

Владимирский государственный университет

К. В. КУЛИКОВ

**ОСНОВЫ ПРИМЕНЕНИЯ
ОПЕРАЦИОННЫХ СИСТЕМ LINUX**

Учебное пособие

Владимир 2022

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»

К. В. КУЛИКОВ

ОСНОВЫ ПРИМЕНЕНИЯ ОПЕРАЦИОННЫХ СИСТЕМ LINUX

Учебное пособие

Электронное издание



Владимир 2022

ISBN 978-5-9984-1601-9

© ВлГУ, 2022

© Куликов К. В., 2022

УДК 004.451.9
ББК 32.972.11

Рецензенты:

Доктор технических наук, профессор
зав. кафедрой информационных систем и программной инженерии
Владимирского государственного университета
имени Александра Григорьевича и Николая Григорьевича Столетовых
И. Е. Жигалов

Кандидат технических наук
инженер программист 2-й категории ООО «СТЦ»
Г. А. Лобачев

Куликов, К. В.

Основы применения операционных систем Linux [Электронный ресурс] : учеб. пособие / К. В. Куликов ; Владим. гос. ун-т им. А. Г. и Н. Г. Столетовых. – Владимир : Изд-во ВлГУ, 2022. – 244 с. – ISBN 978-5-9984-1601-9. – Электрон. дан. (2,79 Мб). – 1 электрон. опт. диск (CD-ROM). – Систем. требования: Intel от 1,3 ГГц ; Windows XP/7/8/10 ; Adobe Reader ; дисковод CD-ROM. – Загл. с титул. экрана.

Рассматриваются основные инструментальные средства использования операционных систем на базе ядра Linux: командный интерфейс, справочная система, команды работы с файлами, создание скриптов, работа с компьютерным оборудованием и сетью, управление процессами и пакетами, базовые средства безопасности.

Предназначено для студентов бакалавриата очной формы обучения по направлению подготовки 09.03.01 «Информатика и вычислительная техника».

Рекомендовано для формирования профессиональных компетенций в соответствии с ФГОС ВО.

Табл. 25. Ил. 12. Библиогр.: 11 назв.

ISBN 978-5-9984-1601-9

© ВлГУ, 2022
© Куликов К. В., 2022

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	9
Глава 1. ВВЕДЕНИЕ В LINUX	10
1.1. Эволюция Linux и популярные операционные системы....	10
1.1.1. Роль ядра.....	10
1.1.2. Приложения.....	11
1.1.3. Роль открытого исходного кода	12
1.1.4. Распределения Linux	13
1.1.5. Аппаратные платформы.....	17
1.2. Выбор операционной системы	18
1.2.1. Пункты принятия решений	18
1.2.2. Microsoft Windows	20
1.2.3. Apple macOS	21
1.2.4. BSD.....	21
1.2.5. Другие коммерческие UNIX.....	22
1.2.6. Linux	22
1.2.7. Android	23
1.3. Основные приложения с открытым исходным кодом.....	24
1.3.1. Серверные приложения.....	25
1.3.2. Настольные приложения	29
1.3.3. Инструменты консоли	31
1.3.4. Средства разработки.....	33
1.4. Программное обеспечения с открытым исходным кодом и лицензирование	36
1.4.1. Фонд свободного программного обеспечения и открытый исходный код	37
1.4.2. Дополнительные условия.....	39
1.4.3. Другие схемы лицензирования.....	40
1.4.4. Бизнес-модели с открытым исходным кодом.....	41
1.5. Графический и неграфический режимы.....	42
1.6. Командная строка	44
1.7. Виртуализация и облачные вычисления	45
1.8. Использование Linux для работы.....	47
1.9. Безопасность Linux-компьютера.....	48
1.10. Защита данных в сети Интернет	49

Глава 2. ОСНОВНЫЕ КОМАНДЫ	51
2.1. Интерфейс командной строки (CLI)	51
2.2. Доступ к терминалу	51
2.2.1. Запрос	52
2.2.2. Shell	53
2.2.3. Команды форматирования	53
2.2.4. Параметры команд	54
2.3. История команд	56
2.4. Переменные оболочки BASH	57
2.5. Переменная PATH	58
2.6. Команда export	59
2.7. Команда which	60
2.8. Команда type	60
2.9. Псевдонимы	61
2.10. Символы раскрытия	62
2.10.1. Asterisk (звездочка *)	63
2.10.2. Знак вопроса (?)	63
2.10.3. Квадратные скобки []	64
2.10.4. Восклицательный знак (!)	65
2.11. Цитирование	65
2.11.1. Двойные кавычки	65
2.11.2. Одинарные кавычки	66
2.11.3. Символ обратной косой черты (\)	66
2.11.4. Обратные кавычки	67
2.12. Операции управления	67
2.12.1. Точка с запятой	68
2.12.2. Двойной амперсанд (&&)	68
2.12.3. Две вертикальных черты	69
Глава 3. ПОЛУЧЕНИЕ ПОМОЩИ	71
3.1. Справочные страницы	71
3.1.1. Просмотр справочных страниц	71
3.1.2. Управление отображением справочной страницы	72
3.1.3. Разделы справочной страницы	73
3.1.4. Раздел Synopsis справочной страницы	73
3.1.5. Поиск по справочной странице	74
3.1.6. Справочные страницы, отсортированные по разделам ...	74
3.2. Команда Info	75
3.2.1. Отображение документации для команды	76

3.2.2. Перемещение во время просмотра документации	76
3.3. Дополнительные источники помощи	78
3.3.1. Использование параметра --help	78
3.3.2. Дополнительная документация системы	79
3.4. Поиск файлов, команд и документации	79
3.4.1. Где находятся команды?	80
3.4.2. Как найти любой файл или каталог с помощью команды locate?	80
3.4.3. Подсчет количества файлов.....	81
3.4.4. Ограничение вывода.....	82
3.5. Поиск файлов с помощью команды “find”	82
3.5.1. Поиск по имени файла	83
3.5.2. Отображение сведений о файле	83
3.5.3. Поиск файлов по размеру	83
 Глава 4. РАБОТА С ФАЙЛАМИ И КАТАЛОГАМИ.....	85
4.1. Общие сведения о файлах и каталогах.....	85
4.1.1. Путь к каталогу	86
4.1.2. Домашний каталог	87
4.1.3. Текущий каталог	88
4.1.4. Изменение каталогов	88
4.1.5. Абсолютные и относительные пути	89
4.2. Вывод списка файлов в каталоге	90
4.2.1. Список с выделением цветом типов файлов.....	90
4.2.2. Список скрытых файлов	91
4.2.3. Список каталогов	92
4.2.4. Рекурсивный список	92
4.2.5. Сортировка списка.....	93
4.3. Операции с файлами и каталогами	94
4.3.1. Подробный режим копирования	95
4.3.2. Как избежать перезаписи данных при копировании	95
4.4. Жесткие и относительные ссылки	97
4.5. Сжатие файлов	101
4.6. Архивирование файлов	104
4.7. ZIP-файлы	106
4.8. Стандарт иерархии файловой системы	107

Глава 5. ПОТОКИ, ПЕРЕНАПРАВЛЕНИЯ И РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ	114
5.1. Потоки командной строки	114
5.2. Перенаправление ввода/вывода	115
5.2.1. STDIN	116
5.2.2. STDOUT	116
5.2.3. STDERR	116
5.2.4. Перенаправление STDOUT	116
5.2.5. Перенаправление STDERR	117
5.2.6. Перенаправление нескольких потоков	117
5.3. Просмотр файлов с помощью команды less	118
5.3.1. Экран помощи в less	119
5.3.2. Команды перемещения в less	119
5.3.3. Команды поиска в less	120
5.4. Сортировка файлов или входных данных	120
5.5. Просмотр статистики файлов с помощью команды wc....	121
5.6. Использование команды cut для фильтрации содержимого файла	122
5.7. Использование команды grep для фильтрации содержимого файла	123
5.8. Базовые регулярные выражения	124
5.8.1. Базовые регулярные выражения – символ . (точка).....	125
5.8.2. Базовые регулярные выражения – символы [].....	125
5.8.3. Базовые регулярные выражения – символ *	126
5.8.4. Базовые регулярные выражения – символы ^ и \$	126
5.8.5. Базовые регулярные выражения – символ \.....	127
5.9. Расширенные регулярные выражения	127
Глава 6. СОЗДАНИЕ СКРИПТОВ	129
6.1. Скрипты оболочки	129
6.2. Редактирование скриптов оболочки	130
6.3. Основы создания скриптов	131
6.3.1. Переменные	132
6.3.2. Условные выражения	134
6.3.3. Циклы	137
Глава 7. КОМПЬЮТЕРНОЕ ОБОРУДОВАНИЕ.....	140
7.1. Процессоры.....	140
7.2. Материнская плата и шина	143

7.2.1. dmidecode	143
7.2.2. Оперативная память	144
7.2.3. Периферийные устройства	145
7.2.4. Устройства универсальной последовательной шины (USB)	147
7.3. Слой аппаратных абстракций (HAL)	147
7.4. Дисковые устройства.....	148
7.5. Оптические диски	150
7.6. Устройства отображения видео.....	151
7.7. Управление устройствами	152
 Глава 8. УПРАВЛЕНИЕ ПАКЕТАМИ И ПРОЦЕССАМИ	153
8.1. Управление пакетами	153
8.1.1. Управление пакетами Debian	153
8.1.2. Управление пакетами RPM.....	156
8.2. Ядро Linux	159
8.3. Иерархия процессов.....	161
8.4. Команда ps (Process).....	163
8.5. Команда top.....	164
8.6. Команда free	166
8.7. Лог-файлы.....	167
8.8. Команда dmesg	169
 Глава 9. КОНФИГУРАЦИЯ СЕТИ.....	171
9.1. Базовая сетевая терминология.....	171
9.2. Терминология сетевых функций	172
9.3. IP-адреса.....	174
9.4. Конфигурация сетевых устройств	175
9.4.1. Настройка сети с помощью GUI	176
9.4.2. Настройка сети с использованием файлов конфигурации.....	179
9.5. Сетевые утилиты.....	182
9.5.1. Команда ifconfig.....	182
9.5.2. Команда route	183
9.5.3. Команда ping	184
9.5.4. Команда netstat	185
9.5.5. Команда dig	187
9.5.6. Команда host.....	187
9.5.7. Команда ssh	188

Глава 10. СИСТЕМНАЯ И ПОЛЬЗОВАТЕЛЬСКАЯ БЕЗОПАСНОСТЬ	191
10.1. Аккаунты пользователей.....	191
10.2. Каталог /etc	191
10.3. Группы	199
10.4. Суперпользователь	202
10.5. Команда SU	203
10.6. Команда sudo	203
10.7. Команда who.....	206
10.8. Команда w.....	207
10.9. Управление группами.....	208
10.10. Создание группы.....	209
10.11. Редактирование групп	211
10.12. Удаление групп	212
10.13. Файл /etc/default/useradd.....	212
10.14. Файл /etc/login.defs.....	213
10.15. Создание пользователя.....	214
10.16. Выбор пароля	218
10.17. Установка пароля.....	220
10.18. Команда chage	221
10.19. Изменение пользователя	221
10.20. Удаление пользователя	223
Глава 11. ПРАВА ДОСТУПА	225
11.1. Владелец файла	225
11.2. Команды newgrp и groups	226
11.3. Команды chgrp и stat.....	227
11.4. Команда chown.....	229
11.5. Разрешения	230
11.6. umask	233
11.7. Параметр setuid	234
11.8. setgid разрешение для файла.....	236
11.9. setgid разрешение для директории.....	237
11.10. Настройка разрешений setgid	238
11.11. Липкий бит	239
ЗАКЛЮЧЕНИЕ.....	241
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	242

ВВЕДЕНИЕ

Рассмотрены теоретические основы использования операционной системы Linux.

Первая глава содержит историю и теоретические основы операционной системы, информацию о распространении и лицензировании системы, описание основных способов взаимодействия с системой.

Вторая глава содержит описание основных команд терминала.

Третья глава содержит информацию о командах операционной системы Linux, с помощью которых можно получить справочную информацию.

Четвертая глава содержит информацию о командах, позволяющих выполнять операции над файлами и каталогами, такие как копирование, перемещение, создание, удаление, архивирование и сжатие.

Пятая глава содержит описание возможностей работы с потоками, поиском и сортировкой файлов, получением сведений и фильтрации содержимого файлов, а также использования базовых и расширенных регулярных выражений.

Шестая глава описывает процесс создания скриптов, а именно работу с переменными, условными выражениями и циклами.

Седьмая глава содержит сведения по работе с аппаратным обеспечением компьютеров.

Восьмая глава демонстрирует возможности операционных систем по управлению прикладным программным обеспечением и отслеживанию рабочих процессов.

Девятая глава описывает возможности по использованию, настройке и диагностике сетевых средств.

Десятая глава содержит информацию об основных файлах и командах для обеспечения безопасности на уровне разграничения пользователей, создания, редактирования и настройки групповой политики системы.

Одиннадцатая глава содержит описания основных команд, требуемых для настройки прав доступа пользователей и групп. Рассматриваются механизмы выдачи разрешений на файлы и директории.

В процессе изучения материала требуется использовать заранее подготовленный компьютер с предустановленной операционной системой Linux. Рекомендуется применить средства виртуализации. Все приведенные команды желательно воспроизвести на такой системе для получения практических навыков.

Глава 1. ВВЕДЕНИЕ В LINUX

1.1. Эволюция Linux и популярные операционные системы

Определение слова Linux зависит от контекста, в котором он используется. Linux означает ядро системы, которое является центральным контроллером всего, что происходит на компьютере (подробнее об этом позже). Люди, которые говорят, что их компьютер «работает под Linux», обычно имеют в виду ядро и набор инструментов, которые поставляются вместе с ним (так называемый дистрибутив). Если есть «опыт использования Linux», скорее всего, говорится о самих программах, хотя в зависимости от контекста можно говорить и о том, как точно настроить ядро. Каждый из этих компонентов будет исследован, чтобы точно понимать, какие роли играет каждый из них.

Дальше рассмотрим термин UNIX. Первоначально UNIX была операционной системой, разработанной в AT&T Bell Labs в 1970-х годах. Она была изменена и раздвоена (люди изменили ее, и эти изменения послужили основой для других систем), так что в настоящее время существует множество различных вариантов UNIX-систем. Однако UNIX теперь является товарным знаком и спецификацией, принадлежащей отраслевому консорциуму под названием Open Group. Только программное обеспечение, которое было сертифицировано Open Group, может назвать себя UNIX. Несмотря на принятие всех требований спецификации UNIX, Linux не сертифицирован, поэтому Linux действительно не UNIX! Он просто... UNIX-подобный.

1.1.1. Роль ядра

Ядро операционной системы похоже на диспетчер воздушного движения в аэропорту. Ядро диктует, какая программа какие части памяти получает, запускает и останавливает программы. Когда приложение что-либо считывает/записывает на диске, оно должно попросить операционную систему сделать это. Если два приложения запрашивают один и тот же ресурс, ядро решает, кто его получает, или в некоторых случаях останавливает одно из приложений, чтобы сохранить остальную часть системы.

Ядро также занимается переключением приложений. У компьютера всегда ограниченное количество процессоров и ограниченный

объем памяти. Ядро обеспечивает приостановку старой задачи и старт новой задачи, если задач больше, чем процессоров. Когда текущая задача выполняется достаточное количество времени, процессор приостанавливает задачу, чтобы другая задача могла решаться. Это называется приоритетной многозадачностью. «Многозадачность» означает, что компьютер выполняет сразу несколько задач, а «приоритетность», что ядро решает, когда нужно переключать фокус между задачами. С быстрым переключением задач кажется, что компьютер решает много задач одновременно, даже с одним вычислительным ядром.

Каждое приложение может решить, что у него большой блок памяти в системе, а ядро поддерживает эту иллюзию, переназначает меньшие блоки памяти, разделяет блоки памяти с другими приложениями или даже заменяет блоки, которые не были затронуты на диске.

Когда компьютер запускается, он загружает небольшой фрагмент кода, называемый загрузчиком. Задача загрузчика - загрузить ядро и запустить его. Пользователи Microsoft Windows или Apple macOS, чаще не видят загрузчика, но в мире UNIX — это обычно видно, чтобы пользователь мог настроить способ загрузки компьютера. Загрузчик всегда виден, если в системе установлено несколько разных операционных систем.

Загрузчик загружает ядро Linux, а затем передает управление. Затем Linux продолжает запускать программы, необходимые для того, чтобы сделать компьютер полезным, например, подключение к сети или запуск веб-сервера.

1.1.2. Приложения

Как и диспетчер воздушного движения, ядро бесполезно без управления. Если ядро — это башня, то приложения — это самолеты. Приложения обращаются к ядру и получают взамен ресурсы, такие как память, процессор и диск. Ядро также абстрагирует сложные детали от приложений. Приложение не знает, находится ли блок диска на твердотельном диске от производителя А, на вращающемся металлическом жестком диске от производителя В или даже в общем сетевом файле. Приложения просто следуют API-интерфейсу прикладного программирования ядра (API), поэтому не нужно беспокоиться о деталях реализации.

Когда пользователи думают о приложениях, то склонны думать о текстовых процессорах, веб-браузерах и почтовых клиентах. Ядру все равно, работает ли что-то, с чем сталкивается пользователь, сетевая служба, ведущая к удаленному компьютеру, или внутренняя задача. Итак, из этого получаем абстракцию, называемую процессом. Процесс — это всего лишь одна задача, которая загружается и отслеживается ядром. Для приложения может потребоваться несколько процессов, поэтому ядро заботится о запуске процессов, запуске и остановке их по запросу и раздаче системных ресурсов.

1.1.3. Роль открытого исходного кода

Linux началась в 1991 году в качестве хобби проекта Линуса Торвальдса. Он сделал источник исходного кода свободно доступным, а другие присоединились к нему, чтобы сформировать эту новую операционную систему. Linux не была первой системой, которую разрабатывала группа, но поскольку это был проект с нуля, ранние разработчики имели возможность влиять на направление проекта и следить, чтобы ошибки из других UNIX не повторялись.

Программные проекты имеют форму исходного кода, который представляет из себя набор компьютерных инструкций, читаемый пользователем. Исходный код может быть написан на любом из сотен разных языков. Linux написан на языке C, который тесно связан с историей исходной UNIX.

Исходный код не разбирается непосредственно компьютером, поэтому он должен быть скомпилирован в машинные инструкции компилятором. Компилятор собирает все исходные файлы и генерирует то, что может быть запущено на компьютере, например, ядро Linux.

Исторически сложилось так, что большинство программ было выпущено по лицензии с закрытым исходным кодом, что означает, что пользователь получает право использовать машинный код, но не может видеть исходный код. Часто лицензия специально говорит о том, что пользователь не должен пытаться переводить машинный код обратно в исходный код, чтобы выяснить, что он делает!

Философия с открытым исходным кодом заключается в том, что любой имеет право получить программное обеспечение (ПО) и изме-

нить его для собственного использования. Linux принял эту философию с большим успехом. Люди взяли источник, внесли изменения и поделились ими с остальной группой.

Наряду с этим был проект GNU (рекурсивный акроним GNU's Not Unix!). Основатель проекта GNU Ричард Столлман – основатель движения свободного программного обеспечения. GNU – набор свободного программного обеспечения, которое можно использовать в качестве операционной системы. GNU был эффективным при создании инструментов, которые сочетаются с операционной системой UNIX, такими как компиляторы и программы пользовательского интерфейса (CLI – command line interface, интерфейса командной строки). Исходные коды были свободно доступны, поэтому Linux смог настроить рекламу своих инструментов и представить полную систему. Таким образом, большинство инструментов, которые являются частью системы Linux, взяты из инструментов GNU. Часто говорят не просто Linux, а GNU/Linux, подразумевая сочетания ядра Linux и инструментов GNU.

Существует много разных вариантов открытого исходного кода, и они будут рассмотрены в следующей главе.

1.1.4. Распределения Linux

Возьмите ядро Linux и инструменты GNU, добавьте еще несколько пользовательских приложений, таких как клиент электронной почты, и у вас есть полная система Linux. Люди начали собирать все это программное обеспечение в дистрибутив почти сразу после того, как Linux стал пригодным для использования. Полноценные дистрибутивы также включают инструменты для управления системой и диспетчером пакетов, которые помогут добавить и удалить программное обеспечение после завершения установки.

Подобно UNIX, существует множество различных вариантов дистрибутивов. В наши дни существуют дистрибутивы, ориентированные на запуск серверов, настольных компьютеров или даже отраслевых инструментов, таких как различная электроника или сетевое оборудование. Основных игроков на рынке можно свести к двум основным линейкам: Red Hat и Debian. Наиболее заметным отличием является менеджер пакетов, хотя найдутся и другие различия: от местоположения файлов до политических философий.

Red Hat началась как простой дистрибутив, который представил Red Hat Package Manager (RPM). Разработчик создал компанию вокруг нее, которая пыталась коммерциализировать рабочий стол Linux для бизнеса. Со временем компания Red Hat стала больше ориентироваться на серверные приложения и выпустила Red Hat Enterprise Linux (RHEL), которая была платной в течение длительного цикла выпуска. Цикл выпуска определяет, как долго обновляется программное обеспечение. Бизнес ценит стабильность и требует длительных циклов выпуска. Любитель или стартапер, возможно, захочет получить последнее программное обеспечение и выбрать более короткий цикл выпуска. Чтобы удовлетворить последнюю группу, Red Hat спонсирует проект Fedora, который делает персональный рабочий стол, содержащий новейшее программное обеспечение, но все еще построенный на той же базе, что и корпоративная версия.

Поскольку все в Red Hat Enterprise Linux является открытым исходным кодом, появился проект под названием CentOS, который перекомпилировал все пакеты RHEL и предоставил их бесплатно. CentOS и другие подобные (такие как Scientific Linux) в значительной степени совместимы с RHEL и интегрируют несколько новых программ, но не предлагают платной поддержки Red Hat.

Scientific Linux - пример конкретного использования, основанного на Red Hat. Проект представляет собой спонсируемый Fermilab дистрибутив, предназначенный для научного вычисления. Среди многочисленных приложений Scientific Linux используется с ускорителями частиц, включая Большой адронный коллайдер в CERN.

Open SUSE изначально получен из Slackware, но включает множество аспектов Red Hat. Оригинальная компания была приобретена Novell в 2003 году, которая затем была приобретена Attachmate Group в 2011 году. Затем группа Attachmate объединилась с Micro Focus International. Благодаря всем слияниям и поглощениям SUSE удалось продолжить и расти. Хотя Open SUSE является настольным и доступным для широкой публики, SUSE Linux Enterprise содержит проприетарный код и продается как серверный продукт.

Debian — это вложение еще больших усилий сообщества, что способствует использованию программного обеспечения с открытым исходным кодом и соблюдению стандартов. В Debian появилась своя система управления пакетами, основанная на формате файла .deb. В то

время как Red Hat поддерживается Intel и AMD, Debian поддерживает многие из этих платформ напрямую.

Ubuntu является самым популярным дистрибутивом Debian. Это создание Canonical - компании, которая была создана для дальнейшего роста Ubuntu и зарабатывания денег, оказывая поддержку.

Linux Mint был запущен как ответвление Ubuntu Linux, но использующее репозитории Ubuntu. Существуют разные версии, все они бесплатны, но некоторые из них включают проприетарные кодеки, которые нельзя распространять без ограничений лицензии в некоторых странах. Linux Mint делят с Ubuntu лидирующие позиции для тех, кто знакомится с Linux впервые и переходит, например, с Microsoft Windows.

Российскими компаниями разработано несколько операционных систем. Почти все они созданы на базе ядра Linux: Альт Линукс, Astra Linux, ROSA Linux, Эльбрус, Ред ОС, ГосЛинукс и др. Однако не верно полагать, что все новые операционные системы разрабатываются на ядре Linux. Есть операционные системы, разрабатываемые с нуля, например, KasperskyOS или ОСПВ «МАКС» (операционная система реального времени). Такие операционные системы как правило используются для специализированного оборудования и не рассчитаны на массовый рынок.

Необходимо понимать, что есть сотни, если не тысячи других дистрибутивов Linux, которые доступны в настоящий момент. Важно понимать, что, хотя существует множество разных дистрибутивов Linux, большинство из используемых в них программ и команд остаются теми же или очень похожими.

1.1.4.1. Что такое команда?

Самый простой ответ на вопрос «Что такое команда?» заключается в том, что команда — это программная инструкция, которая при выполнении в командной строке выполняет действие на компьютере.

Если рассматривать команду в такой терминологии, то можно увидеть, что происходит, когда команда выполняется. Когда команда вводится на исполнение, операционной системой запускается процесс, который может считывать данные, манипулировать данными и выводить данные. С этой точки зрения, команда запускает процесс в операционной системе, который затем заставляет компьютер выполнять задание.

Однако есть еще один способ взглянуть на команду: посмотреть на ее источник. Источник — это то, откуда команда «происходит», и в оболочке CLI есть несколько разных источников команд:

1. Примером команды, которая встроена в оболочку, является команда `cd`, поскольку она является частью оболочки `bash`. Когда пользователь вводит команду `cd`, оболочка `bash` уже выполняет и знает, как интерпретировать эту команду, не требуя запуска дополнительных программ.
2. Примером команды, которые хранятся в файлах, просматриваемых оболочкой, является `ls`. Если вы наберете команду `ls`, оболочка будет искать в каталогах, перечисленных в переменной `PATH`, файл с именем `ls`, который она может выполнить. Эти команды также могут быть выполнены путем ввода полного пути к команде.
3. Псевдоним может переопределять встроенную команду, функцию или команду, найденную в файле. Псевдонимы могут быть полезны для создания новых команд, созданных из существующих функций и команд.
4. Функции также могут быть созданы с использованием существующих команд для создания новых команд, переопределения команд, встроенных в оболочку или команд, хранящихся в файлах. Псевдонимы и функции обычно загружаются из файлов инициализации при первом запуске оболочки, которые обсуждаются ниже в этом разделе.

Хотя псевдонимы будут подробно рассмотрены в следующем разделе, этот краткий пример может помочь в понимании концепции команд.

Псевдоним — это, по существу, псевдоним для другой команды или серии команд. Например, команда `cal 2023` отобразит календарь на 2023 год. Предположим, что пользователь часто выполняет эту команду. Вместо выполнения полной команды каждый раз можно создать псевдоним, называемый `mycal`, и запустить псевдоним, как показано далее:

```
test@test-VirtualBox:~$ alias mycal="cal 2023"
test@test-VirtualBox:~$ mycal

2023
```

Январь							Февраль							Март						
Вс	Пн	Вт	Ср	Чт	Пт	Сб	Вс	Пн	Вт	Ср	Чт	Пт	Сб	Вс	Пн	Вт	Ср	Чт	Пт	Сб
1	2	3	4	5	6	7				1	2	3	4				1	2	3	4
8	9	10	11	12	13	14	5	6	7	8	9	10	11	5	6	7	8	9	10	11
15	16	17	18	19	20	21	12	13	14	15	16	17	18	12	13	14	15	16	17	18
22	23	24	25	26	27	28	19	20	21	22	23	24	25	19	20	21	22	23	24	25
29	30	31					26	27	28					26	27	28	29	30	31	

Апрель							Май							Июнь						
Вс	Пн	Вт	Ср	Чт	Пт	Сб	Вс	Пн	Вт	Ср	Чт	Пт	Сб	Вс	Пн	Вт	Ср	Чт	Пт	Сб
						1		1	2	3	4	5	6					1	2	3

1.1.5. Аппаратные платформы

Linux начинал работать на компьютере с конкретным контроллером жесткого диска. Диапазон поддержки вырос, поскольку люди построили поддержку другого оборудования. В конце концов, Linux начал поддерживать другие чипы, включая аппаратное обеспечение, которое было создано для запуска конкурентных операционных систем!

Типы оборудования выросли от скромного чипа Intel до суперкомпьютеров. Позже, более мелкие, поддерживаемые Linux, чипы были разработаны, чтобы соответствовать потребительским устройствам, называемыми встроенными устройствами. Поддержка Linux стала повсеместной, так что часто стало проще создавать аппаратные средства для поддержки Linux, а затем использовать Linux в качестве плацдарма для своего пользовательского программного обеспечения, чем создавать с нуля нестандартное аппаратное и программное обеспечение.

В конце концов, сотовые телефоны и планшеты начали работать под управлением Linux. Компания, купленная позже Google, придумала платформу Android, которая представляет собой комплект Linux и программное обеспечение, необходимое для работы с телефоном или планшетом. Это дает меньшее количество усилий по выходу на рынок,

а компании могут потратить свое время на разработку программного обеспечения для пользователей, а не изобретать велосипед каждый раз.

На российском рынке существует мобильная операционная система Аврора, основанная на Sailfish OS, которая была построена на базе проекта Mer, который является ответвлением проекта MeeGo компаний Nokia, Intel и Linux Foudation на ядре Linux.

Помимо телефонов и планшетов, Linux можно найти на многих потребительских устройствах. Беспроводные маршрутизаторы почти всегда запускают Linux, потому что он имеет богатый набор сетевых функций. Телевизионные приставки и встроенные Smart функции телевизоров тоже используют различные операционные системы на базе ядра Linux. Несмотря на то, что эти устройства имеют ядро Linux, конечные пользователи не обязаны это знать. Специализированное пользовательское программное обеспечение взаимодействует с пользователем, а Linux обеспечивает стабильную платформу за счет ядра. В отличие от универсальных и полноценных дистрибутивов в таких операционных системах чаще всего нет программ GNU.

1.2. Выбор операционной системы

Linux — это UNIX-подобная операционная система, а это означает, что она не прошла официальную сертификацию и поэтому не может использовать официальный товарный знак UNIX. Существует много альтернатив; некоторые из них похожи на UNIX, а некоторые из них сертифицированы как UNIX. Существуют также не-Unix-операционные системы, такие как Microsoft Windows.

Самый важный вопрос, который следует задать при определении конфигурации машины, - «Что будет делать эта машина?» Если вам нужно запустить специализированное программное обеспечение, которое работает только в Oracle Solaris, тогда это то, что вам нужно. Если вам нужно читать и писать документы Microsoft Office, вам понадобится Windows или что-то, что может работать с LibreOffice или OpenOffice.

1.2.1. Пункты принятия решений

Первое, что нужно решить, это определить роль машины. Нужен ли графический интерфейс, веб-браузер и другие подобные программы? Если это так, должен быть рабочий стол. Будет ли машина

использоваться в качестве веб-сервера, или она будет находиться где-то в серверной стойке? Если да, то в этом случае необходим сервер.

Серверы обычно находятся в стойке и совместно используют клавиатуру и монитор со многими другими компьютерами, поскольку доступ к консоли используется только для настройки и устранения неполадок с сервером. Сервер будет работать в неграфическом режиме, который освобождает ресурсы для реальной цели компьютера. На рабочем столе в основном будет работать графический интерфейс.

Затем определяются функции машины. Требуется ли конкретное программное обеспечение для выполнения или какие функции необходимо выполнять? Нужно иметь возможность управлять сотнями или тысячами этих машин одновременно? Каков набор навыков команды, управляющей компьютером и программным обеспечением?

Также нужно определить срок службы и степень риска для сервера. Операционные системы и обновления программного обеспечения происходят на периодической основе, называемой циклом выпуска. Поставщики программного обеспечения будут поддерживать старые версии программного обеспечения в течение определенного периода времени, прежде чем перестанут предлагать обновлений, которые называются циклом обслуживания (или жизненным циклом). Например, основные выпуски Fedora Linux выпускаются примерно каждые 6 месяцев. Версии считаются End of Life (EOL) после двух основных версий плюс один месяц, поэтому у вас есть от 7 до 13 месяцев после установки Fedora, прежде чем нужно будет ее обновить. Сравните это с вариантом коммерческого сервера Red Hat Enterprise Linux, может пройти до 13 лет, прежде чем потребуется обновление.

Циклы обслуживания и выпуска важны, потому что в среде корпоративного сервера отнимается много времени и, следовательно, редко, выполняется серьезная модернизация на сервере. Вместо этого сам сервер заменяется, когда в приложение требуются основные обновления или замены, требующие обновления операционной системы. Точно так же медленный цикл релиза важен, потому что приложения часто нацелены на текущую версию операционной системы, и желательно избегать накладных расходов на обновление серверов и операционных систем, чтобы «идти в ногу со временем». Существует большая работа, связанная с обновлением сервера, и в роли сервера часто

приходится выполнять множество настроек, которые трудно переносить на новый сервер. Это требует гораздо большего тестирования, чем если бы было обновлено только приложение.

Если заниматься разработкой программного обеспечения или традиционной работой на рабочем столе, часто требуется новейшее программное обеспечение. Более новое программное обеспечение имеет улучшенные функциональные возможности и внешний вид, что способствует получению удовольствия от использования компьютера. Рабочий стол часто хранит свою работу на удаленном сервере, поэтому рабочий стол можно очистить.

