Денис Колисниченко

Руководство по командам и shell-программированию в Linux

Санкт-Петербург

«БХВ-Петербург»

2011

УДК 681.3.068
ББК 32.973.26-018.1
К60

Колисниченко Д. Н.

К60 Руководство по командам и shell-программированию в Linux. — СПб.: БХВ-Петербург, 2011. — 288 с.: ил. — (БЛЦ)

ISBN 978-5-9775-0619-9

Рассмотрены команды Linux, основы работы в командной строке, а также настройка системы с помощью программ, обладающих только текстовым интерфейсом. Работа с системой выполняется только в режиме консоли, что требует определенной квалификации пользователя. Подробно описаны наиболее полезные команды Linux, особенности файловой системы Linux, системы инициализации, загрузчики GRUB и GRUB2. С позиции пользователя оценены интерактивные возможности оболочки zsh. Даны практические примеры разработки сценариев на языках оболочек bash и tcsh. Рассмотрено управление пакетами для наиболее актуальных на данный момент дистрибутивов. Для энтузиастов Linux написана отдельная глава о разработке собственного дистрибутива Linux и создании загрузочного LiveCD.

Для системных администраторов, программистов и квалифицированных пользователей Linux

УДК 681.3.068
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Григорий Добин*
Редактор	*Владимир Красовский*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Виктория Пиотровская*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 26.08.10.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 23,22.
Тираж 2000 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 г. выдано Федеральной службой по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12.

ISBN 978-5-9775-0619-9

Оглавление

Введение

Linux — особенная операционная система, и сейчас мы поговорим о ее особенностях. Начнем с залога популярности Linux — лицензии GPL, по которой и распространяется эта операционная система. Согласно GPL, Linux распространяется абсолютно свободно. Заметьте, я не сказал "бесплатно". Многие думают, что Linux — это бесплатная операционная система. Отчасти это так. Но главное то, что она свободная, т. е. всем желающим доступен исходный код ядра (как и любых других Linux-программ) и вы можете распространять без всяких ограничений как любой дистрибутив Linux, так и исходные коды программ. Вы можете установить Linux на любое количество компьютеров и даже создать свой дистрибутив Linux и распространять его под другим названием — главное, чтобы он распространялся под лицензией GPL.

Сравните это с лицензией на Windows, где вы имеете право установить приобретенный дистрибутив только на несколько компьютеров (количество указывается в лицензии) и где вы не можете распространять дистрибутив. А в мире Linux никаких ограничений нет. Скачали дистрибутив (необязательно даже покупать его в магазине), установили на любое количество компьютеров (пока "болванка" не испортится), затем передали диск друзьям (желательно, пока он не испортился) — пусть они его распространяют дальше. А фанаты Linux могут даже создавать собственные дистрибутивы — как с нуля, так и на базе одного из существующих дистрибутивов (что намного проще). Кстати, в *приложении* этой книги мы как раз и поговорим о создании собственного дистрибутива.

Теперь, думаю, вам стало ясно, почему Linux настолько популярна. Но свободное распространение — это далеко не все. Судите сами: если операционная система бесплатная, но не соответствует требованиям пользователей, то ее установят, попробуют поработать и на следующий день деинсталлируют.

С технической точки зрения можно выделить следующие особенности Linux.

- *Реальная многозадачность* — Linux, как и ее родственник UNIX, использует режим разделения времени центрального процессора (time-sharing system). Работает это так: планировщик выделяет каждому процессу фиксированный интервал для выполнения. По окончанию выделенного времени планировщик приостанавливает выполнение процесса и передает управление другому процессу.

А другие операционные системы используют режим *вытесняющей многозадачности*, когда процесс сам должен уступить место "под солнцем", т. е. сам приостановить свое выполнение и передать право на выполнение другому процессу. Все бы хорошо, но некоторые процессы могут "узурпировать" все процессорное время, и якобы многозадачная операционная система превращается в… однозадачную со всеми вытекающими последствиями.

- *Многопользовательский доступ* — Linux не только многозадачная, но и многопользовательская операционная система. Это означает, что в системе могут *одновременно* работать несколько пользователей. Как именно — это уже другой вопрос. Один пользователь может находиться непосредственно за компьютером, а остальные — подключены по сети. Но не забывайте, что Linux можно установить и на мейнфрейм — суперкомпьютер с несколькими терминалами. Правда, такие компьютеры постепенно отходят, а их место занимают так называемые тонкие клиенты, когда терминал подключен к компьютеру не физически, а по сети. Но в любом случае, несмотря на способ соединения, Linux является многопользовательской системой.
- *Страничная организация памяти* — память в Linux организована в виде страниц по 4 Кбайт каждая. Когда физическая оперативная память заканчивается, включается механизм подкачки и неиспользуемые данные сбрасываются в область подкачки на жесткий диск. Правда, такой организацией памяти и механизмом подкачки сейчас никого не удивишь — все современные операционные системы работают примерно так же.
- *Загрузка выполняемых модулей "по требованию"* — чтобы ядро системы поддерживало определенное устройство или функцию (например, протокол), нужно добавить программный код в состав ядра. Но ядро, поддерживающее все возможности и все устройства, будет просто огромным. В Linux эта проблема решается загрузкой модулей, которые добавляют поддержку определенных устройств. Например, вам нужна поддержка звуковой платы Creative — загрузите модуль, реализующий поддержку этой платы. При этом в память загружаются только те модули, которые необходимы для полноценной работы конкретной системы, что позволяет оптимизировать использование ресурсов компьютера.
- *Совместное использование исполняемых программ* — если несколько пользователей запустили одну и ту же программу, то в память загружается всего одна копия этой программы, а не несколько, что позволяет экономить оперативную память.
- *Общие библиотеки* — библиотеки содержат наборы процедур, которые используются различными приложениями. Вместо того чтобы скомпилировать в один исполнимый файл все процедуры, приложение использует библиотеку (которая также может использоваться другими приложениями), что позволяет существенно экономить место на диске.
- *Поддержка различных файловых систем* — Linux поддерживает много различных файловых систем, в том числе и файловые системы Windows. О поддержке файловых систем мы поговорим в *главе 4*.

