

АЛЕКСЕЙ БЕРЕСНЕВ

Администрирование GNU/Linux с нуля

2-е издание

Работа в командной строке
оболочки Bash и утилиты
GNU/Linux в подробностях

Файловые системы
GNU/Linux

Системные и сетевые
службы GNU/Linux

Серверы Apache, Sendmail,
Postfix, BIND, SAMBA и др.

Графическая система
X Window

СИСТЕМНЫЙ
АДМИНИСТРАТОР

Алексей Береснев

**Администрирование
GNU/Linux
с нуля**
2-е издание

Санкт-Петербург

«БХВ-Петербург»

2010

УДК 681.3.06
ББК 32.973.26-018.2
Б48

Береснев А. Л.

Б48 Администрирование GNU/Linux с нуля. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2010. — 576 с.: ил. + (Дистрибутивы на CD-ROM) — (Системный администратор)

ISBN 978-5-9775-0518-5

Дается необходимый набор знаний в области администрирования GNU/Linux. Материал не привязан к какому-либо конкретному дистрибутиву GNU/Linux, а рассмотрены общие процедуры организации и поддержки этих систем: работа в оболочке Bash, утилиты командной строки, файловые системы, управление загрузкой, системные и сетевые службы GNU/Linux. Приводится множество примеров, связанных с решением повседневных задач системного администрирования. Материал подобран так, чтобы читатель имел возможность подготовиться к сдаче сертификационных экзаменов Linux Professional Institute LPI-101 и LPI-102. Прилагаемый компакт-диск содержит образ установочного диска Ubuntu Server 9.10 и пакет с исходным кодом открытой версии Sun VirtualBox. Во втором издании учтены современные требования LPI, текст обновлен, примеры изменены в сторону наглядности.

Для системных администраторов

УДК 681.3.06
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>Григорий Добин</i>
Редактор	<i>Анна Кузьмина</i>
Компьютерная верстка	<i>Натальи Караваевой</i>
Корректор	<i>Виктория Пиотровская</i>
Дизайн серии	<i>Инны Тачиной</i>
Оформление обложки	<i>Елены Беляевой</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 30.03.10.

Формат 70×100^{1/16}. Печать офсетная. Усл. печ. л. 46,44.

Тираж 1500 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

Введение.....	1
История создания GNU/Linux	1
Что означает свобода распространения программного обеспечения?.....	4
Сертификация LPI.....	4
Для кого предназначена книга?	5
Благодарности	7

ЧАСТЬ I. НАЧАЛО РАБОТЫ В GNU/LINUX.....9

Глава 1. Первый раз устанавливаем GNU/Linux.....	11
Предварительные требования для установки GNU/Linux.....	11
Установка GNU/Linux.....	12

Глава 2. Работа в оболочке Bash	25
Учетные записи и вход в сеанс	25
Как вводить команды в shell?.....	28
Смена пароля пользователя.....	29
Идентификация пользователя	30
Кто сейчас работает в системе?	31
Что такое оболочка?.....	32
Структура командной строки.....	32
Популярные оболочки GNU/Linux	35
Встроенные и системные команды	36
Редактирование и исполнение команд.....	37
Переменные оболочки и окружения	38
История команд	41
Автоматическое дополнение командной строки	43
Псевдонимы команд (aliases)	44
Командная подстановка.....	45
Вычисление арифметических выражений	47
Шаблоны подстановки и перечисление	48

Глава 3. Помощь и документация	51
Сообщения об ошибках	51
Встроенная помощь оболочки Bash	52
Страницы помощи man	52
Файлы страниц man	55
GNU Texinfo	57
Документация программ	58
Источники информации в Интернете.....	59
 ЧАСТЬ II. ОСНОВЫ.....	 61
 Глава 4. Работа с файлами и каталогами.....	 63
Система файлов и каталогов	63
Имена файлов и команда <i>ls</i>	65
Перемещение по файловой системе	68
Создание и удаление файлов и каталогов.....	68
Копирование, перемещение и переименование файлов.....	72
Поиск файлов.....	74
Быстрый поиск файлов <i>locate</i>	76
Определение содержимого файла	77
Устройство файловой системы	78
Использование жестких связей.....	80
Использование символических ссылок.....	84
 Глава 5. Процессы.....	 87
Процессы и задания	87
Фоновый режим выполнения заданий	91
Жизненный цикл процесса	93
Мониторинг процессов.....	95
Сигналы.....	101
Перехват и обработка сигналов в Bash	103
Управление приоритетом процессов.....	104
 Глава 6. Права доступа и права владения	 107
Права владения файлами	107
Права доступа, устанавливаемые на файлы	108
Права доступа к каталогам	110
Изменение прав владения.....	111
Установка прав доступа.....	114
Автоматическая установка прав доступа к вновь создаваемым файлам.....	118
Специальные биты прав доступа: SUID, SGID и sticky bit	119

Классификация регулярных выражений.....	169
Поиск текста с помощью <i>grep</i>	171
Использование обратных ссылок	175
Использование регулярных выражений с <i>sed</i>	177
Регулярные выражения в <i>awk</i>	179

Глава 10. Написание сценариев Bash.....	181
Сценарии оболочки.....	181
Использование переменных оболочки.....	183
Экранирование (quotation).....	186
Интерактивная установка значений переменных	187
Позиционные параметры.....	188
Команда <i>test</i>	191
Условное исполнение команд.....	194
Команда <i>case</i>	198
Циклы	199
Функции	203
 ЧАСТЬ IV. АДМИНИСТРИРОВАНИЕ	 207
 Глава 11. Работа с носителями информации.....	 209
Физическая структура накопителя на жестких магнитных дисках.....	209
Имена жестких магнитных дисков	211
Создание разделов с использованием <i>fdisk</i>	213
Создание файловой системы.....	216
Проверка целостности файловой системы	219
Монтирование файловых систем.....	221
Работа с разделом подкачки.....	224
Файл информации о файловых системах <i>/etc/fstab</i>	227
Мониторинг дисковых ресурсов.....	229
 Глава 12. Резервное копирование	 231
Планирование резервного копирования	231
Команда <i>dd</i>	233
Утилиты для сжатия файлов	234
Команда <i>tar</i>	236
Команда <i>cpio</i>	239
Команда <i>pack</i>	241
Программы <i>dump</i> и <i>restore</i>	243
 Глава 13. Запуск, останов GNU/Linux и уровни выполнения.....	 246
Инициализация операционной системы и переход на заданный уровень исполнения	246
Остановка и перезагрузка системы	251

Глава 14. Загрузчики.....	254
Последовательность процесса загрузки.....	254
Загрузчик GRUB.....	255
Загрузчик LILO.....	258
Глава 15. Отложенное и регулярное выполнение заданий.....	262
Отложенное выполнение заданий	262
Автоматизация выполнения регулярных задач.....	264
Глава 16. Системные журналы	267
Служба syslog	267
Служба ротации журналов	271
Глава 17. Управление пользователями	273
Хранение учетных записей пользователей.....	273
Регистрация, удаление и блокирование учетных записей пользователей.....	274
Управление паролями	279
Управление группами пользователей	280
Профили пользователей	282
Квотирование дискового пространства.....	285
Мониторинг активности пользователей	291
Глава 18. Управление программным обеспечением.....	293
В чем состоит управление программным обеспечением?	293
Сборка и установка программного обеспечения из пакетов	
с исходным кодом	296
Управление библиотеками	299
Менеджер пакетов RPM	303
Система управления пакетами Debian	308
Глава 19. Установка аппаратного обеспечения.....	315
Установка нового оборудования	315
Работа с модулями ядра.....	316
Файлы устройств и udev	321
Устройства PCI.....	325
Установка SCSI-устройств	326
Установка сетевых адаптеров Ethernet.....	327
Работа со звуковыми картами	329
Поддержка USB.....	330
Устройства PCMCIA.....	331
Сборка и установка ядра Linux	331

ЧАСТЬ V. СЕТИ.....	337
Глава 20. Сетевые средства GNU/Linux	339
ТСР/IP	339
Адресация IPv4	341
Адресация IPv6	344
Настройка сетевого интерфейса Ethernet.....	346
Настройка маршрутизатора по умолчанию	347
Настройка разрешения имен	350
Поиск и устранение проблем с сетью	353
Глава 21. Сервисы сети.....	356
Идентификация служб сети.....	356
Запуск сетевых служб	358
Использование супердемона <i>inetd</i> и фильтра <i>tcpd</i>	359
Программа <i>tcpd</i>	360
Использование супердемона <i>xinetd</i>	362
Глава 22. Службы удаленного доступа	367
Служба telnet.....	367
Службы удаленного доступа (r-services).....	370
Система SSH	373
Глава 23. Служба FTP	378
Как работает служба FTP	378
Настройка сервера <i>vsftpd</i>	379
Клиенты FTP	381
Глава 24. Файловая система NFS.....	384
Настройка сервера NFS	384
Использование сервера NFS.....	386
Отличия протокола NFSv4	388
Глава 25. SMB/CIFS-сервер SAMBA	390
Состав пакета SAMBA.....	390
Настройка SAMBA.....	391
Запуск и работа системы SAMBA	394
Монтирование файловых ресурсов SMB	397
Использование сетевых принтеров	398
Запуск SAMBA в режиме PDC	399
Сервер SAMBA в режиме члена домена.....	401
Программа <i>winbind</i>	402

Глава 26. DNS-сервер BIND.....	404
Организация DNS	404
Конфигурационный файл BIND	407
Записи о ресурсах DNS	409
Запуск DNS-сервера BIND	416
Тестирование сервера DNS	417
Делегирование	419
Журналы DNS.....	420
Глава 27. Сервер DHCP	423
Работа DHCP.....	423
Настройка сервера DHCP	424
Глава 28. Web-сервер Apache.....	426
Конфигурационный файл Apache	426
Контейнеры.....	431
Запуск и управление Apache	433
Личные Web-страницы	434
Ограничение доступа к Web-ресурсу.....	436
Виртуальные узлы.....	438
Глава 29. Электронная почта	440
Организация электронной почты.....	440
Файл конфигурации программы Sendmail.....	442
Файл конфигурации sendmail.mc.....	446
Запуск Sendmail	450
Почтовые псевдонимы.....	452
Очередь почтовых сообщений.....	454
Тестирование Sendmail	455
Преимущества использования Postfix	458
Конфигурационные файлы Postfix	462
Виртуальный хостинг	465
POP3/IMAP-сервер Dovecot	466
Глава 30. Печать в GNU/Linux	471
Система печати CUPS	471
Команды CUPS	473
Управление принтерами в CUPS	475
Управление очередью печати	477

Глава 31. Сервер NTP	480
Сервис синхронизации времени	480
Утилита <i>ntpdate</i>	481
Пакет <i>ntp</i>	482
Глава 32. Система X Window	484
Организация X Window	484
Конфигурирование X Window	486
Сервер шрифтов	491
Запуск X-сервера из командной строки	493
Менеджер X-сеанса <i>xdm</i>	497
X-приложения.....	499
Шрифты.....	500
Ресурсы X-приложений	502
Удаленный запуск X-приложений.....	504
Использование <i>xdm</i> для удаленного входа в сеанс	507
 ПРИЛОЖЕНИЯ	 511
Приложение 1. Работа с VMWare Workstation и Sun VirtualBox.....	513
Создание виртуальной машины в Sun VirtualBox.....	513
Создание виртуальной машины в VMWare Workstation.....	518
 Приложение 2. Примеры использования текстовых утилит GNU	 523
Копирование с помощью команды <i>tee</i>	523
Нумерация строк с помощью команды <i>cat</i>	524
Нумерация строк с помощью команды <i>nl</i>	525
Команда <i>csplit</i>	525
Команда <i>sed</i>	527
Команда <i>tac</i>	527
Команда <i>awk</i>	528
Команды <i>expand</i> и <i>unexpand</i>	528
Команда <i>pr</i>	529
Команды <i>sort</i> и <i>uniq</i>	530
Команда <i>wc</i>	530
Команда <i>tr</i>	530
Команда <i>grep</i>	531

Приложение 3. Пример использования telnet для тестирования MTA	532
Приложение 4. Пример файлов конфигурации и описания зон сервера DNS BIND	533
Конфигурация named	533
Зона указателей на корневые серверы	535
Зона localhost	538
Обратная зона для 127.0.0	538
Зона class.edu	539
Обратная зона для 192.168.0	539
Приложение 5. Сложные варианты установки GNU/Linux.....	540
Требования к аппаратному обеспечению для установки GNU/Linux на платформе x86/64	540
Подготовка к установке GNU/Linux на компьютерах с архитектурой x86/64.....	542
Установка GNU/Linux.....	548
Приложение 6. Описание компакт-диска.....	550
Предметный указатель	551

Введение

Во введении приведены некоторые исторические данные о GNU/Linux, а также обсуждается, что такое свободное программное обеспечение, и перечисляются основные виды свободных лицензий. Читатель узнает о том, что такое LPI и какие виды профессиональной сертификации доступны для GNU/Linux. Здесь описано, как читать эту книгу и, конечно, приведены благодарности автора.

История создания GNU/Linux

Операционная система GNU/Linux относится к классу UNIX-подобных операционных систем, наследуя от UNIX множество черт. Операционная система UNIX была создана Кеном Томпсоном и его коллегами в Bell Laboratories фирмы AT&T в 1969 г. Для тех лет это была одна из самых передовых операционных систем, обеспечивающих многозадачность и возможность одновременной работы многих пользователей. В настоящее время имеются две основные ветви UNIX-систем: UNIX System V (продолжение разработок AT&T) и BSD (Berkeley Software Distribution, которая ранее разрабатывалась в университете Berkeley). То есть все современные UNIX-системы можно отнести к первой или второй ветви. Так, операционная система Sun Solaris 10 является представителем ветви UNIX System V (кратко SVR4 — System V Release 4), а FreeBSD 8.0 — наследницей BSD.

Операционная система GNU/Linux не является прямой наследницей какой-либо из этих двух ветвей UNIX-систем. Она сочетает в себе черты, присущие обоим ветвям, поскольку ее разрабатывает множество людей, имеющих, естественно, различные предпочтения. В отличие от команды разработчиков FreeBSD, разработку GNU/Linux в целом, как единой операционной системы, никто не координирует. Поэтому имеется множество различных наборов программного обеспечения (дистрибутивов), являющихся, несмотря на совершенную несхожесть друг с другом, GNU/Linux.

Что же такое GNU/Linux? Название Linux является зарегистрированной торговой маркой Линуса Бенедикта Торвальдса, а GNU — наименование проекта "GNU is not UNIX" (рекурсивный акроним, который можно расшифровывать бесконечно). Проект GNU основал в 1984 г. Ричард Столлмэн в FSF (Free Software Foundation). Линус Торвальдс, будучи в 1991 г. студентом, экспериментировал с операционной системой MINIX (она была разработана профессором Эндрью Танненбаумом в учебных целях) и ассемблером процессора i386.

25 августа 1991 г. Линус Торвальдс распространил в группе новостей **comp.os.minix.usenet** сообщение о том, что он разработал на основе MINIX новую операционную систему для АТ-совместимых компьютеров, и пригласил всех заинтересованных лиц участвовать в ее разработке. В своем сообщении он специально указал, что его эксперименты не более чем хобби, и он "... не является таким профессионалом, как специалисты из GNU...".

Действительно, детище Линуса так и осталось бы, вероятно, только экспериментом, но его сообщением заинтересовался Ричард Столлмэн — основатель FSF. К тому моменту в FSF уже семь лет проводились работы в рамках проекта GNU, целью которого было создание свободно распространяемой по лицензии GPL Copу Left совместимой с SVR4 операционной системы (суть лицензии GPL будет пояснена позже).

В FSF к 1991 г. было создано огромное количество широко используемого программного обеспечения, например, оболочка Bash (Bourne again shell), компилятор gcc и пр. Однако для реализации проекта GNU не хватало стабильно работающего ядра операционной системы. Исходно проект GNU был ориентирован на ядро HURD, но работа по созданию этого ядра до сих пор далека от завершения, поэтому тандем — ядро от Линуса Торвальдса плюс утилиты, библиотеки, компилятор, оболочка и прочее от GNU — был отличной альтернативой GNU/HURD.

В течение 1991—1992 гг. проект GNU/Linux активно развивался, и в 1993 г. появился первый дистрибутив GNU/Linux, собранный в компании Red Hat — ныне ведущим поставщиком GNU/Linux-дистрибутивов. Первые дистрибутивы вряд ли можно было рекомендовать для промышленного использования, но они повлекли массовый интерес к новой операционной системе со стороны широких масс разработчиков и производителей аппаратного обеспечения. К середине 90-х гг. прошлого века появились вполне стабильные и надежные дистрибутивы, в которых поставлялось большое количество портированного в GNU/Linux программного обеспечения.

Залогом успеха GNU/Linux явились два аспекта: бесплатность и свобода распространения дистрибутивов и достаточная для промышленного использова-

ния стабильность множества серверных приложений. Эти две особенности GNU/Linux позволяли строить "малобюджетные" серверы для небольших и средних приложений.

С переносом на GNU/Linux офисных приложений дело обстоит существенно хуже вплоть до конца девяностых годов. Однако на сегодняшний момент имеются отличные графические оболочки: GNOME и KDE, обилие оконных менеджеров, средства офисной работы, редакторы и электронные таблицы. В настоящее время уже есть прецеденты массового перевода на GNU/Linux компьютерных систем крупных фирм и даже муниципальных структур очень больших городов, что доказывает зрелость операционной системы GNU/Linux.

В настоящее время в России пользуются наибольшим распространением следующие дистрибутивы GNU/Linux:

- ❑ Red Hat и Fedora — простые в установке и настройке дистрибутивы (Fedora ориентирована на свободное распространение);
- ❑ SlackWare — очень популярный в России дистрибутив, предназначенный для профессионалов и требующий существенных усилий для настройки;
- ❑ Mandriva — дистрибутив, изготавливаемый существенной переделкой дистрибутивов Red Hat. Этот дистрибутив обладает массой полезных и удобных утилит, облегчающих работу для обычных пользователей настольных рабочих станций;
- ❑ Debian — дистрибутив с очень большим количеством программных пакетов. Команда разработки этого дистрибутива гарантирует свободу его распространения;
- ❑ Ubuntu — дистрибутив, стабильно занимающий первую строку в рейтинге популярности, базирующийся на Debian;
- ❑ Novell SUSE и Open SUSE — исключительно популярные дистрибутивы. Отличаются энциклопедической подборкой программного обеспечения, высокой надежностью, удобством и качественной документацией;
- ❑ Gentoo — проект, по методам установки программ тяготеющий к FreeBSD;
- ❑ ASP — российский дистрибутив, разработчики которого декларируют высокую совместимость с Red Hat. Отличается простотой и ясностью установки и высоким качеством русификации;
- ❑ ALT — отличный российский дистрибутив, разработчики которого стараются обеспечить его высочайшую защищенность и надежность.

Что означает свобода распространения программного обеспечения?

Чаще всего программное обеспечение, распространяемое в рамках свободных лицензий, таких как GPL или BSD, доступно совершенно бесплатно. Однако свобода программного обеспечения вовсе не подразумевает обязательную бесплатность. Некоторые дистрибутивы GNU/Linux предоставляются за деньги. Основная идея свободы программного обеспечения заключается в свободе его модификации и использования. По определению FSF программное обеспечение является свободным, если:

- ☐ программа может выполняться с любой целью без ограничения (свобода 0);
- ☐ имеется возможность изучения и модификации исходного кода программного обеспечения в соответствии со своими потребностями (свобода 1);
- ☐ разрешается свободно распространять копии программы (свобода 2);
- ☐ имеется возможность улучшать программное обеспечение и публиковать улучшения для всеобщего блага (свобода 3).

Следует отметить, что программное обеспечение, доступное в рамках GPL, не является общественной собственностью (Public Domain). Программа является собственностью ее авторов.

Сертификация LPI

Очень часто приходится слышать мнение о том, что наличие профессионального сертификата в какой-либо IT-области ровным счетом ничего не значит. Позволю себе не согласиться. Почему? Все очень просто: еще совсем недавно я считал, что многое понимаю в UNIX- и GNU/Linux-системах, я искренне полагал, что являюсь хорошим системным администратором. И это несмотря на то, что я не понимал, что значит право 750, установленное на каталог.

Мои иллюзии быстро развеялись с подачи Сергея Третьякова, сыгравшего огромную роль в появлении этой книги. Он предложил мне познакомиться с сертификационными требованиями LPI — Linux Professional Institute. Посетив www.lpi.org, я стал читать требования к администраторам GNU/Linux-систем. Каково же было мое удивление, когда я увидел, что не способен ответить на 90% вопросов, которые в качестве примеров предлагались на сайте!

Это был поворотный момент в моей жизни: от иллюзий я перешел к методичной проработке экзаменационных вопросов, и для меня стал открываться

волшебный мир GNU/Linux. На подготовку к моей первой сертификации LPI-I я потратил год, через год я получил следующий сертификат LPI-II. Фактически первый вариант этой книги был написан в 2001 г. Теперь, почти через 10 лет после этого, я считаю, что знаю примерно 1,5% от того объема, который мне хотелось бы знать о GNU/Linux.

Дорогие коллеги! Можно сколь угодно высоко ценить себя, как профессионала и настоящего системного администратора, но наличие сертификата дает вам реальное ощущение своего профессионализма, т. к. без должных знаний и умений экзамен сдать невозможно. В то же время наличие сертификата вовсе не заставляет остановиться в своем развитии. Чем больше вы знаете о GNU/Linux, тем больше хочется им заниматься и углублять свои знания. И не только в нем — ведь рядом есть такие замечательные вещи, как FreeBSD и Open Solaris!

Почему я предпочел сертификацию LPI, а не Red Hat или Novell SUSE? Все очень просто — сертификация LPI не привязана к конкретному дистрибутиву GNU/Linux, она требует знания GNU/Linux вообще. В то же время любые виды сертификаций, разработанные в коммерческих фирмах, слишком много внимания уделяют частным аспектам, связанным лишь с их дистрибутивом. Это не значит, что они хуже. Но в мире свободного программного обеспечения огромную роль играет широта охвата информации. Нельзя ограничиваться знанием единственной разновидности GNU/Linux. Именно эта широта охвата и важна для успешной сдачи экзаменов LPI.

Эта книга написана так, чтобы вы могли, должным образом попрактиковавшись, сдать экзамены LPI-101 и LPI-102, необходимые для получения сертификата LPI-I. Конечно, материал книги поможет вам подготовиться и к другим экзаменам, например, для сертификации по Red Hat или Novell SUSE.

Для кого предназначена книга?

Этой книгой могут с успехом воспользоваться не только специалисты, готовящиеся к сдаче сертификационных экзаменов по GNU/Linux, но и все, кому интересен GNU/Linux и кто хочет углубить свои знания о нем. Книга не требует значительной предварительной подготовки. По материалам, которые легли в основу этой книги, обучилось множество людей с разным уровнем подготовки и опытом: студенты, опытные инженеры, прекрасные дамы и солидные джентльмены.

Для людей, которые еще не работали с GNU/Linux, эта книга предоставит базовые знания, которые обеспечат им вхождение в мир свободного про-

граммного обеспечения. Опытные специалисты, не первый год работающие с GNU/Linux, найдут в книге много нового для себя. И тех, и других я очень прошу уделить особое внимание первым главам книги. Они дают базисные знания, без которых развитие специалиста становится похожим на броуновское движение. А опытный администратор без знаний текстовых утилит и работы в оболочке — это все равно, что орел с одним крылом.

Вы можете ненавидеть редактор `vi`, но тогда вы должны обеспечить себе иное средство редактирования текстовых файлов, которое будет у вас под рукой в момент аварии системы. Поэтому желательно изучать главы подряд — их последовательность выверена годами. Но вы также можете читать их в произвольном порядке.

В мире свободы никто не волен диктовать вам свои предпочтения. Здесь можно только советовать и помогать друг к другу. Если вы нашли досадную ошибку в программе, не спешите ругать ее автора — напишите ему письмо с предложением помощи, а лучше — сразу направьте ему исправления в программе. Если к вам обращается начинающий коллега с вопросом, который кажется вам глупым — не смейтесь над ним, лучше помогите ему, вспомнив себя в его годы.

Изучение GNU/Linux требует большой настойчивости и регулярной практической работы. Эта книга не подходит для прочтения вдали от компьютера. Поэтому самое важное, что вам нужно для работы с этой книгой, — компьютер с GNU/Linux под рукой. Какой дистрибутив вы используете — это ваше дело. Большая часть примеров пригодна к использованию в любых дистрибутивах GNU/Linux. В то же время предостерегаю вас от механического копирования примеров. Все, что здесь написано, требует осмысления, сравнения с действительностью и практической проверки. Я давно отучил себя говорить в GNU/Linux слова "никогда" и "всегда". В мире свободного ПО действительность не является черно-белой, здесь есть полутона, да еще и цвета разнообразные. Именно за то, что GNU/Linux постоянно заставляет думать, его так ценю я и многие другие люди. Жаль, конечно, что я сам не могу заглянуть к вам в экран и помочь советом.

В тексте используется несложное типографское соглашение: если текст выделен шрифтом `Courier`, значит, это либо имя команды, либо исполняемого файла, либо пользователя, либо процесса. Иногда, казалось бы, одни и те же строки напечатаны разными шрифтами. Например, `Bash` и `bash`. Все просто: `Bash` — имя собственное (Bourne Again Shell), а `bash` — имя команды. Надеюсь, что запутаться здесь сложно.

Благодарности

В детстве я не подавал особых надежд, и моим родителям стоило больших усилий заставить меня работать. Теперь, когда у меня взрослая дочь, я вижу, какого труда стоило родителям мое воспитание. Поэтому книга посвящена моим родителям.

Я уже упоминал Сергея Третьякова, который в свое время и предложил мне написать эту книгу. Без его вдумчивых советов и корректировок у меня вряд ли что-либо получилось. Исключительную важность для меня всегда имели критические замечания и дружеские комментарии Юрия Белякова, Дмитрия Вострецова, Анатолия Анохина и Евгения Сафонова. Их комментарии всегда отличались высоким профессионализмом и энциклопедической широтой. Кроме них мне всегда оказывает любую помощь Алексей Залецкий. Его знания в области сетевых технологий совершенно необъятны.

Я хотел бы перечислить здесь еще сотню имен, но лучше выражу им свою признательность за всестороннюю помощь лично. Жаль, что не могу выразить свою благодарность всему свободному сообществу, GNU/Linux, BSD и других направлений, за их титанический труд. Не будь их усилий, мне не о чем бы было писать в этой книге.

Одно я знаю точно: если бы не многолетняя поддержка, любовь и ласка важнейшего человека в моей жизни — Татьяны, я ничего не добился бы в жизни!



ЧАСТЬ I

**Начало работы
в GNU/Linux**



Первый раз устанавливаем GNU/Linux

В этой главе рассматривается обычный способ установки GNU/Linux и требования, которые должны быть выполнены для его успешной установки.

Предварительные требования для установки GNU/Linux

Для основных ныне используемых дистрибутивов GNU/Linux можно выделить следующие стандартные варианты процедуры установки на компьютерах x86/64:

- ☐ с комплекта установочных CD/DVD-дисков;
- ☐ с FTP/HTTP-сервера;
- ☐ с NFS-сетевых файлового ресурса.

Возможна установка как со SCSI, так и с IDE CD/DVD-дисков. Многие дистрибутивы также могут быть установлены и с других носителей. Например, с USB-накопителей.

Установка с CD/DVD обычно производится с загрузочного диска. Раньше, в случаях, когда аппаратура не позволяла загрузиться с CD/DVD-диска, требовалось создать загрузочную дискету. Современные ядра Linux уже не позволяют загрузиться с дискеты, поэтому при невозможности загрузки с CD/DVD приходится применять способ установки через сеть, с USB-носителя или какой-либо другой.

Сейчас роль загрузочной дискеты играет небольшой образ CD, позволяющий загрузиться и продолжить установку через сеть. Этот образ обычно находится на первом диске установочного комплекта.

Возможна ли установка конкретного дистрибутива GNU/Linux с некоторого требуемого вида носителя, зависит от производителя этого дистрибутива.

Информация о возможных видах установки данного дистрибутива может быть найдена либо на дисках установочного комплекта (файлы README или INSTALL), либо на сайте производителя этого дистрибутива.

GNU/Linux может быть установлен на жесткий магнитный диск совместно с другими операционными системами, причем для нормальной установки необходимо минимум два раздела (первичных или логических). Один из них используется для корневой файловой системы, а другой — для раздела подкачки. Дополнительная информация о вариантах установки GNU/Linux находится в *приложении 5*.

Процедура установки GNU/Linux, описанная далее в этой главе, предполагает, что на компьютере имеется CD-ROM и сетевая плата. Если установка выполняется непосредственно на компьютер, то вполне достаточно иметь 512 Мбайт ОЗУ, если же установка производится на виртуальную машину, желательно иметь минимум 1 Гбайт физической памяти на компьютере.

ЗАДАНИЯ

- Проверьте требования к аппаратному обеспечению для предпочитаемого вами дистрибутива.
- Может ли он быть установлен с CD или он требует DVD?

Установка GNU/Linux

В качестве примера здесь рассмотрена установка одного из наиболее популярных дистрибутивов GNU/Linux — Ubuntu Server 9.10. Установка Ubuntu показывает все обычные шаги, которые необходимо выполнить при установке GNU/Linux:

- ☐ выбор языка установки;
- ☐ выбор локали и раскладки клавиатуры;
- ☐ выбор имени узла;
- ☐ настройка временной зоны;
- ☐ разметка жестких дисков;
- ☐ выбор пароля для пользователя `root`;
- ☐ регистрация обычных пользователей системы;
- ☐ настройки сети;
- ☐ выбор программного обеспечения для установки;
- ☐ дополнительные настройки, зависящие от устанавливаемого программного обеспечения.

Для установки необходимо иметь установочный CD или файл с его образом. Следует загрузиться с этого диска. Если вы собираетесь устанавливать Ubuntu в качестве гостевой виртуальной операционной системы, надо соответствующим образом подготовить виртуальную машину. Процесс подготовки виртуальной машины описан в *приложении 1*.

После загрузки с CD вам будет предложено выбрать язык установки. Если выбран русский язык, вы увидите экран приглашения к установке (рис. 1.1).

Следующее диалоговое окно предназначено для определения локали устанавливаемой системы (рис. 1.2).

Далее будет предложено выбрать раскладку клавиатуры (рис. 1.3).

Теперь необходимо установить имя узла сети (рис. 1.4).

Следующий диалог (рис. 1.5) предназначен для установки временной зоны.

Диалог разметки диска на разделы показан на рис. 1.6. Для простоты здесь предлагается использовать автоматические настройки по умолчанию.

Если была выбрана автоматическая разметка диска, то будет выведен диалог выбора диска для автоматической разметки (рис. 1.7).

После выбора разметки диска вы можете либо подтвердить, либо отвергнуть разметку (рис. 1.8).

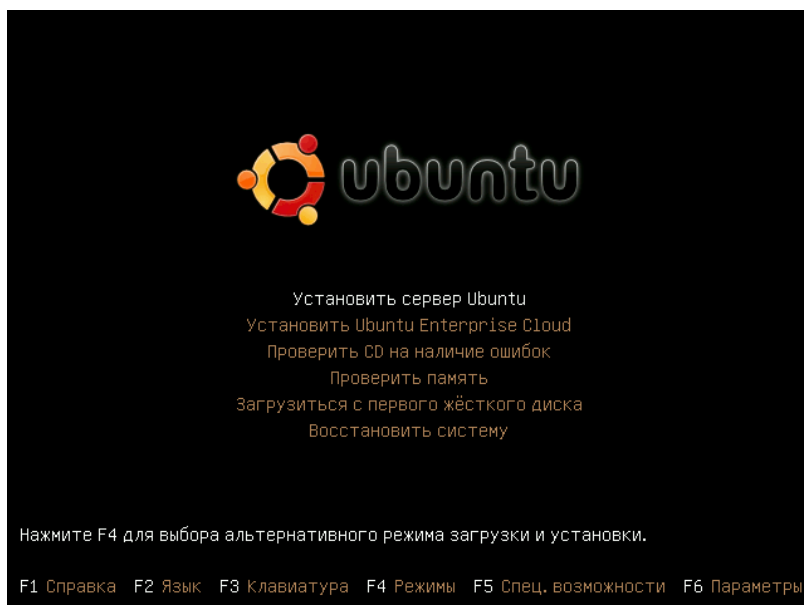


Рис. 1.1. Приглашение к установке

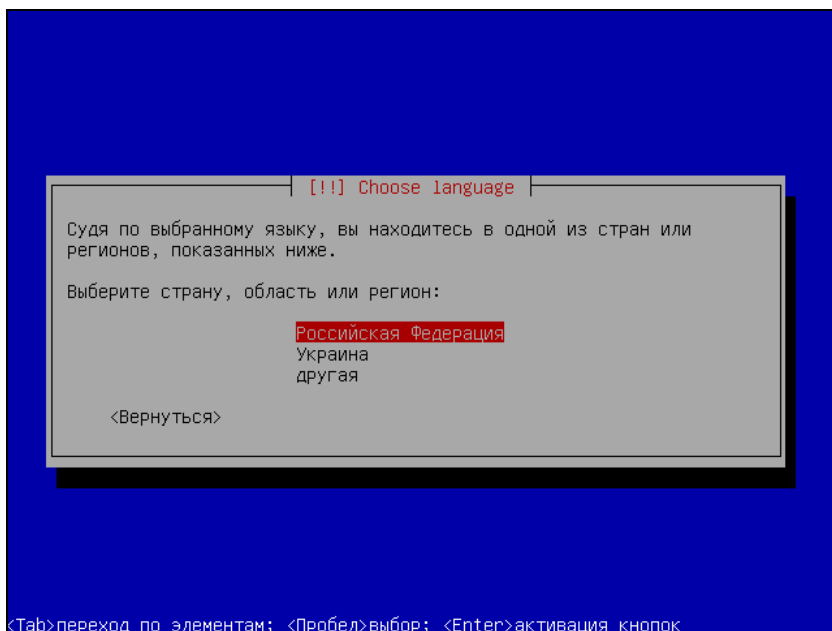


Рис. 1.2. Выбор локали установки

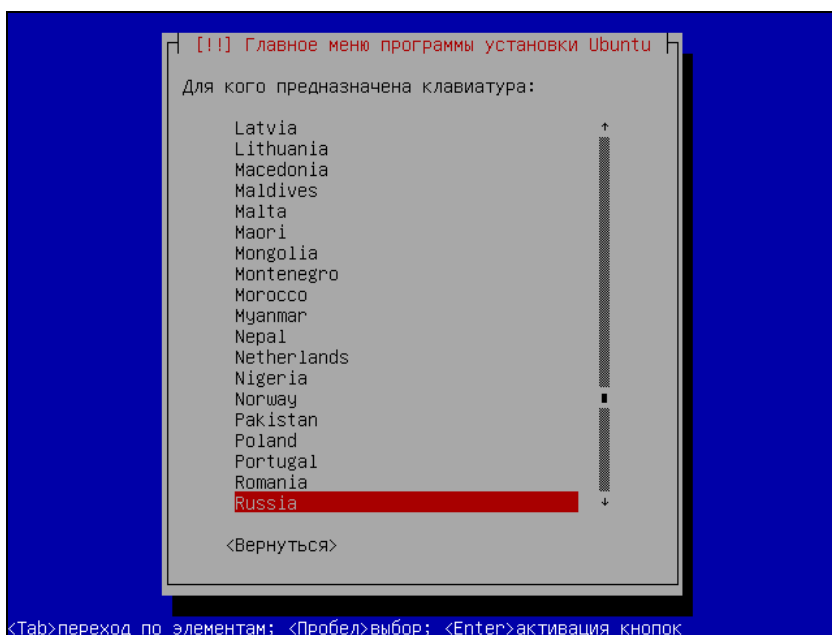


Рис. 1.3. Выбор раскладки клавиатуры

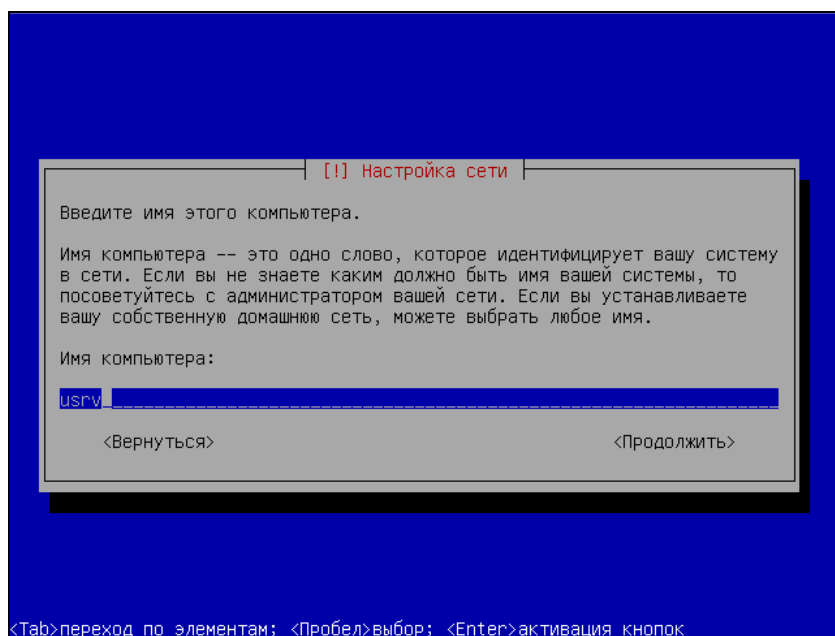


Рис. 1.4. Установка имени узла

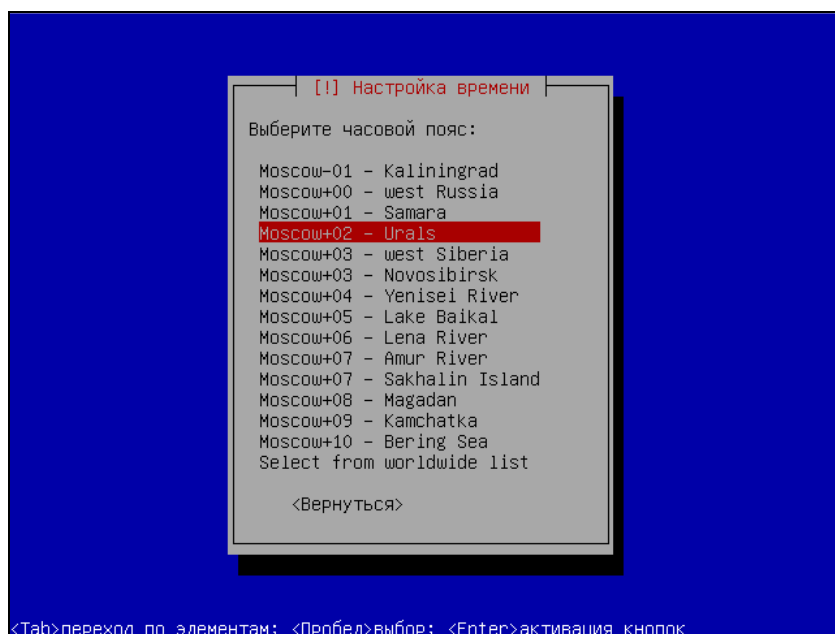


Рис. 1.5. Установка временной зоны

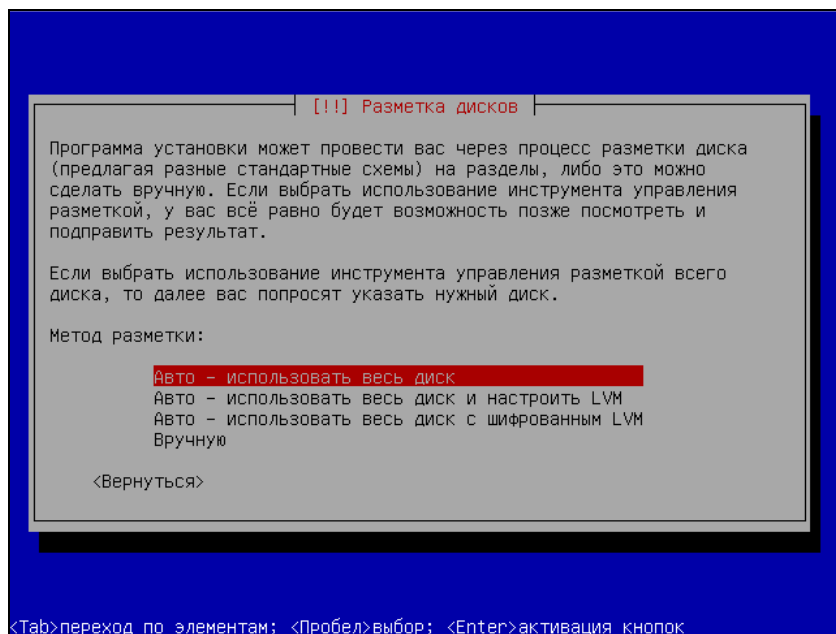


Рис. 1.6. Разметка диска

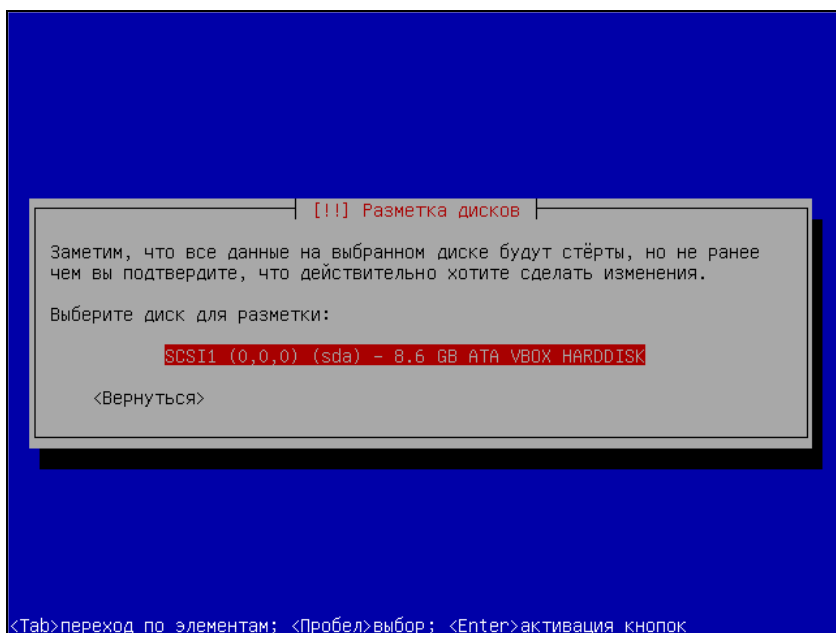


Рис. 1.7. Выбор диска для автоматической разметки

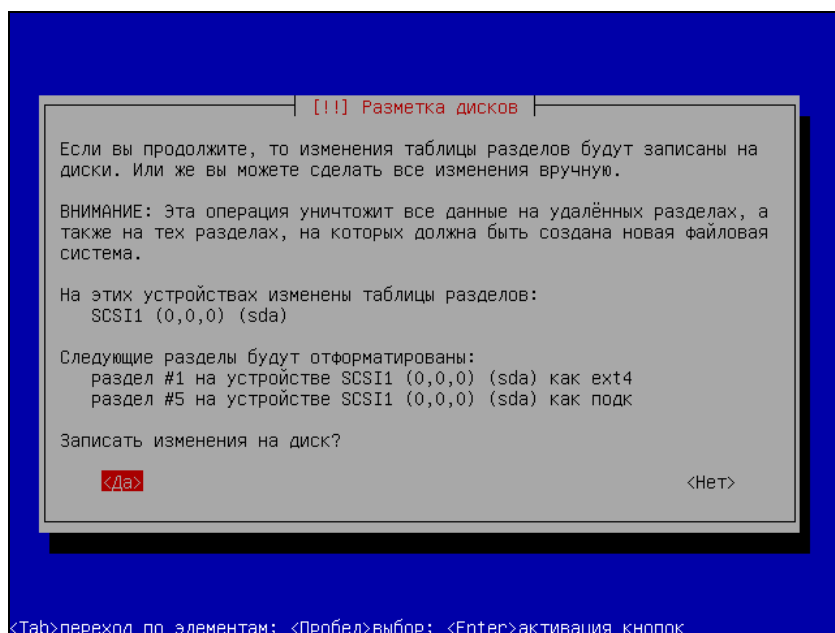


Рис. 1.8. Подтверждение разметки

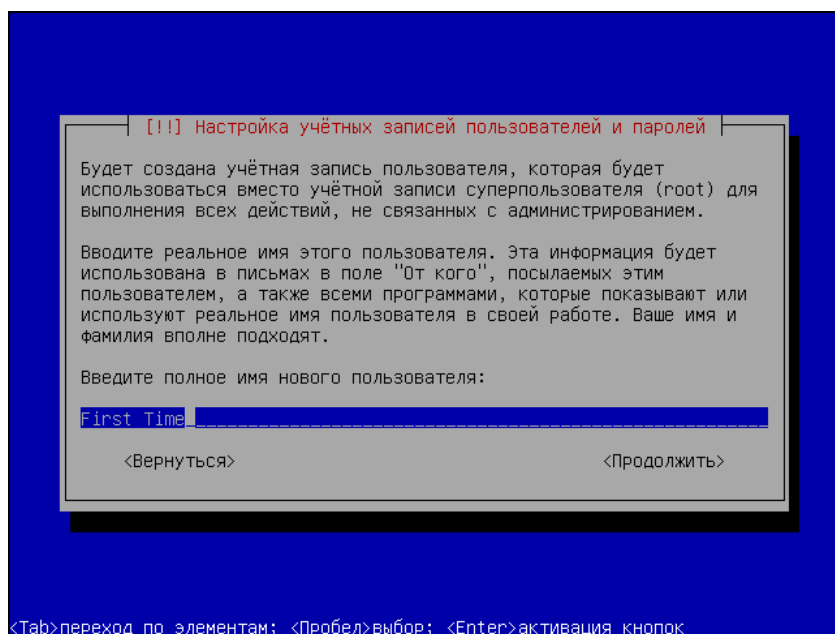


Рис. 1.9. Регистрация пользователя

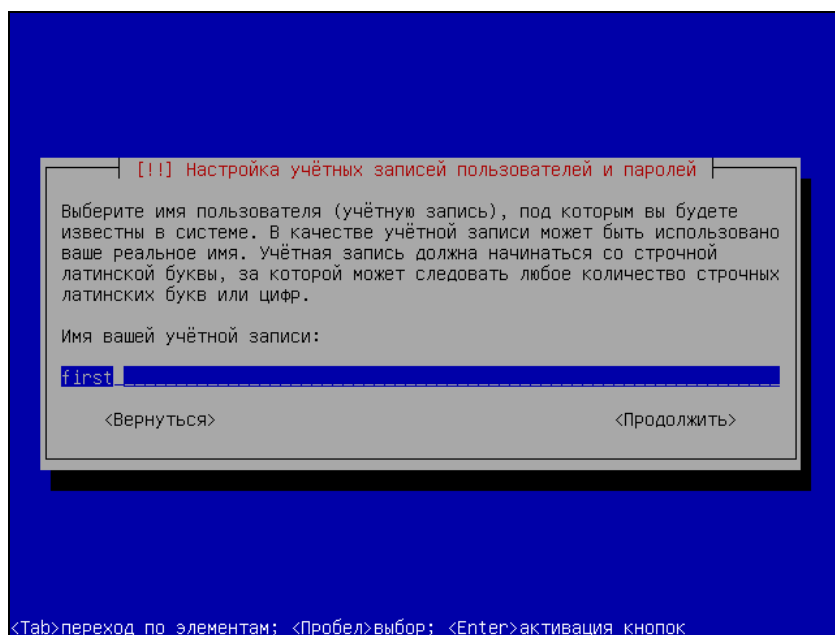


Рис. 1.10. Ввод имени пользователя

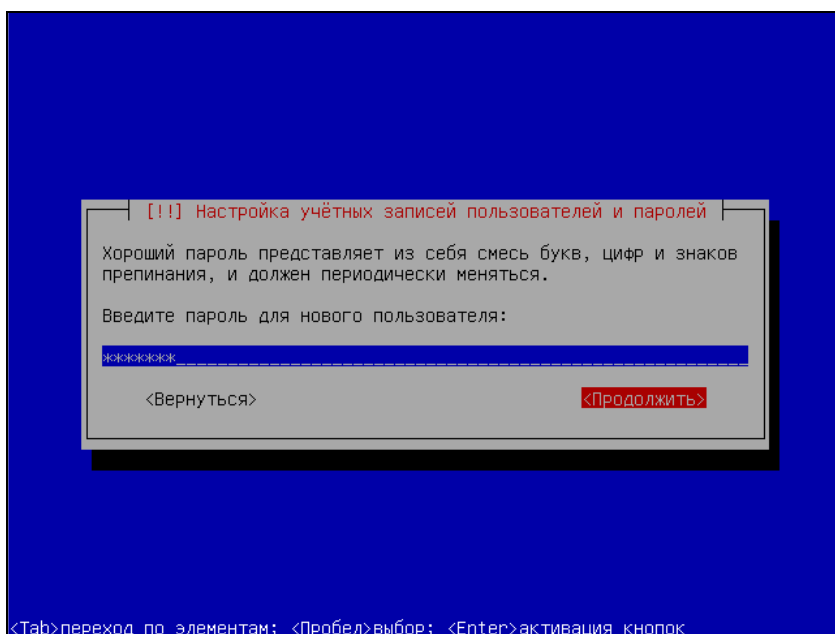


Рис. 1.11. Ввод пароля пользователя

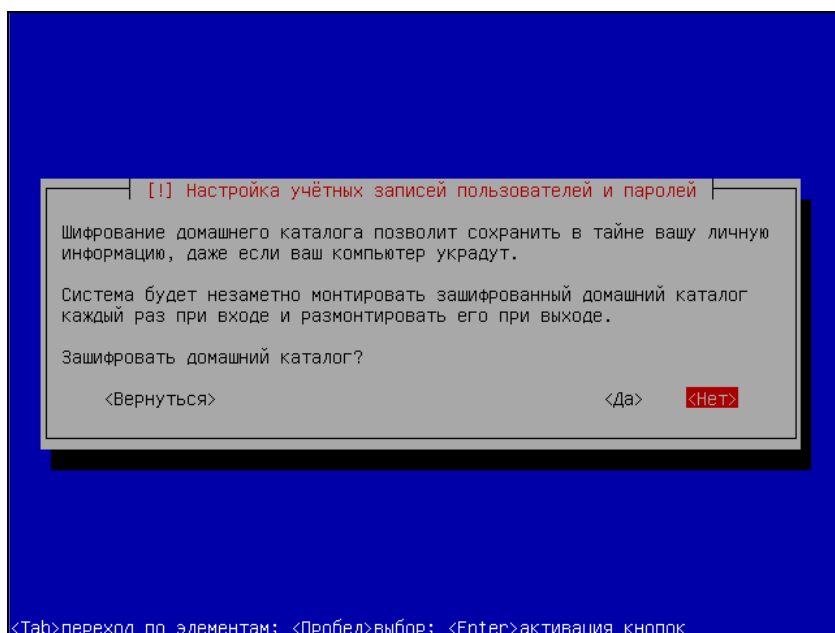


Рис. 1.12. Диалог шифрования домашнего каталога

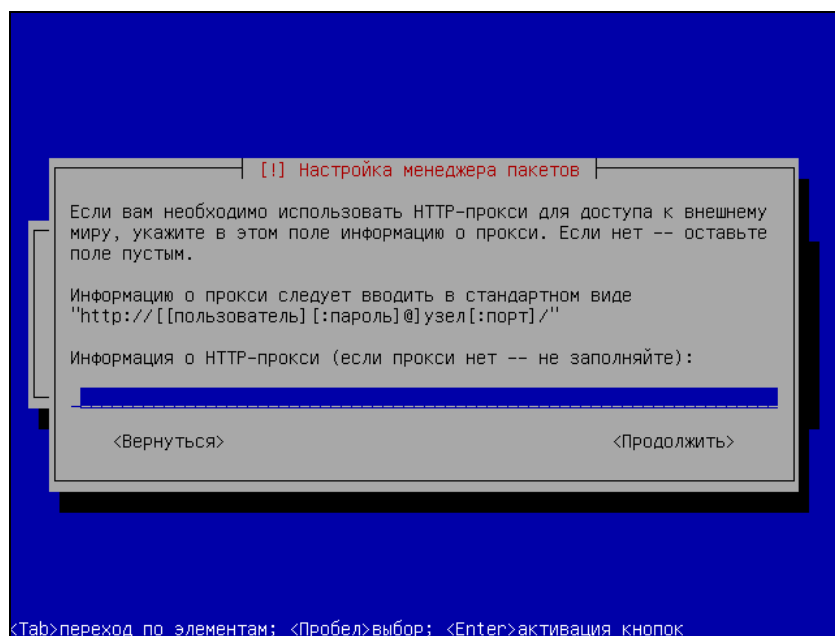


Рис. 1.13. Прокси-сервер

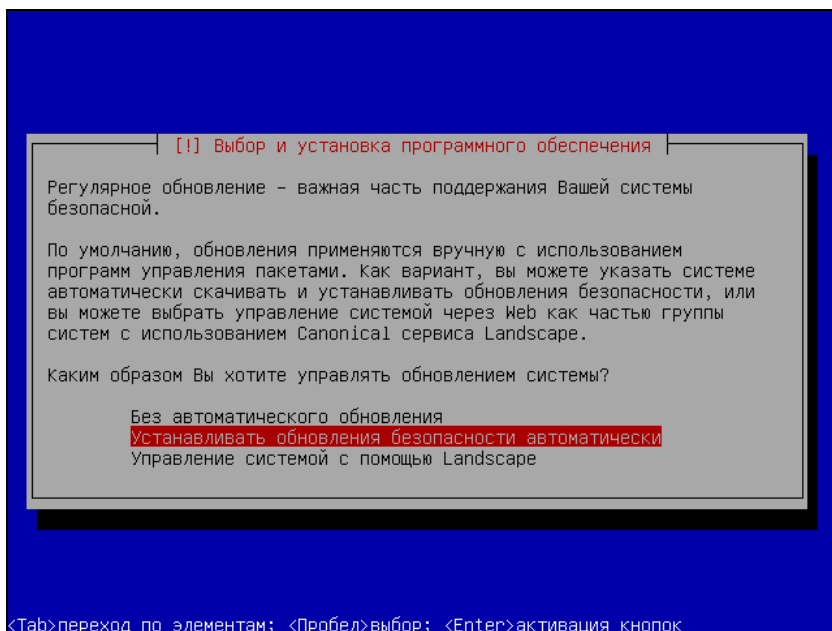


Рис. 1.14. Настройка обновлений

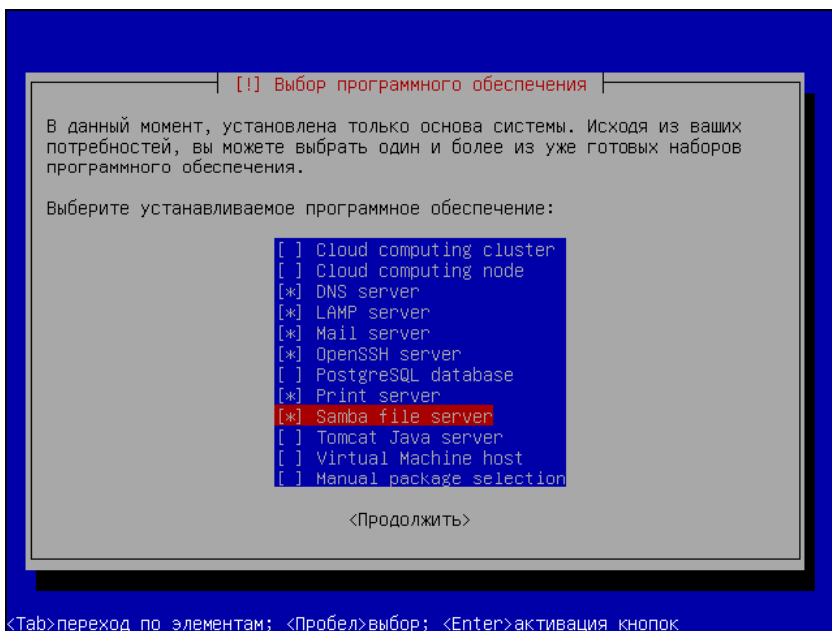


Рис. 1.15. Ввод имени пользователя

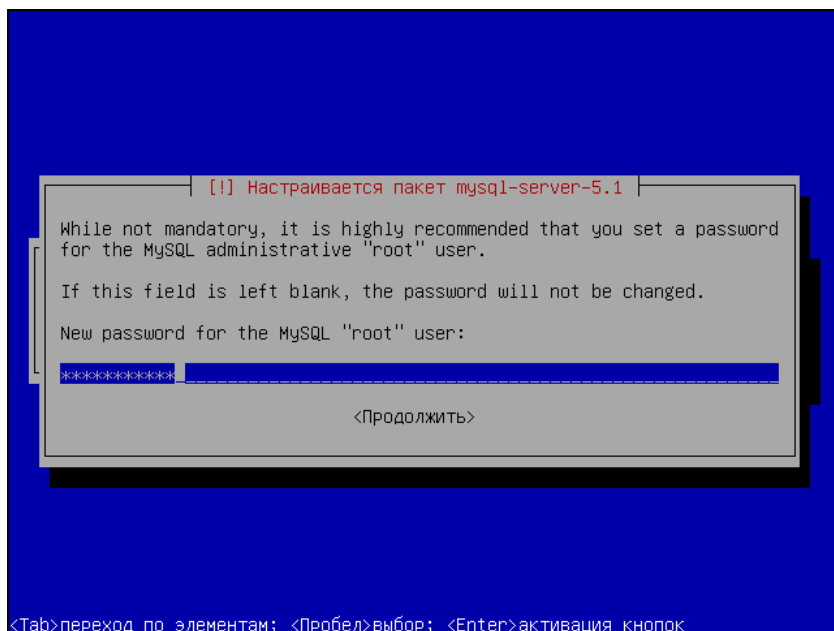


Рис. 1.16. Пароль администратора MySQL

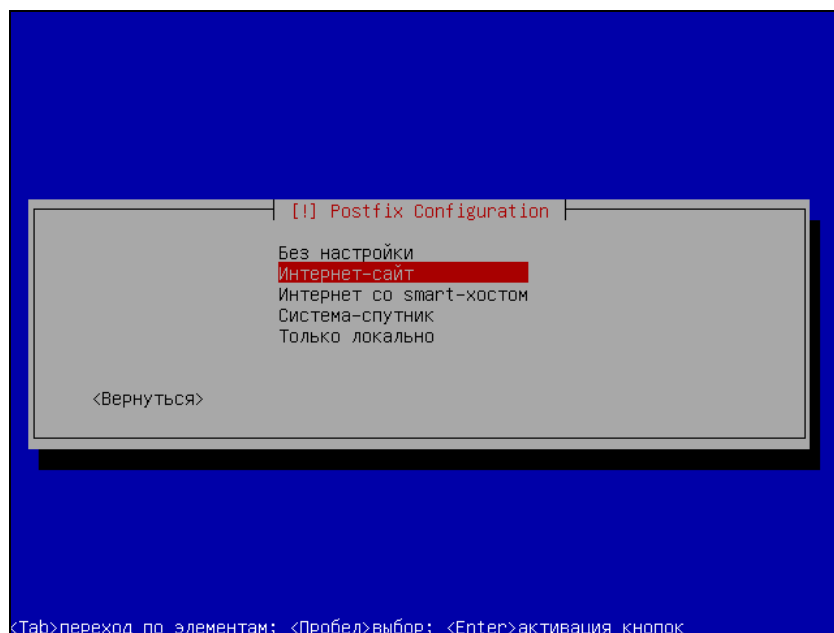


Рис. 1.17. Настройка Postfix

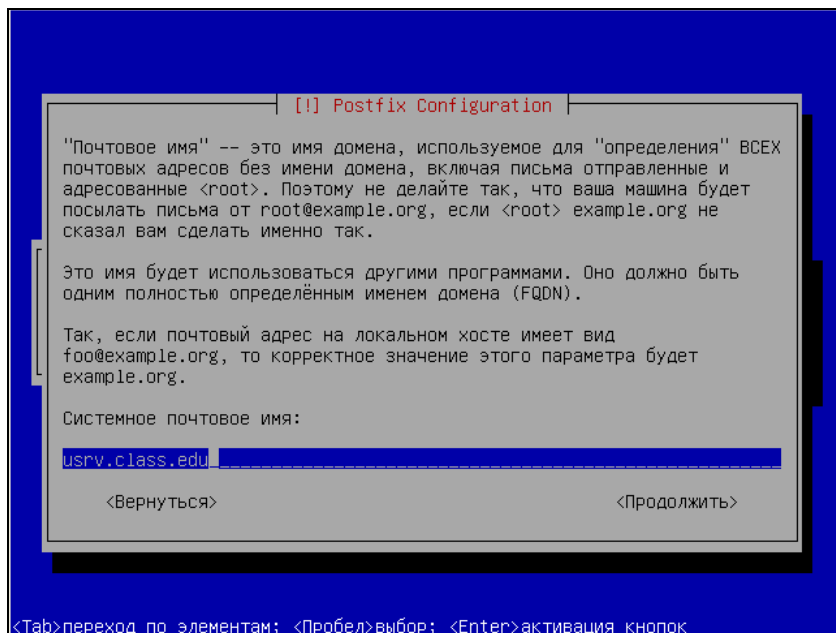


Рис. 1.18. Продолжение настройки Postfix

```
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/

System information as of Sat Jan 30 22:04:48 YEKT 2010

System load: 0.08      Memory usage: 15%   Processes:      90
Usage of / : 14.7% of 7.49GB  Swap usage:  0%   Users logged in: 0

Graph this data and manage this system at https://landscape.canonical.com/

101 packages can be updated.
59 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

first@usrv:~$ _
```

Рис. 1.19. Сеанс работы в Ubuntu

Далее программа установки предложит зарегистрировать в системе пользователя. Особенностью Ubuntu является то, что от имени пользователя, регистрируемого на данном этапе установки, будет производиться администрирование системы (с помощью команды `sudo`). В других системах обычно предлагается ввести пароль для суперпользователя (его имя — `root`), а затем зарегистрировать непривилегированного пользователя.

В диалоговом окне, показанном на рис. 1.9, предлагается ввести имя и фамилию пользователя (эти данные будут сохранены в поле комментария в учетной записи пользователя).

Следующее диалоговое окно предназначено для определения имени учетной записи пользователя. Программа установки сделает свое предложение на основе человеческого имени пользователя, введенного в предыдущем диалоговом окне. Естественно, можно ввести другое имя пользователя (рис. 1.10).

Далее будет предложено ввести пароль для пользователя (рис. 1.11), а затем подтвердить его повторным набором.

Программа установки Ubuntu предлагает зашифровать содержимое домашнего каталога пользователя. Здесь предлагается оставить каталог не зашифрованным (рис. 1.12).

После ввода данных о пользователе программа установки переходит к процессу выбора и установки программного обеспечения. Так как Ubuntu рассчитана на установку самых свежих версий программ, входящих в дистрибутив, с публичных серверов FTP и HTTP, программа установки предлагает ввести данные прокси-сервера. Если его нет, следует оставить поле ввода пустым (рис. 1.13).

Следующее диалоговое окно предназначено для настройки обновлений системы. Представляется удобным и безопасным устанавливать обновления безопасности автоматически (рис. 1.14).

Диалоговое окно выбора устанавливаемого программного обеспечения позволяет определить, какие группы программ будут установлены (рис. 1.15). Здесь также можно перейти к индивидуальному выбору пакетов для установки (последний выбор меню), однако это кропотливый труд. Нужные пакеты можно установить позже.

Следующее диалоговое окно, показанное на рис. 1.16, выводится в силу того, что была выбрана установка сервера MySQL. В нем предлагается ввести пароль администратора MySQL и подтвердить его повторным вводом.

Так как ранее была выбрана установка почтовой системы Postfix, следующее диалоговое окно выведет список стандартных вариантов ее работы, а затем будет предложено выбрать конкретный вариант (рис. 1.17).

Для правильной работы почтовой системы необходимо ввести полное доменное имя сервера, что делается в следующем диалоговом окне (рис. 1.18).

После сбора сведений программа установки Ubuntu скачивает необходимые пакеты и устанавливает их. По завершении процесса установки предлагается перезагрузить систему. На рис. 1.19 показана работающая система и первый вход в сеанс.

ЗАДАНИЯ

- Узнайте объем ОЗУ вашего компьютера.
- Проверьте наличие свободного дискового пространства.



Глава 2

Работа в оболочке Bash

В этой главе вы познакомитесь с процедурой входа в сеанс и узнаете, как GNU/Linux идентифицирует пользователей. Вы попытаетесь выполнить первые простые команды и научитесь менять пароль. Вы также узнаете, как получить список пользователей, работающих в системе. Изучив эту главу, вы приобретете базовые навыки по работе в оболочке Bash, познакомитесь с шаблонами для имен файлов, узнаете, как использовать командную подстановку, историю команд и механизм продолжения имен.

Учетные записи и вход в сеанс

GNU/Linux — это многопользовательская операционная система. Администратор регистрирует пользователей, т. е. заводит *учетные записи* (accounts) для пользователей системы. Учетная запись содержит необходимую для входа в сеанс информацию о пользователе.

В системе не могут быть зарегистрированы два пользователя, имеющие одинаковые имена. Системные утилиты, позволяющие регистрировать пользователей, не допускают этого.

Имя пользователя в GNU/Linux может содержать разнообразные символы. Однако в целях исключения возможных конфликтных ситуаций с программным обеспечением и из соображений совместимости с другими системами настоятельно рекомендуется придерживаться следующих правил:

- ☐ используйте буквы английского алфавита, цифры и символ подчеркивания;
- ☐ избегайте использования символа пробела;
- ☐ запрещено использование символа двоеточия;
- ☐ следует избегать использования метасимволов (@, #, \$ и проч.);
- ☐ не используйте специальные символы, например, символ табуляции.

В современных версиях GNU/Linux не налагается жестких ограничений на длину имени пользователя, как это было до недавнего времени во многих UNIX-системах. Тем не менее, из соображений совместимости разумно ограничивать максимальную длину имени пользователя 32 символами.

Обычная схема аутентификации (т. е. проверки подлинности) пользователя, производимая при попытке его входа в систему, требует ввода пользователем его пароля. Пароль является важнейшим инструментом защиты системы от несанкционированного доступа. Имеются несложные правила выбора паролей и работы с ними:

- ☐ пароль должен содержать не менее восьми символов;
- ☐ пароль должен быть составлен из символов в верхнем и нижнем регистрах, а также цифр и метасимволов;
- ☐ не следует выбирать пароли, основанные на имени пользователя или на его характерных чертах;
- ☐ пароль не должен быть словарным словом;
- ☐ у пароля должно быть ограничение срока действия;
- ☐ необходимо также устанавливать минимальный срок действия пароля;
- ☐ при смене пароля новый пароль должен значительно отличаться от старого;
- ☐ желательно использовать пароли, которые можно запомнить;
- ☐ нельзя произносить пароли вслух;
- ☐ следует искоренять попытки пользователей обмениваться паролями.

Процедуру входа в систему правильнее называть *входом в сеанс*, который может представлять собой работу в текстовой оболочке, например, Bourne Shell, либо же работу в графической системе X Window.

В GNU/Linux можно использовать *виртуальные терминалы*. С их помощью можно открывать сразу несколько сеансов на одном физическом терминале. Для переключения между ними используются комбинации клавиш: `<Alt>+<F1>` — переход на первый виртуальный терминал, `<Alt>+<F2>` — на второй и т. д. Обычно запускается шесть виртуальных терминалов, что определяется количеством экземпляров программы `getty` (или аналогичной), называемой *сервером терминала*.

X Window запускается на первом свободном виртуальном терминале — обычно на седьмом. Для переключения на виртуальный терминал из графики необходимо использовать `<Ctrl>+<Alt>+<F#>`, где # — номер требуемого терминала.

В примере 2.1 показано характерное приглашение для ввода имени пользователя при входе в сеанс в текстовом режиме.

Пример 2.1. Приглашение для ввода имени пользователя

```
GNU/Linux on TTY2  
login:
```

После приглашения `login:` следует ввести имя пользователя, зарегистрированного в системе. Ввод следует завершить нажатием клавиши `<Enter>`. После этого на экран будет выведено приглашение для ввода пароля (пример 2.2).

Пример 2.2. Приглашение для ввода пароля

```
password:
```

Ввод пароля не сопровождается отображением вводимых символов. Если при вводе имени пользователя или пароля была допущена ошибка, на экране будет отображено сообщение о неудачной попытке входа в сеанс (пример 2.3).

Пример 2.3. Сообщение об ошибке при попытке входа в сеанс

```
GNU/Linux on TTY2  
login:student  
password:  
login incorrect
```

Одна из наиболее распространенных ошибок, возникающих при попытке входа в сеанс, заключается в том, что пользователи ошибочно вводят свой пароль в тот момент, когда система ожидает ввода имени пользователя.

Для выхода из сеанса необходимо набрать команду `exit` или команду `logout`. Удобно также использовать комбинацию клавиш `<Ctrl>+<D>`. Это сочетание клавиш генерирует символ EOF — конец файла, который воспринимается оболочкой, как команда на завершение сеанса. Однако это сочетание клавиш может быть заблокировано с помощью специальной команды `set -o ignoreeof`.

После выхода из сеанса на экране вновь появится приглашение `login:`.

ЗАДАНИЯ

- Пройдите процедуру регистрации и войдите в сеанс. Выйдите из сеанса.
- Попытайтесь войти в сеанс, используя неверное имя пользователя. Выдается ли при этом приглашение ввести пароль?
- Попытайтесь войти в сеанс с неверным паролем. Какое сообщение выдается?
- Войдите в сеанс и выполните в нем команду `set -o ignoreeof`. Удастся ли после этого покинуть сеанс с помощью комбинации клавиш `<Ctrl>+<D>`?

Как вводить команды в shell?

Оболочка предоставляет интерфейс командной строки, в котором команды обычно вводятся с клавиатуры. Команда выполняется при нажатии клавиши `<Enter>`. Отменить ввод команды можно сочетанием `<Ctrl>+<C>`.

Когда оболочка готова исполнять команды пользователя, она выводит специальное приглашение. Оно может быть настроено в соответствии с предпочтениями пользователя, но чаще всего оно выглядит так, как показано в примере 2.4.

Пример 2.4. Типичный вид приглашения командной строки

```
[tania@work tmp]$
```

В примере 2.4 пользователь `tania` работает на компьютере с именем `work` и текущим каталогом является `tmp`.

Вид приглашения командной строки легко изменить с помощью переменной окружения `PS1`, однако следует придерживаться таких правил:

- ❑ строка приглашения должна заканчиваться символом доллара (\$) или знаком "больше" (>), если это сеанс простого пользователя (в C Shell часто используют %);
- ❑ если в сеанс зашел суперпользователь (`root`), то приглашение командной строки заканчивается символом решетки (#) — пример 2.5.

Пример 2.5. Вид приглашения командной строки сеанса суперпользователя

```
[root@work root]#
```

Если команда введена неверно, то система выводит соответствующее сообщение об ошибке (пример 2.6).

Пример 2.6. Ошибочный ввод команды

```
$ hwo
bash: hwo: command not found
```

Оболочка сообщает, что команда `hwo` не найдена.

Часто бывает необходимо очистить экран от команд, введенных ранее, и результатов их работы. Это можно сделать с помощью команды `clear` или `<Ctrl>+<L>`.

ЗАДАНИЕ

Попробуйте ввести команду `WHO` или `Who`. Что при этом происходит?

Смена пароля пользователя

В современных версиях GNU/Linux имеется специальный набор библиотек для аутентификации и авторизации пользователей, так называемая система PAM (Pluggable Authentication Module). Эта система, в частности, позволяет настраивать правила выбора нового пароля пользователя.

Администратор системы может изменить пароль любого пользователя системы. Обычные настройки PAM, касающиеся паролей:

- ☐ пароль, вводимый для пользователя администратором, проверяется на предмет удовлетворения элементарным требованиям безопасности, при нарушении которых выдается предупреждение;
- ☐ пользователь может изменить собственный пароль. Перед установкой нового пароля пользователю будет предложено ввести его старый пароль в целях безопасности. Если новый пароль не удовлетворяет требованиям безопасности, то пароль изменен не будет.

Сменить пароль можно командой `passwd`, как показано в примере 2.7.

Пример 2.7. Диалог изменения пароля пользователя

```
$ passwd
Changing password for colobok.
Enter current password:
Enter new password:
Re-type new password:
passwd: all authentication tokens updated successfully.
```

Если пароль для пользователя меняет администратор, то он должен указать имя пользователя в качестве аргумента (пример 2.8).

Пример 2.8. Смена пароля пользователя администратором

```
# passwd colobok
Changing password for user colobok.
New password:
Re-type new password:
passwd: all authentication tokens updated successfully.
```

ЗАДАНИЯ

- Придумайте себе новый пароль и попробуйте установить его.
- Удастся ли установить в качестве пароля строку `linux123?`

Идентификация пользователя

Каждый пользователь GNU/Linux имеет два идентификатора — UID и GID, которые предназначены для определения его прав доступа к файлам в системе. UID — идентификатор пользователя (User ID), GID — идентификатор первичной группы пользователя (Group ID). Каждый пользователь в Linux может быть одновременно членом многих групп, но только одна из них устанавливается на вновь создаваемые пользователем файлы и каталоги — первичная группа GID.

Для получения UID, GID и прочих групп используйте команду `id` (пример 2.9).

Пример 2.9. Идентификаторы пользователя

```
$ id
uid=503(colobok) gid=503(colobok) группы=503(colobok),22(cdrom)
```

Из информации, выведенной командой `id`, легко увидеть, что UID пользователя `colobok` — 503, первичной группой для этого пользователя является группа `colobok` с GID 503. Еще пользователь входит в группу `cdrom`, идентификатор которой 22.

В Linux для обычных пользователей назначаются UID, начиная с некоторого числа, например 100 или 500 (настройки можно увидеть в файле `/etc/login.defs`). Для привилегированных и специальных пользователей отводится нижний диапазон UID.

ЗАДАНИЯ

- Узнайте свои UID и GID.
- К каким группам вы принадлежите?
- Какие идентификаторы имеет пользователь `root`?
- Как получить только UID пользователя с помощью команды `id`? Подсказка: используйте команду `id --help` для получения помощи по команде.

Кто сейчас работает в системе?

Команда `who` выводит пользователей, работающих сейчас в системе (пример 2.10).

Пример 2.10. Команда `who`

```
$ who
tania      tty3          Sep 21 19:10 (localhost)
zhuck      tty4          Sep 21 19:10 (localhost)
colobok    pts/0         Sep 21 19:45 (black.class.edu)
```

Команда `who` отобразила список из трех пользователей системы. Во втором столбце команда показывает терминал, на котором работает этот пользователь. Пользователь `colobok` вошел в сеанс с удаленной машины через сеть, поэтому для него используется псевдотерминал. Имена псевдотерминалов начинаются со строки `pts`.

Вы можете узнать имя текущего терминала, используя команду `tty`.

Информацию о пользователях, работающих в настоящее время в системе, команда `who` извлекает из файла `/var/run/utmp` (файл бинарный). Еще более важно бывает знать, какие пользователи и когда входили в сеанс ранее. Информацию об этом можно получить с помощью команды `last`, которая извлекает данные из бинарного файла `/var/log/wtmp`.

ЗАДАНИЯ

- Используйте команду `who` с опцией `-n`. Удобнее ли она?
- Какие действия предпринимают в данный момент пользователи данной системы?
- Получите список пользователей, входивших в сеанс в вашу систему ранее.
- Выполните команду `who /var/log/wtmp`. Работает ли она?

Что такое оболочка?

Командная оболочка (shell) взаимодействует с пользователем с помощью *интерфейса командной строки (Command Line Interface, CLI)*. Оболочка позволяет пользователю запускать программы и выполнять команды операционной системы, а также выполнять сценарии оболочки.

Оболочка выводит приглашение командной строки, заканчивающееся в сеансе обычного пользователя символом доллара \$ (как правило). В сеансе суперпользователя (root) оболочка использует в качестве приглашения символ решетки (#), предупреждая о возможности порчи системы вследствие ошибочных действий.

В GNU/Linux имеется множество различных оболочек, однако стандартной является оболочка Bourne Again Shell — Bash.

Оболочка запускается при входе пользователя в сеанс. Какая оболочка будет запущена, определяется учетной записью пользователя. Переменная окружения SHELL указывает на оболочку, загружаемую при входе в сеанс (пример 2.11).

Пример 2.11. Переменная окружения SHELL

```
$ echo $SHELL  
/bin/bash
```

Символ доллара (\$) перед переменной используется для извлечения значения из нее.

Задания

- Определите имя исполняемого файла оболочки, запускаемой при входе в сеанс.
- Попробуйте выполнить команду из примера 2.11, используя вместо клавиши <Enter> комбинацию клавиш <Ctrl>+<J>.

Структура командной строки

В общем виде командная строка состоит из следующих трех частей:

- имя команды — имя исполняемого файла или встроенной команды оболочки;
- опции — дополнительные инструкции, сообщающие команде детали действий, которые она должна выполнить;
- аргументы — объекты, с которыми работает команда.

Существуют четыре основных формата командной строки, поддерживаемых GNU/Linux. Их основное отличие — стиль указания опций.

В формате UNIX98 (иначе — POSIX-формат) опции указывают в виде одиночных букв, перед которыми ставится символ `-` (тире). Формат UNIX98 краток и удобен, т. к. опции чаще всего можно указывать друг за другом.

В примере 2.12 опции следуют друг за другом после единственного символа тире.

Пример 2.12. Команда в стиле UNIX98

```
$ ls -dl /etc/default
```

В примере 2.12 команда `ls` выполнена с опциями `-d` (отображать информацию о каталоге, а не о файлах в нем) и `-l` (выводить подробную информацию). Каталог `/etc/default` указан в качестве аргумента.

В BSD-формате тире перед опциями может отсутствовать, причем также можно указывать несколько опций подряд (пример 2.13).

Пример 2.13. Команда в стиле BSD

```
$ ps aux
```

Команда `ps` выводит список процессов в системе. Три используемые опции: `a`, `u`, `x` изменяют формат вывода информации о процессах в системе. Интересно, что в BSD-системах во многих командах допускается указывать или опускать тире перед опциями. При этом работа команды чаще всего не изменяется. В GNU/Linux есть команды, способные работать с опциями в UNIX98- и BSD-формате, причем использование опций в разных стилях приводит к изменениям в поведении команды.

В длинной нотации GNU опции записываются целыми словами, перед которыми надо указывать двойное тире (`--`). Удобство этого формата состоит в интуитивной ясности опций. В соответствии со стандартом программирования GNU все команды поддерживают специальную опцию `-help` (пример 2.14), выводящую краткую справку о команде. Также поддерживается опция `--version` для вывода версии программы.

Пример 2.14. Команда в стиле GNU

```
$ gzip --help
```

Команда `gzip` позволяет сжимать файлы. Однако в данном случае она просто выводит информацию о себе, т. к. установлена опция `--help`.

Команды, связанные с графической системой X Window, традиционно используют собственный формат длинных опций, в котором указывается единственный символ тире перед опцией.

Если после длинной опции в стиле GNU должно следовать значение, в таком случае между опцией и значением должен быть символ "равно". Далее приведен пример 2.15, в котором длинная опция GNU устанавливает значение, передаваемое программе.

Пример 2.15. Установка значений в длинных опциях GNU

```
$ ./configure --prefix=/opt/sfw
```

В примере 2.15 с помощью опции `--prefix` команде `configure` передан дополнительный параметр — путь к целевому каталогу.

Команды, связанные с графической системой X Window, традиционно используют собственный формат длинных опций, в котором указывается единственный символ тире перед опцией.

Пример 2.16. Команда в стиле X Window

```
$ xterm -display :0.0
```

Команда в примере 2.16 запускает графический эмулятор терминала — программу `xterm`. Опция `-display` отмечена единственным символом тире.

В командах GNU/Linux встречаются и другие варианты указания опций. Для каждой команды в документации обязательно описан синтаксис ее командной строки.

Задания

- Выполните команду `ls -l ~`, выводящую содержимое вашего домашнего каталога в подробном формате.
- Разберите структуру командной строки предыдущей команды. Где в командной строке имя команды, опции и аргументы?
- Выполните команду `ls --help`. Какой формат опций использован?

Популярные оболочки GNU/Linux

Первая оболочка, которая появилась в UNIX, — это Bourne Shell, названная по имени ее создателя. Эта оболочка была рассчитана на терминалы — теле-тайпы с крайне ограниченными возможностями редактирования текста. Поэтому в Bourne Shell нет никаких возможностей редактирования командной строки. В GNU/Linux вместо нее используется Bourne Again Shell — `bash`. Имя этой оболочки образовано игрой слов и значит "Возрожденная оболочка".

В GNU/Linux наиболее часто используются следующие оболочки:

- ❑ `bash` — Bourne Again Shell (используется по умолчанию);
- ❑ `pdksh` — свободная реализация оболочки Korn Shell (`ksh`);
- ❑ `tcsh` — Enhanced C shell, улучшенный вариант C Shell (`csh`);
- ❑ `zsh` — Z Shell, дальнейшее развитие `ksh`.

Эти оболочки обладают различной функциональностью и даже разными командами. Поэтому при запуске сценария оболочки (скриптов) необходимо убеждаться в его соответствии используемой оболочке.

Оболочки `ksh` и `bash` способны корректно выполнять сценарии, написанные для Bourne Shell, т. к. являются ее наследницами и совместимы с ней. Но сценарии C Shell не могут быть корректно выполнены в Bash.

Оболочки, установленные в системе, указаны в файле `/etc/shells` (пример 2.17).

Пример 2.17. Файл `/etc/shells`

```
$ cat /etc/shells
/bin/sh
/bin/bash
/sbin/nologin
/bin/ksh
/bin/tcsh
/bin/csh
```

Загрузка оболочки достигается запуском ее исполняемого файла (пример 2.18).

Пример 2.18. Запуск Enhanced C Shell

```
$ tcsh
$ ps
PID TTY          TIME CMD
```

```
2414 pts/1    00:00:00 bash
3047 pts/1    00:00:00 tcsh
3070 pts/1    00:00:00 ps
```

В примере 2.18 команда `tcsh` запускает оболочку Enhanced C Shell. Команда `ps` выводит список процессов, из которого видно, что оболочка `tcsh` запущена.

Запускать другую оболочку имеет смысл, например, для выполнения в ней сценария, рассчитанного на эту оболочку.

Задания

- Получите список оболочек, установленных на вашей системе.
- Временно запустите оболочку `tcsh` и выйдите из нее.
- Какая версия оболочки `tcsh` установлена в вашей системе?
- Поддерживает ли оболочка `tcsh` GNU-опцию `--help`?
- Имеется ли в вашей системе какая-либо версия Korn Shell?

Встроенные и системные команды

Все команды GNU/Linux делятся на два больших класса: встроенные и системные. Встроенные команды выполняются самой оболочкой, а системные команды — это исполняемые файлы. *Встроенные команды* представляют собой процедуры оболочки. Пример встроенной команды — `cd`. Она изменяет текущий каталог. *Системные команды* обычно находятся в одном из следующих каталогов: `/bin`, `/sbin`, `/usr/bin`, `/usr/sbin`, `/usr/local/bin` или `/usr/local/sbin`.

Пользователь может написать собственные системные команды и использовать их. Такие команды чаще всего размещаются в домашнем каталоге пользователя или подкаталоге `bin` домашнего каталога.

Для многих встроенных команд имеются системные аналоги. Например, для встроенной Bash-команды `pwd`, выводящей имя текущего каталога, имеется системный аналог `/bin/pwd`. Если команда вызвана без указания пути к ней, то выполняется ее встроенный в оболочку вариант (при его наличии). Если при вызове команды указан путь, то всегда выполняется системная команда.

Задания

- Используя команду `ls /bin`, получите список системных команд. Есть ли среди них знакомые вам команды?
- Выполните встроенную команду `pwd --help` и системную `/bin/pwd --help`. Есть ли разница?
- Выполните команду `help type`. Для чего нужна команда `type`?
- Является ли команда `disable` встроенной в оболочку?

Редактирование и исполнение команд

При работе в командной строке можно использовать клавиши управления курсором и клавиши редактирования. Однако во многих случаях привычные клавиши управления курсором не работают. Так, например, очень часто при неверных настройках раскладки клавиатуры не работают клавиши управления курсором и клавиша .

В табл. 2.1 приведены клавиатурные сочетания, которые могут быть использованы при работе в командной строке.

Таблица 2.1. Клавиатурные сочетания Bash

Клавиши	Действие
<Ctrl>+	Курсор влево
<Ctrl>+<F>	Курсор вправо
<Ctrl>+<A>	Курсор в начало строки
<Ctrl>+<E>	Курсор в конец строки
<Ctrl>+<H>	Удаление символа перед курсором
<Ctrl>+<D>	Удаление символа в позиции курсора
<Ctrl>+<J>	Ввод (аналогично нажатию клавиши <Enter>)
<Ctrl>+<L>	Очистка экрана
<Alt>+<T>	Перемена мест аргументов
<Ctrl>+<C>	Остановка выполнения команды или сброс командной строки
<Ctrl>+<Z>	Приостановление выполнения задания
<Ctrl>+<R>	Поиск команды в истории

Нажатие комбинации клавиш <Ctrl>+<Z> приводит к немедленной приостановке активного задания. О сочетании <Ctrl>+<Z> будет рассказано позже.

В случае если необходимо ввести длинную команду, которая не помещается в одну строку, нужно воспользоваться символом обратной косой черты (\) и продолжить ввод на следующей строке. Наоборот, можно вводить несколько команд в одной строке, разделяя их символом точки с запятой (;).

Пример 2.19. Ввод нескольких команд в одной командной строке

```
$ cd /tmp; pwd
/tmp
```

В примере 2.19 объединены команды `cd /tmp` и `pwd`. Первая меняет текущий каталог на `/tmp`, а вторая — выводит имя текущего каталога.

Если команды отделены друг от друга с помощью двух амперсандов (`&&`), то вторая команда будет выполнена только в случае успешного выполнения первой. Напротив, при необходимости выполнять вторую команду только в случае неудачи первой следует использовать две вертикальные черты `||`. Эти синтаксические конструкции оболочки позволяют организовать условное выполнение команд (пример 2.20).

Пример 2.20. Условное выполнение команд

```
$ cd /tmp && pwd
/tmp
```

В примере 2.20 первая команда `cd /tmp` меняет текущий каталог на `/tmp`. Если эта команда завершается успехом, то выполняется команда `pwd`, которая выводит имя текущего каталога.

ЗАДАНИЯ

- Введите команду `ls -FR ~` так, чтобы имя команды, опции и аргументы занимали отдельные строки. То есть каждый элемент командной строки должен быть введен в отдельной строке. Используйте `\`.
- Выполните команды `hostname` и `date` в единой командной строке.
- Введите командную строку `/etc ls`. Как поменять местами команду и аргумент?

Переменные оболочки и окружения

Bash позволяет временно сохранять данные в *переменных оболочки*. Переменные оболочки размещаются в памяти автоматически при присвоении им значения через знак равенства. В примере 2.21 переменной `VAR1` присваивается значение `Privet!`.

Пример 2.21. Переменная оболочки

```
$ VAR1=Privet!
$ echo $VAR1
Privet!
```

Для извлечения значения переменной необходимо перед ее именем установить знак доллара (\$). Команда `echo` выводит значение переменной `VAR1`.

Имя переменной должно состоять только из букв и цифр. Допускается применение символа подчеркивания. Первый символ в имени переменной должен быть либо буквой, либо символом подчеркивания. Пробелы в имени переменной не допускаются. Желательно (но не обязательно) использовать в именах переменных только большие буквы для того, чтобы не путать имена команд и имена переменных.

Если переменная должна содержать строку с пробелами, то строку следует экранировать с помощью одиночных или двойных кавычек (пример 2.22). Другой вариант: перед пробелами можно ставить экранирующий символ обратной косой черты (\).

Пример 2.22. Переменная, содержащая строку с пробелом

```
$ VAR1='Bolshoy Privet!'
$ echo $VAR1
Bolshoy Privet!
```

При необходимости добавления строки к значению переменной имя переменной следует взять в фигурные скобки для отделения имени переменной от последующей строки (пример 2.23).

Пример 2.23. Экранирование имени переменной

```
$ VAR1=${VAR1}ZZ
$ echo $VAR1
Bolshoy Privet!ZZ
```

Для получения списка всех переменных, определенных в текущей оболочке, следует использовать команду `set` без аргументов. Если вы хотите уничтожить переменную, то для этого достаточно выполнить команду `unset`, указав в качестве ее аргумента имя переменной.

Переменные оболочки доступны только в той оболочке, в которой они были описаны. Однако можно сделать переменную доступной для дочерних процессов этой оболочки, преобразовав ее в переменную окружения с помощью команды `export` (пример 2.24). Команда `export` записывает переменную в окружение оболочки, которое копируется в окружение дочерних процессов оболочки.

Пример 2.24. Экспортирование переменной

```
$ VAR1=Privet
$ export VAR1
$ pdksh
$ echo $VAR1
Privet
$ exit
```

Переменная `VAR1` в примере 2.24 получила значение `Privet`, а затем экспортирована. Это сделало ее доступной в дочернем процессе — в порожденной оболочке `pdksh`.

Все переменные окружения могут быть получены с помощью команды `env`.

Окружение — это один из способов передачи информации процессов в системе друг другу (табл. 2.2). Часто изменение значения какой-либо переменной окружения приводит к изменению поведения программ. Например, если значение переменной `HOME` установлено неверно, то команда `cd` не будет возвращать вас в домашний каталог.

Таблица 2.2. Важнейшие переменные окружения

Переменная окружения	Содержимое
HOME	Путь к домашнему каталогу пользователя
LOGNAME и USER	Имя пользователя
MAIL	Путь к почтовому ящику пользователя
PATH	Путь поиска исполняемых файлов
PS1	Вид приглашения оболочки
PWD	Имя текущего каталога
OLDPWD	Имя предыдущего каталога
SHELL	Оболочка, указанная в учетной записи
TERM	Тип терминала
LANG	Тип локализации (локаль)
HOSTNAME	Имя хоста
SHLVL	Номер оболочки (при входе в сеанс — 1, для дочерней — 2 и т. д.)

Переменная окружения `PS1` определяет вид приглашения оболочки (пример 2.25). Вид приглашения кодируется специальными символами. Символ `\u` устанавливает вывод имени пользователя. Символ `\w` устанавливает вывод имени текущего каталога, а символ `\h` отображает в строке приглашения Bash имя хоста. В Bash существуют также и другие специальные символы для переменной `PS1`. Они описаны в документации на Bash.

Пример 2.25. Переменная окружения `PS1`

```
[user1@work tmp]$ echo $PS1  
[\u@\h \W]\$
```

При выходе из сеанса установленные значения переменных будут стерты, т. к. оболочка завершит работу. Для автоматической установки при входе в сеанс необходимых переменных окружения их следует инициализировать в файлах профиля.

Переменные окружения, общие для всех пользователей, хранятся в файле `/etc/profile`, а настройки, специфичные для конкретных пользователей, хранятся в одном из файлов домашнего каталога пользователя: либо в `~/.bash_profile`, либо `~/.bash_login`, либо `~/.profile`. Эти файлы выполняются автоматически при каждом входе в сеанс. При запуске оболочки Bash из командной строки выполняется файл профиля `~/.bashrc`.

Подробнее о файлах профиля будет рассказано далее.

Задания

- Выполните команду `cd && ls -a`. Имеются ли в каталоге файлы профиля?
- Используя окружение, получите имена текущего и домашнего каталогов.
- Создайте новую переменную `NEWVAR` со значением 1982 и проверьте, доступна ли она в порожденной оболочке.
- Экспортируйте `NEWVAR` и проверьте, доступна ли она в порожденной оболочке.
- Получите списки переменных оболочки и переменных окружения с их значениями.

История команд

Оболочка Bash позволяет выполнять уже исполненные команды. Выполненные команды сохраняются в файле `~/.bash_history` (в переменной окружения `HISTFILE` можно указать другой файл). Количество команд, запоминаемых

в файле истории, устанавливается с помощью переменной окружения HISTFILESIZE.

Историю команд можно получить с помощью `history` (пример 2.26).

Пример 2.26. Команда `history`

```
$ history
...
685 echo $HISTFILE
686 echo $HISTFILESIZE
687 history
```

В примере 2.26 показаны лишь последние команды из файла истории. Перед каждой командой из файла истории выводится ее номер, с помощью которого эту команду можно вызвать заново. Наиболее простой способ для этого — ввести в командной строке знак восклицания и номер команды для повтора (пример 2.27).

Пример 2.27. Повторный вызов команды по номеру в истории

```
$ !685
echo $HISTFILE
```

Последнюю выполненную команду можно выполнить снова, если ввести в командной строке два знака восклицания (`!!`).

Удобно вызывать из истории команды, вводя после знака восклицания первые символы их имен. Например, если необходимо вновь выполнить команду `ls /tmp`, достаточно ввести в командной строке `!l`. При этом история будет просмотрена с конца до тех пор, пока не будет найдена команда с подходящими первыми символами.

Можно вызвать команду из истории, указав строку символов, содержащуюся в любом месте командой строки. Для этого следует ввести эту строку после знака восклицания и знака вопроса (`!?`).

Исключительно удобное сочетание клавиш `<Ctrl>+<R>` позволяет производить поиск команд в истории по любым символам в командной строке. Нажмите комбинацию клавиш `<Ctrl>+<R>`, и вам будет предложено ввести символы из командной строки, причем найденная команда будет отображаться до своего исполнения (пример 2.28).

Пример 2.28. Поиск команды в истории с помощью комбинации клавиш <Ctrl>+<R>

```
$
(reverse-i-search)`ec': echo $USER
$ echo $USER
colobok
```

Пример 2.28 показывает, как производится поиск команды после нажатия комбинации клавиш <Ctrl>+<R>. Пользователь ввел два символа из имени команды (не обязательно лидирующие), найденная команда была показана, далее пользователь нажал клавишу <Enter>, и найденная команда была выполнена.

Задания

- Выполните последнюю введенную команду заново.
- Найдите в файле истории команду `echo` и выполните ее.
- Вызовите последнюю введенную команду `echo` по первым двум буквам ее имени.
- Вызовите команду, содержащую подстроку `ho`.
- Используя <Ctrl>+<R>, выполните команду, у которой в строке была опция `-F`.
- Выведите полный список команд в файле истории.
- Установите максимальное количество команд в файле истории равным 9999. Сохранится ли это значение при выходе из сеанса?
- В чем отличие команды `!!` от нажатия комбинации клавиш <Ctrl>+<P>?

Автоматическое дополнение командной строки

Bash предоставляет удобный механизм дополнения имен файлов и команд по первым символам их имен. Bash пытается продолжить введенные символы как имя команды или файла после нажатия на клавишу <Tab>. Если оболочка не может продолжить имя файла, то выводится звуковой сигнал. Это может происходить по двум причинам:

- ☐ файла или команды с таким именем не существует;
- ☐ имеется несколько вариантов продолжения строки.

Во втором случае при повторном нажатии клавиши табуляции Bash выводит список возможных подстановок, ориентируясь на который пользователь

может ввести еще несколько символов командной строки и снова нажать клавишу табуляции.

Механизм продолжения в Bash действует не только для имен файлов и команд. Если строка начинается с одного из символов: \$, ~ или @, то Bash попытается дополнить строку как:

- имя переменной оболочки (\$);
- имя пользователя (~);
- имя хоста (@).

Пример 2.29. Продолжение командной строки

```
$ ls /etc/sys<Tab><Tab>
sysconfig/  sysctl.conf  syslog.conf
$ ls /etc/sys
```

В примере 2.29 пользователь начал вводить команду `ls /etc/sysctl.conf`, но после ввода части командной строки `ls /etc/sys` он решил воспользоваться механизмом продолжения имен файлов. После первого нажатия клавиши `<Tab>` был получен звуковой сигнал, свидетельствующий либо об отсутствии файла, либо о наличии нескольких вариантов продолжения. После второго нажатия клавиши табуляции был получен список продолжений. Далее пользователь может продолжить ввод команды.

Задания

- Введите команду `ls -ld /`, а затем получите список всех возможных продолжений командной строки.
- Получите значение переменной окружения `HISTFILESIZE`, пользуясь механизмом дополнения имен.

Псевдонимы команд (aliases)

Многие команды, требуемые в повседневной работе, слишком длинны и неудобны. Для ускорения набора часто повторяющихся сложных команд им можно назначить псевдонимы с помощью встроенной команды `alias`.

Команда `alias`, выполненная без аргументов, выводит список всех уже определенных псевдонимов (пример 2.30).

Пример 2.30. Команда alias

```
$ alias
alias ls='ls --color=tty'
alias vi='vim'
```

Новый псевдоним также создается командой `alias` (пример 2.31).

Пример 2.31. Создание нового псевдонима

```
$ alias ll='ls -l'
$alias
alias ls='ls --color=tty'
alias vi='vim'
alias ll='ls -l'
```

Здесь создан новый псевдоним с именем `ll`, указывающий на команду `ls -l`. Для удаления псевдонима достаточно указать его в качестве аргумента команде `unalias`. Команда `unalias -a` удаляет все описанные в оболочке псевдонимы.

ЗАДАНИЯ

- Получите список псевдонимов, определенных в текущем сеансе.
- Псевдоним `ls` выводит имена файлов с пометкой их цветами. Почему при вызове команды `/bin/ls` этого не происходит?
- Создайте псевдоним `lf` для команды `ls -F` и выполните команду `lf /bin`.
- Удалите только что созданный псевдоним.

Командная подстановка

Командная подстановка (command substitution) — это один из мощнейших механизмов Bash. Суть его сводится к тому, что результат выполнения одной команды автоматически передается в качестве аргументов другой команде. Внутренняя команда в командной подстановке должна быть заключена в открывающие обратные кавычки ```. Результаты ее работы автоматически подставляются в командную строку внешней команде. Командная подстановка выглядит следующим образом:

```
внешняя_команда `внутренняя_команда`
```

Вместо открывающих кавычек (``) можно использовать конструкцию `$()`, т. е.:

```
внешняя_команда $(внутренняя_команда)
```

Исследуем в качестве примера две команды: `which`, которая выводит путь к заданной команде, и `ls -l`, отображающую подробную информацию о файле. Командная подстановка позволяет выполнить их так, что результат работы `which` (внутренняя команда) будет подставлен в качестве аргумента команде `ls -l` (внешняя).

Пример 2.32. Командная подстановка

```
$ which rpm
/bin/rpm
$ ls -l /bin/rpm
-rwxr-xr-x 1 rpm rpm 93692 Oct 12 /bin/rpm
$ ls -l `which rpm`
-rwxr-xr-x 1 rpm rpm 93692 Oct 12 /bin/rpm
```

Первая команда, приведенная в примере 2.32, определяет путь к программе `rpm` — `which rpm`. Путь `/bin/rpm`, полученный в результате работы первой команды, вручную подставлен в качестве аргумента команде `ls -l`, которая выдала информацию об этом исполняемом файле. Последняя команда примера 2.32 демонстрирует командную подстановку: команда `ls -l `which rpm`` выполняет автоматическую подстановку результата работы `which` команде `ls -l`.

Результаты, выводимые при работе внутренней команды в командной подстановке, могут быть занесены в переменную.

Задания

- С помощью командной подстановки выведите командой `ls -l` подробную информацию о самом исполняемом файле команды `ls`.
- Пользуясь механизмом командной подстановки, получите список процессов в системе, исполняющихся от вашего имени. Для этого следует использовать команду `ps -u`, выводящую список процессов для заданного пользователя, и команду `whoami`, выводящую имя текущего пользователя.
- Создайте переменную `MYGROUP`, содержащую информацию о вашей первичной группе. Используйте для этого `id -gn`.

Вычисление арифметических выражений

В командной строке можно вычислять выражения, заключенные либо в квадратные скобки, либо в двойные круглые скобки (пример 2.33). Перед скобками должен стоять символ `$`, а результаты выражений можно передавать как аргумент какой-либо команде или назначать переменной.

Пример 2.33. Выселение значения арифметического выражения

```
$ echo $((192*512))
98304
```

В примере 2.33 вычислено значение выражения $192 \cdot 512$. Команда `echo` выводит в стандартный поток вывода строку, указанную в качестве аргумента.

В примере 2.34 результат выражения занесен в переменную.

Пример 2.34. Назначение вычисленного значения переменной

```
$ v1=5
$ v2=$((v1/2))
$ echo $v2
2
```

Переменной `v1` присвоено значение 5. Далее производится присвоение переменной `v2` результата деления значения `v1` на два. В силу того что в Bash возможно вычисление только целочисленных выражений, то в результате значением стало `v2` число 2.

Задания

- Получите десятичное выражение для 16 мегабайт.
- Назначьте это значение переменной `mb16`. Может ли имя переменной быть `16mb`?
- Назначьте переменной `TTL` значение, соответствующее числу секунд в двух неделях. Выведите ее значение на экран.
- Получите удвоенное значение переменной окружения `HISTFILESIZE`.
- Выполните команды `echo $((7%3))` и `echo $((7%4))`. Что вычислено?

Шаблоны подстановки и перечисление

Оболочки специальным образом интерпретируют некоторые метасимволы, например, *, подставляя вместо них имена файлов. Такие метасимволы называются *шаблонами подстановки*. Символы подстановки позволяют оболочке произвести поиск файлов, имена которых удовлетворяют шаблону, и подставить их в командную строку.

В качестве команды для испытания шаблонов подстановки можно использовать команду `echo`, которая просто выводит на экран то, что ей задано в качестве аргумента.

Символ "звездочка" (*) является шаблоном для любого количества любых символов в именах файлов и даже для их отсутствия. Единственный символ, который не удовлетворяет этому шаблону, — лидирующая точка в именах скрытых файлов. Таким образом, подставив звездочку в качестве аргумента команде `echo`, мы либо увидим в результате саму звездочку, если в каталоге нет файлов, либо оболочка подставит команде `echo` имена всех файлов в каталоге в командную строку (пример 2.35).

Пример 2.35. Шаблон *

```
$ cd /etc/xinetd.d
$ echo *
amanda amandaidx amidxtape chargen chargen-udp cvs daytime daytime-udp
echo echo-udp eklogin ekrb5-telnet gssftp klogin krb5-telnet kshell ktalk
rsync telnet tftp time time-udp uucp
```

В примере 2.35 производится переход в каталог `/etc/xinetd.d`, в котором есть несколько файлов, имена которых подставляются оболочкой в виде аргументов команде `echo`, т. к. они удовлетворяют шаблону `*`.

Имеются особые имена файлов, начинающиеся с точки, например, `.bashrc`. Такие файлы называются *скрытыми*, т. к. их имена не выводятся командой `ls` без специальных опций. Шаблоном для имен скрытых файлов является `.*` (точка и звездочка). Вообще, любые шаблоны для скрытых файлов должны начинаться с символа "точка".

Символ `?` заменяет один символ в имени файла, который должен находиться в той позиции, где находится знак вопроса (пример 2.36).

Пример 2.36. Шаблон ?

```
$ echo .??????  
.bashrc .config .emacs~ .gconfd .gnome2 .isotmp .mcoprc .themes
```

В примере 2.36 получен список скрытых файлов (имена начинаются с точки), в именах которых после точки имеется шесть любых символов.

Можно еще более сузить диапазон поиска, используя набор символов, заключенных в квадратные скобки. Применение такого шаблона обозначает, что в данном месте должен находиться один любой символ из заданного множества. Для того чтобы указать допустимый диапазон символов в шаблоне, необходимо использовать квадратные скобки, а в них — требуемый диапазон. Например, `[0-9]` — шаблон подходит для любых цифр, а `[a-zA-Z]` — шаблон для букв английского.

Пример 2.37. Шаблон множества

```
$ echo .[bcd]*  
.bash_history .bash_logout .bash_profile .bashrc .cshrc .ddd
```

Пример 2.37 демонстрирует вывод имен скрытых файлов, в именах которых после точки стоит символ `b`, `c` или `d`.

Если необходимо указать набор символов, не входящих во множество, следует установить знак восклицания после открывающей скобки. Например, `[!abc]` — множество любых символов, кроме `a`, `b` или `c`.

Очень удобен, хотя и не относится к шаблонам, механизм *перечисления* Bash. Он позволяет задать множество вариантов, которое должна перебрать оболочка, составляя последовательно все варианты, заданные в фигурных скобках (пример 2.38).

Пример 2.38. Перечисление

```
$ echo .bash{rc,_profile}  
.bashrc .bash_profile
```

В примере 2.38 использован механизм перечисления для обращения к двум файлам: `.bashrc` и `.bash_profile`. Имена этих файлов имеют общую подстроку `.bash`, короткая вынесена за фигурные скобки. В фигурных скобках через запятую перечислены варианты продолжения: `rc` и `_profile`.

Задания

- Составьте шаблон для любых имен файлов, вторая буква которых должна быть гласной буквой английского алфавита.
- Как сделать то же, что и в предыдущем задании, но так, чтобы вторая буква не была гласной, а могла быть любым другим символом?
- Как получить с помощью шаблонов все имена файлов длиной шесть символов, последний из которых либо не цифра, либо буква A?
- Проверьте, можно ли использовать символ звездочки внутри квадратных скобок. Если да, то что значит такая конструкция?
- Получите список файлов в /etc, имена которых соответствуют шаблону `rc?.d`.



Глава 3

Помощь и документация

В этой главе читатель узнает, как получать помощь в GNU/Linux, где искать и находить ответы на практические вопросы и как находить нужную документацию. Вы познакомитесь со страницами помощи `man` и гипертекстовой системой GNU Texinfo. Здесь также обсуждаются вопросы поиска информации по GNU/Linux в Интернете.

Сообщения об ошибках

Пользователи часто вводят команды, задавая неверные опции, аргументы или неправильно конструируя команду. Большинство команд выводит в таких случаях сообщения об ошибках и подсказки о том, как их следует использовать (пример 3.1).

Пример 3.1. Ошибочный вызов команды

```
$ pwd -h
bash: pwd: -h: invalid option
pwd: usage: pwd [-PL]
```

Команда `pwd`, выводящая путь к текущему каталогу, была выполнена с неверной опцией. Поэтому было получено сообщение об ошибке и краткая подсказка.

Команды GNU выдают подсказку при использовании опции `--help` (пример 3.2).

Пример 3.2. Опция `--help` команд GNU

```
$wc --help
Usage: wc[OPTION]...[FILE]...
Print line, word, and byte counts for each FILE
```

ЗАДАНИЯ

- Проверьте, работает ли опция `--help` с командой `id`.
- Попробуйте вместо `--help` использовать `-h` с командой `id`.
- Вызовите команду `wc --help` без аргументов, изучите подсказку по ней.

Встроенная помощь оболочки Bash

Оболочка Bash предоставляет собственную помощь по встроенным командам. Список встроенных команд Bash можно увидеть с помощью команды `help`.

Если этой команде указать аргумент — имя встроенной команды Bash, то по этой команде будет выведена подсказка (пример 3.3).

Пример 3.3. Встроенная команда помощи `help`

```
$ help pwd
```

```
pwd: pwd [-PL]
```

Print the current working directory. With the `-P` option, `pwd` prints the physical directory, without any symbolic links; the `-L` option makes `pwd` follow symbolic links.

ЗАДАНИЯ

- Получите помощь по встроенной команде `cd`. Для чего предназначена переменная окружения `CDPATH`?
- Получите помощь по команде `alias`. Что позволяет сделать опция `-p`?

Страницы помощи `man`

Система `man` (от англ. *manual* — руководство) имеется в любой UNIX-системе. Это основное средство получения подробной информации о командах, структуре файлов конфигурации, системным вызовам и прочему. Система `man` не рассчитана на обучение, она предоставляет подробное описание команд и конфигурационных файлов.

Для получения помощи необходимо вызвать команду `man` с аргументом — именем команды или иного требуемого объекта (пример 3.4).

Пример 3.4. Получение помощи `man`

```
$ man ls
```

В GNU/Linux имеется множество файлов и команд с одинаковыми именами. Как же объяснить `man`, какая информация нужна? Например, имеется команда для изменения пароля пользователя `passwd`, а учетные записи хранятся в файле `/etc/passwd`. Поэтому все страницы `man` разделены на секции, приведенные в табл. 3.1.

Таблица 3.1. Секции `man`

Секция	Информация
1	Описание команды пользователя
2	Описание системных вызовов ядра
3	Описание библиотек
4	Информация о файлах устройств и иных специальных файлах
5	Форматы конфигурационных файлов
6	Помощь по играм
7	Макросы, кодировки, информация для программистов
8	Команды системного администрирования
9	Процедуры и функции ядра

Часто используются секции с другими именами, например, `n` или `1x`, соответственно, для команд языка TCL и для пользовательских команд с графическим интерфейсом.

Для указания команде `man` требуемой секции ее номер вводят в командной строке `man`, а затем — имя требуемой страницы помощи (пример 3.5).

Пример 3.5. Получение помощи из заданной секции `man`

```
$ man 3 zlib
```

Эта команда выводит информацию из третьей секции `man` о библиотеке `zlib`. Сама по себе система `man` не занимается отображением страниц помощи на экран. Она находит среди всех страниц помощи нужную, форматирует ее и передает программе постраничного просмотра, используемой в системе по умолчанию. Обычно в GNU/Linux используется `less`. В табл. 3.2 приведены команды для `less`.

Таблица 3.2. Команды *less*

Команда	Действие
<Ctrl>+<N>, <↓>	Следующая строка
<Ctrl>+<P>, <↑>	Предыдущая строка
<Ctrl>+<V>, <PgDn>	Страница вниз
<Alt>+<V>, <PgUp>	Страница вверх
<Пробел>	Следующая страница
</>строка	Поиск подстроки вниз
<?>строка	Поиск подстроки вверх
<n>	Найти следующее вхождение искомой подстроки
<q>	Выход

Часто необходимо получить помощь о команде или файле, не зная точного названия. В этом случае помогают опция `-k` команды `man` или команда `apropos`. Каждая страница `man` начинается с обязательного раздела `NAME`, содержащего описание объекта поиска. Команда `man -k` ищет строку, заданную после опции, во всех имеющихся страницах, просматривая раздел `NAME` (пример 3.6).

Пример 3.6. Поиск в `man` по подстроке

```
$ man -k clock
CLOG_csnc           (4) - synchronize clocks for adjusting times in
merge
adjtimex            (2) - tune kernel clock
alarm               (2) - set an alarm clock for delivery of a signal
clock               (3) - Determine processor time
clockdiff           (8) - measure clock difference between hosts
hwclock             (8) - query and set the hardware clock (RTC)
```

Тот же самый результат будет получен при выполнении команды `apropos clock`.

Если необходимо в разделе `NAME` отыскивать точное вхождение строки, то следует использовать команду `man -f` (пример 3.7) или же `whatis`.

Пример 3.7. Поиск в man по ключевому слову

```
$ man -f clock
clock                (3)  - Determine processor time
```

Команда `apropos` отыскивает подстроку, а команда `whatis` — слово целиком.

Задания

- Получите помощь о самой системе `man`.
- Для чего нужна опция `-P` команды `man`?
- Получите все страницы, называющиеся `groff`.

Файлы страниц man

Страницы `man` форматируются с помощью специального языка разметки `ROFF`. В GNU была создана собственная программа `groff`, реализующая функции программы `nroff`, предназначенной для интерпретации `ROFF` и использующейся в коммерческих UNIX-системах.

Страницы помощи `man` хранятся в отдельных файлах, имя которых состоит из имени раздела и суффикса — секции `man`. Например, файл `rm.1` относится к разделу информации о команде `rm`, раздел принадлежит первой секции системы `man`. Чаще всего страницы `man` хранятся в сжатом с помощью утилит `gzip` или `bzip2` виде, поэтому к расширению имени файла добавляется, соответственно, `gz` или `bz2`. Стандартное место хранения страниц `man` — каталог `/usr/share/man`.

В каталоге `/usr/share/man` имеются подкаталоги с именами `man1`, `man2`, ..., `man9`. Несложно догадаться, что это — каталоги, в которых хранятся страницы соответствующих разделов. Так, например, в каталоге `/usr/share/man/man8` хранятся страницы помощи по восьмой секции `man`. В случае если система локализована и в ней установлены национальные страницы помощи, например, русифицированная система `man`, то в каталоге `/usr/share/man` создается специальный подкаталог с локализованными страницами. Русские страницы `man` установлены в каталоге `/usr/share/man/ru`. Дополнительные приложения и команды, установленные в системе, обычно размещают свои страницы помощи в каталоге `/usr/share/local/man`.

Часто стандартного списка секций не достаточно, поэтому используют специальные секции, предназначенные для помощи по таким программам, как

графическая подсистема X Window. Для этой подсистемы выделена специальная секция — `lx`.

В системе могут быть установлены разные системы помощи `man`, в том числе и в нестандартных местах файловой системы. Переменная окружения `MANPATH` позволяет устанавливать пути доступа к различным каталогам, содержащим страницы `man`. Для установки переменной `MANPATH` можно использовать специальную команду `manpath`.

Имеется специальный файл конфигурации `/etc/man.config`, который позволяет настраивать параметры системы `man`.

Для разработки собственных страниц `man` можно использовать любой текстовый редактор, однако файл страницы помощи должен следовать формату `groff`. Для этого в файле должны быть выделены разделы, приведенные в табл. 3.3, и он должен быть размечен с помощью макросов `groff`.

Таблица 3.3. Стандартные разделы страниц `man`

Раздел	Назначение
NAME	Указывает информацию, которая будет использована при поиске по ключевому слову
SYNOPSIS	Формат вызова программы, опции и аргументы
DESCRIPTION	Описание объекта (программы, файла, библиотеки и т. п.)
OPTIONS	Подробное описание опций
FILES	Файлы, связанные с данной командой
AUTHOR	Имя автора с указанием адреса электронной почты
SEE-ALSO	Указатели на другие страницы <code>man</code> и иную документацию
COPYRIGHT	Права собственности, политика распространения и т. п.

Задания

- Найдите все файлы страниц `man`, касающиеся любых объектов, называющихся `exit`.
- Вы собираетесь разработать игру для X Window — `xzombie`. Как следует назвать файл страницы `man` для этой программы?
- В каком каталоге следует установить эту страницу?
- Вы собираетесь опубликовать обсуждаемую игру в виде пакета, устанавливаемого с помощью системы установки программного обеспечения. В какой стандартный каталог, в таком случае, следует установить страницу помощи?

GNU Texinfo

Система Texinfo была разработана как свободная альтернатива man в рамках проекта GNU. Страницы man по многим командам и утилитам GNU предлагают обратиться к системе Texinfo, как к более полному источнику информации. Texinfo представляет собой гипертекстовую документацию в специальном формате, организованную по иерархическому принципу. Точки разветвления называются узлами (node). В системе Texinfo имеется возможность переходить между соседними узлами, родительскими и дочерними узлами, а также по специальным гипертекстовым ссылкам.

Документация Texinfo имеет особое значение при работе со сложными программами, для которых создать полные и подробные страницы помощи man затруднительно или вообще невозможно. Примерами таких программ являются коллекция компиляторов GNU `gcc` и система Kerberos. Так как документация на них очень сложна и обширна, ее гораздо удобнее изучать с помощью связанных ссылками страниц.

Для получения базовой помощи по системе Texinfo достаточно выполнить команду `info` без аргументов. В таком случае команда `info` выводит страницу документации, отображающую наивысший уровень разделов информационной системы.

Команде `info` можно указывать аргумент — имя интересующей программы или системы. Если соответствующая информация имеется, будет отображена гипертекстовая страница Texinfo. Если в самой Texinfo соответствующей информации нет, но существует подходящая страница man, то она и будет выведена.

Пример 3.8. Вызов помощи Texinfo по команде `ls`

```
$ info ls
```

Команда в примере 3.8 выведет помощь Texinfo по команде `ls`.

Файлы документации Texinfo обычно находятся в каталоге `/usr/share/info`.