Отдельные версии программного обеспечения можно охарактеризовать как «бета» или «стабильные». Одной из замечательных особенностей создания разработки с открытым исходным кодом является то, что можно выпускать новое программное обеспечение и быстро получать отзывы от пользователей. Если выпуск программного обеспечения находится в состоянии, когда у него много новых функций, которые не были тщательно протестированы, его обычно называют «бета-версией». После того, как эти функции были протестированы, программное обеспечение переходит в «стабильную» стадию. Если нужны последние функции, то надо выбрать дистрибутив, который имеет быстрый цикл выпуска и упрощает использование бета-версии ПО.

Другая слабосвязанная концепция - обратная совместимость. Это относится к возможности совместимости более поздней ОС с программным обеспечением, созданным для более ранних версий. Обычно это вызывает беспокойство, если нужно обновить ОС, но нет возможности обновить прикладное ПО.

Стоимость чаще всего является фактором «номер один». Сам Linux может быть бесплатным, но может потребоваться заплатить за поддержку. Microsoft имеет платную серверную лицензию и может иметь дополнительные расходы на поддержку в течение всего срока службы сервера. Выбранная ОС может работать только на определенном наборе аппаратных средств, что дополнительно влияет на стоимость.

1.2.2. Microsoft Windows

Microsoft разбивает операционные системы в соответствии с назначением машины: рабочий стол или сервер? Настольная версия

Windows подверглась различным схемам именования вплоть до последней (на момент написания этого материала), которая представляет собой просто Windows 11. Новые версии рабочего стола выходили каждые 3-5 лет и, как правило, поддерживались в течение многих лет. Обратная совместимость также является приоритетом для Microsoft, даже для того, чтобы объединить технологии виртуальных машин, чтобы пользователи могли запускать более старое программное обеспечение.

В области сервера есть Windows Server. В названии версии обычно указывается год выпуска/начала продаж. Сервер работает с графическим интерфейсом, но в значительной степени в качестве конкурентного ответа на Linux, добился огромных успехов в средствах командной строки благодаря PowerShell. Также можно сделать сервер похожим на рабочий стол с дополнительным пакетом Desktop Experience.

1.2.3. Apple macOS

Apple выпускает проприетарную операционную систему macOS (ранее называлась Mac OS X и OS X), которая прошла сертификацию UNIX. macOS частично основана на ПО проекта FreeBSD.

В настоящий момент macOS — это, в основном, настольная ОС, которая имеет дополнительные пакеты, помогающие управлять сетевыми службами, которые позволяют многим рабочим столам macOS взаимодействовать, например, обмениваться файлами или иметь сетевой учет.

macOS на рабочем столе обычно является личным решением, так как многие находят систему более простой в использовании. Растущая популярность macOS обеспечила здоровую поддержку со стороны поставщиков программного обеспечения. macOS также довольно популярен в таких творческих отраслях, как производство видео. Это одна из областей, где приложения оказывает влияние на выбор операционной системы и, следовательно, выбор аппаратуры, поскольку macOS работает только на аппаратном обеспечении Apple.

1.2.4. BSD

Существует несколько проектов с открытым исходным кодом BSD (Berkely Software Distribution), таких как OpenBSD, FreeBSD и NetBSD. Во многих отношениях это альтернативы Linux, поскольку они используют большое количество общего ПО. BSD обычно реали-

зуются в роли сервера, хотя существуют варианты, использующие рабочие столы GNOME и KDE, которые были разработаны для настольных целей.

1.2.5. Другие коммерческие UNIX

Некоторые из наиболее популярных коммерческих UNIX:

- Oracle Solaris
- IBM AIX
- HP-UX

Каждый из них работает на оборудовании от их соответствующих создателей. Аппаратное обеспечение обычно является большим и мощным, предлагая такие функции, как процессор и память с возможностью «горячей» замены, или интеграция с устаревшими системами мейнфреймов, также предлагаемыми поставщиком.

Если ПО не требует определенного оборудования или приложения, требуется некоторая избыточность, встроенная в аппаратное обеспечение, большинство людей предпочитают выбирать эти параметры, поскольку они уже являются пользователями продуктов компании. Например, IBM AIX работает с широким спектром оборудования IBM и может совместно использовать аппаратное обеспечение с мейнфреймами. Таким образом, AIX используют в компаниях, у которых уже есть большой опыт использования оборудования IBM, или которые используют программное обеспечение IBM, такое как WebSphere.

1.2.6. Linux

Один из аспектов, в котором Linux сильно отличается от альтернатив, заключается в том, что после того, как администратор выбрал Linux, дальше все равно придется выбирать дистрибутив. Дистрибутив совмещает ядро Linux, утилиты и инструменты управления в устанавливаемый пакет и предоставляет способ установки и обновления пакетов после первоначальной установки.

Некоторые операционные системы доступны только одному поставщику, например, macOS и Windows, с поддержкой системы через поставщика. В Linux существует множество вариантов - от коммерческих предложений для сервера или рабочего стола - до пользовательских предложений, предназначенных для превращения старого компьютера в сетевой брандмауэр.

Часто поставщики приложений выбирают подмножество дистрибутивов для поддержки. Различные дистрибутивы имеют разные версии ключевых библиотек, и компаниям сложно поддерживать все эти разные версии.

В разных дистрибутивах есть разные циклы выпуска, иногда короткие (каждые шесть месяцев), иногда длинные (два года и более). Каждая версия может поддерживаться только в течение определенного разумного промежутка времени. Некоторые версии Linux, имеют долгосрочную поддержку (LTS) 5 лет или более, в то время как другие будут поддерживаться только в течение двух лет или менее.

Существует несколько дистрибутивов различных версий: стабильные, тестовые и неустойчивые. Разница заключается в том, что в нестабильных версиях производитель освобождается от ответственности за надежное функционирование системы. Когда функции были интегрированы в систему в течение длительного времени, и многие из проблем были решены, ПО переходит через тестирование в стабильный выпуск. Распространение Debian предупреждает пользователей о проблемах с использованием выпуска «sid» со следующим предупреждением:

«Sid» подвержен значительным изменениям и обновлениям библиотек. Это может привести к очень «нестабильной» системе, содержащей пакеты, которые невозможно установить из-за отсутствия библиотек или зависимостей, которые невозможно установить и т.д. Вы используете их на свой страх и риск!

Другие релизы зависят от бета-дистрибутивов. Например, дистрибутив Fedora выпускает бета-версию или предварительные выпуски своего ПО перед полным выпуском, чтобы свести к минимуму ошибки. Fedora часто считается общей бета-версией RedHat. Функции добавляются и изменяются в релизе Fedora прежде, чем обнаруживаются в дистрибутиве RedHat Enterprise.

1.2.7. Android

Android, спонсируемый Google, является самым популярным в мире дистрибутивом Linux. Он принципиально отличается от аналогов.

Android использует виртуальную машину Dalvik с Linux, предоставляя надежную платформу для мобильных устройств, таких как телефоны и планшеты. Однако, не имея традиционных пакетов, которые часто распространяются с Linux (например, GNU и Xorg), Android, как правило, несовместим с настольными дистрибутивами Linux.

Эта несовместимость означает, что пользователь RedHat или Ubuntu не может загружать ПО из магазина Google Play. Возможно в будущем будет реализован полный функционал запуска из других систем через функции эмуляции. Подобный функционал частично реализован в Windows. Аналогично, терминальному эмулятору в Android не хватает многих команд его коллег из Linux. Однако можно использовать BusyBox или другое ПО с Android, чтобы большинство команд могли работать. Желющие использовать GNU команды на мобильных устройствах с Android могут установить приложение Termux или поискать его аналоги.

1.3. Основные приложения с открытым исходным кодом

Ядро Linux может запускать широкий спектр ПО на многих аппаратных платформах. Компьютер может действовать как сервер, что означает, что он в основном обрабатывает данные от имени другого пользователя или может работать как рабочий стол, что означает, что пользователь будет взаимодействовать с ним напрямую. Машина может запускать ПО или может использоваться как машина разработки в процессе создания ПО. Даже можно запускать несколько ролей, поскольку нет никакого различия в Linux между ролями машины; это просто вопрос настройки приложений.

Одно из преимуществ заключается в том, что можно моделировать практически все аспекты производственной среды: от разработки до тестирования и проверки на сокращенном аппаратном обеспечении, что экономит затраты и время. Изучая Linux, можно запускать серверные приложения на своем рабочем столе, которые запускаются большим интернет-провайдером. Конечно, нельзя справиться с объемом нагрузки, который может быть у реального сервера, поскольку у него будет гораздо более дорогое оборудование. Но можно имитировать практически любую конфигурацию, не требуя мощного лицензирования оборудования или сервера.

Программное обеспечение Linux обычно относится к одной из трех категорий:

1. Серверное ПО - ПО, которое не имеет прямого взаимодействия с монитором и клавиатурой машины, на которой он работает. Его цель - обслуживать информацию на других компьютерах, называемых клиентами. Иногда серверное ПО может не взаимодействовать с другими компьютерами, а просто обрабатывать данные.
2. Настольное ПО - веб-браузер, текстовый редактор, музыкальный плеер или другое ПО, с которым взаимодействует человек. Во многих случаях подобное ПО взаимодействует с сервером через компьютерную сеть и интерпретирует данные для конечного пользователя. Здесь настольным ПО является клиент.
3. Инструменты - свободная категория ПО, которая существует, чтобы упростить управление системой. Это может быть инструмент, который поможет настроить дисплей или что-то еще, что обеспечивает стабильную работу оболочки Linux, или даже более сложные инструменты, которые конвертируют исходный код в то, что компьютер может выполнить (системное программное обеспечение - СПО).

Кроме того, есть мобильные приложения. Мобильное приложение очень похоже на настольное приложение, но оно работает на телефоне или планшете вместо настольного компьютера.

Любая задача, которую требуется выполнить в Linux, может быть решена с помощью разных приложений. Существует много веб-браузеров, множество веб-серверов и множество текстовых редакторов (преимущества каждого из них являются предметом многих священных войн UNIX). Это ничем не отличается от мира, закрытого ПО. Однако преимущество открытого исходного кода заключается в том, что, если кому-то не нравится, как работает, например, веб-сервер, можно начать создавать свой собственный или изменить код существующего.

1.3.1. Серверные приложения

Linux часто используют для запуска серверных приложений из-за его надежности и эффективности. При рассмотрении серверного ПО наиболее важным вопросом является: «Какой сервис запускается?». Если требуется обслуживать веб-страницы, понадобится ПО веб-сервера, а не почтовый сервер!

Одно из ранних применений Linux как раз было применение в качестве веб-серверов. Веб-сервер размещает контент для веб-страниц, который просматривается веб-браузером с использованием протокола гипертекстовой передачи (HTTP) или его зашифрованного варианта (HTTPS). Сама веб-страница может быть статичной. Это означает, что, когда веб-браузер запрашивает страницу, веб-сервер просто отправляет в ответ содержимое файла. Сервер также может выдавать динамический контент. Это означает, что запрос отправляется веб-сервером в приложение, которое генерирует контент. WordPress - один из популярных примеров. Это набор скриптов на языке PHP. При каждом запросе к сайту скрипты формируют содержимое отображаемой страницы. Пользователи могут разрабатывать контент через свой браузер в приложении WordPress, и ПО превращает его в полностью функциональный веб-сайт. Каждый раз, когда совершаются покупки в Интернете, пользователи обращаются к динамическому содержимому сайта.

Apache является доминирующим веб-сервером, который используется сегодня. Первоначально Apache был автономным проектом. Позднее была организована организация Apache Software Foundation, которая поддерживает более ста программных проектов с открытым исходным кодом.

Другой популярный веб-сервер - nginx был разработан российским программистом Игорем Владимировичем Сысоевым. Он фокусируется на производительности, используя более современные ядра UNIX и делает часть того, что может сделать Apache. Более 65% веб-сайтов работают на nginx или Apache.

Электронная почта всегда была популярной для серверов Linux. При обсуждении серверов электронной почты всегда полезно посмотреть на 3 разных роли, необходимых для обмена сообщениями через электронную почту между людьми:

5. Агент передачи почты (Mail Transfer Agent, MTA) – указывает как пересылаются сообщения от сервера к серверу и какой сервер в итоге должен получить сообщение электронной почты; использует протокол SMTP для перемещения электронной почты между серверами.
6. Агент доставки почты (Mail Delivery Agent, MDA; также называемый агент локальной доставки) - заботится о сохранении электронной почты в почтовом ящике пользователя.

7. POP / IMAP-сервер. Протокол почтового отделения (The Post Office Protocol) и протокол доступа к Интернет-сообщениям (Internet Message Access Protocol) — это два протокола обмена сообщениями, которые позволяют почтовому клиенту, работающему на компьютере, «общаться» с сервером, чтобы получить электронное письмо.

Иногда часть ПО реализует несколько компонентов. В мире с закрытым исходным кодом Microsoft Exchange реализует все компоненты, поэтому нет возможности сделать индивидуальный выбор. В мире с открытым исходным кодом существует много вариантов.

Наиболее известными МТА являются sendmail и Postfix. Кроме того, можно использовать что-то вроде procmail, которое позволяет определять настраиваемые фильтры для обработки почты и фильтрации.

Dovecot - популярный сервер POP / IMAP благодаря простоте использования и низкому обслуживанию. Cyrus IMAP - еще один вариант.

Для совместного использования файлов Samba является явным победителем. Samba позволяет машине Linux выглядеть как компьютер Windows, чтобы он мог обмениваться файлами и участвовать в домене Windows. Samba реализует серверные компоненты, такие как создание файлов для совместного использования и определения ролей сервера Windows, а также клиента, чтобы машина Linux могла использовать общий доступ к файлам Windows.

Если в локальной сети есть компьютеры Apple, проект Netatalk позволяет машине Linux вести себя как файловый сервер Apple.

Собственный протокол обмена файлами для UNIX называется Сетевой файловой системой (Network File System, NFS). NFS обычно является частью ядра, что означает, что удаленная файловая система может быть установлена как обычный диск, что делает доступ к файлам прозрачным для других приложений.

По мере увеличения компьютерной сети потребуется реализовать какой-либо каталог. Самый старый каталог называется системой доменных имен (Domain Name System, DNS) и используется для преобразования имени, подобного <http://www.linux.com> в IP-адрес, такой как 192.168.100.100, который является уникальным идентификатором

этого компьютера в Интернете или локальной сети. DNS также содержит такую глобальную информацию, как адрес MTA для данного имени домена. Организация может запустить собственный DNS-сервер для размещения своих имен, обращенных к интернету, а также для работы в качестве внутреннего каталога. Консорциум Internet Software Consortium поддерживает самый популярный DNS-сервер, называемый bind. Так же называется процесс, который запускает службу.

DNS в основном сосредоточен на именах компьютеров и IP-адресах и недоступен для поиска. Другие каталоги возникли для хранения другой информации, такой как учетные записи пользователей и роли безопасности. Легкий протокол доступа к каталогам (Lightweight Directory Access Protocol, LDAP) является наиболее распространенным каталогом, который также поддерживает Microsoft Active Directory. В LDAP объект хранится в древовидной структуре, и позиция этого объекта в дереве может использоваться для получения информации об объекте в дополнение к тому, что хранится с самим объектом. Например, администратор Linux может быть сохранен в ветке дерева под названием «ИТ-отдел», который находится под веткой «Отделы». Таким образом, весь технический персонал можно найти в ветке под названием «ИТ-отдел». OpenLDAP – самая популярная реализация каталога на основе LDAP.

Одной из частей сетевой инфраструктуры является протокол динамической конфигурации хоста (Dynamic Host Configuration Protocol, DHCP). Когда компьютер загружается, ему требуется IP-адрес для локальной сети, чтобы он мог быть однозначно идентифицирован. Задача DHCP – прослушивать запросы и назначать бесплатный адрес из пула DHCP. Консорциум Internet Software Consortium также поддерживает сервер ISC DHCP, который является наиболее распространенным.

База данных хранит информацию, а также позволяет легко извлекать и запрашивать данные. Самыми популярными базами данных являются MySQL и PostgreSQL. Можно ввести данные в базу данных, а затем использовать язык под названием Structured Query Language (SQL) для формирования запросов.

1.3.2. Настольные приложения

В экосистеме Linux имеется множество настольных приложений. Можно найти игры, приложения, инструменты для творчества и многое другое. Этот раздел является очень кратким обзором того, что есть.

Прежде чем смотреть на отдельные приложения, полезно взглянуть на среду рабочего стола. Рабочий стол Linux запускает систему под названием X Window, также известную как X11. Сервер Linux X11 — это X.org, который предоставляет возможность ПО работать в графическом режиме и принимать ввод с клавиатуры и мыши. Окна и ярлыки обрабатываются другим ПО, называемым диспетчером окон или средой рабочего стола. Диспетчер окон — это простая версия среды рабочего стола, поскольку она предоставляет собой код для рисования меню и управления окнами приложений на экране. Уровень среды рабочего стола в таких функциях, как окна входа в систему, сеансы, файловый менеджер и другие утилиты. Таким образом, текстовая рабочая станция Linux становится графическим рабочим столом с добавлением X-Windows и среды рабочего стола или оконного менеджера.

Существует много менеджеров окон, например, Compiz, FVWM и другие. Настольные среды — это в первую очередь KDE и GNOME, оба из которых имеют свои собственные оконные менеджеры. И KDE, и GNOME — это зрелые проекты с невероятным количеством утилит, выбор которых часто является вопросом личных предпочтений. Как правило авторы дистрибутивов Linux заранее сами выбирают одну из используемых сред рабочего стола. Для некоторых дистрибутивов можно выбрать один из вариантов. Например, при установке Linux Mint можно выбрать среды Cinnamon, MATE или XFCE.

Очень важны базовые приложения для выполнения рабочих задач, такие как текстовый редактор, программы для работы с электронными таблицами и презентациями. В совокупности такие программы известны как офисный пакет, в основном из-за пакета программ Microsoft Office, который является доминирующим на рынке.

OpenOffice (иногда называемый OpenOffice.org) и LibreOffice предлагают полный офисный пакет, который стремится к совместимости с Microsoft Office как с точки зрения функций, так и с форматами файлов. Эти два проекта также являются примером того, как политика влияет на открытый исходный код.

В 1999 году Sun Microsystems приобрела относительно малоизвестную немецкую компанию, которая создала офисный пакет для Linux под названием StarOffice. Вскоре после этого Sun переименовала его в OpenOffice и выпустила его под лицензией с открытым исходным кодом. Чтобы еще больше усложнить ситуацию, StarOffice оставался проприетарным продуктом, который выпускался и имел общий код с OpenOffice. В 2010 году Sun была приобретена компанией Oracle, которая позже передала проект в Apache Foundation.

У Oracle была плохая история поддержки проектов с открытым исходным кодом, которые она приобретает, поэтому вскоре после приобретения Oracle проект был разветвлен, чтобы стать LibreOffice. В этот момент появились две группы людей, разрабатывающих один и тот же программный продукт. Большая часть сил была направлена на проект LibreOffice, поэтому он включен по умолчанию во многих дистрибутивах Linux. Пакет LibreOffice может быть установлен на Linux, Windows и macOS.

Для просмотра веб-страниц используются разные браузеры. Двумя основными являются Firefox и Google Chrome. Оба являются веб-браузерами с открытым исходным кодом, которые являются быстрыми, многофункциональными и имеют отличную поддержку для веб-разработчиков. Эти два пакета являются хорошим примером того, как разнообразие выгодно для открытого исходного кода - улучшения для одного стимулируют другую команду, чтобы быть лучше других. В результате в Интернете есть два отличных браузера, которые выходят за рамки того, что можно сделать в Интернете и работать на разных платформах. Многие браузеры, такие как Opera, Yandex, Edge, Chrome, используют в своей основе браузер под названием Chromium.

Проект Mozilla выпускает популярный Thunderbird, полнофункциональный почтовый клиент для настольных компьютеров. Thunderbird подключается к серверу по протоколам POP или IMAP, локально отображает электронную почту и отправляет электронную почту через внешний SMTP-сервер. Другими известными почтовыми клиентами являются Evolution и KMail, которые являются почтовыми клиентами проекта GNOME и KDE. Стандартизация через POP, IMAP означает возможность легкого переключения между почтовыми клиентами без потери данных. Веб-адрес электронной почты также является еще одним вариантом доступа к электронной почте через браузер.

Для обработки мультимедиа файлов существуют программы Blender, GIMP и Audacity, которые созданы для создания 3D-фильмов, обработки 2D-изображений и редактирования звука соответственно. У них были разные успехи на профессиональных рынках. Например, Blender используется для всего, от любительских до голливудских фильмов.

1.3.3. Инструменты консоли

История разработки UNIX показывает значительное совпадение навыков разработки программного обеспечения и системного администрирования. Инструменты, которые позволяют управлять системой, имеют функции компьютерных языков, такие как циклы, а некоторые компьютерные языки широко используются для автоматизации задач администрирования систем. Таким образом, следует учитывать эти дополнительные навыки.

На базовом уровне достаточно взаимодействия с системой Linux через оболочку (shell) независимо от того, происходит ли подключение к системе удаленно или локально. Задача оболочки - принимать команды, такие как манипуляции с файлами и запускающие приложения, и передавать их в ядро Linux для выполнения. Здесь отображается типичное взаимодействие с оболочкой Linux:

```
sysadmin@localhost:~$ ls -l /tmp/*.gz
-rw-r--r-- 1 sean root 246841 Mar 5 2021 /tmp/fdboot.img.gz
sysadmin@localhost:~$ rm /tmp/fdboot.img.gz
```

Пользователю предоставляется подсказка, которая обычно заканчивается значком доллара \$, чтобы указать непривилегированную учетную запись. Все, что перед приглашением, в этом случае `sysadmin@localhost:~`, представляет собой настраиваемое приглашение, которое предоставляет дополнительную информацию пользователю. На приведенном выше рисунке `sysadmin` - это имя текущего пользователя, `localhost` - это имя сервера, а `~` - текущий каталог (в UNIX символ тильды является короткой формой домашнего каталога пользователя). Более подробно команды Linux будут рассмотрены в дальнейших главах, но, чтобы закончить объяснение, первая команда `ls` перечисляет файлы, получает некоторую информацию о файле и затем пользователь удаляет этот файл с помощью команды `rm`.

Оболочка Linux предоставляет богатый язык для обработки файлов и настройки среды, без выхода из оболочки. Например, можно

написать одну командную строку, которая находит файлы с содержимым, соответствующим определенному шаблону, извлекает полезную информацию из файла и затем копирует новую информацию в новый файл.

Linux предлагает множество shell оболочек на выбор, в основном отличающихся тем, как и что можно настроить, и синтаксисом встроенного языка сценариев. Двумя основными семействами являются оболочка Bourne и оболочка C. Оболочка Bourne была названа в честь создателя, а оболочка C была названа, потому что синтаксис сильно зависит от языка C. Поскольку обе эти оболочки были изобретены в 1970-х годах, есть более современные версии, Bourne Again Shell (Bash) и tcsh (tee-see-shell). Bash — это оболочка по умолчанию для большинства систем, хотя tcsh почти всегда тоже доступна.

На основе Bash и tcsh были созданы другие оболочки, такие как оболочка Korn (ksh) и zsh. Выбор инструмента в основном личный. Но, если человек умеет комфортно работать с Bash, тогда он может эффективно работать на большинстве Linux-систем. После изучения Bash можно попробовать новые оболочки.

Еще большее разделение исторически сложилось среди текстовых редакторов. Текстовый редактор используется на консоли для редактирования файлов конфигурации и скриптов. Два основных лагеря «старой школы» — это vi (или более современные vim) и emacs. Оба являются очень мощными инструментами для редактирования текстовых файлов, они отличаются форматом команд и тем, как пишутся плагины для них. Плагины могут быть чем угодно: от синтаксического выделения программных проектов до встроенных календарей.

Оба vim и emacs сложны в изучении. Это плохо, если требуется простое редактирование небольшого текстового файла. Поэтому pico и nano доступны на большинстве систем (последний из них является производным от первого) и обеспечивают очень простое редактирование текста.

Даже если не использовать vi (vim), надо получить базовое представление о работе с ним, потому что vi находится в каждой системе Linux. Например, если потребуется восстанавливать неисправную систему Linux, запустив ее в режиме восстановления дистрибутива, наверняка будет доступ к vi.

Если есть система Linux, нужно будет добавлять, удалять и обновлять программное обеспечение. Раньше это означало загрузку исходного кода, настройку, компиляцию и копирование файлов в каждой системе. К счастью, разработчики дистрибутивов создали пакеты, которые являются сжатыми копиями приложения. Менеджер пакетов заботится о том, чтобы отслеживать, какие файлы принадлежат к какому пакету, и даже загружать обновления с удаленного сервера, называемого репозиторием. В системах Debian такие инструменты включают `dpkg`, `apt-get` и `apt`. В системах с Red Hat используются `rpm` и `yum`. Этот вопрос будет рассмотрен позднее.

1.3.4. Средства разработки

Неудивительно, что, поскольку программное обеспечение построено на вкладах программистов, Linux имеет отличную поддержку для разработки программного обеспечения. Оболочки построены так, чтобы быть программируемыми, и в каждой системе есть мощные редакторы. Существует также множество инструментов для разработки.

Компьютерные языки предоставляют возможность программисту вводить инструкции в более понятном для человека формате, и эти инструкции переводятся в инструкции, исполняемые компьютером. Языки попадают в один из двух лагерей: интерпретируются или компилируются. Интерпретируемый язык переводит написанный код в машинный код по мере запуска программы, а скомпилированный язык переводится сразу.

Сам Linux был написан на компилируемом языке C, так как его основное преимущество заключается в том, что сам язык тесно связан со сгенерированным машинным кодом, чтобы квалифицированный программист мог писать небольшой и эффективный код. Это было очень важно во времена, когда компьютерная память измерялась в Кбайтах. Даже с большими размерами памяти сегодня C все еще помогает писать код, который должен работать быстро, например, в операционной системе.

C был использован на протяжении многих лет. Существует C++, который добавляет поддержку объектов к C (другой стиль программирования), а Objective C принял другое направление и активно используется в продуктах Apple.

Язык Java отличается от подхода с компиляцией. Вместо того, чтобы компилировать машинный код, Java сначала представляет собой гипотетический процессор, называемый виртуальной машиной Java (JVM), и компилирует для него весь код. Затем каждый компьютер запускает программное обеспечение JVM для перевода инструкций JVM (называемых байт-кодом) в собственные инструкции.

Дополнительный перевод с Java может заставить думать, что это будет медленно. Однако JVM довольно прост, поэтому его можно быстро и надежно реализовать на любом устройстве от мощного компьютера до устройства с низким энергопотреблением, которое подключается к телевизору. Скомпилированный файл Java также можно запускать на любом компьютере, реализующем JVM!

Еще одно преимущество компиляции промежуточного кода заключается в том, что JVM может предоставлять услуги для приложения, которые обычно не будут доступны при исполнении на обычном процессоре. Выделение памяти для программы - сложная проблема, но она встроена в JVM. Это также означает, что разработчики JVM могут сосредоточить свои улучшения на JVM в целом, поэтому любой прогресс, который они делают, мгновенно доступен для приложений.

С другой стороны, интерпретируемые языки переводятся в машинный код по мере их выполнения. Дополнительная вычислительная мощность, затрачиваемая на это, часто может быть компенсирована повышением производительности, которую получает программист, не останавливаясь на компиляции. Интерпретируемые языки также предлагают больше возможностей, чем компилируемые языки, что означает, что часто требуется меньше кода. Сам интерпретатор языка обычно написан на другом языке, таком как C, а иногда даже Java! Это означает, что интерпретируемый язык запускается на JVM, который переводится во время выполнения в фактический машинный код.

Perl - интерпретируемый язык. Первоначально Perl был разработан для обработки текста. На протяжении многих лет он пользовался поддержкой системных администраторов и по-прежнему продолжает совершенствоваться и использоваться во всем: от автоматизации до создания веб-приложений.

PHP — это язык, который изначально был создан для создания динамических веб-страниц. Файл PHP читается веб-сервером, таким как Apache. Специальные теги в файле указывают, что части кода

должны интерпретироваться как инструкции. Веб-сервер объединяет все разные части файла и отправляет его в веб-браузер в итоговом HTML. Основными преимуществами РНР являются то, что их легко освоить и использовать практически в любой системе. Из-за этого многие популярные проекты построены на РНР. Известные примеры включают WordPress (блоги и сайты визитки) и даже части крупных проектов, таких как Facebook.

Ruby — это еще один язык, на который повлияли Perl и Shell, а также многие другие языки. Ruby позволяет решать сложные задачи программирования относительно легкими способами, и с включением структуры Ruby on Rails является популярным выбором для создания сложных веб-приложений. Ruby также является языком, на котором работают многие из ведущих инструментов автоматизации, таких как Chef и Puppet, которые упрощают управление большим количеством Linux-систем.

Python — это один из самых популярных интерпретируемых языков. Подобно Ruby, он упрощает задачи и имеет фреймворк под названием Django, который упрощает создание веб-приложений. Python обладает отличными способностями к статистической обработке и является фаворитом в академических кругах.

Язык — это просто инструмент, который позволяет указать компьютеру, какую задачу требуется выполнить. Библиотека объединяет общие задачи в отдельный пакет, который может быть использован разработчиком. Например, ImageMagick - одна из таких библиотек, которая позволяет программистам манипулировать изображениями в коде. ImageMagick также поставляется с некоторыми инструментами командной строки, которые позволяют обрабатывать изображения из оболочки и использовать возможности сценариев.

OpenSSL — это криптографическая библиотека, которая используется во всем, от веб-серверов до командной строки. Она предоставляет стандартный интерфейс для добавления криптографии в Perl-скрипт, например.

На гораздо более низком уровне находится библиотека C. Она обеспечивает базовый набор функций для чтения и записи файлов и дисплеев, которые используются приложениями и другими языками.

1.4. Программное обеспечения с открытым исходным кодом и лицензирование

Когда говорят о покупке программного обеспечения, есть три различных компонента.

1. Собственность. Кто владеет интеллектуальной собственностью программного обеспечения?
2. Денежный перевод. Как переходят деньги?
3. Лицензирование. Что получает покупатель? Что может сделать с программным обеспечением? Может ли использовать его только на одном компьютере? Может ли дать его кому-то еще?

В большинстве случаев право собственности на программное обеспечение остается за лицом или компанией, которая его создала. Пользователям предоставляется только лицензия на использование программного обеспечения. Это вопрос законодательства об авторском праве. Перевод денег зависит от бизнес-модели создателя. Это лицензирование, которое действительно отличает программное обеспечение с открытым исходным кодом от программного обеспечения с закрытым исходным кодом.

Два контрастных примера.

В Microsoft Windows корпорация Microsoft владеет интеллектуальной собственностью. Сама лицензия, лицензионное соглашение с конечным пользователем (EULA), является специальным юридическим документом, с которым должен согласиться пользователь/покупатель, указав согласие, чтобы установить программное обеспечение. Microsoft сохраняет исходный код и распространяет только двоичные копии через авторизованные каналы. Для большинства потребительских продуктов разрешено устанавливать программное обеспечение на одном компьютере и не разрешено делать копии диска, кроме резервной копии. Не разрешается перепроектировать программное обеспечение. Оплата производится за одну конкретную копию программного обеспечения, которая получает незначительные обновления, но не крупные обновления.

Linux принадлежит Линусу Торвальдсу. Он разместил код под лицензией GNU Public License версии 2 (GPLv2). Эта лицензия, среди прочего, говорит о том, что исходный код должен быть доступен всем, и что каждый может вносить любые изменения, которые захочет. Одно

из предостережений заключается в том, что, если вносятся изменения и они распространяются, то требуется поместить изменения под одну и ту же лицензию, чтобы другие могли воспользоваться новым кодом. GPLv2 также говорит, что не разрешено взимать плату за распространение исходного кода, кроме фактических затрат на это (например, копирование на съемные носители).

В общем, когда компания или частное лицо что-то создают, они также получают право решать, как его использовать и распространять. Бесплатное программное обеспечение с открытым исходным кодом (FOSS) относится к программному обеспечению, в котором этого права нет, и всем разрешено просматривать исходный код и распространять его. Линус Торвалдс сделал это с Linux - хотя он и создал Linux, он не может запретить использовать его на любом компьютере, потому что он отказался от этого права через лицензию GPLv2.

Лицензирование программного обеспечения является политической/юридической проблемой, и неудивительно, что существует много разных мнений в этом вопросе. Крупные организации разрабатывают свою собственную лицензию, которая воплощает их конкретные взгляды, но для многих легче выбрать существующую лицензию, чем придумать свою собственную. Например, университеты, такие как Массачусетский технологический институт и Калифорнийский университет, разработали свои лицензии. Такие проекты, как Apache Foundation, и такие группы, как Фонд свободного программного обеспечения, создали свои собственные лицензии для продвижения своей повестки.

1.4.1. Фонд свободного программного обеспечения и открытый исходный код

Две группы могут считаться самыми влиятельными силами в мире с открытым исходным кодом: Фонд свободного программного обеспечения (FSF) и Инициатива с открытым исходным кодом (OSI).

Фонд свободного программного обеспечения был основан в 1985 году Ричардом Столлманом. Цель FSF - продвигать бесплатное программное обеспечение. По мнению FSF, проприетарное программное обеспечение (программное обеспечение, распространяемое по лицензии с закрытым исходным кодом) является плохим. FSF также высту-

пает за то, что лицензии на программное обеспечение должны обеспечивать открытость изменений. По их мнению, тот, кто изменит Free Software, должен будет поделиться своими изменениями. Эта конкретная философия называется copyleft.

