

Командная строка Linux и автоматизация рутинных задач

2-е издание

Ядро 3.x

Оболочки `bash`, `tcsh`, `zsh`

Система инициализации `systemd`

Автоматизация на языках оболочек `bash` и `tcsh`

Особенности файловой системы Linux

Криптографическая файловая система `eCryptfs`

Команды системного администратора

Команды для обработки текста и для работы в сети

Загрузчики `GRUB` и `GRUB2`

Управление пакетами. Создание `RPM`-пакетов

Конфигурационные файлы сети
Настройка сети вручную

Краткое руководство по созданию собственного дистрибутива

УДК 004.4
ББК 32.973.26-018.2
К60

Колисниченко Д. Н.

К60 Командная строка Linux и автоматизация рутинных задач. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2014. — 368 с.: ил. — (Системный администратор)

ISBN 978-5-9775-3319-5

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, криптографическая файловая система eCryptfs, система инициализации systemd, загрузчики GRUB и GRUB2, ядро 3.x. 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.

Во втором издании полностью переработан материал по созданию собственных RPM-пакетов, настройке сети и Интернета, появилось описание псевдофайловой системы /proc и ряда полезных утилит: chage, pwck, grpck, groupmod, groupdel, sed, dd и др.

*Для системных администраторов, программистов
и квалифицированных пользователей Linux*

УДК 004.4
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Евгений Рыбаков</i>
Зав. редакцией	<i>Екатерина Капалыгина</i>
Редактор	<i>Григорий Добин</i>
Компьютерная верстка	<i>Ольги Сергиенко</i>
Корректор	<i>Зинаида Дмитриева</i>
Дизайн серии	<i>Инны Тачиной</i>
Оформление обложки	<i>Марины Дамбиевой</i>

Подписано в печать 31.03.14.
Формат 70×100^{1/16}. Печать офсетная. Усл. печ. л. 29,67.
Тираж 1500 экз. Заказ №
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

Первая Академическая типография "Наука"
199034, Санкт-Петербург, 9 линия, 12/28

ISBN 978-5-9775-3319-5

© Колисниченко Д. Н., 2014
© Оформление, издательство "БХВ-Петербург", 2014

Оглавление

Введение	13
Что нового во втором издании?	16
ЧАСТЬ I. КОМАНДНАЯ СТРОКА	19
Глава 1. Введение в командную строку	21
1.1. Вход в систему	21
1.2. Команды <i>poweroff</i> , <i>halt</i> , <i>reboot</i> , <i>shutdown</i>	24
1.3. Как работать в консоли	24
1.4. Графические терминалы	25
Глава 2. Командные интерпретаторы	27
2.1. Файл <i>/etc/shells</i>	27
2.2. Оболочка <i>sh</i>	28
2.3. Оболочка <i>csh</i>	28
2.4. Оболочка <i>ksh</i>	29
2.5. Оболочка <i>bash</i>	29
2.6. Оболочка <i>zsh</i>	30
2.7. Оболочка <i>tsh</i>	31
2.8. Оболочка <i>ash</i>	31
2.9. Выбор оболочки	31
Глава 3. Базовые команды Linux	33
3.1. О командах Linux	33
3.2. Команда <i>arch</i> : сведения об архитектуре компьютера	33
3.3. Команда <i>banner</i> : текстовый баннер	34
3.4. Команда <i>chsh</i> : изменение командного интерпретатора	34
3.5. Команды <i>cksum</i> и <i>md5sum</i> : вычисление контрольной суммы файла	35
3.6. Команда <i>clear</i> : очистка экрана	36
3.7. Команда <i>date</i> : вывод даты и времени	36
3.8. Команда <i>echo</i> : вывод сообщения	37
3.9. Команда <i>exit</i> : выход из системы	37
3.10. Команда <i>env</i> : установка переменных окружения	38
3.11. Команды <i>man</i> и <i>info</i> : вывод справки	38