- *Поддержка разных аппаратных платформ* — Linux может работать не только на платформах x86/x64. Поддерживаются аппаратные платформы ARM, DEC Alpha, SUN Sparc, M68000 (Atari и Amiga), MIPS, PowerPC и др.

Мы перечислили далеко не все технические особенности Linux. Полный список может занять еще несколько страниц, введение затянется и книга покажется вам скучной. А этого нельзя допустить, поскольку данная книга — не просто справочник по командам ОС, а нечто большее — учебник по командной строке.

Технические особенности — это прекрасно, но чем же Linux хорош для обычного пользователя? Начнем с безопасности. Linux не страшны обычные вирусы, которые постоянно поражают Windows-компьютеры и которых насчитывается несколько сотен тысяч. Для Linux создано не более 1000 вирусов, да и то встретить такой вирус — это что-то из области ненаучной фантастики. К тому же вирус может причинить ущерб системе, только если пользователь откровенно не соблюдает правила безопасности, например всегда работает под учетной записью root.

Если раньше у Linux были проблемы с русским языком и наличием пользовательских приложений, то сейчас все иначе. С русским, как и с другими языками, проблем нет, поскольку все современные дистрибутивы уже давно перешли на UTF-8. А пользовательских приложений (работающих в графическом режиме) тоже достаточно — медиапроигрыватели, офисные пакеты, браузеры, почтовые клиенты — в общем все, что нужно обычному пользователю. К тому же для Linux разработаны эмуляторы (`wine`, `cedega`), позволяющие запускать Windows-игры.

Со временем Linux становится все проще и проще в использовании. Помню, как устанавливал в 1999 году свой первый дистрибутив (Red Hat). Перечитал множество документации (документации по Red Hat не нашел, зато воспользовался руководством по установке Slackware, где была подробно описана программа `fdisk` — в Red Hat того времени при установке как раз и вызывался `fdisk`) и только потом приступил к установке. На все про все потратил весь рабочий день (это было первое знакомство с Linux), а настройку системы отложил на следующий день. Но настроить за один день не получилось — то нужно было настроить монитор, то русифицировать X Window, то устранить некоторые проблемы в GNOME, установить русские шрифты для принтера и т. д. Работы хватало. Чтобы получить полностью рабочую систему (учитывая, что Linux я видел в первый раз в жизни), мне понадобилось около недели.

Сейчас все иначе. На установку системы в зависимости от дистрибутива и производительности компьютера уходит 15–40 минут. После этого в большинстве случаев вы получаете полностью рабочую систему. Вам останется только настроить соединение с Интернетом и доустановить необходимые программы. Если вы знаете, что делаете, то все это займет еще час, пусть два. А если не знаете, максимум — 1 день. Установка программ тоже стала намного проще — вам больше не нужно вручную бороться с зависимостями, нужно только указать менеджеру пакетов, какой пакет вам нужно установить, — все дополнительные пакеты будут установлены автоматически.

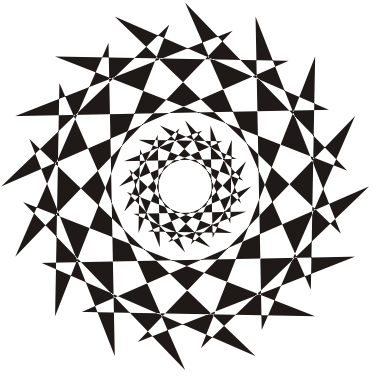
Конечно, есть частные случаи, когда не удается сразу заставить работать то или иное устройство, например Wi-Fi-адаптер. Но это частные случаи. Помню,

в Windows тоже пытался 4 часа настроить звуковую плату, причем драйвер у меня был...

В современных книгах по Linux все больше внимания уделяется графическим программам и графическому интерфейсу. Это правильно — нужно соответствовать времени, ведь Linux уже больше не воспринимается без графического интерфейса.

Но не нужно забывать, что изначально в Linux не было графического интерфейса. А была командная строка — вы вводили команду и получали результат выполнения (список файлов, список процессов, содержимое файла и т. д.). Даже когда появился графический интерфейс, многие пользователи предпочитали работать в командной строке. Сейчас все поменялось. Большинство пользователей работает исключительно в графическом интерфейсе, а некоторые даже не знают о существовании командной строки! Современные пользователи Linux уподобились Windows-пользователям, которые могут работать только с графическим интерфейсом и не знают ни одной команды операционной системы.

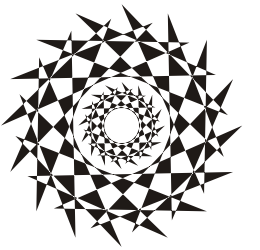
Эта книга посвящена командной строке Linux. В ней вообще не будут рассматриваться графические программы и графический интерфейс Linux. Только консоль, команды и файлы конфигурации — ничего больше. Зачем это нужно? А затем, что многие так называемые проблемы возникают от незнания и решаются вводом той или иной команды. Не нужно воспринимать данную книгу как сухой и скучный справочник по командам. Если вам нужен такой справочник, то вообще не нужно тратиться на покупку книги — он у вас всегда под рукой — это команда `man`. Данная книга — это учебник по использованию команд Linux. В ней вы найдете описание синтаксиса команд, описание параметров и, конечно же, практические примеры. Чувствую, введение затянулось, поэтому самое время приступить к чтению книги.



ЧАСТЬ I

Командная строка

Часть I полностью посвящена командной строке. Мы научимся правильно использовать командную строку, познакомимся с различными командными интерпретаторами, а также рассмотрим множество полезных команд Linux.