В системе Texinfo можно пользоваться перечисленными в табл. 3.4 командами.

Таблица 3.4. Команды Texinfo

Команда	Действие
<n>	Следующий узел
<p>	Предыдущий узел
<u>	Переход к родительскому узлу
<l>	Предыдущая страница
<Ctrl>+<V>, <PgDn>	Страница вниз
<Alt>+<V>, <PgUp>	Страница вверх
<Alt>+< >	В начало страницы
<Alt>+< > >	В конец страницы
<s>строка	Поиск строки на странице
<q>	Выход из Texinfo

Задания

- Получите помощь по команде `who`, пользуясь системой `info`.
- Найдите раздел документации `info`, относящийся к теме переходов по гипертекстовым ссылкам, перейдите в родительский узел.

Документация программ

В каталоге `/usr/share/doc` находятся подкаталоги с именами вида: "приложение-версия". Например, `/usr/share/doc/httpd-2.2.13` — каталог, содержащий некоторую документацию для сервера HTTP Apache версии 2.2.13. В дистрибутивах SUSE каталоги документации пакетов, установленных в системе, находятся в `/usr/share/doc/packages`.

В этих каталогах размещается дополнительная документация от разработчиков программного обеспечения. Это могут быть подробные руководства по использованию программ или же просто файлы `README`.

Насколько подробна документация, находящаяся в этих каталогах, зависит от разработчика. Многие большие программы сопровождаются весьма обширной документацией, и тогда документация может быть выделена в отдельный пакет.

Наиболее важные файлы в каталогах `/usr/share/doc/*` :

- ❑ **README** — краткая информация о продукте, последние изменения, не вошедшие в документацию, предупреждения и, возможно, инструкции по установке;
- ❑ **INSTALL** — инструкции по установке.

По многим программам документация поставляется в формате HTML.

Задания

- Проверьте, какая документация для оболочки Bash имеется в вашей системе.
- Имеется ли дополнительная информация о системе `man`?

Источники информации в Интернете

При необходимости получения помощи по Linux необходимо вначале обратиться к сайту производителя дистрибутива. Серьезные производители дистрибутивов сопровождают их значительным количеством документации. Так как настройка приложений отличается в различных дистрибутивах, то начинать следует именно отсюда.

Далее при необходимости за получением документации следует обратиться на сайт производителя конкретной программы.

Важным ресурсом для получения помощи по GNU/Linux является сайт The Linux Documentation Project <http://www.tldp.org>. Он содержит следующую информацию: HOWTO — пошаговые инструкции, FAQ — ответы на часто задаваемые вопросы, а также ссылки на разнообразные обучающие документы и программы (Tutorials).

HOWTO — подробная рецептурная информация по разнообразным аспектам использования GNU/Linux. Чаще всего HOWTO (как сделать) — это тематические описания различных программ, методов установки, настройки и работы с ними, созданные авторами-добровольцами на основе собственного опыта работы с описываемыми ими программами и системами. Профессиональный уровень и достоверность многих документов не всегда достаточны, однако HOWTO — это главный способ получения пошаговых инструкций для настройки GNU/Linux-систем, полезный как для начинающего пользователя, так и для профессионала.

FAQ — часто задаваемые вопросы (Frequently Asked Questions). Представляют собой подборки вопросов, задаваемых пользователями программ, на которые чаще всего отвечают сами разработчики этих программ. Если вы в процессе настройки GNU/Linux-системы натолкнулись на проблему,

то вполне вероятно, что кто-либо также наталкивался на нее, и проблема уже решена. Именно такого рода информация может быть найдена в FAQ.

Также можно обратиться за дополнительной информацией по следующим адресам:

- ❑ <http://www.debian.org> — официальный сайт Debian GNU/Linux;
- ❑ <http://www.ubuntu.com> — сайт популярного дистрибутива Ubuntu;
- ❑ <http://www.opensuse.org> — популярный дистрибутив Open SUSE;
- ❑ <http://www.redhat.com> — сайт компании Red Hat;
- ❑ <http://www.gentoo.org> — сайт Gentoo Linux, изобилующий руководствами;
- ❑ <http://www.howtoforge.com> — неплохой набор учебных материалов;
- ❑ <http://www.sf.net> — база данных программного обеспечения Source Forge;
- ❑ <http://www.ibm.com> — на сайте IBM имеется большое количество статей и документации по GNU/Linux;
- ❑ <http://www.gnu.org> — сайт проекта FSF GNU;
- ❑ <http://www.linuxshop.ru> — форум системных администраторов;
- ❑ <http://linux.slashdot.org> — новостной сайт с постоянными обновлениями.

ЗАДАНИЯ

- В вашей системе может быть установлена документация HOWTO. Проверьте ее наличие в каталоге `/usr/share/doc`.
- На сайте <http://www.linuxgazette.org> имеется архив разнообразных статей по тематике GNU/Linux. Исследуйте его.



ЧАСТЬ II

ОСНОВЫ

Глава 4



Работа с файлами и каталогами

Большинство объектов, с которыми приходится иметь дело в GNU/Linux, — это файлы. В данной главе вы изучите устройство файловой системы и освоите команды манипулирования файлами и каталогами. Также здесь рассматриваются специальные типы файлов и жесткие связи.

Система файлов и каталогов

Логически файловая структура в GNU/Linux организована в виде единой древовидной иерархии. Древовидная структура организована с помощью каталогов, содержащих файлы и подкаталоги. Каждый каталог может иметь множество подкаталогов, но у каждого подкаталога имеется только один родительский каталог. На каких физических носителях ни хранились бы файлы, в GNU/Linux они всегда находятся на одной из ветвей единой древовидной файловой структуры.

Вершиной файловой структуры является *корневой каталог* (root directory). Имя корневого каталога: `/`. У корневого каталога нет родительского каталога, вернее, он сам является для себя родительским.

Файлы в GNU/Linux являются основополагающими объектами, поскольку вся работа с данными, устройствами компьютера, процессами и прочим обеспечивается посредством файлов.

Обычные файлы (plain files) обеспечивают хранение данных в компьютере. Они представляют собой именованный набор блоков данных на устройстве хранения.

Древовидная структура образуется за счет использования каталогов, которые могут содержать файлы и другие каталоги. Каталоги являются особым типом файлов, предназначенным для поддержки иерархической структуры файловой

системы. По сути, каталог — это таблица, содержащая перечень имен файлов, находящихся в нем.

В корневом каталоге обычно не содержатся какие-либо пользовательские файлы (чаще всего в корневом каталоге находятся исключительно подкаталоги). Файлы пользователей принято хранить в их домашних каталогах, системные файлы хранятся в специальных каталогах и т. д.

Последовательность имен каталогов, которые требуется пройти от корневого каталога для доступа к файлу, называется *путем* (path). Для разделения имен вложенных каталогов применяется символ /.

В GNU/Linux пользователь имеет большую свободу в назначении имен файлам. Единственное ограничение: имя файла не может содержать символы / и \0 (null). Разумно назначать файлам осмысленные имена и избегать излишнего использования метасимволов (например, звездочки или решетки) в именах файлов, т. к. это может привести к проблемам в работе многих приложений.

Прописные и строчные буквы различаются (case sensitive), т. е. имена файлов TheFile и thefile относятся к двум разным файлам.

В различных каталогах могут находиться разные файлы с одинаковыми именами. Поэтому для однозначной идентификации файла необходимо применять *полное* или *абсолютное имя файла*. Оно состоит из пути (path) к нему в дереве каталогов и собственно имени файла. Таким образом, имена файлов бывают двух типов:

- *абсолютные* — те имена, которые начинаются с символа "косая черта" (/) — корневого каталога — и указывают последовательность подкаталогов, которые необходимо пройти для достижения файла;
- *относительные* — их имена не начинаются с косой черты и, следовательно, показывают путь доступа к файлам относительно текущего каталога.

Имя файла может содержать точки (.). В GNU/Linux, в отличие, например, от MS-DOS, никакого особого значения точки в именах файлов не имеют. Однако для удобства принято считать часть имени файла, находящуюся после точки, — суффиксом (или иначе — расширением) имени файла. Суффиксы сообщают пользователю информацию о типе файла. Так, например, файл myarch.tgz является tar-архивом, сжатым утилитой gzip. Может быть несколько суффиксов: tarball.tar.gz.

Файлы, у которых точка является первым символом в имени, являются скрытыми и командой ls не выводятся. Тем не менее, список этих файлов можно получить, пользуясь командой ls с опцией -a (all) или опцией -A.

Для вывода списка всех файлов используется команда ls -a, в том числе и скрытых, в текущем каталоге. Эта команда выводит имя файла .hidden.

Помимо него выведены еще два имени файлов — `.` (точка), т. е. имя текущего каталога, и `..` — имя родительского каталога. Команда `ls` — а также выводит имена скрытых файлов наряду с обычными именами, но имена текущего и родительского каталога — нет.

ЗАДАНИЯ

- Имя домашнего каталога — `~`. Выведите его содержимое.
- Выведите содержимое домашнего каталога, включая скрытые файлы.
- Можно ли использовать с командой `ls` опцию `--all`?

Имена файлов и команда `ls`

К файлам, находящимся в текущем каталоге, не требуется указывать путь доступа. Если аргументом файловой команды является имя файла без пути, то действие команды будет применено к файлу в текущем каталоге.

Команда `pwd` выводит полное имя текущего каталога (пример 4.1).

Пример 4.1. Имя текущего каталога

```
$ pwd
/home/user1
```

Полные имена файлов иначе называются *абсолютными именами* (absolute pathname), вместо них можно пользоваться также *относительными именами* (relative pathnames), в которых путь к файлу указывается относительно текущего каталога. Имена файлов, не начинающиеся с символа `/`, являются относительными. Например, `anna/referat.txt` — относительное имя файла, находящего в подкаталоге `anna` текущего каталога.

Команда `ls` выводит содержимое каталогов, указанных в качестве аргументов, или содержимое текущего каталога, если аргументов нет.

Пример 4.2. Вывод содержимого нескольких каталогов

```
$ ls /usr/local /etc/default/
/etc/default/:
nss  useradd

/usr/local:
bin  etc  games  include  lib  libexec  sbin  share  src
```

Обычно при регистрации в системе нового пользователя ему назначается его домашний каталог, в котором он может хранить личные файлы. При входе в сеанс обычных пользователей текущими становятся их домашние каталоги. Имена домашних каталогов чаще всего совпадают с именами пользователей — владельцев этих каталогов.

Стандартное место для размещения домашних каталогов пользователей в GNU/Linux — каталог /home. Например, домашний каталог пользователя anna — /home/anna.

При пользовании оболочкой Bash существует короткий путь для указания имени домашнего каталога: имя ~ указывает на домашний каталог пользователя, вошедшего в систему, а ~имяпользователя — на домашний каталог указанного пользователя.

Опция -l команды ls позволяет получить подробную информацию о файлах.

Пример 4.3. Получение подробной информации о файлах

```
$ ls -l /etc/default/  
итого 16  
-rw-r--r-- 1 root root 962 Apr  2  2009 nss  
-rw----- 1 root root  96 Фев 12  2009 useradd
```

Первая строка вывода команды ls -l /etc/default сообщает о суммарном дисковом пространстве в 512-байтных блоках, которое занимают файлы. Это пространство больше, чем размер файлов. Причиной этого является то, что каждый из этих двух файлов физически занимает в файловой системе 4 Кбайт (блок данных).

Далее команда ls -l выводит строки с информацией о файлах. В первом столбце этих строк сообщается тип файла, далее права доступа к файлу, количество имен файла (жестких связей), владелец файла, первичная группа владельца, размер файла, дата изменения и имя файла. Права владения и права доступа, а также возможность наличия у файла более одного имени будут рассмотрены далее.

Обсудим имеющиеся в GNU/Linux типы файлов и соответствующие им обозначения в первом столбце листинга команды ls -l:

- - — обычные файлы;
- d — каталоги;
- l — символические ссылки (содержат указатели на другие файлы);

- ❑ `b` — блочные устройства (специальные файлы, предназначенные для обращения к устройствам, информация на которые записывается и считывается оттуда блоками, например, жесткий диск);
- ❑ `c` — символьные устройства (специальные файлы, предназначенные для посимвольного ввода/вывода с таких устройств, как терминал или мышь);
- ❑ `p` — именованный канал (PIPE или FIFO, они являются одним из вариантов организации межпроцессного взаимодействия);
- ❑ `s` — сокет (sockets, предназначенные для организации сетевого межпроцессного взаимодействия).

Другая часто используемая опция команды `ls` — это опция `-F` (пример 4.4). При указании этой опции после имен каталогов выводится `/`, после имен исполняемых файлов — `*`, после символических ссылок — `@`.

Пример 4.4. Опция `-F` команды `ls`

```
$ ls -F ~  
Desktop/ intro.txt scr1.sh*
```

Команда `ls` с опцией `-F` вывела содержимое домашнего каталога пользователя с использованием символов подсказки. Здесь `Desktop` — каталог, т. к. после его имени выводится знак `/`. Файл `intro.txt` — обычный файл. А сценарий `scr1.sh` является исполняемым файлом, т. к. после его имени выведен символ `*`.

Для получения информации собственно о каталогах, а не о файлах, содержащихся в них, необходимо воспользоваться опцией `-d` команды `ls` (пример 4.5). Чаще всего опция `-d` применяется совместно с опцией `-l` команды `ls`.

Пример 4.5. Получение подробной информации о каталоге

```
$ ls -ld /etc  
drwxr-xr-x  87 root      root          6064 Окт  7 06:16 /etc
```

Если бы опция `-d` здесь отсутствовала, то была получена информация не о каталоге, а о файлах, содержащихся в нем.

Задания

- Узнайте, какая опция команды `ls` позволяет вывести файлы в порядке, отсортированном по времени изменения этих файлов.
- Можно ли получить список файлов, отсортированный по их размеру?
- Получите несортированный список файлов в каталоге `/tmp`.

Перемещение по файловой системе

Команда `cd` предназначена для смены текущего каталога. Для перехода в заданный каталог необходимо вызвать команду `cd` с аргументом — именем каталога.

Пример 4.6. Смена текущего каталога

```
$ pwd
/home/user1
$ cd /etc
$ pwd
/etc
```

В данном листинге команда `cd /etc` сделала текущим каталог `/etc`.

Если команда `cd` вызвана без аргументов, она осуществляет переход в домашний каталог пользователя. Если же ее аргумент — знак "тире", то команда `cd` переходит в предыдущий каталог.

Задания

- Перейдите в подкаталог `/tmp`.
- С помощью одной команды перейдите в подкаталог `local/bin` каталога `/usr`.
- Перейдите в домашний каталог. Проверьте, какой каталог является текущим.
- Вернитесь в предыдущий каталог.
- Изучите в `man`, какие переменные окружения связаны с командой `cd`.

Создание и удаление файлов и каталогов

Простейший способ создать пустой файл заключается в использовании перенаправления "пустого ввода" в файл с помощью команды `> файл` (пример 4.7). Особенностью данного способа является то, что если файл существовал до выполнения этой команды, его содержимое будет стерто.

Пример 4.7. Создание файла

```
$ > empty
$ ls -l empty
-rw-rw-r-- 1 user1 user1 0 Окт 12 21:33 empty
```

Команда `touch` дает возможность создать один или несколько файлов. Имена создаваемых файлов задаются аргументами команды `touch` (пример 4.8). Если в качестве аргумента указан файл, который уже существует, то у этого файла в результате выполнения команды `touch` будет изменена дата модификации.

Пример 4.8. Команда `touch`

```
$ date >f1;cat f1;ls -l f*;sleep 60;touch f1 f2;date;cat f1;ls -l f*
ЧТВ Окт 15 21:26:34 YEKST 2009                # cat f1
-rw-rw-r-- 1 user1 user1 37 Окт 15 21:26 f1    # ls -l f*
ЧТВ Окт 15 21:27:34 YEKST 2009                # cat f1
ЧТВ Окт 15 21:26:34 YEKST 2009                # date
-rw-rw-r-- 1 user1 user1 37 Окт 12 21:27 f1    # ls -l f*
-rw-rw-r-- 1 user1 user1  0 Окт 12 21:27 f2    # ls -l f*
```

Командная строка примера 4.8 содержит сразу восемь команд. Первая из них `date > f1` создает файл `f1`, содержащий текущую дату. Это достигается с помощью перенаправления потока вывода команды `date`. Команда `cat f1` выводит содержимое этого файла, команда `ls -l f*` — подробную информацию о файле `f1`. Обратите внимание на время его модификации — 21:27:34. Далее команда `sleep 60` задерживает ход выполнения последующих команд на 60 секунд. После этого команда `touch f1 f2` изменяет дату модификации файла `f1` и создает новый пустой файл `f2`. Команда `date` затем выводит текущее время. Команда `cat f1` снова выводит содержимое файла `f1` для демонстрации того, что оно не изменилось. Но команда `ls -l` показывает, что даты модификации файлов `f1` и `f2` соответствуют времени выполнения команды `touch`.

Для удаления файла необходимо воспользоваться командой `rm` (пример 4.9).

Пример 4.9. Удаление файлов

```
$ ls
f1 f2 f3
$ rm -f f1 f2
$ rm -i f3
rm: удалить пустой обычный файл `f3'? y
$ ls
$
```

Изначально в текущем каталоге находились три файла: `f1`, `f2` и `f3` (пример 4.9). Первые два из них были удалены командой `rm -f`. Эта команда удаляет файлы без запроса на подтверждение, т. к. используется опция `-f`. Наоборот, если необходимо выводить запрос на удаление каждого файла, указанного в качестве аргумента команды `rm`, требуется использовать опцию `-i`.

Команда `rm`, вызванная без опций, не задает никаких вопросов по поводу необходимости удаления файлов, как это происходит с опцией `-i`. При использовании шаблонов подстановки в качестве аргументов команды `rm` настоятельно рекомендуется предварительно проверить шаблон командой `ls`.

Также в целях безопасности рекомендуется использовать псевдоним для команды `rm`, активизирующий ее интерактивный режим работы по умолчанию (пример 4.10).

Пример 4.10. Псевдоним для команды `rm`

```
$ alias rm='rm -i'
```

Приведенный в примере 4.10 псевдоним следует разместить в файле профиля пользователя `.bash_profile`, либо в `.bashrc`. Использование его включит интерактивный режим работы команды `rm` по умолчанию. Важно понимать, что затруднительно, а часто просто невозможно восстановить удаленный файл.

Если необходимо рекурсивно удалить каталог со всем его содержимым, надо использовать команду `rm` с опцией `-r` (пример 4.11).

Пример 4.11. Рекурсивное удаление каталога с содержимым

```
$ ls -R d1
d1:
direc1  file3

d1/direc1:
Remember
$ rm -rf d1
$ ls -R d1
ls: d1: No such file or directory
```

Каталог `d1` (пример 4.11) содержал файлы и подкаталоги. После его удаления командой `rm -rf` команда `ls` сообщает о его отсутствии.

Командой `rm -rf` следует пользоваться с особой осторожностью, т. к. установленная опция `-f` запрещает выводить какие-либо сообщения об удалении файлов. Разумно перед использованием такой команды переходить в родительский каталог удаляемого каталога, и затем указать относительное имя удаляемого каталога. Этого правила следует придерживаться обязательно при работе в сеансе суперпользователя.

Команда `mkdir` создает каталоги, а при использовании опции `-p` команды `mkdir` можно создать целую ветвь вложенных каталогов (пример 4.12).

Пример 4.12. Создание каталогов

```
$ mkdir dir1
$ cd dir1
$ ls
$ mkdir -p mydir1/mydir2/mydir3
$ ls -R mydir1
mydir1:
mydir2

mydir1/mydir2:
mydir3

mydir1/mydir2/mydir3:
```

Пример 4.12 демонстрирует, как с помощью команды `mkdir` был создан каталог `dir1`. Далее пользователь перешел в созданный каталог командой `cd`. Затем, используя ключ `-p` команды `mkdir`, пользователь создал целую ветвь вложенных каталогов: `mydir1/mydir2/mydir3`.

Команда `rmdir` позволяет удалять каталоги, если они пустые (пример 4.13).

Пример 4.13. Удаление каталогов

```
$ mkdir emptyDir
$ ls -F
emptyDir/  mydir1/

$ rmdir *
rmdir: `mydir1': Directory not empty
```

В примере 4.13 был создан еще один каталог `emptyDir`. Затем пользователь попытался удалить оба каталога, однако удален был лишь пустой каталог `emptyDir`. Каталог `mydir1` был пропущен, поскольку он непустой.

Команда `rmdir -p` способна удалить ветвь вложенных пустых каталогов. Если в пути некоторый каталог будет содержать файлы, то будут удалены все пустые каталоги в пути до первого непустого каталога.

Напомню, что команда `rm -rf` удаляет каталог со всем его содержимым.

ЗАДАНИЯ

- С помощью одной команды создайте цепочку каталогов `dir1/dir2/dir3/dir4` в домашнем каталоге. Проверьте, созданы ли они.
- Создайте в каталоге `dir1/dir2` файл `Bubuka`.
- Попробуйте удалить цепочку `dir1/dir2/dir3/dir4`. Проверьте, какие каталоги удалены.
- Удалите каталог `dir1` со всем содержимым.

Копирование, перемещение и переименование файлов

Команда `cp` применяется для копирования (пример 4.14):

- ❑ команда `cp srcFile tagFile` копирует `srcFile` в `tagFile`;
- ❑ команда `cp file1 file2 fileN dir` копирует указанные файлы `file1`, `file2`, `fileN` в каталог `dir`;
- ❑ команда `cp -R dir1 dir2` копирует каталог `dir1` в каталог `dir2` рекурсивно, создавая в каталоге `dir2` копию каталога `dir1` со всеми файлами, содержащимися в исходном каталоге `dir1`.

Пример 4.14. Копирование файлов

```
$ ls -F mydir/  
f1  
$ cp mydir/f{1,2}  
$ ls -F mydir/  
f1 f2
```

С помощью команды `cp mydir/f{1,2}` создается копия файла `f1` с именем `f2` в том же каталоге `mydir`, где находится исходный файл. Команда `cp mydir/f{1,2}`

использует механизм перебора и эквивалентна команде `cp mydir/f1 mydir/f2`, однако она значительно короче.

Скопируем теперь каталог `mydir` со всем содержимым в каталог `/tmp` (пример 4.15).

Пример 4.15. Рекурсивное копирование каталога

```
$ ls
mydir
$ cp -R mydir/ /tmp/
$ ls -R /tmp/mydir
/tmp/mydir:
f1 f2
```

Команда `mv` используется для перемещения и переименования:

- ☐ команда `mv oldName newName` переименовывает `oldName` в `newName`;
- ☐ команда `mv file1 file2 fileN dir` перемещает заданные файлы в каталог `dir`;
- ☐ команда `mv oldName newName` переименовывает каталог `oldName` в `newName`.

Переместим каталог `/tmp/mydir` в домашний каталог (пример 4.16).

Пример 4.16. Перемещение каталога с содержимым

```
$ mv /tmp/mydir/ /var/tmp
$ ls -R /var/tmp/mydir
f1 f2
```

Команда `mv` работает с каталогами точно так же, как и с файлами, т. е. для перемещения каталога не надо указывать дополнительных опций.

ЗАДАНИЯ

- Создайте каталог `Toppler`, содержащий два файла: `high11` и `low11`. Скопируйте оба эти файла в каталог `/tmp`.
- Переместите эти два файла из каталога `/tmp` в домашний каталог, используя символы подстановки.
- Рекурсивно скопируйте каталог `Toppler` в `/tmp`.
- Переименуйте в `/tmp` этот каталог в `Roller`.
- Переместите полученный каталог со всем содержимым в домашний каталог так, чтобы на экране отображалась подробная информация об этом процессе.

Поиск файлов

Команда `find` ищет файлы по указанным критериям. Формат команды:

`find` *опции места_поиска критерии модификаторы*

Здесь *места_поиска* — каталоги, начиная с которых будет осуществлен поиск. Поиск также производится и в подкаталогах, указанных в качестве мест поиска каталогов. По умолчанию имена найденных файлов печатаются на экране. Критериями поиска могут быть любые атрибуты файла, например, имя файла, его размер, владелец файла, тип, даты доступа, модификации и пр. Однако команда `find` не обеспечивает поиска файла по содержимому.

В командной строке `find` после тире указывают не только опции, но и критерии поиска, а также так называемые *модификаторы*. Критерии поиска указывают, что искать, а модификаторы и опции — как искать и выводить найденную информацию.

Наиболее часто используют следующие критерии:

- ☐ `-name` — поиск по имени файла или файловому шаблону;
- ☐ `-iname` — то же с игнорированием регистра;
- ☐ `-type` — поиск по типу файла;
- ☐ `-size` — для поиска по размеру или диапазону возможных размеров;
- ☐ `-empty` — поиск пустых файлов;
- ☐ `-mtime` — по дате модификации;
- ☐ `-perm` — поиск по правам доступа;
- ☐ `-user` и `-group` — по принадлежности файла.

Например, в текущем каталоге требуется найти все файлы, имя которых начинается со строки `d1`. Соответствующая команда поиска приведена в примере 4.17.

Пример 4.17. Поиск по имени файла

```
$ find . -name "d1*"
./d1
```

Обратите внимание, что если критерий поиска по подстроке в имени файла использует шаблоны подстановки, то такой шаблон должен быть экранирован кавычками. В противном случае до исполнения команды оболочка заменит шаблоны подходящими именами файлов, и команда будет работать неверно.

Можно ужесточить критерий поиска, потребовав от предыдущей команды отыскивать только каталоги с заданным именем (пример 4.18).

Пример 4.18. Поиск только каталогов по заданному имени

```
$ find . -name "d1*" -type d -ls
12042      0 drwxr-xr-x    2 user1 user1 192 Окт  7 21:58 ./d1
```

Более того, в последнем примере использован модификатор `-ls` команды `find`, позволяющий отображать информацию о найденных файлах в подробном формате.

Если критерии поиска необходимо объединить по логическому условию ИЛИ, то следует использовать `-o` (пример 4.19).

Пример 4.19. Поиск по критериям, объединенным условием ИЛИ

```
$ find . -name "d1*" -o -empty
./d1
./d1/s
./d1/f1
./d1/f2
```

В этом примере продемонстрирован поиск по двум критериям, объединенным условием ИЛИ. Первый критерий — поиск файлов, начинающихся со строки `d1`, а второй — поиск пустых файлов (условие поиска `-empty`).

Для поиска файлов определенного типа необходимо задавать критерий `-type` *тип*, где *тип* один из:

- ☐ `b` — файл блочного устройства;
- ☐ `c` — файл символьного устройства;
- ☐ `d` — каталог;
- ☐ `f` — обычный файл;
- ☐ `p` — именованный канал;
- ☐ `s` — сокет;
- ☐ `l` — символическая ссылка.

Имеется возможность исполнять команды с найденными `find` файлами. Для этого необходимо использовать модификатор `-exec` (пример 4.20).

Пример 4.20. Выполнение команд над найденными файлами

```
$find . -name "*core*" -exec rm -f {} \;
```

В примере 4.20 производится поиск и удаление всех файлов, имена которых содержат в себе строку "core". Смысл конструкции `{ } \;` состоит в следующем: фигурные скобки будут заменены именами найденных файлов, которые и станут аргументами исполняемых команд. С помощью экранированной точки с запятой `;` команду `find` информируют о том, где заканчивается команда, указанная после `-exec`.

ЗАДАНИЯ

- Найдите в домашнем каталоге все пустые файлы.
- Найдите в каталоге `/bin` все файлы, больше 1 Кбайт, но меньшие 10 Кбайт.
- Сделайте то же, но сузив диапазон размеров файлов с 1,5 до 2 Кбайт.

Быстрый поиск файлов *locate*

В GNU/Linux используется специальная индексированная база данных, в которую помещаются все имена всех файлов в системе. Эта база данных позволяет быстро производить поиск файла по подстроке в его имени. Поиск в этой базе данных выполняется с помощью команды `locate`. Эта команда не воспринимает файловые шаблоны поиска, а только строки. Имеются версии этой команды, работающие с регулярными выражениями — эффективными шаблонами для поиска строк.

Индексирование базы данных производится на регулярной основе автоматически (обычно в ночное время) командой `updatedb`.

Отыщем, например, все имена файлов, содержащих строку `spice` (пример 4.21).

Пример 4.21. Поиск файлов по заданной строке с помощью *locate*

```
$ locate spice
/usr/share/jed/lib/spicemod.sl
/usr/share/jed/lib/spicemod.slc
/usr/share/vim/syntax/spice.vim
```

Команда `locate` вывела найденные имена файлов.

ЗАДАНИЯ

- Произведите поиск всех файлов, содержащих в имени строку `user`.
- Создайте новый файл `filemy`. Попробуйте найти его с помощью `locate`.
- Войдите в сеанс суперпользователя, проиндексируйте базу данных поиска и вновь повторите поиск от имени простого пользователя.

Определение содержимого файла

Часто бывает необходимо определить тип содержимого файла, что позволяет делать команда `file` (пример 4.22), пытающаяся осуществить это с помощью базы данных сигнатур файлов, называемой `magic numbers` (магические числа).

Пример 4.22. Определение содержимого файла

```
$ file /etc/issue
/etc/issue: ASCII text

$ file /bin/ls
/bin/ls: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), for GNU/Linux 2.6.9, stripped
```

В примере 4.22 были определены типы содержимого двух файлов: `/etc/issue` и `/bin/ls`. Первый из них содержит текст, а второй — машинный код в формате ELF (Executable and Linking Format, формат исполняемых и компонуемых файлов).

Перед выводом на экран незнакомого файла рекомендуется узнать его тип с помощью команды `file`. В противном случае на экран может быть выведен бинарный файл, и настройки терминала могут быть испорчены.

ЗАДАНИЯ

- Определите типы содержимого текущего каталога, файлов `/bin/bash` и `~/.bashrc`.
- Узнайте, где находится файл с магическими числами.
- Как команда `file` определяет тип файла?
- Можно ли создать собственный тип содержимого файла?

Устройство файловой системы

Файловая система построена из трех основных компонентов (табл. 4.1):

- ☐ суперблок (superblock);
- ☐ массив индексных дескрипторов (inode list);
- ☐ блоки хранения данных.

Таблица 4.1. Устройство файловой системы

Магнитный диск						
Суперблок	Массив индексных дескрипторов	Данные				
		Блок1	Блок2	Блок3 ...		БлокN

Суперблок содержит основную информацию, необходимую для монтирования и работы файловой системы:

- ☐ тип файловой системы;
- ☐ размер и количество блоков в файловой системе;
- ☐ количество индексных дескрипторов;
- ☐ время последнего монтирования;
- ☐ информация о том, была ли отмонтирована файловая система;
- ☐ счетчик числа монтирований;
- ☐ список свободных блоков и индексных дескрипторов

и т. п.

Если суперблок файловой системы испорчен, то монтирование файловой системы без его восстановления невозможно. Современные файловые системы специально сохраняют в заранее известных блоках диска копии суперблока для возможности их использования при восстановлении файловой системы.

Индексные дескрипторы или, иначе, *метаданные* — это данные о самих файлах. В индексных дескрипторах сохраняются следующие сведения о файлах:

- ☐ владелец и группа пользователей файла;
- ☐ права доступа к файлу;
- ☐ тип файла;
- ☐ количество имен у файла (link count);
- ☐ дата доступа к файлу (файл открыт на чтение);

- ☐ дата модификации (файл открыт на запись);
- ☐ дата изменения метаданных;
- ☐ количество блоков, занятое файлом;
- ☐ указатели на блоки данных файла.

Каталоги предоставляют собой особый тип файлов, содержащих таблицу, в которой хранятся имена файлов, находящихся в данном каталоге, и соответствующие им номера индексных дескрипторов (inode). Эти записи связывают имена файлов с их индексными дескрипторами, а те, в свою очередь, предоставляют информацию о местонахождении блоков данных файла. В блоках данных нет никакой информации, способной указать номер индексного дескриптора файла, которому принадлежит этот блок данных. Точно так же, в индексном дескрипторе не хранится имя файла, который данный индексный дескриптор описывает. Именно каталоги определяют положение файла в дереве файловой системы, т. к. сам файл не содержит информации о месте его нахождения.

На один и тот же индексный дескриптор может указывать несколько имен файлов. Это фиксируется *счетчиком имен файла* (link counter) в индексном дескрипторе. Если у файла имеется несколько имен, то говорят, что между этими именами существует *жесткая связь* (hard link). То есть разные имена файлов указывают на одни метаданные и, следовательно, на одни и те же блоки данных. Все имена файла абсолютно эквивалентны, и изменение содержимого этих файлов будет совершенно синхронным. Нет никакой возможности определить, какое имя у файла было изначально, а какое появилось потом.

У каталогов имеются как минимум два имени:

- ☐ обычное имя каталога, находящееся в родительском каталоге (например, /home/user1);
- ☐ имя "точка" (.) — имя текущего каталога.

При появлении в любом каталоге подкаталога количество имен у этого каталога увеличивается на единицу, т. к. в дочернем каталоге всегда содержится имя "две точки" (..) — имя родительского каталога.

Количество имен у файла можно определить с помощью команды `ls -l`, а получить номера индексных дескрипторов можно, используя `ls -i` (пример 4.23).

Пример 4.23. Определение номера индексного дескриптора

```
$ ls -ldi /etc
6 drwxr-xr-x  87 root      root      6064 Oct 14 00:13 /etc
```

Из полученного листинга видно, что номер inode каталога /etc равен 6, а количество имен у него 87, следовательно, количество подкаталогов в нем равняется 85.

Подробную информацию об индексном дескрипторе файла можно получить, используя команду `stat` (пример 4.24).

Пример 4.24. Команда `stat`

```
$ stat /etc
  File: `/etc'
  Size: 6064          Blocks: 12          IO Block: 4096   Directory
Device: 305h/773d    Inode: 6              Links: 87
Access: (0755/drwxr-xr-x)  Uid: (  0/   root)   Gid: (  0/   root)
Access: 2009-06-17 23:19:59.000000000 +0600
Modify: 2009-12-14 00:13:23.000000000 +0500
Change: 2009-12-14 00:13:23.000000000 +0500
```

Зарезервированы три номера для индексных дескрипторов:

- ☐ 0 — свободный индексный дескриптор (для удаленных файлов в индексный дескриптор записывается номер 0);
- ☐ 1 — зарезервировано для будущего использования, но в GNU/Linux используются для файловых систем, порожденных ядром (проверьте каталоги /proc и /sys);
- ☐ 2 — inode корневой файловой системы и точек монтирования файловых систем.

Задания

- Определите номер inode родительского каталога корневого каталога.
- Сравните номер inode вашего домашнего каталога в /home и имени "точка" (.) в нем самом.
- Определите количество имен у файла /bin/gzip.
- Получите подробную информацию об inode этого файла.

Использование жестких связей

Жесткая связь имеет место между несколькими именами файла, указывающими на одни и те же метаданные (inode). Наличие у файла другого имени, т. е. наличие жесткой связи, можно обнаружить, изучив вывод команды `ls -l`

(пример 4.25). Эта команда в третьем столбце показывает количество имен у файла (link count). Если у файла link count имеет значение, большее единицы, значит, файл жестко связан.

Пример 4.25. Жесткая связь

```
$ ls -li /bin/*grep
13598 -rwxr-xr-x 3 root root 85228 Sep 11 2009 /bin/grep
13598 -rwxr-xr-x 3 root root 85228 Sep 11 2009 /bin/egrep
13598 -rwxr-xr-x 3 root root 85228 Sep 11 2009 /bin/fgrep
```

Листинг, выводимый командой `ls -li /bin/*grep`, показывает, что имена `/bin/grep`, `/bin/egrep` и `/bin/fgrep` принадлежат одному и тому же файлу с тремя именами. Номер inode у всех этих файлов одинаков, следовательно, они эквивалентны.

Поскольку метаданные у файлов, между которыми установлена жесткая связь, одинаковы, то и блоки данных тоже одинаковы. Следовательно, любые изменения, производимые с одним файлом, зеркально отразятся на файле, жестко связанном с ним.

Для создания жесткой связи с файлом применяется команда `ln`. Первый аргумент команды — имя файла, а второй — имя жесткой связи с ним (пример 4.26).

Пример 4.26. Создание жесткой связи

```
$ ln file1 newname
$ ls -li file1
4676 -rw-rw-r-- 2 user1 users 0 Dec 14 20:43 file1
$ ls -li newname
4676 -rw-rw-r-- 2 user1 users 0 Dec 14 20:43 newname
```

В примере 4.26 создано новое имя для файла `file1` — `newname`. Команда `ls -li` демонстрирует, что inode у этих файлов одинаковые.

Следует помнить, что жесткие связи могут быть установлены только в пределах одной файловой системы, т. е. нельзя установить жесткую связь между файлом на жестком диске и, например, дискете. Это вызвано тем, что у жестко связанных файлов должен быть один и тот же индексный дескриптор.

Пример 4.27. Эквивалентность имен жестко связанных файлов

```
$ echo 'This is a hard link' > file1
$ cat newname
This is a hard link
```

В примере 4.27 информация в файл была записана с использованием имени файла `file1`, а чтение информации было произведено из `newname`.

Удаление одной жесткой связи с файлом уменьшает количество имен (`link count`) на единицу (пример 4.28), однако реально файл удаляется только тогда, когда счетчик количества имен уменьшается до нуля.

Пример 4.28. Количество имен файла

```
$ ls -l file1 newname
-rw-rw-r--  2 user1  user1      20 Dec 14 20:43 file1
-rw-rw-r--  2 user1  user1      20 Dec 14 20:43 newname
$ rm -f file1
$ ls -l file1 newname
ls: file1: No such file or directory
-rw-rw-r--  1 user1  user1      20 Dec 14 20:43 newname
```

Пример 4.28 демонстрирует, что при удалении одного из имен файла счетчик имен уменьшился на единицу.

Установить жесткую связь с каталогом с помощью команды `ln` нельзя, т. к. для каталогов жесткие связи имеют особый смысл (пример 4.29).

Пример 4.29. Попытка создания жесткой связи для каталога

```
$ ls -ld LPI
drwxrwx---  10 prof  users      544 May  2 10:15 LPI
$ ln LPI l1
ln: `LPI': не допускается создавать жесткие ссылки на каталоги
```

У каждого каталога имеются как минимум два имени — имя каталога, записанное в его родительском каталоге, и имя "точка". Таким образом, у вновь созданного каталога существуют сразу два имени. А у его родительского каталога после создания нового каталога количество имен увеличивается на единицу, т. к. в дочернем каталоге имеется имя "две точки", являющееся жесткой связью с именем родительского каталога (пример 4.30).

Пример 4.30. Жесткая связь каталога с подкаталогами

```
$ ls -ldi
 3555 drwx----- 16 user1  users      1120 Dec 15 22:15 .
$ mkdir dir1
$ ls -ild dir1
 1598 drwxr-x---   2 user1  users        48 Dec 15 23:03 dir1
$ ls -ial dir1
total 2
 1598 drwxr-x---   2 user1  users        48 Dec 15 23:03 .
 3555 drwx----- 17 user1  users      1144 Dec 15 23:03 ..
$ ls -ldi
 3555 drwx----- 17 user1  users      1144 Dec 15 23:03 .
```

В примере 4.30 был создан каталог `dir1`, inode которого — 1598. Имя "точка", находящееся в каталоге `dir1`, — имя текущего каталога, имеет тот же индексный дескриптор 1598. Таким образом, у нового каталога `dir1` существуют два имени. Имя родительского каталога — "две точки", находящиеся в каталоге `dir1`, имеет inode 3555, значение которого совпадает с inode родительского каталога. При сравнении выводов первой и последней команд становится заметно, что после создания каталога `dir1` количество имен у его родительского каталога увеличилось на единицу.

Команда `cp` имеет специальную опцию `-l`, которая позволяет (в пределах одной файловой системы) вместо копирования создавать жесткие связи с исходными файлами (пример 4.31).

Пример 4.31. Команда `cp -l`

```
$ ls -li f???
 4629 -rw-r--r--   1 user1  users        0 Dec 11 12:30 f112
 4631 -rw-r--r--   1 user1  users        0 Dec 11 12:30 f113
 4632 -rw-r--r--   1 user1  users        0 Dec 11 12:30 f117
$ cp -l f??? dir1
$ ls -li dir1
total 0
 4629 -rw-r--r--   2 user1  users        0 Dec 11 12:30 f112
 4631 -rw-r--r--   2 user1  users        0 Dec 11 12:30 f113
 4632 -rw-r--r--   2 user1  users        0 Dec 11 12:30 f117
```


В примере 4.31 показано, что при использовании команды `cp` с опцией `-l` для исходных файлов были созданы жесткие связи с такими же именами в каталоге `dir1`.

Задания

- Создайте в текущем каталоге файл `file1` и жесткую связь с файлом `link1`.
- Можете ли вы перенести этот файл в каталог `/tmp` и почему?
- Попробуйте создать жесткую связь командой `ln` с домашним каталогом. Удастся ли это?
- Как можно создать жесткую связь с именем домашнего каталога? Какое может быть имя у этой связи и с помощью какой команды она создается?

Использование символических ссылок

Символические ссылки — это особый вид файлов, представляющий собой указатели на другие файлы. Символические ссылки создают командой `ln -s`, где первый аргумент — имя уже существующего файла, а второй аргумент — либо имя символической ссылки, либо каталога, где будет образован файл символической ссылки с таким же именем, что и исходный файл (пример 4.32).

Пример 4.32. Создание символической ссылки на файл

```
$ ln -s file1 slink1
$ ls -li file1 slink1
 4639 -rw-rw-r--  1 user1  users      0 Dec 11 15:55 file1
 1604 lrwxrwxrwx  1 user1  users      5 Dec 16 01:37 slink1 ->
file1
$ echo 123 > file1
$ cat slink1
123
```

В примере 4.32 на файл `file1` создана символическая ссылка `slink1`. Метаданные у `file1` и `slink1` различаются, поскольку это совершенно разные файлы. Тем не менее, к файлу можно обращаться с помощью ссылки.

Можно создавать ссылки на каталоги (пример 4.33).

Пример 4.33. Создание символической ссылки на каталог

```
$ ln -s ~/dir1 /tmp
$ ls -ldi dir1 /tmp/dir1
 4635 drwxr-xr-x  2  user1  users  144 Dec 16 00:26 dir1
163286 lrwxrwxrwx  1  user1  users   16 Dec 16 01:12 /tmp/dir1 ->
/home/user1/dir1
```

В этом примере создается символическая ссылка на каталог `dir1` в каталоге `/tmp`. Заметно, что символическая ссылка имеет совершенно иной тип файла и номер `inode`, она не ограничена одной файловой системой. Обратите также внимание, что, во-первых, символические ссылки можно устанавливать на каталоги, а, во-вторых, при создании символической ссылки в другом каталоге следует указывать полное имя исходного файла. Если второе требование не выполняется, то:

- ☐ либо ссылка будет оборванной (или иначе — висящей), т. е. символическая ссылка будет указывать на несуществующий файл;
- ☐ либо ссылка будет указывать на существующий файл в целевом каталоге, имя которого (случайно или намеренно) совпадет с исходным файлом, однако ссылка будет указывать не на исходный файл.

Символическая ссылка может оказаться оборванной (пример 4.34) в случае, если:

- ☐ файл, на который она указывает, перемещен, переименован или удален;
- ☐ нет достаточных прав доступа на файл, указываемый символической ссылкой;
- ☐ файл находится в файловой системе, которая сейчас не смонтирована.

Пример 4.34. Оборванная символическая ссылка

```
$ mv file1 file11
$ cat slink1
cat: slink1: No such file or directory
```

После переименования файла `file1` в `file11` ссылка `slink1` стала оборванной.

Приведенный далее пример 4.35 демонстрирует, что символическая ссылка может быть создана и на специальный файл.

Пример 4.35. Создание ссылки на файл устройства

```
$ ln -s /dev/fd0 .  
$ ls -l fd0  
lrwxrwxrwx    1 user1    users          8 Dec 16 01:13 fd0 -> /dev/fd0
```

Длина файла символической ссылки равна длине имени файла, на который она указывает.

Команда `cp` обладает специальной опцией `-s`, которая позволяет вместо копирования файлов создавать на них символические ссылки (пример 4.36).

Пример 4.36. Опция `-s` команды `cp`

```
$ cp -s ~/f??? /tmp  
$ ls -l /tmp/f???  
lrwxrwxrwx    1 user1    users    16 Dec 16 01:52 /tmp/f112 -> /home/user1/f112  
lrwxrwxrwx    1 user1    users    16 Dec 16 01:52 /tmp/f113 -> /home/user1/f113  
lrwxrwxrwx    1 user1    users    16 Dec 16 01:52 /tmp/f117 -> /home/user1/f117  
lrwxrwxrwx    1 user1    users    16 Dec 16 01:52 /tmp/f122 -> /home/user1/f122
```

В примере 4.36 на группу файлов были созданы символические ссылки в каталоге `/tmp`.

Если используется команда `cp`, для которой в качестве аргумента указаны файлы символических ссылок, то будут скопированы не файлы символических ссылок, а файлы, на которые они указывают. Если же необходимо скопировать не исходные файлы, а именно символические ссылки, то следует использовать опцию `-d` команды `cp` (пример 4.37).

Пример 4.37. Копирование символических ссылок

```
$ cp -d /tmp/f122 Documents/  
$ ls -l Documents/f122  
lrwxrwxrwx    1 user1    users          16 Dec 16 02:00 Documents/f122 ->  
/home/user1/f122
```

Задания

- Создайте символическую ссылку на каталог `/usr/share/doc` в вашем домашнем каталоге. Попробуйте выполнить команду `cd ~/doc`.
- Получите список файлов — символических ссылок, находящихся в каталоге `/usr`.
- Создайте в домашнем каталоге символическую ссылку на исполняемый файл `/bin/ls`. Попробуйте воспользоваться новой ссылкой.

Глава 5



Процессы

Процесс — это одно из наиболее важных понятий в GNU/Linux. Работающие программы являются процессами, и умение управлять ими крайне важно. В этой главе вы познакомитесь с понятиями задачи, процесса и потока. Вы научитесь получать информацию о процессах в системе и управлять ими.

Процессы и задания

GNU/Linux — многопользовательская и многозадачная операционная система. Это значит, что одновременно в системе может выполняться множество программ, запущенных различными пользователями. Для обеспечения многозадачности процессор компьютера последовательно переключается на обработку программного кода различных приложений, системных и прочих программ, создавая впечатление, что все они выполняются одновременно. Рано или поздно, по истечении некоторого промежутка времени, каждая ждущая исполнения программа попадает на процессор для выполнения. Естественно, чем больше загружена система, тем большим становится этот промежуток времени. Сильно загруженная система может с точки зрения пользователя потерять интерактивность, т. е. она может настолько медленно отвечать на запросы пользователя, что он будет уверен в том, что система не работает.

Программы представляют собой исполняемые файлы двух типов:

- ☐ бинарные файлы, содержащие инструкции на машинном языке;
- ☐ интерпретируемые сценарии (например, сценарии Bash и программы Perl).

Когда от пользователя поступает команда выполнить некоторую программу, оболочка вначале определяет с помощью "магических чисел" (magic numbers, см. `man file`), к какому из этих двух типов относится данная программа.

Если она относится к первому типу, то программа пополняет очередь других работающих программ, ожидающих своей очереди на исполнение процессором. Во втором случае сначала запускается программа — интерпретатор, предназначенный для исполнения данного вида сценариев (например, `/bin/bash`), который, в свою очередь, относится к программам первого типа.

Важно понимать, что программа — это некоторый код, хранящийся в обычном исполняемом файле. Как только пользователь запускает программу на выполнение, она, в конечном итоге, в виде соответствующего машинного кода попадает в ОЗУ и обрабатывается процессором. Данное описание является несколько упрощенным, однако оно подводит к пониманию, что такое процесс. *Процесс* — это экземпляр программы, исполняемый процессором, либо ожидающий этого момента в очереди. Программа — это исполняемый файл, а процесс — это исполняющийся машинный код.

В GNU/Linux одновременно выполняется множество процессов, причем многие из них запускаются автоматически без вмешательства пользователей. В силу этого факта следует выделить понятие задания (*task* или *job*). *Задание* — это команда, запущенная на исполнение пользователем с помощью оболочки. Запуск задания может приводить к созданию более чем одного процесса. Например, запуск сервера HTTP Apache приводит к параллельному запуску сразу нескольких экземпляров программы `httpd`.

Каждый процесс работает в собственном виртуальном адресном пространстве независимо от других процессов. Пользовательский процесс не может получить доступ к адресному пространству другого процесса. Но ядро Linux — это тоже процесс. Ядро работает, переключая процессор в привилегированный режим. Достигается это с помощью специального прерывания процессора.

Так как программы пользователей не могут получать доступ к системным ресурсам напрямую, они вынуждены обращаться при возникновении такой необходимости к ядру. Ядро предоставляет пользовательским процессам услуги посредством интерфейса системных вызовов. *Системные вызовы* — это функции, реализованные в ядре операционной системы. Когда пользовательский процесс осуществляет системный вызов, его выполнение производится ядром. Код ядра при этом работает в контексте процесса.

Процессы ядра отделены от пользовательских процессов — они работают в пространстве ядра (*kernel space*). Приложения работают в пользовательском пространстве (*user space*) за исключением времени, когда в них производятся системные вызовы.

Команда `time` позволяет увидеть, сколько времени исполнялась программа реально (от запуска до окончания), сколько времени она обслуживалась ядром и сколько она работала в пользовательском пространстве (пример 5.1).

Пример 5.1. Команда time

```
# time updatedb

real    0m0.158s
user    0m0.092s
sys     0m0.064s
```

В примере 5.1 программа `time` определила реальное время работы программы `updatedb`, время работы на стороне пользователя и на стороне ядра.

В GNU/Linux одновременно выполняется множество процессов. Процессор выделяет каждому процессу интервалы времени (*time slices*), в течение которых инструкции процесса выполняются процессором. Такой режим называется *разделением времени*. Используемая в Linux модель управления процессами относится к классу *вытесняющей многозадачности* (*preemptive multitaskings*). В рамках этой модели каждый процесс имеет свой уровень важности в системе или приоритета. Более приоритетные процессы могут вытеснять с исполнения на процессоре менее приоритетные.

Если схематично описать внутреннюю структуру процесса, то в нем обычно можно выделить следующие составляющие части:

- ❑ заголовок процесса, содержащий, в частности, специальные сигнатуры, показывающие формат процесса (например, формат ELF или COFF);
- ❑ инструкции (по историческим причинам их именуют *text*);
- ❑ данные (*data*) — статические данные;
- ❑ куча (*heap*) — область памяти, предназначенная для выделения памяти динамическим переменным и т. п.;
- ❑ стек (*stack*) — структура типа LIFO (*Last Input — First Output*), предназначенная для вызовов подпрограмм и хранения некоторых переменных.

Процессы могут обращаться к каким-либо файлам. С каждым файлом ассоциируется целое число, называемое *дескриптором файла*. Когда ядро создает процесс, с ним автоматически ассоциируются три потока ввода/вывода:

- ❑ стандартный поток ввода (*stdin*) — дескриптор 0;
- ❑ стандартный поток вывода (*stdout*) — дескриптор 1;
- ❑ стандартный поток ошибок (*stderr*) — дескриптор 2.

Процессы создаются другими процессами с помощью системного вызова `fork()`. Между процессами устанавливаются отношения наследства. Про-

цесс, который породил другие процессы, называется *родительским* (parent). А порожденные процессы называются *дочерними* (child).

У каждого процесса есть один и только один родитель, но процесс может иметь несколько потомков. Это позволяет выстраивать иерархическую древовидную систему процессов.

Когда процесс-родитель завершает свою работу, он может завершить работу своих дочерних процессов. Если это не происходит, то "осиротевшие" процессы наследуются процессом с номером 1 — `init`.

Ядро Linux поддерживает специальную таблицу процессов, содержащую информацию о процессах в системе. Таблица процессов динамически увеличивается при росте количества процессов в системе (увеличение таблицы процессов ограничено).

Важнейшие идентификаторы процесса:

- ❑ PID (Process ID) — уникальный порядковый номер процесса в системе, предназначенный для идентификации процесса;
- ❑ PPID (Parent Process ID) — PID родительского процесса, позволяющий выстроить иерархию процессов;
- ❑ UID (User ID) — идентификатор пользователя, от имени которого выполняется процесс;
- ❑ GID (Group ID) — идентификатор группы пользователей, от имени которой выполняется процесс.

Процесс запускается с терминала, поэтому процесс связывается с терминалом. Но, все же, есть процессы, не связанные с терминалами.

Процессы принято подразделять на три категории, перечисленные далее.

- ❑ *Процессы ядра*. Они выполняются в `kernel space`. Примером такого процесса в Linux является процесс упорядочивания процессов в очереди (scheduling). Процессы ядра располагаются в оперативной памяти и не имеют соответствующих им программ в виде исполняемых файлов. Соответствующий им код находится в файле ядра и модулях ядра.
- ❑ *Демоны* — это фоновые неинтерактивные процессы, запускаемые путем загрузки соответствующих им исполняемых файлов. Обычно они выполняют свою работу, никак не проявляя себя с точки зрения пользователей, т. е. они не ассоциированы с терминалами. Они предназначены для выполнения таких задач, как обслуживание соединений по какому-либо сетевому протоколу или регулярное выполнение рутинных операций в системе.
- ❑ Все остальные процессы называются *прикладными*. Они, как правило, порождаются в рамках сеанса работы пользователя.

ЗАДАНИЯ

- Получите страницу помощи, касающуюся стандартных потоков ввода/вывода.
- Какая команда позволяет узнать имя файла устройства того терминала, на котором вы вошли в сеанс?

Фоновый режим выполнения заданий

Обычно пользователи вводят задания интерактивно с помощью команд оболочки, наблюдая на экране за ходом и результатами их выполнения.

В GNU/Linux можно запускать команды в *фоновом режиме*. Это позволяет пользователю выполнять несколько программ одновременно.

Только одно задание на терминале может работать в интерактивном режиме (foreground). Если на терминале имеются другие активные задания, то они выполняются в фоновом режиме (background).

Для запуска команды в фоновом режиме в конце командной строки необходимо поставить символ `&` (амперсанд). При запуске фонового задания выводится его номер (пример 5.2).

Пример 5.2. Запуск задания в фоновом режиме

```
#updatedb &  
[1]546
```

В примере 5.2 команда `updatedb` запущена в фоновом режиме, т. к. в конце командной строки установлен символ `&`. Номер задания выводится в квадратных скобках, в этом примере — 1. Число, выводимое после квадратных скобок, — PID процесса.

Для мониторинга состояний фоновых заданий предназначена команда `jobs`, которая позволяет просмотреть статус фоновых заданий. Она отображает номер задания, имя команды и статус задания (пример 5.3).

Пример 5.3. Команда `jobs`

```
#find ~basile -name "*core*" -user basile -exec rm -f {} \; &  
[1]548  
#find ~anna -name "*core*" -user anna -exec rm -f {} \; &  
[2]551  
#jobs  
[1]-Done find ~basile -name "*core*" -user basile -exec rm -f {} \;  
[2]+Running find ~anna -name "*core*" -user anna -exec rm -f {} \;
```


В этом примере были запущены два задания на поиск и удаление файлов с подстрокой `core` в именах файлов, принадлежащих пользователям `basile` и `anna`. Заданиям назначены номера 1 и 2. Команда `jobs` показала, что задание 1 выполнено, а задание 2 выполняется.

Обозначения `%%` и `%+` указывают последнее запущенное фоновое задание, а `%` — предпоследнее задание. Аналогично, информация о заданиях, выводимых командой `jobs`, отображает символы `+` и `-` для индикации последнего и предпоследнего заданий.

Команда `fg %номерзадания` переводит задание с номером `номерзадания` в интерактивный режим. Так, команда `fg %1` переводит задание с номером 1 в интерактивный режим.

Наоборот, для перевода задания в фоновый режим из интерактивного необходимо приостановить его выполнение нажатием комбинации клавиш `<Ctrl>+<Z>`, а затем выполнить команду `bg` с аргументом `%номерзадания`.

Можно указывать задания по первым символам их командной строки, предваряя их `%` (пример 5.4).

Пример 5.4. Обращение к фоновым заданиям по именам команд

```
#jobs
[2]+Running find ~anna -name "*core*" -user anna -exec rm -f {} \;
#fg %fi
```

В примере 5.4 выполняющееся в фоновом режиме задание было переведено в интерактивный режим командой `fg %fi`, которая использовала идентификацию задания не по его номеру, а по двум первым буквам в имени команды.

Завершить фоновое задание можно командой `kill %номерзадания` (пример 5.5).

Пример 5.5. Снятие фонового задания с выполнения

```
#jobs
[2]+Running find ~anna -name "*core*" -user anna -exec rm -f {} \;
#kill %2
#jobs
[2]+Terminated find ~anna -name "*core*" -user anna -exec rm -f {} \;
```

В примере 5.5 задание с номером 2 было завершено командой `kill`.

ЗАДАНИЯ

- Найдите пустые файлы в домашнем каталоге в фоновом режиме.
- Запустите в фоновом режиме два задания: `sleep 200` и `sleep 2000`, выведите информацию о состоянии заданий.
- Снимите с выполнения 2-е задание, выведите информацию о заданиях.

Жизненный цикл процесса

Как было отмечено ранее, процессы создаются другими процессами. Когда процессу требуется создать дочерний процесс, используется системный вызов `fork()`.

Предположим, что пользователь выполнил в оболочке команду `ps -f`, желая получить список процессов, запущенных на данном виртуальном терминале. В таком случае `ps -f` является дочерним процессом оболочки. Пример 5.6 демонстрирует, что команда `ps -f` является дочерним процессом оболочки: PPID команды `ps -f 1751` соответствует PID оболочки.

Пример 5.6. Идентификаторы дочернего процесса

```
$ ps -f
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
user1	1751	1745	0	21:01	pts/0	00:00:00	bash
user1	1870	1751	0	21:32	pts/0	00:00:00	ps -f

Команда `ps -f` будет подробнее рассмотрена далее, сейчас нам необходимо разобраться в том, каков был жизненный цикл процесса.

Когда пользователь ввел в командной строке команду `ps -f`, эта команда была проверена оболочкой `bash`, не является ли она встроенной. Так как эта команда не является встроенной, для ее выполнения оболочка произвела системный вызов `fork()`.

Системный вызов `fork()` приводит к тому, что ядро копирует адресное пространство процесса, произведшего этот вызов, в свободное адресное пространство в памяти. Начиная с этого момента, в системе имеются два совершенно одинаковых процесса `bash`. В обоих процессах выполнение производится с одной и той же инструкции, следующей после `fork()`.

При создании процесса ядро назначает ему уникальный идентификатор процесса PID. Так и сейчас, для копии родительской оболочки — дочернего процесса `bash`, в нашем случае, система назначила PID 1870, а PPID новой оболочки соответствует PID породившей ее оболочки — 1751.

Итак, в настоящий момент в системе имеются два одинаковых процесса `bash`. Но требовалось получить процесс команды `ps -f`. Поэтому в дочернем процессе `bash` должен быть произведен системный вызов класса `exec()` (см. `man 3 exec`), который позволяет загрузить программу `/bin/ls` и поместить прочитанный исполняемый код в адресное пространство дочернего процесса.

В то же время родительский процесс `bash` должен быть приостановлен с помощью системного вызова `wait()`, ожидающего сигнала о завершении работы дочернего процесса. До тех пор, пока этот сигнал получен не будет, родительский процесс `bash` не может продолжить работу.

Как же отличить родительский и дочерний процессы `bash` для того, чтобы определить, в каком из них надо вызвать `exec()`, а в каком — `wait()`? Их можно различить с помощью анализа значения, возвращаемого функцией `fork()`. Она возвращает PID дочернего процесса. Родительский процесс `bash` породил дочерний процесс, и, следовательно, `fork()` возвратил PID дочернего процесса. В порожденном процессе `fork()` возвращает 0.

После вызова `exec()` дочерний процесс уже не является процессом `bash` — он выполняет инструкции команды `ps`. Родительский процесс ждет завершения дочернего процесса.

После того как команда `ps -f` завершает свою работу системным вызовом `exit()`, ее адресное пространство освобождается. Родительский процесс `bash`, который с помощью системного вызова `wait()` ожидал завершения работы дочернего процесса, продолжит после этого свою работу.

Нормальный жизненный цикл процесса в системе может быть нарушен вследствие наступления какого-либо события. Например, процесс может получить сигнал от другого процесса, который может привести к его преждевременной остановке.

Информация о дочерних процессах в таблице процессов может быть важна для родительского процесса. Поэтому информация о них не стирается из таблицы процессов до вызова родительским процессом функции `wait()` (см. `man 2 wait`).

Возможна ситуация, когда дочерний процесс уже завершил свою работу, а родительский процесс не произвел еще системного вызова `wait()`. В таком случае информация об уже несуществующем дочернем процессе сохраняется в таблице процессов, т. к. эта информация может потребоваться родительскому процессу. Запись в таблице о завершившемся дочернем процессе помечается как `defunct` или, иначе, *процесс-зомби* (*zombie*).

Задания

- Как, по вашему мнению, изменяется общая картина жизненного цикла процесса в системе по сравнению с описанным, если процесс запускается в фоновом режиме?
- Выполните команду `exec ls -R /etc`. Изучите ее поведение.

Мониторинг процессов

Основным инструментом для исследования процессов является команда `ps`. Она выводит состояние процессов на момент ее выполнения в системе. Без опций она выводит список процессов, связанных с текущим терминалом (пример 5.7).

Пример 5.7. Команда `ps`

```
$ ps
  PID TTY          TIME CMD
 1751 pts/0    00:00:00 bash
 1890 pts/0    00:00:00 ps
```

Столбец `PID` (см. пример 5.7) отображает идентификаторы процессов, `TTY` — имена терминалов, `TIME` — суммарное процессорное время, затраченное процессом с момента его старта. Столбец `CMD` — командная строка процесса.

Подробную информацию можно получить с помощью опции `-f` (full format).

Пример 5.8. Подробный формат вывода команды `ps -f`

```
$ ps -f
  UID      PID  PPID  C  STIME TTY          TIME CMD
user1    1751   1745  0  21:01 pts/0    00:00:00 bash
user1    1970   1751  0  23:14 pts/0    00:00:00 ps -f
```

Как видно из примера 5.8, в вывод добавляются дополнительные столбцы с информацией о `UID` владельца процесса, родителе процесса (`PPID`), столбец `STIME` показывает время запуска процесса, столбец `C` — фактор загрузки процессора (требуется планировщику для расчета приоритета процесса).

Еще более подробную информацию предоставляет опция `-l` (long format).

Пример 5.9. Длинный формат вывода команды `ps -l`

```
$ ps -l
F S  UID    PID  PPID  C PRI  NI ADDR      SZ  WCHAN TTY          TIME CMD
0 S   500   1751  1745  0  76   0  -   781 11c541 pts/0    00:00:00 bash
0 R   500   1988  1751  0  77   0  -   617      - pts/0    00:00:00 ps
```

В примере 5.9 столбец `F` показывает флаги процесса (см. `man ps`).

Столбец `s` — статус процесса, который может быть:

- ☐ `D` — процесс приостановлен и не может быть прерван (например, ожидает окончания ввода/вывода);
- ☐ `R` — процесс выполняется или находится в очереди;
- ☐ `S` — процесс приостановлен;
- ☐ `T` — процесс трассируется;
- ☐ `Z` — процесс помечен как `defunct` (zombie).

Столбец `NI` показывает величину `nice number`. Эта величина устанавливается пользователем и участвует в вычислении приоритета процесса планировщиком. Столбец `SZ` — количество памяти, занимаемое процессом, а `WCHAN` — это информация о системном вызове, произведенном процессом.

Исключительно полезна опция `-e`, позволяющая вывести список всех процессов в системе (аналог — опция `-A`). Чаще всего для получения списка всех процессов используют команду `ps -ef`, дающую подробную информацию о процессах.

Бывает необходимо вывести список каких-либо конкретных процессов, выбранных по заданному критерию. Далее приведены некоторые опции фильтрации `ps`:

- ☐ `-u` — фильтрация по `UID`;
- ☐ `-t` — фильтрация по терминалу;
- ☐ `-p` — фильтрация по `PID` искомого процесса;
- ☐ `-c` — фильтрация по командной строке.

Требуется, например, вывести список процессов, запущенных на втором виртуальном терминале (пример 5.10).

Пример 5.10. Фильтрация процессов по заданному терминалу

```
$ ps -ft tty2
UID      PID  PPID  C STIME TTY          TIME CMD
root      1557    1  0 06:18 tty2      00:00:00 /sbin/mingetty tty2
```

Вывести список процессов с заданной командной строкой можно, указав после опции `-c` команды `ps` имя требуемой команды, например, `ps -c bash`. Имеется также команда `pgrep`, позволяющая получить PID процессов по заданной командной строке, например, `pgrep bash`.

Все приведенные выше команды соответствуют POSIX-формату, однако GNU-версия программы `ps` поддерживает также опции и в BSD-стиле. Наиболее популярной командой в таком формате является `ps aux` — она выводит список всех процессов в системе с указанием их владельцев. В примере 5.11 приведен небольшой фрагмент из ее вывода.

Пример 5.11. Опции команды `ps` в BSD-стиле

```
$ ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.1	1268	60	?	S	Oct08	0:04	init

Эта команда отобразила в виде процентов уровень загрузки процессора и памяти данным процессом. Столбец `vsz` — объем используемой процессом виртуальной памяти, а `rss` — физической памяти.

Опции POSIX, BSD и длинные опции GNU команды `ps` можно комбинировать.

Пример 5.12. Комбинация опций команды `ps` в POSIX- и BSD-стилях

```
$ ps -f U root
```

UID	PID	PPID	C	STIME	TTY	STAT	TIME	CMD
root	1	0	0	06:18	?	S	0:04	init
root	2	1	0	06:18	?	SW	0:00	[keventd]

Пример 5.12 содержит фрагмент вывода команды `ps -f U root`. В этом примере для выбора процессов пользователя `root` была использована опция в BSD-стиле `u`, а POSIX-опция `-f` была задана для указания подробного формата вывода.

В процессах могут быть созданы последовательности параллельно выполняющихся инструкций, называемых *потоками* (threads) или *облегченными процессами* (LWP, Light Weight Process). Для получения информации о потоках необходимо использовать опцию `-L` (пример 5.13).

Пример 5.13. Получение информации о потоках

```
$ ps -fLC swriter.bin
```

UID	PID	PPID	LWP	C	NLWP	STIME	TTY	TIME	CMD
user1	2891	2881	2891	0	5	Oct13 ?	00:00:50	swriter.bin	-writer
user1	2891	2881	2892	0	5	Oct13 ?	00:00:00	swriter.bin	-writer
user1	2891	2881	2893	0	5	Oct13 ?	00:00:00	swriter.bin	-writer
user1	2891	2881	2894	0	5	Oct13 ?	00:00:00	swriter.bin	-writer
user1	2891	2881	2895	0	5	Oct13 ?	00:00:00	swriter.bin	-writer

В примере 5.13 демонстрируется, что у процесса `swriter.bin` (Open Office) PID равен 2891. В этом процессе имеется пять потоков со своими идентификаторами, выводимыми в столбце `LWP`.

Для постоянного мониторинга процессов используется утилита `top`, которая отображает исполняющиеся процессы, отнимающие большую часть процессорного времени и наиболее сильно использующие память.

Утилита `top` регулярно обновляет информацию о процессах. Для выхода из нее необходимо набрать `q`. Эта команда позволяет, не выходя из интерактивного просмотра процессов, посылать процессам сигналы с помощью нажатия `k`. Нажатие `i` отключает вывод `top` неактивных процессов. В первой строке экрана вывода команды приводятся данные о средней загрузке системы (load averages) за последние 1, 5 и 15 минут.

Удобно использовать также команду `w`, демонстрирующую список всех вошедших в сеанс пользователей и запущенные ими задания (пример 5.14).

Пример 5.14. Команда w

```
$ w
```

```
18:17:10 up 22 min, 2 users, load average: 0.30, 0.38, 0.37
USER      TTY      FROM          LOGIN@      IDLE        JCPU      PCPU      WHAT
user1     pts/0    :0.0          5:56pm     1.00s      0.05s     0.00s    w
```

Исключительно важным инструментом, позволяющим получить информацию о процессах в системе, является каталог `/proc` (пример 5.15). То, что в нем находится, не является в полном смысле файлами — это псевдофайловая система, порожаемая ядром. Она позволяет выводить информацию о процессах, получать и устанавливать параметры ядра на лету.

Пример 5.15. Файловая система /proc

```
$ ls /proc
1      1508 1590 3710 846      driver      kcore      mtrr       sys
1016   1518 1591 3718 966      execdomains kmsg       net        sysvipc
11     1537 1620 4     967      fb          ksyms      partitions tty
1260   1556 1687 5     apm       filesystems loadavg    pci        uptime
1277   1557 1745 5090 asound    fs          locks      scsi       version
1290   1558 1751 5233 bus       ide         mdstat     self
1296   1559 2     6     cmdline  interrupts  meminfo    slabinfo
1314   1560 3     7     cpuinfo   iomem       misc       splash
1499   1561 3429 7156 devices ioports     modules    stat
1507   1563 3709 8     dma       irq         mounts     swaps
```

Подкаталоги /proc с именами, состоящими из цифр, соответствуют PID процессов.

Определим, например, PID текущей оболочки и исследуем соответствующий ее процессу каталог (пример 5.16).

Пример 5.16. Файлы в /proc

```
$ pgrep bash
1751
$ ls /proc/1751
cmdline cwd environ exe fd maps mem mounts root stat statm status
$ cat /proc/1751/cmdline
bash
```

Файл `cmdline` содержит в себе командную строку процесса, а файл `status` — подробную информацию о статусе процесса. Подкаталог `fd` предназначен для мониторинга файлов, открытых процессом. В нем содержатся символические ссылки на реально открытые файлы процессом. Имена этих символических ссылок соответствуют номерам файловых дескрипторов открытых файлов.

Исключительно важной является команда `fuser`, позволяющая определить, какие процессы открыли заданный файл. В примере 5.17, приведенном далее, пользователь смонтировал DVD в каталоге `/mnt/dvd`, перешел в этот каталог и получил информацию о процессе, использующем этот каталог.

Пример 5.17. Команда `fuser`

```
$ mount /mnt/dvd
$ cd /mnt/dvd/
$ /sbin/fuser .
.:                               1751c
$ ps -p 1751
  PID TTY          TIME CMD
 1751 pts/0    00:00:00 bash
```

Команда `fuser` вывела PID процесса оболочки, т. к. для нее этот каталог является текущим. Этот факт команда `fuser` подтверждает, выводя `c` после PID процесса.

Команда `fuser` выводит следующие символы после PID процессов:

- `c` — текущий каталог;
- `e` — исполняемый файл в момент его работы;
- `f` — открытый файл;
- `r` — корневой каталог;
- `m` — разделяемая библиотека либо отображаемый в память файл.

Опция `-m` команды `fuser` позволяет указать, что имя файла является именем смонтированного блочного устройства (пример 5.18).

Пример 5.18. Опция `-m` команды `fuser`

```
$ /sbin/fuser -m /dev/dvd
/dev/dvd:                1751c
```

Для просмотра списка процессов в виде дерева, отображающего отношения родительских и дочерних процессов, необходимо выполнить команду `ps tree`.

Пример 5.19. Команда `ps tree`

```
$ps tree
init--2*[automount]
    | -cron
    | -6*[getty]
    | -inetd
```

В примере 5.19 приведен фрагмент иерархического списка процессов.

ЗАДАНИЯ

- Получите информацию о процессах в обычном и подробном форматах.
- Выведите список процессов, запущенных пользователями, вошедшими в сеанс.
- Получите список всех процессов, связанных с терминалами.
- Выведите PID процессов, связанных с каталогом `/usr`.
- Получите иерархический список процессов с помощью команды `ps` (не `ps tree`).

Сигналы

Сигналы — это один из способов межпроцессного взаимодействия. Они обеспечивают возможность обмена процессами элементарными сообщениями. Получив сигнал, процесс может отреагировать на него по-разному в зависимости от полученного сигнала и действий, запрограммированных в коде процесса. Так, например, процесс может перечитать файл конфигурации или завершить работу.

Список сигналов, используемых в системе, можно получить с помощью команды `kill -l`, а их подробное описание доступно с помощью `man 7 signal`.

Наиболее часто используются сигналы, приведенные в списке далее:

- ❑ 1 — `HUP`, разрыв связи с терминалом (`Hang Up` — положить трубку). Многие демоны используют этот сигнал, как команду перечитать их конфигурационный файл и продолжить работу с измененными настройками. Оболочка `Bash` реагирует на этот сигнал завершением сеанса;
- ❑ 2 — `INT`, прерывание процесса. Генерируется при нажатии комбинации клавиш `<Ctrl>+<C>`;
- ❑ 3 — `QUIT`, сброс процесса. Генерируется при нажатии комбинации клавиш `<Ctrl>+<\\>`;
- ❑ 15 — `TERM`, сигнал для корректного завершения процесса. Этот сигнал команда `kill` посылает по умолчанию;
- ❑ 9 — `KILL`, аварийное завершение процесса.

Каким образом то или иное приложение реагирует на получение некоторого сигнала, зависит от того, как эта программа написана. В программе получение сигнала может перехватываться и обрабатываться специальным образом.

Сигнал `KILL` не может быть перехвачен. Этот сигнал приводит к немедленному и, таким образом, часто некорректному снятию процесса с исполнения. При этом файлы, открытые процессом, не закрываются нормальным способом, что может привести к потере данных.

Если необходимо послать сигнал некоторым процессам, то сначала требуется узнать PID этих процессов, а затем с помощью команды `kill` послать им требуемый сигнал, номер которого указывается после тире. Если номер сигнала или его имя не задано после дефиса, то команда `kill` посылает целевым процессам сигнал 15 (`TERM`). Так, например, можно попытаться послать сигнал оболочке `bash` (пример 5.20).

Пример 5.20. Реакция процессов на сигналы

```
$ ps
  PID TTY          TIME CMD
 2179 pts/0    00:00:00 bash
 2180 pts/0    00:00:00 ps
$ kill 2179
$ kill -2 2179

$ kill -1 2179
```

login:

Пример 5.20 демонстрирует, что оболочка `bash` игнорирует сигнал 15 (`TERM`), сигнал 2 (`INT`) очищает командную строку, а получив сигнал 1 (`HUP`), оболочка завершает работу и выходит из сеанса.

Посылать сигналы процессам могут только их владелец и суперпользователь. Если родительским процессом получен сигнал, приводящий к его остановке, то в нормальной ситуации будут сняты с выполнения все его дочерние процессы.

Удобно пользоваться командами `killall` и `pkill`. Они действуют подобно команде `kill`, позволяя, как и `kill`, послать требуемый сигнал нескольким заданным процессам. В отличие от команды `kill`, для команд `pkill` и `killall` достаточно указать имя командной строки процесса.

Например, следующая команда приведет к немедленному аварийному останову всех копий демона `httpd` (пример 5.21).

Пример 5.21. Команда `killall`

```
# killall -9 httpd
```

В результате передачи сигнала 9 (KILL) всем процессам `httpd` они будут сняты с исполнения, т. к. перехватить такой сигнал невозможно.

ЗАДАНИЯ

- Запустите порожденную оболочку `bash`. Исследуйте, посылая родительской оболочке сигналы `TERM`, `INT`, `QUIT` и `HUP`, что при этом происходит.
- От имени обычного пользователя пошлите сигнал `KILL` любому процессу, запущенному от имени другого пользователя. Что произойдет?
- Запустите в фоновом режиме команду `sleep 1000`. Проверьте, на какие сигналы из следующих: `TERM`, `INT`, `QUIT` и `HUP`, реагирует эта команда.

Перехват и обработка сигналов в Bash

В оболочке `Bash` имеется встроенная команда `trap`, которая позволяет перехватывать сигналы и реагировать на них каким-либо заданным способом. Первым аргументом ее является команда, которую следует выполнить при получении оболочкой сигнала. Второй аргумент задает сигнал, который должен быть обработан.

Введите команды, показанные в примере 5.22, для установки ловушки сигнала `INT`.

Пример 5.22. Перехват сигналов в оболочке Bash

```
$ trap "echo Получен сигнал INT" INT
$ trap -p
trap -- 'echo Получен сигнал INT' SIGINT
$ Получен сигнал INT
```

Команда `trap` установила ловушку для сигнала `INT` — команду `echo`. Команда `trap -p` вывела список установленных обработчиков сигналов. Далее пользователь нажал комбинацию клавиш `<Ctrl>+<C>`, передающую сигнал `INT` оболочке. При этом сигнал был перехвачен обработчиком, выполнившим команду `echo`.

ЗАДАНИЯ

- Запрограммируйте оболочку так, чтобы при получении ею сигнала `TERM` создавался файл `pwd.txt`, содержащий информацию о текущем каталоге.
- Запустите порожденную оболочку. Работает ли в ней созданный обработчик?

Управление приоритетом процессов

В GNU/Linux используется режим выполнения процессов с разделением времени. В каждый момент времени центральный процессор выполняет инструкции одного-единственного процесса, а все остальные процессы находятся в режиме ожидания. Все процессорное время разделено на части — *time slices* (другое название — *TIC*). Маловероятно, чтобы процесс находился на исполнении процессора в течение всего времени *TIC*. Он может быть снят с исполнения более "важным" для системы процессом. Поэтому говорят о *приоритете процессов* в GNU/Linux.

Процессы, обладающие в системе бóльшим приоритетом, исполняются быстрее. Процесс выполняется тем быстрее, чем: чаще процесс попадает на исполнение, чем полнее он использует промежуток времени *time slice*.

Работа по обслуживанию очереди процессов осуществляется планировщиком. Планировщик вычисляет для каждого процесса величину, которую можно увидеть в поле `PR` листинга, выводимого командой `ps -l` (пример 5.23). Чем ниже величина `PR`, тем выше приоритет процесса, следовательно, быстрее он выполняется, поэтому для избежания путаницы далее в тексте вместо слов "увеличение и уменьшение приоритета" будут использованы, соответственно, "улучшение и уменьшение приоритета". Величина `PR` постоянно изменяется, обеспечивая для процессов, которые давно не были на исполнении процессором, улучшение приоритета, и, наоборот, для процессов, которые были исполнены только что, — его ухудшение.

Пример 5.23. Приоритет процессов

```
$ ps -l
```

F	S	UID	PID	PPID	C	PR	NI	ADDR	SZ	WCHAN	TTY	TIME	CMD
0	S	500	1747	1741	0	74	0	-	783	11c541	pts/0	00:00:00	bash
0	R	500	2510	1747	4	77	0	-	618	-	pts/0	00:00:00	ps

На основании вычисленной величины приоритета `PR` ядро определяет, какой процесс следующим попадет на исполнение. В примере 5.23 заметна также величина `NI` — *nice number*. Это число, устанавливаемое пользователем, называется иначе *относительным приоритетом*.

С помощью *nice number* пользователь может влиять на вычисляемую планировщиком величину приоритета процесса. Чем ниже значение *nice number*, тем лучше будет приоритет процесса и тем быстрее он будет работать. В GNU/Linux значение *nice number* задается в пределах от -20 до 19 . По умолчанию *nice number* равно 0 .

Для обычных пользователей отведен диапазон положительных значений `nice number`. В область отрицательных значений эту величину может устанавливать только суперпользователь. То есть обычные пользователи могут жертвовать производительностью своих приложений ради общего быстродействия системы.

Значение `nice number` можно установить с помощью команды `nice`. После нее в качестве аргумента задается команда, которая должна быть исполнена с измененным приоритетом. По умолчанию команда `nice` увеличивает значение `nice number` на 10, ухудшая, таким образом, приоритет этого процесса. Если требуется указать иное значение увеличения `nice number`, то его следует указать после опции `-n`.

Запустим, например, `Bash` с ухудшенным приоритетом (пример 5.24).

Пример 5.24. Запуск процесса с измененным значением `nice number`

```
$ nice -n 19 bash
$ ps -l
F S  UID  PID  PPID  C PRI  NI ADDR  SZ  WCHAN TTY          TIME CMD
0 S   500  1747  1741  0  74   0  -   784 11c541 pts/0    00:00:00 bash
0 S   500  2559  1747  0  78  19  -   780 11c541 pts/0    00:00:00 bash
0 R   500  2560  2559  0  79  19  -   618      - pts/0    00:00:00 ps
$ nice
19
```

В примере 5.24 продемонстрировано, как с помощью команды `nice -n 19 bash` была запущена оболочка `bash` с ухудшенным приоритетом. В поле `NI` для этой оболочки команда `ps -l` выводит значение 19. Поле `PRI` этого же листинга показывает, что приоритет запущенной оболочки `bash` действительно ухудшен (78 у дочерней оболочки против 74 у родительской).

Для установки иного значения `nice number` для уже исполняющегося процесса следует использовать команду `renice`. Эта команда обычно доступна только для суперпользователя. Новое значение `nice number` указывается в качестве аргумента команды `renice`. С помощью этой команды можно изменить `nice number` для конкретного процесса, заданного с помощью его `PID` после опции `-p` или же в качестве второго аргумента. Например, для изменения `nice number` оболочки `bash` из предыдущего примера можно выполнить команду, показанную в примере 5.25.

Пример 5.25. Команда `renice`

```
# renice 10 2559
2559: old priority 19, new priority 10
# ps -l -p 2559
F S    UID    PID    PPID    C PRI  NI ADDR      SZ WCHAN  TTY          TIME CMD
0 S    500    2559    1747    0  76   10  -       782 read_c pts/1    00:00:00 bash
```

Данная команда установила новое значение приоритета (поле `NI`) для оболочки, которая была исходно запущена с `nice number` 19. Заметно, что приоритет процесса (см. поле `PRI`) также изменился.

С помощью команды `renice` можно изменять приоритет всех процессов для заданного после опции `-u` пользователя (`-g` для группы пользователей) — пример 5.26.

Пример 5.26. Изменение приоритета процессов заданного пользователя

```
# renice 0 -u user1
```

ЗАДАНИЯ

- От имени обычного пользователя попытайтесь запустить оболочку `bash` со значением `nice number`, равным `-1`. Какое сообщение выводится?
- От имени суперпользователя запустите команду индексирования базы данных поиска в следующем виде: `time nice -n 19 updatedb`. А затем выполните такую же команду, в которой значение `nice number` для `updatedb` будет `-5`. Сравните полученные результаты.

Глава 6



Права доступа и права владения

В этой главе рассматриваются основополагающие понятия системы контроля доступа DAC — права владения и права доступа. От правильной установки прав доступа на файлы напрямую зависит защищенность системы. Здесь вы также узнаете, что такое специальные биты прав доступа и как они влияют на безопасность системы.

Права владения файлами

Любая UNIX- и GNU/Linux-система реализует политику DAC (Discretionary Access Control) — добровольное управление доступом к файлам. Суть ее состоит в том, что пользователи сами определяют права доступа к своим файлам.

Каждый файл имеет два идентификатора, определяющие его принадлежность — владелец файла и группа пользователей. Эта информация сохраняется не в самом файле, а в его метаданных (inode).

Любой файл принадлежит одному-единственному пользователю. Этот пользователь называется *владельцем* (или пользователем — user) файла. Когда пользователь системы создает файл, то владельцем файла устанавливается именно он.

Первичная группа пользователя (GID) в обычных условиях определяет группу пользователей, которая будет установлена для этого файла.

Права доступа к файлам могут быть определены с помощью команды `ls -l`.

Пример 6.1. Права доступа к файлу

```
$ id
uid=501(user1) gid=100(users) группы=100(users)
$ > file
$ ls -l file
-rw-r--r--  1 user1  users          0 Dec 12 18:03 file
```


В примере 6.1 пользователь `user1` создал файл. Созданный файл принадлежит пользователю `user1` (третий столбец вывода команды `ls -l`) и имеет группу пользователей `users` (четвертый столбец). Первичная группа пользователя (`users`) была установлена, как группа пользователей файла.

В метаданных сохраняется информация не об именах пользователя и группы, а `UID` и `GID` пользователя. В соответствии с требованиями `DAC` в `GNU/Linux` существуют три базовых класса доступа к файлу:

- ☐ User access (`u`) — права доступа владельца файла;
- ☐ Group access (`g`) — права доступа группы владельцев файла;
- ☐ Other access (`o`) — права доступа для всех остальных.

Никакие другие категории не предусматриваются, поэтому каждый пользователь может быть либо владельцем файла, либо входить в группу пользователей, либо относиться к категории всех остальных.

Сами по себе права владения файлами не предоставляют информации о том, что может пользователь делать с данным файлом. Система прав доступа к файлу, описываемая далее, определяет, какие возможности работы с файлом имеет конкретный пользователь системы.

Задания

- Какая группа пользователей установлена для вашего каталога?
- Какие владелец и группа установлены на файл `/bin/ls`?
- Проверьте информацию, полученную выше, с помощью команды `stat`.

Права доступа, устанавливаемые на файлы

Права доступа к файлу хранятся в метаданных файла и кодируются тремя триадами битов (табл. 6.1). Права доступа можно задавать в символической и восьмеричной нотациях. Символическая нотация основана на буквенных обозначениях прав владения и прав доступа, а восьмеричная связана с фактическим представлением этих прав в виде триад битов.

Таблица 6.1. Триады битов прав доступа

u g o								
r	w	x	r	w	x	r	w	x

Порядок триад:

- ☐ старшая триада соответствует правам доступа владельца файла (u);
- ☐ средняя триада — правам доступа группы владельцев (g);
- ☐ младшая триада — правам доступа всех остальных пользователей (o).

Порядок бит в триадах:

- ☐ установленный в 1 старший бит в каждой триаде (4 в восьмеричной нотации) обозначает разрешение на чтение данного файла и в символической нотации обозначается `r--`;
- ☐ установленный в 1 средний бит в каждой триаде (2 в восьмеричной нотации) обозначает разрешение на изменение данного файла: `-w-`;
- ☐ установленный в 1 младший бит в каждой триаде (1 в восьмеричной нотации) обозначает разрешение на исполнение данного файла: `--x`.

Например, запись `rxwxr-x--x` (751) обозначает, что пользователь файла имеет все права на доступ к нему (`rxw` или 7), группа пользователей имеет права на чтение и исполнение файла (`r-x` или 5), остальные имеют права на исполнение (`--x` или 1).

Символьная и восьмеричная нотация записи прав доступа эквивалентны. Восьмеричное значение получается сложением степеней двойки, соответствующих номеру бита в триаде. Например, права доступа в символической нотации `rxwxr-xr--` в восьмеричной нотации записываются как 754, где $7 = 4 + 3 + 1$, $5 = 4 + 1$, $4 = 4$.

Для того чтобы увидеть права доступа к файлу, достаточно набрать команду `ls -l`, при этом права доступа к файлам выводятся в первой колонке.

Задания

- Получите список файлов в домашнем каталоге в подробном формате. Какие права доступа установлены на них?
- Переведите из восьмеричной формы записи прав доступа в символическую 641.
- Переведите запись прав доступа `rw-r-----` в восьмеричную форму.
- Определите, кто является владельцем домашнего каталога пользователя `mail`, какова группа пользователей.
- Имеется файл `script.sh` с правами 750, владелец `root`, группа `sys`. Может ли пользователь `user2`, являющийся членом групп `user2` и `tty`, что-либо изменить в тексте файла?
- Может ли пользователь `user2` запустить на исполнение этот файл?
- Какие права должны быть добавлены на файл, чтобы `user2` мог читать и исполнять этот файл?

Права доступа к каталогам

Права доступа, устанавливаемые на каталоги, имеют несколько иной смысл, чем права на файлы. Для каталогов используются следующие права:

- ❑ `x` — (`search`) право обращаться к метаданным файлов в каталоге, а также входить в каталог с помощью команды `cd` и получать доступ к файлам, находящимся в этом каталоге;
- ❑ `r` — право на чтение имен файлов, находящихся в каталоге, т. е. на выполнение команды `ls`. Без наличия права `search` (`x`) не позволяет получать подробную информацию о файлах и какой-либо доступ к ним, поскольку прав на доступ к метаданным файлов в каталоге нет;
- ❑ `w` — право на запись в каталог, т. е. право переименовывать, удалять, создавать файлы и пр. Без наличия права `search` (`x`) не позволяет осуществлять запись в каталог, т. к. для записи требуется доступ к метаданным файлов.

Без наличия права `search` (`x`), установленного на каталог, операции обращения к файлам, находящимся в нем, не могут быть осуществлены. Поэтому для каталогов права доступа должны быть либо нечетные, либо отсутствовать. Далее приведены права доступа к каталогам, которые имеют практический смысл:

- ❑ `0` (`---`) — прав нет;
- ❑ `1` (`--x`) — имеется право перехода в каталог, можно обращаться к файлам в нем, однако нельзя выполнять команду `ls` и осуществлять какие-либо манипуляции файлами, например, переименование или удаление;
- ❑ `3` (`-wx`) — в каталог можно переходить, разрешены любые манипуляции файлами и можно к ним обращаться, однако получить список файлов командой `ls` невозможно;
- ❑ `5` (`r-x`) — в каталог можно переходить и получать подробную информацию о файлах командой `ls -l`, разрешено обращаться к файлам, однако нельзя осуществлять манипуляции файлами, например, переименование или удаление;
- ❑ `7` (`rwX`) — полные права.

Для получения прав доступа к каталогу следует использовать команду `ls -ld`.

Пример 6.2. Права доступа к каталогу

```
$ ls -ld
drwxr-x--x  10 user1  users      4096 Dec 12 18:03 .
```

В примере 6.2 на текущий каталог установлены права 751, т. е. владелец этого каталога (`user1`) имеет все права на этот каталог, группа (`users`) не имеет прав переименовывать и удалять файлы, поскольку не имеет прав на запись, а для всех остальных каталог является "темным". Остальные могут переходить в этот каталог и обращаться к файлам, находящимся в нем. Однако необходимо знать, какие имена имеют файлы. Команда `ls` имена файлов не отобразит, т. к. прав на чтение каталога нет. Переименовывать или удалять файлы также не удастся, т. к. права у остальных на запись в каталог нет.

ЗАДАНИЯ

- Определите, какие права установлены на ваш домашний каталог.
- Какие права установлены на домашний каталог пользователя `mail`?
- На каталог `d1` установлены права `drwx--x---`. Владелец каталога `adm`, группа `sys`. Может ли пользователь `user1`, принадлежащий группе `sys`, переименовать файл `f1` в этом каталоге?
- Мог бы пользователь `user1` узнать, какие права установлены на файл `f1`?
- Предположим, что на файл `f1` установлены права 640, владелец `adm`, группа `sys`. Сможет ли `user1` просмотреть содержимое файла? Сможет ли он запустить на исполнение этот файл?
- Если бы на этот каталог были бы установлены права 751, то смог бы пользователь `user1` переименовать файл `f1`?
- На что должны быть установлены права, чтобы `user1` смог удалить файл `f1` из каталога `d1`? Какие эти права?
- В вашем домашнем каталоге находится файл `enigma` с правами 600, владельцем `root` и группой `root`. Можете ли вы удалить этот файл?

Изменение прав владения

Права владения файлами изменяют с помощью следующих команд:

- ❑ `chown` — эта команда позволяет менять как владельца файла или каталога, так и группу пользователей файла;
- ❑ `chgrp` — позволяет менять группу пользователей файла.

Команду `chown` может выполнять только суперпользователь. Обычный пользователь может менять командой `chgrp` группу для файла, если, во-первых, он владеет этим файлом и, во-вторых, он входит в группу, которую он устанавливает на файл.

Приведенная в примере 6.3 команда меняет владельца файла.

Пример 6.3. Изменение владельца файла

```
# ls -l f1
-rw-r--r--  1 tania    prof          8 Oct 22 21:04 f1
# chown user1 f1
# ls -l f1
-rw-r--r--  1 user1    prof          8 Oct 22 21:04 f1
```

В примере 6.3 видно, что первый аргумент команды — имя нового владельца файла, а далее идут файлы или каталоги, права на владение которыми изменяются.

В примере 6.4 показана смена группы пользователей файлов `f1` и `text.c`.

Пример 6.4. Изменение группы

```
# ls -l f1 text.c
-rw-r--r--  1 user1    prof          8 Oct 22 21:04 f1
-rw-r--r--  1 prof     prof        175 Dec 13 21:24 text.c
# chgrp tania f1 text.c
# ls -l f1 text.c
-rw-r--r--  1 user1    tania          8 Oct 22 21:04 f1
-rw-r--r--  1 prof     tania        175 Dec 13 21:24 text.c
```

С помощью команды `chown` можно одновременно изменить владельца и группу, причем новый владелец вовсе не обязан быть членом той группы, которая будет установлена на файл (пример 6.5).

Пример 6.5. Изменение владельца и группы командой `chown`

```
# chown tania:sys f1
# ls -l f1
-rw-r--r--  1 tania    sys          8 Oct 22 21:04 f1
```

В примере 6.5 продемонстрировано, как одновременно поменять владельца и группу с помощью команды `chown`.

Исключительно полезной является опция `-c` GNU-версий команд `chown` и `chgrp`. Использование ее позволяет получать подробную информацию об изменяемых правах владения (пример 6.6).

Пример 6.6. Опция -c команды chown

```
# chgrp -c tania fl
изменена группа `fl' на tania
```

Обе команды, `chown` и `chgrp`, имеют опцию `-R`, позволяющую рекурсивно изменять права владения на каталоги и их содержимое (пример 6.7).

Пример 6.7. Рекурсивное изменение прав владения

```
# ls -Rl scores/
scores/:
total 1
drwxrwxr-x    2 prof   prof           80 Aug 24 16:20 rnd_tutorial
scores/rnd_tutorial:
total 4
-rw-rw-r--    1 prof   prof       1040 Aug 24 16:20 000.score
# chown -R tania:tania scores/
# ls -Rl scores/
scores/:
total 1
drwxrwxr-x    2 tania   tania           80 Aug 24 16:20 rnd_tutorial
scores/rnd_tutorial:
total 4
-rw-rw-r--    1 tania   tania       1040 Aug 24 16:20 000.score
```

В примере 6.7 владельцем и группой пользователей стали `tania` (владелец и группа).

При использовании рекурсивной смены прав владения бывает очень удобно получать подробную информацию об этом процессе. Для этого предназначена опция `-v` команд `chown` и `chgrp` (пример 6.8).

Пример 6.8. Опция -v команд chown и chgrp

```
# chgrp -Rv users scores/
изменена группа `scores/' на users
изменена группа `scores//rnd_tutorial' на users
изменена группа `scores//rnd_tutorial/000.score' на users
```

Пример демонстрирует, что с опцией `-v` для каждого файла, на который изменяются права владения, выдается подробная информация.

ВНИМАНИЕ!

Неосторожное использование команд `chown` и `chgrp`, особенно с опцией `-R`, может привести к выводу всей системы из строя!

Задания

- Находясь в сеансе обычного пользователя, попытайтесь изменить права владения на любой файл в вашем домашнем каталоге. Что происходит?
- Перейдите в сеанс суперпользователя и, находясь в домашнем каталоге обычного пользователя (с именем которого вы осуществляете повседневную работу), измените права владения на любой файл так, чтобы он принадлежал пользователю `daemon`.
- Измените группу того же файла на `sys`.
- Одной командой верните исходных владельца и группу этого файла.

Установка прав доступа

Команда `chmod` предназначена для изменения прав доступа к файлам и каталогам, указанным в качестве аргументов. Права доступа должны быть указаны либо в восьмеричной, либо в символьной нотации. Изменять права доступа к файлу могут только суперпользователь и владелец файла.

В примере 6.9 показано использование команды `chmod` для изменения прав доступа к файлу в восьмеричной нотации.

Пример 6.9. Изменение прав доступа в восьмеричной нотации

```
$ ls -l text.c
-rw-r--r--  1 prof  tania           175 Dec 13 21:24 text.c
$ chmod 660 text.c
$ ls -l text.c
-rw-rw----  1 prof  tania           175 Dec 13 21:24 text.c
```

Команда `chmod 660 text.c` установила права 660 на файл `text.c`.

Команда `chmod` позволяет также устанавливать права на доступ к файлу, указывая их в символьной нотации. Для этого применяется следующая форма команды:

`chmod класс изменение права файлы`

Здесь *класс* — один из следующих:

- ☐ *u* — доступ владельца;
- ☐ *g* — доступ группы владельцев;
- ☐ *o* — доступ всех остальных;
- ☐ *a* — доступ всех групп пользователей.

Аргумент *изменение*:

- ☐ *+* — разрешить (добавить);
- ☐ *-* — запретить (убрать);
- ☐ *=* — установить (переписать).

Аргумент *права*:

- ☐ *r* — чтение;
- ☐ *w* — запись;
- ☐ *x* — выполнение.

Предположим, что для файла `text.c` требуется удалить право на запись для группы и добавить право на чтение для остальных. Пример 6.10 демонстрирует это.

Пример 6.10. Изменение прав доступа в символьной нотации

```
$ chmod g-w,o+r text.c
$ ls -l text.c
-rw-r--r--  1 prof  tania           175 Dec 13 21:24 text.c
```

Если используются операции разрешения *+* или запрета *-* прав на файл, то они не изменяют те биты прав доступа, которые не относятся к требуемому изменению.

Наоборот, использование операции назначения *=* стирает те права, которые были установлены ранее, и назначает новые. Так, например, установим на файл `text.c` права на чтение и запись для владельца и группы и запретим всем остальным какой-либо доступ к файлу (пример 6.11).

Пример 6.11. Установка прав доступа со сбросом предыдущих прав

```
$ chmod ug=rw,o= text.c
$ ls -l text.c
-rw-rw----  1 prof  tania           175 Dec 13 21:24 text.c
```


Аналогично командам `chown` и `chgrp`, команда `chmod` способна рекурсивно изменять права доступа к каталогам и всему их содержимому, если она вызвана с опцией `-R`. Однако этой возможностью следует пользоваться с особой осторожностью, принимая во внимание концептуальные отличия прав на файлы от прав на каталоги — на файлы в большинстве случаев устанавливаются четные права (отсутствие прав на исполнение), а на каталоги наоборот — нечетные (без права на `search` каталоги не позволят обращаться к метаданным файлов внутри них).

Приведенная в примере 6.12 команда снимает права на запись для каталога `scores`.

Пример 6.12. Рекурсивная установка прав доступа

```
$ ls -lR scores
scores:
total 1
drwxrwxr-x    2 prof   users           80 Aug 24 16:20 rnd_tutorial
scores/rnd_tutorial:
total 4
-rw-rw-r--    1 prof   users          1040 Aug 24 16:20 000.score
$ chmod -R g-w scores
$ ls -lR scores
scores:
total 1
drwxr-xr-x    2 prof   users           80 Aug 24 16:20 rnd_tutorial
scores/rnd_tutorial:
total 4
-rw-r--r--    1 prof   users          1040 Aug 24 16:20 000.score
```

Видно, что и с самого каталога, и со всего его содержимого были удалены права на запись для группы пользователей.

Рекурсивное изменение прав на каталог и его содержимое используют для дополнения или запрета доступа на чтение или запись. Часто права на каталоги и на файлы устанавливаются отдельно командой `find` с модификатором `-exec chmod`.

Например, в каталоге `scores` требуется установить для файлов права `644`, не затрагивая при этом права на каталоги (пример 6.13).

Пример 6.13. Установка прав на выбранные командой `find` файлы

```
$ find scores -type f -exec chmod 644 {} \;  
$ ls -lR scores  
scores:  
итого 1  
drwxr-xr-x    2 prof  users           80 Авг 24 16:20 rnd_tutorial  
scores/rnd_tutorial:  
итого 4  
-rw-r--r--    1 prof  users          1040 Авг 24 16:20 000.score
```

GNU-версия команды `chmod` позволяет использовать опцию `-v` для получения подробной информации о файлах (пример 6.14), для которых изменяются права, и опцию `-c` для получения подробностей изменения прав.

Пример 6.14. Использование опции `-v` команды `chmod`

```
$ chmod -Rv g-w scores  
права доступа `scores` изменены на 0755 (rwxr-xr-x)  
права доступа `scores/rnd_tutorial` изменены на 0755 (rwxr-xr-x)  
права доступа `scores/rnd_tutorial/000.score` изменены на 0644(rw-r--r--)
```

Однако опция `-v` выдает подробную информацию всегда, а `-c` только тогда, когда права действительно изменяются (пример 6.15).

Пример 6.15. Использование опции `-c` команды `chmod`

```
$ chmod -c g-w text.c  
права доступа `text.c` изменены на 0640 (rw-r-----)  
$ chmod -v 660 text.c  
права доступа `text.c` изменены на 0660 (rw-rw----)
```

Задания

- Создайте цепочку каталогов `d1/d2/d3`, а в них — файлы `d1/f1`, `d1/d2/f2`, `d1/d2/d3/f3`. Установите на файл `d1/f1` права 400. Можете ли вы изменить этот файл?
- Рекурсивно добавьте права на чтение и запись для каталога `d1`. Можете вы теперь изменить файл `f1`?
- Отнимите у каталога `d1` права на чтение и запись. Можно ли теперь получить информацию о файлах в этом каталоге?
- С помощью команды `find` при условии установки `-exec` измените права на каталоги и все подкаталоги `d1` на 750.
- Командами `find`, `xargs` и `chmod` установите права на все обычные файлы в каталоге `d1` равными `-rw-r--r--`.

Автоматическая установка прав доступа к вновь создаваемым файлам

Команда `umask` предназначена для автоматической установки прав доступа к вновь создаваемым файлам и каталогам. Она позволяет задавать значение битовой маски, которая будет "вычитаться" из прав 777 для каталогов и 666 для файлов. При вызове этой команды без аргумента она возвратит текущее значение маски (пример 6.16).

Пример 6.16. Маска прав доступа

```
$ umask
0022
```

Установка другого значения `umask` никоим образом не отразится на уже существующих файлах и каталогах, она участвует только в процессе определения прав на вновь создаваемые файлы и каталоги (пример 6.17).

Пример 6.17. Изменение `umask`

```
$ umask 002
$ mkdir dir1
$ > file1
$ ls -ld dir1 file1
drwxrwxr-x    2 user1  user1      48 Dec 14 20:43 dir1
-rw-rw-r--    1 user1  user1       0 Dec 14 20:43 file1
$ umask 077
$ mkdir dir2
$ > file2
$ ls -ld dir2 file2
drwx-----    2 user1  user1      48 Dec 14 20:44 dir2
-rw-----    1 user1  user1       0 Dec 14 20:44 file2
```

В примере 6.17 продемонстрировано, что при установленном значении `umask` 002 на каталоги устанавливаются права 775, а на файлы — 664. В то же время величина `umask`, установленная в 077, дает в результате, соответственно, 700 — для каталогов и 600 — для файлов.

В табл. 6.2 приведены наиболее часто применяемые значения `umask`.

Таблица 6.2. Значения `umask`

<code>umask</code>	Каталоги	Файлы
002	775	664
007	770	660
022	755	644
027	750	640
077	700	600

Значение `umask` можно задавать также и в символьной нотации (пример 6.18).

Пример 6.18. Установка `umask` в символьной нотации

```
$ umask g=rwx,g=rx,o=
$ umask
0027
```

При задании значения `umask` в символьной нотации всего лишь требуется указать в качестве аргумента права, которые должны будут иметь новые каталоги.

Задания

- Установите значение `umask 000`. Проверьте, с какими правами создаются новые каталоги и файлы.
- В символьной форме установите такое значение `umask`, чтобы вновь создаваемые файлы имели права доступа 644.
- Требуется, чтобы владелец создаваемого каталога мог создавать, читать и записывать файлы в каталог, а также переходить в него. Члены группы владельцев должны иметь права на создание и удаление файлов в этом каталоге. Все остальные никаких прав иметь не должны. Каким должно быть значение маски для того, чтобы удовлетворить перечисленным требованиям?

Специальные биты прав доступа: SUID, SGID и sticky bit

Помимо битов, устанавливающих разрешения на доступ к файлу, существуют специальные атрибуты, для которых предназначена еще одна триада битов:

- ☐ sticky bit (save text mode) — бит "липучка";
- ☐ SUID — бит подмены UID;
- ☐ SGID — бит подмены GID.

Sticky bit кодируется восьмеричной 1 (двоичная 001), SGID кодируется восьмеричной 2 (010), а SUID — 4 (100). В символьной нотации применяются символы `t` для sticky bit, `s` для SUID и SGID. Эти символы всегда выводятся в позиции, где обычно стоит флаг разрешения на исполнение (`x`). При этом:

- ❑ SUID отображается в виде буквы `s` в старшей триаде битов, отображающей права владельца;
- ❑ SGID отображается в виде буквы `s` в средней триаде битов, отображающей права группы;
- ❑ sticky bit отображается в виде буквы `t` в младшей триаде битов, отображающей права для всех остальных.

Прописные символы `S` и `T` выводятся при отсутствии в этих триадах прав на исполнение, что обычно сигнализирует о ненормальном состоянии прав доступа.

Пример 6.19. Специальные биты прав доступа

```
$ ls -ld /tmp /bin/ping
-rws--x--x   1 root      root          32908 Окт 16   2002 /bin/ping
drwxrwxrwt   94 root      root          3488 Дек 14  20:31 /tmp
```

Приведенный пример 6.19 показывает, что на файл системной команды `ping` установлен бит SUID (символ `s` в старшей триаде вместо прав на исполнение), а на каталог `/tmp` установлен sticky bit (символ `t` в триаде прав остальных).

Атрибут sticky bit в GNU/Linux для файлов не используется, в ранних версиях UNIX он был предназначен для того, чтобы оставить в памяти образ программы (save text mode).

Обычный процесс наследует права доступа к системным ресурсам от пользователя, запустившего процесс, и его первичной группы (UID и GID), однако для исполняемых файлов, на которые установлены биты SUID и/или SGID, это не так.

При установленном на исполняемый файл бите SUID процесс выполняется не от имени пользователя, запустившего его, а от имени владельца исполняемого файла команды. Аналогично, при установленном бите SGID процесс выполняется не от имени первичной группы пользователя, запустившего процесс, а от имени группы пользователей файла.

У каждого процесса имеются четыре идентификатора:

- ❑ RUID — Real UID, который всегда равен UID пользователя, выполнившего команду;

- ❑ **RGID** — Real GID, который всегда равен GID пользователя, выполнившего команду;
- ❑ **EUID** — Effective UID, который равен либо RUID, либо если на исполняемый файл установлен бит SUID, то UID владельца файла;
- ❑ **EGID** — Effective GID, который равен либо RGID, либо если на исполняемый файл установлен бит SGID, то GID владельца файла.

В подавляющем большинстве случаев подмена владельца или группы осуществляется на `root` или какого-либо привилегированного пользователя или группу. Например, при выполнении команды `ping` (см. пример 6.19), несмотря на то, что ее запустил обычный пользователь, она будет исполняться от имени `root`, т. к. он владеет ее исполняемым файлом. Это используется, например, в таких программах, как `passwd`, которые требуют временного предоставления доступа обычному пользователю к тем ресурсам, к которым он не имеет доступа. Естественно, такие программы требуют особого подхода к разработке, т. к. представляют серьезную угрозу для безопасности системы. На файлы сценариев Shell биты SUID и SGID устанавливать можно, но они действовать не будут.

Установка sticky bit на каталог, в отношении которого пользователь имеет права на чтение и на запись, позволяет запретить удалять и изменять пользователю чужие файлы в этом каталоге. Это используется при установке прав доступа к каталогу `/tmp`, открытому на запись всем, поскольку иначе пользователь может удалить чужие временные файлы, находящиеся в этом каталоге, что может повлечь нежелательные последствия.

При установке атрибута SGID на каталог вновь созданные файлы в этом каталоге будут наследовать группу владельцев по группе владельцев каталога (так называемый "стиль BSD"), вместо RGID процесса, создающего файл (пример 6.20).

Пример 6.20. Каталог с установленным битом SGID

```
$ cd dir1
$ ls -ld
drwxrwsr-x    2 tania    users          48 Dec 14 20:43 .
$ id
uid=500(prof) gid=500(prof) группы=500(prof),100(users)
$ > file
$ ls -l
total 0
-rw-r-----    1 prof    users          0 Dec 14 22:00 file
```

В каталоге, на который установлен бит SGID, был создан файл (см. пример 6.20). При этом группа владельцев файла была назначена не по первичной группе пользователя, создавшего файл, а по группе пользователей каталога, в котором файл был создан.

Команда `chmod` позволяет установить особые биты доступа на файлы и каталоги. Для этого команда `chmod` может быть выполнена со следующими аргументами:

- ☐ `u+s` — для установки на файл бита SUID;
- ☐ `g+s` — для установки на файл или каталог бита SGID;
- ☐ `o+t` — для установки на каталог бита sticky bit.

Не обязательно устанавливать специальные биты в символьной нотации. Приведенная в примере 6.21 команда `chmod 2775 d1` устанавливает бит SGID на каталог.

Пример 6.21. Установка SGID на каталог в восьмеричной нотации

```
$ mkdir d1
$ ls -ld d1
drwxr-x---  2 prof  prof          48 Dec 14 22:31 d1
$ chmod 2770 d1
$ ls -ld d1
drwxrws---  2 prof  prof          48 Dec 14 22:31 d1
```

В табл. 6.3 указаны результаты установки различных специальных прав доступа на файлы и каталоги.

Таблица 6.3. Специальные биты прав доступа

Права	Эффект для каталогов	Эффект для файлов
-rws--x--x	—	Команда выполняется от имени владельца файла
-rwx--s--x	—	Команда выполняется от имени группы пользователей файла
drwxrws---	На файлы, создаваемые в каталоге, будет установлена такая же группа, как у каталога	—
drwxrwxrwt	В каталоге можно удалять или переименовывать только собственные файлы	—

Задания

- Проверьте, какие биты прав доступа установлены на исполняемый файл команды `passwd`.
- С помощью `awk` получите список всех процессов в системе, для которых RUID не равен EGID.
- На какой-либо виртуальной консоли запустите команду `passwd` от имени обычного пользователя. Выполните ту же команду, что и в предыдущем пункте. Имеется ли в требуемом списке процесс `passwd`?
- Используя `find`, запишите в файл `SUGID.txt` все имена файлов, на которые установлены биты SUID или SGID.
- Найдите в системе все каталоги с установленными битами SGID или sticky bit.
- Создайте каталог и (будучи `root`) установите на этот каталог группу `users`. Установите бит SGID на этот каталог. В сеансе обычного пользователя создайте в этом каталоге файл. Проверьте, кому он принадлежит и какая группа пользователей у этого файла.



ЧАСТЬ III

**Утилиты
командной строки**



Глава 7

Редактор *vi*

Многие считают редактор *vi* анахронизмом. Но на самом деле *vi* — это удивительно удобный и мощный инструмент редактирования конфигурационных файлов и программ. Администратор GNU/Linux должен иметь достаточный навык работы с *vi*, т. к. при аварии *vi* будет, возможно, единственным средством редактирования текста.

Запуск *vi* и режимы его работы

Полноэкранный текстовый редактор *vi* (от англ. *visual*) универсален, но сейчас он чаще всего используется для редактирования исходных текстов программ и конфигурационных файлов. Команды его подобраны таким образом, что он будет работать даже на терминалах, не обладающих клавишами управления курсором. В современных версиях GNU/Linux обычно устанавливается редактор текста *vim* (*vi improved*), являющийся дальнейшим развитием *vi*. Имеется также графическая оконная версия *gvim* редактора *vi*. Она предоставляет современный интерфейс меню, что значительно облегчает работу для начинающих.

Запускают *vi* следующим образом:

- ❑ *vi* — в таком случае *vi* будет запущен для ввода текста;
- ❑ *vi -* — текст будет считан из стандартного потока ввода (*stdin*);
- ❑ *vi filename* — файл будет открыт для редактирования;
- ❑ *view filename* или *vi -R filename* — файл будет открыт для просмотра;
- ❑ *vi +[num] filename* — курсор будет поставлен на строку с номером *num* или на последнюю строку, если *num* не указан;
- ❑ *vi +/regexp filename* — файл будет открыт для редактирования и курсор будет установлен на первое вхождение регулярного выражения *regexp*.

Для "срочной эвакуации" из `vi` следует нажать клавишу `<Esc>`, а затем набрать команду `:q!`. Эта команда обеспечит выход из `vi` без сохранения изменений.

Редактор `vim` обладает развитой системой помощи, которая может быть получена посредством команды `:help`, если ее набрать после нажатия клавиши `<Esc>`.

Редактор `vi` обладает тремя различными режимами работы.

- ❑ *Командный режим*, в котором `vi` оказывается при его запуске и при нажатии клавиши `<Esc>`. В этом режиме осуществляется перемещение курсора, просмотр и редактирование текста.
- ❑ *Режим ввода текста*, в который `vi` переходит при вызове любой из команд вставки или добавления текста, например, `i`. В этом режиме не следует пользоваться клавишами перемещения курсора по тексту. Этот режим используется исключительно для ввода нового текста. Для выхода из этого режима применяется клавиша `<Esc>`.
- ❑ *Режим двоеточия*, или, иначе, режим последней строки, в который `vi` переходит при нажатии клавиши `<:>` в командном режиме. В этом режиме работают такие команды, как открытие нового файла или установка нумерации строк. Официальное название этого режима — *ex mode*, т. к. в нем можно использовать команды однострочного редактора `ex` (развитая версия `ed`).

Если вам необходимо ввести новый текст, то надо в командном режиме набрать команду `i` (`insert`), в результате чего `vi` перейдет в режим вставки, в котором можно набирать текст. Выйти из режима вставки можно с помощью нажатия клавиши `<Esc>`. Команда `u` — отказ от изменений (`undo`). В классическом `vi` эта команда отменяет последнее действие, а в `vim` можно последовательно отменять несколько команд.

Для сохранения изменений в тексте можно использовать команду `zz`, которая обеспечит выход из `vi` с сохранением изменений в редактируемом файле. Вместо команды `zz` можно воспользоваться командой `:x`.

Задания

- Откройте в режиме только для чтения файл `/etc/passwd` так, чтобы курсор был установлен на вашу учетную запись.
- Выйдите из `vi` без сохранения изменений.
- Создайте новый файл `text.txt`, открыв его в `vi` и выйдя с сохранением изменений.

- Войдите в режим вставки, выполнив команду *i*. Введите произвольный текст и покиньте режим вставки (клавиша <Esc>).
- Проверьте, работает ли команда отказа от изменений *u*.
- Пользуясь встроенной системой помощи *vim*, найдите, какие еще варианты перехода в режим вставки имеются.
- Выйдите с сохранением изменений.

Перемещение курсора по тексту в vi

В редакторе vi перемещение по тексту можно осуществлять с помощью привычных клавиш управления курсором и клавиш <PgUp>, <PgDn>. Однако vi был разработан с учетом поддержки даже таких старых или специализированных терминалов, которые этими клавишами не обладают.

Как же обойтись без клавиш управления курсором? В командном режиме редактора vi для ввода команд используют только обычные клавиши алфавитно-цифровой клавиатуры, а также клавиши <Ctrl>, <Esc> и <Enter>.

Для перемещения курсора по тексту подобно клавишам управления курсором используются четыре команды, на клавиатуре соответствующие им клавиши расположены рядом (<H>, <J>, <K> и <L>):

- *h* — смещает курсор влево на одну позицию;
- *j* — на строку вниз;
- *k* — на строку вверх;
- *l* — на позицию вправо.

Для смещения курсора сразу на несколько строк или позиций перед любой из указанных команд можно набирать в виде цифры количество строк или позиций.

Например, следующая команда переместит курсор на двадцать строк вверх: *20k*.

Для прокрутки страниц используются сочетания: <Ctrl>+<F> (от англ. *forward*) — вперед и <Ctrl>+ (от англ. *backward*) — назад. По аналогии: <Ctrl>+<D> — вниз (*down*), <Ctrl>+<U> — вверх (*up*).

Другие часто используемые команды перемещения по тексту приведены в табл. 7.1.

