2. Настройка службы irda

Теперь нужно перезапустить службу irda командой service irda restart. Если у вас инфракрасный порт встроен в ноутбук, воспользуйтесь командой irattach. Для проверки настроек используйте программу irdadump, в выводе которой должна появиться информация о вашем устройстве.

Настройка BlueTooth

Рассмотрим подключение USB-BlueTooth. В файле /etc/bluetooth/hcid.conf класс нашего устройства (Local device class) следует изменить на class 0x520104;. В том же файле программу ввода PIN следует изменить на pin_helper /etc/bluetooth/pin.sh;.

PIN-код для доступа к телефону указывается в файле /etc/bluetooth/pin.sh в такой форме:

```
#!/bin/sh
echo "PIN:123"
```

Пример 3. Указание PIN-кода

Где 123 — это PIN-код. В /etc/bluetooth/rfcomm.conf нужно внести bind yes; (по умолчанию указывается no) и адрес телефона (параметр device).

Теперь можно запустить службу — service bluetooth start. В телефоне необходимо включить доверительный режим (введён пароль компьютера — 123).

Соединение через GPRS (в GSM)

Поскольку скорость соединения может достигать 171200 бит/сек, и тарифицируется соединение по объёму переданной информации (вне зависимости от направления), этот режим является наиболее выгодным для владельцев GSM-телефонов. Подробности можно выяснить у вашего оператора. Телефон может быть подключен через последовательный порт (/dev/ttyS?), USB (появится /dev/usb/ttyUSB0), инфракрасный порт (/dev/ircomm0) или BlueTooth (/dev/rfcomm0). Если вы сомневаетесь в конкретных настройках для своего телефона, ищите в http://google.com слова «*модель_вашего_телефона* GPRS Linux».

При настройке дозвона (аналогично модемному соединению) Вам нужно указать дополнительную строку инициализации (в kppp она будет второй) примерно такого вида: AT+CGDCONT=1,"IP","*точка.доступа*". Точную информацию можно найти на сайте оператора (так например, для оператора СМАРТС *точка.доступа* будет “internet.smarts.ru”, а для Мегафон-ПОВОЛЖЬЕ “internet.volga”.

Номер телефона для дозвона зависит от модели телефона:

Модель телефона	Номер телефона
Siemens, Motorola	*99***1#
Nokia	*99#

Если в настройках телефона, в разделе «точка доступа по умолчанию» корректно занесены все настройки, то с номером *99***1# можно избежать добавления дополнительной строки инициализации — все настройки будут взяты из телефона. Это может быть удобно при работе с несколькими телефонами, подключенными к разным операторам сотовой связи.

Логин/пароль произвольны, хотя часто в качестве логина указывают название оператора (ncc, mts, nw, beeline), а пароль — пустой или совпадает с логином. Если набор номера даёт ERROR, скорее всего услуга GPRS у вас отключена оператором.

Соединение в режиме простой передачи данных (GSM)

В этом режиме скорость передачи ограничена значением 9600 бит/секунду, соединение тарифицируется по времени, практически как обычный разговор, что выходит очень дорого. Использовать его стоит только если телефон или оператор не поддерживают GPRS.

Имя пользователя и пароль обычно не важны, попробуйте указывать просто название.

Настройка соединения через SkyLink (CDMA-2000)

Необходимо указать строку инициализации AT+CRM=1;&C0, указать номер телефона #777, указать логин mobile и пароль internet.

Возможные проблемы:

- возможно, соединение не будет устанавливаться с погасшим экраном
- не забудьте выставить одинаковые скорости в телефоне и программе подключения
- не пытайтесь определять связь с телефоном по работе в терминальной программе
- если ничего не получается, попробуйте выключить и включить телефон.

История и право

Свободные программы и сообщество

Свободное или несвободное

Когда деревья были маленькими...

Когда разработка одного экземпляра вычислительной машины длилась годами, а сам этот экземпляр выглядел как магазин продажи холодильников, программы для этой машины числились наряду с мелкими и средними деталями для неё. Скажем, для записи данных на перфоленту прежде всего необходим перфоратор, затем — процессор, который будет этим перфоратором управлять, а уж затем — всякие там провода, программы для записи данных и прочая необходимая мелочь. Красноречиво этот факт подтверждает термин «операционная система», точнее, его исходное английское написание — «operating system». Дескать, есть некоторая «система» (system) — это высокоорганизованный набор электронных деталей. Но система на настолько сложна, что «использовать» (operate) эту систему лучше по определённым правилам, тут вот даже кой-какие программки для этого написаны, они и задают правила «использования системы» (operating system). И том же говорят и наши термины: «аппаратное обеспечение ЭВМ» и «программное обеспечение ЭВМ», подразумевающие, что утилиты и компиляторы так же крепко привязаны к этой вот конкретно машине, как микросхемы и вентиляторы.

Различение «набора деталей» (system) и «набора программ» (operating system) далось машиностроителям не вдруг. Сначала открыли цикл ПКМ: «программирование — компиляция — отладка (неуспешная) — программирование — и. т. д.... — отладка (успешная)». С чугунной деталью так не поступают: выточил деталь — поставил в машину — она взорвалась вместе с деталью... Затем выяснилось, что, в отличие от детали, программу удобно разрабатывать совместно, когда каждый может подключиться к работе и добавить несколько нужных лично ему функций; надо только определиться, кто отвечает за всю программу целиком и принимает решения.

Только после распространения языка программирования «Си» и операционной системы UNIX стало зарождаться понятие **кроссплатформенности**, которое сделало это отличие совершенно очевидным, и навсегда оторвало программный продукт от жёсткой привязки к «system» и даже «operating system». Кроссплатформенность, то есть возможность использования одного и того же программного продукта независимо от того, где он запускается («платформы» или «среды»), открыла новые возможности в совместной разработке. К одному и тому же проекту стали присоединяться программисты, работающие на разных компьютерах. Они свободно обменивались друг с другом программами, но ещё чаще — исходными текстами к ним, так как выполняться скомпилированная программа может только в рамках одной платформы, а исходные тексты всё равно нужны для доработки и исправления.

Безущербное копирование и продажа воздуха

Где-то в это время, в конце семидесятых прошлого века, стало очевидным и свойство, которое так сильно — и технологически, и экономически — отличает программные продукты от продуктов других производств. Это свойство **безущербного копирования**.

Что нужно сделать, чтобы вместо единственной хорошей детали стало две одинаковых? Купить чугунную болванку, а рабочий выточит из неё деталь. Вторая деталь нам ни к чему, мы её отдали; чего мы при этом лишаемся, ведь первая-то при нас? Разумеется, денег: болванку мы покупали, а токарю — платили, это его труд создал копию. Копирование причиняет ущерб, который, по-честному, надо возмещать. Например, тем, что детали отныне мы будем не дарить, а продавать.

Так. А что нужно для того, чтобы из единственной хорошей программы стало две одинаковых? Выполнить операцию копирования, больше ничего. Если тот, кому вы собираетесь эту программу

подарить, принесёт с собой записываемый лазерный диск, вам это ни во что не станет. Копирование экземпляра программы **безущербное** для хозяина этого экземпляра.

Тут между разработчиками программ случился раскол. Одни (будем называть их «эгоистами») говорили: «надо запретить свободное распространение не только исходных текстов, но и вообще самих программ. Программы надо продавать! Это ж сколько денег можно выудить прямо из воздуха. А вы, господа альтруисты (назовём их так), просто олухи царя небесного, маргиналы, пиджаков не носите, питаетесь кофе и гамбургерами, и ничего не знаете о том, как вести бизнес».

«Альтруисты» возражали: «нам всё равно, что носить, мы любим кофе, и это наши программы, мы их написали, мы хозяева — что хотим, то и делаем. А если мы не будем свободно распространять исходные тексты, как вообще тогда программы разрабатывать? В одиночку, что ли?»

«Зачем в одиночку? — напирали эгоисты — Из воздуха-то можно много денег понаделать, мы наймём вам помощников, сколько надо, только не показывайте никому трудов своих, а для пущей крепости хозяевами вашим программ будем считаться мы, а уж мы-то позаботимся о том, чтобы каждая копия превращалась в зелёную бумажку».

Мировое сообщество в прошлом веке в целом знать не знало ни о каком безущербном копировании, так что с удовольствием начало вырабатывать законы, это копирование запрещающие. Главный ход, конечно, состоит в том, что право распоряжаться такой вот «эгоистской» программой (в отличие от чугунной детали) должно всегда оставаться у одного человека — правовладельца. Он и будет получать доход с продажи, его права и защищают законы.

Это нужно знать, между прочим. Допустим, вы пошли в магазин, и купили там втридорога лазерный диск с копией «операционной системы» или каким-нибудь другим программным продуктом, предоставляемым без исходных текстов и без возможности их получить. Ущерб, который потерпит ваш карман от такого безущербного копирования, наводит на мысль о том, что ваша копия сопровождается правовладельческой лицензией. По сути дела, обладая копией — экземпляром — программы, вы *не становитесь хозяином* программы как произведения. Вам бы дальнейшее её распространение не причинило никакого ущерба, но права на это у вас нет, такое право сохраняется только за «продавцом» или другим правовладельцем программы.

Правовладельческие программы называют также **ПО ЗК** — Программное Обеспечение с Закрытым Кодом, потому что исходный текст этих программ использовать запрещено, а чаще всего и смотреть на него нельзя, или «проприетарными» (от proprietary, «собственнический»), что отражает правовладельческую суть дела.

Свобода программного обеспечения и GNU Public License

Люди, названные нами «альтруистами», — программисты университетской школы и вообще люди, ищущие интереса прежде выгоды, поначалу просто не обращали внимания на воцаряющийся правовладельческий строй. Они продолжали свободно обмениваться текстами и совместно дорабатывать программы друг друга. Однако чувствовали они себя неуютно, и неспроста. В начале восьмидесятых последовала серия громких судебных процессов по алгоритму «ты написал принадлежащую мне программу и украл её у меня, передав товарищу». Беспокойство нарастало.

Между тем стало вполне понятно, что именно отличает традиционный свободный способ существования программы от «правовладельческого». Достаточно, чтобы любой, кто получил копию программы, вместе с ней получал *те же* права, что и сам автор, в первую очередь право её изменять и копировать. При этом *каждый* становится «хозяином программы как произведения». Тогда можно будет с чистой совестью продолжать работать совместно и свободно: тиражировать программу, продавать её, раздавать за так, улучшать и ухудшать, тиражировать то, что получилось, и т. д.

Принципы свободы программного обеспечения несколько позже сформулировал видный деятель Фонда Свободных Программ (FSF) Ричард Столлман:

- Свобода запускать программу в любых целях
- Свобода изучать работу программы и вносить в неё изменения
- Свобода распространять копии программы

- Свобода публиковать внесённые изменения и распространять копии изменённой программы

Смысл этих принципов ясен: во-первых, пользоваться, во-вторых исправлять то, что не нравится, в-третьих, привлекать всех, кто пожелает, в-четвёртых, работать совместно. Стоит один из четырёх принципов нарушить, и работать над программой сообща будет невозможно.

Перечисленные четыре принципа свободы ничем не ограничивают потенциального хозяина копии программного продукта. Он может делать с этой копией что угодно, в том числе *нарушать* своими действиями хоть все четыре принципа. Скажем, исправлять ошибки и распространять результат с правовладельческой лицензией, то есть объявить себя хозяином всех последующих копий. Исправления и исходный текст исправленной программы он, конечно, никому не откроет — это «ноу-хау», рычаг для добычи прибыли из воздуха. Немало в прошлом свободных программ и их частей заключено нынче во мрак несвободных программных продуктов. Плохо здесь совсем не то, что кто-то зарабатывает на чужом, в сущности, труде; зарабатывает — это хорошо!

Плохо, что общественной пользы от такого сокрытия никакой: никто не узнает, что это были за ошибки; свой талант человек, считай, в землю зарыл — в надежде, что вырастет золотое дерево. Эффективность разработки при этом — мизерная: второму такому же умнику придётся начинать с *того же самого* места, с открытого исходного кода, и третьему, и десятому. И вот сидят они по своим углам, никому ничего не говорят, и исправляют одни и те же ошибки без надежды поделиться результатами успеха с коллегами.

После нескольких болезненных судебных столкновений с правовладельческими лицензиями по схеме «программировал-то ты, а хозяин — я», Ричард Столлман и его коллеги из сообщества профессиональных программистов GNU разработали собственную *лицензию* — соглашение между хозяином копии программы и предполагаемым получателем этой копии. Она получила название GNU Public License (Общественная Лицензия GNU). Только после принятия условий этой лицензии человек может считаться хозяином полученной им копии программного продукта и исходных текстов.

Общественная Лицензия GNU декларирует пресловутые «четыре свободы», которые гарантированы хозяину копии, содержит некоторые юридические тонкости относительно их соблюдения и — главное — накладывает на хозяина единственное строгое ограничение: если новоиспечённый хозяин захочет распространять программу или изменённый её вариант, условия её распространения должны быть *не хуже* GPL, то есть включать в себя «четыре свободы» и вот это самое требование их дальнейшего соблюдения. Программисту, знакомому с понятием «наследование» идея вполне прозрачна, законникам пришлось объяснять, а со стороны это выглядит совсем просто: требования свободы нельзя нарушать, и всё. Чаще всего проще не мучиться, изобретая собственные лицензии, а публиковать плоды совместных трудов по той же GPL.

GPL не имеет прямого отношения к сути свободного ПО, это *инструмент*, оружие, которое в мире правовладельческих лицензий помогает сообществу отстаивать право на свободу.

Сообщество кого?

Важнейшее прикладное следствие свободы программного продукта — свободная совместная его разработка. Всякий, кто хочет и может поучаствовать в общем деле, может приступать к работе без каких-либо не имеющих отношения к вопросу ограничений. В каком качестве заинтересованный человек может присоединиться к сообществу?

Ядро (Core Team)

Во-первых, ответственный и квалифицированный человек может взять на себя часть разработки программного продукта, и, что гораздо важнее и сложнее, принятие решений по всем принципиальным вопросам. Таких людей, как правило, совсем немного, а доля их участия в жизни продукта велика. Неважно, почему они согласны тянуть лямку впереди всех, важно, что они это делают, имеют достаточно рабочего времени и личных достоинств. Они составляют **ядро** сообщества. Как правило, это умудрённые опытом программисты или эксперты, словом, те, к чьему мнению заслуженно прислушиваются. Именно от мнения ядра зависит, как будет развиваться продукт, как и когда вносить в

него предложенные поправки, когда и как делать выпуски (release) и т. п.

Ядро соответствует понятию «команда разработчиков» старой, индивидуалистической схемы. Здесь более или менее срабатывают и практические рекомендации старой школы: ядро должно быть небольшим, чтобы могло само с собой договариваться, должно иметь разделение труда, чтобы каждый занимался в первую очередь тем, к чему имеет больший талант, должно иметь «главного» для волевого разрешения неразрешимых вопросов и т. п., словом, типичный «team» из пяти-семи человек.

Сообщество разработчиков (Development Team)

Сообщество разработчиков — самая главная часть любого свободного программного продукта. Дело в том, что полдюжины человек в ядре, конечно, не могут *слишком* быстро заниматься разработкой. Что ещё важнее — они не могут в одиночку управляться со всей эксплуатационной историей, то есть отслеживать, как, когда и с каким успехом применялась программа, какие были ошибки, были ли они исправлены и как. Именно свобода ПО предоставляет любому, кто поучаствовал в эксплуатационной истории, сделать эту историю публичной и внести её в русло общей работы.

Самый простой способ подключиться к сообществу — найти ошибку в программе, исправить её и написать разработчикам. Вполне вероятно, что вы, в силу специфики работы, наткнулись на эту ошибку впервые. Программа свободная, исходные тексты имеются, право на публичное её изменение — тоже, так что толковый программист в состоянии ошибку найти и исправить. Если он при этом будет настолько ленив и неблагодарен, что не сообщит авторам, всё сообщество укажет на него с осуждением. Если узнает, конечно.

Что всего приятнее — практически любой человек, мало-мальски сведущий в программировании, может поправить ошибку. Титаном мысли для этого можно и не быть. Если предложенная тобою заплата (patch) написана из рук вон плохо, ответственный разработчик из Core Team предпочтёт скорее переписать это место наново, чем выбросить и забыть, на то он и ответственный.

Свободные проекты очень много внимания и ресурсов уделяют поддержке сообщества разработчиков. Техническая документация по проекту (руководство по сборке и установке), как правило, не уступает пользовательской. Для разработчиков заводятся специальные списки почтовой рассылки, в которых они обмениваются информацией друг с другом и задают вопросы, и — что тоже очень важно — автоматизированные системы отслеживания ошибок и предложений (т. н. «bug tracker»-системы).

Эта политика весьма эффективна ещё и вот в каком плане. Программист, вместо того, чтобы кидаться к клавиатуре и начинать программировать порученный ему продукт с нуля, берётся за мышь и педантично обшаривает такие ресурсы, как SourceForge, FreshMeat или NonGNU, на предмет подходящего свободного проекта, который решает те же задачи. С очень высокой степенью вероятности такой проект найдётся, хотя, возможно, и не будет обладать какими-то специфическими для конкретной ситуации способностями. Всё что останется программисту — это модифицировать свободный продукт, дополнив его этими возможностями. Добросовестный и благодарный человек не преминет поделиться с сообществом, а от происков любителей правовладельческого лицензирования его защищает GPL: если программу вообще собираются распространять, исходный код необходимо открыть, а уж от этого до интеграции изменений ядром разработчиков в основной проект дорожка прямая.