3.12. Команда <i>printenv</i> : вывод значения переменной окружения.....	38
3.13. Команда <i>reset</i> : сброс терминала	38
3.14. Команда <i>sleep</i> : пора спать	39
3.15. Команда <i>startx</i> : запуск графического интерфейса X.Org	39
3.16. Команда <i>tee</i> : перенаправление ввода	39
3.17. Команда <i>true</i> : успешное завершение.....	39
3.18. Команда <i>yes</i> : возвращает у.....	40
3.19. Команда <i>uname</i> : название и версия дистрибутива Linux.....	40
3.20. Конвертирование кодировок, звуковых и графических файлов.....	40

Глава 4. Файловая система. Команды для работы с файловой системой 43

4.1. Файловые системы, поддерживаемые Linux.....	43
4.1.1. Выбор файловой системы.....	45
4.1.2. Linux и файловые системы Windows	46
4.1.3. Сменные носители.....	46
4.2. Особенности файловой системы Linux.....	46
4.2.1. Имена файлов в Linux	46
4.2.2. Файлы и устройства.....	47
4.2.3. Корневая файловая система и монтирование.....	48
4.2.4. Стандартные каталоги Linux.....	49
4.3. Команды для работы с файлами и каталогами.....	50
4.3.1. Работа с файлами.....	50
4.3.2. Работа с каталогами.....	52
4.4. Команда <i>ln</i> : создание ссылок.....	54
4.5. Команды <i>chown</i> , <i>chmod</i> и <i>chattr</i>	55
4.5.1. Команда <i>chmod</i> : права доступа к файлам и каталогам	55
4.5.2. Команда <i>chown</i> : смена владельца файла.....	57
4.5.3. Специальные права доступа (SUID и SGID)	57
4.5.4. Команда <i>chattr</i> : атрибуты файла, запрет изменения файла.....	57
4.6. Монтирование файловых систем	59
4.6.1. Команды <i>mount</i> и <i>umount</i>	59
4.6.2. Файлы устройств и монтирование	60
Жесткие диски.....	60
Приводы оптических дисков.....	62
Дискеты.....	62
Флешки и внешние жесткие диски.....	62
4.6.3. Опции монтирования файловых систем	63
4.6.4. Монтирование разделов при загрузке.....	64
4.6.5. Подробно о UUID и файле <i>/etc/fstab</i>	66
4.6.6. Монтирование Flash-дисков	68
4.7. Настройка журнала файловой системы <i>ext3</i>	69
4.8. Файловая система <i>ext4</i>	70
4.8.1. Сравнение <i>ext3</i> и <i>ext4</i>	70
4.8.2. Совместимость с <i>ext3</i>	72
4.8.3. Переход на <i>ext4</i>	72
4.9. Особые команды	73
4.9.1. Команда <i>mkfs</i> : создание файловой системы	73
4.9.2. Команда <i>fsck</i> : проверка и восстановление файловой системы	73