FSF также выступает против патентов на программное обеспечение и выступает в качестве сторожевого пса для организаций по стандартизации, выступая, когда предлагаемый стандарт может нарушать принципы свободного программного обеспечения, включая такие элементы, как Digital Rights Management (DRM), которые могут ограничить то, что могли бы сделать с сервисом.

FSF разработали собственный набор лицензий, таких как GPLv2 и GPLv3, а также лицензии меньшего размера GPL версии 2 и 3 (LGPLv2 и LGPLv3). Меньшие лицензии во многом похожи на обычные лицензии, за исключением того, что они содержат положения о привязке к несвободному программному обеспечению. Например, в GPLv2 нельзя распространять программное обеспечение, которое использует закрытую исходную библиотеку (например, драйвер устройства), но это позволяет вариант меньшего размера.

Изменения между версиями 2 и 3 в основном сосредоточены на использовании Free Software на закрытом аппаратном устройстве. Например, TiVo — это компания, которая строит телевизионный цифровой видеомаягнитофон на своем собственном оборудовании и использует Linux в качестве основы для своего программного обеспечения. Хотя TiVo выпустила исходный код для своей версии Linux, как требуется в GPLv2, аппаратное обеспечение не будет запускать модифицированные бинарные файлы. В глазах FSF это противоречило духу GPLv2, поэтому они добавили конкретное предложение к версии 3 лицензии. Линус Торвальдс согласен с TiVo по этому вопросу и решил остаться с GPLv2.

Инициатива с открытым исходным кодом была основана в 1998 году Брюсом Перенсом и Эриком Раймондом (ESR). Они считают, что свободное программное обеспечение было слишком политически заряженным и что требовались менее экстремальные лицензии, особенно в отношении копилефт-аспектов лицензий FSF. OSI полагает, что источник должен быть доступен не только свободно, но также и не следует устанавливать никаких ограничений на использование программного

обеспечения независимо от того, какое у него предназначение. В отличие от FSF, OSI не имеет собственного набора лицензий. Вместо этого OSI имеет ряд принципов и добавляет другие лицензии в этот список, если они соответствуют этим принципам, называемым лицензиями с открытым исходным кодом. Таким образом, программное обеспечение, соответствующее лицензии с открытым исходным кодом, является открытым исходным кодом.

Некоторые из лицензий с открытым исходным кодом — это семейство лицензий BSD, которые намного проще, чем GPL. Они просто заявляют, что можно распространять исходный код и двоичные файлы до тех пор, пока сохраняются уведомления об авторских правах и не подразумевается, что первоначальный создатель поддерживает новую версию. Другими словами, «делайте то, что хотите с помощью этого программного обеспечения, просто не говорите, что его написали».

Лицензии FSF, такие как GPLv2, также являются лицензиями с открытым исходным кодом. Однако многие лицензии с открытым исходным кодом, такие как BSD и MIT, не содержат положений о копи-лефт и, таким образом, неприемлемы для FSF. Эти лицензии называются разрешительными лицензиями на бесплатное программное обеспечение, поскольку они позволяют разрешать распространение программного обеспечения. Можно взять лицензионное программное обеспечение BSD и включить его в закрытый программный продукт.

1.4.2. Дополнительные условия

Вместо того, чтобы останавливаться на более тонких моментах Open Source vs. Free Software, сообщество начало ссылаться на все это как бесплатное и открытое исходное программное обеспечение (FOSS). Английское слово «Free» может означать «бесплатно, без каких-либо затрат», так и «бесплатно, без ограничений». Эта двусмысленность привела к включению слова libre в ссылку на последнее определение. Таким образом, получаем Free / Libre / Open Source Software (FLOSS).








Хотя эти условия удобны, они скрывают различия между двумя школами мысли. По крайней мере, когда используется программное обеспечение FOSS, нужно помнить, что не нужно платить за него, и можно распространять его по своему усмотрению.

1.4.3. Другие схемы лицензирования

Лицензии FOSS в основном связаны с программным обеспечением. Люди размещали такие работы, как рисунки и чертежи по лицензии FOSS, но это не было целью.

Когда программное обеспечение размещено в общественном достоянии, автор отказался от всех прав, включая авторское право на произведение. В некоторых странах это используется по умолчанию, когда работа выполняется государственным органом. В некоторых странах работа, защищенная авторским правом, становится общественным достоянием после смерти автора или длительного периода ожидания.

Организация Creative Commons (CC) создала лицензии Creative Commons, которые пытаются перенести принципы лицензий FOSS для непрограммных объектов. Лицензии CC также могут использоваться для ограничения коммерческого использования, если это желание владельца авторских прав. Лицензии CC:

- Атрибут (CC BY ) - очень похоже на лицензию BSD, можно использовать CC BY-контент для любых целей, но требуется указывать владельца авторских прав;
- Атрибут ShareAlike (CC BY-SA ) - копия версия лицензии Attribution. Требуется указание авторства и производные работы распространяются по той же лицензии, что и оригинал;
- Атрибут No-Derivs (CC BY-ND ) - можно распространять контент на тех же условиях, что и CC-BY, но нельзя его изменять
- Атрибут NonCommercial (CC BY-NC ) - так же, как CC BY, но нельзя использовать в коммерческих целях
- Атрибут NonCommercial-ShareAlike (CC-BY-NC-SA ) - Создает лицензию CC BY-NC, но требует, чтобы изменения были по той же лицензии.
- Атрибут NonCommercial-No-Derivs (CC-BY-NC-ND ) - Можно делиться содержимым, которое будет использоваться в некоммерческих целях, но нельзя изменять контент.
- No Rights Reserved (CC0 ) - это общедоступная версия Creative Commons, не накладывающая ограничений.

Вышеупомянутые лицензии можно суммировать как ShareAlike или без ограничений, и разрешено ли коммерческое использование или изменения.

1.4.4. Бизнес-модели с открытым исходным кодом

Если бесплатно отдавать свое программное обеспечение, как можно зарабатывать на этом?

Самый простой способ заработать деньги – продавать поддержку или гарантию вокруг программного обеспечения. Можно зарабатывать деньги, устанавливая людям программное обеспечение, помогая людям, когда у них есть проблемы, или исправлять ошибки за деньги, фактически являясь консультантом.

Также можно взимать плату за услугу или подписку, которая расширяет программное обеспечение.

Можно совместить продажи с оборудованием или добавить дополнительное закрытое исходное программное обеспечение для продажи вместе с бесплатным программным обеспечением. Приборы и встроенные системы, часто используют Linux. Многие потребительские Wi-Fi маршрутизаторы и развлекательные устройства следуют этой модели.

Также можно разрабатывать программное обеспечение с открытым исходным кодом как часть своей работы. Если создать инструмент, применимый на обычной работе, можно убедить своего работодателя позволить открыть его исходный код. Это может быть ситуация, когда работали над программным обеспечением в одной компании, но лицензирование в качестве открытого исходного кода позволяло бы другим людям с подобной проблемой помогать и даже вносить свой вклад в разработку.

В 1990-х годах Джеральд Комбс работал в интернет-провайдере и начал писать собственный инструмент анализа сети, потому что подобные инструменты в то время были очень дорогими. В настоящее время в проекте, получившем название Wireshark, участвует более 600 человек. Сейчас его часто считают лучше коммерческих аналогов. Это привело к созданию компании вокруг Джеральда для поддержки Wireshark и продажи продуктов и поддержки, которые делают его более полезным. Позднее эта компания была куплена крупным сетевым поставщиком, который поддерживает его разработку.

Другие компании получают такую огромную прибыль из программного обеспечения с открытым исходным кодом, что нанимают программистов для работы над программным обеспечением в полном объеме. Поисковая система Google наняла создателя языка Python. Даже Линус Торвалдс нанят Linux Foundation для работы в Linux. Американская телефонная компания AT&T получает такую выгоду от проектов Ruby и Rails для своего ресурса Yellow Pages, что у них есть сотрудник, который ничего не делает, кроме работы над этими проектами.

Ещё один способ, которым люди косвенно зарабатывают деньги через открытый исходный код, заключается в том, что это открытый способ оценить свои навыки. Одно дело сказать на собеседовании, что выполняли определенные задания на своей предыдущей работе, а другое продемонстрировать свое творение и поделиться им со всем миром. Потенциальные работодатели видят качество работы автора. Аналогичным образом, компании обнаружили, что открытые источники не критических частей их внутреннего программного обеспечения привлекают интерес людей.

1.5. Графический и неграфический режимы

Прежде чем стать эффективным администратором систем Linux, нужно научиться использовать Linux с рабочим столом и получить базовые навыки в области информационно-коммуникационных технологий (ИКТ). Это не только поможет при работе с пользователями, но и поможет быстрее улучшить свои навыки. Кроме того, жизнь системного администратора — это больше, чем просто работа с сервером — есть электронная почта и работа с документацией.

Linux можно использовать одним из двух способов: графически и неграфически. В графическом режиме приложения открываются в окнах, которые можно изменять и перемещать. Есть меню и инструменты, которые помогут найти то, что требуется. Здесь будет использоваться веб-браузер, инструменты для редактирования графики и электронной почты. На рисунке 1 отображен пример графического рабочего стола Ubuntu с панелью меню популярных приложений слева, документом, открытым в LibreOffice, и страницей, открытой с помощью веб-браузера Firefox.

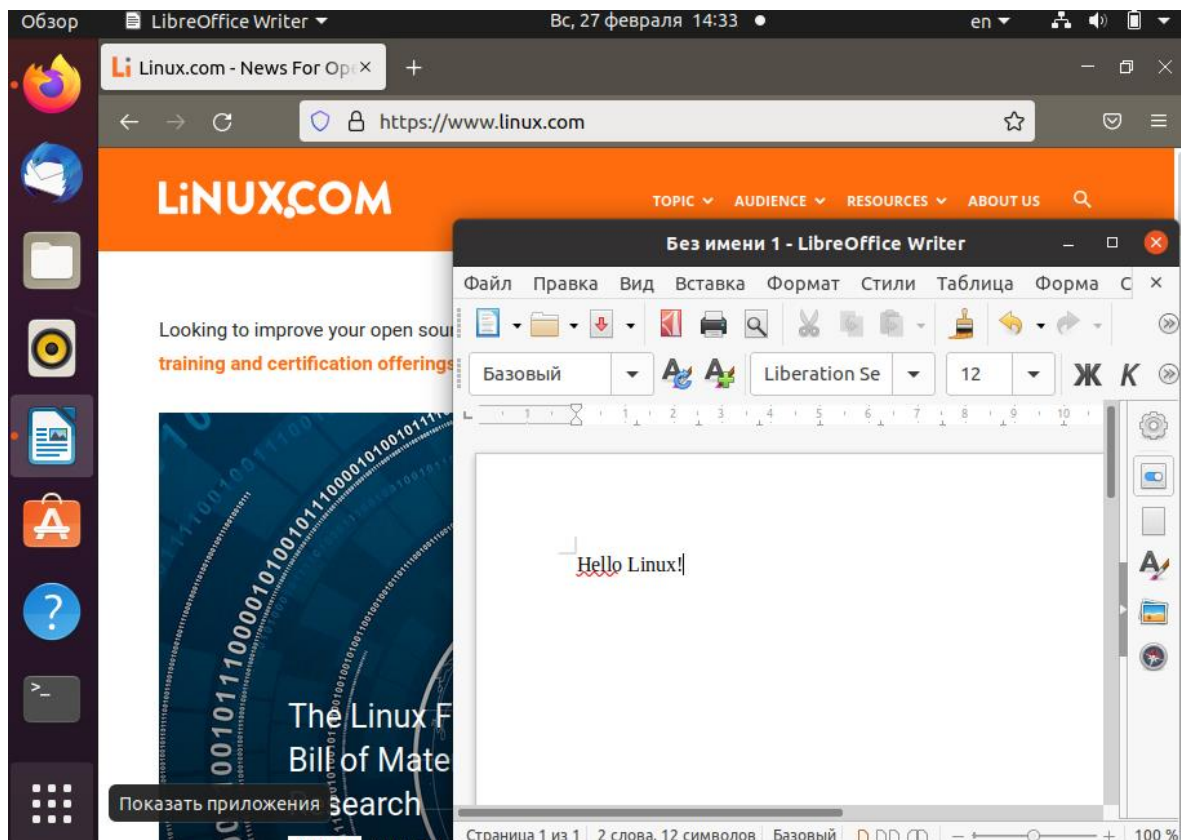


Рис. 1. Графический режим

В графическом режиме можно открыть несколько оболочек, что очень полезно при выполнении задач на нескольких удаленных компьютерах. Для локального входа требуется использовать имя пользователя и пароль. После входа в систему пользователь попадает на рабочий стол, где может запускать приложения.

Неграфический режим начинается с текстового входа, показанного ниже. Будет предложено ввести имя пользователя и после этого ввести пароль. Если вход успешно выполнен, пользователь попадает прямо в оболочку. В некоторых случаях пароль пользователя заменяется на файл-ключ. Например, при подключении по протоколу SSH в качестве идентификатора может быть применен зашифрованный ключ.

```
ubuntu 20.04 ubuntu tty2
ubuntu login:
```

В неграфическом режиме нет окон для перемещения. Несмотря на то, что есть текстовые редакторы, веб-браузеры и почтовые клиенты, это только текст. Именно так UNIX начал свою работу до того, как были установлены нормальные графические среды. Большинство

серверов также будут работать в этом режиме, так как люди не контактируют с ними напрямую, что делает графический интерфейс пустой тратой ресурсов. Ниже приведен пример экрана, который можно увидеть после входа в систему.

```
ubuntu 20.04 ubuntu tty2

ubuntu login: username
Password:

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.

Welcome to Ubuntu 20.04 (GNU/Linux 3.8.0-19-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

212 packages can be updated.
91 updates are security updates.

username@ubuntu:~$ w
 17:27:22 up 14 min,  2 users,  load average: 1.73, 1.83, 1.69
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU WHAT
username  tty2                    20:08    14.35  0.05s  0.00s w
```

Можно увидеть исходное приглашение для входа в верхней части с новым текстом, добавленным ниже. Во время входа в систему можно увидеть некоторые сообщения, называемые сообщением дня (MOTD), что дает возможность системному администратору передавать информацию пользователям. После MOTD следует командная строка. В приведенном выше примере пользователь ввел команду `w`, которая показывает, кто вошел в систему. Когда новые команды вводятся и обрабатываются, окно прокручивается, а старый текст теряется сверху. Сам терминал несет ответственность за сохранение любой истории, например, чтобы позволить пользователю прокручивать вверх и видеть ранее введенные команды.

1.6. Командная строка

Командная строка представляет собой простой текстовый ввод, который позволяет исполнять текстовые команды, начиная от одного командного слова и заканчивая сложными сценариями. Если входить в текстовый режим, то сразу попадаем в консоль. Если входить в систему графически, нужно будет запустить графическую оболочку, которая

представляет собой текстовую консоль с окном вокруг нее, для которого можно изменять размер и которую можно перемещать.

Рабочие столы Linux могут быть разными, поэтому, для выхода в текстовый режим надо найти в меню одну из программ: terminal, console, x-term и т. п. Эти программы - графические оболочки, отличающиеся в основном внешностью, а не функциональностью. Если есть инструмент поиска, такой как Ubuntu One, в нем можно найти подходящую программу, как показано на рисунке 2.

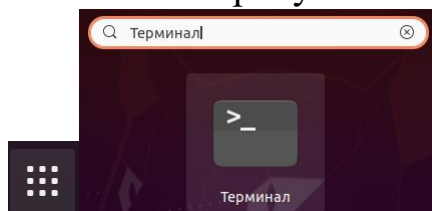


Рис. 2. Запуск программы терминала

Подобные инструменты устроены похожим образом во всех современных операционных системах и позволяют быстро находить в системе программы без пролистывания через меню.

1.7. Виртуализация и облачные вычисления

Linux - многопользовательская операционная система, что означает, что разные пользователи могут работать в одной и той же системе одновременно и по большей части не могут навредить другим пользователям. Однако у этого правила есть ограничения - пользователи могут загружать дисковое пространство или занимать слишком много ресурсов памяти и процессора, при этом делать систему медленной для всех. Совместное использование системы в многопользовательском режиме также требует, чтобы все работали как непривилегированные пользователи, поэтому позволить каждому пользователю запустить собственный веб-сервер очень сложно.

Виртуализация — это процесс, когда один физический компьютер, называемый хостом, запускает несколько копий операционной системы, каждый из которых называется гостем. Хост запускает программное обеспечение, называемое гипервизором, которое переключает управление между различными гостями, как это делает ядро Linux для отдельных процессов.

Виртуализация работает, потому что серверы проводят большую часть своего времени на холостом ходу и не нуждаются в таких физических ресурсах, как монитор и клавиатура. Теперь можно использовать мощный процессор и распространять его возможности на нескольких виртуальных машинах, таким образом поддерживая более справедливый обмен между гостями. Основным ограничением является, как правило, память, а с технологиями гипервизора и многоядерными процессорами можно поставить больше виртуальных машин на один хост, чем когда-либо.

В виртуализованной среде один хост может запускать десятки гостевых операционных систем и при поддержке самого процессора гости даже не знают, что они работают на виртуальной машине. Каждый гость получает свой собственный виртуальный процессор, оперативную память, диск и сам общается с сетью. Даже не нужно запускать одну и ту же операционную систему для всех гостей, что еще больше уменьшает количество необходимых физических серверов.

Виртуализация предлагает способ снизить потребление энергии и сократить пространство центров обработки данных над эквивалентным парком физических серверов. Теперь используются просто программные конфигурации, поэтому легко развернуть новую машину для тестирования и уничтожить ее, когда ее полезность прошла.

Виртуальную машину можно использовать для знакомства с новыми операционными системами и для целей тестирования программного обеспечения.

Если нужно запускать несколько экземпляров операционной системы на одном физическом компьютере и подключаться к ней по сети, то расположение машины на самом деле не имеет значения. Облачные вычисления используют этот подход и позволяют иметь виртуальную машину в удаленном центре данных и оплачивать только те ресурсы, которые используются. Поставщики облачных вычислений могут воспользоваться масштабами экономии, чтобы предлагать вычислительные ресурсы по ценам лучшим, чем выходило бы при покупке собственного оборудования, оплате его размещения, охлаждения, подключения к сети и обслуживания.

Виртуальные серверы — это только один из аспектов облачных вычислений. Через облако можно получить файловое хранилище, базы данных или даже программное обеспечение. Ключевая особенность в

большинстве этих продуктов заключается в том, что потребитель платит за то, что он использует, например, определенную сумму за гигабайт данных в месяц, а не покупает аппаратное и программное обеспечение, а затем размещает его самостоятельно. Некоторые ситуации более подходят для облака, нежели другие. Следует учитывать проблемы безопасности и производительности, а также стоимость и функциональность.

Linux играет ключевую роль в облачных вычислениях. Большинство виртуальных серверов основаны на ядре Linux, а Linux часто используется для размещения приложений, работающих с облачными вычислениями.

1.8. Использование Linux для работы

Основными инструментами, используемыми в большинстве офисов, являются:

- Текстовый редактор.
- Таблицы.
- Презентации.
- Веб-браузер.

OpenOffice, или более активный его наследник LibreOffice, решают первые три вопроса, являясь для многих заменой Microsoft Office. Текстовый процессор Writer используется для редактирования текстовых документов, таких как отчеты и заметки. Таблицы Calc полезны для работы с числами, например, для обобщения данных о продажах и будущих прогнозов. Пакет презентаций Impress используется для создания слайдов с такими функциями, как текст, графика и анимация. Слайды могут быть напечатаны или отображены на экране и проекторе, чтобы поделиться с аудиторией.

LibreOffice помимо собственных форматов файлов может работать с проприетарными файловыми форматами, такими как файлы Microsoft Office или Adobe Portable Document Format (PDF). Кроме того, благодаря использованию расширений LibreOffice можно интегрировать с программным обеспечением Wiki, чтобы предоставить мощное решение для интрасети.

Linux является приоритетным для браузеров Firefox и Google Chrome. Таким образом, можно рассчитывать на доступность новейшего программного обеспечения для платформы и своевременный доступ к исправлениям ошибок и новым функциям.

1.9. Безопасность Linux-компьютера

Пользователи должны придерживаться некоторых базовых принципов для обеспечения безопасности своих данных.

Самое первое, что нужно сделать, это использовать хороший уникальный пароль, особенно на локальном компьютере. Хороший пароль имеет длину не менее 10 символов и содержит смесь цифр, букв (верхний и нижний регистр) и специальных символов. Для генерации и сохранения паролей можно использовать пакет KeePass. Нужно только иметь пароль для входа на компьютер и пароль, чтобы открыть файл KeePass. Современные браузеры содержат аналогичные средства генерации и сохранения паролей для доступа к интернет-сайтам. В организациях как правило придерживаются более строгих требований к парольной системе и принудительно требуют смены пароля каждого сотрудника через заданный период времени.

Обязательно нужно периодически проверять обновления операционной системы и программного обеспечения. При наличии рабочего стола в операционных системах есть наглядные средства для выполнения этих функций. Выходят подсказки пользователю о доступных обновлениях и предлагается выполнить обновление. Если есть обновления, связанные с безопасностью, будет немедленно предложено их установить. В противном случае можно получать обновления раз в неделю или реже. Все, что пользователю нужно сделать, это нажать кнопку «Установить», ввести пароль и обновления будут выполнены в автоматическом режиме.

Ещё один элемент защиты – брандмауэр. Это программное обеспечение, которое фильтрует сетевой трафик. Linux имеет встроенные средства фильтрации сетевого трафика iptables и множество программ для его настройки. Для большинства пользователей достаточно просто изменить статус на «включено», что заблокирует весь нежелательный трафик, поступающий на компьютер, если только его не инициировал сам пользователь. Изменение правил брандмауэра требует четкого по-

нимания совершаемых действий. На рабочих компьютерах организаций изменение настроек сетевого трафика осуществляются администратором и не разрешается простому пользователю.

1.10. Защита данных в сети Интернет

Когда пользователь просматривает страницы в Интернет, он оставляет цифровой след. Большая часть этой информации игнорируется, некоторые из них собираются для сбора статистики для рекламы, а некоторые могут использоваться для вредоносных целей. Как правило, пользователь не должен доверять сайтам, с которыми взаимодействует. Нужно использовать отдельные пароли на каждом веб-сайте, чтобы, если сайт взломан, пароль не мог использоваться для доступа к другим сайтам. Использование KeePass, упомянутого ранее, является самым простым способом сделать это. Кроме того, желательно ограничивать информацию, которую пользователь предоставлял сайтам. Хотя предоставление девичьей фамилии и даты рождения матери может помочь разблокировать учетную запись в социальной сети, вместо этого лучше использовать средства двухфакторной аутентификации. В настоящее время большинство крупных сервисов используют аутентификацию с помощью номера мобильного телефона. Такой вариант лучше, чем использование постоянного логина и пароля и однозначно идентифицирует пользователя.

Cookies — это основной механизм, который используют веб-сайты для отслеживания посетителей. Иногда это отслеживание помогает, например, чтобы знать, что находится в корзине покупок интернет-магазина, или чтобы осуществлять автоматический вход в систему при возвращении на сайт. Когда пользователь просматривает веб-страницы, веб-сервер может отправить ему файл cookie, который представляет собой небольшой фрагмент текста, вместе с веб-страницей. Браузер хранит этот файл и отправляет его с каждым запросом на тот же сайт.

Тем не менее, на многих сайтах есть встроенные скрипты, которые поступают от третьих лиц, такие как рекламные баннеры или невидимые пиксели аналитики. Если на разных сайтах есть одинаковые пиксели отслеживания, от одного рекламодателя, то один и тот же файл cookie будет отправлен при просмотре обоих сайтов. Рекламодатель во всех таких случаях узнает, что посетил пользователь. При достаточно

широком охвате, например, в социальных сетях и т. д., веб-сайт может получить представление о том, какие веб-сайты пользователь часто посещает, и выяснить его интересы и демографические данные.

Существуют различные стратегии борьбы с этим. Один из них - игнорировать все файлы cookie. Другой - ограничить пиксели отслеживания, которые принимает браузер, либо путем их полного блокирования, либо периодической чисткой.

Настройки, связанные с cookie для разных браузеров, могут отличаться. Но как правило надо зайти в раздел «Приватность и защита», а дальше выбрать одну из схем: Стандартную, Строгую или Персональную. Нужно только помнить, что некоторые сайты в более строгом режиме могут отображаться некорректно. Настройка конфиденциальности может сделать пользователя более анонимным в Интернете, но это также может вызвать проблемы с некоторыми сайтами, зависящими от сторонних файлов cookie. Если это произойдет, возможно, придется явно разрешать сохранение некоторых файлов cookie.

Контрольные вопросы

1. Что означает понятие «Открытый исходный код»?
2. Под какой лицензией распространяется ядро Linux?
3. Linux – это Unix?
4. Что общего между Apple macOS и Linux?
5. Какие есть преимущества использования неграфического консольного режима над графическим?
6. В каких сферах и на каком аппаратном обеспечении используются операционные системы с ядром Linux?
7. Что означает акроним GNU?

Глава 2. ОСНОВНЫЕ КОМАНДЫ

2.1. Интерфейс командной строки (CLI)

Введенный в массы компанией Apple на компьютере Macintosh и популяризированный Microsoft, графический интерфейс пользователя (graphical user interface, GUI) обеспечивает простой и удобный способ управления системой. Без графического интерфейса некоторые инструменты для графики и видео не были бы реализуемы.

До популярности GUI интерфейс командной строки (CLI) был предпочтительным или даже единственным способом управления компьютером. CLI полагается исключительно на ввод с клавиатуры. Все, что пользователь хочет, чтобы компьютер выполнял, ретранслируется путем ввода команд, а не щелчка мышью по значкам.

Для тех, кто никогда не использовал CLI, сначала это может оказаться сложным, потому что требует запоминания команд и их вариантов. Однако CLI обеспечивает более точное управление, большую скорость и возможность легко автоматизировать задачи с помощью сценариев. Хотя в Linux есть много графических сред, можно более эффективно управлять Linux, используя интерфейс командной строки.

Интерфейс командной строки (CLI) является текстовым интерфейсом компьютера, в котором пользователь вводит команды, а компьютер их выполняет. Среда CLI предоставляется приложением на компьютере, известном как терминал.

Терминал принимает то, что пользователь вводит и передает в оболочку (shell). Оболочка интерпретирует то, что пользователь ввел, преобразуя в инструкции, которые могут быть выполнены операционной системой. Если командой производится вывод данных, то этот текст отображается в терминале. Если возникают проблемы с командой, отображается сообщение об ошибке.

2.2. Доступ к терминалу

Существует много способов доступа к окну терминала. Некоторые системы будут загружаться непосредственно в терминальном ре-

жиме. Это часто происходит с серверами, поскольку графический интерфейс пользователя (GUI) может быть ресурсоемким и может не понадобиться для выполнения операций на сервере.

Хорошим примером сервера, который не обязательно требует графического интерфейса, является веб-сервер. Веб-серверы должны запускаться как можно быстрее, а графический интерфейс просто замедляет работу системы.

В системах, загружаемых с графическим интерфейсом, обычно есть два способа доступа к терминалу: 1) терминал на основе графического интерфейса; 2) виртуальный терминал.

Терминал GUI представляет собой программу в среде GUI, которая эмулирует окно терминала. К терминалам GUI можно получить доступ через систему меню. Например, на машине CentOS можно нажать «Приложения» в строке меню, затем «Системные инструменты» и, наконец, «Терминал»:

Виртуальный терминал можно запускать одновременно с графическим интерфейсом, но он требует от пользователя входа в систему через виртуальный терминал, прежде чем они смогут выполнять команды. В большинстве систем есть несколько виртуальных терминалов, к которым можно получить доступ, нажав комбинацию клавиш, например: Ctrl-Alt-F1

Примечание. На виртуальных машинах виртуальные терминалы могут быть недоступны.

2.2.1. Запрос

В окне терминала отображается запрос; приглашение появляется, когда никакие команды не выполняются, и когда весь вывод команды был напечатан на экране. Приглашение предназначено для указания пользователю ввести команду.

Структура подсказки может различаться между дистрибутивами, но обычно будет содержать информацию о пользователе и системе. Ниже приведена общая структура приглашения:

```
username@hostname:~$
```

В приглашении указывается имя пользователя, входящего в систему (`username`), имя системы (`hostname`) и текущий каталог (`~`). Символ `~` используется как сокращение для домашнего каталога пользователя (обычно домашний каталог для пользователя находится в каталоге

/home и называется именем учетной записи пользователя, например: /home/username).

2.2.2. Shell

Оболочка — это интерпретатор, который преобразует команды, введенные пользователем, в действия, выполняемые операционной системой. В среде Linux предусмотрены различные типы оболочек, некоторые из которых существуют уже много лет.

Наиболее часто используемая оболочка для дистрибутивов Linux называется оболочкой BASH. Это оболочка, которая предоставляет множество дополнительных функций, таких как история команд, что позволяет легко выполнять ранее выполненные команды.

У оболочки BASH также есть другие популярные функции:

- Сценарии: возможность размещения команд в файле и выполнения файла, в результате чего выполняются все команды. Эта функция также имеет некоторые функции программирования, такие как условные операторы и возможность создавать функции (подпрограммы).
- Псевдонимы: способность создавать короткие «прозвища» для более длинных команд.
- Переменные: переменные используются для хранения информации во время сеанса пользователя в оболочке BASH. Эти переменные могут использоваться для изменения работы команд и функций, а также для обеспечения важной системной информации.

Примечание. Предыдущий список — это просто краткое изложение некоторых из многих функций, предоставляемых оболочкой BASH.

2.2.3. Команды форматирования

Многие команды могут использоваться сами по себе без дальнейшего ввода. Некоторые команды требуют дополнительного ввода для правильной работы. Этот дополнительный ввод представлен в двух формах: параметрах (опциях, options) и аргументах.

Типичный формат команды:

команда [параметры] [аргументы]

Параметры используются для изменения основного поведения команды, а аргументы используются для предоставления дополнительной информации (например, имени файла или имени пользователя).

Каждый параметр и аргумент обычно разделяются пробелом, хотя параметры часто могут объединяться вместе.

Linux чувствителен к регистру. Команды, параметры, аргументы, переменные и имена файлов должны вводиться точно так, как показано.

Покажем пример на команде `ls`. Сама по себе команда `ls` будет перечислять файлы и каталоги, содержащиеся в текущем рабочем каталоге:

```
sysadmin@localhost:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
sysadmin@localhost:~$
```

Команда `ls` будет подробно рассмотрена в следующей главе. Цель введения этой команды состоит в том, чтобы продемонстрировать, как работают аргументы и параметры. На этом этапе не следует беспокоиться о том, что является результатом команды, а скорее сосредоточиться на понимании того, что такое аргументы и параметры.

Аргумент может быть передан команде `ls`, чтобы указать, в какой директории будет отображаться содержимое. Например, команда `ls /etc/ppp` будет отображать содержимое каталога `/etc/ppp` вместо текущего каталога:

```
sysadmin@localhost:~$ ls /etc/ppp
ip-down.d ip-up.d
sysadmin@localhost:~$
```

Поскольку команда `ls` принимает несколько аргументов, можно сразу перечислить содержимое нескольких каталогов, набрав команду `ls /etc/ppp /etc/ssh`

```
sysadmin@localhost:~$ ls /etc/ppp /etc/ssh
/etc/ppp:
ip-down.d ip-up.d
/etc/ssh:
moduli          ssh_host_dsa_key.pub  ssh_host_rsa_key      sshd_configssh_config
ssh_host_ecdsa_key  ssh_host_rsa_key.pub
ssh_host_dsa_key    ssh_host_ecdsa_key.pub  ssh_import_id
sysadmin@localhost:~$
```

2.2.4. Параметры команд

Параметры могут использоваться с командами для расширения или изменения поведения команды. Параметры часто являются одиночными буквами; однако иногда они также будут «словами». Как правило, более старые команды используют одиночные буквы, в то время

как новые команды используют полные слова. Однобуквенным параметрам предшествует одна тире. Параметрам полного слова предшествуют два символа тире.

Например, можно использовать параметр `-l` с командой `ls` для отображения дополнительной информации о файлах. Команда `ls -l` будет отображать файлы, содержащиеся в текущем каталоге, и предоставлять дополнительную информацию, такую как разрешения, размер файла и другую информацию:

```
sysadmin@localhost:~$ ls -l
total 0
drwxr-xr-x 1 sysadmin sysadmin 0 Jan 29  2022 Desktop
drwxr-xr-x 1 sysadmin sysadmin 0 Jan 29  2022 Documents
drwxr-xr-x 1 sysadmin sysadmin 0 Jan 29  2022 Downloads
drwxr-xr-x 1 sysadmin sysadmin 0 Jan 29  2022 Music
drwxr-xr-x 1 sysadmin sysadmin 0 Jan 29  2022 Pictures
drwxr-xr-x 1 sysadmin sysadmin 0 Jan 29  2022 Public
drwxr-xr-x 1 sysadmin sysadmin 0 Jan 29  2022 Templates
drwxr-xr-x 1 sysadmin sysadmin 0 Jan 29  2022 Videos
sysadmin@localhost:~$
```

В большинстве случаев параметры могут использоваться вместе с другими параметрами. Например, команда `ls -l -h` или `ls -lh` будет перечислять файлы с подробной информацией, но отображает размеры файлов в формате, удобном для чтения, вместо значения по умолчанию (байты):

```
sysadmin@localhost:~$ ls -l /usr/bin/perl
-rwxr-xr-x 2 root root 10376 Feb  4  2021 /usr/bin/perl
sysadmin@localhost:~$ ls -lh /usr/bin/perl
-rwxr-xr-x 2 root root 11K Feb  4  2021 /usr/bin/perl
sysadmin@localhost:~$
```

В предыдущем примере показано, как можно комбинировать однобуквенные варианты: `-lh`. Порядок комбинированных параметров не важен.