Глава 1

Введение в командную строку

1.1. Вход в систему

Настоящий линуксоид должен уметь работать в консоли. Ведь когда система Linux только появилась, существовала одна консоль, о графическом интерфейсе не было и речи. Знаете, почему UNIX и Linux отталкивали обычных пользователей? Потому что не было хорошего графического интерфейса. Раньше в Linux работали одни профессионалы. Сейчас все изменилось — в Linux очень удобный графический интерфейс, который с удовольствием используют и профессионалы (дождались наконец-то!), забывая о командной строке. Наш дистрибутив вообще ориентирован на работу в графическом режиме, а в официальных руководствах, которые можно найти в Интернете, о консоли вообще не упоминается. А ведь она есть! В этой главе мы поговорим о том, как правильно работать в консоли. Совсем необязательно работать полностью в текстовом режиме, вы можете использовать материал данной главы для эффективной работы с Терминалом — эмулятором консоли.

Обычные пользователи в консоль ни ногой — даже принципиально, мол, зачем возвращаться в DOS? Под "DOS" имелась в виду командная строка Linux. Да, ее вид не очень дружелюбный, но это только кажется. Стоит вам поработать в консоли, и вы поймете все ее преимущества. Начнем с того, что командная строка Linux намного удобнее командной строки DOS — об этом мы еще поговорим. В консоли можно выполнять те же операции, что и в графическом режиме, причем все намного быстрее. Хотите бороздить просторы Интернета? Пожалуйста, но без картинок. Не так красиво, но зато сэкономите трафик. А на обмен электронными сообщениями это никак не влияет. В консоли также можно работать и с документами, правда, тоже о графике можно забыть. Консоль позволяет эффективно использовать ресурсы старых компьютеров. Да, в графическом режиме на стареньком "Пентиуме" не поработаешь, зато в текстовом режиме его можно быстро превратить в очень полезный для всей сети компьютер — в шлюз, через который его более мощные собратья будут получать доступ к Интернету.

По умолчанию в современных дистрибутивах при входе в систему запускается графический менеджер регистрации (рис. 1.1). Однако из всех правил могут быть исключения. Пример тому дистрибутив Slackware — в нем сначала нужно выполнить вход в консоли (см. далее), а потом для запуска графического интерфейса ввести команду `startx`.

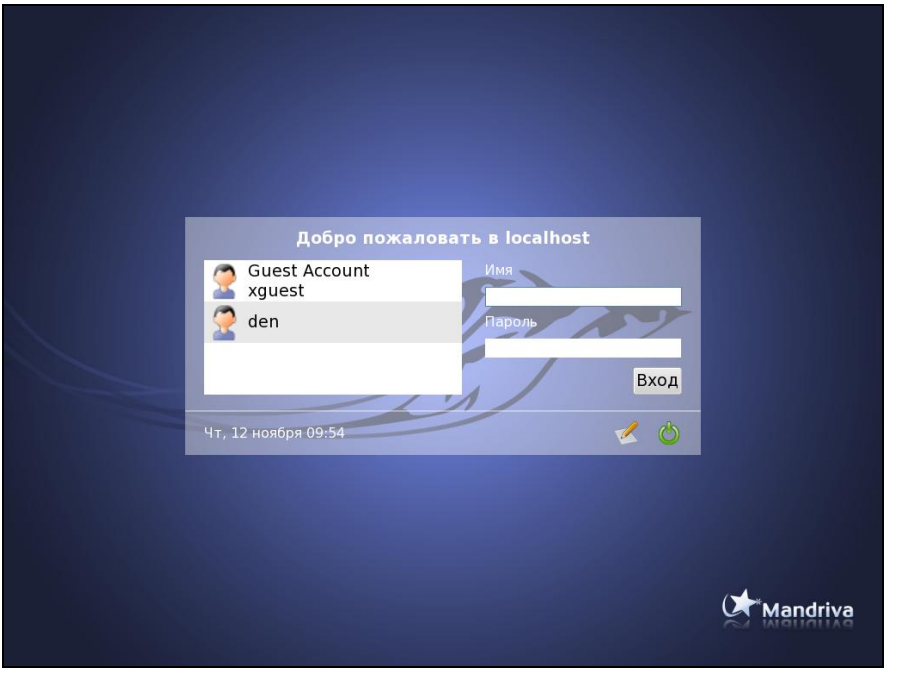


Рис. 1.1. Графический вход в систему (Mandriva 2010)

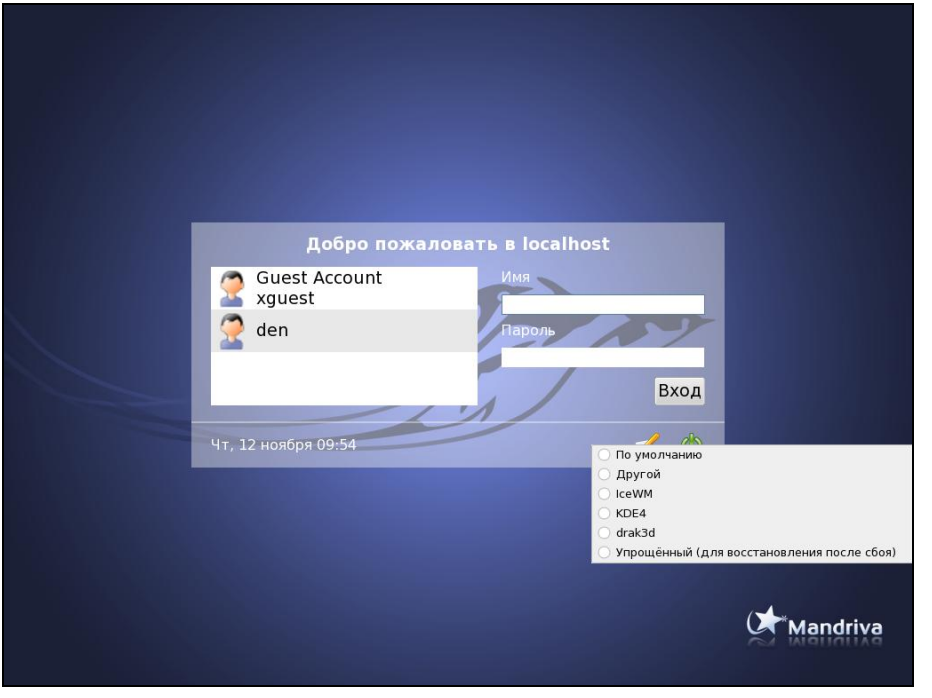


Рис. 1.2. Выбор типа сеанса (Mandriva 2010)