Само собой, этот Добросовестный Программист автоматически становится членом сообщества разработчиков, что, помимо прочего, сулит и некоторые служебные перспективы.

Сообщество пользователей (Users Community)

Бессмысленно говорить о каком-то программном продукте, не упоминая тех, кто им будет пользоваться. Старая «хакерская» практика — разработчик и пользователь суть одно лицо, потому что программировать гораздо интереснее, чем запускать программу — преобразовалась за эти годы до неузнаваемости. Мы не знаем, почему те или иные члены сообщества разработчиков занимаются нашей программой — из-за денег, славы, спортивного интереса, производственной необходимости или на голом энтузиазме; они работают — и огромное им спасибо. Но почему мы, пользователи, хотим видеть программу мощной, толковой и надёжной — совершенно понятно: программа нам *нужна*.

Конечно, пользователь-программист или программист-пользователь продолжают играть немаловажную роль в развитии свободного ПО. Такой человек в сообществе решает сразу три задачи: активно тестирует продукт, находя ошибки и выдумывая, что ещё «в супе не хватает», сам формулирует программистское задание, и сам же его выполняет. Поэтому сообществу всегда выгодно привлечь активного пользователя в ряды разработчиков либо убедить талантливого программиста пользоваться свободным ПО.

Но уверенно надеяться на чудесное слияние столь разных ролей можно только в редких случаях «программирования для программирования», чаще всего пользователем может быть кто угодно, а иногда профессия типичного пользователя *не* позволяет ему быть программистом. Например, пользователь компилятора — скорее всего программист, музыкального проигрывателя — совсем неважно кто, а пользователь кассового аппарата скорее всего имеет самое смутное представление о том, каким образом аппарат цифры показывает.

В конце концов, трёхступенчатый процесс поиска эксплуатационных трудностей, формулировки задачи и программирования давно уже освоен «классической» схемой разработки: пользователь отыскивает повод для улучшения, пользователь **и** программист совместно формулируют задачу, программист её решает. И от программиста, и от пользователя в этом случае требуется желание, возможность и умение взаимодействовать. Не очень-то и редкие качества, если учитывать, что человек — животное общественное и вообще склонен общаться с себе подобными. Однако даже это может стать камнем преткновения: несвободный, «закрытый» способ разработки пошёл по экстенсивному пути развития; для решения каждой задачи нанимается специальный отдельный человек (тестер, консультант, менеджер проекта).

Преткновение как раз в том, в чём закрытый способ разработки отказывает решительно всем, даже самим пользователям: в *заинтересованности*. Общественная схема начинает работать только тогда, когда и пользователь, и разработчик хотят (неважно, по каким причинам) *решать* возникшую задачу (а не только «чтобы была решена»). Но ведь сообщество образуется как раз из заинтересованных, из тех, кому для работы не жалко трудов. Дополнительное требование — пользователь и разработчик должны быть *в состоянии* решить задачу, то есть иметь достаточную квалификацию каждый в своей области. Сообщество вокруг свободного ПО — не для неучей, что правда, то правда. Впрочем, мало найдётся людей, столь невосприимчивых к знаниям, чтобы остаться неучами вопреки активному интересу и всеобщему образованию.

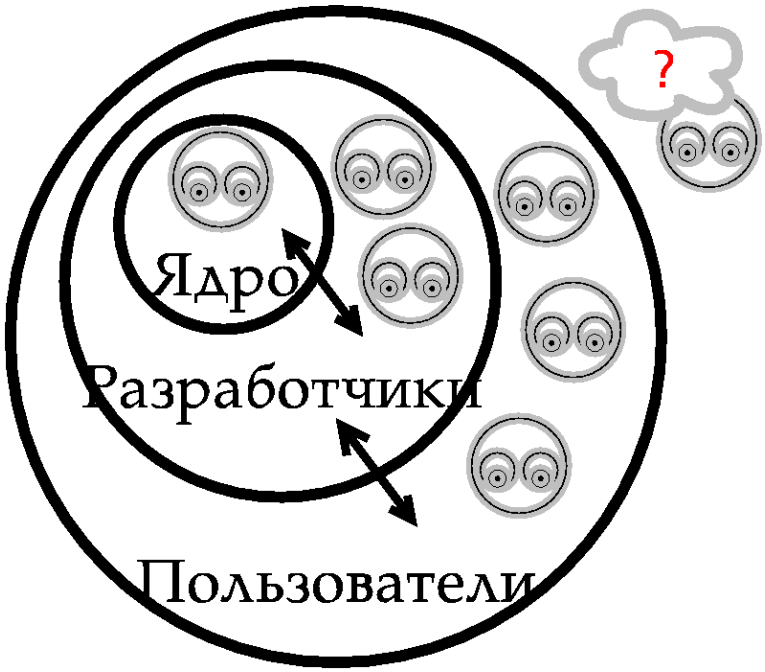


Иллюстрация 1. Сообщество вокруг свободного программного продукта

Единое информационное пространство

Для того, чтобы приведённая выше схема заработала, одного желания мало. Недостаточно и умения. Делать что-то *сообща* можно только если есть *возможность* сообщать и общаться. Передавать друг другу информацию — во-первых, и ориентироваться в ней — во-вторых.

В самом деле. Группы в сообществе — ядро, разработчики и пользователи — представляются эдакими тремя комнатами, где каждый знаком с каждым, все затруднения активно обсуждаются между собой, а двери из комнаты в комнату никогда не закрываются: всякий результат обсуждения нужно довести до сведения товарищей по сообществу, да и просто так поговорить, чтобы быть в курсе, не мешает. А ведь в действительности даже ядро сообщества может состоять из людей, которые живут на различных континентах, и лично каждый не с каждым и знаком. Что и говорить о всемирном сообществе разработчиков или пользователей?

Переоценить вклад глобальных сетей в развитие свободного ПО невозможно. Именно за счёт свободного распространения информации людям не обязательно постоянно находиться в одной комнате в обществе друг друга. Именно за счёт связности сети Интернет все пользователи и разработчики имеют *возможность* собираться вместе, обсуждать проблемы, обмениваться новостями, узнавать друг о друге, словом, вести себя так, как если бы все находились в «одной комнате».

Ещё важнее простота, оперативность и надёжность связи *между* группами, без которой не может жить никакой проект свободного ПО. Если на схеме убрать стрелочки, она превратится в три беспомощных замкнутых множества, в каждом из которых будет куча нерешаемых задач. Ядро не сможет быстро и оперативно разрабатывать программу, внешние разработчики — договориться о том, какие именно задачи решать, куда идти, да и решать ли вообще, а пользователи — добиться удобной и безошибочной работы программы.

Живые люди

Прежде, чем говорить о *технических* возможностях, которые обеспечивают взаимодействие людей в сообществе, стоит сказать пару слов о том, зачем это техническое обеспечение им нужно.

Необходимость общения в сообществе очевидна (сами слова-то почти одинаковые). Столь же очевидно, что никто не сможет заставить члена сообщества общаться, кроме него самого, кроме желания побольше узнать и помочь. С другой стороны, редко у кого такое желание — часть *профессии*. Иными словами, за жажду знаний, как и за миссионерство, нечасто платят. Это означает, что сам *способ* общения и приёма-передачи знаний должен быть настолько прост и вместе с тем соблазнителен, чтобы им добровольно стало пользоваться как можно больше хороших и нужных людей.

Живых людей, которыми движет *желание* работать и *необходимость* общаться (а не наоборот). Из чего следует, что чем удобнее *инструмент* и *структура* обмена информацией, тем проще уступить желанию, смирившись с необходимостью. (В скобках заметим, что закрытый путь разработки решает эту проблему привычным экстенсивным способом: нанимаются специалисты по PR, профессиональные проповедники и автоответчики во плоти, обязанность которых — отвечать на вопросы, даже если в ответах нет никакого смысла).

Кузница, Горнило и Багзилла

Каналы связи *между* группами существуют и сами по себе — за счёт пользователей-разработчиков, которых в свободных проектах немало. Однако этого крайне недостаточно. Как всякий труд, совместная разработка требует дисциплины и формализации, причём чем больше в её составе людей и чем меньше между ними социальных связей, тем строже должна быть дисциплина и требовательнее формальность. В случае свободного сообщества, когда коллеги по проекту могут не быть даже знакомы, вопрос дисциплины и формы встаёт особенно остро: без этого культурные, социальные, психологические и прочие различия возьмут верх, и результатом будет неуправляемое вече. С другой стороны, нельзя перегибать палку, иначе свободно пришедший в сообщество человек плюнет на все формальности и свободно уйдёт.

Структура

Первым делом сообществу стоит договориться о том, кто какие права и возможности имеет, и каким образом их осуществляет. Где хранятся исходные тексты программы и согласно какой дисциплине их менять. Как принимать сообщения об ошибках и реагировать на них. Какая документация существует по программному продукту, где её взять и как с ней работать. Что необходимо знать, прежде чем включаться в сообщество. Где, наконец, находятся *готовые* версии программного продукта, и чем одна отличается от другой.

Вопросов много, и способы ответа на каждый из них мировое сообщество нащупывало десятилетиями. Дело осложнялось тем, что разработчики — движущая сила любого сообщества — редко когда оказывались толковыми веб-дизайнерами. Не в смысле умения проектировать что-либо, а в смысле умения это наглядно показывать. Впрочем, со временем необходимость выдумывать что-то совсем уж своё отпала, потому что выработались эффективные шаблоны организации труда. Тогда-то и стали появляться сетевые проекты, которые предлагают уже готовый набор таких шаблонов — **порталы**. Сообществу остаётся только согласиться использовать этот набор в работе и наполнить шаблоны содержательной информацией.

Так, например, устроен упомянутый выше проект Source Forge: система хранения и обновления исходных текстов, хранилище готовых программ, система отслеживания ошибок, форумы для обсуждения, списки рассылки и т. д. — всё для успешной организации сообщества. Зарегистрироваться, положить первые десятки строк программы — и пошло-поехало. Кстати, на сегодня в SF зарегистрировано больше десяти тысяч проектов и много больше миллиона пользователей.

Лицо

Что меньше всего стоит передоверять универсальным службам, вроде SF, так это — эстетическую и идеологическую составляющую любого сообщества. В конце концов, технические средства — это всего только инструмент, а как им распорядиться решает, ядро при поддержке всех единомышленников. Немалую роль здесь играет всем уже сегодня привычное понятие «сайт» — сетевой WWW-ресурс, посвящённый самому программному продукту, его задачам, способам использования, дисциплине разработки и т. п. От того, как устроен сайт проекта, в немалой степени зависит и интерес новых посетителей сайта, и активность самого сообщества. Сайт — это лицо проекта.

Сайт совсем не обязан быть огромным и сложным. Скорее наоборот: чем легче будет посетителю ориентироваться, тем выше вероятность, что он найдёт себе подходящее место в сообществе. На сайте обычно лежат материалы трёх видов. Во-первых, научно-популярные и идеологические, которые описывают, зачем проект нужен самим его создателям, и чем он может помочь пользователям. Во-вторых — информационная часть с пользовательской и технической документацией, что делает включение в сообщество вопросом чисто техническим. И в-третьих — технически-структурная часть, которая описывает все элементы взаимодействия в сообществе, а также отсылает к используемым ресурсам. Нередки ситуации, когда несколько титульных страниц проекта, особенно небольшого, — дело рук разработчиков, а всю структурную и ресурсную часть (хранение исходных текстов и готовых программ, отслеживание ошибок, информационные рассылки и прочее) берёт на себя какой-нибудь портал.

Обратная связь

Несколько раз уже упомянутая служба отслеживания ошибок — чуть ли не самый важный из механизмов, формирующих сообщество. Дело в том, что распространение свободных программ, равно как и информации о них, через сайт — дело благое, но чрезвычайно неблагодарное, если не предусмотреть *обратной связи* от тех, кто эту информацию усвоил, а программу использует. Стрелки на схеме *обязаны* быть двунаправленными, иначе ядро не получит никакой выгоды от сообщества разработчиков, и все они вместе — от сообщества пользователей.

Чтобы сообщить об ошибке разработчику, необходимо иметь: ошибку, разработчика, и — главное! — *механизм*, позволяющий оформить подобное сообщение. Причём в случае распределённой совместной работы этот механизм должен сопровождаться известной дисциплиной оформления, чтобы первый попавшийся разгильдяй не принялся докучать ответственному разработчику посланиями вроде «Дорогие учёные. У меня который год в подполе происходит подземный стук. Объясните, пожалуйста, как он происходит». В неменьшей степени это относится и к прямой работе сообщества над исходными текстами программы.

Стоит ещё раз подчеркнуть, что средства совместной разработки должны быть в первую очередь удобны технически, то есть самим разработчикам и активным пользователям. Именно с этим связано разнообразие и солидная история как систем отслеживания ошибок, так и систем совместного написания текста программ. В числе первых назовём GNATS, BTS, BugZilla, Mantis, Eventum и множество других, как родившихся в недрах больших сообществ для собственных нужд, так и специально предназначенных для шаблонов совместной работы любого сообщества. Системы совместного ведения архива исходных текстов традиционно называются системами «контроля версий» (version control), хотя их возможности давно уже этим не исчерпываются. Среди них — всемирно популярная CVS, её наследники SubVersion и GNU Arch, специализированные на особые задачи git или monotone, крайне простые (darcs) и весьма мощные (вплоть до поддержки автоматической сборки получившейся программы — aegis), и некоторые другие.

Всякому, кто хочет включиться в сообщество разработчиков, придётся научиться взаимодействовать и с тем, и с другим. Только так можно сохранить относительный порядок при полной свободе добровольно собравшихся вместе людей.

Списки, Блоги и Вики

Для организации взаимодействия *внутри* группы, особенно вне рамок исходных текстов, нужны другие средства. Нужно то самое произвольное общение, которое привиделось в форме «трёх комнат с открытыми дверями». Где формальная сторона была бы сведена к минимуму, а преобладала бы, насколько это возможно, общечеловеческая.

Лет пятнадцать-двадцать назад было трудно себе представить, что в обмене символами посредством компьютера есть так уж много человеческого. Бытовало мнение, что электронная почта — удел профессионалов-компьютерщиков, уже не различающих электронную и явную реальности, а, скажем, сетевой дневник (web log, или «the blog») — это такая форма электронного эксгибиционизма полностью исключённого из «настоящего» социума киберпанка. Скорей уж инструментом, приближающим электронное к человеческому, мыслился ужасный электрод, вставляемый непосредственно в голову и навевающий электронные сны, либо передающий мысли по проводам. Что было и продолжает оставаться фантастикой. Гора не шла к Магомету.

Сейчас мы стали свидетелями того, как охотно Магомет, то есть современное человечество, двинулся к горе. Электронная почта заменила бумажную. Слово «форум» вызывает стойкие ассоциации с веб-сайтом, и только после некоторого напряжение вспоминаешь его словарное значение. В последнее время наблюдается взрыв увлечения электронными дневниками (не в последнюю очередь, кстати, из-за того, что исходный текст «движков» некоторых из них открыт, и теперь каждый может основать свой сайт с дневниками). Купить вещь в Интернет-магазине с доставкой на дом зачастую дешевле, чем влачиться за нею куда-то на склад, не говоря уже о посещении чистого и сверкающего, но людного и дорогого магазина.

Первое, что было придумано, а скорее — возникло само собой для информационного сплочения сообщества — тематические списки рассылки[1]. Как правило, каждое сообщество имеет несколько таких списков — для разработчиков, для пользователей, для обсуждения стратегических вопросов, технические, дизайнерские, наконец, для «роботов», собирающих статистику; крупные проекты, например, дистрибутивы, имеют их до сотни. Список рассылки — то самое место, куда надо задавать вопросы: если человек читает рассылку, ему это для чего-нибудь нужно. Он поможет вам, а когда само попадёт впросак, кто-нибудь поможет ему.

Для общения в совсем необязательном режиме — трёпа, тусовки и прочего, что в цифровом мире заменяет совместное времяпрепровождение — используются пресловутые «форумы», сетевые журналы и обмен текстами в реальном времени (chat). Это ресурсы уже не совсем сообщества, а, так сказать, пригород его, тропки, которыми можно в сообщество прийти. А можно и не прийти, если процесс обмена словами важнее совместной работы. Так что эти форумы — ещё и фильтр ответственного отношения к делу. Гулякам праздным там живётся лучше, чем в дисциплинированной среде сообщества.

Первое исключение из этого правила: ресурсы Internet Relay Chat (irc), которые достаточно стары, как следствие — слегка сложноваты для освоения гуляками и имеют свои культурные традиции защиты от дурака. IRC-каналы часто используют профессионалы для оперативного решения задач в стиле «вопрос-ответ». Второе: всё чаще дневники (Blogger, Live Journal и т. п.) стали использоваться с самыми серьёзными целями: информация для сообщества, обсуждение принципиальных вопросов и т. п. Причина этого — особый способ доступа к информации в сетевых дневниках, когда пользователь читает не то, что ему захотели прислать, а те источники информации, что он предварительно согласился читать. Получается отличный фильтр действительно заинтересованных, что, собственно, и надо сообществу.

