Linux **на практике** Как установить? Как подобрать программы? Как начать работу?

alt linux

снаружи

на примере дистрибутива
ALT Linux 3.0 Compact

DVD
прилагается

библиотека alt linux

ДМК ПРЕСС

УДК 004.2
ББК 32.973.26-018.2
А45

А45 ALT Linux снаружи. ALT Linux изнутри:
В серии: «Библиотека ALT Linux». — М.: ALT Linux; Издательский дом ДМК-пресс, 2006. — 416 с.; ил.

ISBN 5-9706-0029-6

Пингвины живут даже на юге Африки, не говоря уже о зоопарках и «Сокровищнице Дикой Жизни» Лас-Вегаса (средняя забортная температура летом — +38).

Что это за зверь такой — Linux-пингвин? Что ему делать на моём компьютере? Любят ли пингвины Интернет, и *как* они его любят? Как использовать пингвина в мирных целях? Нуждается ли он в запчастях? А для начала — как засунуть пингвина в холодильник, то есть установить Linux на собственный компьютер? Обо всём этом и о многом другом и рассказывает эта половина книги, первой в серии «Библиотека ALT Linux».

Нашему «пингвину» — операционной системе и внушительной подборке программ на все случаи жизни — будет домом любой холодильник, что серверный, что персональный. Персональный даже интереснее: каждый может сделать со своим пингвином что угодно, достаточно только понять, что же на самом деле происходит внутри холодильника. Правда, для этого надо уметь читать книжки не только с начала, но и с конца.

Пингвин прилагается.

Главный редактор: Д. А. Мовчан `dm@dmk-press.ru`

Редактор серии: К. А. Маслинский `kirill@altlinux.ru`

Корректор: М. Л. Романова

Оглавление

К читателю 7

Часть I Первый день

Глава 1

Перед установкой 10

Что такое ALT Linux 10
ALT Linux 3.0 Compact:
первый листок новой ветки 12
Оборудование 15
Сохранение данных и меры предосторожности .. 16

Глава 2

Установка 18

Начало установки: загрузка системы 18
Введение в программу установки 19
Установка базовой системы 19
Первоначальная настройка системы 26
Завершение установки 27
Первая помощь 28
Настройка загрузки 30

Глава 3

С чего начать? 34

Что нужно знать о Linux пользователю 34
Документация 43
Центр управления ALT Linux 46
Пользователи 47

Часть II Второй день

Глава 4

Что тут есть? 50

Прикладные программы для Linux 50
Установка и удаление программ 71

Глава 5

Интернет 77

Управление сетью 77
Удалённое подключение к Интернет по модему .. 78
Подключение к Интернет через мобильный телефон 81
Подключение через ADSL 84
Настройка почтового клиента 85

Глава 6

Практическое руководство по OpenOffice.org 88

Общая информация об офисном пакете OpenOffice.org 88

Текстовый редактор OpenWriter 90
Электронные таблицы 115
Использование OpenDraw 124
Создание презентаций 139
Работа с базами данных в OpenOffice.org 145
Получение дополнительной информации 150
Благодарности 150

Глава 7

Совместимость 151

WINE: среда для запуска win-приложений на платформе Unix 151

Часть III Оборудование

Глава 8

Работа с оборудованием в Linux: «Сага о Драйверах» 161

Что такое «оборудование»? 162
Как распознаётся оборудование? 163
Что такое «драйвер» и где он находится? 166
Опять «устройство»? 167
Кто виноват и что делать? 168

Глава 9

Принтер 171

Управление принтерами 171
Подсистема печати (CUPS) 174

Глава 10

Видеооборудование 182

Свойства экрана (x11) 182

Глава 11

Жёсткие диски 185

Структура жёсткого диска 185
Именование дисков и разделов в Linux 187
Планирование диска 189

Глава 12

Системная плата 193

Настройка системных часов 193

Глава 1

Перед установкой

Что такое ALT Linux

Алексей Новодворский, Кирилл Маслинский

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[1] — наш ежедневно обновляемый репозиторий пакетов. На его основе создаются все дистрибутивы ALT Linux. Поддержи-

[1] `http://sisyphus.ru`

ваемая 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[1] 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[2].

Sisyphus (Сизиф) — персонаж греческой мифологии. Миф о Сизифе[3], который непрерывно катил в гору камни, символизирует постоянный труд команды по усовершенствованию решений, заложенных в репозиторий. «Миф о Сизифе»[4] — 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 поль-

[1] `http://wiki.sisyphus.ru/changes`

[2] `http://wiki.sisyphus.ru/`

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

[4] `http://www.philosophy.ru/library/camus/01/0.html`

зователи) — 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].