4.9.3. Команда <i>chroot</i> : смена корневой файловой системы	74
4.9.4. Установка скорости CD/DVD	74
4.9.5. Монтирование каталога к каталогу	74
4.9.6. Команды поиска файлов	74
4.10. Многофункциональная команда <i>dd</i>	76
4.10.1. Копирование файлов с помощью команды <i>dd</i>	76
4.10.2. Разделение файла на несколько частей	77
4.10.3. Создание резервной копии жесткого диска	78
4.10.4. Создание архива с резервной копией всего жесткого диска	78
4.10.5. Уничтожение всех данных раздела жесткого диска	78
4.11. <i>eStorfs</i> : краткое руководство для секретного агента	79
Глава 5. Процессы	82
5.1. Команды <i>kill</i> , <i>killall</i> , <i>xkill</i> и <i>ps</i>	82
5.2. Программа <i>top</i> : кто больше всех расходует процессорное время?	84
5.3. Команды <i>nice</i> и <i>renice</i> : изменение приоритета процесса	86
5.4. Перенаправление ввода/вывода	86
5.5. Команда <i>fuser</i> : кто открыл ресурс?	87
Глава 6. Запись CD/DVD из консоли	88
6.1. Команда <i>dd</i> : создание образа диска	88
6.2. Команды <i>cdrecord</i> и <i>dvdrecord</i> : запись образа на болванку	89
6.3. Команды очистки перезаписываемых дисков	90
6.4. Команда <i>mkisofs</i> : создание ISO-образа	90
6.5. Преобразование образов дисков	91
6.6. Создание и монтирование файлов с файловой системой	91
Глава 7. Команды для работы с текстом	92
7.1. Команда <i>cmp</i> : сравнение двух файлов	92
7.2. Команда <i>column</i> : разбивка текста на столбцы	92
7.3. Команда <i>comm</i> : еще одна команда для сравнения файлов	93
7.4. Команда <i>diff</i> : сравнение файлов	93
7.5. Команда <i>diff3</i> : сравнение трех файлов	94
7.6. Команда <i>egrep</i> : расширенный текстовый фильтр	95
7.7. Команда <i>expand</i> : замена символов табуляции пробелами	96
7.8. Команда <i>fmt</i>	96
7.9. Команда <i>fold</i>	97
7.10. Команда <i>grep</i> : текстовый фильтр	97
7.11. Команды <i>more</i> и <i>less</i> : постраничный вывод	97
7.12. Команды <i>head</i> и <i>tail</i> : вывод начала и хвоста файла	97
7.13. Команда <i>look</i>	98
7.14. Команда <i>sort</i> : сортировка файлов	98
7.15. Команда <i>split</i> : разбиение файлов на несколько частей	98
7.16. Команда <i>unexpand</i> : замена пробелов символами табуляции	99
7.17. Команды <i>vi</i> , <i>nano</i> , <i>ee</i> , <i>mcedit</i> , <i>pico</i> : текстовые редакторы	99
7.18. Команда <i>wc</i> : подсчет слов в файле	103
7.19. Некоторые команды преобразования символов и форматов	104
7.20. Команда <i>sed</i> : потоковый текстовый редактор	104

Глава 8. Команды для работы с сетью и Интернетом	106
8.1. Команда <i>ifconfig</i> : управление сетевыми интерфейсами	106
8.2. Маршрутизация.....	108
8.2.1. Команда <i>netstat</i> : просмотр таблицы маршрутизации	108
8.2.2. Команда <i>route</i> : изменение таблицы маршрутизации	112
8.3. Команда <i>pppoeconf</i> : настройка DSL-соединения	114
8.4. Команда <i>pppconfig</i> : настройка модемного (PPP) соединения.....	118
8.5. Команда <i>wvdial</i> : настройка PPP-соединения.....	119
8.6. Текстовые браузеры	121
8.7. Команда <i>ftp</i> : FTP-клиент	121
8.8. Команда <i>wget</i> : загрузка файлов	122
8.9. Команды для диагностики сети	124
8.10. Настройка сети вручную. Конфигурационные файлы	128
8.10.1. Конфигурационные файлы Fedora	130
8.10.2. Конфигурационные файлы openSUSE	132
8.10.3. Конфигурационные файлы Debian/Ubuntu	134
8.11. Команда <i>mii-tool</i>	134
8.12. Сетевой сканер <i>nmap</i>	135
8.12.1. Что такое <i>nmap</i> ?.....	135
8.12.2. Где мне взять <i>nmap</i> ?	136
8.12.3. Примеры использования <i>nmap</i>	136
Глава 9. Команды системного администратора	139
9.1. Программы разметки диска	139
9.1.1. Программа <i>fdisk</i>	139
9.1.2. Программа <i>parted</i>	142
9.1.3. Введение в GPT. Утилиты для работы с GPT	146
9.2. Информация о системе и пользователях	147
9.2.1. Команда <i>uptime</i> : информация о работе системы.....	147
9.2.2. Команда <i>users</i> : информация о пользователях	147
9.2.3. Команды <i>w</i> , <i>who</i> , <i>ftpwho</i> и <i>whoami</i> : информация о пользователях	148
9.2.4. Мониторинг работы системы	148
9.3. Планировщик <i>at</i>	151
9.3.1. Команда <i>at</i> : добавление задания.....	151
9.3.2. Команды <i>atq</i> и <i>atrm</i> : очередь заданий и удаление задания	151
9.4. Планировщик <i>cron</i>	151
9.5. Планировщик <i>anacron</i>	153
9.6. Команда <i>date</i> : вывод и установка даты и времени	154
9.7. Команды <i>free</i> и <i>df</i> : информация о системных ресурсах.....	154
9.8. Команда <i>ssh</i> : удаленный вход в систему	155
9.9. Устройства и драйверы	156
9.10. Псевдофайловая система <i>/proc</i>	159
9.10.1. Информационные файлы	160
9.10.2. Файлы, позволяющие изменять параметры ядра.....	160
9.10.3. Файлы, изменяющие параметры сети.....	161
9.10.4. Файлы, изменяющие параметры виртуальной памяти.....	162
9.10.5. Файлы, позволяющие изменить параметры файловых систем.....	162
9.10.6. Как сохранить изменения.....	163