Наконец, в последнее время набирает обороты практика совместной разработки сайтов. Эта практика и сопутствующая ей технология получила название «Wiki» (от гавайского wiki-wiki, «быстро-быстро»). Практика в том, чтобы сделать процесс наполнения сайта содержимым *очень* простым. Настолько простым, чтобы туда написать мог *кто угодно*, было бы желание. Ответственному за сайт придётся только просматривать новые поступления на предмет очевидного хулиганства (пресечь которое в Wiki-технологии достаточно просто) и, возможно, самому добавлять страницы структурирующего характера (чтобы сайт не превращался в свалку бессвязных текстов). Несмотря на молодость технологии, большинство крупных проектов уже обзавелось своими Wiki — официальными и неофициальными, которые служат отличным хранилищем данных и поисковой системой по ним. Надо ли говорить, что качество и наполнение Wiki-сайта целиком зависит от усилий сообщества?

Что такое хорошо и что такое плохо

Условия успеха

Сообщество вокруг программного продукта добивается успехов, если созданы условия для совместной работы: свободное распространение, три части сообщества — ответственная (ядро), компетентная (разработчики) и заинтересованная (пользователи), доступное и толковое информационное пространство, в первую очередь — простые и надёжные каналы связи в группах и между ними.

Тогда каждый занимает в сообществе место по силам и желанию, и делает только то, что может и должен. Чем грамотнее спланировано информационное пространство, тем меньше накладных расходов и тем больше приходит компетентных людей, которые по тем или иным причинам готовы развивать программный продукт. Проект, фактически, пишет сам себя, остаётся только направлять его движение и следить за тем, чтобы сообществу хорошо жилось.

Конечно, в настоящей жизни бывает всякое, любая правовая, политическая или экономическая неприятность может пагубно сказаться на жизни программы, но успех крупных свободных программных проектов в наибольшей степени зависит от умения управлять сообществом.

За чертой остаются все, кто по какой-то причине не может включиться в сообщество. Это не означает, что, изолируясь от сообщества, человек не может получить определённую выгоду от использования

свободного программного продукта. Большинство свободных программ вполне «дистрибутивны»: их распространяют и используют как по отдельности, так и в составе **дистрибутива** — тематического или системного сборника программ, с помощью которых пользователь может решать задачи определённого направления. Как правило, дистрибутив даже единственного программного продукта содержит достаточно информации, чтобы пользоваться им автономно. Но в этом случае пользователь лишается *главного* преимущества — поддержки сообщества, непосредственной помощи со стороны знающих людей, которую невозможно ни запрограммировать, ни задокументировать.

К тому же любая толковая просьба о помощи — сама по себе уже *помощь* сообществу! Если вопрос возник, а ответа на него нет ни в документации, ни в списках рассылки, скоро с такими же трудностями столкнутся и другие пользователи. Так отчего бы не прояснить вопрос до того, как станет неудобно многим? А сообщение об ошибке или короткая методичка по найденному вами решению задачи — это уже *услуга* сообществу, которую легко и приятно оказать в корыстных целях: ошибка будет исправлена, а ваш текст опубликован. Взаимодействие с сообществом — это *естественный* способ существования пользователя свободного ПО; если, например, вы страстно хотите каких-либо изменений в программе, но боитесь в этом признаться, вы их не дождётесь. А если вы их изложите сообществу, да подкрепите просьбу добрым словом, пивом, сотней рублей, сотней долларов — чем-нибудь вполне адекватным объёму работ, ответ не заставит себя ждать.

Граждане и гражданки! Проявляйте социальную активность! Включайтесь в сообщество пользователей!

Унесённые призраками

Теперь нетрудно понять, когда разработка ПО по открытой схеме себя не оправдывает. Как только какое-нибудь из перечисленных «условий успеха» нарушается, ограничивается одна из свобод, сразу возникают затруднения, подчас — совершенно непреодолимые в рамках свободной модели.

Нашлись какие-то правовладельческие документы и соблюдать «четыре свободы Столлмана» нельзя? Прощай, открытая совместная разработка! Только своими силами, да ещё и подписку о неразглашении давать придётся.

В стране запрещён или не поощряется Интернет и вообще открытая передача данных? Тогда нет смысла в совместном проекте — без обратной-то связи. Не случайно в Китае так много сильных программистов и хакеров и так мало сообществ свободного ПО (да есть ли они вообще?).

Если пользователи не заинтересованы в самой программе, работают из-под палки или просто ничего не понимают и не хотят понимать, возникает огромный зазор между ними и разработчиками. Тогда каждый, кто и пользователь и разработчик одновременно, бесценен. А таких может не оказаться. Например, до недавнего времени инструментарий *менеджера* — программные средства управления персоналом, сетевого менеджмента и т. п. — был представлен в свободном ПО очень скудно. По причине, как это ни смешно, застарелой нелюбви разработчиков и управленцев друг к другу. Только в последние лет пять свободное ПО начало входить и на этот рынок.

Другой подобный пример — компьютерные игры, где пропасть, разделяющая разработчика и пользователя-игрока преодолевается только на могучей волне энтузиазма; а чаще — не преодолевается и замыкается в ПО ЗК. Правда, и здесь наметились сдвиги: архитектуру компьютерной игры («движок») теперь иногда делают открытой, надеясь на помощь сообщества в главном — в наполнении.

Если потенциальных пользователей очень мало, скажем, дюжина человек на весь мир, то открытость или закрытость разработки никакой роли не играет: всё равно придётся работать с каждым индивидуально, а ждать чудесного самозарождения сообщества разработчиков вообще не приходится. С другой стороны, здесь могут сыграть как раз индивидуальные пристрастия: в научном мире, к которому вполне могут относиться эти двенадцать пользователей, принято открыто обмениваться если не всей информацией, то уж точно — инструментами (а программа для учёного — именно инструмент).

Если в проекте большая «расходная статья» на непрограммистские задачи (скажем, много работы для

художника, электронщика, сценариста и т. п.), а «доходная статья» (техническое обслуживание, заказная доработка, обучение и т. п.) — маленькая или вообще не предусмотрена, делать его свободным, к сожалению, невыгодно. Ситуация, характерная опять-таки для больших компьютерных игр — и для узкопрофильных разработок, где оплачивается участие многих специалистов а возможности пользовательского сообщества невелики.

И всё-таки главное препятствие развитию свободного ПО — у нас в головах. Человечество тысячелетиями вырабатывало культуру права собственности, имея перед собой исключительно материальные объекты, копирование которых требует затрат. Объекты нематериальные, «технологии», появились в массовом порядке менее двух веков назад, а «программным продуктам», безущербность копирования которых очевидна, ещё не скоро исполнится полвека. Неудивительно, что люди, которые *сами не программируют* (управленцы, экономисты, юристы и прочие), всё ещё не готовы отличить собственность на чугунную болванку от собственности на программу. Культура нематериальной собственности только вырабатывается.

Напоследок

Время одиночек, которые самостоятельно, от начала до конца, прорабатывают весь программный продукт, прошло. Для того, чтобы написать хорошую программу, требуются усилия *различных* специалистов: от системного архитектора до кодировщика и от художника до эксперта в прикладной области. Собрать всех этих людей воедино можно либо на добровольной, свободной основе, либо «кнутом и пряником» — деньгами и крепостной зависимостью, правовладением.

Правовладельческая «команда» устроена так: неважно, как человек изначально относится к своим обязанностям и что думает, главное — платить ему и создавать условия для эффективной работы. Тогда он либо начнёт приносить пользу, либо лишится места. Ущерб, который он в этом случае может нанести компании, надо ограничить запретительной лицензией.

Свободное сообщество устроено строго наоборот: неважно, *почему* человек хочет участвовать в разработке, важно, чтобы он *хотел* этого, мог быть полезен и относился к работе ответственно. Ущерб, который может быть нанесён сообществу путём сокрытия информации, ограничивается открытой лицензией GPL.

Важно помнить, что свобода — это в первую очередь ответственность. И дело не только в том, что в свободном мире каждый обязан отвечать за свою работу и не может свалить ответственность на другого. В свободном мире безответственные действия не приносят выгоды. Например, крахом обернётся попытка продавать воздух в стиле ПО ЗК: любой свободный продукт стоит столько, сколько заслуживает, как это и полагается настоящими требованиями рынка; а если искусственно завышать цену одного экземпляра, всегда найдутся желающие приобрести и продавать его по цене естественной.

Что же приносит доход в мире свободного ПО? А что везде: труд, оперативность, информация, качество.

Доработка
: Свободное ПО хорошо тем, что *кто угодно* может адаптировать его под свои нужды. Или заплатить кому-нибудь из сообщества разработчиков за такую адаптацию. Причём чем качественней продукт, чем больше вокруг него сообщество тем *меньше* для самого сообщества себестоимость доработки, тем *больше* разница между тем, что клиент готов заплатить, и затратами.

Заказная разработка
: Сообщество свободного ПО — своего рода биржа труда профессиональных разработчиков. Она имеет свою специфику (например, попадают туда не от безработицы, а напротив, вследствие работы, отчего народ там разборчив к предложениям), грамотно манипулируя которой можно в короткий срок получить в своё распоряжение работоспособную команду с приличной профессиональной историей. Главное — не нарушать принципов их свободы.

Техническая поддержка
: Услуги — работа компетентного человека здесь и сейчас — во всём мире и во все времена ценились высоко. Кому же знать специфику программного продукта, как не активному члену сообщества? Частенько человек *работает* в команде разработчиков близко к ядру, а то и вовсе в нём, а *зарабатывает* на жизнь, скажем, системным администрированием.

Консультация и обучение

То же самое относится и к услугам, так сказать, социального плана. В каждом сообществе наверняка находится человек, не только сведущий, но и умеющий грамотно изложить свои сведения — устно ли, письменно ли. Таких людей немного и оплачиваются их труд довольно высоко. О том, какую пользу они приносят сообществу, строя то самое вожделенное информационное пространство, и говорить не надо.

Время одиночек, наверное, прошло. Прошло и время вождей, ведущих за собою толпы, неведомо куда под неведомо какими лозунгами. Человек — свободная личность — хочет выбирать свой путь самостоятельно, и следовать своему собственному пути, опираясь на опыт и помощь других свободных людей. Не такая уж и малая, в сущности, часть общества — пользователи и разработчики свободных программ — подают пример такого свободного взаимодействия. Возможно, этот пример нельзя скопировать на произвольную повседневность. Даже наверняка нельзя. Повседневность каждого из нас — если, конечно, это наш свободный выбор — невозможно скопировать. Это как программа, написанная лично тобой.

Тем не менее этот пример актуален. Свободные программы для свободных людей.

1Наверное, уже и не надо объяснять, что такое «список рассылки»? Это такая форма электронной почты, когда один человек её отправляет, а несколько — подписчики — получают.

Частное и общественное

Написание компьютерных программ — не так уж давно возникшая форма интеллектуальной деятельности. В написании программы действительно много общего с написанием какого-нибудь литературного или другого нетривиального текста, поэтому совершенно естественно, что с точки зрения правовых отношений программы попали в один класс с такими текстами — «произведений».

Эти правовые отношения регулируются законодательством об авторском праве и на сегодняшний день тесно ассоциируются с широко распространённой системой экономического использования этих прав: торговля экземплярами произведения и запрет на тиражирование произведения (создание новых экземпляров) для всех, кроме обладателя прав. В этой модели распространения к произведению относятся как к собственности правообладателя. Программы, которые распространяются по такой модели, называются точным, но не очень благозвучным в русской огласовке термином **проприетарные**.

Однако не менее широко распространена и другая модель, в которой к произведению относятся как к общественному достоянию, плоду интеллектуального творчества, который должен быть доступен любому и не принадлежать никому в отдельности[1]. Типичный пример — произведения давно умерших классиков, хотя и многие здравствующие авторы (например, учёные) распространяют свои произведения по этой, общественной модели. Несмотря на молодость, программное обеспечение тоже может следовать общественной модели распространения, и здесь возникают два ключевых понятия: **свободное ПО** и **открытые системы**.

Заметьте, что речь не идёт о платном и бесплатном. Экземпляры произведений классиков тоже продаются за деньги, разница в том, что правомерно издавать (тиражировать) произведение, находящееся в общественном достоянии, может любой.

У программ для компьютера есть всё-таки несколько существенных свойств, которые отличают их от текстов на естественном языке, поэтому и при распространении ПО возникают некоторые специфические особенности. Чтобы разобраться в смысле понятий «открытые системы» и «свободное ПО», нам потребуется сформулировать два свойства компьютерных программ:

1. Очень часто, хотя и не обязательно, программа существует в двух видах: производится она в одной форме — в виде **исходного текста**, а распространяется и используется в другой — в виде скомпилированной **двоичной программы**, машинных кодов, по которым невозможно однозначно восстановить исходный текст.
2. Объектом авторского права (то есть **произведением**) является исходный текст программы. Двоичная форма — это уже экземпляр произведения, и в его отношении действуют примерно те же нормы, что и к экземпляру книги, с некоторыми технологическими оговорками. Правовая разница между произведением и экземпляром произведения, а также применение авторского права к программному обеспечению неюридическим языком хорошо раскрывается в Anti Copyright FAQ Фёдора Зуева.

Вообще говоря, свобода и открытость — независимые признаки, поэтому мы рассмотрим их по отдельности.

Степени открытости

Нулевая степень

То, что большинство программ используются в двоичной форме и не требуют для работы наличия исходного текста, приводит к возможности распространять двоичные экземпляры программы, никому

не показывая исходные тексты. Это подкрепляется рассуждением: конечного пользователя в первую очередь интересует программа как работающий продукт, а не то, как и почему она работает. Такое программное обеспечение широко распространено на сегодняшнем рынке и может быть точно обозначено как **программное обеспечение с закрытым исходным текстом**.

Минимальная открытость: спецификация форматов и интерфейсов

Если каждую программу сделать «вещью в себе», которая работает одной ей ведомыми способами с одной ей понятными данными, то будет невозможно какое бы то ни было взаимодействие разных программ и их совместное использование. Недостатки очевидны: на каждого автора или производителя программы наваливается необходимость все задачи решать самостоятельно, не имея возможности перепоручить часть работы другой программе; невозможен будет обмен данными между пользователями, если у них нет одной и той же программы (а что будет, если разные версии программы обрабатывают данные слегка по-разному?). Поэтому естественно, что идеальная модель полностью закрытого ПО никогда не была реализована.

С другой стороны, чтобы обеспечить взаимодействие программ, совершенно не требуется целиком открывать их исходный текст. Достаточно чётко описать, каким способом можно обращаться к программе, чтобы добиться определённого результата, и в каком виде программа возвратит этот результат. Такое описание называется **спецификация интерфейса взаимодействия** или **API**.

Чтобы обеспечить обмен данными, необходимо составить набор правил, в соответствии с которыми определённые данные могут быть переведены в форму, доступную для обработки программой, и наоборот, как из этой формы восстановить смысл закодированных в ней данных. Такой набор правил называется **спецификацией формата данных**.

В качестве примера можно привести текстовый документ. Программа представляет документ в виде некоторого файла. Если у нас нет спецификации формата этого файла, то единственный способ получить содержащиеся в нём данные — прибегнуть к помощи этой самой программы, которая неизвестным же для нас образом их извлечёт и отобразит в понятном виде. Теперь представим себе, что это не просто текстовый документ, а государственная бумага, подписанная электронно-цифровой подписью официального лица. Поскольку мы не знаем, как именно программа делает из файла читаемый документ, то даже удостоверившись с помощью электронно-цифровой подписи в подлинности *файла*, мы не можем гарантировать, что видим *тот же самый текст*, который был подписан официальным лицом. Например, на одной из сторон программа настроена иначе и не отобразила примечания... Катастрофические государственные последствия очевидны.

Такое невозможно, если мы располагаем строгой спецификацией формата файла, содержащего документ. Мы всегда, даже без участия какой бы то ни было специальной программы, сможем восстановить из файла содержащийся в нём текст, если нам известен полный набор правил, по которым это делается. При необходимости мы можем создать собственную программу, которая будет работать с файлами в этом формате.

Неполная открытость

Приведённый выше пример показывает, что есть случаи, когда требования пользователя программы шире, чем просто работающая программа: иногда необходимо точно знать устройство и принцип работы программы, например, чтобы иметь возможность исключить недостоверность и различное толкование данных. В такой ситуации автор или производитель вправе предоставить исходные тексты своей программы некоторому закрытому сообществу — клиенту или государственному органу — обычно на условиях неразглашения.

Тем не менее, такая условная открытость не может добавить общественного доверия, она только расширяет круг тех лиц, на авторитетном заявлении которых держится уверенность в тех или иных результатах работы программы.

Другая возможность — публичная демонстрация не полного исходного текста программы, а только отдельных его фрагментов. Принципиально такой подход не добавляет открытости и общественного

доверия.

Максимальная открытость

Программное обеспечение, исходные тексты которого опубликованы или предоставляются любому по первому запросу, называется **программным обеспечением с открытым исходным текстом**, это несколько многословный аналог более лаконичного термина **Open Source Software**.

Но даже если публике представлен исходный текст программы целиком, этого ещё не достаточно, чтобы считать программу полностью открытой, а её работу — полностью прозрачной для пользователя. Для этого ещё необходима уверенность в том, что используемая программа в двоичном виде действительно была получена непосредственно из данного исходного текста. Гарантировать это можно, если не только исходный текст, но и инструментарий, с помощью которого он преобразуется в двоичный вид, будут открыты для публики. В таком случае двоичную программу можно просто воспроизвести (заново скомпилировать), что даёт возможность контроля и аудита программного обеспечения.