См. также Что такое свободные программы [изнутри, стр. 59]
Что такое сообщество . . . . [изнутри, стр. 70]

ALT Linux 3.0 Compact: первый листок новой ветки

Георгий Курячий

После лета ожидания вышел новый дистрибутив ALT Linux — Compact 3.0. Дистрибутив и в самом деле новый, это видно и на первый взгляд, а также и на второй, и на третий, и после того, как расковыряешь со словами «А что там внутри?».

Полностью переписана программа установки. Яркая зелень самых весенних оттенков («жимолость» — услужливо подсказывает текстовый загрузчик) и совершенно новый интерфейс, обогащённый ненавязчивой и вдумчиво встроенной документацией, — вот отличительные признаки нового инсталлятора. Эксперименты на детях (фестиваль «Цифровой Мир-2005») показали, что документацию можно читать с глубоким интересом, а можно и не читать — Compact 3.0 установится всё равно.

Compact 3.0 — дистрибутив «офисного» плана. Основная его задача — умещаться на одном CD и закрывать «конторско-домашние» функции компьютера: можно работать с собственно офисными документами (пакет OpenOffice.org), читать электронную почту и болтать в сети (thunderbird и psi), слушать музыку и смотреть фильмы (amaroK, xine), работать с графикой (GIMP) и т. д. В нём нет особого разнообразия программ, решающих одну и ту же задачу:

[1] http://www.altlinux.ru/content/view/3/16/

скажем, метафора рабочего стола обеспечивается пакетом KDE, а про Gnome есть только упоминания в документации. Для тех, кто не любит «тяжёлые» графические среды есть оконный диспетчер IceWM, а больше на CD-версии диспетчеров окон нет. Знакомым с Linux не надо объяснять, почему в CD версии нет антивируса (без которого некоторые системы мгновенно превращаются в рассадник нечисти), а незнакомые с Linux в утверждение «в Linux вирусов нет» обычно не верят, привыкли.

Новый пункт меню в KDE — «ALT Linux Control Center» — открывает другую сторону принципиальной новизны `Compact 3.0`, модульную систему настройки всего на свете по имени «Alterator». Alterator — это программный скелет, позволяющий по-быстрому решить типичную задачу администратора «настроить такую-то часть системы», формализовать это решение и написать графический интерфейс к нему. Модули «ALT Linux Control Center» не просто напоминают пункты в программе установки, они ими являются! Более того, всякий, кому необходимо пройти Путь Админа «решил — формализовал — сделал GUI», может ходить, а взять Alterator и ехать.

`Compact 3.0` выходит сразу в трёх ипостасях: CD-версия, Travel CD и DVD-версия. Travel CD — аналог Live CD различных дистрибутивов, самый известный из которых — Knoppix, основанный на Debian: стоит только загрузиться с CD — и готова работающая операционная система — с сетью, распознаванием внешних устройств, работающим KDE и т. п., причём жёсткий диск (если он вообще есть) по умолчанию не используется. Работать можно где угодно, а рабочие файлы хранить в сети или на Flash. Дополнительное свойство Travel CD — возможность быть X-терминалом. В этом режиме загружается только графическая оболочка и менеджер дисплеев `kdm`, который ищет в локальной сети сервер X-терминалов. Что такое «сервер X-терминалов»? А какая разница! Его можно с лёгкостью изготовить, например, из того же `kdm` на машине с установленным Compact-ом, изменив в секции `[Xdmcp]` файла `/etc/X11/kdm/kdmrc` строчку `Enable=false` на `Enable=true`[1]. После выбора одного из серверов показывается обычная `dm`-подсказка («login/password»), и пользователь X-терминала регистрируется в системе и запускает программы на *сервере*, возможно, и не подозревая об этом — до тех

[1] При этом kdm начнёт широковещательно рассылать сообщения вида «Сюда! Сюда заходите!»

пор, пока не захочет послушать музыку (усложнение архитектуры всякими Network Audio System или ESD делать не стали).