ЧАСТЬ II. ОПЕРАЦИОННАЯ СИСТЕМА	165
Глава 10. Загрузчики Linux.....	167
10.1. Основные загрузчики	167
10.2. Конфигурационные файлы GRUB и GRUB2	168
10.2.1. Конфигурационный файл GRUB.....	168
10.2.2. Конфигурационный файл GRUB2. Команды <i>grub-mkconfig</i> и <i>update-grub</i>	171
10.3. Команды установки загрузчиков.....	175
10.4. Установка тайм-аута выбора операционной системы. Редактирование параметров ядра.....	175
10.5. Установка собственного фона загрузчиков GRUB и GRUB2.....	178
10.6. Постоянные имена и GRUB.....	179
10.7. Восстановление загрузчика GRUB/GRUB2	180
10.8. Две и более ОС Linux на одном компьютере	181
10.9. Загрузка с ISO-образов.....	182
10.10. Установка пароля загрузчика	183
10.10.1. Загрузчик GRUB.....	183
10.10.2. Загрузчик GRUB2.....	185
Глава 11. Системы инициализации Linux.....	187
11.1. Начальная загрузка Linux.....	187
11.2. Система инициализации <i>init</i>	189
11.2.1. Команда <i>init</i>	190
11.2.2. Команда <i>service</i>	191
11.2.3. Редакторы уровней запуска	191
11.2.4. Параллельная загрузка сервисов, или как сделать старую систему <i>init</i> быстрее.....	192
11.3. Система инициализации <i>upstart</i>	193
11.3.1. Как работает <i>upstart</i> ?	193
11.3.2. Конфигурационные файлы <i>upstart</i>	194
11.4. Система инициализации <i>systemd</i>	195
11.4.1. Идеальная система инициализации.....	195
11.4.2. Введение в <i>systemd</i>	196
11.4.3. Основные особенности <i>systemd</i>	198
11.4.4. Сравнение <i>init</i> , <i>upstart</i> и <i>systemd</i>	198
11.4.5. Немного практики	200
11.4.6. Команды системного администратора.....	203
11.5. Система инициализации <i>Slackware</i>	204
Глава 12. Команды управления пользователями.....	207
12.1. Многопользовательская система.....	207
12.2. Пользователь <i>root</i>	208
12.2.1. Максимальные полномочия.....	208
12.2.2. Как работать без <i>root</i>	208
Команда <i>sudo</i>	209
Команда <i>su</i>	209
Проблемы с <i>sudo</i> в <i>Ubuntu</i> и <i>Kubuntu</i>	210
Ввод серии команд <i>sudo</i>	211
12.2.3. Переход к традиционной учетной записи <i>root</i>	211
Преимущества и недостатки <i>sudo</i>	211