Смысл открытых систем

Даже минимальная открытость уже даёт независимость от конкретного производителя ПО, гарантии целостности и однозначности данных, открывает дорогу взаимодействию и совместной работе программ. Бо́льшая открытость ведёт к повышению общественного доверия к программе. Если же программное обеспечение применяется в общественной и государственной сфере, его открытость является гарантией соблюдения принципов гражданского общества[2].

Степени свободы

По российскому законодательству программное обеспечение не может быть запатентовано, поэтому все связанные с ним имущественные и неимущественные отношения в России регулируются только законодательством о программах для ЭВМ (ЗоПЭВМ) и об авторском праве и смежных правах.

Право автора подписывать произведение своим именем и прочие неимущественные авторские права всегда сохраняются за автором и не могут быть переданы другому лицу; в дальнейшем изложении эта сторона авторского права не будет обсуждаться. Имущественные же авторские права касаются разных форм тиражирования произведения, в том числе передачи в эфир, переработки и т. д. В момент создания произведения все имущественные права принадлежат исключительно автору, а всем остальным тиражирование произведения запрещается законом. С имущественными правами автор волен поступать по своему усмотрению, главным образом, передавать их любому лицу или организации. Условия передачи определяются в **авторском договоре**, и могут быть произвольными в рамках допустимых договорных обязательств.

В современном мире появилась новая форма заключения договора: одна из сторон предлагает текст договора в электронном виде, а другая, прочтя его с экрана, принимает условия, нажав на какую-нибудь кнопку. В частности, таким образом производитель или автор программы может распространять вместе со своей программой и авторский договор. Текст договора может быть и просто приложен к программе. Для обозначения такой формы договора часто используют слово «лицензия» — кальку с английского license. Однако в России юридического смысла это слово не имеет, правильно в данном случае говорить об авторском договоре.

Общественная модель распространения произведений предполагает известную степень свободы в тиражировании произведения. Законодательством об авторском праве предусмотрен срок, по прошествии которого произведение переходит в общественное достояние, фактически имущественные права на произведение в этот момент уничтожаются[3]. Однако пока, видимо, ещё ни одна компьютерная программа не достигла преклонного возраста, достаточного для перехода в общественное достояние. Тем не менее, в силах автора сделать своё произведение распространяемым по общественной модели — вопрос только в объёме имущественных прав, которые автор или законный правообладатель готовы передать в общественное достояние.

Нулевая степень свободы

Минимальная степень свободы — использовать программу любым способом и с любой целью. Для российских пользователей ПО эта свобода действительно «нулевая», в том смысле, что она присутствует всегда, что бы ни говорилось в авторском договоре. По российскому законодательству обладатель имущественных авторских прав волен ограничивать право пользователей на *тиражирование* своего произведения, однако у него нет никаких прав каким бы то ни было образом ограничивать владельца экземпляра произведения в *использовании* программы.

Нулевая свобода — это некоторая гарантия личной свободы пользователя от посягательств производителей ПО, но она ещё не является свободой для самого произведения.

Свобода распространения

Первая свобода собственно для произведения — свобода распространения. Для автора она означает, с одной стороны, снятие всяких ограничений на тиражирование произведения (и потенциально — его более широкое распространение); с другой стороны, фактический отказ от получения вознаграждения за передачу имущественных прав.

Известно огромное число примеров свободно распространяемых программ. Однако свобода распространения никак не предполагает, что должны быть доступны исходные тексты программы, вполне можно распространять программу только в двоичном виде. Зачастую именно так и происходит: производитель ПО стремится максимально широко распространить свою программу среди пользователей, даже отказываясь от платы за экземпляры, при этом не делает свою программу открытой. Отпущенная «в свободное плавание» программа ещё не может считаться вполне общественным достоянием, если её исходный текст — который и является собственно произведением — недоступен публике.

Свобода модификации

Одно из имущественных авторских прав, оговорённых в законодательстве, — право на переработку произведения. Это право приобретает очень большое значение, если в качестве произведения рассматривается компьютерная программа. В отличие от литературных и прочих текстов, программа должна *работать*. Это значит, что в ней необходимо исправлять ошибки, приспосабливать для работы в новых системах и т. п. Если исходный текст программы открыт, то в силах любого компетентного человека внести необходимые исправления. Для общественности эта возможность имеет значение, если есть свобода распространения модифицированных версий программы. Ведь автор не всегда доступен, не всегда заинтересован и имеет возможность вносить исправления или переносить программу на другие системы.

Если в авторском договоре, сопровождающем программу, автор передаёт право модифицировать и распространять модифицированные версии программы, то такую программу можно с полным правом отнести к **свободному ПО** (Free Software), в том понимании, которое было сформулировано Ричардом Столлманом[4]. Естественно, свобода модификации предполагает, что открыт исходный текст программы.

На сегодняшний день распространено довольно много типовых форм авторских договоров для распространения свободных программ. Те из них, которые удовлетворяют требованиям к свободному ПО, перечислены на сайте Фонда свободного программного обеспечения. Названную свободу модификации в полной мере реализует лицензия BSD[5].

Максимальная свобода

Может ли общественность злоупотребить данной ей свободой в обращении с программой? Вполне. Самое страшное злоупотребление, которое можно себе представить — модифицировав программу, запретить свободное распространение модифицированной версии и даже закрыть её исходный текст. Таким образом программа может быть изъята из свободного обращения и переведена в частное владение. В соответствии с лицензией BSD такое поведение является абсолютно законным и

предусмотренным: объём передаваемых авторских прав достаточен для таких действий.

Однако не всегда автор, отдавая своё произведение общественности, согласится на такое его использование. Чтобы избежать «закрепощения» своей программы, автор может воспользоваться тем же самым законным инструментом — авторским договором. Главный прототип такого договора (да и вообще главный прототип всех свободных лицензий) — GPL, общественная лицензия GNU (GNU **G**eneral **P**ublic **L**icense), впервые сформулированная тем же Ричардом Столлманом. Эта лицензия, помимо предоставления всех необходимых свобод, включает условие **copyleft**: никто не имеет права, сделав модифицированную версию свободной программы, распространять её, не соблюдая *всех* принципов свободного ПО, ограничивая тем самым права *других пользователей* по отношению к программе. Говоря короче, запрещает модификацию свободной программы делать несвободной.

Любой авторский договор, включающий такое условие, может быть назван «copyleft». Это игра слов с умыслом: по-английски авторское право называется «copyright», буквально «копироватьправо», а «copyleft», соответственно, «копироватьлево». Действительно, условие «copyleft» прямо противоположно по смыслу авторскому праву: авторское право призвано ограничить пользователя в копировании и распространении копий продукта, а «авторское лево», наоборот, строго запрещает его ограничивать. «Авторское лево» реализует идею о том, что интеллектуальные достижения человека не могут и не должны находиться в чьей-то частной собственности, и сохраняет свободу наилучшим способом — пользуясь теми самыми механизмами ограничения, которые предоставлены законодательством об авторском праве.

Несмотря на то, что лицензия BSD с юридической точки зрения передаёт бОльший объём прав, в нашем изложении она оказалась ниже на шкале свободы именно по той причине, что она даёт меньше гарантий свободы программе.

Смысл свободного ПО

Причины, по которым люди и организации выбирают свободную модель распространения своего программного обеспечения, очень разнообразны и индивидуальны. Одна из важнейших и самых общих причин для авторов — это стремление к свободе интеллектуальной деятельности. Яркое выражение этого стремления и опасений за интеллектуальную свободу можно найти в очень коротком антиутопическом рассказе Ричарда Столлмена «Право читать». Однако здесь не всё универсально, и в общем случае существуют две противоположные стратегии поведения по отношению к полученным интеллектуальным результатам: спрятать подальше или распространить пошире, предполагающие, соответственно, частную и общественную модели распространения. Научные и университетские традиции склоняются в пользу второй стратегии, но стоит науке подойти достаточно близко к технологии — как нередко актуализируется частная модель, возникают патенты и закрытые результаты.

Для индивидуальных авторов на некоммерческом поприще общественная модель — это хорошая возможность для самореализации: на свободно распространяемых программах всегда стоят имена их авторов; здесь же и возможности для социализации — вокруг удачных и востребованных свободных программ всегда складывается сообщество разработчиков и пользователей.

Общественная модель распространения ПО представляет хорошие возможности и для бизнеса. К началу XXI века уже стало традиционным строить бизнес в области программного обеспечения из двух компонент: собственно разработки и торговли лицензиями (правом производить ограниченное количество экземпляров), причём вторая часть значительно прибыльнее первой, поскольку расходы не увеличиваются пропорционально числу проданных лицензий. На рынке свободного ПО вторая компонента просто отпадает — производить экземпляры разрешено всем в неограниченных количествах, а первая — собственно разработка — остаётся в полном объёме. В результате преимущество получают те, кто делает реальную разработку — это открывает дорогу конкуренции и инновациям. В целом, на рынке свободного ПО значительно ниже финансовый порог вхождения при достаточно высоком интеллектуальном пороге.

Свобода и открытость Linux

Если быть точным, слово Linux обозначает только **ядро** операционной системы, хоть и самый главный, но всё же только один из компонентов сложнейшей системы, какой является операционная среда, в которой работает пользователь. Всё остальное — утилиты, графическая среда, пользовательские приложения, — это независимые друг от друга *свободные* программы, над каждой из которых работает своя команда разработчиков и пользователей. Само по себе ядро Linux тоже является совершенно независимой свободной программой. Современные дистрибутивы, включающие в себя все эти компоненты, тоже обозначают словом Linux, но это уже его расширительное понимание.

Сформировать целостную операционную систему из независимо разрабатываемых компонентов может только одно — соблюдение открытых стандартов на форматы и интерфейсы, что позволяет организовать взаимодействие программ и обмен данными. В этом отношении Linux наследует традициям открытого проектирования семейства операционных систем, обозначаемых общим и несколько собирательным именем Unix. Исторически в Unix сложился фактический стандарт на интерфейс операционной системы (**API**), который впоследствии был даже зафиксирован в виде официального стандарта под именем POSIX. Ядро Linux было написано финским студентом, Линусом Торвальдсом, как независимая и свободная реализация Unix API с открытым исходным текстом.

К моменту первой публикации Linux в 1991 году уже существовало значительное количество свободных программ для Unix, самые важные разрабатывались в рамках проекта GNU Ричарда Столлмана. Однако не существовало свободного Unix-совместимого ядра. Сложилось так, что ядро Linux стало тем связующим звеном, которое позволило собрать вместе ПО, которое уже было свободным, и создать первую полностью *свободную* и *открытую* операционную систему. Это одновременно дало толчок к появлению нового свободного ПО. Поэтому сегодня свободное и открытое ПО в первую очередь ассоциируются с Linux, а Linux — со свободным ПО.

Но это совершенно не означает, что свободного и открытого нет за пределами Linux. Сам смысл свободы и открытости гарантирует то, что они никогда не окажутся неотделимы от конкретной ОС, или любого другого имени собственного. Свободное ПО пишется для самых разнообразных платформ, в том числе для Windows.

Открытость стандартов гарантирует, что инструментарий может быть перенесён на любые платформы, а следствие — многоплатформенность приложений. Пример тому — проект cygwin, реализующий инструментарий Unix на платформе Windows.

1Нужно отметить, что это относится только к имущественной стороне авторского права, неимущественные же права, в частности, право на имя, на сохранение целостности произведения и т. п., в конечном итоге, право на репутацию автора, всегда сохраняются за автором, вне зависимости от модели распространения. Такие права не могут быть проданы или переданы другому лицу.

2Если и делать государственный документооборот электронным, то этот механизм должен быть полностью открытым, так как он служит ограничению прав. Это полностью аналогично тому, как должны быть полностью открыты для публики тексты законов.

3На сегодняшний день по российскому законодательству этот срок весьма долог — 70 лет с момента смерти автора.

4Имеются в виду 4 свободы Столлмана, декларированные в манифесте созданного им Фонда свободного программного обеспечения (Free Software Foundation).

5BSD — **B**erkeley **S**oftware **D**istribution, пакет совместимого с UNIX программного обеспечения, разработанный в университете Беркли и распространявшийся свободно.

GNU без Linux

К 1990 году в рамках проекта GNU были разработаны и постоянно развивались свободные программы, составляющие основной инструментарий для разработки программ на языке Си: текстовый редактор Emacs, компилятор языка Си gcc, отладчик программ gdb, командная оболочка Bash, библиотека важнейших функций для программ на Си libc. Все эти программы были написаны для операционных систем, похожих на UNIX. Это означает, что в них использовался стандартный для UNIX механизм запроса ресурсов компьютера, необходимых программе — **системные вызовы**, которые исполняются **ядром** операционной системы. При помощи системных вызовов программы получают доступ к оперативной памяти, файловой системе, устройствам ввода и вывода. Благодаря тому, что системные вызовы выглядели более-менее стандартно во всех реализациях UNIX, программы GNU могли работать (с минимальными изменениями или вообще без изменений) в любой UNIX-подобной операционной системе.

С помощью имевшихся инструментов GNU можно было бы писать программы на Си, пользуясь *только* свободными программными продуктами, однако свободного UNIX-совместимого **ядра**, на основе которого могли бы работать все эти инструменты, не существовало. В такой ситуации разработчики GNU вынуждены были использовать одну из патентованных реализаций UNIX, т. е. вынуждены были следовать принятым в этих операционных системах архитектурным решениям и технологиям и основывать на них свои собственные разработки. Идеал Столлмана о научной разработке ПО, свободной от решений, движимых коммерческими целями, был недоступен, пока в основе свободной разработки лежало патентованное UNIX-совместимое **ядро**, исходные тексты которого оставались тайной для разработчиков.

Linux — ядро

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. Прототипом для будущего ядра стала операционная система MINIX: совместимая с UNIX операционная система для персональных компьютеров, которая загружалась с дискет и умещалась в очень ограниченной в те времена памяти персонального компьютера. MINIX был создан Энди Танненбаумом в качестве учебной операционной системы, *демонстрирующей* архитектуру и возможности UNIX, но непригодной для полноценной работы с точки зрения программиста. Так же, MINIX можно было использовать только в некоммерческих целях. Именно полноценное ядро для своего ПК и хотел сделать Линус Торваьдс. Название для своего ядра он соорудил из собственного имени, заменив последнюю букву и сделав его похожим на анаграмму слова UNIX.

Совместимость с UNIX в этот 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 он 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 GPL. Благодаря этому и получавшей всё большее распространение сети Internet очень многие получили возможность самостоятельно компилировать и тестировать это ядро,

участвовать в обсуждении и исправлении ошибок, и 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 свидетельствует 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 бо́льшего сообщества, чем во времена до версии 0.1, изменилась и роль самого Линуса Торвальдса, который теперь не главный разработчик, но главный авторитет, который традиционно оценивает исходные тексты, которые должны быть включены в ядро и даёт своё добро на их включение. Тем не менее, общая модель свободной разработки сообществом сохраняется. В настоящее время параллельно всегда разрабатывается два варианта ядра. Стабильная версия, считающаяся достаточно надёжной и пригодной для пользователей, её номер заканчивается на чётное число, например, «2.4». Номер соответствующей экспериментальной версии ядра оканчивается на нечётное число — «2.5». Экспериментальная версия адресована в первую очередь разработчикам ядра, тестирующим новые возможности.

GNU и Linux

Однако как нельзя сделать операционную систему без ядра, так и ядро будет бесполезно без утилит, которые использовали бы его возможности. Благодаря проекту GNU Линус Торвальдс сразу имел возможность использовать в Linux свободные утилиты: Bash, компилятор gcc, tar, gzip и многие другие уже известные и широко используемые приложения, которые могли работать с его UNIX-совместимым ядром. Так Linux сразу попал в хорошее окружение и в сочетании с утилитами GNU представлял собой очень интересную среду для разработчиков программного обеспечения даже на самой ранней стадии своего развития.

Принципиальным шагом вперёд было именно то, что из ядра Linux и утилит и приложений GNU *впервые* стало возможно сделать полностью свободную *операционную систему*, т. е. работать с компьютером и, более того, разрабатывать новое программное обеспечение, пользуясь только свободным программным обеспечением. Идеал полностью некоммерческой разработки Столлмана теперь мог быть реализован в жизни.

Однако появление теоретической возможности воплощения идеала не означало его немедленной практической реализации. Совместимость Linux и утилит GNU была обусловлена тем, что и то, и другое писалось с ориентацией на одни и те же стандарты и практику. Однако, в рамках этой практики (множество различных UNIX-систем) оставался большой простор для несовместимости и различных решений. Поэтому на начальном этапе разработки ядра каждое заработавшее под Linux приложение GNU было для Линуса очередным достижением: первыми стали Bash и gcc. Таким образом, сочетание GNU и Linux было возможностью создать свободную операционную систему, но само по себе ещё не составляло такой системы, потому что Linux и различные утилиты GNU оставались разрозненными программными продуктами, которые писали разные люди, не всегда принимая в расчёт то, что делают другие. Основное же качество системы — согласованность её компонентов.

Возникновение дистрибутивов

После определённого периода разработки под Linux уже стабильно работал ряд важнейших утилит GNU. Скомпилированное ядро Linux с небольшим комплектом скомпилированных уже в Linux утилит GNU составляло набор инструментов для разработчика программного обеспечения, желающего использовать свободную операционную систему на своём персональном компьютере. В таком виде Linux уже не только годился для разработки Linux, но и представлял собой операционную систему, в которой можно было уже выполнять какие-то прикладные задачи. Конечно, первое, чем можно было заниматься в Linux — писать программы на Си.