DVD-версия ALT Linux Compact 3.0 — тот же CD, в который добавлено почти всё, что было работающего в репозитории пакетов, включая, скажем, антивирус ClamAV, OpenOffice.org 2.0, выпуск которого случился аккурат во время выпуска Compact, или всевозможные серверные пакеты, вроде Apache. Надо признать, что не все из восьми с лишним тысяч пакетов Compact 3.0-DVD «дистрибутивно» включаются в установку по умолчанию. Обратите внимание: это проверенно работающие, оттестированные пакеты! Однако в их установке могут возникнуть некоторые особенности, о которых стоит знать заранее. Главная особенность — «UTF-икация» дистрибутива в объёме CD-версии. Системная консоль и прочие терминалы, среда KDE, русифицированные утилиты — всё это использует Unicode (более точно — кодировку UTF-8). Достоинства Unicode неоспоримы: вы больше не увидите на экране изречений типа «uAAAOO xIDOIO» или «гЮДЮРЭ БНОПНЯ», потому что все языки умеют отображаться в Unicode, где толкование символов однозначно. Не увидите, если будете использовать программы из CD-версии дистрибутива и довольно обширное — но, увы, неполное и не подсчитанное — подмножество программ из DVD-версии. Оставшиеся можно «вылечить», запуская их в стандартной кодовой странице koi8-r (например, так: «LC_ALL=ru_RU.KOI8-R audacity» для запуска редактора звуков audacity); некоторые придётся вообще лишать языка с помощью «LC_ALL=POSIX»), а при запуске в системной консоли стараться не использовать псевдографику (например, запускать «mc -a» вместо «mc»). Осталось заметить, что DVD-версия может работать и в режиме Travel CD, когда Linux загружается и работает без установки и без использования жёсткого диска.

Внутри ALT Linux Compact 3.0 тоже заметно отличается от предыдущих дистрибутивов ALT Linux: использовано новое ядро (2.6.12), мощная и гибкая система сетевых сценариев etcnet сменила довольно путаную netscripts, вместо XFree86 используется Xorg и т. д. Об этом и о многом другом можно прочитать на сайте ALT Linux[1], а также на дружественных этому проекту сайтах.

См. также Какие программы есть в Linux . . . . . [стр. 50]
Офисный пакет OpenOffice.org . . . . . [стр. 88]

[1] http://www.altlinux.ru

Оборудование

Кирилл Маслинский

В настоящее время ядро Linux поддерживает практически любое современное оборудование и очень многое из старого. Большая часть оборудования будет настроена автоматически при условии правильно выставленных параметров BIOS. Однако всегда есть возможность, что с тем или иным экзотическим устройством могут возникнуть сложности. Некоторое старое оборудование не может быть настроено автоматически, поэтому после завершения установки вам может понадобиться выяснить некоторые параметры таких устройств и указать их в системных конфигурационных файлах.

В целом при установке не стоит уделять слишком много внимания настройке оборудования. Жизненно важные для системы устройства наверняка будут определены и настроены автоматически, а всё остальное гораздо удобнее будет настроить в уже установленной системе.

В Интернет можно найти довольно много разных списков оборудования, совместимого с Linux (Linux Hardware Compatibility List). Однако такие списки очень быстро устаревают, так как разработка ядра Linux не стоит на месте, и круг поддерживаемых устройств постоянно расширяется. Кроме того, ни один из таких списков не является полным — при современном разнообразии оборудования полный список вряд ли возможен в принципе. Некоторые сведения о работе оборудования в дистрибутивах ALT Linux пользователи и разработчики размещают на сайте freesource.info[1].

Настройка BIOS

BIOS (Basic Input/Output System) — это первая программа, которая выполняется при включении питания компьютера. В частности, она позволяет указать, на каком устройстве находится операционная система и начинает процесс загрузки ОС. Она также позволяет производить начальную настройку оборудования ещё до загрузки операционной системы.

Если вы собираетесь использовать принтер, подключённый непосредственно к вашему компьютеру, убедитесь, что параллельный порт установлен на EPP (или на ECP+EPP, но в этом случае могут

[1] http://www.freesource.info/wiki/HCL?v=i55

возникать проблемы), а не на SPP. Если этого не сделать, принтер всё равно сможет печатать, но не будет автоматически определяться, и его придётся настраивать вручную. Убедитесь также, что принтер включён и правильно подсоединён к компьютеру.

См. также | О понятии «драйвер» в Linux . . . . . [стр. 161]

Сохранение данных и меры предосторожности

Кирилл Маслинский

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 у вас 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 вашей системе установлена Microsoft Windows, и вы прежде не устанавливали GNU/Linux, то программа установки должна будет изменить размер ваших Windows-разделов на диске. От этой операции могут пострадать ваши данные, поэтому предварительно надо выполнить следующие действия:

  – Запустить scandisk для раздела Windows. Программа изменения размера может обнаружить некоторые очевидные ошибки, но scandisk 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.