Для входа в систему вам нужно указать имя пользователя и пароль. После этого загрузится KDE или GNOME (в зависимости от того, какая графическая среда установлена в вашем дистрибутиве по умолчанию). Конечно, может быть загружена какая-то другая графическая среда, но обычно по умолчанию устанавливается KDE или GNOME. Для выбора графической среды нужно нажать кнопку **Тип сеанса** (или **Сеанс** — в Fedora и некоторых других дистрибутивах, а в некоторых дистрибутивах эта кнопка может быть представлена графической пиктограммой), как показано на рис. 1.2.

Сейчас вы находитесь в графическом режиме. Для того чтобы перейти из графического режима в консоль (рис. 1.3), нажмите клавиатурную комбинацию <Ctrl>+<Alt>+<F*n*>, где *n* — номер консоли (от 1 до 6). Чтобы перейти на первую консоль, нужно нажать комбинацию клавиш <Ctrl>+<Alt>+<F1>, на вторую — <Ctrl>+<Alt>+<F2> и т. д. Обратите внимание, что так можно перейти в консоль только из графического режима. Если вы уже находитесь в консоли, то для переключения между консолями служат комбинации клавиш <Alt>+<F1> ... <Alt>+<F6>, а также <Alt>+<F7> — для перехода в графический режим. Для лучшего запоминания эти комбинации клавиш приведены в табл. 1.1.

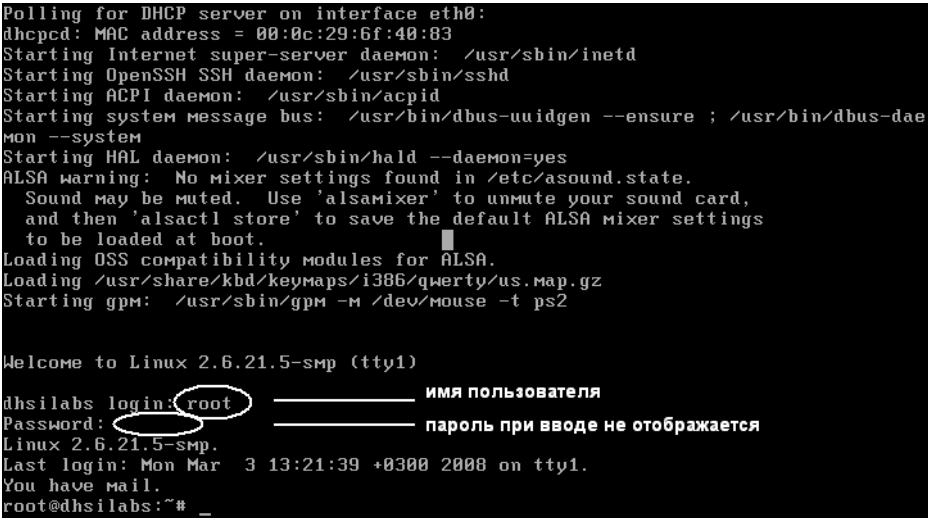


Рис. 1.3. Регистрация в консоли (Slackware)

Таблица 1.1. *Клавиши переключения между консолями и графическим режимом*

Комбинация клавиш	**Назначение**
<Ctrl>+<Alt>+<F*n*> (*n* от 1 до 6)	Переключение из графического режима в консоль с номером *n*
<Alt>+<F*n*> (*n* от 1 до 6)	Переключение между консолями
<Alt>+<F7>	Переключение из консоли в графический режим

1.2. Команды *poweroff*, *halt*, *reboot*, *shutdown*

Для выхода из консоли (чтобы ею никто не воспользовался во время вашего отсутствия) предусмотрена команда `logout`, она же команда `exit`.

Для перезагрузки компьютера существует команда `reboot`. Кроме нее вы можете использовать еще две команды — `halt` и `poweroff`:

- команда `halt` завершает работу системы, но не выключает питание. Вы увидите сообщение **`System is halted`**, свидетельствующее о возможности выключения питания. Эта команда предназначена для старых компьютеров, не поддерживающих расширенное управление питанием;
- команда `poweroff` завершает работу системы и выключает ее питание.

Самая "продвинутая" команда — `shutdown` — позволяет завершить работу и перезагрузить систему в назначенное время. Предположим, что вы хотите уйти пораньше, но компьютер нужно выключить ровно в 19:30 (вдруг некоторые пользователи задержались на работе, а вы выключите сервер, — некрасиво получится). Вот тут-то вам и поможет команда `shutdown`:

```
# shutdown -h 19:30 [сообщение]
```

> ***Примечание***
>
> Здесь и далее решетка (`#`) означает, что команда должна быть выполнена от имени пользователя root. Если перед командой ничего не указано или же указан символ доллара (`$`), команду можно выполнить от имени обычного пользователя.

Сообщение `[сообщение]` можно и не указывать, все равно Windows-пользователи его не увидят.