Таблица 7.1. Команды перемещения по тексту в *vi*

Команда	Действие
w	На слово вправо
b	На слово влево
e	Установить курсор в конец слова
0	Курсор в начало строки без учета отступа, т. е. в самое начало
^	Курсор на первый символ строки, не являющийся пробелом или табуляцией
\$	Курсор в конец строки
)	Курсор на начало следующего предложения
(Курсор на начало предыдущего предложения
}	Курсор в начало следующего абзаца
{	Курсор в начало предыдущего абзаца
G	На последнюю строку документа
#G	Переместить курсор на # — номер строки (например, 20G)
H	Установить курсор на первую строку экрана
L	Курсор на последнюю строку экрана
z+	Поместить текущую строку вверх экрана
z-	Поместить текущую строку вниз экрана
z.	Поместить текущую строку в центр экрана

Задания

- Запустите *vi* так, чтобы он читал данные из потока стандартного ввода, и введите несколько строк так, чтобы в начале строк были символы табуляции или пробелы.
- В введенном тексте проверьте отличия команд позиционирования *^* и *0*.
- Перейдите в конец файла с помощью одной команды. Находится ли курсор в начале строки?
- Выполните команду: `<Esc> :r! ps aux`. Вы должны увидеть список процессов, сгенерированный командой `ps aux`. Вы можете редактировать этот текст?
- Перейдите на тридцатую строку текста. Проверьте ее номер, нажав `<Ctrl>+<G>`.
- Позиционируйте текущую строку в центр экрана.
- Проверьте, работают ли команды *H* и *L*.
- Отличается ли действие команд *z+* и *z-* от команд *H* и *L*?
- Выключите режим нумерации строк командой: `<Esc> :set nonumber`.
- Работают ли клавиши управления курсором и клавиши `<PgUp>` и `<PgDn>`?

Команды *vi* редактирования текста

Несмотря на то, что при правильных настройках терминала редактор *vim* позволяет перемещаться по тексту с помощью обычных клавиш управления курсором даже в режиме вставки текста, а клавиши `` и `<Backspace>` ведут себя ожидаемым образом, полагаться на привычные приемы редактирования текста нельзя.

Во-первых, оригинальный *vi* (не *vim*) не поддерживает многие привычные средства редактирования, а во-вторых, даже современный редактор *vim* может не распознать тип терминала (например, из-за того, что переменная окружения `TERM` установлена неверно). Такие эффекты довольно часто наблюдаются при работе в удаленном сеансе с помощью сетевых эмуляторов терминала. В таких случаях обычные клавиши редактирования работать не будут.

Команды для редактирования текста приведены в табл. 7.2.

Таблица 7.2. Команды редактирования текста

Команда	Действие
<code>i</code>	Переход в режим вставки в позиции курсора
<code>I</code>	Переход в режим вставки в начале строки
<code>a</code>	Добавление после текущего символа
<code>A</code>	Добавление в конец строки
<code>o</code>	Вставка строки после текущей строки с переходом в режим вставки
<code>O</code>	Вставка строки до текущей строки с переходом в режим вставки
<code>r</code>	Замена символа в текущей позиции
<code>R</code>	Переход в режим замещения
<code>x</code>	Удаление символа в позиции курсора
<code>X</code>	Удаление предыдущего символа
<code>s</code>	Замена текущего символа и переход в режим вставки
<code>S</code> <code>cc</code>	Обе эти команды заменяют текущую строку с переходом в режим вставки
<code>cw</code> <code>cW</code>	Замена слова до пробела, табуляции (обе команды) или знака препинания (<code>cw</code>)
<code>dw</code> <code>dW</code>	Удаление слова до пробела, табуляции (обе) или знака препинания (<code>dw</code>)
<code>dd</code> <code>D</code>	Удаление строки полностью (<code>dd</code>) или вправо от курсора (<code>D</code>)

Таблица 7.2 (окончание)

Команда	Действие
<code>yy Y</code>	Копирование строки в буфер обмена
<code>yw yW</code>	Копирование слова до пробела, табуляции (обе) или знака препинания (<code>dw</code>)
<code>P</code>	Вставка из буфера после текущей позиции
<code>p</code>	Вставка из буфера до текущей позиции
<code>~</code>	Смена регистра текущего символа
<code>J</code>	Объединение строк
<code>u</code>	Отмена действия (от англ. <i>undo</i>)

Многие команды, перечисленные в таблице, допускают использование перед ними числовых квантификаторов. Они указывают, сколько раз должно быть выполнено требуемое действие.

Например, команда `10x` удалит десять символов, начиная с текущей позиции. Команда `2Y` запомнит в буфере обмена две строки, а команда `10P` десять раз произведет их вставку (только в vim).

ЗАДАНИЯ

- Запустите `vi`, наберите команду `11o` и введите строку `192.168.1.1`. Далее нажмите клавишу `<Esc>`. Что произошло?
- Удалите последнюю строку.
- С помощью команды `sw` измените IP-адрес, заданный в последней строке, на `192.168.200.1`.
- В предпоследней строке измените адрес на `192.168.200.121`. Какая команда уместнее в этом случае?
- Перейдите в начало файла и удалите с помощью одной команды четыре строки.
- Требуется ввести IP-адрес `10.10.10.10`. Как это можно сделать, введя в режиме вставки строку `10.` только один раз?

Команды поиска и замены строк

Для поиска строки по шаблону можно воспользоваться следующими командами:

- ☐ `/шаблон` — для поиска с текущей позиции до конца файла;
- ☐ `?шаблон` — для поиска с текущей позиции в начало файла.

Эти команды установят курсор на начало первой найденной строки, удовлетворяющей шаблону. Для продолжения поиска надо набрать команду `n`.

Поиск с заменой найденных строк в `vi` осуществляется с помощью команды режима последней строки `:s/шаблон/замена/`.

Для демонстрации команд замены текста здесь использован текст, приведенный в примере 7.1.

Пример 7.1. Исходный текст

```
The top ten & ten CC. The best Ten in the World of tens.
```

```
Another ten.
```

```
Another Ten.
```

В приведенном тексте несколько раз встречается слово `ten`, а также `Ten`. Предполагается, что курсор находится в первой строке текста.

Для замены в текущей строке первого вхождения строки `ten` на `10` следует ввести команду, приведенную в примере 7.2.

Пример 7.2. Поиск и замена строк

```
:s/ten/10/
```

В результате команды из примера 7.2, получим текст, показанный в примере 7.3.

Пример 7.3. Текст после замены слова `ten` на строку `10`

```
The top 10 & ten CC. The best Ten in the World of tens.
```

```
Another ten.
```

```
Another Ten.
```

Первое вхождение строки `ten` было заменено на `10` в текущей строке, но не в других строках. Обратите внимание: было заменено именно первое вхождение строки.

Если же необходимо в текущей строке заменить все вхождения, а не только первое, то в конец этой команды надо добавить модификатор `g` (от англ. *go*) (пример 7.4).

Пример 7.4. Поиск всех вхождений шаблона в строке

```
:s/ten/10/g
```

Результат работы команды из примера 7.4 показан в примере 7.5.

Пример 7.5. Текст после замены всех вхождений `ten` строкой `10`

```
The top 10 & 10 CC. The best Ten in the World of 10s.
```

```
Another ten.
```

```
Another Ten.
```

Бывает необходимо заменить строки без учета их регистра, например, `ten` и `Ten` на `10`. Это позволяет сделать модификатор `i` (пример 7.6).

Пример 7.6. Поиск и замена с игнорированием регистра

```
:s/ten/10/i
```

В примере 7.7 показан текст после выполнения команды из примера 7.6.

Пример 7.7. Текст после поиска с игнорированием регистра

```
The top 10 & 10 CC. The best 10 in the World of 10s.
```

```
Another ten.
```

```
Another Ten.
```

Для замены в нескольких строках в приведенные выше команды требуется добавить указание диапазона поиска. Например, для поиска по всему файлу и замены всех вхождений искомой строки можно ввести команду, приведенную в примере 7.8.

Пример 7.8. Поиск всех вхождений строки по всему файлу

```
:%s/ten/10/gi
```

Оператор `%` указывает диапазон поиска — весь файл. Модификатор `i` добавлен для игнорирования регистра при поиске.

В результате выполнения команды из примера 7.8 текст будет выглядеть так, как это показано в примере 7.9.

Пример 7.9. Текст после замены строк по всему тексту с игнорированием регистра

The top 10 & 10 CC. The best 10 in the World of 10s.
Another 10.
Another 10.

Диапазон поиска можно указывать:

- ☐ номерами начальной и конечной строк, например: `:3,5s/ten/10/;`
- ☐ номером конечной строки для замены с текущей строки: `:,5s/ten/10/;`
- ☐ смещением от текущей строки: `:+5s/ten/10/;`
- ☐ символом конца файла: `:3,$s/ten/10/.`

Задания

- Перейдите на начало файла, использованного в примерах данного раздела, и найдите все строки, содержащие 10.
- Замените во всех строках, начиная с текущей и до конца файла, вхождения строки 10 на строку 127.
- Замените только в текущей строке 127 на 721.
- Откройте в режиме для чтения файл `/etc/passwd` и замените все вхождения символов двоеточия строкой `%%`.
- Работает ли команда отклонения изменений `ц`, если изменения были порождены командой замены строк `:s`?

Команды режима двоеточия

Режим двоеточия (*ex mode*) предназначен для обеспечения совместимости с однострочным редактором *ex*, являющимся расширенной версией редактора *ed*. Эти редакторы, несмотря на весьма почтенный возраст, до сих пор поставляются в современных версиях GNU/Linux. Это связано с исключительным удобством пакетной обработки текста этими редакторами и родственным им неинтерактивным редактором *sed*, о котором будет рассказано позже.

Команды режима двоеточия можно разделить на три категории:

- ☐ файловые;
- ☐ команды для выполнения команд оболочки;
- ☐ служебные.

Под файловыми командами в *vi* понимают разнообразные команды, предназначенные для манипулирования файлами. Команды, предназначенные для

выполнения команд оболочки, возвращают результаты работы команд оболочки в редактируемый текст. Команда `:s` для замены текста, описанная в предыдущем разделе, является примером служебной команды режима двоеточия.

Все файловые команды в `vi` выполняются в режиме последней строки. Исключение составляет команда `zz`, которая позволяет покинуть `vi` с сохранением редактируемого файла.

Таблица 7.3 демонстрирует наиболее важные команды режима двоеточия в `vi`.

Таблица 7.3. Команды режима двоеточия

Команда	Действие
<code>:q</code>	Выход из редактора
<code>:q!</code>	Выход из редактора без сохранения изменений
<code>:wq</code>	Выход из редактора с сохранением изменений
<code>:x</code>	Выход с сохранением, если текст был изменен
<code>:w</code>	Сохранение редактируемого файла
<code>:w имя</code>	Задание имени файлу и сохранение или запись с другим именем
<code>:e имя</code>	Открытие файла для редактирования
<code>:e! имя</code>	Открытие файла с отказом от изменений в редактируемом файле
<code>:r имя</code>	Вставка текста из файла в позиции курсора
<code>:r! команда</code>	Выполнение команды оболочки и вставка ее текстового вывода в файл
<code>:y</code>	Копирование строки в буфер (можно указывать диапазон строк)
<code>:d</code>	Удаление строк
<code>:set showmode</code>	Показывать текущий режим работы
<code>:set number</code>	Показывать номера строк

Особенностью служебных команд `vi`, начинающихся с `:set`, является то, что для отмены этого режима надо сделать такую же команду, но с добавкой префикса `no`.

Команда, выполненная в примере 7.10, включит нумерацию строк. Для отключения этого режима следует выполнить команду, приведенную в примере 7.11.

Пример 7.10. Включение нумерации строк

```
:set number
```

Пример 7.11. Выключение нумерации строк

```
:set nonumber
```

Команда `:set nonumber` отключит вывод номеров страниц.

ЗАДАНИЯ

- Запустите `vi` и вставьте в новый файл результаты работы команды `ps -ef`.
- Сохраните файл с именем `ps.txt`.
- Внесите в текст любые изменения и отмените их полностью.
- Командой `: , $y` скопируйте строки, начиная от текущей и до конца файла.
- Вставьте их в начало файла, используя команды `lg` и `P`.
- Удалите все строки, содержащие слово `root`. Команда `:/root/d`.



Глава 8

Текстовые файлы и потоки

Знание инструментов обработки текста исключительно важно для автоматизации повседневной работы администратора. GNU/Linux предоставляет множество удобных и эффективных средств обработки текста. В этой главе вы познакомитесь с важнейшими текстовыми утилитами GNU/Linux.

Перенаправление потоков ввода/вывода

При открытии файла его имя ассоциируется с так называемым файловым дескриптором — целым числом, которым затем оперируют все остальные процедуры чтения или записи. Со всяким процессом при его создании автоматически связываются три потока:

- ❑ стандартный поток ввода (`stdin`), файловый дескриптор которого — 0;
- ❑ стандартный поток вывода (`stdout`), файловый дескриптор — 1;
- ❑ стандартный поток вывода ошибок (`stderr`), файловый дескриптор — 2.

Стандартные потоки не надо специально открывать в программах, т. к. они автоматически ассоциируются с процессом при его создании.

Поток ввода открыт на чтение, а потоки вывода и ошибок — на запись. Обычно по умолчанию стандартный поток ввода связан с клавиатурой, а стандартные потоки вывода и ошибок — с дисплеем.

Оболочка Bash позволяет перенаправить стандартные потоки, для чего используются следующие операторы:

- ❑ `<` — оператор перенаправления стандартного потока ввода;
- ❑ `>` или `1>` — операторы перенаправления стандартного потока вывода;
- ❑ `2>` — оператор перенаправления стандартного потока вывода ошибок.

Например, перенаправив стандартный поток ввода команды `mail`, можно отправить электронное письмо с текстом, содержащимся в файле `letter` (пример 8.1).

Пример 8.1. Перенаправление потока ввода

```
$ mail -s 'Privet!' user1 < letter
```

Команда `mail` (пример 8.1) отправит электронное письмо пользователю `user1`. Тема письма (Subject) указана после опции `-s`, а текст письма передан через стандартный поток ввода команде `mail` из файла `letter`.

Следующая команда перенаправит поток вывода в файл `ls.txt` (пример 8.2).

Пример 8.2. Перенаправление потока вывода

```
$ ls > ls.txt
```

В файле `ls.txt` окажется список файлов из текущего каталога, выведенный командой `ls` в стандартный поток вывода.

Аналогично в файл можно перенаправить поток ошибок (пример 8.3).

Пример 8.3. Перенаправление потока ошибок

```
$ ls -ld /etc /ctc 2> ls.err
drwxr-xr-x  87 root      root          6064 Dec 15 18:57 /etc
$ cat ls.err
ls: /ctc: No such file or directory
```

В примере 8.3 поток вывода ошибок был перенаправлен в файл `ls.err`. В качестве аргументов команды `ls -ld` были заданы каталог `/etc` и несуществующий каталог `/ctc`. Информация о каталоге `/etc` была выведена в стандартный поток вывода (на экран), а о `/ctc` — в стандартный поток вывода ошибок, который был перенаправлен в файл `ls.err`. Содержимое файла `ls.err` было выведено с помощью команды `cat`.

Модифицируем предыдущую команду так, чтобы поток вывода был перенаправлен в файл `ls.txt` одновременно с перенаправлением потока ошибок в `ls.err` (пример 8.4).

Пример 8.4. Перенаправление потока вывода и потока вывода ошибок

```
$ ls -ld /etc /ctc > ls.txt 2> ls.err
```

Если же необходимо использовать один и тот же файл `ls.txt` для записи потока вывода и потока ошибок, то следует применить оператор сцепления потоков `&` (пример 8.5).

Пример 8.5. Сцепление потоков вывода и вывода ошибок

```
$ ls -ld /etc /etc > ls.txt 2>&1
```

Оболочка Bash позволяет сделать это еще проще (пример 8.6).

Пример 8.6. Сцепление потоков в Bash

```
$ ls -ld /etc /etc &> ls.txt
```

Следует особо подчеркнуть, что операции перенаправления потоков вывода и вывода ошибок в файл стирают его содержимое, записывая новое содержимое взамен старого. Этим можно пользоваться для стирания содержимого файлов и создания новых пустых файлов. Так, например, для стирания содержимого файла `ls.txt` можно использовать команду перенаправления (пример 8.7).

Пример 8.7. Очистка содержимого файла с помощью перенаправления

```
$ > ls.txt
```

Эффект замещения старого содержимого файлов, возникающий при перенаправлении в них потоков вывода и ошибок, часто бывает нежелателен. Оболочка Bash позволяет исключить стирание содержимого файлов при перенаправлении в них потоков вывода или ошибок с помощью команды `set -o noclobber`.

Однако все же есть способ даже при установленной опции `noclobber` переписать содержимое существующего файла с помощью операции перенаправления вывода или вывода ошибок. Для этого можно воспользоваться операторами:

- ❑ `>|` — перенаправление потока вывода с гарантированной перезаписью файла;
- ❑ `2>|` — перенаправление потока ошибок с гарантированной перезаписью файла.

Пример 8.8. Гарантированная перезапись файла

```
$ set -o noclobber
$ ls -l > ls.txt
bash: ls.txt cannot overwrite existing file
$ ls -l >| ls.txt
```

Последняя команда примера запишет подробную информацию о содержимом текущего каталога в существующий файл `ls.txt`, не обращая внимания на то, что файл уже существует и установлена опция оболочки `noclobber`.

Сбросить опцию `noclobber` можно с помощью команды `set +o noclobber`.

Можно дописать в существующий файл информацию из потока вывода вместо перезаписи файла. Для этого оболочка предоставляет операторы:

- ❑ `>>` — для перенаправления потока вывода на добавление к файлу;
- ❑ `2>>` — для перенаправления потока вывода ошибок на добавление к файлу.

Многие команды, работающие с текстом, позволяют использовать поток ввода вместо открытия на чтение файла. Для завершения ввода с клавиатуры следует нажать комбинацию клавиш `<Ctrl>+<D>` для передачи в поток ввода символа окончания потока. После этого команда, читающая стандартный поток ввода, завершит свою работу (пример 8.9).

Пример 8.9. Передача данных через поток ввода с клавиатуры

```
$ cat > f1
Privet
<Ctrl>+<D>
$ cat f1
Privet
```

В примере 8.9 команда `cat > f1` получила через стандартный ввод текст `Privet` с клавиатуры, который был записан через стандартный поток вывода в файл `f1`.

В сценариях оболочки часто возникает необходимость передать блок текста, находящийся непосредственно в тексте сценария, какой-либо команде этого же сценария. В таких случаях используют конструкцию *here document* (документ здесь) — `<<` (два знака "меньше"). В таком случае для окончания ввода используется любой удобный символ вместо символа конца файла. Этот символ должен быть указан после `<<` и должен быть единственным символом

в строке. Так, в примере 8.10 пользователю `user1` послано письмо. Тело письма передается через поток ввода команды `mail` с помощью `here document`. В качестве символа окончания потока ввода установлена точка.

Пример 8.10. Конструкция `here document`

```
$ mail -s HereDoc user1 << .
Some Text here!
.
```

Задания

- Получите список всех процессов и перенаправьте вывод в файл `ps.txt`.
- Выполните команду поиска всех обычных файлов в каталоге `/usr` так, чтобы найденные имена файлов были записаны в файл `SysComm` в домашнем каталоге, а поток ошибок был записан в ноль-устройство `/dev/null`.
- Выполните ту же команду, но так, чтобы потоки вывода и ошибок были записаны в файл `SysComm`.
- Допишите в конец файла `SysComm` информацию о текущей дате и времени, выводимую командой `date`.
- Установите опцию `noclobber` и сотрите содержимое файла `ps.txt` с помощью перенаправления вывода. Снимите опцию `noclobber`.

Конвейеры и фильтры

Конвейер (`pipe`) позволяет направить поток вывода одного процесса на поток ввода другого процесса. Для организации конвейера необходимо поставить символ конвейера — вертикальную черту (`|`) между командами, потоки которых необходимо объединить. Естественно, первой должна быть указана команда, поток вывода которой должен быть передан второй команде (пример 8.11).

Пример 8.11. Конвейер

```
$ last | view -
```

Выводимый командой `last` список входивших в сеанс пользователей передан через конвейер команде просмотра `view` (вызов `vi` на чтение, нажмите `:q` для выхода). Знак "дефис" после команды `view` сообщает, что текст должен быть получен из потока ввода, в который передает свой вывод команда `last`.

Конвейер может состоять из множества команд, причем команды в середине конвейера обязательно должны быть фильтрами. Фильтр — это команда,

способная принимать поток данных через стандартный поток ввода, обрабатывать их и выводить обработанные данные в стандартный поток вывода.

Правила построения конвейера таковы:

1. Первая команда конвейера должна выводить информацию в поток вывода.
2. Команды, не находящиеся по краям конвейера, должны быть фильтрами.
3. Последняя команда в конвейере должна читать стандартный поток ввода.

Пример фильтра — команда `tee`. Она никак не меняет полученные из стандартного потока ввода данные, а лишь копирует их в стандартный поток вывода и в файлы, заданные ей в качестве аргументов (пример 8.12).

Пример 8.12. Команда `tee`

```
$ ps -o comm= -o pid= | tee ps.txt | sort
bash          32642
ps            13882
sort          13884
tee           13883
$ cat ps.txt
ps            13882
tee           13883
sort          13884
bash          32642
```

В примере 8.12 команда `ps` выводит список процессов без заголовка в два столбца: команда и PID. Содержимое файла `ps.txt` содержит вывод команды `ps`, т. к. команда `tee` получила его из стандартного потока ввода. Эти же данные `tee` передала через поток вывода далее команде `sort` для сортировки. На экран выводится сортированный список процессов, а в файле `ps.txt` — он не отсортированный.

Команду `tee` часто используют для отладки работы сложных конвейеров, состоящих из множества команд. Эту команду удобно устанавливать в месте конвейера, которое вызывает подозрения. Так как в файле, указанном в качестве аргумента `tee`, будет находиться та же информация, что была передана в данном месте конвейера, то по ней легко можно будет определить наличие и суть ошибок.

Большинство фильтров предназначено для обработки исключительно текста, но это вовсе не обязательно. Команда `tee`, например, может обрабатывать и бинарные данные. При этом стоит позаботиться лишь о том, чтобы бинарные

данные не были выведены на экран, например, перенаправив поток вывода в `/dev/null`.

Задания

- Проведите поиск файлов символических ссылок в каталоге `/usr/share/doc` так, чтобы их список был выведен в отсортированном виде с помощью фильтра `sort`.
- Выведите отсортированный список пользователей, вошедших в сеанс, в системе в файл `seans.txt` и на экран одновременно.

Команда *echo*

Команда `echo` выводит на экран аргумент — текстовую строку (пример 8.13).

Пример 8.13. Команда *echo*

```
[user1@aida ~]$ echo "It's cool"
It's cool
[user1@aida ~]$ echo $PS1
[\u@\h \W]$
```

В первом случае (см. пример 8.13) команда `echo` просто выводит строку, заданную ей в качестве аргумента. Далее выводится значение переменной окружения `PS1`, для чего перед именем переменной поставлен символ извлечения значения переменной `$`.

Команда `echo` автоматически вставляет после выводимой строки символ перевода строки. Для отмены этого можно использовать опцию `-n`.

Важна опция `-e`, позволяющая команде `echo` интерпретировать восьмеричные числа как символы ASCII (пример 8.14).

Пример 8.14. Вывод ASCII-символов

```
$ echo -e '\0101\t\0102'
A      B
```

В этой команде (пример 8.14) выводятся буквы `A` и `B`, заданные их восьмеричными кодами ASCII, которые разделены символом табуляции. Шестнадцатеричные коды ASCII также могут быть указаны: `echo -e '\x66'`.

Очистить экран можно командой из примера 8.15.

Пример 8.15. Очистка экрана

```
$ echo -ne '\033c'
```

Задания

- Определите из `man ascii` восьмеричный код системного звукового сигнала и выведите его с помощью `echo`.
- Опция `-n` команды `echo` подавляет добавление перевода строки. С помощью `man`-системы определите, как это можно сделать иначе.

Просмотр файлов с помощью *more* и *less*

Команды `more` и `less` позволяют постранично просматривать текстовые файлы, которые можно задать в качестве аргументов или же передать в стандартном потоке ввода. Обобщающее название этих команд — *пейджеры* (pager). Первый из них исторически присутствует во множестве операционных систем. Пейджер `less` разработан позже, он более удобен, чем `more`, и используется в GNU/Linux по умолчанию.

Одно из наиболее удобных свойств `less` заключается в том, что в отличие от `more`, для прокрутки экрана в нем можно пользоваться обычными клавишами управления курсором `<↑>` и `<↓>`, а также клавишами `<PgUp>` и `<PgDn>`. Многие из команд управления прокруткой `more` и `vi` поддерживаются в `less`. В табл. 8.1 показано несколько наиболее часто используемых команд `more` и `less`.

Таблица 8.1. Команды *more* и *less*

Действие	<code>more</code>	<code>less</code>
Выход из пейджера	q	q
Получение встроенной помощи по командам	h	h
Прокрутка страницы вперед	<Пробел> <PgDn>	
Прокрутка страницы назад <Ctrl>+		<PgUp>
Прокрутка на строку вперед <Enter>		<Ctrl>+<N>
Прокрутка на строку назад		<Ctrl>+<P>
Переход в конец файла		G
Переход в начало файла		g
Поиск по шаблону (регулярному выражению)	/ шаблон	/ шаблон

Задания

- В чем разница между командами `less /etc/passwd` и `less < /etc/passwd`? Заметны ли отличия в командах при их выполнении?
- Команда `man` обладает опцией для явного указания пейджера, в котором будет отображаться найденная страница помощи. Опробуйте ее действие.
- Как можно запустить `man` так же, как и в предыдущем задании, но используя переменную окружения `PAGER`?
- Команда `less` позволяет открыть несколько файлов, указав их в качестве аргументов. Изучите эту возможность.

Объединение файлов с помощью `cat`

Команда `cat` выводит в стандартный поток вывода содержимое файла (не обязательно текстового). Если аргументами задано несколько файлов, то их содержимое выводится последовательно, объединяя в выводимом потоке содержимое этих файлов (название `cat` происходит от англ. *concatenate* — сцеплять). Команда `cat` может объединять и двоичные файлы. Не следует только выводить содержимое двоичных файлов на терминал, т. к. это может нарушить его настройки.

В примере 8.16 команда `split` (будет описана далее) делит бинарный файл `/bin/ls` на две части: `xaa` и `xab`, которые помещаются в текущий каталог. Далее команда `cat` из этих двух частей собирает файл `ls` в текущем каталоге. Файлы `ls` и `/bin/ls` имеют одинаковое содержимое, что проверяется командой `md5sum` (вычисляет хэш MD5). То есть при слиянии частей файла команда `cat` не изменила содержимое файла. Для проверки работоспособности копии программы `ls`, собранной из частей, на файл `ls` устанавливается право на исполнение для владельца. Последняя команда примера демонстрирует работоспособность исполняемого бинарного файла, собранного из частей.

Пример 8.16. Объединение частей бинарного файла командой `cat`

```
$ ls -l /bin/ls
-rwxr-xr-x 1 root root 100584 Oct 24 10:20 /bin/ls
$ split -b51k /bin/ls
$ ls -l x??
-rw-r--r-- 1 user1  users 52224 Jan  3 13:32 xaa
-rw-r--r-- 1 user1  users 48360 Jan  3 13:32 xab
$ cat x?? > ls
$ md5sum ls /bin/ls
```

```
768c9ee4991bad14bdd444d5a2cb6b5a  ls
768c9ee4991bad14bdd444d5a2cb6b5a  /bin/ls
$ chmod u+x ls
$ ./ls -l ls /bin/ls
-rwxr-xr-x 1 root  root  100584 Oct 24 10:20 /bin/ls
-rwxr--r-- 1 user1 users 100584 Jan  3 13:32 ls
```

У команды `cat` имеется похожая команда `tac`, позволяющая выводить строки файла, заданного в качестве аргумента, в обратном порядке.

Задания

- Команда `cat` обладает опцией, устанавливающей нумерацию выводимых строк. Определите, какая она.
- Найдите опцию `cat`, позволяющую этой команде удалять последовательно повторяющиеся переводы строки, оставляя лишь один перевод строки.
- Выведите содержимое файла `/etc/passwd` в обратном порядке следования строк.
- Для чего предназначена опция `-b` команды `tac`?

Команды *head* и *tail*

Команды `head` и `tail` выводят по умолчанию десять первых и последних строк соответственно потока или файла. Требуемое количество строк может быть указано.

Пример 8.17. Команда `head`

```
$ head -3 /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
```

В примере 8.17 выведены первые три строки файла `/etc/passwd`.

Команда `tail` предоставляет удобную опцию `-f`, которая позволяет выводить последние строки файла, отображая новые строки, появляющиеся в конце файла. Это бывает полезно, например, при мониторинге файлов журналов.

Задания

- Получите имена трех наиболее объемных файлов в каталоге `/usr/bin`.
- Выполните команду `tail -f /var/log/messages` и подключите к компьютеру флэш-накопитель с USB-разъемом. Выводится ли соответствующее сообщение?

Вырезание текста с помощью *cut*

Команда *cut* выводит только указанные в командной строке символы, байты или поля из строк файла, направляя выводимые данные в стандартный поток вывода. Если файл для чтения не задан, то осуществляется ввод из стандартного потока ввода.

Требуемые для вывода символы строки указывают после опции *-c* через запятую или тире (пример 8.18).

Пример 8.18. Команда *cut*

```
$ ls -ld
drwx----- 51 user1  user1      3752 Oct 22 21:04 .
$ ls -ld | cut -c1-10,35-43
drwx----- 3752
```

Пример 8.18 показывает, как из информации, выводимой командой *ls -ld*, были извлечены требуемые диапазоны символов так, что в результате осталась информация только о правах доступа к каталогу и о его размере.

Можно указать номера требуемых байтов в строке для вывода, используя опцию *-b*.

Если предполагается, что строка разбита на поля, разделенные табуляцией, то с помощью команды *cut -f* можно вывести требуемые поля строк. Например, если требуется вывести из файла */etc/hosts*, содержащего соответствия IP-адресов именам хостов, только заданные в нем IP-адреса без имен хостов, можно выполнить команду, приведенную в примере 8.19.

Пример 8.19. Вывод требуемых полей с помощью *cut*

```
$ cut -f1 /etc/hosts
127.0.0.1
172.16.0.25
```

Во многих файлах используются другие символы-разделители. Разделитель полей можно указать после опции *-d*. Предположим, требуется получить список пользователей системы и их UID. Это просто сделать путем вывода первого и третьего полей файла */etc/passwd*. Разделитель полей в этом файле — двоеточие (пример 8.20).

Пример 8.20. Установка символа разделителя полей для `cut`

```
$ cut -f1,3 -d: /etc/passwd
```

Задания

- Получите столбец из имен пользователей, находящийся в данный момент в сеансе.
- Получите столбец, составленный из символов строк — имен файлов в текущем каталоге таким образом, чтобы были выведены символы с пятого до последнего.

Потоковый редактор `sed`

Потоковые редакторы представляют собой фильтры, редактирующие проходящий через них поток текста на основе заранее заданных команд. Все изменения, вносимые в текст потоковыми редакторами, производятся в потоке и не затрагивают содержимое файла, откуда этот поток текста считан. Потоковые редакторы обладают встроенными языками программирования для написания сценариев обработки текста.

Потоковый редактор `sed` позволяет либо прочесть входной поток для обработки из стандартного потока ввода, либо осуществить чтение текстового файла, заданного в качестве аргумента. Обработанный текст передается в стандартный поток вывода. Редактор `sed` работает с целыми строками текста.

Перед командами `sed` нужно указывать адреса строк, с которыми производятся какие-либо действия. Адреса можно указывать следующими способами:

- номером строки;
- диапазоном строк с указанием первого и последнего номеров строк в диапазоне через запятую (например, команда `sed '2,4d'` удалит в потоке строки со второй по четвертую включительно). Можно указать диапазон до последней строки. Последняя строка обозначается символом доллара (\$);
- с помощью регулярного выражения или, в простейшем случае, подстроки, которая должна находиться в адресуемых строках.

В примере 8.21 приведена команда, которая выведет список процессов, не связанных с какими-либо терминалами (их имена содержат подстроку `tty`), но не псевдотерминалами (например, `pts/1`). Команда `sed` здесь удаляет из потока все строки, в которых имеется подстрока `tty`. Искомая подстрока (регулярное выражение), используемая для адресации удаляемых в потоке строк, должна быть задана в косых чертах.

Пример 8.21. Удаление строк по подстроке

```
$ ps -e | sed '/tty/d'
```

Эта команда (см. пример 8.21) выведет список процессов, либо не связанных с терминалами вообще, либо связанных с псевдотерминалами.

Наиболее часто используются следующие команды языка `sed`:

- ❑ `d` — удаление;
- ❑ `p` — вывод строки (от англ. *print*);
- ❑ `s` — замена искомой подстроки на заданную подстроку;
- ❑ `i` — вставка строки;
- ❑ `a` — добавление строки;
- ❑ `c` — изменение всей строки на заданную строку;
- ❑ `q` — выход без дальнейшей обработки и вывода строк;
- ❑ `y` — команда транслитерации символ в символ;
- ❑ `=` — команда печати номера строки.

Предположим, что требуется увидеть все строки учетных записей, начиная с первой и заканчивая строкой, в которой встречается строка `daemon` (пример 8.22).

Пример 8.22. Команда `q` редактора `sed`

```
$ sed /daemon/q /etc/passwd
root:x:0:0:System Administrator:/root:/bin/bash
bin:x:1:1:bin:/:/dev/null
daemon:x:2:2:daemon:/:/dev/null
```

Из примера 8.22 видно, что здесь была использована команда выхода `q`. Как только редактор `sed` встретил строку `daemon`, вывод был завершен.

Пример 8.23 демонстрирует, как `sed` применяется для замены подстрок. Необходимо в потоке, полученном из файла `/etc/passwd`, все вхождения `bash` заменить на `zsh`.

Пример 8.23. Замена строк с помощью `sed`

```
$ sed s/bash/zsh/ /etc/passwd
```

В примере 8.23 после команды `s` в косых чертах указана подстрока для поиска — `bash`. Далее указана подстрока замены — `zsh`.

В примере 8.24 приведен более сложный вариант замены подстрок в потоке. Требуется вывести содержание файла `/etc/group` (файл с информацией о группах пользователей и членстве в них) следующим образом:

- ☐ должны быть выведены только группы, включающие пользователя `user1`;
- ☐ все знаки двоеточия должны быть заменены символом подчеркивания.

Пример 8.24. Сложная команда `sed`

```
$ sed -n /user1/s/:/_/gp /etc/group
adm_x_4_root,adm,daemon,user1
wheel_x_10_root,user1
users_x_100_user1
```

В примере 8.24 во входном потоке выбираются только строки, содержащие подстроку `user1`. Команда `s` заменяет все вхождения символа двоеточия в каждой такой строке на подчеркивание. После косой черты установлен модификатор `g`. Именно он заставляет команду `s` заменить все вхождения искомым подстроки (двоеточия) на подстроку — заменитель (подчеркивание). Если бы он отсутствовал, то в каждой адресуемой строке был бы заменен только первый символ двоеточия. После модификатора `g` следует команда `p`. Эта команда печатает адресуемые строки, т. е. те, в которых здесь произведена замена. Использование команды `p` вынуждает добавить опцию `-n` команды `sed`. Без этой опции `sed` выводит в `stdout` все входные строки. При этом были бы выведены все строки потока с дублированием обработанных строк.

Пример 8.25 демонстрирует, как можно выделить требуемые строки в потоке, вставив перед ними заданную строку (например, строку-разделитель). Итак, требуется получить список всех последних входов в систему пользователя `user1` так, чтобы после каждой строки, где упоминается этот пользователь, выводилась строка решеток.

Пример 8.25. Добавление строк с помощью `sed`

```
$ last | sed '/user1/a \
#####'
```

В примере 8.25 конструкция `/user1/a` включает режим добавления ко всем строкам, в которых есть подстрока `user1`. Команда `a` заставляет `sed` добавить после каждой адресуемой строки содержимое нижестоящей строки в команде (здесь — решетки).

Пример 8.26 показывает файл сценария `sed`, выполняющего ту же задачу.

Пример 8.26. Сценарий `sed`

```
$ cat sed.scr
/user1/a \
#####
$ last | sed -f sed.scr
```

Содержимое файла `sed.scr` выведено командой `cat`. Последняя строка примера показывает, что для указания `sed` имени файла сценария надо задавать опцию `-f`.

ЗАДАНИЯ

- С помощью `sed` получите список только тех процессов, которые связаны с каким-либо терминалом (фильтр по строкам `pts` и `tty`).
- В полученном списке процессов с помощью `sed` замените все строки `user` на `provider`.

Потоковый редактор `awk`

Утилита `awk`, GNU-версия которой называется `gawk`, предназначена для потокового редактирования текста и обладает специализированным языком программирования для обработки табулированных данных. Она исходно разрабатывалась, как генератор отчетов.

Сценарий для `awk` в общем виде выглядит, как показано в примере 8.27.

Пример 8.27. Общий вид сценария `awk`

```
BEGIN
Команды формирования заголовка отчета
/шаблон/ { команды обработки строк }
END
Команды формирования завершения отчета
```

Сценарий `awk` может состоять из трех частей:

- ❑ команды формирования заголовка — следуют после `BEGIN`;
- ❑ команды основного цикла обработки строк заключаются в фигурные скобки;
- ❑ команды формирования завершения отчета — следуют после `END`.

Команды могут находиться в файле сценария, имя которого должно быть указано после опции `-f`, либо может быть задано в качестве аргумента `awk`. Если шаблон для команды не указан, то обработке подвергаются все строки. Если же, наоборот, шаблон задан, а команда — нет, то в `stdout` выводятся только строки, подходящие шаблону. Например, следующая команда выведет из файла `/etc/passwd` все строки, содержащие подстроку `bash` (пример 8.28).

Пример 8.28. Вывод строк по шаблону с помощью `awk`

```
$ awk /bash/ /etc/passwd
root:x:0:0:System Administrator:/root:/bin/bash
user1:x:500:100::/home/user1:/bin/bash
```

В этом случае не было команды, но был задан шаблон — строка `bash`.

Теперь выведем только имя пользователей, использующих `bash`, и их `UID`. Для этого необходимо вывести первое и третье поле этого файла. Утилита `awk` позволяет обращаться к полям строк, разделенных с помощью пробелов или табуляций, следующим образом: `$0` — вся строка, `$1` — первое поле, `$2` — второе и т. д.

В нашем случае разделителем полей является двоеточие, поэтому данный символ надо указать после опции `-F` (пример 8.29).

Пример 8.29. Вывод требуемых полей

```
$ awk -F: '/bash/{print $1,$3}' /etc/passwd
root 0
user1 500
```

Вывод требуемых полей обеспечивает команда `print`.

Для вывода только тех пользователей, чьи `UID` больше 100, можно отказаться от шаблона и профильтровать строки по условию `$3>100` (пример 8.30).

Пример 8.30. Фильтрация по числовому значению

```
$ awk -F: '$3>100{print $1,$3}' /etc/passwd
user1 500
```

Язык `awk` обладает собственными заранее определенными переменными и разрешает пользователю определять свои переменные. Одной из наиболее часто используемых встроенных переменных является `NR` — номер строки в потоке. Так, в предыдущем примере перед именами пользователей можно вывести номера строк, считанных из файла `/etc/passwd` (пример 8.31).

Пример 8.31. Переменная `NR` редактора `awk`

```
$ awk -F: '$3>100{print NR,$1,$3}' /etc/passwd
48 user1 500
```

Пример 8.32 демонстрирует, как можно получить список только тех файлов, длина имен которых больше 15 символов.

Пример 8.32. Фильтрация строк по длине

```
$ ls | awk 'length($0)>15'
```

Функция `length()` языка `awk` позволяет проверять длину строк. В данном случае фильтруются только те строки, длина которых больше 15 символов.

Задания

- Используя `awk`, из полученного списка процессов удалите второй столбец.
- С помощью `awk` пронумеруйте полученный в результате список.

Сравнение файлов и каталогов

Команда `diff` сравнивает текстовые файлы, указанные в качестве аргументов.

Пример 8.33. Поиск отличий в текстовых файлах с помощью `diff`

```
$ ps > ps1.txt
$ ps > ps2.txt
$ diff ps1.txt ps2.txt
3c3
```

```
< 3084 pts/0      00:00:00 ps
---
> 3088 pts/0      00:00:00 ps
```

В примере 8.33 список текущих процессов был выведен в два различных файла. Так как вызов команды `ps` производился дважды, то PID процессов у этих команд различался. Это и подтверждает вывод команды `diff`. Оба получившихся файла содержат по три строки. Последние, третьи строки отличаются (3с3).

Опция `q` отменяет вывод отличающихся строк и печатает только сообщение о том, имеются ли отличия. Опция `-i` применяется в случае, если необходимо игнорировать регистр в сравниваемых файлах.

Часто используемые опции `-u` и `-c` выводят информацию, соответственно, в обобщенном (unified) и контекстном (context) форматах (пример 8.34).

Пример 8.34. Опция `-u` команды `diff`

```
$ diff -u ps1.txt ps2.txt
--- ps1.txt      2009-11-20 23:18:16 +0500
+++ ps2.txt      2009-11-20 23:18:21 +0500
@@ -1,3 +1,3 @@
     PID TTY          TIME CMD
     2164 pts/0      00:00:00 bash
-   3084 pts/0      00:00:00 ps
+   3088 pts/0      00:00:00 ps
```

Особенностью этих форматов является вывод информации об именах сравниваемых файлов. Эта информация может быть использована для реконструкции одного из сравниваемых файлов по-другому с помощью известных отличий этих файлов. Для этого используется утилита `patch`, позволяющая обновлять содержимое файлов с помощью так называемых "заплаток" (patches). Файлы-"заплатки" создаются с помощью утилиты `diff`, а обновление файлов производится утилитой `patch`.

Для побайтного сравнения бинарных файлов удобно применять команду `cmp`. Если же необходимо просто выявить факт отличия файлов, пользуются командой проверки контрольной суммы `sum` или же командами вычисления хэш-функций `md5sum`, `sha256sum` или подобными.

Задания

- Определите, какая опция `diff` рекурсивно сравнивает каталоги.
- Введите команду, сравнивающую содержимое вашего домашнего каталога с домашним каталогом суперпользователя (от имени суперпользователя).
- Получите файл `patch.txt` с отличиями файлов `ps1.txt` и `ps2.txt` в контекстном формате. Удалите файл `ps2.txt` и восстановите его содержимое с помощью файлов `ps1.txt` и `patch.txt`.

Замена символов табуляции на пробелы

Команда `expand` предназначена для преобразования символов табуляции в пробелы. По умолчанию символ табуляции заменяется восемью пробелами, однако можно точно указать требуемое количество пробелов для замены табуляции. Для этого требуемое количество пробелов указывается как опция (пример 8.35).

Пример 8.35. Команда `expand`

```
$ echo -e 'Строка\tc\tтабуляцией'
Строка      c      табуляцией
$ echo -e 'Строка\tc\tтабуляцией' | expand -3
Строка      c      табуляцией
```

В примере 8.35 команда `echo` выводит на экран строку, в которой имеются символы табуляции. Команда `expand -3` заменяет каждый символ табуляции тремя пробелами, что заметно по изменению форматирования.

Можно точно указать список позиций табуляции, пользуясь опцией `-t`, после которой необходимо указать список позиций.

Имеется команда `unexpand`, заменяющая пробелы символами табуляции.

Задания

- Выведите содержимое файла `.bashrc`, заменив каждый символ табуляции двумя пробелами.
- Как заменить только первые в строке символы табуляции на пробелы?

Простое форматирование текста

Команда `fmt` предназначена для форматирования текста (пример 8.36). Команда объединяет и разделяет строки так, чтобы результирующая строка имела заданную длину (по умолчанию — 75 символов). Команда `fmt` пытается разделить длинные строки, начиная с длины строки, меньшей на 7%, чем установленная максимальная длина строки. Опция `-w` позволяет указать ширину текста в символах.

Если необходимо только разделять строки, но не объединять их, то следует использовать опцию `-s`.

Пример 8.36. Команда `fmt`

```
$ ls -l /home | fmt -w50 -s
total 24
drwx-----  47 user1    users      4096 Nov
24 23:22 user1
drwx-----   6 colobok  colobok    4096 Nov
21 19:07 colobok
```

ЗАДАНИЕ

Получите отформатированный командой `fmt` список файлов в каталоге `/usr/share/doc`.

Подготовка текста к печати

Команда `pr` готовит к печати файл, выводя его содержимое в стандартный поток вывода (`stdout`). При подготовке к печати текст форматируется в соответствии с параметрами страницы, и добавляются страничные заголовки.

После символа `+` можно указать номер страницы, с которой необходимо начинать печать. Например, команда `pr +3` выведет в стандартный поток вывода подготовленный к печати текст файла, начиная с третьей страницы.

Количество колонок, на которые необходимо разбить текст, указывается в качестве опции: `pr -2` — данная команда форматирует вывод текста для печати в две колонки на странице. Команда `pr` с опцией `-m` форматирует текст файлов, указанных в командной строке, для печати в разных колонках содержимого разных файлов. При этом не надо указывать количество колонок, т. к. оно определяется автоматически. Опция `-b` позволяет балансировать

вывод колонок на последней странице, т. е. обеспечивать приблизительно равное количество строк в них.

После опции `-h` можно указать строку заголовка для печати на страницах.

Далее приведены другие часто использующиеся опции команды `pr`:

- ☐ опция `-a` устанавливает режим печати поперечных колонок;
- ☐ замена табуляции пробелами — опция `-e`;
- ☐ замена пробелов табуляцией — опция `-i`;
- ☐ печать управляющих символов в виде символического кода — опция `-c`;
- ☐ опция `-v` задает режим вывода восьмеричных кодов непечатаемых символов;
- ☐ опция `-t` отменяет печать заголовков;
- ☐ опция `-l` устанавливает длину страницы;
- ☐ опция `-w` устанавливает ширину страницы;
- ☐ опция `-o` устанавливает величину левого отступа;
- ☐ опция `-f` подавляет вставку прогонов страниц;
- ☐ вывод номеров строк — опция `-n`.

Задания

- Выведите отформатированное в два столбца содержимое текущего каталога с заголовком, в котором будет указано имя текущего каталога.
- Сделайте то же самое, но без вывода заголовка страницы и с выводом последовательного номера перед каждым именем файла.

Сортировка строк

Команда `sort` предназначена для сортировки строк файлов — аргументов. Если файлы не указаны, команда сортирует строки из стандартного потока ввода. По умолчанию команда `sort` сортирует строки в алфавитном порядке по возрастанию. При необходимости сортировки строк в порядке убывания используется опция `-r`.

Команда `sort` способна также сортировать текстовый поток не только по целым строкам, но и по отдельным полям строк. Разделителем полей по умолчанию считается пробел. Если используется иной разделитель полей, его следует указать после опции `-t`. Для указания номера поля для сортировки используется опция `-k`.

Опция `-n` позволяет задать команде `sort` не алфавитный, а числовой порядок сортировки. Пример 8.37 показывает, как отсортировать учетные записи пользователей, в порядке возрастания их `UID`.

Пример 8.37. Сортировка по заданному полю

```
$ sort -t: -k3 -n /etc/passwd
```

В файле `/etc/passwd` разделителем полей является двоеточие. Это установлено опцией `-t`. Сортировка была выполнена по третьему полю (опция `-k3`), в котором содержатся UID пользователей. Сортировка была проведена в числовом порядке, что было установлено опцией `-n`.

Задания

- Получите список групп пользователей в системе, отсортированный по GID в обратном числовом порядке.
- С помощью утилит `find`, `head` и `sort` получите список из десяти файлов в домашнем каталоге, занимающих наибольшее дисковое пространство.

Вывод неповторяющихся строк

Фильтр `uniq` удаляет повторения строки в сортированном потоке. Команда в примере 8.38 выведет строки файлов `f1` и `f2` так, что если какая-либо строка имеется в обоих файлах, то в поток вывода попадет только одна ее копия.

Пример 8.38. Команда `uniq`

```
cat f1 f2 | sort | uniq
```

Опция `-c` команды `uniq` позволяет подсчитать количество вхождений каждой строки во входном потоке. Опция `-d` позволяет вывести только дублирующиеся строки, что помогает, например, узнать, какие строки имеются одновременно в разных файлах. Опция `-u` выводит только уникальные (не дублированные) строки. Для игнорирования регистра при сравнении строк можно установить опцию `-i`.

Если строки входного потока разделены на поля, то можно пропустить заранее заданное количество полей до определения уникальности строки. Для пропуска первых полей, разделенных пробелами, необходимо указать их количество после опции `-f`. Например, команда `uniq -f6` пропустит шесть первых полей входного потока при определении уникальности строк.

Задание

Выведите список тех файлов в домашнем каталоге, первые три символа в именах которых совпадают не менее, чем у двух файлов.

Объединение строк двух файлов по общему полю

Команда `join` построчно объединяет содержимое заранее отсортированных файлов по общему полю. Выходная информация направляется в стандартный поток вывода. По умолчанию предполагается, что поля отделяются друг от друга пробелами или табуляцией. Опция `-t` позволяет указать требуемый символ-разделитель полей.

Объединение производится по первым полям строк файлов. Если необходимо произвести объединение строк файлов не по первым полям строк, то номера этих полей необходимо указать после опций `-j1` — для первого файла и `-j2` — для второго файла. Содержимое файлов должно быть отсортировано этим полям.

Для того чтобы вывести вместе с объединенными строками непарные строки, которые обычно не выводятся, необходимо использовать опцию `-a`. После этой опции следует указать номер файла, из которого будут выведены непарные строки. Наоборот, опция `-v` позволяет вместо объединенных строк вывести только непарные строки того файла, номер которого указан после этой опции.

Задания

- Имеются два каталога без подкаталогов. Как с помощью `join` получить список файлов, которые имеются в обоих каталогах?
- Как сделать то же самое для файлов, которые имеются только в одном из каталогов и отсутствуют в другом?
- Как получить список файлов кроме тех, которые имеются в обоих каталогах?

Подсчет количества и нумерация строк

Команда `wc` позволяет подсчитывать количество символов, слов и строк в файле, указанном в качестве аргумента (пример 8.39) или в стандартном потоке ввода.

Пример 8.39. Команда `wc`

```
$ wc /etc/hosts
      3          8      92 /etc/hosts
```

В этом примере подсчитано количество строк, слов и символов в файле `/etc/hosts`.

При необходимости можно установить вывод только числа:

- ☐ строк — с использованием опции `-l`;
- ☐ слов — при установленной опции `-w`;
- ☐ символов — при использовании опции `-c`.

Например, для определения количества пользователей, работающих в настоящее время в сеансе, можно использовать команду из примера 8.40.

Пример 8.40. Подсчет количества пользователей в сеансе

```
$ who | wc -l
1
```

Команда `nl` позволяет пронумеровать строки в тексте, считанном из файла или из потока ввода. Команда `cat -b` также нумерует строки, однако возможности команды `nl` намного шире. Опции `nl` позволяют устанавливать формат нумерации строк, особым образом нумеровать пустые строки, устанавливать шаг нумерации и т. п.

Задания

- Определите, используя файл `/etc/passwd`, содержащий данные об учетных записях пользователей, сколько пользователей зарегистрировано в системе.
- Определите, для скольких пользователей оболочка по умолчанию — `bash`.
- Сколько имеется пользователей, UID которых больше 100?
- Выведите пронумерованный список файлов в текущем каталоге.
- Сделайте то же, но с шагом нумерации, равным двум.

Замена символов с помощью команды *tr*

Команда `tr` читает поток ввода и преобразует его (пример 8.41). Она позволяет выполнять преобразования, приведенные далее в списке.

- ☐ Замена символов в потоке. Задают два набора символов. Символы из первого набора заменяются соответствующими по порядку символами из второго набора.
- ☐ При использовании опции `-d` удаляются символы, указанные в наборе.
- ☐ Исключение повторения символов в потоке. Символы, повторы которых должны быть исключены, задаются в наборе — опция `-s`.

- ❑ Замена символов в потоке с последующим устранением их повторов. При этом задаются два набора символов и устанавливается опция `-s`. Вначале команда `tr` заменяет символы, а затем устраняет повторения символов из второго набора.
- ❑ Удаление символов из потока с последующим устранением повторов. Этот режим требует установки опций `-d` и `-s`. Вначале удаляются символы из первого набора, а затем устраняются повторы из второго.

Пример 8.41. Замена символов с помощью `tr`

```
$ echo tarelka | tr a-z A-Z  
TARELKA
```

Здесь были заданы два набора символов — все буквы английского алфавита в нижнем регистре, которые заменяются буквами в верхнем регистре.

Удалить символы перевода строки можно командой, показанной в примере 8.42.

Пример 8.42. Удаление символов

```
$ ls / | tr -d '\n'  
binbootdevetchomelibmntoptprocrootssbinswaptmpusrvar
```

Здесь в качестве входного потока был использован вывод команды `ls /`, из которого были удалены все символы перевода строк.

Устранение повторов выполняет опция `-s` (пример 8.43).

Пример 8.43. Устранение повторов

```
$ echo root | tr -s o  
rot
```

В этом примере были устранены повторения символа `o`.

Опция `-c` команды `tr` позволяет инвертировать смысл задаваемого множества символов, т. е. удалить при использовании `-d` все, кроме символов, указанных в наборе. Например, команда `tr -dc 0-9` удалит во входном потоке все, кроме цифр.

С опцией `-t` команда `tr` обрезает длину первого набора по длине второго, для того чтобы количество символов в них равнялось.

Можно указать класс символов из набора predefined символов (табл. 8.2).

Таблица 8.2. *Предопределенные шаблоны `tr`*

Класс	Символы
<code>[:alnum:]</code>	Символы алфавита в любом регистре и цифры
<code>[:alpha:]</code>	Символы алфавита в любом регистре
<code>[:blank:]</code>	Пустое множество
<code>[:cntrl:]</code>	Управляющие символы
<code>[:digit:]</code>	Десятичные цифры
<code>[:graph:]</code>	Все символы, которые могут быть напечатаны, кроме пробела
<code>[:lower:]</code>	Алфавитные символы в нижнем регистре
<code>[:print:]</code>	Все символы, которые могут быть напечатаны
<code>[:punct:]</code>	Все символы пунктуации
<code>[:space:]</code>	Пробел или табуляция
<code>[:upper:]</code>	Алфавитные символы в верхнем регистре
<code>[:xdigit:]</code>	Шестнадцатеричные цифры

В примере 8.44 символы в нижнем регистре заменены символами в верхнем регистре.

Пример 8.44. Использование предопределенных шаблонов

```
$echo "str" | tr [:lower:] [:upper:]
STR
```

ЗАДАНИЯ

- Выведите последние три строки файла `/etc/passwd`, заменив разделители-двоеточия разделителями-пробелами.
- Подсчитайте с помощью команды `wc`, сколько в получившемся потоке пробелов.
- Сколько символов — десятичных цифр содержится в файле `/etc/sysctl.conf`?

Слияние строк

При необходимости вывести в одну строку содержимое файла можно использовать команду `paste -s`. Так, приведенная в примере 8.45 команда выведет в виде одной строки содержимое файла `/etc/hosts`.

Пример 8.45. Команда `paste`

```
$ paste -s /etc/hosts
192.168.1.1 note 127.0.0.1 localhost.localdomain localhost
```

Без опции `-s` команда `paste` выводит строки файлов-аргументов параллельно, т. е. первая строка первого файла вместе с первой строкой второго файла и т. д.

Если `paste` должна читать из стандартного потока ввода, то вместо имени файла указывают символ тире. Колонок вывода будет столько, сколько тире (пример 8.46).

Пример 8.46. Параллельный вывод строк

```
$ ls / | paste - - - - - - -
bin    boot  dev   etc    home   lib     lost+found  media
mnt    opt   proc  root   sbin   selinux  srv         sys
tmp    usr   var   windows
```

В примере 8.46 вывод содержимого корневого каталога производится в восемь столбцов, т. к. задано восемь тире.

Задания

- Получите список всех PID процессов, связанных с терминалами, так, чтобы весь список был выведен в одну строку, а после каждого PID процесса был указан соответствующий ему терминал. *Подсказка:* выделите нужные поля с помощью `awk` из списка всех процессов в системе.
- Получите пронумерованный список всех пользователей в системе так, чтобы он выводился в виде одной строки. Для номера должно отводиться три позиции.
- Измените предыдущую команду, добавив в нее `sed` и `tr`, так, чтобы перед каждым номером строки и после самих строк были вставлены переводы строки.

Получение дампа

Команда `od` позволяет получить восьмеричный, десятичный или шестнадцатеричный дамп потока или файла. В первом столбце команда `od` выводит смещение в потоке (адрес). Далее выводятся значения считанных из потока символов.

Например, для получения восьмеричного дампа строки `ABC` может быть использована команда, показанная в примере 8.47.

Пример 8.47. Восьмеричный дамп

```
$ echo ABC | od -tOc
0000000 101 102 103 012
0000004
```

Опции `-tOc` команды `od` задают вывод дампа в восьмеричном формате, а опция `-C` указывает, что входной поток должен быть интерпретирован как символичный.

Бывает удобно отобразить дамп не в виде численных значений, а с помощью "именованных символов". Для этого используются опции `-ta` (пример 8.48).

Пример 8.48. Дамп с выводом именованных символов

```
$ echo ABC | od -ta
0000000  A  B  C  nl
0000004
```

В таком формате печатаемые символы отображаются как обычно, а для непечатаемых символов используются символьные обозначения.

Можно установить формат для вывода смещения. Для этого необходимо воспользоваться опцией `-A`, указав далее базис, где:

- ☐ `-Ad` — десятичное знаковое целое;
- ☐ `-Au` — десятичное беззнаковое целое;
- ☐ `-Ao` — восьмеричное целое;
- ☐ `-Ax` — шестнадцатеричное целое;
- ☐ `-An` — подавить вывод адреса смещения.

Аналогичные модификаторы можно использовать после опции `-t` для получения дампа в различных форматах.

Помимо указания базиса вывода дампа, часто необходимо указывать формат интерпретации последовательности байтов, чем, фактически, задается формат чтения данных из памяти. Для всех целых базисов можно указать следующие форматы:

- -c — символьный (char);
- -s — короткое целое (short);
- -i — целое (int);
- -l — длинное целое (long).

Помимо `od` для получения дампов можно использовать `hexdump`.

Задания

- Убедитесь с помощью `man ascii`, что отображаемый командой в примере 8.47 дамп соответствует символам ABC, а также найдите, какому символу соответствует восьмеричный код 012.
- Получите восьмеричный дамп файла `/etc/hosts`.
- Получите шестнадцатеричный дамп первых двух байтов файла `/bin/ls`. Смещение должно быть указано в десятичном формате. Поток должен быть интерпретирован как набор коротких целых (short).
- С какого по порядку байта дампы файлов `/bin/ls` и `/bin/pwd` отличаются?

Разделение файлов на части

Команды `split` и `csplit` не являются текстовыми фильтрами. Они предназначены для разделения файлов на части, которые записываются в другие файлы со специальными именами. Эти файлы образуются в текущем каталоге.

Команда `split` разделяет содержимое файла на части, которые записываются в отдельные файлы. По умолчанию команда записывает в отдельный файл каждые 1000 строк текста. Имена файлов начинаются со строки префикса и дополняются символами нумерации, начиная с `aa`. Если префикс не указан, то используется префикс по умолчанию — символ `x`, а имена файлов, в которые будут записаны части разделенного файла, будут иметь вид: `хаа`, `хаб`, `хас`, ...

Опция `-l` команды `split` позволяет указать, сколько строк должно содержаться в частях файла (пример 8.49).

Пример 8.49. Команда `split`

```
$ split -l9 /etc/passwd
$ ls x*
хаа хаб
```

В примере 8.49 содержимое файла `/etc/passwd` разделено так, чтобы в частях файла содержалось по четыре строки (кроме последнего файла). В результате выполнения команды исходный файл разделен на два файла — `haa` и `hab`.

Опция `-b` команды `split` позволяет указать, сколько байтов должно содержаться в частях файла. Так можно разделять на части бинарные файлы. Команда `split -b5k` разделит файл на части, содержащие по 5 Кбайт кроме, возможно, последней части.

Опция `-cn` позволяет записывать в выходные файлы столько целых строк исходного файла, сколько можно записать, так, чтобы размер выходных файлов не превысил `n` байт. Например, команда `split -c1024` разделит файл на части, не превышающие по размеру 1 Кбайт и содержащие целые строки исходного текста.

Команда `csplit` позволяет разделить текстовый файл на часть до строки, содержащей образец — регулярное выражение, и часть, начиная со строки, содержащей его. Например, команда `csplit /etc/hosts /et/` разделит файл на две части. Первая часть будет содержать все строки до строки, содержащей `et`, а вторая — все остальные строки. Команда `csplit` именует части исходного файла префиксом `xx` с нумерацией.

Следует отметить, что команда `csplit` по умолчанию делит исходный файл на две части при удачном поиске, т. е. прекращает работу сразу после первого вхождения образца. Во второй части исходного файла могут содержаться вхождения образца, по которым исходный файл может быть разделен далее.

Можно указать число повторов поиска образца в фигурных скобках. Например, команда `csplit file /etc/ {1}` повторит поиск регулярного образца один раз. В случае если повторный поиск неудачен, команда `csplit` завершает свое выполнение с выводом сообщения об ошибке, а все части исходного файла стираются. Опция `-k` отменяет удаление выходных файлов в случае ошибки повторного поиска по регулярному выражению.

Задания

- Разделите файл `/etc/passwd` на части по 10 строк, находящиеся в текущем каталоге. Имена файлов должны начинаться с `passwd`.
- Разделите файл `/etc/passwd` на две части по первому вхождению строки `uusr`.
- Как заставить команду `csplit` продолжить разделение исходного файла на части до тех пор, пока это возможно без использования опции `-k`?

Команда *xargs*

Команда *xargs* использует данные, передаваемые ей из стандартного потока ввода, в качестве аргументов для конструирования команды. Наиболее часто она используется с командой *find*. Например, обе приведенные далее команды делают одно и то же — ищут и удаляют core-файлы, остающиеся в системе после программных сбоев (пример 8.50).

Пример 8.50. Команда *xargs*

```
$ find /usr -type f -name "core.*" -exec rm -f {} \;  
$ find /usr -type f -name "core.*" | xargs rm -f
```

В первом случае обработка найденных файлов производится командой *find*, которая вызывает команду *rm*. Во втором случае имена найденных файлов отправляются через конвейер команде *rm*. Второй вариант обычно работает быстрее. Но если в командную строку подставляется слишком много аргументов, это может привести к ошибке *bash*.

ЗАДАНИЯ

- В домашнем каталоге найдите все созданные в течение последних трех дней нескрытые пустые каталоги и удалите их, используя команду *xargs*.
- Какая команда позволяет сравнить быстродействие обоих вариантов команды *find*, приведенной выше?

Глава 9



Регулярные выражения

Регулярные выражения — это мощнейшее средство поиска текста. Очень часто рутинные задачи администрирования связаны с поиском строк текста, например, в файлах системных журналов. Поэтому умение работать с регулярными выражениями важно для эффективной работы администратора. В этой главе вы изучите базовые и расширенные регулярные выражения и освоите важнейшие утилиты для работы с ними.

Классификация регулярных выражений

Регулярные выражения — это специальные шаблоны, используемые для указания строк, например, при поиске. Регулярные выражения строятся из обычных алфавитно-цифровых символов и метасимволов. Метасимволы позволяют адресоваться одновременно к одному или более обычным алфавитно-цифровым символам.

По историческим причинам регулярные выражения подразделяются на:

- ☐ обычные регулярные выражения (basic regexp);
- ☐ регулярные выражения с расширенным синтаксисом (extended regexp).

Не все текстовые утилиты, работающие с регулярными выражениями, поддерживают расширенный синтаксис регулярных выражений.

Метасимволы, составляющие регулярные выражения, представлены двумя классами: шаблонами и квантификаторами.

Шаблоны — специальные символы, заменяющие один или более обычных символов.

Квантификаторы — указатели количества вхождений символа или набора символов, находящихся в регулярном выражении непосредственно перед ними.

То есть шаблоны указывают на то, что должно находиться в данном месте искомой строки, а квантификаторы — сколько раз оно должно там встречаться.

В табл. 9.1 приведены наиболее общеупотребительные регулярные выражения.

Таблица 9.1. Регулярные выражения

Метасимвол	Класс	Расширенный	Описание
^	Шаблон		Начало строки
\$	Шаблон		Конец строки
\<	Шаблон *		Начало слова
\>	Шаблон *		Конец слова
.	Шаблон		Любой символ, включая пробел
[]	Шаблон		Набор символов
()	Шаблон *		Группировка символов
	Шаблон *		Инфиксный оператор (или)
*	Квантификатор		Вхождение любое количество раз
+	Квантификатор *		Вхождение не менее одного раза
?	Квантификатор *		Вхождение не более одного раза
{n}	Квантификатор *		Вхождение n раз
{n, }	Квантификатор *		Вхождение не менее n раз
{n, m}	Квантификатор *		Вхождение от n до m раз

Помимо перечисленных регулярных выражений имеется несколько расширенных их наборов. Например, в языке Perl используется больше регулярных выражений.

Множества символов, задаваемые в квадратных скобках, имеют следующий смысл: в данном месте должен находиться один любой символ из входящих во множество. Например, множество [0-3] включает в себя все числа от нуля до трех, а регулярному выражению ^[0-3]\$ удовлетворяют все строки, в которых находится единственный символ — число от нуля до трех (^ — начало строки, \$ — конец строки).

Помимо множеств, заданных с помощью перечисления символов в квадратных скобках, существуют заранее определенные классы символов, приведенные в табл. 9.2.

Таблица 9.2. Заранее определенные множества символов

Класс	Значение
[:alnum:]	Множество алфавитно-цифровых символов
[:alpha:]	Алфавитные символы
[:blank:]	Пустые символы — пробел и табуляция
[:cntrl:]	Символы с восьмеричными кодами от 000 до 037 и 177
[:digit:]	Десятичные цифры
[:graph:]	Изображаемые символы: алфавитно-цифровые и пунктуация
[:lower:]	Буквы в нижнем регистре
[:print:]	Печатаемые символы: алфавитно-цифровые, пробел и пунктуация
[:space:]	Табуляция, вертикальная табуляция, пробел, перевод строки
[:upper:]	Буквы в верхнем регистре
[:xdigit:]	Шестнадцатеричные символы

Задания

- Составьте регулярное выражение для пустой строки.
- Как выявить в текстовом файле все строки, содержащие символ с восьмеричным кодом 007?
- Какой класс можно использовать для этого?

Поиск текста с помощью *grep*

Команда `grep` производит поиск в указанных файлах строк по регулярному выражению. Команда выводит в случае удачного поиска имя файла и строку, удовлетворяющую заданному регулярному выражению. Если входные файлы не заданы, то команда производит ввод из стандартного потока ввода. При этом используются три разновидности команды `grep` (далее идет речь о GNU-версии команды `grep`):

- ☐ `grep` — интерпретирует регулярные выражения с обычным (basic) синтаксисом;
- ☐ `grep -F` или `fgrep` — не интерпретирует регулярные выражения, воспринимая их как обычные текстовые строки;
- ☐ `grep -E` или `egrep` — позволяет работать с расширенным синтаксисом регулярных выражений.

Команды `grep`, `egrep` и `fgrep`, разработанные в проекте GNU, обычно реализованы в виде жестких связей (или символических ссылок). То есть все эти три команды являются одним файлом. В других разновидностях UNIX-подобных систем это не так. Приведенные далее примеры не всегда будут работать так же в других операционных системах, например, в Sun Solaris или FreeBSD.

Для поиска всех пользователей системы, использующих оболочку `bash`, достаточно выполнить команду `grep` с обычным регулярным выражением (пример 9.1).

Пример 9.1. Поиск с помощью обычного регулярного выражения

```
$ grep 'bash$' /etc/passwd
root:x:0:0:System Administrator:/root:/bin/bash
tania:x:502:502::/home/tania:/bin/bash
figus:x:503:503::/home/figus:/bin/bash
user1:x:504:100::/home/user1:/bin/bash
```

Регулярное выражение взято в апострофы для предотвращения интерпретации символа доллара оболочкой (пример 9.1). Доллар здесь обозначает конец строки.

Теперь, например, требуется выделить из всех файлов в каталоге `/etc/sysconfig/` все строки, содержащие IP-адреса в стандарте IPv4. Такую задачу удобнее всего решать с помощью расширенного синтаксиса регулярных выражений, т. е. используя команду `egrep` (пример 9.2).

Пример 9.2. Поиск с помощью расширенных регулярных выражений

```
$ egrep '[0-9]{1,3}(\.[0-9]{1,3}){3}' /etc/sysconfig/*
```

В этой команде обеспечивается поиск последовательности из одной, двух или трех любых десятичных цифр, которая должна быть троекратно продолжена такой же последовательностью. Причем каждая последовательность отделена от предыдущей последовательности точкой. Командой `fgrep` удобно пользоваться для поиска строк, содержащих символы, имеющие особое значение, как регулярные выражения. Так, например, можно получить список всех процессов, не связанных с какими-либо терминалами (пример 9.3).

Пример 9.3. Использование fgrep

```
$ ps -e | fgrep '?'
```

Команда `fgrep` воспринимает знак вопроса просто как обычный символ, который должен быть найден в строке.

Опция `-i` позволяет производить поиск без учета регистра. Приведенный далее пример 9.4 демонстрирует, как с помощью поиска с игнорированием регистра найти все процессы в системе, где в командной строке есть подстрока `x` или `x.`

Пример 9.4. Поиск с игнорированием регистра

```
$ ps -e | grep -i x
1546 ?          00:00:01 xfs
1599 ?          00:00:19 x
1752 ?          00:00:00 xvt
```

Если требуется подсчитать число вхождений регулярного выражения, то для этого используется опция `-c` (пример 9.5).

Пример 9.5. Подсчет вхождений регулярного выражения

```
$ egrep -c '^.{1,3}:' /etc/passwd
7
```

В приведенном примере 9.5 подсчитано, сколько имеется пользователей, длины имен которых не превышают три символа.

Опция `-v` команды `grep` инвертирует алгоритм поиска: команда начинает искать строки, не удовлетворяющие регулярному выражению. Так, предположим, что требуется найти список всех групп из файла `/etc/group`, в которые не входит пользователь `user1` (пример 9.6).

Пример 9.6. Инверсия условия поиска

```
$ grep -v user1 /etc/group
```

Если в регулярном выражении не указаны какие-либо ограничители, связанные с началом или концом слова, то команда `grep` производит поиск подстроки, а не слова. В команде `egrep` начало и конец слова можно обозначить

регулярными выражениями, соответственно, `\<` и `\>`. Например, в файле `/etc/sysctl.conf` для поиска всех строк, в которых встречается слово `Enable`, но не `Enables`, можно использовать команду, приведенную в примере 9.7.

Пример 9.7. Поиск слова

```
$ egrep '\<Enable\>' /etc/sysctl.conf
# Enable the magic-sysrq key
# Enable tcp_syncookies
```

Однако в таком случае гораздо удобнее воспользоваться опцией `-w` команды `grep`, которая устанавливает режим поиска по целому слову (пример 9.8).

Пример 9.8. Поиск слова с помощью опции `-w`

```
$ grep -w Enable /etc/sysctl.conf
# Enable the magic-sysrq key
# Enable tcp_syncookies
```

При необходимости можно также установить режим поиска по целой строке, для чего предназначена опция `-x`. Приведенная в примере 9.9 команда позволит отыскать в каталоге `/etc` все файлы, которые содержат строки, состоящие из одного символа.

Пример 9.9. Поиск целой строки с помощью опции `-x`

```
$ grep -x '.' /etc/*
```

Использование опции `-n` позволяет установить режим вывода номеров строк, в которых найдены искомые строки. Так, например, требуется получить строки файла `/etc/hosts`, содержащие подстроки `local`, и номера этих строк (пример 9.10).

Пример 9.10. Отображение номеров строк

```
$ grep -n local /etc/hosts
1:127.0.0.1          localhost.localdomain localhost
```

Здесь в первом поле вывода перед двоеточием указан номер строки, в которой найдена искомая подстрока (см. пример 9.10).

Опция `-r` заставляет команду `grep` работать рекурсивно, обрабатывая файлы в подкаталогах. Например, в каталоге `/usr/share/doc/HOWTO/` требуется найти все файлы, содержащие строку `trainer` (пример 9.11).

Пример 9.11. Рекурсивный поиск

```
$ grep -r trainer /usr/share/doc/HOWTO/
```

Задания

- Найдите опцию, позволяющую `grep` выводить имена файлов перед найденными строками.
- С помощью `grep` получите список всех пользователей системы, с $10 < GID < 100$.
- Отфильтруйте все сообщения, находящиеся в файле `/var/log/messages`, так, чтобы были выведены сообщения о событиях, происшедших с 8:30 до 12:30.
- Найдите в каталоге `/etc/rc.d` и его подкаталогах все файлы, содержащие слово `fsck`. В выводе должны присутствовать имена найденных файлов.
- С помощью команд `ps`, `grep`, `sort`, `uniq` подсчитайте, сколько процессов в настоящий момент связано с каждым виртуальным терминалом (`tty`) в системе.

Использование обратных ссылок

Существует специальная конструкция, позволяющая в регулярном выражении обращаться к уже найденной с помощью символов группировки подстроке. Она называется *обратной ссылкой* (back reference) и записывается так: `\1`, что значит повтор строки, удовлетворившей регулярному выражению — группировке, которое было указано в регулярном выражении до вхождения символа обратной ссылки. Например, требуется получить список всех пользователей системы, у которых имя (первое поле `/etc/passwd`) совпадает с именем исполняемого файла оболочки (последнее поле `/etc/passwd`). Такую задачу можно решить, используя обратную ссылку (пример 9.12).

Пример 9.12. Поиск с обратной ссылкой с помощью `egrep`

```
$ egrep '^(.+):.*\1$' /etc/passwd
sync:x:5:0:sync:/:/bin/sync
shutdown:x:6:0:shutdown:/:/sbin/shutdown
halt:x:7:0:halt:/:/sbin/halt
```

В этом регулярном выражении шаблоном для имени пользователя является `^(.+):`, т. е. строка из ненулевого количества любых символов. При этом специально использован символ группирования, т. к. с помощью него записывается строка символов, соответствующая имени пользователя. Далее эта же найденная строка с помощью обратной ссылки `\1` сравнивается с именем файла оболочки.

Вообще говоря, приведенный пример 9.12 не будет работать в других операционных системах, если в них не используется GNU-версия `grep`. Это связано с концептуальной невозможностью использования обратных ссылок в программах поиска подстрок, реализующих расширенный синтаксис регулярных выражений. GNU-версия `egrep` (он же `grep`) обеспечивает возможность поиска строк с использованием обратных ссылок.

Далее приведен более корректный пример 9.13, который не использует `egrep`.

Пример 9.13. Поиск с обратной ссылкой с помощью `grep`

```
$ grep '^(..*\):.*\1$' /etc/passwd
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
```

В регулярном выражении может быть задано более одного символа обратной ссылки. Если используется более одной обратной ссылки, то должно быть указано несколько символов группировки, а номера у обратных ссылок должны соответствовать порядковым номерам вхождений символов группировки в регулярном выражении.

Например, требуется получить список всех IP-адресов и имен хостов из файла `/etc/hosts` (который связывает IP-адрес и имя компьютера в сети) только для тех хостов, чьи IP-адреса начинаются либо со 127, либо со 192 (пример 9.14). Кроме того, у этих хостов должны быть одновременно указаны полностью определенное доменное имя (Fully Qualified Domain Name, FQDN) и короткое имя (или наоборот — порядок не важен).

Пример 9.14. Использование нескольких обратных ссылок

```
$ egrep '^(192|127)\..*([[:alnum:]]+)\..*\2$' /etc/hosts
127.0.0.1                localhost.localdomain localhost
```

В примере 9.14 первый символ группирования предназначен для организации перечисления (инфиксного оператора ИЛИ), а второй — для организации поиска по обратной ссылке. Так как искомые имена компьютеров могут быть определены по второму символу группирования, то используется оператор обратной ссылки \2.

Задания

- Получите список всех учетных записей пользователей вашей системы, у которых имя домашнего каталога (последнего каталога в пути) совпадает с именем пользователя.
- Проверьте, имеются ли в файле /var/log/messages (файл системного журнала) сообщения, у которых во времени совпадают часы и минуты (например, 09:09).

Использование регулярных выражений с *sed*

Регулярные выражения позволяют адресоваться к строкам, обрабатываемым *sed*. Регулярное выражение указывается в косых чертах. Так, например, для вставки строки-разделителя после учетных записей, использующих в качестве оболочки *bash*, с предварительным удалением строк учетных записей пользователей, использующих оболочки, имена которых не заканчиваются на *sh*, можно задать команду, приведенную в примере 9.15.

Пример 9.15. Регулярные выражения в *sed*

```
$ sed -e '/[^s][^h]$/d' -e '/bash/a \
@@@@@@@@@@@@@@@@@@@@@' /etc/passwd
root:x:0:0:System Administrator:/root:/bin/bash
@@@@@@@@@@@@@@@@@@@@@
bamboo:x:501:100:~/home/bamboo:/bin/csh
tania:x:502:502:~/home/tania:/bin/bash
@@@@@@@@@@@@@@@@@@@@@
figus:x:503:503:~/home/figus:/bin/bash
@@@@@@@@@@@@@@@@@@@@@
user1:x:504:100:~/home/user1:/bin/bash
```

Опция `-e` команды `sed` используется для указания того, что дальше в командной строке будет задан сценарий `sed` (пример 9.15). Первый сценарий удаляет все строки, не заканчивающиеся на строку `sh`, а второй добавляет разделительную строку только после строк, заканчивающихся на `bash`.

Очень удобно использовать `sed` для замены строк в потоке, адресуясь к строкам с помощью регулярных выражений. Например, требуется заменить в потоке, считанном из `/etc/group`, вхождения `daemon` в конце строк на `angel` (пример 9.16).

Пример 9.16. Замена с помощью регулярных выражений

```
$ sed 's/daemon$/angel/' /etc/group
```

К сожалению, `sed` работает только с очень ограниченным базовым набором регулярных выражений. Однако при замене подстроки имеется возможность вставлять в подстроку-замену исходную подстроку, удовлетворившую шаблону. Так, например, требуется вывести список всех файлов текущего каталога с суффиксом `.txt` так, чтобы к имени файлов была добавлена строка `.bak` (пример 9.17).

Пример 9.17. Обращение к найденной строке с помощью `&`

```
$ ls *.txt | sed 's/txt$/&.bak/'
dir.txt.bak
distfiles.txt.bak
ps.txt.bak
```

Для обращения к подстроке, удовлетворившей найденному регулярному выражению, в `sed` используется оператор `&`.

Задания

- С помощью `sed` получите список только процессов, связанных с виртуальными терминалами с первого по четвертый.
- Произведите замену строки `user` в потоке, считанном из файла `/etc/passwd`, на строку `extrauser`.
- Используя регулярное выражение `^.*bash` и оператор `&`, получите закомментированный символом "решетка" список всех пользователей с оболочкой `bash`.

Регулярные выражения в *awk*

В *awk* регулярные выражения могут использоваться аналогично тому, как они применяются в *sed* — для указания строк, которые должны подлежать обработке. Так, следующая команда выведет список только таких процессов в системе, которые были запущены в промежутке с 15:20 до 15:29.

Пример 9.18. Регулярные выражения в *awk*

```
$ ps -ef | awk '/[[[:blank:]]15:2[0-9][[:blank:]]/{print $5,$8}'
15:20 /usr/lib/openoffice/program/soffice.bin
15:29 [kjournald]
```

Эта команда (см. пример 9.18) выводит два столбца из потока, генерируемого командой *ps -ef*, столбец с информацией о времени запуска процесса и столбец с именем команды, запустившей процесс.

В этом случае пришлось указать наличие пробельных символов до и после столбца со временем запуска команды, т. к. далее в выводе команды *ps -ef* находится столбец, показывающий суммарное время выполнения процесса процессором. Наличие дополнительных регулярных выражений для пробельных символов сильно загромождает команду. Однако *awk* позволяет проверять на соответствие регулярному выражению не только целые строки, этот редактор отлично справляется с проверкой отдельных полей строк. Далее приведен пример команды, которая выполняет ту же задачу, что и предыдущая (пример 9.19).

Пример 9.19. Поиск регулярного выражения в поле

```
$ ps -ef | awk '$5~/15:2[0-9]/{print $5,$8}'
15:20 /usr/lib/openoffice/program/soffice.bin
15:29 [kjournald]
```

Синтаксис этой команды намного яснее предыдущей: здесь проверяется совпадение только пятого поля строки с регулярным выражением. Это достигается с помощью оператора *~*, который требует удовлетворения данного поля регулярному выражению.

Наоборот, при необходимости можно получить список строк, не удовлетворяющих регулярному выражению, с помощью оператора *!~*. Например, команда, приведенная в примере 9.20, выдаст список только таких процессов, которые *не* были запущены с 15:20 до 15:29.

Пример 9.20. Инверсия регулярного выражения

```
$ ps -ef | awk '$5!~/15:2[0-9]/{print $5,$8}'
```

Команда `awk` позволяет работать с расширенным синтаксисом регулярных выражений. Так, например, можно получить список всех процессов, в командной строке которых есть `bash` или `cs`h (пример 9.21).

Пример 9.21. Расширенные регулярные выражения в `awk`

```
$ ps -e | awk '$4~/(\b|c)sh/'
1749 pts/0    00:00:00 bash
7372 pts/0    00:00:00 cs
```

Особо следует отметить, что `awk` не поддерживает обратных ссылок, а символ группирования здесь используется лишь вместе с оператором перечисления ИЛИ (инфикс `|`). Однако практически это и не требуется, т. к. в `awk` имеется возможность представления строки в виде набора полей, которые можно сравнивать между собой.

Задания

- С помощью `awk` получите список учетных записей пользователей, для которых в качестве оболочки установлены `/bin/nologin`, `/bin/false`, `/dev/null` или же в седьмом поле `/etc/passwd` для них ничего не указано.
- Как с помощью `awk` проверить, есть ли в тексте символы звукового сигнала?
- С помощью `awk` получите из файла `/etc/bashrc` строки с табуляцией.
- Получите список пользователей из `/etc/passwd`, у которых имя пользователя содержится в имени домашнего каталога (например, `user` — `/home/user`).

Глава 10



Написание сценариев Bash

Возможность использования сценариев в Bash предоставляет администратору GNU/Linux возможность легко автоматизировать рутинные задачи, отнимающие массу времени. В этой главе вы научитесь составлять простые сценарии Bash и познакомитесь с основными конструкциями языка сценариев. Вы изучите команды ветвления, научитесь использованию циклов и функций.

Сценарии оболочки

Сценарий оболочки представляет собой текстовый файл, содержащий программу, состоящую из системных и встроенных команд. Они предназначены для автоматизации выполнения задач, чаще всего связанных с администрированием.

Оболочка последовательно интерпретирует и выполняет команды, заданные в сценарии. Эти же команды могут быть выполнены простым последовательным вызовом их в командной строке оболочки.

Для файлов сценариев оболочки Bash принято устанавливать расширение `.sh`.

Сценарии оболочки могут быть исполнены двумя различными путями.

- ❑ Имя файла сценария можно указать в качестве аргумента командной строки при явном запуске исполняемого файла оболочки, например, `bash`. В этом случае файл сценария должен быть доступен для чтения.
- ❑ Если в сценарии содержится неявный вызов оболочки, а файл сценария доступен для чтения и исполнения, то сценарий можно запустить так же, как и обычные системные команды.

Далее приведен пример запуска сценария путем явного вызова оболочки и указания имени сценария в качестве аргумента (пример 10.1).

Пример 10.1. Запуск сценария с явным вызовом оболочки

```
$ bash myscr1.sh
Privet!
```

При отладке сценариев исключительно важны опции `-v` и `-x` `bash`. Опция `-v` переводит оболочку в режим подробного информирования о работе. В этом режиме отображаются команды сценария перед их интерпретацией. Опция `-x` отображает результаты интерпретации команд. При неявном указании оболочки в первой строке сценария должна находиться строка с полным именем исполняемого файла оболочки (пример 10.2).

Пример 10.2. Неявный вызов оболочки

```
$ cat myscr1.sh
#!/bin/bash
echo 'Privet!'
```

Строка `#!/bin/bash` указывает, с помощью какой оболочки должен быть интерпретирован и выполнен сценарий.

Запуск сценария при неявном вызове оболочки возможен, если на файл сценария имеются разрешения для чтения и исполнения (пример 10.3).

Пример 10.3. Установка прав на исполнение сценария

```
$ chmod a+rx myscr1.sh
$ ./myscr1.sh
Privet!
```

Обратите внимание на то, что нахождение исполняемого файла в текущем каталоге не означает возможности его запуска без указания пути к нему. При отсутствии имени текущего каталога в переменной `PATH` имя текущего каталога (точка) должно быть указано в пути к файлу сценария — `./`.

Часто возникает необходимость в процессе исполнения сценария считать и исполнить содержимое другого файла сценария в контексте вызывающего сценария. Если просто вызвать сценарий из другого сценария, то вызываемый сценарий будет исполнен в собственной оболочке. Переменные, значения которых были установлены в вызываемом сценарии, не будут известны в вызывающем сценарии.

При необходимости выполнения сценария в контексте вызывающего сценария (оболочки) нужно использовать inline-подстановку (пример 10.4).

Пример 10.4. Использование inline-подстановки

```
$ . .bashrc
```

Команда "точка" является inline-подстановкой. В примере 10.4 содержимое файла сценария `.bashrc` будет выполнено в контексте вызывающей оболочки.

Inline-подстановка часто используется для считывания в сценарии переменных, заданных в другом файле (пример 10.5).

Пример 10.5. Установка значений переменных из файла

```
$ cat myscr2rc
VAR1='Snova Privet!'
$ cat myscr2.sh
#!/bin/bash
. myscr2rc
echo $VAR1
$ ./myscr2.sh
Snova Privet!
```

В примере 10.5 в коде сценария `myscr2.sh` была выполнена inline-подстановка содержимого файла `myscr2rc`, в котором была установлена переменная `VAR1`.

Задания

- В сценарии `s1.sh` определите переменную `v1` и выведите ее значение.
- Запустите сценарий, указывая оболочку явно.
- Измените сценарий для неявного вызова оболочки.
- Перепишите сценарий `s1.sh` таким образом, чтобы из него вызывался сценарий `s2.sh`, который и печатал бы значение переменной `v1`.
- Измените сценарий `s1.sh` так, чтобы переменная `v1` считывалась из файла `s1rc`.

Использование переменных оболочки

Переменные оболочки предназначены для временного хранения строковых значений. Память, необходимая для размещения значений переменных, выделяется оболочкой динамически по мере надобности.

Все переменные оболочки хранят строки. Однако переменные, значения которых представляют собой целые числа, допускают выполнение простейших арифметических операций (пример 10.6).

Пример 10.6. Вычисление арифметических выражений в Bash

```
$ V1=10
$ echo $(( V1+1 ))
11
```

Имя переменной должно использовать только символы английского алфавита, цифры и символ подчеркивания. Имя переменной должно начинаться либо с буквы, либо с символа подчеркивания. Имена переменных чувствительны к регистру.

Для исключения неверной интерпретации оболочкой командной строки, в которой непосредственно за именем переменной следует некоторый символ, можно отделить имя переменной с помощью фигурных скобок (пример 10.7).

Пример 10.7. Экранирование имени переменной

```
$ f1=str
$ echo $f1
str
$ echo $f11
$ echo ${f1}1
str1
```

В примере 10.7 переменной `f1` присвоено значение `str`. Добавить к строке `str` символ `1` нельзя без экранирования имени переменной фигурными скобками, т. к. иначе оболочка попытается вывести значение переменной `f11`.

Имеется возможность использовать значение по умолчанию для переменной. Если переменная определена, то используется ее значение. В противном случае используется значение по умолчанию (пример 10.8).

Пример 10.8. Значения по умолчанию

```
$ V1=abcdef
$ echo ${V1:-09876}
abcdef
$ echo ${V2:-09876}
09876
```

```
$ echo $v1
abcdef
$ echo $v2
```

В примере 10.9 переменная `v1` определена, а `v2` — нет. Поэтому для переменной `v1` конструкция (значение по умолчанию) `${v1:-09876}` дает значение переменной, а для `v2` — возвращает значение по умолчанию `09876`. Собственно значения переменных `v1` и `v2` не изменяются.

Если же необходимо назначить переменной значение по умолчанию в случае, когда она не определена, следует использовать конструкцию `${переменная:=значение}` (пример 10.9).

Пример 10.9. Установка значения по умолчанию

```
$ echo ${v2:=12345}
12345
$ echo $v2
12345
```

Так как переменная `v2` в примере 10.9 не была определена, то вместо нее было использовано значение `12345`, которое при этом было назначено этой переменной.

При необходимости использовать вместо определенной переменной заданную строку применяют конструкцию `${переменная:+значение}` (пример 10.10).

Пример 10.10. Замещение значением по умолчанию

```
$ echo ${v2:+'nu i dela...'}
nu i dela...
```

Здесь вместо установленного значения переменной была использована строка — значение по умолчанию.

Задания

- Какая команда оболочки извлекает значение переменной, имя которой содержится в другой переменной?
- Как экспортировать переменную, чье имя содержится в другой переменной?
- Куда помещаются экспортированные переменные?
- Поместите в переменную `ENV` все имена переменных окружения.
- Где можно увидеть переменные окружения, установленные для процесса `bash`?

Экранирование (quotation)

При необходимости указать в качестве аргумента какой-либо команды или назначить переменной значение, содержащее пробелы или метасимволы, они должны быть защищены от интерпретации оболочкой. Механизм защиты специальных символов оболочки от интерпретации называется *экранированием* (quotation).

Например, требуется назначить переменной `STR1` значение `Bolshoy Privet`. Эта строка содержит пробел, который должен быть предотвращен от интерпретации оболочкой. Экранировать его можно, например, установив перед ним символ обратной косой черты (пример 10.11).

Пример 10.11. Экранирование обратной косой чертой

```
$ STR1=Bolshoy\ Privet
$ echo $STR1
Bolshoy Privet
```

Обратная косая черта устраняет интерпретацию метасимвола, следующего за ней.

Для экранирования специальных символов применяются:

- ❑ одиночные кавычки `' '`;
- ❑ двойные кавычки `" "`;
- ❑ символ обратной косой черты `\`.

Одиночные кавычки используются парой и устраняют интерпретацию специального значения всех метасимволов, заключенных в них, кроме других одиночных кавычек (пример 10.12).

Пример 10.12. Экранирование одиночными кавычками

```
STR1='Stroka soderzhit $TERM'
aberes@newnote aberes $ echo $STR1
Stroka soderzhit $TERM
```

Заметно, что символ доллара, имеющий особое значение в оболочке, защищен от интерпретации одиночными кавычками.

Двойные кавычки также используются парой (пример 10.13) и устраняют интерпретацию метасимволов кроме:

- ☐ других двойных кавычек `"`;
- ☐ символа доллара `$`;
- ☐ обратной косой черты `\`;
- ☐ обратной кавычки ```.

Пример 10.13. Экранирование с помощью двойных кавычек

```
$ STR1="'Stroka soderzhit $TERM'"
aberes $ echo $STR1
'Stroka soderzhit xterm'
```

В примере 10.13 показано, что использование двойных кавычек не предохраняет от интерпретации оболочкой символов доллара. В то же время одиночные кавычки, заключенные в двойные, не оказали воздействия на интерпретацию символа доллара.

ЗАДАНИЯ

- Задайте переменной `v1` значение `$1000`. Выведите значение переменной.
- Назначьте переменной `v1` строку, составленную из ее предыдущего значения, взятого в одиночные кавычки.

Интерактивная установка значений переменных

Для получения значения переменной непосредственно от пользователя предназначена команда `read`, которая читает данные из стандартного потока ввода (пример 10.14).

Пример 10.14. Ввод значения

```
$ echo -n 'Введите строку:'; read var; echo $var
Введите строку:Добрый день!
Добрый день!
```

В примере 10.14 с клавиатуры должно быть введено значение переменной `var`. Опция `-n` команды `echo` использована здесь для отключения перевода строки после вывода строки приглашения.

Можно одновременно ввести значения для нескольких переменных сразу:
`read var1 var2 var3.`

Если в строке, поступающей из потока стандартного ввода для команды `read`, содержится больше значений, чем имеется аргументов у команды `read`, то все значения, для которых нет соответствующих переменных, в виде целой строки присваиваются последней переменной в списке.

Наоборот, если задано больше переменных, чем введено значений, то переменным, для которых не хватило значений, присваивается пустая строка.

ЗАДАНИЕ

Напишите сценарий оболочки, считывающий значения трех переменных и выводящий их значения в стандартный поток вывода. Проверьте его работу, вводя два, три и четыре значения.

Позиционные параметры

Позиционные параметры позволяют сценариям оболочки получать информацию, задаваемую в командной строке при их запуске.

В Bash имеется десять позиционных параметров: `$0`, `$1`, ..., `$9`. Они содержат:

- ☐ `$0` — имя команды;
- ☐ параметры от `$1` до `$9` — значения девяти аргументов командной строки.

Пример 10.15. Позиционные параметры

```
$ cat param.sh
#!/bin/bash
echo $1
echo $2
$ ./param.sh first second
first
second
```

В сценарии `param.sh` (пример 10.15) содержатся две команды `echo`, выводящие содержимое первого и второго позиционных параметров, которое соответствует первому и второму аргументу командной строки.

Кроме позиционных параметров в Bash используются специальные параметры:

- ☐ `$*` — строка, составленная из значений всех аргументов командной строки;
- ☐ `$@` — содержит строку, составленную из значений всех аргументов командной строки, разделенных пробелами (аналогично, но не тождественно `$*`);

- ❑ \$# — количество аргументов командной строки;
- ❑ \$? — код возврата предыдущей команды;
- ❑ \$\$ — PID оболочки.

В примере 10.16 сценарий выводит все аргументы командной строки.

Пример 10.16. Параметр \$*

```
$ cat comline.sh
#!/bin/bash
echo $*

$ ./comline.sh 11 aa 22 bbb
11 aa 22 bbb
```

Позиционные параметры не позволяют устанавливать их значения, т. к. их значения назначаются автоматически оболочкой.

При необходимости получить значения аргументов (опций) командной строки, содержащей более девяти аргументов, используют команду `shift`. Команда `shift`, вызванная без аргументов, сдвигает позиционные параметры вправо (пример 10.17). То есть \$1 получит значение, которое имел \$2, параметр \$2 — значение \$3 и т. д.

Пример 10.17. Команда shift

```
$ cat param.sh
#!/bin/bash
echo $1
echo $2
echo And now all parameters are shifted.
shift
echo $1
echo $2

$ ./param.sh first second third
first
second
And now all parameters are shifted.
second
third
```


В этом сценарии (пример 10.17) сначала выводятся аргументы командной строки без сдвига. То есть выводятся первый и второй аргументы командной строки, а после использования команды `shift` — второй и третий, хотя использованы все те же позиционные параметры `$1` и `$2`.

Для сдвига более чем на одну позицию вправо необходимо указать величину сдвига, используя аргумент команды `shift`. Например, `shift 4` сдвинет параметры на четыре позиции вправо.

Вернуться после сдвига параметра к их предыдущим значениям нельзя. Поэтому при необходимости запомнить более девяти аргументов командной строки следует сохранить в переменных параметры, значения которых будут потеряны при сдвиге.

При необходимости назначить новые значения позиционным параметрам текущей оболочки можно воспользоваться командой `set`, которая позволяет сконструировать командную строку заново, назначая новые значения сразу всем позиционным параметрам (пример 10.18).

Пример 10.18. Установка аргументов с помощью `set`

```
$ cat param.sh
#!/bin/bash
echo $1
echo $2
set 1 2
echo $1
echo $2

$ ./param.sh first second
first
second
1
2
```

Здесь продемонстрировано, что изменение всех аргументов командой `set` приводит к изменению позиционных параметров.

При необходимости сохранить значение одного или нескольких позиционных параметров следует использовать команду `set`, указав в требуемых позициях позиционные параметры, не подлежащие изменению (пример 10.19).

Пример 10.19. Сохранение значений позиционных параметров при вызове `set`

```
set $1 $2 newarg $4
```

В этом случае (см. пример 10.19) командная строка текущей оболочки содержит только четыре аргумента. Команда `set` изменяет все их сразу, а для сохранения значений первому, второму и четвертому аргументам просто присваиваются их же старые значения. Третий аргумент получает здесь новое значение — `newarg`.

Задания

- Напишите сценарий, выводящий имя команды, количество аргументов и PID.
- Проверьте работу сценария, используя явный вызов оболочки `bash`.
- Исследуйте работу опции `-v bash`.
- Измените сценарий так, чтобы он выводил все аргументы командной строки.
- Поместите в сценарий команду `shift`. Как он работает?
- Командой `set` установите в сценарии новые значения позиционных параметров.

Команда `test`

Команда `test` позволяет проверить заданные условия. Виды проверок, выполняемых командой `test`:

- ☐ проверка файлов на предмет выполнения заданных условий;
- ☐ сравнение файлов;
- ☐ проверка установки опций оболочки;
- ☐ сравнение строк;
- ☐ сравнение целых чисел.

Если тест выполнен успешно, команда `test` передает нулевой код возврата. Так, используя команду `test -e`, можно проверить существование файла (пример 10.20).

Пример 10.20. Проверка существования файла

```
$ test -e /etc/passwd
$ echo $?
0
$ test -e not_existent_file
$ echo $?
1
```

В первом случае команда `test` вернула нулевой код возврата, поскольку файл `/etc/passwd` существует. Во втором случае команда вернула код ошибки.

Команда `test` в сценариях обычно вызывается в другой форме, совершенно эквивалентной показанной ранее. В этой форме вместо строки `test` указывают квадратные скобки и условие в них: `[условие]`. Задача из предыдущего примера может быть решена иным способом (пример 10.21).

Пример 10.21. Другая форма команды `test`

```
$ [ -e /etc/passwd ]  
$ echo $?  
0
```

Команда `[-e /etc/passwd]` эквивалентна команде `test -e /etc/passwd`.

Наиболее часто используемые опции команды `test`, связанные с проверкой файлов:

- ❑ `-e` — файл существует;
- ❑ `-f` — файл является обычным файлом (plain file);
- ❑ `-d` — файл является каталогом;
- ❑ `-h` или `-L` — файл является символической ссылкой;
- ❑ `-r` — файл доступен для чтения;
- ❑ `-w` — файл доступен для записи;
- ❑ `-x` — файл доступен для исполнения;
- ❑ `-s` — файл не пуст;
- ❑ `-N` — файл был модифицирован.

Формат вызова `test`, который используется при сравнении файлов:

- ❑ `[file1 -nt file2]` — возвращает истину, если первый файл имеет более позднюю дату модификации;
- ❑ `[file1 -ot file2]` — возвращает истину, если первый файл имеет более раннюю дату модификации;
- ❑ `[file1 -ef file2]` — проверка жесткой связи (hard link).

Опция `-o` позволяет проверять установку опций оболочки (пример 10.22).

Пример 10.22. Проверка установленных опций оболочки

```
$ set -o noclobber
$ [ -o noclobber ]
$ echo $?
0
$ set +o noclobber
$ [ -o noclobber ]
$ echo $?
1
```

Для сравнения строк применяется следующий формат вызова команды `test`:

- ❑ `[str1 = str2]` — проверка на совпадение строк;
- ❑ `[str1 != str2]` — проверка на несовпадение строк;
- ❑ `[str1 < str2]` — истина, если при сортировке строка `str1` окажется раньше, чем `str2` (по аналогии сравнение `>`);
- ❑ `[-z str]` — истина, если длина строки нулевая;
- ❑ `[-n str]` — истина, если длина строки ненулевая.

Сравнение целых чисел (пример 10.23) производится с помощью следующих опций команды `test`:

- ❑ `-eq` — равенство;
- ❑ `-ne` — неравенство;
- ❑ `-lt` — меньше;
- ❑ `-le` — меньше или равно;
- ❑ `-gt` — больше;
- ❑ `-ge` — больше или равно.

Пример 10.23. Сравнение чисел

```
$ [ 1 -lt 2 ]
$ echo $?
0
$ [ 1 -eq 2 ]
$ echo $?
1
```

Задания

- Как проверить, является ли файл специальным файлом блочного устройства?
- Как, используя `test`, проверить, содержит ли переменная `STR1` значение `str1`?
- Проверьте, установлен ли запрет на нажатие комбинации клавиш `<Ctrl>+<D>` для выхода из сеанса.
- С помощью `test` проверьте, имеется ли жесткая связь между `gzip` и `gunzip`.

Условное исполнение команд

Операторы условного исполнения команд `&&` и `||` могут быть применены для управления потоком исполнения команд. Они позволяют исполнять команды в зависимости от успешности выполнения предыдущих команд.

Если оператор `&&` установлен между двумя командами, то вторая из них будет исполнена только в случае успеха предыдущей (пример 10.24).

Пример 10.24. Выполнение команды в случае успеха предыдущей команды

```
$ ls /tmp &> /dev/null && cd /tmp
$ pwd
/tmp
```

В примере 10.24 первая команда проверяет наличие каталога, переход в который осуществляет вторая команда `cd /tmp` при условии успешного исполнения первой.

Более изящный вариант выполнения этой же задачи заключается в использовании команды `test` для проверки существования целевого каталога (пример 10.25).

Пример 10.25. Использование команды `test` с условными операторами

```
$ [ -d /tmp ] && cd /tmp
$ pwd
/tmp
```

Команда `test` в примере 10.25 проверяет существование каталога.

Если необходимо выполнять команду в случае неудачного завершения работы другой команды, удобно применять оператор `||`. Например, требуется перейти в каталог `d1`. В случае отсутствия этот каталог необходимо создать и перейти в него. Эту задачу можно решить следующим образом (пример 10.26).

Пример 10.26. Выполнение команды при неудаче предыдущей команды

```
$ [ -d d1 ] || mkdir d1 ; cd d1
$ pwd
/tmp/d1
```

Если каталог отсутствует (пример 10.26), то он создается.

Оболочка предоставляет также специальную команду `if`, которая позволяет управлять последовательностью исполнения целых блоков команд. В качестве примера предположим, что требуется обеспечить выход из некоторого сценария с ошибкой, если в командной строке установлено отличное от единицы число аргументов командной строки. При этом команда должна сообщать об ошибке и выдавать подсказку о правильном варианте ее использования (пример 10.27).

Пример 10.27. Команда `if`

```
#!/bin/bash
if [ $# -ne 1 ]
then
    cat <<- ERR
        Недостаточно аргументов.
        Использование:
        if.sh file
        Аргумент file должен быть обычным файлом.
    ERR
    exit 1
fi
ls -l $1
```

Команда `if` исполняет блок команд после команды `then`, только в случае получения нулевого кода возврата команды, указанной в качестве ее аргумента. В данном случае аргумент команды `if` — это команда `test`, которая проверяет количество аргументов сценария. Если количество аргументов не равно единице, то выполняется блок операторов после `then` до команды `fi`, заканчивающей `if`.

Конструкция `cat <<- ERR` — это "here document". Она позволяет указывать целый блок данных непосредственно в теле сценария, передавая его в данном случае в стандартный поток ввода команде `cat` для вывода на экран. Блок

данных ограничен строкой `ERR`. В этом примере используется оператор `<<-` вместо `<<` для игнорирования табуляций перед блоком данных.

Результат — в примере 10.28.

Пример 10.28. Результаты работы программы с командой `if`

```
$ ./if.sh
Недостаточно аргументов.
Использование:
if.sh file
Аргумент file должен быть обычным файлом.
$ ./if.sh if.sh
-rwxr--r--    1 user1  users      172 Oct  8 19:34 if.sh
```

Команда `if` допускает использование команды `elif` для выполнения дополнительной проверки (пример 10.29).

Пример 10.29. Команда `elif`

```
#!/bin/bash
if [ $# -ne 1 ]
then
    cat <<- ERR
        Недостаточно аргументов.
        Использование:
        if.sh file
        Аргумент file должен быть обычным файлом.
    ERR
    exit 1
elif [ ! -f $1 ]
then
    echo -n 'Тип файла '
    file $1
    exit 1
fi
ls -l $1
```

Если аргументом задан специальный файл, то выводится его тип (пример 10.30).

Пример 10.30. Проверка работы сценария с `elif`

```
$ ./if.sh .
```

```
Тип файла .: directory
```

В команде `if` можно также использовать `else` для указания блока операторов, которые должны исполняться в случае, если предыдущие проверки не закончились успехом (пример 10.31).

Пример 10.31. Команда `else`

```
#!/bin/bash
```

```
if [ $# -ne 1 ]
```

```
then
```

```
    cat <<- ERR
```

```
        Недостаточно аргументов.
```

```
        Использование:
```

```
        if.sh file
```

```
        Аргумент file должен быть обычным файлом.
```

```
    ERR
```

```
    exit 1
```

```
elif [ ! -f $1 ]
```

```
then
```

```
    echo -n 'Тип файла '
```

```
    file $1
```

```
    exit 1
```

```
else
```

```
    ls -l $1
```

```
fi
```

Задания

- Напишите сценарий, проверяющий имя текущего каталога и выводящий сообщение об ошибке, если оно короче пяти символов.
- Требуется проверить, является ли файл обычным или каталогом. Если это обычный файл, то сценарий должен выводить имя файла и его размер. В случае если размер файла превышает 1 Кбайт, то размер должен выводиться в килобайтах. Если размер превышает 1 Мбайт — в мегабайтах.

Команда *case*

При создании сценариев часто возникает задача проверить значение, содержащееся в переменной, на совпадение с заданными шаблонами. При этом в зависимости от шаблона, с которым произошло совпадение, должны выполняться те или иные заданные действия. Для выполнения такой задачи предназначена команда *case*. Шаблоны сравнения *case* такие же, как и файловые шаблоны.

Синтаксис команды *case* в общем показан в примере 10.32.

Пример 10.32. Команда *case*

```
case слово in
    шаблон1 )
        команды
        ;;
    шаблон2 )
        команды
        ;;
esac
```

Сравнение слова с шаблонами производится последовательно. Как только найдется совпадение, выполняются соответствующие команды без дальнейших проверок.

Допустим, что в текущем каталоге располагаются символические ссылки на сценарии запуска служб GNU/Linux. Требуется написать сценарий, который будет запускать службы, передавая сценарию аргумент *start*, если в качестве аргумента будет использована символическая ссылка, начинающаяся с буквы *s* или *с*. Если же ссылка будет начинаться с буквы *к* или *к*, то службы должны останавливаться и, следовательно, их сценариям должен передаваться аргумент *stop* (пример 10.33).

Пример 10.33. Команда *case*

```
$ cat case.sh
#!/bin/bash
[ $# -ne 1 ] && exit 1;
FIRST=`echo $1 | cut -c1`
case $FIRST in
```

```
[Ss] )
    echo "Запускается $1"
    ./$1 start
    ;;

K|k )
    echo "Останавливается $1"
    ./$1 stop
    ;;

* )
    echo "Статус $1"
    ./$1 status
    ;;
```

esac

В этом сценарии вначале проверяется количество аргументов. Если оно не равно 1, то осуществляется выход с ошибкой. Далее в переменную `FIRST` помещается первая буква аргумента командной строки.

Команда `case` проверяет соответствие содержащейся в переменной `FIRST` буквы шаблону `[Ss]`. Этот шаблон обозначает множество по аналогии с обычными файловыми шаблонами. То есть этому шаблону удовлетворяют либо `S`, либо `s`.

Если буква в `FIRST` удовлетворяет этому шаблону, служба запускается.

Для остановки службы используется иной шаблон, имеющий в данном случае тот же смысл: либо `K`, либо `k`. Вертикальная черта обозначает "ИЛИ". Однако ее можно применять даже для целых строк или шаблонов.

Если содержимое переменной `FIRST` не совпадает и с этим шаблоном, то выводится информация о текущем статусе службы.

Задания

- Изучите любой сценарий в `/etc/init.d`. Как используется команда `case`?
- Напишите сценарий, анализирующий с помощью `case` список пользователей, находящихся в настоящий момент в системе. Если имеется хотя бы один сеанс `root`, должно выдаваться предупреждающее сообщение.

Циклы

В случаях, когда надо организовать последовательное выполнение одного и того же набора инструкций, удобно использовать программные циклы. Для программирования в оболочке доступны три вида циклов:

❑ `for` — эта команда позволяет создавать циклы типа перебора значений;

- ❑ `while` — цикл, выполняющийся до тех пор, пока истинно некоторое условие;
- ❑ `until` — цикл, выполняющийся до тех пор, пока некоторое условие ложно.

Допустим, что имеется набор значений, которые должны быть последовательно присвоены некоторой переменной, требующейся для произведения каких-либо операций. В этом случае удобно использовать команду `for` (пример 10.34).

Пример 10.34. Цикл `for`

```
#!/bin/bash

for DIR in /etc /tmp /var
do
    echo -n "Права доступа к $DIR "
    ls -ld $DIR | cut -c2-11
done
```

В данном сценарии переменная `DIR` последовательно принимает три значения: `/etc`, `/tmp` и `/var`. Это делает команда `for`, в которой список значений указан после `in`. Тело цикла начинается после команды `do` и ограничивается `done`.

Результат работы сценария представлен в примере 10.35.

Пример 10.35. Результаты работы цикла `for`

```
$ ./for.sh
Права доступа к /etc  rwxr-xr-x
Права доступа к /tmp  rwxrwxrwt
Права доступа к /var  rwxr-xr-x
```

Если в команде `for` не указан список значений директивой `in`, то переменная цикла будет принимать значения, соответствующие аргументам командной строки.

Изменим предыдущий сценарий так, чтобы имена каталогов можно было бы задавать в качестве аргументов в командной строке (примеры 10.36 и 10.37).

Пример 10.36. Перебор аргументов командной строки с помощью for

```
#!/bin/bash
[ $# -lt 1 ] && exit 1
for DIR
do
    if [ -d $DIR ]; then
        echo -n "Права доступа к $DIR "
        ls -ld $DIR | cut -c2-11
    elif [ ! -e $DIR ]; then
        echo "$DIR не существует"
    fi
done
```

В этом сценарии переменная `DIR` последовательно принимает значения аргументов.

Пример 10.37. Результат перебора аргументов командной строки

```
$ ./for.sh /usr /rsu /root
Права доступа к /usr rwxr-xr-x
/rsu не существует
Права доступа к /root rwx-----
```

Для организации итеративных циклов удобно использовать команды `while` и `until`.

Сценарий из примера 10.38 в цикле выводит значения от 1 до 5 раз в секунду (пример 10.39).

Пример 10.38. Итеративный цикл

```
#!/bin/bash

i=1
while [ $i -le 5 ]; do
    echo -n $i ' ' ; sleep 1
    i=$((i+1))
done
echo
```

Пример 10.39. Результаты работы итеративного цикла

```
$ ./while.sh
1 2 3 4 5
```

Цикл `while` работает до тех пор, пока команда, указанная в качестве ее аргумента, возвращает успешный код завершения. Наоборот, `until` работает, пока команда-аргумент заканчивается неудачей.

В качестве примера приведем сценарий, последовательно удаляющий из полного доменного имени узла (FQDN) подстроки до первой точки (пример 10.40). Сначала должно быть выведено полное имя узла, затем домен, потом родительский домен, и так далее, пока строка не станет пустой.

Пример 10.40. Цикл `until`

```
#!/bin/bash
HN=`hostname`
until [ -z $HN ]; do
    echo $HN
    HN=`echo -n $HN | tr '.' '\n' | sed 'ld' | tr '\n' '.'`
done
```

В сценарии используется переменная `HN`, которой вначале присваивается значение — доменное имя узла. Затем из него в цикле удаляется подстрока от начала строки до первой встретившейся точки. Для этого в строке — имени узла точки сначала заменяются переводами строк, затем удаляется первая строка, и в заключение переводы строки снова заменяются точками.

Цикл `until` работает до тех пор, пока содержимое переменной `HN` не станет пустым.

Результат работы сценария представлен в примере 10.41.

Пример 10.41. Результаты работы цикла `until`

```
$ hostname
nechto.zamislovatoe.tmn.ru
$ ./until.sh
nechto.zamislovatoe.tmn.ru
zamislovatoe.tmn.ru
tmn.ru
ru
```

Задания

- Напишите сценарий, в котором переменная `USERS` будет содержать список пользователей в системе. В цикле выведите содержимое этой переменной так, чтобы для каждого пользователя также выводилось имя его домашнего каталога.
- Напишите сценарий, выводящий посекундно в цикле имена файлов текущего каталога и их порядковый номер.

Функции

Написание хорошо структурированных программ требует возможности создания подпрограмм. В оболочке Bash это реализуется с помощью функций, представляющих собой именованные блоки кода (пример 10.42). При определении функции в оболочке (даже просто в командной строке) ее можно вызвать по имени.

Пример 10.42. Синтаксис функции

```
function ИМЯ()  
{  
    КОМАНДЫ  
}
```

Код функции описывают в начале сценария до первого вызова функции. Для вызова функции достаточно просто указать ее имя.

Для демонстрации использования функций оболочки модифицируем программу `if.sh` (см. примеры 10.27, 10.29, 10.31). Измененный код сценария `if.sh` представлен в примере 10.43, а результат его работы — в примере 10.44.

Пример 10.43. Использование функций

```
#!/bin/bash  
function err_msg()  
{  
    cat <<- ERR  
        Недостаточно аргументов.  
        Использование: if.sh file  
        Аргумент file должен быть обычным файлом.  
    ERR  
}
```

```

if [ $# -ne 1 ]
then
    err_msg
    exit 1
elif [ ! -f $1 ]
then
    echo -n 'Тип файла '
    file $1
    exit 1
fi
ls -l $1

```

Пример 10.44. Проверка работы функции в Bash

```

$ ./if.sh
Недостаточно аргументов.
Использование: if.sh file
Аргумент file должен быть обычным файлом.
$ ./if.sh /etc/passwd
-rw-r--r--    1 root      root          2033 Oct 19 23:24 /etc/passwd

```

Для передачи аргументов в функцию их указывают после имени функции, разделяя пробелами. В теле функции обращение к переданным аргументам производится с помощью позиционных параметров.

Модификация `if.sh`, с передачей аргументов в функцию, показана в примере 10.45, а результат работы — в примере 10.46.

Пример 10.45. Передача значений в функции

```

#!/bin/bash
function err_msg()
{
    echo "Ошибка: $1"
    cat <<- ERR
        Недостаточно аргументов.
        Использование: if.sh file
        Аргумент file должен быть обычным файлом.
    ERR
}

```

```
if [ $# -ne 1 ]
then
    err_msg '001'
    exit 1
elif [ ! -f $1 ]
then
    echo -n 'Тип файла '
    file $1
    exit 1
fi
ls -l $1
```

Здесь в функцию передается строка — код ошибки. В теле функции этот код считывается из позиционного параметра.

Пример 10.46. Проверка передачи аргументов в функции

```
$ ./if.sh
Ошибка: 001
Недостаточно аргументов.
Использование: if.sh file
Аргумент file должен быть обычным файлом.
```

Функции часто описывают в отдельных файлах, считывая в сценариях содержимое этих файлов с помощью inline-подстановки.

Задания

- Переделайте сценарий `if.sh` так, чтобы функция, выводящая сообщение об ошибке, находилась в отдельном файле сценария `err.sh`.
- Измените сценарий `case.sh` (см. пример 10.32) так, чтобы команды, выполняющиеся при совпадении переменной `FIRST`, были реализованы в функциях.



ЧАСТЬ IV

Администрирование



Глава 11

Работа с носителями информации

Одна из важнейших задач системного администратора GNU/Linux заключается в поддержании работоспособности файловых систем. В этой главе вы познакомитесь с организацией хранения данных на жестких магнитных дисках, научитесь создавать дисковые разделы, файловые системы и разделы подкачки. Монтирование файловых систем и поддержка их в рабочем состоянии также обсуждаются в этой главе.