Первоначально, чтобы получить компьютер с работающей системой Linux, разработчики пользовались специальными комплектами дискет со скомпилированным ядром Linux и утилитами: с этих дискет можно было загрузить Linux и работать в нём. Однако это не слишком удобно, когда нужно работать в Linux постоянно, да и объём дискет накладывал очень сильные ограничения на дальнейшее расширение

системы и включение новых утилит.

Когда задача получить компьютер с постоянно работающей на нём системой Linux стала востребованной и довольно распространённой, разработчики в хельсинкском и техасском университетах создают собственные наборы дискет, с которых скомпилированное ядро и основные утилиты можно записать на жёсткий диск, после чего загружать операционную систему прямо с него. Эти наборы дискет — первые прототипы современных **дистрибутивов** Linux — комплекты программного обеспечения, на основе которых можно получить работающую операционную систему на своём компьютере. Нужно отметить, что в дистрибутив Linux с самого начала входили программные продукты GNU. На самом деле, всякий раз, когда говорится «операционная система Linux», подразумевается «ядро Linux и утилиты GNU». Фонд свободного ПО даже рекомендует называть это операционной системой GNU/Linux.

Однако скопировать все нужные программы на жёсткий диск ещё недостаточно, чтобы получить подходящую для нужд пользователя операционную среду (пусть даже это очень профессиональный пользователь). Поэтому первые наборы дискет можно только условно назвать дистрибутивами. Чтобы получить работающую операционную систему, тредуются какие-то специальные средства установки и настройки программного обеспечения. Именно наличие таких средств и отличает современные дистрибутивы Linux. Другое важнейшая задача дистрибутива — регулярное обновление. Программное обеспечение, особенно свободное, — одна из самых быстро развивающихся областей, поэтому мало один раз установить Linux, нужно ещё регулярно его обновлять. Первым дистрибутивом в современном понимании, получившим широкое распространение, стал Slackware, созданный Патриком Фолькердингом (кстати, этот дистрибутив сохранился и до наших дней). Он был широко известен пользователям Linux уже к 1994 году.

Несмотря на то, что с появлением первых дистрибутивов установка Linux уже не требует самостоятельной компиляции всех программ из исходных текстов, использование Linux оставалось уделом разработчиков: пользователь этой операционной системы в тот период её развития мог заниматься почти исключительно программированием. По крайней мере, чтобы решать в ней другие повседневные прикладные задачи (например, чтение электронной почты, написание статей и т. п.), он должен был сначала некоторое время позаниматься программированием и даже разработкой самой системы Linux, чтобы создать для себя соответствующие прикладные программы или заставить их работать в Linux.

Однако разработчики — тоже люди, которые пишут и электронные письма, и статьи, и даже рисуют картинки. Всё программное обеспечение для Linux было открытым, поэтому вскоре стало появляться всё больше прикладных программ для Linux, которые использовались всё большим сообществом, отчего становились надёжнее и получали всё новую функциональность. В конце концов возникает идея, что из Linux и GNU-приложений для Linux целенаправленными усилиями небольшой группы разработчиков можно делать целостные операционные системы, подходящие для очень широкого круга пользователей и продавать эти системы пользователям за деньги как аналог и альтернативу существующим патентованным операционным системам.

Выгода операционной системы, целиком состоящей из свободного программного обеспечения, очевидна — собирающие эту систему не должны никому платить за входящие в неё программы. Более того, дальнейшая разработка и обновление имеющихся программ ведётся сообществом разработчиков также совершенно бесплатно, не нужно платить сотрудникам, которые занимались бы этим. В итоге затраты фирмы, собирающей **дистрибутив** Linux для пользователя, ограничиваются оплатой программистов, интегрирующих разрозненные приложения в систему и пишущих программы для стандартизации процедур установки и настройки системы, чтобы облегчить эти задачи неподготовленному пользователю, а также затратами на само издание получившегося дистрибутива. Для конечного покупателя это означает принципиальное снижение цены на операционную систему.

Первой успешной компанией, работающей по такой схеме, стала RedHat, появившаяся в 1995 году. RedHat адресовала свои разработки не только программистам профессионалам, но и обыкновенным пользователям и системным администраторам, для которых компьютер — в первую очередь офисное рабочее место или рабочий сервер. Ориентируясь на уже существующие на рынке предложения для такого класса пользователей, RedHat всегда уделял большое внимание разработке приложений с

графическим интерфейсом для выполнения типичных задач по настройке и администрированию системы. Бизнес RedHat развивался довольно успешно, в 1999 году эта компания акционировалась — сразу после выпуска акции росли в цене очень энергично, однако потом ажиотаж улёгся. В настоящее время доля RedHat на рынке серверов и рабочих станций Linux очень велика. Благодаря RedHat в сообществе пользователей Linux очень широкое распространение получил формат пакетов RPM.

Практически одновременно с RedHat появился проект Debian. Его задача была примерно той же — сделать целостный дистрибутив Linux и свободного программного обеспечения GNU[1], однако этот проект был задуман как принципиально некоммерческий, проводимый в жизнь сообществом разработчиков, нормы взаимодействия в котором полностью соответствовали бы идеалам свободного ПО. Сообщество разработчиков Debian — международное, участники которого взаимодействуют через Internet, а нормы взаимодействия между ними определяются специальными документами — **полиси** (policy).

Сообщество разработчиков не извлекает никакой прибыли от продажи Debian, его версии распространяются свободно, доступны в Интернет, могут распространяться и на твёрдых носителях (CD, DVD), но и в этом случае их цена редко сильно превышает стоимость носителя и наценку, окупающую затраты на издание. Первоначально разработка Debian спонсировалась Фондом свободного программного обеспечения. Адресатами дистрибутивов Debian всегда в первую очередь были профессиональные пользователи, так или иначе связанные с академической разработкой программного обеспечения, которые готовы читать документацию и собственными руками организовать нужный **профиль** системы, соотвествующий именно их задачам. Ориентация на такую аудиторию предопределила некоторые тенденции развития Debian: в нём никогда не было обилия «простых» графических средств настройки среды, всевозможных **мастеров**, однако всегда уделялось много внимания средствам последовательной и единообразной интеграции программного обеспечения в единую систему. Именно в Debian появился менеджер пакетов (APT). В настоящее время Debian — самый популярный дистрибутив Linux среди пользователей, являющихся профессионалами в области информационных технологий.

Всякий раз, когда свободное программное обеспечение оказывается востребованным, немедленно возникает множество альтернативных решений — так произошло и с дистрибутивами Linux. После 1995 года возникло (и продолжает возникать) огромное количество коммерческих компаний и свободных сообществ, которые ставят своей задачей подготовку и выпуск дистрибутивов Linux. У каждого из них — свои особенности, своя целевая аудитория, свои приоритеты. К настоящему времени на рынке дистрибутивов выделилось несколько лидеров, которые предлагают более или менее универсальные решения и наиболее широко известны и используются. Помимо уже названных RedHat и Debian следует назвать в ряду дистрибутивов, ориентированных на рядового пользователя, немецкий SuSE и французский Mandrake, среди адресованных специалистам — Gentoo. Но помимо «крупных» игроков на рынке дистрибутивов есть горадо большее количество менее распространённых дистрибутивов. Теперь перед пользователем, желающим установить Linux, встаёт вопрос выбора дистрибутива. Критерии выбора — и задачи, которые предполагается решать с помощью Linux, и уровень подготовки пользователя, и технологии, и предстоящие контакты с тем сообществом, которое занимается разработкой дистрибутива.

История Linux в России

Получилось так, что в международном сообществе разработчиков, начинавших и продолжавших развивать Linux, все в той или иной степени могли объясниться по-английски. Это и неудивительно, поскольку исторически английский оказался языком компьютерной науки и операционной системы UNIX, глобальной сети Internet, программирования. В международном сообществе разработчиков программного обеспечения английский выполнял и выполняет роль, подобную латыни в научном сообществе средневековой Европы. Но если Linux предполагается использовать не только для программирования и общения с программистами, но и для повседневных задач, необходима локализация — т. е. возможность общаться с комьютером и при помощи компьютера и на других языках.

Локализация — комплексный процесс, затрагивающий самые разные стороны системы. Для

полноценной поддержки того или иного языка в системе необходимо обеспечить возможность ввода на этом языке (поддержка раскладок клавиатуры и кодировок), вывода (экранных шрифтов), печати, а затем уже необходимо переводить интерфейс различных приложений на данный язык, разрабатывать средства подготовки электронных и бумажных публикаций на этом языке и т. д. В этой лекции мы кратко рассмотрим только историю локализации Linux в России для русского языка, т. е. русификации Linux.

Первой компанией, поставившей своей целью выпуск дистрибутивов Linux для русскоговорящих пользователей, была УрбанСофт, открытая в Петербурге в 1992 году. Весь её бизнес состоял в выпуске и продаже CD-дисков с дистрибутивами свободного программного обеспечения. В первую очередь это были дистрибутивы RedHat, а также Debian, в которые включались разработанные силами УрбанСофт пакеты для русификации.

Несколько позже в Москве IPLabs Linux Team выпускает Linux Mandrake Russian Edition — модифицированный (чтобы соответствовать нуждам русского пользователя) вариант дистрибутива Mandrake Linux. Впоследствии эта команда начинает выпускать дистрибутивы, которые отличаются от Mandrake уже не только наличием пакетов для русификации, но и другими принципиальными возможностями. В конце концов команда разработчиков создаёт фирму ALT Linux и начинает выпускать дистрибутивы под маркой ALT Linux.

Также появляется компания ASPLinux, целью которой является выпуск RedHat с модификациями для поддержки русского языка, название продукта совпадает с именем компании.

Все перечисленные Российские производители дистрибутивов Linux существуют и по сей день, и продолжают с большей или меньшей активностью выпускать дистрибутивы.

Зачем нужна командная строка

Интерпретатор командной строки (shell)

Проницательный читатель, несомненно, заметит, что как только речь заходит об устройстве Linux и более или менее профессиональной работе с этой ОС, в примерах немедленно возникает и начинает доминировать командная строка. Из чего несложно сделать вывод, что это главный (и стандартный) интерфейс управления системой в Linux. Тот же проницательный читатель наверняка задастся вопросом — а кто же выполняет команды, введённые в командной строке? Ответ «система» окажется неправильным: в Linux нет отдельного объекта под именем «система». Система — она на то и система, чтобы состоять из многочисленных компонентов, взаимодействующих друг с другом.

Правильный ответ, как водится, оказывается более сложным. Операционная система нужна в частности для того, чтобы программы можно было писать, не думая о подробностях устройства компьютера и его деталей, начиная от процессора и жёсткого диска (скажем, на уровне «открыть файл», а не последовательности команд перемещения головки жёсткого диска). Операционная система управляет оборудованием сама, а программам предоставляет «язык» довольно высокого уровня абстракции, покрывающий все их необходимости, т. н. **API**[1]. Но для команд пользователя такой язык не годится, поскольку он всё равно слишком низкоуровневый (для решения даже самой простой задачи пользователя необходимо выполнить несколько таких операций), да и воспользоваться им можно, только написав программу (чаще всего — на языке Си). Возникает необходимость выдумать для пользователя другой — более высокоуровневый и более удобный — язык управления системой. Все команды, которые можно ввести в командной строке, сформулированы именно на этом языке.

Из чего проницательному читателю несложно заключить, что *обрабатывать* эти команды, переводя их на язык операционной системы, должна тоже какая-нибудь специальная программа, и именно с ней ведёт диалог пользователь, работая с командной строкой. Так оно и есть: программа эта называется **интерпретатор командной строки** или **командная оболочка** («shell»). «Оболочкой» она названа как раз потому, что всё управление системой идёт как бы «изнутри» неё: пользователь общается с нею на удобном ему языке (с помощью текстовой командной строки), а она общается с другими частями системы на удобном *им* языке (вызывая запрограммированные функции).

Конечно, командных интерпретаторов в Linux несколько. Самый простой из них, появившийся в ранних версиях UNIX, назывался sh, или «Bourne Shell» — по имени автора, Стивена Борна (Stephen Bourne). Со временем его — везде, где только можно — заменили на более мощный, bash, «Bourne Again Shell»[2]. bash превосходит sh во всём, особенно в возможностях *редактирования* командной строки. Помимо sh и bash в системе может быть установлен «The Z Shell», zsh, *самый* мощный на сегодняшний день командный интерпретатор (шутка ли, 22 тысячи строк документации), или tcsh, обновлённая и тоже очень мощная версия старой оболочки «C Shell», синтаксис команд которой похож на язык программирования Си.

Синтаксис командной строки

Итак, что же представляет собой этот более удобный для пользователя язык? Больше всего общение на этом языке напоминает письменный диалог с системой — поочерёдный обмен текстами. Высказывание пользователя на этом языке — это команда, каждая команда — это отдельная строка. Пока не нажат *enter*, строку можно редактировать, затем она передаётся оболочке. Оболочка *разбирает* полученную команду — переводит её на язык системных объектов и функций, после чего отправляет системе на выполнение.

Результат выполнения очень многих команд также представляет собой текст, выдаваемый в качестве «ответа» пользователю. Хотя это и не обязательно — команда может выполнять свою работу совершенно молчаливо. Кроме того, если в процессе выполнения команды возникли какие-то особые обстоятельства (например, ошибка), оболочка включит в ответ пользователю **диагностические сообщения**.

Команда и параметры

Простейшая команда состоит из одного «слова», например, команда cal, которая выводит календарь на текущий месяц.

```
[methody@localhost methody]$ cal
   Декабря 2005
Вс Пн Вт Ср Чт Пт Сб
            1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24

[methody@localhost methody]$
```

Пример 1. Команда cal

А если нужно посмотреть календарь на будущий месяц? Верно, не следует для этого изобретать отдельную команду[3], cal вполне справится с этой задачей, только её поведение нужно немного модифицировать:

```
[methody@localhost methody]$ cal 1 2006
   Января 2006
Вс Пн Вт Ср Чт Пт Сб
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

Пример 2. Команда cal с параметрами

Выходит, команда cal 1 2006 состоит как минимум из двух частей — собственно команды cal и «всего остального». Это остальное, что следует за командой, называют **параметрами** (или *аргументами*), причём они вводятся для того, чтобы изменить поведение команды. В большинстве случаев при разборе командной строки *первое* слово считается именем команды, а остальные — её параметрами.

Слова

При разборе командной строки shell использует понятие **разделитель** (delimiter). Разделитель — это символ, разделяющий слова; таким образом командная строка — это последовательность *слов* (которые имеют значение) и *разделителей* (которые значения не имеют). Для shell разделителями являются символ пробела, символ табуляции и символ перевода строки. Количество разделителей между двумя соседними словами значения не имеет.

Для того, чтобы разделитель попал *внутрь* слова (и получившаяся строка с разделителем передалась как *один* параметр), всю нужную подстроку надо окружить одинарными или двойными кавычками:

```
[methody@localhost methody]$ echo One      Two      Three
 One Two Three
[methody@localhost methody]$ echo One      "Two      Three"
 One Two      Three
[methody@localhost methody]$ echo 'One
>
> Ой. И что дальше?
> А, кавычки забыл!'

 One

 Ой. И что дальше?
 А, кавычки забыл!
[methody@localhost methody]$
```

Пример 3. Закавычивание в командной строке

Всё сказанное означает, что у команды столько параметров, сколько *слов* с точки зрения shell следует за ней в командной строке. Первое слово в тройке передаётся команде как первый параметр, второе — как второй и т. д. В первом случае команде echo было передано *три* параметра — «One», «Two» и «Three». Она их и вывела, разделяя пробелом. Во втором случае параметров было *два*: «One» и «Two Three». В результате эти *два* параметра были также выведены через пробел. В третьем случае параметр был всего *один* — от открывающего апострофа «'One» до закрывающего «...забыл!'». Всё время ввода bash услужливо выдавал подсказку «> » — в знак того, что набор командной строки продолжается, но в режиме ввода содержимого кавычек.

Ключи

Для решения разных задач одни и те же действия необходимо выполнять слегка по-разному. Например, для синхронизации работ в разных точках земного шара лучше использовать единое для всех время (по Гринвичу), а для организации собственного рабочего дня — местное время (с учётом сдвига по часовому поясу и разницы зимнего и летнего времени). И то, и другое время показывает команда date, только для работы по Гринвичу ей нужен дополнительный параметр «-u» (он же «--universal»).