Если нужно завершить работу системы прямо сейчас, вместо времени укажите `now`:

```
# shutdown -h now
```

Для перезагрузки системы есть опция `-r`:

```
# shutdown -r now
```

1.3. Как работать в консоли

Работа в консоли заключается во вводе нужной команды. Вы вводите команду (например, создания каталога, просмотра файла, вызова редактора и т. д.) и нажимаете клавишу <Enter>. Команда содержит как минимум имя запускаемой программы. Кроме имени программы команда может содержать параметры, которые будут переданы программе, а также символы перенаправления ввода/вывода (об этом чуть позже). Естественно, вам нужно знать имя программы, а также параметры, которые нужно ей передать. Если вы помните название программы, а назначение параметров забыли, вспомнить поможет команда `man`. Man (от англ. *manual*) — это справочная система Linux. В ней есть информация о каждой программе, которая установлена в вашей системе. Как система знает все обо всех программах? Все очень просто. Разработчики программ под Linux договорились, что вместе с программой будет поставляться специальный man-файл — файл справочной системы.

Понятно, если разработчик не добросовестный, он может и не создать файл справочной системы, но это происходит очень редко. Чтобы получить справку по какой-нибудь программе, нужно ввести команду:

```
man имя_программы
```

Вы никак не можете запомнить, как пишется та или иная команда? Если вы помните хотя бы, на какую букву она начинается, то воспользуйтесь функцией автодополнения командной строки: введите первые буквы команды и нажмите клавишу <Tab>. При первом нажатии система попытается дополнить команду, если это возможно. Иногда дополнить команду невозможно. Например, вы ввели букву `a` и нажали клавишу <Tab>. Ясное дело, в системе есть несколько команд, которые начинаются на букву "a". Тогда система не дополнит командную строку. Если вы хотите просмотреть все командаы на букву "a", тогда нажмите еще раз клавишу <Tab>.

> ***Примечание***
>
> Описанная здесь функция автодополнения работает в командной оболочке bash (которая используется по умолчанию). В следующей главе будут рассмотрены особенности и других оболочек.

Вам лень писать (даже с автодополнением) длинные команды? Тогда можно создать псевдонимы команд. Для этого в файл .bash_profile добавьте строки вида:

```
alias псевдоним='команда'
```

Например:

```
alias cfg-net='system-config-network'
```

Для того чтобы изменения вступили в силу, выйдите из консоли (команда `logout`) и заново зарегистрируйтесь.

Пожалуй, для полноценной работы с консолью вам нужно знать еще одну команду — `clear`. Данная команда очищает консоль (терминал). Очень полезная команда, особенно когда вы хотите все начать с "чистого листа".

1.4. Графические терминалы

Понимаю, что большинство дистрибутивов оснащены графическим интерфейсом, который к тому же запускается по умолчанию. Поэтому большинство пользователей не будут жертвовать удобным и привычным интерфейсом ради консоли.

Вместо того чтобы переключиться в консоль, можно использовать терминалы — эмуляторы консоли. Терминал — это графическая программа (рис. 1.4), в окне которой вы можете вводить команды и видеть результат их выполнения. Запустить терминал можно через меню GNOME/KDE (**Система | Стандартные | Терминал** или **Система | Системные | Терминал** — в зависимости от дистрибутива).

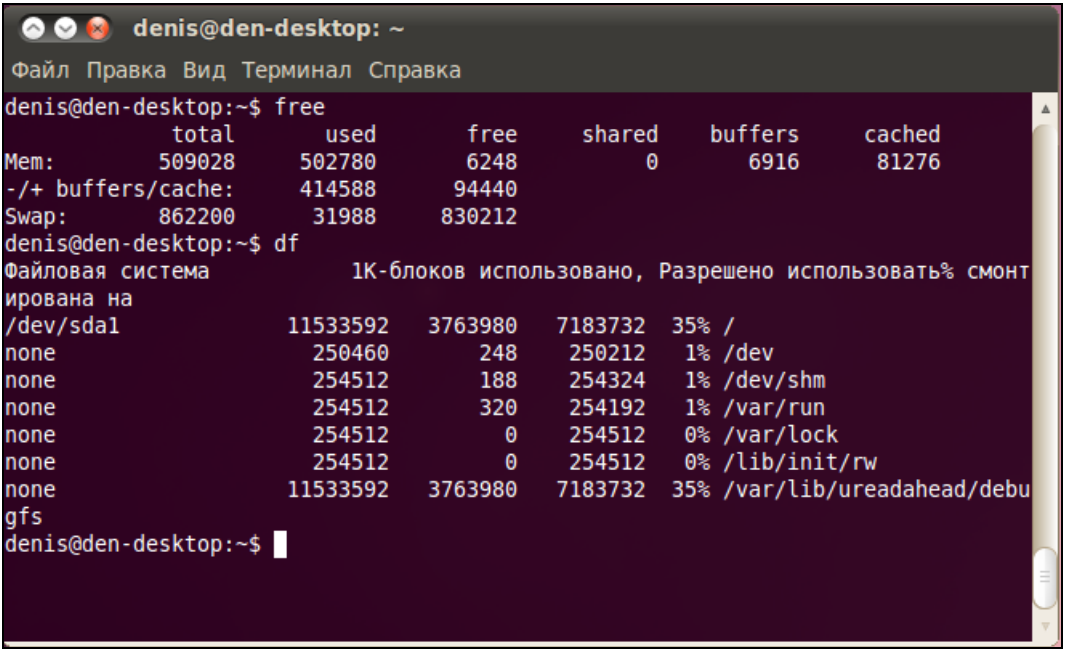
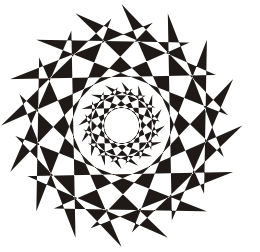


Рис. 1.4. Терминал



Глава 2

Командные интерпретаторы

2.1. Файл /etc/shells

По умолчанию во всех современных дистрибутивах используется командный интерпретатор `bash`. Основное предназначение `bash`, как и любой другой оболочки, — выполнение команд, введенных пользователем. Пользователь вводит команду, `bash` ищет программу, соответствующую команде, в каталогах, указанных в переменной окружения `PATH`. Если такая программа найдена, то `bash` запускает ее и передает введенные пользователем параметры. В противном случае выводится сообщение о невозможности выполнения команды.