Традиционная учетная запись root в Ubuntu.....	212
Традиционная учетная запись root в Mandriva	213
Вход в качестве root в Fedora.....	214
12.3. Создание, удаление и модификация пользователей стандартными средствами.....	215
12.3.1. Команды <i>adduser</i> и <i>passwd</i>	215
12.3.2. Команды <i>usermod</i> и <i>chage</i>	216
12.3.3. Команда <i>userdel</i>	217
12.3.4. Команды <i>pwck</i> и <i>grpck</i>	218
12.3.5. Подробно о создании пользователей	218
12.4. Группы пользователей.....	219
12.5. Команды квотирования.....	219

Глава 13. Ядро..... 223

13.1. Команда <i>dmesg</i> : вывод сообщений ядра	223
13.2. Параметры ядра	229
13.3. Компиляция ядра	233
13.3.1. Установка исходных кодов ядра	234
13.3.2. Настройка ядра	234
13.3.3. Компиляция ядра	238
13.4. RT-ядро.....	240
13.5. Особенности компиляции ядра в других дистрибутивах Linux.....	240

ЧАСТЬ III. ПРОГРАММИРОВАНИЕ И АВТОМАТИЗАЦИЯ В LINUX 243

Глава 14. Программирование на языке C. Утилиты для программиста 245

14.1. Команда <i>gcc</i> : компилятор	245
14.2. Команда <i>make</i> : сборка проекта	247
14.3. Команды из пакета <i>binutils</i>	248
14.4. Другие полезные команды.....	249
14.5. Команда <i>gdb</i> : отладка программ	249

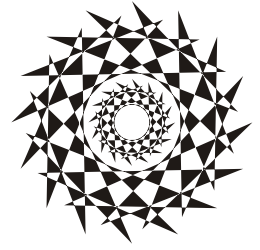
Глава 15. Автоматизация рутинных задач с помощью *bash* 252

15.1. Настройка <i>bash</i>	252
15.2. Автоматизация задач с помощью <i>bash</i>	254
15.3. Привет, мир!.....	255
15.4. Использование переменных в собственных сценариях.....	255
15.5. Передача параметров сценарию	257
15.6. Массивы и <i>bash</i>	257
15.7. Циклы	258
15.8. Условные операторы	259
15.9. Функции.....	260
15.10. Примеры сценариев.....	260
15.10.1. Сценарий мониторинга журнала	260
15.10.2. Переименование файлов	261
15.10.3. Преобразование систем счисления	262
15.10.4. Проверка прав пользователя.....	262
15.10.5. Генератор имени временного файла	263
15.10.6. Проверка свободного дискового пространства с уведомлением по электронной почте	263

Глава 16. Автоматизация задач с помощью <i>tcs</i>h.....	264
16.1. Использование <i>tcs</i> h.....	264
16.2. Конфигурационные файлы <i>tcs</i> h.....	265
16.3. Создание сценариев на <i>tcs</i> h.....	266
16.3.1. Переменные, массивы и выражения.....	266
16.3.2. Чтение ввода пользователя.....	269
16.3.3. Переменные оболочки <i>tcs</i> h.....	269
16.3.4. Управляющие структуры.....	272
Условный оператор <i>if</i>	272
Условный оператор <i>if.then.else</i>	273
Оператор <i>foreach</i>	274
Оператор <i>while</i>	274
Оператор <i>switch</i>	275
16.3.5. Встроенные команды <i>tcs</i> h.....	276
Глава 17. Автоматизация обработки задач средствами <i>gawk</i>.....	278
17.1. Введение в <i>gawk</i>	278
17.2. Основы языка.....	278
17.2.1. Образцы и действия.....	278
17.2.2. Операторы.....	279
17.2.3. Переменные.....	280
17.2.4. Ассоциативные массивы.....	280
17.2.5. Функции.....	280
17.2.6. Вывод с помощью <i>printf</i>	281
17.2.7. Управляющие структуры.....	282
Условный оператор <i>if.else</i>	282
Цикл <i>while</i>	282
Цикл <i>for</i>	282
17.3. Примеры.....	283
Глава 18. Собственный сервер для РНР-программиста.....	286
18.1. Зачем нужен сервер РНР-программисту?.....	286
18.2. Web-сервер.....	286
18.2.1. Установка Apache и РНР.....	286
18.2.2. Тестирование настроек Web-сервера.....	287
18.2.3. Конфигурационные файлы сервера. Команды запуска и останова сервера.....	289
18.3. Сервер баз данных MySQL.....	289
18.3.1. Установка сервера.....	289
18.3.2. Команды управления пользователями MySQL-сервера.....	290
18.3.3. Команды запуска и останова сервера.....	291
18.3.4. Программа MySQL Administrator.....	291
18.4. Быстрая настройка FTP-сервера.....	293
ЧАСТЬ IV. УПРАВЛЕНИЕ ПАКЕТАМИ.....	297
Глава 19. Введение в пакеты. Программы <i>rpm</i> и <i>dpkg</i>.....	299
19.1. Что такое пакет?.....	299
19.2. Репозитории пакетов.....	301
19.3. Программы для управления пакетами.....	302