Если ни scandisk, ни defrag не установлены под Windows, то обратитесь за инструкциями по их установке в документации по Windows.

Полной гарантией от проблем с потерей данных является резервное копирование!

См. также О резервном копировании стандартными средствами Linux . . . . . [изнутри, стр. 165]

Глава 2

Установка

Начало установки: загрузка системы

Кирилл Маслинский

Загрузка с установочного диска дистрибутива ALT Linux 3.0 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 оборудования. Можно получить справку по любому пункту меню, выбрав этот пункт и нажав `F1`.

Прямо в меню загрузки можно выбрать язык, на котором будет проходить установка и работа в системе: по нажатию `F3` внизу экрана откроется меню со списком языков. Изначально все сообщения отображаются на русском языке.

По нажатию `F2` открывается меню доступных видеорежимов (разрешения экрана). Это разрешение будет использоваться во время установки и загрузки установленной системы. По умолчанию выбирается максимальное разрешение из возможных.

Чтобы начать процесс установки, нужно 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 жёстком диске. Если вы пропустили нужный момент, перезагрузите компьютер и вовремя выберите «Установку».

Начальный этап установки не требует вмешательства пользователя: происходит автоматическое определение оборудования, запуск компонентов программы установки. По умолчанию в это время отображается индикатор выполнения. Если вы хотите знать, что именно происходит на этом этапе — нажмите `F4` и в появившемся в

нижней части экрана меню выберите пункт `native` или `verbose`. Если индикатор уже успел появиться на экране, увидеть, что «скрыто за ним» можно, нажав клавишу *Ecsape*.

Введение в программу установки

Процесс установки разделён на шаги: каждый шаг посвящён настройке или установке определённого свойства системы. Шаги нужно проходить последовательно, однако при необходимости можно вернуться к уже пройденному шагу и изменить настройки. Для этого достаточно щёлкнуть мышью на названии того шага, к которому вы хотите вернуться. Передвигаться между шагами можно также кнопками «Назад» и «Далее», расположенными внизу экрана.

Программа установки разделена на два этапа: сначала производится установка **базовой системы**, а затем — её **первоначальная настройка**. Обратите внимание, что на каждом этапе свой список шагов и вернуться к шагам предыдущего этапа, после того как он закончен, уже невозможно.

Технические сведения о ходе установки можно просмотреть, нажав *Ctrl+Alt+F1*, вернуться к программе установки — *Ctrl+Alt+F7*.

Установка базовой системы

Этот этап установки — наиболее ответственный для пользователя: здесь подготавливается площадка для установки ALT Linux (выделяется место на жёстком диске), а также определяются такие свойства системы, которые будет затруднительно изменить после установки. На этом этапе не нужно спешить с ответами на вопросы, а если что-то непонятно — стоит обратиться к справке, там содержатся краткие пояснения к каждому шагу. Более подробное руководство можно почитать прямо в ходе установки — для этого нажмите кнопку «Меню» и выберите «Руководство». Переход по некоторым ссылкам из текста краткой справки тоже открывает окно руководства. Чтобы закрыть руководство, нужно нажать на кнопку в правом верхнем углу экрана.

Есть возможность *прервать* процесс установки системы: для этого нажмите на кнопку «Меню» в нижней части экрана и вы-

берите пункт «Прервать установку». Незаконченная установка может иметь разные последствия, в зависимости от того, какие шаги уже были выполнены. Совершенно *безопасно* выходить до шага «Разбиение диска», поскольку до этого момента никаких изменений на жёстком диске компьютера не производится. Между шагами «Разбиение диска» и «Установка начального загрузчика» прерывать установку нельзя: после может не загрузиться ни одна из установленных систем.

Если все шаги успешно пройдены, то будет загружена установленная базовая система. При этом полной перезагрузки компьютера не происходит, просто экран на некоторое время гаснет и отображаются служебные сообщения о запуске компонентов системы. Далее на экране появится окно следующего этапа установки — «первоначальной настройки системы» [стр. 26].

Лицензионное соглашение

Перед продолжением установки следует внимательно прочитать условия лицензии. Лицензия относится ко всему дистрибутиву ALT Linux. Если вы не согласны с условиями лицензии, нажмите на клавишу «Отказаться», что немедленно прекратит установку. Нажимая «Принять» для продолжения установки, вы тем самым принимаете условия лицензии.