Кроме `bash` существуют и другие оболочки — `sh`, `csh`, `ksh`, `zsh` и пр. Все командные оболочки, установленные в системе, прописаны в файле /etc/shells. Список оболочек может быть довольно длинным. В листинге 2.1 представлен файл /etc/shells дистрибутива Fedora (установка по умолчанию).

Листинг 2.1. Файл /etc/shells дистрибутива Fedora

```
/bin/ash
/bin/bash
/bin/csh
/bin/false
/bin/ksh
/bin/sh
/bin/tcsh
/bin/true
/bin/zsh
/usr/bin/csh
/usr/bin/ksh
/usr/bin/bash
/usr/bin/tcsh
/usr/bin/zsh
```

С точки зрения пользователя указанные оболочки мало чем отличаются. Все они позволяют выполнять введенные пользователем команды. Но оболочки используются

не только для выполнения команд, а еще и для автоматизации задач с помощью *сценариев*. Так вот, все эти оболочки отличаются синтаксисом языка описания сценариев.

Примечание

В листинге 2.1 программы /bin/false и /bin/true не являются оболочками. Это "заглушки", которые можно использовать, если вы хотите отключить ту или иную учетную запись пользователя. При входе пользователя в систему запускается установленная для него оболочка. Для каждого пользователя имеется возможность задать свою оболочку (изменить оболочку пользователь может самостоятельно командой `chsh`). Так вот, если для пользователя задать оболочку /bin/false (или /bin/true), он не сможет войти в систему. Точнее, он войдет в систему, но и сразу выйдет из нее, поскольку обе "заглушки" ничего не делают, а просто возвращают значение 0 (для false) или 1 (для true). Сессия же пользователя длится до завершения работы его оболочки.

2.2. Оболочка *sh*

Самым первым командным интерпретатором (оболочкой) в операционной системе UNIX (да, именно UNIX, поскольку корни Linux уходят в далекие 70-е годы) была `sh` (сокращение от *shell*). Данная оболочка до сих пор используется в современных версиях Linux (и FreeBSD).

Оболочка `sh` была разработана Стивеном Борном (Steve Bourne), поэтому ее второе название — Bourne Shell. Изначально `sh` была разработана для операционной системы AT&T (разработка Bell Labs). Чуть позже `sh` была усовершенствована и вошла в состав POSIX (Portable Operating System Interface for UNIX — Переносимый интерфейс операционных систем UNIX). Усовершенствованная версия `sh` до сих пор устанавливается (но не используется по умолчанию) в современных версиях FreeBSD.

С точки зрения пользователя оболочка `sh` не очень удобна, поэтому пользователи предпочитают другие оболочки, например `tcsh` или `bash`.

2.3. Оболочка *csh*

Оболочка `csh` (C Shell) по умолчанию используется в FreeBSD. Разработка `csh` началась еще в первых версиях BSD (Linux будет создан лет через 15). Тогда в институте Беркли начали создавать новую оболочку (`csh`), потому что не захотели мириться с ограничениями `sh`.

Внутренний синтаксис `csh` очень напоминает язык программирования C, поэтому он должен был понравиться программистам (а в то время все пользователи компьютеров являлись программистами). Хотя сами программисты отмечали, что синтаксис не очень удобен, даже несмотря на то, что он похож на C.

По сравнению с `sh`, у `csh` есть множество преимуществ: она умеет управлять заданиями, хранит историю ранее введенных команд, а также у `csh` есть сценарии, которые выполняются при входе пользователя (запуске оболочки) и при выходе

пользователя (когда пользователь вводит команду `exit`). В то время у `sh` не было таких сценариев, которые оказались очень удобными.

С точки зрения обычного использования оболочки (а не программирования) `csh` тоже была на высоте.

В последних версиях FreeBSD и Linux вместо `csh` используется ее усовершенствованная версия `tcsh`, а файл /bin/csh — это просто ссылка на /bin/tcsh.

2.4. Оболочка *ksh*

Не хочется делать экскурс в историю UNIX, но пару слов сказать все же придется. Изначально система UNIX появилась в лабораториях компании AT&T, позже появились версии UNIX института Беркли (операционная система называлась BSD). Так уж сложилось исторически, что AT&T и институт Беркли постоянно конкурировали между собой. Как только в Беркли разработали оболочку `csh`, в AT&T принялись разрабатывать собственную оболочку, которая получила название ksh (Korn Shell) — по имени разработчика Дэвида Корна (David Korn).

Оболочка `ksh` по функциям похожа на `csh`: есть поддержка управления заданиями, история команд, позволяет назначать командам псевдонимы, а также создавать конфигурационные файлы для подоболочек.

Несмотря на то что оболочка была разработана в 1986 году, она до сих пор используется в некоторых версиях UNIX по умолчанию, а также устанавливается по умолчанию во всех дистрибутивах Linux (но не используется по умолчанию). Правда, изначально `ksh` — это коммерческий продукт, поэтому в FreeBSD и Linux используется не `ksh`, а ее бесплатная версия — `pdksh`, но для краткости исполнимый файл называется `ksh`.

Начинающим пользователям `ksh` не понравится (лучше использовать `bash`) — она слишком неудобна в использовании, зато у нее довольно развитый синтаксис внутреннего языка, что понравится программистам.

2.5. Оболочка *bash*

Командный интерпретатор `bash` (Bourne Again Shell) был разработан фондом свободного программного обеспечения (Free Software Foundation, FSF). За основу была взята оболочка `sh`. Оболочка стала очень популярной и сейчас используется по умолчанию во всех дистрибутивах Linux.