Физическая структура накопителя на жестких магнитных дисках

Современные накопители на жестких магнитных дисках состоят из одного или нескольких магнитных дисков и магнитных головок (head). Каждая магнитная поверхность диска разбита на дорожки (track). Дорожки одного диаметра на всех магнитных поверхностях образуют цилиндр (cyl). Количество дорожек равно произведению количества цилиндров на количество головок:

$$\text{track} = \text{cyl} \times \text{head}.$$

Каждая дорожка разбита на секторы (sect), стандартный размер которых составляет 512 байтов. Объем диска в байтах равен:

$$\text{capacity} = \text{cyl} \times \text{head} \times \text{sect} \times 512.$$

Стандартное количество секторов в дорожке равняется 63. Поскольку количество байтов в секторе и секторов в дорожке являются постоянными величинами, то основными параметрами диска (так называемой *геометрией*) являются количества цилиндров cyl и головок head.

Команда `fdisk -l` выводит сведения о геометрии диска (пример 11.1).

Пример 11.1. Получение сведений о геометрии диска

```
# fdisk -l
```

```
Disk /dev/sda: 160.0 GB, 160041885696 bytes
255 heads, 63 sectors/track, 19457 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0xf866a219
```

Device	Boot	Start	End	Blocks	Id	System
--------	------	-------	-----	--------	----	--------

```
# echo $(( 19457*255*63*512 ))
160041885696
```

В примере 11.1 использован диск с 255 головками и 19 457 цилиндрами (не правда ли, много головок). Зная эти данные, легко можно вычислить объем этого диска. Размер диска отображается в первой строке.

Диск разбивается на *разделы* (partitions). Обычно каждый раздел предназначен для размещения одной операционной системы, отдельной файловой системы или области размещения страниц подкачки. В нулевом секторе диска хранится таблица разделов, которая указывает начальный и конечный цилиндры для каждого раздела.

Ограничения, исходно наложенные на геометрию жестких дисков, не предусматривали возможности иметь жесткие диски с размером более 520 Мбайт. У жестких дисков того времени предусматривалось наличие максимум 16 головок и 1024 цилиндров, что и давало ограничение в 520 Мбайт. В таблице разделов для хранения номеров цилиндров разделов предназначается только 10 бит, т. е. 1024 (2^{10}) цилиндров максимум. Это ограничение быстро стало не просто существенным, а критическим. Для его преодоления был введен так называемый *режим LBA* (Linear Block Addressing), в котором за счет мнимого увеличения количества головок удавалось виртуально понизить количество цилиндров, оставляя при этом их произведение неизменным. Это не единственное ограничение на объем дискового пространства, пройденное с 80-х годов прошлого века. Подробную информацию об этом можно найти на сайте <http://www.tldp.org> в документе "Hard disks HOWTO".

Другое историческое ограничение связано с тем, что исходно разработчики IBM PC заложили возможность использования лишь *четырех разделов* на жестком диске. Для Linux-систем такое ограничение могло привести к невозможности создания требуемого количества файловых систем, которые должны находиться на разных разделах.

Для преодоления этого ограничения в GNU/Linux (как и в ПО от Microsoft) используются *логические разделы*. В этой модели четыре основных раздела называются *первичными разделами* (primary), причем один из них может быть помечен, как *расширенный раздел* (extended). В расширенном разделе может быть создано неограниченное количество логических разделов (logical partitions).

ЗАДАНИЯ

- Проверьте геометрию диска, установленного на вашем компьютере с помощью `fdisk -l`.
- Что делает команда `sfdisk -s?`

Имена жестких магнитных дисков

Ядро Linux предоставляет унифицированный интерфейс к устройствам — файлы устройств. Большинству устройств (за исключением, разве что, сетевых интерфейсов) в Linux соответствуют специальные файлы устройств, размещающиеся в каталоге `/dev`. Эти файлы не используют блоки данных в файловой системе. Команда `ls -l` для файлов устройств вместо размера выводит два параметра: мажор и минор. Мажор — это номер драйвера для этого вида устройств в ядре Linux, а минор — номер экземпляра устройства, обслуживаемого данным драйвером.

Файлы устройств бывают блочными (в выводе первый `ls -l` символ `b`) и символьными (в выводе первый `ls -l` символ `c`). *Блочные устройства* в Linux — все те, информацию на которые можно записывать исключительно блоками и считывать так же. *Блочные файлы устройств* — интерфейс к устройствам, имеющим файловую систему. Примерами *символьных устройств* являются терминал, клавиатура и мышь. Обмен информацией с такими устройствами осуществляется посимвольно.

Схема именования файлов устройств фиксируется специальным соглашением, которое можно найти в каталоге с исходным кодом ядра (обычно `/usr/src/linux`) в файле `Documentation/devices.txt`. Соглашение устанавливает, что файлы устройств IDE жестких дисков называются:

- ☐ `/dev/hda` — Primary Master;
- ☐ `/dev/hdb` — Primary Slave;
- ☐ `/dev/hdc` — Secondary Master;
- ☐ `/dev/hdd` — Secondary Slave.

Имена SCSI-дисков (и SATA) начинаются с `sd`, первому SCSI-диску соответствует файл устройства `/dev/sda`, второму — `/dev/sdb` и т. д. в соответствии со SCSI ID данного жесткого диска.

В ядре Linux для IDE-дисков предусматривается мажор 3, а минор зависит от номера раздела: 0 — для всего диска, 1 — для первого раздела и т. д. Для разделов SCSI и SATA дисков (мажор 8) предусматривается аналогичный порядок (пример 11.2).

Пример 11.2. Файлы устройств жестких дисков

```
$ ls -l /dev/hda{,1}
brw-rw---- 1 root disk 3, 0 Oct 20 2009 /dev/hda
brw-rw---- 1 root disk 3, 1 Oct 20 2009 /dev/hda1
$ ls -l /dev/sda{,1}
brw-rw---- 1 root disk 8, 0 Oct 20 2009 /dev/sda
brw-rw---- 1 root disk 8, 1 Oct 20 2009 /dev/sda1
```

Первичным разделам жестких дисков соответствуют миноры с первого по четвертый. Таким образом, например, для первичных разделов на первом SCSI-диске создаются файлы устройств `/dev/sda1`, `/dev/sda2`, `/dev/sda3` и `/dev/sda4`.

Логические разделы нумеруются, начиная с пяти. Причем даже если на диске нет всех четырех первичных разделов (один из них должен быть расширенным для создания логических разделов), то все равно первому логическому разделу будет соответствовать устройство с минором 5. Так, например, первому логическому разделу на Secondary Master IDE-диске будет соответствовать файл устройства `/dev/hdc5`.

В примере 11.3 приведен листинг, полученный при выполнении команды `sfdisk -l`, показывающий разделы на диске и их размер в цилиндрах и блоках.

Пример 11.3. Дисковые разделы

```
# sfdisk -l

Disk /dev/sda: 19457 cylinders, 255 heads, 63 sectors/track
Units = cylinders of 8225280 bytes, blocks of 1024 bytes, counting from 0

Device Boot Start      End    #cyls   #blocks  Id System
```

/dev/sda1	0+	1073-	1074-	8624128	27	Unknown
/dev/sda2	1073+	8854-	7782-	62501953+	7	HPFS/NTFS
/dev/sda3	*	8855	19456	10602	85160565	5 Extended
/dev/sda4	0	-	0	0	0	Empty
/dev/sda5	19083+	19456	374-	3004123+	82	Linux swap/Solaris
/dev/sda6	8855+	12510	3656-	29366757	83	Linux
/dev/sda7	12511+	19082	6572-	52789558+	83	Linux

На SATA-диске в примере 11.3 размещены три первичных раздела (/dev/sda1, ..., /dev/sda3), причем третий (/dev/sda3) раздел является расширенным (extended). В нем размещены три логических раздела (/dev/sda5, ..., /dev/sda7). Знаки "плюс" и "минус" сообщают о том, что имело место округление. Если вывести информацию в секторах, его не будет.

ЗАДАНИЯ

- С помощью команды `fdisk -l` изучите таблицу разделов вашего жесткого магнитного диска.
- Какая команда — `fdisk -l` или `sfdisk -l` — дает более подробный результат?
- На каком разделе вашего диска имеется область подкачки swap?

Создание разделов с использованием *fdisk*

Интерактивная утилита `fdisk` позволяет оперировать дисковыми разделами жестких магнитных дисков, предоставляя специальный набор собственных команд.

Имеется также команда `sfdisk`, которая, в отличие от `fdisk`, является утилитой для неинтерактивного редактирования таблицы разделов на жестком диске. При неосторожном ее использовании легко можно утратить все данные на жестком диске, т. к. данные для редактирования таблицы разделов задаются в командной строке `sfdisk`.

Весьма популярна программа `cfdisk`, которая позволяет редактировать таблицу разделов диска с помощью простого интерфейса меню.

Работать с командой `fdisk` может только администратор. Для редактирования таблицы разделов на диске эту команду следует запустить в интерактивном режиме, указав файл устройства для требуемого жесткого диска (пример 11.4).

Пример 11.4. Команда fdisk

```
# fdisk /dev/sda
```

The number of cylinders for this disk is set to 19077.

There is nothing wrong with that, but this is larger than 1024, and could in certain setups cause problems with:

- 1) software that runs at boot time (e.g., old versions of LILO)
- 2) booting and partitioning software from other OSs (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help): m

Command action

- a toggle a bootable flag
- b edit bsd disklabel
- c toggle the dos compatibility flag
- d delete a partition
- l list known partition types
- m print this menu
- n add a new partition
- o create a new empty DOS partition table
- p print the partition table
- q quit without saving changes
- s create a new empty Sun disklabel
- t change a partition's system id
- u change display/entry units
- v verify the partition table
- w write table to disk and exit
- x extra functionality (experts only)

В примере 11.4 команда `fdisk` была выполнена с аргументом — файлом устройства первого SATA-диска. Далее была выполнена встроенная команда `m`, отобразившая список встроенных команд `fdisk`. Список основных команд утилиты `fdisk`:

- ☐ `q` — завершение работы без сохранения изменений;
- ☐ `l` — вывод списка возможных типов разделов;
- ☐ `d` — удаление раздела (для удаления будет запрошен номер удаляемого раздела);

- ❑ `n` — создание нового раздела;
- ❑ `t` — установка типа вновь созданного раздела (для установки типа необходимо ввести номер типа раздела);
- ❑ `a` — выбор активного раздела;
- ❑ `w` — запись измененной таблицы разделов.

Команда `p` утилиты `fdisk` выводит таблицу разделов на диске (пример 11.5).

Пример 11.5. Получение таблицы разделов

Command (m for help): `p`

```
Disk /dev/sda: 20.0 GB, 20003880960 bytes
64 heads, 32 sectors/track, 19077 cylinders
Units = cylinders of 2048 * 512 = 1048576 bytes
Disk /dev/sda: 20.0 GB, 20003880960 bytes
64 heads, 32 sectors/track, 19077 cylinders
Units = cylinders of 2048 * 512 = 1048576 bytes
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sda1		1	19077	19534832	83	Linux

Command (m for help): `q`

Обычная последовательность работы с утилитой `fdisk` при создании нового раздела на жестком диске такова:

1. Выводится список существующих разделов на диске — команда `p`.
2. Удаляются ненужные разделы — команда `d`.
3. Создается новый раздел — команда `n`. При этом будут запрошены: тип раздела (`p` — первичный, `e` — расширенный, `l` — логический), номер раздела, первый и последний цилиндры раздела. Последний цилиндр нового раздела можно указать абсолютно или, после знака `+`, как размер раздела в килобайтах (например, `+15000k`) или мегабайтах (`+1200m`).
4. Если новый раздел должен иметь тип, отличный от принятого по умолчанию (83 — Linux Native), то необходимо указать тип раздела — команда `t` (получение списка возможных типов разделов — команда `l`).
5. Сохранение изменений — команда `w`.

Если диск, на котором редактировалась таблица разделов, содержит корневую файловую систему в одном из разделов, то потребуется перезагрузка.

Задания

- В интерактивном режиме команды `fdisk` получите информацию о таблице разделов на вашем жестком диске.
- Определите код типа раздела для физического тома системы Linux LVM (Logical Volume Manager).
- Какие могут быть созданы типы разделов FAT, каковы их идентификаторы?
- Получите от инструктора информацию о разделе, который требуется создать на жестком диске, и создайте его с помощью команды `fdisk`.

ВНИМАНИЕ!

При ошибочных действиях все данные на диске могут быть утрачены! При создании раздела на диске, где размещена корневая файловая система, потребуется перезагрузка после создания раздела.

Создание файловой системы

Для возможности работы с разделом накопителя на магнитных дисках в разделе должна быть создана файловая система, т. е. должно быть произведено форматирование раздела. При этом в новой файловой системе формируется суперблок, создается массив индексных дескрипторов и выделяется пространство для блоков данных. Стандартной файловой системой в GNU/Linux является `ext2`. Однако помимо `ext2` широко используются следующие файловые системы:

- ❑ `ext3` — современная версия `ext2` с поддержкой журналирования, индексированием каталогов и с улучшенными показателями быстродействия;
- ❑ `ext4` — дальнейшее развитие `ext3` с возможностью создавать очень большие файловые системы (1 экзбайт = 10^{18} байт), ориентированная на работу с экстендами (непрерывная последовательность блоков, принадлежащих файлу) с улучшенными параметрами надежности журналирования;
- ❑ `XFS` — высокопроизводительная файловая система для очень больших объемов хранимой информации (8 экзбайт), разработанная в Silicon Graphics;
- ❑ `JFS` — высокопроизводительная файловая система от IBM, используемая при необходимости хранения больших объемов информации (32 петабайта = 32×10^{15} байт).

В GNU/Linux реализована поддержка и других файловых систем, однако основные используемые файловые системы: `ext2`, `ext3` и `ext4`, две последние

из которых поддерживают *журналирование* (journaling). Это значит, что на диске выделяется место для хранения информации о том, какие операции и с какими блоками в файловой системе выполнялись в последнее время.

Наличие журналирования позволяет быстрее и надежнее обеспечить восстановление файловой системы после сбоя. Операции записи на диск в журналируемых файловых системах оформлены в виде транзакций, которые либо завершаются целиком, либо не принимаются вовсе. Использование журналирования существенно повышает целостность данных и уменьшает вероятность потери данных при сбое.

Для создания файловой системы применяется команда `mkfs`, с аргументом — файлом устройства, соответствующим разделу жесткого диска. Команда `mkfs` по умолчанию создает файловую систему `ext2`. Так, для создания файловой системы `ext2` на втором первичном разделе SATA жесткого диска следует выполнить такую команду от имени суперпользователя (пример 11.6).

Пример 11.6. Создание файловой системы `ext2`

```
# mkfs /dev/sda2
```

При успешном выполнении этой команды на экран будет выведена информация о размере блоков и их количестве в данной файловой системе, о местоположении копий суперблока и пр.

Для создания иной файловой системы ее тип следует указать после опции `-t` команды `mkfs`. Например, для создания на том же разделе диска файловой системы `ext4` надо выполнить команду, приведенную в примере 11.7.

Пример 11.7. Создание файловой системы `ext4`

```
# mkfs -t ext4 /dev/hda2
```

Устройство различных файловых систем значительно отличается, поэтому для создания конкретной файловой системы используется специализированная утилита:

- ❑ `mke2fs` — для создания файловых систем `ext2`, `ext3` и `ext4`;
- ❑ `mkfs.xfs` — для создания XFS;
- ❑ `mkdosfs` — для создания файловой системы FAT.

Команда `mkfs` является удобной оболочкой, которая для создания конкретных типов файловых систем обращается к следующим командам:

- ☐ `/sbin/mkfs.ext2;`
- ☐ `/sbin/mkfs.ext3;`
- ☐ `/sbin/mkfs.ext4;`
- ☐ `/sbin/mkfs.xfs;`
- ☐ `/sbin/mkfs.msdos.`

Эти команды, в свою очередь, являются символическими ссылками либо же жесткими связями со специализированными утилитами, упомянутыми ранее (пример 11.8).

Пример 11.8. Команды для создания файловых систем

```
$ ls -l /sbin/mkfs.*
-rwxr-xr-x 4 root root 55268 Oct 19 21:47 /sbin/mkfs.ext2
-rwxr-xr-x 4 root root 55268 Oct 19 21:47 /sbin/mkfs.ext3
-rwxr-xr-x 4 root root 55268 Oct 19 21:47 /sbin/mkfs.ext4
lrwxrwxrwx 1 root root      7 Nov 30 18:28 /sbin/mkfs.msdos -> mkdosfs
-rwxr-xr-x 2 root root 174296 Oct 19 21:59 /sbin/mkfs.reiserfs
lrwxrwxrwx 1 root root      7 Nov 30 18:28 /sbin/mkfs.vfat -> mkdosfs

$ find /sbin -samefile /sbin/mkfs.ext4
/sbin/mkfs.ext2
/sbin/mke2fs
/sbin/mkfs.ext3
/sbin/mkfs.ext4
```

В примере 11.8 с помощью команды `find` продемонстрировано, что `mke2fs`, `mkfs.ext2`, `mkfs.ext3` и `mkfs.ext4` являются одним и тем же файлом.

Команда `mke2fs` обладает опцией `-c`, которая заставляет команду перед созданием файловой системы проверять поверхность диска на наличие плохих блоков. Опция `-b` позволяет выбирать размер блока.

Задания

- Создайте новый раздел в свободном пространстве жесткого диска.
- Создайте на этом разделе файловую систему `ext3` с предварительной проверкой поверхности диска на плохие блоки. Файловая система должна иметь размер блока 2048 байтов.
- Определите, где находится первая копия суперблока в только что созданной файловой системе.

Проверка целостности файловой системы

Целостность файловой системы может быть нарушена в результате сбоя питания или же аппаратной неисправности. Если работа операционной системы была прервана и файловая система не была отмонтирована, то в суперблоке файловой системы остается специальный флаг (*dirty* — грязный или *not clean*), который сообщает о том, что в файловой системе возможны нарушения. Если же файловая система была отмонтирована правильно, то файловая система находится в состоянии *clean* — "чистая".

Сбой файловой системы обычно сопровождается тем, что информация, находящаяся в кэше для данной файловой системы, не попадает на диск, т. е. не синхронизируется. Может произойти также и следующее: операция синхронизации кэша с диском может быть не доведена до конца вследствие сбоя и запись данных из буфера на диск может прерваться до ее завершения. Это способно привести к целому ряду проблем:

- ☐ искажению или потере информации в блоках данных;
- ☐ появлению в системе блоков данных, которые считаются занятыми, хотя на них не указывает ни один индексный дескриптор;
- ☐ наличию перекрестных ссылок на блоки данных;
- ☐ появлению метаданных с ненулевым счетчиком ссылок, на которые не ссылаются никакие файлы;
- ☐ наличию противоречивых записей в каталогах

и т. п.

Различают два класса возможных нарушений в файловой системе:

- ☐ нарушение целостности данных;
- ☐ нарушение целостности файловой системы.

Для защиты целостности данных могут быть использованы разнообразные методы резервного хранения данных (*backup*). Утилиты восстановления целостности файловой системы не могут гарантировать восстановление данных после сбоя. Гарантом сохранности данных является только наличие резервных копий данных.

Для восстановления целостности файловых систем в GNU/Linux используется утилита *fsck* (пример 11.9), которая аналогично *mkfs* является надстройкой над специализированными утилитами для различных файловых систем.

ВНИМАНИЕ!

Проверка целостности файловой системы должна осуществляться лишь на отмонтированной файловой системе. Невыполнение этого требования может привести к полной потере данных в файловой системе!

Пример 11.9. Команда fsck

```
$ ls -l /sbin/fsck.*
-rwxr-xr-x 4 root root 185112 Oct 19 21:47 /sbin/fsck.ext2
-rwxr-xr-x 4 root root 185112 Oct 19 21:47 /sbin/fsck.ext3
-rwxr-xr-x 4 root root 185112 Oct 19 21:47 /sbin/fsck.ext4
lrwxrwxrwx 1 root root      7 Nov 30 18:28 /sbin/fsck.msdos -> dosfsck
-rwxr-xr-x 2 root root 330200 Oct 19 21:59 /sbin/fsck.reiserfs
lrwxrwxrwx 1 root root      7 Nov 30 18:28 /sbin/fsck.vfat -> dosfsck
```

Если вызвать утилиту `fsck` без опции `-t`, указывающей тип файловой системы, по умолчанию запустится `e2fsck`, проверяющая файловые системы `ext2`, `ext3` и `ext4`.

Пример 11.10. Проверка файловой системы ext3

```
# fsck -t ext3 /dev/sdb2
fsck from util-linux-ng 2.16
e2fsck 1.41.9 (22-Aug-2009)
/dev/sdb2: clean, 11357/2643840 files, 4355140/5277352 blocks
```

Эта команда (пример 11.10) проверит целостность файловой системы `ext3`.

При этом полная проверка осуществляется далеко не всегда, а только при условии:

- ☐ наличия флага `dirty` в суперблоке, что бывает при сбое или выключении питания без отмонтирования файловой системы;
- ☐ достижения максимального разрешенного количества монтирований файловой системы без проверки ее целостности;
- ☐ достижения максимального срока без проверки целостности файловой системы.

Если же необходимо выполнить полную проверку файловой системы в отсутствие любого из приведенных выше условий, то надо использовать опцию `-f` команды `e2fsck`. А для тестирования поверхности диска перед проверкой файловой системы требуется указывать опцию `-c` команды `e2fsck`. Например, приведенная в примере 11.11 команда выполняет полную проверку файловой системы, т. к. превышено максимально разрешенное для данной файловой системы число монтирований без проверки целостности файловой системы.

Пример 11.11. Полная проверка файловой системы ext3

```
# fsck -t ext3 /dev/sdb2
fsck from util-linux-ng 2.16
e2fsck 1.41.9 (22-Aug-2009)
USBDISK has been mounted 27 times without being checked, check forced.
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information

/dev/sdb2: ***** FILE SYSTEM WAS MODIFIED *****
/dev/sdb2: 11357/2643840 files (0.5% non-contiguous), 4355140/5277352
blocks
```

При работе команды `e2fsck` используется специальный каталог `lost+found`, находящийся в корневом каталоге файловой системы устройства. В этом каталоге сохраняются конвертированные в файлы потерянные блоки данных.

ЗАДАНИЯ

- Проверьте целостность файловой системы, созданной в предыдущем разделе.
- Выполните команду проверки этой же файловой системы так, чтобы была выполнена ее полная проверка (найдите подходящую опцию в `man`).

Монтирование файловых систем

В GNU/Linux все файловые системы на блочных устройства сведены в единую файловую систему посредством каталогов — *точек монтирования*.

Процесс подключения файловой системы на блочном устройстве к корневой файловой системе называется *монтированием*. При этом корнем файловой системы на смонтированном устройстве становится точка монтирования — каталог единой корневой файловой системы GNU/Linux.

За исключением файловых систем, для которых имеются специальные настройки в файле `/etc/fstab`, монтирование файловых систем производится суперпользователем.

Стандарт FHS предписывает, что временные точки монтирования файловых систем должны находиться в каталоге `/mnt`, а точки монтирования подклю-

чаемых файловых систем на съемных носителях — в каталоге `/media`. Например, каталог `/media/cdrom` может быть точкой монтирования для CD-ROM.

Команда `mount` позволяет смонтировать файловую систему указанного с помощью опции `-t` типа (по умолчанию `ext2`) в каталог — точку монтирования. Первый аргумент команды `mount` — файл блочного устройства, на котором находится монтируемая файловая система. Второй аргумент — точка монтирования этой файловой системы. Например, для монтирования файловой системы на устройстве `/dev/sdb2` можно выполнить команду, приведенную в примере 11.12.

Пример 11.12. Монтирование файловой системы

```
# mount /dev/sdb2 /mnt
```

Здесь файловая система на устройстве `/dev/sdb2` смонтирована в каталог `/mnt`. На устройстве `/dev/sdb2` файловая система `ext3`, но в нормальном режиме работы Linux опцию `-t` с указанием типа файловой системы указывать не обязательно.

Если же команда `mount` вызвана без аргументов, то она показывает список смонтированных файловых систем, т. е. имена файлов устройств и соответствующих им точек монтирования (пример 11.13).

Пример 11.13. Получение списка смонтированных устройств

```
# mount
/dev/sda6 on / type ext3 (rw,acl,user_xattr)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
debugfs on /sys/kernel/debug type debugfs (rw)
udev on /dev type tmpfs (rw)
devpts on /dev/pts type devpts (rw,mode=0620,gid=5)
/dev/sda7 on /home type ext3 (rw,acl,user_xattr)
/dev/sda2 on /windows/C type fuseblk
(rw,noexec,nosuid,nodev,allow_other,default_permissions,blksize=4096)
fusectl on /sys/fs/fuse/connections type fusectl (rw)
securityfs on /sys/kernel/security type securityfs (rw)
gvfs-fuse-daemon on /home/aberes/.gvfs type fuse.gvfs-fuse-daemon
(rw,nosuid,nodev,user=aberes)
rpc_pipefs on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
/dev/sdb2 on /mnt type ext3 (rw)
```

На дисках CD и DVD чаще всего бывает файловая система iso9660 (пример 11.14). Их надо монтировать с опцией `-r` для монтирования в режиме только для чтения (read only).

Пример 11.14. Монтирование CD-ROM

```
# mount -t iso9660 -r /dev/cdrom /media/cdrom
```

Монтирование файловой системы подменяет индексный дескриптор каталога — точки монтирования. До монтирования индексный дескриптор соответствует каталогу, находящемуся в файловой системе, к которой монтируется устройство. После монтирования индексный дескриптор каталога — точки монтирования принадлежит уже смонтированной файловой системе (пример 11.15).

Пример 11.15. Подмена индексного дескриптора при монтировании

```
# ls -ldi /media/usbdisk
160574 drwxr-xr-x 2 root root    48 Nov 20   2006 /media/usbdisk
# mount /dev/sda1 /media/usbdisk
# ls -ldi /media/usbdisk
2 drwxrwxrwx  9 root users  2048 Nov 25 19:43 /media/usbdisk
```

Из примера 11.15 заметно, что до монтирования временной файловой системы в каталог `/media/usbdisk`, индексный дескриптор каталога был 160574, а после монтирования — 2. У всех каталогов — точек монтирования файловых систем индексный дескриптор имеет номер 2 (пример 11.16).

Пример 11.16. Индексные дескрипторы точек монтирования

```
$ ls -id / /home /media/disk /media/disk-1
2 /  2 /home  2 /media/disk  1 /media/disk-1
```

В этом примере демонстрируется, что для каждой файловой системы номер индексного дескриптора точки монтирования равен 2.

Отмонтировать файловые системы (без специальных настроек в `/etc/fstab`) имеет право только суперпользователь. Для отмонтирования файловой системы применяется команда `umount`, которой в качестве аргумента должен быть задан единственный аргумент — либо точка монтирования, либо файл устройства (пример 11.17).

Пример 11.17. Отмонтирование файловой системы

```
# ls /media/usbdisk/
Books                               DelphinCHIK.JPG
# umount /media/usbdisk/
# ls /media/usbdisk/
```

В этом примере показана работа команды `umount`. После отмонтирования в точке монтирования больше нет доступа к файлам отмонтированной файловой системы.

Хорошим правилом является следующее: не рекомендуется хранить какие-либо файлы в каталогах, являющихся точками монтирования файловых систем.

Задания

- Смонтируйте файловую систему, созданную в предыдущих разделах в `/mnt`.
- Какой каталог находится в новой файловой системе?
- Перейдите в каталог — точку монтирования. Попытайтесь отмонтировать файловую систему. Что при этом происходит?
- С помощью `fuser` узнайте, какой процесс использует файл устройства.
- Что надо сделать для отмонтирования устройства?

Работа с разделом подкачки

Раздел или файл подкачки необходим при работе GNU/Linux для обеспечения временного перемещения страниц памяти из ОЗУ в этот раздел или файл. Такое перемещение бывает крайне необходимо при недостатке физической памяти. При этом страницы памяти, временно не используемые, но, тем не менее, необходимые для работы операционной системы или приложений, временно перемещаются в раздел подкачки. Процесс обмена страницами памяти между ОЗУ и разделом подкачки называется *paging*, а *раздел подкачки* называется *swap-разделом*. Понятия *swapping* и *paging* отличаются. Понятие *swapping* обозначает полное перемещение образа процесса из ОЗУ в раздел подкачки.

Получить информацию об использовании раздела подкачки можно с помощью команды `swapon -s` (пример 11.18).

Пример 11.18. Получение информации о разделе подкачки

```
$ /sbin/swapon -s
```

Filename	Type	Size	Used	Priority
/dev/sda5	partition	3004112	632	-1

Информация, полученная от команды `swapon -s`, демонстрирует следующее: имеется раздел подкачки в первом логическом разделе. Размер раздела — 3 Гбайт, из них использовано в настоящий момент — 632 Кбайт. Столбец приоритета отображает порядок использования `swap`-разделов. Сначала будут использованы разделы подкачки с наивысшим приоритетом, затем — с меньшим.

При необходимости можно добавить в систему дополнительные области подкачки. Они могут быть размещены как в разделах дисков, так и в обычных файлах. Создавать, подключать и отключать разделы подкачки имеет право суперпользователь. Если в системе не хватает ОЗУ для выполнения каких-либо приложений, то создание раздела подкачки может быть единственным методом решения этой проблемы.

При наличии в системе нескольких жестких дисков разумно разместить разделы подкачки на всех или нескольких из них.

Для создания файла подкачки необходимо создать файл, заполненный нулями. Приведенная в примере 11.19 команда создает файл, заполненный 128 Мбайт нулей.

Пример 11.19. Создание файла, заполненного нулями

```
$ dd if=/dev/zero of=swap.file bs=1k count=131072
```

```
131072+0 входных записей
```

```
131072+0 выходных записей
```

```
$ ls -l swap.file
```

```
-rw-r--r-- 1 user1 user1 134217728 Nov 19 17:43 swap.file
```

Команда `mkswap` создает в файле или разделе (указанном при помощи файла устройства) область подкачки, специальным образом размечая ее (пример 11.20).

Пример 11.20. Создание файла подкачки

```
$ /sbin/mkswap swap.file
```

```
Setting up swapspace version 1, size = 134213 kB
```

Эта команда создала в файле `swap.file` пространство подкачки.

Создать раздел подкачки в разделе жесткого диска можно лишь тогда, когда тип раздела установлен равным Linux Swap (82 тип в команде `fdisk`). Для создания раздела подкачки в разделе выполняется та же команда `mkswap` (пример 11.21).

Пример 11.21. Создание раздела подкачки

```
# mkswap -c /dev/sda2
```

В примере 11.22 создается раздел подкачки. При этом используется опция `-c`, которая перед созданием области подкачки проверяет диск на наличие плохих блоков.

Подключить созданный раздел или файл подкачки можно с помощью команды `swapon`, а отключить — с помощью команды `swapoff` (пример 11.22).

Пример 11.22. Подключение и отключение файла подкачки

```
# swapon ~user1/swap.file
# swapon -s
```

Filename	Type	Size	Used	Priority
/dev/sda5	partition	3004112	0	-1
/home/user1/swap.file	file	131064	0	-2

```
# swapoff ~user1/swap.file
# swapon -s
```

Filename	Type	Size	Used	Priority
/dev/sda5	partition	3004112	0	-1

В примере 11.22 был подключен дополнительный файл подкачки с помощью команды `swapon`, а затем этот файл был отключен командой `swapoff`.

Задания

- Создайте файл подкачки `swap1` размером 256 Мбайт.
- Подключите созданный файл подкачки.
- Проверьте, используется ли он.
- Используя команду `free`, получите информацию о доступной памяти в системе.

Файл информации о файловых системах /etc/fstab

При установке GNU/Linux создается файл /etc/fstab, содержащий информацию о файловых системах, автоматически монтируемых при загрузке или по требованию пользователя. Информация в файле /etc/fstab представлена в виде таблицы (пример 11.23).

Пример 11.23. Файл /etc/fstab

#<device>	<mount point>	<type>	<options>	<dump>	<pass>
/dev/sda5	swap	swap	defaults	0	0
/dev/sda6	/	ext3	defaults	1	1
/dev/sda7	/home	ext3	defaults	1	2
proc	/proc	proc	defaults	0	0
sysfs	/sys	sysfs	noauto	0	0
debugfs	/sys/kernel/debug	debugfs	noauto	0	0
usbfs	/proc/bus/usb	usbfs	noauto	0	0
devpts	/dev/pts	devpts	mode=0620,gid=5	0	0

Строки, начинающиеся с символа #, являются комментариями. Таблица, содержащаяся в файле /etc/fstab, состоит из колонок полей, назначение которых следующее:

- ❑ поле <device> содержит имя файла монтируемого устройства;
- ❑ поле <mount point> указывает каталог — точку монтирования;
- ❑ поле <type> содержит тип файловой системы на данном носителе;
- ❑ поле <options> содержит опции команды mount, которые должны быть использованы при монтировании данной файловой системы;
- ❑ поле <dump> указывает, надо ли для данной файловой системы производить автоматическое резервное копирование (backup) командой dump. Если в этом поле находится 1, то резервное копирование разрешено, если 0 — нет;
- ❑ поле <pass> предназначено для порядка проверки целостности файловых систем при загрузке операционной системы. Для корневой файловой системы в этом поле должно быть значение 1. Для других файловых систем,

которые необходимо проверять при загрузке, следует указать 2. Если проверка не требуется, то 0.

Далее приведена табл. 11.1, содержащая часто используемые опции команды `mount`, указываемые в поле `<options>` файла `/etc/fstab`.

Таблица 11.1. Опции монтирования в `/etc/fstab`

Опция	Назначение
<code>defaults</code>	Установки: <code>rw</code> , <code>suid</code> , <code>dev</code> , <code>exec</code> , <code>auto</code> , <code>nouser</code> , <code>asynch</code>
<code>asynch</code>	Асинхронный режим ввода/вывода
<code>auto</code>	Монтировать во время загрузки
<code>noauto</code>	Не монтировать во время загрузки
<code>exec</code>	Разрешение выполнения файлов с машинным кодом
<code>noexec</code>	Запрет выполнения файлов с машинным кодом
<code>suid</code>	Бит SUID устанавливать можно
<code>nosuid</code>	Запрещена установка битов SUID
<code>user</code>	Обычный пользователь может монтировать устройство
<code>nouser</code>	Монтировать разрешено только суперпользователю
<code>ro</code>	Режим только для чтения
<code>rw</code>	Разрешено как чтение, так и запись

Наличие записи в файле `/etc/fstab` означает, что данная файловая система может быть смонтирована с помощью единственного аргумента командной строки `mount`. Для монтирования файловой системы, указанной в `/etc/fstab`, достаточно задать либо точку монтирования, либо файл устройства.

Задания

- Подходит ли какая-либо запись в файле `/etc/fstab`, показанном в примере 11.24, для монтирования CD-ROM? Если нет, то какая запись в `/etc/fstab` должна быть?
- Изучите, для чего применяются опции `user`, `users` и `owner` команды `mount`?

Мониторинг дисковых ресурсов

Команда `df` выводит количество свободного места в блоках на специфицированном устройстве, а если оно не указано, то на всех смонтированных файловых системах. При этом удобно использовать опцию `-h` (пример 11.24) для отображения информации в удобных для пользователя единицах (`human readable format`).

Пример 11.24. Определение свободного пространства на дисках

```
$ df -h
```

Файловая система	Разм	Исп	Дост	Исп%	смонтирована на
/dev/sda3	9,5G	4,2G	4,9G	47%	/
tmpfs	504M	0	504M	0%	/dev/shm
/dev/sda7	11G	2,6G	7,1G	27%	/home
/dev/sdb2	20G	5,9G	13G	31%	/media/disk
/dev/sdb1	7,8G	4,8G	3,1G	62%	/media/disk-1
/dev/sda8	45G	36G	9,0G	80%	/home/disk

Для получения информации о наличии свободных индексных дескрипторов необходимо вызвать команду `df -i` (пример 11.25).

Пример 11.25. Получение количества свободных индексных дескрипторов

```
$ df -i
```

Файловая система	Инодов	Испол	Своб	Исп %	смонтирована на
/dev/sda3	2560864	178864	2382000	7%	/
tmpfs	128832	1	128831	1%	/dev/shm
/dev/sda7	2736384	10484	2725900	1%	/home
/dev/sdb2	2643840	956	2642884	1%	/media/disk
/dev/sdb1	0	0	0	-	/media/disk-1
/dev/sda8	0	0	0	-	/home/disk

Для того чтобы узнать, сколько пространства занимают файлы в каталоге, следует использовать команду `du`, отображающую количество блоков, занимаемых каждым каталогом и каждым его подкаталогом. Можно использовать `du -h` для отображения информации в удобных единицах. Также можно отобразить лишь суммарную информацию о каталоге. Для этого применяется `du -s` (пример 11.26).

Пример 11.26. Определение суммарного занятого пространства

```
$ du -sh ~  
467M    /home/user1
```

В этом примере получена информация о суммарном пространстве, используемом домашним каталогом пользователя.

Задания

- Выведите информацию об использовании дискового пространства на всех смонтированных блочных устройствах в системе.
- Получите информацию о суммарном размере файлов в домашнем каталоге.



Глава 12

Резервное копирование

Залог сохранности данных — это регулярное и неукоснительное выполнение резервного копирования данных. В этой главе читатель познакомится с основами создания политики резервного копирования, а также изучит основные программы для архивирования и сжатия данных.

Планирование резервного копирования

При планировании резервного копирования надо определить следующее:

- ☐ какие данные должны быть скопированы и где они находятся (в каких каталогах или файловых системах а также, возможно, на каких компьютерах);
- ☐ какой тип носителей данных будет использован для резервного копирования;
- ☐ с какой периодичностью будет производиться копирование данных;
- ☐ как будет осуществляться ротация резервных носителей данных;
- ☐ кто уполномочен производить резервное копирование;
- ☐ можно ли выполнять резервное копирование полностью автоматически;
- ☐ в какие моменты времени должно осуществляться резервное копирование;
- ☐ где и как будет производиться хранение носителей резервных копий;
- ☐ каков будет порядок восстановления утраченных данных из резервных копий;
- ☐ каково допустимое время, необходимое для восстановления данных.

В процессе принятия решения об организации резервного копирования надо учесть частоту изменения данных в файловой системе. В большинстве систем

содержимое следующих каталогов меняется относительно редко после установки системы:

- ☐ /boot — файлы, необходимые для загрузки ядра Linux;
- ☐ /etc — конфигурационные файлы, база данных паролей и сценарии инициализации системы и запуска демонов;
- ☐ /lib — основные жизненно важные библиотеки;
- ☐ /opt — дополнительное программное обеспечение;
- ☐ /bin и /sbin — исполняемые файлы, жизненно важные для системы;
- ☐ /usr — вторичная иерархия файловой системы, содержащая статические файлы.

Большинство каталогов с программным обеспечением и библиотеками устанавливается с дистрибутивных носителей, поэтому, при условии их сохранности, резервное копирование этих каталогов совсем не обязательно за редкими исключениями. Такими исключениями, например, могут явиться каталоги /usr/local и /opt, в которые может быть установлено программное обеспечение не из дистрибутивного комплекта.

В настоящее время наиболее распространены следующие типы носителей:

- ☐ магнитные ленты, для которых используются ленточные накопители (streamers);
- ☐ CD оптические диски — емкость до 700 Мбайт на один диск;
- ☐ DVD оптические диски — емкость до 4,7 Гбайт на один диск.

Применение того или иного вида аппаратуры для резервного копирования определяется такими факторами, как:

- ☐ стоимость аппаратуры и носителей;
- ☐ количество информации, которое может быть скопировано на носитель;
- ☐ возможность повторной записи и допустимое количество циклов записи/чтения;
- ☐ скорость копирования данных на носитель и считывания с него;
- ☐ возможность автоматической смены носителей в случае заполнения носителя.

Резервное копирование данных должно производиться тогда, когда эти данные не изменяются. В противном случае нельзя гарантировать сохранности этих данных. Желательно регулярно выполнять резервное копирование в часы минимальной нагрузки на систему. Обычно минимальная нагрузка имеет место в диапазоне от 2 часов ночи до 5 утра (в зависимости от специфики работы системы).

Резервное копирование может производиться в интерактивном режиме, требующем присутствия человека (*attended backup*). Чаще всего такой режим копирования применяется при каких-либо незапланированных ситуациях.

Другой вид резервного копирования — неинтерактивный (*unattended backup*). Неинтерактивное резервное копирование идеально подходит для заранее запланированных регулярных процедур копирования, осуществляемых в нерабочее время.

Хорошо, если резервное копирование производится достаточно быстро, однако гораздо важнее, чтобы быстро осуществлялось восстановление данных, поскольку обычно резервное копирование есть процедура регулярная, а восстановление данных требуется в случае сбоя системы, который заранее предсказать невозможно. Суммарное время на восстановление данных состоит как минимум из:

- ☐ времени, необходимого для поиска и доставки носителя резервной копии;
- ☐ времени на поиск требуемой информации на носителе (если поиск допускается);
- ☐ времени копирования информации с резервного носителя.

Если в архивной копии находится полная копия файловой системы, то время восстановления с нее может быть недопустимо большим.

Поэтому различают два типа резервного копирования:

- ☐ *полное*, при котором в архив помещается полная копия файловой системы;
- ☐ *инкрементальное*, при котором архив разбивается на несколько частей, в которых хранятся копии файлов, изменившихся с момента последнего копирования.

В GNU/Linux для инкрементального резервного копирования файловых систем *ext2*, *ext3* и *ext4* применяется утилита *dump*.

Задания

- Найдите утилиту, позволяющую управлять накопителем на магнитной ленте. *Подсказка:* воспользуйтесь контекстной помощью по страницам *man*-системы.
- Определите, какой файл устройства соответствует магнитной ленте.

Команда *dd*

Команда *dd* осуществляет низкоуровневое поблочное копирование из файла, указанного в командной строке после префикса *if=*, в файл, указанный после *of=*. По умолчанию используются блоки размером 512 байтов (один сектор).

Обычно команда `dd` не применяется для резервного копирования файловых систем, однако она с успехом может быть использована для создания образов файловых систем и для восстановления файловых систем из образов. Так, для поблочного копирования содержимого CD в файл можно использовать команду из примера 12.1.

Пример 12.1. Команда `dd`

```
dd if=/dev/cdrom of=image.img bs=1k
```

В этом примере все содержимое CD блоками размером 1 Кбайт (аргумент `bs=1k`) копируется в файл `image.img`.

Файловые системы можно копировать с помощью `dd` в размонтированном состоянии.

Если аргументы с префиксами `if=` и `of=` отсутствуют, то команда `dd` копирует поток ввода в поток вывода.

Кроме того, часто используются следующие опции команды `dd`:

- ☐ `count` — количество блоков, которое должно быть скопировано;
- ☐ `skip` — количество блоков во входном файле, которое надо пропустить;
- ☐ `seek` — количество блоков в выходном файле, которое надо пропустить.

Задания

- Скопируйте первые 512 байтов из файла устройства загрузочного жесткого диска в файл `first_sect.img` и определите тип его содержимого.
- Получите шестнадцатеричный дамп первых четырех байтов файла `first_sect.img`.

Утилиты для сжатия файлов

В UNIX- и GNU/Linux-системах команды архивирования отделены от утилит сжатия файлов. Все утилиты сжатия осуществляют компрессию файлов, указанных в качестве аргументов. При этом к исходным названиям файлов добавляются стандартные расширения, перечисленные далее, а права доступа и владения сохраняются.

В GNU/Linux используются следующие утилиты сжатия:

- ☐ `gzip` — применяется наиболее часто, сжатые файлы имеют расширения `.gz`;
- ☐ `bzip2` — часто обеспечивает лучшую степень сжатия, чем `gzip`, сжатые файлы имеют расширения `.bz2`;

- ❑ `compress` — стандартная UNIX-утилита сжатия, в GNU/Linux используется реже, чем предыдущие, сжатые файлы имеют расширение `.Z`.

Для сжатия файлов достаточно указать их в качестве аргументов (пример 12.2).

Пример 12.2. Сжатие файлов командой `gzip`

```
$ ls -l distfiles.*
-rw-r--r--  1 user1  user1    239263 Nov 26 23:10 distfiles.lst
-rw-r--r--  1 user1  user1    1967881 Nov 22 15:01 distfiles.txt
$ gzip distfiles.*
$ ls -l distfiles.*
-rw-r--r--  1 user1  user1     63298 Nov 26 23:10 distfiles.lst.gz
-rw-r--r--  1 user1  user1    187311 Nov 22 15:01 distfiles.txt.gz
```

В этом примере командой `gzip` были сжаты два файла. После сжатия к их названиям был добавлен суффикс `.gz`.

Далее приведены часто используемые опции команды `gzip`:

- ❑ `-d` — декомпрессия сжатых файлов, как `gunzip`;
- ❑ `-c` — вывести сжатое содержимое файлов в поток вывода, без изменения файлов;
- ❑ `-r` — сжать содержимое каталога рекурсивно;
- ❑ `-s` — установить иной, чем `.gz`, суффикс;
- ❑ `-t` — тестировать содержимое архива;
- ❑ `-v` — подробный вывод информации о работе команды;
- ❑ `-l` — вывести информацию об уровне сжатия файлов.

Декомпрессию сжатых `gzip`-файлов выполняет команда `gunzip` (пример 12.3).

Пример 12.3. Распаковка сжатых файлов

```
$ gunzip -v distfiles.*
distfiles.lst.gz:      73.6% -- replaced with distfiles.lst
distfiles.txt.gz:     90.5% -- replaced with distfiles.txt
```

Команда `zcat` распаковывает и выводит в поток содержимое сжатых `gzip`-файлов.

Что касается команды `bzip2`, то в ней реализован иной алгоритм сжатия, часто обеспечивающий более высокий уровень компрессии (пример 12.4). Многие опции команды `bzip2` функционально идентичны соответствующим опциям `gzip`.

Пример 12.4. Сжатие файлов командой `bzip2`

```
$ bzip2 -v distfiles.*
distfiles.lst:  3.879:1, 2.063 bits/byte, 74.22% saved, 239263 in, 61689 out.
distfiles.txt: 13.576:1, 0.549 bits/byte, 93.14% saved, 1967881 in, 135009 out.
$ ls -l distfiles.*
-rw-r--r--    1 user1   user1      61689 Nov 26 23:10 distfiles.lst.bz2
-rw-r--r--    1 user1   user1     135009 Nov 22 15:01 distfiles.txt.bz2
```

Если сравнить результаты работы `bzip2` с `gzip`, то заметно, что степень сжатия у первой команды выше, правда, команда `bzip2` работает чуть медленнее.

ЗАДАНИЯ

- Создайте пустой файл и сожмите его `gzip`. Уменьшился ли размер этого файла?
- Получите шестнадцатеричный дамп этого файла.
- Получите в домашнем каталоге сжатые `gzip`- и `bzip2`-копии файла `/bin/mount`. Исходный файл не должен быть изменен. Сравните результаты сжатия.
- Проверьте целостность сжатых файлов.
- Как с помощью `bzip2` сделать сжатую копию некоторого файла с расширением, характерным для `gzip`?

Команда `tar`

Один из наиболее часто используемых инструментов резервного копирования — команда `tar` (tape archive). Аргументы команды `tar` — это файлы и каталоги, которые должны быть помещены в архив. Имя архива указывают после опции `f` команды. Команда в примере 12.5 поместит каталог `/home` в архив на магнитной ленте.

Пример 12.5. Создание архива `tar`

```
# tar cvf /dev/st0 /home
```

Опция `c` команды `tar` предназначена для создания архива (`create`), а опция `v` заставляет команду `tar` выводить информацию об обрабатываемых файлах.

Команда `tar` позволяет создавать архивы, сжимая их при создании утилитами `gzip` с опцией `z` или `bzip2` с опцией `j`. Могут быть использованы и другие утилиты сжатия. В примере 12.6 каталоги `/bin` и `/sbin` будут помещены в архив `binaries.tar.gz`.

Пример 12.6. Создание сжатого с помощью `gzip` архива `tar`

```
$ tar czvf binaries.tar.gz /{,s}bin
```

Стандартным расширением для архивов `tar` является `.tar`. Если архив сжат, то к имени архива принято добавлять соответствующий суффикс (например, `.gz`).

Для просмотра содержимого архива следует использовать опцию `t` (`type`). Если архив сжат, то необходимо установить соответствующую опцию (пример 12.7).

Пример 12.7. Просмотр содержимого `tar`-архива, сжатого с помощью `gzip`

```
$ tar tzvf binaries.tar.gz
```

Для извлечения файлов из архива следует использовать опцию `x` (`extract`). При использовании GNU-версии команды `tar` содержимое архива будет извлечено в текущий каталог. При этом будут созданы все подкаталоги, которые находятся в архиве. В примере 12.8 содержимое архива `binaries.tar.gz` извлекается в текущий каталог.

Пример 12.8. Извлечение содержимого архива `tar`

```
$ tar xzvf binaries.tar.gz
```

Если вместо имени файла архива после опции `f` в командной строке `tar` указан дефис (`-`), то архив будет принят из стандартного потока ввода. Так, приведенная в примере 12.9 команда по действию аналогична предыдущей.

Пример 12.9. Получение архива из стандартного потока ввода

```
$ zcat binaries.tar.gz | tar xvf -
```

Часто используемые опции GNU-версии команды `tar` приведены в списке:

- ☐ `-A` — добавление файлов `tar`-архива в существующий архив (слияние);
- ☐ `-c` — сравнение содержимого архива с заданным каталогом;
- ☐ `--delete` — удаление файлов из архива;
- ☐ `-r` — добавление файлов в конец архива;
- ☐ `-u` — обновление архива версиями файлов, более новыми, чем в архиве;
- ☐ `-b` — указывает размер блока ($n \times 512$ байтов);
- ☐ `-C` — изменение каталога;
- ☐ `-h` — разыменовывать символические ссылки, т. е. сохранять в архиве не файлы символических ссылок, а файлы, на которые они указывают;
- ☐ `-l` — при создании архива оставаться в пределах текущей файловой системы и не переходить в смонтированные к подкаталогам файловые системы;
- ☐ `-L` — указать длину ленты ($n \times 512$ байтов);
- ☐ `-m` — не восстанавливать дату модификации файлов при извлечении их из архива;
- ☐ `-p` — сохранять при восстановлении файлов оригинальные права владения и права доступа к ним;
- ☐ `-M` — указывает, что архив состоит из нескольких томов;
- ☐ `-P` — сохранять в архиве файлы с абсолютными именами;
- ☐ `-N` — помещать в архив только те файлы, которые были созданы или изменены после специфицированной даты;
- ☐ `-O` — разархивировать файлы в стандартный поток вывода;
- ☐ `-t` — взять имена файлов для извлечения из архива или помещения в архив из заданного после опции файла;
- ☐ `-Z` — использовать утилиту сжатия `compress`.

Задания

- От имени суперпользователя создайте в его домашнем каталоге полный архив файлов в каталоге `/etc`. Архив должен называться `etc.tar.gz`. При создании архива в нем должны быть записаны файлы с их полными именами.
- Получите содержимое архива для просмотра списка файлов в нем.
- Извлеките из созданного архива единственный файл `/etc/hosts` так, чтобы он был сохранен в каталоге `/tmp` с тем же именем.
- Разделите созданный архив на части, размером по 50 Кбайт, используя для этого утилиту `split`. Каким образом из полученных частей собрать целый архив?

Команда `cpio`

Команда `cpio` является вторым по значимости инструментом архивирования после `tar`. Особенность `cpio` в том, что имена файлов для архивирования она принимает через стандартный поток ввода.

Команда `cpio` способна работать в трех режимах, определяемых опциями:

- ❑ `-o` — для копирования в архив (`copy-out`), в котором архивируемые файлы помещаются в архив, а сам поток байтов архива копируется в выходной (`output`) файл;
- ❑ `-i` — для копирования из архива (`copy-in`), в котором файлы извлекаются из архива, который передается команде на вход (`input`);
- ❑ `-p` — проходной режим (`pass-through`), при использовании которого файлы копируются из одного каталога в другой без реального создания архива.

Чаще всего список файлов, которые должны быть помещены в архив, передается на стандартный поток ввода команды `cpio` с помощью команды `find`.

В примере 12.10 в архив будут помещены только те файлы в домашнем каталоге, которые были изменены за последние сутки (режим `copy-out`).

Пример 12.10. Создание архива `cpio`

```
$ find ~tania -mtime 1 | cpio -ov > /tmp/tania.cpio
```

Эта команда найдет все файлы в домашнем каталоге пользователя `tania`, измененные в течение последних 24 часов, передаст их список в стандартный поток ввода утилиты `cpio`, которая поместит их в файл архива `/tmp/tania.cpio`. Опция `-v` используется для получения подробной информации о процессе архивирования.

Для передачи в стандартный поток вывода информации о файлах в архиве используется режим `copy-in` команды `cpio` с опцией `-t` (`type`) — пример 12.11.

Пример 12.11. Получение содержимого архива `cpio`

```
$ cpio -ivt < /tmp/tania.cpio
```

В этом примере содержимое архива было считано командой `cpio` из стандартного потока ввода. Команда `cpio` работала в режиме `copy-in` (опция `-i`).

Извлечь файлы из архива можно с помощью такой же команды, но без опции `-t`. При этом обычно устанавливают опцию `-d`, вынуждающую команду `cpio` создавать отсутствующие каталоги при восстановлении файлов.

Эта команда извлечет в текущий каталог файлы из `tania.cpio` (пример 12.12).

Пример 12.12. Извлечение файлов из архива `cpio`

```
$ cpio -ivd < /tmp/tania.cpio
```

В проходном режиме команды `cpio`, который работает при установленной опции `-p`, архив не создается. Файлы, имена которых передаются команде `cpio` через стандартный поток ввода, рекурсивно копируются в целевой каталог.

Далее приводится список других часто используемых опций `cpio`:

- ☐ `-b` — устанавливает размер блока 5120 байтов вместо 512 байтов по умолчанию;
- ☐ `-F` — указывает имя файла для ввода или вывода архива;
- ☐ `-o` — указывает имя файла для записи архива;
- ☐ `-I` — указывает имя файла для считывания архива;
- ☐ `-A` — добавление файлов к архиву, указанному после опции `-F` или `-o`;
- ☐ `-E` — извлекать из архива файлы, имена которых удовлетворяют шаблону;
- ☐ `-f` — не копировать файлы, имена которых удовлетворяют заданному шаблону;
- ☐ `-n` — не переводить `UID` и `GID` в имена пользователей и групп;
- ☐ `-r` — интерактивно переименовывать файлы;
- ☐ `-m` — сохранять дату модификации файлов;
- ☐ `-u` — безусловно заменять новыми существующие файлы.

Задания

- Создайте `cpio`-архив всех файлов устройств, принадлежащих вам.
- Выведите содержимое этого архива на экран.
- Извлеките из архива в домашний каталог файлы, в имени которых есть `fd`.
- Найдите опцию `cpio`, которая при создании архива позволяет не изменять время доступа к файлам, помещаемым в архив.

Команда *pax*

Архиватор *pax* (Portable Archive eXchanger) способен работать с архивами разнообразных форматов (в том числе *tar* и *cpio*). Он работает в трех режимах:

- ☐ извлечения файлов из архива (опция *-r*);
- ☐ записи файлов в архив (опция *-w*);
- ☐ копирования (одновременно опции *-rw*).

Пример 12.13. Создание *pax*-архива

```
$ pax -wv -f Tania.pax Tania
Tania
Tania/Pl.ppt
Tania/insect.ppt
Tania/P.ppt
pax: ustar vol 1, 4 files, 0 bytes read, 32409600 bytes written.
$ file Tania.pax
Tania.pax: tar archive
```

Команда *pax -wv -f Tania.pax Tania*, показанная в примере 12.13, создала архив *Tania.pax*, содержащий каталог *Tania*. Опция *-v* была использована для отображения процесса создания архива (*verbose*). Созданный архив *Tania.pax* имеет формат *tar*.

При необходимости установки иного формата можно воспользоваться опцией *-x* (например, *-x cpio*). Также можно использовать компрессию с помощью *gzip*. Для включения компрессии предназначена опция *-z*.

Вывести содержимое архива можно командой, приведенной в примере 12.14.

Пример 12.14. Получение содержимого архива

```
$ pax -v -f Tania.pax
drwxrwxr-x  2 user1  user1          0 Dec  3  Tania
-r-xr-xr-x  1 user1  user1    2641408 Mar  8  Tania/Pl.ppt
-rwxr-xr-x  1 user1  user1    27053568 Dec  3  Tania/insect.ppt
-r-xr-xr-x  1 user1  user1    2702336 Mar  8  Tania/P.ppt
pax: ustar vol 1, 4 files, 32409600 bytes read, 0 bytes written.
```

Извлечение файлов из архива достигается при указании опции `-r`. При этом можно использовать опцию `-s`, позволяющую изменять имена извлекаемых из архива файлов подобно команде `s` потокового редактора `sed` (пример 12.15).

Пример 12.15. Извлечение файлов из архива с изменением имен

```
$ cd /var/tmp/  
$ ls  
$ pax -r -s '/Tania/./' -f ~/Tania.pax 2> /dev/null  
$ ls  
insect.ppt  P1.ppt  P.ppt
```

Команда, выполненная в примере 12.15, извлекла в текущий каталог файлы из архива. При этом в именах извлекаемых файлов каталог `Tania` был заменен точкой. Поэтому файлы, содержащиеся в подкаталоге `Tania`, были извлечены в текущий каталог.

Команду `pax` можно применять для рекурсивного копирования в проходном режиме (пример 12.16).

Пример 12.16. Копирование каталогов с помощью проходного режима `pax`

```
# pax -rwv -pe ~user1/Tania .  
./home/user1/Tania  
./home/user1/Tania/P1.ppt  
./home/user1/Tania/insect.ppt  
./home/user1/Tania/P.ppt  
# ls -l home/user1/Tania/  
total 31680  
-rwxr-xr-x 1 user1 user1 27053568 Dec  3  insect.ppt  
-r-xr-xr-x 1 user1 user1  2641408 Mar  8  P1.ppt  
-r-xr-xr-x 1 user1 user1  2702336 Mar  8  P.ppt
```

В этом примере суперпользователь рекурсивно скопировал в текущий каталог содержимое каталога `/home/user1/Tania`. Так как были использованы опции `-pe` (`preserve everything`), права на файлы и даты модификации изменены не были.

Задания

- Создайте архив `pax` в формате `cpio` из файлов вашего домашнего каталога, модифицированных за последние сутки.
- Получите содержимое архива.
- Извлеките файлы из архива, заменив в пути к ним строку `home` на `emo`.

Программы *dump* и *restore*

Программа `dump` предназначена для выполнения инкрементального архивирования файловых систем `ext2`, `ext3` и `ext4`. Полный архив включает в себя копии всех файлов файловой системы, а инкрементальный — только те файлы, дата изменения которых позже некоторой заданной даты. Определение этой даты достигается с помощью установки уровней резервного копирования, которые задают моменты отсчета, начиная с которых проверяется актуальность файлов.

Полное копирование называется *копированием нулевого уровня*. При выполнении резервного копирования нулевого уровня все файлы помещаются в архив.

Для всех последующих уровней резервного копирования справедливо правило: в архив уровня N помещаются все файлы, созданные или изменившиеся за время, прошедшее с момента последнего копирования уровня M , меньше, чем N ($M < N$).

Например, в архив второго уровня попадут все файлы, изменившиеся с момента последнего копирования нулевого или первого уровня.

Программа `dump` предлагает десять уровней копирования с 0 — полный архив до 9.

В общем виде командная строка `dump` выглядит следующим образом:

```
dump <-уровень> [опции] <файлы>
```

где *-уровень* — уровень резервного копирования, например `-0` — полное; *файлы* — либо файл устройства копируемой файловой системы, либо точка монтирования ее, либо файлы для полного архивирования.

Наиболее важные опции команды `dump`:

- ❑ `-f файл` — файл устройства или обычный файл, куда будет помещен архив;
- ❑ `-z` — выполнять компрессию `gzip`;
- ❑ `-L метка` — задать метку тома;
- ❑ `-u` — сделать запись о резервном копировании в файл `/etc/dumpdates`.

Далее приведен пример использования команды `dump` для получения полного архива файловой системы `/home`, который записывается на магнитную ленту (пример 12.17).

Пример 12.17. Создание полного архива `dump`

```
dump -0u -f /dev/st0 /home
```

Эта команда выполнит полное резервное копирование (уровень 0) на магнитную ленту, записав информацию об этом в файл `/etc/dumpdates` (опция `-u`).

В примере 12.18 создается архив второго уровня файловой системы `/dev/sda3` с меткой `Level2`.

Пример 12.18. Пример создания инкрементального архива второго уровня

```
dump -2u -L 'Level2' -f /dev/st0 /dev/sda3
```

Команда `restore` позволяет извлечь файлы из архива, созданного командой `dump`. Имеется возможность извлечь все или некоторые файлы из архива.

При использовании команды `restore` обязательно должен быть указан режим работы с помощью одной из опций:

- ☐ `-i` — включение интерактивного режима работы команды;
- ☐ `-r` — режим неинтерактивного восстановления всех файлов из архива;
- ☐ `-t` — вывод содержимого архива;
- ☐ `-x` — восстановление файлов с возможностью селективного выбора файлов.

С помощью опции `-f` команды `restore` указывают файл архива. Например, имеется полный архив на магнитной ленте. Необходимо его восстановить (пример 12.19).

Пример 12.19. Восстановление из полного архива

```
restore -rf /dev/st0
```

Опция `-r` задает режим восстановления, опция `-f` указывает файл с архивом.

Опция `-i` задает интерактивный режим работы команды `restore`. В этом режиме `restore` предоставляет встроенную командную строку, в которой могут быть выполнены следующие команды:

- ☐ `add` — пометить указанный каталог или файл для восстановления;
- ☐ `cd` — сменить текущий каталог;
- ☐ `delete` — удалить указанный файл или каталог из списка на восстановление;
- ☐ `extract` — восстановить файлы;
- ☐ `help` — отобразить помощь по встроенным командам;
- ☐ `ls` — вывести список файлов;
- ☐ `pwd` — вывести полный путь к текущему каталогу;

- ❑ `quit` — выйти из программы, даже если список на восстановление не пуст;
- ❑ `setmodes` — установить права доступа и даты на восстанавливаемые файлы;
- ❑ `verbose` — включить режим подробного информирования.

Другие важные опции команды `restore`:

- ❑ `-a` — подавить выдачу вопроса о номере тома, содержащего требуемые для восстановления файлы (в режимах `-x` и `-i`);
- ❑ `-m` — работа с многотомным архивом (см. опцию `-m` команды `dump`);
- ❑ `-v` — работа с многотомными неленточными архивами (например, CD-ROM);
- ❑ `-N` — симуляция восстановления файлов;
- ❑ `-o` — восстанавливать права доступа к файлам без выдачи запросов.

Далее приведен пример 12.20 проверки наличия в архиве некоторого файла. Должно быть введено полное имя файла так же, как оно сохранено в архиве.

Пример 12.20. Восстановление заданного файла

```
$ /sbin/restore -x -ao -f proceed.0.dump    home/aberes/8marta/iptraf.sh
$ ls
home  proceed.0.dump
$ ls -R home/
home/:
aberes
home/aberes:
8marta
home/aberes/8marta:
iptraf.sh
```

Опция `-x` позволяет селективно извлечь файл из архива. Опция `-a` отменяет выдачу запроса о номере тома, а `-o` автоматически без запроса устанавливает права доступа к восстанавливаемым файлам. После восстановления в текущем каталоге образовалось дерево подкаталогов, содержащее восстановленный файл.

Задания

- Определите, какие файловые системы смонтированы. Сделайте полный архив одной из файловых систем.
- Измените любой файл в этой файловой системе и выполните резервное копирование первого уровня.
- Откройте командой `restore` архив первого уровня в интерактивном режиме и восстановите файл, содержащийся в нем.



Глава 13

Запуск, останов GNU/Linux и уровни выполнения

В этой главе дается описание уровней исполнения системы. Вы узнаете, как происходит инициализация GNU/Linux и запуск сервисов. Вы научитесь правильно выключать и перезагружать систему.

Инициализация операционной системы и переход на заданный уровень исполнения

Нормальный ход загрузки GNU/Linux предусматривает запуск процесса `/sbin/init` после завершения загрузки и инициализации ядра. Процесс `init` имеет `PID = 1` и запускает разные системные службы. Процесс `init` также порождает серверы терминалов `getty` (или им подобные), которые выводят на терминал приглашение `login:` и передают управление команде `/bin/login` при попытке входа в сеанс.

В GNU/Linux-системах имеется конфигурационный файл, задающий поведение процесса `init` при загрузке системы — это файл `/etc/inittab`. Для разновидностей GNU/Linux, следующих спецификации System V, определяется понятие *уровней исполнения* (runlevels). Уровень исполнения — это вариант работы операционной системы, определяющийся конкретным набором программ, запускающимся на нем.

В Red Hat, Novell SUSE и подобных ОС определены следующие уровни исполнения:

- ❑ 0 — останов системы (`halt`);
- ❑ 1 — административный однопользовательский режим (`single user mode`);
- ❑ 2 — многопользовательский режим без поддержки файловой системы NFS;
- ❑ 3 — стандартный многопользовательский режим (`multiuser`);

- ❑ 4 — не используется;
- ❑ 5 — многопользовательский режим с запуском X-сервера;
- ❑ 6 — перезагрузка (reboot).

Часто определяется уровень исполнения для однопользовательского режима — S.

В некоторых GNU/Linux-системах используется система инициализации, следующая стилю BSD или подобная ему. Например, в Slackware используется BSD-подобная система инициализации, где определены только останов, перезагрузка системы, однопользовательский режим и многопользовательский режим, а уровней исполнения в обычном понимании нет. В Gentoo Linux применяется оригинальная система инициализации, напоминающая BSD, на которой определены такие режимы работы, как boot и default, а обычных уровней исполнения нет.

Дальнейшее изложение относится лишь к системам GNU/Linux, следующим System V-спецификации процесса инициализации операционной системы.

Процесс `init` запускает процессы инициализации системы, основываясь на конфигурационных командах в файле `/etc/inittab`. Формат записей в файле `/etc/inittab` следующий: `id:runlevels:action:process`, где:

- ❑ `id` — уникальный идентификатор записи в файле `/etc/inittab`, состоящий из 1—4 символов (для старых систем — не более 2-х);
- ❑ `runlevels` — уровни исполнения, на которых выполняется действие;
- ❑ `action` — действие, которое должно быть выполнено;
- ❑ `process` — процесс, который должен быть запущен.

Далее приводится список распространенных действий в `/etc/inittab`:

- ❑ `initdefault` — строка определяет уровень исполнения по умолчанию;
- ❑ `sysinit` — процесс должен быть запущен единожды при инициализации операционной системы до запуска любых других процессов (не считая, естественно, самого процесса `init`);
- ❑ `wait` — процесс должен быть запущен при переходе на данный уровень исполнения, причем процесс `init`, породивший данный процесс, должен дожидаться завершения этого процесса перед переходом к следующим действиям;
- ❑ `ctrlaltdel` — процесс, который должен быть запущен при нажатии в системной консоли комбинации клавиш `<Ctrl>+<Alt>+`;
- ❑ `powerfail` — задает процесс, который должен быть запущен при получении от системы управления питанием сигнала об отсутствии питания;

- ❑ `powerokwait` — задает процесс, запускаемый `init` при возобновлении питания;
- ❑ `respawn` — процесс должен быть запущен заново, если он остановился.

Обсудим важнейшие фрагменты файла `/etc/inittab`.

Сначала рассмотрим пример 13.1.

Пример 13.1. Уровень исполнения по умолчанию

```
id:5:initdefault:
```

Эта строка определяет уровень исполнения по умолчанию — здесь 5-й, т. е. многопользовательский с запуском X-сервера.

В строке, приведенной в примере 13.2, задается выполнение сценария `/etc/rc.d/rc.sysinit` при инициализации системы перед запуском других процессов.

Пример 13.2. Скрипт первичной инициализации системы

```
si::sysinit:/etc/rc.d/rc.sysinit
```

Блок настроек, показанных в примере 13.3, определяет поведение системы при переходе на заданный уровень исполнения.

Пример 13.3. Скрипты перехода на заданный уровень исполнения

```
10:0:wait:/etc/rc.d/rc 0
11:1:wait:/etc/rc.d/rc 1
12:2:wait:/etc/rc.d/rc 2
13:3:wait:/etc/rc.d/rc 3
14:4:wait:/etc/rc.d/rc 4
15:5:wait:/etc/rc.d/rc 5
16:6:wait:/etc/rc.d/rc 6
```

Идея проста: при переходе на уровень исполнения с заданным номером (0—6) происходит вызов сценария `/etc/rc.d/rc`, которому в качестве аргумента подставляется цифра — номер требуемого уровня исполнения. Уровень исполнения, на котором работает каждая строка, задается во втором поле записей. Дальнейшая инициализация, выполняемая на данном уровне исполнения, производится сценарием `/etc/rc.d/rc`.

В примере 13.4 показана строка настройки для запуска программы `getty`, выводящей приглашение на вход в сеанс `login`: и ожидающей попытки входа.

Пример 13.4. Запуск сервера терминала

```
1:2345:respawn:/sbin/mingetty tty1
```

Данная строка распространяется на уровни исполнения со 2-го по 5-й и не действует на однопользовательский режим. Обратите внимание, что метка представлена одной цифрой, совпадающей с номером терминала, на котором запущена программа `mingetty`. Фактически количество доступных виртуальных консолей в системе определяется количеством этих строк в файле `/etc/inittab`.

Если в файле `/etc/inittab` произведены изменения, то они вступят в силу либо после перезагрузки системы, либо после команды `/sbin/init q`. Еще один способ сделать это — передать процессу `init` (PID = 1) сигнал `SIGHUP`.

Итак, при переходе на требуемый уровень исполнения в Red Hat срабатывает сценарий `/etc/rc.d/rc`, деятельность которого ограничивается запуском сценариев в каталогах с именами вида: `/etc/rc.d/rc#.d`.

В системах System V каталога `/etc/rc.d` нет и имена каталогов инициализации имеют вид: `/etc/rc#.d`, где вместо знака решетки подставляется номер требуемого уровня исполнения, например, каталог `/etc/rc2.d` соответствует 2-му уровню исполнения.

Если выполнить команду `ls -l /etc/rc3.d`, то можно убедиться, что содержащиеся в нем файлы представляют собой символические ссылки на файлы инициализационных сценариев, находящиеся в каталоге `/etc/init.d`.

Имена файлов — символических ссылок формируются следующим образом:

1. Сначала устанавливается либо буква `s` для сценариев, запускающих некоторую службу, или буква `k` для сценариев, останавливающих какую-либо службу (все имена файлов, начинающихся с чего-либо другого, игнорируются).
2. Далее следует порядковый номер запуска сценария, предназначенный для упорядочивания процесса запуска и останова служб.
3. Завершается имя ссылки строкой — именем (условным) службы.

Итак, все ссылки, имена которых начинаются с `s` (`start`), запускают что-либо, а ссылки, имена которых начинаются с `k` (`kill`), — останавливают. Порядковые номера предотвращают несвоевременные попытки запуска каких-либо сервисов. Например, служба электронной почты запускается ссылкой `S80sendmail` после инициализации сети ссылкой `S10network`.

Например, для инициализации сетевых интерфейсов следует набрать команду, показанную в примере 13.5.

Пример 13.5. Запуск сценария инициализации сетевого интерфейса

```
/etc/init.d/network start
```

Сценарий `/etc/rc.d/rc`, перебирая файлы символических ссылок в каталогах инициализации, для всех символических ссылок, имена которых начинаются с `s`, добавляют через пробел к имени ссылки слово `start`, запуская так службы. А для ссылок с именами, начинающимися с `k`, — слово `stop`.

В Red Hat и подобных системах запуск и останов служб удобно выполнять с помощью команды `/sbin/service` (пример 13.6).

Пример 13.6. Запуск служб в Red Hat

```
service network start
```

Эта команда инициализирует сетевой интерфейс.

Таким образом, для того чтобы некоторая служба автоматически запускалась на заданном уровне исполнения, требуется иметь сценарий инициализации этой службы (желательно поместить его в стандартном каталоге `/etc/init.d`) и символические ссылки, размещаемые в каталогах `/etc/rc#.d`, где вместо решетки следует подставить требуемые уровни исполнения.

В Red Hat, SUSE и подобных системах удобно пользоваться командой `chkconfig`, позволяющей настроить поведение сценариев инициализации на разных уровнях исполнения. Приведенная в примере 13.7 команда отобразит все сценарии запуска служб и уровни выполнения, на которых они запускаются.

Пример 13.7. Получение таблицы запуска служб

```
chkconfig --list
```

Для добавления службы в список автоматического запуска надо вызвать команду `chkconfig` с опцией `--add`, а для удаления службы — с опцией `--del`. Для определения уровней исполнения, на которых будет работать требуемый сценарий, необходимо использовать опцию `--level`. Команда в примере 13.8 определяет, что сетевой интерфейс должен работать на 2, 3, 4 и 5-м уровнях исполнения.

Пример 13.8. Настройка запуска службы по уровням исполнения

```
chkconfig --level 2345 network on
```

В Ubuntu имеется аналогичная `chkconfig` утилита `sysv-rc-conf`, обладающая удобными интерактивными возможностями.

Команда `runlevel` показывает текущий уровень исполнения (пример 13.9).

Пример 13.9. Определение уровня исполнения

```
# runlevel
N 5
```

Команда `runlevel` выводит два значения — предыдущий уровень исполнения (в примере он неизвестен — `N`) и текущий (здесь — `5`).

Перейти на другой уровень исполнения можно с помощью команды `/sbin/init` (пример 13.10).

Пример 13.10. Переход в однопользовательский режим

```
/sbin/init 1
```

Задания

- Какая разновидность программы `getty` используется в вашей системе?
- Как командой `who` вывести уровень исполнения?
- Измените конфигурацию системы так, чтобы в системе запускалось на одну виртуальную консоль больше.
- Определите, запускается ли на текущем уровне исполнения служба `httpd`.

Остановка и перезагрузка системы

Для немедленного останова или перезагрузки системы можно использовать команды, соответственно, `/sbin/init 0` или `/sbin/init 6`. Однако для этого удобнее вызывать команды `halt` для останова или `reboot` для перезагрузки.

Команда `halt` вносит в файл `/var/log/wtmp` запись о том, что система была остановлена в это время. Далее для останова вызывается команда `shutdown -h now`, останавливающая систему. Этого не произойдет при использовании команды `halt` опцией `-f` (*force*), заставляющей систему остановиться без вызова `shutdown`.

Если команда `halt` вызвана с опцией `-n`, то перед остановом не будет произведена операция сброса содержимого кэша на диск. А при использовании команды `halt -d` кроме этого запись в файл `/var/log/wtmp` произведена не будет.

Остановка системы с последующим отключением питания будет произведена в результате выполнения команд `halt -p` или `poweroff`.

Обычно команды `poweroff` и `reboot` реализованы в виде символических ссылок на файл команды `/sbin/halt`.

Основной командой для безопасной остановки или перезагрузки системы является `/sbin/shutdown`. С ее помощью можно осуществлять как немедленную, так и отложенную остановку системы. Причем эта команда посылает пользователям предупреждение о том, что система останавливается. Процессам, работающим в этот момент, посылается сигнал `SIGTERM`, получив который приложения могут корректно завершить свою работу.

Команда `shutdown` посылает сигнал процессу `init` для перехода на 0-й или 6-й уровень исполнения при вызове с опцией, соответственно, `-h` или `-r` (пример 13.11).

Пример 13.11. Остановка системы

```
/sbin/shutdown -h now
```

Данная команда осуществит немедленную остановку системы, т. к. в качестве времени останова системы указан параметр `now`. Если же необходимо остановить или перезагрузить систему в заданное время, то его следует указать в качестве аргумента.

Пример 13.12. Планируемая перезагрузка

```
/sbin/shutdown -r 17:00 'System will be rebooted at 17:00!'
```

В этом примере перезагрузка системы будет произведена в 17:00, причем пользователи будут оповещены об этом с помощью строки сообщения.

Вместо использования точного указания времени можно указывать время задержки перед остановом (пример 13.13). Если задержка измеряется секундами, то количество секунд следует указать после опции `-t`.

Пример 13.13. Остановка системы через 10 минут

```
/sbin/shutdown -h +10
```

В данном случае (см. пример 13.13) останов будет выполнен через 10 минут. Реально между этими двумя вариантами задания задержки существует разница: после опции `-t` задается время задержки в секундах до того, как `shutdown` передаст сигнал `init` для перехода на другой уровень исполнения. Если же используется указание либо времени, либо задержки в минутах, то при этом реальное действие самой команды `shutdown` будет произведено с заданной задержкой. При этом пользователи, вошедшие в сеанс, могут продолжать работать до начала останова, но новые сеансы не будут открыты.

В табл. 13.1 указаны часто используемые опции команды `shutdown`.

Таблица 13.1. Опции команды `shutdown`

Опция	Назначение
<code>-c</code>	Отменить начавшийся останов системы
<code>-f</code>	Создает файл <code>/fastboot</code> , наличие которого позволяет не проверять файловую систему при загрузке
<code>-F</code>	Создает файл <code>/forcefsck</code> , наличие которого вынуждает проверять файловую систему при загрузке
<code>-h</code>	Остановка системы
<code>-r</code>	Перезагрузить систему
<code>-k</code>	Послать пользователям сообщение, но не останавливать систему

Задания

- Пошлите пользователям системы сообщение об останове системы средствами `shutdown` без выполнения реального останова.
- Будет ли создан файл `/for fsck` в таком же режиме работы команды `shutdown`, как в предыдущем задании, если команда была вызвана с опцией `-F`?



Глава 14

Загрузчики

В этой главе вы познакомитесь с процессом загрузки GNU/Linux и с двумя стандартными загрузчиками — LILO и GRUB.

Последовательность процесса загрузки

В BIOS компьютеров с архитектурой x86/64 входит программа, которая называется POST (Power On Self Test). Эта программа проверяет устройства компьютера при включении питания. После удачного завершения программы POST запускается другая программа в BIOS, которая называется Hardware Bootstrap Loader (аппаратный загрузчик). Аппаратный загрузчик обращается к загрузочному устройству, указанному в CMOS, для поиска на нем загрузочного сектора и загрузке его содержимого.

Если загрузка осуществляется с магнитного диска, то производится поиск программы загрузки в нулевом секторе. Для жестких магнитных дисков загрузочный сектор называют MBR (Master Boot Record, главная загрузочная запись). Аппаратный загрузчик опознает программный загрузчик по сигнатуре последних двух байтов загрузочного сектора (510 и 511 байты, начиная с 0 байта) — 0xAA55 и загружает его в ОЗУ. Если программный загрузчик в MBR отсутствует, то его поиск осуществляется в первом секторе активного раздела. Размер программного кода в загрузочном секторе меньше 512 байт, поэтому он не может выполнять сложных действий.

Стандартные программные загрузчики, применяемые в GNU/Linux, — GRUB и LILO. Они состоят из двух частей. Одна из них, называемая *начальным загрузчиком*, помещается либо в MBR, либо в первый сектор активного раздела. Задача начального загрузчика — загрузка второй части загрузчика. Вторая часть загрузчика — более сложная программа, которая может загрузить ядро Linux (или другое ядро) и выполнить необходимые действия для загрузки

операционной системы (например, распаковка электронного диска с примитивным образом корневой файловой системы).

Загрузки, используемые в GNU/Linux, способны передавать ядру Linux различные конфигурационные параметры и дополнительные команды.

После загрузки ядра и его инициализации производится запуск программы `/sbin/init` (если в параметрах, переданных ядру, не указано иное). Конфигурационный файл программы `/sbin/init` — `/etc/inittab`. Процесс `init` всегда имеет PID, равный 1. Эта программа осуществляет дальнейшую инициализацию операционной системы GNU/Linux, зависящую от выбранного режима работы (например, однопользовательский или многопользовательский режим).

В зависимости от выбранного уровня исполнения команда `init` исполняет тот или иной набор специальных командных файлов, называемых сценариями инициализации. Именно эти сценарии определяют, какие программы будут работать на том или ином уровне исполнения и какова будет функциональность системы.

Один из процессов, которые запускает `/sbin/init`, — сервер терминала `getty` или аналогичный. Программа `getty` выводит приглашение войти в сеанс `login:`. Введенное имя пользователя передается программе `login` как аргумент. Программа `login` выводит приглашение ввести пароль `password:`, который проверяется в базе данных паролей. Если аутентификация производится успешно, то запускается оболочка, указанная для пользователя в файле `/etc/passwd`. Перед выводом приглашения `login:` на экран печатается содержимое файла `/etc/issue` (или `/etc/issue.net`, если подключение осуществляется через сеть). А после входа в сеанс на экран выводится содержимое файла `/etc/motd` (Message Of The Day, сообщение дня).

Задания

- Отредактируйте содержимое файла `/etc/issue` и проверьте вывод на экран его содержимого, выйдя из сеанса.
- Отредактируйте файл `/etc/motd` и проверьте, выводится ли на экран его содержимое при входе в сеанс.

Загрузчик GRUB

В настоящее время загрузчик GRUB (Grand Unified Boot loader) является стандартным в GNU/Linux. Он позволяет загружать не только Linux, но и ядра множества других операционных систем. Загрузчик GRUB способен обращаться к файловой системе на загрузочном носителе для поиска файла ядра.

Загрузчик GRUB распознает различные типы файловых систем, например, ext2, ext3, ext4, XFS, JFS. GRUB способен работать с дисками, имеющими более 1024 цилиндров. При этом обеспечивается загрузка ядер ОС с разделов, расположенных в любом месте диска.

GRUB может загружать ядра разнообразных операционных систем, например, GNU/Linux, GNU/HURD, FreeBSD, SUN Solaris и др. Для операционных систем, загрузка ядер которых не поддерживается, загрузчик GRUB обеспечивает цепную загрузку (chainload), т. е. передачу управления загрузчику другой операционной системы. Также GRUB поддерживает автоматическую декомпрессию загружаемых файлов, сжатых в формате ZIP. GRUB обладает встроенной командной оболочкой, позволяющей гибко управлять процессом загрузки, а также автоматизировать его, задавая в конфигурационном файле загрузчика последовательности команд. Загрузчик GRUB поддерживает загрузку через локальную сеть посредством протокола TFTP (Trivial File Transfer Protocol) и выполняет управление загрузкой с удаленного последовательного терминала.

В GRUB используется особая система именования дисковых устройств. Для обращения к дискам используются обозначения вида: (hd1) — второй жесткий диск в системе (нумерация начинается с нуля), hd — жесткие диски. Имена дисков заключаются в круглые скобки. Раздел жесткого диска указывают в виде номера раздела через запятую после номера диска. Нумерация разделов начинается с нуля. Например, второй раздел первого жесткого диска именуется (hd0,1).

Команда `grub` в командной строке запускает собственную командную оболочку GRUB, в которой, например, можно определить структуру жесткого диска с помощью встроенной GRUB-команды `geometry`, которой необходимо указать в качестве аргумента имя жесткого диска (пример 14.1).

Пример 14.1. Получение геометрии диска

```
grub> geometry (hd0)
drive 0x80: C/H/S = 19457/255/63, The number of sectors = 312581808,
/dev/disk/by-id/ata-HITACHI_HTS541616J9SA00_SB2404GJGZV
WLS
Partition num: 0, Filesystem type unknown, partition type 0x27
Partition num: 1, Filesystem type unknown, partition type 0x7
Partition num: 4, Filesystem type unknown, partition type 0x82
Partition num: 5, Filesystem type is ext2fs, partition type 0x83
Partition num: 6, Filesystem type is ext2fs, partition type 0x83

grub> quit
```

Команда `help` предоставляет помощь в командной оболочке GRUB. Команда `root` предназначена для указания раздела жесткого диска, на котором размещается ядро ОС, т. е. загрузочного раздела. Например, для указания второго раздела первого жесткого диска следует выполнить команду `root (hd0,1)`.

Команда `setup` позволяет установить загрузчик, т. е. записать исполняемый код загрузчика в требуемый раздел жесткого диска или MBR (Master Boot Record). Аргумент команды `setup` должен указывать требуемый раздел. Например, для установки GRUB в MBR первого диска следует выполнить команду из примера 14.2.

Пример 14.2. Установка загрузчика GRUB

```
setup (hd0)
```

Для выхода из командной оболочки GRUB следует использовать команду `quit`.

В командной оболочке GRUB работает механизм продолжения имен команд, аналогичный такому же механизму в оболочке Bash. Для продолжения имени команды в GRUB следует нажать клавишу <Tab>.

При загрузке с помощью GRUB можно также использовать оболочку GRUB, например, для того, чтобы установить значения некоторых параметров загрузки ядра. Для перехода в оболочку GRUB при загрузке необходимо во время работы загрузчика нажать клавишу <c>. Для выхода из командной оболочки GRUB и продолжения загрузки следует нажать клавишу .

При работе загрузчик GRUB использует файлы, находящиеся в каталоге `/boot/grub`. Основной файл конфигурации загрузчика `/boot/grub/menu.lst` задает последовательности команд, выполняемых загрузчиком при загрузке ОС. В целях обратной совместимости может присутствовать файл символической ссылки `/boot/grub/grub.conf`, указывающий на основной файл конфигурации GRUB. В файле `menu.lst` может быть задано несколько различных вариантов ядер для загрузки и/или несколько вариантов различных параметров загрузки. Здесь также могут быть заданы команды передачи управления по цепи загрузчикам других операционных систем.

Команда `default` указывает образ для загрузки по умолчанию. Все образы нумеруются с нуля в порядке их появления в файле `menu.lst`. Задержка в секундах перед загрузкой образа по умолчанию задается командой `timeout`. При желании использовать графическое меню загрузки можно установить графическое фоновое изображение с помощью команды `splashimage` (пример 14.3). Описание образа для загрузки должно включать в себя команду `title`, отображающую в меню строку с именем или описанием загружаемого образа,

возможно, иной загрузочный раздел, нежели раздел по умолчанию, с помощью команды `root`, а также, возможно, имя образа начального электронного диска с первичной файловой системой `initrd`.

Пример 14.3. Файл конфигурации `/boot/grub/menu.lst`

```
default 2
timeout 8

title openSUSE 11.2 - 2.6.31.8-0.1 (pae)
    root (hd0,5)
    kernel /boot/vmlinuz-2.6.31.8-0.1-pae root=/dev/sda6
    initrd /boot/initrd-2.6.31.8-0.1-pae

title Windows
    rootnoverify (hd0,1)
    chainloader +1
```

В примере 14.3 показан файл конфигурации GRUB, позволяющий загружать ядро Linux и по цепной загрузке MS Windows.

Из собственной оболочки GRUB загрузчик может быть установлен с помощью команды оболочки `setup`. Например, `setup (hd0)` — установка в MBR.

При необходимости установить GRUB из оболочки Bash (или иной) следует использовать системную команду `grub-install`, передав ей в качестве параметра имя загрузочного раздела.

Если загрузчик GRUB установлен, то для дальнейшего изменения его поведения следует править файл `/boot/grub/menu.lst`, который при загрузке будет прочитан загрузчиком GRUB без каких-либо других подготовительных действий.

Задания

- Установите загрузчик GRUB в конфигурации, аналогичной конфигурации LILO из предыдущего параметра.
- Как можно использовать GRUB для передачи управления загрузчику LILO, если он установлен в первом разделе первого жесткого магнитного диска?

Загрузчик LILO

LILO еще недавно являлся стандартным загрузчиком GNU/Linux, но в настоящее время он отсутствует во многих современных дистрибутивах. Если меню-интерфейс LILO не используется, то приглашение загрузчика LILO для ввода идентификатора загружаемого образа системы выглядит, как в примере 14.4.

Пример 14.4. Приглашение загрузчика LILO

LILO:

Обычно это приглашение выводится в течение нескольких секунд, и если пользователь не ввел командную строку, то загружается образ GNU/Linux, установленный в конфигурации LILO по умолчанию. Этот же образ ядра будет загружен, если пользователь нажмет клавишу <Enter>. Нажатие клавиши <Tab> выведет список возможных вариантов загрузки и/или различных ядер Linux.

Помимо ввода имен образов для загрузки в строке приглашения LILO можно вводить дополнительные параметры загрузки и команды ядру. Если встречается параметр, который не опознается ядром, то этот параметр передается процессу init.

Например, если надо вместо /sbin/init после старта ядра запустить интерактивную оболочку, следует указать в приглашении LILO строку, показанную в примере 14.5.

Пример 14.5. Передача параметра ядру

LILO: Linux-2.6.31 init=/bin/sh

В этом случае стартовая инициализация системы производиться не будет. Вместо этого сразу же после запуска ядра будет запущена интерактивная оболочка. Такой режим, например, может быть использован для восстановления системы после сбоя.

Далее приведена табл. 14.1 с наиболее часто используемыми параметрами ядра.

Таблица 14.1. Параметры, передаваемые ядру

Параметр	Значение
root=...	Имя устройства с корневой файловой системой
mem=...	Объем ОЗУ
nosmp	Отмена поддержки SMP (Symmetric Multi Processors)
1	Загрузка в однопользовательский режим
ro	Монтирование файловой системы в режиме только для чтения

Настройки загрузчика LILO хранятся в файле /etc/lilo.conf (пример 14.6, табл. 14.2).

Пример 14.6. Файл /etc/lilo.conf

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
default=Linux-2.6.31
lba32
prompt
timeout=50
image=/boot/vmlinuz-2.6.31
    label=Linux-2.6.31
    root=/dev/sda6
    initrd=/boot/initrd-2.6.31.img
    read-only
other=/dev/sda2
    label=windows
    table=/dev/hda
```

В примере 14.6 корневая файловая система для ядра Linux расположена в разделе /dev/hda6. Помимо этого здесь можно загружать MS Windows.

Таблица 14.2. Конфигурационные директивы /etc/lilo.conf

Параметр	Значение
boot=...	Место расположения загрузочной записи
install=...	Указывает файл загрузчика для его установки
default=...	Образ для загрузки по умолчанию. Указывает метку (label) образа
label=...	Метка образа
lba32	Использование логической 32-разрядной адресации для загрузки
prompt	Включение интерактивного режима работы LILO
timeout=...	Количество десятых долей секунды для ожидания выбора
delay=...	Количество десятых долей секунды задержки
image=...	Имя файла ядра
root=...	Местонахождение корневой файловой системы
initrd=...	Имя файла образа исходной файловой системы
read-only	Монтирование файловой системы в режиме только для чтения
append=...	Передает параметр ядру. Например, append="mem=128М"
other	Для загрузки иной файловой системы

После изменения настроек LILO следует установить измененный экземпляр загрузочной записи. Это делается с помощью команды `/sbin/lilo`. Рекомендуется предварительно проверить правильность конфигурации с помощью команды `/sbin/lilo -t`, которая имитирует установку загрузчика LILO.

Важнейшие опции командной строки `/sbin/lilo`:

- ❑ `-R` — командная строка загрузчика, используемая при следующей загрузке;
- ❑ `-C` — использовать другой, чем `/etc/lilo.conf`, файл конфигурации;
- ❑ `-i` — указывает файл загрузочной записи (по умолчанию `/boot/boot.b`);
- ❑ `-v` — увеличивает уровень подробности сообщений;
- ❑ `-u` — восстанавливает предыдущий загрузчик (`uninstall`).

При работе загрузчика LILO могут возникать проблемы, о которых он сообщает с помощью системы кодов (табл. 14.3). Если на экране при загрузке появляется приглашение `LILO:` не полностью, а только частично, значит, возникла проблема.

Таблица 14.3. Коды сообщений LILO

Сообщение	Значение
Нет сообщения LILO	не загружен аппаратным загрузчиком
<code>L<код ошибки></code>	Вторая (многосекторная) часть не загружается
<code>LI</code>	Вторая часть LILO загружена, но не исполняется
<code>LIL</code>	Нет таблицы дескрипторов образов в <code>map</code> -файле
<code>LIL?</code>	Инициализация невозможна
<code>LIL-</code>	Таблица дескрипторов в <code>map</code> -файле испорчена
<code>LILO</code>	Обе части загрузчика выполнены успешно

Задания

- Используя LILO, запустите GNU/Linux в однопользовательском режиме. Для этого необходимо в командной строке загрузчика указать параметр `1`.
- В конфигурационном файле LILO создайте дополнительный раздел, позволяющий загружать GNU/Linux в однопользовательском режиме.
- Перед реальной установкой новой конфигурации LILO проверьте ее.



Глава 15

Отложенное и регулярное выполнение заданий

В этой главе вы узнаете, как выполнять команды в заданные моменты времени. В главе рассказано о том, как устанавливать одноразовое и регулярное исполнение команд. Понимание этого важно для автоматизации рутинных задач системного администрирования.

Отложенное выполнение заданий

Часто возникает необходимость выполнить какую-либо команду в заданный момент будущего или же в тот момент, когда загрузка системы минимальна. Команда `at` позволяет указать момент времени, в который должна быть исполнена команда. Команда `at` ставит задания в очередь, которая обслуживается демоном `atd`.

Заданная команда будет выполнена один раз в указанное время (пример 15.1).

Пример 15.1. Отложенное выполнение задания

```
$ date
Tue Jan 5 19:02:21 YEKT 2010
$ at 22:20
warning: commands will be executed using /bin/sh
at> ps -ef | mail -s 'At command' user1
at> <EOT>
job 1 at 2010-01-05 22:20
```

В 19:02 была запущена команда `at 22:20` (пример 15.1). Команда `at`, вызванная так, позволяет ввести команды, которые необходимо выполнить в будущем. В этом примере была введена команда, посылающая в заданное время

пользователю `user1` список процессов в подробном формате. Ввод команд завершается нажатием комбинации клавиш `<Ctrl>+<D>`.

Помимо указания времени выполнения команды можно задавать период времени, через который должна быть выполнена заданная команда (пример 15.2).

Пример 15.2. Установка времени выполнения относительно текущего момента

```
$ at now +10 minutes
```

В этом случае команда `at` выполнит команду, заданную в ее интерактивном сеансе, через десять минут относительно текущего момента времени.

Команды для исполнения можно передать команде `at` через стандартный поток ввода. Также можно опцией `-f` указать имя файла с командами. Если указана опция `-m`, то команда `at` посылает вызвавшему ее пользователю сообщение по электронной почте о том, что задание выполнено.

Получить список заданий можно, используя команду `atq` или `at -l` (пример 15.3).

Пример 15.3. Вывод списка заданий

```
$ atq
1          2010-01-05 22:20 a user1
```

При исполнении этой команды обычный пользователь получает список только его заданий, а суперпользователь — список заданий всех пользователей.

Для удаления задания из очереди следует вызвать команду `atrm` или `at -d`. Номер задания должен быть указан в качестве аргумента (пример 15.4).

Пример 15.4. Удаление задания

```
$ echo 'wall Hello!' | at -m +1 minutes
warning: commands will be executed using /bin/sh
job 2 at 2010-01-05 20:04
$ atq
2          2010-01-05 20:04 a user1
$ atrm 2
$ atq
```


В примере 15.4 была задана команда `wall`, которая должна была вывести сообщение "Hello!" на терминалы всех пользователей, находящихся в сеансе, через минуту после ее вызова. Однако эта команда снята с исполнения командой `atrm`.

Команда `batch` отличается от `at` тем, что она выполняет задание не через четко определенный промежуток времени, а в тот момент, когда средняя загрузка системы уменьшается до 0,8. Значение средней загрузки системы берется из файла `/proc/loadavg`. В остальном команда `batch` аналогична команде `at`.

В системе могут быть файлы `/etc/at.allow` и `/etc/at.deny`. Если существует файл `/etc/at.allow`, то только пользователи, перечисленные в нем, могут вызывать команду `at`. Если этого файла нет, то проверяется наличие файла `/etc/at.deny`, где указывают пользователей, которым запрещено вызывать `at`.

Задания

- Создайте файл, содержащий команду, позволяющую послать суперпользователю электронное письмо, содержащее сведения о средней загрузке системы.
- Установите на исполнение через 1 день команду в этом файле.
- Получите список заданий `at`.

Автоматизация выполнения регулярных задач

Утилита `cron` позволяет выполнять задание на регулярной основе с заданной периодичностью. В различных версиях GNU/Linux используются разные пакеты `cron`, работа которых отличается в деталях. Основой регулярного выполнения заданий `cron` являются таблицы, в которых кодируется периодичность выполнения заданий.

В `vixie-cron` имеется общесистемная таблица заданий `cron`, находящаяся в файле `/etc/crontab`, и индивидуальные таблицы периодических заданий пользователей, находящиеся в каталоге `/var/spool/cron`. Файл системной таблицы заданий `/etc/crontab` имеет семь полей (пример 15.5). Первые пять полей используются для указания периодичности выполнения задания. Шестое поле указывает пользователя, с правами которого должно быть исполнено задание. В седьмом поле задают саму команду.

Пример 15.5. Файл /etc/crontab

```
SHELL=/bin/sh
PATH=/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
25 6 * * * root run-parts /etc/cron.daily
47 6 * * 7 root run-parts /etc/cron.weekly
52 6 1 * * root run-parts /etc/cron.monthly
```

В таблице `crontab`, показанной в примере, установлены две переменные окружения — `SHELL` и `PATH`, и определены четыре задания. Первая из них указывает оболочку для запуска задания, а вторая определяет пути доступа к исполняемым командам. В примере 15.5 командой является `run-parts` с аргументом — именем каталога. Скрипт `run-parts` выполняет задания, расположенные в каталогах `/etc/cron.hourly` (ежечасно), `/etc/cron.daily` (ежедневно), `/etc/cron.weekly` (еженедельно) и `/etc/cron.monthly` (ежемесячно).

Формат задания времени следующий:

- ☐ минуты часа (0—59);
- ☐ час суток (0—23);
- ☐ календарное число (день месяца 1—31);
- ☐ месяц (1—12);
- ☐ день недели, начиная с воскресенья (0—6).

Так, например, задание, определенное в последней строке `crontab` из примера 15.5, будет запускаться в 6 часов 52 минуты первого числа каждого месяца.

Системную таблицу заданий имеет право изменять лишь суперпользователь. Обычные пользователи имеют право устанавливать свои собственные таблицы заданий. Таблицы заданий `cron` для обычных пользователей не имеют поля, в котором указывается имя пользователя, поэтому в них всего шесть полей.

Для манипулирования заданиями `cron` существует команда `crontab`. Если ей будет указан файл, содержащий таблицу заданий `cron`, то старая таблица `cron` будет стерта, а новая из файла будет сохранена (пример 15.6).

Пример 15.6. Создание таблицы `cron`

```
$ echo '30 21 * * * wall Hello!' > crontab.eg
$ crontab crontab.eg
```

В примере 15.6 был создан файл `crontab.eg`, содержащий единственную строку с заданием. Эта таблица была считана и установлена командой `crontab`.

Команда `crontab` с опцией `-l` выводит список заданий `cron` (пример 15.7).

Пример 15.7. Получение таблицы заданий `cron`

```
$ crontab -l
30 21 * * * wall Hello!
```

Обращение к таблицам пользователей возможно с помощью опции `-u` (пример 15.8).

Пример 15.8. Получение таблицы заданий конкретного пользователя

```
# crontab -l -u user1
30 21 * * * wall Hello!
```

Для редактирования текущей таблицы заданий в текстовом редакторе можно использовать команду `crontab -e`. При вызове этой команды будет запущен текстовый редактор по умолчанию, в нем будет открыта текущая таблица заданий `cron`, которую можно отредактировать. Другой вариант редактирования:

1. Таблица заданий сохраняется в файле: `crontab -l > crontab.eg`.
2. Временный файл редактируется: `vi crontab.eg`.
3. Отредактированная таблица заданий запоминается: `crontab crontab.eg`.

Для удаления таблицы заданий выполняют команду `crontab -r` (пример 15.9).

Пример 15.9. Удаление таблицы `cron`

```
$ crontab -r
$ crontab -l
crontab: no crontab for user1
```

Можно ограничивать доступ пользователей к команде `crontab` с помощью файлов `/etc/cron.allow` и `/etc/cron.deny`. Если существует файл `/etc/cron.allow`, то только пользователи, указанные в нем, имеют право применять команду `crontab`. В файле `/etc/cron.deny` указывают пользователей без права вызова `cron`.

Задания

- Ежедневно в 10:30 требуется получать список пользователей, входивших в сеанс в последнее время. Создайте соответствующую таблицу `cron`.
- В сеансе `root` выведите таблицу, которая создана в предыдущем пункте.



Глава 16

Системные журналы

Журналы — это важнейший источник информации о деятельности системы. Журналы предназначены для контроля деятельности системы и ее наладки. В этой главе читатель познакомится со службой `syslog` и программой ротации журналов.

Служба `syslog`

В GNU/Linux принято сохранять информацию разной степени детализации о процессе работы программ в специальных текстовых файлах, называемых *журналами*. Стандартное место расположения журналов — это каталог `/var/log`. Все журнальные файлы принадлежат к одной из двух категорий: системные журналы `syslog` и журналы прикладных программ. Не все журналы в `/var/log` обслуживаются `syslog`.

Служба `syslog` представлена демоном `syslogd`, запускаемым при старте операционной системы автоматически и работающим в фоновом режиме. Демон `syslogd` является сетевой службой и способен принимать сообщения из сети.

В GNU/Linux сообщения, поступающие от различных служб, обычно не записываются в один-единственный журнал. Напротив, принято называть файлы журналов так, чтобы по их имени можно было судить об источнике сообщений.

Конфигурационным файлом демона `syslogd` является `/etc/syslog.conf` (пример 16.1). Структура строк конфигурации, каждая из которых направляет некоторый поток сообщений в заданный файл (или на удаленный компьютер — сервер ведения журналов), представлена двумя полями:

- ❑ определение сообщения (`selector`) — поле, в котором указывают классы программ, сообщения от которых должны помещаться в данный поток;
- ❑ поле действия (`action`) указывает, куда должен быть записан поток сообщений.

Пример 16.1. Файл /etc/syslog.conf

#SELECTOR	ACTION
*.crit;lpr,cron,mail.none	/var/log/critical
daemon.info;daemon.!err	-/var/log/daemons
daemon.=err	/var/log/daemons.err
authpriv.*	/var/log/messages
kern.*;kern.!=info	/var/log/kernel
cron.info	-/var/log/cron
lpr.info	-/var/log/lpr
mail.warning	-/var/log/mail/mail

Из примера 16.1 заметно, что поле определения сообщения состоит из двух частей, разделенных точкой. Первая часть — источник сообщения (facility), а вторая — уровень важности (priority) сообщения. Эти две части уникально определяют все возможные сообщения, обрабатываемые syslog.

Служба syslog работает со следующими источниками сообщений (facility):

- ☐ auth — сообщения служб авторизации и безопасности;
- ☐ authpriv — сообщения служб авторизации и безопасности;
- ☐ cron — сообщения служб at и cron;
- ☐ daemon — сообщения, поступающие от демонов;
- ☐ kern — сообщения ядра;
- ☐ lpr — сообщения службы печати;
- ☐ mail — сообщения, поступающие от служб электронной почты;
- ☐ mark — источник для вставки служебных меток в журналы;
- ☐ news — сообщения службы новостей;
- ☐ security — то же, что и auth;
- ☐ syslog — собственные сообщения syslog;
- ☐ user — сообщения пользовательских программ;
- ☐ uucp — сообщения службы uucp;
- ☐ local0, ..., local7 — произвольно используемые источники.

Все сообщения разделены также по следующим уровням важности (priority), приведенным в порядке убывания важности:

- ☐ emerg — система не работоспособна (другое название — panic);
- ☐ alert — требуется немедленное вмешательство;

- ☐ `crit` — критическое событие;
- ☐ `err` — ошибка (другое название — `error`);
- ☐ `warning` — предупреждение (другое название — `warn`);
- ☐ `notice` — нормальное, но значимое событие;
- ☐ `info` — информационное сообщение;
- ☐ `debug` — отладочная информация.

Пара `источник.важность` устанавливает, что сообщения от этого источника с уровнями важности, *не меньшими указанного*, записываются в этот канал (пример 16.2).

Пример 16.2. Определение канала журналирования

```
lpr.info                -/var/log/lpr
```

При такой настройке (см. пример 16.2) все сообщения службы печати с уровнями важности `info` и выше будут записаны в файл `/var/log/lpr`. Знак "тире" перед именем файла разрешает асинхронную запись в журнал.

При необходимости запретить передачу в канал сообщений с уровнем важности не ниже указанного ставят знак восклицания перед уровнем важности (пример 16.3).

Пример 16.3. Исключение информации определенных уровней важности

```
daemon.info;daemon.!err    -/var/log/daemons
```

В этом случае (см. пример 16.3) все сообщения от демонов с уровнями важности от `info` до `warning` будут записываться в журнал `/var/log/daemons`, а сообщения с уровнями важности, начиная с `err`, записаны туда не будут.

Если же необходимо записывать в журнал сообщения только с одним уровнем важности, то перед уровнем важности ставят знак "равно" (пример 16.4).

Пример 16.4. Запись сообщений требуемого уровня важности

```
daemon.=err                /var/log/daemons.err
```

В примере 16.4 в журнал попадут лишь сообщения об ошибках и ничего более.

Можно исключить сообщения с заданным уровнем важности (пример 16.5).

Пример 16.5. Исключение заданного уровня важности

```
kern.*;kern.!=info /var/log/kernel
```

Информационные сообщения ядра в журнал записаны не будут (см. пример 16.5).

Звездочка обозначает любые источники или любые уровни важности (пример 16.6).

Пример 16.6. Указание любых уровней важности

```
authpriv.* /var/log/messages
```

В примере 16.6 указано, что будут записаны все любые сообщения авторизации.

Если в канал не должны быть записаны любые сообщения от каких-либо источников, то удобно использовать директиву `none` (пример 16.7).

Пример 16.7. Исключение заданных источников

```
*.crit;lpr,cron,mail.none /var/log/critical
```

После применения настроек из примера 16.7 в файл `/var/log/critical` будут направляться сообщения от всех источников о критических и более важных событиях, кроме любых сообщений от служб печати, почты, `at` и `cron`.

После внесения изменений в файл `/etc/syslog.conf` вовсе не обязательно перезапускать `syslogd`. Достаточно послать ему сигнал `HUP` (пример 16.8).

Пример 16.8. Информирование `syslogd` об изменении конфигурации

```
kill -1 syslogd
```

Программа `logger` позволяет проверить настройки `syslogd`. Опция `-p` программы `logger` предназначена для указания источника и важности сообщения (пример 16.9).

Пример 16.9. Посылка сообщения для журналирования

```
logger -p mail.err Proverka
```

Команда в примере 16.9 пошлет сообщение в канал `mail` с уровнем важности `err`.

ЗАДАНИЯ

- Требуется записывать сообщения от источника `auth` с уровнем важности не ниже `info` в файл `/var/log/mylog`. Сделайте соответствующую настройку.
- Протестируйте созданный журнал с помощью `logger`.
- Записываются ли в этот журнал сообщения о входе в сеанс и выходе из него?

Служба ротации журналов

С течением времени накапливающиеся сообщения в файлах журналов могут переполнить файловую систему. Для предотвращения этого предназначена программа `logrotate`, обеспечивающая ротацию журналов. Стандартный путь вызова `logrotate` — запуск ее с помощью `cron`.

Файл `/etc/logrotate.conf` содержит настройки `logrotate`, в котором определяются действия, производимые с журналами. Программа `logrotate` способна производить следующие действия с файлами журналов:

- ☐ удалять;
- ☐ переименовывать;
- ☐ сжимать с помощью программ-компрессоров;
- ☐ создавать новые пустые файлы журналов;
- ☐ посылать файлы журналов по электронной почте.

Ротация осуществляется следующим образом (пример 16.10 для журнала `messages`):

1. При первой ротации файл `messages` переименовывается в `messages.1`.
2. При второй ротации `messages.1` переименовывается в `messages.2`, а файл `messages` переименовывается в `messages.1`.
3. При третьей ротации файл `messages.2` переименовывается в `messages.3`, файл `messages.1` в `messages.2` и т. д.

Пример 16.10. Ротированные журналы

```
# ls -w 1 /var/log/messages*  
/var/log/messages  
/var/log/messages.1  
/var/log/messages.2
```

Утилита `logrotate` удаляет архивные копии старых журналов по достижении заданного количества копий (пример 16.11).

Пример 16.11. Файл конфигурации /etc/logrotate.conf

```
weekly
rotate 4
create
compress
notifempty
include /etc/logrotate.d
/var/log/utmp {
    monthly
    create 0664 root utmp
    rotate 4
}
```

Настройки по умолчанию находятся в начале файла /etc/logrotate.conf. Для каждого конкретного файла журнала можно указывать отдельные настройки, как это сделано для файла /var/log/utmp (см. пример 16.11).

Настройки `daily`, `weekly` и `monthly` определяют периодичность ротации.

Настройка `rotate` определяет количество старых журналов, которое должно храниться до удаления. А настройка `create` позволяет указывать права доступа и владения создаваемых журнальных файлов.

Для сжатия файлов старых журналов предназначена настройка `compress`. Кроме этого, можно просто копировать файлы при помощи настройки `copy`, оставляя при этом оригинальные файлы журналов нетронутыми.

Настройка `notifempty` позволяет не осуществлять ротацию пустых файлов.

Директива `include` позволяет включать в файл конфигурации дополнительные настройки, указанные в файле — аргументе этой директивы. Если аргументом является каталог, то в основной файл конфигурации включается содержимое всех конфигурационных файлов, находящихся в этом каталоге.

Настройка `mail` позволяет посылать журналы по электронной почте. А директивы `prerotate` и `postrotate` указывают сценарии, которые будут исполнены, соответственно, до и после ротации.

С помощью настройки `size` можно указывать размер файла журнала, при превышении которого должна осуществляться его ротация.

ЗАДАНИЕ

Создайте настройки ротации для журналов /var/log/mylog. Журнал должен ротироваться ежедневно. Старые журналы должны сжиматься утилитой `gzip`.



Глава 17

Управление пользователями

Управление пользователями — важнейшая задача системного администрирования. В этой главе вы познакомитесь с принципами аутентификации, форматами файлов для хранения учетных записей и изучите команды для управления учетными записями.

Хранение учетных записей пользователей

В GNU/Linux процедура аутентификации пользователя при входе в сеанс может быть проведена разными способами. Вот некоторые из них:

- ☐ хранение учетных записей в файлах;
- ☐ аутентификация с помощью системы Kerberos;
- ☐ аутентификация в NIS/NIS+;
- ☐ аутентификация в LDAP;
- ☐ использование специализированных систем аутентификации (например, TCB).

Файл `/etc/passwd` содержит учетные записи пользователей, а в файле `/etc/shadow` хранятся зашифрованные пароли. Каждая запись в этих файлах соответствует одному пользователю системы. Поля записей разделены двоеточиями.

Структура записей в файле `/etc/passwd` следующая:

- ☐ имя пользователя;
- ☐ пароль — содержит символ `x` при использовании теневых паролей;
- ☐ UID пользователя;
- ☐ GID пользователя;

- ☐ справочная информацию о пользователе;
- ☐ домашний каталог пользователя;
- ☐ оболочка.

Права доступа, устанавливаемые на файл `/etc/passwd`, позволяют читать этот файл всем пользователям. Поэтому хранение зашифрованного пароля во втором поле этого файла представляет реальную угрозу безопасности, т. к. любой злоумышленник, имеющий доступ к данной системе, может воспользоваться программами подбора паролей для взлома системы. Использование системы теневых паролей существенно снижает эту опасность, т. к. файл `/etc/shadow`, где хранятся зашифрованные пароли, не позволяет его читать никому, кроме суперпользователя.

Структура файла `/etc/shadow` такова:

- ☐ имя пользователя;
- ☐ зашифрованный пароль;
- ☐ количество дней с 1 января 1970 г. до момента последней смены пароля;
- ☐ минимальное время жизни пароля;
- ☐ максимальное время жизни пароля;
- ☐ время до момента устаревания пароля, начиная с которого пользователь будет получать предупреждения о необходимости;
- ☐ период времени с момента устаревания пароля, по истечении которого учетная запись пользователя будет заблокирована;
- ☐ срок жизни учетной записи;
- ☐ девятое поле зарезервировано и в настоящее время в GNU/Linux не используется.

Задания

- Просмотрите файл `/etc/shadow` (с правами `root`). У всех ли пользователей содержимое второго поля выглядит приблизительно одинаково?
- Какие символы могут содержаться в зашифрованной строке пароля в `/etc/shadow`?

Регистрация, удаление и блокирование учетных записей пользователей

Правами регистрации пользователей в системе обладает суперпользователь. Для добавления учетной записи пользователя применяется команда `useradd`. В качестве аргумента для этой команды должно быть указано имя пользователя (пример 17.1).

Пример 17.1. Регистрация пользователя

```
# useradd user1
# id user1
uid=504(user1) gid=504(user1) groups=504(user1)
```

В примере 17.1 в системе был зарегистрирован новый пользователь — `user1`. Для него были созданы домашний каталог и приватная группа — `user1`. Приватная группа пользователей состоит из единственного пользователя. Приватная группа может и не создаваться — это зависит от настроек конкретной системы.

Регистрация пользователя приводит к появлению соответствующих записей в файлах `/etc/passwd` и `/etc/shadow` (пример 17.2).

Пример 17.2. Учетные записи в `/etc/passwd` и `/etc/shadow`

```
# grep user1 /etc/passwd
user1:x:504:504::/home/user1:/bin/bash
# grep user1 /etc/shadow
user1:!!:14257:::::::
```

Создание приватной группы и домашнего каталога пользователя характерно для Red Hat Linux и подобных дистрибутивов. В других GNU/Linux-системах требуется предварительно создать все группы, в которых должен участвовать пользователь, если они еще не зарегистрированы. Кроме этого, создание домашнего каталога производится только при использовании опции `-m` команды `useradd`, либо опции `-d`, указывающей путь к домашнему каталогу.

В Red Hat создание приватной группы можно запретить, задавая опцию `-n`. При этом для вновь зарегистрированных пользователей в системе будет установлена группа по умолчанию (см. далее) в качестве первичной группы.

Настройки для команды `useradd` находятся в файле `/etc/default/useradd` и могут быть получены с помощью опции `-D` команды `useradd` (пример 17.3).

Пример 17.3. Настройки команды `useradd` по умолчанию

```
# useradd -D
GROUP=100
HOME=/home
INACTIVE=-1
```

```
EXPIRE=  
SHELL=/bin/bash  
SKEL=/etc/skel
```

Выведенная информация командой `useradd -D` говорит о следующем:

- ☐ `GROUP=100` — GID для вновь регистрируемых пользователей — 100 (для Red Hat эта настройка игнорируется при создании приватной группы пользователя);
- ☐ `HOME=/home` — домашние каталоги пользователей создаются в каталоге `/home`;
- ☐ `INACTIVE=-1` — блокирование учетной записи пользователя при устаревании его пароля не произойдет;
- ☐ `SHELL=/bin/bash` — оболочка для вновь регистрируемых пользователей;
- ☐ `SKEL=/etc/skel` — каталог "скелета", из которого в домашние каталоги вновь регистрируемых пользователей копируются файлы профиля и настроек.

Каталог `/etc/skel` обычно содержит файлы профиля для вновь регистрируемых пользователей и другие служебные файлы, которые копируются при регистрации пользователя в его домашний каталог.

Часто используемые опции команды `useradd`:

- ☐ `-s` — указывает исполняемый файл оболочки по умолчанию;
- ☐ `-d` — путь к домашнему каталогу пользователя;
- ☐ `-m` — опция, указывающая на необходимость создать домашний каталог;
- ☐ `-M` — не создавать домашний каталог;
- ☐ `-k` — путь к альтернативному каталогу скелета;
- ☐ `-u` — назначить UID пользователю;
- ☐ `-g` — назначить GID (первичную группу) пользователю;
- ☐ `-G` — список групп пользователя (через запятые);
- ☐ `-e` — календарная дата, после которой учетная запись будет заблокирована;
- ☐ `-f` — количество дней, которое должно пройти после срока устаревания пароля до блокировки учетной записи.

Приведенная в примере 17.4 команда регистрирует пользователя без создания для него домашнего каталога (опция `-M`) с первичной группой `users` (опция `-g`). Вход в сеанс запрещен, т. к. в качестве оболочки указан файл `/sbin/nologin`. Учетная запись пользователя будет заблокирована 23 февраля 2011 г. (опция `-e`).