19.4. Программа <i>rpm</i> для всех дистрибутивов, совместимых с Red Hat.....	303
19.5. Программа <i>rpmbuild</i> : простая сборка пакетов исходного кода	304
19.6. Программа <i>dpkg</i> : управление DEB-пакетами.....	304
19.7. Команда <i>alien</i> : установка RPM-пакетов.....	306
19.8. Создание RPM-пакетов	307
19.8.1. Подготовка окружения.....	307
19.8.2. Создание файла спецификаций	308
19.8.3. Сборка пакета.....	310
Глава 20. Управление пакетами в Debian/Ubuntu.....	311
20.1. Программы для управления пакетами	311
20.2. Программа <i>apt-get</i>	311
20.2.1. Установка пакетов. Источники пакетов	311
20.2.2. Основные команды программы <i>apt-get</i>	312
Обновление источников	313
Удаление и переустановка пакетов	313
Обновление пакета и системы	314
Очистка кэша пакетов.....	314
Опции программы <i>apt-get</i>	315
Подключение репозитория Medibuntu в Ubuntu.....	315
Корова в <i>apt-get</i>	316
20.3. Программа <i>aptitude</i>	316
Глава 21. Управление пакетами в Fedora.....	317
21.1. Использование программы <i>yum</i>	317
21.2. Управление источниками пакетов.....	319
21.3. Установка пакетов через прокси-сервер.....	321
21.4. Плагины для программы <i>yum</i>	321
Глава 22. Управление пакетами в openSUSE. Менеджер пакетов <i>zypper</i>	322
Глава 23. Управление пакетами в Slackware	326
23.1. Особенности Slackware	326
23.2. Управление пакетами	327
23.2.1. Команда <i>installpkg</i> : установка пакетов	328
23.2.2. Команда <i>removepkg</i> : удаление пакетов	329
23.2.3. Команда <i>upgradepkg</i> : обновление пакетов	330
23.3. Нет нужного пакета — вам поможет программа <i>rpm2tgz</i>	330
23.4. Программа <i>slackpkg</i> : установка пакетов из Интернета.....	330
Глава 24. Управление пакетами в Mandriva	332
24.1. Команда <i>urpmi</i> : установка пакетов.....	332
24.2. Команда <i>urpme</i> : удаление пакетов	337
24.3. Поиск пакета. Получение информации о пакете.....	337
ЧАСТЬ V. СРЕДСТВА РЕЗЕРВНОГО КОПИРОВАНИЯ И ДИСТРИБУТИВОСТРОЕНИЯ.....	339
Глава 25. Создание дистрибутива	341
25.1. Зачем нужно создавать еще один дистрибутив?	341