Параметр `-h` также имеет форму полного слова: `--human-readable`.

Параметры часто можно использовать с аргументом. Фактически, некоторые параметры требуют собственных аргументов. Можно использовать параметр и аргумент с командой `ls`, чтобы отобразить содержимое другого каталога, выполнив команду `ls -l /etc/ppp`

```
sysadmin@localhost:~$ ls -l /etc/ppp
total 0
drwxr-xr-x 1 root root 10 Jan 29  2022 ip-down.d
drwxr-xr-x 1 root root 10 Jan 29  2022 ip-up.d
sysadmin@localhost:~$
```


2.3. История команд

Когда команды выполняются в терминале, они сохраняются в «истории». Это предназначено для упрощения выполнения одной и той же команды позже, поскольку не нужно будет повторно указывать всю команду.

Чтобы просмотреть список истории терминала, используется команда `history`:

```
sysadmin@localhost:~$ date
Сб 20 апр 2022 08:57:51 MSK
sysadmin@localhost:~$ ls
Desktop Documents Downloads Music Pictures Public Templates
Videos
sysadmin@localhost:~$ cal 5 2023
    Май 2023
Вс Пн Вт Ср Чт Пт Сб
   1  2  3  4  5  6
   7  8  9 10 11 12 13
  14 15 16 17 18 19 20
  21 22 23 24 25 26 27
  28 29 30 31
sysadmin@localhost:~$ history
  1 date
  2 ls
  3 cal 5 2023
  4 history
sysadmin@localhost:~$
```

Нажатие клавиши «Вверх» ↑ приведет к отображению предыдущей команды в строке подсказки. Можно нажимать клавишу «Вверх» повторно, чтобы перебирать ранее выполненные команды. Нажатие клавиши `Enter` снова запустит отображаемую команду. После нахождения интересующей команды можно её отредактировать, используя клавиши со стрелками влево и вправо, чтобы переместить курсор для редактирования. Другими полезными ключами для редактирования являются клавиши `Home`, `End`, `Backspace` и `Delete`.

Для запуска команды из списка, который выводится командой `history`, можно выполнить эту команду, введя восклицательный знак, а затем номер рядом с командой, например: `!3`

```
sysadmin@localhost:~$ history
  1 date
  2 ls
  3 cal 5 2023
  4 history
sysadmin@localhost:~$ !3
cal 5 2023
    Май 2023
Вс Пн Вт Ср Чт Пт Сб
```

```

1  2  3  4  5  6
7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
sysadmin@localhost:~$

```

В таблице 1 можно увидеть дополнительные параметры команды `history`.

Таблица 1. Некоторые дополнительные примеры команды `history`

Пример	Значение
<code>history 5</code>	Показать последние пять команд из списка истории
<code>!!</code>	Выполнить последнюю команду еще раз
<code>! -5</code>	Выполнить пятую команду из нижней части списка истории
<code>! ls</code>	Выполнить самую последнюю команду <code>ls</code>

2.4. Переменные оболочки BASH

Переменная оболочки `BASH` позволяет оболочке хранить данные. Эти данные могут использоваться для предоставления критической информации о системе или для изменения поведения работы оболочки `BASH` (или других команд).

Переменным присваиваются имена, и они временно хранятся в памяти. Когда закрывается окно терминала или оболочка, все переменные теряются. Тем не менее, система автоматически воссоздает многие из этих переменных при открытии новой оболочки.

Чтобы отобразить значение переменной, можно использовать команду `echo`. Команда `echo` используется для вывода в терминале. В приведенном ниже примере команда отобразит значение переменной `HISTSIZE`:

```

sysadmin@localhost:~$ echo $HISTSIZE
1000
sysadmin@localhost:~$

```

Переменная `HISTSIZE` определяет максимальное количество предыдущих команд для хранения в списке истории. Чтобы отобразить значение переменной, используется знак `$` (доллар) перед именем переменной. Чтобы изменить значение переменной, символ `$` использовать не надо:

```

sysadmin@localhost:~$ HISTSIZE=500
sysadmin@localhost:~$ echo $HISTSIZE

```

```
500
```

```
sysadmin@localhost:~$
```

Существует много переменных оболочки, доступных для оболочки BASH, а также переменные, которые будут влиять на различные команды Linux. Обсуждение всех переменных оболочки выходит за рамки этой главы, однако по мере продвижения этого материала будут рассмотрены дополнительные переменные оболочки.

2.5. Переменная PATH

Одной из самых важных переменных оболочки BASH для понимания является переменная PATH.

Термин «PATH» относится к списку, который определяет, в каких каталогах будут находиться команды и программы, исполняемые оболочкой. Если ввести незнакомую команду, то на экран выведется ошибка «команда не найдена». Это связано с тем, что оболочка BASH не смогла найти команду под этим именем в любом из каталогов, включенных в PATH. Следующая команда отображает содержимое переменной PATH в текущей оболочке:

```
sysadmin@localhost:~$ echo $PATH
/home/sysadmin/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
sysadmin@localhost:~$
```

На основе полученного вывода оболочка сначала ищет команду в каталоге /home/sysadmin/bin. Если команда найдена в этом каталоге, она выполняется. Если она не найдена, оболочка будет искать команду в каталоге /usr/local/sbin и так далее.

Если команда не найдена ни в одном каталоге, указанном в переменной PATH, выводится сообщение об ошибке:

```
sysadmin@localhost:~$ zed
-bash: zed: command not found
sysadmin@localhost:~$
```

Если в системе установлено специальное программное обеспечение, может потребоваться изменить переменную PATH. Например, следующая команда добавит каталог /usr/bin/custom в переменную PATH:

```
sysadmin@localhost:~$ PATH=/usr/bin/custom:$PATH
sysadmin@localhost:~$ echo $PATH
```

```
/usr/bin/custom:/home/sysadmin/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
sysadmin@localhost:~$
```

2.6. Команда export

В оболочке BASH, используются два типа переменных. Переменные среды, такие как PATH и HOME, используются BASH при интерпретации команд и выполнении задач. Локальные переменные часто связаны с задачами, основанными на пользователях, и являются строчными. Чтобы создать локальную переменную, нужно ввести имя переменной=содержимое:

```
sysadmin@localhost:~$ variable1='Something'
```

Для обращения к содержимому переменной, нужно использовать знак \$ (доллар):

```
sysadmin@localhost:~$ echo $variable1
Something
```

Чтобы просмотреть переменные среды, используется команда env (поиск по результату с помощью grep, как показано здесь, будет рассмотрен в последующих главах). В этом случае поиск переменной variable1 в переменных среды не приводит к выходу:

```
sysadmin@localhost:~$ env | grep variable1
sysadmin@localhost:~$
```

После команды export переменная variable1 является переменной среды. На этот раз она находится в поиске:

```
sysadmin@localhost:~$ export variable1
sysadmin@localhost:~$ env | grep variable1
variable1=Something
```

Команда export также может использоваться для создания переменной среды:

```
sysadmin@localhost:~$ export variable2='Else'
sysadmin@localhost:~$ env | grep variable2
variable2=Else
```

Чтобы изменить значение переменной среды, нужно заново написать, чему она равна:

```
sysadmin@localhost:~$ variable1=$variable1 '$variable2'
sysadmin@localhost:~$ echo $variable1
Something Else
```

Экспортируемые переменные можно удалить с помощью команды `unset`:

```
sysadmin@localhost:~$ unset variable2
```

2.7. Команда `which`

Могут быть ситуации, когда в системе существуют разные версии одной и той же команды. Если команда не ведет себя так, как ожидалось, или если команда недоступна, можно воспользоваться командой `which`. Эта команда показывает, где оболочка находит команду или какую версию она использует. Было бы скучно вручную искать в каждом каталоге, указанном в переменной `PATH`. Вместо этого можно использовать команду `which`, чтобы отобразить полный путь к рассматриваемой команде:

```
sysadmin@localhost:~$ which date
/bin/date
sysadmin@localhost:~$ which cal
/usr/bin/cal
sysadmin@localhost:~$
```

Команда `which` ищет местоположение команды, выполнив поиск по переменной `PATH`.

2.8. Команда `type`

Команда `type` может использоваться для определения информации о различных командах. Некоторые команды берутся из определенного файла:

```
sysadmin@localhost:~$ type which
which is hashed (/usr/bin/which)
```

Этот вывод будет аналогичен выводу команды `which` (как обсуждалось в предыдущем разделе, в котором отображается полный путь команды):

```
sysadmin@localhost:~$ which which
/usr/bin/which
```

Команда `type` также может идентифицировать команды, встроенные в оболочку `bash`:

```
sysadmin@localhost:~$ type echo
echo is a shell builtin
```

В этом случае выход команды `which` значительно отличается от выхода команды `type`:

```
sysadmin@localhost:~$ which echo
/bin/echo
```

Используя параметр `-a`, команда `type` также может указать варианты исполнения команд:

```
sysadmin@localhost:~$ type -a echo
echo is a shell builtin
echo is /bin/echo
```

Команда `type` также может идентифицировать псевдонимы для других команд:

```
sysadmin@localhost:~$ type ll
ll is aliased to `ls -alF'
sysadmin@localhost:~$ type ls
ls is aliased to `ls --color=auto'
```

Вывод этих команд указывает, что `ll` является псевдонимом для `ls -alF`, и даже `ls` является псевдонимом для `ls --color=auto`

```
sysadmin@localhost:~$ which ll
sysadmin@localhost:~$ which ls
/bin/ls
```

Команда `type` поддерживает другие параметры и может одновременно показывать информацию по нескольким командам. Чтобы отобразить только одно слово, описывающее команду, можно использовать параметр `-t`:

```
sysadmin@localhost:~$ type -t echo ll which
builtin
alias
file
```

2.9. Псевдонимы

Псевдоним может использоваться для замены длинной команды с параметрами заменяемой её короткой формой — одним словом. Когда оболочка видит выполняемый псевдоним, она заменяет его на более длинную последовательность, прежде чем приступить к интерпретации команд.

Например, команда `ls -l` обычно псевдоним `l` или `ll`. Поскольку эти более мелкие команды легче набирать, быстрее запускается командная строка `ls -l`.

Можно определить, какие псевдонимы установлены в оболочке с помощью команды `alias`:

```
sysadmin@localhost:~$ alias
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -aLF'
alias ls='ls --color=auto'
```

Псевдонимы, которые видно из предыдущих примеров, были созданы с помощью файлов инициализации. Эти файлы предназначены для автоматического создания процесса создания псевдонимов и более подробно рассматриваются далее.

Новые псевдонимы могут быть созданы путем ввода псевдонима `alias name=command`, где `name` — это имя, которое сохраняется за псевдонимом, а `command` — это команда, которую нужно выполнить при запуске псевдонима.

Например, можно создать псевдоним `lh` для отображения длинного списка файлов, отсортированного по размеру с помощью команды `alias lh='ls -Shl'`. Ввод `lh` теперь должен привести к тому же выводу, что и команда `ls -Shl`:

```
sysadmin@localhost:~$ alias lh='ls -Shl'
sysadmin@localhost:~$ lh /etc/ppp
total 0
drwxr-xr-x 1 root root 10 Jan 29 2021 ip-down.d
drwxr-xr-x 1 root root 10 Jan 29 2021 ip-up.d
```

Псевдонимы, созданные таким образом, будут сохраняться только при открытой оболочке. Когда оболочка будет закрыта, новые псевдонимы будут потеряны. Кроме того, каждая оболочка имеет свои собственные псевдонимы, поэтому, если создается псевдоним в одной оболочке и затем открывается другая оболочка, то в новой оболочке такого псевдонима не будет.

2.10. Символы раскрытия

Символы раскрытия (`glob`, в русской литературе встречаются термины «джокер» и «универсализация файловых имен») имеют особое значение для оболочки. В отличие от команд, запускаемых оболочкой, или параметров и аргументов, которые оболочка передает командам, символы раскрытия интерпретируются самой оболочкой, прежде чем

она попытается выполнить любую команду. Это означает, что символы раскрытия могут использоваться с любой командой.

Символы раскрытия являются мощными инструментами, потому что позволяют указывать шаблоны, которые соответствуют именам файлов в каталоге, поэтому вместо одновременного управления одним файлом можно легко выполнять команды, которые будут влиять на многие файлы. Например, с помощью символов раскрытия можно управлять всеми файлами с определенным расширением или с определенной длиной имени файла.

2.10.1. Asterisk (звездочка *)

Символ `*` используется для представления любых символов в имени файла (любое количество раз, включая отсутствие символов). Например, предположим, что нужно отобразить все файлы в каталоге `/etc`, начинающиеся с буквы `t`:

```
sysadmin@localhost:~$ echo /etc/t*  
/etc/terminfo /etc/timezone  
sysadmin@localhost:~$
```

Шаблон `t*` означает «соответствовать любому файлу, который начинается с символа `t` и может иметь любое количество произвольных символов после `t`».

Можно использовать символ `*` в любом месте в шаблоне имени файла. Например, следующее будет соответствовать любому имени файла в каталоге `/etc`, который заканчивается на `.d`:

```
sysadmin@localhost:~$ echo /etc/*.d  
/etc/apparmor.d /etc/bash_completion.d /etc/cron.d /etc/depmod.d /etc/fstab.d  
/etc/init.d /etc/insserv.conf.d /etc/ld.so.conf.d /etc/logrotate.d /etc/modprobe.d  
/etc/pam.d /etc/profile.d /etc/rc0.d /etc/rc1.d /etc/rc2.d /etc/rc3.d /etc/rc4.d  
/etc/rc5.d /etc/rc6.d /etc/rcS.d /etc/rsyslog.d /etc/sudoers.d /etc/sysctl.d  
/etc/update-motd.d
```

В следующем примере будут отображаться все файлы в каталоге `/etc`, начинающиеся с буквы `r` и заканчивающиеся на `.conf`:

```
sysadmin@localhost:~$ echo /etc/r*.conf  
/etc/resolv.conf /etc/rsyslog.conf
```

2.10.2. Знак вопроса (?)

Вопросительный знак представляет любой символ. Каждый знак вопроса соответствует точно одному символу, не больше и не меньше.

Предположим, требуется отобразить все файлы в каталоге /etc, которые начинаются с буквы t и имеют ровно 7 символов после символа t:

```
sysadmin@localhost:~$ echo /etc/t???????
/etc/terminfo /etc/timezone
sysadmin@localhost:~$
```

Символы раскрытия могут использоваться вместе, чтобы составить еще более сложные шаблоны. Команда `echo /etc/*????????????????????` будет печатать только файлы в каталоге /etc с двадцатью или более символами в имени файла:

```
sysadmin@localhost:~$ echo /etc/*????????????????????
/etc/bindresvport.blacklist /etc/ca-certificates.conf
sysadmin@localhost:~$
```

Символы * и ? также могут использоваться вместе для поиска файлов с трехбуквенными расширениями, запустив `echo /etc/*.???` команда:

```
sysadmin@localhost:~$ echo /etc/*.???
/etc/blkid.tab /etc/issue.net
sysadmin@localhost:~$
```

2.10.3. Квадратные скобки []

Скобки используются для указания соответствия одному символу, представляя диапазон возможных символов. Например, `echo /etc/[gu]*` будет печатать любой файл, который начинается с символа g или u и содержит ноль или более дополнительных символов:

```
sysadmin@localhost:~$ echo /etc/[gu]*
/etc/gai.conf /etc/groff /etc/group /etc/group- /etc/gshadow /etc/gshadow-
/etc/ucf.conf /etc/udev /etc/ufw /etc/update-motd.d /etc/updatedb.conf
sysadmin@localhost:~$
```

Квадратные скобки также могут использоваться для представления ряда символов. Например, команда `echo /etc/[a-d]*` будет печатать все файлы, начинающиеся с любой буквы между a и d:

```
sysadmin@localhost:~$ echo /etc/[a-d]*
/etc/adduser.conf /etc/adjtime /etc/alternatives /etc/apparmor.d
/etc/apt /etc/bash.bashrc /etc/bash_completion.d /etc/bind /etc/bindresvport.blacklist
/etc/blkid.conf /etc/blkid.tab /etc/ca-certificates /etc/ca-certificates.conf
/etc/calendar /etc/cron.d /etc/cron.daily /etc/cron.hourly /etc/cron.monthly
/etc/cron.weekly /etc/crontab /etc/dbus-1 /etc/debconf.conf /etc/debian_version
/etc/default
/etc/deluser.conf /etc/depmod.d /etc/dpkg
sysadmin@localhost:~$
```

Команда `echo /etc/*[0-9]*` отображает любой файл, содержащий хотя бы одно число:

```
sysadmin@localhost:~$ echo /etc/*[0-9]*
/etc/dbus-1 /etc/iproute2 /etc/mke2fs.conf /etc/python2.7 /etc/rc0.d
/etc/rc1.d /etc/rc2.d /etc/rc3.d /etc/rc4.d /etc/rc5.d /etc/rc6.d
sysadmin@localhost:~$
```

Диапазон основан на текстовой таблице ASCII. Эта таблица определяет список символов, упорядочивая их в определенном стандартном порядке. Если указан неверный порядок, совпадение не будет найдено:

```
sysadmin@localhost:~$ echo /etc/*[9-0]*
/etc/*[9-0]*
sysadmin@localhost:~$
```

2.10.4. Восклицательный знак (!)

Восклицательный знак используется в сочетании с квадратными скобками для отрицания диапазона. Например, команда `echo [!DP]*` будет отображать любой файл, который не начинается с D или P.

2.11. Цитирование

Существует три типа кавычек, которые имеют особое значение для оболочки BASH: двойные кавычки, одинарные кавычки и обратные кавычки. Каждый набор кавычек указывает оболочке на то, что она должна обрабатывать текст внутри кавычек особым образом.

2.11.1. Двойные кавычки

Двойные кавычки приводят к тому, что перестают интерпретироваться некоторые метасимволы, включая символы раскрытия. В двойных кавычках `*` - это просто звездочка, `?` - это только знак вопроса и т.д. Это означает, что когда используется вторая команда `echo` ниже, оболочка BASH не преобразует шаблон раскрытия в имена файлов, которые соответствуют шаблону:

```
sysadmin@localhost:~$ echo /etc/[DP]*
/etc/DIR_COLORS /etc/DIR_COLORS.256color /etc/DIR_COLORS.lightbgcolor /etc/PackageKit
sysadmin@localhost:~$ echo "/etc/[DP] *"
/etc/[DP] *
sysadmin@localhost:~$
```

Это полезно, когда нужно отображать что-то на экране, что обычно является специальным символом для оболочки:

```
sysadmin@localhost:~$ echo "The glob characters are *, ? and [ ]"
The glob characters are *, ? and [ ]
sysadmin@localhost:~$
```

Двойные кавычки позволяют заменять команды, заменять переменные и разрешать некоторые другие метасимволы оболочки, которые еще не обсуждались. Например, в следующей демонстрации отображается значение переменной `PATH`:

```
sysadmin@localhost:~$ echo "The path is $PATH"
The path is /usr/bin/custom:/home/sysadmin/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
sysadmin@localhost:~$
```

2.11.2. Одинарные кавычки

Одиночные кавычки не позволяют оболочке выполнять интерпретацию специальных символов (в том числе символы раскрытия, переменные, подстановку команд и другие метасимволы, которые еще не обсуждались).

Например, если надо, чтобы символ `$` просто означал `$`, а не как индикатор для поиска значения переменной, нужно выполнить вторую команду, показанную ниже:

```
sysadmin@localhost:~$ echo The car costs $100
The car costs 00
sysadmin@localhost:~$ echo 'The car costs $100'
The car costs $100
```

2.11.3. Символ обратной косой черты (\)

Можно использовать альтернативный метод, для одинарной кавычки, если нужно зафиксировать один отдельный символ в строке. Например, предположим, что нужно напечатать следующее: «Стоимость услуг составляет \$100, а путь - `$PATH`». Если поместить это в двойные кавычки, записи `$100` и `$PATH` считаются переменными. Если поместить все в одинарные кавычки, `$100` и `$PATH` не будут являться переменными. Но что, если требуется, чтобы `$PATH` рассматривался как переменная, а `$100` - нет?

Если поместить символ обратной косой черты перед другим символом, он обрабатывается как символ «одиночных кавычек». Третья команда ниже демонстрирует использование символа `\`, а две другие демонстрируют, как переменные будут обрабатываться в двойных и одинарных кавычках:

```
sysadmin@localhost:~$ echo "The service costs $100 and the path is $PATH"
```

```
The service costs 00 and the path is /usr/bin/custom:/home/sysadmin/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
sysadmin@localhost:~$ echo 'The service costs $100 and the path is $PATH'
The service costs $100 and the path is $PATH
sysadmin@localhost:~$ echo The service costs \$100 and the path is $PATH
The service costs $100 and the path is /usr/bin/custom:/home/sysadmin/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
sysadmin@localhost:~$
```

2.11.4. Обратные кавычки

Обратные кавычки используются для указания команды внутри другой команды - процесса, называемого подстановкой команд. Это дает очень мощный и сложный механизм использования команд.

Рассмотрим пример. Для начала обратим внимание на вывод команды `date`:

```
sysadmin@localhost:~$ date
Сб 20 апр 2022 09:04:21 MSK
```

Теперь обратим внимание на вывод команды `echo Today is date`:

```
sysadmin@localhost:~$ echo Today is date
Today is date
sysadmin@localhost:~$
```

В предыдущей команде слово `date` обрабатывается как обычный текст, а оболочка просто передает слово `date` команде `echo`. Но, вероятно, есть потребность выполнить команду `date` и получить вывод этой команды, отправленной команде `echo`. Чтобы выполнить это, нужно запустить `echo` следующим образом.

```
sysadmin@localhost:~$ echo Today is `date`
Today is Сб 20 апр 2022 09:07:48 MSK
sysadmin@localhost:~$
```

2.12. Операции управления

Операции управления позволяют использовать сразу несколько команд или запускать дополнительные команды в зависимости от успеха предыдущей команды. Обычно эти управляющие операторы используются в сценариях, но их также можно использовать и в командной строке.

2.12.1. Точка с запятой

Точка с запятой может использоваться для запуска нескольких команд одной за другой. Каждая команда выполняется независимо и последовательно; независимо от результата первой команды, вторая будет выполняться после завершения первой, затем третьей и т. д.

Например, если нужно распечатать календарь на три месяца, можно выполнить в командной строке:

```
sysadmin@localhost:~$ cal 1 2023; cal 2 2023; cal 3 2023
Январь 2023
Вс Пн Вт Ср Чт Пт Сб
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31

Февраль 2023
Вс Пн Вт Ср Чт Пт Сб
      1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28

Март 2023
Вс Пн Вт Ср Чт Пт Сб
      1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
```

2.12.2. Двойной амперсанд (&&)

Двойной амперсанд && действует как логический "и". Если первая команда выполнена успешно, тогда будет выполняться вторая команда (справа от &&). Если первая команда завершилась неудачей, вторая команда не будет запущена.

Чтобы лучше понять, как это работает, сначала рассмотрим концепцию отказа и успеха команд. Команды успешны, когда они работают правильно и терпят неудачу, когда есть ошибка. Например, рассмотрим командную строку `ls /etc/xml`. Команда будет успешной, если каталог `/etc/xml` доступен и завершится неудачей, если это не так.

В примере ниже, первая команда будет успешной, потому что каталог `/etc/xml` существует и доступен, а вторая команда сообщает об ошибке:

```
sysadmin@localhost:~$ ls /etc/xml
catalog catalog.old xml-core.xml xml-core.xml.old
sysadmin@localhost:~$ ls /etc/junk
ls: cannot access /etc/junk: No such file or directory
sysadmin@localhost:~$
```

То, как можно использовать успешное завершение первой команды, для запуска второй показано ниже:

```
sysadmin@localhost:~$ ls /etc/xml && echo success
catalog catalog.old xml-core.xml xml-core.xml.old
success
sysadmin@localhost:~$ ls /etc/junk && echo success
ls: cannot access /etc/junk: No such file or directory
sysadmin@localhost:~$
```

В первом примере выше команда `echo` выполняется, поскольку команда `ls` выполнена успешно. Во втором примере команда `echo` не была выполнена, так как команда `ls` потерпела неудачу.

2.12.3. Две вертикальных черты

Две вертикальных черты `||` являются логическим "или". Они работают аналогично `&&`; в зависимости от результата первой команды, вторая команда будет либо запущена, либо будет пропущена. Если первая команда выполняется успешно, вторая команда пропускается. Если первая команда выполнена с ошибкой, тогда будет выполнена вторая команда. Другими словами: «Либо выполнить первую команду, либо вторую».

В следующем примере команда `echo` будет выполняться только в случае сбоя команды `ls`:

```
sysadmin@localhost:~$ ls /etc/xml || echo failed
catalog catalog.old xml-core.xml xml-core.xml.old
sysadmin@localhost:~$ ls /etc/junk || echo failed
ls: cannot access /etc/junk: No such file or directory
failed
sysadmin@localhost:~$
```

Контрольные вопросы

1. Как можно получить доступ к интерфейсу командной строки CLI?
2. Как вывести историю введенных команд и повторить одну из команд?
3. Для чего нужна команда `export`?
4. Какие существуют типы исполняемых команд?
5. В каком случае команда может выполняться с установленными заранее параметрами, если пользователь их не вводит?

6. Перечислите символы раскрытия.
7. В каких случаях необходимо использовать символы цитирования?
8. Какие варианты есть для запуска из консоли нескольких команд?
9. Основная цель использования символов раскрытия (glob)?

Глава 3. ПОЛУЧЕНИЕ ПОМОЩИ

3.1. Справочные страницы

В операционной системе Linux есть тысячи команд с различными опциями, что делает эти команды мощными инструментами.

Однако, такое количество команд представляет дополнительную сложность, которая может создать путаницу. Знание того, где найти справочную информацию во время работы в Linux, является важным навыком для любого пользователя. Справочная информация может напомнить, как работает та или иная команда, а также может быть информационным ресурсом при изучении новых команд.

UNIX – это операционная система, на основе которой был создан Linux. Разработчики UNIX создали справочные документы, называемые справочными страницами.

Справочные страницы используются для описания функций команд. Они предоставляют базовое описание цели команды, а также предоставляют подробную информацию о параметрах команды.

3.1.1. Просмотр справочных страниц

Для просмотра справочной страницы нужно выполнить команду `man command`. Например, команда `man cal` отобразит на дисплее справочную страницу для команды `cal`:

```
CAL(1)                                BSD General Commands Manual          CAL(1)

NAME
cal, ncal -- displays a calendar and the date of Easter

SYNOPSIS
cal [-3hjy] [-A number] [-B number] [[month] year]
cal [-3hj] [-A number] [-B number] -m month [year]
ncal [-3bhjJpwySM] [-A number] [-B number] [-s country_code] [[month]
year]
ncal [-3bhJeoSM] [-A number] [-B number] [year]
ncal [-CN] [-H yyyy-mm-dd] [-d yyyy-mm]

DESCRIPTION
The cal utility displays a simple calendar in traditional format and ncal
offers an alternative layout, more options and the date of Easter. The new
format is a little cramped but it makes a year fit on a 25x80 terminal. If
arguments are not specified, the current month is displayed.

The options are as follows:
-h      Turns off highlighting of today.
```


3.1.2. Управление отображением справочной страницы

Команда `man` использует «пейджер» (от англ. page - страница) для отображения документов. Обычно в качестве «пейджера» используется команда `less`, но в некоторых дистрибутивах это может быть команда `more`. Обе команды похожи в том, как они работают.

Для просмотра возможностей перемещения по справке, можно нажать символ `h`. Появится страница справки:

SUMMARY OF LESS COMMANDS

Commands marked with * may be preceded by a number, N.
Notes in parentheses indicate the behavior if N is given.
A key preceded by a caret indicates the Ctrl key; thus ^K is ctrl-K.

```
h  H          Display this help.
q  :q  Q  :Q  ZZ  Exit.
```

MOVING

```
e  ^E  j  ^N  CR  *  Forward one line (or N lines).
y  ^Y  k  ^K  ^P  *  Backward one line (or N lines).
f  ^F  ^V  SPACE *  Forward one window (or N lines).
b  ^B  ESC-v     *  Backward one window (or N lines).
z                               *  Forward one window (and set window to N).
w                               *  Backward one window (and set window to N).
ESC-SPACE                 *  Forward one window, but don't stop at end-of-file.
d  ^D                     *  Forward one half-window (and set half-window to N)
u  ^U                     *  Backward one half-window (and set half-window to N)
ESC-)  RightArrow *  Left one half screen width (or N positions).
HELP -- Press RETURN for more, or q when done
```

Если дистрибутив использует команду `less`, будет доступно большое количество команд. В таблице 2 представлены наиболее полезные команды.

Таблица 2. Клавиши в команде `less`

Команда	Описание
Return (или Enter)	Спуститься на строчку вниз
Space	Спуститься на страницу вниз
/term	Поиск по термину
N	Найти следующий элемент поиска
1G	В начало
G	В конец
h	Справка
q	Заккрыть справочную страницу

3.1.3. Разделы справочной страницы

Справочные страницы разбиты на разделы. Каждый раздел предназначен для предоставления конкретной информации о команде. Хотя есть общие разделы, некоторые разработчики также создают разделы, которые можно увидеть только на определенной справочной странице.

В таблице 3 описаны некоторые из наиболее распространенных разделов, которые можно найти в справочных страницах:

Таблица 3. Разделы справочных страниц

Название раздела	Назначение
Name	Название команды и описание
Synopsis	Примеры выполнения команды
Description	Более подробное описание команды
Options	Список параметров для команды, а также описание того, как они используются.
Files	Список файлов, связанных с командой, а также описание того, как они используются. Эти файлы могут использоваться для настройки более продвинутых функций команды.
Author	Имя человека, который создал страницу и (иногда) его контактные данные.
Reporting Bugs	Сведения о том, как сообщить о проблемах с командой.
Copyright	Основные сведения об авторских правах.
See also	Сведения о том, где можно найти дополнительную информацию. Этот раздел может содержать другие команды, связанные с этой командой.

3.1.4. Раздел Synopsis справочной страницы

Раздел Synopsis справочной страницы предоставляет краткий пример использования команды. Например, Synopsis справочной страницы для команды `cal`:

```
SYNOPSIS
cal [-3hjy] [-A number] [-B number] [[[day] month] year]
```

Квадратные скобки `[]` используются, чтобы указать, что данные параметры не обязательны для выполнения команды. Например, `[-3hjy]` означает, что можно использовать параметры `-h`, `-j`, `-y`, `1` или `3`, но ни один из них не требуется для нормальной работы команды `cal`.

Второй набор квадратных скобок (`[[[day] month] year]`) демонстрирует другую функцию; можно указать год сам по себе (без

указания месяца и дня), но, если указать месяц, нужно также указать год. Если указать день, то также нужно указать месяц и год.

Несколько компонент раздела Synopsis, которые могут вызвать путаницу, встречаются в команде date:

```
SYNOPSIS
date [OPTION]... [+FORMAT]
date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]
```

Есть два синтаксиса для команды date. Первый используется для отображения даты в системе, а второй - для установки даты.

3.1.5. Поиск по справочной странице

Чтобы выполнить поиск термина на справочной странице, нужно нажать клавишу / и ввести термин, затем нажать клавишу Enter. Поиск будет осуществлен с текущей позиции до конца страницы.

Если термин не найден, или достигли конца страницы, программа сообщит, что «шаблон» не найден (нажмите Return). Если совпадение найдено, и требуется перейти к следующему найденному совпадению, надо нажать n. Чтобы вернуться к предыдущему совпадению, надо нажать N.

3.1.6. Справочные страницы, отсортированные по разделам

До сих пор речь шла о справочных страницах для команд. Однако, иногда файлы конфигурации также содержат справочные страницы. Файлы конфигурации (иногда называемые системными файлами) содержат сведения, которые используются для хранения информации об операционной системе или службах.

Кроме того, существует несколько типов команд (пользовательские команды, системные команды и команды администрирования), а также другие функции, требующие документации, такие как библиотеки и компоненты ядра.

В результате типичный дистрибутив Linux содержит тысячи справочных страниц. Чтобы упорядочить все эти справочные страницы, они разбиваются по разделам.