Пример 17.4. Учетная запись с ограничением срока действия

```
# useradd -M -g users -s /sbin/nologin -e 2011-02-23 classuser
# id classuser
uid=505(classuser) gid=100(users) groups=100(users)
# grep classuser /etc/shadow
classuser:!!:14616::::::15028:
```

Последняя команда примера демонстрирует запись в файле теневых паролей для пользователя `classuser`. Восьмое поле записи содержит число дней с 1 января 1970 г. до дня, когда учетная запись пользователя будет заблокирована.

Если пользователь не имеет право входить в сеанс, то в качестве оболочки должен быть установлен один из следующих вариантов:

- ❑ `/bin/false` — системная команда, всегда возвращающая код ошибки;
- ❑ `/dev/null` — специальный файл-поглотитель;
- ❑ `/sbin/nologin` — системная команда, возвращающая при запуске код ошибки и сообщение о невозможности входа в сеанс.

Часто возникает необходимость произвести некоторые изменения в учетной записи уже зарегистрированного пользователя, например, поменять ему оболочку по умолчанию. Для этого предназначена команда `usermod`. Далее приведен пример 17.5 смены оболочки по умолчанию для пользователя.

Пример 17.5. Смена оболочки в учетной записи

```
# usermod -s /bin/false classuser
```

Большая часть опций команд `useradd` и `usermod` совпадает. Так, например, для добавления новой группы, в которой участвует пользователь, можно использовать команду, приведенную в примере 17.6.

Пример 17.6. Добавление пользователя в группу

```
# usermod -G "`id -G classuser | tr ' ' ','`" classuser
# id classuser
uid=505(classuser) gid=100(users) groups=100(users),14(uucp)
```

Команда `id -G` выводит список групп, в которые входит пользователь, разделенных пробелами. Далее пробелы заменяются запятыми с помощью команды `tr`, т. к. список групп для команды `usermod` должен быть задан через запятую. К списку групп, в которых пользователь уже принимает участие, добавляется список новых групп, а далее полученный список подставляется в командную строку `usermod` с помощью командной подстановки.

Используя команду `usermod`, можно также указать для пользователя его новое имя с помощью опции `-l`. А опции `-L` и `-U` позволяют, соответственно, блокировать и разблокировать возможность входа в сеанс для данного пользователя.

Удалить учетную запись пользователя можно командой `userdel` (пример 17.7).

Пример 17.7. Удаление пользователя

```
# userdel classuser
# id classuser
id: classuser: No such user
```

В этом примере учетная запись для пользователя `classuser` была удалена.

Перед удалением учетной записи пользователя необходимо решить, что делать с файлами пользователя, если таковые в системе имеются. Сама команда `userdel` удаления файлов в домашнем каталоге пользователя не производит. Поэтому все файлы пользователя, учетная запись которого подлежит удалению, должны быть найдены и либо удалены, либо помещены в архив, либо переданы другому пользователю.

Задания

- Зарегистрируйте пользователя `test1`, для которого запрещен вход в сеанс, имеющего домашний каталог `/home/nouser` и являющегося членом групп `users` и `mail`. Пользователь должен иметь UID, равный 1000.
- Создайте учетную запись для пользователя `test2` с настройками по умолчанию, но без создания приватной группы. Проверьте, создан ли домашний каталог пользователя, наполнен ли он файлами, и кому он принадлежит.
- Измените имя пользователя `test2` на `test3`.
- Удалите пользователя `test3`.
- Помимо файла `/etc/default/useradd` имеется еще один конфигурационный файл, влияющий на поведение команды `useradd`. Найдите его и изучите его содержание. Какая настройка позволяет изменять минимальный UID для новых пользователей?

Управление паролями

Правила установки, использования и управления паролями являются важнейшей частью системной политики. Обычно они включают в себя, минимум, следующее:

- ☐ определение категорий пользователей, которые имеют право самостоятельного выбора паролей с помощью команды `passwd`;
- ☐ правила выбора паролей и требования к их уровню сложности;
- ☐ сроки устаревания паролей;
- ☐ длительности периодов запрета на изменение паролей.

Четко сформулированная политика управления паролями значительно облегчает администрирование системы. Так, например, установив правила выбора паролей, их минимальную длину и требуемый уровень сложности, достаточно настроить модуль контроля паролей системы PAM для автоматической проверки соответствия выбираемого пользователем пароля требованиям безопасности.

Команда `passwd` помимо изменения паролей предоставляет и другие возможности. Далее приведен список часто применяемых опций команды `passwd`:

- ☐ `-l` — блокирование учетной записи;
- ☐ `-u` — деблокирование учетной записи;
- ☐ `-s` — получение текущего состояния пароля;
- ☐ `-d` — удаление пароля;
- ☐ `-n` — период запрета смены пароля (минимальное время жизни пароля);
- ☐ `-x` — максимальный срок использования пароля;
- ☐ `-w` — установка количества дней до момента устаревания пароля, начиная с которого будут выдаваться предупреждения о необходимости смены пароля;
- ☐ `-i` — срок с момента устаревания до блокировки пароля.

Пример 17.8. Блокирование учетной записи

```
# id lisa
uid=503(lisa) gid=503(lisa) groups=503(lisa),22(cdrom)
# cat /etc/shadow
lisa:$2a$08$Z4jZgPzM2GDVguFd4TRF3ubB:14316::::::
# passwd -l lisa
# cat /etc/shadow
lisa:!*$2a$08$Z4jZgPzM2GDVguFd4TRF3ubB:14316::::::
```


В примере 17.8 после блокирования учетной записи в первой позиции второго поля файла `/etc/shadow` перед шифрованным паролем пользователя появляется знак восклицания. При разблокировании учетной записи он исчезает.

ЗАДАНИЯ

- Зарегистрируйте пользователя `test4` с настройками по умолчанию и установите для него пароль. Изучите содержимое соответствующей пользователю записи в файле `/etc/shadow`.
- Установите дату устаревания пароля для пользователя на 31 декабря текущего года. Проверьте, что изменилось в `/etc/shadow`.
- Удалите пароль пользователя и проверьте изменения в `/etc/shadow`.
- Заблокируйте учетную запись `test4`.

Управление группами пользователей

С помощью создания групп пользователей системный администратор может эффективно управлять деятельностью в системе целыми коллективами пользователей, предоставляя им разрешения на доступ к системным ресурсам. Каждый файл располагает в метаданных триадой битов, кодирующей права доступа для группы пользователей. Следовательно, изменяя членство пользователя в группах, администратор изменяет, таким образом, привилегии пользователя на доступ к различным файлам в системе, не меняя при этом права пользователя на принадлежащие ему файлы.

Информация о группах пользователей хранится в файле `/etc/group` в виде строк.

Формат записей в `/etc/group`:

- имя группы;
- пароль группы (при отсутствии пароля — `x`);
- GID группы;
- список пользователей, принадлежащих к данной группе.

Для добавления новой группы необходимо воспользоваться командой `groupadd`, которая добавляет новую запись в файл `/etc/group` (пример 17.9).

Пример 17.9. Добавление группы

```
# groupadd class
# grep class /etc/group
class:x:505:
```

В этом примере добавлена новая группа `class`.

Пользователи, для которых группа является первичной, имеют информацию об этом в `GID`, хранящемся в четвертом поле файла `/etc/passwd`. В то же время имена пользователей, входящих в группу, которая не является для них первичной, записываются через запятую в четвертом поле файла `/etc/group`.

Пример 17.10. Члены группы

```
# grep sys /etc/group
sys:x:3:root,bin,adm
```

В группу `sys` входят пользователи `root`, `bin` и `adm` (пример 17.10).

Для явного указания `GID` используют опцию `-g` (пример 17.11).

Пример 17.11. Регистрация группы с заданным `GID`

```
# groupadd -g 512 project
```

В примере 17.11 создана новая группа `project` с заданным `GID`.

Для удаления группы используют команду `groupdel` (пример 17.12).

Пример 17.12. Удаление группы

```
# groupdel project
```

В примере 17.12 удалена группа пользователей `project`.

Группа пользователей может быть создана для работы над каким-либо проектом. В таком случае бывает удобно одного из пользователей сделать администратором группы и делегировать ему право добавлять уже зарегистрированных в системе пользователей в эту группу и удалять их из группы при необходимости.

Назначить администратора группы можно командой `gpasswd -A` (пример 17.13).

Пример 17.13. Назначение администратора группы

```
# gpasswd -A ivanov developers
```

Здесь администратором группы `developers` назначается пользователь `ivanov`.

Системный администратор может добавлять пользователей в группу с помощью команды `gpasswd -m` (пример 17.14).

Пример 17.14. Добавление пользователей в группу

```
# gpasswd -M marta,lisa developers
```

Пользователи `marta` и `lisa` были добавлены в группу `developers`.

Администратор группы может:

- ☐ добавить пользователя в группу командой `gpasswd -a` (пример 17.15);
- ☐ удалить пользователя из группы командой `gpasswd -d`.

Пример 17.15. Добавление пользователя в группу

```
$ gpasswd -a semko developers
```

Здесь администратор группы `developers` добавил в группу пользователя `semko`.

ЗАДАНИЕ

- Создайте группу пользователей `xusers` с GID, равным 1010.
- Зарегистрируйте себя в качестве участника группы `xusers`.
- Как изменить имена и GID групп? Измените имя группы на `yusers`.

Профили пользователей

При входе пользователей в сеанс автоматически выполняются специальные файлы сценариев, называемые *профилями пользователей*. Обычный подход к хранению настроек оболочки состоит в разделении настроек (профилей) на глобальный профиль (Master Profile) и пользовательские профили (Login Profiles). Кроме профилей имеются еще и специальные файлы настроек оболочек (resource files), которые также являются сценариями оболочек. Отличие профилей от файлов ресурсов состоит в том, что сценарии профилей исполняются единожды при входе пользователя в сеанс, а файлы ресурсов запускаются при запуске оболочки из командной строки.

Если оболочка Bash запущена интерактивно при входе пользователя в сеанс (т. е. является оболочкой по умолчанию), то сначала выполняется общий для всех пользователей файл `/etc/profile`, а затем индивидуальный профиль пользователя, находящийся в его домашнем каталоге. При запуске оболочки Bash

последовательно пытается найти пользовательский профиль в файлах с именами:

- ❑ `~/.bash_profile`;
- ❑ `~/.bash_login`;
- ❑ `~/.profile`.

В файлах профилей обычно устанавливаются такие переменные окружения, как:

- ❑ `PATH` — имена каталогов, в которых Bash ищет исполняемые файлы;
- ❑ `TERM` — тип терминала;
- ❑ `USER` — имя пользователя (устанавливается с помощью `id -un`);
- ❑ `HOME` — путь к домашнему каталогу пользователя;
- ❑ `MAIL` — путь к почтовому ящику пользователя.

Переменные окружения, устанавливаемые в файлах профилей, должны быть экспортированы с помощью команды `export`. В примере 17.16 к списку каталогов в переменной окружения `PATH` добавляется каталог `bin` в домашнем каталоге.

Пример 17.16. Назначение значения переменной `PATH`

```
PATH=$PATH:$HOME/bin
export PATH
```

Имена каталогов, содержащихся в переменной `PATH`, разделяются двоеточиями. Помимо переменных окружения в файлах профиля также устанавливается `umask`.

При необходимости исполнить файл профиля из командной строки следует использовать команду `source` (пример 17.17).

Пример 17.17. Команда `source`

```
# source /etc/profile
```

Эта команда является встроенной и выполняет в текущей оболочке команды из файла, указанного в качестве аргумента.

В противоположность профилям файл ресурсов оболочки `~/.bashrc` выполняется только при интерактивном запуске оболочки Bash из командной строки, а не при входе в сеанс. Для того чтобы дополнительные настройки оболочки

срабатывали не только при запуске оболочки из командной строки (т. е. из уже запущенной оболочки), но и при запуске Bash по умолчанию при входе в сеанс, вызов инструкций в файле `~/.bashrc` часто производится из пользовательского профиля. Типичное содержимое файла пользовательского профиля показано в примере 17.18.

Пример 17.18. Профиль пользователя

```
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

BASH_ENV=$HOME/.bashrc
export BASH_ENV
```

Здесь приведен пример содержимого файла пользовательского профиля, в котором проверяется наличие в домашнем каталоге пользователя файла ресурсов оболочки, и, если он есть, содержимое его выполняется в контексте текущей оболочки. Это достигается с помощью *inline-подстановки* — команды "точка" (`.`). Вызов `. ~/.bashrc` приводит к тому, что переменные, псевдонимы и функции, определенные в файле ресурсов, будут доступны в текущей оболочке. Inline-подстановка всегда используется для передачи из одного файла сценария оболочки в другой сценарий переменных, псевдонимов и функций.

Переменная окружения `BASH_ENV`, определенная в примере 17.18, предназначена для информирования оболочки, запускаемой неинтерактивно (например, для выполнения сценария), что должны быть использованы ресурсы, определенные в файле, имя которого содержится в этой переменной.

Довольно часто в файле `~/.bashrc` находится inline-вызов общесистемного файла ресурсов `/etc/bashrc`. Это не обязательно, но очень удобно, т. к. в этом файле можно определить, например, псевдонимы для команд, которыми часто пользуются различные пользователи системы, вместо определения этих псевдонимов в частных файлах ресурсов оболочки `~/.bashrc`.

Итак, далее приведен список действий, которые обычно выполняются автоматически при входе в сеанс Bash:

1. Исполняется общесистемный скрипт профиля `/etc/profile`.
2. Выполняется пользовательский скрипт профиля в его домашнем каталоге (например, `~/.bash_profile`).
3. В пользовательском профиле проверяется наличие в домашнем каталоге файла ресурсов оболочки `~/.bashrc`, и, при его наличии, он исполняется.

4. Если выполняется файл ресурсов оболочки, то обычно в нем вызывается общесистемный файл ресурсов `/etc/bashrc`.
5. При запуске оболочки из командной строки выполняются пункты 3 и 4 списка.

ЗАДАНИЯ

- Измените значение `umask` на `027` для всех пользователей системы, оболочкой по умолчанию для которых является `Bash`.
- Установите в собственном профиле оболочки псевдоним `l` для команды `ls -ll`.
- Каким образом сделать так, чтобы этот же псевдоним устанавливался и для всех вновь регистрируемых пользователей в системе с оболочкой `Bash`?
- В каком файле удобнее всего добавить к переменной окружения `PATH` путь к каталогу `bin`, находящемуся в домашнем каталоге обычного пользователя?

Квотирование дискового пространства

Если в системе работает множество пользователей, то даже при большом объеме накопителей может обнаружиться недостаток дискового пространства, вызванный деятельностью пользователей. Возможный подход к решению данной проблемы заключается в создании специального раздела для каталогов пользователей. В таком случае, естественно, файлы пользователей не смогут превзойти размер раздела. Однако такое решение создаст иную проблему — одни пользователи будут фактически лишать других пользователей дискового пространства.

Кардинальное решение состоит во введении в системе квотирования. Квотирование позволяет ограничить размер дискового пространства, занимаемого файлами пользователя, выделяя пользователю определенные дисковые ресурсы — квоту.

Для установки системы квотирования, прежде всего, необходимо проверить, установлен ли пакет `quota`. В системах, использующих в качестве менеджера пакетов `RPM`, это можно сделать с помощью команды `rpm -q quota`. Эта команда должна вывести полное имя пакета с его версией (пример 17.19).

Пример 17.19. Проверка наличия пакета `quota`

```
$ rpm -q quota
quota-3.17-3.1
```

Помимо наличия пакета `quota`, который позволяет проверять объем дисковых ресурсов, занимаемых пользователем, в системе должно быть установлено ядро с поддержкой квоты. Поддержка квоты включается в разделе конфигурирования файловых систем при настройке ядра перед его сборкой.

Процесс создания квот состоит из трех шагов:

1. Определение файловых систем, ресурсы которых будут котируются.
2. Создание базы данных квот для каждой котируемой файловой системы.
3. Установка индивидуальных значений квот для пользователей и групп.

Фактически определение котируемых файловых систем заключается в их монтировании с опциями:

- ❑ `usrquota` — для установки индивидуальных пользовательских квот;
- ❑ `grpquota` — для установки квот для групп пользователей.

Эти опции могут быть установлены вместе или по отдельности в зависимости от требований, предъявляемых к системе. Команда монтирования допускает эти опции, но игнорирует их, т. к. они предназначены для программного обеспечения котирувания. Поддержка квот включается при монтировании файловых систем, поэтому опции `usrquota` и `grpquota` указывают в файле `/etc/fstab`.

Пример 17.20. Установка опции монтирования `usrquota`

```
/dev/sda6 /home ext3 defaults,noatime,usrquota 1 2
```

В примере 17.20 каталог `/home` является точкой монтирования для файловой системы `/dev/sda6`, для которой будут установлены пользовательские квоты.

После внесения изменений в файл `/etc/fstab` котируемая файловая система должна быть перемонтирована для того, чтобы опции котирувания вступили в силу. Для проверки правильности монтирования следует выполнить команду `mount` без каких-либо аргументов. В списке смонтированных файловых систем эта команда должна отобразить опции котирувания для файловых систем.

Пример 17.21. Проверка опций монтирования для поддержки квоты

```
$ mount
...
/dev/hda6 on /home type ext3 (rw,noatime,usrquota)
```

В примере 17.21 приведен фрагмент вывода команды `mount`, который подтверждает, что файловая система, смонтированная в каталоге `/home`, подлежит квотированию, т. к. при ее монтировании была использована опция `usrquota`.

После монтирования квотируемых файловых систем с установленными опциями квотирования необходимо создать базу данных квот, в которой хранится информация о файловых ресурсах, занимаемых пользователями. База данных хранится в файлах:

- `quota.user` — для пользовательских квот;
- `quota.group` — для квот групп пользователей.

Файлы базы данных квот должны располагаться в каталогах — точках монтирования квотируемых файловых систем. Если квотируется файловая система, смонтированная в каталоге `/home`, то файлы базы данных квотирования должны находиться в нем.

Файлы базы данных будут созданы автоматически с помощью команды `quotacheck`, которая предназначена для проверки целостности базы данных квот. Обычно эта команда вызывается автоматически при монтировании квотируемых файловых систем во время загрузки операционной системы. Важнейшие опции команды `quotacheck`:

- `-c` — создать новую базу данных квот. При использовании этой опции предыдущие настройки квот для пользователей и групп будут уничтожены;
- `-m` — не перемонтировать файловую систему в режиме только для чтения при создании или проверке квот;
- `-a` — проверить квоты на всех смонтированных с опциями квотирования файловых системах. Если эта опция не используется, то для команды должен быть аргумент, указывающий проверяемую файловую систему;
- `-u` — проверить только пользовательские квоты;
- `-g` — проверить только квоты для групп;
- `-v` — выдавать дополнительную информацию в процессе работы.

Показанная в примере 17.22 команда создает новую базу данных пользовательских квот во всех квотируемых файловых системах.

Пример 17.22. Создание базы данных квот

```
# quotacheck -caumv
```

После работы этой команды в нашем примере будет создан файл базы данных пользовательских квот для файловой системы `/home`. В данном случае он называется `/home/quota.user`.

Далее следует приступить к определению квот для пользователей и/или групп. Квотированию могут подлежать:

- ☐ суммарный объем файлов данного пользователя или группы пользователей;
- ☐ количество inode, т. е. количество файлов, пользователя или группы.

В пакете квотирования используется модель ограничений, в которой указывают ограничение, которое не может быть превзойдено (*hard quota*). Также указывают ограничение, при котором пользователь получает сообщение о превышении квоты, но блокирование записи еще не производится (*soft quota*). Время, на которое разрешается превышать *soft quota*, называется *grace period*. По умолчанию он установлен равным 7 суткам. По истечении этого срока операции записи блокируются. Этот период времени не может быть установлен индивидуально для каждого пользователя — он распространяется на всех.

Таким образом, установка индивидуальных квот для пользователей или групп заключается в следующем:

1. Определяется *grace period* для всех пользователей и групп. Если эта операция не производится, то используется значение по умолчанию.
2. Устанавливаются настройки *soft*- и *hard*-квот на объем файлов и их количество (*inode*) для одного или нескольких пользователей индивидуально. Любой из них может быть использован в качестве образца для установки квот для других пользователей.
3. Используя настройки квот для одного из пользователей как шаблон, производится определение квот для других пользователей.

Продолжительность *grace period* может быть установлена с помощью команды `edquota -t`. При вызове этой команды будет запущен текстовый редактор по умолчанию (например, `vi`), в окне редактирования которого можно будет изменить текущее значение *grace period* для количества и объема файлов. Редактирование производится во временном файле, причем если работа редактора будет завершена с записью в этот временный файл, то команда `edquota` считает содержимое временного файла и установит новое значение *grace period*.

Период времени *grace period* может быть задан в таких единицах времени:

- ☐ `seconds` — секунды;
- ☐ `minutes` — минуты;
- ☐ `hours` — часы;
- ☐ `days` — дни;

- weeks — недели;
- months — месяцы.

После установки `grace period` можно перейти к установке пользовательских и групповых квот. Пользовательская квота настраивается командой `edquota -u`.

Пример 17.23. Определение квоты для пользователя

```
# edquota -u asimonov
```

В данном случае файловая квота будет установлена для пользователя `asimonov`.

Как и при установке `grace period`, будет вызван редактор по умолчанию, в окне редактирования которого можно будет увидеть уже занятый файлами пользователя объем дискового пространства и количество использованных `inode`. Также будут отображены столбцы для установки новых значений `soft`- и `hard`-квот для суммарного объема файлов и количества `inode`. Именно эти столбцы должны быть отредактированы для установки новых значений. После выхода из редактора с сохранением временного файла он будет считан командой `edquota`, и будут установлены новые значения квот.

Как только квота установлена для одного из пользователей системы, настройки для него могут быть использованы в качестве шаблона для других пользователей. Для этого предназначена команда `edquota -p <шаблон>`. Преимуществом этой команды является неинтерактивный режим работы. То есть при вызове этой команды редактор текста не запускается, а настройки для пользователя, используемые как шаблон, будут просто скопированы в базе данных квот для других пользователей, указанных после опции `-u` (пример 17.24).

Пример 17.24. Определение квоты по шаблону

```
# edquota -p asimonov -u bsmirnov akozlov rrumin zkravchenko
```

В этом случае настройки пользовательской квоты для `asmirnov` будут использованы как шаблон квот пользователей: `bsmirnov`, `akozlov`, `rrumin` и `zkravchenko`.

Если требуется установить квоты для групп пользователей, то вместо опции `-u` команды `edquota` надо использовать опцию `-g`, после которой должна быть указана группа пользователей для квотирования.

После определения пользовательских и/или групповых квот необходимо включить программное обеспечение квотирования в ядре. Это достигается

с помощью вызова команды `quotaon`. Если в командной строке не будет задана опция `-a`, включающая квотирование для всех смонтированных с опциями квотирования файловых систем, то в качестве аргумента указывают квотируемую файловую систему (пример 17.25).

Пример 17.25. Включение квотирования

```
# quotaon -vu /dev/sda6
```

Квотирование файловой системы `/dev/sda6`. Опция `-u` включает пользовательские квоты, а опция `-v` выдает сообщение о включении квоты.

Вызов команды `quotaon` обычно осуществляется автоматически при загрузке операционной системы. Выключить квотирование можно с помощью команды `quotaoff`. Для нее также необходимо либо указать опцию `-a` для отключения квотирования всех файловых систем, либо указать квотируемую файловую систему.

Пользователь может получить информацию об ограничениях, установленных квотой для него, с помощью команды `quota`. Она сообщает об объеме и количестве файлов пользователя. Если пользователь превысит значение `soft quota`, то ему выдается соответствующее сообщение. С этого момента начнется обратный отсчет времени `grace period`. Как только этот период времени будет исчерпан, или же при попытке превысить `hard quota`, операции записи будут блокироваться.

Администратор может получать информацию о состоянии пользовательских квот с помощью команды `repquota`.

ЗАДАНИЕ

- Проверьте, установлено ли программное обеспечение квотирования.
- Если файловая система `/home` смонтирована на отдельном разделе, укажите для нее опцию монтирования с пользовательскими квотами в файле `/etc/fstab` и перемонтируйте ее.
- Создайте в квотируемой файловой системе файл базы данных квот.
- Установите `grace period` равным 2 минутам.
- Определите для `user1` следующие квоты: 100 Мбайт (`soft`) и 120 Мбайт.
- Включите квотирование и испытайте его с помощью команды `dd`.
- Превзойдите `soft quota`. Что происходит по прошествии `grace period`?
- Используя пользователя `user1` в качестве шаблона, установите квоты для пользователя `user2`. Что показывает после этого команда `repquota`?

Мониторинг активности пользователей

Команда `who` позволяет получить список пользователей, находящихся в настоящее время в сеансе (пример 17.26). Информация об этом берется из специального двоичного файла `/var/run/utmp`.

Пример 17.26. Команда `who`

```
$ who
user1      :0                Jan 21 15:10
```

С помощью этой же команды, используя соответствующие опции, можно получать и другую информацию. Важнейшие опции команды `who`:

- ❑ `-b` — время последней загрузки системы;
- ❑ `-h` — печать заголовка;
- ❑ `--login` — информация о системных процессах, контролирующих вход в сеанс;
- ❑ `-q` — печатает все имена пользователей в сеансе и их количество;
- ❑ `-w` — текущий статус всех сеансов;
- ❑ `-u` — подробная информация о сеансах (пример 17.27);
- ❑ `-a` — полная информация о статусе процессов, контролирующих вход в сеанс.

Пример 17.27. Подробный отчет `who` о сеансах

```
$ who -uH
NAME      LINE      TIME          IDLE          PID COMMENT
user1     tty7      Jan  4 22:44   old          4237 (:0)
user1     pts/0     Jan  4 23:48   .            5758 (:0.0)
user1     pts/1     Jan  4 23:55   00:24        5758 (:0.0)
```

Для получения отчета о сеансах пользователей, которые уже завершились, необходимо воспользоваться информацией, сохраняемой в файле `/var/log/wtmp`. Этот бинарный файл имеет ту же структуру, что и `/var/run/utmp`, поэтому его содержимое можно отобразить, указав его в качестве аргумента команды `who`. Однако для этого есть команда `last` (пример 17.28).

Пример 17.28. Команда last

```
$ last
user1 pts/1 :0.0 Mon Jan 4 23:55 still logged in
user1 pts/0 :0.0 Mon Jan 4 23:48 still logged in
user1 tty7 :0 Mon Jan 4 22:44 still logged in
reboot system boot 2.6.31.8-0.1-des Mon Jan 4 22:42 (3+01:42)
user1 pts/1 :0.0 Sat Jan 2 20:19 - 22:40 (2+02:20)
```

Эта команда выводит информацию об открытых и законченных сеансах в обратном хронологическом порядке.

Имеется также стандартный файл журнала /var/log/lastlog, в котором также в бинарном виде хранится информация о последних входах в сеанс. Для получения информации, находящейся в этом файле, требуется использовать команду `lastlog`. Она выводит в упорядоченном виде отчет о последних входах пользователей в сеанс.

Опции команды `lastlog`:

- ☐ `-b` — показывает входы в сеанс, совершенные ранее указанного количества дней;
- ☐ `-t` — показывает записи `lastlog`, за указанный период в днях;
- ☐ `-u` — показывает записи `lastlog` для указанного пользователя (пример 17.29).

Пример 17.29. Получение отчета о пользователе

```
$ lastlog -u user1
Username      Port      Latest
user1         :0        Mon Nov 30 13:04:01 +0500 2009
```

Задания

- Определите, когда была последний раз загружена система.
- С помощью опции `-a` команды `who` получите подробную информацию о пользователях и статусе системы.
- Командой `who` получите список пользователей, входивших в сеанс ранее.
- Сравните предыдущий список со списком, выводимым командой `last`.
- Получите отчет по входам в сеанс суперпользователя с помощью `lastlog`.
- Кто входил в сеанс за последние пять дней?



Глава 18

Управление программным обеспечением

Одной из основных задач системного администрирования, возникающей сразу после установки системы и актуальной в течение всей ее эксплуатации, является управление программным обеспечением. В этой главе вы познакомитесь с важнейшими системами управления программным обеспечением, а также узнаете, как собирать программное обеспечение из архивов с исходным кодом.

В чем состоит управление программным обеспечением?

Процесс управления программным обеспечением имеет следующие составляющие.

- ❑ *Установка нового программного обеспечения.* Необходимость установки может быть связана с изменениями требований к системе в процессе эксплуатации или, например, с проблемами с безопасностью.
- ❑ *Обновление программного обеспечения.* Одна из наиболее распространенных причин взлома систем — использование устаревшего программного обеспечения, приводящее к проблемам с безопасностью. Регулярное обновление программного обеспечения — одна из рутинных задач системного администрирования.
- ❑ *Проверка подлинности нового программного обеспечения.* Устанавливаемое и обновляемое программное обеспечение должно поступать из надежных и проверяемых источников.
- ❑ *Удаление программного обеспечения.* Необходимость удаления продиктована требованием наличия в системе только того программного обеспечения,

которое действительно нужно, т. к. любая программа может содержать ошибки, приводящие к брешам в безопасности системы.

- ❑ *Проверка целостности программного обеспечения.* Эта задача связана с защитой от возможной порчи программного обеспечения при сбоях в системе или в результате чьей-либо несанкционированной деятельности в системе. Проверка целостности заключается в анализе размеров файлов, прав доступа и владения, контрольных сумм, времени модификации и прочее.
- ❑ *Создание собственных пакетов или пересборка существующих.* Сборка новых пакетов больше связана с деятельностью разработчиков программного обеспечения или создателей пакетов, отвечающих за их поддержку (maintainers). Однако пересборка существующего пакета может потребоваться и в работе обычного системного администратора. Например, при необходимости оптимизации программного обеспечения для конкретной аппаратной платформы.
- ❑ Менее распространенная задача в мире свободного программного обеспечения — *регистрация и лицензирование программного обеспечения.* В последнее время GNU/Linux часто используется для работы коммерческого программного обеспечения, которое должно быть зарегистрировано и лицензировано.

Существует несколько вариантов установки программного обеспечения:

- ❑ сборка и установка из архивов с исходным кодом (tarballs);
- ❑ установка из архивов с бинарным машинным кодом (binaries);
- ❑ установка из бинарных пакетов (package) с помощью систем управления пакетами (package manager);
- ❑ сборка бинарного пакета из пакета с исходным кодом (source package) с последующей установкой;
- ❑ сборка и установка программного обеспечения из исходного кода с помощью порта — сценария автоматизации (основной способ установки пакетов в Gentoo, где порты называются portage по аналогии с port во FreeBSD).

В подавляющем большинстве GNU/Linux-дистрибутивов имеется *система управления пакетами*. Она в значительной мере упрощает и стандартизирует процесс управления программным обеспечением. Основываясь на информации, предоставляемой на сайте **www.distrowatch.org**, можно утверждать, что наиболее распространены четыре системы управления пакетами:

- ❑ RPM — Red Hat Package Manager. Применяется в RH и подобных ему системах, SUSE и многих других дистрибутивах. Предоставляет возможности

установки бинарных пакетов и позволяет собирать бинарные пакеты самостоятельно;

- ❑ система управления пакетами Debian. Кроме Debian используется в собранных на его основе дистрибутивах, например, в Ubuntu. Предоставляет широкие возможности по управлению пакетами;
- ❑ система портов Gentoo. Этот дистрибутив ориентирован на сборку программного обеспечения с помощью специальных сценариев из архивов с исходным кодом. Позволяет также устанавливать заранее собранные пакеты;
- ❑ система управления пакетами, принятая в SlackWare. Здесь применяются пакеты в виде бинарных архивов в формате TAR.

Преимущества систем управления программным обеспечением:

- ❑ осуществляется единообразное управление программным обеспечением;
- ❑ программы устанавливаются в стандартные места файловой системы;
- ❑ управление программным обеспечением простое и прозрачное;
- ❑ во многих системах есть разграничение ролей пользователей, способных выполнять разные функции в управлении программным обеспечением;
- ❑ легко проверить целостность программного обеспечения.

Недостатки систем управления пакетами, ориентированных на бинарные пакеты:

- ❑ сложно установить часть программного обеспечения из пакета;
- ❑ трудно устанавливать программы в нестандартные места файловой системы, например, в домашние каталоги пользователей;
- ❑ трудно, а иногда и невозможно устанавливать программы из других дистрибутивов или из предыдущей версии этого же дистрибутива;
- ❑ пакеты необходимо собирать заново для оптимизации под данную систему, а также для добавления или удаления некоторой функциональности.

В соответствии со стандартом FHS программное обеспечение, устанавливаемое с помощью систем управления пакетами, размещается в каталогах:

- ❑ /bin;
- ❑ /sbin;
- ❑ /lib;
- ❑ /usr/bin;
- ❑ /usr/sbin;

- ❑ /usr/lib;
- ❑ /usr/X11R6;
- ❑ /opt.

Каталог /opt обычно используется для программного обеспечения, не поставляемого в составе дистрибутива. В каталог /usr/X11R6 помещаются файлы, относящиеся к системе X Window. Файлы помощи для программного обеспечения, устанавливаемого с помощью систем управления пакетами, должны размещаться в /usr/share/man, а документация — в /usr/share/doc.

Программное обеспечение, устанавливаемое самостоятельно с помощью сборки из архивов с исходным кодом, размещается в подкаталогах каталога /usr/local.

При установке программного обеспечения очень часто возникает конфликт пакетов или зависимостей, который может быть вызван причинами, приведенными далее.

- ❑ Два пакета взаимно исключают совместную работу. Так, например, нельзя использовать два сервера SMTP (Simple Mail Transfer Protocol). При попытке установить программу postfix в системе, где установлена почтовая программа sendmail, возникнет конфликт.
- ❑ Библиотеки, с которыми может работать пакет, имеют другие версии.
- ❑ Устанавливаемый пакет может требовать наличия других программ или библиотек, отсутствующих в настоящий момент в системе.
- ❑ Может возникать также и конфликт версий конфигурационных файлов.

ЗАДАНИЯ

- В каких случаях требуется разрешить обычным пользователям устанавливать программы в домашних каталогах?
- Какие проблемы с этим связаны?
- Просмотрите подкаталоги /usr/local.

Сборка и установка программного обеспечения из пакетов с исходным кодом

Очень часто бывает необходимо самостоятельно собрать программу из архива с исходным кодом, например, потому, что пакеты с этим ПО еще не собраны.

В подавляющем большинстве случаев, хотя и не всегда, для сборки и установки программы из архива с исходным кодом достаточно выполнить следующие действия:

1. Получить файл архива.
2. Разархивировать его.
3. Прочитать файл README или INSTALL.
4. Сконфигурировать сценарий сборки.
5. Собрать программу.
6. Установить программу.

В качестве иллюстрации к процессу сборки из архива с исходным кодом здесь приводится пример сборки DNS-сервера BIND (Berkeley Internet Names Daemon) последней на момент написания данной главы версии — BIND 9.7.0rc1.

Первый шаг заключается в получении архива с исходным кодом (пример 18.1).

Пример 18.1. Получение архива с исходным кодом

```
$ wget http://ftp.isc.org/isc/bind9/9.7.0rc1/bind-9.7.0rc1.tar.gz
```

Программа `wget` — один из наиболее удобных инструментов для получения файлов с анонимных FTP-серверов.

Далее требуется извлечь из архива его содержимое (пример 18.2).

Пример 18.2. Распаковка архива

```
$ tar xzvf bind-9.7.0rc1.tar.gz
```

В результате этого должен быть создан каталог с именем пакета. В данном случае — `bind-9.7.0rc1`. Необходимо перейти в этот каталог. Очень важно изучить содержимое файла README и INSTALL, если они существуют (в этом примере есть README).

Следующий этап заключается в конфигурировании сценария сборки пакета. В GNU/Linux сборка программы производится с помощью программы `make`. Она выполняет действия, заданные в файле Makefile, называемые "целями" (target). Первая цель в Makefile выполняется по умолчанию, т. е. без задания утилите `make` аргумента.

Обратите внимание, что пока еще в текущем каталоге нет файла `Makefile`, а есть заготовка для его создания — `Makefile.in`. Дело в том, что, во-первых, на различных платформах сборка должна осуществляться по-разному, а во-вторых, можно собрать программу разными опциями, заданными программистом.

Процесс создания `Makefile` для данной системы в соответствии с требованиями, предъявляемыми к программе, называется *конфигурированием*. Он выполняется с помощью сценария `configure`. Стандартная опция этого сценария — `--help`. С ее помощью можно получить подсказку о возможностях конфигурирования (пример 18.3).

Пример 18.3. Список опции сборки

```
$ ./configure --help | less
```

Точка и косая черта перед именем исполняемого файла сценария требуются, т. к. каталог, в котором находится этот сценарий, не указан в переменной `PATH`.

Одна из наиболее важных опций конфигурирования — `--prefix`. С ее помощью можно устанавливать путь к базовому каталогу для инсталляции программы и ее компонентов. По умолчанию этот путь — `/usr/local`, и установка файлов будет произведена в подкаталоги этого каталога.

Для `BIND` опция `--disable-largefiles` отключает поддержку 64-битной адресации блоков файлов, а `--disable-ipv6` отключает поддержку IPv6 (пример 18.4).

Пример 18.4. Установка опций сборки

```
./configure --disable-largefiles --disable-ipv6
```

В результате работы сценария конфигурации будет создан файл `Makefile` на основе шаблона `Makefile.in`.

Процесс сборки запускается с помощью команды `make` (пример 18.5).

Пример 18.5. Сборка программы

```
$ make
```

После сборки необходимо установить собранное программное обеспечение. Если установка осуществляется в каталог `/usr/local`, то перед инсталляцией

требуется перейти в сеанс суперпользователя. Рекомендуется пользоваться командой `su` без выхода из текущего каталога (как это произойдет в случае `su -`). Установка осуществляется на основе команд в `Makefile`, соответствующих цели `install` (пример 18.6).

Пример 18.6. Установка программы

```
$ su
# make install
```

Можно запустить одну из установленных программ для того, чтобы убедиться в ее работоспособности. Узнаем версию установленного демона `BIND` (пример 18.7).

Пример 18.7. Проверка работоспособности установленной программы

```
# /usr/local/sbin/named -v
BIND 9.7.0rc1
```

Программа демона `BIND` при запуске ее с опцией `-v` выдала версию демона.

ЗАДАНИЯ

- Получите последнюю версию пакета демона `BIND`.
- Соберите программное обеспечение, установите пакет и проверьте его версию.

Управление библиотеками

Практически все программы требуют для своей сборки или работы наличия в системе файлов библиотек. Библиотеки представляют собой файлы со специальной структурой, содержащие скомпилированный бинарный код (так называемый объектный код), предназначенный для подключения к машинному коду, содержащемуся в исполняемых файлах программ. Процесс подключения кода в библиотеках к коду самой программы называется *компоновкой* (*linking*).

В зависимости от типа компоновки различают библиотеки разных видов:

- ☐ статические библиотеки (*static library*);
- ☐ разделяемые библиотеки (*shared library*).

По соглашению файлы статических библиотек заканчиваются суффиксом `.a` (archive), а файлы разделяемых библиотек — `.so` (shared object).

Статические библиотеки представляют собой наборы объектных файлов, объединенных вместе с помощью утилиты `ar`. Скомпилированный код, находящийся в библиотеке, может быть использован в коде программ. Это позволяет избавиться от необходимости многократного повторного написания кода.

Так как требуемый код из статических библиотек помещается на стадии компоновки в результирующий программный код, находящийся в исполняемом файле, то для выполнения таких программ никаких библиотек уже не требуется. То есть библиотека используется только на стадии сборки программы, но не на стадии исполнения, что является существенным преимуществом таких программ.

Статически собранные программы удобны для применения в случае аварийного восстановления системы и во всех остальных случаях, когда доступ к требуемым библиотекам затруднен или невозможен. Однако размер статически собранной программы обычно значительно превышает размер этой же программы, собранной с использованием разделяемых библиотек.

Идея разделяемой библиотеки состоит в том, что один и тот же код, находящийся в файле библиотеки, используется при исполнении разных программ. То есть код из библиотеки не помещается в код программ на стадии компоновки, он вызывается из файла библиотеки по мере его необходимости. Тем не менее, код, находящийся в библиотеке, жестко связан с кодом программы.

Использование разделяемых библиотек существенно экономит дисковое пространство, т. к. множество программ используют код библиотек совместно. При запуске компилятора `gcc` по умолчанию производится сборка программ с помощью разделяемых библиотек.

В качестве примера соберем несложную программу, выводящую на терминал строку "Privet!". Код ее на языке C приводится в примере 18.8.

Пример 18.8. Простая программа на C

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

int main()
{
    printf("%s\n", "Privet!");
    return 0;
}
```

Данная программа использует библиотечную функцию форматированной печати `printf()`, которая выводит на терминал строку, указанную ей в качестве второго аргумента. Первый аргумент — строка форматирования. Функция `printf()` описана в заголовочном файле `stdio.h`.

Соберем эту программу с использованием разделяемых библиотек (пример 18.9).

Пример 18.9. Компиляция программы

```
$ gcc -o simple.dynamic simple.c
$ file simple.dynamic
simple.dynamic: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
for GNU/Linux 2.6.4, dynamically linked (uses shared libs), not stripped
```

В результате выполнения этой команды будет создан исполняемый файл `simple.dynamic` из исходного кода в файле `simple.c`. Вывод команды `file` подтверждает то, что программа собрана с поддержкой разделяемых библиотек.

При статической сборке используют опцию `-static` компилятора `gcc` (пример 18.10).

Пример 18.10. Сборка статически скомпонованной программы

```
$ gcc -static simple.c -o simple.static
$ file simple.static
simple.static: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
for GNU/Linux 2.6.4, statically linked, not stripped
```

Сравним теперь размеры полученных файлов (пример 18.11).

Пример 18.11. Размеры динамически и статически скомпонованных программ

```
$ ls -l simple.*
-rwxr-xr-x  1 user  user      6793 Dec 28 23:08 simple.dynamic
-rwxr-xr-x  1 user  user    485882 Dec 28 23:08 simple.static
$ ./simple.static
Privet!
$ ./simple.dynamic
Privet!
```

Легко заметить, что размеры полученных файлов значительно отличаются, несмотря на то, что программы делают одно и то же.

Как определить, с какими разделяемыми библиотеками собран файл? Ответ на этот вопрос предоставляет команда `ldd` (пример 18.12).

Пример 18.12. Зависимости от разделяемых библиотек

```
$ ldd simple.dynamic
linux-gate.so.1 => (0xffffe000)
libc.so.6 => /lib/libc.so.6 (0xb767b000)
/lib/ld-linux.so.2 (0xb7813000)
```

Итак, программа примера зависит лишь от библиотеки `libc.so.6`. Это основная библиотека стандартных функций C, имеющая шестой номер версии. При сборке программ эта библиотека подключается автоматически, поэтому в примере она нигде не была указана в опциях компилятора.

По соглашению все имена библиотек начинаются с префикса `lib`. Когда команда `ldd` выводит список зависимостей от библиотек, указывается не имя библиотеки, а имя файла символической ссылки на библиотеку (пример 18.13).

Пример 18.13. Имена разделяемых библиотек

```
$ ls -l /lib/librt.so.1
lrwxrwxrwx 1 root root 14 Nov 8 03:27 /lib/librt.so.1 -> librt-2.10.1.so
```

Символические ссылки, указывающие на файлы разделяемых библиотек, называются `SO-name` (или `soname`) и имеют особый смысл. С помощью этих символических ссылок облегчается задача обновления библиотек. В исполняемом файле указано, от какой библиотеки с известным `SO-name` он зависит. Символическая же ссылка с таким именем может указывать на библиотеки с различными версиями. Так, в рассматриваемом случае `SO-name` `librt.so.1` может указывать на предыдущую версию библиотеки `librt-2.10.1.so`. Таким образом, имеется возможность обновления библиотек без внесения изменений в исполняемые файлы.

При сборке программ с библиотеками в опциях компилятора (для `gcc` опция `-l`) не указывают префикс имен библиотек `lib`. Так, для библиотеки `librt.so.1` в опциях компилятора имя будет просто `rt`.

Исполняемые файлы, собранные с поддержкой разделяемых библиотек, требуют очень быстрого обращения к коду в файлах этих библиотек. Для этого

имеется специальная база данных — хешированный файл `/etc/ld.so.cache`. Этот файл бинарный. Он генерируется с помощью команды `ldconfig`.

При установке новых библиотек команда `ldconfig` обычно вызывается автоматически. При этом индексируются библиотеки, находящиеся в стандартных местах файловой системы: `/lib`, `/usr/lib` и `/usr/local/lib`, а также в каталогах, указанных в файле `/etc/ld.so.conf` (пример 18.14).

Пример 18.14. Файл `/etc/ld.so.conf`

```
$ cat /etc/ld.so.conf
/usr/local/lib
/usr/lib/openssl/lib
/usr/X11R6/lib
/usr/qt/3/lib
/opt/kde3/lib
```

ЗАДАНИЯ

- Определите, от каких библиотек зависит исполняемый файл демона BIND `/usr/local/sbin/named`, собранный и установленный в предыдущем разделе. Определите имена файлов библиотек и SO-name.
- Установлены ли в системе какие-либо библиотеки без участия системы управления пакетами (см. `/usr/local`)?

Менеджер пакетов RPM

В Red Hat Linux и подобных ему дистрибутивах используется менеджер пакетов RPM. Менеджер пакетов RPM реализован с помощью группы программ, главная из которых — `rpm`. Далее приведен список важнейших режимов работы RPM:

- ☐ запрос — включается опцией `-q` или `--query`;
- ☐ проверка целостности файлов пакета — включается опцией `-v` или `--verify`;
- ☐ проверка электронной подписи пакета — опции `-K` или `--checksig`;
- ☐ установка пакета — опции `-i` или `--install`;
- ☐ обновление пакета — работает с опцией `-U` или `--upgrade`;
- ☐ обновление версии пакета — опции `-F` или `--freshen`;
- ☐ удаление установленных пакетов — опции `-e` или `--erase`.

Используя режим запроса (опция `-q`), можно, например, узнать точную версию установленного в системе пакета (пример 18.15).

Пример 18.15. Получение полного имени пакета в RPM

```
$ rpm -q bash
bash-4.0-18.4.1.i586
```

В этом примере был осуществлен запрос к базе данных RPM. В качестве ключа запроса использовалось краткое название установленного в системе пакета — `bash`. В результате исполнения запроса было получено полное имя пакета.

Для получения более подробной информации о пакете следует воспользоваться опциями `-qi`. Использование опций `-qi` выводит подробную информацию о пакете, в том числе суммарный размер файлов, установленных из пакета (поле `Size`), и группу пакетов, к которой этот пакет принадлежит (поле `Group`).

В примере 18.16 выводится список файлов, установленных из пакета.

Пример 18.16. Получение списка файлов, установленных из пакета

```
$ rpm -ql logrotate
/etc/cron.daily/logrotate
/etc/logrotate.conf
/etc/logrotate.d
/usr/sbin/logrotate
/usr/share/doc/logrotate-3.7.8
/usr/share/doc/logrotate-3.7.8/CHANGES
/usr/share/man/man8/logrotate.8.gz
/var/lib/logrotate.status
```

В этом примере получен список файлов, установленных из пакета `logrotate`.

Можно ограничиться лишь выводом списка конфигурационных файлов, установленных из пакета, используя для этого опции `-qc`. Кроме этого, с помощью опций `-qd` можно получить информацию о справочных файлах и документации, установленных из этого пакета.

Если задано сочетание опций `-qa`, то будет получен список из всех пакетов, установленных в системе. Этим удобно пользоваться, если название пакета неизвестно.

Так, например, можно получить информацию обо всех пакетах, в имени которых имеется строка "ftp" (пример 18.17).

Пример 18.17. Поиск пакета по подстроке

```
$ rpm -qa | grep ftp
```

Можно узнать, из какого пакета установлен некоторый файл (пример 18.18).

Пример 18.18. Определение пакета по известному файлу

```
$ rpm -qf /usr/sbin/named  
bind-9.6.1P2-1.1.1
```

Сочетание опций `-qf` требует в качестве ключа запроса задать имя файла.

Для получения информации о пакете, еще не установленном в системе, можно указать опции `-qp`, позволяющие извлекать информацию из файла пакета (пример 18.19).

Пример 18.19. Получение информации непосредственно из RPM-пакета

```
$ rpm -qip Downloads/bind-9.6.1P2-1.2.src.rpm  
Name           : bind                      Relocations: (not relocatable)  
Version        : 9.6.1P2                  Vendor: openSUSE  
...
```

В примере 18.18 получена информация о пакете, который не установлен в системе, из файла пакета. Часть вывода пропущена для краткости.

Опция `-v` переключает `rpm` в режим проверки целостности файлов, установленных из пакета (пример 18.20).

Пример 18.20. Проверка целостности пакета

```
$ rpm -V openldap-servers  
S.5....T c /usr/share/openldap/migration/migrate_common.ph
```

Выводимая информация означает следующее: `s` — у файла не совпадает размер, `5` — не совпадает сигнатура `md5`, `T` — время модификации изменено.

Опция `-k` позволяет проверять электронную подпись файлов пакетов. Перед проверкой следует удостовериться, что электронный сертификат производителя пакета (дистрибутива) установлен в системе. Обычно сертификат нахо-

дится на первом диске установочного комплекта. Во многих дистрибутивах электронный сертификат производителя импортируется еще на стадии установки системы.

Пример 18.21. Импортирование сертификата

```
# rpm --import /media/cdrom/RPM-GPG-KEY*
```

Команда `rpm --import` импортировала ключ (см. пример 18.21), находящийся на первом диске установочного комплекта, из файлов, удовлетворявших шаблону `RPM-GPG-KEY*`.

Далее импортированный ключ может использоваться для проверки цифровых подписей пакетов от данного производителя (пример 18.22).

Пример 18.22. Проверка подлинности пакета

```
$ rpm -K Downloads/bind-9.6.1P2-1.2.src.rpm
Downloads/bind-9.6.1P2-1.2.src.rpm: rsa sha1 (md5) pgp md5 OK
```

В приведенном примере подтверждена надежность проверяемого пакета.

Установка или обновление пакета осуществляется с помощью опций:

- ❑ `-i` — пакет будет установлен при отсутствии в системе его предыдущей версии;
- ❑ `-U` — пакет будет установлен или обновлен в любом случае (Upgrade);
- ❑ `-F` — возможно только обновление версии пакета. Если предыдущей версии в системе нет, то установка произведена не будет.

При установке или обновлении пакетов в качестве аргумента для `rpm` указывают имена файлов пакетов (`.rpm`-файлы). Имеется возможность проверить последствия установки без проведения реальной установки пакетов с помощью опции `--test`.

Эта же опция полезна для проверки последствий удаления пакетов из системы без реального удаления (пример 18.23).

Пример 18.23. Проверка возможности удаления пакета

```
$ rpm --test -e mailx
error: Failed dependencies:
    mailx is needed by (installed) nagios-3.0.6-4.1.i586
    /usr/bin/mailx is needed by (installed) lsb-4.0-4.1.i586
```

Бывают случаи, когда установка или удаление пакета должны быть произведены, невзирая на нарушение зависимостей. Для этого предназначена опция `--nodeps`.

В дистрибутивах Red Hat имеется удобная утилита `yum`, значительно автоматизирующая поиск и установку пакетов и учитывающая зависимости пакетов. Важнейшие способы использования `yum`:

- ❑ `yum search <ИМЯ>` — поиск пакета по имени;
- ❑ `yum list available` — список доступных пакетов;
- ❑ `yum grouplist` — список всех установленных групп пакетов;
- ❑ `yum repolist` — список включенных репозиториях (источников) пакетов;
- ❑ `yum info <ИМЯ>` — вывод информации о пакете;
- ❑ `yum provides <ИМЯ>` — поиск пакета по его содержимому;
- ❑ `yum install <ИМЯ>` — установка пакета.

В других дистрибутивах обычно имеются аналогичные средства. Например, в SUSE пакеты можно устанавливать посредством программы конфигурирования `yast` и специализированной утилиты `zypper`.

Вот пример использования `zypper` для поиска и установки пакета (пример 18.24).

Пример 18.24. Использование `zypper` в Open SUSE

```
$ zypper se atftp
Loading repository data...
Reading installed packages...

S | Name | Summary | Type
--+-----+-----+-----
  | atftp | Advanced TFTP Server and Client | package
  | atftp | Advanced TFTP Server and Client | srcpackageerror

$ su
# zypper install atftp
```

Здесь показано, как в Open SUSE можно найти пакет по его имени и установить его с помощью `zypper`.

Задания

- Как в RPM увеличить информативность получаемых данных, например, при проверке целостности установленных пакетов?
- Получите имена всех пакетов, принадлежащих той же группе, которой принадлежит пакет, из которого был установлен менеджер пакетов RPM.
- Определите, каким образом можно получить список всех возможных ключей запроса к базе данных RPM?
- Выведите информацию о полном имени, группе и размере пакета `quota`.
- Получите список из десяти пакетов, файлы из которых занимают на диске максимальное место (на основе данных из самих пакетов).
- Используя опцию `--test`, исследуйте, можно ли удалить пакет `samba`?
- Какие опции выводят при установке пакета индикатор прогресса (`progress bar`) и процент выполнения установки?

Система управления пакетами Debian

Хотя система управления пакетами Debian позволяет устанавливать программное обеспечение как с CD/DVD-ROM или архивов на жестких дисках, так и с сетевых источников, структура ее в первую очередь ориентирована на удобство сетевой установки непосредственно с FTP-зеркала архива Debian. Структура размещения файлов на дисках установочного комплекта Debian GNU/Linux следует структуре архивов на FTP-серверах Debian.

Обычно в архивах представлены три типа дистрибутивов Debian:

- ☐ `stable` — стабильные;
- ☐ `unstable` — не прошедшие должного тестирования дистрибутивы;
- ☐ `frozen` — старые дистрибутивы, обновление которых прекращено.

Архив программного обеспечения Debian разделен на несколько секций. Критериями деления являются:

- ☐ соответствие программ критериям свободы, принятым в Debian;
- ☐ наличие или отсутствие экспортных ограничений на программное обеспечение, связанных с особенностями законодательства США;
- ☐ в пакете находятся скомпилированные программы или же исходный код;
- ☐ принадлежность пакета либо к стабильной ветви, либо к экспериментальной.

По критерию свободы программного обеспечения в архиве Debian выделены секции, приведенные в списке далее:

- ☐ `main` — программное обеспечение, полностью соответствующее критерию свободы Debian. Все пакеты, принадлежащие этой секции, не зависят от каких-либо пакетов, принадлежащих другим секциям;
- ☐ `contrib` — свободное программное обеспечение (по критерию Debian);
- ☐ `non-free` — программное обеспечение, не соответствующее критерию свободы;
- ☐ программное обеспечение, подпадающее под экспортные ограничения США, размещается в иерархии `non-US`. В иерархии `non-US` также выделены секции `main`, `contrib` и `non-free`.

Кроме того, все пакеты классифицированы по уровню приоритета:

- ☐ `required` — пакет должен быть установлен для правильной работы ОС;
- ☐ `important` — важные пакеты для UNIX подобных ОС;
- ☐ `standard` — утилиты командной строки;
- ☐ `optional` — дополнительные пакеты и пакеты X Window;
- ☐ `extra` — пакеты для специфического или частного применения, возможно, конфликтующие с пакетами из более приоритетных секций.

При работе с системой управления пакетами Debian используются программы:

- ☐ `dselect` — высокоуровневая программа управления пакетами, с интерфейсом меню и возможностью индивидуального выбора пакетов;
- ☐ `aptitude` — исключительно удобная высокоуровневая программа управления пакетами, ориентированная на текстовый интерфейс. Облегчает управление пакетами, т. к. визуализирует иерархическую структуру классификации пакетов, позволяя быстрее находить требуемые пакеты;
- ☐ `tasksel` — программа, позволяющая управлять целыми группами пакетов, относящимися к заранее определенным категориям (так называемые `tasks` — наборы пакетов). Эта программа идеальна при отсутствии необходимости индивидуального управления пакетами, т. к. позволяет разом установить целый набор пакетов, связанных с выполнением какой-либо задачи, например, программирования на языке Python;
- ☐ `dpkg` — низкоуровневая утилита с интерфейсом командной строки для индивидуального управления пакетами. Позволяет установить программное обеспечение непосредственно из файла пакета (`.deb`-файла);
- ☐ `apt-get` и `apt-cache` — утилиты с интерфейсом командной строки для индивидуального управления пакетами и манипуляций их источниками.

Утилиты `dselect`, `aptitude` и `tasksel` предоставляют удобный пользовательский интерфейс, базируясь на функциональности программ `dpkg` и `apt-get`. Есть множество других утилит для управления пакетами в Debian, в том числе и с графическим интерфейсом, которые также базируются на `dpkg` и `apt-get`.

Программы `dpkg` и `apt-get` предполагают, что база данных системы управления пакетами Debian размещается в каталоге `/var/lib/dpkg`. Здесь находятся файлы с информацией о списке доступных и установленных пакетов и о статусе пакетов.

Программа `dselect` предоставляет следующие возможности (в виде меню):

- ☐ выбор метода доступа к источнику пакетов;
- ☐ обновление списка доступных пакетов;
- ☐ выбор набора пакетов, установка флагов удаления и добавления пакетов;
- ☐ установка и обновление выбранных пакетов;
- ☐ настройка установленных, но не сконфигурированных пакетов;
- ☐ удаление ненужных пакетов.

На этапе выбора программного обеспечения с помощью `dselect` фактически происходит изменение флагов состояния пакетов в файле `/var/lib/dpkg/status`.

В режиме выбора пакетов в `dselect` можно отобразить как список пакетов, так и комментариев к пакету. Важнейшие команды, работающие в режиме выбора пакетов:

- ☐ + — выбрать пакет для установки;
- ☐ = — запретить обновление пакета;
- ☐ : — снять запрет обновления пакета;
- ☐ - — удалить пакет;
- ☐ _ — удалить пакет и его файлы конфигурации;
- ☐ i и I — изменить режим просмотра;
- ☐ v и V — изменить подробности информации;
- ☐ o и O — изменить сортировки.

Статус пакетов отображается с помощью флагов `ЕIOM`:

- ☐ E (Error) — ошибка;
- ☐ I (Installed state) — состояние установки;
- ☐ O (Old mark) — предыдущее состояние;
- ☐ M (Mark) — выбранное состояние.

Значением поля `e` может быть:

- ☐ пробел — при установке ошибок не возникло;
- ☐ `R` — наличие серьезной ошибки;
- ☐ `I` — при установке возникла ошибка, которую можно игнорировать.

Значением поля `i` может быть:

- ☐ пробел — пакет не установлен;
- ☐ `*` — пакет установлен;
- ☐ `-` — пакет не установлен или удален;
- ☐ `U` — пакет распакован, но не настроен (unpacked);
- ☐ `c` — не сконфигурирован полностью (half-configured);
- ☐ `I` — пакет установлен с ошибками или не полностью (half-installed).

Значениями полей `o` и `m` могут быть:

- ☐ `*` — пакет был установлен (поле `o`) или должен быть установлен (поле `m`);
- ☐ `-` — пакет был удален (поле `o`) или должен быть удален (поле `m`);
- ☐ `=` — пакет был запрещен к обновлению (поле `o`) или должен быть запрещен к обновлению (поле `m`);
- ☐ `_` — пакет был удален вместе с файлами конфигурации (поле `o`) или должен быть удален вместе с файлами конфигурации (поле `m`);
- ☐ `n` — пакет новый и не устанавливался.

Команда `dpkg` удобна для непосредственного обращения к файлам пакетов Debian. Наиболее часто используемые опции `dpkg`:

- ☐ `-I` — отобразить информацию о пакете;
- ☐ `-f` — вывести только требуемую информацию о пакете по заданным полям;
- ☐ `-c` — отобразить список файлов, содержащихся в пакете;
- ☐ `-x` — извлечь файлы из пакета в заданный каталог;
- ☐ `-i` — установить пакет.

Пример использования `dpkg` обращения к файлу пакета для получения информации о зависимостях и версии пакета (пример 18.25).

Пример 18.25. Проверка версии и зависимостей пакета

```
$ dpkg -f virtualbox-3.0_3.0.0-49315_i386.deb Depends Version
Version: 3.0.0-49315_Ubuntu_jaunty
```



```
Depends: libc6 (>= 2.7), libcurl3 (>= 7.16.2-1), libgcc1 (>= 1:4.1.1),
libqt4-network (>= 4.5.0~+rc1), libqtcore4 (>= 4.5.0~+rc1), libqtgui4 (>=
4.5.0~+rc1), libstdl1.2debian (>= 1.2.10-1), libssl0.9.8 (>= 0.9.8f-5),
libstdc++6 (>= 4.2.1), libx11-6, libxcursor1 (>= 1.1.2), libxext6,
libxml2 (>= 2.6.27), libxmu6, libxslt1.1 (>= 1.1.18), libxt6, zlib1g (>=
1:1.1.4), psmisc, adduserVersion: 0.3.17-1
```

Для удаления пакетов с помощью `dpkg` указывается опция `-r`. При таком удалении пакетов конфигурационные файлы не удаляются. Если необходимо произвести полное удаление пакета, то следует использовать опцию `-P`.

Команда `dpkg -l` позволяет получить список всех установленных в системе пакетов или сведения о пакете, указанном в качестве аргумента (пример 18.26).

Пример 18.26. Проверка наличия пакета

```
$ dpkg -l bash
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Cfg-files/Unpacked/Failed-cfg/Half-inst/trig-
aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Имя Версия Описание
+++-+-----+-----+
=====
ii bash 4.0-5ubuntu2 The GNU Bourne Again SHell
```

При необходимости получить список файлов, установленных из пакета, следует использовать опцию `-L` команды `dpkg` (пример 18.27).

Пример 18.27. Получение списка файлов, установленных из пакета

```
$ dpkg -L info
```

В примере 18.27 список файлов, установленный из пакета `info`, не показан для краткости.

Программа `dpkg` предоставляет возможность определить, из какого пакета был установлен некоторый файл. Для этого следует использовать опцию `-s` (пример 18.28).

Пример 18.28. Определение пакета, из которого был установлен файл

```
$ dpkg -S /etc/bash.bashrc
bash: /etc/bash.bashrc
```

Файл `/etc/bash.bashrc` был установлен из пакета `bash`.

Как и `dpkg`, команда `apt-get` предназначена для установки пакетов из командной строки. Но, в отличие от `dpkg`, `apt-get` работает с источниками пакетов (комплекты CD/DVD-ROM, FTP и т. д.) и учитывает зависимости пакетов, позволяя автоматизировать их установку.

Пакет, который требуется установить, при использовании `apt-get` указывают не с помощью имени файла пакета, а при помощи имени самого пакета. При этом получение, установка и конфигурирование пакета будут произведены автоматически.

Перед установкой пакетов следует обновить их список командой `apt-get update`. Список источников пакетов находится в файле `/etc/apt/sources.list`. Причем если среди источников имеются как локальные (например, CD-ROM), так и удаленные, то рекомендуется в начале файла `/etc/apt/sources.list` указывать локальные источники.

При необходимости добавления в список источников нового CD-диска с пакетами Debian следует использовать команду `apt-cdrom add`. Существует также удобная команда для настройки сетевых источников пакетов. Она называется `netselect` и может быть установлена из одноименного пакета. С помощью `netselect` можно выбрать сервер, передача пакетов с которого осуществляется максимально быстро.

Поиск пакетов выполняется с помощью команды `apt-cache` (пример 18.29).

Пример 18.29. Поиск пакета с помощью `apt-cache`

```
$ apt-cache search '^mc$'
mc - midnight commander - a powerful file manager
```

Обратите внимание, что утилита `apt-cache` производит поиск на основе регулярных выражений, трактуя аргумент, как подстроку поиска. Поэтому в примере 18.29 в качестве аргумента была указана строка из символов `mc` и ничего более.

Для установки пакета с помощью `apt-get` достаточно указать его имя в качестве аргумента команды `apt-get install` (пример 18.30).

Пример 18.30. Установка пакета с помощью `apt-get`

```
# apt-get install mc
```

В примере 18.30 команда `apt-get` загрузит и установит пакет `mc`. Вывод команды пропущен для краткости.

Чаще всего производятся следующие действия с `apt-get`:

- ☐ `check` — проверка состояния системы управления пакетами;
- ☐ `update` — обновление информации о доступных пакетах;
- ☐ `upgrade` — обновление системы установкой новых версий пакетов;
- ☐ `dselect-upgrade` — обновление системы в соответствии с изменением поля `status` доступных и установленных пакетов;
- ☐ `dist-upgrade` — аналогично `upgrade`, но с более интеллектуальным подходом к разрешению возможных конфликтов пакетов;
- ☐ `install` — установка пакета;
- ☐ `remove` — удаление пакета;
- ☐ `clean` — очистка каталогов `/var/cache/apt/archives` и `/var/cache/apt/archives/partial`, которые, соответственно, содержат файлы пакетов Debian и файлы пакетов, находящиеся в состоянии приема из источника, например, с FTP-сервера;
- ☐ `source` — получение исходного пакета для сборки требуемого пакета.

Наиболее важные опции команды `apt-get`:

- ☐ `-d` — команда `apt-get` только загрузит пакеты, но не будет устанавливать их;
- ☐ `-s` — команда не будет производить реальных действий, а только симулирует последствия планируемых действий, например, возникновение конфликтов;
- ☐ `-y` — отключение интерактивного режима работы программы, на все вопросы, которые обычно задает `apt-get`, будет дан утвердительный ответ;
- ☐ `-u` — команда покажет обновленные пакеты и сведения об их состоянии.

Задания

- С помощью `apt-cdrom` добавьте в список источников диск из установочного комплекта.
- Используя `apt-get`, установите программу `netselect`.
- Определите, какое зеркало FTP архива Debian является наиболее предпочтительным с точки зрения скорости приема пакетов с него.



Глава 19

Установка аппаратного обеспечения

Системный администратор должен уметь устанавливать аппаратное обеспечение и соответствующие драйверы к нему. В этой главе предоставлены основные сведения о том, как это делается в GNU/Linux. Здесь также рассказано о ядре Linux и работе с модулями ядра.

Установка нового оборудования

Подключение оборудования в компьютерах с архитектурой x86/64 может быть произведено двумя основными способами:

- ☐ установкой плат дополнительного оборудования в слоты расширения;
- ☐ с помощью подключения к внешним шинам или портам.

При установке оборудования в слоты расширения на материнской плате это оборудование непосредственно подключается к локальным шинам компьютера. Оборудование, подключаемое к внешним шинам, часто подключается к системе посредством плат расширения, включенных в локальную шину. Как локальные, так и внешние шины компьютера с помощью согласующего оборудования и контроллеров управления подключаются к центральному процессору.

Для того чтобы оборудование могло осуществлять операции ввода/вывода, ему должны быть выделены следующие ресурсы:

- ☐ IRQ — канал прерывания;
- ☐ IO/Base — базовый адрес ввода/вывода;
- ☐ DMA — канал прямого доступа к ОЗУ.

Прерывание (IRQ) идентифицируется своим номером и позволяет сигнализировать о завершении операции ввода/вывода, осуществляемой данным

устройством. При получении прерывания процессор должен приостановить выполнение текущего задания и переключиться на обработку прерывания с помощью специальной программы (interrupt handler). Список прерываний, использованных установленными в системе устройствами, можно увидеть в файле `/proc/interrupts`.

Регистры памяти, имеющиеся в устройствах расширения, отображаются в специальную область ОЗУ, доступную ядру операционной системы и называемую *памятью устройств*. Эта память предназначена для реализации операций ввода/вывода, т. е. в нее записывается передаваемая при этих операциях информация. *Базовый адрес ввода/вывода* IO/Base указывает начало области памяти для операций ввода/вывода данного устройства. Принято записывать адреса ввода/вывода в шестнадцатеричном виде. Увидеть занятые устройствами адреса ввода/вывода можно в файле `/proc/ioproports`.

ЗАДАНИЯ

- Получите список занятых в системе прерываний.
- Какое-либо устройство в вашей системе использует базовый адрес порта ввода/вывода `0x320`?

Работа с модулями ядра

Современные ядра Linux построены в соответствии с модульной архитектурой. Ядра Linux поддерживают включение модулей ядра без необходимости пересборки ядра и перезагрузки системы. Это позволяет изменять функциональность ядра в работающей системе, а также включать в ядро и удалять драйверы устройств "на лету".

Модули ядра представляют собой объектные файлы с кодом, который можно подключать к работающему ядру Linux. Модуля ядра компилируются при сборке ядра. Процедура сборки ядра будет рассмотрена далее в этой главе. Модули располагаются в каталоге `/lib/modules/<имя ядра>`. Имена подкаталогов `/lib/modules` всегда точно соответствуют именам ядер, имеющихся в системе. Имя работающего сейчас ядра можно узнать, выполнив команду `uname -r` (пример 19.1).

Пример 19.1. Каталог `/lib/modules`

```
$ ls /lib/modules
2.6.31.8-0.1-desktop  2.6.31.8-0.1-xen
```

```
$ uname -r
```

2.6.31.8-0.1-desktop

```
$ ls /lib/modules/`uname -r`
build misc modules.alias.bin modules.builtin.bin modules.dep.bin
modules.symbols source vdso kernel modules.alias modules.builtin
modules.dep modules.order modules.symbols.bin systemtap weak-updates
```

Обратите внимание, что в каталоге `/lib/modules` имеется подкаталог (или подкаталоги) с именем ядра. Основная часть модулей ядра содержится в подкаталоге `kernel/`.

В ядрах серии 2.6 в `kernel/` обычно имеются следующие подкаталоги:

- ☐ `arch` — модули ядра, зависящие от архитектуры;
- ☐ `crypto` — криптографические модули ядра;
- ☐ `drivers` — драйверы для различных устройств;
- ☐ `fs` — модули поддержки файловых систем;
- ☐ `kernel` — служебный модуль;
- ☐ `lib` — библиотеки, используемые ядром;
- ☐ `net` — поддержка сетевой инфраструктуры;
- ☐ `sound` — поддержка звуковой инфраструктуры.

Для получения списка модулей, установленных в настоящее время в ядро, следует выполнить команду `/sbin/lsmmod`, которая выводит в форматированном виде информацию из файла `/proc/modules` (пример 19.2).

Пример 19.2. Фрагмент списка загруженных модулей ядра

```
# lsmmod
Module                Size  Used by
vboxnetadp             9312   0
vboxnetflt            18632   0
vboxdrv               202888   2 vboxnetadp,vboxnetflt
nls_iso8859_1           4000   0
nls_cp437              5664   0
vfat                  12544   0
fat                   58400   1 vfat
nfs                   342032   0
lockd                  81228   1
```

Информацию о модуле выводит команда `/sbin/modinfo` (пример 19.3).

Пример 19.3. Вывод информации о модуле ядра

```
# modinfo vboxdrv
filename:      /lib/modules/2.6.31.8-0.1-desktop/misc/vboxdrv.ko
version:      3.1.2 (0x00100001)
license:      GPL
description:   VirtualBox Support Driver
author:       Sun Microsystems, Inc.
srcversion:   F03437979AB40995D202F18
depends:
vermagic:     2.6.31.8-0.1-desktop preempt mod_unload modversions 686
parm:         force_async_tsc:force the asynchronous TSC mode (int)
```

Информация о модулях включается в сами модули, хотя она при необходимости может быть скрыта.

Новые модули могут быть загружены в работающее ядро Linux с помощью команды `/sbin/insmod` (пример 19.4). Установить модуль в ядро может только суперпользователь. В качестве аргумента для команды `insmod` указывают имя файла модуля.

Пример 19.4. Установка модуля командой `insmod`

```
# insmod \
> /lib/modules/2.6.31.8-0.1-desktop/kernel/drivers/scsi/scsi_debug.ko
# lsmod | grep scsi
scsi_debug          62688  0
```

В примере 19.4 был установлен модуль `scsi_debug.ko`. Команда `lsmod` отобразила в списке модулей установленный модуль.

Удалить модуль из ядра можно с помощью команды `/sbin/rmmod` (пример 19.5).

Пример 19.5. Удаление модуля командой `rmmod`

```
# rmmod scsi_debug
# lsmod | grep scsi
```

Команда `rmmod` в примере 19.5 удалила модуль `scsi_debug.ko`.

Часто модули ядра для работоспособности требуют наличия других модулей ядра. В каталоге, содержащем модули ядра (`/lib/modules/`uname -r``), находится текстовый файл `modules.dep` с зависимостями модулей друг от друга. Файл `modules.dep` генерируется с помощью утилиты `/sbin/depmod`.

Вместо команд `insmod` и `rmmod` предпочтительнее вызывать команду `modprobe`, обеспечивающую более интеллектуальное поведение (пример 19.6). Эта команда использует файл `modules.dep` для автоматического учета зависимостей модулей, что позволяет загружать модули ядра, не задумываясь об их зависимостях. Они будут учтены, и требуемые модули будут загружены.

Пример 19.6. Установка модуля командой `modprobe`

```
# modinfo ext4
filename:      /lib/modules/2.6.31.8-0.1-desktop/kernel/fs/ext4/ext4.ko
...
depends:       jbd2,crc16
...
# modprobe -v ext4
insmod /lib/modules/2.6.31.8-0.1-desktop/kernel/lib/crc16.ko
insmod /lib/modules/2.6.31.8-0.1-desktop/kernel/fs/jbd2/jbd2.ko
insmod /lib/modules/2.6.31.8-0.1-desktop/kernel/fs/ext4/ext4.ko
```

Обратите внимание на то, что команда `modprobe -v`, выполненная в примере 19.6, загрузила сразу три модуля в силу того, что модуль для файловой системы `ext4` зависит от `ext4.ko` и `jbd2.ko`. Опция `-v` была использована для вывода информации о загрузке модулей. Вывод команды `modinfo` в примере был сокращен.

По умолчанию команда `modprobe` устанавливает модуль в ядро, при вызове с опцией `-r` она способна удалять модули (пример 19.7).

Пример 19.7. Удаление модулей командой `modprobe`

```
# modprobe -v -r ext4
rmmod /lib/modules/2.6.31.8-0.1-desktop/kernel/fs/ext4/ext4.ko
rmmod /lib/modules/2.6.31.8-0.1-desktop/kernel/fs/jbd2/jbd2.ko
rmmod /lib/modules/2.6.31.8-0.1-desktop/kernel/lib/crc16.ko
```

При необходимости выгрузить все модули ядра, не используемые в настоящий момент, можно использовать команду `modprobe -r` без аргументов.

Одно из полезных свойств команды `modprobe` — способность находить файлы модулей, имена которых соответствуют заданному шаблону (пример 19.8). Для этого следует использовать опцию `-l`. В именах модулей в таком случае допускается использовать шаблоны, аналогичные шаблонам для имен файлов в оболочке.

Пример 19.8. Поиск модулей по шаблону командой `modprobe`

```
# modprobe -l 'e100*'
kernel/drivers/net/e1000/e1000.ko
kernel/drivers/net/e1000e/e1000e.ko
kernel/drivers/net/e100.ko
```

Использование опции `-a` команды `modprobe` позволяет установить сразу несколько модулей, имена которых соответствуют заданному в качестве аргумента шаблону.

Если модуль требует передачи ядру некоторой информации, то ее можно указывать в качестве аргументов команды `modprobe` (пример 19.9).

Пример 19.9. Передача параметров модулю

```
# modprobe e1000 speed=100
```

Команда `modprobe` имеет конфигурационный файл `/etc/modprobe.conf`. Этот файл позволяет указывать параметры, передаваемые модулям. Помимо этого в данном файле можно задавать команды, которые надо выполнить до или после загрузки или удаления модулей. Также в этом файле могут задаваться псевдонимы для модулей, делающие работу с ними более удобной. Фрагмент файла `/etc/modprobe.conf` приведен в примере 19.10.

Пример 19.10. Файл `/etc/modprobe.conf`

```
alias eth0 e1000
options e1000 Speed=100
```

В файле `/etc/modprobe.conf`, приведенном в примере 19.10, указано, что драйвером для Ethernet-адаптера `eth0` является `e1000`. Это модуль драйвера Intel Gigabit Ethernet, настроенный на скорость 100 Мбит/с, что установлено директивой `options`.

Задания

- Получите список модулей, загруженных в ядро в вашей системе.
- Узнайте, кто является автором модуля `e1000`.
- Определите зависимости модуля `e1000` от других модулей.
- Для какого оборудования предназначен модуль `e1000`?

Файлы устройств и `udev`

GNU/Linux предоставляет доступ к устройствам посредством файлов устройств. В *главе 11* было рассказано о файлах устройств. Вы уже знаете, что они бывают двух типов: символьные и блочные. Также вам известно, что вместо размера файлов устройств команда `ls -l` выводит два значения: мажор и минор. Мажор — это номер драйвера для данного устройства в ядре, минор — дополнительный параметр, чаще всего нумерующий экземпляр устройства.

Традиционно файлы устройств создавались с помощью команды `mknod` и размещались в каталоге `/dev`. Причем из соображений универсальности создавались файлы устройств для аппаратуры, не установленной в компьютере. Однако разнообразных файлов устройств становится все больше и больше, поэтому при использовании статически созданных файлов устройств возникают три проблемы:

- ❑ нехватка мажоров и миноров для адресации файлов устройств;
- ❑ загромождение `/dev` файлами отсутствующих реально устройств (например, в Fedora 1 в каталоге `/dev` имелось порядка 1800 файлов устройств);
- ❑ неудобство в работе подключаемых на лету устройств (hot-pluggable devices), например, USB.

Проблема нехватки мажоров и миноров решается относительно просто, но проблема загромождения `/dev` не так проста. Для ее решения во многих UNIX-системах используют динамически создаваемые файлы устройств с помощью файловой системы `devfs`. Идея такова: в ядре имеется информация, какие устройства установлены в системе (при наличии соответствующих драйверов). Если в системе появляется новое устройство, для него динамически создается новый файл устройства.

Есть еще одна проблема: поскольку имена файлов устройств, создаваемых `devfs`, не являются жестко связанными с конкретными аппаратными средствами, это очень затрудняет работу множества средств автоматизации. Например, если для USB-флэш один раз создается файл устройства `/dev/sdb`, а другой — `/dev/sdc`, то как определить точку монтирования?

Для решения этих проблем в ядрах 2.6.13 и выше используется udev — дальнейшее развитие devfs, предоставляющее возможность "стабилизации" именования динамически создаваемых файлов устройств с помощью специальных правил (rules).

Ядро Linux предоставляет файловую систему sysfs, монтируемую в каталог /sys. В этой файловой системе хранится служебная информация, с помощью которой udev создает и удаляет файлы устройств в /dev.

В udev входят:

- ❑ udevd — демон, отслеживающий сообщения ядра и передающий их udev;
- ❑ udevcontrol — программа для управления udev;
- ❑ udevinfo — программа для получения информации об устройствах (см. пример 19.13);
- ❑ ata_id, edd_id, scsi_id и vold_id — программы для идентификации устройств;
- ❑ udevmonitor — отладочная программа для отслеживания сообщений ядра;
- ❑ udevtest — симулятор udev.

Содержимое каталога /dev находится на временной файловой системе, и файлы устройств создаются там динамически. Статические файлы устройств копируются в /dev при старте системы из /lib/udev/devices (пример 19.11).

Пример 19.11. Статические файлы устройств в /lib/udev/devices

```
$ ls /lib/udev/devices/  
console core fd kmsg lp0 net null ppp ptmx pts shm stderr  
stdin stdout tty tty0 tty1 zero
```

Конфигурационные файлы для udev размещаются в /etc/udev. Прежде всего это правила (rules), с помощью которых определяются имена для создаваемых файлов устройств. Всякий раз, когда устройство добавляется или удаляется из системы, ядро генерирует событие uevent для информирования демона udevd, который в соответствии с правилами в /etc/udev/rules.d/*.rules создает или удаляет файлы устройств.

Каждому устройству сопоставляется специальный идентификатор MODALIAS. Список поддерживаемых устройств конкретным драйвером находится в соответствующем этому драйверу модуле ядра. Программа depmod читает эти идентификаторы и записывает их в файл modules.alias в каталоге с модулями ядра (пример 19.12).

Пример 19.12. Файл modules.alias

```
# modinfo -a kingsun_sir
alias:          usb:v07C0p4200d*dc*dsc*dp*ic*isc*ip*

# grep -i kingsun /lib/modules/`uname -r`/modules.alias
alias usb:v07C0p4200d*dc*dsc*dp*ic*isc*ip* kingsun_sir
```

В примере получен идентификатор модуля для IrDA-USB KingSun с помощью команды `modinfo -a`. Этот же MODALIAS имеется в файле `modules.alias`.

Наличие файла `modules.alias` обеспечивает автоматическое подключение требуемых модулей ядра для устройств, опознанных `udev`. Причем на ранних стадиях загрузки полная инициализация `/dev` невозможна, т. к. при этом может быть еще не готова инфраструктура `udev`. Поэтому после полной инициализации `udev` производит сбор информации в специальных триггерах для устройств в каталоге `/sys` о событиях `uevent`, которые были отложены для последующей обработки `udev`.

Получить информацию об устройстве, опознанном `udev`, можно с помощью команды `udevinfo` (пример 19.13).

Пример 19.13. Получение информации udev об устройстве

```
$ udevinfo -n /dev/sdb -q all
P: /block/sdb
N: sdb
S: disk/by-id/usb-Myson_Century__Inc._USB_Mass_Storage_Device_100
S: disk/by-path/usb-100:0:0:0
E: ID_VENDOR=Myson_Century,_Inc.
E: ID_MODEL=USB_Mass_Storage_Device
E: ID_REVISION=b007
E: ID_SERIAL=Myson_Century,_Inc._USB_Mass_Storage_Device_100
E: ID_TYPE=floppy
E: ID_BUS=usb
E: ID_PATH=usb-100:0:0:0
```

Опция `-n` команды `udevinfo` позволяет указать требуемое устройство соответствующим ему файлом, а опция `-q all` выводит всю доступную `udev` информацию об устройстве. Обратите внимание на строку `P: /block/sdb` — это путь для поиска информации об устройстве в `sysfs` (пример 19.14).

Пример 19.14. Информация об устройстве в sysfs

```
$ find /sys -name '*block*sdb'
/sys/devices/pci0000:00/0000:00:1d.7/usb1/1-7/1-7:1.0/host2/target2
:0:0/2:0:0:0/block:sdb
```

Найденный в примере 19.14 файл — это символическая ссылка на подкаталог в `/sys`, содержащий файлы с информацией об устройстве.

Можно проследить события `uevent`, происходящие в системе при добавлении или удалении устройства. Это делается с помощью `udevadm monitor` (пример 19.15).

Пример 19.15. Мониторинг событий uevent

```
# udevadm monitor --env
monitor will print the received events for:
UDEV - the event which udev sends out after rule processing
KERNEL - the kernel uevent

KERNEL[1263115065.575286] add
/devices/pci0000:00/0000:00:1d.7/usb2/2-2 (usb)
UDEV_LOG=3
ACTION=add
DEVPATH=/devices/pci0000:00/0000:00:1d.7/usb2/2-2
```

В примере 19.15 приведен фрагмент вывода информации о работе `udev` при подключении USB-носителя.

Если подключенное устройство не опознается системой, первое, что имеет смысл сделать — изучить сообщения ядра с помощью команды `dmesg` (пример 19.16).

Пример 19.16. Команда dmesg

```
# dmesg
[137937.482077] usb 2-2: new high speed USB device using ehci_hcd and
address 5
...
[137939.043375] sd 8:0:0:0: [sdb] Attached SCSI removable disk
```

В примере 19.16 приведен укороченный фрагмент вывода сообщений ядра, выполненный командой `dmesg` при подключении USB-носителя.

ЗАДАНИЯ

- Проверьте, используется ли в вашей системе udev.
- Получите с помощью udevinfo информацию о жестком диске.

Устройства PCI

Для получения списка PCI-устройств, работающих в системе, достаточно выполнить команду `lspci` (пример 19.17).

Пример 19.17. Команда `lspci`

```
# lspci
00:00.0 Host bridge: Intel Corporation Mobile PM965/GM965/GL960 Memory
Controller Hub (rev 0c)
...
04:00.0 Ethernet controller: Broadcom Corporation NetLink BCM5787M
Gigabit Ethernet PCI Express (rev 02)
...
```

В примере 19.17 показан фрагмент списка PCI-устройств, обнаруженных в системе.

Можно также получить подробную информацию об устройстве (пример 19.18).

Пример 19.18. Подробная информация об устройстве PCI

```
# lspci -s 04:00.0 -v
04:00.0 Ethernet controller: Broadcom Corporation NetLink BCM5787M
Gigabit Ethernet PCI Express (rev 02)
    Subsystem: Lenovo Device 20d5
    Flags: bus master, fast devsel, latency 0, IRQ 32
    Memory at fe000000 (64-bit, non-prefetchable) [size=64K]
    Expansion ROM at <ignored> [disabled]
    Capabilities: [48] Power Management version 3
    Capabilities: [50] Vital Product Data
    Capabilities: [58] Vendor Specific Information <?>
    Capabilities: [e8] MSI: Enable+ Count=1/1 Maskable- 64bit+
    Capabilities: [d0] Express Endpoint, MSI 00
    Capabilities: [100] Advanced Error Reporting
    Capabilities: [13c] Virtual Channel <?>
```

```
Capabilities: [160] Device Serial Number 00-1a-6b-ff-fe-cd-bd-36
Capabilities: [16c] Power Budgeting <?>
Kernel driver in use: tg3
```

Полученная в примере подробная информация о сетевом адаптере PCI выведена командой `lspci` с опциями `-v` (подробный вывод) и `-s` (выбор устройства).

ЗАДАНИЯ

- Получите список устройств PCI в системе.
- Проверьте IRQ сетевого адаптера с помощью подробного вывода `lspci`.

Установка SCSI-устройств

SCSI расшифровывается как Small Computer Systems Interface — системный интерфейс для малых компьютеров. Современный интерфейс SAS (Serial Attached SCSI) — дальнейшее развитие этого стандарта. Наиболее распространенными типами SCSI-устройств являются накопители на магнитных дисках.

Для обеспечения работоспособности SCSI-устройств требуется:

1. Подключить SCSI-устройства и хост-адаптер к шине и терминировать ее.
2. Установить хост-адаптер SCSI и драйвер к нему.
3. Установить SCSI ID для устройств. По соглашению для хост-адаптера SCSI следует установить SCSI ID 7.
4. Просканировать с помощью программы, реализованной в SCSI BIOS, SCSI-шину для проверки исправности устройств и правильности установки SCSI ID.
5. Если жесткие магнитные диски SCSI не были форматированы утилитой в SCSI BIOS, отформатировать их.
6. При необходимости установить драйверы для SCSI-устройств.

Поддержка SCSI в ядрах Linux серии 2.6 представлена драйверами трех типов (см. `/usr/src/linux/Documentation/scsi/scsi.txt`).

❑ Верхний уровень, наиболее приближенный к процессам пользователя (user space), реализован с помощью драйверов типов устройств:

- `sd` — драйвер жестких магнитных дисков SCSI, модуль ядра `sd_mod.ko`;
- `sr` — драйвер SCSI CD/DVD-дисков, модуль ядра `sr_mod.ko`;
- `st` — драйвер для ленточных устройств, модуль `st.ko`;
- `sg` — условное обобщенное символьное SCSI-устройство, модуль `sg.ko`.

- ❑ Средний обобщающий функциональность SCSI-уровень — `scsi_mod.ko`.
- ❑ Нижний уровень представлен драйверами для хост-адаптеров SCSI, модули для которых находятся в каталоге `/lib/modules/$(uname -r)/kernel/drivers/scsi/`.

Команда `lsscsi` выводит список установленных SCSI-устройств (пример 19.19).

Пример 19.19. Команда `lsscsi`

```
# lsscsi
[0:0:0:0]    disk      ATA          HITACHI HTS54161 SB4I   /dev/sda
[3:0:0:0]    cd/dvd   MATSHITA   DVD-RAM UJ-860   RB01   /dev/sr0
[8:0:0:0]    disk      silicon-power  PMAP   /dev/sdb

# lsscsi -v 8:0:0:0
[8:0:0:0]    disk      silicon-power  PMAP   /dev/sdb
    dir: /sys/bus/scsi/devices/8:0:0:0
[/sys/devices/pci0000:00/0000:00:1d.7/usb2/2-2/2-2:1.0/host8/
target8:0:0/8:0:0:0]
```

Выведенный командой `lsscsi` в примере 19.19 список устройств содержит также и SATA- и USB-устройства, т. к. в Linux обращение к ним производится через SCSI-инфраструктуру. Вторая команда примера показывает, как можно получить подробную информацию об устройстве.

Задания

- Если в системе используется CD-RW-дисковод, то проверьте, загружены ли модули для SCSI CD и эмуляции SCSI.
- Получите список SCSI-устройств в системе.

Установка сетевых адаптеров Ethernet

Драйверы сетевых адаптеров находятся в `/lib/modules/$(uname -r)/kernel/drivers/net/`.

Распространенные компьютеры с платформой x86 в настоящее время чаще всего комплектуются сетевыми адаптерами Ethernet стандарта PCI. Для их установки достаточно просто добавить в ядро соответствующий модуль.

Пример 19.20. Загрузка драйвера сетевой платы

```
# /sbin/modprobe -v e100
```


В примере 19.20 установлен драйвер для платы Intel EtherExpress100.

Для сетевых Ethernet-интерфейсов в ядрах Linux применяются имена вида `eth0` — для первого сетевого интерфейса, `eth1` — для второго и т. д. Для них в Linux не создаются файлы устройств в каталоге `/dev`.

Если модулю ядра драйвера сетевого адаптера необходимо передать какой-либо параметр, его можно указать в `/etc/modprobe.conf` (пример 19.21).

Пример 19.21. Параметры в `/etc/modprobe.conf`

```
alias eth0 e1000
options e1000 speed=100
```

В примере 19.21 для модуля драйвера сетевой платы Intel PRO/1000 устанавливается скорость 100 Мбит/с.

Система `udev` фиксирует имена сетевых устройств для того, чтобы при их добавлении или удалении сетевым адаптерам назначались старые имена и их настройка не сбилась. Привязка имен к интерфейсам достигается с помощью MAC-адресов в файле правил `udev /etc/udev/rules.d/70-persistent-net.rules` (пример 19.22).

Пример 19.22. Привязка имен сетевых интерфейсов

```
# PCI device 0x1022:0x2000 (pcnet32)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
ATTR{address}=="08:00:27:df:ba:12", ATTR{type}=="1", KERNEL=="eth*",
NAME="eth0"

# PCI device 0x1022:0x2000 (pcnet32)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
ATTR{address}=="08:00:27:00:27:d4", ATTR{type}=="1", KERNEL=="eth*",
NAME="eth1"
```

Задания

- Проверьте с помощью `lspci`, установлен ли в системе адаптер PCI Ethernet.
- Узнайте, имеются ли специальные опции, которые можно передавать драйверу Ethernet. Можно ли его включить в режиме 100 Мбит/с полного дуплексного режима?

Работа со звуковыми картами

В ядрах серии 2.6 звуковая подсистема реализована с помощью ALSA, причем она обеспечивает поддержку старой системы OSS.

Драйвер ALSA для звуковых карт ISA поддерживает расширение ISA PnP, поэтому необходимость конфигурирования ее с помощью утилит пакета `isapnptools` отпадает. При использовании ALSA параметры звуковых карт можно просматривать и изменять с помощью файловой системы `/proc` в подкаталоге `asound/`.

В ALSA основным модулем является `snd.ko`, он требуется для всех остальных драйверов ALSA. Все имена модулей драйверов ALSA начинаются с префикса `snd-`.

Для эмуляции PCM OSS применяется модуль `snd-pcm-oss.ko`. Для обеспечения работы звуковых карт USB в ALSA используется модуль `snd-usb-audio.ko`.

В примере 19.23 показаны загруженные модули ALSA.

Пример 19.23. Модули системы ALSA

```
$ /sbin/lsmmod | grep snd
snd_pcm_oss          51616  0
snd_mixer_oss        19072  1 snd_pcm_oss
snd_seq              64752  0
snd_seq_device        8620  1 snd_seq
snd_hda_codec_conexant 34240  1
snd_hda_intel         31680  2
snd_hda_codec         94688  2 snd_hda_codec_conexant,snd_hda_intel
snd_hwdep             8708  1 snd_hda_codec
snd_pcm              96324  3 snd_pcm_oss,snd_hda_intel,snd_hda_codec
snd_timer            25960  2 snd_seq,snd_pcm
snd                   75236  14
snd_pcm_oss,snd_mixer_oss,snd_seq,snd_seq_device,snd_hda_codec_conexant,
snd_hda_intel,snd_hda_codec,snd_hwdep,snd_pcm,snd_timer
snd_page_alloc       10600  2 snd_hda_intel,snd_pcm
```

Задания

- Получите сообщения ядра, выданные при его загрузке, используя `dmesg`. Содержатся ли в выводе этой команды сведения о звуковой карте? Также проверьте содержимое файла `/var/log/messages`.
- Проверьте наличие модулей ALSA среди загруженных модулей ядра.

Поддержка USB

Считывать и изменять настройки USB-устройств можно в `/proc/bus/usb`. Однако имеется специальная команда `lsusb`, выводящая список USB-устройств (пример 19.24).

Пример 19.24. Команда `lsusb`

```
# lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 005: ID 13fe:1f23 Kingston Technology Company Inc.
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 002: ID 147e:2016 Upek Biometric Touchchip/Touchstrip
Fingerprint Sensor
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 005 Device 004: ID 045e:00e1 Microsoft Corp. Wireless Laser Mouse
6000 Reciever
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

Можно также получить подробную информацию об устройстве (пример 19.25).

Пример 19.25. Фрагмент подробного вывода `lsusb`

```
# lsusb -d 147e:2016 -v
Bus 003 Device 002: ID 147e:2016 Upek Biometric Fingerprint Sensor
Device Descriptor:
  bLength                18
  bDescriptorType        1
  bcdUSB                  1.00
  ...
```

В примере 19.25 получена подробная информация о биометрическом сенсоре (приведен фрагмент вывода). Опция `-d` позволяет указать требуемое USB-устройство.

ЗАДАНИЕ

- Проверьте наличие в вашей системе USB-устройств. Если они есть, то определите, какие модули для USB загружены сейчас.
- Получите подробную информацию о USB-накопителе (флэш).

Устройства PCMCIA

Несмотря на свое название PCMCIA (PC Memory Card International Association, Международная ассоциация карт памяти для PC), с помощью таких карт можно подключать и иные устройства. Для подключения PCMCIA-карт необходимо иметь PCMCIA-бридж. В настоящее время имеются два типа карт: PC card — устаревшие 16-битные устройства, напоминающие устройства ISA, и современные 32-битные устройства CardBus. Они трактуются, как специализированные PCI-устройства.

Ранее для работы с PCMCIA-устройствами использовался пакет `pcmcia-cs`, главной утилитой которого являлась `/sbin/cardmgr`, с помощью которой обрабатывались события PCMCIA-устройств. Ныне он устарел и заменен пакетом `pcmciautils`. Обработка событий сейчас выполняется средствами `udev`.

В состав пакета `pcmciautils` входит команда `lspcmcia`, которая выводит список опознанных PCMCIA-устройств в системе (пример 19.26).

Пример 19.26. Команда `lspcmcia`

```
# lspcmcia
Socket 0 Bridge:          [yenta_cardbus]      (bus ID: 0000:15:00.0)

# lspci -s 0000:15:00.0
15:00.0 CardBus bridge: Ricoh Co Ltd RL5c476 II (rev ba)
```

Единственное устройство PCMCIA в примере 19.26 — бридж.

Основная утилита для управления PCMCIA-картами — `pccardctl`.

ЗАДАНИЕ

- Проверьте наличие PCMCIA-бриджа в системе.

Сборка и установка ядра Linux

Самостоятельная сборка действительно рабочего ядра Linux — длительный и кропотливый процесс, требующий высокого профессионализма. Разработчики дистрибутива добиваются наилучшей работы ядра, тщательно интегрируя его с программным обеспечением, входящим в дистрибутив. Нет никакой гарантии, что самостоятельно собранное ядро, которое на первый взгляд кажется рабочим, действительно будет обеспечивать стабильную работу системы. Собирать ядро необходимо лишь при наличии действительно обоснован-

ной необходимости и только после тщательного изучения документации от производителя используемого дистрибутива.

Чаще всего необходимость пересборки ядра в системах, не связанных с разработкой новых ядер Linux, возникает вследствие следующих причин:

- ☐ в коде ядра, используемого в системе, обнаруживается серьезная уязвимость;
- ☐ необходимость обновления кода ядра, связанная с используемым ПО;
- ☐ отсутствие или несоответствие драйвера для какого-либо устройства;
- ☐ отсутствие кода в ядре или модулях ядра для поддержки какой-либо функции;
- ☐ необходимость расширения функциональности ядра;
- ☐ необходимость оптимизации работы ядра.

Для пересборки ядра требуется:

- ☐ наличие компилятора, пригодного для пересборки ядра;
- ☐ наличие GNU-пакета `make`;
- ☐ наличие исходного кода ядра и, возможно, пакетов обновления (patches);
- ☐ при пересборке ядра из дистрибутива необходимо иметь пакеты с исходным кодом ядра и заголовочными файлами;
- ☐ порядка 2 Гбайт дискового пространства;
- ☐ достаточные вычислительные ресурсы;
- ☐ время на конфигурирование и сборку ядра.

До появления ядер 2.6 использовалась специальная система именования ядер, позволяющая отличить стабильные ядра от экспериментальных. У стабильных ядер второе число в версии являлся четным числом, а у экспериментальных ядер — нечетным. Например, ядро 2.4.22 из стабильной ветви, а 2.5.59 из экспериментальной.

Начиная с ядра 2.6.11, разработчики Linux перешли на новую систему нумерации ядер, составленную из четырех цифр. В ней последняя цифра меняется при незначительных изменениях кода в рамках основной версии ядра, имя которой составлено из трех цифр. Изменения четвертой цифры в номере отражают, чаще всего, стабилизирующие исправления в коде. Изменение третьей цифры в номере ядра отражает появление новых возможностей и драйверов в коде.

Получить архив с исходным кодом ядра можно с сайтов **www.kernel.org**, **ftp.kernel.org** или с зеркал (пример 18.27).

Пример 19.27. Получение и распаковка архива

```
$ wget ftp://ftp.kernel.org/pub/linux/kernel/v2.6/linux-2.6.32.tar.bz2
$ wget ftp://ftp.kernel.org/pub/linux/kernel/v2.6/patch-2.6.32.3.bz2
$ tar xjf linux-2.6.32.tar.bz2
```

После этого можно перейти к обновлению распакованного кода ядра с помощью полученного патча (пример 19.28).

Пример 19.28. Обновление ядра 2.6.32 до 2.6.32.3

```
$ cd linux-2.6.32

$ bzcat ../patch-2.6.32.3.bz2 | patch -p1
patching file ...

$ head -4 Makefile
VERSION = 2
PATCHLEVEL = 6
SUBLEVEL = 32
EXTRAVERSION = .3
```

В примере на ядро 2.6.32 наложен патч 2.6.32.3, в результате чего версия ядра обновилась до 2.6.32.3, что видно в Makefile.

Важно соблюдать правила обновления кода ядра с помощью патчей:

- ❑ патчи, обновляющие ядро в четвертой цифре версии, накладываются на первую версию ядра этой ветки, т. е., например, патч 2.6.32.3 накладывается на ядро 2.6.32, а не на 2.6.32.2;
- ❑ патчи, обновляющие ядро в третьей цифре ядра, обязательно накладываются кумулятивно, т. е. 2.6.30 обновляется до 2.6.31 и лишь затем до 2.6.32.

Перед сборкой ядра настоятельно рекомендуется прочесть файл README. Для сборки ядра 2.6 следует пройти следующие стадии.

- ❑ `make mrproper` — предварительная очистка и удаление ненужных файлов.

ПРЕДУПРЕЖДЕНИЕ

Выполнение этой команды среди прочего приводит к удалению файла конфигурации ядра `.config`.

Эту команду следует выполнять после неудачной сборки ядра. Она должна быть выполнена после распаковки кода ядра и его обновления.

- ❑ Конфигурирование ядра заключается в создании текстового файла `.config` с настройками для сборки. Для конфигурирования ядра следует выполнить одну из этих команд:
 - `make oldconfig` — построение конфигурационного файла по умолчанию или на основе предыдущих настроек;
 - `make silentoldconfig` — то же, что и предыдущее, но без выдачи запросов;
 - `make menuconfig` — программа конфигурации ядра с интерфейсом меню;
 - `make xconfig` — программа конфигурации ядра с графическим интерфейсом;
 - `make gconfig` — то же самое, но с интерфейсом Gtk;
 - имеются другие варианты — см. README.
- ❑ `make` — команда построения ядра и модулей.
- ❑ `su` — для перехода в сеанс суперпользователя *без смены текущего каталога*.
- ❑ `make modules_install install` — команда установки модулей ядра и самого ядра. После нее модули ядра будут записаны в каталог `/lib/modules`, а само ядро будет установлено в `/boot`. Конфигурация загрузчика также обновляется.
- ❑ `make clean` — эта команда удаляет оставшиеся от предыдущих шагов или процедур сборки объектные файлы, файлы ядра, модулей и прочего. Эта команда производит чистку только в подкаталогах каталога с исходным кодом ядра и не предназначена для удаления файлов в каталогах `/boot` и `/lib/modules`.

После успешной сборки и установки ядра следует проверить настройки загрузчика, т. к. они обновляются автоматически. Далее можно перезагрузиться и испытать новое ядро.

Если необходимо скомпилировать и установить драйвер, не входящий в код ядра, а предоставленный производителем или же написанный кем-либо еще, в общем случае необходимо выполнить команды, показанные в примере 19.29.

Пример 19.29. Сборка модуля ядра от стороннего поставщика

```
# make -C /lib/modules/`uname -r`/build M=`pwd` modules
# make -C /lib/modules/`uname -r`/build M=`pwd` clean
# make -C /lib/modules/`uname -r`/build M=`pwd` modules_install
```

Возможно, программное обеспечение, не входящее в поставляемый код ядра, имеет другую последовательность сборки и установки. Необходимо тщательно изучить документацию на устанавливаемый продукт.

ЗАДАНИЯ

- Скачайте свежее ядро с сайта **kernel.org**.
- Сконфигурируйте ядро Linux, отметив нужные для данной системы опции и драйверы.
- Соберите первый вариант собственного ядра Linux.
- Проверьте, установлены ли в каталог `/lib/modules` модули для только что собранного ядра.
- Проверьте настройки загрузчика и испытайте новое ядро.



ЧАСТЬ V

Сети

Глава 20



Сетевые средства GNU/Linux

В этой главе приводятся основные сведения о протоколах семейства TCP/IP. Здесь вы узнаете, как настраивать сетевые интерфейсы и маршрутизацию. Также здесь рассмотрены подходы к устранению проблем с сетью.

TCP/IP

Стандартной отправной точкой для изучения стека TCP/IP является общепризнанная модель OSI (Open System Interconnection), определившая семь уровней.

- ☐ *Прикладной уровень* (application layer). На этом уровне осуществляется взаимодействие пользователя с программным обеспечением.
- ☐ *Уровень представления* (presentation layer). Здесь осуществляются преобразования данных, необходимые для предоставления информации, передаваемой программами нижележащих уровней, в требуемом для прикладного уровня виде.
- ☐ *Сеансовый уровень* (session layer). На этом уровне работают программы, обеспечивающие аутентификацию, вход и выход из сеанса.
- ☐ *Транспортный уровень* (transport layer). Программы этого уровня обеспечивают прием/передачу данных с заданной надежностью.
- ☐ *Сетевой уровень* (network layer). Этот уровень определяет адреса сетей и узлов в сети, а также определяет передачи данных между сетями.
- ☐ *Канальный уровень* (data link). На этом уровне обеспечивается передача информации как форматированного потока битов.
- ☐ *Физический уровень* (physical layer) — самый нижний уровень сетевого взаимодействия, обеспечивающий передачу потока битов посредством

сетевого аппаратного обеспечения и физического носителя, связывающего вычислительные системы.

Если отобразить стек протоколов TCP/IP на модель OSI, то окажется, что в нем реализованы лишь четыре уровня сетевого взаимодействия.

- ❑ *Прикладной.* Этот уровень в TCP/IP представлен прикладными протоколами, например: HTTP, FTP, SMTP, POP3 и т. д. С помощью этих протоколов обеспечивается функционирование прикладного программного обеспечения.
- ❑ *Транспортный.* В TCP/IP имеются два транспортных протокола: надежный и ориентированный на соединение протокол TCP (Transmission Control Protocol) и более быстрый, но не надежный, протокол UDP (User Datagram Protocol).
- ❑ *Сетевой.* Данный уровень представлен протоколом IP (Internet Protocol). Этот протокол обеспечивает передачу пакетов между узлами и сетями.
- ❑ *Канальный уровень.* Он представлен в TCP/IP протоколами адресации физических устройств, такими, как ARP (Address Resolution Protocol).

Достоинством вертикально структурированных стеков протоколов является то, что при реализации программ для конкретного уровня не требуется учитывать детали реализации программ, принадлежащих другим уровням.

Данные, сформированные программой прикладного уровня, разбиваются на пакеты, которые упаковываются в пакеты нижележащего уровня (инкапсуляция). При приеме этих упакованных пакетов происходит обратный процесс: принятые пакеты распаковываются и извлекаются из более пакетов нижележащего уровня (декапсуляция).

Существуют документы, описывающие функционирование стека протоколов TCP/IP, Интернета и стандартных сетевых служб. Эти документы называются RFC (Request For Comments). Список RFC размещен на сайте IETF (Internet Engineering Task Force) www.ietf.org, www.rfc.net или www.rfc-editor.org.

Список разнообразных протоколов находится в файле `/etc/protocols` (пример 20.1).

Пример 20.1. Фрагмент файла `/etc/protocols`

```
$ sed -n '10,20p' /etc/protocols
ip      0      IP      # internet protocol, pseudo protocol number
icmp    1      ICMP     # internet control message protocol
...
```

ЗАДАНИЯ

- К какому уровню стека TCP/IP следует отнести протокол IMAP, позволяющий управлять удаленным почтовым ящиком и читать в нем почту?
- Какой транспортный протокол следует предпочесть для передачи TV - трансляции посредством Интернета (например, на стадии проектирования специализированной программы для TV-трансляции)?
- К какому уровню модели OSI можно было бы отнести протокол ICM P (Internet Control Message Protocol)? Он, в частности, позволяет проверить работоспособность и правильность настроек сетевого оборудования.

Адресация IPv4

Назначение протокола IP состоит в передаче пакетов между узлами сети, возможно, находящимися в различных сетях. Поэтому в IP-адресах закодированы адреса сетей и адреса узлов в этих сетях. Задача оборудования, обеспечивающего передачу пакетов между разными сетями (маршрутизацию), состоит в определении адреса сети, в которую направляется пакет, и определения пути в эту сеть.

Так как узел-источник IP-пакета ожидает, вероятно, получить некоторый ответный результат от узла назначения, то в заголовке IP-пакета присутствует как адрес назначения пакета, так и адрес источника.

В настоящий момент различают две спецификации этого протокола: IPv4, в котором для IP-адресов используется 4 байта (см. RFC 791), и более современная спецификация IPv6, в котором используются 16-байтные адреса (см. RFC 2373).

Адреса IPv4 могут быть записаны как с помощью шестнадцатеричной нотации, так и с помощью так называемой записи dotted-decimal. Последний вариант используется шире. Пример IPv4-адреса, записанного в формате dotted-decimal: 192.168.111.253. В этом варианте IP-адрес записывается четырьмя десятичными числами, разделенными точками. Так как каждое десятичное число кодирует один байт, то каждое это число принадлежит диапазону от 0 до 255.

Если все биты, кодирующие номер узла в сети, установлены в 0, то такой адрес является адресом всей этой сети. Напротив, если все биты установлены в 1, то этот адрес является широковещательным адресом для данной сети (broadcast address).

В соответствии с исходной спецификацией адреса IPv4 разделены на пять классов в зависимости от того, сколько битов адреса идентифицируют сеть.

- ❑ В сетях класса A адрес сети кодируется с помощью старшего байта, три младших байта используются для кодирования адреса узла в данной сети.

Причем первый бит старшего байта должен быть установлен в 0, следовательно, адрес сети класса А может находиться в диапазоне значений старшего байта от 0 до 127. То есть всего имеется 128 сетей класса А. При этом оставшиеся 24 бита позволяют закодировать адреса $2^{24} - 2$ узла сети, т. е. 16 777 214 узлов. Пример IPv4-адреса класса А: 81.190.1.250.

- В сетях класса В адрес сети кодируется в двух старших байтах IP-адреса. В первых двух битах старшего байта должно находиться двоичное значение 10. Поэтому первый байт для адресов класса В может принимать десятичные значения от 128 до 191. Имеется $2^{14} = 16\,384$ сети класса В, в каждой из которых может быть $2^{16} - 2 = 65\,534$ узлов. Пример IPv4-адреса класса В: 180.127.10.71.
- В сетях класса С адрес сети кодируется в трех старших байтах IP-адреса. В первых трех битах старшего байта должно находиться двоичное значение 110. Первый байт адресов класса С принадлежит диапазону от 192 до 223. Существует $2^{21} = 2\,097\,152$ сети класса С, а в каждой из них может быть $2^8 - 2 = 254$ узла. Пример IPv4-адреса класса С: 200.217.110.1.
- Сети класса D имеют особое назначение — они предназначены для обеспечения группового вещания (multicast). Суть его заключается в том, что в отличие от обычного режима передачи IP-пакетов, в этом режиме посланный IP-пакет может приниматься несколькими узлами, входящими в группу multicast. Для адресов этого класса в первых четырех битах адреса указывают значение 1110, поэтому возможны десятичные значения первого байта от 224 до 239.
- Имеется также сеть экспериментального назначения Е.

Для сетей А, В, С и D определено понятие *маски сети*. Маска сети, как и адрес IPv4, представлена четырьмя байтами. Биты этих байтов, установленные в 1, определяют часть IP-адреса, идентифицирующую сеть. Биты, установленные в 0, находятся в позициях, соответствующих номеру узла в этой сети. Для сетей А, В, С и D имеются следующие собственные маски сетей (native mask):

- класс А — маска 255.0.0.0;
- класс В — маска 255.255.0.0;
- класс С — маска 255.255.255.0;
- класс D — маска 255.255.255.255.

Маска сети предоставляет простую возможность определять, к какой сети (адресу сети) принадлежит заданный IP-адрес с помощью логической операции И. Например, для адреса 192.168.111.253 маска сети будет 255.255.255.0 (сеть класса С), а операция определения номера сети приведена в табл. 20.1.

Таблица 20.1. IPv4-адрес и маска сети

	В десятичном виде	В двоичном виде
IP-адрес	192.168.111.253	1100 0000 1010 1000 0110 1111 1111 1101
Маска	255.255.255.0	1111 1111 1111 1111 1111 1111 0000 0000
Адрес сети	192.168.111.0	1100 0000 1010 1000 0110 1111 0000 0000

Маршрутизаторы являются устройствами, передающими IP-пакеты между различными сетями. Решение о передаче IP-пакета в ту или иную сеть принимается маршрутизатором на основе анализа адреса сети назначения IP-пакета.

В памяти маршрутизатора имеется таблица (таблица маршрутизации), в которой адресам сетей сопоставлены IP-адреса других маршрутизаторов, позволяющих передать пакет непосредственно в требуемую сеть или же передать другим маршрутизаторам, находящимся "ближе" к сети назначения.

В одну и ту же сеть могут вести несколько маршрутов. Обычно среди них имеются более "короткие" и более "протяженные".

Так как сложно составить таблицу маршрутизации, содержащую пути ко всем сетям, то используется маршрутизатор по умолчанию (default gateway). В качестве такового назначают маршрутизатор, позволяющий передавать пакеты в большее число сетей (в Интернете). Для обозначения адреса назначения "по умолчанию" (default) используется IP-адрес 0.0.0.0. Этот адрес называют в качестве номера сети, в которую способен передавать пакеты маршрутизатор "по умолчанию" (default gateway).

Адрес 127.0.0.1, принадлежащий сети А 127.0.0.0, зарезервирован для обращения к так называемому закольцовываемому сетевому интерфейсу (loop-back interface). Этому интерфейсу не соответствует никакой реальный сетевой интерфейс, он реализован программно и позволяет обеспечивать работоспособность сетевой подсистемы на локальном компьютере, не подключенном к сети.

Имеются также зарезервированные блоки адресов для частных сетей, описанные в RFC 1918. Такие адреса можно использовать свободно, не регистрируя их.

Имеются следующие блоки частных адресов:

- ☐ класс А: 10.0.0.0—10.255.255.255, одна сеть;
- ☐ класс В: 172.16.0.0—172.31.255.255, шестнадцать сетей;
- ☐ класс С: 192.168.0.0—192.168.255.255, двести пятьдесят шесть сетей.

ЗАДАНИЯ

- К какому классу принадлежит IP-адрес 63.127.33.254?
- Какой широковещательный адрес у этой сети?
- Какая маска у этой сети?
- Запишите указанный выше адрес в шестнадцатеричном формате.

Адресация IPv6

Протокол IPv4 не обеспечивает достаточного адресного пространства. Поэтому был разработан новый протокол IPv6. Его особенности:

- ☐ расширенное адресное пространство. В IPv6 вместо 32-разрядных адресов IPv4 используются 128-разрядные адреса;
- ☐ в адресах IPv6 нет понятия класса сети. Неудачный выбор схемы классовой адресации в IPv4 был основной причиной дефицита адресов в Интернете;
- ☐ упрощенный формат заголовка IP-пакета, позволяющий более эффективно обрабатывать IPv6-пакеты на маршрутизаторах;
- ☐ заголовки расширения, позволяющие после основного IPv6-заголовка пакета включать заголовки с дополнительной информацией в стандартном виде;
- ☐ поддержка качества обслуживания QOS (Quality Of Service);
- ☐ встроенная поддержка IPSec, позволяющая создавать виртуальные частные сети;
- ☐ поддержка IPMobile, удобная для перемещающихся пользователей.

В IPv6 имеются три типа адресов:

- ☐ Unicast — аналогичны обычным адресам IPv4, назначаемым интерфейсам;
- ☐ Multicast — адреса группового вещания;
- ☐ Anycast — адреса для достижения любого из устройств.

Широковещательных адресов в IPv6 нет, поскольку в IPv4 в связи с использованием широковещания возникает множество проблем. В том числе и с безопасностью.

В IPv6 можно использовать транспортные протоколы TCP и UDP. Имеется также собственный транспортный протокол, ориентированный на надежную работу с потоками информации — SCTP.

Формат представления адресов IPv6 описан в RFC 2373, в котором установлено, что IPv6-адреса записывают в виде последовательности из шестнадцати пар шестнадцатеричных чисел. Каждые две пары шестнадцатеричных чисел разделяют двоеточием (два байта). Таким образом, в IPv6-адресе должно быть восемь четырехразрядных шестнадцатеричных чисел, разделенных двоеточиями (пример 20.2).

Пример 20.2. IPv6-адрес

```
fe80:0000:0000:0000:0250:8bff:fe5f:7ceb
```

Так как в большинстве IPv6-адресов имеются длинные последовательности нулей, то записывать их в полном виде не удобно. Вместо этого предлагается пользоваться сокращенной нотацией. В ней непрерывные последовательности нулей могут быть сокращены. При этом в IPv6-адресах в месте, где были сокращены нули, записывают два двоеточия :: (пример 20.3). Сокращение нулей можно проводить лишь один раз.

Пример 20.3. IPv6-адрес в сокращенной нотации

```
fe80::250:8bff:fe5f:7ceb
```

В примере 20.3 показано, что нули, стоящие после fe80, сокращены.

Адрес loopback-интерфейса в IPv6 в сокращенном виде записывают: ::1, т. к. в нем 127 нулей и последняя единица.

Если IPv6-адрес начинается с fe80, значит, этот адрес получен с помощью процедуры автоматического конфигурирования из MAC-адреса сетевого интерфейса. То есть такие адреса не надо назначать вручную, они определяются автоматически.