```
[methody@localhost methody]$ date
Вск Сен 19 23:01:17 MSD 2004
[methody@localhost methody]$ date -u
Вск Сен 19 19:01:19 UTC 2004
```

Пример 4. Команда date с ключом

Такого рода параметры называются *модификаторами выполнения* или **ключами** (options)[4]. Ключ принадлежит данной конкретной команде и сам по себе смысла не имеет, чем отличается от прочих параметров (например, имён файлов, чисел), которые имеют *собственный* смысл, не зависящий ни от какой команды. Каждая команда может распознавать некоторый набор ключей и соответственно изменить своё поведение. В результате «один и тот же» ключ, например, «-s» может значить для разных команд совершенно разные вещи.

Для формата ключей нет жёсткого стандарта, однако существуют договорённости, нарушать которые в наше время уже неприлично. Во-первых, если параметр начинается на «-», это — **однобуквенный**

ключ. За «-», как правило, следует один символ, чаще всего — буква, обозначающая действие или свойство, которое этот ключ придаёт команде. Так проще отличать ключи от других параметров — и пользователю при наборе командной строки, и программисту, автору команды.

Во-вторых, желательно, чтобы имя ключа было *значащим* — как правило, это первая буква названия действия или свойства, обозначаемого ключом. Например, ключ «-a» в man и who происходит от слова «**A**ll» (всё), и изменяет работу этих команд так, что они начинают показывать информацию, о которой обычно умалчивают. А в командах cal и who смысл ключа «-m» — разный:

```
[methody@localhost methody]$ who -m
 methody  tty1         Sep 20 13:56 (localhost)
[methody@localhost methody]$ cal -m
     Сентября 2004
 Пн Вт Ср Чт Пт Сб Вс
        1  2  3  4  5
  6  7  8  9 10 11 12
 13 14 15 16 17 18 19
 20 21 22 23 24 25 26
 27 28 29 30
```

Пример 5. Использование ключа «-m» в разных командах

Для who ключ «-m» означает «**M**e», то есть «Я», и в результате who работает похоже на whoami[5]. А для cal ключ «-m» — это команда выдать календарь, считая первым днём *понедельник* («**M**onday»), как это принято в России.

Свойство ключа быть, с одной стороны, предельно коротким, а с другой стороны — информативным, называется **аббревиативностью**. Не только ключи, но и имена наиболее распространённых команд Linux обладают этим свойством.

В-третьих, иногда ключ изменяет поведение команды таким образом, что меняется и толкование параметра, следующего в командной строке за этим ключом. Выглядит это так, будто ключ *сам* получает параметр, поэтому ключи такого вида называются **параметрическими**. Как правило, их параметры — имена файлов различного применения, числовые характеристики и прочие *значения*, которые нужно передать команде.

```
[methody@localhost methody]$ date -s 00:00
date: невозможно установить дату: Operation not permitted
Чтв Дек 29 00:00:00 MSK 2005
[methody@localhost methody]$ date
Чтв Дек 29 14:17:38 MSK 2005
```

Пример 6. Использование date -s

Ключ «-s» команды date позволяет установить системное время в то значение, которое указывается в качестве параметра этого ключа. Однако в данном примере эта операция не удалась, о чём свидетельствует выведенное **сообщение об ошибке**. Для изменения системных часов требуются привилегии системного администратора, а в нашем примере эта команда выполнялась от имени обычного пользователя. Тем не менее, date всё равно отобразил время, которое нужно было установить, хотя системные часы и остались не изменёнными.

Аббревиативность ключей трудно соблюсти, когда их у команды *слишком* много. Некоторые буквы латинского алфавита (например, «s» или «o») используются очень часто, и могли бы служить сокращением сразу нескольких команд, а некоторые (например, «z») — редко, под них и название-то

осмысленное трудно придумать. На такой случай существует другой, **полнословный** формат: ключ начинается на *два* знака «-», за которыми следует *полное* имя обозначаемой им сущности. Таков, например, ключ «--help» (аналог «-h»):

```
[methody@localhost methody]$ head --help
Использование: head [КЛЮЧ]... [ФАЙЛ]...
Печатает первые 10 строк каждого ФАЙЛА на стандартный вывод.
Если задано несколько ФАЙЛОВ, сначала печатает заголовок с именем файла.
Если ФАЙЛ не задан или задан как -, читает стандартный ввод.

Аргументы, обязательные для длинных ключей, обязательны и для коротких.
 -c, --bytes=[-]N        напечатать первые N байт каждого файла;
                       если перед N стоит `-', напечатать все, кроме N
                       последних байт каждого файла
 -n, --lines=[-]N        напечатать первые N строк каждого файла, а не 10;
                       если перед N стоит `-', напечатать все, кроме N
                       последних строк каждого файла
 -q, --quiet, --silent    не печатать заголовки с именами файлов
 -v, --verbose           всегда печатать заголовки с именами файлов
    --help     показать эту справку и выйти
    --version  показать информацию о версии и выйти


N может иметь суффикс-множитель: b 512, k 1024, m 1024*1024.

Об ошибках сообщайте по адресу <bug-coreutils@gnu.org>.
```

Пример 7. Ключ–help

Видно, что некоторые ключи head имеют и однобуквенный, и полнословный формат, а некоторые — только полнословный. Так обычно и бывает: часто используемые ключи имеют аббревиатуру, а редкие — нет. *Значения* параметрических *полнословных* ключей принято передавать не следующим параметром командной строки, а с помощью конструкции «=*значение*» непосредственно после ключа.

В-четвёртых, есть некоторые менее жёсткие, но популярные договорённости о *значении* ключей. Ключ «-h» («**H**elp») обычно (но, увы, не всегда) заставляет команды выдать *краткую справку*. Наконец, бывает необходимо передать команде *параметр, а не ключ*, начинающийся с «-». Для этого нужно использовать ключ «--»:

```
[methody@localhost methody]$ head -1 -filename-with-
head: invalid option -- f
Попробуйте `head --help' для получения более подробного описания.
[methody@localhost methody]$ head -1 -- -filename-with-
Первая строка файла -filename-with-
```

Пример 8. Параметр–не ключ, начинающийся на «-»

Ключ «--» (первый «-» — признак ключа, второй — сам ключ) обычно запрещает команде интерпретировать все последующие параметры командной строки как ключи, независимо от того, начинаются ли они на «-» или нет. Только после «--» head согласилась с тем, что -filename-with- — это имя файла.

Синопсис

Из всего вышесказанного ясно, что для каждой команды существует свой собственный небольшой язык — его составляют те ключи и обязательные и необязательные параметры, которые принимает и интерпретирует команда. Чтобы окинуть возможности команды одним взглядом, в различной документации по Linux приводится **синопсис** — сжатое перечисление всех возможных параметров команды. Выглядит это примерно так:

```
cal [-smjy13] [[month] year]
```

Пример 9. Синопсис для команды cal

В синопсисе даётся *формализованное* описание способов использования объекта (с данном случае — того, как и с какими параметрами запускать команду cal). Параметры перечисляются в том же порядке, в котором их нужно вводить в командной строке, необязательные параметры, как правило, даются в квадратных скобках, обязательные — вообще без скобок, а ключи для компактности собираются в один параметр, в котором каждая буква — это отдельный ключ. Приведённый пример читается так: у команды cal нет обязательных параметров, есть (необязательные) ключи (-s, -m и т. д.), и необязательный параметр year, перед которым может присутствовать необязательный же month (но не может быть указан month без year).

Откуда берутся команды

Дочитав предыдущий раздел, проницательный читатель должен был подумать примерно так: ага, ну с командами и параметрами (т. е. с грамматикой командной строки) мы немного разобрались, вооружите же нас теперь списком всех команд Linux (иначе говоря, словарём), и мы примемся за работу. Почему же нигде не напечатан такой список? Точнее, списков команд много разных и все они очевидно неполные и не во всём сходятся. Ответ на этот вопрос состоит из двух частей.

Часть 1: команды и утилиты

Shell, командный интерпретатор, является «оболочкой» не только для пользователя, но и для команд: сам он почти никакие команды не исполняет, передаёт системе. Его задача сводится к тому, чтобы разобрать командную строку, выделить из неё команду и параметры, а затем запустить **утилиту**[6] — программу, имя которой совпадает с именем команды.

Если смотреть «изнутри» командного интерпретатора, то работа с командной строкой происходит примерно так: пользователь вводит строку (команду), shell считывает её, иногда — преобразует по определённым правилам, получившуюся строку разбивает на команду и параметры, а затем запускает утилиту, передавая ей эти параметры. Утилита, в свою очередь, анализирует параметры, выделяет среди них ключи, и делает что попросили, попутно выводя данные для пользователя, после чего завершается. По завершении утилиты возобновляется работа «отступившего на задний план» командного интерпретатора, он снова считывает командную строку, разбирает её, вызывает команду... Так продолжается до тех пор, пока пользователь не скомандует оболочке *завершиться* самой (с помощью команды logout или управляющего символа *Ctrl+D*).

Однако часть команд (меньшую) оболочка всё же выполняет самостоятельно, не вызывая никаких утилит. Некоторые — самые нужные — команды встроены в bash, даже несмотря на то, что они имеются в виде утилит (например, echo). Работает встроенная команда так же, но так как времени на её выполнение уходит существенно меньше, командный интерпретатор выберет именно её, если будет такая возможность. В bash тип команды можно определить с помощью команды type. Собственные команды bash называются **builtin** (встроенная команда), а для утилит выводится **путь**, содержащий название каталога, в котором лежит файл с соответствующей программой, и имя этой программы. Ключ «-a» («**a**ll», конечно), заставляет type вывести *все* возможные варианты интерпретации команды, а ключ

«-t» — вывести тип команды вместо пути.

```
[methody@localhost methody]$ type date
 info is /bin/date
[methody@localhost methody]$ type echo
 echo is a shell builtin
[methody@localhost methody]$ type -a echo
 echo is a shell builtin
 echo is /bin/echo
[methody@localhost methody]$ type -a -t echo
 builtin
 file
```

Пример 10. Определение типа команды

Собственных команд в командном интерпретаторе немного. В основном это — операторы языка программирования и прочие средства управления самим интерпретатором. Все команды, выполняющие содержательную работу для пользователя, представлены в Linux в виде отдельных утилит. Вот и первая часть ответа на вопрос обо всех командах Linux: их столько же, сколько есть программ (утилит), написанных для Linux. Их список — это список установленных в системе утилит, и в разных системах он будет различным.

Часть 2: всему своё руководство

Каждый объект системы: все **утилиты**, все **демоны** Linux, все функции **ядра** и **библиотек**, структура большинства **конфигурационных файлов**, наконец, многие умозрительные, но важные понятия — должны обязательно сопровождаться документацией, описывающей их назначение и способы использования. Поэтому от пользователя системы не требуется *заучивать* все возможные варианты взаимодействия с ней. Достаточно *понимать* основные принципы её устройства и уметь находить справочную информацию. Эйнштейн говорил на этот счёт так: «Зачем запоминать то, что всегда можно посмотреть в справочнике?».

Больше всего *различной* полезной информации содержится в **страницах руководства** (**manpages**). Каждая страница руководства (для краткости — просто «руководство») посвящена какому-нибудь одному объекту системы. Для того, чтобы посмотреть страницу руководства, нужно дать команду man *объект*:

```
[methody@localhost methody]$ man cal
 CAL(1)          BSD General Commands Manual          CAL(1)

 NAME
    cal - displays a calendar

 SYNOPSIS
    cal [-smjy13] [[month] year]

 DESCRIPTION
    Cal displays a simple calendar.  If arguments are not specified,
       the current month is displayed.  The options are as follows:
. . .
```

Пример 11. Просмотр страницы руководства

«Страница руководства» занимает, как правило, больше одной страницы *экрана*. Для того, чтобы читать было удобнее, man запускает программу постраничного просмотра текстов — less. Управлять программой less просто: страницы перелистываются пробелом, а когда читать надоест, надо нажать «q» (Quit). Перелистывать страницы можно и клавишами *Page Up*/*Page Down*, для сдвига на *одну* строку вперёд можно применять *enter* или стрелку вниз, а на одну строку назад — стрелку вверх. Переход на начало и конец текста выполняется по командам «g» и «G» соответственно (Go). Полный список того, что можно делать с текстом в less, выводится по команде «H» (Help).

Страница руководства состоит из **полей** — стандартных разделов, с разных сторон описывающих объект. При первом изучении руководства стоит начать с полей NAME (краткое описание объекта) и DESCRIPTION (развёрнутое описание объекта, достаточное для того, чтобы им воспользоваться). Одно из самых важных полей руководства находится в конце текста. Если в процессе чтения NAME или DESCRIPTION пользователь понимает, что не нашёл в руководстве того, что искал, он может захотеть посмотреть, а есть ли *другие* руководства или иные источники информации *по той же теме*. Список таких источников содержится в поле SEE ALSO:

```
[methody@localhost methody]$ man man
. . .
SEE ALSO
       apropos(1), whatis(1), less(1), groff(1), man.conf(5).
. . .
```

Пример 12. Поле SEE ALSO руководства

В поле SEE ALSO обнаружились ссылки на руководства по less, groff (программе форматирования страницы руководства), структуре **конфигурационного файла** для man, а также по двум сопутствующим командам whatis и apropos, которые помогают отыскать нужное руководство. Обе они работают с базой данных, состоящей из полей NAME всех страниц руководства в системе. Различие между ними — в том, что whatis ищет только среди имён объектов (в *левых* частях полей NAME), а apropos — по всей базе. В результате у whatis получается список кратких описаний объектов с *именами*, включающими в себя искомое слово, а у apropos — список, в котором это слово *упоминается*.

В системе может встретиться несколько объектов *разного* типа, но с одинаковым названием. Часто совпадают, например, имена **системных вызовов** (функций **ядра**) и утилит, которые позволяют пользоваться этими функциями из командной строки. При ссылке на руководство по объекту системы принято непосредственно после имени объекта ставить в круглых скобках номер раздела, в котором содержится руководство по этому объекту: man(1), less(1), passwd(5). По такому формату легко опознать, что имеется в виду руководство.

В системе руководств Linux девять разделов, каждый из которых содержит страницы руководства к объектам определённого типа. Все разделы содержат по одному руководству с именем «intro», в котором в общем виде и на примерах рассказано, что за объекты имеют отношение к данному разделу. Список разделов с названиями можно получить командой whatis intro.

По умолчанию man просматривает все разделы и показывает *первое найденное* руководство с заданным именем. Чтобы посмотреть руководство по объекту из определённого раздела, необходимо в качестве первого параметра команды man указать номер раздела, например, man 8 passwd.

Другой источник информации о Linux и составляющих его программах — справочная подсистема info. Страница руководства, несмотря на обилие ссылок различного типа, остаётся «линейным» текстом, структурированным только логически. Документ info — это настоящий гипертекст, в котором множество небольших страниц объединены в дерево. В каждом разделе документа info всегда есть оглавление, из которого можно перейти сразу к нужному подразделу, откуда всегда можно вернуться обратно. Кроме того, info-документ можно читать и как *непрерывный* текст, поэтому в каждом подразделе есть ссылки на предыдущий и последующий подразделы. Можно догадаться, что подробное руководство по тому, как перемещаться между страницами в info можно получить по команде info info.

Команда info, введённая без параметров, предлагает пользователю список всех документов info, установленных в системе.

Если некоторый объект системы не имеет документации ни в формате man, ни в формате info, это нехорошо. В этом случае можно надеяться, что при нём есть *сопроводительная документация*, не имеющая, увы, ни стандартного формата, ни тем более — ссылок на руководства по другим объектам системы. Такая документация (равно как и примеры использования объекта), обычно помещается в каталог /usr/share/doc/*имя_объекта*.

Документация в подавляющем большинстве случаев пишется на простом английском языке. Если английский — не родной язык для автора документации, она будет только проще. Традиция писать по-английски идёт от немалого вклада США в развитие компьютерной науки вообще и Linux в частности. Кроме того, английский становится языком международного общения во всех областях, не только в компьютерной. Необходимость писать на языке, который будет более или менее понятен большинству пользователей, объясняется постоянным развитием Linux. Дело не в том, что страницу руководства нельзя перевести, а в том, что её придётся переводить *всякий раз*, когда изменится описываемый ею объект! Например, выход новой версии программного продукта сопровождается изменением его возможностей и особенностей работы, а следовательно, и новой версией документации. Тогда *перевод* этой документации превращается в «moving target», сизифов труд.

Тем не менее, некоторые наиболее актуальные руководства всё-таки существуют в переводе на русский язык. Наиболее свежие версии таких переводов на русский собраны в пакете man-pages-ru — достаточно установить этот пакет, и те руководства, для которых есть перевод, man будет по умолчанию отображать на русском языке.

Переменные окружения

Помимо параметров, передаваемых в командной строке, в Linux есть ещё один способ модифицировать поведение программы — для этого используются **переменные окружения**. Чтобы объяснить принцип работы переменных окружения, потребуется небольшой экскурс в механизм взаимодействия процессов в Linux.

Выполняющаяся программа называется в Linux **процессом**. Каждый запускаемый процесс система снабжает неким информационным пространством, которое этот процесс вправе изменять как ему заблагорассудится — это и есть **окружение** (по-английски environment). Правила пользования этим пространством просты: в нём можно задавать именованные хранилища данных (**переменные окружения**), в которые записывать какую угодно информацию (присваивать значение переменной окружения), а впоследствии эту информацию считывать (подставлять значение переменной).

Процессы — это основные действующие лица в системе. Когда пользователь отдаёт команды в командной строке, то новые процессы для выполнения этих команд (внешние утилиты и т. п.) запускает другой процесс — тот самый командный интерпретатор, который общается с пользователем и принимает от него команды.

Создание одного процесса другим называется *порождением процесса* и происходит в два этапа: сначала создаётся точная копия исходного, *родительского* процесса (системный вызов fork()), а затем копия процесса подменяется новым, *дочерним* (системный вызов exec()). Для нас сейчас важно, что при этой подмене сохраняются все свойства исходного процесса, и, в частности, окружение.

Вернёмся к работе командного интерпретатора: выполняя команду, он запускает нужную утилиту в качестве дочернего процесса, дожидается окончания её работы (при помощи ещё одного системного вызова, wait()), анализирует результат и продолжает работу. Запущенная утилита получает от родительского процесса (командного интерпретатора) информацию двух типов: *параметры командной строки* (не в том виде, в котором их ввёл пользователь, а после обработки по правилам командного интерпретатора, в виде последовательного списка) и *окружение*, то есть все переменные и их значения, которые были определены в окружении родительского процесса.

Одна и та же утилита может быть использована *одним и тем же* способом, но в изменённом окружении — и выдавать различные результаты. Пользователь может явно изменить окружение для

запускаемого процесса, присвоив некоторое значение переменной окружения в командной строке *перед* именем команды. Командный интерпретатор, увидев «=» внутри первого слова командной строки, приходит к выводу, что это — операция присваивания, а не имя команды, и запоминает, как надо изменить окружение команды, которая последует после.