Имеется девять разделов справочных страниц по умолчанию:

1. Исполняемые программы или команды оболочки
2. Системные вызовы (функции, предоставляемые ядром)
3. Библиотечные вызовы (функции внутри программных библиотек)
4. Специальные файлы (обычно в каталоге /dev)

5. Форматы файлов и соглашения, например, /etc/passwd
6. Игры
7. Разное (включая макропакеты и соглашения)
8. Команды системного администрирования (обычно только для root)
9. Подпрограммы ядра

Команда `man` ищет заданную команду в каждом из этих разделов по порядку, пока не найдет первое "совпадение". Например, при выполнении команды `man cal` в первом разделе (исполняемые программы или команды оболочки) выполняется поиск справочной страницы `cal`. Если команда не найдена, то она ищется во втором разделе и т.д. Если после поиска по всем разделам не будет найдено ни одной справочной страницы, появится сообщение об ошибке:

```
sysadmin@localhost:~$ man zed
No manual entry for zed
sysadmin@localhost:~$
```

3.2. Команда `Info`

Справочные страницы являются большими источниками информации, но они, как правило, имеют несколько недостатков. Один из недостатков заключается в том, что каждая справочная страница представляет собой отдельный документ, не связанный с другими справочными страницами. Хотя некоторые справочные страницы имеют раздел `See Also`, который может ссылаться на другие справочные страницы, но в основном эти страницы не связаны друг с другом.

Команда `info` также предоставляет документацию по командам и функциям операционной системы. Назначение этой команды отличается от справочных страниц: предоставить источник документации, который обеспечивает логическую организационную структуру, что упрощает чтение документации.

В документах `info` информация разбивается на категории, как содержание книги. Гиперссылки ссылаются на страницы с информацией по отдельным темам для каждой команды или функции. Вся документация объединена в единую «книгу», в которой можно перейти на верхний уровень документации и просмотреть оглавление, представляющее всю доступную документацию.

Другое преимущество документации команды `info` над справочными страницами `man` заключается в том, что стиль изложения информационных документов более удобен для изучения темы. Справочные страницы – это справочный ресурс, а информационные документы – это учебное руководство.

3.2.1. Отображение документации для команды

Чтобы отобразить документацию для команды, нужно выполнить команду `info command` (вместо слова `command` подставляется название команды, о которой требуется получить информацию). Например, ниже показан результат выполнения команды `info ls`:

```
File: coreutils.info, Node: ls invocation, Next: dir invocation, Up: Directory
listing

10.1 `ls': List directory contents
=====

The `ls' program lists information about files (of any type, including directo-
ries). Options and file arguments can be intermixed arbitrarily, as usual.

For non-option command-line arguments that are directories, by
default `ls' lists the contents of directories, not recursively, and
omitting files with names beginning with `.'. For other non-option
arguments, by default `ls' lists just the file name. If no non-option
argument is specified, `ls' operates on the current directory, acting
as if it had been invoked with a single argument of `.'.

By default, the output is sorted alphabetically, according to the
locale settings in effect.(1) If standard output is a terminal, the

output is in columns (sorted vertically) and control characters are
output as question marks; otherwise, the output is listed one per line
and control characters are output as-is.
--zz-Info: (coreutils.info.gz)ls invocation, 58 lines --Top-----
Welcome to Info version 5.2. Type h for help, m for menu item.
```

Первая строка содержит некоторую информацию о местоположении в документации. Документация разбита на узлы, и в приведенном выше примере представлен узел `ls invocation`. Если бы пользователь перешел к следующему узлу (как к следующей главе в книге), он бы оказался в узле `dir invocation`. Если бы пользователь поднялся на уровень выше, он бы оказался в узле `directory listing`.

3.2.2. Перемещение во время просмотра документации

Введя букву `h` при чтении документа `info`, можно получить список команд перемещения:

```
Basic Info command keys
```

```

l      Close this help window.
q      Quit Info altogether.
H      Invoke the Info tutorial.
Up     Move up one line.
Down   Move down one line.
DEL    Scroll backward one screenful.
SPC    Scroll forward one screenful.
Home   Go to the beginning of this node.
End     Go to the end of this node.
TAB    Skip to the next hypertext link.
RET    Follow the hypertext link under the cursor.
l      Go back to the last node seen in this window.
[      Go to the previous node in the document.
]      Go to the next node in the document.
p      Go to the previous node on this level.
n      Go to the next node on this level.
u      Go up one level.
-----Info: *Info Help*, 466 lines --Top-----

```

Для возвращения на предыдущую открытую страницу надо нажать l. Это вернет к документу и позволит продолжить чтение. Чтобы полностью выйти, надо нажать q.

В таблице 4 приводится сводка полезных команд:

Таблица 4. Перемещения info

Команда	Описание
Down Arrow	спуститься вниз на одну строчку
Space	спуститься вниз на одну страницу
s	поиск термина
[к предыдущему узлу
]	к следующему узлу
u	подняться на уровень вверх
TAB	перейти к следующей гиперссылке
HOME	в начало
END	в конец
h	справка
L	закрыть окно справки
q	закрыть документ

Если пролистать документ, в конечном итоге можно увидеть меню для команды ls:

```

* Menu:

* Which files are listed::
* What information is listed::
* Sorting the output::
* Details about version sort::
* General output formatting::
* Formatting file timestamps::
* Formatting the file names::

----- Footnotes -----

```

```
(1) If you use a non-POSIX locale (e.g., by setting `LC_ALL' to
`en_US'), then `ls' may produce output that is sorted differently than
you're accustomed to. In that case, set the `LC_ALL' environment
variable to `C'.
--zz-Info: (coreutils.info.gz)ls invocation, 58 lines --Top-----
```

Пункты меню являются гиперссылками, переход по которым приведет к узлам, которые содержат больше информации о команде `ls`. Например, если переместить курсор в строку "`* Sorting the output::`" и нажать клавишу `Enter`, пользователь перейдет к узлу, который описывает сортировку выходных данных команды `ls`:

```
File: coreutils.info, Node: Sorting the output, Next: Details about version s\ort,
Prev: What information is listed, Up: ls invocation
10.1.3 Sorting the output

-----
These options change the order in which `ls' sorts the information it
outputs. By default, sorting is done by character code (e.g., ASCII
order).

`-c'
`--time=ctime'
`--time=status'
    If the long listing format (e.g., `-l', `-o') is being used, print
    the status change time (the `ctime' in the inode) instead of the
    modification time. When explicitly sorting by time (`--sort=time'
    or `-t') or when not using a long listing format, sort according
    to the status change time.

`-f'
    Primarily, like `-U'--do not sort; list the files in whatever
    order they are stored in the directory. But also enable `-a' (lis
--zz-Info: (coreutils.info.gz)Sorting the output, 68 lines --Top-----
```

Чтобы вернуться к предыдущему узлу, надо нажать `u`. Для перехода в то место, в котором были до перехода к узлу, надо нажать `l`.

3.3. Дополнительные источники помощи

Во многих случаях либо справочные страницы, либо информационная документация предоставят необходимую информацию. Однако в некоторых случаях может потребоваться поиск в других местах.

3.3.1. Использование параметра `--help`

Многие команды могут предоставить базовую информацию, очень похожую на краткий обзор, найденный на страницах руководства, в случае применения опции `--help`. Это полезно для изучения базового использования команды:

```

sysadmin@localhost:~$ ps --help
***** simple selection *****
-A all processes
-N negate selection
-a all w/ tty except session leaders
-d all except session leaders
-e all processes
T all processes on this terminal
a all w/ tty, including other users
g OBSOLETE -- DO NOT USE
r only running processes
x processes w/o controlling ttys
***** output format *****
-o,o user-defined -f full
-j,j job control s signal
-O,O preloaded -o v virtual memory

***** selection by list *****
-C by command name
-G by real group ID (supports names)
-U by real user ID (supports names)
-g by session OR by effective group name
-p by process ID
-s processes in the sessions given
-t by tty
-u by effective user ID (supports names)
U processes for specified users
t by tty
***** long options *****
--Group --User --pid --cols --ppid
--group --user --sid --rows --info
--cumulative --format --deselect

-l,l long u user-oriented --sort --tty --forest --version
-F extra full X registers --heading --no-headi
***** misc options *****
-V,V show version L list format codes f ASCII art forest
-m,m,-L,-T,H threads S children in sum -y change -l format
-M,Z security data c true command name -c scheduling class
-w,w wide output n numeric WCHAN,UID -H process hierarchy
sysadmin@localhost:~$

```

3.3.2. Дополнительная документация системы

В большинстве систем существует каталог, в котором находится дополнительная документация. Это часто место, где разработчики, которые создают дополнительное (стороннее) программное обеспечение, могут хранить файлы документации. Как правило, это место, куда пойдут системные администраторы, чтобы узнать, как настроить сложные программные службы. Однако иногда обычные пользователи также находят эту документацию полезной.

Эти файлы документации часто называют "readme" файлы, так как файлы обычно имеют имена, такие как readme или readme.txt. Расположение этих файлов может различаться в зависимости от используемого дистрибутива (чаще всего это /usr/share/doc и /usr/doc).

3.4. Поиск файлов, команд и документации

Команда `whatis` (или `man -f`) сообщит, в каком разделе хранится справочная страница. При использовании этой команды можно столкнуться с необычными выходными данными, такими как:

```

sysadmin@localhost:~$ whatis ls
ls (1)                - list directory contents

```



```
ls (lp)                - list directory contents
sysadmin@localhost:~$
```

На основе этих выходных данных есть две команды, которые перечисляют содержимое каталога. Простой ответ на то, почему есть две команды `ls`, заключается в том, что UNIX имел два основных варианта, в результате чего некоторые команды разрабатывались "параллельно". Это привело к тому, что некоторые команды ведут себя по-разному в разных вариантах UNIX. Многие современные дистрибутивы Linux включают команды из обоих вариантов UNIX.

Однако это создает некоторую проблему: при выполнении команды `ls`, какая команда выполняется? В центре внимания следующих нескольких разделов будет ответ на этот вопрос, а также описаны инструменты, чтобы найти, где эти файлы находятся в системе.

3.4.1. Где находятся команды?

Для поиска расположения команды или справочных страниц для команды используется команда `whereis`. Эта команда выполняет поиск команд, исходных файлов и справочных страниц в определенных местах, где обычно хранятся эти файлы:

```
sysadmin@localhost:~$ whereis ls
ls: /bin/ls /usr/share/man/man1p/ls.1.gz /usr/share/man/man1/ls.1.gz
sysadmin@localhost:~$
```

Справочные страницы обычно легко различаются от самих команд, так как они обычно сжимаются с помощью команды `gzip`, в результате чего имя файла заканчивается на `.gz`.

Интересно то, что есть две справочные страницы, но только одна команда (`/bin/ls`). Это связано с тем, что команда `ls` может использоваться с параметрами/функциями, которые описаны любой из справочных страниц. Таким образом, при изучении того, что можно сделать с помощью команды `ls`, можно исследовать обе справочные страницы. К счастью, это больше исключение, так как большинство команд имеют только одну справочную страницу.

3.4.2. Как найти любой файл или каталог с помощью команды `locate`?

Команда `whereis` предназначена специально для поиска команд и справочных страниц. Это полезно, но есть моменты, когда необходимо найти произвольный файл или каталог, а не только файлы, которые являются командами или справочными страницами.

Чтобы найти любой файл или каталог, можно использовать команду `locate`. Эта команда выполняет поиск в базе данных всех файлов и каталогов, которые находились в системе при создании базы данных. Обычно команда для создания этой базы данных выполняется каждую ночь.

```
sysadmin@localhost:~$ locate gshadow
/etc/gshadow
/etc/gshadow-

/usr/include/gshadow.h
/usr/share/man/cs/man5/gshadow.5.gz
/usr/share/man/da/man5/gshadow.5.gz
/usr/share/man/de/man5/gshadow.5.gz
/usr/share/man/fr/man5/gshadow.5.gz
/usr/share/man/it/man5/gshadow.5.gz
/usr/share/man/man5/gshadow.5.gz
/usr/share/man/ru/man5/gshadow.5.gz
/usr/share/man/sv/man5/gshadow.5.gz
/usr/share/man/zh_CN/man5/gshadow.5.gz
sysadmin@localhost:~$
```

Любые файлы, которые создали в течении дня, обычно не будут доступны для поиска с помощью команды `locate`. Если есть доступ к системе от имени пользователя `root` (администратора системы), можно вручную обновить базы данных `locate`, выполнив команду `updatedb`. Обычные пользователи не могут обновить файл базы данных.

При использовании команды `locate` в качестве обычного пользователя выходные данные могут быть ограничены из-за прав доступа к файлам. Если у пользователя нет доступа к файлу или каталогу в файловой системе из-за разрешений, команда `locate` не вернет эти имена. Это функция безопасности, предназначенная для предотвращения "изучения" файловой системы с помощью базы данных `locate`. Пользователь `root` (администратор системы) может осуществлять поиск по всем файлам в базе `locate`.

3.4.3. Подсчет количества файлов

Выходные данные команды `locate` могут быть довольно большими. Когда ищут файлы с таким именем, как `passwd`, команда поиска будет выдавать каждый файл, который содержит строку для смены пароля, а не только файлы с именем `passwd`.

Во многих случаях можно начать с подсчета количества совпадений. Это можно сделать с помощью параметра `-c` команды `locate`:

```
sysadmin@localhost:~$ locate -c passwd
97
sysadmin@localhost:~$
```

3.4.4. Ограничение вывода

Можно ограничить выводимые данные при выполнении команды `locate` с помощью параметра `-b`. Эта опция будет включать только списки, которые удовлетворяют условиям поиска в базовом имени файла. Базовое имя - это часть имени файла, не включая имена каталогов.

```
sysadmin@localhost:~$ locate -c -b passwd
83
sysadmin@localhost:~$
```

Из предыдущих выходных данных видно, что все равно будет много результатов при использовании опции `-b`. Чтобы ограничить вывод еще больше, можно поместить символ `\` перед поисковым термином. Этот символ ограничивает выходные данные именами файлов, которые точно соответствуют термину:

```
sysadmin@localhost:~$ locate -b "\passwd"
/etc/passwd
/etc/cron.daily/passwd
/etc/pam.d/passwd
/usr/bin/passwd
/usr/share/doc/passwd
/usr/share/lintian/overrides/passwd
sysadmin@localhost:~$
```

3.5. Поиск файлов с помощью команды “find”

Одна из проблем, с которой сталкиваются пользователи при работе с файловой системой, вспомнить место, где хранятся файлы. Большинство файлов, с которыми пользователь будет работать, - это те, которые он создает сам. В результате чаще будет искать файлы в своем домашнем каталоге. Однако иногда может потребоваться поиск в других местах файловой системы, чтобы найти файлы, созданные другими пользователями.

Команда `find` - это очень мощный инструмент, который можно использовать для поиска файлов в файловой системе. Эта команда может выполнять поиск файлов по имени, в том числе с использованием символов подстановки, если полное имя не известно. Кроме того,

можно искать файлы на основе метаданных файла, таких как тип файла, размер файла и владелец.

3.5.1. Поиск по имени файла

Для поиска файла по имени используется опция `-name`:

```
sysadmin@localhost:~$ find /etc -name hosts
find: `/etc/dhcp': Permission denied
find: `/etc/cups/ssl': Permission denied
find: `/etc/pki/CA/private': Permission denied
find: `/etc/pki/rsyslog': Permission denied
find: `/etc/audisp': Permission denied
find: `/etc/named': Permission denied
find: `/etc/lvm/cache': Permission denied
find: `/etc/lvm/backup': Permission denied
find: `/etc/lvm/archive': Permission denied
/etc/hosts
find: `/etc/ntp/crypto': Permission denied
find: `/etc/polkit-1/localauthority': Permission denied
find: `/etc/sudoers.d': Permission denied
find: `/etc/sssd': Permission denied
/etc/avahi/hosts
find: `/etc/selinux/targeted/modules/active': Permission denied
find: `/etc/audit': Permission denied
sysadmin@localhost:~$
```

3.5.2. Отображение сведений о файле

Получение подробной информации о файле может быть полезным при использовании команды `find`, потому что только имя файла не может быть достаточной информацией, чтобы найти правильный файл.

Например, может быть семь файлов с именем `hosts`, но если известно, что файл `hosts`, который нужен, был недавно изменен, то отметка времени изменения файла будет полезна для просмотра.

Чтобы просмотреть эти сведения о файле, используются параметры `-ls` для команды `find`:

```
sysadmin@localhost:~$ find /etc -name hosts -ls 2> /dev/null
  41    4 -rw-r--r--    1 root    root      158 Jan 12 2020 /etc/hosts
 6549    4 -rw-r--r--    1 root    root     1130 Jul 19 2021 /etc/avahi/hosts
sysadmin@localhost:~$
```

3.5.3. Поиск файлов по размеру

Одним из полезных вариантов поиска является опция, которая позволяет искать файлы по размеру. Опция `-size` позволяет искать файлы, которые больше или меньше указанного размера, а также искать точный размер файла.

При указании размера файла можно указать размер в байтах (с), килобайтах (k), мегабайтах (M) или Гигабайтах (G). Например, ниже приведен поиск файлов в структуре каталогов /etc размером ровно 10 байт:

```
sysadmin@localhost:~$ find /etc -size 10c -ls 2>/dev/null
  432    4 -rw-r--r--    1 root    root          10 Jan 28  2022 /etc/adjtime
 8814    0 drwxr-xr-x    1 root    root          10 Jan 29  2022 /etc/ppp/ip-d
own.d
 8816    0 drwxr-xr-x    1 root    root          10 Jan 29  2022 /etc/ppp/ip-u
p.d
 8921    0 lrwxrwxrwx    1 root    root          10 Jan 29  2022 /etc/ssl/cert
s/349f2832.0 -> EC-ACC.pem
 9234    0 lrwxrwxrwx    1 root    root          10 Jan 29  2022 /etc/ssl/cert
s/aeb67534.0 -> EC-ACC.pem
 73468    4 -rw-r--r--    1 root    root          10 Nov 16 20:42 /etc/hostname
sysadmin@localhost:~$
```

Контрольные вопросы

1. Как используя справочные страницы узнать о команде вызова справочных страниц?
2. В чем отличие команды `man` от команды `info`?
3. Какую надо нажать клавишу, чтобы выйти из просмотра справочной страницы.
4. Какие существуют способы получения информации по командам помимо `man` и `info`?
5. Если пользователь читает синопсис команды на странице руководства, что означают элементы в квадратных скобках?
6. Какие команды можно использовать для поиска файлов?
7. Какие две команды пейджера используются командой `man` для управления перемещением в документе?
8. Какой из символов обрабатывает следующий за ним символ, как если бы он был заключен в одинарные кавычки?

Глава 4. РАБОТА С ФАЙЛАМИ И КАТАЛОГАМИ

4.1. Общие сведения о файлах и каталогах

При работе в операционной системе Linux необходимо знать, как управлять файлами и каталогами. Некоторые дистрибутивы Linux имеют приложения графического интерфейса, которые позволяют управлять файлами, но администраторам важно знать, как выполнять эти операции через командную строку. Командная строка содержит богатый набор команд, которые позволяют управлять файлами.

Файлы используются для хранения данных, таких как текст, графика и программы. Каталоги (или "папки", директории) используются для обеспечения иерархической организационной структуры. Эта структура несколько отличается от структуры в системах Microsoft Windows.

В системе Windows верхний уровень структуры каталогов называется "Мой компьютер". Каждое физическое устройство (жесткий диск, DVD-диск, USB флэш-диск, сетевой диск и т. д.) отображается под «Моим компьютером», каждому назначена буква диска, например C: или D:. Визуальное представление этой структуры показано на рисунке 3.

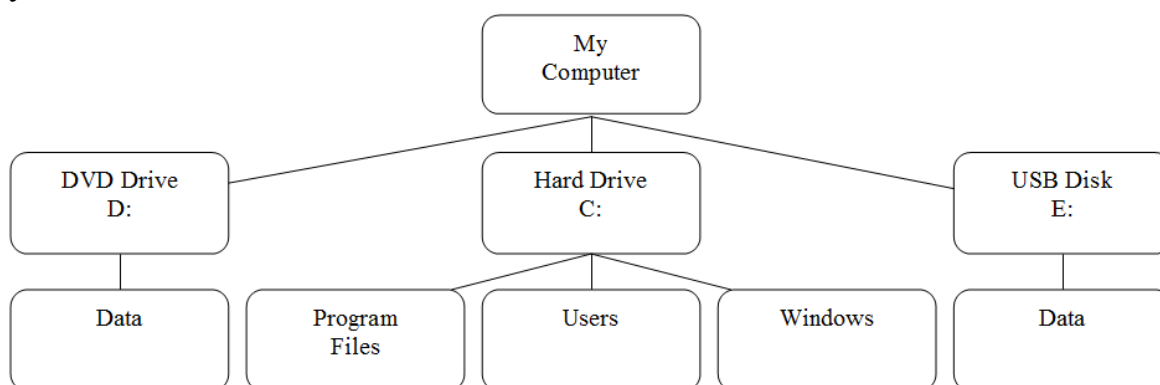


Рис. 3. Дерево каталогов Windows

Как и в Microsoft Windows, структура каталогов Linux имеет верхний уровень, однако она называется не «Мой компьютер», а корневой каталог и обозначается символом /. В Linux также нет дисков. Каждое физическое устройство доступно в каталоге, а не в букве диска.

Визуальное представление типичной структуры каталогов Linux показано на рисунке 4.

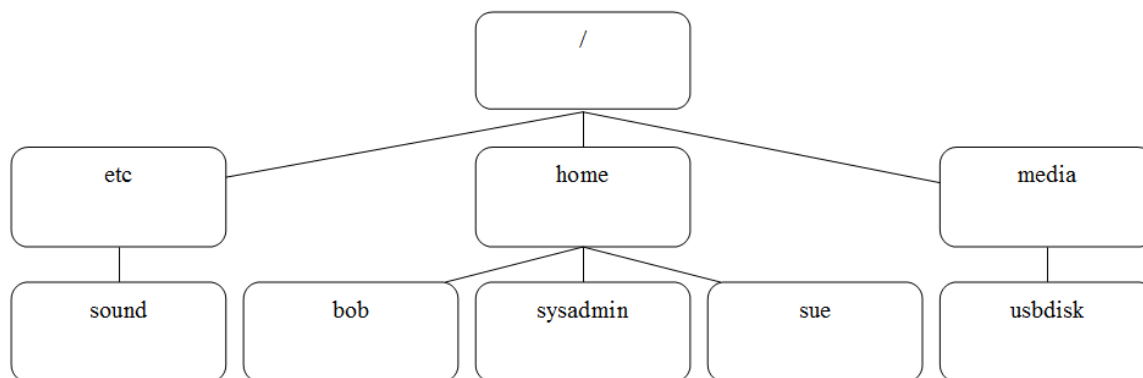


Рис. 4. Дерево каталогов Linux

Для просмотра корневой файловой системы надо ввести `ls /`:

```
sysadmin@localhost:~$ ls /  
bin    dev    lib    libx32  mnt    root  snap    sys  var  
boot   etc    lib32  lost+found  opt    run    srv      tmp  
cdrom  home   lib64  media    proc   sbin   swapfile  usr
```

В корневом каталоге существует множество каталогов, включая, например, каталог `/boot`, содержащий файлы для загрузки компьютера.

4.1.1. Путь к каталогу

Используя схему в предыдущем разделе, можно увидеть, что существует каталог с именем `sound` в каталоге с именем `etc`, который находится в каталоге `/`. Путь позволяет указать точное расположение каталога. Для каталога `sound` путь будет `/etc/sound`. Первый символ `/` представляет корневой каталог, в то время как каждый следующий символ `/` используется для разделения имен каталогов.

Такой путь называется абсолютным путем. В абсолютном пути всегда указывается направление к каталогу (или файлу), начиная с корневого каталога.

На рисунке 5 показаны три дополнительных абсолютных пути.

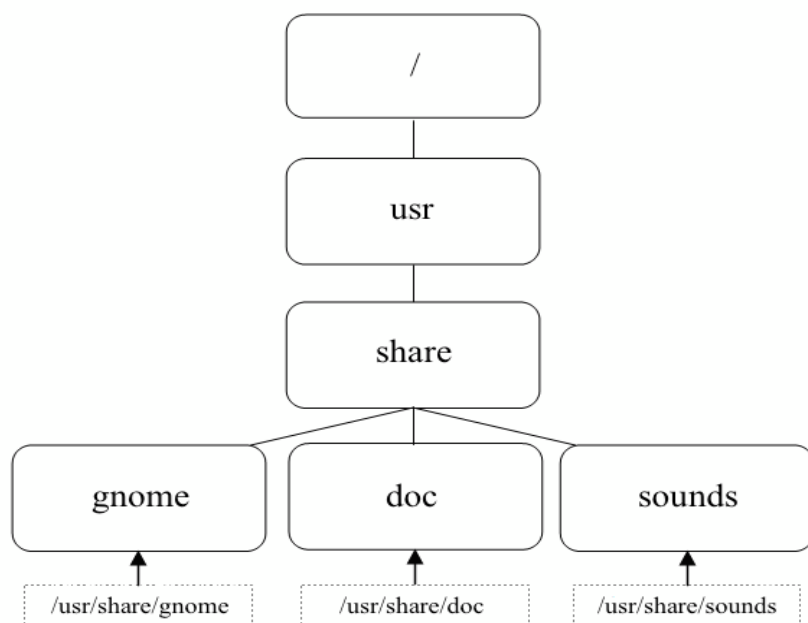


Рис. 5. Абсолютные пути к каталогам

4.1.2. Домашний каталог

Термин «домашний каталог» часто вызывает путаницу у начинающих пользователей Linux. Для начала, в большинстве дистрибутивов Linux есть каталог под названием `home` в корневом каталоге: `/home`.

В этом каталоге `/home` будет каталог для каждого пользователя в системе. Имя каталога будет совпадать с именем пользователя, поэтому у пользователя с именем "bob" будет домашний каталог с именем `/home/bob`.

Домашний каталог является очень важным каталогом. Для начала, когда пользователь открывает оболочку (shell), он автоматически попадает в свой домашний каталог.

Кроме того, домашний каталог является одним из немногих каталогов, где у пользователя есть полный доступ для создания и удаления дополнительных файлов и каталогов. Большинство других каталогов в файловой системе Linux защищены правами доступа к файлам.

В большинстве дистрибутивов Linux доступ к файлам в домашнем каталоге имеют только сам пользователь и администратор системы (пользователь `root`). Это можно изменить с помощью разрешений для файлов.

Есть специальный символ, который можно использовать для представления домашнего каталога: `~` (тильда). Если домашний каталог `/home/sysadmin`, можно просто ввести `~` в командной строке

вместо `/home/sysadmin`. Можно обратиться к домашнему каталогу другого пользователя с помощью нотации `~user`, где `user` - это имя учетной записи пользователя, на домашний каталог которого требуется сослаться. Например, `~bob`:

```
sysadmin@localhost:~$ cd ~
sysadmin@localhost:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
sysadmin@localhost:~$
```

Список показывает подкаталоги, содержащиеся в домашнем каталоге. Изменение каталогов требует внимания к деталям:

```
sysadmin@localhost:~$ cd downloads
-bash: cd: downloads: No such file or directory
sysadmin@localhost:~$
```

Почему приведенная выше команда привела к ошибке? Это происходит потому, что среды Linux чувствительны к регистру. Для перехода в каталог `Downloads` требуется правильно написать название этого каталога (с заглавной буквы `D`):

```
sysadmin@localhost:~$ cd Downloads
sysadmin@localhost:~/Downloads$
```

4.1.3. Текущий каталог

Текущий каталог - это каталог, в котором работает пользователь в данный момент в терминале. Когда впервые открывается терминал, текущий каталог - это домашний каталог пользователя. Текущий каталог можно изменить переходом в другие каталоги командой `cd`.

Находясь в среде командной строки, можно определить текущий каталог с помощью команды `pwd`:

```
sysadmin@localhost:~$ pwd
/home/sysadmin
sysadmin@localhost:~$
```

4.1.4. Изменение каталогов

Для перехода в другой каталог, используется команда `cd` (`change directory` - изменить каталог). Например, следующая команда изменит текущий каталог на каталог с именем `/etc/sound/events`:

```
sysadmin@localhost:~$ cd /etc/sound/events
sysadmin@localhost:/etc/sound/events$
```

При попытке изменить каталог, который не существует, появится сообщение об ошибке:

```
sysadmin@localhost:/etc/sound/events$ cd /etc/junk
-bash: cd: /etc/junk: No such file or directory
sysadmin@localhost:/etc/sound/events$
```

Для возврата в свой домашний каталог, можно либо ввести команду `cd` без аргументов, либо использовать команду `cd` с символом `~` в качестве аргумента:

```
sysadmin@localhost:/etc/sound/events$ cd
sysadmin@localhost:~$ pwd
/home/sysadmin
sysadmin@localhost:~$ cd /etc
sysadmin@localhost:/etc$ cd ~
sysadmin@localhost:~$ pwd
/home/sysadmin
sysadmin@localhost:~$
```

4.1.5. Абсолютные и относительные пути

Путь - это, по сути, описание того, где находится файл или каталог в файловой системе. Можно также считать, что путь - это направление, указывающее системе, где найти файл или каталог. Например, `cd /etc/perl/Net` означает "переход в каталог `Net`, который находится в каталоге `perl`, который находится в каталоге `etc`, который находится в корневом каталоге".

Путь, который начинается с корневого каталога, называется абсолютным путем. Во многих случаях желательно использовать предоставление абсолютного пути. Например, если находясь в своем домашнем каталоге требуется перейти в каталог `/etc/perl/Net`, то имеет смысл предоставления абсолютного пути команде `cd`:

```
sysadmin@localhost:~$ cd /etc/perl/Net
sysadmin@localhost:/etc/perl/Net$
```

Однако, что делать, если пользователь находится в каталоге `/etc/perl` и хочет перейти в каталог `/etc/perl/Net`? Было бы утомительно вводить полный путь, чтобы добраться до каталога, который находится всего на один уровень ниже текущего местоположения. В такой ситуации требуется использовать относительный путь:

```
sysadmin@localhost:/etc/perl$ cd Net
sysadmin@localhost:/etc/perl/Net$
```

Относительный путь в качестве точки отсчета использует текущее местоположение.

Существует удобный метод для относительного пути, который можно использовать для перемещения на один уровень вверх в структуре каталогов: (..). Независимо от того, в каком каталоге находится пользователь, (..) всегда представляет один каталог выше текущего каталога (за исключением случая корневого каталога /):

```
sysadmin@localhost:/etc/perl/Net$ pwd
/etc/perl/Net
sysadmin@localhost:/etc/perl/Net$ cd ..
sysadmin@localhost:/etc/perl$ pwd
/etc/perl
sysadmin@localhost:/etc/perl$
```

Относительные и абсолютные пути используются не только для команды `cd`. Каждый раз, когда указывается файл или каталог, можно использовать относительные или абсолютные пути.

В то время как две точки (..) используются для ссылки на каталог уровнем выше текущего, одиночная точка (.) используется для ссылки на текущий каталог.

4.2. Вывод списка файлов в каталоге

Команда `ls` (сокращенно `list`) может использоваться для отображения содержимого каталога, а также подробной информации о файлах, которые находятся в каталоге.

Сама по себе команда `ls` выведет список файлов в текущем каталоге:

```
sysadmin@localhost:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
sysadmin@localhost:~$
```

4.2.1. Список с выделением цветом типов файлов

В Linux существует множество различных типов файлов. В таблице 5 приведен краткий обзор некоторых наиболее распространенных типов файлов:

Таблица 5

Тип	Назначение
plain file	Обычный файл
directory	Файл каталога (содержит другие файлы)
executable	Файл, который можно запустить как программу
symbolic link	Файл, указывающий на другой файл

Во многих дистрибутивах Linux учетные записи обычных пользователей изменяются таким образом, что команда `ls` отображает имена файлов с цветовой кодировкой по типу файла. Например, каталоги могут быть окрашены в синий цвет, исполняемые файлы будут отображаться зеленым цветом, а символические ссылки голубым (светло-синим).

Это происходит из-за использования параметра `--color` для команды `ls`. Причина, по которой `ls`, автоматически выполняет эту раскраску, заключается в том, что существует псевдоним для команды `ls`, поэтому она выполняется с параметром `--color`:

```
sysadmin@localhost:~$ alias
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
sysadmin@localhost:~$
```

В некоторых случаях не желательно видеть все цвета (иногда они могут немного отвлекать или некорректно отображаться на экране). Чтобы избежать автоматической раскраски, нужно поместить символ обратной косой черты `\` перед командой для игнорирования псевдонима.

4.2.2. Список скрытых файлов

При использовании команды `ls` для отображения содержимого каталога не все файлы отображаются автоматически. По умолчанию команда `ls` не отображает скрытые файлы. Скрытый файл - это любой файл (или каталог), начинающийся с точки.

Чтобы отобразить все файлы, включая скрытые, используется параметр `-a` команды `ls`:

```
sysadmin@localhost:~$ ls -a
.          .bashrc    .selected_editor  Downloads  Public
..         .cache     Desktop          Music      Templates
.bash_logout .profile  Documents        Pictures   Videos
```

Почему эти файлы скрыты? Большинство скрытых файлов - это файлы, предназначенные для настройки работы Linux, оболочки или программ. Например, файл `bashrc` в домашнем каталоге настраивает

функции оболочки, такие как создание или изменение переменных и псевдонимов.