Оболочка `bash` может использоваться также и для запуска сценариев `sh`, поэтому `sh` во многих системах уже не устанавливается, а файл /bin/sh — это ссылка на /bin/bash.

С точки зрения пользователей `bash` намного удобнее, чем `ksh`. Вы можете легко редактировать командную строку, просматривать историю команд, создавать псевдонимы команд, создавать переменные окружения и использовать их в собственных сценариях. Как и в `csh`, в `bash` есть сценарии, которые вызываются при запуске оболочки и при выходе из нее.

Синтаксис `bash` довольно прост, поэтому бóльшая часть сценариев, разрабатываемых в Linux, пишется именно на `bash`.

2.6. Оболочка *zsh*

Оболочки `bash` и `tcsh` (современная версия `csh`) будут рассмотрены в *части III*, оболочка `ksh` используется редко. Поэтому сейчас мы поближе познакомимся с оболочкой `zsh`, которая становится все более популярной.

До того как я не познакомился с `zsh`, я считал самой удобной оболочку `bash`. Однако это не так.

Что же удобного в `zsh`? Во-первых, навигация. В `bash` для перехода в каталог /dir/subdir1/subdir2 нужно ввести команду:

```
cd /dir/subdir1/subdir2
```

Можно использовать автодополнение `bash` — вводить начальные символы каталога и нажимать клавишу <Tab>. Это будет выглядеть примерно так:

```
cd /dir/sub [Tab]/subdi [Tab]
```

В `zsh` можно ввести:

```
/d/s/s
```

Затем нажать клавишу <Tab> — вы перейдете в нужный каталог. Например, для перехода в /etc/sysconfig/network, нужно ввести `/e/s/n` и нажать клавишу <Tab>. Кстати, команда `cd` уже не нужна.

Покажу еще один трюк. Предположим, у нас есть каталог files, а в нем есть каталоги f1 и f2. Внутри каждого каталога f* есть каталоги source и last. То есть структура каталогов будет примерно такой:

/files/f1/sources/last

/files/f2/sources/last

Пусть мы находимся в каталоге /files/f1/sources/last, для перехода в каталог /files/f2/sources/last введите команду:

```
cd 1 2
```

Но одной лишь навигацией возможности `zsh` не ограничиваются. Можно, например, использовать вот такое перенаправление:

```
< /var/log/messages
```

Оболочка запустит программу, указанную в переменной `$PAGER`. В большинстве случаев это аналогично команде:

```
cat /var/log/messages | less
```

Все возможности `zsh` в этой главе мы рассматривать не будем — их намного больше, чем вам кажется. Если вы заинтересовались, то прочитайте следующие страницы:

- **http://opennet.ru/base/dev/zsh_intro.txt.html**;
- **http://citkit.ru/articles/1083/**;
- **http://alexott.net/ru/writings/zsh/index.html**;
- **http://habrahabr.ru/blogs/linux/82537/**.

2.7. Оболочка *tcsh*

Оболочка `tcsh` является модифицированной версией `csh`. Буква t в названии означает TENEX: изначально оболочка была разработана для операционной системы TENEX (использовалась в далеком прошлом на компьютерах DEC PDP-10).

В `tcsh` усовершенствована фкнкция редактирования командной строки, есть автозавершение команд (как в `bash`). Кроме того, `tcsh` может распознавать потенциально опасные команды. Если вы от имени root попытаетесь удалить все файлы, оболочка потребует подтверждения.

Оболочка `tcsh` очень удобна в использовании, но ее синтаксис сценариев сложнее, чем у `bash`. Однако в *части III* мы все же рассмотрим разработку сценариев на `tcsh`, чтобы вы смогли оценить сложность создания разработки сценариев на `bash` и на `tcsh`.

2.8. Оболочка *ash*

Almquist shell (`ash`) — самая простая командная оболочка. Это самая маленькая оболочка, доступная для UNIX (у нее самые низкие требования к дисковому пространству).

У `ash` всего 24 встроенных команд и 10 опций командной строки. Обычно `ash` используется при загрузке Linux в однопользовательском режиме (или в режиме восстановления).

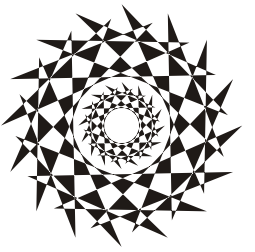
Оболочка `ash` совместима с `sh`, с ее помощью можно проверить сценарии на совместимость с традиционным синтаксисом `sh`. А в операционной системе NetBSD оболочка `ash` используется вместо /bin/sh.

2.9. Выбор оболочки

Какую оболочку выбрать? Первым делом нужно оценить простоту использования оболочки. Ведь вы будете использовать эту оболочку каждый день, поэтому простота использования должна быть на первом месте.

Затем нужно оценить простоту синтаксиса оболочки. Конечно, это только в том случае, если вы планируете разрабатывать собственные сценарии. Также не нужно забывать, что вы можете использовать одну оболочку, а разрабатывать сценарии — на языке другой оболочки. Например, в повседневной работе вы можете использовать `zsh`, а разрабатывать сценарии на языке `bash`.

Довольно удобны в использовании оболочки `bash`, `tcsh` и `zsh`. Скорее всего, вы выберете одну из них. А вот для программирования вы будете использовать или `bash`, или `tcsh` (синтаксис `zsh` не очень понятен).



Глава 3

Базовые команды Linux

3.1. О командах Linux

Все команды Linux можно условно разделить на несколько групп:

- *команды общего назначения* — эти команды могут понадобиться в любой момент как при работе в консоли, так и при написании собственных сценариев;
- *команды для работы с файлами и каталогами* — эти команды будут рассмотрены в *главе 4* вместе с основами файловой системы Linux, без которых данные команды не будут понятны читателю;
- *команды обработки текста* — будут рассмотрены в *главе 7*;
- *команды для работы с сетью и Интернетом* — выйти в Интернет в Linux можно даже без запуска графического интерфейса, что и будет показано в *главе 8*;
- *команды системного администратора* — любой системный администратор просто обязан знать команды, представленные в *главе 9*.