Допустим, у Ethernet-интерфейса имеется MAC-адрес 00:50:8B:5F:7C:EB. Тогда автоматически сконфигурированный IPv6-адрес будет fe80::250:8bff:fe5f:7ceb. IPv6-адрес формируется автоматически добавлением после третьего байта MAC-адреса (с правой стороны) двух байтов fffe, что заметно в приведенных адресах.

Задания

- Выполните команду `/sbin/ifconfig`. Имеется ли автоматически сконфигурированный адрес IPv6?
- Проверьте соответствие MAC-адреса и IPv6-адреса (MAC-адрес — `hwaddr`).

Настройка сетевого интерфейса Ethernet

Команда `ifconfig` позволяет настроить сетевой интерфейс (пример 20.4). Имена сетевых интерфейсов в GNU/Linux зависят от их типа. Для интерфейсов Ethernet используются имена `eth0` — для первого интерфейса, `eth1` — для второго и т. д.

Пример 20.4. Настройка IPv4-адреса

```
# ifconfig eth0 212.193.90.5
```

В простейшем случае достаточно указать имя сетевого интерфейса и его IP-адрес.

Команда `ifconfig` позволяет узнать настройки сетевого интерфейса (пример 20.5).

Пример 20.5. Получение настроек сетевого интерфейса

```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:03:0D:10:DD:C7
          inet addr:212.193.90.5  Bcast:212.193.90.255  Mask:255.255.255.0
          inet6 addr: fe80::203:dff:fe10:ddc7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1234123 errors:0 dropped:0 overruns:0 frame:0
          TX packets:619932 errors:0 dropped:0 overruns:0 carrier:21
          collisions:34647 txqueuelen:100
          RX bytes:1796372646 (1713.1 Mb)  TX bytes:41619121 (39.6 Mb)
          Interrupt:20 Base address:0xcf00
```

Каждый адаптер Ethernet имеет уникальный аппаратный адрес (MAC-адрес), определенный при его производстве. В выводе команды `ifconfig` он отображается в поле `HWaddr`. В этом примере — `HWaddr 00:03:0D:10:DD:C7`.

Для получения информации о приеме/передаче пакетов через все сетевые интерфейсы в системе применяют команду `netstat -i` (пример 20.6).

Пример 20.6. Получение состояния интерфейсов

```
# netstat -i
Kernel Interface table
```

Iface	MTU	Met	RX-OK	RX-ERR	RX-DRP	RX-OVR	TX-OK	TX-ERR	TX-DRP	TX-OVR	Flg
eth0	1500	0	1439206	0	0	0	722726	0	0	0	BMRU
lo	16436	0	6	0	0	0	6	0	0	0	LRU

В полях `RX-OK` и `TX-OK` указывается, соответственно, информация о принятых и отправленных пакетах. А поля `RX-ERR` и `TX-ERR` отображают статистику ошибок.

Проверить работоспособность сетевого интерфейса можно с помощью команды `ping`, которая посылает ICMP-пакеты и ждет ответных пакетов (пример 20.7).

Пример 20.7. Команда `ping`

```
# ping -c2 212.193.90.5
PING 212.193.90.5 (212.193.90.5) 56(84) bytes of data.
64 bytes from 212.193.90.5: icmp_seq=1 ttl=64 time=0.028 ms
64 bytes from 212.193.90.5: icmp_seq=2 ttl=64 time=0.020 ms
--- 212.193.90.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.020/0.022/0.028/0.006 ms
```

Опция `-c2` указывает команде `ping`, что надо послать два пакета ICMP. После проверки прохождения пакетов команда `ping` выдает результирующую статистику по количеству отправленных, принятых и потерянных пакетов.

ЗАДАНИЯ

- Получите статистику по приему/передаче пакетов.
- Как командой `ifconfig` можно отключить сетевой интерфейс?
- Остановите сетевой интерфейс и попробуйте выполнить команду `ping`, направляя пакеты на остановленный интерфейс. Что при этом происходит?
- Какая опция `ping` позволяет посылать ICMP-пакеты потоком без задержки?

Настройка маршрутизатора по умолчанию

Если компьютер должен обмениваться IP-пакетами лишь в пределах локальной сети или соединения "точка — точка", то маршрутизацию настраивать не требуется. Сетевой интерфейс, которому назначен IP-адрес, принадлежит соответствующей сети.

Маршрутизатор — это компьютер или специализированное устройство, позволяющее передавать пакеты из одной сети в другую. Для того чтобы компьютер мог быть маршрутизатором, он должен иметь как минимум два сете-

вых интерфейса (не обязательно физических). Кроме этого, ядро должно обеспечивать передачу IP-пакетов между интерфейсами (forwarding).

Таблица маршрутизации находится в ядре, и в ней указаны маршруты направления IP-пакетов в различные сети. Маршрут связывает адрес сети назначения, адрес маршрутизатора и интерфейс, через который можно достичь этой сети.

Опция `-r` команды `netstat` предназначена для получения таблицы маршрутизации, а опция `-n` позволяет выводить ее содержимое в числовом виде (пример 20.8).

Пример 20.8. Таблица маршрутизации

```
# ifconfig eth0 192.168.1.3
# netstat -rn
Kernel IP routing table
Destination      Gateway          Genmask          Flags   MSS Window  irtt Iface
192.168.1.0      0.0.0.0         255.255.255.0    U        0  0          0 eth0
127.0.0.0        127.0.0.1       255.0.0.0        UG        0  0          0 lo
```

Первая команда в этом примере конфигурирует интерфейс `eth0`, назначая ему адрес `192.168.1.3`. Вторая выводит таблицу маршрутизации. Несложно заметить, что в таблице маршрутизации появилась сеть `192.168.1.0`, т. к. установленный для интерфейса `eth0` IP-адрес принадлежит этой сети. Поскольку маршрутизатор не требуется для передачи пакетов внутри сети, то в поле `Gateway` указано `0.0.0.0`.

Сконфигурированный интерфейс способен обмениваться IP-пакетами с другими сетевыми интерфейсами узлов своей сети.

Протокол ARP предназначен для преобразования IP-адресов в MAC-адреса сетевых устройств. Результаты этого преобразования кэшируются. Кэш ARP можно просмотреть с помощью команды `arp -an` (пример 20.9).

Пример 20.9. Просмотр кэша ARP

```
$ /sbin/arp -an
? (192.168.1.1) at 00:0C:6E:1F:E8:FA [ether] on eth0
```

В кэше ARP здесь находится единственный IP-адрес `192.168.1.1` сетевого интерфейса, имеющего MAC-адрес `00:0C:6E:1F:E8:FA`.

Предположим теперь, что в сети `192.168.1.0` имеется маршрутизатор с IP-адресом интерфейса в этой сети `192.168.1.1`. Допустим, что этот маршрутиза-

тор позволяет пакетам из сети 192.168.1.0 следовать в любые другие сети. В таком случае он является маршрутизатором по умолчанию. Для того чтобы поместить информацию о нем в таблицу маршрутизации, необходимо выполнить команду в примере 20.10.

Пример 20.10. Маршрутизатор по умолчанию

```
# route add default gw 192.168.1.1
# netstat -rn
Kernel IP routing table
Destination    Gateway        Genmask         Flags   MSS Window  irtt Iface
192.168.1.0    0.0.0.0        255.255.255.0   U        0 0          0 eth0
127.0.0.0      127.0.0.1      255.0.0.0       UG        0 0          0 lo
0.0.0.0        192.168.1.1    0.0.0.0         UG        0 0          0 eth0
```

В поле `Destination` находится значение 0.0.0.0, или маршрут по умолчанию (default может быть записан 0.0.0.0). В поле `Gateway` находится IP-адрес интерфейса маршрутизатора, принадлежащего сети 192.168.1.0.

В состав большинства дистрибутивов GNU/Linux входит утилита `traceroute`, позволяющая проследживать процесс прохождения IP-пакетов через маршрутизаторы по мере следования его к узлу назначения (пример 20.11).

Пример 20.11. Фрагмент листинга, выводимого утилитой `traceroute`

```
# traceroute 66.35.212.174
traceroute to 66.35.212.174 (66.35.212.174), 30 hops max, 40 byte packets
...
19 66.35.212.174 (66.35.212.174) 263.627 ms 263.687 ms 263.592 ms
```

Вызванная без аргументов, эта команда осуществляет пробы всех маршрутизаторов, которые встречаются на пути прохождения IP-пакетов к узлу назначения. Если какой-либо узел не посылает ответные пакеты, например, в силу их фильтрации, то вместо адресов и имен узлов сети в списке будут выводиться звездочки.

Задания

- Установите заведомо неверный адрес маршрутизатора по умолчанию.
- Попробуйте выполнить команду `ping`, указав в качестве аргумента правильный адрес маршрутизатора. Попробуйте также выполнить предыдущую

команду с любым существующим IP-адресом узла сети, находящегося вне вашей сети.

- Измените адрес маршрутизатора по умолчанию на правильный и снова выполните команду `ping` с адресом любого реального узла вне вашей сети.
- Выполните команду `traceroute` с этим же адресом.

Настройка разрешения имен

Сетевое имя машины, работающей под управлением GNU/Linux, может быть получено и установлено с помощью команды `hostname` (пример 20.12).

Пример 20.12. Установка имени хоста

```
$ hostname  
newnote.class.edu
```

Эта команда выполняется на стадии инициализации системы, а строка имени машины находится в одном из файлов в `/etc`. Для RH-подобных систем этот файл — `/etc/sysconfig/network`, в SUSE — `/etc/HOSTNAME`.

Различают *короткие* и *полные имена узлов* (FQDN, Fully Qualified Domain Name). Короткие имена состоят из строк, в которых отсутствуют символы точек. В данном примере короткое имя узла — `newnote`. Полное имя образуется из короткого с помощью добавления доменной части — имени DNS домена, которому принадлежит данный компьютер. В этом примере узел `newnote` принадлежит домену `class.edu`. Поэтому `newnote.class.edu` — это полное имя узла.

Имена узлов используются для более понятного указания их в сети. Для человека гораздо более удобно использовать имена компьютеров вместо их IP-адресов. Таким образом, требуется специальная служба для разрешения имен узлов, т. е. преобразования имени узла в его IP-адрес. Часто требуется также осуществить обратное преобразование: из IP-адреса в имя узла. Преобразование осуществляется с помощью специальной библиотеки `resolver`.

Существуют следующие способы преобразования:

- ❑ с помощью файла `/etc/hosts`. Этот способ подходит для небольших сетей;
- ❑ с помощью обращения к службе DNS (Domain Name Service);
- ❑ с помощью обращения к службам NIS, NIS+, LDAP и прочим.

Создание статичного списка соответствий IP-адресов именам узлов сети — наиболее простой способ получения разрешения имен. Он годится для

небольших сетей, т. к. на каждом узле сети должны быть копии файла `/etc/hosts`. В каждой строке этого файла указывают IP-адрес и имена узла (пример 20.13).

Пример 20.13. Файл `/etc/hosts`

```
$ cat /etc/hosts
127.0.0.1          localhost
192.168.1.1        mycomp.mynet.net    mycomp
```

Если имена узлов и их IP-адреса указаны верно, то в этом случае `resolver` обеспечит преобразование имен узлов в их адреса (пример 20.14).

Пример 20.14. Автоматическое преобразование имен в IPv4-адреса

```
$ ping -c2 mycomp
PING mycomp (192.168.1.1) 56(84) bytes of data.
64 bytes from mycomp (192.168.1.1): icmp_seq=1 ttl=64 time=0.245 ms
64 bytes from mycomp (192.168.1.1): icmp_seq=2 ttl=64 time=0.091 ms
--- 192.168.1.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.088/0.141/0.245/0.073 ms
```

Настройки `resolver` хранятся в файле `/etc/resolver.conf` (пример 20.15).

Пример 20.15. Файл `/etc/resolv.conf`

```
$ cat /etc/resolv.conf
nameserver 192.168.1.254
search mydom.com, mydom.net
```

Директива `nameserver` задает IP-адрес DNS-сервера, обслуживающего данный компьютер. С помощью последней директивы — `search` — задают список доменов, имена которых будут автоматически подставляться к именам узлов при их поиске.

Таким образом, в большинстве случаев для преобразования имени узла в его IP-адрес требуется либо указать его адрес и имя в `/etc/hosts`, либо настроить доступ к DNS-серверу, указав его IP-адрес в файле `/etc/resolv.conf`.

Имеются еще два конфигурационных файла, настройки которых влияют на поведение библиотеки `resolver` — `/etc/host.conf` и `/etc/nsswitch.conf`.

Для `resolver` они важны с точки зрения порядка разрешения имен: будет ли сначала произведено обращение к файлу `/etc/hosts` или же к DNS (пример 20.16).

Пример 20.16. Файл `/etc/host.conf`

```
order host, bind
multi on
```

Здесь задан следующий порядок работы `resolver`: при разрешении имени узла сначала осуществляется просмотр записей в файле `/etc/hosts`, а затем — обращение к DNS. Директива `multi on` заставляет `resolver` выводить все адреса узлов, имеющих несколько IP-адресов. Файл `/etc/nsswitch.conf` (Name Service Switch) предназначен для сообщения библиотеке `resolver` и другим системным вызовам, где искать требуемую информацию. Файл `/etc/nsswitch.conf` содержит централизованную конфигурационную информацию о доменах и связанных с ними базах данных:

- ☐ `aliases` — почтовые псевдонимы почтового транспортного агента;
- ☐ `ethers` — источник информации о MAC-адресах;
- ☐ `group` — источник информации о группах пользователей;
- ☐ `hosts` — порядок разрешения имен узлов;
- ☐ `netgroup` — список пользователей каждого хоста для службы NIS;
- ☐ `network` — источник информации об именах и адресах сетей;
- ☐ `passwd` — список поиска источников паролей пользователей;
- ☐ `protocols` — список поиска протоколов, используемых в сети;
- ☐ `publickey` — местоположение ключей шифрования;
- ☐ `rpc` — список поиска имен и номеров вызываемых удаленных процедур;
- ☐ `services` — список поиска сетевых служб;
- ☐ `shadow` — список поиска зашифрованных паролей.

При использовании DNS и `/etc/hosts` в файле `/etc/nsswitch.conf` должны содержаться строки, представленные в примере 20.17.

Пример 20.17. Настройка разрешения имен

```
hosts:          files dns
networks:       files dns
```

Первая настройка задает порядок разрешения имен узлов: сначала будет произведено обращение к `/etc/hosts`, а далее — к DNS. Вторая настройка задает аналогичный порядок для сетей: сначала к `/etc/networks`, а затем к DNS.

ЗАДАНИЯ

- Проверьте наличие имени вашего компьютера в файле `/etc/hosts` и выполните команду `ping`, указав ей в качестве аргумента имя вашего компьютера.
- Проверьте правильность указания DNS-сервера в файле `/etc/resolv.conf`. Попробуйте выполнить команду `ping` для одного из известных вам имен узлов в сети. Разрешается ли имя данного узла?

Поиск и устранение проблем с сетью

Время от времени системному администратору приходится сталкиваться с достаточно сложными проблемами в сетевом взаимодействии программного обеспечения. Однако часто проблемы при работе сети бывают вызваны несколькими причинами:

- ☐ неисправностью в работе сетевых кабелей или активного сетевого оборудования;
- ☐ несоответствием или отсутствием драйвера сетевого адаптера;
- ☐ неверными настройками IP-адресов узлов сети;
- ☐ "засорением" кэша ARP;
- ☐ неверным указанием IP-адреса маршрутизатора по умолчанию;
- ☐ ошибочной установкой DNS-сервера или неверными записями в `/etc/hosts`;
- ☐ блокирующими настройками фильтра IP-пакетов.

Хорошим подходом к поиску аппаратных неисправностей является проверка работоспособности сетевой инфраструктуры неработающего узла сети с помощью заведомо исправных сетевых кабелей и активного оборудования.

Если сетевой адаптер стандарта PCI исправен, то информация о нем должна присутствовать в выводе команды `/sbin/lspci` (пример 20.18).

Пример 20.18. Проверка наличия сетевой платы

```
$ /sbin/lspci | grep -i ether
01:08.0 Ethernet controller: Intel Corp. 82801BD PRO/100 VE (CNR)
```


Далее, используя команды `ifconfig` и `netstat -rn`, следует убедиться в правильности настройки IP-адреса и адреса маршрутизатора по умолчанию.

Прохождение пакетов командой `ping` нужно производить в следующем порядке:

1. Проверить работоспособность сетевой инфраструктуры в ядре `ping 127.0.0.1`.
2. Проверить работоспособность сетевого адаптера с помощью команды `ping`, которой указан IP-адрес, привязанный к данному адаптеру.
3. Проверить прохождение пакетов в пределах данной сети, послав командой `ping` ICMP-пакеты любому узлу в данной сети или маршрутизатору.
4. Проверить прохождение пакетов к внешнему сетевому адаптеру маршрутизатора.
5. Послать ICMP-пакеты по любому известному IP-адресу сети.

В качестве последней проверки можно протестировать работоспособность библиотеки `resolver`, выполнив `ping` с указанием имени компьютера в качестве аргумента.

Если команда `ping` с аргументом — IP-адресом узла назначения в этой же сети показывает, что ICMP-пакеты не проходят, но настройки адресов верные, то, возможно, "засорен" кэш ARP (Address Resolution Protocol). Проверить содержимое кэша ARP можно с помощью команды `arp -a`.

Задача протокола ARP состоит в сопоставлении IP-адресов сетевых интерфейсов их MAC-адресам, поэтому в информации, выводимой командой `arp -a`, должны содержаться правильные соответствия. В противном случае неверные записи должны быть стерты с помощью команды `arp -d <имя узла>`.

Невозможность приема и передачи ICMP-пакетов (как и других пакетов IP) может быть вызвана их блокированием на передающем и/или принимающем узле. Рассмотрение фильтров IP выходит за рамки данной книги, но проверить отсутствие таких блокировок можно, используя команду `iptables -L`.

Если ICMP-пакеты, посланные командой `ping`, правильно пересылаются узлу назначения только в случае указания его с помощью IP-адреса, но это не происходит при указании имени узла, то `resolver` не работает. При этом следует проверить правильность указания IP-адреса DNS-сервера в файле `/etc/resolv.conf` и правильность соответствий IP-адресов в файле `/etc/hosts`. Имеются специальные команды: `host`, `dig` и `nslookup`, позволяющие тести-

ровать правильность работы DNS. Подробно эти команды будут рассмотрены в *главе 26*, посвященной службе DNS, т. к. для проверки resolver вполне достаточно лишь одной из них — `host`.

При необходимости получения административной информации о каком-либо домене DNS можно использовать команду `whois`.

ЗАДАНИЕ

Закомментируйте символом "решетка" (#) строку, устанавливающую IP-адрес DNS-сервера в файле `/etc/resolv.conf`. Убедитесь с помощью команды `host` в том, что имена узлов не разрешаются в их IP-адреса.

Глава 21



Сервисы сети

Сетевые службы, работающие в GNU/Linux, могут запускаться по-разному. Здесь рассматриваются варианты их запуска. В этой главе вы узнаете о том, как определять статус сетевых служб, как запускать и останавливать их.

Идентификация служб сети

Процесс взаимодействия программ в сети проходит в соответствии с требованиями протокола прикладного уровня (Application Layer). В TCP/IP для идентификации служб, которые могут быть запущены на каком-либо узле сети, используются номера портов в диапазоне от 0 до 65 535 (шестнадцать битов). Номера портов назначаются IANA (Internet Assigned Numbers Authority).

Многие номера портов, меньшие 1024 и некоторые большие, закреплены за определенными службами сети (Well Known Services). Вот список некоторых:

- ❑ 20 — данные, передаваемые сервисом FTP;
- ❑ 21 — управляющая информация для сервиса FTP;
- ❑ 22 — протокол SSH;
- ❑ 25 — транспортный почтовый протокол SMTP;
- ❑ 23 — удаленный доступ с помощью Telnet;
- ❑ 53 — протокол DNS;
- ❑ 80 — протокол HTTP;
- ❑ 110 — доставка почты по протоколу POP3;
- ❑ 119 — служба доставки новостей NNTP;
- ❑ 139 — протокол NetBIOS;
- ❑ 143 — удаленный доступ к почтовым ящикам по протоколу IMAP;

- ❑ 161 — протокол управления сетью SNMP;
- ❑ 389 — протокол доступа к службам каталогов LDAP.

Список имен служб находится в файле `/etc/services` (пример 21.1).

Пример 21.1. Файл `/etc/services`

```
ftp-data      20/tcp
ftp           21/tcp
fsp           21/udp      fspd
ssh           22/tcp              # SSH Remote Login Protocol
ssh           22/udp
telnet        23/tcp
smtp          25/tcp      mail
```

Службы, номера портов которых меньше 1024, называются *привилегированными*. Серверные приложения, открывающие данные порты, могут быть запущены лишь суперпользователем. В противоположность привилегированным портам порты с номерами от 1024 до 65 535 называют *эфемерными*, поскольку большинство из них не закреплено за какими-либо службами сети и используются клиентскими приложениями для связи с серверами.

Для связи между двумя сетевыми приложениями каждое из них должно открыть сокет (socket). Список активных соединений выводит команда `netstat`.

Если необходимо получить список не только установленных соединений, но и перечень портов, открытых для соединения, то можно использовать опцию `-a` команды `netstat`. Ограничить вводимую информацию лишь TCP-или UDP-портами можно, используя опции, соответственно, `-t` или `-u` (пример 21.2).

Пример 21.2. Получение списка открытых TCP-портов

```
$ netstat -ta
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp      0      0 *:ssh                   :*:                     LISTEN
```

Этот пример показывает, что на данном компьютере открыт порт для соединений по протоколу SSH (Secure Shell).

Для открытых портов в поле статуса установлен идентификатор `LISTEN`. Если соединение установлено, то в поле статуса будет `ESTABLISHED` (пример 21.3).

Пример 21.3. Установленные соединения на клиентском компьютере

```
# netstat -ta
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 black:32771             work:ssh                 ESTABLISHED
tcp        0      0 *:ssh                   *:*
```

Обратите внимание, что на клиентском компьютере (в данном случае — `ssh`) открыт сокет, использующий порт 32 771. Через этот сокет производится обмен информацией с сокетом 22/tcp (служба `ssh`) на компьютере, где запущен сервер `sshd`.

Для вывода информации в числовом виде предназначена опция `-n` команды `netstat`.

Открытые, но не используемые порты, представляют собой брешь в системе безопасности, поскольку каждый прослушиваемый порт представляет собой потенциальную цель атаки.

ЗАДАНИЯ

- Для какой службы используется порт 8080 TCP?
- Выполните команду `netstat -a`. В каком порядке выводится информация об открытых портах в системе?
- Имеются ли какие-либо открытые сокеты в подкаталогах `/tmp`? Используйте для определения этого команду `netstat` без аргументов.
- Получите информацию об открытых портах UDP.
- Как получить PID процессов, открывших эти порты (см. опцию `-p` команды `netstat`)?

Запуск сетевых служб

В GNU/Linux используются два пути запуска сетевых сервисов:

- ☐ запуск сетевых служб с помощью инициализационных сценариев в каталоге `/etc/init.d`. Такие службы именуются *самостоятельными* (*stand-alone*);
- ☐ запуск служб с помощью супердемона `inetd` (или `xinetd`), который открывает заданные в файле `/etc/inetd.conf` (`/etc/xinetd.conf`) порты и запускает при поступлении запроса соответствующий демон сетевой службы.

Запуск самостоятельных служб (stand-alone) приводит к более интенсивному расходованию памяти, но такой способ запуска позволяет быстрее обрабатывать большие потоки запросов к службам. Использование служб, запускаемых супердемоном, значительно экономит память системы, но приводит к дополнительным накладным расходам на запуск служб.

ЗАДАНИЯ

- Определите, какие порты TCP и UDP открыты в вашей системе.
- Используя опцию `-p` команды `netstat`, определите PID процессов, прослушивающих эти порты. Какие из этих портов открыты супердемоном?

Использование супердемона *inetd* и фильтра *tcpd*

Супердемон `inetd` открывает порты, перечисленные в файле его конфигурации `/etc/inetd.conf`. В этом же файле указывают серверные программы, управление которым передает супердемон для реального обслуживания соединений. Фрагмент файла конфигурации супердемона `inetd` — `/etc/inetd.conf` приведен в примере 21.4.

Пример 21.4. Файл конфигурации `inetd`

```
#Service Socket Protocol Flags User Path Args
telnet stream tcp nowait root /usr/sbin/tcpd in.telnetd
#shell stream tcp nowait root /usr/sbin/tcpd in.rshd
#login stream tcp nowait root /usr/sbin/tcpd in.rlogind
#exec stream tcp nowait root /usr/sbin/tcpd in.rexecd
```

Каждой службе, запускаемой с помощью супердемона `inetd`, соответствует одна строка в файле `/etc/inetd.conf`. В каждой строке конфигурации имеется семь полей:

- **Service** — имя службы в соответствии с определениями в файле `/etc/services`;
- **Socket** — тип сокета: `stream` для TCP и `dgram` — UDP;
- **Protocol** — тип транспортного протокола может быть `tcp`, `udp` и `raw`;
- **Flags** — определяет поведение только сокетов типа `dgram`. Для них в данном поле могут находиться значения либо `wait`, либо `nowait`. Для всех остальных типов сокетов необходимо использовать только `nowait`. В случае

если dgram-сервер должен освобождать сокет сразу же после завершения обслуживания, то тогда необходимо использовать значение `nowait`. Такой сервер является многопоточным (multi-treated). В противном случае необходимо использовать однопоточный сервер (single-treated), для чего нужно использовать значение `wait`;

- ❑ `User` — определяет UID процесса сервера, обслуживающего данное соединение;
- ❑ `Path` — путь к исполняемому файлу сервера. В Linux запуск серверов всегда производится посредством фильтра `tcpd` (TCP Wrapper);
- ❑ `Args` — командные опции и аргументы, передаваемые серверу `tcpd`.

Так, строка конфигурации, приведенная в примере 21.4, настраивает супердемон `inetd` для запуска посредством фильтра `tcpd` сервера `telnet`, имеющего исполняемый файл `in.telnetd`. При этом соединение будет осуществляться с помощью транспортного протокола TCP, поэтому тип сокета — `stream`.

Обычно в файле конфигурации `/etc/inetd.conf` имеются заранее настроенные закомментированные строки настройки сервисов. Для того чтобы супердемон `inetd` стал прослушивать новый порт, достаточно найти соответствующую ему строку в файле `/etc/inetd.conf`, раскомментировать ее и послать процессу супердемона сигнал `SIGHUP` для того, чтобы он перечитал файл конфигурации (пример 21.5).

Пример 21.5. Передача сигнала `SIGHUP` супердемону

```
# pkill -1 inetd
```

Напротив, если необходимо исключить прослушивание некоторой службы супердемоном `inetd`, следует закомментировать символом "решетка" (`#`) соответствующую строку в `/etc/inetd.conf` и также послать сигнал `SIGHUP` процессу супердемона.

Задания

- Определите, используется ли в вашей системе супердемон `inetd`.
- При использовании в системе `inetd` активизируйте службу `telnet`.

Программа `tcpd`

Программа `tcpd` (TCP Wrapper) представляет собой средство фильтрации IP-пакетов, позволяющее ограничить список IP-адресов, с которых могут поступать запросы на сервисы, предоставляемые супердемоном `inetd`.

Поведение `tcpd` задается с помощью двух файлов:

- ☐ `/etc/hosts.allow` — разрешает соединения;
- ☐ `/etc/hosts.deny` — запрещает соединения.

Порядок работы директив в `/etc/hosts.allow` и `/etc/hosts.deny` приведен далее.

1. Если в файле `/etc/hosts.allow` для заданной пары "узел сети — служба" находится совпадение, то для данного узла доступ к этой службе разрешается.
2. Если в предыдущем файле совпадение не найдено, то осуществляется просмотр файла `/etc/hosts.deny`. При наличии в нем совпадения для заданной пары "узел сети — служба" доступ к данной службе с этого узла запрещается.
3. При отсутствии совпадений в обоих файлах доступ разрешается.

Формат записей в файлах `/etc/hosts.allow` и `/etc/hosts.deny` таков:

имя демона: список доступа

В списке доступа определяются группы адресов, с которых могут поступать запросы на конкретные сервисы:

- ☐ `ALL` — все адреса;
- ☐ `EXCEPT` — исключения из списка;
- ☐ `KNOWN` — хосты, соответствие IP-адресов именам которых можно проверить;
- ☐ `LOCAL` — локальный адрес;
- ☐ `PARANOID` — хосты, имена которых не соответствуют их IP-адресам (для проверки используется обратное преобразование имен в DNS: из IP-адреса в имя хоста);
- ☐ `UNKNOWN` — хосты, имена которых не могут быть разрешены, а также сюда относятся пользователи, не зарегистрированные в целевой системе.

Пример 21.6. Конфигурация `tcpd`

```
cat /etc/hosts.deny
ALL: mail.yahoo.com, .badspammers.org
ALL EXCEPT ftp: compl.smallnet.ru, .rcpmk.ru
telnet: ALL EXCEPT LOCAL
```

В примере 21.6 произведены такие настройки:

- ☐ запрещается доступ ко всем службам для запросов, исходящих с хоста `mail.yahoo.com` и домена `badspammers.org`;

- ❑ разрешен доступ к ftp с хоста `comp1.smallnet.ru` и для домена `rcmpk.ru`;
- ❑ к telnet доступ запрещен отовсюду, кроме локального хоста.

Для проверки правильности конфигурации файлов `/etc/hosts.allow` и `/etc/hosts.deny` используется команда `tcpdchk`. Кроме этого, имеется утилита `tcpdmatch`, которая пытается определить, каким образом `tcpd` будет обрабатывать запросы к `inetd`. Ее удобно использовать при наличии сложных правил фильтрации.

Задания

- Запретите с помощью `/etc/hosts.deny` доступ отовсюду ко всем службам, кроме `telnet`, доступ к которому должен быть разрешен в `/etc/hosts.allow`. Проверьте конфигурацию с помощью `tcpdchk`.
- Сделайте это же исключительно с помощью файла `/etc/hosts.deny`.
- Как запретить доступ к `telnet` с узлов, имена которых не могут быть получены с помощью обратного преобразования IP-адреса в имя хоста посредством DNS?
- Проверьте командой `tcpdmatch` последовательность фильтрации.

Использование супердемона *xinetd*

В GNU/Linux широко используется современная альтернатива супердемону `inetd` — расширенный демон `xinetd` (extended Internet services daemon), обладающий лучшими качествами, чем `inetd` в плане возможностей конфигурирования и безопасности.

Как и `inetd`, супердемон `xinetd` позволяет запускать как многопоточные службы (multi-treated), создавая новый серверный процесс для каждого запроса, так и однопоточные (single-treated), в которых один экземпляр серверного процесса обслуживает все поступающие запросы (обычно только для протокола UDP).

Одним из преимуществ `xinetd` перед `inetd` является возможность запуска простыми пользователями непривилегированных служб. Супердемон `xinetd` не требует наличия имени сервиса в файле `/etc/services`.

Супердемон `xinetd` позволяет управлять доступом к любому сервису на основе:

- ❑ адреса удаленного хоста;
- ❑ времени доступа;
- ❑ имени удаленного хоста;
- ❑ домена, к которому принадлежит удаленный хост.

Супердемон `xinetd` обеспечивает возможность реконфигурации в ходе выполнения. Некоторые характеристики `xinetd` приведены далее в списке.

- ☐ `xinetd` может останавливать процессы служб, удаленных из конфигурации;
- ☐ может останавливать процессы служб, не удовлетворяющих ограничениям;
- ☐ позволяет ограничивать количество экземпляров серверов для каждого сервиса;
- ☐ может ограничить количество экземпляров сервера, запущенных для обслуживания соединений с одного адреса;
- ☐ супердемон `xinetd` позволяет ограничить размер создаваемых журнальных файлов;
- ☐ способен налагать ограничения на исходящие соединения;
- ☐ позволяет останавливать сервисы, расходующие ресурсы свыше ограничения;
- ☐ предоставляет широкие возможности по настройке журналирования;
- ☐ не ограничивает возможное количество передаваемых демону аргументов;
- ☐ можно связывать сервисы с заданными сетевыми интерфейсами;
- ☐ запускаемые `xinetd` службы могут отсутствовать в `/etc/services`;
- ☐ супердемоны `xinetd` и `inetd` могут работать параллельно.

Супердемон `xinetd` настраивается с помощью конфигурационного файла `/etc/xinetd.conf`. Конфигурационный файл `/etc/inetd.conf` может быть преобразован в файл `/etc/xinetd.conf` командой `itox`.

В общем виде формат записей конфигурации `xinetd` таков:

```
service <служба>
{
    <атрибут> <оператор> <значение> <значение> ...
    ...
}
```

Оператор присваивания `<оператор>` может быть одним из:

- ☐ `=` — присвоение значения атрибуту;
- ☐ `--` — удаление атрибута;
- ☐ `+=` — добавление атрибута.

Файл `/etc/xinetd.conf` содержит настройки по умолчанию (пример 21.7).

Пример 21.7. Файл `/etc/xinetd.conf`

```
defaults
{
    log_type = SYSLOG authpriv
    log_on_success = PID HOST
    log_on_failure = HOST
    instances = 25
    per_source = 5
    only_from = 127.0.0.1
}
```

`includedir /etc/xinetd.d`

Команда `includedir` заставляет супердемон прочитать все индивидуальные конфигурационные файлы служб в каталоге `/etc/xinetd.d`.

Настройка `log_type` определяет тип журналирования. Здесь журналирование будет осуществляться с помощью службы `syslog`, причем запись сообщений будет осуществляться в канал `authpriv`. При успешном запуске службы (`log_on_success`) в журнал будут записаны номер процесса запущенного для обслуживания соединения демона и имя хоста, инициировавшего соединение. В случае отказа в запуске (`log_on_failure`) в журнал будет занесено лишь имя хоста.

Настройка `instances` ограничивает максимальное количество экземпляров демонов для каждой конкретной службы, а `per_source` — максимальное количество демонов для обслуживания соединений из одного источника.

Ограничение `only_from = 127.0.0.1` разрешает обращение к службам только с локального хоста.

Любые настройки из секции `defaults` файла `/etc/xinetd.conf` могут быть переопределены в файлах индивидуальной настройки служб в каталоге `/etc/xinetd.d`. Удобно задавать такие имена файлов индивидуальных настроек служб, чтобы они соответствовали именам служб в `/etc/services`, хотя это не является требованием.

Далее приведен пример 21.8 содержимого файла индивидуальной настройки для службы FTP, реализованной с помощью демона `vsftpd` (Very Secure FTP Daemon).

Пример 21.8. Файл настроек службы

```
cat /etc/xinetd.d/vsftpd
service ftp
{
    socket_type      = stream
    wait             = no
    user             = root
    server           = /usr/sbin/vsftpd
    server_args      = /etc/vsftpd/vsftpd.conf
    log_on_success   += DURATION USERID
    log_on_failure   += USERID
    nice             = 10
    disable          = yes
}
```

Настройка `disable = yes` запрещает супердемону `xinetd` прослушивать порт для запуска данной службы. Если необходимо, чтобы супердемон прослушивал данный порт и запускал демон этой службы, следует установить `disable = no` и послать сигнал `SIGHUP` супердемону (пример 21.9).

Пример 21.9. Включение службы `xinetd`

```
# cat /etc/xinetd.d/vsftpd
service ftp
{
    socket_type      = stream
    wait             = no
    user             = root
    server           = /usr/sbin/vsftpd
    server_args      = /etc/vsftpd/vsftpd.conf
    log_on_success   += DURATION USERID
    log_on_failure   += USERID
    nice             = 10
    disable          = no
}

# pkill -1 xinetd
# netstat -ta
Active Internet connections (servers and established)
```

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	*:ssh	*:*	LISTEN
tcp	0	0	*:ftp	*:*	LISTEN

Из примера видно, что после того, как в файле настроек `/etc/xinetd.d/vsftpd` установка `disable = yes` была заменена на `disable = no` и супердемону `xinetd` был передан сигнал `SIGHUP`, порт FTP стал доступен для соединений.

Сервер `vsftpd` будет в данном случае исполняться от имени суперпользователя, что диктует настройка `user = root`. Причем он будет выполняться с пониженным приоритетом, что определено настройкой `nice = 10`.

При запуске супердемоном процесса `vsftpd` последнему в качестве аргумента будет передано имя файла его конфигурации `/etc/vsftpd/vsftpd.conf`.

Настройки `log_on_failure` и `log_on_success` изменены по сравнению с настройками по умолчанию: к ним добавляется информация о соединении, где `USERID` — идентификатор пользователя, а `DURATION` — длительность сеанса.

Задания

- С помощью `xinetd` разрешите доступ к `echo-udp` из вашей локальной сети.
- Настройте разрешение на доступ к этой службе с 9:00 до 10:30.
- Как с помощью `xinetd` перенаправить обращение службе на другой узел сети? *Подсказка:* изучите директиву `redirect` файла конфигурации `/etc/xinetd.conf`.
- Необходимо запустить службу `telnet` не на порту 23, а на порту 2323. Как это сделать? *Подсказка:* изучите примеры `man xinetd.conf`.

Глава 22



Службы удаленного доступа

Службы удаленного доступа позволяют получить терминальный доступ к удаленной системе или выполнить некоторую команду на удаленной системе. В этой главе вы познакомитесь со службами удаленного доступа. Здесь также обсуждаются вопросы безопасности, связанные с этими службами.

Служба telnet

Служба telnet позволяет открывать сеанс на удаленном компьютере. Клиентская часть этой службы представлена программой telnet, а серверная — in.telnetd. Сервер telnet по умолчанию прослушивает 23-й порт TCP и запускается супердемоном.

В примере 22.1 приведен файл конфигурации службы telnet для супердемона xinetd.

Пример 22.1. Конфигурация xinetd для службы telnet

```
service telnet
{
    socket_type      = stream
    wait            = no
    user            = root
    server          = /usr/sbin/in.telnetd
    log_on_success  += DURATION USERID
    log_on_failure  += USERID
    disable         = yes
}
```

В большинстве дистрибутивов служба telnet по умолчанию не активна (настройка `disable = yes`), поэтому для ее запуска следует активизировать (`disable = no`) и послать сигнал `SIGHUP` супердемону.

Для входа в сеанс на удаленной машине с помощью telnet необходимо выполнить команду telnet, указав имя или адрес удаленной машины (пример 22.2).

Пример 22.2. Вход в удаленный сеанс telnet

```
$ telnet note.class.edu
Trying 192.168.11.25...
Connected to note.class.edu.
Escape character is '^]'.
Connection closed by foreign host.
```

В данном примере сеанс связи с сервером telnet был закрыт, о чем было получено сообщение "Connection closed by foreign host". В данном случае это вызвано настройкой `only_from = 127.0.0.1` в файле конфигурации `xinetd` (пример 22.3).

Пример 22.3. Разрешение для подключения из конкретной сети

```
service telnet
{
...
    only_from      = 192.168.11.0/25
    disable        = no
}
```

```
$ telnet note.class.edu
Trying 192.168.11.25...
Connected to note.class.edu.
Escape character is '^]'.
Welcome! (note.class.edu)
note.class.edu login: apox
Password:
Last login: Wed Jun 30 22:34:19 on :0
```

В примере 22.3 явно разрешено устанавливать соединения со службой telnet, запущенной на данном узле с любых узлов сети 192.168.11.0/25.

Сервер telnet вывел строку сообщения из файла /etc/issue.net — "Welcome! (note.class.edu)" и приглашение ввести имя пользователя и пароль для аутентификации пользователя. После успешной аутентификации был запущен удаленный сеанс. В нем можно выполнять любые разрешенные для данного пользователя команды на удаленном компьютере.

При необходимости можно временно выйти из терминального режима работы клиента telnet в командный (пример 22.4). Для этого следует ввести escape-последовательность, указанную при запуске сеанса: <Ctrl>+<]>.

Пример 22.4. Терминальный режим работы клиента telnet

```
apox@note apox $
<Ctrl>+<]>
telnet> help
Commands may be abbreviated.  Commands are:
close                close current connection
logout              forcibly logout remote user and close the connection
display             display operating parameters
mode                try to enter line or character mode ('mode ?' for more)
open                connect to a site
quit                exit telnet
send                transmit special characters ('send ?' for more)
set                 set operating parameters ('set ?' for more)
unset               unset operating parameters ('unset ?' for more)
status              print status information
toggle              toggle operating parameters ('toggle ?' for more)
slc                 change state of special charaters ('slc ?' for more)
z                   suspend telnet
!                   invoke a subshell
environ             change environment variables ('environ ?' for more)
?                   print help information
telnet> quit
Connection closed.
```

Для выхода из удаленного сеанса можно либо воспользоваться обычными средствами завершения работы в оболочке (например, `exit`), либо ввести команду `quit`.

В силу того, что использование telnet не предусматривает никаких средств криптографической защиты средствами самого протокола telnet, то настоя-

тельно рекомендуется не использовать telnet для связи в сетях, не являющихся доверенными. Вместо этой службы следует использовать SSH (Secure Shell).

Клиентская часть службы telnet, тем не менее, исключительно удобна для тестирования разнообразных служб, работающих с собственными протоколами. Далее приведен пример проверки возможности связаться с сервером SSH, работающим с 22-м портом TCP (пример 22.5).

Пример 22.5. Использование клиента telnet для проверки работы SSH

```
$ telnet note.class.edu 22
Trying 192.168.11.25...
Connected to note.class.edu.
Escape character is '^]'.
SSH-1.99-OpenSSH_3.7.1p2
^]
telnet> quit
Connection closed.
```

В примере 22.5 продемонстрировано, что клиент telnet способен устанавливать соединение по порту, указанному в командной строке. Здесь указан порт 22 — SSH.

ЗАДАНИЯ

- Установите и запустите сервер telnet. Настройте супердемон так, чтобы соединения с вашей машиной могли быть установлены из вашей сети.
- Подключитесь с помощью telnet к SMTP-серверу на вашей машине (порт 25 TCP).

Службы удаленного доступа (r-services)

Служба telnet не является единственной возможностью для обеспечения входа в сеанс на удаленном компьютере. Имеется еще один набор команд для удаленного входа в сеанс и исполнения команд на удаленной машине. Имена этих команд начинаются с буквы r (от англ. *remote* — удаленный), поэтому их принято именовать r-командами:

- ❑ `rexec` — выполняет команду на удаленной машине, используя аутентификацию на базе имен пользователей и паролей, сервер для этой команды — `in.rexecd`;
- ❑ `rlogin` — удаленный сеанс, сервер для этой команды — `in.rlogind`;

- ❑ `rsh` — позволяет либо выполнить команду на удаленной машине, либо войти в сеанс с помощью `rlogin`, сервер для этой команды — `in.rshd`;
- ❑ `rcp` — команда удаленного копирования файлов, сервер — `in.rshd`.

Вывод `netstat` после активизации `r`-служб приведен в примере 22.6.

Пример 22.6. Открытые порты `r`-служб

```
$ netstat -ta
```

Active Internet connections (servers and established)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	*:exec	*:*	LISTEN
tcp	0	0	*:login	*:*	LISTEN
tcp	0	0	*:shell	*:*	LISTEN

Команды `rsh`, `rcp` и `rlogin` позволяют устанавливать доверительные отношения между узлами сети. Доверительные отношения означают, что пользователь, вошедший в сеанс на узле, являющемся доверенным для другого узла, может без аутентификации открыть сеанс на этом узле. Для этого создается либо общесистемный файл `/etc/hosts.equiv`, устанавливающий доверенные узлы и пользователей на них, либо пользователь может создать файл `.rhosts` в своем домашнем каталоге, указав в нем доверенные узлы (пример 22.7).

Пример 22.7. Файл `.rhosts`

```
# cat ~user/.rhosts
elan.class.edu
```

Здесь в домашнем каталоге пользователя `user` находится файл `.rhosts`. Этот файл предоставляет пользователю возможность открывать удаленный сеанс на этом узле с компьютера `elan.class.edu` без ввода пароля.

В файле `/etc/hosts.equiv` можно указывать узлы сети и пользователей на них для установления доверенных отношений (пример 22.8).

Пример 22.8. Файл доверительных отношений `/etc/hosts.equiv`

```
# cat /etc/hosts.equiv
+buddy
-bubuka
elan.class.edu -root
elan.class.edu +susel
```

В этом примере пользователь `buddy` имеет возможность открывать сеанс на данном узле без аутентификации с любого узла сети, пользователю `bubuka` доверенный вход запрещен. С узла `elan.class.edu` доверенный вход разрешен для пользователя `susel` и запрещен для `root`.

Пример 22.9. Сеанс `rlogin`

```
$ rlogin note.class.edu
Last login: Fri Mar 19 08:51:06 on tty2
susel> hostname
note.class.edu
susel> exit
rlogin: connection closed.
```

В примере 22.9 пользователь `susel` инициировал сеанс на узле `note.class.edu`, используя для этого `rlogin`. Так как данному пользователю на этом узле разрешен доверенный вход в сеанс, то пароль на вход запрошен не был.

Команда `rsh` позволяет выполнить команду на удаленной машине (пример 22.10).

Пример 22.10. Запуск команды на удаленном компьютере

```
$ rsh note.class.edu ls
tanya.rtf
```

Команда `rsh`, будучи вызвана пользователем, для которого установлены доверительные отношения, выполняет указанную в качестве аргумента команду `ls` на удаленном хосте без запроса пароля.

Пример 22.11 демонстрирует копирование файла с удаленной машины.

Пример 22.11. Копирование с удаленной машины

```
$ rcp note.class.edu:~/tanya.rtf .
$ ls *.rtf
tanya.rtf
```

Здесь, как и в предыдущем случае, не требуется ввода пароля пользователя, поскольку между узлами установлены доверительные отношения.

Как и `telnet`, `r`-команды не дают возможности шифрования, предоставляя, таким образом, злоумышленнику легкую возможность перехвата сетевого

трафика. Возможность установки доверительных отношений между узлами сети представляет собой особую опасность, т. к. IP-адреса могут быть подвргнуты подмене (spoofing).

Вместо telnet и r-команд рекомендуется использовать SSH.

Задания

- Сконфигурируйте xinetd для активизации r-служб.
- Настройте доверительный вход и проверьте работоспособность r-команд.
- Влияет ли наличие файла .rhosts на поведение команды rexec?
- Какая опция rcp позволяет копировать файлы рекурсивно?

Система SSH

Система SSH (Secure Shell) предоставляет защищенную с помощью криптографии надежную альтернативу r-командам и службе telnet. Она позволяет открывать зашифрованный канал связи между удаленными узлами, предоставляет возможности аутентификации с использованием публичных и частных ключей (несимметричное шифрование), защищает от подмены IP-адресов.

В GNU/Linux используется свободная версия системы SSH — OpenSSH.

Пакет OpenSSH представлен тремя основными программами:

- ❑ `sshd` — сервер SSH, прослушивающий 22-й порт TCP;
- ❑ `ssh` — клиент службы SSH, позволяющий инициировать удаленный сеанс;
- ❑ `scp` — программа для удаленного копирования.

Серверная и клиентская части системы OpenSSH обладают разными файлами конфигурации: сервер `sshd` имеет конфигурационный файл `/etc/ssh/sshd_config`, а клиенты `ssh` и `scp` — `/etc/ssh/ssh_config`.

Сервер OpenSSH запускается самостоятельно (stand-alone) с помощью сценария в каталоге `/etc/init.d`, поэтому для автоматического старта сервера OpenSSH при переходе в многопользовательский режим следует надлежащим образом сконфигурировать систему инициализации.

Для инициирования сеанса на удаленной машине с запущенным сервером OpenSSH достаточно на клиентском узле просто выполнить команду `ssh`, указав ей в качестве аргумента имя или IP-адрес узла назначения (пример 22.12).

Команда `scp` скопировала с удаленного компьютера файл `linux.tar` и поместила его в каталог `/tmp` на локальной машине.

Можно в явном виде указать имя пользователя на удаленной машине (пример 22.15).

Пример 22.15. Явное указание имени пользователя в `scp`

```
susel$ scp webadm@note:/var/www/html/susel.php
webadm@note's password:
susel.php                                100%   22KB   0.0KB/s   00:00
```

В этом примере пользователь `susel` скопировал с удаленной машины файл, пройдя процедуру аутентификации на ней, как пользователь `webadm`.

Несмотря на то, что команды `ssh` и `scp` используют зашифрованный канал, во многих случаях аутентификацию с помощью пароля нельзя признать безопасной. В таких случаях можно использовать криптографическую аутентификацию. SSH предоставляет возможность использовать аутентификацию по протоколам RSA и DSA.

Первое, что необходимо сделать для обеспечения возможности аутентификации с помощью RSA, — это создать командой `ssh-keygen` пару ключей несимметричного шифрования (пример 22.16).

Пример 22.16. Генерация ключей

```
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/spa/.ssh/id_rsa):
Created directory '/home/spa/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/spa/.ssh/id_rsa.
Your public key has been saved in /home/spa/.ssh/id_rsa.pub.
The key fingerprint is:
3b:51:43:a3:67:5b:5b:27:fa:f6:fc:84:c3:6c:9b:67 spa@note.class.edu
```

Эта команда создает пару ключей RSA. Они помещаются в подкаталог `.ssh` домашнего каталога пользователя, вызвавшего команду. Файл `id_rsa` содержит частный ключ, доступ к которому должен быть предоставлен лишь его

владельцу. Файл `id_rsa.pub` должен быть помещен на удаленный узел, с которым необходимо обеспечить связь по зашифрованному каналу.

Парольная фраза, которую требуется ввести здесь — это "пропуск" к частному ключу. Допускается не вводить ее, но в таком случае частный ключ не будет защищен.

Несимметричное шифрование используется лишь на стадии аутентификации. После подтверждения аутентичности пользователя дальнейшая связь осуществляется с применением симметричного шифрования, т. к. оно обеспечивает приемлемый уровень быстродействия системы, а несимметричное — нет.

После создания пары ключей несимметричного шифрования, необходимых для аутентификации пользователя, требуется поместить публичный ключ на удаленный хост. Для этого можно зайти на него, используя `ssh`, и скопировать с помощью `scp` публичный ключ с собственного узла на удаленный узел (пример 22.17).

Пример 22.17. Копирование публичного ключа на удаленный узел

```
$ ssh mainfrank.class.edu
spa@mainfrank.class.edu's password:
Last login: Fri Jul  2 10:13:06 2004
spa% hostname
mainfrank.class.edu
spa% scp note.class.edu:~/.ssh/id_rsa.pub secret.pub
id_rsa.pub                                100%  22KB  0.0KB/s   00:00
spa% cat secret.pub >> .ssh/authorized_keys ; rm -f secret.pub
spa% chmod 600 .ssh/authorized_keys
spa% exit
```

На первом шаге этой процедуры пользователь `spa` открывает удаленный сеанс на узле `mainfrank.class.edu`, куда требуется передать публичный ключ.

Далее он копирует, находясь в сеансе на удаленном узле, публичный ключ со своего компьютера во временный файл на `mainfrank.class.edu`.

После этого ключ добавляется к базе данных публичных ключей, находящейся в файле `.ssh/authorized_keys`, права доступа к которому должны быть ограничены лишь его владельцем.

После выхода из сеанса следующий инициированный сеанс будет открыт при условии успешной аутентификации с помощью частного и публичного ключей.

ЗАДАНИЯ

- Запустите службу SSH.
- Проверьте работоспособность команд `ssh` и `scp`.
- Создайте пару ключей RSA и поместите публичный ключ на желаемый сервер. Оставьте парольную фразу пустой — в таком случае при успешном обмене ключами вводить пароль для доступа к удаленному узлу не потребуется.



Глава 23

Служба FTP

FTP — одна из наиболее распространенных служб в Интернете. Прочитав эту главу, вы получите сведения о базовой настройке FTP-сервера и его защите.

Как работает служба FTP

Служба FTP реализует одноименный протокол для передачи файлов с одного узла сети на другой (FTP — File Transfer Protocol). Сервер FTP позволяет по запросу клиента FTP передавать требуемые файлы на клиентский узел.

При запуске FTP-сервера он открывает порт TCP 21, предназначенный для инициации FTP-соединения и передачи управляющей информации. При инициации клиентом FTP-соединения происходит аутентификация пользователя, при успешном прохождении которой запускается сеанс FTP. После этого пользователь получает возможность копирования файлов между удаленным и локальным компьютерами. Передача данных производится в отдельном TCP-соединении по порту 20.

Таким образом, при работе службы FTP открываются сразу два TCP-соединения:

- ❑ первое иницируется клиентским узлом с порта источника с номером выше 1024 (обычно более 30 000) и портом назначения 21 на серверной стороне; это соединение используется для передачи управляющей информации;
- ❑ второе иницируется серверным узлом для обмена данными, использующим 20-й порт источника на серверном узле и порт назначения с номером выше 1024 на клиентском узле; причем клиент предварительно информирует сервер о номере открытого для передачи данных порта.

Данная схема описывает работу *активного* FTP-соединения, вызывающего трудности при организации работы IP-фильтра, поскольку порт назначения

для передачи данных на клиентской машине заранее неизвестен. Для обхода описанной проблемы используется *пассивный* режим FTP. В этом случае соединение для передачи данных инициируется не сервером с порта 20 TCP, а клиентом с порта выше 1024 на порт сервера с номером выше 1023. Номер порта на сервере для передачи данных заранее сообщается клиенту.

В настоящее время в GNU/Linux используется множество различных программ, реализующих FTP-серверы. Среди них: BSD `ftpd`, `wuftp`, `proftpd` и `vsftpd`. Большинство из них может работать как в самостоятельном режиме (*stand-alone*), так и с помощью супердемона. Имеется также множество реализаций FTP-клиентов. Наиболее популярны: `ftp`, `lftp`, `nftp` и `wget`.

Часто серверы FTP предоставляют возможность анонимного использования. Для этих целей используются специальные имена анонимных пользователей: `anonymous` или `ftp`. При анонимном доступе к FTP-серверу вместо пароля вводят почтовый адрес или вообще ничего. Большинство анонимных FTP-серверов сконфигурировано только для копирования данных с них (*download*), но имеются и поддерживающие загрузку файлов на них (*upload*). В последнем случае следует предпринимать особые меры предосторожности, т. к. анонимный сервер с возможностью загрузки файлов на него — отличная цель для атак и иной незаконной деятельности.

По историческим причинам протокол FTP различает два режима передачи файлов: текстовый (`ascii`) и бинарный (`binary`). Передача бинарного файла в текстовом режиме приведет к искажению передаваемой информации.

ЗАДАНИЯ

- Определите, какие программы FTP установлены в вашей системе.
- Какие конфигурационные файлы используются сервером FTP в вашей системе?
- Поддерживают ли имеющиеся клиенты FTP пассивный режим?

Настройка сервера *vsftpd*

Конфигурационный файл сервера `vsftpd` — `/etc/vsftpd.conf` (пример 23.1).

Пример 23.1. Конфигурационный файл *vsftpd*

```
anonymous_enable=YES
dirmessage_enable=YES
connect_from_port_20=YES
xferlog_enable=YES
```

```
xferlog_std_format=YES  
xferlog_file=/var/log/vsftpd/vsftpd.log  
nopriv_user=nobody
```

Смысл этих и других настроек `vsftpd` таков:

- ☐ настройка `anonymous_enable=YES` разрешает анонимный вход в сеанс FTP;
- ☐ `dirmessage_enable=YES` разрешает использовать файлы сообщений в каталогах FTP-сервера;
- ☐ `connect_from_port_20=YES` разрешает работу в режиме активного FTP;
- ☐ ведение журнала передачи файлов определяется настройкой `xferlog_enable`;
- ☐ стандартный формат журнала задается `xferlog_std_format=YES`;
- ☐ имя файла журнала задает `xferlog_file`;
- ☐ имя непривилегированного пользователя — `nopriv_user`;
- ☐ `local_enable=YES` — разрешение реальным пользователям использовать FTP;
- ☐ `write_enable=YES` — разрешение загружать файлы для пользователей;
- ☐ `local_umask=022` — `umask` для файлов, копируемых на сервер;
- ☐ `anon_upload_enable=YES` — разрешение загружать на сервер файлы анонимам;
- ☐ `anon_mkdir_write_enable=YES` — разрешение создавать каталоги анонимам;
- ☐ `chown_uploads=YES` — менять владельца загруженных на сервер файлов, имя нового владельца задается с помощью `chown_username`;
- ☐ `ascii_upload_enable` и `ascii_download_enable` — текстовый режим;
- ☐ `ftpd_banner` — задает приветствие, выдаваемое сервером в начале сеанса;
- ☐ `deny_email_enable=YES` — настройка, позволяющая использовать файл с адресами e-mail анонимных пользователей, для которых вход запрещен;
- ☐ `banned_email_file` — имя файла с e-mail запрещенных пользователей;
- ☐ `chroot_list_enable=YES` — разрешение подменять корневой каталог на домашний каталог пользователя (`chroot`);
- ☐ `chroot_list_file` — файл со списком пользователей, для которых требуется осуществлять подмену корневого каталога домашним;
- ☐ `ls_recurse_enable=YES` — разрешает использовать опцию `-R` команды `ls`;
- ☐ `listen=YES` — используется для запуска `vsftpd` в самостоятельном режиме;
- ☐ `background=YES` — переводит сервер в режим демона.

Сервер `vsftpd` запускается и в самостоятельном режиме, и с помощью супер-демона.

Часто файлы, доступные для анонимных пользователей, размещаются в каталоге `/var/ftp/pub`. Обычно это домашний каталог пользователя `ftp`, который, тем не менее, не является владельцем этого каталога. Сервер `vsftpd` требует, чтобы данный каталог не был доступен для записи для пользователя `ftp`.

Вход в сеанс может быть запрещен для конкретного пользователя с помощью файла `/etc/ftpusers`. Пользователи, указанные в нем, не имеют доступа к FTP-серверу.

ЗАДАНИЕ

- Настройте `vsftpd` для разрешения реальным пользователям входить в сеанс с возможностью загружать файлы на сервер.

Клиенты FTP

Команда `ftp` запускает клиента FTP. Он обладает собственными встроенными командами, список которых можно получить, используя команду `help` (пример 23.2).

Пример 23.2. Команды клиента `ftp` (фрагмент)

```
$ ftp
ftp> help
Commands may be abbreviated.  Commands are:
!          debug          mdir          sendport      site
$          dir            mget          put           size
...
ftp> quit
```

Важнейшие встроенные команды `ftp`:

- ☐ `open` — устанавливает соединение с FTP-узлом;
- ☐ `user` — задает имя пользователя для входа в сеанс FTP;
- ☐ `ascii` — устанавливает режим передачи ASCII-файлов;
- ☐ `binary` — устанавливает режим передачи не ASCII-файлов;
- ☐ `ls` — выводит содержимое FTP-каталога в подробном формате;
- ☐ `dir` — получение списка файлов текущего FTP-каталога;
- ☐ `cd` — изменяет текущий FTP-каталог;

- ❑ `get` — копирует файл с удаленного компьютера;
- ❑ `mget` — получает несколько файлов с FTP-узла;
- ❑ `put` — копирует файл на удаленный компьютер;
- ❑ `mput` — копирует несколько файлов на удаленный компьютер;
- ❑ `reget` — продолжает копирование файла с места, где была прервана связь;
- ❑ `restart` — перезапускает передачу файла с заданной позиции;
- ❑ `size` — отображает размер файла;
- ❑ `prompt` — отменяет или разрешает запрос подтверждения на передачу файлов;
- ❑ `bye` — завершает сессию;
- ❑ `!` — выход в shell.

Аргументом можно указать адрес сервера FTP для подключения (пример 23.3).

Пример 23.3. Открытие сеанса FTP с помощью клиента `ftp` (фрагмент)

```
$ ftp sinix.mainfrank.de
Name (sinix.mainfrank.de:susel): test1
Password:
ftp> ls
drwx-----   3 1001      100          128 Mar 17 14:37 Desktop
ftp> put lsmod.asp
1305 bytes sent in 5.1e-05 secs (2.5e+04 Kbytes/sec)
ftp> bye
221 Goodbye.
```

Другим популярным FTP-клиентом является программа `wget`, не работающая интерактивно. Все параметры для получения файлов задают в ее командной строке. Далее приведен пример 23.4 использования команды `wget` для получения файла.

Пример 23.4. Команда `wget` (фрагмент)

```
$ wget ftp://newnote.net-burg.com/pub/*.stat* .
```

Команда `wget` позволяет указывать в URL шаблоны имен файлов.

Часто используемые опции `wget`:

- ☐ `-b` — после запуска перейти в фоновый режим;
- ☐ `-c` — возобновление загрузки в случае разрыва соединения;
- ☐ `-v` — режим подробного информирования о действиях (*verbose*);
- ☐ `-q` — режим минимального информирования о действиях (*quiet*);
- ☐ `-i <file>` — взять URL файлов для скачивания из заданного файла;
- ☐ `-Y <on/off>` — передача файлов посредством *проxy*-сервера;
- ☐ `-r` — рекурсия;
- ☐ `-l <num>` — ограничить рекурсию до *num* уровня вложенности каталогов;
- ☐ `-x` — создавать требуемые каталоги;
- ☐ `--passive-ftp` — включить пассивный режим FTP.

ЗАДАНИЕ

- Получите свежую версию `wget`, используя саму команду `wget`.

Глава 24



Файловая система NFS

Сетевая файловая система NFS чрезвычайно широко используется в мире UNIX- и GNU/Linux-систем. В этой главе вы узнаете, что такое разделяемые ресурсы NFS, как их экспортируют и монтируют. Здесь также будут показаны основные утилиты NFS.

Настройка сервера NFS

Сетевая файловая система NFS (Network File System) позволяет предоставлять файловые ресурсы одних компьютеров в сети для других. Эта служба чаще всего используется в UNIX- и GNU/Linux-сетях.

Файловая система NFS позволяет монтировать сетевые дисковые ресурсы с помощью обычной команды `mount`, поэтому с точки зрения пользователя сетевые ресурсы NFS не отличаются от локальных дисковых ресурсов.

Служба NFS основана на использовании протокола RPC (Remote Procedure Call). Удаленная процедура (Remote Procedure) — это программа, выполняющаяся на удаленном компьютере и производящая заданные действия по требованию клиента. Этот протокол требует отображения удаленных процедур на порты протокола TCP или UDP. Эту функцию выполняет специальная программа Portmapper (в GNU/Linux ей соответствует исполняемый файл `portmap` либо `rpcbind`).

Программа RPC-сервер сообщает Portmapper, какой порт он будет использовать для обслуживания клиентских соединений. Клиент должен связаться с демоном `portmap` на сервере для определения номера порта, прослушиваемого сервером RPC.

Основной конфигурационный файл сервера NFS — `/etc/exports`. В этом файле определяется список ресурсов, предоставляемых этим файловым сервером, и ограничения на доступ к этим ресурсам.

Допустим, что требуется предоставить каталог `/home/export` в общее использование, как публичный ресурс с непривилегированным доступом только для чтения. Доступ к данному ресурсу должен быть разрешен лишь для узлов сети `192.168.199.0/24` (24 бита в сетевой части — сеть класса C) — пример 24.1.

Пример 24.1. Настройка разделяемого ресурса NFS

```
/home/export 192.168.199.0/24(ro,async,all_squash,insecure)
```

Каталог `/home/export` предоставляется для анонимного доступа только для чтения в пределах сети `192.168.199.0/24`. Настройка `all_squash` предназначена для отображения запросов на доступ к ресурсу на непривилегированного пользователя `nobody`. Директива `insecure` разрешает использовать для соединения непривилегированный порт клиента (большой 1024), а директива `async` — асинхронный ввод/вывод.

В файле `/etc/exports` могут быть наложены ограничения на доступ к разделяемым ресурсам NFS с помощью указания:

- ☐ единственного имени узла или его IP-адреса клиента;
- ☐ адреса сети или имени сети из файла `/etc/networks`, причем при указании адреса сети должна быть указана маска сети или префикс (например, `192.168.199.0/24` — при указании префикса, или `192.168.199.0/255.255.255.0`);
- ☐ шаблона `*` или `?` в FQDN (полностью специфицированном доменном имени) имени узла (например, `*.class?.mynet.edu` — разрешено использование ресурса всех узлов из поддоменов `class1`, `class2` и т. д. домена `mynet.edu`).

Далее приведен список основных опций конфигурирования `/etc/exports`:

- ☐ `secure` — разрешает запросы только с привилегированных портов (меньше 1024);
- ☐ `rw` — разрешает доступ для записи;
- ☐ `async` — разрешает отвечать клиенту до того, как будут завершены предыдущие операции записи, что повышает быстродействие;
- ☐ `no_wdelay` — разрешает одновременное выполнение нескольких запросов на запись к диску для повышения быстродействия;
- ☐ `root_squash` — отображать запросы на доступ к ресурсу с `UID=0` на непривилегированный (по умолчанию `nobody`);
- ☐ `no_root_squash` — не отображать суперпользователя на непривилегированного пользователя при доступе к ресурсу;
- ☐ `anonuid` и `anongid` — указывают `UID` и `GID` анонимного пользователя.

ЗАДАНИЯ

- Установите анонимный доступ для чтения к каталогу `/home/export` для узлов, принадлежащих только тому же домену, что и ваш узел. Все подключения должны рассматриваться как анонимные. Подключения должны быть разрешены с непривилегированных портов.
- Разрешите доступ для записи к каталогу `/home/pub` с отображением `root` на непривилегированного пользователя. Для данного ресурса должна быть разрешена асинхронная запись.

Использование сервера NFS

Стартовать сервер NFS можно с помощью сценария в `/etc/init.d` (пример 24.2).

Пример 24.2. Запуск NFS-сервера

```
# /etc/init.d/nfs start
```

В GNU/Linux имеется специальная команда `exportfs` для управления ресурсами NFS-сервера. Она обычно вызывается непосредственно из сценария запуска NFS-сервера с опцией `-a` для экспортирования всех ресурсов, описанных в `/etc/exports`. Используя `exportfs -u`, можно отменить экспорт ресурса.

Проверить, какие ресурсы предоставлены в данной системе, можно с помощью команды `showmount -e` (пример 24.3).

Пример 24.3. Получение списка экспортированных NFS-ресурсов

```
# showmount -e
Export list for netsrv.class.edu:
/home/export 192.168.199.0/24
```

Эта команда демонстрирует, что на данном компьютере имеется NFS-ресурс `/home/export`, доступный для узлов сети `192.168.199.0/24`.

После запуска NFS-сервера можно проверить зарегистрированные Portmapper вызовы RPC (пример 24.4).

Пример 24.4. Проверка зарегистрированных RPC

```
# rpcinfo -p
program vers proto  port  service
100000    4    tcp    111   portmapper
```

100000	3	tcp	111	portmapper
100000	2	tcp	111	portmapper
100000	4	udp	111	portmapper
100000	3	udp	111	portmapper
100000	2	udp	111	portmapper
100005	1	udp	57125	mountd
100005	1	tcp	35007	mountd
100005	2	udp	57125	mountd
100005	2	tcp	35007	mountd
100005	3	udp	57125	mountd
100005	3	tcp	35007	mountd
100024	1	udp	37954	status
100021	1	udp	54572	nlockmgr
100021	3	udp	54572	nlockmgr
100021	4	udp	54572	nlockmgr
100021	1	tcp	52437	nlockmgr
100021	3	tcp	52437	nlockmgr
100021	4	tcp	52437	nlockmgr
100003	2	udp	2049	nfs
100003	3	udp	2049	nfs
100003	4	udp	2049	nfs
100003	2	tcp	2049	nfs
100003	3	tcp	2049	nfs
100003	4	tcp	2049	nfs
100024	1	tcp	38634	status

Команда `rpcinfo -p` показывает соответствие вызовов RPC портам TCP и UDP, а также имя соответствующего сервера RPC. В столбце `vers` выводится версия протокола. Например, для `nfs` поддерживаются версии 2, 3 и 4.

Смонтировать файловый ресурс NFS можно так, как это показано в примере 24.5.

Пример 24.5. Монтирование файловой системы NFS

```
# mount -t nfs -r netsrv.class.edu:/home/export /mnt
# mount
...
netsrv.class.edu:/home/export on /mnt type nfs (ro,addr=netsrv.class.edu)
```

При монтировании необходимо указать имя или IP-адрес компьютера, предоставляющего ресурс, через двоеточие имя каталога этого ресурса на сервере NFS и точку монтирования.

Задания

- Экспортируйте ресурс /home с доступом только на чтение.
- Получите список зарегистрированных RPC.

Отличия протокола NFSv4

Новая версия протокола NFSv4 значительно улучшила надежность и защищенность протокола NFS по сравнению со второй и третьей версиями. В протоколе NFSv4 введено понятие состояния, поэтому он использует в качестве транспорта протокол TCP.

Конфигурация NFSv4 несколько отличается от предыдущих версий NFS, т. к. в четвертой версии экспортируемые ресурсы сведены в единую псевдофайловую систему, которая также должна быть экспортирована (пример 24.6).

Пример 24.6. Файл /etc/exports для NFSv4

```
/export      *(fsid=0,crossmnt,ro,root_squash,sync,no_subtree_check)
/export/pub   *(ro,root_squash,sync,no_subtree_check)
```

Псевдофайловая система /export содержит в этом примере единственный ресурс — каталог /export/pub. Для самой псевдофайловой системы, являющейся корневой для разделяемых ресурсов NFSv4, должна быть установлена опция `fsid=0`. Опция `crossmnt` позволяет монтировать все ресурсы NFSv4 в корневой псевдофайловой системе.

Реальные файловые системы можно смонтировать к разделяемым ресурсам NFSv4 так, как показано в примере 24.7.

Пример 24.7. Монтирование каталога к разделяемому ресурсу NFSv4

```
mount --bind /home/user1/eBook /export/pub
```

Здесь каталог /home/user1/eBook смонтирован к разделяемому ресурсу NFSv4 /export/pub.

И на клиентской стороне и на сервере должен работать сервер `rpc.idmapd`, который обычно поставляется в пакете `nfs-client` или `nfs-common`. В примере 24.8 показано, как монтируется разделяемый ресурс NFSv4.

Пример 24.8. Монтирование разделяемого ресурса NFSv4

```
mount -t nfs4 netsevr.class.edu:/pub /mnt
```

Обратите внимание, что в имени разделяемого ресурса NFSv4 не указывают имя корневого каталога псевдофайловой системы. Указывают имя ресурса внутри нее.

ЗАДАНИЕ

- Измените конфигурацию для NFSv3 из предыдущего задания на NFSv4.



Глава 25

SMB/CIFS-сервер SAMBA

Пакет SAMBA позволяет работать GNU/Linux-машине в сети MS Windows. В этой главе вы познакомитесь с основными вариантами конфигурации SAMBA и ее важнейшими утилитами.

Состав пакета SAMBA

Пакет SAMBA представляет собой набор программ, реализующий сервер SMB/CIFS (Server Message Block) в UNIX/Linux-системах. Протокол SMB и его дальнейшее развитие — CIFS (Common Internet File System) являются наследниками протокола NetBIOS, разработанного в IBM. Эти протоколы лежат в основе функциональности сетей MS Windows, поэтому пакет SAMBA обеспечивает возможности интеграции GNU/Linux-систем в Windows-сети.

Используя SAMBA, работающая с GNU/Linux система способна использовать и предоставлять ресурсы для Windows-сети. В настоящий момент реализованы два вида разделяемых в сети Windows ресурсов: файловые ресурсы и разделяемые принтеры.

В распространенных дистрибутивах GNU/Linux пакет SAMBA разделен на несколько пакетов: `samba`, `samba-server`, `samba-client` и т. д. Важнейшие программы SAMBA 3.4.x:

- ☐ `smbd` — сервер SMB/CIFS;
- ☐ `nmbd` — сервер разрешения NetBIOS-имен;
- ☐ `smbclient` — клиентская программа для обращения к ресурсам сети;
- ☐ `testparm` — утилита, для проверки правильности конфигурации SAMBA;
- ☐ `smbstatus` — программа, выводящая статус SAMBA;
- ☐ `nmblookup` — утилита, позволяющая проверять разрешение имен NetBIOS;

- ❑ `smbpasswd` — команда для установки и изменения паролей пользователей;
- ❑ `mount.cifs` — команда монтирования сетевых файловых ресурсов SMB/CIFS;
- ❑ `smbtar` — сценарий для архивирования файловых ресурсов;
- ❑ `winbindd` — демон, позволяющий получать информацию от серверов Windows;
- ❑ `wbinfo` — программа, позволяющая запрашивать информацию от `winbindd`;
- ❑ `findsmb` — скрипт Perl для получения списка SMB-машин в сети;
- ❑ `smbcontrol` — посылает сигналы демонам `smbd` и `nmdbd`;
- ❑ `smbspool` — посылает задания на печать на SMB-сервер печати;
- ❑ `pdbedit` — программа для управления SAMBA-пользователями;
- ❑ `swat` — программа для настройки и управления SAMBA с Web-интерфейсом.

Задания

- Определите версию SAMBA, установленную в вашей системе.
- Проверьте, установлена ли программа `mount.cifs`. Из какого пакета?

Настройка SAMBA

Основной конфигурационный файл, задающий поведение серверов `smbd` и `nmdbd`, — это `/etc/samba/smb.conf`. Настройка SAMBA может быть осуществлена непосредственным редактированием файла `smb.conf` либо с помощью многочисленных утилит, в число которых входит программа SWAT. Для активизации SWAT следует сконфигурировать супердемон для прослушивания порта 901 TCP. Обратиться к SWAT можно с помощью Web-браузера по 901-му порту.

Далее приводится пример 25.1 несложной конфигурации сервера SAMBA, позволяющего ему предоставлять ресурсы машин в сеть Windows.

Пример 25.1. Простейшая конфигурация SAMBA

```
# Global parameters
[global]

    netbios name = NOTE
    workgroup = CLASS
    security = user
    guest account = nobody
```

```
[homes]
    Home directories
    read only = No

[PUB]
    comment = Public share
    path = /home/samba/pub
    read only = No
    guest ok = Yes

[BUX]
    path = /home/samba/buxgalter
    browseable = No
    valid users = buxgalter
```

В примере 25.1 использованы следующие параметры настроек SAMBA:

- ☐ netbios name — имя компьютера в сети NetBIOS;
- ☐ workgroup — имя рабочей группы NetBIOS;
- ☐ security — вариант аутентификации для получения доступа к ресурсам:
 - share — права доступа проверяются непосредственно при доступе к ресурсу;
 - user — права доступа проверяются в момент подключения к компьютеру, предоставляющему ресурсы (используется по умолчанию);
 - domain — компьютер является членом домена NT, и проверка пользователей производится на PDC (Primary Domain Controller) или BDC (Backup Domain Controller);
 - server — аутентификация на SMB-сервере, не являющемся PDC или BDC;
 - ads — SAMBA будет работать как член домена ADS (Active Domain Service);
- ☐ guest account — имя анонимного пользователя;
- ☐ comment — необязательный комментарий, устанавливаемый на ресурс;
- ☐ path — путь доступа на UNIX-машине к каталогу, предоставляемому в качестве разделяемого файлового ресурса, либо для принтеров — к очереди печати;
- ☐ read only — доступ только для чтения;
- ☐ guest ok — разрешение гостевого доступа к данному ресурсу;
- ☐ browseable — разрешение отображать этот ресурс в сетевом окружении.

Файл `smb.conf` содержит секции, каждая из которых имеет свое имя, указываемое в квадратных скобках. Имеются три особые секции:

- ❑ `global` — раздел общих настроек;
- ❑ `printers` — описывает методы подключения к принтерам;
- ❑ `homes` — настраивает доступ к домашним каталогам пользователей.

В приведенном примере 25.1 в разделе `global` установлено NetBIOS-имя машины и принадлежность ее к рабочей группе `class`. Также задан порядок аутентификации и авторизации — права доступа к ресурсам этой машины будут проверяться при попытке доступа к ней. Также здесь установлено имя пользователя UNIX, которое будет использовано для гостевого доступа с минимальными правами. Также здесь разрешено использование пустых паролей, что удобно при разрешении гостевого доступа к некоторым ресурсам на данном сервере. Исключительно важно правильно установить кодовые страницы, используемые клиентской и серверной операционными системами для правильного хранения файлов.

Обобщающий ресурс `homes` описывает доступ к домашним каталогам пользователей. Путь к ним здесь установлен с помощью макроса `%u` — имя пользователя. Настройка `path = /home/%u` задает нахождение домашних каталогов в `/home`, а имена каталогов совпадают с именами их пользователей.

Для каждого разделяемого ресурса в файле `smb.conf` устанавливается собственный раздел с именем этого ресурса. В примере 25.1 имеются два файловых ресурса:

- ❑ `pub` — для публичного доступа;
- ❑ `bux` — только для пользователя `buxgalter`.

Причем для `pub` разрешен доступ для записи, а для `bux` — нет. Более того, для ресурса `bux` запрещено его отображение в сетевом окружении.

Настроив файл `smb.conf`, следует обязательно проверить правильность установок с помощью программы `testparm` (пример 25.2).

Пример 25.2. Проверка правильности конфигурации `smb.conf`

```
$ testparm -s | less
```

Если среди настроек в файле `smb.conf` имеется ошибка, то `testparm` выдаст сообщение об этом. Кроме того, `testparm` выводит полный список опций конфигурации с их значениями. Опция `-s` требуется для того, чтобы перед выводом полного списка конфигурационных опций не надо было нажимать клавишу <Enter>.

ЗАДАНИЯ

- Какая настройка `security` требуется в системе, которая должна предоставлять общедоступный гостевой файловый ресурс на запись без пароля?
- Создайте файл настроек для такого доступа и проверьте его правильность.
- Предположим, что в примере 25.1 требуется разрешить доступ к ресурсу `BUX` группе `users`. Как это сделать?

Запуск и работа системы SAMBA

Запуск демонов `smbd` и `nmbd` производится сценарием в `/etc/init.d` (пример 25.3).

Пример 25.3. Запуск SAMBA

```
# /etc/init.d/samba start
Starting samba... [ ok ]
# netstat -ta
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:netbios-ssn          :::*                     LISTEN
```

После старта сервер `smbd` прослушивает 139-й порт TCP (`netbios-ssn`).

Команда `nmblookup` проверяет разрешение NetBIOS-имен, за которое отвечает сервер `nmbd` (пример 25.4).

Пример 25.4. Проверка разрешения NetBIOS-имен

```
$ nmblookup NOTE
querying NOTE on 192.168.111.127
192.168.111.25 NOTE<00>
```

Команда `nmblookup` посылает широковещательный запрос с целью определить IP-адрес компьютера в данной сети, который имеет требуемое NetBIOS-имя.

В сети NetBIOS используется *мастер-браузер* — компьютер, обслуживающий базу данных NetBIOS-имен компьютеров в данной сети. Им бывает PDC либо один из компьютеров одноранговой сети. При отсутствии PDC производятся выборы мастер-браузера. Для определения, какой компьютер в сети является мастер-браузером, можно использовать команду из примера 25.5.

Пример 25.5. Определение мастер-браузера

```
$ nmblookup -M -- -
querying __MSBROWSE__ on 192.168.111.127
192.168.111.25 __MSBROWSE__<01>
```

В пакете SAMBA имеется удобная интерактивная утилита `smbclient`, предоставляющая возможности доступа к SMB-серверам. С ее помощью можно получить список ресурсов на сервере SMB (примеры 25.6 и 25.7).

Пример 25.6. Получение списка разделяемых ресурсов (фрагмент)

```
$ smbclient -L //NOTE
Password:
Domain=[CLASS] OS=[Unix] Server=[Samba 3.4.2]

    Sharename      Type            Comment
    -----
    homes          Disk           Home directories
    PUB            Disk           Public share
```

На приглашение ввести пароль можно просто нажать клавишу <Enter>.

Данная команда позволяет получить список разделяемых ресурсов на сервере. Обратите внимание, что в этом примере разделяемый ресурс `vux` не представлен в списке, т. к. в его настройках присутствует параметр `browseable = no`.

Пример 25.7. Доступ к публичному ресурсу (фрагмент)

```
$ smbclient //NOTE/pub
Password:
Domain=[CLASS] OS=[Unix] Server=[Samba 3.4.2]
smb: \> help
?                altname          archive          blocksize        cancel
cd               chmod           chown           del              dir
...
smb: \> ls
.                D              0      9  3 12:35:00 2009
..              D              0      9  3 12:29:56 2009
linux.tar       215040      25  3 17:20:24 2009
```

```
38153 blocks of size 262144. 21261 blocks available
smb: \> get linux.tar
getting file \linux.tar of size 0 as linux.tar (104994.9 kb/s) (average
105000.0 kb/s)
smb: \> quit
```

Из примера 25.7 видно, что клиентская программа `smbclient` предоставляет аналогичные `ftp` встроенные команды, например, `help`. Для получения файлов с разделяемого SMB-ресурса можно использовать команды `get` и `mget`.

Для получения доступа к ресурсу `bux` пользователь `buxgalter` должен быть зарегистрирован в базе данных учетных записей SAMBA `smbpasswd` с помощью одноименной команды (пример 25.8). Перед выполнением команды `smbpasswd` он уже должен быть зарегистрирован, как UNIX-пользователь.

Пример 25.8. Регистрация в SAMBA пользователя

```
# useradd -M -d /home/samba/buxgalter/ buxgalter
# smbpasswd -a buxgalter
New SMB password:
Retype new SMB password:
unable to open passwd database.
Added user buxgalter.
```

Пользователь `buxgalter` зарегистрирован в базе данных учетных записей SAMBA. Тип хранилища базы данных задается настройкой `passwd backend`. По умолчанию используется тип хранилища `tdbsam`.

Пользователь `buxgalter` может теперь с помощью `smbclient` получить доступ к файлам, расположенным на ресурсе `bux`. Команда `smbclient` позволяет явно указать имя пользователя для доступа к ресурсу с помощью опции `-U` (пример 25.9).

Пример 25.9. Подключение к ресурсу от имени заданного пользователя

```
$ smbclient //NOTE/BUX -U buxgalter
```

В этом случае должен быть введен верный пароль пользователя.

Аналогично осуществляется доступ к домашним каталогам пользователей, которые зарегистрированы в SAMBA с помощью `smbpasswd` (пример 25.10).

Пример 25.10. Подключение к домашнему каталогу

```
$ smbclient //NOTE/user1
Password:
Domain=[CLASS] OS=[Unix] Server=[Samba 3.4.2]
smb: \> ls
.                D            0   10 3 17:11:15 2009
..               D            0    9 3 12:29:56 2009
.qt              DH           0   10 3 16:18:02 2009
GvR              D            0    6 3 18:58:20 2009
LPI              D            0   10 3 12:25:33 2009
UML              D            0   18 3 22:22:05 2009
smb: \> quit
```

ЗАДАНИЯ

- Измените права доступа к ресурсу `BUX` так, чтобы он был доступен для записи, а также к нему имели доступ все члены группы `users`.
- Настройте ресурс `BUX` так, чтобы новые файлы, размещаемые в нем, принадлежали группе `users`, а права доступа к ним устанавливались бы 660.

Монтирование файловых ресурсов SMB

Команда `mount.cifs` предназначена для монтирования сетевых файловых ресурсов SMB. Эта программа реализована как демон, контролирующий события, происходящие во время монтирования и использования файловых ресурсов (пример 25.11).

Пример 25.11. Монтирование SMB-ресурса

```
# mount -t cifs //NOTE/PUB /mnt/disk -o username=guest,password=''
# ls /mnt/disk/
linux.tar
```

В этом примере был смонтирован гостевой файловый ресурс `PUB` с компьютера `NOTE`. При монтировании были использованы следующие опции:

- ☐ `username` — устанавливает пользователя, от имени которого осуществляется доступ к ресурсу;
- ☐ `password` — пароль этого пользователя.

Для получения статуса SAMBA следует использовать программу `smbstatus`.

ЗАДАНИЕ

Имеется разделяемый файловый ресурс SMB, доступ к которому должен производиться от имени конкретного пользователя с некоторым паролем. Нужно сделать так, чтобы при монтировании не требовалось вводить ни имя пользователя, ни его пароль. В командной строке пароль вводиться не должен.

Использование сетевых принтеров

Для предоставления возможностей сетевой печати с помощью SAMBA следует установить разновидность системы печати. Это можно сделать с помощью директивы глобальной конфигурации `printing`. При использовании CUPS следует установить также параметр `printcap name = cups`.

В `smb.conf` предусмотрена специальная секция `[printers]`, определяющая установки сетевой печати. В секции `[printers]` должна быть директива `printable = yes`, а директива `path` должна указывать путь к каталогу спула печати (пример 25.12).

Пример 25.12. Установки сетевой печати

```
[printers]
comment = "Public printer"
path = /var/spool/samba
printeable = yes
guest ok = yes
```

При существовании клиентских станций под управлением Windows 2000/XP в разделе `[printers]` удобно установить параметр `use client driver = yes`. Это позволит использовать драйвер принтера клиентской ОС.

Система CUPS способна осуществлять специальный тип обработки потока печати RAW. Он позволяет пользоваться протоколом печати без дополнительной его обработки фильтром печати. Если используется режим RAW, то в файлах `/etc/cups/mime.convs` и `/etc/cups/mime.types` должны присутствовать строки, приведенные в примерах 25.13 и 25.14.

Пример 25.13. Настройка в `/etc/cups/mime.convs`

```
application/octet-stream    application/vnd.cups-raw      0          -
```

Пример 25.14. Настройка в /etc/cups/mime.types

```
application/octet-stream
```

ЗАДАНИЯ

- Создайте разделяемый ресурс SAMBA для принтера.
- Разрешите доступ к этому ресурсу лишь для группы `users`.

Запуск SAMBA в режиме PDC

Наличие в сети PDC означает, что, единожды пройдя процесс аутентификации на PDC, пользователь автоматически получает доступ ко всем компьютерам — членам домена (Single Sign On, SSO).

Для обеспечения возможности входа клиентов в сеанс в домене должен существовать сервис `NETLOGON`. При запуске SAMBA в режиме PDC уровень безопасности должен быть установлен `security = user` (пример 25.15).

Пример 25.15. Настройка для SAMBA в режиме PDC

```
[globals]
netbios name = pdclass
workgroup = class
security = user
passdb backend = tdbsam
os level = 64
preferred master = yes
local master = yes
domain master = yes
domain logons = yes
logon path = \\%N\profiles\%U
logon home = \\%N\win95\%U
logon drive = N
logon script = welcome.cmd

[netlogon]
path = /home/samba/netlogon
read only = yes
guest ok = yes
browseable = no
```

Директива `netbios name` задает имя сервера SAMBA в сети NetBIOS. Имя домена определяется настройкой `workgroup`. Уровень важности сервера, который указывается им в широковещательных пакетах, генерируемых для участия в процессе выборов мастер-браузера сети, определяется настройкой `os level`. По умолчанию используется значение 20. Уровень 64 гарантирует победу в выборах.

Настройки `preferred master = yes` и `domain master = yes` также предназначены для победы SAMBA в процессе выборов мастер-браузера сети. Причем вторая директива предназначена для гарантирования того, что сервер станет браузером всего домена. Локальный браузер в сети NetBIOS предназначен для обеспечения просмотра локальной подсети. Параметр `local master = yes` гарантирует победу в выборах в подсети. Параметр `domain logons = yes` предназначен для обеспечения входа в домен клиентов Windows 95/98.

Спецификация местоположения перемещаемых профилей пользователей (roaming profiles) для клиентов Windows 2000/XP определяется параметром `logon path`. В примере используется макрос `%N` — NetBIOS имя машины, `%U` — имя пользователя.

Рабочие станции Windows NT способны использовать параметр `logon drive`, задающий букву диска для отображения на нее сетевого ресурса — домашнего каталога пользователя. Директива `logon script` задает имя сценария, срабатывающего при входе пользователя в сеанс.

При запуске SAMBA в режиме PDC необходимо обеспечить разделяемый ресурс `NETLOGON`, предназначенный для хранения сценариев, срабатывающих при входе пользователей в сеанс. Директива `logon script` указывает имя файла сценария относительно каталога, определенного в `NETLOGON` (пример 25.16). Файл сценария, срабатывающего при входе пользователя в сеанс, должен содержать переводы строки в стиле MS-DOS: CR/LF.

Пример 25.16. Файл скрипта `logon script`

```
NET USE Z: \\PDCLASS\Pub
```

Разделяемый ресурс `PROFILES` позволяет хранить индивидуальные настройки рабочего окружения пользователей — профили. Каждый пользователь должен иметь в этом разделяемом ресурсе индивидуальный подкаталог. Этот ресурс должен быть доступен для чтения всем, а для записи — его пользователю (пример 25.17).

Пример 25.17. Ресурс PROFILES

```
[profiles]
  path = /home/samba/profiles
  read only = no
  create mask = 0600
  directory mask = 0700
```

ЗАДАНИЯ

- Создайте ресурсы NETLOGON и PROFILES.
- Создайте `logon script`, отображающий ресурс PUB на диск P: для `user1`.
- Запустите SAMBA в режиме PDC.

Сервер SAMBA в режиме члена домена

Машины — члены домена автоматически предоставляют свои ресурсы зарегистрировавшимся в домене пользователям. При этом повторной регистрации на данной машине не требуется. Для запуска сервера SAMBA в режиме члена домена необходимо установить уровень безопасности `security = domain`.

На сервере SAMBA, являющемся контроллером домена, должна быть создана *доверенная учетная запись* (trusted account) для машины — члена домена. Перед ручным созданием доверенной учетной записи необходимо создать специальную UNIX учетную запись для машины — члена домена. При этом имя UNIX учетной записи должно заканчиваться на символ доллара (пример 25.18).

Пример 25.18. Создание учетной записи для машины — члена домена

```
# useradd -M -d /home/samba -s /bin/false -g nobody 'membra$'
# usermod -L 'membra$'
```

После создания UNIX учетной записи сервера — члена домена она должна быть зарегистрирована в SAMBA. Для этого используется команда `smbpasswd` с опциями `-a` (добавление) и `-m` (доверенная учетная запись машины) — пример 25.19.

Пример 25.19. Регистрация учетной записи сервера — члена домена

```
# smbpasswd -a -m memba
```


Машина — член домена должна после этого войти в домен (пример 25.20).

Пример 25.20. Регистрация в домене

```
# net join MEMBER -S PDCLASS -U root
```

Администратор `root` должен быть зарегистрирован в SAMBA заранее.

Задания

- Настройте компьютер с SAMBA и сконфигурируйте его, как член домена.
- Зарегистрируйте на PDC нужные учетные записи.
- Зарегистрируйте сервер — член домена.

Программа *winbind*

Пакет `winbind`, входящий в комплект SAMBA, позволяет пользователям домена Windows NT или Active Directory единообразно аутентифицироваться и использовать ресурсы UNIX/Linux-машин. При использовании `winbind` UNIX/Linux-машина работает с NT-пользователями и группами так, как будто это обычные UNIX-пользователи и группы.

Демон `winbindd` запускается с помощью сценария в каталоге `/etc/init.d`. При этом требуется, чтобы `smbd` и `nmbd` уже были запущены. Работоспособность демона `winbindd` проверяется с помощью команды `wbinfo` (пример 25.21).

Пример 25.21. Проверка работоспособности `winbindd` (фрагмент)

```
$ wbinfo -u
wdom\Administrator
wdom\Guest
```

В примере 25.21 получен список пользователей домена `wdom`. Для вывода информации о группах используйте опцию `-g`.

Задача демона `winbindd` заключается в предоставлении ответов на запросы клиентов NSS (Name Service Switch) и PAM (Pluggable Authentication Module). Пример такого клиента — программа `ntlm_auth`. Она возвращает 0, если пользователь аутентифицирован, и 1 в противном случае. Распространенное применение программы `ntlm_auth` — в прокси-сервере Squid для аутентификации пользователей, обращающихся к Web-ресурсам.

Имеется модуль PAM `pam_winbind.so`, позволяющий проводить аутентификацию обычных UNIX/Linux-пользователей в домене Windows. Сервис `winbindd` и библиотека `libnss_winbind.so` предоставляют необходимую инфраструктуру для использования домена Windows, как источника информации о пользователях, группах и узлах сети. В примере 25.22 приведен фрагмент файла `/etc/nsswitch.conf`, позволяющий сделать это.

Пример 25.22. Фрагмент `/etc/nsswitch.conf` для использования `winbindd`

```
passwd:      files winbindd
group:       files winbindd
hosts:       files dns wins
```

При таких настройках информация о пользователях и группах будет взята из локальных файлов, а затем из домена Windows. Информация об узлах сети будет извлечена из локальных файлов, затем DNS, затем с помощью WINS (Windows Internet Name Service).

Задания

- Проверьте наличие `winbind` в системе.
- При наличии домена Windows подключитесь к нему с помощью `windind` и проверьте его работу с помощью `wbinfo`.

Глава 26



DNS-сервер BIND

Служба доменных имен DNS — одна из наиболее важных служб, на которых держится современный Интернет. В этой главе вы познакомитесь с ее устройством, научитесь конфигурировать DNS и проверять ее работоспособность.

Организация DNS

Аббревиатура DNS расшифровывается как *служба доменных имен* (Domain Name Service). Она пришла на смену файлу, устанавливающему соответствия имен компьютеров их адресам, который использовался на заре ARPANET (сейчас он называется `/etc/hosts`). DNS во многом решила проблему конфликтов имен. Конфликт возникает в случае, когда узлы с разными адресами претендуют на одно и то же имя в сети.

В системе DNS все пространство имен разбито на домены, имена узлов в которых должны быть уникальны. В то же время в разных доменах могут использоваться одинаковые имена узлов. Имена `www.bamboo.be` и `www.bamboo.de` различны, т. к. находятся в разных доменах — `bamboo.be` и `bamboo.de`.

Сервер DNS BIND (Berkeley Internet Name Daemon) является де-факто стандартным сервером DNS в Интернете. Он предоставляет требуемую инфраструктуру для организации иерархической распределенной базы данных DNS, хранящей информацию о соответствии имен хостов их IP-адресам. База данных DNS является распределенной, поскольку информация о различных доменах может находиться на разных серверах DNS. Нет ни одного сервера DNS, который бы имел полную информацию о соответствии всех имен компьютеров в сети их адресам. Пространство имен в DNS организовано в виде дерева, корень которого имеет имя "точка" (`.`). Этот домен (точка) содержит

поддомены, называемые *доменами верхнего уровня* (TLDs, Top Level Domains). Они условно подразделяются на три типа:

- ❑ исходные (generic) — com, gov, mil, edu, net, org, int;
- ❑ территориальные (geographic) — ru, su, be, uk и т. д.;
- ❑ дополнительные, например — biz, news и info.

Корневые домены содержат поддомены, называемые доменами второго уровня. Так, например, `class.edu` — это домен второго уровня, являющийся поддоменом корневого домена `edu`. В доменах второго уровня также могут быть созданы поддомены. Так, может быть создан поддомен `tjumen.class.edu`.

Множество имен компьютеров, входящих в домен, за исключением имен всех компьютеров, входящих во все поддомены этого домена, называется *зоной*. Каждая зона должна обслуживаться сервером имен, который сопоставляет имена компьютеров их адресам. При этом сервер имен обычно может обслуживать сразу несколько зон, но каждая зона должна обладать хотя бы одним сервером имен.

Обычно для обеспечения надежности системы DNS и деятельности сервера имен, сопоставляющего имена компьютеров их адресам в некоторой зоне, устанавливаются подчиненные (slave) серверы. Сервер имен зоны, являющийся первоисточником информации о соответствии имен компьютеров их адресам, называется мастером (master). Информация о зонах находится на мастер-сервере в специальных текстовых файлах, содержащих так называемые записи о ресурсах (Resource Records, RR).

Подчиненные (slave) серверы получают информацию о зоне с мастер-сервера (master) с помощью копирования файлов ресурсов. Операция копирования файлов ресурсов называется *трансфером зоны* (zone transfer).

Информация, получаемая от мастер-серверов и подчиненных серверов, считается клиентской частью DNS одинаково достоверной (хотя это далеко не всегда соответствует истине). Поэтому ответы, получаемые от мастер-серверов и подчиненных серверов, называются авторитетными.

В процессе разрешения имени, производимого сервером по поручению клиента, сервер может обращаться с запросами к корневым серверам и серверам поддоменов, рекурсивно углубляясь в дерево доменных имен. При этом в кэше сервера обычно фиксируются результаты выполнения промежуточных запросов.

Серверы, способные углубляться в дерево доменных имен, в процессе получения ответа на запрос клиента, работают в рекурсивном режиме и кэшируют промежуточные результаты запросов. Серверы, обслуживающие корневые

домены (root servers), работают в нерекурсивном режиме и не кэшируют информацию в целях защиты от DNS-подлогов (DNS spoofing).

Время, в течение которого DNS-сервер может пользоваться кэшированной информацией, называется *временем жизни записи* (time to live, TTL).

В процессе обслуживания запроса клиента в кэше какого-либо сервера DNS может оказаться информация, требуемая клиенту. Уровень доверия к такой информации ниже, чем к авторитетной, т. к. она может устареть и не соответствовать истине. Поэтому такие ответы называются неавторитетными, и в них содержится дополнительная информация клиенту о том, где (вероятно) может быть получена авторитетная информация по данному запросу.

Сервер DNS может быть запущен в рекурсивном режиме, кэшируя информацию, необходимую для повторных запросов клиентов, но при этом не содержать описания зон (исключая зоны для соответствия имени localhost адресу 127.0.0.1 и наоборот). Такой сервер называется *исключительно кэширующим* (cache only). Кэширующие серверы используются для минимизации нагрузки на авторитетные серверы.

Процедура регистрации доменного имени заключается в выборе доменного имени и фиксации адресов серверов, являющихся авторитетными для регистрируемой зоны. Считается, что для надежности каждая зона должна обслуживаться не менее, чем двумя серверами (мастером и подчиненным), находящимися в разных IP-сетях.

Ранее доменами второго уровня централизованно управлял Network Information Center (NIC), на сегодняшний момент управление децентрализовано, и домен второго уровня можно зарегистрировать у коммерческих регистраторов. В территориальных доменах регистрация обычно осуществляется национальными регистраторами. Для создания поддомена домена второго уровня достаточно всего лишь обратиться к администратору данного домена.

Процесс передачи прав авторитетности для какой-либо зоны другому серверу имен называется *делегированием*. Так, например, право вести записи о ресурсах поддомена может быть делегировано серверу DNS, запущенному на одной из машин, принадлежащих поддомену.

В системе BIND основой является демон `named`, отвечающий на запросы о преобразовании имен машин в IP-адреса и наоборот.

Файлы конфигурации BIND позволяют настраивать работу демона `named` так, чтобы он являлся мастер-сервером имен зоны и был одновременно подчиненным сервером для других зон.

Разделяют два типа зон:

- ❑ зоны прямого отображения, в которых имена узлов сопоставляются их IP-адресам;
- ❑ зоны обратного отображения, сопоставляющие IP-адреса узлов именам узлов.

Зоны обратного отображения всегда являются поддоменами специального домена `in-addr.arpa`. Имена зон обратного отображения формируются из IP-адреса сети, переписанного наоборот. Так, отображение IP-адресов узлов сети `192.168.0.0` на имена узлов задается в зоне `0.168.192.in-addr.arpa`. Такая непривычная форма записи имен зон обратного отображения позволяет производить и прямое, и обратное отображение с помощью одних и тех же программ.

Обычно на DNS-сервере имеется специальный файл указателей на корневые серверы, что позволяет серверу обращаться к ним.

Задания

- Какие преимущества и недостатки имеются у кэширующих серверов?
- Почему мастер-серверы и подчиненные серверы должны быть в разных IP-сетях?

Конфигурационный файл BIND

В настоящее время широко используются восьмая и девятая версии сервера BIND. Конфигурационный файл для этих версий BIND — `/etc/named.conf` (пример 26.1). Ранее использовалась четвертая версия BIND, файл конфигурации которой `/etc/named.boot` значительно отличался синтаксисом.

Пример 26.1. Файл конфигурации `/etc/named.conf` для BIND 9

```
options {
    pid-file "/var/run/named/named.pid";
    directory "/var/named";
};

// Зона указателей на корневые серверы
zone "." {
    type hint;
    file "db.root";
};
```

```
// Зона отображения имени localhost на 127.0.0.1
zone "localhost" {
    type master;
    file "db.localhost";
};

// Зона обратного отображения 127.0.0.1 на localhost
zone "0.0.127.in-addr.arpa" {
    type master;
    file "db.127.0.0.1";
};

// Зона прямого отображения для class.edu
zone "class.edu" {
    type master;
    file "db.class.edu";
};

// Закомментированный пример организации подчиненного сервера
/* zone "classic.edu" {
    type slave;
    masters { 192.168.1.254; };
    file "slave/bak.classic.edu";
}; */

// Зона обратного отображения для сети 192.168.2
zone "2.168.192.in-addr.arpa" {
    type master;
    file "db.192.168.2";
};
```

В файле конфигурации `named.conf` могут применяться три типа комментариев:

- ❑ `#` — однострочные комментарии в стиле Shell;
- ❑ `//` — однострочные комментарии в стиле C;
- ❑ `/* ... */` — многострочные комментарии в стиле C.

Раздел `options`, приведенный в примере, устанавливает местонахождение файла, в котором будет находиться PID процесса (директива `pid-file`), а также указывает имя каталога, в котором находятся файлы описания зон (директива `directory`).

Разделы `zone` указывают зоны, известные серверу имен. Для каждой зоны в этих разделах фиксируется ее имя, например, `class.edu` — для зоны пря-

мого отображения или `2.168.192.in-addr.arpa` — для зоны обратного отображения. Среди них могут быть зоны четырех типов (директива `type`):

- ☐ `hint` — зона указателей на корневые серверы;
- ☐ `master` — зона, для которой данный сервер является мастер-сервером;
- ☐ `slave` — для данной зоны этот сервер является подчиненным;
- ☐ `stub` — особый тип зоны, применяемый для делегирования.

Для зон типа `master` достаточно указать лишь файл описания зоны с помощью директивы `file`. В этот файл администратор DNS должен внести корректные записи о соответствии имен узлов зоны их IP-адресам, а также другую дополнительную информацию, описанную далее.

Для зон типов `slave` и `stub` должен быть указан IP-адрес мастер-сервера, что делается с помощью директивы `masters`. В приведенном примере для зоны `classic.edu` сервер мог бы быть подчиненным (в примере использован многострочный комментарий). Файлы описания зон, для которых сервер является подчиненным, принято сохранять в специальном подкаталоге — в примере `slave/`. Эти файлы будут получены сервером при успешном *зонном трансфере* (*zone transfer*).

Зона указателей на корневые серверы типа `hint` нужна для начального заполнения кэша сервера BIND IP-адресами корневых серверов. Файл `db.root` (другое его распространенное название — `named.ca`) содержит имена и IP-адреса корневых серверов. Он может быть получен по адресу: **`ftp://ftp.rc.internic.net`**.

В пакете BIND 9 поставляется удобная команда, позволяющая проверять синтаксис файла `named.conf` — `named-checkconf`.

Задания

- Предположим, что требуется установить сервер BIND для сети `172.16.0.0`, имена узлов в ней — `comp1.cheremiska.ru`, `comp2.cheremiska.ru` и т. д. до `comp254.cheremiska.ru`. IP-адреса узлов соответствуют числу в их имени. Какие зоны должны быть описаны в `named.conf`?
- Как бы вы назвали файлы описания зон в таком случае?
- Создайте соответствующий заданным выше требованиям файл конфигурации и проверьте его синтаксис с помощью команды `named-checkconf`.

Записи о ресурсах DNS

Файлы описания зон конкретизируют ресурсы DNS, составляющие данную зону. Так, например, в зонах прямого отображения основную массу записей составляют соответствия имен узлов их IP-адресам.

В файле конфигурации `named.conf` содержатся указатели на файлы описания зон. Эти указатели можно найти по ключевому слову `zone`. То есть имя зоны, заданное в файле `named.conf`, связано с файлом описания зоны директивой `file`.

Если в файле описания зоны требуется изменить текущее имя зоны (заданное в `named.conf`), то можно использовать директиву `$ORIGIN` (пример 26.2).

Пример 26.2. Директива `$ORIGIN`

```
$ORIGIN class.edu.
```

В примере 26.2 текущего имени зоны установлено в `class.edu`.

Для обращения к текущему имени зоны в BIND используется специальный метасимвол `@`. То есть имя зоны, заданное в `named.conf`, либо заданное директивой `$ORIGIN`, может быть получено с помощью `@`.

Большинство записей в файлах описания зон является *записями о ресурсах* (resource record, RR). Эти записи составляют базу данных зоны и должны начинаться с первой позиции строки. Единственный символ, который применяется для установки комментариев — это точка с запятой (`;`). Части строк, находящиеся правее этого символа, не интерпретируются сервером BIND.

Записи RR имеют пять полей: имя, время жизни, класс, тип и поле данных. Поле времени жизни часто опускают, используя TTL по умолчанию. Поле данных также может быть разделено на несколько подполей. В поле имени указывают наименование ресурса. Поле "класс" практически всегда имеет значение `IN` (Internet), а поле "тип" указывает тип записи о ресурсах.

Директива `$TTL` — это время жизни ресурсов в кэше по умолчанию (пример 26.3).

Пример 26.3. Директива `$TTL`

```
$TTL 1D
```

В данном случае указано время жизни TTL, равное суткам.

Файлы описания зон чаще всего содержат следующие типы записей о ресурсах:

- ❑ SOA (Start of Authority) — определяет зону полномочий сервера. После этой записи всегда следуют записи RR, описывающие зону, за которую данный сервер отвечает, как мастер-сервер;

- ❑ NS (Name Server) — для перечисления DNS-серверов данной зоны;
- ❑ A (Address) — устанавливают соответствие между именем хоста и его IP-адресом. Эти записи необходимы для обеспечения прямого разрешения имен (из имени в IP-адрес) и встречаются только в файлах зон прямого отображения;
- ❑ PTR (Pointer) — предназначены для установления обратного соответствия IP-адресов DNS-именам хостов. Эти записи участвуют в процедуре обратного разрешения (из адреса в имя) и встречаются в файлах зон обратного отображения;
- ❑ CNAME (Canonical Name) — используются для установки псевдонимов имен хостов. Они позволяют задать для хоста несколько альтернативных имен;
- ❑ MX (Mail Exchanger) — определяют серверы электронной почты (SMTP-агенты), обслуживающие данную зону или узел.

Далее представлен пример 26.4 описания зоны прямого соответствия для зоны localhost, т. е. приведено содержание файла db.localhost.

Пример 26.4. Зона прямого отображения для localhost

```
$TTL 86400
@ IN SOA localhost. root.localhost. (
    2009122001 ; s/n
    1W          ; refresh
    1D          ; retry
    2W          ; expire
    1H          ; negative TTL
)
IN NS localhost.
IN A 127.0.0.1
```

Запись SOA имеет несколько полей данных (они начинаются сразу после надписи SOA), которые могут быть записаны в одну строку, однако традиционно они размещаются вертикально и экранируются круглыми скобками.

Поля данных SOA:

- ❑ имя мастер-сервера, обслуживающего данную зону. Для зоны localhost, естественно, имя сервера тоже localhost;
- ❑ почтовый адрес администратора сервера, в котором вместо привычного символа @ — разделителя имени пользователя и доменной части используется символ "точка", т. к. @ имеет особый смысл;

- ❑ последовательный номер (serial number, *s/n*), необходимый для подчиненных серверов. Они ориентируются по нему, не производились ли изменения в зоне. Если да, то должен быть произведен трансфер зоны. Этот номер всегда растет;
- ❑ периодичность опроса подчиненных серверов мастер-сервером на предмет проверки, не изменился ли последовательный номер (*refresh*). То есть не надо ли произвести трансфер зоны. Может быть указан в секундах. В данном примере: 1w — одна неделя;
- ❑ периодичность повторного опроса в случае неудачи (*retry*). Если мастер-сервер в момент очередной проверки оказался не доступен, то с заданной периодичностью проводятся повторные попытки проверки;
- ❑ время устаревания данных зоны на подчиненных серверах (*expire*). Если мастер-сервер так и не ответил за указанное время, то по прошествии его информация о зоне на подчиненном сервере стирается;
- ❑ время жизни записей о неудачах разрешения имен (*negative TTL*).

Далее приведена запись *NS*, в первом поле которой необходимо указать имя зоны. Эта запись задает имя сервера имен для данной зоны. Их может быть несколько (обычно не более десяти), в таком случае в файле описания зоны указывают сразу несколько записей типа *NS* — по одной для каждого сервера имен.

В примере 26.4 первое поле записи *NS* пустое. В таком случае *BIND* автоматически подставляет значение первого поля предыдущей записи *RR*. В нашем случае предыдущая запись — это запись *SOA*. В ее первом поле указан символ @, т. е. текущее имя зоны (в данном случае — *localhost*). Поэтому запись *NS* здесь, фактически, выглядит так, как показано в примере 26.5.

Пример 26.5. Запись *NS* для зоны *localhost*

```
localhost. IN NS localhost.
```

Обратите особое внимание на точки, завершающие имена. Если этих точек не будет, то *BIND* автоматически подставит к именам узлов имя текущего домена, что в данном случае приведет к ошибке.

Последняя запись в данном файле типа *A* устанавливает прямое соответствие имени узла его IP-адресу. В примере 26.6 показано, как выглядит запись *A* для *localhost* с учетом автоматической подстановки в первом поле.

Пример 26.6. Запись *A* для *localhost*

```
localhost. IN A 127.0.0.1
```

То есть имени `localhost` соответствует адрес `127.0.0.1`.

Далее приведен пример 26.7 зоны обратного отображения для адреса `127.0.0.1`, т. е. распечатано содержимое файла `db.127.0.0.1`.

Пример 26.7. Зона обратного отображения для `localhost`

```
$TTL 86400
@ IN SOA localhost. root.localhost. (
    2009122001 ; s/n
    1W          ; refresh
    1D          ; retry
    2W          ; expire
    1H          ; negative TTL
)
IN NS  localhost.
1 IN PTR localhost.
```

В примере 26.7 новой является лишь запись типа `PTR`, задающая обратное соответствие. В ее первом поле находится последняя цифра IP-адреса, к которой добавляется имя текущего домена. Здесь оно — `0.0.127.in-addr.arpa`. То есть последняя запись выглядит так, как показано в примере 26.8.

Пример 26.8. Запись `PTR` для `localhost`

```
1.0.0.127.in-addr.arpa. IN PTR localhost.
```

Рассмотрим теперь содержимое файла `db.class.edu` для зоны прямого соответствия `class.edu` (пример 26.9).

Пример 26.9. Зона прямого отображения `class.edu`

```
$TTL 86400
@ IN SOA ns1 root.ns1 (
    2009122001 ; s/n
    1W          ; refresh
    8H          ; retry
    1D          ; expire
    1H          ; negative TTL
)
IN NS  ns1.class.edu.
```

```

IN NS ns.provider.net.
IN MX 10 relay.class.edu.
ns1    IN A 192.168.2.253
relay  IN A 192.168.2.254
murr   IN A 192.168.2.252
www    IN CNAME murr
$GENERATE 1-251 ws$ A 192.168.2.$

```

В этом примере указаны два сервера имен, обслуживающих данную зону. Мастер-сервер указан в записи SOA — ns1.class.edu. Подчиненный сервер принадлежит провайдеру, что является распространенной практикой.

Запись mx указывает на почтовый сервер, обслуживающий данную зону — relay.class.edu. Число 10 в этой записи указывает уровень приоритета сервера, который актуален при наличии нескольких почтовых серверов для зоны. Чем меньше число в данном случае, тем выше приоритет почтового сервера.

Записи типа A задают прямое отображение имен узлов на их IP-адреса. Для одного из узлов — murr.class.edu с помощью записи типа CNAME указан псевдоним www.

Директива \$GENERATE исключительно удобна для массового сопоставления единообразных имен узлов их IP-адресам. Запись с этой директивой заменяет множество записей типа A. То же самое можно было сделать, 251 раз написав строки RR показанного в примере 26.10 вида.

Пример 26.10. Генерируемые директивой \$GENERATE записи RR

```

ws1 IN A 192.168.2.1
ws2 IN A 192.168.2.2
ws3 IN A 192.168.2.3
...
ws251 IN A 192.168.2.251

```

Пример 26.11 — это содержимое файла db.192.168.2 зоны обратного соответствия для сети 192.168.2.0 (зона 2.168.192.in-addr.arpa).

Пример 26.11. Зона обратного отображения для class.edu

```

$TTL 86400
@ IN SOA ns1.class.edu. root.ns1.class.edu. (
    2009122001 ; s/n
    1W        ; refresh

```

```

8H      ; retry
1D      ; expire
1H      ; negative TTL
)
IN NS ns1.class.edu.
254 IN PTR relay.class.edu.
253 IN PTR ns1.class.edu.
252 IN PTR murr.class.edu.
$GENERATE 1-251 $ PTR ws$.class.edu.

```

Легко заметить, что в этом файле преобладают записи обратного отображения типа PTR. Обратное отображение для множества названных единообразно узлов также осуществляется с помощью директивы \$GENERATE.

Последний файл описания ресурсов, используемый в этом примере, — это файл указателей на корневые серверы db.root (пример 26.12).

Пример 26.12. Зона указателей на корневые серверы (фрагмент)

```

; formerly NS.INTERNIC.NET
;
.                3600000    IN    NS      A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET. 3600000    A      198.41.0.4
;
; formerly NS1.ISI.EDU
;
.                3600000    NS      B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET. 3600000    A      126.9.0.107

```

В этом файле указаны серверы имен корневого домена и их IP-адреса.

В заключение этого раздела следует отметить, что в состав BIND 9 включена удобная утилита для проверки файлов описания зон — `named-checkzone`. С ее помощью можно отыскивать синтаксические ошибки в описаниях зон.

Задания

- Составьте необходимые файлы описания зон для реализации задания из предыдущего раздела (`cheremiska.ru`).
- Проверьте синтаксис этих файлов с помощью `named-checkzone`.
- Получите обновленный файл `db.root` с официального сайта.

Запуск DNS-сервера BIND

Сервер BIND запускается с помощью сценария в каталоге `/etc/init.d` (пример 26.13).

Пример 26.13. Запуск BIND

```
# /etc/init.d/named start
* Starting named...          [ ok ]
```

Запуск демона `named` осуществляется от имени непривилегированного пользователя (обычно пользователя `named`), что достигается с помощью опции `-u` (пример 26.14).

Пример 26.14. Запуск BIND от имени непривилегированного пользователя

```
# ps -ef | grep named
named    25224 25225  0 01:20 ?                00:00:00 /usr/sbin/named -u named
root     25231 12151  0 01:22 pts/1          00:00:00 grep named
```

Сервер BIND использует в работе 53-е порты TCP и UDP (пример 26.15).

Пример 26.15. Порты, открытые BIND

```
# netstat -tau
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 localhost:domain        *:*                     LISTEN
udp        0      0 localhost:domain        *:*
```

Управление сервером BIND может осуществляться либо с помощью передачи ему сигналов, либо для BIND 8 — командой `ndc`, а в BIND 9 — `rndc`. Команда `rndc` имеет собственный конфигурационный файл `/etc/rndc.conf`. При этом в файлах конфигурации `/etc/named.conf` и `/etc/rndc.conf` должен быть одинаковый ключ шифрования. Файл `/etc/rndc.conf` легко сгенерировать командой `rndc-confgen`. Генерируемая конфигурация содержит строки, которые должны быть включены в `/etc/rndc.conf`, а также в `/etc/named.conf`.

Задания

- Запустите BIND и проверьте его статус.
- Определите, на какие сигналы и как реагирует сервер BIND.

- Остановите сервер BIND с помощью передачи ему соответствующего сигнала.
- Удастся ли теперь стартовать BIND с помощью сценария в `/etc/init.d`?
- Если сервер не стартует — изучите проблему и запустите сервер.
- Изучите работу `rndc-confgen`. Включите полученный текст в файлы `/etc/named.conf` (удалив комментарии) и `/etc/rndc.conf`.
- Опробуйте `rndc`.