Выбор языков

Сейчас нужно выбрать те языки, с которыми понадобится работать в будущей системе. Для каждого выбранного языка будут установлены все компоненты, необходимые для ввода и отображения текста на этом языке. Сюда относятся системные сообщения и интерфейс программ, формат дат, символов валюты и др., шрифты, кодировки и раскладки клавиатуры.

Английский язык находится на особом положении, поскольку ввод/вывод на английском поддерживается в любой системе Linux, для этого не нужно ничего дополнительно устанавливать. Выбирать «English» на данном этапе установки следует только в том случае, если вам необходимо установить локали [стр. 21], соответствующие английскому языку.

Среди всех установленных в системе языков один считается **основным**. Этот язык указывается первым в списке доступных в

системе языков и используется по умолчанию большинством программ. Изначально в качестве основного выбран тот язык, на котором проходит установка системы. После установки основной язык в любой момент можно будет сменить (на любой другой из установленных в системе). Все остальные языки, установленные в системе, можно называть **дополнительными**.

> При установке нужно выбрать **все** языки, на которых вам потребуется работать. После того, как система будет установлена, установить ещё один дополнительный язык будет весьма затруднительно.

О локалях

Правильнее следует здесь говорить не о выборе языка, а о выборе **локали**. Выбирая установку того или иного языка, вы тем самым запрашиваете установку всех локалей, имеющихся для этого языка.

Локаль определяется сочетанием языка и страны, название локали состоит из стандартных двухбуквенных кодов языка и страны (например, en_US — английский в США, en_GB — английский в Великобритании, ru_RU — русский в России). «Системному» английскому языку, который всегда присутствует в Linux, соответствует специальная локаль, она называется **C** или **POSIX**. Локаль определяет, как стандартные системные понятия и сообщения следует оформлять в соответствии с нормами, принятыми для данного сочетания язык/страна. Сюда входят названия дней недели и месяцев, алфавитный порядок сортировки для данного языка, символ национальной валюты для данной территории и т. д.

Раскладка клавиатуры для основного языка

Раскладка клавиатуры — это привязка букв, цифр и специальных символов к клавишам на клавиатуре. Сейчас предлагается выбрать раскладку клавиатуры для **основного языка**.

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. Между раскладками для разных языков можно будет переключаться с помощью той клавиши, которая указана после типа раскладки.

Для русского языка доступны три вида раскладки:

- Russian Win (точка и запятая вводятся крайней правой кнопкой в нижнем ряду);
- Russian (точка и запятая вводятся сочетаниями клавиш *Shift+7*, *Shift+6*);
- Russian (Yawerty) (русские буквы привязаны, где возможно, к клавишам соответствующих по написанию или произношению английских букв).

Разбиение жёсткого диска

Зачем нужно разбиение диска?

Чтобы установить Linux, необходимо свободное место на жёстком диске компьютера. Рекомендуется отводить для установки Linux не меньше, чем 4 Гб. Дисковое пространство, как правило, разбивается на несколько областей — **разделов**, для установки Linux требуется создать несколько разделов. Сейчас нужно определить, как их разместить на диске, в результате на диск будет записана **таблица разделов**. Подробнее о технологии разбиения жёсткого диска можно почитать в разделе «Структура жёсткого диска» [стр. 185].

Запись таблицы разделов на диск — необратимая операция, в результате которой **данные**, имеющиеся на диске, **могут быть утрачены**. Если на диске есть данные, которые нужно сохранить (установленная ранее операционная система, пользовательские файлы и т. п.), **не спешите** на этом этапе установки и внимательно **прочитайте справку**.

Пока таблица разделов не записана на диск (запись происходит в тот момент, когда вы нажимаете кнопку «Далее»), можно отменить любые сделанные изменения, вернув диск к исходному состоянию. Для этого следует нажать кнопку «Сброс».

Выбор диска для установки Linux

В колонке «Устройство» отображается список тех дисков, на которые возможно установить Linux. Если в этом списке дисков

ISBN 5-97060-0029-6

Почему Linux? Чего можно добиться с Linux? Когда он удобен и эффективен? Почему он так привлекателен?

alt linux

изнутри

на примере репозитория
ALT Linux Sisyphus

библиотека alt linux

ДМК ПРЕСС

УДК 004.2
ББК 32.973.26-018.2
А45

А45 ALT Linux снаружи. ALT Linux изнутри:
В серии: «Библиотека ALT Linux». — М.: ALT Linux; Издательский дом ДМК-пресс, 2006. — 416 с.; ил.