4.2.3. Список каталогов

При использовании команды `ls -d` она ссылается на текущий каталог, а не на содержимое внутри него. Без каких-либо других опций это довольно бессмысленно:

```
sysadmin@localhost:~$ ls -d
.
```

Чтобы использовать команду `ls -d` со смыслом, необходимо добавить параметр `-l`. В примере ниже первая команда перечисляет сведения о содержимом в каталоге `/home/sysadmin`, в то время как вторая команда перечисляет сам каталог `/home/sysadmin`.

```
sysadmin@localhost:~$ ls -l
total 0
drwxr-xr-x 1 sysadmin sysadmin  0 Apr 15  2022 Desktop
drwxr-xr-x 1 sysadmin sysadmin  0 Apr 15  2022 Documents
drwxr-xr-x 1 sysadmin sysadmin  0 Apr 15  2022 Downloads
drwxr-xr-x 1 sysadmin sysadmin  0 Apr 15  2022 Music
drwxr-xr-x 1 sysadmin sysadmin  0 Apr 15  2022 Pictures
drwxr-xr-x 1 sysadmin sysadmin  0 Apr 15  2022 Public
drwxr-xr-x 1 sysadmin sysadmin  0 Apr 15  2022 Templates
drwxr-xr-x 1 sysadmin sysadmin  0 Apr 15  2022 Videos
drwxr-xr-x 1 sysadmin sysadmin 420 Apr 15  2022 test
sysadmin@localhost:~$ ls -ld
drwxr-xr-x 1 sysadmin sysadmin 224 Nov  7 17:07 .
sysadmin@localhost:~$
```

4.2.4. Рекурсивный список

Возможны случаи, когда нужно отобразить все файлы в каталоге, а также все файлы во всех подкаталогах в каталоге. Это называется рекурсивным перечислением.

Для выполнения рекурсивного перечисления используется параметр `-R` команды `ls`:

```
sysadmin@localhost:~$ ls -R /etc/ppp
/etc/ppp:
chap-secrets  ip-down.ipv6to4  ip-up.ipv6to4  ipv6-up  pap-secrets

ip-down      ip-up            ipv6-down      options  peers

/etc/ppp/peers:
sysadmin@localhost:~$
```

В предыдущем примере файлы в каталоге `/etc/ppp` были перечислены первыми. После этого были перечислены файлы в каталоге

/etc/ppp/peers (в этом случае файлов не было, но, если бы какие-либо файлы были в этом каталоге, они были бы отображены).

Нужно быть осторожными с этим параметром. Например, при выполнении команды `ls -R` для корневого каталога `/` будут перечислены все файлы в файловой системе, включая все файлы на любом подключенном USB-устройстве и DVD-диске в системе.

4.2.5. Сортировка списка

По умолчанию команда `ls` сортирует файлы в алфавитном порядке по имени файла. Иногда бывает полезно сортировать файлы по разным критериям.

Для сортировки файлов по размеру можно использовать опцию `-S`.

```
sysadmin@localhost:~$ ls /etc/ssh moduli
ssh_host_dsa_key.pub  ssh_host_rsa_key      sshd_config
ssh_config            ssh_host_ecdsa_key     ssh_host_rsa_key.pub
ssh_host_dsa_key      ssh_host_ecdsa_key.pub ssh_import_id
sysadmin@localhost:~$ ls -S /etc/ssh moduli
ssh_host_dsa_key      ssh_host_ecdsa_key
sshd_config           ssh_host_dsa_key.pub  ssh_host_ecdsa_key.pub
ssh_host_rsa_key      ssh_host_rsa_key.pub
ssh_config            ssh_import_id
sysadmin@localhost:~$
```

В двух командах одни и те же файлы и каталоги перечислены, но в другом порядке.

Параметр `-lS` выведет список файлов от наибольшего к наименьшему и отобразит фактический размер файла.

```
sysadmin@localhost:~$ ls -lS /etc/ssh
total 160
-rw-r--r-- 1 root root 125749 Apr 29  2021 moduli
-rw-r--r-- 1 root root   2489 Jan 29  2022 sshd_config
-rw----- 1 root root   1675 Jan 29  2022 ssh_host_rsa_key
-rw-r--r-- 1 root root   1669 Apr 29  2021 ssh_config
-rw----- 1 root root    668 Jan 29  2022 ssh_host_dsa_key
-rw-r--r-- 1 root root    607 Jan 29  2022 ssh_host_dsa_key.pub
-rw-r--r-- 1 root root    399 Jan 29  2022 ssh_host_rsa_key.pub
-rw-r--r-- 1 root root    302 Jan 10  2020 ssh_import_id
-rw----- 1 root root    227 Jan 29  2022 ssh_host_ecdsa_key
-rw-r--r-- 1 root root    179 Jan 29  2022 ssh_host_ecdsa_key.pub
sysadmin@localhost:~$
```

Также можно сортировать файлы по времени их изменения. Это можно сделать с помощью параметра `-t`. Этот параметр можно использовать отдельно, но, он более полезен в сочетании с параметром `-l`:

```

sysadmin@localhost:~$ ls -tl /etc/ssh
total 160
-rw----- 1 root root 668 Jan 29 2022 ssh_host_dsa_key
-rw-r--r-- 1 root root 607 Jan 29 2022 ssh_host_dsa_key.pub
-rw----- 1 root root 227 Jan 29 2022 ssh_host_ecdsa_key
-rw-r--r-- 1 root root 179 Jan 29 2022 ssh_host_ecdsa_key.pub
-rw----- 1 root root 1675 Jan 29 2022 ssh_host_rsa_key
-rw-r--r-- 1 root root 399 Jan 29 2022 ssh_host_rsa_key.pub
-rw-r--r-- 1 root root 2489 Jan 29 2022 sshd_config
-rw-r--r-- 1 root root 125749 Apr 29 2021 moduli
-rw-r--r-- 1 root root 1669 Apr 29 2021 ssh_config
-rw-r--r-- 1 root root 302 Jan 10 2020 ssh_import_id
sysadmin@localhost:~$

```

Важно помнить, что дата изменения в каталогах представляет время последнего добавления или удаления файла из каталога. Если файлы в каталоге были изменены много дней или месяцев назад, может быть сложно определить, когда они были изменены, так как для более старых файлов указана только дата. Для получения более подробной информации о времени модификации можно использовать опцию `--full-time` для отображения полной метки времени (включая часы, секунды, минуты...). Параметр `--full-time` принимает параметр `-l` автоматически.

```

sysadmin@localhost:~$ ls -t --full-time /etc/ssh
total 160
-rw----- 1 root root 668 2022-01-29 03:17:33.000000000 +0000 ssh_host_dsa_key
-rw-r--r-- 1 root root 607 2022-01-29 03:17:33.000000000 +0000 ssh_host_dsa_key.pub
-rw----- 1 root root 227 2022-01-29 03:17:33.000000000 +0000 ssh_host_ecdsa_key
-rw-r--r-- 1 root root 179 2022-01-29 03:17:33.000000000 +0000 ssh_host_ec-
dsa_key.pub
-rw----- 1 root root 1675 2022-01-29 03:17:33.000000000 +0000 ssh_host_rsa_key
-rw-r--r-- 1 root root 399 2022-01-29 03:17:33.000000000 +0000 ssh_host_rsa_key.pub
-rw-r--r-- 1 root root 2489 2022-01-29 03:17:33.000000000 +0000 sshd_config
-rw-r--r-- 1 root root 125749 2021-04-29 23:58:51.000000000 +0000 moduli
-rw-r--r-- 1 root root 1669 2021-04-29 23:58:51.000000000 +0000 ssh_config
-rw-r--r-- 1 root root 302 2020-01-10 18:48:29.000000000 +0000 ssh_import_id
sysadmin@localhost:~$

```

4.3. Операции с файлами и каталогами

Команда `cp` (сокращение от `copy`) используется для копирования файлов. Необходимо указать источник и место назначения. Источник - файл, который требуется скопировать. Место назначения - это место, где должна находиться копия. Следующая команда копирует файл `/etc/hosts` в домашний каталог:

```

sysadmin@localhost:~$ cp /etc/hosts ~
sysadmin@localhost:~$ ls
Desktop  Downloads  Pictures  Templates  hosts
Documents  Music      Public    Videos
sysadmin@localhost:~$

```

4.3.1. Подробный режим копирования

Опция `-v` заставит команду `cp` сообщить об успешном выполнении:

```
sysadmin@localhost:~$ cp -v /etc/hosts ~
`/etc/hosts' -> `/home/sysadmin/hosts'
sysadmin@localhost:~$
```

Если местом назначения является каталог, то полученный новый файл будет иметь то же имя, что и исходный файл. Если требуется, чтобы новый файл имел другое имя, необходимо указать новое имя:

```
sysadmin@localhost:~$ cp /etc/hosts ~/hosts.copy
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures   Templates  hosts
Documents  Music      Public     Videos     hosts.copy
sysadmin@localhost:~$
```

4.3.2. Как избежать перезаписи данных при копировании

Команда `cp` может быть разрушительной для существующих данных, если файл назначения уже существует. Если файл назначения существует, команда `cp` перезапишет содержимое существующего файла содержимым исходного файла.

Есть два варианта, которые могут быть использованы для защиты от случайной перезаписи. При использовании команды `cp` с опцией `-i` она запросит разрешение перед перезаписью файла. Если нет желания автоматически отвечать на каждый запрос, используется опция `-n`. По сути, это означает "не переписывать".

Для перемещения файла используется команда `mv` (сокращение от `move`). В следующем примере файл `hosts`, созданный ранее, перемещается из текущего каталога в каталог `Videos`:

```
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures   Templates  example.txt  hosts.copy
Documents  Music      Public     Videos     hosts
sysadmin@localhost:~$ mv hosts Videos
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures   Templates  example.txt
Documents  Music      Public     Videos     hosts.copy
sysadmin@localhost:~$ ls Videos
hosts
sysadmin@localhost:~$
```

При перемещении файла он удаляется из исходного расположения и помещается в новое. При этом могут возникнуть трудности, по-

тому что пользователям нужны определенные разрешения для удаления файлов из каталога. Если у пользователя нет необходимых разрешений, появится сообщение об ошибке "отказано в разрешении" (Permission denied):

```
sysadmin@localhost:~$ mv /etc/hosts .
mv: cannot move `/etc/hosts' to `./hosts': Permission denied
sysadmin@localhost:~$
```

Команда `mv` используется не только для перемещения файла, но и для переименования файла. Например, следующие команды переименуют `newexample.txt` файл в `myexample.txt`:

```
sysadmin@localhost:~$ cd Videos
sysadmin@localhost:~/Videos$ ls
hosts  newexample.txt
sysadmin@localhost:~/Videos$ mv newexample.txt myexample.txt
sysadmin@localhost:~/Videos$ ls
hosts  myexample.txt
sysadmin@localhost:~/Videos$
```

Для удаления файла используется команда `rm`:

```
sysadmin@localhost:~$ ls
Desktop  Downloads  Pictures  Templates  sample
Documents Music      Public   Videos
sysadmin@localhost:~$ rm sample
sysadmin@localhost:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
sysadmin@localhost:~$
```

В примере файл был удален без каких-либо подтверждений. Это может вызвать проблемы при удалении нескольких файлов с помощью символов раскрытия, например, `rm *.txt`. Поскольку эти файлы удаляются без подтверждения, пользователь может в конечном итоге удалить файлы, которые не были предназначены для удаления.

Кроме того, файлы удаляются безвозвратно. В качестве меры предосторожности пользователи должны использовать опцию `-i` при удалении нескольких файлов: `rm -i *.txt`

Можно удалять каталоги с помощью команды `rm`. Однако использование команды `rm` по умолчанию (без параметров) не удалит каталог:

```
sysadmin@localhost:~$ rm Videos
rm: cannot remove `Videos': Is a directory
sysadmin@localhost:~$
```

Если требуется удалить каталог, используется опция `-r` для команды `rm`: `rm -r Videos`

Важное замечание: когда пользователь удаляет каталог, все файлы и подкаталоги удаляются без каких-либо интерактивных вопросов. Лучше всего использовать опцию `-i` с командой `rm`.

Можно также удалить каталог с помощью команды `rmdir`, но только если каталог пуст.

Для создания каталога, используется команда `mkdir`:

```
sysadmin@localhost:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates
sample.txt
sysadmin@localhost:~$ mkdir test
sysadmin@localhost:~$ ls
Desktop  Downloads  Pictures  Templates  test
Documents  Music      Public    sample.txt
sysadmin@localhost:~$
```

4.4. Жесткие и относительные ссылки

Рассмотрим следующий сценарий: файл, глубоко зарытый в файловой системе, называется `/usr/share/doc/superpuperpackage/data/2021/someversion/valuable-information.txt`. Предположим, этот файл регулярно обновляется пользователем и необходим для доступа. Длинное имя файла неудобно для ввода, но файл должен находиться в этом месте и, поскольку он обновляется на регулярной основе, нельзя просто сделать копию файла.

Это такая ситуация, когда пригождаются жесткие и символические (мягкие) ссылки. Можно создать файл, который будет связан с тем, который имеет длинный путь. Этот новый файл можно поместить в домашний каталог или в любое другое удобное место. При доступе к новому файлу «ссылке» пользователь получит доступ к содержимому исходного файла.

Каждый метод связывания, жесткий и символический, обеспечивает одинаковый общий доступ, но использует разные методы. Для каждого метода существуют плюсы и минусы, поэтому важно знать и методы, и когда их использовать.

Чтобы понять жесткие ссылки, полезно понять немного о том, как файловая система отслеживает файлы. Для каждого созданного файла в файловой системе находится блок данных, в котором хранится метainформация файла. Метаинформация включает информацию о файле,

такую как разрешения, владельцы и временные метки. Метаинформация не включает имя файла или содержимое файла, но она включает практически всю информацию о файле.

Эта метаинформация называется таблицей inode. Таблица inode также включает указатели на другие блоки в файловой системе, называемые блоками данных, где хранятся данные.

Каждый файл имеет уникальный идентификационный номер в дисковом разделе, называемый номером inode. Команда `ls -i` отображает номер inode файла. Как и в случае с пользователями и группами, то, что действительно определяет файл, - это не его имя, а номер, который ему был назначен.

Таблица inode не включает имя файла. Для каждого файла есть запись, которая хранится в области данных каталога (блок данных), включающая в себя связь между номером inode и именем файла. Когда пользователь пытается получить доступ к файлу, система использует эту таблицу для преобразования имени файла в номер inode. Затем извлекаются данные файла, просматривая информацию для файла в таблице inode.

Если два файла имеют одинаковый номер inode, они по существу являются одним и тем же файлом. Можно получить доступ к данным файла, используя любое имя файла.

Когда пользователь выполняет команду `ls -li`, число, которое появляется для каждого файла между разрешениями и владельцем, является числом подсчета ссылок:

```
sysadmin@localhost:~$ echo data > file.original
sysadmin@localhost:~$ ls -li file.*
278772 -rw-rw-r--. 1 sysadmin sysadmin 5 Oct 25 15:42 file.original
```

Номер подсчета ссылок указывает, сколько жестких ссылок было создано. Когда число равно единице, тогда файл имеет только одно имя, связанное с inode.

Для создания жестких ссылок используется команда `ln`. Первый аргумент - существующее имя файла, а второй аргумент - новое имя файла. Когда используется команда `ln` для создания жесткой ссылки, число ссылок будет увеличиваться на единицу для каждого нового имени файла:

```
sysadmin@localhost:~$ ln file.original file.hard.1
sysadmin@localhost:~$ ls -li file.*
278772 -rw-rw-r--. 2 sysadmin sysadmin 5 Oct 25 15:53 file.hard.1
278772 -rw-rw-r--. 2 sysadmin sysadmin 5 Oct 25 15:53 file.original
```

Символическая ссылка - это просто файл, указывающий на другой файл. В системе уже есть несколько символических ссылок, в том числе несколько в каталоге /etc:

```
sysadmin@localhost:~$ ls -l /etc/grub.conf
lrwxrwxrwx. 1 root root 22 Feb 15 2020 /etc/grub.conf -> ../boot/grub/grub.conf
```

В приведенном примере можно увидеть, что файл /etc/grub.conf «указывает на» файл ../boot/grub/grub.conf. Если попытаться просмотреть содержимое файла /etc/grub.conf, оно будет следовать указателю и покажет содержимое файла ../boot/grub/grub.conf.

Чтобы создать символическую ссылку, используется параметр -s с командой ln:

```
sysadmin@localhost:~$ ln -s /etc/passwd mypasswd
sysadmin@localhost:~$ ls -l mypasswd
lrwxrwxrwx. 1 sysadmin sysadmin 11 Oct 31 13:17 mypasswd -> /etc/passwd
```

Хотя жесткие и символические ссылки имеют один и тот же конечный результат, разные методы создают разные преимущества и недостатки. Фактически преимущество одного метода компенсирует недостаток другой техники. Преимущество жестких ссылок - нет единой точки отказа. Одним из преимуществ использования жестких ссылок является то, что каждое имя файла для содержимого файла эквивалентно. Если есть пять файлов, которые связаны друг с другом, то удаление любых четырех из этих файлов не приведет к удалению фактического содержимого файла. Файл связан с уникальным номером inode. Пока один из файлов с жесткой привязкой остается, то этот номер inode все еще существует. Это означает, что данные файла все еще существуют.

Символические ссылки, имеют одну точку отказа: исходный файл. Рассмотрим следующий пример, в котором доступ к данным не удастся, если исходный файл будет удален:

```
sysadmin@localhost:~$ echo "привет" > test.txt
sysadmin@localhost:~$ ln -s test.txt mytest.txt
sysadmin@localhost:~$ more test.txt
привет
sysadmin@localhost:~$ more mytest.txt
привет
sysadmin@localhost:~$ rm test.txt
sysadmin@localhost:~$ more mytest.txt
mytest.txt: No such file or directory
sysadmin@localhost:~$ ls -l mytest.txt
lrwxrwxrwx. 1 sysadmin sysadmin 8 Oct 31 13:29 mytest.txt -> test.txt
```

Иногда бывает трудно узнать, где существуют жесткие ссылки на файл. Если пользователь видит обычный файл со списком ссылок, который больше одного, он может использовать команду `find` с критериями поиска `-inum` для поиска других файлов с одним и тем же номером `inode`. Чтобы найти номер `inode`, можно сначала использовать команду `ls -li`:

```
sysadmin@localhost:~$ ls -li file.original
278772 file.original
sysadmin@localhost:~$ find / -inum 278772 2> /dev/null
/home/sysadmin/file.hard.1
/home/sysadmin/file.original
```

Мягкие ссылки гораздо более наглядны, не требуя дополнительных команд за пределами команды `ls` для определения ссылки:

```
sysadmin@localhost:~$ ls -li mypasswd
lrwxrwxrwx. 1 sysadmin sysadmin 11 Oct 31 13:17 mypasswd -> /etc/passwd
```

Поскольку каждая файловая система (раздел) имеет отдельный набор индексов, невозможно создать жесткие ссылки ведущие на разные разделы:

```
sysadmin@localhost:~$ ln /boot/vmlinuz-2.6.32-358.6.1.el6.i686 Linux.Kernel
ln: creating hard link `Linux.Kernel' => `/boot/vmlinuz-2.6.32-358.6.1.el6.i686': Invalid cross-device link
```

В предыдущем примере была создана жесткая ссылка между файлом в файловой системе `/boot` и корневой файловой системой `/`. Команда не удалась, потому что каждая из этих файловых систем имеет уникальный набор номеров индексных дескрипторов, которые нельзя использовать вне файловой системы.

Однако, поскольку символическая ссылка указывает на другой файл с использованием имени пути, можно создать мягкую ссылку на файл в другой файловой системе:

```
sysadmin@localhost:~$ ln -s /boot/vmlinuz-2.6.32-358.6.1.el6.i686 Linux.Kernel
sysadmin@localhost:~$ ls -li Linux.Kernel
lrwxrwxrwx. 1 sysadmin sysadmin 11 Oct 31 13:17 Linux.Kernel -> /boot/vmlinuz-2.6.32-358.6.1.el6.i686
```

Другим ограничением жестких ссылок является то, что они не могут быть созданы для каталогов. Причина этого ограничения заключается в том, что сама операционная система использует жесткие ссылки для определения иерархии структуры каталогов. В следующем примере показано сообщение об ошибке, которое отображается при попытке создания жесткой ссылки на каталог:

```
sysadmin@localhost:~$ ln /bin binary
ln: `/bin': hard link not allowed for directory
```

Допускается ссылка на каталоги с использованием символической ссылки:

```
sysadmin@localhost:~$ ln -s /bin binary
sysadmin@localhost:~$ ls -l binary
lrwxrwxrwx. 1 sysadmin sysadmin 11 Oct 31 13:17 binary -> /bin
```

4.5. Сжатие файлов

Архивация файлов используется, когда один или несколько файлов должны быть переданы или сохранены как можно более эффективно. Существует два подхода:

- Архивирование - объединение нескольких файлов в один, что устраняет издержки в отдельных файлах и облегчает их передачу.
- Сжатие - уменьшение размера файлов путем удаления избыточной информации.

Можно архивировать несколько файлов в один архив, а затем сжимать его, или можно сжать отдельный файл. Первое называется архивированием, а второе - сжатием. Когда берется архив, распаковывается и извлекается один или несколько файлов, речь идет о разархивировании.

Несмотря на то, что дисковое пространство относительно дешевое, архивация и сжатие по-прежнему имеют значение:

- Если нужно сделать доступным в сети большое количество файлов, таких как исходный код приложения или коллекции документов, людям легче загрузить сжатый архив, чем загружать отдельные файлы.
- Файлы журналов имеют привычку заполнять диски, поэтому их полезно разбивать по дате и сжимать старые версии.
- При резервном копировании каталогов проще хранить их все в одном архиве, чем в версии каждого файла.
- Часто бывает быстрее сжать файл перед отправкой на накопитель или по медленной сети и распаковать его на другом конце, чем отправить его несжатым.

Сжатие файлов делает их меньше, удаляя дублирование из файла и сохраняя его таким образом, что файл может быть полностью восста-

новлен. Файл с удобочитаемым текстом может иметь часто используемые слова, замененные чем-то меньшим, или изображение со сплошным фоном может представлять пятна этого цвета с помощью кода. Как правило, сжатая версия файла не используется напрямую, а распаковывается перед использованием. Алгоритм сжатия - это процедура, которую выполняет компьютер для кодирования исходного файла, и в результате делает его меньше.

Если говорить о сжатии, существует два типа:

- без потерь: информация не удаляется из файла. Сжатие файла и распаковка оставляет результат идентичный оригиналу.
- с потерями: информация может быть удалена из файла при сжатии, так что распаковка файла приведет к файлу, который немного отличается от оригинала. Например, изображение с двумя тонко различными оттенками зеленого может быть уменьшено, рассматривая эти два оттенка как одинаковые.

Как правило, человеческие глаза и уши не замечают незначительных недостатков в изображениях и аудио, особенно когда они отображаются на мониторе или воспроизводятся на динамиках. Сжатие с потерями часто приносит пользу, потому что это приводит к меньшим размерам файлов, и люди не могут определить разницу между оригиналом и версией с измененными данными. Для данных, которые должны оставаться нетронутыми, таких как документы, журналы и программное обеспечение, требуется сжатие без потерь.

Большинство форматов изображений, таких как GIF, PNG и JPEG, реализуют сжатие с потерями. Можно выбрать какого качества изображение требуется сохранить. Более низкое качество приводит к меньшему файлу, но после декомпрессии пользователь может заметить артефакты, такие как грубые края или обесцвечивание. Высокое качество будет очень похоже на исходное изображение, но размер файла будет ближе к оригиналу.

Сжатие уже сжатого файла не сделает его меньше. Это часто забывается, когда дело доходит до изображений, так как они уже хранятся в сжатом формате. При сжатии без потерь это многократное сжатие не является проблемой, но, если файл сжимается и распаковывается несколько раз, используя алгоритм с потерями, в конечном итоге можно получить что-то неузнаваемое.

Linux предоставляет несколько инструментов для сжатия файлов, наиболее распространенным является `gzip`. Вот как выглядит файл журнала до и после сжатия:

```
bob:tmp $ ls -l access_log*
-rw-r--r-- 1 sean sean 372063 Oct 11 21:24 access_log
bob:tmp $ gzip access_log
bob:tmp $ ls -l access_log*
-rw-r--r-- 1 sean sean 26080 Oct 11 21:24 access_log.gz
```

В приведенном выше примере существует файл с именем `access_log`, который составляет 372 063 байта. Файл сжимается при вызове команды `gzip` с именем файла в качестве единственного аргумента. После выполнения этой команды исходный файл исчез и сжатая версия с расширением `.gz` остается на его месте. Размер файла теперь составляет 26 080 байт, что дает коэффициент сжатия около 14:1.

Можно узнать коэффициент сжатия, используя параметр `-l`:

```
bob:tmp $ gzip -l access_log.gz
      compressed      uncompressed   ratio uncompressed_name
          26080             372063   93.0% access_log
```

Здесь можно видеть, что коэффициент сжатия задан как 93%, что является обратным соотношению 14:1, т. е. 13/14. Кроме того, когда файл распаковывается, он будет называться `access_log`.

Противоположная `gzip` команда `gunzip`. В качестве альтернативы можно использовать `gzip -d`. После того, как `gunzip` делает свою работу, файл `access_log` возвращается к своему первоначальному размеру:

```
bob:tmp $ gunzip access_log.gz
bob:tmp $ ls -l access_log*
-rw-r--r-- 1 sean sean 372063 Oct 11 21:24 access_log
```

`Gzip` также может выступать в качестве фильтра, который означает, что он не читает и не записывает что-либо на диск, а получает данные через входной канал и записывает их в выходной канал.

```
bob:tmp $ mysqldump -A | gzip > database_backup.gz
bob:tmp $ gzip -l database_backup.gz
      compressed      uncompressed   ratio uncompressed_name
          76866             1028003   92.5% database_backup
```

Команда `mysqldump -a` выводит содержимое локальных баз данных MySQL в консоль. Символ `|` говорит: "перенаправьте выходные данные предыдущей команды на вход следующей". Программой для получения выходных данных является `gzip`, который распознает,

что имена файлов не были даны, поэтому он должен работать в режиме канала. Наконец, `> database_backup.gz` означает "перенаправить выходные данные предыдущей команды в файл с именем `database_backup.gz`. Проверка этого файла командой `gzip -l` показывает, что сжатая версия составляет 7,5% от размера оригинала.

Есть еще пара команд, которые работают практически одинаково с `gzip` и `gunzip`. Это `bzip2` и `bunzip2`. Утилиты `bzip` используют другой алгоритм сжатия (называемый сортировкой блоков Burrows-Wheeler, в отличие от кодирования Lempel-Ziv, используемого `gzip`), который может сжимать файлы сильнее, чем `gzip`, за счет большего времени процессора. Можно распознать эти файлы по расширению `bz` или `bz2` вместо `.gz`.

4.6. Архивирование файлов

Если есть несколько файлов для отправки кому-то, можно сжать каждый по отдельности. Будет меньший объем данных, чем при отправке несжатых файлов, но все равно придется иметь дело с несколькими файлами одновременно.

Архивирование является решением этой проблемы. Традиционная утилита Unix для архивации файлов называется `tar` (сокращение `tape archive` - архив на магнитной ленте). `Tar` использовался для потоковой передачи большого количества файлов на ленту для резервного копирования или передачи файлов. `Tar` принимает несколько файлов и создает один выходной файл, который можно снова разделить на исходные файлы на другом конце передачи.

`Tar` имеет 3 режима:

- Создание: создать новый архив из серии файлов
- Извлечение: извлечь один или несколько файлов из архива
- Список: показать содержимое архива без извлечения

Вот как выглядит файл `tar`, также называемый TAR-архивом, который создается из нескольких журналов доступа:

```
bob:tmp $ tar -cf access_logs.tar access_log*
bob:tmp $ ls -l access_logs.tar
-rw-rw-r-- 1 sean sean 542720 Oct 12 21:42 access_logs.tar
```

Для создания архива требуются два именованных параметра. Первый, `-c`, определяет режим. Вторым, `-f`, говорит `tar` ожидать имя

файла в качестве следующего аргумента. Первый аргумент в приведенном выше примере создает архив с именем `access_logs.tar`. Остальные аргументы воспринимаются как имена входных файлов, либо как символ раскрытия. В данном примере используется символ раскрытия для включения всех файлов, которые начинаются с `access_log`.

В приведенном выше примере приводится отображение параметров созданного файла. Окончательный размер составляет 542720 байт, что немного больше, чем входные файлы. TAR-файлы могут быть сжаты. Для этого используется опция `-z`:

```
bob:tmp $ tar -czf access_logs.tar.gz access_log*
bob:tmp $ ls -l access_logs.tar.gz
-rw-rw-r-- 1 sean sean 46229 Oct 12 21:50 access_logs.tar.gz
bob:tmp $ gzip -l access_logs.tar.gz
```

compressed	uncompressed	ratio	uncompressed_name
46229	542720	91.5%	access_logs.tar

В приведенном выше примере показана та же команда, что и в предыдущем примере, но с добавлением параметра `z`. Выходные данные гораздо меньше, чем сам архив, и полученный файл совместим с `gzip`.

В то время как UNIX не обрабатывает расширения файлов, существует соглашение о том, что необходимо использовать расширение `.tar` для файлов `tar`, и `.tar.gz` или `.tgz` для сжатых файлов `tar`. Вместо `gzip` можно использовать `bzip2`, используя опцию `j` вместо опции `z`. Тогда получаются расширения файлов `.tar.bz2`, `.tbz`, или `.tbz2`. Можно увидеть содержимое `tar`-файла, используя опцию `t`:

```
bob:tmp $ tar -tjf access_logs.tbz
logs/
logs/access_log.3
logs/access_log.1
logs/access_log.4
logs/access_log
logs/access_log.2
```

В этом примере используются 3 опции:

- `t`: список файлов в архиве
 - `j`: распаковать с помощью `bzip2`, прежде чем читать
 - `f`: работать с заданным именем файла `access_log.tbz`
- Отображается содержимое сжатого архива

4.7. ZIP-файлы

Де-факто основным типом для архивирования и сжатия в операционных системах Microsoft Windows является ZIP-файл. Он не так распространен в Linux, но хорошо поддерживается командами `zip` и `unzip`.

Режим `zip` по умолчанию - добавление файлов в архив и сжатие.

```
bob:tmp $ zip logs.zip logs/*
adding: logs/access_log (deflated 93%)
adding: logs/access_log.1 (deflated 62%)
adding: logs/access_log.2 (deflated 88%)
adding: logs/access_log.3 (deflated 73%)
adding: logs/access_log.4 (deflated 72%)
```

Первым аргументом в приведенном выше примере является имя архива, с которым необходимо работать, в данном случае это `logs.zip`. После этого идёт список файлов, которые будут добавлены. Выходные данные показывают файлы и коэффициент сжатия.

`Zip` не будет рекурсивно обрабатывать подкаталоги по умолчанию, что отличается от поведения `tar`. То есть, просто добавление `logs` вместо `logs/*` добавит пустой каталог. Если нужно рекурсивно обработать вложенные подкаталоги, необходимо использовать параметр `-r`:

```
bob:tmp $ zip -r logs.zip logs
adding: logs/ (stored 0%)
adding: logs/access_log.3 (deflated 73%)
adding: logs/access_log.1 (deflated 62%)
adding: logs/access_log.4 (deflated 72%)
adding: logs/access_log (deflated 93%)
adding: logs/access_log.2 (deflated 88%)
```

В приведенном выше примере добавляются все файлы в каталоге `logs`, поскольку используется параметр `-r`. Первая строка выходных данных указывает, что каталог был добавлен в архив, в остальном выходные данные аналогичны предыдущему примеру.

Перечисление файлов в `zip` выполняется командой `unzip` и опцией `-l`:

```
bob:tmp $ unzip -l logs.zip
Archive:  logs.zip
  Length      Date    Time    Name
-----
      0  10-14-2020  14:07   logs/
  1136  10-14-2020  14:07 logs/access_log.3
   362  10-14-2020  14:07 logs/access_log.1
   784  10-14-2020  14:07 logs/access_log.4
```

90703	10-14-2020	14:07	logs/access_log
153813	10-14-2020	14:07	logs/access_log.2
-----			-----
246798			6 files

Справочные страницы `zip` и `unzip` описывают другие действия, которые можно выполнить с помощью этих инструментов, например, заменить файлы в архиве, использовать различные уровни сжатия и даже использовать шифрование.

4.8. Стандарт иерархии файловой системы

Среди стандартов, поддерживаемых Linux Foundation, есть стандарт иерархии файловой системы (Filesystem Hierarchy Standard, FHS), текст которого размещается по адресу <http://www.pathname.com/fhs/>.

Стандартом является набор правил или рекомендаций, которые рекомендуется соблюдать. Тем не менее, эти рекомендации могут быть нарушены либо целыми дистрибутивами, либо администраторами на отдельных машинах.