Тестирование сервера DNS

Для проверки работы сервера DNS предназначены три основные команды:

- ☐ `nslookup` — интерактивная утилита, обладающая возможностями тестировать как сервер DNS, так и его клиента — библиотеку `resolver` (пример 26.16);
- ☐ `host` — команда для тестирования клиента;
- ☐ `dig` — команда для тестирования сервера (Domain Information Grouper).

Пример 26.16. Использование `nslookup` из командной строки

```
$ nslookup www.nissan.com
Server:          217.76.182.3
Address:         217.76.182.3#53

Non-authoritative answer:
www.nissan.com canonical name = nissan.com.
Name:   nissan.com
Address: 137.118.26.70
```

В примере 26.16 получен IP-адрес требуемого узла сети.

Далее приведен пример 26.17 работы в интерактивном режиме `nslookup`.

Пример 26.17. Интерактивный режим `nslookup`

```
$ nslookup
> www.toyota.jp
Server:          217.76.182.3
Address:         217.76.182.3#53

Non-authoritative answer:
```



```
Name:    www.toyota.jp
Address: 203.181.119.185
```

```
$ host www.kia.com
www.kia.com has address 207.178.243.78
```

Опция `-l` команды `host` выводит полный список узлов в данной зоне при обращении запроса к авторитетному серверу (если это не запрещено на сервере).

В примере 26.18 представлена команда `dig` и результат ее работы.

Пример 26.18. Команда `dig`

```
$ dig www.hiundai.com
; <<>> DiG 9.5.0 <<>> www.hiundai.com
;; global options: printcmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 60345
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 0
;; QUESTION SECTION:
;www.hiundai.com.                IN      A
;; ANSWER SECTION:
www.hiundai.com.                3600    IN      A      209.216.204.91
;; AUTHORITY SECTION:
hiundai.com.                    172799  IN      NS      ns2.domain-name-host.com.
hiundai.com.                    172799  IN      NS      ns3.domain-name-host.com.
hiundai.com.                    172799  IN      NS      ns1.domain-name-host.com.
;; Query time: 681 msec
;; SERVER: 217.76.183.34#53(217.76.183.34)
;; MSG SIZE rcvd: 120
```

Для проверки обратного соответствия следует использовать опцию `-x` команды `dig` (пример 26.19).

Пример 26.19. Проверка обратного соответствия с помощью `dig`

```
$ dig -x 207.178.243.78
; <<>> DiG 9.5.0 <<>> -x 207.178.243.78
;; global options: printcmd
;; Got answer:
```

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 42802
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0
;; QUESTION SECTION:
78.243.178.207.in-addr.arpa.      IN      PTR
;; ANSWER SECTION:
78.243.178.207.in-addr.arpa. 86400 IN    PTR      www.kia.com.
;; AUTHORITY SECTION:
243.178.207.in-addr.arpa. 86400 IN      NS      nsl.anides.com.
;; Query time: 542 msec
;; SERVER: 217.76.182.3#53(217.76.182.3)
;; MSG SIZE rcvd: 95
```

При использовании команды `dig` для тестирования сервера, отличного от заданного по умолчанию в `/etc/resolv.conf`, следует установить его IP-адрес в командной строке `dig` после символа `@`.

ЗАДАНИЯ

- Проверьте правильность работы сервера DNS с помощью `nslookup`, `host` и `dig`.
- Проверьте трансфер зоны.
- Можно ли проверить трансфер зоны средствами `dig`?

Делегирование

Создание поддомена сводится к организации дочернего домена, а делегирование — это передача ответственности за дочерний домен. На первом этапе настраивается сервер, авторитативный для дочернего домена, а затем производится делегирование.

В примере 26.20 для филиала в Омске создан поддомен `omsk.class.edu` и установлен собственный DNS-сервер `lessy.omsk.class.edu` (пример 26.21).

Пример 26.20. Настройки мастер-сервера поддомена

```
zone "omsk.class.edu" {
    type master;
    file "db.omsk.class.edu";
};
```

Пример 26.21. Записи о ресурсах в db.omsk.class.edu

```
$TTL 1d
@ IN SOA lessy.omsk.class.edu. postmaster.omsk.class.edu. (
    2009122901
    3H
    1H
    1W
    1H )
    IN NS      lessy.omsk.class.edu.
lessy IN      A      192.168.149.2
```

Для указания того, что поддомен делегирован в файле описания зоны родительского DNS-сервера, необходимо указать наличие серверов имен для этого поддомена. В примере 26.22 приведен фрагмент файла db.class.edu родительского сервера DNS.

Пример 26.22. Делегирование

```
omsk IN      NS      lessy.omsk.class.edu.
lessy.omsk.class.edu. IN      A      192.168.149.2
```

Запись ns в примере 26.22 делегирует ответственность за зону `omsk.class.edu` серверу `lessy.omsk.class.edu`.

Задания

- Создайте зону `zone.class.edu` с любыми доменными именами.
- Делегируйте ответственность за зону `zone.class.edu`.

Журналы DNS

Настройки журналирования определяются директивой `logging` файла конфигурации `named.conf`. Информация, сохраняемая при работе DNS в журналах, определяется двумя сущностями: каналом и категорией. Канал определяет, какие и куда будут записаны данные. Данные могут записываться в файл, либо они будут передаваться демону `syslog`, либо в "черную дыру" `null`.

При записи в файл следует указать его имя, а при передаче информации для журналирования службе `syslog` необходимо указать поток (facility) `syslog`. По умолчанию используется `local0`.

Категория определяет тип информации, которая должна попадать в журналы, а также указывает каналы, в которые информация этой категории должна заноситься.

Каналы позволяют фильтровать сообщения по уровню важности (severity). Имеются следующие уровни важности:

- ☐ `critical` — события, приводящие к полной неработоспособности;
- ☐ `error` — ошибки, при которых сервер не функционирует нормально;
- ☐ `warning` — предупреждения о некритичных ошибках;
- ☐ `notice` — сообщения, не являющиеся информацией об ошибках;
- ☐ `info` — сообщения информационного толка;
- ☐ `debug [level]` — отладка;
- ☐ `dynamic` — уровень настраивается с помощью `rndc trace`.

Категории сообщений:

- ☐ `config` — сообщения о настройках;
- ☐ `lame-server` — обнаружение некорректного делегирования;
- ☐ `notify` — информация о рассылке служебных уведомлений об изменениях зон;
- ☐ `queries` — регистрация запросов;
- ☐ `security` — сообщения, связанные с безопасностью;
- ☐ `update` — сообщения о динамических обновлениях;
- ☐ `xfer-in` — сообщения о получении зон;
- ☐ `xfer-out` — сообщения о передаче зон;
- ☐ `general` — сообщения общего плана;
- ☐ `client` — обработка запросов;
- ☐ `database` — сообщения внутренней базы данных BIND;
- ☐ `dnssec` — обработка подписанных транзакций DNSSEC;
- ☐ `network` — сетевые операции.

Для настройки журналирования в файле конфигурации `/etc/named.conf` необходимо указать директиву `logging` (пример 26.23).

Пример 26.23. Настройка журналов

```
logging {  
    channel qry {  
        file "/var/log/queries.log";
```

```
        severity info;
        print-time yes;
    };
channel common {
    syslog daemon;
    severity dynamic;
    print-category yes;
    print-severity yes;
};
category queries { qry; };
category default { common; };
};
```

В примере 26.23 определены два канала журналирования: `qry` для записи информации о запросах и `common` для всех остальных сообщений.

В канале `common` уровень важности сообщений может быть отрегулирован с помощью команды `rndc trace`, т. к. уровень важности задан — `dynamic`. Для сообщений, записываемых в канал `common`, будут указаны категория и уровень важности, а для запросов будет дополнительно фиксироваться время запроса.

Сообщения о запросах будут записываться в файл самим BIND, а все остальные сообщения передаваться службой `syslog` от имени (facility) `daemon`. Сообщения о запросах будут записываться лишь после разрешения командой `rndc querylog`, т. к. объем журнала о запросах растет стремительно.

Задания

- Настройте журналирование по примеру 26.23.
- Выполните любой запрос к серверу. Записывается ли информация в журнал запросов? Как включить запись запросов?
- Какие сообщения записываются в канал `common`?
- Настройте канал `common` так, чтобы сообщения передавались службой `syslog` от имени `local0` (facility), и создайте журнал `syslog` для записи этих сообщений.



Глава 27

Сервер DHCP

Сервер DHCP позволяет передавать клиентским машинам информацию о сетевых настройках. В этой главе вы узнаете, как настроить сервер DHCP.

Работа DHCP

Администрирование сети становится исключительно неудобным, если количество рабочих станций со статическими IP-адресами приближается к нескольким десяткам. Даже в небольших сетях гораздо удобнее автоматически настраивать сетевые интерфейсы и передавать клиентам настройки DNS, маршрутизатора по умолчанию, сервера точного времени и т. п. Протокол DHCP является усовершенствованием протокола BOOTP (Bootstrap protocol).

Последовательность взаимодействия DHCP-сервера и клиента такова:

1. Клиент посылает широковещательный запрос DHCPDISCOVER.
2. Сервер отвечает на запрос пакетом DHCPOFFER, в котором клиенту предлагаются сетевые настройки.
3. Все полученные ответы DHCPOFFER анализируются клиентом, затем клиент посылает выбранному DHCP-серверу запрос DHCPREQUEST.
4. DHCP-сервер подтверждает получение запроса и гарантирует предложенные настройки, высылая подтверждение DHCPACK.

Сервер DHCP предоставляет клиентам сетевые настройки на ограниченное время, называемое *lease time*. По истечении этого срока клиент может запросить продления действия настроек.

Сервер DHCP также предоставляет возможность зафиксировать за определенным клиентом его сетевые настройки так, чтобы он получал их всегда. Это делается с помощью привязки по MAC-адресу клиента.

Обычные клиентские запросы включают:

☐ сетевые настройки:

- IP-адрес и сетевая маска;
- IP-адрес маршрутизатора;
- доменное имя компьютера;
- настройки DNS;

☐ настройки приложений:

- сервер печати;
- сервер времени;
- сервер загрузки TFTP (Trivial File Transfer Protocol);
- сервер WINS

и т. д.

Имеются два обычных пути предоставления клиентам информации сервером DHCP:

- ☐ на основе MAC-адреса. В таком случае можно обеспечить постоянство параметров конфигурации, предоставляемых клиенту;
- ☐ с помощью пула адресов. При таком подходе клиент получает IP-адрес динамически по принципу обычной очереди "первый пришел — первый обслужен". Когда клиент не работает в течение заданного периода времени, его конфигурация освобождается и может быть использована сервером для настройки другого клиента.

В распространенных дистрибутивах GNU/Linux всегда имеются два разных пакета: один содержит клиентскую часть (обычно называется `dhclient`), другой — серверную. Исполняемый файл сервера DHCP — `dhcpd`.

Задания

- Определите версию сервера DHCP, установленного в вашей системе.
- Как называется пакет с клиентской частью DHCP, установленный в системе?

Настройка сервера DHCP

Основной конфигурационный файл, задающий поведение сервера `dhcpd`, — это `/etc/dhcp.conf`. В примере 27.1 приводится несложная конфигурация сервера DHCP.

Пример 27.1. Простая конфигурация /etc/dhcpd.conf

```
# Global parameters
default-lease-time 600;
max-lease-time 7200;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.0.255;
option routers 192.168.0.254;
option domain-name-servers 192.168.0.1, 192.168.0.2;
option domain-name "class.edu";

# Dynamic pool
subnet 192.168.1.0 netmask 255.255.255.0 {
    range 192.168.1.10 192.168.1.100;
    range 192.168.1.150 192.168.1.200;
}
```

Здесь использованы следующие параметры настроек сервера DHCP:

- ☐ `default-lease-time` — время аренды (lease time) IP-адреса по умолчанию;
- ☐ `max-lease-time` — максимальное время аренды;
- ☐ `option subnet-mask` — маска подсети;
- ☐ `option broadcast-address` — широковещательный адрес;
- ☐ `option routers` — маршрутизаторы;
- ☐ `option domain-name-servers` — серверы DNS;
- ☐ `option domain-name` — доменное имя;
- ☐ `subnet` — секция настроек пула адресов:
 - `range` — диапазон адресов для предоставления клиентам.

Сервер DHCP запускается с помощью скрипта в /etc/init.d (пример 27.2).

Пример 27.2. Запуск сервера DHCP

```
# /etc/init.d/dhcpd start
Starting dhcp server...
```

[ok]

ЗАДАНИЯ

- Определите настройки для DHCP-сервера: пул адресов, маршрутизатор, DNS и т. п.
- Создайте файл настроек для заданных настроек.
- Запустите сервер DHCP, проверьте, получают ли клиенты IP-адреса и другую информацию с сервера.

Глава 28



Web-сервер Apache

Сервер Apache, называемый `httpd`, сегодня является стандартным и наиболее распространенным в Интернете сервером HTTP. В этой главе рассматривается базовая конфигурация Apache и управление им.

Конфигурационный файл Apache

Apache — это целое семейство различных проектов, первым из которых был Web-сервер. Web-сервер представлен демоном `httpd`, поэтому проект Apache по разработке Web-сервера называется `httpd`.

Конфигурационные файлы Web-сервера `httpd` в GNU/Linux в различных дистрибутивах размещаются в разных каталогах: `/etc/httpd/conf/`, или `/etc/apache2`, или ином. Главный конфигурационный файл называется `httpd.conf`, причем часто выделенные для каких-либо групп настроек индивидуальные конфигурационные файлы подключаются к `httpd.conf` с помощью директивы `include`.

В целом, настройки `httpd` можно разделить на три основные категории:

- ☐ секция глобальных настроек;
- ☐ настройки для главного сервера;
- ☐ настройки для виртуальных узлов.

Кроме этих секций можно выделить настройки для обработки заданных типов файлов, MIME-расширений и спецификаций обработки национальных языков.

К глобальным настройкам сервера относятся те из них, которые влияют на работу сервера Apache в целом. Далее приводятся основные глобальные настройки:

- ☐ `ServerRoot` — базовый каталог установки, относительно которого в файловой системе GNU/Linux размещаются конфигурационные файлы Apache, а также некоторые файлы, необходимые в его работе;

- ❑ `PidFile` — файл, в который при запуске сервера Apache заносится PID главного процесса сервера;
- ❑ `LoadModule` — команда загрузить модуль расширения Apache, скомпилированный в виде разделяемой библиотеки;
- ❑ `Timeout` — количество секунд перед посылкой сигнала о потере соединения;
- ❑ `KeepAlive` — разрешение или запрет поддержки долгоживущих соединений, доступных в рамках протокола HTTP/1.1, позволяющих оставлять соединение в неразорванном состоянии после отправки сервером требуемой информации;
- ❑ `KeepAliveTimeout` — количество секунд ожидания следующего запроса, по прошествии которого долгоживущее соединение будет разорвано;
- ❑ `MaxKeepAliveRequests` — максимально возможное количество запросов, разрешенное для долгоживущих соединений;
- ❑ `StartServers` — количество дочерних процессов `httpd`, которые стартуют при запуске сервера Apache;
- ❑ `MinSpareServers` — минимальное количество ждущих соединения дочерних процессов `httpd` (при появлении соединений запускается столько копий `httpd`, чтобы количество ожидающих соединения дочерних процессов было не меньше этой величины);
- ❑ `MaxSpareServers` — максимальное количество ждущих соединения дочерних процессов `httpd` (лишние процессы останавливаются);
- ❑ `MaxClients` — ограничение на максимальное количество дочерних процессов `httpd`, определяющее возможное количество соединений;
- ❑ `MaxRequestsPerChild` — максимальное количество запросов, которое разрешено обслуживать одному дочернему процессу `httpd`, при достижении которого он гарантированно останавливается для исключения возможных утечек памяти.

Далее приведен пример 28.1 секции глобальных настроек файла `httpd.conf` (с сокращениями строк `LoadModule`).

Пример 28.1. Секция глобальных настроек `httpd.conf`

```
ServerRoot "/etc/httpd"
PidFile run/httpd.pid
Timeout 120
KeepAlive Off
MaxKeepAliveRequests 100
```

```
KeepAliveTimeout 15
<IfModule prefork.c>
StartServers      8
MinSpareServers   5
MaxSpareServers   20
ServerLimit       256
MaxClients        256
MaxRequestsPerChild 4000
</IfModule>
Listen 80
LoadModule auth_basic_module modules/mod_auth_basic.so
LoadModule auth_digest_module modules/mod_auth_digest.so
LoadModule authn_file_module modules/mod_authn_file.so
LoadModule authn_alias_module modules/mod_authn_alias.so
LoadModule authn_anon_module modules/mod_authn_anon.so
...
User apache
Group apache
```

Следует отметить, что модули могут быть не только подключены с помощью директивы `LoadModule`, но и установлены непосредственно в код Apache на стадии его компиляции и сборки. Для определения, какие модули были установлены в код на стадии компиляции, выполните команду `httpd -l` (пример 28.2).

Пример 28.2. Статически собранные с `httpd` модули

```
$ /usr/sbin/httpd -l
Compiled in modules:
  core.c
  prefork.c
  http_core.c
  mod_so.c
```

Обратите внимание на модуль `prefork.c` — это один из нескольких различных модулей MPM (Multi-Processing Modules), имеющихся для Apache v2. Эти модули ориентированы на разные операционные системы или на различные модели многозадачной обработки. Например, модуль `prefork.c` ориентирован на традиционную для UNIX модель порождения дочерних процессов, а `worker.c` — на работу с потоками.

Среди настроек главного сервера наиболее важны следующие:

- ☐ `Listen` — заставляет Apache слушать только указанный порт;
- ☐ `User` — пользователь, от имени которого будут запущены процессы `httpd`;
- ☐ `Group` — группа, от имени которой будут запущены дочерние процессы `httpd`;
- ☐ `DocumentRoot` — каталог размещения файлов Web-сайта;
- ☐ `UserDir` — имя каталога для размещения личных Web-страниц пользователей;
- ☐ `DirectoryIndex` — список имен файлов, который будет просмотрен при попытке доступа к корневому документу каталога;
- ☐ `AccessFileName` — имя файла, содержащего директивы управлением доступом к каталогу;
- ☐ `TypesConfig` — задает местонахождение файла конфигурации MIME;
- ☐ `DefaultType` — задает MIME-тип документа по умолчанию;
- ☐ `MimeMagicFile` — задает местонахождение файла магических чисел для определения MIME-типа документа;
- ☐ `HostnameLookups` — задает тип занесения информации о запросах клиентов либо по IP-адресу, либо по имени узла;
- ☐ `ErrorLog` — файл журнала, в который будут записываться любые ошибки (сообщения), встретившиеся в процессе работы Apache;
- ☐ `LogLevel` — уровень информативности журналирования;
- ☐ `LogFormat` — директива, определяющая формат вывода информации в журнал;
- ☐ `CustomLog` — указывает местонахождение файла журнала;
- ☐ `ServerSignature` — разрешает выводить в страницах, сгенерированных сервером Apache, информацию о версии, виртуальном узле и т. п.;
- ☐ `Alias` — устанавливает псевдонимы для путей доступа к каталогам сервера (длина строки псевдонима меньше, чем длина строки пути доступа);
- ☐ `ScriptAlias` — делает то же, что и `Alias`, но для каталогов с CGI-сценариями;
- ☐ `IndexOptions` — определяет тип индексирования каталогов;
- ☐ `AddIcon` — указывает файл значка для отображения в конкретных ситуациях, например, при выводе страницы с содержимым каталога разным типам файлов сопоставляются разные значки;
- ☐ `AddIconByEncoding` — задает значки для конкретных MIME-кодировок;

- ☐ AddIconByType — задает значки для определенных MIME-видов документов;
- ☐ DefaultIcon — значок по умолчанию;
- ☐ AddDescription — добавляет строку комментария к определенным типам документов (например, .gz - gzipped document);
- ☐ ReadmeName — задает имя файла README, добавляемого в конец списка файлов в каталоге (directory index);
- ☐ HeaderName — задает имя файла HEADER, добавляемого в начало списка файлов в каталоге (directory index);
- ☐ IndexIgnore — имена файлов, которые будут игнорироваться при индексации;
- ☐ AddEncoding — позволяет браузерам заданным образом обрабатывать информацию на лету (например, отображать содержимое сжатых gzip-файлов);
- ☐ AddLanguage — отображает заданные расширения имен файлов на определенный язык документа (например, AddLanguage ru .ru);
- ☐ AddCharset — задает отображение определенного расширения имен файлов на конкретную кодировку;
- ☐ LanguagePriority — порядок следования языков;
- ☐ AddType — задает отображение расширения имен файлов на тип MIME;
- ☐ AddHandler — задает тип обработки по расширению имен файлов;
- ☐ ErrorDocument — позволяет настраивать сообщения об ошибках;
- ☐ BrowserMatch — задает особенности вывода информации для браузеров.

В файле конфигурации `httpd.conf` часто встречаются условные директивы вида `<IfModule>`. С их помощью устанавливают такие настройки Apache, которые возможны, если присутствует данный модуль (пример 28.3).

Пример 28.3. Проверка наличия модуля

```
<IfModule mod_alias.c>
    Alias /doc /usr/share/doc
</IfModule>
```

В этом примере установлен псевдоним `doc` для каталога `/usr/share/doc`. Однако такой псевдоним будет установлен только в случае наличия модуля `mod_alias.c`. Имея такой псевдоним (при наличии соответствующих разрешений), можно будет обращаться с помощью Web-браузера к содержимому каталога `/usr/share/doc`, используя URL **`http://localhost/doc`**.

Другим видом условных директив является `<IfDefine>`. Эти директивы содержат в себе блоки настроек Apache, которые работают только при условии определения некоторого дополнительного параметра. Этот параметр может быть задан при помощи опции `-D` исполняемого файла `httpd`.

Часто Apache поставляется с весьма обширной документацией в HTML-формате. Например, в дистрибутивах Fedora традиционно документация на Apache доступна после запуска Apache по URL **`http://localhost/manual`**.

ЗАДАНИЯ

- Какая версия Apache установлена в вашей системе? Проверьте, как описаны настройки `StartServers` и `MinSpareServers` в вашей системе.
- Предположим, что сервер Apache скомпилирован и установлен непривилегированным пользователем в подкаталог `bin` его домашнего каталога. Каковы условия, необходимые для запуска Apache непривилегированным пользователем?

Контейнеры

Директивы настройки Apache находятся в файле конфигурации `httpd.conf` в одном из двух видов контекста:

- ☐ в контексте конфигурации сервера — глобальные настройки;
- ☐ в контексте контейнера.

Область действия директив, указанных в контексте контейнера, ограничивается объектом, связанным с данным контейнером. Например, для обеспечения возможности обращаться к каталогу `/usr/share/doc`, используя псевдоним `doc` (см. пример 28.3), можно создать контейнер, как в примере 28.4.

Пример 28.4. Контейнер Directory

```
<Directory /usr/share/doc>
    Options Indexes FollowSymLinks
    Order deny,allow
    Deny from all
    Allow from 127.0.0.1
</Directory>
```

Данный контейнер устанавливает права доступа к каталогу `/usr/share/doc`, получить содержимое которого можно, обратившись по URL **`http://localhost/doc`**, т. к. имеется псевдоним `doc`, указывающий на этот каталог.

Опция `Indexes` позволяет Apache автоматически выводить список файлов и подкаталогов, если в данном каталоге отсутствует файл `index.html`. Опция `FollowSymLinks` позволяет следовать символическим ссылкам в этом каталоге.

Директива `Order` предназначена для указания интерпретации следующих далее директив: `Deny`, устанавливающей запрет доступа к ресурсу, и `Allow`, разрешающей доступ. В этом примере к каталогу `/usr/share/doc` разрешен доступ посредством Apache только для клиентов с локального узла.

Контейнеры, используемые наиболее часто:

- ❑ `<Directory>` — применяются для указания директив, область действия которых ограничена заданным каталогом (при задании имени каталога с помощью регулярного выражения перед ним следует указывать знак `~`);
- ❑ `<DirectoryMatch>` — то же, что и предыдущий контейнер, но исходно предназначенный для указания имен каталогов с помощью регулярных выражений (знак `~` указывать не нужно);
- ❑ `<Files>` — для директив, связанных с заданными файлами (при задании имени файла с помощью регулярного выражения перед ним следует указывать знак `~`);
- ❑ `<FilesMatch>` — для директив, связанных с заданными файлами, указываемыми с помощью регулярных выражений (знак `~` указывать не нужно);
- ❑ `<Location>` — для указания директив, связанных с заданными URL;
- ❑ `<LocationMatch>` — указания настроек для URL, заданных регулярными выражениями;
- ❑ `<VirtualHost>` — особый вид контейнера, связанный с виртуальными узлами, т. е. с Web-сайтами, обслуживаемыми данным сервером, но имеющими отличные доменные имена или IP-адреса;
- ❑ `<Limit>` — контейнер, ограничивающий возможности использования заданных команд протокола HTTP (например, `GET` или `POST`);
- ❑ `<LimitExcept>` — контейнер, выполняющий обратное по смыслу действие по сравнению с предыдущим;
- ❑ `<IfDefine>` — контейнер, содержимое которого действительно только при указании некоторой дополнительной функциональности Apache (например, подключения SSL) с помощью опции `-D` команды `httpd`;
- ❑ `<IfModule>` — контейнер, содержимое которого действительно только при подключении указанного модуля.

Некоторые виды контейнеров допускают вложенность. Так, например, для обеспечения возможности проводить мониторинг активности Apache можно воспользоваться модулем `mod_status.c` (пример 28.5).

Пример 28.5. Модуль mod_status.c

```
<IfModule mod_status.c>
  <Location /server-status>
    SetHandler server-status
    Order deny,allow
    Deny from all
    allow from 127.0.0.1
  </Location>
</IfModule>
```

Контейнер `<IfModule>` здесь содержит контейнер `<Location>`, который позволяет в случае наличия модуля `mod_status.c` получить с локального узла информацию о состоянии сервера, обратившись по адресу **http://localhost/server-status**.

Директива `SetHandler`, использованная в этом примере, задает обработчик для всех файлов, предоставляемых сервером. В данном примере именно эта директива позволяет получить статус сервера с помощью модуля `mod_status.c`.

Среди программ пакета `httpd` обычно имеется удобная утилита `a2enmod`, автоматизирующая подключение модулей Apache.

ЗАДАНИЯ

- Проверьте ваш файл конфигурации Apache на наличие директив, позволяющих получить информацию о статусе сервера.
- Имеются ли в вашем конфигурационном файле Apache директивы, позволяющие использовать модуль `mod_info.c`? Как обратиться к его обработчику?
- С помощью каких директив в вашем файле конфигурации устанавливается запрет доступа к файлам, имена которых начинаются с `.ht`?

Запуск и управление Apache

Запустить или остановить Apache можно с помощью сценария в каталоге `/etc/init.d` (пример 28.6).

Пример 28.6. Запуск Apache

```
# /etc/init.d/httpd start
* Starting httpd... [ ok ]
# netstat -ta
```


Active Internet connections (servers and established)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	*:www	*:*	LISTEN

Обычно Apache настроен на прослушивание порта 80 TCP, как в этом случае.

Запустить сервер Apache можно также и с помощью специальной утилиты `apache2ctl` (пример 28.7).

Пример 28.7. Запуск с помощью `apache2ctl`

```
# /usr/sbin/apache2ctl start
/usr/sbin/apache2ctl start: httpd started
```

Эта же утилита позволяет остановить сервер Apache, а также проверить его статус и правильность конфигурации. Для проверки правильности конфигурации сервера необходимо выполнить команду, приведенную в примере 28.8.

Пример 28.8. Проверка синтаксиса конфигурации

```
$ /usr/sbin/apache2ctl configtest
Syntax OK
```

Задания

- Изучите команду `apache2ctl`. Как с ее помощью получить статус сервера?
- Проверьте правильность конфигурации Apache.
- Запустите сервер и проверьте его статус.
- Обратитесь к серверу с помощью браузера для проверки его работоспособности.

Личные Web-страницы

При необходимости разрешить пользователям поддерживать личные Web-страницы можно использовать модуль `mod_userdir.c`. Этот модуль позволяет обращаться к домашним каталогам пользователей.

Само по себе подключение модуля `mod_userdir.c` еще не обеспечивает доступ к личным Web-страницам. Требуется еще указать метод трансляции URL на имена каталогов (пример 28.9).

Пример 28.9. Каталог для личных Web-страниц

```
<IfModule mod_userdir.c>
    UserDir public_html
</IfModule>
```

В этом случае личные Web-страницы должны быть расположены в подкаталогах `public_html` домашних каталогов пользователей, а для обращения к ним надо будет использовать URL вида **`http://hostname/~username`**.

Помимо этого, следует также обеспечить требуемые права доступа к каталогу, содержащему личные Web-страницы. То есть, во-первых, каталоги должны быть доступны для чтения (например, права `755`), а во-вторых, разумно указать ограничения для доступа к личным страницам (пример 28.10).

Пример 28.10. Ограничения доступа к каталогам с личными страницами

```
<Directory /home/*/public_html>
    AllowOverride All
    Options MultiViews -Indexes Includes FollowSymLinks
    Order allow,deny
    Allow from all
</Directory>
```

В этом примере директива `AllowOverride All` позволяет изменять любые предыдущие настройки доступа к этим каталогам с помощью файлов `.htaccess`.

На каталоги, содержащие личные Web-страницы, здесь установлены следующие настройки по умолчанию с помощью директивы `Options`:

- ☐ `MultiViews` — опция, позволяющая серверу предоставлять клиентам запрашиваемую страницу в разных формах, например, в разной кодировке, что обеспечивается модулем `mod_negotiation.c`;
- ☐ `-Indexes` — запрет автоматического предоставления содержимого каталога при отсутствии файла `index.html`;
- ☐ `Includes` — разрешение использования SSI (Server Side Includes, включения на стороне сервера);
- ☐ `FollowSymLinks` — разрешение следовать символическим ссылкам.

Задания

- Зарегистрируйте пользователя с первичной группой `apache` (или иной — в зависимости от GID запуска демона `httpd` в вашей системе), имеющего домашний каталог. Создайте в его домашнем каталоге подкаталог `public_html` с группой владельцев `http`. На домашний каталог пользователя установите права 750, а на `public_html` — 2750 (установлен SGID).
- От имени только что зарегистрированного пользователя создайте файл `index.html` (или воспользуйтесь любым существующим в системе) в каталоге `public_html`. Получите доступ к нему с помощью браузера.
- Разместите в каталоге `public_html` несколько любых файлов и удалите `index.html`. Что надо сделать для получения доступа к ним?
- Разместите в каталоге `public_html` символические ссылки на файлы из домашнего каталога. Удастся ли пользоваться ими с помощью браузера?

Ограничение доступа к Web-ресурсу

Часто возникает необходимость предоставить доступ к ресурсу не всем пользователям, а только тем, кто прошел процедуру аутентификации. Естественно, для этого необходимо иметь механизм для накопления сведений о зарегистрированных пользователях системы.

Простейшим методом обеспечения этого является подключение модуля `mod_auth.c`.

Для аутентификации пользователей размещают специальный файл `.htaccess` в каталоге, к которому требуется ограничить доступ. В файле конфигурации Apache для этого должна присутствовать строка, приведенная в примере 28.11.

Пример 28.11. Указание имени файла управления доступом

```
AccessFileName .htaccess
```

В контейнере, связанном с защищаемым ресурсом, должно быть разрешение изменять правила аутентификации, что достигается с помощью директивы `AllowOverride AuthConfig`. Так, например, для ограничения доступа к каталогу `/var/www/html/secret` можно организовать в соответствии с примером 28.12.

Пример 28.12. Разрешение изменять правила аутентификации

```
<Directory /var/www/html/secret>
    Options Indexes FollowSymLinks
    AllowOverride AuthConfig
</Directory>
```

В этом примере наиважнейшей является настройка `AllowOverride AuthConfig`, т. к. она разрешает изменять правила аутентификации, указывая их в файле конфигурации `.htaccess`. Файл `.htaccess` должен быть размещен в защищаемом каталоге и должен быть доступен только для чтения пользователю, от имени которого запускается Apache.

При использовании простого метода аутентификации, основанного на наличии файла, в котором находятся имена авторизованных пользователей и их пароли, в файл `.htaccess` следует поместить строки, показанные в примере 28.13.

Пример 28.13. Защита ресурса паролем

```
AuthName "Unauthorized access is strongly prohibited!"
AuthType Basic
AuthUserFile /etc/httpd/htpasswd
Require valid-user
```

Директива `AuthName` позволяет задать сообщение, выводимое в диалоговом окне аутентификации при попытке доступа к этому ресурсу. Директива `AuthType` задает тип аутентификации с использованием передачи паролей, закодированных `uuencode`. Этот способ прост, но не безопасен. Файл, в котором хранятся имена пользователей и зашифрованные пароли, указывается с помощью директивы `AuthUserFile`.

Файл, содержащий имена авторизованных пользователей, может быть создан с помощью команды `htpasswd`. Эта утилита позволяет задавать пароли для пользователей защищаемого ресурса (пример 28.14).

Пример 28.14. Сохранение имен пользователей и паролей

```
# htpasswd -c /etc/httpd/htpasswd webuser
New password:
Re-type new password:
Adding password for user webuser
# htpasswd /etc/httpd/htpasswd weberus
New password:
Re-type new password:
Adding password for user weberus

# cat /etc/httpd/htpasswd
webuser:Zi6v4CjM7y5J2
weberus:83Ygmk1dHcLzM
```

Обратите внимание, что при первом вызове команды `htpasswd` была использована опция `-с` для создания файла паролей. При последующем использовании этой команды опцию `-с` устанавливать не надо, т. к. она сотрет файл паролей.

После выполнения указанных процедур правами на доступ к защищаемому ресурсу будут пользоваться только авторизованные пользователи, а при попытке получения доступа к защищаемому ресурсу будет выводиться приглашение ввести пароль и имя пользователя.

Задания

- Защитите личную страницу пользователя, зарегистрированного в предыдущем разделе, от неавторизованного доступа.
- Как сделать так, чтобы к личной странице пользователя имели доступ только члены группы `users`?

Виртуальные узлы

Виртуальный хостинг заключается в поддержке на одном реальном сервере множества Web-сайтов с разными доменными именами или IP-адресами. Возможность предоставления различных Web-страниц в зависимости от номера порта, к которому обращается клиент, также относится к виртуальному хостингу.

В рамках данной книги рассматривается лишь организация виртуальных узлов на основе их имен (Name Based Virtual Hosting).

Для организации виртуальных узлов, базирующихся на именах, необходимы две вещи. Во-первых, чтобы с IP-адресом узла, на котором запущен сервер Apache, были связаны желаемые доменные имена виртуальных узлов. Достигается это с помощью псевдонимов доменной системы имен DNS, которые реализуются записями `CNAME` в файлах описания зон DNS (см. главу 26, посвященную DNS).

Во-вторых, описания виртуальных узлов должны присутствовать в файле конфигурации Apache или же быть загружены из отдельного файла, что удобно в случае наличия большого количества виртуальных узлов.

Предположим, что с помощью `CNAME`-записей на сервере DNS для компьютера `comp1.class.edu` имеются два имени с одним IP-адресом (пример 28.15).

Пример 28.15. Записи `CNAME` в зоне прямого отображения DNS

```
www.class.edu. IN CNAME comp1.class.edu.  
web.class.edu. IN CNAME comp1.class.edu.
```

В примере 28.16 приведены директивы конфигурации Apache, позволяющие получать те или иные Web-страницы в зависимости от доменного имени целевого сайта.

Пример 28.16. Виртуальные хосты

```
NameVirtualHost *:80

<VirtualHost *:80>
    ServerName www.class.edu
    DocumentRoot /www/www.class.edu
</VirtualHost>

<VirtualHost *:80>
    ServerName web.class.edu
    DocumentRoot /www/web.class.edu
</VirtualHost>
```

Легко заметить, что в двух этих контейнерах с помощью директив `ServerName` задается имя виртуального узла, а директивы `DocumentRoot` задают местоположение Web-документов этих сайтов — каталоги `/www/www.class.edu` и `/www/web.class.edu`.

Директива `NameVirtualHost *:80` сообщает серверу, что виртуальные узлы не должны быть связаны с конкретным IP-адресом, но они будут доступны по порту 80. В контейнерах `<VirtualHost *:80>` звездочка указана, поскольку в данном случае используется виртуальный хостинг на основе имен.

Задания

- Предположим, что у вашей машины имеются два доменных имени, связанных с ее адресом: `www.machine.com` и `site.machine.com`. Установите соответствующие настройки в файле конфигурации Apache для организации виртуального хостинга на основе имен.
- Перенесите директивы конфигурации виртуального хостинга в отдельный файл.
- Настройте парольный доступ к `site.machine.com`.



Глава 29

Электронная почта

Несмотря на свой почтенный возраст, SMTP транспортный агент Sendmail является стандартным и широко распространенным почтовым сервером. В главе продемонстрированы вопросы, связанные с созданием его конфигурационного файла, настройкой и запуском сервера в простой конфигурации. В этой главе также рассмотрена базовая конфигурация другого популярного MTA — Postfix. Кроме того, в главе рассматриваются протоколы POP3 и IMAP.

Организация электронной почты

В процедуре передачи электронной почты участвуют три вида программ:

- ☐ пользовательский агент электронной почты MUA (Mail User Agent);
- ☐ транспортный агент MTA (Mail Transfer Agent);
- ☐ агент доставки MDA (Mail Delivery Agent).

Пользовательский агент MUA предназначен для непосредственного взаимодействия с пользователем в процессе создания писем, манипуляций ими и отправки почты с помощью MTA. Примерами MUA являются программы mutt и Novell Evolution.

Транспортный агент электронной почты MTA предназначен для приема почты от пользовательского агента MUA и передачи почты к месту назначения. В процессе передачи MTA взаимодействует с другими MTA, передавая почту от узла к узлу. Примерами MTA являются Sendmail, Exim и Postfix.

Доставочные агенты MDA предназначены для размещения почты в почтовые ящики. Как только почтовое сообщение появляется на узле назначения, доставочный агент передает письмо в ящик пользователя — адресата сообщения.

В качестве MDA в GNU/Linux может быть использована, например, программа `procmail`.

Почтовый ящик представляет собой файл или каталог. В этот файл помещаются входящие сообщения.

Каждое электронное письмо состоит из двух частей: заголовка и тела письма. Заголовок — часть электронного письма, содержащая информацию о том, кто отправил письмо, кто является получателем, о времени создания письма, о теме сообщения (Subject), а также служебные доставочные отметки.

Транспортные агенты MTA взаимодействуют при помощи протоколов передачи электронной почты. Наиболее часто встречается протокол передачи SMTP (Simple Mail Transfer Protocol) или его развитая версия ESMTP (Extended SMTP). До сих пор встречается также и применение устаревшего протокола UUCP (Unix-to-Unix copy). Реже используется протокол X.400.

В процессе передачи почты транспортными агентами к адресату особую роль играет система DNS. Она сообщает серверам MTA, куда следует направить электронную почту для данного адресата. Для этого предназначены специальные записи о ресурсах — `mx` (Mail Exchanger).

Записи `mx` в файлах описания зон DNS указывают транспортному агенту MTA, куда следует направить письмо. Запись `mx` может быть указана для отдельного хоста или для целого домена (пример 29.1).

Пример 29.1. Записи `mx` зоны прямого соответствия

```
class.edu. IN MX 5 mail.class.edu.  
class.edu. IN MX 10 relay.myisp.net.
```

В примере 29.1 приведены записи `mx` для основного почтового сервера домена `class.edu` — `mail.class.edu` и резервного сервера — `relay.myisp.net`. Сервер MTA, запущенный на узле `mail.class.edu`, имеет более высокий приоритет по сравнению с резервным сервером почты.

Процедура получения почты адресатом, обслуживаемым последним MTA в цепочке передачи, может быть выполнена несколькими способами.

- ❑ Если пользователь зарегистрирован на данном узле и имеет возможность использовать интерактивную оболочку, то он может просматривать содержимое своего почтового ящика с помощью таких команд, как `mail`, `mutt`, а также программ с графическим интерфейсом.
- ❑ Почта может быть получена с помощью протокола POP3. При этом пользователь инициирует соединение с сервером POP3, запущенным на узле,

где находятся полученные электронные сообщения, и получает почту. При таком способе доставки основное место хранения полученных сообщений — компьютер пользователя.

- ❑ При необходимости размещения почтового ящика пользователя не на стороне клиента (т. е. не на компьютере пользователя), а на сервере следует использовать протокол IMAP. При этом пользователь читает почту и управляет своим почтовым ящиком через сеть.

Почтовым ящиком можно управлять через сеть, используя специальные программы, предоставляющие Web-интерфейс для доступа к почтовому ящику. Эти программы обеспечивают взаимодействие с сервером IMAP, запущенным на узле, где размещаются почтовые ящики, и позволяют читать почту и управлять сообщениями, используя обычный Web-браузер.

Задания

- Используя запрос к серверу DNS, узнайте, какой MTA предназначен для получения адресованной вам почты.
- Какие преимущества и недостатки имеются в использовании систем POP3 и IMAP с точки зрения пользователя?
- Какую систему (POP3 или IMAP) следует предпочесть для доставки почты клиенту, связанному с сервером с помощью медленной модемной линии?

Файл конфигурации программы Sendmail

Программа Sendmail — это наиболее распространенный MTA. Несмотря на солидный возраст, эта программа до сих пор устанавливает стандарт работы MTA и обладает исключительно широкими возможностями.

Основной файл конфигурации программы Sendmail — `sendmail.cf`. В GNU/Linux файлы конфигурации Sendmail располагают в каталоге `/etc/mail`.

В файле конфигурации `sendmail.cf` определяются правила обработки почты. В нем могут быть определены классы (`classes`), макросы (`macros`) и опции (`options`):

- ❑ классы определяют общие фразы, которые используются для задания наборов правил обработки для сообщений определенного типа;
- ❑ макросы представляют собой наборы значений для упрощения набора длинных строк текста в файле конфигурации;
- ❑ опции представляют собой набор параметров для обеспечения работы Sendmail.

Каждая строка конфигурационного файла `sendmail.cf` начинается с одиночного символа, определяющего действие, которое должно быть выполнено. Строки комментариев начинаются с символа `#`.

Используются следующие виды строк конфигурации `sendmail.cf`:

- ☐ `c` — определяет класс;
- ☐ `d` — определяет макрос;
- ☐ `f` — определяет имя файла, содержащего классы;
- ☐ `h` — определяет поля заголовка в посланиях и действия с ними;
- ☐ `k` — определяет базы данных;
- ☐ `m` — определяет почтовые хосты;
- ☐ `o` — определяет опции `Sendmail`;
- ☐ `p` — определяет приоритеты;
- ☐ `r` — определяет наборы правил для обработки адресов;
- ☐ `s` — определяет группы наборов правил.

Например, определение имени почтового узла, называемого `Smart Host`, через который осуществляется ретрансляция всей исходящей почты, делается с помощью макроса в файле `sendmail.cf` (пример 29.2).

Пример 29.2. Настройка `Smart Host`

```
DSmailrelay.provider.ru
```

В этом примере в файл настроек `Sendmail` занесена настройка, вынуждающая его посылать для дальнейшей отправки всю исходящую почту на узел `mailrelay.provider.ru`. Это сделано с помощью определения макроса с именем `s`, которому присвоено значение `mailrelay.provider.ru`.

Имеются заранее определенные макросы. Далее приведены некоторые из них:

- ☐ `$a` — дата отправки из поля `Date:` заголовка письма;
- ☐ `$b` — текущая дата в формате `sendmail`;
- ☐ `$c` — число передач письма серверами МТА;
- ☐ `$d` — текущая дата в системном формате UNIX;
- ☐ `$f` — адрес отправителя;
- ☐ `$h` — идентификатор сообщения в очереди;
- ☐ `$j` — полное доменное имя узла;

- ❑ `$k` — доменная часть узла;
- ❑ `$m` — имя процесса демона `sendmail`;
- ❑ `$n` — PID процесса демона `sendmail`;
- ❑ `$p` — формат адреса отправителя по умолчанию;
- ❑ `$r` — протокол, используемый для приема сообщения;
- ❑ `$s` — имя хоста отправителя;
- ❑ `$t` — числовое выражение текущего времени;
- ❑ `$u` — получатель сообщения;
- ❑ `$v` — номер версии `Sendmail`;
- ❑ `$w` — имя хоста, на котором запущен данный экземпляр `Sendmail`;
- ❑ `$x` — полное имя отправителя;
- ❑ `$z` — рабочий каталог получателя;
- ❑ `$_` — проверенный адрес отправителя;
- ❑ `${bodytype}` — тип тела сообщения;
- ❑ `${client_addr}` — IP-адрес SMTP-клиента;
- ❑ `${client_name}` — имя хоста SMTP-клиента;
- ❑ `${client_port}` — номер TCP-порта SMTP-клиента;
- ❑ `${opMode}` — текущий режим работы (например, отладочный);
- ❑ `${deliveryMode}` — текущий режим доставки.

Классы с предназначены для группировки критериев, по которым будут выбираться пути обработки сообщения. Формат определения класса приведен в примере 29.3.

Пример 29.3. Определение класса

```
Ccphrase1 phrase2 ... phraseN
```

Здесь `c` — имя класса, а `phrase` — критерии, которые объединены в класс. Если имя класса многосимвольное, то его следует взять в фигурные скобки. Так, например, для определения класса локальной обработки сообщений с именем `w` (т. е. сообщений, которые не надо передавать через сеть, а достаточно поместить в почтовый ящик адресата на этой же машине) следует использовать директиву `cw` (пример 29.4).

Пример 29.4. Определение класса локальных имен

```
Cwlocalhost
```

Классы часто задают с помощью внешних текстовых файлов, а не в конфигурационном файле `sendmail.cf`. Для извлечения значений классов из файлов предназначены строки конфигурации `F`. Формат строк `F` показан в примере 29.5.

Пример 29.5. Определение класса с помощью файла

```
Fc filename
```

Здесь `c` — имя класса, а `filename` — имя файла, содержащего определение класса. Имя класса подчиняется тем же правилам, что и в строках `c`.

Предположим, что сервер SMTP может передавать (relay) почту для нескольких доменов, имена которых указывают в файле `/etc/mail/relay-domains` (пример 29.6).

Пример 29.6. Определение доменов, для которых разрешена передача почты

```
FR-o /etc/mail/relay-domains
```

Здесь класс `r`, задающий имена доменов, для которых разрешена передача почты, определяется с помощью строк в файле `/etc/mail/relay-domains`.

Помимо задания классов в текстовых файлах широко используются базы данных. Для извлечения значений класса из базы данных предназначены строки `k` (пример 29.7).

Пример 29.7. Определение класса в базе данных

```
Kmapname mapclass arguments
```

Здесь `mapname` — имя файла базы данных, `mapclass` — тип индексирования базы данных (например, `hash`), а `argument` — дополнительные аргументы.

Это применяется, например, для предотвращения рассылки несанкционированных сообщений с помощью файла `access.db`, разрешающего доступ к МТА (пример 29.8).

Пример 29.8. Включение файла `access.db`

```
Kaccess hash -T<TMPF> -o /etc/mail/access.db
```

В этом примере определена база данных `access`, находящаяся в файле `/etc/mail/access.db`.

Опции команды `sendmail`, задаваемые с помощью `-o` и `-O` в командной строке, могут быть заданы непосредственно в файле конфигурации `sendmail.cf`. Для этого используются директивы `O`. Формат директив `O`: `Oo value` или `O option=value`, где `o` — односимвольное имя опции, а `option` — длинное имя опции (пример 29.9).

Пример 29.9. Задание времени ожидания сетевого соединения

```
O Timeout.connect=1m
```

Большую часть файла конфигурации `sendmail.cf` занимают настройки процесса преобразования почтовых адресов, что делается с помощью определенных правил разборки `R` и их наборов `S`.

Задания

- Найдите в файле конфигурации `sendmail.cf` строку, задающую имя централизованного почтового узла (Smart Host). Закомментирована ли она?
- Найдите в файле конфигурации строку, позволяющую указать файл, в который заносят альтернативные имена узла, на котором запущен Sendmail.
- Найдите строку, в которой задается имя файла почтовых псевдонимов `alias`.

Файл конфигурации `sendmail.mc`

Просмотрев файл конфигурации `sendmail.cf` программы Sendmail, можно убедиться в его исключительной сложности. Случайно изменив одну из настроек, можно привести Sendmail в неработоспособное состояние.

В настоящее время в GNU/Linux используется заранее созданный разработчиками Sendmail набор шаблонов, написанных на языке `m4`, которые позволяют быстро и просто настраивать Sendmail, не зная деталей настроек файла `sendmail.cf`.

При использовании `m4` вначале создают файл макросов `sendmail.mc`, а затем с помощью препроцессора `m4` его преобразуют в файл `sendmail.cf` (пример 29.10).

Пример 29.10. Преобразование `sendmail.cf` в `sendmail.mc`

```
m4 sendmail.mc > sendmail.cf
```

Обычно в каталоге `/etc/mail` находится специально разработанный файл `Makefile`, позволяющий с помощью утилиты `make` выполнить это преобразование, а также перестроить при необходимости базы данных в этом каталоге. Для перестроения файла конфигурации и баз данных в каталоге `/etc/mail` надо выполнить команду, показанную в примере 29.11. Эта команда может обновить конфигурационные базы данных и даже пригодна для рестарта `Sendmail`.

Пример 29.11. Перестройка конфигурационных файлов

```
make all
```

Прежде чем использовать язык `m4`, следует убедиться, установлен ли в системе соответствующий пакет (он может называться по-разному). Проверить, установлен ли данный пакет в RPM-системах, можно командой, показанной в примере 29.12.

Пример 29.12. Проверка наличия пакета с макросами `m4` для `Sendmail`

```
$ rpm -qa | grep sendmail.*cf.*  
sendmail-cf-8.14.3-4
```

Файлы с макросами для генерации конфигурационного файла `sendmail.cf` находятся в отдельном каталоге, например, `/usr/share/sendmail-cf`.

Макропроцессор `m4` чувствителен к регистру задаваемых символов. В директивах `FEATURE` и `DEFINE` файла `sendmail.mc` используются круглые скобки, команды в которых должны быть взяты в кавычки, где открывающая кавычка — обратный апостроф (```), а закрывающая — обычный апостроф (`'`) — пример 29.13.

Пример 29.13. Директива `m4` — определение операционной системы

```
OSTYPE(`linux')dn1
```

В результате действия этой директивы к файлу `sendmail.cf` будет подключен файл `/usr/share/sendmail-cf/ostype/linux.m4`.

Каждый макрос и директива в файле макросов должны записываться с начала строки, в каждой строке может быть только одна директива или макрос. Строки, следующие после директивы `dn1`, пропускаются препроцессором `m4` и представляют собой комментарии. Первыми двумя директивами, устанавливаемыми в файле макросов, обычно являются `divert(-1)` и `divert(0)`.

Директива `divert(-1)` очищает буферы макросов от временных данных, не требующихся в результирующем файле. Директива `divert(0)` обозначает новый файл макросов.

Директива `include` предназначена для подключения к файлу макросов содержимого других файлов макросов (пример 29.14).

Пример 29.14. Подключение макросов m4 для Sendmail

```
include(`/usr/share/sendmail-cf/m4/cf.m4')dnl
```

Макрос `VERSIONID` устанавливает версию получаемого файла конфигурации.

Директива, показанная в примере 29.15, не имеет отношения к доменным именам DNS. Она устанавливает разновидность макросов для построения результирующего файла конфигурации. В GNU/Linux вы можете использовать настройки `generic`.

Пример 29.15. Определение стандартных макросов конфигурации

```
DOMAIN(`generic')dnl
```

Макрос `MAILER` предназначен для указания обработчика почтового трафика (пример 29.16). Эти макросы должны устанавливаться в файле конфигурации последними, причем локальная программа доставки всегда добавляется автоматически.

Пример 29.16. Установка программы локальной доставки

```
MAILER(`procmail')dnl
```

Макрос включения посылки писем по протоколу SMTP всегда должен находиться до макроса включения программы `procmail` (пример 29.17).

Пример 29.17. Порядок макросов MAILER

```
MAILER(`smtp')dnl  
MAILER(`procmail')dnl
```

Директива `define(MACRO, VALUE)` предназначена для определения макроса `MACRO` со значением `VALUE`. Например, для определения имени централизованного почтового узла (Smart Host) в `sendmail.mc` указывают настройку, показанную в примере 29.18.

Пример 29.18. Определение Smart Host

```
define(`SMART_HOST', `mailrelay.provider.ru')
```

Здесь определен макрос SMART_HOST со значением mailrelay.provider.ru.

В примере 29.19 приведен простой файл конфигурации sendmail.cf.

Пример 29.19. Файл sendmail.mc

```
divert(-1)dnl
include(`/usr/share/sendmail-cf/m4/cf.m4')dnl
VERSIONID(`Simple config')dnl
OSTYPE(`linux')dnl
define(`confDEF_USER_ID',`8:12')dnl
define(`confTO_CONNECT', `1m')dnl
define(`confDONT_PROBE_INTERFACES',true)dnl
define(`PROCMAIL_MAILER_PATH',`/usr/bin/procmail')dnl
define(`ALIAS_FILE', `/etc/aliases')dnl
FEATURE(`smrsh',`/usr/sbin/smrsh')dnl
FEATURE(always_add_domain)dnl
FEATURE(local_procmail,`,`procmail -t -Y -a $h -d $u')dnl
FEATURE(`accept_unresolvable_domains')dnl
MAILER(smtp)dnl
MAILER(procmail)dnl
```

В этом файле настроек задано следующее:

- ☐ настройка `define(`confDEF_USER_ID',`8:12')dnl` задает UID и GID пользователя, от имени которого работает Sendmail (пользователь mail);
- ☐ настройка `define(`confTO_CONNECT', `1m')dnl` задает время ожидания соединения — 1 минута;
- ☐ используя директиву `define(`confDONT_PROBE_INTERFACES',true)dnl`, можно запретить Sendmail добавление имен хоста, связанных с IP-адресами интерфейсов, в класс w, задающий имена узла;
- ☐ путь к программе procmail, являющейся почтовым фильтром и используемой в GNU/Linux для локальной доставки, задается с помощью директивы `define(`PROCMAIL_MAILER_PATH',`/usr/bin/procmail')dnl`;
- ☐ директива `define(`ALIAS_FILE', `/etc/aliases')dnl` задает имя файла почтовых псевдонимов;

- ❑ с помощью ограниченной оболочки Sendmail — smrsh осуществляется передача почты для обработки другим программам. Использование smrsh диктуется настройкой `FEATURE('smrsh', `/usr/sbin/smrsh')dnl`;
- ❑ настройка `FEATURE(always_add_domain)dnl` позволяет автоматически добавлять адресу отправителя доменное имя;
- ❑ опции фильтра procmail для локальной доставки задаются директивой `FEATURE(local_procmail, `', `procmail -t -Y -a $h -d $u')dnl`;
- ❑ разрешение на прием почты из доменов, имена узлов в которых не удастся проверить в DNS, задается настройкой `FEATURE('accept_unresolvable_domains')dnl`.

ЗАДАНИЯ

- Измените настройки в `sendmail.mc` так, чтобы для доставки исходящей почты использовался почтовый сервер вашего провайдера.
- Преобразуйте файл `sendmail.mc` в `sendmail.cf`. Видна ли информация, заданная директивой `VERSIONID`?
- Скопируйте файл `sendmail.cf` с именем `sendmail.cf.old`. Переделайте файл конфигурации `sendmail.cf`, удалив из `sendmail.mc` строку конфигурации `FEATURE(always_add_domain)dnl`. Получите после такого преобразования файла `sendmail.cf` список отличий старого и нового файлов конфигурации.

Запуск Sendmail

Sendmail автоматически запускается при переходе в многопользовательский режим, как stand-alone служба с помощью сценария в `/etc/init.d` (пример 29.20).

Пример 29.20. Запуск Sendmail

```
# /etc/init.d/sendmail start
* Starting sendmail...                [ ok ]
# netstat -ta
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address          State
tcp      0      0 localhost:smtp          *:*                      LISTEN
tcp      0      0 *:smtp                  *:*                      LISTEN
```

Sendmail прослушивает 25-й порт TCP — SMTP.

Далее приведены наиболее важные опции команды `sendmail`:

- ☐ `-bd` — работа в фоновом режиме в качестве демона;
- ☐ `-bD` — запуск в интерактивном режиме (`run foreground`);
- ☐ `-bi` — построение базы данных псевдонимов, аналогично команде `newaliases`;
- ☐ `-bm` — доставить почту обычным образом (по умолчанию);
- ☐ `-bp` — аналогично команде `mailq` отображает содержимое очереди сообщений;
- ☐ `-bs` — использовать для ввода и вывода протокол SMTP по RFC 821;
- ☐ `-bt` — запустить в тестовом режиме;
- ☐ `-bv` — режим проверки имен получателя и отправителя без доставки;
- ☐ `-c` — указывает файл конфигурации вместо `/etc/sendmail.cf`;
- ☐ `-d` — позволяет установить режим отладки;
- ☐ `-F` — устанавливает полное имя отправителя в поле `From:` сообщения;
- ☐ `-f` — устанавливает полное имя отправителя, задаваемое в поле `From:` сообщения, на имя пользователя, вошедшего в сеанс;
- ☐ `-h` — устанавливает количество передач посланий (`hop count`) через MTA;
- ☐ `-i` — указывает метку конца сообщения (по умолчанию — точка);
- ☐ `-N` — позволяет установить выдачу уведомления о состоянии сообщения;
- ☐ `-n` — при использовании этой опции Sendmail не пересылает почту и не использует почтовые псевдонимы;
- ☐ `-p` — устанавливает используемый почтовый протокол;
- ☐ `-q` — позволяет определить периодичность обработки сообщений в очереди. Если эта опция задана без параметров, то Sendmail один раз обрабатывает очередь и прекращает работу;
- ☐ `-R` — определяет, какое сообщение будет выдаваться при невозможности доставки сообщения получателю;
- ☐ `-t` — переводит Sendmail в режим чтения сообщений;
- ☐ `-U` — Sendmail не будет принимать сообщения от других MTA, а принимать будет только сообщения от MUA;
- ☐ `-v` — режим подробного информирования о действиях Sendmail;
- ☐ `-x` — позволяет определить файлы для сохранения отчетов;
- ☐ `-O` — задает опции Sendmail в формате `-O option=value`;

- ❑ `-o` — позволяет задавать другие опции Sendmail в формате `-o x value`, где `x` — имя односимвольной опции sendmail;
- ❑ `--` — конец списка опций, все, что следует далее в командной строке, интерпретируется, как адрес.

ЗАДАНИЯ

- Изучите сценарий запуска `/etc/init.d/sendmail` программы Sendmail и определите, с какими опциями он запускается в вашей системе.
- Запустите Sendmail так, чтобы периодичность проверки очереди сообщений составляла 30 минут.
- Очистите с помощью команды `sendmail` с соответствующей опцией очередь сообщений в режиме подробного отображения информации.

Почтовые псевдонимы

С помощью файла `.forward`, который должен находиться в домашнем каталоге пользователя, можно переадресовать входящую почту пользователя на другой адрес.

В файле `~/.forward` можно указывать сразу несколько адресов почты, на которые следует перенаправить копии сообщения. Каждый адрес должен быть в отдельной строке. Предположим, например, что в домашнем каталоге пользователя `user1` находится файл `.forward`, содержащий строки, приведенные в примере 29.21.

Пример 29.21. Файл `.forward`

```
lexa
ivanov
robertino@loretti.biz
```

В результате почта, посылаемая пользователю `user1`, будет также послана по адресам `ivanov`, `lexa` и `robertino@loretti.biz`.

При использовании перенаправления с помощью `~/.forward` сообщения будут одновременно направляться как по старому, так и по новому адресу.

С помощью файла `~/.forward` можно вести примитивный список рассылки.

Файл `/etc/aliases` позволяет создать псевдонимы для почтовых адресов пользователей сервера электронной почты. В каждой строке `/etc/aliases` устанавливается соответствие реального имени пользователя и его псевдонимов (пример 29.22).

Псевдонимы могут быть указаны в одном из форматов:

- ☐ формат для переадресовки: `name: alias1, alias2, ... aliasN;`
- ☐ формат для передачи почты программе: `name: |program;`
- ☐ формат для записи почты в файл: `name: file;`
- ☐ пересылка по списку адресов, содержащихся в файле: `name: : include:file.`

Пример 29.22. Файл `/etc/aliases` (фрагмент)

```
adm:                root
...
www:                webmaster

root:               ivanov
rassyl: sid@client.ru, bubuka@mail.ru, spamoed@net.domena.ru
```

В первом столбце указывают почтовый псевдоним, почта для которого переадресовывается на адрес, указанный в правом столбце. В последней строке приведен пример организации простого списка рассылки.

Имя пользователя, указываемое в левом поле, должно быть локальным. Не допускается указывать адреса с доменной частью. Однако псевдоним может указывать на адрес пользователя другого хоста.

Для создания файла базы данных псевдонимов `/etc/aliases.db` на основе `/etc/aliases` используют команду `newaliases` (пример 29.23) либо `sendmail -bi`.

Пример 29.23. Индексация почтовых псевдонимов

```
# newaliases
/etc/aliases: 14 aliases, longest 10 bytes, 152 bytes total
```

Задания

- Создайте в вашем домашнем каталоге файл `~/forward`, содержащий инструкцию по перенаправлению вашей почты пользователю `postmaster`.
- В файле `~/forward` поместите строку, которая позволит сохранять полученную почту в файле `~/mail_archive`.
- Настройте файл `/etc/aliases` так, чтобы был создан простой список рассылки `maillist`, получатели сообщений для которого должны быть указаны в файле `/etc/mail/maillist`.

Очередь почтовых сообщений

Каталог очереди почтовых сообщений, в котором находятся сообщения, ожидающие обработки, — `/var/spool/mqueue`. Время нахождения отправляемого сообщения в очереди зависит, естественно, от производительности компьютера и, в гораздо большей степени, от широты полосы пропускания линии и от доступности МТА назначения.

Для получения информации о состоянии очереди сообщений необходимо использовать команду `mailq` (пример 29.24) или `sendmail -bp`.

Пример 29.24. Проверка очереди сообщений

```
# mailq -v
-Queue ID- --Size-- ----Arrival Time---- -Sender/Recipient-----
477E7919EE      370 Sun Jan 31 14:31:29  emerald@localhost.class.edu
(Name service error for name=bobin.org type=MX:Host not found ,try again)
                                robin@bobin.org

-- 0 Kbytes in 1 Request.
```

В этом примере узел адресата не найден при обращении к DNS-серверу, поэтому сообщение находится в очереди и не отправлено.

Опция `-q` задает периодичность проверки почтовой очереди (пример 29.25).

Пример 29.25. Задание периодичности проверки очереди

```
# sendmail -qlh30m -bd
```

В примере 29.25 программа Sendmail запущена в фоновом режиме демона и будет проверять очередь с периодичностью в полтора часа.

Опцию `-q` можно использовать при задании обработки очереди по условию. При этом в сообщениях, находящихся в очереди, ищется определенная строка:

- ☐ `-qI` — поиск строки с заданным идентификатором;
- ☐ `-qR` — поиск по получателю сообщения;
- ☐ `-qS` — поиск по отправителю сообщения.

Задания

- Получите состояние очереди почтовых сообщений.
- Если в очереди на отправку имеются неотправленные сообщения, иницируйте их отправку.

Тестирование Sendmail

Для проверки работоспособности Sendmail можно использовать клиент telnet (пример 29.26).

Пример 29.26. Тестирование Sendmail с помощью telnet

```
$ telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 bblob.glott.com ESMTP Sendmail
HELO localhost
250 bblob.glott.com
MAIL FROM: uitzgen@localhost
250 Ok
RCPT TO: robin@bobin.org
250 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Bolshoj Privet!
.
250 Ok: queued as 477E7919EE
QUIT
221 Bye
Connection closed by foreign host.
```

В примере 29.26 воспроизведен процесс переговоров почтового клиента с сервером MTA Sendmail по протоколу SMTP.

Далее приведен список команд SMTP, которые часто используются при тестировании сервера:

- ❑ HELO — представление клиента SMTP-серверу;
- ❑ MAIL — определение отправителя сообщения;
- ❑ RCPT — определение получателя сообщения;
- ❑ DATA — определение начала сообщения;
- ❑ SEND — посылка сообщения непосредственно на терминал пользователя;
- ❑ RSET — сброс SMTP-соединения;

- ☐ VRFY — проверяет имя пользователя системы;
- ☐ EXPN — запрашивает список псевдонимов;
- ☐ HELP — запрашивает список команд;
- ☐ QUIT — выход из сеанса.

В примере после представления серверу HELO указано имя узла отправителя (клиентский узел). После фразы MAIL FROM: находится адрес отправителя. Фраза RCPT TO: указывает получателя. После команды DATA идет текст письма, заканчивающийся одиночной точкой в строке. Команда QUIT завершает соединение.

Исключительно удобным инструментом для проверки работоспособности электронной почты является программа mail. Несмотря на минималистский пользовательский интерфейс, она обладает достаточными возможностями для отладки сервера SMTP.

Для отправки сообщения с помощью программы mail из командной строки достаточно выполнить команду, показанную в примере 29.27.

Пример 29.27. Отправка почты с помощью mail

```
$ mail maximus@vergilius.it
Subject: Filioque
Cc:
Bcc:
End data with .
Текст послания.
.
```

Аргументом командной строки mail является почтовый адрес получателя письма. При этом пользователю будет предложено ввести тему письма после строки Subject:. Тему письма можно указать и после опции -s команды mail. Приглашения Cc: и Bcc: позволяют ввести адрес получателя копии сообщения и получателя скрытой копии сообщения. Текст письма завершается одиночной точкой. Точка должна быть единственным символом в строке.

Для проверки поступления почты команду mail вызывают без аргументов. В таком режиме команда mail работает интерактивно, отображая заголовки полученных сообщений и их номера. Прочитать сообщение можно, введя его номер.

Далее приведен список основных команд интерактивного режима работы `mail`:

- `?` — отображает краткую подсказку о командах;
- `-` — выводит предыдущее сообщение;
- `+` — выводит следующее сообщение;
- `n` — выводит следующее сообщение;
- `!` — позволяет выполнить команду `Shell`;
- `p` — печатает сообщение вместе с заголовком;
- `d` — удаляет сообщение или несколько сообщений;
- `u` — снимает пометку на удаление;
- `m` — позволяет создавать и отправлять почту;
- `r` — отправляет ответ;
- `s` — сохраняет сообщение вместе с заголовком;
- `w` — сохраняет сообщение без заголовка;
- `U` — помечает сообщения как непрочитанные;
- `x` — выход без изменения почтового ящика;
- `q` — выход с записью изменений.

Текст письма можно передать команде `mail` через поток ввода (пример 29.28).

Пример 29.28. Передача тела письма через стандартный поток ввода

```
ls *.txt | mail eji.class.edu
```

Здесь список файлов, выведенный командой `ls *.txt`, будет передан через конвейер команде `mail`, которая поместит его в тело сообщения и pošлет адресату.

Особенно полезно в процессе отладки работы сервера `Sendmail` бывает использовать команду `mail` с опцией `-v` (пример 29.29). Эта опция выводит сведения о процессе передачи почты почтовому серверу для дальнейшей передачи.

Пример 29.29. Режим подробного вывода информации `mail`

```
$ mail -v emerald@susel.class.edu < text.c
```

```
emerald... Connecting to [127.0.0.1] via relay...
```

```
220 susel.class.edu ESMTP Sendmail 8.14.3;Sun, 31 Jan 2010 20:52:47 +0500
```



```
>>> EHLO susel.class.edu
250-susel.class.edu Hello localhost.localdomain [127.0.0.1], pleased to
meet you
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-8BITMIME
250-SIZE
250-DSN
250-ETRN
250-AUTH DIGEST-MD5 CRAM-MD5
250-DELIVERBY
250 HELP
>>> MAIL From:<emerald@susel.class.edu> SIZE=16
AUTH=emerald@susel.class.edu
250 2.1.0 <emerald@susel.class.edu>... Sender ok
>>> RCPT To:<emerald@susel.class.edu>
>>> DATA
250 2.1.5 <emerald@susel.class.edu>... Recipient ok
354 Enter mail, end with "." on a line by itself
>>> .
250 2.0.0 i72EqlMg001038 Message accepted for delivery
emerald... Sent (i72EqlMg001038 Message accepted for delivery)
Closing connection to [127.0.0.1]
>>> QUIT
221 2.0.0 susel.class.edu closing connection
```

Фактически использование `-v` заставляет команду `mail` выводить процесс переговоров клиентской программы `mail` и сервера `sendmail`.

Задания

- Пошлите письмо пользователю `test` так, чтобы скрытая копия его была доставлена пользователю `root`. Адресат скрытой копии должен быть указан с помощью опции командной строки.
- Проверьте почту для `root`. Как ответить на скрытую копию?
- Отправьте письмо командой `mail` в информативном режиме работы.

Преимущества использования Postfix

Почтовый сервер Postfix изначально разрабатывался Витсом Венемой (Wietse Venema) для достижения высоких показателей производительности при условии обеспечения надежности и безопасности работы. Его безусловным

преимуществом перед Sendmail является простота и ясность конфигурирования. При этом Postfix обеспечивает практически полное перекрытие возможностей Sendmail.

Файлы конфигурации Postfix содержат множество комментариев, значительно облегчающих настройку. Для запуска Postfix в базовой конфигурации обычно достаточно поменять всего лишь три-четыре параметра.

В процессе работы Postfix запускает несколько процессов, управляемых процессом `master`. Эти процессы работают от имени непривилегированных пользователей.

Большинство внешних программ, рассчитанных на непосредственный запуск процесса `sendmail`, способно работать с Postfix, т. к. пакет Postfix предоставляет собственную программу `sendmail`.

В системе Postfix запускается главный процесс `master`, работающий с правами `root`. Он по необходимости запускает демоны Postfix, выполняющие всю работу почтовой системы. Поведение процесса `master` задается файлом `/etc/postfix/master.cf`. Процесс `master` играет роль супервизора, запуская и останавливая дочерние процессы, необходимые для работы Postfix. Он работает все время, пока функционирует Postfix.

Входящая почта помещается во входную очередь (`incoming queue`). Если клиент использует протокол SMTP, то за прием сообщения отвечает демон `smtpd`. Сообщение, принятое `smtpd`, подвергается предварительной проверке и передается демону `cleanup`.

Локальные сообщения, переданные командой `sendmail`, поставляющейся для совместимости с Sendmail, размещаются в очереди `maildrop` с помощью привилегированной команды `postdrop`.

Из очереди `maildrop` сообщения извлекаются демоном `pickup`. Он выполняет проверку сообщения для защиты Postfix и передает сообщение демону `cleanup`.

Все сообщения из локальных источников направляются непосредственно демону `cleanup`. К локальным источникам относятся:

- сообщения, связанные с работой самого Postfix;
- сообщения, исходящие от локального доставочного агента `local`;
- сообщения, возвращаемые источнику с помощью программы `bounce`.

Сервер `cleanup` при необходимости приводит в порядок заголовки сообщения. Он способен выполнять преобразование адресов, а также может осуществлять несложную фильтрацию сообщений по регулярным сообщениям.

Преобразование адресов выполняется с помощью сервера `trivial-rewrite`. В результате работы сервера `cleanup` сообщение помещается в очередь `incoming`, и сервер `cleanup` информирует менеджер очереди о появлении нового сообщения.

Менеджер `qmgr`, обслуживающий очередь сообщений, является главной составной частью системы доставки сообщений. Для доставки сообщений менеджер `qmgr` взаимодействует со следующими программами:

- ❑ `smtp` — клиент SMTP для передачи почты;
- ❑ `lmtp` — клиент LMTP локальной сетевой доставки (см. RFC 2033);
- ❑ `local` — локальный агент доставки, способный записывать сообщение в файл или передавать команде;
- ❑ `virtual` — агент для виртуального хостинга;
- ❑ `pipe` — для доставки сообщения внешней команде;
- ❑ `discard` — для отбрасывания писем или возврата их в очередь для повторения попыток доставки в будущем;
- ❑ `error` — для отбрасывания писем без попытки отправки.

Менеджер `qmgr` использует небольшую собственную очередь `active`, в которой размещаются обрабатываемые письма из очереди `incoming`. Имеется также очередь `deferred`, в которой менеджер очереди размещает сообщения, которые не могут быть доставлены.

Для обрабатываемых менеджером очереди писем с помощью программы `trivial-rewrite` проверяются адреса получателей. При этом может быть использована таблица `transport`, определяющая маршруты доставки сообщений, и таблица `relocated` для пользователей с изменившимися почтовыми адресами.

Клиент `smtp` пытается доставить исходящую почту с помощью протокола SMTP. Клиент `lmtp` доставляет локальную почту таким программам, как Cyrus IMAP, с помощью протокола LMTP. Агент доставки `local` способен доставлять почту в локальные почтовые ящики. Он может быть сконфигурирован для разнообразных форматов почтовых ящиков. Агент локальной доставки `virtual` предназначен для поддержки виртуального хостинга и способен записывать сообщения в локальные почтовые ящики в формате `mailbox` и `qmail maildir`.

Программа `anvil` обеспечивает контроль количества сессий и запросов к `smtpd`.

Серверы `bounce`, `defer` и `trace` обслуживают собственные журналы о доставке или не доставке для каждого сообщения. Сервер `trace` выводит ин-

формацию о том, как Postfix обрабатывает почту подобно `sendmail -bv` и `sendmail -v`.

Сервер `flush` позволяет производить срочную доставку почты с помощью ETRN (fast flush). Он ведет индивидуальные журналы для каждого отправляемого сообщения.

Специальный сервис поиска `proximap` обеспечивает поиск в различных конфигурационных таблицах без возможности их изменения.

Применяется для решения проблем, связанных с запуском Postfix с подменой корневого каталога (в `chrooted environment`), а также для консолидации множественных процедур поиска, требуемых разнообразным процессам Postfix.

Использование сервера `scache` значительно повышает производительность Postfix в отправке почты, т. к. он кэширует соединения с внешними SMTP-серверами, передавая их клиентским процессам `smtp`. Производительность повышается, в том числе и оттого, что при использовании кэширования соединений имеется возможность пропускать неотвечающие серверы, передавая клиентам `smtp` соединения с готовыми к работе серверами.

Программа `showq` предназначена для получения информации о статусе очередей Postfix, что необходимо для работы команд `mailq` и `postqueue`.

Сервер `spawn` запускает внешнюю для Postfix программу с клиентом, соединенным посредством сокета или канала с потоками ввода/вывода внешней программы.

Команда `postfix` предназначена для выполнения административных задач: старта, останова Postfix и т. п.

Системная команда `sendmail`, поставляющаяся с Postfix, предназначена для обеспечения совместимости с программами, обращающимися к Sendmail с помощью его прямого вызова.

Команда `mailq` позволяет выводить список сообщений в очереди. То же самое делает команда `postqueue -p`.

Команда `postqueue -f` "выталкивает" сообщения из очереди, пытаясь выполнить их немедленную доставку.

Для совместимости с Sendmail в пакете Postfix поставляется утилита `newaliases`, индексирующая файл `/etc/aliases`. Команда `postalias` выполняет те же функции, предоставляя также и дополнительные возможности.

Утилита `postcat` предназначена для просмотра файлов внутри очередей Postfix.

Команда `postconf` позволяет отображать и изменять параметры конфигурации Postfix (из `main.cf`). Также она выводит информацию о поддерживаемых в системе типах блокировок и типах таблиц поиска.

Программа `postdrop` вызывается Postfix-командой `sendmail` для помещения сообщения в очередь `maildrop`.

Программа `postkick` создает внутренние каналы связи для использования, например, в сценариях оболочки.

Команда `postlog` обеспечивает Postfix-совместимое журналирование.

Программа `postmap` предназначена для поддержки таблиц отображений. Например, для поддержки таблиц `canonical` и `virtual`. Она обеспечивает аналогичную функциональность с командой `makemap`.

Программа `postsuper` обслуживает очереди Postfix, удаляя ненужные временные файлы из них. При изменении структуры каталогов очередей эта программа перемещает файлы в очередях в правильные каталоги, соответствующие новой структуре. Она запускается при запуске Postfix.

ЗАДАНИЯ

- Удалите из системы пакет `Sendmail` и установите вместо него `Postfix`.
- Получите список файлов, установленных из пакета `Postfix`. Имеются ли в нем программы, упомянутые в данном разделе?

Конфигурационные файлы Postfix

По умолчанию конфигурационные файлы Postfix размещаются в каталоге `/etc/postfix`. Среди них имеются два главных файла конфигурации Postfix:

❑ `main.cf` — он определяет функциональность Postfix в целом;

❑ `master.cf` — настраивает работу процесса `master`.

В файле `/etc/postfix/main.cf` могут присутствовать десятки конфигурационных параметров, но чаще всего возникает необходимость изменить лишь несколько параметров. Большинство параметров Postfix имеет разумные установки по умолчанию, и поэтому они не требуют изменения.

Определение параметра Postfix производится в стиле переменных оболочки. Для извлечения значения из параметра Postfix указывается символ `$` (пример 29.30).

Параметр Postfix может быть использован до присвоения ему значения. То есть обращение `$variable` синтаксически верно даже до присвоения значения `variable`.

Пример 29.30. Определение параметров в `/etc/postfix/main.cf`

```
myhostname = host.domain.tld
myorigin = $myhostname
```

В примере 29.30 установлено значение параметра `myhostname`. Параметр `myorigin` установлен в значение `myhostname`.

В процессе работы Postfix активно использует разнообразные таблицы. Они могут храниться в разных форматах. В том числе: Berkeley DB, LDAP и SQL, а также другие типы баз данных (пример 29.31).

Пример 29.31. Определение базы данных

```
virtual_alias_maps = hash:/etc/postfix/virtual
```

При наличии такой установки в файле конфигурации `main.cf` Postfix будет использовать хешированную базу данных `/etc/postfix/virtual.db`.

Если любой параметр конфигурации Postfix был изменен, то Postfix следует информировать об изменении конфигурационного файла командой `postfix reload`.

Первый важный момент настройки Postfix заключается в определении идентичности системы и ее роли в сети. При этом требуется ответить на следующие вопросы:

- ☐ какое доменное имя использовать при отправке почты;
- ☐ для каких доменов будет выполняться прием почты;
- ☐ какие клиенты имеют право переправлять почту через данную систему;
- ☐ в каком направлении переправлять полученную почту;
- ☐ какой метод доставки использовать: прямой или опосредованный?

Для того чтобы посмотреть, какие параметры Postfix имеют настройки, отличающиеся от настроек по умолчанию, можно выполнить команду `postconf -n`.

Параметр `myorigin` устанавливает доменное имя адресов источников почтовых сообщений, отправляемых данной системой.

По умолчанию параметр `myorigin` установлен в значение, содержащееся в параметре `myhostname`, который, в свою очередь, по умолчанию содержит имя данной машины.

Обычно бывает необходимо скрывать имя хоста в адресах исходящей почты, оставляя лишь имя домена. Например, требуется, чтобы почта, отправляемая пользователем `user1` с хоста `compl.class.edu`, имела адрес источника `user1@class.edu`, а не `user1@compl.class.edu`. Для этого параметр `myorigin` должен быть установлен в `$mydomain` (пример 29.32).

Пример 29.32. Скрытие имени хоста из адреса отправителя

```
myorigin = $mydomain
```

Параметр `mydestination` указывает, для каких доменов принимается почта, т. е. какая почта должна быть доставлена на данную машину вместо отправки ее далее.

По умолчанию параметр `mydestination` настроен так, что почта принимается лишь для данной машины. Если машина должна принимать почту для домена, то в списке `mydestination` должен быть указан также `$mydomain` (пример 29.33).

Пример 29.33. Организация почтового сервера для домена

```
mydestination = $myhostname localhost.$mydomain localhost $mydomain
```

Настройки Postfix по умолчанию заставляют его отправлять почту (relay) в любом направлении для всех клиентов, принадлежащих авторизованным сетям. Авторизованные сети определяются параметром `mynetworks`. Его установки по умолчанию считают авторизованными клиентов, принадлежащих той же сети, что и машина с Postfix.

По умолчанию Postfix выполняет доставку почты для клиентов, не принадлежащих доверенным сетям, направляемую лишь в авторизованных направлениях. Авторизованные направления определяются параметром `relay_domains` (пример 29.34). По умолчанию авторизованными являются домены, указанные в параметре `mydestination`, а также их поддомены.

Пример 29.34. Настройка разрешения передавать почту по умолчанию

```
relay_domains = $mydestination
```

Файл `main.cf` для сервера небольшой сети может выглядеть так, как в примере 29.35.

Пример 29.35. Простая конфигурация Postfix

```
myorigin = $mydomain
mynetworks = 127.0.0.0/8 10.0.0.0/24
relay_domains =
#relayhost = $mydomain
```

Здесь настройка `mynetworks` указывает доверенные сети. Настройка `relay_domains` запрещает обработку почты для всех сетей, не являющихся

доверенными. Настройку `relayhost` можно использовать для отправки почты через почтовый шлюз.

ЗАДАНИЯ

- Создайте простой файл конфигурации Postfix.
- Как можно проверить его синтаксис с помощью команды `postfix`?

Виртуальный хостинг

Сущность виртуального почтового хостинга заключается в том, что почтовый сервер обслуживает почту для доменов, не связанных с именем данного хоста.

Простейшую форму виртуального хостинга можно получить, включив имя домена, для которого должна приниматься почта, в параметр `mydestination`. Этот параметр указывает доменные имена, для которых данная система будет рассматриваться, как конечная точка назначения.

Вариант использования параметра `mydestination` для организации хостинга для домена `nigde.ego.net` приведен в примере 29.36.

Пример 29.36. Включение обслуживаемого домена в список `mydestination`

```
mydestination = $myhostname localhost.$mydomain nigde.ego.net
```

Приведенная конфигурация в `main.cf` настроит Postfix таким образом, что для почты, следующей в домен `nigde.ego.net`, данный сервер будет рассматриваться, как конечная точка назначения. Приведенный пример практически не подходит для реальных условий, т. к. приводит к конфликту имен. Например, если `class.edu` — основное доменное имя для данного сервера, то `user@class.edu` и `user@nigde.ego.net` будут соответствовать одному и тому же пользователю.

Конфигурация Postfix для обеспечения виртуального хостинга базируется на использовании:

- виртуальных псевдонимов;
- виртуальных почтовых ящиков.

В этой главе показан первый способ — виртуальные псевдонимы.

При использовании виртуальных псевдонимов имена пользователей виртуальных доменов, для которых осуществляется хостинг, отображаются на реальных локальных пользователей системы. Далее приведен пример 29.37 настроек в файле `main.cf` для включения виртуального хостинга с использованием виртуальных псевдонимов.

Пример 29.37. Виртуальные псевдонимы

```
virtual_alias_domains = nigde.ego.net
virtual_alias_maps = hash:/etc/postfix/virtual
```

Настройка `virtual_alias_domains` позволяет указать домен `nigde.ego.net`, как виртуальный псевдоним. В случае отсутствия этой настройки Postfix будет отвергать почту для домена `nigde.ego.net`.

В таблице `virtual` задают отображения пользователей виртуального домена на реальных пользователей локальной системы (пример 29.38).

Пример 29.38. Содержимое таблицы `virtual`

```
postmaster@nigde.ego.net postmaster
info@nigde.ego.net       sekretar
sales@nigde.ego.net      otdelprodazz@tjumen.class.edu
```

В файл таблицы `virtual` можно добавить запись "почтового ящика по умолчанию", позволяющий доставлять почту в домен `nigde.ego.net` без явного указания имени пользователя в таблице `virtual` (пример 29.39).

Пример 29.39. Почтовый ящик по умолчанию

```
@nigde.ego.net      otstoinik
```

Изложенный подход имеет существенный недостаток, заключающийся в необходимости использования UNIX учетных записей.

ЗАДАНИЯ

- Создайте настройки для обслуживания виртуального домена `experim.biz`.
- Для домена `experim.biz` должен быть создан почтовый ящик по умолчанию.

POP3/IMAP-сервер Dovecot

Протокол POP3 предназначен для обеспечения работы пользовательских агентов электронной почты MUA. Этот протокол предоставляет возможность пользователю с помощью клиентской программы MUA подключаться к своему почтовому ящику, обслуживаемому сервером POP3, для получения электронной почты. Протокол POP3 ориентирован на хранение сообщений на стороне клиента и описан в RFC 1939.

При подключении по протоколу POP3 пользователь может получить всю почту, пришедшую к нему со времени последнего подключения. Пользователь может прочесть только ту почту, которая уже получена с сервера POP3.

При получении почтового сообщения пользовательским агентом MUA это сообщение может быть либо стерто, либо оставлено на сервере.

Для тестирования работоспособности сервера POP3 можно воспользоваться командой `telnet`, используя подключение по протоколу POP3 (порт 110) — пример 29.40.

Пример 29.40. Тестирование сервера POP3 с помощью `telnet`

```
$ telnet localhost 110
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.
+OK dovecot ready.
USER aberes
+OK
PASS password
+OK Logged in.
LIST
+OK 2 messages:
1 833
2 810
.
RETR 1
+OK 833 octets
Return-Path: <aberes@black.class.edu>
Received: from black.class.edu (localhost.localdomain [127.0.0.1])
    by black.class.edu (8.13.4/8.13.4) with ESMTP id k18IV0kh007986
    for <aberes@black.class.edu>; Sun,31 Jan 2010 23:31:28 +0500
Received: (from aberes@localhost)
    by black.class.edu (8.13.4/8.13.4/Submit) id k18IVO6E007985
    for aberes; Sun,31 Jan 2010 23:31:24 +0500
Date: Sun,31 Jan 2010 23:31:24 +0500
From: aberes@black.class.edu
Message-Id: <201001311831.k18IVO6E007985@black.class.edu>
To: aberes@black.class.edu
```

X-IMAPbase: 1140961700 2

Status: 0

X-UID: 1

Content-Length: 113

X-Keywords:

PID	TTY	TIME	CMD
5294	pts/2	00:00:00	bash
7983	pts/2	00:00:00	ps
7984	pts/2	00:00:00	mail

.

DELE 1

+OK Marked to be deleted.

QUIT

+OK Logging out, messages deleted.

Connection closed by foreign host.

В этом примере было проведено соединение с сервером POP3. С помощью команд USER и PASS пользователь вошел в сеанс POP3. Далее командой LIST был получен список сообщений, затем сообщение 1 было получено командой RETR и удалено командой DELE.

Протокол IMAP ориентирован на хранение почтовых сообщений на стороне сервера. На компьютеры клиентов передаются исходно лишь заголовки почтовых сообщений.

По умолчанию для протокола IMAP определен порт 143 TCP.

Используя IMAP, пользователь может манипулировать письмами на сервере, получая на свой компьютер лишь те письма, которые ему нужны.

Перед каждой командой IMAP должен находиться специальный идентификатор, который позволяет клиенту определить, к какой команде клиента относится ответ сервера (пример 29.41).

Пример 29.41. Тестирование сервера IMAP с помощью telnet

```
$ telnet localhost 143
```

```
Trying 127.0.0.1...
```

```
Connected to localhost.localdomain (127.0.0.1).
```

```
Escape character is '^]'.
```

```
* OK dovecot ready.
```

```
A1 LOGIN aberes password
A1 OK Logged in.
A2 SELECT Inbox
* FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
* OK [PERMANENTFLAGS (\Answered \Flagged \Deleted \Seen \Draft *)] Flags
  permitted.
* 1 EXISTS
* 0 RECENT
* OK [UNSEEN 1] First unseen.
* OK [UIDVALIDITY 1140961700] UIDs valid
* OK [UIDNEXT 3] Predicted next UID
A2 OK [READ-WRITE] Select completed.
A3 FETCH 1 BODY[HEADER]
* 1 FETCH (FLAGS (\Seen) BODY[HEADER] {693}
Return-Path: <aberes@black.class.edu>
Received: from black.class.edu (localhost.localdomain [127.0.0.1])
  by black.class.edu (8.13.4/8.13.4) with ESMTP id k1QDluHi005078
  for <aberes@black.class.edu>; Sun,31 Jan 2010 18:47:56 +0500
Received: (from aberes@localhost)
  by black.class.edu (8.13.4/8.13.4/Submit) id k1QDlu5f005077
  for aberes; Sun,31 Jan 2010 18:47:56 +0500
Date: Sun,31 Jan 2010 18:47:56 +0500
From: aberes@black.class.edu
Message-Id: <201001311347.k1QDlu5f005077@black.class.edu>
To: aberes@black.class.edu
Status: O
X-UID: 2
Content-Length: 113
X-IMAPbase: 1140961700 2
X-Keywords:
)
A3 OK Fetch completed.
A4 FETCH 1 BODY[TEXT]
* 1 FETCH (BODY[TEXT] {117}
  PID TTY          TIME CMD
  3746 pts/2      00:00:00 bash
  5075 pts/2      00:00:00 ps
  5076 pts/2      00:00:00 mail
)
```

```
A4 OK Fetch completed.  
A5 LOGOUT  
* BYE Logging out  
A5 OK Logout completed.  
Connection closed by foreign host.
```

В этом сеансе перед каждой командой клиента установлены идентификаторы A1—A5. Используются команды LOGIN для входа в сеанс IMAP, SELECT для выбора каталога сообщений, FETCH для получения заголовков и тела сообщения и LOGOUT для выхода.

Один из наиболее популярных серверов POP3 и IMAP — Dovecot. Он может быть запущен с помощью сценария в /etc/init.d.

Конфигурационный файл Dovecot — /etc/dovecot.conf. Он прекрасно закомментирован и отличается простотой. Минимальная рабочая конфигурация, приведенная в примере 29.42, состоит всего из двух строк.

Пример 29.42. Минимальная конфигурация Dovecot

```
protocols = imap pop3  
listen = *
```

Настройка protocols определяет, какие протоколы будут поддерживаться, а настройка listen указывает IP-адреса интерфейсов, которые должны прослушиваться.

ЗАДАНИЯ

- Установите сервер Dovecot.
- Создайте простую конфигурацию и протестируйте работу по протоколам POP3 и IMAP.



Глава 30

Печать в GNU/Linux

В этой главе вы познакомитесь с основной системой печати GNU/Linux — CUPS. Вы узнаете, как печатать и как управлять демоном печати, а также познакомитесь с полезными утилитами для печати.

Система печати CUPS

Система CUPS использует протокол IPP (Internet Printing Protocol — 631 порт TCP), предназначенный для управления принтерами и базирующегося на протоколе HTTP. В CUPS имеются файлы описания принтеров PPD (PostScript Printer Description), которые предоставляют информацию о возможностях принтеров. В основе системы CUPS находится демон — планировщик `cupsd`, обслуживающий очередь заданий на печать. Конфигурационные файлы CUPS находятся в `/etc/cups`. Наиболее важные из них:

- ❑ `cupsd.conf` — основной файл настроек сервера;
- ❑ `printers.conf` — описания и настройки принтеров;
- ❑ `classes.conf` — описания для целых классов (групп) принтеров;
- ❑ `client.conf` — индивидуальные настройки для клиентов.

Формат файлов конфигурации аналогичен используемому в сервере Apache, многие настройки имеют совершенно одинаковые названия директив.

Пример типичного содержания файла `/etc/cups/cupsd.conf` (пример 30.1).

Пример 30.1. Файл `cupsd.conf`

```
DocumentRoot /usr/share/cups/docs
LogLevel info
User lp
```

```

Group lp
Port 631
SystemGroup lp
<Location />
    Order Deny,Allow
    Deny From All
    Allow From 127.0.0.1
</Location>
<Location /admin>
    AuthType Basic
    AuthClass System
    Order Deny,Allow
    Deny From All
    Allow From 127.0.0.1
</Location>

```

В этом файле конфигурации использованы следующие настройки:

- ☐ настройка DocumentRoot задает базовый каталог для файлов CUPS;
- ☐ директива LogLevel — уровень важности сообщений для журналирования;
- ☐ User и Group — пользователь и группа, от имени которых работает сервер;
- ☐ Port — номер порта, прослушиваемый сервером;
- ☐ SystemGroup — группа, необходимая для доступа к принтеру;
- ☐ <Location> — дают права доступа к ресурсам.

Допустим, что в системе установлен USB-принтер HP LJ1100. В примере 30.2 приведена соответствующая конфигурация в файле настроек принтеров.

Пример 30.2. Файл printers.conf

```

<DefaultPrinter laser>
Info HP1100
Location In my room
DeviceURI usb:/dev/usb/lp0
State Idle
Accepting Yes
JobSheets none none
QuotaPeriod 0
PageLimit 0
KLimit 0
</Printer>

```

В примере 30.2 использованы следующие настройки:

- ☐ Info — описание принтера;
- ☐ Location — местонахождение принтера;
- ☐ DeviceURI — универсальный идентификатор подключения принтера;
- ☐ State — исходное состояние принтера;
- ☐ Accepting — разрешение посылать задания на печать для данного принтера;
- ☐ JobSheets — наличие расписания;
- ☐ QuotaPeriod — период квоты;
- ☐ PageLimit — ограничение на максимальное количество печатаемых страниц;
- ☐ KLimit — ограничение на максимальный объем заданий на печать.

ЗАДАНИЯ

- Сконфигурируйте ваш принтер с помощью Web-интерфейса.
- Установите ограничение для заданий: максимальное количество печатаемых страниц — не более 100, максимальный объем заданий на печать — не более 10 Мбайт.

Команды CUPS

Основная команда для печати в CUPS — `lp`. Так, для печати на принтер по умолчанию документа `test.txt` достаточно выполнить команду, приведенную в примере 30.3.

Пример 30.3. Печать файла в CUPS

```
$ lp test.txt
request id is dj-1 (1 file(s))
```

Далее приведены основные опции команды `lp`:

- ☐ `-d` — имя принтера, в очередь которого должно быть послано задание;
- ☐ `-h` — имя узла сети для отправки задания на него;
- ☐ `-i` — идентификатор задания;
- ☐ `-n` — количество копий;
- ☐ `-q` — устанавливает приоритет задания в очереди;
- ☐ `-u` — имя пользователя;

❑ -n — дополнительные опции для данного задания;

❑ -P — задает список страниц для печати.

Например, для вывода на принтер dj документа test.txt и печати двух его экземпляров следует использовать команду, показанную в примере 30.4.

Пример 30.4. Печать двух копий файла

```
lp -d dj -n 2 test.txt
request id is dj-2 (1 file(s))
```

Для печати в системе CUPS можно также использовать команду lpr (пример 30.5).

Пример 30.5. Использование lpr для печати в CUPS

```
lpr -#2 -P dj test.txt
```

Если в системе имеется PostScript-принтер, то для печати на нем текстового файла его разумно преобразовать с помощью утилиты a2ps (пример 30.6).

Пример 30.6. Преобразование текстового файла в PostScript-файл

```
$ a2ps -o test.ps test.txt
```

Эта команда преобразует текстовый файл test.txt в PostScript-файл test.ps.

Далее полученный файл может быть просто выведен на PostScript-принтер. Если не используется опция -o, то вывод осуществляется в очередь принтера по умолчанию.

Для печати PostScript-файла на принтере, не поддерживающем PostScript, можно воспользоваться утилитой gs, обрабатывающей Ghostscript-файлы (Ghostscript — открытый формат, совместимый с PostScript) — пример 30.7.

Пример 30.7. Печать Ghostscript

```
$ gs -dSAFER -dNOPAUSE -sDEVICE=hpdj855c -sOutputFile=- -q test.ps | lp
```

В этом примере утилита gs читает файл test.ps, указанный в качестве аргумента. Вывод через конвейер осуществляется в стандартный поток ввода команды lp.

Команда `gs` — это интерпретатор, и при вызове ее без аргументов можно использовать встроенные команды интерпретатора в интерактивном режиме.

Часто используются следующие опции команды `gs`:

- ❑ `-dSAFER` — открывает файл, заданный команде `gs`, только для чтения;
- ❑ `-dNOPAUSE` — не приостанавливать печать после вывода каждой страницы;
- ❑ `-sDEVICE` — указывает тип устройства, на которое производится вывод;
- ❑ `-sOutputFile` — имя файла для вывода;
- ❑ `-sPAPERSIZE` — размер бумаги, например, `-sPAPERSIZE=a4`.

Задания

- Как напечатать файл `test` на принтере `laser`, подключенном к узлу `trainer`?
- Каким образом напечатать документ с задержкой в пять минут?

Управление принтерами в CUPS

Для конфигурирования и управления принтерами в системе CUPS может быть вызвана Web-утилита с помощью обращения в браузере по адресу **`http://localhost:631`**.

Если же необходимо использовать командную строку, то можно воспользоваться командой `lpadmin`. Так, например, для установки ограничения на максимальное количество страниц в задании можно сделать следующее (пример 30.8).

Пример 30.8. Ограничение на максимальное количество страниц

```
# lpadmin -p laser -o job-page-limit=100

# cat /etc/cups/printers.conf
<DefaultPrinter laser>
...
PageLimit 100
KLimit 0
</Printer>
```

Команда `lpadmin` установила ограничения для задач, помещаемых на принтер `laser` (что указано опцией `-p`), на максимальное количество страниц в задании, равное 100.

Пример 30.9. Разрешение печати для заданного пользователя

```
# lpadmin -p laser -u allow:user1

# cat /etc/cups/printers.conf
<DefaultPrinter laser>
...
AllowUser user1
</Printer>
```

В примере 30.9 настроено разрешение на доступ к принтеру пользователю user1. Это изменение также будет отражено в файле /etc/cups/printers.conf.

Современные принтеры предоставляют широкие возможности для конфигурирования, задаваемые в файлах PPD (PostScript Printer Definition) — пример 30.10.

Пример 30.10. Опции настройки принтера

```
# lpoptions -l
Resolution/Output Resolution: 150dpi 300dpi *600dpi 1200dpi
Duplex/Double-Sided Printing: *None DuplexNoTumble DuplexTumble
PageSize/Media Size: Letter Legal Executive Tabloid A3 *A4 A5 B5 EnvISOB5
Env10 EnvC5 EnvDL EnvMonarch
InputSlot/Media Source: *Default Tray1 Tray2 Tray3 Tray4 Manual Envelope Auto
PageRegion/PageRegion: Letter Legal Executive Tabloid A3 A4 A5 B5
EnvISOB5 Env10 EnvC5 EnvDL EnvMonarch
Option1/Duplexer: True *False
```

В примере 30.11 показано, как изменить разрешение принтера на 300 dpi.

Пример 30.11. Изменение настроек принтера

```
# lpadmin -p laser -o Resolution=300dpi
# lpoptions -l
Resolution/Output Resolution: 150dpi *300dpi 600dpi 1200dpi
...
```

Задания

- Ограничьте максимальный размер задания на печать 10 Мбайт.
- Запретите всем пользователям, кроме входящих в группу lp, печать на принтере.
- Установите худшее качество печати и опробуйте печать.

Управление очередью печати

Команда `lpstat` показывает состояние очередей печати CUPS (пример 30.12).

Пример 30.12. Получение состояния очередей печати

```
# lpstat -a
laser accepting requests since Jan 01 00:00
```

Подробную информацию предоставляет опция `-t` команды `lpstat` (пример 30.13).

Пример 30.13. Состояние системы печати CUPS

```
# lpstat -t
scheduler is running
system default destination: laser
device for laser: usb:/dev/usb/lp0
laser accepting requests since Jan 01 00:00
printer laser is idle.  enabled since Jan 01 00:00
```

Часто необходимо запретить вывод заданий на печать, оставляя возможность постановки заданий на печать. Например, для смены картриджа принтера. Достигается это с помощью команды `cupsdisable` (пример 30.14).

Пример 30.14. Запрет на печать

```
# cupsdisable laser
# lpstat -t
scheduler is running
system default destination: laser
device for laser: usb:/dev/usb/lp0
laser accepting requests since Jan 01 00:00
printer laser disabled since Jan 01 00:00 -
    Paused
```

При этом задания будут накапливаться в очереди (пример 30.15).

Пример 30.15. Постановка заданий на печать при запрете печати

```
$ lp -d laser lsmod.asp
request id is laser-1 (1 file(s))
$ lpstat
```

```
laser-1          user1          2048   Sat 16 Jan 2010 01:06:39
$ lp -d laser smbldap-howto.fr.html
request id is laser-2 (1 file(s))
$ lpstat
laser-1          user1          2048   Sat 16 Jan 2010 01:06:39
laser-2          user1          139264  Sat 16 Jan 2010 01:08:57
```

Как видно из этого примера, задания на печать накапливаются в очереди.

Командой `reject` можно запретить постановку заданий на печать (пример 30.16).

Пример 30.16. Запрет постановки заданий в очередь печати

```
# reject -r 'Ushel na bazu!' laser
# lpstat -t
scheduler is running
system default destination: laser
device for laser: usb:/dev/usb/lp0
laser not accepting requests since Jan 01 00:00 -
    Ushel na bazu!
printer laser disabled since Jan 01 00:00 -
    Ushel na bazu!

laser-1          user1          2048   Sat 16 Jan 2010 01:06:39
laser-2          user1          139264  Sat 16 Jan 2010 01:08:57
```

Опция `-r` команды `reject` позволяет указать причину отказа в приеме заданий.

Задания в очереди можно перемещать. Так, для немедленной печати задания `laser-2` необходимо выполнить команду, показанную в примере 30.17.

Пример 30.17. Перемещение задания в очереди

```
# lp -i laser-2 -H immediate
# lpstat -u user1
laser-2          user1          139264  Sat 16 Jan 2010 01:08:57
laser-1          user1          2048   Sat 16 Jan 2010 01:06:39
```

Опция `-i` команды `lp` указывает задание, а `-H immediate` — перемещает его вперед.

Команда `cancel` снимает задание с печати, причем задания могут быть указаны как индивидуально, так и группой. Например, для удаления из очереди

всех заданий от пользователя `user1` следует указать его после опции `-u` (пример 30.18).

Пример 30.18. Удаление заданий на печать пользователя

```
# lpstat -u user1
laser-2                user1          139264    Sat 16 Jan 2010 01:08:57
laser-1                user1          2048     Sat 16 Jan 2010 01:06:39
# cancel -u user1
# lpstat -u user1
```

Команда `accept` разрешает ставить задания в очередь печати (пример 30.19).

Пример 30.19. Разрешение постановки заданий на печать

```
# accept laser
# lpstat -t
scheduler is running
system default destination: laser
device for laser: usb:/dev/usbblp0
laser accepting requests since Jan 01 00:00
printer laser disabled since Jan 01 00:00 -
    reason unknown
```

Команда `cupsenable` позволяет разрешить печать (пример 30.20).

Пример 30.20. Разрешение печати

```
# cupsenable laser
# lpstat -t
scheduler is running
system default destination: laser
device for laser: usb:/dev/usbblp0
laser accepting requests since Jan 01 00:00
printer laser is idle.  enabled since Jan 01 00:00
```

ЗАДАНИЯ

- Распечатайте любой текстовый файл.
- Запретите постановку заданий в очередь. Продолжается ли печать текста?
- Запретите печать. Продолжается ли печать текста?
- Снимите задание с печати. Продолжается ли печать текста?

Глава 31



Сервер NTP

Поддержка точного времени на системных часах компьютеров исключительно важна с точки зрения задач аутентификации и безопасности. Сервис NTP позволяет синхронизировать время на компьютерах. В этой главе вы познакомитесь с ним.

Сервис синхронизации времени

Поддержка точного времени на компьютерах и других сетевых устройствах нужна не только для правильной работы приложений. Вот лишь некоторые аспекты работы GNU/Linux-систем, зависящие от точности системных часов:

- ☐ выполнение отложенных заданий `at`;
- ☐ выполнение периодических заданий `cron`;
- ☐ запись сообщений в системные журналы;
- ☐ возможность выполнения анализа событий, происходящих на нескольких устройствах сети одновременно;
- ☐ правильная работа систем аутентификации и авторизации;
- ☐ исключение несанкционированных модификаций в системе, скрываемых с помощью манипуляций с системным временем и временными метками модификации файлов.

При загрузке операционной системы системные часы инициализируются от аппаратных часов. Текущее время и дату системных часов выводит и устанавливает команда `date`. Аппаратные часы можно установить командой `hwclock`. В большинстве GNU/Linux-систем при их работе генерируются аппаратные прерывания (прерывания таймера) с частотой 100 Гц (clock tick). Ядро Linux увеличивает счетчик времени с помощью обработчика прерывания

таймера. Однако в сильно загруженных системах ядро не успевает обработать все прерывания таймера. Это связано с наличием более приоритетных задач. Таким образом, для высоко загруженных систем характерно стабильное отставание системных часов.

Протокол NTP (Network Time Protocol) используется для синхронизации времени на сервере и клиенте с точностью до миллисекунд в локальных сетях и десятков миллисекунд в глобальных. Текущая версия протокола — NTPv4.

Протокол NTP предусматривает наличие нескольких уровней серверов (tiers — "ярусы"). Серверы NTP первого уровня синхронизируются с помощью атомных часов или иных эталонных приборов точного времени. Серверы второго и третьего уровней синхронизируются по серверам первого уровня и предназначены для обслуживания клиентских запросов. Имеются серверы точного времени с публичным доступом. Узнать их адреса можно, обратившись на <http://pool.ntp.org>.

В GNU/Linux распространены два разных пакета для работы с сервисом NTP. Первый называется `ntp` и содержит в себе демон `ntpd` и прочие программы. Второй пакет — это `ntpdate`, содержащий одноименную клиентскую программу.

ЗАДАНИЯ

- Проверьте наличие в системе пакетов `ntp` и `ntpdate`. Если их нет — установите.
- Какие программы входят в состав пакета `ntp`?

Утилита `ntpdate`

Команда `ntpdate` предназначена для синхронизации времени с NTP-сервера при ее вызове. Она обычно вызывается либо с помощью скрипта при загрузке системы, либо на регулярной основе с помощью `cron`. Безусловное преимущество использования команды `ntpdate` заключается в простоте ее использования (пример 31.1).

Пример 31.1. Команда `ntpdate`

```
# ntpdate -v ntp.ubuntu.com
30 Jan 14:31:16 ntpdate[2466]: ntpdate 4.2.4p6@1.1549-o Fri Dec 4
18:08:43 UTC 2009 (1)
30 Jan 14:31:15 ntpdate[2466]: step time server 91.189.94.4 offset -
1.378534 sec
```


В примере 31.1 команда `ntpdate` синхронизировала системное время с NTP-сервером **ntp.ubuntu.com**. Опция `-v` включает режим подробного информирования о работе команды `ntpdate`.

Задания

- Узнайте на **<http://pool.ntp.org>** имена публичных серверов NTP в Российской Федерации.
- Синхронизируйте системное время командой `ntpdate` с одним из этих серверов.

Пакет *ntp*

Демон `ntpd`, поставляющийся в составе пакета `ntp`, радикально отличается от программы `ntpdate`. Команда `ntpdate` запускается единожды и корректирует системное время, изменяя системные часы, возможно, довольно значительно. В отличие от нее демон `ntpd` работает постоянно и поддерживает синхронизацию системного времени намного точнее. Демон `ntpd` вычисляет отклонение системного времени от точного времени, постоянно подстраивая его. Таким образом, больших изменений системного времени при использовании демона `ntpd` не происходит, что гораздо больше подходит для постоянно работающих серверов.

В конфигурационном файле сервера `ntpd` — `/etc/ntp.conf` в простейшем случае достаточно указать лишь имена серверов NTP, с которыми необходимо синхронизироваться (пример 31.2).

Пример 31.2. Фрагмент файла конфигурации `ntp.conf`

```
...  
server ntp.ubuntu.com  
...
```

В примере 31.2 конфигурации демона `ntpd` указано, что в данном случае синхронизация системного времени будет производиться на публичном сервере `ntp.ubuntu.com`.

Пакет `ntp` содержит в себе несколько драйверов для подключения локальных устройств точного времени. Каждый драйвер идентифицируется с помощью номера. Список поддерживаемых драйверов можно найти в файле `refclock.html` документации на демон `ntpd`. В файле конфигурации подключение

локальных устройств производится так, как будто это устройства в IP-сети. Для локальных устройств используются IPv4-адреса `127.127.t.u`, где:

- `t` — тип эталонного таймера;
- `u` — номер интерфейса.

Так, например, для подключения устройства Motorola Oncore GPS используется тип 30. Для первого устройства номер порта 0 (пример 31.3).

Пример 31.3. Синхронизация от локального GPS-приемника

```
...  
server 127.127.30.0  
...
```

Запускается демон `ntpd` с помощью скрипта в `/etc/init.d` (пример 31.4).

Пример 31.4. Запуск `ntpd`

```
# /etc/init.d/ntp start  
Starting NTP server ntpd [ ok ]
```

ЗАДАНИЯ

- Сконфигурируйте `ntpd` на использование сервера NTP в РФ.
- Запустите демон `ntpd`.



Глава 32

Система X Window

В этой главе описана графическая система X Window. Здесь вы узнаете, как настроить X-сервер. Вы научитесь запускать X-клиенты и настраивать их поведение, а также познакомитесь с файлами ресурсов и удаленным входом в сеанс X Window.

Организация X Window

Разработка системы X Window началась в первой половине 80-х годов прошлого века в Массачусетском технологическом институте (MIT) в рамках проекта Athena, который финансировался фирмами IBM и DEC. Целью проекта являлось построение распределенной графической среды, позволяющей единообразно работать с различным оборудованием и разными операционными системами. Сейчас используется 11-я версия X (выпуски 6 или 7), поэтому протокол называется X11.

Система X Window построена в рамках архитектуры "клиент — сервер". Сервер предназначен для работы с устройством отображения, например, видеоадаптером.

Сервер передает видеоадаптеру команды для отображения графических примитивов. Видеоадаптер является устройством вывода графической информации. В то же время X-сервер способен работать и с различными устройствами ввода информации, такими как мышь, клавиатура и джойстик.

Клиентские программы взаимодействуют с X-сервером с помощью X-протокола. Он представляет собой обычный протокол прикладного уровня TCP/IP, которому присвоен порт 6000 TCP. Поэтому X-клиент и X-сервер могут с успехом работать по сети. То есть программы клиент и сервер вполне могут находиться на различных машинах и никаких дополнительных программ для этого не требуется.

X-клиенты могут быть условно подразделены на четыре категории:

- *обычные X-приложения* (X applications). Примерами таковых являются браузер mozilla, программа просмотра PDF xpdf, текстовый редактор emacs;
- *оконные менеджеры* (Window Managers). Они предназначены для обеспечения возможности управления окнами с помощью устройств ввода и предоставления удобного пользовательского интерфейса. Примерами оконных менеджеров являются Window Maker (исполняемый файл wmaker), Ice WM (icewm) и Xfwm;
- *графические рабочие окружения* (Desktop Environment), представляющие собой большие комплексы программного обеспечения, включающие в себя собственные оконные менеджеры, файловые менеджеры, средства офисной работы и иное пользовательское прикладное программное обеспечение. X-приложения, созданные для работы в составе рабочей среды, обеспечивают единообразный пользовательский интерфейс. Наиболее распространены KDE (K Desktop Environment), GNOME (GNU Objects Model Environment) и Xfce;
- *менеджеры сеанса* (X Session Managers). Программы особого рода, предназначенные для обеспечения возможности непосредственного запуска X-сеанса без необходимости предварительного входа в обычный неграфический сеанс Shell. Кроме этого, менеджеры X-сеанса отвечают за перезапуск X-сервера в случае его остановки. Наиболее распространены xdm, kdm и gdm. Менеджеры X-сессий используют протокол XDMCP (X Display Manager Control Protocol). Запуск X-сеанса осуществляется иначе, чем при использовании обычного текстового терминала. При обычном входе в сеанс запускается Shell, связанный с текстовым терминалом, а в X — менеджер окон, рабочая среда (desktop) или иное X-приложение.

X-протокол открыт, и поэтому существует множество его реализаций, предназначенных, прежде всего, для запуска на различных аппаратных платформах. На платформе IA-32 могут быть использованы различные X-системы, например:

- XFree86 — до выпуска 4.4rc2 применялась в GNU/Linux повсеместно (в настоящее время используется редко из-за ограничительных условий лицензии);
- Xorg — реализация X-системы (ответвление от XFree86 4.4rc2 в 2004 г.).

Написание программ для X-системы обычно требует интенсивного использования инструментальных библиотек. Различные библиотеки предоставляют

разные средства и требуют использования разных прикладных интерфейсов программирования (Application Program Interface, API). Несмотря на то, что X-протокол является стандартным, существует множество несовместимых платформ для X-системы.

В GNU/Linux две X-платформы получили наибольшее распространение:

- ❑ Qt — эту платформу использует KDE;
- ❑ Gtk+ — используется множеством X-приложений, средами GNOME и Xfce.

В GNU/Linux требования стандарта FHS предписывают размещать программное обеспечение X-сервера в каталоге `/usr/X11R6` или `/usr/X11`. Оконные менеджеры и иные X-приложения часто также размещаются в этом каталоге. Традиционно X-сервер — это исполняемый файл `x` или `xorg` (пример 32.1).

Пример 32.1. X-сервер

```
$ ls -l `which X`  
lrwxrwxrwx 1 root root 4 Nov 30 2009 /usr/bin/X -> /usr/bin/Xorg
```

Вместе с X Window может работать сервер шрифтов X Font Server (`xfst`), который управляет шрифтами, установленными в системе, предоставляя их X-серверу для отображения. Если X-сервер использует в работе сервер шрифтов, то X-сервер является для сервера шрифтов клиентом.

ЗАДАНИЯ

- Изучите структуру каталога `/usr/X11R6`.
- Определите версию X-сервера. Установлен X-сервер Xorg или XFree86?
- Какие оконные менеджеры имеются в вашей системе?

Конфигурирование X Window

Конфигурационным файлом для XFree86 является `XF86Config` (в XFree86 версий 4.x используется файл `XF86Config-4`, т. к. формат конфигурационного файла для предыдущих версий XFree86 отличался, и только при его отсутствии используется `XF86Config`). В Xorg конфигурационный файл называется `xorg.conf`. Эти файлы в GNU/Linux размещаются в каталоге `/etc/X11`.

Формат конфигурационных файлов `XF86Config` и `xorg.conf` отличается лишь в деталях. Для создания этого файла обычно применяются автоматизированные процедуры, хотя он может быть создан и вручную.

Для создания конфигурационного файла можно использовать:

- ☐ утилиты, специфичные для дистрибутива, например, в SUSE — `sax2`;
- ☐ команду `X -configure`, которая производит автоматическую проверку оборудования и сохраняет новый файл конфигурации в текущем каталоге;
- ☐ отдельно поставляющиеся утилиты `xorgcfg` или `xf86cfg`;
- ☐ Shell-сценарий `xorgconfig` или `xf86config`, позволяющий конфигурировать X-сервер, отвечая на последовательность вопросов.

Обычно в файле `xorg.conf` имеются следующие секции:

- ☐ `ServerLayout` — указывает идентификаторы используемого X-сервером экрана и устройств ввода;
- ☐ `Module` — подключает модули расширения X-сервера;
- ☐ `InputDevice` — описывает используемые устройства ввода, например, мышь или клавиатуру;
- ☐ `Files` — задает пути к библиотеке цветов RGB и каталогам шрифтов;
- ☐ `ServerFlags` — устанавливает дополнительные флаги X-сервера;
- ☐ `Monitor` — определяет параметры используемого монитора;
- ☐ `Device` — описывает видеоадаптер;
- ☐ `Screen` — задает параметры отображения информации на экране, например, глубины цвета.

Для использования русской раскладки клавиатуры в секции `InputDevice`, связанной с клавиатурой директивой `Driver "Keyboard"`, можно использовать конфигурацию, приведенную в примере 32.2.

Пример 32.2. Настройки клавиатуры для русской раскладки

```
Section "InputDevice"
    Identifier "Keyboard[0]"
    Driver "keyboard"
    Option "Protocol" "Standard"
    Option "XkbModel" "pc104"
    Option "XkbRules" "xfree86"
    Option "XkbLayout" "us,ru"
    Option "XkbOptions" "grp:ctrl_shift_toggle,grp_led:scroll"
EndSection
```

Настройки для клавиатуры:

- ❑ количество клавиш указывает Option "XkbModel" "pc104";
- ❑ опция Option "XkbLayout" "us,ru" задает раскладку клавиатуры;
- ❑ опция Option "XkbOptions" "grp:ctrl_shift_toggle,grp_led:scroll" задает метод переключения раскладки — сочетание клавиш <Ctrl>+<Shift>;
- ❑ при переключении раскладки клавиатуры на русский язык будет загораться светодиод Scroll Lock на клавиатуре для индикации текущей раскладки.

Для описания мыши используются директивы, приведенные в примере 32.3.

Пример 32.3. Настройки мыши