25.2. Инструменты для создания дистрибутива	342
25.3. Этапы создания дистрибутива	343
25.4. Процесс создания дистрибутива	343
25.5. Развитие дистрибутива	346
Глава 26. Средства резервного копирования. Создание LiveCD-диска	347
26.1. Необходимость в "живой" резервной копии	347
26.2. Средства клонирования Linux	348
26.3. Clonezilla	349
26.4. Remastersys Backup	356
26.5. Linux Live	358
Заключение	361
Предметный указатель	363



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 сэкономяте трафик. А на обмен электронными сообщениями это никак не влияет. В консоли также можно работать и с документами, правда, тоже о графике следует 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 ис-



Рис. 1.1. Графический вход в систему (openSUSE 12.1)

ключения. Пример тому дистрибутив Slackware — в нем сначала нужно выполнить вход в консоли (см. рис. 1.3), а потом для запуска графического интерфейса ввести команду `startx`.

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>+<Fn>`, 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.

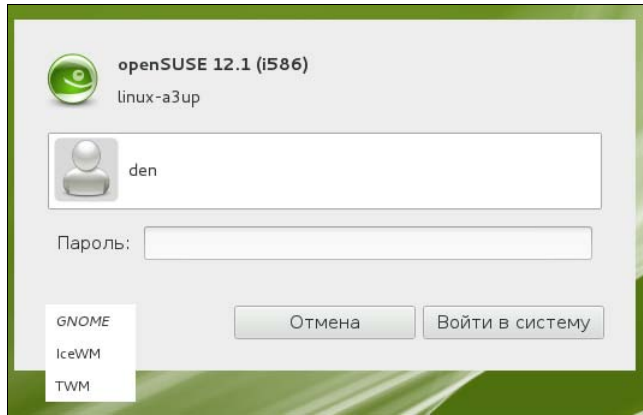


Рис. 1.2. Выбор типа сеанса (openSUSE 12.1)

```

Polling for DHCP server on interface eth0:
dhcpcd: MAC address = 00:0c:29:6f:40:83
Starting Internet super-server daemon: /usr/sbin/inetd
Starting OpenSSH SSH daemon: /usr/sbin/sshd
Starting ACPI daemon: /usr/sbin/acpid
Starting system message bus: /usr/bin/dbus-uuidgen --ensure ; /usr/bin/dbus-daemon --system
Starting HAL daemon: /usr/sbin/hald --daemon=yes
ALSA warning: No mixer settings found in /etc/asound.state.
Sound may be muted. Use 'alsamixer' to unmute your sound card,
and then 'alsactl store' to save the default ALSA mixer settings
to be loaded at boot.
Loading OSS compatibility modules for ALSA.
Loading /usr/share/kbd/keymaps/i386/qwerty/us.map.gz
Starting gpm: /usr/sbin/gpm -m /dev/mouse -t ps2

Welcome to Linux 2.6.21.5-smp (tty1)

dhsilabs login: root _____ ИМЯ ПОЛЬЗОВАТЕЛЯ
Password: _____ ПАРОЛЬ ПРИ ВВОДЕ НЕ ОТОБРАЖАЕТСЯ
Linux 2.6.21.5-smp.
Last login: Mon Mar  3 13:21:39 +0300 2008 on tty1.
You have mail.
root@dhsilabs:~# _

```

Рис. 1.3. Регистрация в консоли (Slackware)

Таблица 1.1. Клавиши переключения между консолями и графическим режимом

Комбинация клавиш	Назначение
<Ctrl>+<Alt>+<Fn> (n от 1 до 6)	Переключение из графического режима в консоль с номером n
<Alt>+<Fn> (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 того, 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 следует 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. Раз-