Стандарт FHS классифицирует каждый системный каталог несколькими способами:

- Каталог можно классифицировать как разделяемый или нет, то есть, может ли быть предоставлен доступ к каталогу из сети для использования несколькими машинами.
- Каталог помещается в категорию наличия либо статических файлов (содержимое файла не изменяется), либо файлов переменных (содержимое файла может измениться).

Чтобы придерживаться этих классификаций, часто приходится ссылаться на поддиректории ниже верхнего уровня каталогов. Например, каталог `/var` нельзя классифицировать как разделяемый или не разделяемый, но один из его подкаталогов, каталог `/var/mail`, является разделяемым. И наоборот, каталог `/var/lock` не должен быть разделяемым.

Стандарт FHS определяет четыре иерархии каталогов, используемых при организации файловой системы. В таблице 6 перечислена иерархия верхнего уровня или корня.

Таблица 6

Директо- рия	Назначение директории
/	База структуры или корень файловой системы, этот каталог объединяет все каталоги, независимо от того, являются ли они локальными разделами, съемными устройствами или сетевыми ресурсами
/bin	Для хранения необходимых бинарных исполняемых файлов, таких как команды ls, cp и rm, и быть частью корневой файловой системы.
/boot	Сохраняет файлы, необходимые для загрузки системы, такие как ядро Linux и связанные с ним файлы конфигурации
/dev	Заполняется файлами, которые представляют собой аппаратные устройства и другие специальные файлы, такие как файлы /dev/null и /dev/zero
/etc	Содержит файлы основных конфигураций хоста, такие как файлы /etc/hosts или /etc/passwd
/home	Расположение домашних каталогов пользователей
/lib	Основные библиотеки для поддержки исполняемых файлов в каталогах /bin и /sbin
/lib<qual>	Основные библиотеки, созданные для конкретной архитектуры. Например, каталог /lib64 для 64-разрядных процессоров AMD / Intel x86
/media	Точка монтирования для съемных носителей, установленных автоматически
/mnt	Точка монтирования для временного монтирования файловых систем вручную
/opt	Дополнительное место установки программного обеспечения стороннего производителя
/proc	Виртуальная файловая система для ядра, чтобы сообщить о процессах и другой информации
/root	Домашний каталог пользователя root
/sbin	Существенные системные исполняемые файлы, используемые пользователем root
/sys	Виртуальная файловая система, содержащая информацию об аппаратных устройствах, подключенных к системе
/srv	Местоположение, на котором могут размещаться сайты
/tmp	Каталог, где всем пользователям разрешено создавать временные файлы, которые должны быть очищены во время загрузки (но часто это не так)
/usr	Вторая иерархия несущественных файлов для многопользовательского использования.

/usr/local	Третья иерархия файлов для программного обеспечения, не входящих в дистрибутив
/var	Иерархия /var содержит файлы, которые меняются со временем
/var/cache	Файлы, используемые для кэширования данных приложений
/var/log	Каталог, в котором хранится большинство файлов журнала
/var/lock	Хранятся файлы блокировки для общих ресурсов
/var/spool	Хранятся файлы пула для печати и почты
/var/tmp	Временные файлы, которые необходимо сохранить между перезагрузками

Вторая и третья иерархии, расположенные в каталогах /usr и /usr/local, повторяют шаблон многих ключевых каталогов, найденных в первой иерархии или корневой файловой системе. Четвертая иерархия, каталог /var, также повторяет некоторые из каталогов верхнего уровня, таких как lib, opt и tmp.

Если корневая файловая система и ее содержимое считаются необходимыми для загрузки системы, каталоги /var, /usr и /usr/local считаются несущественными для процесса загрузки. В результате корневая файловая система и ее каталоги могут быть единственными, доступными в определенных ситуациях, таких как загрузка в однопользовательский режим, среда, предназначенная для устранения неполадок системы.

Каталог /usr предназначен для хранения программного обеспечения для использования несколькими пользователями. Каталог /usr иногда разделяется по сети и монтируется только для чтения. Общие каталоги второго уровня описаны в таблице 7.

Таблица 7

Директория	Назначение директории
/usr/bin	Бинарные исполняемые файлы для обычного пользователя, используемые, когда система находится в многопользовательском режиме
/usr/include	Файлы, которые должны быть включены для компиляции программного обеспечения из дистрибутива
/usr/lib	Библиотеки для поддержки исполняемых файлов в каталогах /usr/bin и /usr/sbin
/usr/lib<qual>	Нестандартные библиотеки, построенные для конкретной архитектуры

/usr/libexec	Исполняемые программы, которые будут использоваться другими программами, а не непосредственно пользователями
/usr/sbin	Системные двоичные исполняемые файлы для использования администратором в многопользовательском режиме
/usr/share	Хранится документация по программному обеспечению и другие данные приложений
/usr/src	Исходный код для компиляции ядра

Иерархия /usr/local предназначена для установки программного обеспечения, которое не связано с дистрибутивом. Часто этот каталог используется для программного обеспечения, которое скомпилировано из исходного кода. Общие каталоги третьего уровня, найденные в каталоге /usr/local, описаны в таблице 8.

Таблица 8

Директория	Назначение директории
/usr/local/bin	Локальные программные двоичные файлы для обычного использования пользователями
/usr/local/etc	Локальные файлы конфигурации программного обеспечения
/usr/local/include	Файлы, которые необходимо включить для компиляции локального исходного кода
/usr/local/lib	Библиотечные файлы для поддержки исполняемых файлов в каталогах /usr/local/bin и /usr/local/sbin
/usr/local/libexec	Локальные исполняемые программы, которые будут использоваться другими программами, а не напрямую пользователями
/usr/local/sbin	Локальные двоичные исполняемые файлы для использования системным администратором
/usr/local/share	Хранятся локальные страницы программного обеспечения, информационные страницы и другая информация о локальных приложениях
/usr/local/src	Часто размещается исходный код для компиляции локального программного обеспечения

Хотя стандарт FHS полезен для детального понимания компоновки каталогов, используемых большинством дистрибутивов Linux, ниже приведено более обобщенное описание компоновки каталогов, как они на самом деле существуют в типичном дистрибутиве Linux.

Домашние каталоги пользователей. В каталоге /home обычно есть каталог для каждой учетной записи пользователя. Например,

пользователь Alex будет иметь домашний каталог /home/alex. Как правило, доступ к этой директории будет иметь только пользователь Alex. Не назначая специальные разрешения для других каталогов, пользователь может создавать файлы только в своем домашнем каталоге, каталоге /tmp и /var/tmp.

Каталоги двоичных исполняемых файлов. Каталоги двоичных исполняемых файлов содержат программы, которые запускают пользователи и администраторы. Каталоги, предназначенные для использования непривилегированными пользователями, включают /bin, /usr/bin и /usr/local/bin. Иногда стороннее программное обеспечение также сохраняет свои исполняемые файлы в каталогах, таких как /usr/local/application/bin и /opt/application/bin. Кроме того, для каждого пользователя нет ничего необычного, чтобы иметь собственный каталог bin, расположенный в домашнем каталоге, например /home/alex/bin.

Каталоги/sbin в основном предназначены для использования системным администратором (пользователем root). Обычно они включают каталоги /sbin, /usr/sbin и /usr/local/sbin, хотя сторонние административные приложения также могут использовать каталоги, такие как /usr/local/application/sbin или /opt/application/sbin.

Переменная PATH может не содержать всех каталогов bin и/sbin. Чтобы иметь возможность выполнять команду в одном из этих каталогов, каталог должен быть включен в список переменных PATH или пользователю необходимо указать полный путь к команде, например: /sbin/ifconfig.

Директории программного обеспечения. В отличие от операционной системы Windows, где приложения могут иметь все свои файлы, установленные в подкаталоге каталога C:\Program Files, приложения в Linux могут иметь свои файлы в нескольких каталогах, расположенных по всей файловой системе Linux. Для дистрибутивов, полученных из Debian, чтобы получить список местоположений файлов можно выполнить команду dpkg -L. В дистрибутивах Red Hat аналогичная команда rpm -ql.

Бинарные файлы исполняемой программы могут находиться в каталоге /usr/bin, если они включены в операционную систему, или они могут находиться в каталогах /usr/local/bin или /opt/application/bin, если программа установлена отдельно.

Данные для приложения могут храниться в одном из следующих подкаталогов: `/usr/share`, `/usr/lib`, `/opt/application` или `/var/lib`.

Файлы, связанные с документацией, могут храниться в одном из следующих подкаталогов: `/usr/share/doc`, `/usr/share/man` или `/usr/share/info`.

Глобальные файлы конфигурации для приложения, скорее всего, будут храниться в подкаталоге каталога `/etc`, в то время как персонализированные файлы конфигурации приложения (определенные для пользователя), вероятно, находятся в скрытом подкаталоге домашнего каталога пользователя.

Директории библиотек. Библиотеки - это файлы, содержащие код, который используется совместно несколькими программами. В большинстве случаев имена файлов библиотеки заканчиваются расширением файла `.so`, что означает общий объект.

Может присутствовать несколько версий библиотеки. При этом код может быть различным в каждом файле, даже если он может выполнять аналогичные функции. Одна из причин того, что код может быть разным, заключается в том, что он скомпилирован для работы на процессорах разного типа. Например, это типично для систем, которые используют код, предназначенный для 64-разрядных процессоров Intel / AMD, для 32-разрядных библиотек и 64-разрядных библиотек.

Библиотеки, которые поддерживают основные двоичные программы, найденные в каталогах `/bin` и `/sbin`, обычно располагаются в `/lib` или `/lib64`.

Для поддержки исполняемых файлов `/usr/bin` и `/usr/sbin` обычно используются каталоги библиотек `/usr/lib` и `/usr/lib64`.

Для поддержки приложений, не распространяемых с операционной системой, часто используются каталоги библиотеки `/usr/local/lib` и `/opt/application/lib`.

Директории данных. Каталог `/var` и многие его подкаталоги могут содержать данные, которые будут часто меняться. Если система используется для электронной почты, то для хранения пользовательских данных электронной почты обычно используется `/var/mail` или `/var/spool/mail`. Если система используется для печати на принтере, каталог `/var/spool/cups` используется для временного хранения заданий печати.

В зависимости от того, какие события регистрирует система и сколько активности в системе будет определять, настолько большой будет файл журнала. В загруженной системе в файлах журналов может быть большой объем данных. Эти файлы хранятся в каталоге `/var/log`.

Хотя файлы журналов могут быть чрезвычайно полезны для устранения неполадок, они могут вызвать проблемы. Одна из основных проблем для таких каталогов заключается в том, что в процессе активной работы всей системы может быть быстро заполнено дисковое пространство. Для решения этой проблемы каталог `/var` часто выносят в отдельный раздел, в том числе для подключения отдельного устройства хранения.

Контрольные вопросы

1. Какая команда позволит изменить текущий рабочий каталог?
2. Что выполняет команда `ls` без параметров и аргументов?
3. Какая опция для команды `ls` будет отображать все файлы, включая скрытые?
4. На что указывает первый символ в табличном выводе команды `ls`?
5. Какую команду можно использовать для переименования файла и чем её можно заменить?
6. Какой параметр можно использовать с командой `rm` для вывода запроса перед удалением?
7. Какой флаг(опцию) надо передать команде `tar`, чтобы создать новый архив?
8. Какая команда покажет, что находится внутри сжатого архива?
9. Как понять, какой путь файловой системы указан?
10. В чем отличие файловой системы Windows и Linux?
11. Скрытые файлы - это файлы, которые начинаются с какого символа?
12. Какова разница между архивированием и сжатием?
13. Чем отличается жесткая ссылка на файл от мягкой?

Глава 5. ПОТОКИ, ПЕРЕНАПРАВЛЕНИЯ И РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ

5.1. Потоки командной строки

Большое количество файлов в типичной файловой системе представляют собой текстовые файлы. Текстовые файлы содержат просто текст без каких-либо функций форматирования.

Поскольку существует так много этих файлов в типичной системе Linux, существует большое количество команд, чтобы помочь пользователям управлять текстовыми файлами. Существуют команды для просмотра и изменения этих файлов различными способами.

Кроме того, оболочке доступны функции управления выводом команд, поэтому вместо размещения вывода в окне терминала выходные данные могут быть перенаправлены в файл или другую команду. Эти функции перенаправления предоставляют пользователям гибкую и мощную среду для работы.

Символ `|` (вертикальная черта) может использоваться для передачи выходных данных одной команды другой и называется конвейером или каналом. Вместо печати на экране вывод одной команды становится вводом для следующей команды. Это может быть мощным инструментом, особенно при поиске конкретных данных. Конвейер часто используется для уточнения результатов начальной команды.

Команды `head` и `tail` будут использоваться во многих примерах ниже, чтобы проиллюстрировать использование конвейера. Эти команды могут использоваться для отображения только первых или последних нескольких строк файла (или, при использовании с контейнером, вывода предыдущей команды).

По умолчанию команды `head` и `tail` отображают десять строк. Например, следующая команда отобразит первые десять строк файла `/etc/sysctl.conf`:

```
sysadmin@localhost:~$ head /etc/sysctl.conf
#
# /etc/sysctl.conf - Configuration file for setting system variables
# See /etc/sysctl.d/ for additional system variables
# See sysctl.conf (5) for information.
#
```

```
#kernel.domainname = example.com

# Uncomment the following to stop low-level messages on console
#kernel.printk = 3 4 1 3
sysadmin@localhost:~$
```

Конвейер позволяет пользователям использовать эти команды не только для файлов, но и для вывода других команд. Это может быть полезно при выводе большого каталога, например, каталога `/etc` командой `ls`. Что делать, если требуется перечислить первые несколько файлов каталога `/etc`?

Вместо отображения полного вывода команды `ls`, при передаче в команду `head` будут отображаться только первые десять строк:

```
sysadmin@localhost:~$ ls /etc | head
adduser.conf
adjtime
alternatives
apparmor.d
apt
bash.bashrc
bash_completion.d
bind
bindresvport.blacklist

blkid.conf
sysadmin@localhost:~$
```

Полный вывод команды `ls` передается в команду `head` оболочкой, а не печатается на экране. Команда `head` принимает эти выходные данные (от `ls`) как "входные данные", и вывод `head` затем печатается на экран.

Несколько каналов можно использовать последовательно для связывания нескольких команд в длинный конвейер. Если три команды передаются вместе, вывод первой команды переходит во вторую команду. Выходные данные второй команды затем передаются третьей команде. Выходные данные третьей команды будут напечатаны на экране.

5.2. Перенаправление ввода/вывода

Перенаправление ввода-вывода позволяет передавать информацию из командной строки в различные потоки. Прежде чем обсуждать перенаправление, важно понять стандартные потоки.

5.2.1. STDIN

Стандартный ввод или STDIN - это информация, обычно вводимая пользователем с клавиатуры. Когда команда запрашивает данные у оболочки, оболочка предоставляет пользователю возможность вводить команды, которые, в свою очередь, отправляются в команду как STDIN.

5.2.2. STDOUT

Стандартный вывод, или STDOUT, является нормальным выводом команд. Когда команда функционирует правильно (без ошибок), вывод, который она производит, называется STDOUT. По умолчанию STDOUT отображается в окне терминала (экране), где выполняется команда.

5.2.3. STDERR

Стандартная ошибка или STDERR - это сообщения об ошибках, создаваемые командами. По умолчанию STDERR отображается в окне терминала (экране), где выполняется команда.

Перенаправление ввода-вывода позволяет пользователю перенаправлять STDIN для ввода данных, поступающих из файла, и STDOUT / STDERR, для вывода в файл. Перенаправление осуществляется с помощью символов меньше и больше: < и > .

5.2.4. Перенаправление STDOUT

С помощью символа > выходные данные могут быть перенаправлены в файл:

```
sysadmin@localhost:~$ echo "Line 1"
Line 1
sysadmin@localhost:~$ echo "Line 1" > example.txt
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures   Templates example.txt test
Documents Music      Public     Videos   sample.txt
sysadmin@localhost:~$ cat example.txt
Line 1
sysadmin@localhost:~$
```

При перенаправлении команда echo не отображает выходные данные, так как STDOUT был перенаправлен в файл example.txt вместо экрана. Можно увидеть новый файл, введя команду ls. Вновь созданный файл содержит выходные данные команды echo при просмотре файла с помощью команды cat.

STDOUT известен как поток (или канал) #1.

5.2.5. Перенаправление STDERR

STDERR можно перенаправить аналогично STDOUT. STDERR известен как поток #2.

При использовании символа > для перенаправления по умолчанию предполагается поток #1, если не указан другой поток. Таким образом, поток #2 должен быть задан при перенаправлении STDERR в виде добавления цифры 2 к знаку больше: 2>.

Выполнив следующую команду, получим ошибку, так как указанный каталог не существует:

```
sysadmin@localhost:~$ ls /fake
ls: cannot access /fake: No such file or directory
sysadmin@localhost:~$
```

В приведенном выше примере нет ничего, что подразумевает, что выходные данные являются STDERR. Вывод явно содержит сообщение об ошибке, но как определить, что он отправляется в STDERR? Один простой способ определить это - перенаправить STDOUT:

```
sysadmin@localhost:~$ ls /fake > output.txt
ls: cannot access /fake: No such file or directory
sysadmin@localhost:~$
```

В приведенном выше примере STDOUT был перенаправлен в файл output.txt. Таким образом, вывод, который отображается, не может быть STDOUT, потому что он был бы перенаправлен в файл output.txt.

5.2.6. Перенаправление нескольких потоков

Можно одновременно направить как STDOUT, так и STDERR. Следующая команда создаст как STDOUT, так и STDERR, поскольку один из указанных каталогов существует, а другой нет:

```
sysadmin@localhost:~$ ls /fake /etc/ppp
ls: cannot access /fake: No such file or directory
/etc/ppp:
chap-secrets  ip-down      ip-down.ipv6to4  ip-up         ip-up.ipv6to4
ipv6-down     ipv6-up      options          pap-secrets    peers
```

Если в файл отправляется только STDOUT, STDERR все равно будет напечатан на экране:

```
sysadmin@localhost:~$ ls /fake /etc/ppp > example.txt
ls: cannot access /fake: No such file or directory
sysadmin@localhost:~$ cat example.txt
/etc/ppp:
chap-secrets
ip-down
```

```
ip-down.ipv6to4
ip-up
ip-up.ipv6to4
ipv6-down
ipv6-up
options
pap-secrets
peers
sysadmin@localhost:~$
```

Если в файл отправляется только STDERR, STDOUT все равно будет напечатан на экране:

```
sysadmin@localhost:~$ ls /fake /etc/ppp 2> error.txt
/etc/ppp:
hap-secrets  ip-down  ip-down.ipv6to4  ip-up  ip-up.ipv6to4
ipv6-down    ipv6-up  options          pap-secrets  peers
sysadmin@localhost:~$ cat error.txt
ls: cannot access /fake: No such file or directory
sysadmin@localhost:~$
```

Как STDOUT, так и STDERR могут быть отправлены в файл с помощью набора символов `&>`, который означает "1> и 2>":

```
sysadmin@localhost:~$ ls /fake /etc/ppp &> all.txt
sysadmin@localhost:~$ cat all.txt
ls: cannot access /fake: No such file or directory
/etc/ppp:
chap-secrets
ip-down
ip-down.ipv6to4
ip-up
ip-up.ipv6to4
ipv6-down
ipv6-up
options
pap-secrets
peers
```

5.3. Просмотр файлов с помощью команды `less`

При просмотре небольших файлов с помощью команды `cat` никаких проблем не возникает, но возникает проблема для больших файлов. Команда `cat` не позволяет легко приостановить или прокрутить отображение текста на экране, поэтому все содержимое файла сбрасывается на экран, но пользователь в итоге видит только последние строки.

Для больших файлов можно использовать команду “пейджера” для просмотра содержимого. Команды “пейджера” будут отображать одну страницу данных за раз, что позволяет двигаться вперед и назад в файле с помощью клавиш перемещения.

Есть две часто используемые команды “пейджера”:

- Команда `less`: эта команда предоставляет расширенные возможности подкачки. Обычно это пейджер по умолчанию, используемый командами, такими как команда `man`.
- Команда `more`: у этой команды меньше возможностей, чем у команды `less`, у него есть одно важное преимущество: команда `less` не всегда включена во все дистрибутивы Linux (а в некоторых дистрибутивах она не устанавливается по умолчанию). Команда `more` всегда доступна.

Команды `less` и `more` позволяют "перемещаться" по документу с помощью команд нажатия клавиш.

5.3.1. Экран помощи в `less`

При просмотре файла с помощью команды `less` можно использовать клавишу `h` для отображения экрана справки. Экран позволит увидеть, какие другие команды доступны. В следующем примере выполняется команда `less /usr/share/dict/words`. После отображения документа была нажата клавиша `h`, отображающая экран справки:

```
SUMMARY OF LESS COMMANDS
  Commands marked with * may be preceded by a number, N.
  Notes in parentheses indicate the behavior if N is given.

h  H          Display this help.
q  :q  Q  :Q  ZZ  Exit.
-----
                                MOVING

e  ^E  j  ^N  CR  *  Forward one line (or N lines).
y  ^Y  k  ^K  ^P  *  Backward one line (or N lines).
f  ^F  ^V  SPACE  *  Forward one window (or N lines).
b  ^B  ESC-v      *  Backward one window (or N lines).
z                                *  Forward one window (and set window to N).
w                                *  Backward one window (and set window to N).
ESC-SPACE          *  Forward one window, but don't stop at end-of-file.
d  ^D              *  Forward one half-window (and set half-window to N).

u  ^U              *  Backward one half-window (and set half-window to N).
ESC-)  RightArrow  *  Left one half screen width (or N positions).
ESC-(  LeftArrow   *  Right one half screen width (or N positions).
HELP -- Press RETURN for more, or q when done
```

5.3.2. Команды перемещения в `less`

В таблице 9 показаны клавиши, которые можно использовать для перемещения при запуске команды `less`.

Таблица 9

Команда	Клавиша
Следующее окно	Spacebar
Предыдущее окно	b
Следующая строка	Enter
Выход	q
Справка	h

При простом использовании `less` в качестве пейджера, самый простой способ продвижения - нажать клавишу Пробел (`Spacebar`).

5.3.3. Команды поиска в `less`

Есть два способа поиска в команде `less`: можно искать вперед или назад от текущей позиции, используя шаблоны, называемые регулярными выражениями.

Чтобы начать поиск вперед от текущей позиции, используется клавиша `/`. Затем вводится текст или шаблон для сопоставления и нажимается клавиша `Enter`.

Если совпадение найдено, то курсор переместится на совпадение в документе.

Чтобы начать поиск назад от текущего положения, нужно нажать `?`, ввести текст или шаблон для сопоставления и нажать клавишу `Enter`. Курсор переместится назад к первому найденному совпадению или сообщит, что шаблон не найден.

5.4. Сортировка файлов или входных данных

Команда `sort` может использоваться для изменения порядка вывода строк на основе содержимого одного или нескольких полей. Поля определяются разделителем полей, содержащимся в каждой строке, который по умолчанию содержит пробелы (пробелы и табуляции).

В следующем примере создается небольшой файл с помощью команды `head` для захвата первых 5 строк файла `/etc/passwd` и отправки выходных данных в файл `mypasswd`:

```
sysadmin@localhost:~$ head -5 /etc/passwd > mypasswd
sysadmin@localhost:~$
sysadmin@localhost:~$ cat mypasswd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
```

```
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
sysadmin@localhost:~$
```

Если произвести сортировку файла `mypasswd`:

```
sysadmin@localhost:~$ sort mypasswd
bin:x:2:2:bin:/bin:/bin/sh
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
root:x:0:0:root:/root:/bin/bash
sync:x:4:65534:sync:/bin:/bin/sync
sys:x:3:3:sys:/dev:/bin/sh
sysadmin@localhost:~$
```

Если файл или входные данные могут быть разделены другим разделителем, таким как запятая или двоеточие, параметр `-t` позволяет указать этот разделитель полей. Чтобы указать поля для сортировки, используется параметр `-k` с аргументом для указания номера поля (начиная с 1 для первого поля).

Другие часто используемые опции для команды `sort`: `-n` - выполнить числовую сортировку и `-r` - выполнить обратную сортировку.

В следующем примере параметр `-t` используется для разделения полей символом двоеточия и выполняет числовую сортировку с использованием третьего поля каждой строки:

```
sysadmin@localhost:~$ sort -t: -n -k3 mypasswd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
sysadmin@localhost:~$
```

Параметр `-r` может быть использован для реверсирования сортировки, в результате чего более высокие числа в третьем поле отображаются в верхней части выходных данных:

```
sysadmin@localhost:~$ sort -t: -n -r -k3 mypasswd
sync:x:4:65534:sync:/bin:/bin/sync
sys:x:3:3:sys:/dev:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
root:x:0:0:root:/root:/bin/bash
sysadmin@localhost:~$
```

5.5. Просмотр статистики файлов с помощью команды `wc`

Команда `wc` позволяет печатать до трех статистических данных для каждого файла, а также общее количество этих статистических

данных, если указано более одного файла. По умолчанию команда `wc` предоставляет количество строк, слов и байтов (1 байт = 1 символ в текстовом файле):

```
sysadmin@localhost:~$ wc /etc/passwd /etc/passwd-
 35   56 1710 /etc/passwd
 34   55 1665 /etc/passwd-
 69  111 3375 total
sysadmin@localhost:~$
```

В приведенном выше примере показан результат выполнения: `wc /etc/passwd /etc/passwd-`. Выходные данные имеют четыре столбца: количество строк в файле, количество слов в файле, количество байтов в файле и имя файла или общее количество.

Если пользователя интересуют только конкретные статистические данные, то можно использовать опцию `-l`, чтобы отобразить только количество строк, `-w` показывает только количество слов, `-c` показывает только количество байт.

Команда `wc` может быть полезна для подсчета количества строк, выводимых другой командой через конвейер. Например, если требуется узнать общее количество файлов в каталоге `/etc`, можно выполнить `ls /etc | wc -l`:

```
sysadmin@localhost:~$ ls /etc/ | wc -l
136
sysadmin@localhost:~$
```

5.6. Использование команды `cut` для фильтрации содержимого файла

Команда `cut` может извлекать столбцы текста из файла или стандартного ввода. Основное применение команды `cut` - работа с файлами базы данных с разделителями. Эти файлы очень распространены в системах Linux.

По умолчанию считается, что входные данные разделены символом табуляции, но параметр `-d` может указывать альтернативные разделители, такие как двоеточие или запятая.

С помощью параметра `-f` можно указать, какие поля будут отображаться, либо в виде диапазона с переносом, либо в виде списка, разделенного запятыми.

В следующем примере отображаются первое, пятое, шестое и седьмое поля из файла базы данных `mypasswd`:

```
sysadmin@localhost:~$ cut -d: -f1,5-7 mypasswd
root:root:/root:/bin/bash
daemon:daemon:/usr/sbin:/bin/sh
bin:bin:/bin:/bin/sh
sys:sys:/dev:/bin/sh
sync:sync:/bin:/bin/sync
sysadmin@localhost:~$
```

С помощью команды `cut` можно также извлечь столбцы текста на основе позиции символа с помощью опции `-с`. Это может быть полезно для извлечения полей из файлов базы данных фиксированной ширины. Например, ниже будет показан только тип файла (символ #1), разрешения (символы #2-10) и имя файла (символы #50+) выходных данных команды `ls -l`:

```
sysadmin@localhost:~$ ls -l | cut -c1-11,50-
total 12
drwxr-xr-x Desktop
drwxr-xr-x Documents
drwxr-xr-x Downloads
drwxr-xr-x Music
drwxr-xr-x Pictures
drwxr-xr-x Public
drwxr-xr-x Templates
drwxr-xr-x Videos
-rw-rw-r-- errors.txt
-rw-rw-r-- mypasswd
-rw-rw-r-- new.txt
sysadmin@localhost:~$
```

5.7. Использование команды `grep` для фильтрации содержимого файла

Команда `grep` может использоваться для фильтрации строк в файле или выходных данных другой команды на основе соответствия шаблону. Этот шаблон может быть простым, как точный текст, которому он должен соответствовать или он может быть гораздо более продвинутым с помощью регулярных выражений.

Например, можно найти всех пользователей, которые могут войти в систему с помощью оболочки BASH, используя команду `grep` для фильтрации строк из файла `/etc/passwd`, содержащих символы `bash`:

```
sysadmin@localhost:~$ grep bash /etc/passwd
root:x:0:0:root:/root:/bin/bash
sysadmin:x:1001:1001:System Administrator,,,:/home/sysadmin:/bin/bash
```

```
sysadmin@localhost:~$
```

Чтобы было легче увидеть соответствия, используется параметр `--color`. Этот параметр выделит совпадающие элементы красным цветом.

В некоторых случаях пользователя не интересуют сами строки, которые соответствуют шаблону, а интересует сколько таких строк. С опцией `-c`, можно получить количество строк:

```
sysadmin@localhost:~$ grep -c bash /etc/passwd
2
sysadmin@localhost:~$
```

При просмотре выходных данных команды `grep` может быть трудно определить исходные номера строк. Эта информация может быть полезна, когда пользователь возвращается в файл (например, для его редактирования). Опция `-n` команды `grep` отобразит исходные номера строк:

```
sysadmin@localhost:~$ grep -n bash /etc/passwd
1:root:x:0:0:root:/root:/bin/bash
24:sysadmin:x:1001:1001:System Administrator,,,:/home/sysadmin:/bin/bas
sysadmin@localhost:~$
```

5.8. Базовые регулярные выражения

Регулярное выражение представляет собой набор "нормальных" и "специальных" символов, которые используются для сопоставления простых или сложных шаблонов. Нормальные или обычные символы - это буквенно-цифровые символы, которые соответствуют друг другу. Например, 'a' будет соответствовать a.

Некоторые символы имеют специальные значения при использовании в шаблонах командами, такими как `grep`. Есть как базовые регулярные выражения (доступен широкий спектр команд Linux) и расширенных регулярных выражений (для более продвинутых команд Linux). В таблице 10 показаны основные регулярные выражения.

Таблица 10

Регулярное выражение	Сопоставление
.	Любой одиночный символ
[]	Список или диапазон символов, соответствующий одному символу, если первый символ не является кареткой ^, а затем любой символ
*	Предыдущий символ повторяется ноль или более раз
^	Следующий текст должен появиться в начале строки
\$	Предшествующий текст должен появиться в конце строки

Помимо команды `grep` регулярные выражения поддерживают и некоторые другие команды, включая команды `more` и `less`. Хотя некоторые регулярные выражения без необходимости заключаются в одинарные кавычки, рекомендуется использовать одинарные кавычки вокруг регулярных выражений, чтобы оболочка не пыталась интерпретировать из них особое значение как из символов раскрытия (`glob`):

5.8.1. Базовые регулярные выражения – символ . (точка)

В приведенном ниже примере простой файл сначала создается с помощью перенаправления. Затем команда `grep` используется для демонстрации простого соответствия шаблону:

```
sysadmin@localhost:~$ echo 'abcddd' > example.txt
sysadmin@localhost:~$ cat example.txt
abcddd
sysadmin@localhost:~$ grep --color 'a..' example.txt
abcddd
sysadmin@localhost:~$
```

В предыдущем примере видно, что шаблон `a..` соответствует `abc`. Первый символ `.` соответствовал `b`, второй соответствовал `c`.

В следующем примере шаблон `c..` не будет совпадать ни с чем, поэтому команда `grep` не будет производить никаких выходных данных. Чтобы соответствие было успешным, должно быть два символа между `a` и `c` в `example.txt`:

```
sysadmin@localhost:~$ grep --color 'a..c' example.txt
sysadmin@localhost:~$
```

5.8.2. Базовые регулярные выражения – символы []

Если используется символ `.`, ему может соответствовать любой символ. В некоторых случаях нужно точно указать, какие символы требуется сопоставить. Например, чтобы сопоставить буквенный символ

в нижнем регистре или числовой символ. Для этого можно использовать символы регулярного выражения `[]` и указать допустимые символы внутри символов `[]`.

Например, следующая команда соответствует двум символам: первый - а или b, а второй - а, b, с или d:

```
sysadmin@localhost:~$ grep --color '[ab][a-d]' example.txt
abcddd
sysadmin@localhost:~$
```

Можно либо перечислить каждый возможный символ `[abcd]`, либо указать диапазон `[a-d]`, если диапазон находится в правильном порядке. Например, `[d-a]` не будет работать, потому что это недопустимый диапазон:

```
sysadmin@localhost:~$ grep --color '[d-a]' example.txt
grep: Invalid range end
sysadmin@localhost:~$
```

5.8.3. Базовые регулярные выражения — символ `*`

Символ `*` может использоваться для сопоставления предыдущего символа ноль или более раз. Например, следующее будет соответствовать нулю или более раз символа `d`:

```
sysadmin@localhost:~$ grep --color 'd*' example.txt
abcddd
sysadmin@localhost:~$
```

5.8.4. Базовые регулярные выражения — символы `^` и `$`

Сопоставление с шаблоном можно произвести в любом месте строки. Можно указать, что совпадение происходит в начале или в конце строки. Для указания того, что совпадение происходит в начале строки, шаблон начинается с символа `^`.