Некоторые команды могут относиться к одной из групп, но в этой книге выделены в специальную главу, поскольку заслуживают отдельного разговора. Например, команды управления пользователями и группами выделены в *главу 12*. Можно было бы просто упомянуть команды `adduser` и `passwd` в *главе 9*, но в *главе 12* помимо рассмотрения формата важных конфигурационных файлов приводится описание множества дополнительных команд, так или иначе связанных с пользователями и группами пользователей.

В этой главе будут рассмотрены базовые команды Linux, т. е. команды общего назначения.

3.2. Команда *arch*: вывод архитектуры компьютера

Данная команда поможет узнать тип аппаратной платформы, например: i386, i586, i686 и др.

Пример использования:

```
$ arch
i686
```

3.3. Команда *banner*: текстовый баннер

Команда `banner` выводит строку (максимальная длина — 10 символов), рисуя буквы символом звездочки (`*`). Данную команду можно использовать в своих сценариях для вывода названия сценария. Пример использования:

```
$ banner Denix
```

3.4. Команда *chsh*: изменение командного интерпретатора

Команда `chsh` позволяет изменить командный интерпретатор, вывести список установленных интерпретаторов, а также установить командный интерпретатор по умолчанию. Синтаксис вызова программы:

```
$ chsh [параметры] интерпретатор
```

В качестве параметров вы можете передать:

- `-L` — выводит список установленных командных интерпретаторов (список хранится в файле /etc/shells);
- `-s` — устанавливает командный интерпретатор по умолчанию.

Примеры использования команды:

```
$ chsh zsh
$ chsh -s zsh
```

Первая команда изменяет текущий интерпретатор команд на оболочку `zsh`. Как только пользователь выйдет из системы и снова зайдет, будет запущен его интерпретатор по умолчанию (который установлен в файле /etc/passwd). Вторая команда устанавливает командный интерпретатор по умолчанию для текущего пользователя.

3.5. Команда *cksum*: вычисление контрольной суммы файла

Команда `cksum` вычисляет контрольную сумму (CRC) указанных файлов. Формат вызова:

```
$ cksum файлы
```

Пример:

```
$ cksum file1.txt file2.txt
```

3.6. Команда *clear*: очистка экрана

Команда `clear` очищает экран при работе в консоли (терминале).
Пример использования:

```
$ clear
```

3.7. Команда *date*: вывод даты и времени

Команда `date` относится как к командам общего назначения, так и к командам системного администратора. Обычные пользователи могут только просматривать дату и время в заданном формате, а пользователь с правами root может еще и устанавливать дату и время. Об установке даты и времени мы поговорим в *главе 9*, а сейчас разберемся с выводом даты и времени.

Команда `date` (без параметров) просто выводит текущую дату и время. Но вы можете уточнить формат вывода даты и времени так:

```
$ date +формат
```

Строка `формат` может состоять из модификаторов, указанных в табл. 3.1.

Таблица 3.1. *Модификаторы даты и времени*

Модификатор	Описание
`%%`	Знак %
`%a`	Сокращенное название дня недели (например, Вск)
`%A`	Полное название дня недели (например, Вторник)
`%b`	Сокращенное название месяца (например, Мар)
`%B`	Местное полное название месяца (напр., Март)
`%c`	Дата и время (в формате, заданном в настройках системы)
`%C`	Век
`%d`	День месяца (с предшествующим нулем, например, 07)
`%D`	Дата (в формате %m/%d/%y)
`%F`	Полная дата (в формате %Y-%m-%d)
`%g`	Последние две цифры года, соответствующего номеру недели в году согласно ISO 8601
`%G`	Год, соответствующий номеру недели в году согласно ISO 8601
`%H`	Час (00..23)
`%I`	Час (01..12)
`%j`	Номер дня в году
`%k`	Час (0..23)
`%l`	Час (1..12)
`%m`	Месяц (01..12)
`%M`	Минута (00..59)
`%n`	Вставляет разрыв строки
`%r`	12-часовое время (например, 12:12:07 PM)
`%R`	24-часовой формат часов и минут

Таблица 3.1 *(окончание)*

Модификатор	Описание
`%s`	Число секунд, прошедших с 1970-01-01 00:00:00 UTC (это так называемый timestamp)
`%S`	Секунда
`%t`	Вставляет символ табуляции
`%T`	Время (в формате %H:%M:%S)
`%U`	Номер недели в году (неделя начинается с воскресенья)
`%V`	Номер недели в году (неделя начинается с понедельника)
`%w`	День недели (0..6, где 0 — воскресенье)
`%y`	Последние две цифры года (00..99)
`%Y`	Год (все четыре цифры, например 2010)
`%z`	Часовой пояс в формате '+ччмм' (например, +0200)
`%:z`	Часовой пояс в формате '+чч:мм' (например, +02:00)
`%Z`	Алфавитное сокращение часового пояса (например, EST)

3.8. Команда *echo*: вывод сообщения

Команда `echo` выводит текстовую строку, указанную в качестве аргумента, например:

```
$ echo "Hello world!"
Hello world!
```

Обычно данная команда используется в сценариях командного интерпретатора для вывода сообщений на экран.

3.9. Команда *exit*: выход из системы

Для завершения сеанса работы в системе (при условии, что вы работаете в консоли) нужно использовать команду `exit`. Если не завершить сеанс работы, кто угодно сможет работать в системе под вашим именем (понятно, что во время вашего отсутствия за компьютером).

3.10. Команда *env*: установка переменных окружения

Команда `env` используется для установки переменных окружения во время выполнения команды, например ее можно использовать для установки переменной