работчики 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 команды 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'
```

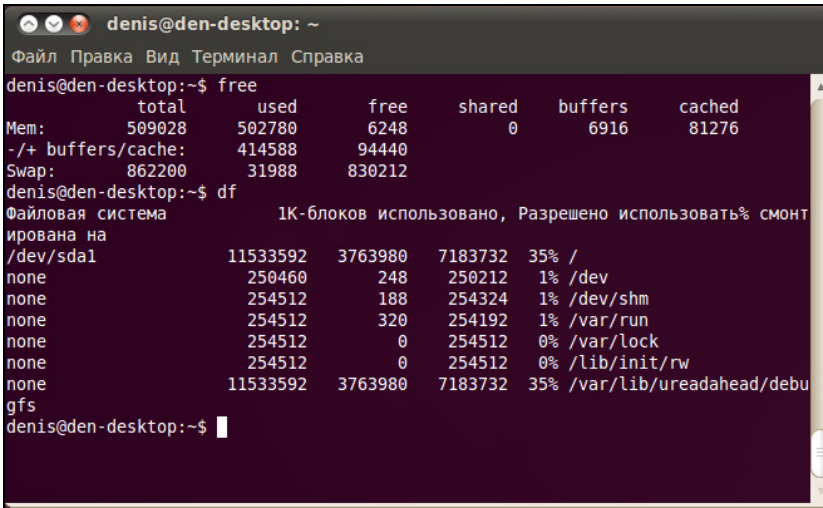
Для того чтобы изменения вступили в силу, выйдите из консоли (команда `logout`) и заново зарегистрируйтесь.

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

1.4. Графические терминалы

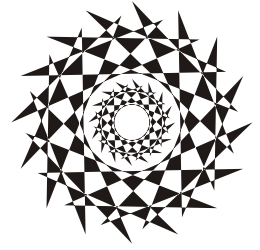
Понимаю, что большинство дистрибутивов оснащены графическим интерфейсом, который к тому же запускается по умолчанию. Поэтому большинство пользователей не будут жертвовать удобным и привычным интерфейсом ради консоли.

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).



```
denis@den-desktop: ~
Файл Правка Вид Терминал Справка
denis@den-desktop:~$ free
              total        used         free       shared    buffers     cached
Mem:           509028        502780          6248           0         6916        81276
-/+ buffers/cache:  414588        94440
Swap:          862200        31988       830212
denis@den-desktop:~$ df
Файловая система          1К-блоков использовано, Разрешено использовать% смонтирована на
/dev/sda1                11533592    3763980    7183732    35% /
none                    250460      248      250212    1% /dev
none                    254512      188      254324    1% /dev/shm
none                    254512      320      254192    1% /var/run
none                    254512       0      254512    0% /var/lock
none                    254512       0      254512    0% /lib/init/rw
none                    11533592    3763980    7183732    35% /var/lib/ureadahead/debu
gfs
denis@den-desktop:~$
```

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 прошлого века) 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 задачами, хранит историю ранее введенных команд, а также у *csh* есть сценарии, ко-

торые 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 возникли 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 создавать 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 придется по вкусу 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 псевдо-

нимы 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 большая 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 знакомства 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 — 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 функция 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 команды и 10 опций командной строки. Обычно `ash` используется при загрузке Linux в однопользовательском режиме (или в режиме восстановления).

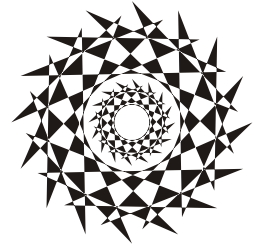
Оболочка `ash` совместима с `sh`, с ее помощью можно проверить сценарии на совместимость с традиционным синтаксисом `sh`. А в операционной системе NetBSD оболочка `ash` используется вместо `/bin/sh`.

2.9. Выбор оболочки

Какую оболочку выбрать? Первым делом нужно оценить простоту работы в ней. Ведь вы будете использовать эту оболочку каждый день, поэтому данный фактор должен быть на первом месте.

Затем нужно оценить простоту синтаксиса оболочки. Конечно, это только в том случае, если вы планируете разрабатывать собственные сценарии. Также не следует 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 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*: сведения об архитектуре 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 4
```

На рис. 3.1 представлена эта команда в действии.



Рис. 3.1. Команда `banner`

ПРИМЕЧАНИЕ

В Ubuntu по умолчанию команда `banner` недоступна, для ее добавления нужно установить пакет `sysvbanner`.

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
```