```
[methody@localhost methody]$ date
 Птн Ноя  5 16:20:16 MSK 2004
[methody@localhost methody]$ LC_TIME=C date
 Fri Nov  5 16:20:23 MSK 2004
```

Пример 13. Утилита date пользуется переменной окружения LC_TIME

Переменная окружения LC_TIME предписывает использовать определённый язык при выводе даты и времени а значение «C» соответствует «стандартному системному» языку (чаще всего — английскому).

Переменные, которые командный интерпретатор **bash** определяет *после* запуска, не принадлежат окружению, и, стало быть, не наследуются дочерними процессами. Чтобы переменная **bash** попала в окружение, её надо *проэкспортировать* командой export:

```
[methody@localhost methody]$ LC_TIME=C
[methody@localhost methody]$ date
 Птн Ноя  5 16:20:16 MSK 2004
[methody@localhost methody]$ export LC_TIME=C
[methody@localhost methody]$ date
 Fri Nov  5 16:20:23 MSK 2004
```

Пример 14. Экспорт переменных shell в окружение

Во время сеанса работы пользователя командный интерпретатор получает довольно богатое окружение, к которому добавляет и собственные настройки. Большинство заранее определённых переменных используются либо самой командной оболочкой, либо утилитами системы, поэтому их изменение приводит к тому, что оболочка или утилиты начинают работать слегка иначе. Просмотреть окружение в **bash** можно с помощью команды set.

К значению любой переменной в bash можно обратиться по имени: вместо конструкции $*имя_переменной* оболочка подставит значение этой переменной. Например, для того, чтобы просмотреть значение некоторой переменной, пользователь может ввести команду echo $*имя_переменной*.

```
[methody@localhost methody] echo $PATH
/home/methody/bin:/bin:/usr/bin:/usr/local/bin:/usr/X11R6/bin:/usr/games
```

Пример 15. Вывод значения переменной

Весьма примечательна переменная окружения PATH. В ней содержится список каталогов, элементы которого разделяются двоеточиями. Если команда в командной строке — не собственная команда shell (вроде cd) и не представлена в виде *пути* к запускаемому файлу (как /bin/ls или ./script), то shell будет искать эту команду среди имён запускаемых файлов во всех каталогах PATH, и только в них. По этой причине исполняемые файлы невозможно запускать просто по имени, если они лежат в текущем

каталоге, и текущий каталог не входит в PATH. В таких случаях можно воспользоваться кратчайшим из возможных путей, «./» (например, вызывая сценарий ./script).

Переменных окружения, влияющих на работу *разных* утилит, довольно много. Например, переменные семейства LC_ (полный их список выдаётся командой locale), определяющие язык, на котором выводятся диагностические сообщения, стандарты на формат даты, денежных единиц, чисел, способы преобразования строк и т. п. Если какая-то утилита требует редактирования файла, этот файл передаётся программе, путь к которой хранится в переменной EDITOR (обычно это /usr/bin/vi), а если потребуется открыть html-файл, то многие утилиты вызовут для этого программу, указанную в переменной BROWSER. Наконец, некоторые переменные вроде UID, USER или PWD просто содержат полезную информацию, которую можно было бы добыть и другими способами.

1В Linux основу API составляют **системные вызовы** и стандартные **библиотечные функции**.

2Игра слов: «Bourne Again» вслух читается как «born again», т. е. «возрождённый».

3Представьте себе язык, в котором для выражения *любой* мысли существует отдельное слово — он был бы невероятно неэффективным, и обязательно нашлась бы мысль, на этом языке невыразимая. В естественных языках для выражения мысли используются мощные средства комбинации и модификации слов, и, соответственно, их значений — грамматика языка. Аналогичный принцип действует и в языке командной оболочки, только здесь «грамматику» принято называть синтаксисом.

4Многие склонны вместо слова «ключ» употреблять слово «опция» как аналог английского option, однако это не признак хорошего стиля.

5Кстати, с незапамятных времён who поддерживает один *нестандартный* набор параметров: who am i делает то же, что и who -m.

6Все программы, которые здесь обсуждаются и будут обсуждаться принято называть утилитами, то есть полезными программами.

Редактирование ввода

Редактирование командной строки

Некоторое время поработав в Linux, понабирав команды в командной строке, приходишь к выводу, что в общении с оболочкой не помешают кое-какие удобства. Одно из таких удобств — возможность редактировать вводимую строку с помощью клавиши *Backspace* (удаление последнего символа), *Ctrl+W* (удаление слова) и *Ctrl+U* (удаление всей строки) — предоставляет сам терминал Linux. Эти команды работают для *любого* построчного ввода в терминале. Если по каким-то причинам в строчку на экране влез мусор, можно нажать *Ctrl+R* (**r**edraw) — система выведет в новой строке содержимое входного буфера.

Командная оболочка поддерживает некоторые базовые операции по редактированию командной строки, которых можно ожидать для любого текстового ввода. Речь идёт о клавишах *Стрелка влево* и *Стрелка вправо*, с помощью которых можно перемещать курсор по командной строке, и клавише *Del*, удаляющей символ *под* курсором, а не позади него. Помимо этого перемещаться в командной строке можно не только по одному сиволу вперёд и назад, но и по словам: команды *ESC*F/*ESC*B или *Alt*+F/*Alt*+B соответственно (от **f**orward и **b**ackward), работают также клавиши *Home* и *End*, или, что то же самое, *Ctrl+A* и *Ctrl+E*.

История команд

Bash располагает весьма мощным механизмом — возможностью работать с **историей команд**. Все команды, набранные пользователем, bash запоминает и позволяет обращаться к ним впоследствии. Для работы с историей команд используются клавиши со стрелками — вверх и вниз. По стрелке вверх (можно использовать и *Ctrl+P*, **p**revious), список поданных команд «прокручивается» от последней к первой, а по стрелке вниз (*Ctrl+N*, **n**ext) — обратно. Соответствующая команда отображается в командной строке как только что набранная, её можно отредактировать и подать оболочке (подгонять курсор к концу строки при этом не обязательно).

Если необходимо добыть из истории какую-то давнюю команду, проще не гонять список истории стрелками, а *поискать* в ней с помощью команды *Ctrl+R* (**r**everse search). При этом выводится подсказка специального вида («(reverse-i-search)»), подстрока поиска (окружённая символами ` и ') и последняя из команд в истории, в которой эта подстрока присутствует:

```
[methody@localhost methody]$
 ^R | (reverse-i-search)`': 
 i  | (reverse-i-search)`i': ls i
 n  | (reverse-i-search)`in': info
 f  | (reverse-i-search)`inf': info
 o  | (reverse-i-search)`info': info
 ^R | (reverse-i-search)`info': man info
 ^R | (reverse-i-search)`info': info "(bash.info.bz2)Commands For History"
```

Пример 1. Поиск по истории команд

Пример представляет символы вводимые пользователем (в левой части до «|») и содержимое последней строки терминала. Это «кадры» работы с одной и той же строкой, показывающие, как она меняется при наборе. Набрав «info», пользователь продолжил поиск этой подстроки, повторяя *Ctrl+R* до тех пор, пока не наткнулся на нужную ему команду, содержащую подстроку «info». Осталось только передать её bash с помощью *Enter*.

Чтобы история команд могла сохраняться *между* сеансами работы пользователя, bash записывает её в файл .bash_history, находящийся в домашнем каталоге пользователя. Делается это в момент *завершения* оболочки: накопленная за время работы история дописывается в конец этого файла. При следующем запуске bash считывает .bash_history целиком. История хранится не вечно, количество запоминаемых команд в .bash_history ограничено (обычно 500 командами, но это можно и перенастроить).

Сокращения

Поиск по истории — удобное средство: длинную командную строку можно не набирать целиком, а выискать и использовать. Однако *давнюю* команду придётся добывать с помощью нескольких *Ctrl+R* — а можно и совсем не доискаться, если она уже выбыла оттуда. Для того, чтобы оперативно заменять длинные команды короткими, стоит воспользоваться **сокращениями** (aliases). В конфигурационных файлах командного интерпретатора пользователя обычно *уже* определено несколько сокращений, список которых можно посмотреть с помощью команды alias без параметров:

```
[methody@localhost methody]$ alias
 alias cd..='cd ..'
 alias cp='cp -i'
 alias l='ls -lapt'
 alias ll='ls -laptc'
 alias ls='ls --color=auto'
 alias md='mkdir'
 alias mv='mv -i'
 alias rd='rmdir'
 alias rm='rm -i'
```

Пример 2. Просмотр заранее определённых сокращений

Выяснилось, что по команде ls вместо *утилиты* /bin/ls bash запускает собственную команду-сокращение, превращающееся в команду ls --color=auto. *Повторно* появившуюся в команде подстроку «ls» интерпретатор уже не обрабатывает, во избежание вечного цикла. Например, команда ls -al превращается в результате в ls --color=auto -al. Точно так же любая команда, начинающаяся с rm, превращается в rm -i (**i**nteractive), в результате чего ни одно удаление не обходится без вопросов в стиле «rm: удалить обычный файл `файл'?». Избавиться от ненужного сокращения можно с помощью команды unalias.

Достраивание

Сокращения позволяют быстро набирать *команды*, однако никак не затрагивают имён *файлов*, которые чаще всего и оказываются параметрами этих команд. Бывает, что набранной строки — пути к файлу и нескольких первых букв его имени — достаточно для *однозначного* указания на этот файл, потому что по введённому пути болшьше файлов, чьё имя начинается на эти буквы, просто нет. Чтобы не дописывать оставшиеся буквы в bash можно нажать клавишу *Tab*. И bash *сам* достроит имя файла до полного (снова воспользуемся методом «кадров»):

```
[methody@localhost methody]$ ls -al /bin/base
 Tab    | [methody@localhost methody]$ ls -al /bin/basename
-rwxr-xr-x  1 root root 12520 Июн  3 18:29 /bin/basename
[methody@localhost methody]$ base
 Tab    | [methody@localhost methody]$ basename
 Tab    | [methody@localhost methody]$ basename ex
 Tab    | [methody@localhost methody]$ basename examples/
 Tab    | [methody@localhost methody]$ basename examples/sample-file
sample-file
```

Пример 3. Использование достраивания

Дальше — больше. Оказывается, и имя команды можно вводить не целиком: оболочка догадается достроить набираемое слово именно до команды, раз уж это слово стоит в начале командной строки. Таким образом, команда basename examples/sample-file была набрана за *восемь* нажатий клавиш («base» и четыре *Tab*)! Не потребовалось вводить начало имени файла в каталоге examples, потому что файл там был всего один.

Выполняя **достраивание** (completion), bash может вывести не всю строку, а только ту её часть, относительно которой у него нет сомнений. Если дальнейшее достраиване может пойти *несколькими* путями, то однократное нажатие *Tab* приведёт к тому, что bash растерянно пискнет[1], а повторное — к выводу *под* командной строкой списка всех возможных вариантов. В этом случае надо подсказать командной оболочке продолжение: дописать несколько символов, определяющих, по какому пути пойдёт достраивание, и снова нажать *Tab*.

Дополнения в bash находятся ещё не на самой вершине удобства и экономии нажатий на клавиши. Если в bash *несколько* типов достраивания (по именам файлов, по именам команд и т. п.), то в zsh их *сколько угодно*: существует способ запрограммировать любой алгоритм достраивания и задать шаблон командной строки, в которой именно этот способ будет применяться.

Генерация имён файлов

Достраивание очень удобно, когда цель пользователя — задать *один* конкретный файл в командной строке. Если же нужно работать сразу с *несколькими* файлами — например для перемещения их в другой каталог с помощью mv, достраивание не помогает. Необходим способ задать одно «общее» имя, которое будет описывать сразу группу файлов, с которыми будет работать команда. В подавляющем большинстве случаев это можно сделать при помощи **шаблона**.

Шаблоны

Символы в шаблоне разделяются на обычные и **специальные**. Обычные символы означают сами себя, а специальные обрабатываются особым образом:

- Шаблону, стостоящему только из обычных символов, соответствует *единственная* строка, состоящая из тех же символов в том же порядке. Например, шаблону «abc» соответствует строка abc, но не aBc или ABC, потому что большие и маленькие буквы различаются.
- Шаблону, состоящему из единственного спецсимвола «*», соответствует *любая* строка любой длины (в том числе и пустая).
- Шаблону, состоящему из единственного спецсимвола «?», соответствует *любая* строка длиной в *один* символ, например, a, + или @, но не ab или 8888.
- Шаблону, состоящему из любых символов, заключённых в квадратные скобки «[» и «]» соответствует строка длиной в *один* символ, причём этот символ должен *встречаться* среди заключённых в скобки. Например, шаблону «[bar]» соответствуют только строки a, b и r, но не c, B, bar или ab. Символы внутри скобок можно не перечислять полностью, а задавать *диапазон*, в начале которого стоит символ с наименьшим ASCII-кодом, затем следует «-», а затем — символ с наибольшим ASCII-кодом. Например, шаблону «[0-9a-fA-F]» соответствует одна шестнадцатеричная цифра (скажем, 5, e или C). Если после «[» в шаблоне следует «!», то ему соответствует строка из одного символа *не* перечисленного между скобками.
- Шаблону, состоящему из *нескольких* частей, соответствует строка, которую можно разбить на столько же подстрок (возможно, пустых), причём первая подстрока будет отвечать первой части шаблона, вторая — второй части и т. д. Например, шаблону «a*b?c» будут соответствовать строки ab@c («*» соответвтует пустая подстрока), a+b=c и aaabbc, но не соответствовать abc («?» соответвтует подстрока c, а для «c» соответствия не находится), @ab@c (нет соответствия для «a») или aaabbbc (из трёх b превое соответствует «b», второе — «?», а вот третье приходится на «c»).

Использование шаблонов

Шаблоны используются в нескольких конструкциях shell. Главное место их применения — командная строка. Если оболочка видит в командной строке шаблон, она немедленно заменяет его на список *файлов*, имена которых ему соответствуют. Команда, которая затем вызывается, получает в качестве параметров список файлов уже безо всяких шаблонов, как если бы этот список пользователь ввёл вручную. Эта способность командного интерпретатора называется **генерацией имён файлов**.