```
Section "InputDevice"
    Identifier      "Mouse[0]"
    Driver         "mouse"
    Option         "Protocol" "IMPS/2"
    Option         "Device"   "/dev/input/mice"
    Option         "ZAxisMapping" "4 5"
    Option         "Emulate3Buttons" "no"
EndSection
```

При использовании двухкнопочных мышей имеет смысл включить эмуляцию трехкнопочной мыши с помощью опции `Emulate3Buttons`. Это позволит эмулировать щелчок средней кнопкой мыши при нажатии одновременно двух кнопок. Для обычных двухкнопочных мышей чаще всего используется протокол PS/2, а для мышей с колесиком — IMPS/2. Для того чтобы можно было пользоваться скроллингом, управляемым колесиком мыши, необходимо указать Option "ZAxisMapping" "4 5".

Мыши, подключаемые к разъему PS/2, используют файл устройства `/dev/psaux`, а USB-мыши обычно используют `/dev/input/mice`.

В секции `Files` установлен путь к библиотеке RGB (пример 32.4). Она определяет удобные имена для сочетаний битов RGB, задающих цвета (см. файл `rgb.txt` в каталоге `/usr/X11R6/lib/X11/` или `/usr/share/X11/`).

Пример 32.4. Путь к библиотеке RGB

```
RgbPath "/usr/X11R6/lib/X11/rgb"
```

Если сервер шрифтов `xf86` не используется, то в секции `Files` имеются пути к каталогам со шрифтами (пример 32.5).

Пример 32.5. Путь к кириллическим шрифтам

```
FontPath      "/usr/X11R6/lib/X11/fonts/cyrillic/"
```

Если используется сервер шрифтов, то эти строки отсутствуют. Вместо них указывается сокет сервера шрифтов (пример 32.6).

Пример 32.6. Настройка для использования сервера шрифтов

```
FontPath      "unix/:-1"
```

Настройка в примере 32.6 заставляет X-сервер использовать сервер шрифтов на локальной машине. При необходимости использования централизованного сервера на выделенной машине следует указать имя или IP-адрес машины и номер порта. Сервер шрифтов обычно работает с портом TCP 7100 (пример 32.7).

Пример 32.7. Подключение к серверу шрифтов через сеть

```
FontPath      "tcp/:192.168.1.1:7100"
```

Секция `Monitor` задает параметры монитора (пример 32.8).

Пример 32.8. Настройки монитора

```
Section "Monitor"
    DisplaySize  331 207
    HorizSync    30-62
    Identifier   "Monitor[0]"
    ModelName    "LENOVO LCD MONITOR"
    Option       "DPMS"
    Option       "PreferredMode" "1280x800"
    VendorName   "LEN"
    VertRefresh  43-60
    UseModes     "Modes[0]"
EndSection
```


Важнейшими параметрами здесь является кадровая частота `VertRefresh` (в герцах) и частота строчной синхронизации `HorizSync` (в килогерцах). Строка `Modeline` задает разрешение экрана и временные параметры, необходимые для корректного отображения изображения на мониторе. Настройка временных параметров может быть произведена с помощью графической программы `xvidtune`.

В секции `Device` описывают видеоадаптер, а точнее указывают его PCI-шинный идентификатор, например, `BusID "PCI:0:2:0"`. Исключительно важно правильно указать драйвер для видеоадаптера, например: `Driver "i810"`.

Секция `Screen` указывает видеорежим и глубину цветов (пример 32.9).

Пример 32.9. Настройки разрешения и глубины цветов

```
Section "Screen"
    DefaultDepth 24
    SubSection "Display"
        Depth        24
        Modes         "1280x800" "1280x768" "640x480"
        Virtual       3840 1200
    EndSubSection
    Device           "Device[0]"
    Identifier       "Screen[0]"
    Monitor          "Monitor[0]"
EndSection
```

Здесь настройка `DefaultColorDepth` задает глубину цвета, т. е. количество битов, используемых для кодировки цветов. В подсекции `Display` указывается видеорежим.

Задания

- Определите, какие утилиты для конфигурирования X-сервера имеются в вашей системе.
- Какая глубина цвета и какое разрешение используются в вашей системе?
- Какой файл устройства используется в вашей системе?
- Какая модель видеоадаптера применяется в вашей системе?
- Проверьте тип мыши и, если она обладает колесиком, проверьте настройки в файле конфигурации, позволяющие управлять скроллингом.

Сервер шрифтов

X-сервер способен самостоятельно обрабатывать шрифты, установленные в системе, однако эту работу можно поручить специализированному серверу шрифтов — программе `xf86`. Этот сервер способен предоставлять X-серверу шрифты через сеть, позволяя, таким образом, создавать централизованное хранилище шрифтов в сети.

Сервер шрифтов лучше, чем X-сервер, обрабатывает шрифты, предоставляя приложениям лучшее качество воспроизведения шрифтов на экране, а также возможности, которые не поддерживаются самим X-сервером. Используя сервер шрифтов, гораздо проще обеспечить поддержку новых форматов шрифтов.

Сервер шрифтов `xf86` запускается на стадии инициализации операционной системы при переходе в многопользовательский режим, как сервер `stand-alone`.

Конфигурационный файл сервера шрифтов — `/etc/X11/fs/config` (пример 32.10).

Пример 32.10. Файл конфигурации сервера шрифтов

```
no-listen = tcp
#client-limit = 10
clone-self = on
catalogue = /usr/X11R6/lib/X11/fonts/75dpi,
            /usr/X11R6/lib/X11/fonts/cyrillic,
            /usr/X11R6/lib/X11/fonts/ukr,
            /usr/X11R6/lib/X11/fonts/100dpi,
            /usr/X11R6/lib/X11/fonts/misc,
            /usr/X11R6/lib/X11/fonts/Type1,
            /usr/X11R6/lib/X11/fonts/Speedo,
            /usr/X11R6/lib/X11/fonts/CID,
            /usr/X11R6/lib/X11/fonts/util,
            /usr/X11R6/lib/X11/fonts/local,
            /usr/X11R6/lib/X11/fonts/truetype,
            /usr/X11R6/lib/X11/fonts/TrueType,
            /usr/X11R6/lib/X11/fonts/freefont,
            /usr/X11R6/lib/X11/fonts/sharefont
default-point-size = 120
default-resolutions = 75,75,100,100
use-syslog = on
```

В примере 32.10 используются следующие настройки сервера шрифтов xfs:

- ☐ `no-listen = tcp` — запрет на соединение с сервером шрифтов по TCP;
- ☐ `client-limit` — ограничивает допустимое количество клиентов;
- ☐ `clone-self = on` — если ограничено максимально допустимое количество клиентов, то при достижении ограничения будет запускаться копия сервера;
- ☐ `catalogue` — список каталогов шрифтов;
- ☐ `default-point-size` — размер шрифта по умолчанию;
- ☐ `default-resolutions` — список поддерживаемых разрешений для масштабируемых шрифтов;
- ☐ `use-syslog` — использовать демон `syslogd` для записи информации в журналы.

Каталоги со шрифтами находятся в `/usr/X11R6/lib/X11/fonts/` (пример 32.11).

Пример 32.11. Каталог со шрифтами KOI8-R

```
$ ls /usr/X11R6/lib/X11/fonts/koi8
encodings.dir      koi10x20.pcf.gz    koi6x13b.pcf.gz    koi8x13.pcf.gz
koi9x18b.pcf.gz    fonts.alias         koi12x24b.pcf.gz    koi6x13.pcf.gz
koi8x16b.pcf.gz    koi9x18.pcf.gz     fonts.dir           koi12x24.pcf.gz
koi6x9.pcf.gz      koi8x16.pcf.gz     koinil2.pcf.gz      fonts.list
koi5x8.pcf.gz      koi7x14b.pcf.gz    koi9x15b.pcf.gz     koi10x16b.pcf.gz
koi6x10.pcf.gz     koi7x14.pcf.gz     koi9x15.pcf.gz
```

В каталогах находятся файлы шрифтов и служебные файлы, необходимые для доступа X-сервера к шрифтам.

Если возникает необходимость установить в систему новый шрифт, то в каталоге с этим шрифтом следует выполнить команду `mkfontdir`, позволяющую создать служебные файлы, обеспечивающие доступ к шрифтам (пример 32.12).

Пример 32.12. Создание каталога шрифтов

```
# ls kwintv/
led-fixed.pcf
# mkfontdir kwintv/
# ls kwintv/
fonts.dir  led-fixed.pcf
# xset fp rehash
```

В этом примере в каталоге kwintv был создан служебный файл с помощью команды `mkfontdir`. Команда `xset fp rehash` сообщает X-серверу о необходимости перечитать каталоги, содержащие шрифты.

ЗАДАНИЯ

- Определите, используется ли сервер шрифтов в вашей системе.
- Настройте X-сервер и сервер шрифтов на взаимодействие через порт TCP.
- Имеются ли в файле конфигурации `xf86` пути к каталогам с русскими шрифтами (в их именах обычно есть строки `cyrillic`, `koi8`, `cp1251` и т. п.)?

Запуск X-сервера из командной строки

X-сервер без каких-либо X-клиентов можно запустить командой `x` (пример 32.13).

Пример 32.13. Запуск X-сервера

```
$ X :0.0 &
X.Org X Server 1.6.5
Release Date: 2009-10-11
X Protocol Version 11, Revision 0
Build Operating System: openSUSE SUSE LINUX
Current Operating System: Linux linux-0qp2 2.6.31.8-0.1-desktop #1 SMP
PREEMPT 2009-12-15 23:55:40 +0100 i686
Build Date: 02 November 2009 12:05:39PM
```

```
Before reporting problems, check http://wiki.x.org
to make sure that you have the latest version.
```

```
Markers: (--) probed, (**) from config file, (==) default setting,
        (++) from command line, (!!) notice, (II) informational,
        (WW) warning, (EE) error, (NI) not implemented, (??) unknown.
(==) Log file: "/var/log/Xorg.0.0.log", Time: Thu Jan 28 10:37:23 2010
(==) Using config file: "/etc/X11/xorg.conf"
(EE) Failed to load module "freetype" (module does not exist, 0)
```

Эта команда запустит X-сервер в фоновом режиме. Строка `:0.0`, указанная в качестве аргумента, значит следующее: данный сервер X является первым экземпляром X-серверов и использует первый экран для отображения.

К системе может быть подключено множество терминалов, а в каждом из них может быть несколько мониторов. Но даже в системе, обладающей единственным физическим экраном, можно запустить несколько X-серверов, если для этого имеется достаточное количество ресурсов в системе. Так, например, для запуска второго X-сервера в системе можно выполнить команду, показанную в примере 32.14.

Пример 32.14. Запуск второго X-сервера

```
$ X :1.0 &
```

При этом оба сервера будут готовы принимать соединения по X-протоколу для отображения графики на экране (пример 32.15).

Пример 32.15. Открытые X-серверами порты

```
$ netstat -tan
```

Active Internet connections (servers and established)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	0.0.0.0:6000	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:6001	0.0.0.0:*	LISTEN

Запущенные X-серверы используют свободные виртуальные терминалы. Так, например, если в системе используется шесть виртуальных терминалов, то первый виртуальный X-терминал будет доступен с помощью сочетания клавиш <Ctrl>+<Alt>+<F7>, а второй — <Ctrl>+<Alt>+<F8>. Завершить работу X-сервера можно сочетанием клавиш <Ctrl>+<Alt>+<Backspace>. Обратите внимание на то, что первый X-сервер прослушивает порт TCP 6000, а второй — 6001. Фактически, номер порта напрямую связан с номером экземпляра X-сервера.

X-клиент, который должен обслуживаться первым X-сервером, будет запущен, если указана опция `-display :0.0` (пример 32.16).

Пример 32.16. Запуск X-клиента

```
$ xterm -display :0.0 &
```

Клиент, который будет обслуживаться вторым сервером, можно запустить, указав после опции `-display` номер второго X-сервера (пример 32.17).

Пример 32.17. Запуск X-клиента, работающего со вторым X-сервером

```
$ xeyes -display :1.0 &
```

Если X-сервер не должен прослушивать порт TCP, а использовал вместо этого UNIX-сокеты для обмена информацией с клиентами, то его следует запустить, используя опцию `-nolisten tcp` (пример 32.18).

Пример 32.18. Запрет на открытие порта TCP

```
$ X -nolisten tcp :0.0 &
```

Использование порта TCP для работы X-сервера не приветствуется с точки зрения безопасности системы и допустимо лишь в системах, где X-сервер и X-клиенты запускаются на различных компьютерах.

Запуск X-сервера без последующего автоматического запуска требуемых X-клиентов не удобен, поэтому разумно использовать программу `startx` — скрипт Shell, являющийся надстройкой над командой `xinit`. Программа `xinit` позволяет запустить X-сервер и эмулятор терминала `xterm` (пример 32.19).

Пример 32.19. Запуск X-сервера с помощью xinit

```
$ xinit
```

Эта команда запустит X-сервер и эмулятор терминала — программу `xterm`.

Если в домашнем каталоге имеется файл `.xinitrc`, то команды для запуска X-клиентов, указанные в нем, будут выполнены вместо запуска программы `xterm` (пример 32.20).

Пример 32.20. Файл ~/.xinitrc

```
xclock -g 50x50-0+0 &  
twm
```

В примере 32.20 после запуска X-сервера автоматически стартуют два клиента: программа `xclock` (отображает системное время) и примитивный менеджер окон `twm`. Опция `-g` программы `xclock` (и многих других X-клиентов) указывает размер и положение окна программы на экране. Обратите внимание, что программа `xclock` запущена в фоновом режиме. В противном случае менеджер окон будет запущен лишь после завершения работы `xclock`.

Если файл `~/xinitrc` отсутствует, то по умолчанию запускается эмулятор терминала `xterm`. При запуске X-сервера программой `xinit` можно задать X-серверу требуемые опции. Для этого их прописывают в файле `~/xserverrc` (пример 32.21). В этом файле указывают имя программы X-сервера для его старта и опции.

Пример 32.21. Файл `~/xserverrc`

```
exec X :0.0 -nolisten tcp
```

Если такая строка будет присутствовать в файле `~/xserverrc`, то команда `xinit` запустит X-сервер в режиме без поддержки TCP-сетевых соединений.

В командной строке `xinit` можно задавать клиентское приложение для старта, а также указывать параметры запуска X-сервера (пример 32.22).

Пример 32.22. Передача параметров в командной строке `xinit`

```
$ xinit icewm -- X :0.0 -nolisten tcp
```

В этом примере будет запущен X-сервер без поддержки соединений по протоколу TCP, после чего будет запущен менеджер окон `icewm`.

Кроме команды `xinit`, запустить X-сервер позволяет также сценарий оболочки `startx`, являющийся надстройкой над `xinit` (пример 32.23).

Пример 32.23. Команда `startx`

```
$ startx
```

В результате работы сценария `startx` команда `xinit` запустит X-сервер.

Особенностью сценария `startx` по сравнению с `xinit` является то, что он позволяет сделать общесистемные файлы `xinitrc` и `xserverrc`. В GNU/Linux эти файлы располагаются обычно в каталоге `/etc/X11/xinit`.

ЗАДАНИЯ

- Проверьте, хватает ли ресурсов в вашей системе для запуска одновременно двух X-серверов.
- Испытайте запуск X-сервера из командной строки в режимах с прослушиванием TCP-порта и без него.
- Настройте файл `~/xinitrc` так, чтобы после старта X-сервера автоматически запускались оконный менеджер `twm` и эмулятор терминала `xterm`. Окно последнего должно находиться приблизительно в центре экрана и занимать около половины его площади.
- Изучите содержимое файла `/etc/X11/xinit/xinitrc`.

Менеджер X-сеанса *xdm*

Запуск X-сервера из командной строки не удобен тем, что после входа в обычный сеанс Shell приходится либо вручную вызывать команду `startx`, либо вызывать ее с помощью какого-либо сценария оболочки.

Менеджеры X-сессии самостоятельно запускают X-сервер и X-приложение, представляющее собой диалоговое окно для ввода имени пользователя и его пароля.

В GNU/Linux чаще всего используются три менеджера сеанса:

- ☐ `xdm` — поставляется вместе с X-сервером;
- ☐ `gdm` — поставляется с GNOME;
- ☐ `kdm` — в составе KDE.

Все менеджеры X-сеанса поддерживают специальный протокол XDMCP (X Display Manager Control Protocol). С помощью этого протокола менеджеры X-сеанса могут управлять X-дисплеем как на локальной, так и на удаленной машине.

Менеджеры сеанса обычно запускаются при переходе в многопользовательский режим, обеспечивая возможность входа в X-сеанс. В RH-подобных дистрибутивах и SUSE запуск менеджера X-сеанса производится на 5-м уровне исполнения, а в Debian — в обычном многопользовательском режиме на 2-м уровне исполнения.

Файл конфигурации `xdm` в GNU/Linux — `/etc/X11/xdm/xdm-config` (пример 32.24).

Пример 32.24. Файл конфигурации `xdm`

```
DisplayManager.errorLogFile:    /var/log/xdm.log
DisplayManager.pidFile:         /var/run/xdm.pid
DisplayManager.keyFile:         /etc/X11/xdm/xdm-keys
DisplayManager.servers:         /etc/X11/xdm/Xservers
DisplayManager.accessFile:      /etc/X11/xdm/Xaccess
DisplayManager.willing:         su nobody -c /etc/X11/xdm/Xwilling
DisplayManager*authorize:       true
! The following three resources set up display :0 as the console.
DisplayManager._0.setup:        /etc/X11/xdm/Xsetup_0
DisplayManager._0.startup:      /etc/X11/xdm/GiveConsole
DisplayManager._0.reset:        /etc/X11/xdm/TakeConsole
```



```
DisplayManager*resources:      /etc/X11/xdm/Xresources
DisplayManager*session:       /etc/X11/xdm/Xsession
DisplayManager*authComplain:   true
DisplayManager.requestPort:    0
```

В этом файле конфигурации для комментирования строк используется знак восклицания.

Наиболее важные настройки в файле конфигурации `xdm-config` указывают дополнительные файлы, настраивающие поведение `xdm` и X-сервера:

- ☐ `Xservers` — указывает X-сервер и его опции;
- ☐ `Xaccess` — файл управления доступом к `xdm` посредством протокола XDMCP;
- ☐ `Xsession` — указывает приложения, запускаемые после входа в X-сеанс;
- ☐ `Xresources` — настраивает параметры отображения (ресурсы) окна `xdm` и X-приложений, указанных в `Xsession`.

Пример 32.25. Файл `/etc/X11/xdm/Xservers`

```
:0 local /usr/X11R6/bin/X :0.0
```

В примере 32.25 строка конфигурации сообщает `xdm`, что он должен быть запущен на локальной машине, а также указывает команду для запуска X-сервера на локальной машине.

Файл `Xsession` представляет собой сценарий, запускаемый после успешной аутентификации пользователя `xdm`. Он предназначен для настройки окружения и запуска оконных менеджеров или рабочих окружений, а также программ, которые должны запускаться автоматически.

Настройки окон, связанных с работой `xdm` (например, диалогового окна для ввода имени пользователя и пароля) и приложений, запускаемых в `Xsession`, определяются в `Xresources`. Формат файла ресурсов будет описан в разд. "Ресурсы X-приложений" далее в этой главе.

Задания

- В файле `/etc/X11/xinit/Xresources` найдите строку ресурсов, определяющую строку приветствия, выводимую в окне приглашения `xdm`.
- Обычно в завершающей части сценария `/etc/X11/xinit/Xsession` содержатся команды для запуска либо GNOME — `gnome-session`, либо KDE — `startkde`. Используя регулярное выражение, проверьте это.
- Находясь на третьем уровне исполнения (в RH), запустите `xdm`.

Х-приложения

Многие Х-приложения поддерживают стандартные опции командной строки. Наиболее часто используют следующие опции:

- ☐ `-bg` — цвет фона окна Х-приложения;
- ☐ `-fg` — цвет текста по умолчанию в окне приложения;
- ☐ `-bd` — цвет обрамления окна;
- ☐ `-rv` — инверсия цветового оформления;
- ☐ `-bw` — толщина обрамления в пикселах;
- ☐ `-display` — указывает, на каком узле какие Х-сервер и экран используются для вывода окна Х-приложения;
- ☐ `-fn` — основной шрифт приложения;
- ☐ `-geometry` — расположение и размер окна Х-приложения;
- ☐ `-iconic` — запуск приложения в свернутом режиме без раскрытого окна;
- ☐ `-name` — указывает имя приложения для поиска его ресурсов, если исполняемый файл имеет другое имя, чем указано в файле ресурсов;
- ☐ `-title` — строка заголовка окна Х-приложения;
- ☐ `-xnlanguage` — указывает кодировку (например, `ru_RU.KOI8-R`);
- ☐ `-xrm` — определяет имя ресурсов для приложения в файле ресурсов.

В качестве примера Х-приложений можно привести эмуляторы Х-терминалов, в изобилии имеющиеся в GNU/Linux. Графические эмуляторы терминала — Х-приложения, предназначенные для работы в командной строке в Х-сессии.

Наиболее популярные эмуляторы терминала:

- ☐ `xterm` — стандартный эмулятор терминала, поставляющийся в Xorg;
- ☐ `rxvt` — проще `xterm`, зато экономно расходует ресурсы системы;
- ☐ `aterm` — базируется на `rxvt` и предоставляет расширенные возможности;
- ☐ `gnome-terminal` — поставляется в составе GNOME и обладает удобной возможностью использования вкладок, позволяющих получать доступ к нескольким оболочкам Shell без открытия новых окон;
- ☐ `konsole` — используется в KDE и также позволяет создавать вкладки.

Для запуска `xterm` с заданными размерами окна, темно-синим цветом фона и светло-синим цветом шрифта, следует использовать команду, показанную в примере 32.26.

Пример 32.26. Опции X-клиентов

```
$ xterm -bg navy -fg cyan -geometry 100x40+20+10 &
```

Амперсанд, установленный в конце командной строки, запускает `xterm` в фоновом режиме. Если этого не сделать, командная строка будет доступна в окне, из которого произведен вызов, только после остановки `xterm`.

В некоторых случаях встречаются зависшие X-приложения, окна которых, возможно, не отображаются на экране вообще. Тем не менее, эти "невидимые" X-приложения потребляют ресурсы системы и замедляют работу других приложений. Для идентификации таких приложений удобно воспользоваться командой `ps` с опцией `-u`, позволяющей получить список процессов, запущенных заданным пользователем. Далее приведен пример 32.27 фрагмента вывода команды `ps tree`, выполненной в эмуляторе терминала `gnome-terminal` рабочего окружения GNOME.

Пример 32.27. Иерархия X-процессов (фрагмент)

```
$ ps tree -Aap 3371
gnome-terminal,3371
|-bash,3373
|  `--man,6360 ps tree
|      `--less,6368
|-bash,6383
|  `--ps tree,6437 -Aap 3371
|-gnome-pty-helper,3372
`--{gnome-terminal},3375
```

ЗАДАНИЯ

- Запустите `xterm` с розовым фоном и желтым цветом шрифта.
- Реагирует ли `gnome-session` на опцию командной строки `-bg`?

Шрифты

Все шрифты принято разделять на три категории:

- ☐ шрифты без засечек (Sans Serif), например, Helvetica и Lucida;
- ☐ с засечками (Serif), например, Courier и Times;
- ☐ специальные шрифты, например, Symbol.

Известно, что шрифты, в которых для различных букв используется своя, наилучшая для данного символа, ширина, читаются лучше.

По критерию постоянства ширины символов шрифты подразделяются на:

- ☐ пропорциональные, классический пример которых — Times;
- ☐ моноширинные, например, Courier. В таких шрифтах размер символов одинаков, что позволяет легко читать тексты программ.

В X Window используется четырнадцать различных (иногда связанных друг с другом) характеристик шрифтов, задающих имя шрифта:

- ☐ Foundry — обладатель прав на данный шрифт;
- ☐ Family Name — имя типа шрифта;
- ☐ Weight Name — толщина линий: Medium — обычные символы, Bold или Demibold — жирные символы;
- ☐ Slant — наклон: r — обычные символы (regular), i — курсив (italic), o — с наклоном (oblique);
- ☐ Setwidth Name — плотность расположения символов: normal — обычная плотность, semicondensed и condensed — уплотненное расположение символов;
- ☐ Add Style Name — стиль шрифта (как правило, не указывается);
- ☐ Pixel size — величина символов в пикселах;
- ☐ Point size — величина символов в десятках типографских пунктов;
- ☐ Resolution X — разрешение экрана по горизонтали в пунктах на дюйм;
- ☐ Resolution Y — разрешение экрана по вертикали в пунктах на дюйм;
- ☐ Spacing — для пропорциональных шрифтов p, m — для моноширинных;
- ☐ Average Width — средняя ширина символа;
- ☐ Charset Registry — алфавит шрифта;
- ☐ Charset Encoding — кодировка шрифта.

Имя шрифта указывается в виде строки, содержащей необходимое число из указанных выше четырнадцати параметров, разделенных тире. Параметры, которые не надо указывать, должны быть заменены звездочками (пример 32.28).

Пример 32.28. Указание шрифта

```
--symbol-***-***-240-**-***-***
```

Здесь указан шрифт Symbol с размером символов 24 типографских пункта.

В случае если под спецификацию шрифта подходит более одного шрифта, то X Window использует первый шрифт из подходящих. X Window способна подбирать наиболее подходящий шрифт на основании имени шрифта.

Утилита `xfontsel` позволяет выбрать требуемый шрифт, действуя по аналогии с фильтром, в котором можно указывать необходимые характеристики шрифтов, получая список тех шрифтов, которые удовлетворяют критериям фильтрации. Команда `xlsfonts -fn` выводит на экран имена шрифтов, удовлетворяющие условию фильтрации (пример 32.29).

Пример 32.29. Фильтрация шрифтов

```
$ xlsfonts -fn *-lucida*-i-*-12-*-**-*-koi8-*  
-b&h-lucida-bold-i-normal-sans-12-120-75-75-p-79-koi8-r  
-b&h-lucida-bold-i-normal-sans-12-120-75-75-p-79-koi8-r  
-b&h-lucida-medium-i-normal-sans-12-120-75-75-p-71-koi8-r  
-b&h-lucida-medium-i-normal-sans-12-120-75-75-p-71-koi8-r
```

В примере 32.29 выведены все шрифты, установленные в системе, с именем `lucida`, с наклонным начертанием, размером 12 пикселей и с кодировкой KOI8-R.

С помощью опции `-fn` можно указать шрифт, который должен использоваться по умолчанию X-приложением (пример 32.30).

Пример 32.30. Запуск X-клиента с требуемым шрифтом

```
xterm -fn *-courier-bold-*-90-*-**-*-cyr-*
```

В окне эмулятора терминала будет использован кириллический шрифт Courier с жирным начертанием символов и величиной 90 пунктов.

ЗАДАНИЯ

- Используя `xfontsel` и `xlsfonts`, получите все русские шрифты в системе.
- Запустите программу `xterm` с любым русским шрифтом наклонного начертания (*italic*) размером 140 пунктов.

Ресурсы X-приложений

Внешний вид X-приложений, написанных для использования в составе рабочих сред KDE и GNOME, может быть настроен с помощью специальных программ, входящих в состав этих сред. Настройки, связанные с этими приложениями, хранятся в специальном образе отформатированных файлов.

X-приложения, написанные с использованием библиотеки X Toolkit, для настройки внешнего вида (т. е. например, шрифтов и цветов) используют так называемые *файлы ресурсов*.

Один из этих файлов — Xresources, определяющий внешний вид менеджера сеанса xdm и связанных с ним приложений.

Общесистемные ресурсы для X-приложений определяются в каталоге /etc/X11/app-defaults в файлах с именами, соответствующими именам X-приложений. При этом лидирующую строчную букву "x" в имени X-приложений в именах файлов ресурсов, соответствующих этим приложениям, заменяют прописной буквой "X". Так, приложению xterm сопоставлен файл ресурсов /etc/X11/app-defaults/Xterm.

Пользователи могут создавать индивидуальные файлы ресурсов ~/.Xresources, содержащие строки настроек оформления внешнего вида X-приложений.

Для исследования влияния файлов ресурсов на X-приложения, написанные с помощью X Toolkit, можно запустить простую программу xlogo, выводящую на экран окно с изображением логотипа X Window. Если же запустить эту программу следующим образом: `xlogo -fg red`, то цвет стилизованной буквы "X" на логотипе будет красным. Этого же эффекта можно добиться, добавив в файл индивидуальных настроек ресурсов ~/.Xresources строку, показанную в примере 32.31.

Пример 32.31. Задание X-ресурса для приложения

```
XLogo*Foreground: red
```

Для того чтобы индивидуальные настройки ресурсов стали известны X-серверу, необходимо выполнить команду `xrdb -merge` (пример 32.32).

Пример 32.32. Подключение X-ресурсов

```
$ xrdb -merge ~/.Xresources
```

После этого при запуске программы xlogo цвет логотипа будет красным.

При указании строк ресурсов для X-приложений нужно учитывать следующее:

- ☐ объекты, предоставляемые библиотекой X Toolkit, выстроены иерархически, причем верхним объектом в иерархии является само X-приложение;
- ☐ по соглашению первая буква в имени приложения пишется заглавной (например, для редактора emacs имя объекта верхнего уровня иерархии — Emacs);

□ для приложений, имена которых начинаются с "x", две первые буквы пишутся прописными (например, для `xterm` — `XTerm`).

Строки комментариев начинаются со знака восклицания (!), а символ решетки (#) — для подключения внешнего файла ресурсов. После символа двоеточия (:) указывают строку значения данного ресурса.

Для разделения объектов в иерархии X-приложения используется символ точки (.). Также может быть использован символ звездочки (*), позволяющий пропустить несколько уровней в иерархии объектов X-приложения.

Задания

- Найдите страницу помощи `man`, описывающую ресурсы X-приложений.
- Настройте файл ресурсов `~/.Xresources` так, чтобы фон `xterm` был темно-синего цвета, а цвет шрифта — желтый.
- Настройте по умолчанию русский шрифт с размером 140 пунктов для `xterm`.
- Введите в индивидуальный файл ресурсов запись, устанавливающую удобный для вас заголовок окна `xterm`.

Удаленный запуск X-приложений

Поскольку система X Window построена на технологии "клиент — сервер", то X-сервер, выполняющийся на некотором компьютере, способен обслуживать X-приложения, выполняющиеся на других компьютерах. Это, в частности, позволяет запускать клиентские X-приложения, требовательные к ресурсам, на мощных компьютерах, отображая при этом графику с помощью X-сервера, запущенного на рабочей станции.

Один из важнейших моментов во взаимодействии X-сервера с удаленными X-клиентами состоит в авторизации X-клиентов, т. е. в предоставлении прав X-клиентам использовать данный X-сервер. В простейшем случае при отсутствии необходимости идентификации пользователей, работающих на удаленных компьютерах, авторизация X-клиентов может быть произведена с помощью программы `xhost`. Вызванная без аргументов, эта команда отображает список имен узлов, с которых разрешен запуск X-приложений на данном сервере (пример 32.33).

Пример 32.33. Команда `xhost`

```
$ xhost
```

```
access control enabled, only authorized clients can connect
```

В этом случае команда `xhost` сообщила, что контроль доступа включен. Так как данная команда не отобразила ни одного имени узла, с которого разрешено взаимодействие X-клиентов, то данный X-сервер не будет отображать графику удаленных приложений. Для разрешения доступа X-приложений отовсюду следует выполнить команду `xhost +`. Эта команда выключает контроль доступа к X-серверу. Для его восстановления необходимо выполнить команду `xhost -`.

Используя команду `xhost`, можно указывать имена узлов, которые авторизованы для запуска X-приложений, взаимодействующих с данным X-сервером (пример 32.34).

Пример 32.34. Разрешение на доступ к X-серверу указанному хосту

```
$ xhost +classfw
classfw being added to access control list
$ xhost
access control enabled, only authorized clients can connect
INET:classfw.class.edu
```

В этом примере в список авторизованных узлов внесен узел `classfw`. Его имя указано в файле `/etc/hosts`. Если имя узла не указано в этом файле, то программа `xhost` сделает попытку обращения к серверу DNS.

Более безопасно воспользоваться средствами авторизации, основанными на шифрованных дайджестах. Такие возможности предоставляет программа `xauth`, позволяющая обмениваться такими дайджестами (MIT-MAGIC-COOKIE-1). Команда, приведенная в примере 32.35, передает дайджест с компьютера `xsrv` на компьютер `xclnt`. В результате этого, компьютер `xclnt` получает доступ к X-серверу на компьютере `xsrv`.

Пример 32.35. Команда `xauth`

```
[user1@xsrv]$ xauth generate xsrv:0
[user1@xsrv]$ xauth list xsrv:0
xsrv/unix:0 MIT-MAGIC-COOKIE-1  20aace2a7dfb831f4e2a7828c4f2f7b8
[user1@xsrv]$ xauth extract - xsrv:0 | ssh xclnt xauth merge -
```

Приведенная в примере 32.36 команда `xauth generate` генерирует дайджест, который формируется из списка, выданного командой `xauth list`. Последняя команда примера пересылает дайджест посредством SSH на клиентский компьютер.

Запуск X-приложений, взаимодействующих с удаленным X-сервером, требует указания узла, на котором запущен X-сервер. Это может быть осуществлено с помощью указания требуемого узла после опции X-приложений `-display` (пример 32.36).

Пример 32.36. Запуск X-клиента, работающего с удаленным X-сервером

```
$ xterm -display bamboo.class.edu:0.0 &
```

В этом примере X-приложение `xterm`, запущенное на узле `classfw`, будет отображать свои окна в X-сервере, запущенном на узле `bamboo.class.edu`.

Вместо указания имени узла с помощью опции `-display` можно воспользоваться переменной окружения `DISPLAY`, которой назначается имя дисплея (пример 32.37).

Пример 32.37. Переменная `DISPLAY`

```
$ DISPLAY="bamboo.class.edu:0.0"
$ export DISPLAY
$ xterm &
```

После присвоения переменной окружения `DISPLAY` значения — имени удаленного узла X-приложения будут пытаться взаимодействовать с X-сервером на этом узле.

Наиболее безопасно использовать удаленные X-сессии, туннелируя их посредством SSH. Туннелирование SSH разрешает настройка в файле конфигурации `/etc/ssh/ssh_config` на компьютере с X-сервером, приведенная в примере 32.38.

Пример 32.38. Разрешение туннелирования X11 в `/etc/ssh/ssh_config`

```
X11Forwarding yes
```

Вместо редактирования конфигурационного файла можно просто использовать опцию `-X` команды `ssh` (пример 32.39).

Пример 32.39. Туннель X11 в SSH

```
workstation$ ssh -X dspless.class.edu netbeans
```

Команда в примере 32.39 запустит на машине `dsplss.class.edu` приложение `netbeans`, графические окна которого будут отображаться на дисплее рабочей станции `workstation`. При этом пакеты протокола X11 будут туннелированы через SSH.

Задания

- Отключите контроль доступа к X-серверу и запустите программу `xterm` на удаленном компьютере так, чтобы ее окно отображалось с помощью этого X-сервера, используя опцию `-display`.
- Сделайте то же с помощью переменной окружения `DISPLAY`. Проверьте, на каком компьютере работает процесс `xterm`.
- Запретите для какого-либо удаленного узла доступ к X-серверу и попытайтесь запустить на нем X-приложение, взаимодействующее с этим X-сервером.

Использование `xdm` для удаленного входа в сеанс

Менеджеры сеанса `xdm`, `gdm` и `kdm` обеспечивают возможность удаленного входа в X-сеанс с помощью протокола XDMCP (порт 177 UDP). Менеджер `xdm` по умолчанию запрещает удаленный вход в X-сеанс, т. к. обращение к порту 177 UDP запрещено с помощью установки в файле `/etc/X11/xdm/xdm-config` (пример 32.40).

Пример 32.40. Запрет на работу XDMCP

```
! SECURITY: do not listen for XDMCP or Chooser requests
! Comment out this line if you want to manage X terminals with xdm
DisplayManager.requestPort:      0
```

Настройка `DisplayManager.requestPort: 0` запрещает обращение к `xdm` с удаленных узлов. Для разрешения такого обращения эта настройка должна быть закомментирована (используйте для комментирования знак восклицания).

Если на удаленной системе запущен сервер XDMCP, например, программа `xdm`, то обратиться к ней можно, запустив X-сервер одним из следующих вариантов:

- ❑ `X -query <сервер>` — в этом случае локальная система будет использоваться в качестве X-терминала для удаленного входа на указанную систему;

- ❑ `X -broadcast` — здесь будут использованы широковещательные запросы для поиска доступных серверов, причем будет использован первый ответивший;
- ❑ `X -indirect <сервер>` — в этом случае предполагается обращение к серверу XDMCP, предоставляющему список систем, на которые может быть осуществлен удаленный вход в X-сессию (chooser).

Так, для прямого обращения к серверу XDMCP на узле `susel.ritter.org` следует использовать команду, показанную в примере 32.41.

Пример 32.41. Обращение к XDMCP-серверу на удаленной машине

```
$ X -query susel.ritter.org
```

Если сервер позволяет производить косвенные обращения к серверам XDMCP, то должна быть выполнена команда, показанная в примере 32.42.

Пример 32.42. Косвенное обращение к серверам XDMCP

```
$ X -indirect susel.ritter.org
```

В результате будет получен список из доступных серверов XDMCP, выбрав один из которых можно будет войти в X-сеанс на выбранном удаленном узле.

Разрешение на доступ к XDMCP задают в файле `/etc/X11/xdm/Xaccess`.

Если к программе `xdm` разрешается обращаться с любого узла, то в этом файле достаточно указать символ звездочки — шаблон для любых узлов: `*`.

Можно ограничить использование `xdm`, указав в `/etc/X11/xdm/Xaccess` домен, узлы которого имеют право использовать этот сервер XDMCP (пример 32.43).

Пример 32.43. Разрешение использовать XDMCP для узлов домена

```
*.ritter.org
```

Например, если для любых узлов из домена `ritter.org` разрешено обращаться к `xdm` с непрямыми запросами, причем в списке доступных серверов должны отображаться лишь узлы `susel.ritter.org` и `ejick.ritter.org`, то в таком случае в файл `/etc/X11/xdm/Xaccess` следует внести строку, показанную в примере 32.44.

Пример 32.44. Указание доступных серверов XDMCP

```
*.ritter.org  CHOOSEsusel.ritter.org ejick.ritter.org
```

Можно также строить список доступных серверов XDMCP, используя широко-вещательные запросы. Так, если требуется разрешить для любых узлов обращаться к любым доступным серверам XDMCP, используя не прямые запросы, то в `/etc/X11/xdm/Xaccess` необходимо внести директиву `CHOOSEBROADCAST`.

Задания

- Настройте сервер `xdm` для обеспечения прямых удаленных запросов к нему из домена `class.edu`. Проверьте возможность удаленного входа в X-сеанс.
- Разрешите для любых узлов использование широковещательных запросов.



ПРИЛОЖЕНИЯ

Приложение 1



Работа с VMWare Workstation и Sun VirtualBox

В этом приложении описаны подготовительные действия для установки GNU/Linux на виртуальных машинах.

Создание виртуальной машины в Sun VirtualBox

Программа Sun VirtualBox позволяет запускать на компьютерах с 32- и 64-битными процессорами Intel и AMD виртуальные машины с гостевыми операционными системами. Sun VirtualBox является гипервизором с поддержкой аппаратной виртуализации. Он может работать в MS Windows, множестве дистрибутивов GNU/Linux, Sun Solaris и OpenSolaris, Mac OS X. Его достоинством является широкая поддержка гостевых ОС, а также несомненная простота и наличие качественной документации.

Получить Sun VirtualBox можно на сайте **www.virtualbox.org**. По лицензии GPLv2 распространяется продукт VirtualBox OSE.

Далее представлен процесс создания виртуальной машины для установки на нее Ubuntu 9.10. В примере основная ОС — MS Windows XP. Под другими ОС установка виртуальной машины идентична.

На рис. П1.1 изображено окно Sun VirtualBox. В левой части окна находится список уже созданных виртуальных машин, в правой — настройки выделенной виртуальной машины. Для создания новой виртуальной машины можно либо щелкнуть мышью на кнопке **Создать**, либо воспользоваться сочетанием клавиш <Ctrl>+<N>, либо сделать это в меню **Машина**.

Диалоговое окно для создания новой виртуальной машины показано на рис. П1.2. Здесь вы должны ввести имя виртуальной машины, тип гостевой ОС и ее разновидность из списка.

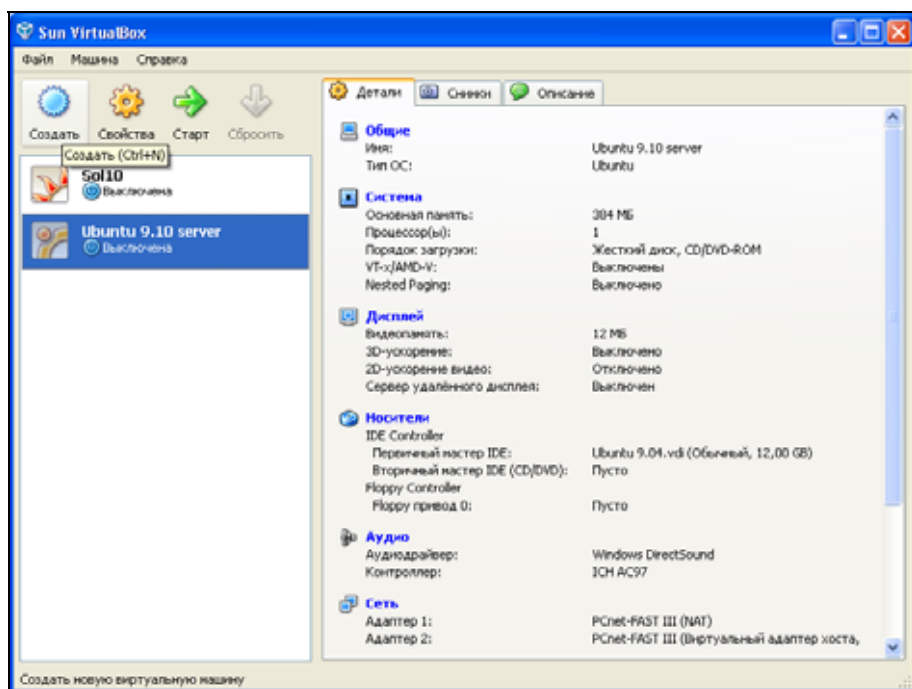


Рис. П1.1. Основное окно Sun VirtualBox

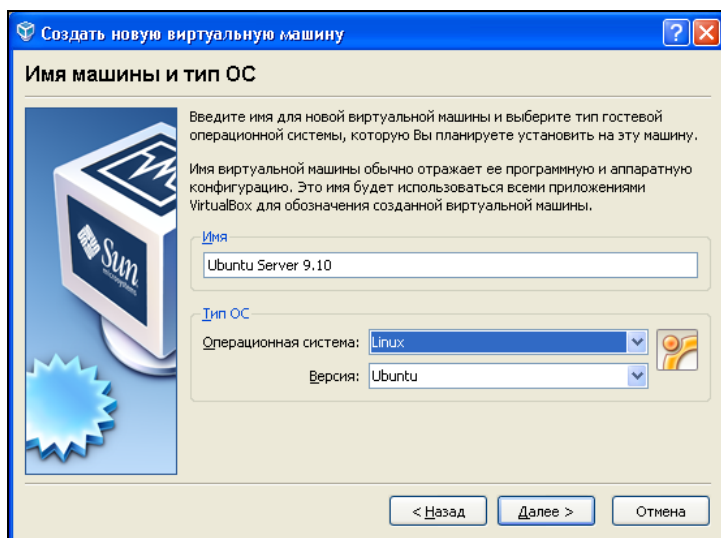


Рис. П1.2. Создание новой виртуальной машины

Следующее диалоговое окно (рис. П1.3) предназначено для определения объема ОЗУ, которое займет виртуальная машина. При этом объем ОЗУ, переданный для работы виртуальной машине, не будет доступен основной операционной системе. Не рекомендуется выдавать виртуальной машине более 50% объема ОЗУ.

Далее будет запущен диалог создания виртуального жесткого диска (рис. П1.4).

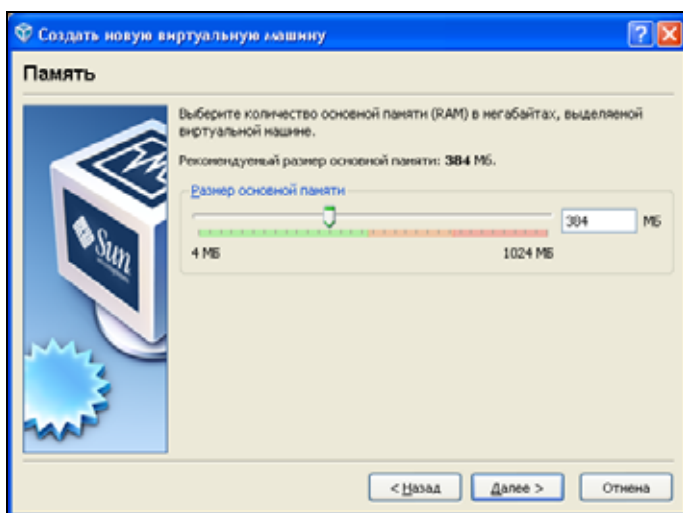


Рис. П1.3. Выделение памяти для виртуальной машины

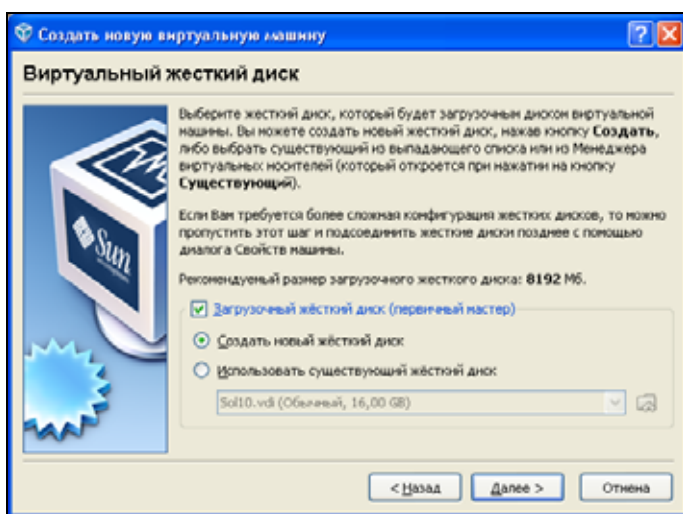


Рис. П1.4. Создание нового виртуального жесткого диска

В следующем диалоговом окне (рис. П1.5) можно выбрать создание динамически расширяющегося или фиксированного образа жесткого диска.

Следующее диалоговое окно позволяет указать размер образа жесткого диска и выбрать его местоположение (рис. П1.6).

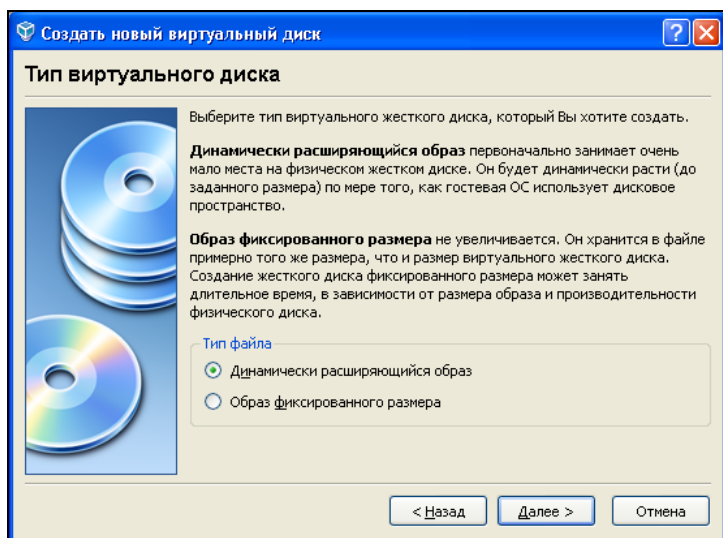


Рис. П1.5. Создание новой виртуальной машины

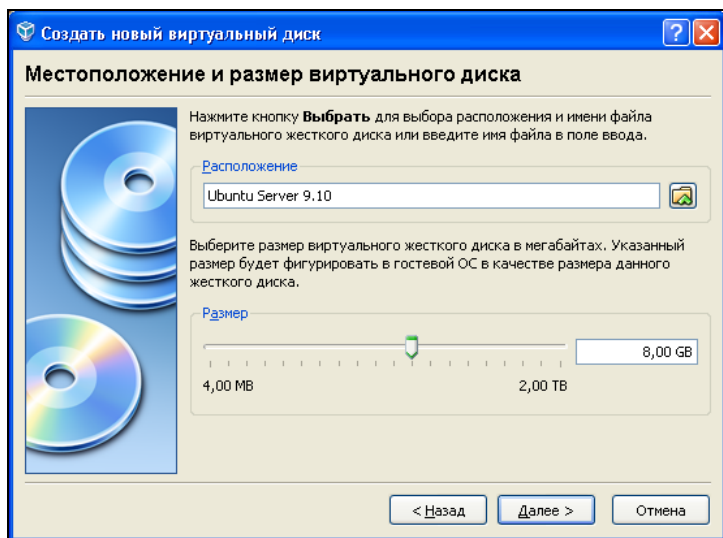


Рис. П1.6. Определение размера и местоположения образа жесткого диска

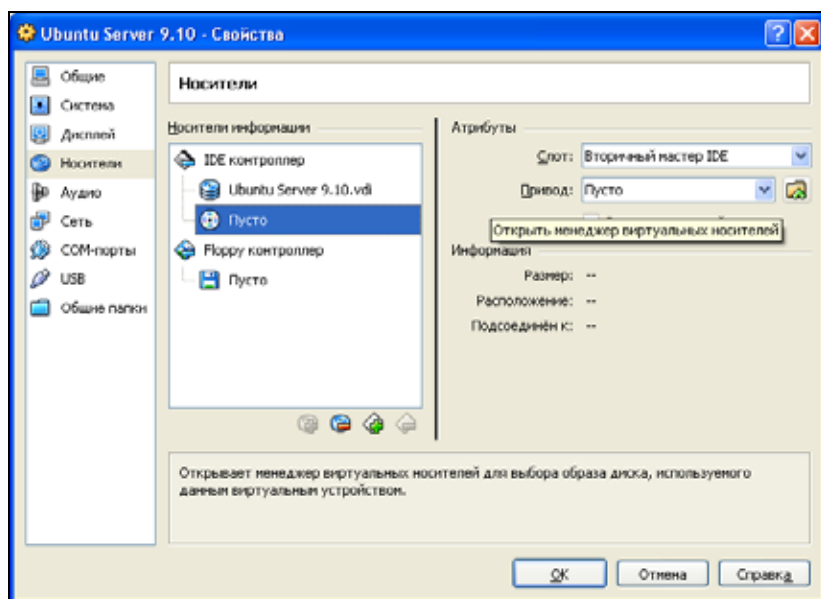


Рис. П1.7. Диалог управления носителями

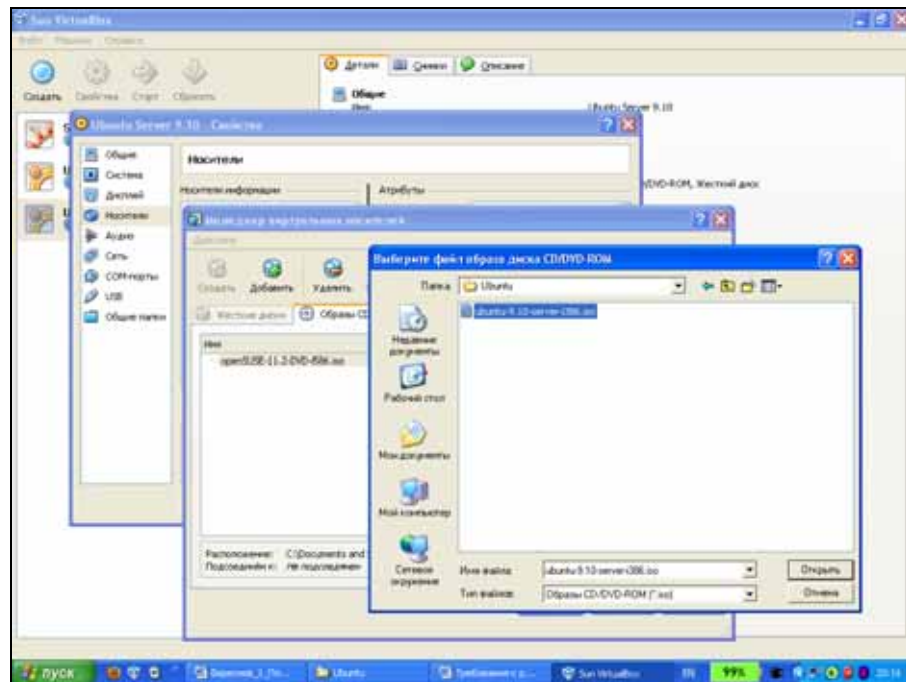


Рис. П1.8. Выбор местоположения образа установочного диска

К этому моменту виртуальная машина уже создана, но нам необходимо подключить либо физический CD, либо образ дистрибутивного диска для установки Ubuntu. На рис. П1.7 показано диалоговое окно настройки носителей. В него можно попасть, щелкнув мышью в основном окне Sun VirtualBox на настройке **Носители** в правой части окна.

В этом диалоговом окне можно открыть менеджер виртуальных носителей, в котором следует указать местоположение образа дистрибутивного диска Ubuntu (рис. П1.8).

После этого виртуальная машина готова к работе и можно приступать к установке Ubuntu. Для этого просто нажмите кнопку **Старт**.

Создание виртуальной машины в VMWare Workstation

Аналогично создается виртуальная машина в VMWare Workstation. Здесь продемонстрировано использование VMWare в среде MS Windows Vista.

Основной экран VMWare показан на рис. П1.9.

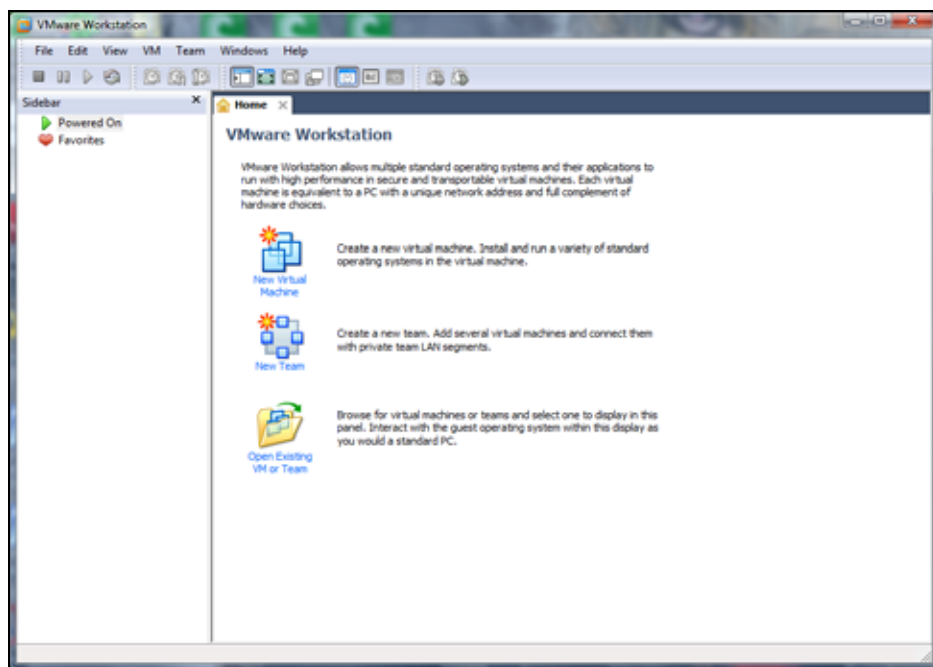


Рис. П1.9. Основной экран VMWare Workstation

Для создания новой виртуальной машины достаточно щелкнуть на значке **New Virtual Machine**. При этом будет запущен мастер создания новой виртуальной машины, первое диалоговое окно которого показано на рис. П1.10.

Следующее диалоговое окно позволяет подключить физический CD, образ CD или же сделать это позже (рис. П1.11).

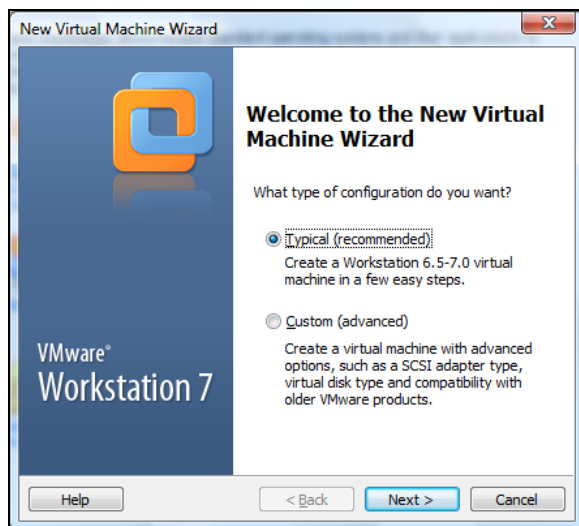


Рис. П1.10. Выбор стандартных или нестандартных установок новой виртуальной машины

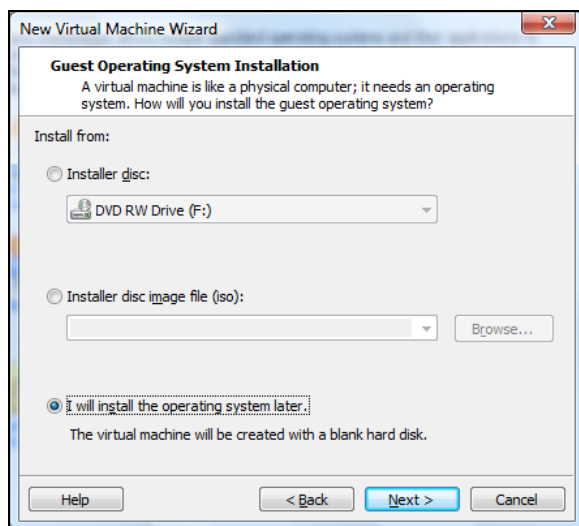


Рис. П1.11. Подключение установочного диска

Далее необходимо указать гостевую операционную систему (рис. П1.12).
В следующем окне надо задать имя машины и ее расположение (рис. П1.13).

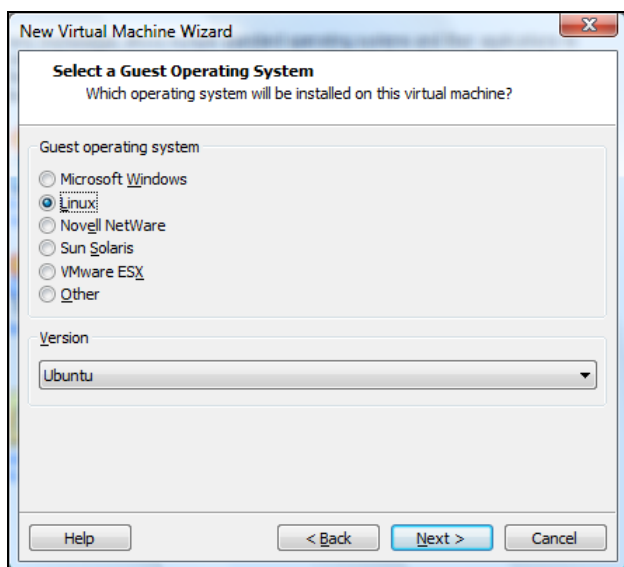


Рис. П1.12. Выбор гостевой ОС

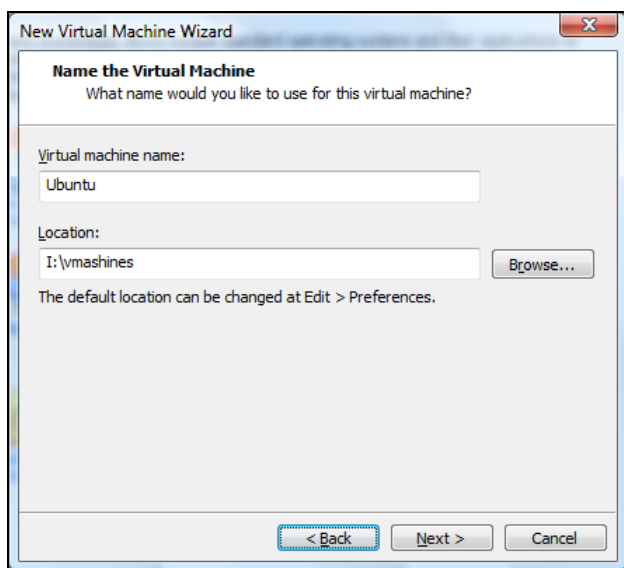


Рис. П1.13. Выбор имени и местоположения виртуальной машины

Размер виртуального жесткого диска задается в следующем окне (рис. П1.14). Далее выводится итоговая информация (рис. П1.15).

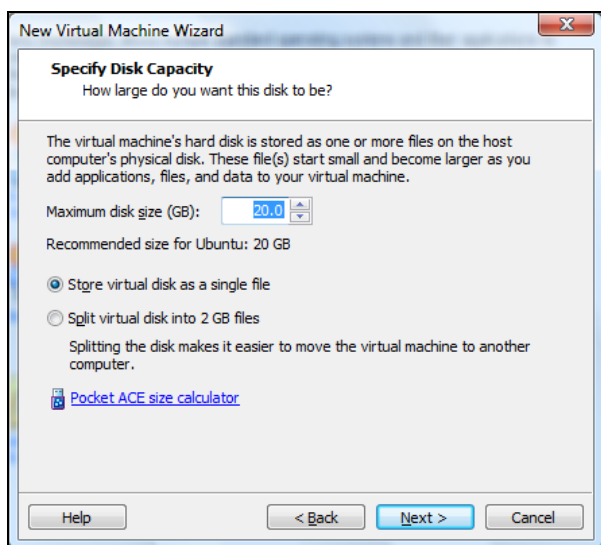


Рис. П1.14. Определение размера виртуального жесткого диска

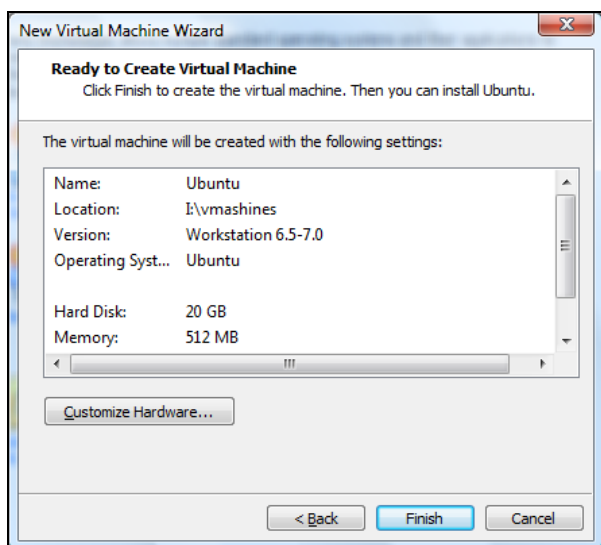


Рис. П1.15. Итоговая информация

Виртуальная машина готова к установке гостевой операционной системы (рис. П1.16).

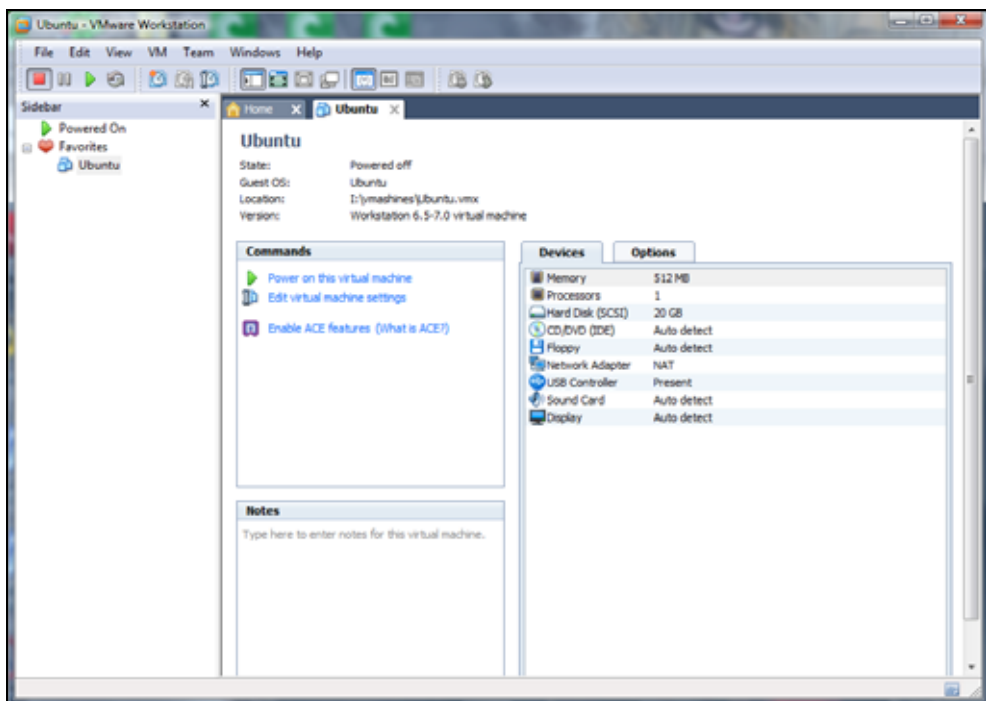


Рис. П1.16. Виртуальная машина готова к запуску

Приложение 2



Примеры использования текстовых утилит GNU

В этом приложении приводятся примеры использования текстовых утилит GNU.

Копирование с помощью команды *tee*

Команда *tee* копирует содержимое стандартного потока ввода в поток вывода и в файлы, заданные в качестве аргументов без каких-либо изменений. Этим можно воспользоваться для создания множества одинаковых копий файла. При этом вовсе не обязательно, чтобы файл был текстовым. В примере П2.1 показано, как создать несколько копий бинарного файла.

Пример П2.1. Команда *tee*

```
$ tee ls1 ls2 ls3 ls4 > /dev/null < /bin/ls
```

```
$ sha256sum ls? /bin/ls
```

```
6fde6afddce5f6a05379b67d14073cd900ac7ab4a4c5280e54813e478dc45b3d  ls1
```

```
6fde6afddce5f6a05379b67d14073cd900ac7ab4a4c5280e54813e478dc45b3d  ls2
```

```
6fde6afddce5f6a05379b67d14073cd900ac7ab4a4c5280e54813e478dc45b3d  ls3
```

```
6fde6afddce5f6a05379b67d14073cd900ac7ab4a4c5280e54813e478dc45b3d  ls4
```

```
6fde6afddce5f6a05379b67d14073cd900ac7ab4a4c5280e54813e478dc45b3d  /bin/ls
```

```
$ chmod u+x ls?
```

```
$ ./ls1 -Fl
```



```
total 592
-rwxr--r-- 1 user1 users 100584 Jan 31 20:05 ls1*
-rwxr--r-- 1 user1 users 100584 Jan 31 20:05 ls2*
-rwxr--r-- 1 user1 users 100584 Jan 31 20:05 ls3*
-rwxr--r-- 1 user1 users 100584 Jan 31 20:05 ls4*
```

В примере П2.1 команда `tee` получила четыре аргумента: `ls1`, `ls2`, `ls3` и `ls4`. В эти файлы скопировано бинарное содержимое файла `/bin/ls`, которое поступило в стандартный поток ввода команды `tee`. Стандартный поток вывода этой команды был перенаправлен в файл устройства `/dev/null` для исключения вывода бинарного содержимого на экран. Получившиеся бинарные файлы имеют одинаковое содержание и являются точными копиями файла `/bin/ls`, что подтверждается проверкой дайджеста SHA256. Для проверки работоспособности файл `ls1` сделан исполняемым. Его работа проверена последней командой примера.

Нумерация строк с помощью команды `cat`

Команда `cat` помимо того, что может объединять содержимое текстовых или бинарных файлов, выводя их в стандартный поток вывода, может еще нумеровать строки текстовых файлов. Для этого предназначены опции `-b` — нумеровать непустые строки и `-n` — нумеровать строки подряд (пример П2.2).

Пример П2.2. Команда `cat`

```
$ cat -b /etc/issue
 1 Welcome to openSUSE 11.2 "Emerald" - Kernel \r (\l).

$ cat -n /etc/issue
 1 Welcome to openSUSE 11.2 "Emerald" - Kernel \r (\l).
 2
 3
```

В примере команда `cat` вывела содержимое одного и того же файла, содержащего пустые строки. При использовании опции `-b` пустые строки не были пронумерованы.

Пример П2.3 показывает, как можно получить нумерованный список файлов.

Пример П2.3. Нумерованный список файлов

```
$ ls *.gz | cat -n
 1 fsck.txt.gz
 2 ls.txt.gz
 3 mkfs.txt.gz
 4 mount.txt.gz
 5 paste.txt.gz
 6 ps.txt.gz
 7 sfdisk.txt.gz
 8 users.txt.gz
```

В примере П2.3 получен нумерованный список файлов с суффиксом `.gz`.

Нумерация строк с помощью команды `nl`

Команда `nl` обладает более широкими возможностями нумерации строк. Например, имеется возможность пронумеровать лишь строки, удовлетворяющие регулярному выражению. Пример П2.4 показывает, как можно пронумеровать только те строки, у которых перед первой точкой от начала строки стоит лишь два символа.

Пример П2.4. Нумерация строк, удовлетворяющих регулярному выражению

```
$ ls *.gz | nl -bp'^..\.'
```

```
      fsck.txt.gz
1     ls.txt.gz
      mkfs.txt.gz
      mount.txt.gz
      paste.txt.gz
2     ps.txt.gz
      sfdisk.txt.gz
      users.txt.gz
```

Опция `-b` команды `nl` устанавливает режим нумерации строк. Стиль `p` нумерации обозначает использование регулярного выражения.

Команда `csplit`

Команда `csplit` делит на части текстовые файлы, основываясь на поиске строк, удовлетворяющих заданному регулярному выражению. По умолчанию

файл делится на две части: в первой части — все строки до вхождения строки, удовлетворяющей регулярному выражению, и во второй части — все остальные строки, начиная с той, которая удовлетворила искомому регулярному выражению. Имеется, однако, возможность продолжить поиск либо заданное количество раз, либо повторить его столько раз, сколько раз встречается строка, удовлетворяющая регулярному выражению (пример П2.5).

Пример П2.5. Деление текстового файла на части

```
$ cat sfdisk.txt
```

```
# sfdisk -l
```

```
Disk /dev/sda: 19457 cylinders, 255 heads, 63 sectors/track
```

```
Units = cylinders of 8225280 bytes, blocks of 1024 bytes, counting from 0
```

Device	Boot	Start	End	#cyls	#blocks	Id	System
/dev/sda1		0+	1073-	1074-	8624128	27	Unknown
end: (c,h,s) expected (1023,254,63) found (1023,239,63)							
/dev/sda2		1073+	8854-	7782-	62501953+	7	HPFS/NTFS
start: (c,h,s) expected (1023,254,63) found (1023,239,63)							
/dev/sda3	*	8855	19456	10602	85160565	5	Extended
/dev/sda4		0	-	0	0	0	Empty
/dev/sda5		19083+	19456	374-	3004123+	82	Linux swap / Solaris
/dev/sda6		8855+	12510	3656-	29366757	83	Linux
/dev/sda7		12511+	19082	6572-	52789558+	83	Linux

```
$ csplit sfdisk.txt /sda[0-9]/ {*}
```

```
209
```

```
120
```

```
124
```

```
63
```

```
60
```

```
75
```

```
60
```

```
60
```

```
$ ls xx??
```

```
xx00 xx01 xx02 xx03 xx04 xx05 xx06 xx07
```

```
$ cat xx06
```

```
/dev/sda6      8855+ 12510      3656- 29366757      83 Linux
```

В примере П2.5 выведено содержимое текстового файла `sfdisk.txt`, содержащего пример работы команды `sfdisk -l`. Этот файл был разделен на части с помощью команды `csplit`. Она разделила файл на столько частей, сколько встретилась искомая строка, удовлетворяющая регулярному выражению `sda[0-9]`. В результате образовались файлы `xx00`, ..., `xx07`, содержимое предпоследнего файла показано.

Команда `sed`

Потоковый редактор `sed` позволяет выполнять довольно сложное неинтерактивное редактирование текста. В документации на `sed` приведено множество полезных примеров применения `sed` (см. `info sed`). В примере П2.6 показано, как с помощью `sed` инвертировать порядок следования строк файла.

Пример П2.6. Инвертирование порядка строк файла с помощью команды `sed`

```
$ cat ldd.txt
linux-gate.so.1 => (0xfffffe000)
libc.so.6 => /lib/libc.so.6 (0xb767b000)
/lib/ld-linux.so.2 (0xb7813000)

$ sed -n -e '1! G' -e '$ p' -e 'h' ldd.txt
/lib/ld-linux.so.2 (0xb7813000)
libc.so.6 => /lib/libc.so.6 (0xb767b000)
linux-gate.so.1 => (0xfffffe000)
```

Команда `tac`

То же самое, но проще, можно сделать командой `tac` — она выводит содержимое файла в обратном порядке (пример П2.7).

Пример П2.7. Инвертирование порядка строк файла с помощью команды `tac`

```
$ cat ldd.txt
linux-gate.so.1 => (0xfffffe000)
libc.so.6 => /lib/libc.so.6 (0xb767b000)
/lib/ld-linux.so.2 (0xb7813000)

$ tac ldd.txt
/lib/ld-linux.so.2 (0xb7813000)
libc.so.6 => /lib/libc.so.6 (0xb767b000)
linux-gate.so.1 => (0xfffffe000)
```

Команда *awk*

Приведенная в примере П2.8 команда сделает копии файлов с суффиксом `.txt` так, что имена копий до суффикса сохранятся, а суффикс будет заменен на `.html`. Например, файл `index.txt` будет скопирован в файл `index.html`.

Пример П2.8. Команда *awk*

```
$ ls *.txt | awk -F. '{print "cp", $0, $1".html"}' | bash
$ ls
index.html index.txt
```

Команда `awk` имеет встроенную функцию `printf`, позволяющую осуществлять форматированный вывод аналогично одноименной функции стандартной библиотеки языка C (пример П2.9).

Пример П2.9. Функция *printf*

```
$ awk -F: '{printf "User:%s\t\t\tUID:%3d\n", $1, $3}' /etc/passwd
User:avahi                UID:103
User:bin                  UID: 1
User:daemon               UID: 2
```

Команды *expand* и *unexpand*

Команда `expand` заменяет символы табуляции пробелами (пример П2.10).

Пример П2.10. Команда *expand*

```
$ echo -e 'A\tB'
A      B

$ echo -e 'A\tB' | od -ta
0000000  A  ht  B  nl
0000004

$ echo -e 'A\tB' | expand | od -ta
0000000  A  sp  sp  sp  sp  sp  sp  sp  B  nl
0000012
```

Команда `unexpand` делает обратную замену. По умолчанию табуляцией заменяются лишь лидирующие пробелы в строке. Опция `-a` позволяет заменять все пробелы (пример П2.11).

Пример П2.11. Команда `unexpand`

```
$ echo -e 'A\tB' | expand | unexpand -a | od -ta
0000000  A  ht  B  nl
0000004
```

Команда `pr`

Команда `pr` предназначена для подготовки текста к печати. В примере П2.12 показано, как с ее помощью можно вывести список файлов в три столбца.

Пример П2.12. Команда `expand`

```
$ ls | pr -3ft
alsa.txt          ls.txt.gz        sfdisk.txt
cat.txt           lsusb.txt        tee.txt
cookie.txt        mkfs.txt.gz      udev.txt
cooo.txt          modalias.txt     users.txt.gz
csplit.txt        modinfo.txt      who.txt
dmesg.txt         mon.txt          xorg.txt
expand.txt        mount.txt.gz     xx00
fsck.txt.gz       nl.txt           xx01
grub.txt          paste.txt.gz     xx02
last.txt          printf.txt       xx03
ldd.txt           pstree.txt       xx04
libmodules.txt    ps.txt.gz        xx05
lsmod.txt         rev.txt          xx06
lspci.txt         rpcinfo.txt      xx07
lspcmcia.txt      rpm.txt          zypper.txt
lsscsi.txt
```

Команды *sort* и *uniq*

Команда `uniq` позволяет отфильтровывать повторяющиеся строки во входном потоке. Входной поток должен быть заранее отсортирован. В примере П2.13 показано, как в текстовый файл, содержащий в себе вывод команды `ps -ef`, через некоторое время добавлен вывод этой же команды. Задача заключается в том, что надо вывести список процессов, изменившихся за это время.

Пример П2.13. Команды *sort* и *uniq*

```
$ ps -ef > ps.txt
```

```
$ ps -ef >> ps.txt
```

```
$ sort ps.txt | uniq -u
```

```
user1    21150   6054   4 00:17 pts/0    00:00:00 ps -ef
```

```
user1    21162   6054   0 00:17 pts/0    00:00:00 ps -ef
```

Команда *wc*

Команда `wc` позволяет подсчитать количество строк, слов и символов в потоке текста. Особенностью GNU-версии команды `wc` является наличие опции `-L`, позволяющей получить максимальную длину строки среди всех обработанных строк (пример П2.14).

Пример П2.14. Команда *wc*

```
$ ls | tr -L
```

```
14
```

Команда *tr*

Команда `tr` позволяет заменять или удалять символы в потоке. Приведенная в примере П2.15 команда позволяет подсчитать количество пробелов в файле.

Пример П2.15. Команда *tr*

```
$ tr -dc ' ' < /etc/motd | wc -c
```

```
4
```

Команда *grep*

Команда `grep` позволяет отфильтровывать строки, удовлетворяющие регулярному выражению с базовым синтаксисом (Basic syntax). А `egrep` использует расширенный синтаксис (Extended Regexp). В примере П2.16 показана команда для получения из журнала `/var/log/messages` только тех сообщений, которые записывались в журнал с 8 до 10 утра.

Пример П2.16. Команда `egrep`

```
$ egrep '^[0-9]{7}:[0-9:]' /var/log/messages
```




Приложение 3

Пример использования telnet для тестирования МТА

Здесь продемонстрирован процесс тестирования работоспособности почтового сервера с помощью клиента telnet. На клиентской машине с ОС MS Windows XP с помощью команды `telnet smtp.class.edu 25`. Последний аргумент команды задает порт SMTP для соединения с МТА. Сеанс работы с сервером МТА показан на рис. ПЗ.1.

В сеансе использованы следующие команды протокола SMTP:

- ☐ HELO — представление клиента серверу;
- ☐ MAIL FROM: — адрес отправителя;
- ☐ RCPT TO: — адрес получателя;
- ☐ DATA — тело письма (его ввод должен быть завершен единственной точкой в строке);
- ☐ QUIT — завершение сеанса.

```
C:\WINDOWS\system32\cmd.exe
220 black.assign-pro.ru ESMTP Postfix (Ubuntu)
HELO this.class.edu
250 black.assign-pro.ru
MAIL FROM: nadia
250 2.1.0 Ok
RCPT TO: olga
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
test
250 2.0.0 Ok: queued as 5B16326206F
QUIT
221 2.0.0 Bye

Подключение к узлу утеряно.
C:\Documents and Settings\BeresnevAI>
```

Рис. ПЗ.1. Сеанс работы с МТА



Приложение 4

Пример файлов конфигурации и описания зон сервера DNS BIND

В этом приложении приводится пример файлов конфигурации DNS-сервера BIND, а также примеры файлов описания зон.

Конфигурация named

В примере П4.1 показан файл конфигурации `/etc/named.conf`.

Пример П4.1. Файл `/etc/named.conf`

```
options {  
    // Файл с PID named  
    pid-file "/var/run/named.pid";  
  
    // Каталог, в котором размещаются зоны  
    directory "/var/named";  
  
    // Серверы, получающие запросы, на которые этот  
    // сервер не имеет данных  
    forwarders {  
        192.168.1.1;  
        172.17.2.53;  
    };  
  
    // Интерфейсы, которые будет прослушивать named  
    listen-on {  
        192.168.0.1;  
        127.0.0.1;
```

```
};  
};  
  
logging {  
    // Канал журналирования для запросов  
    channel qrylog {  
        // Сообщения записываются в файл  
        file "/var/log/namedqry.log";  
        // Уровень важности от info и выше  
        severity info;  
        // Записывать метки времени  
        print-time yes;  
    };  
  
    // Канал журналирования  
    channel baslog {  
        // Запись сообщений посредством syslog  
        syslog local0;  
        // Регулируемый уровень важности сообщений  
        severity dynamic;  
        // Записывать в сообщение категорию  
        print-category yes;  
        // Записывать уровень важности  
        print-severity yes;  
    };  
    // Категория для записи запросов  
    category query { qrylog; };  
    // Все остальные сообщения  
    category default { baslog; };  
};  
  
// Зона указателей на корневые серверы  
zone '.' {  
    type hint;  
    file "root.cache";  
};  
  
// Зона для localhost  
zone 'localhost' {  
    type master;
```

```
file "db.localhost";
};

// Обратная зона для localhost
zone '0.0.127.in-addr.arpa' {
    type master;
    file "db.127.0.0";
};

// Зона, для которой данный сервер является мастером
zone 'class.edu' {
    type master;
    file "master/db.class.edu";
};

// Обратная зона, для которой сервер является мастером
zone '0.168.192.in-addr.arpa' {
    type master;
    file "master/db.192.168.0";
};

// Обратная зона, для которой данный сервер подчиненный
zone '100.168.192.in-addr.arpa' {
    type slave;
    masters { 192.168.100.1; };
    file "slave/bak.192.168.100";
};
```

Зона указателей на корневые серверы

В примере П4.2 показано содержимое файла указателей на корневые серверы.

Пример П4.2. Файл root.cache

```
;      This file holds the information on root name servers needed to
;      initialize cache of Internet domain name servers
;      (e.g. reference this file in the "cache . <file>"
;      configuration file of BIND domain name servers).
;
```

```

;      This file is made available by InterNIC
;      under anonymous FTP as
;          file                /domain/named.root
;          on server           FTP.INTERNIC.NET
;      -OR-                   RS.INTERNIC.NET
;
;      last update:    Feb 04, 2008
;      related version of root zone:    2008020400
;
; formerly NS.INTERNIC.NET
;
.                3600000    IN    NS      A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.  3600000      A      198.41.0.4
A.ROOT-SERVERS.NET.  3600000      AAAA   2001:503:BA3E::2:30
;
; formerly NS1.ISI.EDU
;
.                3600000      NS      B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.  3600000      A      192.228.79.201
;
; formerly C.PSI.NET
;
.                3600000      NS      C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.  3600000      A      192.33.4.12
;
; formerly TERP.UMD.EDU
;
.                3600000      NS      D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.  3600000      A      128.8.10.90
;
; formerly NS.NASA.GOV
;
.                3600000      NS      E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.  3600000      A      192.203.230.10
;
; formerly NS.ISC.ORG
;
.                3600000      NS      F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.  3600000      A      192.5.5.241
F.ROOT-SERVERS.NET.  3600000      AAAA   2001:500:2f::f

```

```

;
; formerly NS.NIC.DDN.MIL
;
.                3600000      NS      G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.  3600000      A      192.112.36.4
;
; formerly AOS.ARL.ARMY.MIL
;
.                3600000      NS      H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.  3600000      A      128.63.2.53
H.ROOT-SERVERS.NET.  3600000      AAAA   2001:500:1::803f:235
;
; formerly NIC.NORDU.NET
;
.                3600000      NS      I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.  3600000      A      192.36.148.17
;
; operated by VeriSign, Inc.
;
.                3600000      NS      J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.  3600000      A      192.58.128.30
J.ROOT-SERVERS.NET.  3600000      AAAA   2001:503:C27::2:30
;
; operated by RIPE NCC
;
.                3600000      NS      K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.  3600000      A      193.0.14.129
K.ROOT-SERVERS.NET.  3600000      AAAA   2001:7fd::1
;
; operated by ICANN
;
.                3600000      NS      L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.  3600000      A      199.7.83.42
;
; operated by WIDE
;
.                3600000      NS      M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.  3600000      A      202.12.27.33
M.ROOT-SERVERS.NET.  3600000      AAAA   2001:dc3::35
; End of File

```

Зона localhost

Зона localhost предназначена для прямого отображения имени узла localhost на адрес 127.0.0.1 (пример П4.3).

Пример П4.3. Зона localhost

```
$TTL 604800 ; Время жизни записей в кэш по умолчанию
@ IN SOA localhost. root.localhost. (
    2      ; Серийный номер
    604800 ; Периодичность проверки
    86400  ; Повторные проверки
    2419200 ; Устаревание зоны
    604800 ) ; Время жизни информ. об отс. записях
;
@ IN NS localhost. ; Сервер имен
@ IN A 127.0.0.1 ; Прямое соответствие
@ IN AAAA ::1 ; То же для IPv6
```

Обратная зона для 127.0.0

Зона обратного отображения 0.0.127.in-addr.arpa задает соответствие имени 1.0.0.127.in-addr.arpa имени localhost (пример П4.4).

Пример П4.4. Обратная зона 0.0.in-addr.arpa

```
$TTL 604800
@ IN SOA localhost. root.localhost. (
    1      ; Серийный номер
    604800 ; Периодичность проверки
    86400  ; Повторные проверки
    2419200 ; Устаревание
    604800 ) ; Негативное кэширование
;
@ IN NS localhost. ; Сервер имен
1 IN PTR localhost. ; Обратное соответствие
```

Зона class.edu

Зона прямого соответствия class.edu (пример П4.5).

Пример П4.5. Зона class.edu

```
$TTL 1D ; Время жизни записей в кэш по умолчанию
@ IN SOA nsserv.class.edu. root.nsserv.class.edu. (
    2010020101      ; Серийный номер
    2D              ; Периодичность проверки
    1D              ; Повторные проверки
    1W              ; Устаревание зоны
    1D      )      ; Вр. жизни информ. об отс. записях
;
IN NS  nsserv.class.edu.      ; Сервер имен
comp1 IN A      192.168.0.11 ; Прямое соответствие
nsserv IN A      192.168.0.1
www    IN A      192.168.0.2
web    IN CNAME  www        ; Псевдоним
```

Обратная зона для 192.168.0

Зона обратного соответствия 0.168.192.in-addr.arpa (пример П4.6).

Пример П4.6. Обратная зона 0.168.192.in-addr.arpa

```
$TTL 1D ; Время жизни записей в кэш по умолчанию
@ IN SOA nsserv.class.edu. root.nsserv.class.edu. (
    2010020101      ; Серийный номер
    2D              ; Периодичность проверки
    1D              ; Повторные проверки
    1W              ; Устаревание зоны
    1D      )      ; Время жизни информ. об отс. записях
;
IN NS  nsserv.class.edu. ; Сервер имен
11 IN PTR comp1.class.edu. ; Обратное соответствие
2  IN PTR www.class.edu.
1  IN PTR nsserv.class.edu.
```




Приложение 5

Сложные варианты установки GNU/Linux

В этом приложении рассматривается установка GNU/Linux, для которой настройки "по умолчанию" не подходят.

Требования к аппаратному обеспечению для установки GNU/Linux на платформе x86/64

Система GNU/Linux может быть установлена практически на любой компьютер с архитектурой x86 (а также на множество других аппаратных платформ). Поддерживаются процессоры от i386 до мощных Xeon. Поддерживаются также и совместимые платформы и клоны, например, AMD и VIA. Современные версии ядра Linux поддерживают также 64-разрядные процессоры AMD и Intel.

До сих пор поддерживаются устаревшие модели процессоров i386 и i486. Однако далеко не все существующие варианты GNU/Linux-дистрибутивов могут быть установлены на эти процессоры.

Все ныне распространенные типы микросхем ОЗУ для платформы x86 поддерживаются GNU/Linux. Среди них:

- ☐ EDO RAM;
- ☐ DRAM;
- ☐ SDRAM;
- ☐ DDRAM.

При использовании устаревших BIOS возможны проблемы при наличии более чем 64 Мбайт ОЗУ. Однако эти проблемы могут быть устранены с помощью установки корректного значения параметра `mem`, передаваемого ядру загрузчиком.

Требования к аппаратному обеспечению во многом зависят от используемого дистрибутива. Можно найти минималистские версии GNU/Linux, способные работать с 32 Мбайт ОЗУ и меньше, но для большинства современных дистрибутивов разумный минимум — 512 Мбайт ОЗУ. Естественно, желание работать с графическими приложениями, базами данных и т. п. увеличивает этот минимум в разы.

Для корневого раздела разумно предусмотреть минимум 6 Гбайт, а размер раздела подкачки обычно выделяют пространство, равное удвоенному объему ОЗУ.

GNU/Linux работает с любыми существующими для x86 видеоадаптерами в текстовом режиме. Работа в графическом режиме зависит от поддержки данного графического оборудования реализацией X Window Xorg. При установке GNU/Linux следует заранее уточнить версию реализации X Window Xorg, использованную в дистрибутиве. Возможно, будет необходимо проверить, поддерживает ли данная реализация установленный видеоадаптер.

Поддерживаются следующие основные разновидности шин:

- ☐ ISA;
- ☐ EISA;
- ☐ VESA Local BUS;
- ☐ MCA;
- ☐ PCI;
- ☐ AGP;
- ☐ USB (1.1 и 2.0);
- ☐ PC Card (PCMCIA);
- ☐ Firewire.

Для установки GNU/Linux нужен стандартный AT-контроллер для жестких дисков и IDE жесткий диск. Также могут быть использованы SCSI-диски. При этом следует убедиться в том, что SCSI хост-адаптер поддерживается в данном ядре Linux (см. Hardware-HOWTO, SCSI-HOWTO, документацию на ядро и информацию от поставщика дистрибутива). Обычно поддержка распространенного SCSI-оборудования в GNU/Linux проблем не вызывает. Кроме IDE- и SCSI-дисков поддерживаются устаревшие RLL- и MFM-диски. Ядра 2.4 и 2.6 поддерживают USB-диски, хотя корневую файловую систему на них обычно не устанавливают. Поддерживаются SATA-диски (Serial ATA), контроллеры SCSI и ATA RAID. Для проверки поддержки конкретного оборудования следует обратиться к документации производителя и, возможно, в Hardware-HOWTO.

GNU/Linux обеспечивает поддержку дисков ATAPI и SCSI CD/DVD. Наличие их не является обязательным в случаях, когда с них не будет производиться установка. Также необязательным является наличие флоппи-дисковода.

Подготовка к установке GNU/Linux на компьютерах с архитектурой x86/64

Когда GNU/Linux устанавливается на домашнем компьютере или на компьютере, выделенном для экспериментов, о подготовке списка комплектации аппаратного обеспечения можно особо не беспокоиться. Но для успешной установки GNU/Linux на промышленный сервер важно подготовить список комплектации оборудования (installation hardware checklist). В этом документе должны быть отражены основные аспекты конфигурации аппаратного обеспечения. Среди них:

- ☐ тип используемых центральных процессоров и их количество;
 - при использовании многопроцессорной схемы — архитектура системы;
 - тип материнской платы и версия основного набора микросхем (chipset);
- ☐ данные о BIOS:
 - версия;
 - поддерживаемые загрузочные устройства;
 - способность работы с большими дисками (более 1024 цилиндров);
 - поддержка PnP;
 - способность статического конфигурирования ресурсов для устройств расширения (IRQ, IO/Base, DMA);
 - настройка аппаратного времени — либо на UTC, либо на местное время;
- ☐ данные об ОЗУ:
 - тип;
 - производитель;
 - объем;
- ☐ системные шины, поддерживаемые данной материнской платой:
 - PCI, PCI-X, PCI-II;
 - AGP;
 - ISA;
 - EISA;

- MCA;
 - VESA Local BUS;
- ☐ шины и порты расширения:
- последовательные порты, количество, тип разъема (9 или 25 контактов);
 - параллельные порты и их тип;
 - USB;
 - Firewire;
 - PC Card (PCMCIA);
 - PS/2;
- ☐ данные о SCSI хост-адаптере:
- производитель хост-адаптера;
 - тип хост-адаптера (ISA, PCI и т. п.);
 - стандарт SCSI-шины;
 - тактовая частота шины;
 - SCSI ID адаптера;
- ☐ наличие и тип флоппи-дисков;
- ☐ данные о дисковой подсистеме:
- объем накопителей и их производители;
 - геометрия дисков (количество головок, цилиндров);
 - IDE:
 - ◇ тип контроллеров (Serial/Parallel ATA);
 - ◇ количество накопителей;
 - ◇ подключение к Primary- или Secondary-контроллеру (или к дополнительной плате расширения);
 - ◇ установка Master/Slave/Cable select;
 - SCSI:
 - ◇ количество накопителей;
 - ◇ тип шины (SE или LVD);
 - ◇ установки SCSI ID и LUN;
 - RAID-устройства:
 - ◇ производитель;
 - ◇ тип контроллера: SCSI или ATA;

- ◇ настройки SCSI ID и LUN или Primary/Secondary и Master/Slave для ATA;
- ◇ количество дисков в RAID и его уровень;
- предлагаемая схема размещения файловых систем (точки монтирования, диски, разделы, размеры);
- использование мультипликативных устройств:
 - ◇ программный RAID (логические устройства md) и его уровень;
 - ◇ LVM или ELVM (Enterprise Logical Volume Manager);
- накопители CD/DVD:
 - производитель;
 - тип интерфейса: ATA или SCSI;
 - настройки контроллеров: SCSI ID или Primary/Secondary и Master/Slave для ATA;
 - возможность загрузки с них;
- данные об устройствах ввода информации:
 - типы интерфейсов подключения устройств (PS/2, IrDA, USB и т. п.);
 - количество клавиш на клавиатуре;
 - мышь:
 - ◇ модель;
 - ◇ интерфейс;
 - ◇ протокол (PS/2, IMPS/2);
 - ◇ количество кнопок;
 - наличие иных устройств ввода (необязательно): джойстики, touch-pad и пр.;
- видеоподсистема:
 - типы и количество установленных видеоадаптеров, модели и производители;
 - количество видеопамяти;
 - поддерживаемые видеорежимы;
 - тип и параметры монитора (кадровая и строчные частоты синхронизации);
- аудиоподсистема и MIDI-устройства:
 - тип и количество установленных аудиоустройств;
 - типы интерфейсов (ISA, PCI и пр.);

☐ настройки оборудования для работы в сети:

- количество, типы, модели и параметры сетевых адаптеров;
- требуемые настройки TCP/IP:
- имя узла и имя домена;
- версии IP (IPv4 или IPv6);
- IP-адреса сетевых интерфейсов;
- IP-адреса шлюзов;
- IP-адреса DNS-серверов.

В зависимости от разновидности дистрибутива GNU/Linux, а также от предназначения устанавливаемой системы, приведенный выше список может подвергнуться значительным изменениям. Однако наличие его позволяет избежать многих проблем с установкой GNU/Linux.

Другим важным моментом при подготовке к установке GNU/Linux является планирование типа или профиля системы, определяемого основным направлением использования системы. От этого профиля зависит набор пакетов программного обеспечения, который требуется установить.

Распространенные программы установки GNU/Linux от основных производителей дистрибутивов часто предлагают следующие профили установки:

- ☐ рабочая станция;
- ☐ сетевой сервер;
- ☐ рабочая станция разработчика.

Набор программных пакетов, который должен быть установлен, может быть полностью определен пользователем либо же выбран и дополнен на основе предопределенного набора. Поэтому до начала установки следует составить список требуемого программного обеспечения для установки.

В настоящее время GNU/Linux поддерживает следующие основные типы аутентификации пользователей для входа в сеанс:

- ☐ на основе локальных файлов базы данных учетных записей — обычная схема на основе теневых паролей;
- ☐ аутентификация в системе Kerberos;
- ☐ аутентификация в NIS/NIS+;
- ☐ аутентификация в LDAP;
- ☐ вход в сеанс с использованием SMB-сервера аутентификации.

Перед установкой следует решить, какая схема будет использоваться.

Особое внимание следует уделить вопросу планирования разбиения дисков на разделы и определению точек монтирования. Неудачный выбор схемы разбиения жесткого диска на разделы может привести к недостатку дискового пространства в одной файловой системе при явном избытке свободного пространства в другой.

Далее приведен вывод команды `du`, показывающий использование дискового пространства файлами, содержащимися в подкаталогах корневого каталога. На основе анализа этой информации можно спланировать требуемое для файловых систем дисковое пространство. Приведенная в примере П5.1 команда была выполнена на типичной GNU/Linux рабочей станции разработчика.

Пример П5.1. Объем, занимаемый файлами в подкаталогах корневого каталога

```
du -sh /*
7,3M    /bin
7,2M    /boot
196K    /dev
102M    /etc
2,7G    /home
418M    /lib
16K     /lost+found
11G     /media
8,0K    /misc
12K     /mnt
0       /net
8,0K    /opt
898M    /proc
300K    /root
21M     /sbin
8,0K    /selinux
8,0K    /srv
0       /sys
136K    /tmp
4,3G    /usr
211M    /var
```

На серверах бывает важно монтировать файловые системы `/var`, `/usr`, `/home` и др. на отдельных разделах диска или на разных дисках. Чаще всего на отдельных файловых системах монтируются каталоги:

- ❑ `/boot` — файлы в нем занимают мало места, но Red Hat и ему подобные дистрибутивы требуют для него минимум 100 Мбайт (более 150 Мбайт

выделять нет смысла), тип файловой системы желательно установить ext2 или ext3;

- ❑ /home — его размер следует выбирать исходя из планируемого количества пользователей, отводя для каждого из них некоторый средний размер занимаемого пространства (например, 1 Гбайт × 10 пользователей = 10 Гбайт);
- ❑ /usr — в этот каталог устанавливается основная масса программного обеспечения, поэтому в зависимости от профиля установки для массовых дистрибутивов рекомендуется выбирать от 800 Мбайт для серверов в минимальной конфигурации до 5—6 Гбайт для рабочих станций разрабатчиков;
- ❑ /opt — каталог для установки дополнительного программного обеспечения, использование которого целиком зависит от воли сборщиков дистрибутива, поэтому он может совсем не использоваться, но он может потребовать и значительного дискового пространства — порядка 1 Гбайт и выше;
- ❑ /var — в большинстве случаев для этого каталога требуется не более 300 Мбайт, однако на серверах баз данных это пространство может быть больше на порядки;
- ❑ /tmp — обычно достаточно 200—300 Мбайт.

В случае если указанные выше каталоги смонтированы на отдельных разделах, то для корневого раздела не требуется более 500 Мбайт дискового пространства.

Если вы устанавливаете GNU/Linux в первый раз, установите все в единственный корневой раздел, размером 6 Гбайт и выше. На промышленных серверах такой вариант установки совершенно исключен.

Обычно программы установки современных дистрибутивов GNU/Linux предоставляют возможность создать на жестких дисках требуемые разделы и определить файловые системы, которые будут размещены на них. Тем не менее, разделы могут быть созданы также и заранее, например, с помощью команды `fdisk`. Помимо прочего следует заранее решить, каким образом будет загружаться устанавливаемый GNU/Linux. В подавляющем большинстве случаев для этого используется загрузчик LILO или GRUB, размещаемый либо в MBR, либо в активном разделе. Практикуется также и способ загрузки с CD/DVD. Гораздо реже на обычных (обладающих собственными дисками) рабочих станциях и серверах встречается загрузка по сети.

Установка GNU/Linux

Различные программы для установки GNU/Linux, поставляемые производителями дистрибутивов, значительно отличаются между собой, однако среди них можно выделить следующие основные стадии.

❑ Подготовительная:

- тип клавиатуры и терминала;
- настройка мыши;
- локализация системы и установка раскладки клавиатуры;
- определение источника установки (CD, HTTP, FTP, NFS и т. д.);
- в случае загрузки по сети — настройка сетевых интерфейсов;
- загрузка требуемых драйверов.

❑ Подготовка файловых систем:

- разбиение диска на разделы;
- создание файловых систем;
- монтирование их к временной корневой файловой системе, используемой на стадии установки.

❑ Установка базовой системы. Базовая система представляет собой минимально функциональный GNU/Linux, пригодный для продолжения дальнейшей установки. Этот этап характерен для Debian. В Red Hat установка базовой системы производится в начале установки программных пакетов.

❑ Определение профиля устанавливаемой системы и выбор программного обеспечения для установки.

❑ Установка программного обеспечения. Обычно эта фаза неинтерактивна, но, например, в Debian многие устанавливаемые пакеты сразу же конфигурируются, поэтому в этом дистрибутиве на этой стадии программа установки требует от пользователя ответов на вопросы относительно конфигурации устанавливаемого программного обеспечения.

❑ Завершающая фаза установки:

- установка и настройка загрузчика;
- установка доменного имени системы;
- установка пароля root;
- определение типа аутентификации: с помощью базы данных паролей, NIS/NIS+, LDAP, Kerberos и т. д.;
- заведение обычных пользователей;

- настройка сети и, возможно, несложная настройка фильтра IP-пакетов;
- определение параметров видеооборудования и настройка X Window;
- конфигурирование дополнительного оборудования, например, звуковой подсистемы;
- конфигурирование профилей пользователей в установленной системе и установка дополнительного ПО, не вошедшего в основной дистрибутив.

В зависимости от привязки используемого дистрибутива к конкретной версии ядра обычно различается стадия установки базового ПО. Например, Red Hat подобные дистрибутивы не выделяют установку базовой подсистемы в отдельную фазу, т. к. дистрибутив рассчитан на использование конкретной версии ядра. А Debian, напротив, рассчитан на возможность использования различных ядер даже не Linux, а, например, HURD. Поэтому программа установки Debian сначала устанавливает базовый комплект и выбранное ядро, а затем устанавливает загрузчик и пытается загрузиться. После успешной перезагрузки программа установки продолжает работу, устанавливая дополнительное ПО.

В некоторых дистрибутивах после установки базового комплекта и установки загрузчика система полностью работоспособна и дальнейшая установка ПО и настройка оборудования выполняется после перезагрузки. В Gentoo Linux процесс установки ПО напоминает систему портов, используемых во FreeBSD: выбранное программное обеспечение автоматически копируется в виде исходного кода для последующей компиляции. В Gentoo даже базовый комплект может быть полностью перекомпилирован.

ВНИМАНИЕ!

Если вы устанавливаете GNU/Linux на диск, где уже есть другие операционные системы, обязательно выполните резервное копирование данных!

Приложение 6



Описание компакт-диска

Содержимое компакт-диска описано в табл. П6.1.

Таблица П6.1. Описание компакт-диска

Каталог	Файл	Описание
\Ubuntu ubuntu-9.10-serv	er-i386.iso	Дистрибутив Ubuntu 9.10
\SunVB VirtualBox	-3.1.2-OSE.tar.bz2	Пакет установки Sun VirtualBox OSE

İ ðãäì àòí ûé óêàçàòâëü

A

Absolute pathname 65
Account 25
ALSA 329
Apache 426

B

Back reference 175
Bourne Again Shell 35
Bourne Shell 35
Broadcast address 341

C

Command Line Interface (CLI) 32
Command substitution 45
Суперблок 78

D

Default gateway 343
DHCP-сервер 423
Discretionary Access Control (DAC) 107
Dovecot 470

E

Extended SMTP (ESMTP) 441

F

Fully Qualified Domain Name (FQDN) 350

G

Getty 26
GID 30

H

Hard link 79
Hardware Bootstrap Loader 254

I

Inline-подстановка 183, 284
IRQ 315

J

Journaling 217

L

Light Weight Process (LWP) 97
Link count 81
Link counter 79
Linking 299
Loopback interface 343

M

Magic numbers 77
Master Boot Record (MBR) 254

O

OSI 339

P

Pager 145
 Paging 224
 Partitions 210
 Path 64
 PCMCIA 331
 Pipe 142
 Plain file 63
 Postfix 458
 Power On Self Test (POST) 254
 Preemptive multitaskings 89

Q

Quotation 186

R

Relative pathnames 65
 Root directory 63
 Runlevels 246

S

Shell 32
 Simple Mail Transfer Protocol (SMTP) 441
 Socket 357

T

Thread 97
 Time to live (TTL) 406
 Trusted account 401

U

UID 30
 USB 330

W

Web-страница 434

X

X Window 484
 X-приложение 499
 ресурсы 503
 удаленный запуск 504
 X-сервер 486

Z

Z Shell 35
 Zombie 94
 Zone transfer 405

A

Агент:
 доставочный 440
 пользовательский 440
 транспортный 440
 Адрес:
 IP 341
 MAC 346
 широковещательный 341
 Адресация IPv6 344
 Архив:
 извлечение 237, 240, 242, 244
 создание 236, 239, 241

Б

Библиотека:
 resolver 350
 разделяемая 300
 статическая 300

B

Виртуальный псевдоним 465
 Виртуальный терминал 26
 Восстановление из архива *См. Архив,*
 извлечение
 Время жизни записи 406
 Вход в сеанс 26
 Вывод на экран строки 144

Д

Дамп 165
 Делегирование 419
 прав 406
 Демон 90
 cupsd 471
 syslogd 267
 Дескриптор файла 89
 Диск жесткий 209
 Домен корневой 405
 Дорожка 209

Ж

Жесткая связь 79, 80
Журнал 267
Журналирование 217

З

Загрузчик:
GRUB 255
LILO 258
аппаратный 254
начальный 254
программный 254
Задание 88
Закольцовывающий сетевой
интерфейс 343
Замена символов 161
Звуковая карта 329
Значение по умолчанию 184
Зомби 94
Зона 405

И

Идентификатор процесса 90
Имя файла:
абсолютное 64, 65
относительное 64, 65
полное 64
Индексные дескрипторы, метаданные 78

К

Каталог 64, 79
/etc/skel 276
/lib/modules 316
/proc 98
домашний 66
корневой 63
переименование 73
смена текущего 68
удаление 70, 71
Клиент resolver 417
Команда:
a2ps 474
alias 44
apachectl 434
apt-get 309
ar 300
arp 348

at 262
awk 152
batch 264
bzip2 236
cancel 478
case 198
cat 146
cd 68
cfdisk 213
chgrp 111
chkconfig 250
chmod 114
chown 111
clear 29
cp 72
cpio 239
crontab 265
csplit 166
cut 148
dd 233
depmod 319
df 229
diff 154
dig 417
disable 477
dpkg 309
du 229
dump 243
e2fsck 220
echo 144
edquota 288
egrep 171
elif 196
else 197
enable 479
env 40
exit 27
expand 156
export 39
fdisk 209, 213
fgrep 171
file 77
find 74
fmt 157
for 200
fsck 219
ftp 381
fuser 99
gpasswd 281

(продолжение рубрики см. на стр. 554)

Команда (продолжение):

grep 171
 groff 55
 groupadd 280
 groupdel 281
 grub 256
 gs 474
 gunzip 235
 gzip 235
 halt 251
 head 147
 help 52
 history 42
 host 417
 hostname 350
 id 30
 if 195
 ifconfig 346
 info 57
 insmod 318
 join 160
 killall 102
 last 31, 291
 lastlog 292
 ldd 302
 less 145
 lilo 261
 locate 76
 logout 27
 lp 473
 lpadmin 475
 lpstat 477
 lsmod 317
 man 52
 manpath 56
 mkdir 71
 mkdosfs 217
 mke2fs 217
 mkfs 217
 mkreiserfs 217
 mkswap 225
 modinfo 318
 modprobe 319
 more 145
 mount 222
 mutt 441
 mv 73
 ndc 416
 netstat 346, 357
 nmblookup 394

nslookup 417
 od 165
 passwd 29, 279
 paste 164
 pax 241
 pgrep 97
 ping 347
 pkill 102
 poweroff 252
 pr 157
 ps 95
 pstree 100
 pwd 51, 65
 quotacheck 287
 quotaoff 290
 quotaon 290
 rcp 371
 read 187
 reboot 251, 252
 reject 478
 repquota 290
 restore 244
 rexec 370
 rlogin 370
 rm 69
 rmdir 71
 rmmmod 318
 rndc 416
 rndc-confgen 416
 rpm 303
 rsh 371
 runlevel 251
 run-parts 265
 scp 373
 sed 149
 set 27, 39, 190
 sfdisk 213
 shift 189
 shutdown 251
 sort 158
 source 283
 split 166
 ssh 373
 swapon 224
 tar 236
 tee 143
 test 191
 testparm 393
 time 88
 top 98

touch 69
tr 161
traceroute 349
tty 31
umask 118
unalias 45
uname -r 316
uniq 159
unset 39
until 201
updatedb 76
useradd 274, 275
userdel 278
usermod 277
w 98
wc 160
wget 382
while 201
who 31, 291
whois 355
xargs 168
xfontsel 502
xrdp 503
yast 307
yum 307
zcat 235
встроенная 36
системная 36

Командная оболочка 32
Командная подстановка 45
Командная строка 32
Компилятор gcc 300
Компоновка 299
Конвейер 142
Конструкция here document 141
Конфигурирование 298

М

Маршрутизатор 343, 347
 по умолчанию 343, 349
Маска сети 342
Менеджер пакетов RPM 303
Механизм перечисления Bash 49
Многозадачность, вытесняющая 89
Модуль ядра 316
Монтирование 221

Н

Неявный вызов оболочки 181

О

Оболочка bash (Bourne-Again Shell) 32
Обратная ссылка 175
Оператор условного исполнения команд 194
Опция оболочки noclobber 140, 141
Очередь почтовых сообщений 454
Ошибка, сообщение 51

П

Пакет quota 285
Пакет SAMBA 390
Память устройств 316
Пейджер 145
Переменная окружения:
 DISPLAY 506
 HISTFILE 41
 HISTFILESIZE 42
 MANPATH 56
 PS1 28, 41
 SHELL 32
Переменные оболочки 38, 183
Перенаправление:
 ввода 68
 поток 138
Печать сетевая 398
Письмо электронное 441
Позиционный параметр 188
Политика DAC 107
Пользователь:
 аутентификация 26, 436
 владелец файла 107
 находящийся в сеансе 291
 работающий в системе 31
Поток 97
Потоковый редактор:
 awk 152, 179
 sed 149, 177
Почтовый ящик 441
Права владения файлами 111
Права доступа:
 к каталогам 110
 к файлу 108
 установка 114, 118
Прерывание 315
Приоритет процессов 104
 относительный 104

Προγραμμα 87
 /sbin/init 255
 aptitude 309
 dselect 309
 gdm 497
 httpd 426
 in.telnetd 367
 kdm 497
 logrotate 271
 mail 456
 make 297, 332
 named 406
 nmbd 390
 sendmail 442
 smbclient 390, 395
 smbd 390
 smbstatus 397
 sshd 373
 startx 495
 SWAT 391
 tasksel 309
 telnet 367
 X 486
 xdm 497
 xfs 491
 xinit 495
 Протокол:
 ARP 348
 ESMTP 441
 FTP 378
 IMAP 468
 IPP 471
 POP3 466
 SMTP 441
 X11 484
 XDMCP 497
 Профиль пользователя 282
 Процесс 88
 демон 90
 дочерний 90
 жизненный цикл 93
 зомби 94
 мониторинг 95
 облегченный 97
 прикладной 90
 родительский 90
 ядра 90
 Псевдоним команды 44
 Псевдотерминал 31
 Путь 64

Р

Раздел жесткого диска 210
 Раздел подкачки 224
 Регулярные выражения 169
 Режим выполнения заданий:
 интерактивный 91, 92
 фоновый 91
 Резервное копирование 231, 236
 инкрементальное 233
 интерактивное 233
 неинтерактивное 233
 нулевого уровня 243
 поблочное 233
 полное 233

С

Сектор 209
 Сервер:
 исключительно кэширующий 406
 мастер 405
 подчиненный 405
 терминала 26
 шрифтов 491
 Сетевая файловая система NFS 384
 Сетевой адаптер 327
 Сжатие данных 234
 Сигнал 101
 Система:
 cron 264
 man 52
 PAM 279
 SSH 373
 sysfs 322
 Texinfo 57
 udev 322
 Система печати CUPS 398
 управление очередью печати 477
 управление принтерами 475
 Система управления пакетами Debian 308
 Системный вызов 88
 exec() 94
 fork() 89, 93
 wait() 94
 Служба:
 DNS 404
 FTP 378
 NFS 384
 telnet 367

привилегированная 357
самостоятельная 358
удаленного доступа 370
Сокет 357
Сообщения об ошибках 51
Специальные биты прав доступа 119
Ссылка символическая 84, 302
Стандартный поток:
 ввода stdin 138
 вывода stdout 138
 вывода ошибок stderr 138
Супердемон:
 inetd 359
 xinetd 362
Суффикс имени файла 64
Сценарий 181
Счетчик имен файла 79

T

Таблица маршрутизации 343, 348
Таблица разделов жесткого диска 210
Текстовый редактор:
 vi 127
 командный режим 128
 перемещение по тексту 129
 поиск и замена строк 132
 редактирование текста 131
 режим ввода текста 128
 режим двоеточия 128, 135
 vim 127
Терминал 31
Точка монтирования 221
Трансфер зоны 405

У

Управление программным
 обеспечением 293
Уровень исполнения 246
Установка:
 GNU/Linux 11
 оборудования 315
Устройство:
 PCI 325
 PCMCIA 331
 SCSI 326
 USB 330
 блочное 211
 символьное 211

Учетная запись 25
 доверенная 401

Ф

Файл 63
 .htaccess 436
 .rhosts 371
 /etc/crontab 264
 /etc/dovecot.conf 470
 /etc/exports 384
 /etc/fstab 227
 /etc/group 280
 /etc/host.conf 351
 /etc/hosts 351
 /etc/hosts.equiv 371
 /etc/inetd.conf 359
 /etc/inittab 255
 /etc/lilo.conf 259
 /etc/login.defs 30
 /etc/modprobe.conf 320
 /etc/named.conf 407
 /etc/nsswitch.conf 351
 /etc/profile 41, 282
 /etc/resolver.conf 351
 /etc/rndc.conf 416
 /etc/samba/smb.conf 391, 424
 /etc/samba/smbpasswd 396
 /etc/services 357
 /etc/shadow 273
 /etc/shells 35
 /etc/ssh/ssh_config 373
 /etc/ssh/sshd_config 373
 /etc/syslog.conf 267
 /proc/interrupts 316
 /proc/modules 317
 /var/log/lastlog 292
 /var/log/wtmp 291
 /var/log/wtmp 31
 /var/run/utmp 31, 291
 ~/bash_history 41
 ~/bash_profile 41, 283
 httpd.conf 426
 main.cf 462
 Makefile 297
 master.cf 459, 462
 sendmail.cf 442, 446
 sendmail.mc 446
 XF86Config 486
 xorg.conf 486
(окончание рубрики см. на стр. 558)

Файл (окончание):

вывод неповторяющихся строк 159
 конфигурации /etc/man.config 56
 копирование 72
 объединение с другим файлом 160
 обычный 63
 переименование 73
 перемещение 73
 подготовка к печати 157
 подкачки 224
 подсчет количества строк, слов и
 символов 160
 поиск строк по регулярному
 выражению 171
 разделение на части 166
 скрытый 48, 64
 содержимое 77
 создание 68
 сортировка строк 158
 текстовый:
 вывод в стандартный поток вывода 146
 вывод отдельных символов 148
 постраничный просмотр 145
 удаление 69
 устройства 321
 Файловая система 78, 216
 ext2 216
 ext3 216

Reiserfs 216
 псевдофайловая 98
 целостность 219

Фильтр 142

Функции в сценариях 203

Х

Хостинг 438

 виртуальный, почтовый 465

Ц

Цилиндр 209

Ш

Шаблоны подстановки 48

Шрифты 500

Э

Экранирование 186

Я

Ядро 88

 сборка и установка 331