ISBN 5-9706-0029-6

Пингвины не вьют гнёзд, а яйца высиживают самцы.

Из чего состоит дистрибутив Linux? Как он работает и почему? Что умеет, и как этими умениями управлять? Кто они такие, эти таинственные «линуксоиды»? пингвины? Если писать программы, то про что их писать? И вообще — что на самом деле такое: «Linux», «Дистрибутив», «Сообщество», «Свободные программы» наконец? Обо всём этом и о некотором другом и написано во второй половине нашей книги.

Раз уж приходится управлять таким сложным созданием, как пингвин, и тем более — таким сложным инструментом, как компьютер, без знаний не обойтись. Если вы не считаете поведение компьютера «предопределением природы», и хотите что-то изменить к лучшему, непременно стоит познакомиться с тем, что именно вы хотите изменить. По крайней мере, мы — разработчики дистрибутивов ALT Linux и участники Linux-сообщества — для себя решили начинать именно со знаний.

В комплект входит дистрибутив ALT Linux Compact 3.0.

Главный редактор: Д. А. Мовчан `dm@dmk-press.ru`

Редактор серии: К. А. Маслинский `kirill@altlinux.ru`

Корректор: М. Л. Романова

Оглавление

К читателю 6

Часть I Основы ОС Linux

Глава 1

Интерпретатор командной строки (shell) 9

Синтаксис командной строки 10
Откуда берутся команды 17
Переменные окружения 22

Глава 2

Файлы 25

Файловая система Linux 25
Работа с файлами 28
Типы файловых систем 34

Глава 3

Права 37

Пользователи в Linux 37
Права доступа в системе Linux 42

Глава 4

Процессы 48

Процессы и управление заданиями 48
Что происходит в системе? 55

Часть II Linux общественный

Глава 5

Не бесплатное, а свободное 59

Программное обеспечение: право и свобода 59
Свободные программы и сообщество 70

Глава 6

Больше, чем дистрибутив 91

Система управления пакетами APT 91
Как получить нужную программу 101
Сборка программ для ALT Linux с использованием hasher 108

Часть III Конструктор

Глава 7

Командная строка 112

Ввод, вывод и конвейер 112
Удобства shell: экономия движений 118

Глава 8

Графический интерфейс в Linux 126
Оконная система X и её реализации ... 126
Цветной бутерброд ... 128
«Чистая» X ... 128
Менеджеры окон ... 132
Оконные менеджеры BlackBox и FluxBox ... 140
Оконный менеджер WindowMaker ... 142
Оконный менеджер IceWM ... 143
Интегрированные графические среды ... 144
GNOME ... 147
KDE ... 152
Зачем нужны «лёгкие» среды? ... 154

Глава 9

Своими руками 156
История одного скрипта ... 156
Самодельные мультфильмы ... 163
Резервное копирование информации ... 165
Настройка почтовой системы в Linux ... 169
Linux-класс за час, или X-терминалы, тонкие и ленивые ... 195

Глава 1

Интерпретатор командной строки (shell)

Георгий Курячий, Кирилл Маслинский

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 за-

[1] В Linux основу API составляют **системные вызовы** и стандартные **библиотечные функции**.

дачи 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»[1]. 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 ко-

[1] Игра слов: «Bourne Again» вслух читается как «born again», т. е. «возрождённый».

манду — 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]$
```

А если нужно посмотреть календарь на будущий месяц? Верно, не следует для этого изобретать отдельную команду[1], `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
```

[1] Представьте 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.

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]$
```

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
```

Такого рода параметры называются *модификаторами выполнения* или **ключами** (options)[1]. 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

[1] Многие склонны вместо слова «ключ» употреблять слово «опция» как аналог английского option, однако это не признак хорошего стиля.

«**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
```

Для `who` ключ «`-m`» означает «**M**e», то есть «Я», и в результате `who` работает похоже на `whoami`[1]. 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
```

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

[1] Кстати, с незапамятных времён `who` поддерживает один *нестандартный* набор параметров: `who am i` делает то же, что и `who -m`.

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>.
```

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-
```

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]
```

В синопсисе даётся *формализованное* описание способов использования объекта (в 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**[1] — 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*).

[1] Все программы, которые здесь обсуждаются и будут обсуждаться, принято называть утилитами, то есть «полезными программами».

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
```

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:
 . . .
```

«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 руковод-

ства 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).
 . . .
```

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
```

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
```

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