```
[methody@localhost methody]$ ls .bash*
 .bash_history  .bash_logout  .bash_profile  .bashrc
[methody@localhost methody]$ /bin/e*
 /bin/ed /bin/egrep /bin/ex
[methody@localhost methody]$ ls *a*
 sample-file
[methody@localhost methody]$ ls -dF *[ao]*
 Documents/  examples/  loop  to.sort*
```

Пример 4. Использование шаблона в командной строке

При использовании шаблонов новичок может натолкнуться на несколько «подводных камней». В приведённом примере только первая команда срабатывает *не* вопреки ожиданиям: шаблон «.bash*» был превращён командной оболочкой в список файлов, начинающихся на .bash, этот список получила в качестве параметров командной строки утилита ls, после чего честно его вывела. «/bin/e*» — это на самом деле опасная команда, с которой в данном случае просто повезло: этот шаблон превратился в список файлов из каталога /bin, начинающихся на «e», и *первым* файлом в списке оказалась безобидная утилита /bin/echo. Поскольку в командной строке ничего, кроме шаблона, не было, именно строка /bin/echo была воспринята оболочкой в качестве *команды*, которой — в качестве *параметров* — были переданы *остальные* элементы списка — /bin/ed, /bin/egrep и /bin/ex.

Что же касается ls *a*, то кажется, что эта команда должна была выдать список файлов в текущем каталоге, имя которых *содержит* «a». Вместо этого на экран вывелось имя файла из подкаталога examples... Впрочем, никакой чёрной магии тут нет. Во-первых, имена файлов вида «.bash*» хотя и содержат «a», но начинаются на точку, и, стало быть, считаются **скрытыми**. Скрытые файлы попадают в результат генерации имён только если точка в начале указана *явно* (как в первой команде примера). Поэтому по шаблону «*a*» в домашнем каталоге bash нашёл только подкаталог с именем examples, его-то он и передал в качестве параметра утилите ls. Что вывелось на экран в результате образовавшейся команды ls examples? Конечно, содержимое каталога. Шаблон в последней команде из примера, «*[ao]*», был превращён в список файлов, чьи имена содержат «a» или «o» — Documents examples loop to.sort, а ключ «-d» потребовал у ls показывать информацию о каталогах, а не об их содержимом. В соответствии с ключом «-F», ls расставил «/» после каталогов и «*» после исполняемых файлов.

Ещё одно отличие генерации имён от стандартной обработки шаблона — в том, что символ «/», разделяющий элементы пути, *никогда* не ставится в соответствие «*» или диапазону. Происходит это не потому, что искажён алгоритм, а потому, что при генерации имён шаблон применяется именно к элементу пути, внутри которого уже нет «/». Например, получить список файлов, которые находятся в каталогах /usr/bin и /usr/sbin и содержат подстроку «ppp» в имени, можно с помощью шаблона «/usr/*bin/*ppp*». Однако *одного* шаблона, который бы включал в этот список ещё и каталоги /bin и /sbin — то есть подкаталоги *другого* уровня вложенности — по стандартным правилам сделать нельзя[2].

Если перед любым специальным символом стоит «\», этот символ лишается специального значения, **экранируется**: пара «*символ*» заменяется командным интерпретатором на «*символ*» и передаётся в командную строку безо всякой дальнейшей обработки:

```
[methody@localhost methody]$ echo *o*
 Documents loop to.sort
[methody@localhost methody]$ echo \*o\*
 *o*
[methody@localhost methody]$ echo "*o*"
 *o*
[methody@localhost methody]$ echo *y*
 *y*
[methody@localhost methody]$ ls *y*
 ls: *y*: No such file or directory
```

Пример 5. Экранирование специальных символов и обработка «пустых» шаблонов

Обратите внимание, что шаблон, которому не соответствует *ни одного* имени файла, bash раскрывать не стал, как если бы все «*» в нём были экранированы. В самом деле, какое из двух зол меньшее: изменять *интерпретацию* спецсимволов в зависимости от содержимого каталога, или сохранять логику интерпретации с риском превратить команду с параметрами в команду *без параметров*? Если бы, допустим, шаблон, которому не нашлось соответствия, попросту удалялся, то команда ls *y* превратилась бы в ls и неожиданно выдала бы содержимое всего каталога. Авторы bash (как и Стивен Борн, автор самой первой командной оболочки — sh) выбрали более непоследовательный, но и более безопасный первый способ[3].

Лишить специальные символы их специального значения можно и другим способом. Разделители (пробелы, символы табуляции и символы перевода строки) перестают восприниматься таковыми, если часть командной строки, их содержащую, окружить двойными или одинарными кавычками. В кавычках престаёт «работать» и генерация имён (как это видно из примера), и интерпретация других специальных символов. Двойные кавычки, однако, допускают выполнение **подстановок** переменной окружения и результата работы команды.

1Все терминалы должны уметь выдавать звуковой сигнал при *выводе* управляющего символа *Ctrl+G*. Для этого не нужно запускать никаких дополнительных программ: «настоящие» терминалы имеют встроенный динамик, а виртуальные консоли обычно пользуются системным («пищалкой»). В крайнем случае разрешается привлекать внимание пользователя другими способами: например, эмулятор терминала screen пишет в служебной строке «wuff-wuff» («гав-гав»).

2Генерация имён файлов в zsh предусматривает специальный шаблон «**», которому соответствуют подстроки с *любым* количеством «/». Пользоваться им следует *крайне осторожно*, понимая, что при генерации имён по такому шаблону выполняется операция, аналогичная не ls, а ls -R или find. Так, использование «/**» в начале шаблона вызовет просмотр *всей* файловой системы!

3Авторы zsh пошли по другому пути: в этой версии shell использование шаблона, которому не соответствует ни одно имя файла, приводит к *ошибке*, и соответствующая команда не выполняется.

Стандартный ввод и стандартный вывод

Программа обычно ценна тем, что может обрабатывать данные: принимать одно, на выходе выдавать другое, причём в качестве данных может выступать практически что угодно: текст, числа, звук, видео... Потоки входных и выходных данных для команды называются *ввод* и *вывод*. Потоков ввода и вывода у каждой программы может быть и по несколько. В Linux каждый процесс, при создании в обязательном порядке получает так называемые *стандартный ввод* (standard input, stdin) и *стандартный вывод* (standard output, stdout) и *стандартный вывод ошибок* (standard error, stderr).

Стандартные потоки ввода/вывода предназначены в первую очередь для обмена текстовой информацией. Тут даже не важно, кто общается с помощью текстов: человек с программой или программы междй собой — главное, чтобы у них был канал передачи данных и чтобы они говорили «на одном языке».

Текстовый принцип работы с машиной позволяет отвлечься от конкретных частей компьютера, вроде системной клавиатуры и видеокарты с монитором, рассматривая единое *оконечное устройство*, посредством которого пользователь вводит текст (команды) и передаёт его системе, а система выводит необходимые пользователю данные и сообщения (диагностику и ошибки). Такое устройство называется *терминалом*. В общем случае терминал — это точка входа пользователя в систему, обладающая способностью передавать текстовую информацию. Терминалом может быть отдельное внешнее устройство, подключаемое к компьютеру через порт последовательной передачи данных (в персональном компьютере он называется «COM port»). В роли терминала может работать (с некоторой поддержкой со стороны системы) и программа (например, xterm или ssh). Наконец, *виртуальные консоли* Linux — тоже терминалы, только организованные программно с помощью подходящих устройств современного компьютера.

При работе с командной строкой, стандартный ввод командной оболочки связан с клавиатурой, а стандартный вывод и вывод ошибок — с экраном монитора (или окном эмулятора терминала). Покажем на примере простейшей команды — cat. Обычно команда **cat** читает данные из всех файлов, которые указаны в качестве её параметров, и посылает считанное непосредственно в стандартный вывод (stdout). Следовательно, команда

```
/home/larry/papers# cat history-final masters-thesis
```

выведет на экран сначала содержимое файла history-final, а затем — файла masters-thesis.

Однако если имя файла не указано, программа **cat** читает входные данные из stdin и немедленно возвращает их в stdout (никак не изменяя). Данные проходят через **cat**, как через трубу. Приведём пример:

```
/home/larry/papers# cat
Hello there.
Hello there.
Bye.
Bye.
Ctrl-D/home/larry/papers#
```

Каждую строчку, вводимую с клавиатуры, программа cat немедленно возвращает на экран. При вводе информации со стандартного ввода конец текста сигнализируется вводом специальной комбинации клавиш, как правило **Ctrl-D**.

Приведём другой пример. Команда **sort** читает строки вводимого текста (также из stdin, если не указано ни одного имени файла) и выдаёт набор этих строк в упорядоченном виде на stdout. Проверим её действие.

```
/home/larry/papers# sort
bananas
carrots
apples
Ctrl+D
apples
bananas
carrots /home/larry/papers#
```

Как видно, после нажатия **Ctrl-D**, **sort** вывела строки упорядоченными в алфавитном порядке.

Перенаправление ввода и вывода

Допустим, вы хотите направить вывод команды sort в некоторый файл, чтобы сохранить упорядоченный по алфавиту список на диске. Командная оболочка позволяет перенаправить стандартный вывод команды в файл, используя символ >. Приведём пример:

```
/home/larry/papers# sort > shopping-list
bananas
carrots
apples
Ctrl-D/home/larry/papers#
```

Можно увидеть, что результат работы команды sort не выводится на экран, однако он сохраняется в файле с именем shopping-list. Выведем на экран содержимое этого файла:

```
/home/larry/papers# cat shopping-list
apples
bananas
carrots
/home/larry/papers#
```

Пусть теперь исходный неупорядоченный список находится в файле items. Этот список можно упорядочить с помощью команды **sort**, если указать ей, что она должна читать из данного файла, а не из своего стандартного ввода, и кроме того, перенаправить стандартный вывод в файл, как это делалось выше. Пример:

```
/home/larry/papers# sort items shopping-list
/home/larry/papers# cat shopping-list
apples
bananas
carrots
/home/larry/papers#
```

Однако можно поступить иначе, перенаправив не только стандартный вывод, но и *стандартный ввод* утилиты из файла, используя для этого символ <:

```
/home/larry/papers# sort < items
apples
bananas
carrots
/home/larry/papers#
```

Результат команды **sort < items** эквивалентен команде **sort items**, однако первая из них демонстрирует следующее: при выдаче команды **sort < items** система ведёт себя так, как если бы данные, которые содержатся в файле items, были введены со стандартного ввода. Перенаправление осуществляется командной оболочкой. Команде **sort** не сообщалось имя файла items: эта команда читала данные из своего стандартного ввода, как если бы мы вводили их с клавиатуры.

Введём понятие *фильтра*. Фильтром является программа, которая читает данные из стандартного ввода, некоторым образом их обрабатывает и результат направляет на стандартный вывод. Когда применяется перенаправление, в качестве стандартного ввода и вывода могут выступать файлы. Как указывалось выше, по умолчанию, stdin и stdout относятся к клавиатуре и к экрану соответственно. Программа sort является простым фильтром — она сортирует входные данные и посылает результат на стандартный вывод. Совсем простым фильтром является программа **cat** — она ничего не делает с входными данными, а просто пересылает их на выход.

Использование состыкованных команд (конвейер)

Выше уже демонстрировалось, как использовать программу sort в качестве фильтра. В этих примерах предполагалось, что исходные данные находятся в некотором файле или что эти исходные данные будут введены с клавиатуры (стандартного ввода). Однако как поступить, если вы хотите отсортировать данные, которые являются результатом работы какой-либо другой команды, например, **ls**?

Будем сортировать данные в обратном алфавитном порядке; это делается опцией -r команды **sort**. Если вы хотите перечислить файлы в текущем каталоге в обратном алфавитном порядке, один из способов сделать это будет таким. Применим сначала команду **ls**:

```
/home/larry/papers# ls
english-list
history-final
masters-thesis
notes
/home/larry/papers#
```

Теперь перенаправляем выход команды **ls** в файл с именем file-list

```
/home/larry/papers# ls > file-list
/home/larry/papers# sort -r file-list
notes
masters-thesis
history-final
english-list
/home/larry/papers#
```

Здесь выход команды **ls** сохранен в файле, а после этого этот файл был обработан командой **sort** -r. Однако этот путь является неизящным и требует использования временного файла для хранения выходных данных программы **ls**.

Решением в данной ситуации может служить создание *состыкованных команд* (pipelines). Стыковку осуществляет командная оболочка, которая stdout первой команды направляет на stdin второй команды. В данном случае мы хотим направить stdout команды **ls** на stdin команды **sort**. Для стыковки используется символ |, как это показано в следующем примере:

```
/home/larry/papers# ls | sort -r
notes
masters-thesis
history-final
english-list
/home/larry/papers#
```

Эта команда короче, чем совокупность команд, и её проще набирать.

Рассмотрим другой полезный пример. Команда

```
/home/larry/papers# ls /usr/bin
```

выдаёт длинный список файлов. Большая часть этого списка пролетает по экрану слишком быстро, чтобы содержимое этого списка можно было прочитать. Попробуем использовать команду more для того, чтобы выводить этот список частями:

```
/home/larry/papers# ls /usr/bin | more
```

Теперь можно этот список «перелистывать».

Можно пойти дальше и состыковать более двух команд. Рассмотрим команду **head**, которая является фильтром следующего свойства: она выводит первые строки из входного потока (в нашем случае на вход будет подан выход от нескольких состыкованных команд). Если мы хотим вывести на экран последнее по алфавиту имя файла в текущем каталоге, можно использовать следующую длинную команду:

```
/home/larry/papers# ls | sort -r | head -1 notes
/home/larry/papers\#
```

где команда **head** -1 выводит на экран первую строку получаемого ей входного потока строк (в нашем случае поток состоит из данных от команды **ls**), отсортированных в обратном алфавитном порядке.

Недеструктивное перенаправление вывода

Эффект от использования символа > для перенаправления вывода файла является деструктивным; иными словами, команда

```
/home/larry/papers# ls > file-list
```

уничтожит содержимое файла file-list, если этот файл ранее существовал, и создаст на его месте новый файл. Если вместо этого перенаправление будет сделано с помощью символов >>, то вывод будет приписан в конец указанного файла, при этом исходное содержимое файла не будет уничтожено. Например, команда

```
/home/larry/papers# ls >> file-list
```

приписывает вывод команды **ls** в конец файла file-list.

Следует иметь в виду, что перенаправление ввода и вывода и стыкование команд осуществляется командными оболочками, которые поддерживают использование символов >, >> и |. Сами команды не способны воспринимать и интерпретировать эти символы.

Путеводитель по coreutils

Антон Бояршинов

Пакет coreutils содержит множество маленьких утилит, которые могут применяться и по отдельности, но особую мощь они обретают будучи объединены между собой и с другими утилитами. Ниже рассмотрены большинство из них (не все). Этот текст не является заменой руководств по этим утилитам, а всего лишь краткой справкой по их возможностям.

Операции над файлами и каталогами

Здесь рассмотрены утилиты, работающие с объектами файловой системы: файлами, каталогами, устройствами, а также с файловыми системами в целом.

cp

копирует файлы и каталоги;

mv

перемещает (переименовывает) файлы;

rm

удаляет файлы и каталоги;

df

выводит отчёт об использовании дискового пространства;

du

оценивает место на диске занимаемого файлами и каталогами;

ln

создаёт ссылки между файлами;

ls

выводит список файлов в каталоге в различных форматах;

mkdir

создаёт каталоги;

touch

изменяет метки времени (последняя модификация, последний доступ) файла;

realpath

вычисляет абсолютное имя файла по относительному;

basename

удаляет из полного имени файла путь;

dirname

удаляет из полного имени файла имя файла;

pwd

выводит имя текущего каталога;

Пример 1. Использование команды du для проверки выявления каталогов, занимающих много места на диске

```
du -b | sort -n
```

Вычисления

Команды **test**, **date** и **expr** совершают вычисления над своими аргументами, остальные могут быть также использованы как фильтры в потоке.

test

сравнивает типы файлов и значения;

date

выводит и устанавливает системную дату, кроме того может быть использована для вычислений над датами;

expr

вычисляет выражения;

md5sum

подсчитывает контрольную сумму по алгоритму MD5;

sha1sum

подсчитывает контрольную сумму по алгоритму SHA1;

wc

подсчитывает количество строк, слов и символов в файле;

factor

разлагает числа на простые множители;

Пример 2. Использование команды test для проверки существования файла

```
if test -e /bin/bash; then echo "bash существует"; else echo "bash не существует, странно"; fi;
```

Пример 3. Использование команды test для сравнения чисел

```
if test 5 -gt 7; then echo "5 > 7"; else echo "7 > 5"; fi;
```

Пример 4. Использование команды wc для подсчёта суммы строк во всех файлах на языке C в текущем каталоге

```
cat *.[ch] | wc -l
```

Пример 5. Использование команды date для получения вчерашней даты

```
date -d yesterday/
```

Фильтры

Приведённые в этом разделе команды, как правило, используются как фильтры в потоке, хотя многие из них могут работать и непосредственно с файлами.

cat

объединяет файлы и выводит их на стандартный вывод;

tac

объединяет файлы и выводит их на стандартный вывод, начиная с конца;

sort

сортирует строки;

uniq

удаляет дублирующиеся строки из отсортированных файлов;

tr

выполняет транслитерацию, сжатие и/или удаление символов из файла;

cut

вырезает части из каждой строки файла;

csplit

делит файлы на части по шаблону;

expand

преобразует табуляции в пробелы;

unexpand

преобразует пробелы в табуляции;

fmt

форматирует текст по ширине;

fold

переносит слишком длинные текстовые строки на следующую строку;

nl

нумерует строки файла;

od

выводит файл в восьмеричном, шестнадцатеричном и других подобных формах;

tee

читает стандартный вход и выводит и в стандартный выход и в файлы;

Пример 6. Использование команд cat, sort и uniq для получения списка адресов писавших вам людей

```
cat *| grep ^From: |sort |uniq
```

Пример 7. Получение списка слов, содержащихся в тексте с количеством вхождений

```
cat big.text.file | tr '[:space:][:punct:]' "\n" |sort |uniq -c |sort -n -r |less
```

Прочее

head

выводит часть файла заданного размера, начиная с начала;

tail

выводит часть файла заданного размера, начиная с конца, также используется для слежения за файлом и вывода добавляющихся в него строк;

echo

выводит на стандартный вывод текст, заданный в аргументе;

false

всегда возвращает код ошибки;

true

всегда возвращает код успеха;

yes

выводит строку (по умолчанию - «yes») бесконечно, пока не прервут.

seq

выводит последовательность номеров;

sleep

создаёт паузу заданной продолжительности (в секундах);

usleep

создаёт паузу заданной продолжительности (в миллисекундах);

comm

построчно сравнивает 2 отсортированных файла;

join

объединяет строки двух файлов по общему полю;

paste

объединяет строки двух файлов по порядку;

split

разбивает файл на части по размеру;

Ну и в завершение:

```
rpm -ql coreutils | grep man | xargs -n1 basename | sed 's/.1.gz//' | xargs man
```

Ресурсы в сети Интернет

- Сайт ALT Linux
- Поиск по ресурсам ALT Linux
- Вопросы и ответы
- Списки рассылки
- Система отслеживания ошибок
- Документация ALT Linux online

Лицензия

Документ подготовлен для печати ООО «Дальком» г. Хабаровск http://dalcom.kha.ru на основании «Руководства пользователя» взятого с общедоступного сайта ALT Linux http://altlinux.ru:
ftp://ftp.altlinux.ru/pub/distributions/ALTLinux/4.0/Desktop/4.0.0/i586/docs/index.html

Вопросы по редактированию или дополнению новыми статьями обращаться:

ALT Linux: org@altlinux.ru

Дальком: dalcom@hotbox.ru