В следующем примере добавляется еще одна строка в файл `example.txt` для демонстрации использования символа `^`:

```
sysadmin@localhost:~$ echo "xyzabc" >> example.txt
sysadmin@localhost:~$ cat example.txt
abcddd
xyzabc
sysadmin@localhost:~$ grep --color "a" example.txt
abcddd
xyzabc
sysadmin@localhost:~$ grep --color "^a" example.txt
abcddd
```

В первых выходных данных `grep` обе строки совпадают, так как они обе содержат букву `a`. Во втором выводе `grep` совпала только строка, которая начиналась с буквы `a`.

Чтобы указать, что совпадение происходит в конце строки, нужно завершить шаблон символом `$`. Например, чтобы найти только строки, которые заканчиваются буквой `c`:

```
sysadmin@localhost:~$ grep "c$" example.txt
xyzabc
sysadmin@localhost:~$
```

5.8.5. Базовые регулярные выражения – символ `\`

В некоторых случаях может потребоваться сопоставить символ, который является специальным символом регулярного выражения. Например, рассмотрим следующее:

```
sysadmin@localhost:~$ echo "abcd*" >> example.txt
sysadmin@localhost:~$ cat example.txt
abcd
xyzabc
abcd*
sysadmin@localhost:~$ grep --color "cd*" example.txt
abcd
xyzabc
abcd*
sysadmin@localhost:~$
```

В выходных данных команды `grep` выше можно увидеть, что каждая строка совпадает, потому что происходит поиск символа `c`, за которым следует ноль или больше символов `d`. Если нужно найти реальный символ `*`, требуется поместить символ `\` перед символом `*`:

```
sysadmin@localhost:~$ grep --color "cd\*" example.txt
abcd*
sysadmin@localhost:~$
```

5.9. Расширенные регулярные выражения

Использование расширенных регулярных выражений требует специальной опции `-E`, предоставляемой команде `grep` для их распознавания. Исторически сложилось так, что существует команда под названием `egrep`, которая похожа на `grep`, но способна понять их использование без дополнительных опций. Команда `egrep` устарела, и чаще используется `grep` с опцией `-E`.

В таблице 11 показаны регулярные выражения, которые считаются "расширенными".

Таблица 11

Регулярное выражение	Сопоставление
?	Соответствует предыдущему символу ноль или один раз, поэтому это необязательный символ
+	Соответствует предыдущему символу, повторенному один или несколько раз
	Чередование или как логический оператор ИЛИ

Некоторые примеры расширенных регулярных выражений показаны в таблице 12.

Таблица 12

Пример	Вывод
grep -E 'colou?r' 2.txt	color или colour
grep -E 'd+' 2.txt	Сопоставить один или несколько символов d (d, dd, ddd, dddd...)
grep -E 'gray grey' 2.txt	Сопоставить либо gray, либо grey

Контрольные вопросы

1. Как написать команду, чтобы направлять сообщения об ошибках в файл?
2. Какие команды можно использовать для просмотра текстового файла с прокруткой?
3. Куда по умолчанию отправляются сообщения об ошибках, генерируемые командами?
4. Какая разница между перенаправлениями символами > и >>?
5. Каково основное предназначение команды grep?
6. В чем отличие регулярных выражений от символов подстановки?

Глава 6. СОЗДАНИЕ СКРИПТОВ

6.1. Скрипты оболочки

Скрипт оболочки – это файл, содержащий исполняемые команды, который был сохранен в текстовом виде. Такие файлы называют ещё сценариями. Когда файл запускается, выполняется каждая его команда. Скрипты оболочки имеют доступ ко всем командам оболочки, включая логические. Таким образом, скрипт может проверить наличие файла или искать конкретный результат и соответственно изменить его поведение. Можно создавать скрипты для автоматизации повторяющихся частей работы, что освобождает время и обеспечивает согласованность при каждом использовании скрипта.

Скрипт может быть простым, как одна команда:

```
echo "Привет, Мир!"
```

Скрипт `test.sh` состоит только из одной строки, которая печатает строку `Привет, Мир!` на консоль.

Запуск скрипта можно выполнить либо путем передачи его в качестве аргумента в оболочку, либо путем ее непосредственного запуска:

```
sysadmin@localhost:~$ sh test.sh
Привет, Мир!
sysadmin@localhost:~$ ./test.sh
-bash: ./test.sh: Отказано в доступе
sysadmin@localhost:~$ chmod +x ./test.sh
sysadmin@localhost:~$ ./test.sh
Привет, Мир!
```

В приведенном выше примере сначала скрипт запускается как аргумент для оболочки. Затем скрипт запускается непосредственно из оболочки. Текущий каталог редко есть в пути поиска `$PATH`, поэтому имя имеет префикс `./`, чтобы указать, что он должен быть запущен из текущего каталога.

Ошибка «Отказано в доступе» означает, что файл скрипта не был помечен как исполняемый. Команда `chmod` используется для изменения разрешений файла, что будет подробно описано в следующей главе.

Существуют различные оболочки с их собственным синтаксисом языка. Поэтому более сложные скрипты будут указывать конкретную оболочку, указав абсолютный путь к интерпретатору в качестве первой строки с префиксом `#!` как показано далее:

```
#!/bin/sh
echo "Привет, Мир!"
```

ИЛИ

```
#!/bin/bash
echo "Привет, Мир!"
```

Два символа `# !` традиционно называются хешем и бэгом соответственно, что приводит к сокращенной форме `shebang`, когда они используются в начале скрипта.

Кстати, `shebang` (или `crunchbang`) используется для традиционных сценариев оболочки и других текстовых языков, таких как Perl, Ruby и Python. Любой текстовый файл, помеченный как исполняемый, будет запускаться под интерпретатором, указанным в первой строке, если скрипт запускается напрямую. Если скрипт вызывается в качестве аргумента для интерпретатора, например `sh`-скрипта или скрипта `bash`, данная оболочка будет использоваться независимо от того, что находится в строке `shebang`.

Для создания текстовых файлов скриптов используются традиционные текстовые редакторы без функций форматирования. Офисные инструменты, такие как LibreOffice, чьи форматы выходных файлов содержат форматирование и другую дополнительную информацию, не подходят для этой задачи.

6.2. Редактирование скриптов оболочки

В UNIX есть много текстовых редакторов. Редактор GNU nano — очень простой, хорошо подходящий для редактирования небольших текстовых файлов. Визуальный редактор `vi` (или его улучшенная версия `vim`) является мощным редактором, но сложен для обучения. Обратим внимание на `nano`.

Если напечатать `nano test.sh`, появится следующий экран:

```
GNU nano 2.2.6                               File: test.sh                               modified

#!/bin/sh
echo "Привет, Мир!"
echo -n "Время - "
date

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Po
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text ^T To Spell
```

В редакторе `nano` есть несколько полезных функций. Можно нажимать на клавиатуре клавиши со стрелками для перемещения курсора и кнопку `backspace`, чтобы удалить текст. В нижней части экрана

можно увидеть некоторые доступные команды, которые чувствительны к контексту и изменяются в зависимости от того, что делает пользователь. Если он находится непосредственно на самой машине Linux, в отличие от подключения по сети, можно использовать также мышь для перемещения курсора и выделения текста.

^X Exit – означает «нажать кнопку Ctrl и X для выхода». Следует нажать Ctrl и X вместе. Если изменения не были сохранены при выходе будет выведено сообщение:

```
Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ?
Y Yes
N No          ^C Cancel
```

На этом этапе можно выйти из программы без сохранения, нажав клавишу N, или сначала сохранить, нажав Y. По умолчанию используется сохранение файла с текущим именем. Можно нажать клавишу Enter, чтобы сохранить и выйти.

После сохранения пользователь вернется к приглашению оболочки, а затем к редактору. Для сохранения изменений без выхода из редактора надо нажать Ctrl и O.

Сочетание Ctrl и K вырезает строку текста. Сочетание Ctrl и U используется для вставки буфера копирования в текущую позицию.

Другие полезные команды редактора nano показаны в таблице 13.

Таблица 13

Команда	Описание
Ctrl + W	поиск документа
Ctrl + W , затем Ctrl+ R	поиск и перемещение
Ctrl + G	показать все доступные команды
Ctrl + Y/V	листать страницу вверх / вниз
Ctrl + C	показать текущую позицию в файле и размер файла

6.3. Основы создания скриптов

Помимо запуска команд, следует ознакомиться с тремя темами:

- переменные, содержащие временную информацию в скрипте;

- условные обозначения, которые позволяют выполнять разные действия на основе написанных пользователем тестов;
- циклы, которые позволяют повторять одно и то же.

6.3.1. Переменные

Переменные являются ключевой частью любого языка программирования. Здесь показано очень простое использование переменных:

```
#!/bin/bash
ANIMAL="пингвин"
echo "Моя любимая птица - $ANIMAL"
```

После строки `shebang` показана директива для назначения переменной некоторого текста. Имя переменной – `ANIMAL`, а знак «равно» обозначает присвоить ей строку «`penguin`». Переменная подобна коробке, в которой можно хранить вещи. После выполнения этой строки в поле под названием «`ANIMAL`» содержится слово «`penguin`».

Важно, чтобы не было пробелов между именем переменной, знаком равенства и элементом, который должен быть присвоен переменной. Если есть пробел, можно получить ошибку, например, «команда не найдена».

Приведенный скрипт выдает строку на консоль командой `echo`. Строка содержит имя переменной, которой предшествует знак доллара. Когда интерпретатор встречает знак доллара, он заменяет имя переменной на её содержимое, – это называется подстановкой или интерпретацией. Результат скрипта – «Моя любимая птица – пингвин».

Чтобы назначить переменную, следует использовать просто имя переменной. Чтобы получить доступ к содержимому переменной, надо сопроводить ее знаком доллара. В примере ниже показывается как переменной присваивается содержимое другой переменной.

```
#!/bin/bash
ANIMAL=пингвин
SOMETHING=$ANIMAL
echo "Моя любимая птица - $SOMETHING"
```

Переменная `ANIMAL` содержит строку «пингвин» (т.к. нет пробелов, показан альтернативный синтаксис без использования кавычек). Затем `SOMETHING` присваивает содержимое `ANIMAL` (т.к. перед `ANIMAL` есть знак доллара).

По желанию можно присвоить переменной интерпретированную строку. Это довольно распространено в больших скриптах, так как можно создать большую команду и выполнить ее.

Другой способ присвоить значение переменной – использовать вывод другой команды в качестве содержимого переменной:

```
#!/bin/bash
CURRENT_DIRECTORY=`pwd`
echo "Вы находитесь в директории $CURRENT_DIRECTORY"
```

Этот шаблон часто используется для обработки текста. Можно взять текст из одной переменной или входного файла и передать его через другую команду, например, `sed` или `awk`, для извлечения определенных частей и сохранения результата в переменной.

Можно получить входные данные от пользователя скрипта и назначить их переменной через команду `read`:

```
#!/bin/bash
echo -n "Введите свое имя: "
read NAME
echo "$NAME, здравствуйте!"
```

Команда `read` может принимать строку прямо с клавиатуры или как часть перенаправления команд.

Можно передать аргументы в свой скрипт:

```
#!/bin/bash
echo "$1, здравствуйте!"
```

Знак доллара, за которым следует число `N`, соответствует `N`-му аргументу, переданному скрипту. Если пример выше будет вызван командой `./test.sh Linux`, то получится вывод строки `Linux, здравствуйте!`. Переменная `$0` содержит имя самого скрипта.

После запуска программы возвращается код выхода, который является целым числом от 0 до 255. Чтобы увидеть, успешно ли выполнена предыдущая команда, можно проверить код выхода через переменную `$?`

```
sysadmin@localhost:~$ grep -q root /etc/passwd
sysadmin@localhost:~$ echo $?
0
sysadmin@localhost:~$ grep -q notuser /etc/passwd
sysadmin@localhost:~$ echo $?
1
```

Команда `grep` использовалась для поиска строки в файле с флагом `-q`, что означает «тихий» режим. `grep`, работая в тихом режиме, возвращает 0, если строка была найдена, и 1 в противном случае. Эта информация может использоваться в последующем выражении.

Аналогичным образом можно установить код выхода собственного скрипта пользователя с помощью команды `exit`:

```
#!/bin/bash
# Произошло что-то плохое!
exit 1
```

В приведенном выше примере показан комментарий `#`. Все, что записано после знака хэш (`#`), можно использовать, чтобы помочь программисту или администратору оставить заметки. `exit 1` возвращает код выхода 1. Это работает в оболочке, даже если скрипт будет запущен из командной строки; если затем ввести `echo $?`, видно, что он возвращает 1.

Код выхода 0 означает «все в порядке». Любые коды выхода, превышающие 0, означают, что произошла какая-то ошибка, которая специфична для программы.

6.3.2. Условные выражения

Можно сделать так, чтобы скрипт выполнял различные функции на основе тестов, называемых ветвями. Оператор `if` является основным оператором для реализации ветвления.

Утверждение `if` выглядит так:

```
if команда1; then
    # выполнять если команда1 имеет код выхода 0
fi
```

Следующий пример будет запускать «некоторые команды» (на самом деле, все до точки с запятой). Используя то, что известно о `grep`, можно написать скрипт, который выполняет разные действия в зависимости от наличия строки в файле паролей:

```
#!/bin/bash

if grep -q root /etc/passwd; then
    echo Пользователь root есть в файле паролей
else
    echo Пользователь root не найден в файле паролей
fi
```

Из предыдущих примеров известно, что код выхода `grep` равен 0, если строка найдена. В приведенном выше примере это используется в

одной строке для печати сообщения, если `root` находится в файле пароля, или в другом сообщении, если это не так. Разница здесь в том, что вместо `fi` для закрытия блока `if` есть еще один. Это позволяет выполнить одно действие, если условие истинно, и другое, если условие ложно. Блок `else` все еще должен быть закрыт ключевым словом `fi`.

Распространенными задачами являются поиск наличия файла или каталога и сравнение строк и чисел. Можно инициализировать файл журнала, если он не существует, или сравнить количество строк в файле, когда он присутствует. Очевидно, что команда `if` поможет, но какая команда используется для сравнения?

Команда `test` дает легкий доступ к операторам сравнения и проверки файлов. Примеры показаны в таблице 14.

Таблица 14

Команда	Описание
<code>test -f /dev/ttyS0</code>	0, если файл существует
<code>test ! -f /dev/ttyS0</code>	0, если файл не существует
<code>test -d /tmp</code>	0, если директория существует
<code>test -x test.sh</code>	0, если файл существует и доступен для выполнения
<code>test 1 -eq 1</code>	0, если числовое сравнение выполнено успешно (равно)
<code>test ! 1 -eq 1</code>	0, если сравнение не выполняется (не равно)
<code>test 1 -ne 1</code>	0, для числового неравенства (не равно)
<code>test "a" = "a"</code>	0, если строки совпадают
<code>test "a" != "a"</code>	0, если строки различаются
<code>test 1 -eq 1 -o 2 -eq 2</code>	-o это логическое ИЛИ: может быть выполнено одно из условий
<code>test 1 -eq 1 -a 2 -eq 2</code>	-a это логическое И: оба условия должны быть выполнены

Важно отметить, что `test` смотрит на целочисленные и строковые сравнения по-разному. `01` и `1` совпадают по цифровому сравнению, но не по сопоставлению строк. Следует помнить, какой тип ввода ожидается.

Есть еще много тестов, таких как `-gt` (больше), и способов проверки, например, один файл более новый, чем другой, и многого другого.

Команда `test` немного усложняет чтение команды `if`, поэтому есть псевдоним для него, называемый `[` (левая квадратная скобка).

Если заключить условия в квадратных скобках, это будет то же самое, что и `test`. Следующие два утверждения идентичны.

```
if test -f /tmp/foo; then
if [ -f /tmp/foo]; then
```

В то время как последняя форма используется чаще, важно понимать, что квадратная скобка – это сама по себе команда, которая работает аналогично `test`, за исключением того, что для нее требуется закрывающая квадратная скобка.

Оператор `if` имеет форму, которая позволяет выполнять несколько сравнений за один раз, используя `elif` (краткий вариант `else if`).

```
#!/bin/bash
if [ "$1" = "привет" ]; then
    echo "Добрый день"
elif [ "$1" = "пока" ]; then
    echo "Рады были вас видеть"
    echo "Ещё увидимся"
else
    echo "Не понятно"
fi
```

В приведенном выше коде сравнивается первый аргумент, переданный скрипту. Если это «привет», выполняется первый блок. Если это не так, скрипт проверяет, «пока». Для «пока» выполняется второй блок с двумя `echo`. В противном случае отправляется третье сообщение. В примере используется оператор сравнения строк вместо числовой версии (`-eq`).

Тесты `if/elif/else` могут стать довольно многословными и сложными. Оператор `case` – альтернатива для упрощения нескольких тестов.

```
#!/bin/bash
case "$1" in
привет|здравствуйте)
    echo "Добрый день"
    ;;
пока)
    echo "Рады были вас видеть"
    echo "Ещё увидимся"
    ;;
*)
    echo "Не понятно"
esac
```

Оператор `case` начинается с описания тестируемого выражения: `case` выражение `in`. Выражение в примере – это `$1`.

Выражение сравнивается с шаблонами, заканчивающимися закрывающей скобкой. Предыдущий пример сначала ищет «привет» или «здравствуйте». Варианты разделены вертикальной чертой |, которая является оператором ИЛИ во многих языках программирования. Ниже приведены команды, которые должны выполняться, если шаблон возвращает истину. Команды заканчиваются двумя точками с запятой. Если тест возвращает ложь, выражение сравнивается со следующими шаблонами.

Шаблон * совпадает с else (в противном случае). Поведение оператора case аналогично if / elif / else в том, что обработка останавливается после первого совпадения. Если ни один из шаблонов не равен искомому выражению – это соответствует значению *.

При правильном понимании условий можно использовать свои скрипты только в случае необходимости.

6.3.3. Циклы

Циклы позволяют выполнять код повторно. Они могут быть полезны во многих ситуациях, например, когда требуется запускать одни и те же команды над каждым файлом в каталоге или повторять несколько действий множество раз. В сценариях оболочки есть два основных цикла: for и while.

Циклы for используются, когда конечная коллекция, по которой планируется итеративное перемещение, например, список файлов или список имен серверов:

```
#!/bin/bash
SERVERS="servera serverb serverc"
for S in $SERVERS; do
    echo "Выполнять действия для $S"
done
```

Сначала скрипт устанавливает переменную, содержащую список имен серверов, разделенных пробелами. Выражение for циклично работает над списком серверов, каждый раз, когда он устанавливает переменную S в текущее имя сервера. Выбор S был произвольным, но следует обратить внимание, что S не имеет знака доллара, а \$SERVERS имеет, показывая, что \$SERVERS будет заменен списком серверов. Список не обязательно должен быть переменной. В следующем примере показаны еще два способа передачи списка.

```
#!/bin/bash
for NAME in Вася Петя Коля; do
    echo "$NAME, привет!"
done
for S in *; do
    echo "Выполнить что-то с $$S"
done
```

Первый цикл функционально такой же, как в предыдущем примере, за исключением того, что список передается в цикл `for` напрямую вместо использования переменной. Использование переменной помогает добавить ясности сценарию, так как можно легко внести изменения в переменную и не смотреть на цикл.

Второй цикл использует символ `*`, который является символом раскрытия (файловым глобом). оболочка подставляет имена всех файлов в текущем каталоге.

Цикл `while` работает со списком неизвестного размера. Его задача состоит в том, чтобы на каждой итерации выполнять проверку (тест), чтобы увидеть, должен ли он выполнять тело цикла. Это похоже на «пока какое-то условие верно, делайте что-нибудь».

```
#!/bin/bash
i=0
while [ $i -lt 10 ]; do
    echo $i
    i=$(( $i + 1 ))
done
echo Расчет окончен
```

В приведенном выше примере показан цикл `while`, который считает от 0 до 9. Переменная счетчика `i` инициализируется на 0. Затем выполняется цикл `while` с тестом `i<10`. Цикл `while` использует такую же команду `test`, что и выражение `if`.

Внутри цикла `while` к текущему значению переменной `i` добавляется 1 через команду `$((арифметическое выражение))` и полученное значение записывается обратно в переменную `i`. Когда `i` становится равно 10, оператор `while` в тесте возвращает `false`, и обработка продолжается после цикла.

Контрольные вопросы

1. Какая команда оболочки принимает ввод с клавиатуры пользователя?
2. С какой строки начинается файл скрипта? Что эта строка означает?
3. Количество пользователей, вошедших в систему, указывается в переменной USERS. Как можно проверить, вошли ли в систему 5 пользователей?
4. Какая информация хранится внутри \$?
5. Какой код проверки условия ищет команда if, чтобы считать условие истинным?
6. Что означает \$1?
7. Какие циклы можно использовать в BASH скриптах?
8. Можно ли использовать для исполнения скрипта интерпретатор языка высокого уровня, например Python? Если да, то что для этого нужно сделать?
9. Какими средствами можно редактировать скрипты оболочки?
10. Как иначе можно записать команду test?

Глава 7. КОМПЬЮТЕРНОЕ ОБОРУДОВАНИЕ

7.1. Процессоры

Одним из многих преимуществ наличия множества разных дистрибутивов Linux является то, что некоторые из них предназначены для работы на определенных аппаратных платформах. Есть дистрибутивы Linux, специально разработанные для всех современных аппаратных платформ.

Каждая из этих аппаратных платформ имеет большое разнообразие в доступных аппаратных компонентах. В дополнение к нескольким различным типам жестких дисков существует множество различных мониторов и принтеров. С популярностью USB-устройств, таких как USB-устройства хранения данных, камеры и сотовые телефоны, количество доступных устройств насчитывает тысячи.

В некоторых случаях это создает проблемы, так как этим аппаратным устройствам обычно требуется специальное программное обеспечение (называемое драйверами), которое позволяет им связываться с установленной операционной системой. Производители оборудования часто предоставляют такое программное обеспечение, но обычно для Microsoft Windows, а не для Linux. Однако операционная система Linux получила широкое распространение в различных областях, что указывает на поддержку Linux и производителями аппаратных средств.

В дополнение к поддержке поставщиков, существует большая поддержка сообщества, которая стремится предоставить универсальные открытые драйверы для Linux-систем. Хотя не у всех аппаратных средств есть необходимые драйверы, их существует достаточное количество.

Аппаратные устройства доступны через специальные файлы в каталоге /dev.

Процессор подключается к остальному оборудованию через материнскую плату, также известную как системная плата. Материнские платы предназначены для работы с конкретными типами процессоров.

Если аппаратная система имеет больше одного процессора, система называется мультипроцессорной. Если несколько процессоров

объединены в один процессорный чип, то он называется многоядерным.

Хотя поддержка доступна для большего количества типов процессоров в Linux, чем в любой другой операционной системе, в настольных и серверных компьютерах обычно используются только два типа процессоров: x86 и x86_64. В системе x86 система обрабатывает данные 32 бита за раз; на x86_64 система обрабатывает данные 64 бита за раз. Система x86_64 также может обрабатывать данные 32 бита за один раз в режиме обратной совместимости. Одним из основных преимуществ 64-разрядной системы является то, что система может работать с большим объемом памяти.

Семейство процессоров x86 было создано Intel в 1978 году с выпуском процессора 8086. С тех пор Intel выпустила много других процессоров, которые улучшают исходный 8086; они известны в общем виде как процессоры x86. Эти процессоры включают в себя 80386 (также известный как i386), 80486 (i486), Pentium серии (i586) и Pentium Pro серии (i686). В дополнение к Intel, другие компании, такие как AMD и Cyrix, также выпустили x86-совместимые процессоры. Хотя Linux способен поддерживать процессоры в поколении i386, многие дистрибутивы ограничивают их поддержку i686 или новее.

Семейство процессоров x86_64, включая 64-разрядные процессоры от Intel и AMD, выпускается с 2000 года. В результате большинство современных процессоров, построенных сегодня, – x86_64. Хотя аппаратное обеспечение было доступно уже более двадцати лет, программное обеспечение для поддержки этого семейства процессоров развивалось гораздо медленнее. До сих пор существует множество программных пакетов, доступных для архитектуры x86, но не x86_64.

Все современные процессоры линейки x86 64-разрядные, поэтому все последние версии дистрибутивов Linux выпускаются только под эти процессоры. Для более старых 32-разрядных процессоров x86 нужно устанавливать устаревшие версии дистрибутивов.

Можно увидеть семейство, к которому принадлежит процессор, с помощью команды `arch`:

```
sysadmin@localhost:~$ arch
x86_64
sysadmin@localhost:~$
```

Более детальную информацию о процессоре дает команда `lscpu`:

```
sysadmin@localhost:~$ lscpu
Архитектура:          x86_64
CPU op-mode(s):      32-bit, 64-bit
Address sizes:        39 bits physical, 48 bits virtual
Порядок байт:         Little Endian
CPU(s):               4
On-line CPU(s) list:  0-3
ID производителя:     GenuineIntel
Имя модели:           Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz
Семейство ЦПУ:        6
Модель:               165
Потоков на ядро:       1
Ядер на сокет:         4
Сокетов:              1
Степпинг:             2
ВогомIPS:              4991.99
Флаги:                fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge
                      mca cmov pat pse36 clflush mmx fxsr sse sse2 ht sysca
                      ll nx rdtscp lm constant_tsc rep_good nopl xtopology
                      nonstop_tsc cpuid tsc_known_freq pni pclmulqdq ssse3
                      cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt aes xsave
                      avx rdrand hypervisor lahf_lm abm 3dnowprefetch invp
                      cid_single fsgsbase avx2 invpcid rdseed clflushopt md
                      _clear flush_l1d arch_capabilities

--More--
```

Первая строка этого вывода показывает, что ЦП используется в 64-битном режиме, так как сообщается об архитектуре `x86_64`. Вторая строка вывода показывает, что процессор способен работать в режиме 32 или 64 бит, поэтому на самом деле это 64-битный процессор.

Самый подробный способ отображения информации о ЦП – просмотр файла `/proc/cpuinfo` с помощью команды `cat`:

```
sysadmin@localhost:~$ cat /proc/cpuinfo
processor           : 0
vendor_id          : GenuineIntel
cpu family         : 6
model              : 165
model name         : Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz
stepping: 2
cpu MHz            : 2495.998
cache size         : 8192 KB
physical id        : 0
siblings: 4
core id            : 0
cpu cores          : 4
apicid             : 0
initial apicid     : 0
fpu                : yes
fpu_exception      : yes
cpuid level        : 22
wp                 : yes
flags               : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
pse36 clflush mmx fxsr sse sse2 ht syscall nx rdtscp lm constant_tsc rep_good nopl
xtopology nonstop_tsc cpuid tsc_known_freq pni pclmulqdq ssse3 cx16 pcid sse4_1 sse4_2
```

```
x2apic movbe popcnt aes xsave avx rdrand hypervisor lahf_lm abm 3dnowprefetch invpcid_single fsgsbase avx2 invpcid rdseed clflushopt md_clear flush_l1d arch_capabilities
--More--
```

Большая часть вывода `lsrpu` и содержимого файла `/proc/cpuinfo` оказывается одинаковой. Флаги ЦП являются очень важным компонентом, поскольку они указывают, какие функции поддерживает процессор, и возможности процессора.

Например, выход из предыдущего примера содержит флаг `lm` (длинный режим), что указывает на то, что этот ЦП способен работать на 64 бита. Существуют также флаги, указывающие, способен ли процессор поддерживать виртуальные машины (возможность иметь несколько операционных систем на одном компьютере). Некоторые функции виртуализации могут быть ограничены в настройках BIOS или установленной операционной системой.

7.2. Материнская плата и шина

Материнская плата (системная плата) является основной аппаратной платой компьютера, через которую центральный процессор (ЦП), оперативное запоминающее устройство (ОЗУ) и другие компоненты соединены друг с другом. Некоторые устройства подключаются непосредственно к материнской плате, а другие устройства подключаются через шину.

7.2.1. `dmidecode`

Системная плата многих компьютеров содержит так называемую базовую систему ввода-вывода (BIOS). BIOS системного управления (SMBIOS) – это стандарт, определяющий структуры данных и способы передачи информации об аппаратном обеспечении компьютера. Команда `dmidecode` может считывать и отображать информацию из SMBIOS.

Для устройств, подключенных непосредственно к материнской плате, администратор может использовать команду `dmidecode` для их просмотра. Существует большое количество информации, предоставляемой выходом этой команды.

В следующем примере видно, что BIOS поддерживает загрузку непосредственно с компакт-диска. Это важно, поскольку установки

операционной системы часто выполняются путем загрузки непосредственно с установочного компакт-диска:

```
# dmidecode 3.3
Getting SMBIOS data from sysfs.
SMBIOS 2.5 present.
10 structures occupying 450 bytes.
Table at 0x000E1000.

Handle 0x0000, DMI type 0, 20 bytes
BIOS Information
    Vendor: innotek GmbH
    Version: VirtualBox
    Release Date: 12/01/2006
    Address: 0xE0000
    Runtime Size: 128 kB
    ROM Size: 128 kB
    Characteristics:
        ISA is supported
        PCI is supported
        Boot from CD is supported
        Selectable boot is supported
        8042 keyboard services are supported (int 9h)
        CGA/mono video services are supported (int 10h)
        ACPI is supported

--More--
```

7.2.2. Оперативная память

На материнской плате обычно имеются слоты, в которые можно подключить оперативную память (RAM, ОЗУ). 32-битные архитектуры могут использовать до 4 гигабайт ОЗУ, в то время как 64-битные архитектуры способны адресовать и использовать гораздо больше ОЗУ.

В некоторых случаях ОЗУ системы может быть недостаточно для обработки всех требований к операционной системе. Каждая программа должна хранить данные в ОЗУ, а сами программы загружаются в ОЗУ при их выполнении.

Чтобы избежать сбоя системы из-за нехватки ОЗУ, используется виртуальная RAM (или пространство подкачки). Виртуальная оперативная память — это пространство на жестком диске, которое используется для временного хранения данных RAM, когда система закончила работу с ОЗУ. Данные, которые хранятся в ОЗУ и которые не были использованы недавно, копируются на жесткий диск, поэтому активные программы могут использовать ОЗУ. При необходимости эти данные с заменой могут быть снова сохранены в ОЗУ позднее.

Чтобы просмотреть объем оперативной памяти в системе, включая виртуальную RAM, выполняется команда `free`. Команда `free`

имеет параметр `-m` для принудительного округления вывода до ближайшего мегабайта и параметра `-g`, чтобы заставить округление до ближайшего гигабайта:

```
sysadmin@localhost:~$ free -m
              total        used         free       shared    buffers     cached
Mem:           1894         356         1537           0          25         177
-/+ buffers/cache:          153         1741
Swap:          4063           0         4063
sysadmin@localhost:~$
```

Результат выполнения команды `free` показывает, что система, в которой она была выполнена, имеет в общей сложности 1894 мегабайта и в настоящее время использует 356 мегабайт.

Объем подкачки составляет примерно 4 гигабайта, хотя ни один из них, по-видимому, не используется. Это имеет смысл, потому что большая часть физической памяти свободна, поэтому в настоящее время нет необходимости использовать виртуальную память.

7.2.3. Периферийные устройства

Материнская плата имеет шины, которые позволяют подключать несколько устройств к системе, включая межсетевой интерфейс периферийных компонентов (PCI) и универсальную последовательную шину (USB). Материнская плата также имеет разъемы для мониторов, клавиатур и мышей.

Ниже приведен пример вывода команды `lspci`. Как видно ниже в выделенных разделах, эта система имеет контроллер VGA (мониторный разъем), контроллер хранения SCSI (тип жесткого диска) и контроллер Ethernet (сетевой разъем):

```
sysadmin@localhost:~$ lspci
00:00.0 Host bridge: Intel Corporation 440BX/ZX/DX - 82443BX/ZX/DX Host bridge (rev 01)
00:01.0 PCI bridge: Intel Corporation 440BX/ZX/DX - 82443BX/ZX/DX AGP bridge (rev 01)
00:07.0 ISA bridge: Intel Corporation 82371AB/EB/MB PIIX4 ISA (rev 08)
00:07.1 IDE interface: Intel Corporation 82371AB/EB/MB PIIX4 IDE (rev 01)
00:07.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 08)
00:07.7 System peripheral: VMware Virtual Machine Communication Interface (rev 10)
00:0f.0 VGA compatible controller: VMware SVGA II Adapter
03:00.0 Serial Attached SCSI controller: VMware PVSCSI SCSI Controller (rev 02)
0b:00.0 Ethernet controller: VMware VMXNET3 Ethernet Controller (rev 01)
```

Выполнение команды `lspci` с параметром `-nn` показывает как числовой идентификатор для каждого устройства, так и исходное текстовое описание:

```
sysadmin@localhost:~$ lspci -nn
```

```

00:00.0 Host bridge [0600]: Intel Corporation 440BX/ZX/DX - 82443BX/ZX/DX Host bridge
[8086:7190] (rev 01)
00:01.0 PCI bridge [0604]: Intel Corporation 440BX/ZX/DX - 82443BX/ZX/DX AGP bridge
[8086:7191] (rev 01)
00:07.0 ISA bridge [0601]: Intel Corporation 82371AB/EB/MB PIIX4 ISA [8086:7110] (rev
08)
00:07.1 IDE interface [0101]: Intel Corporation 82371AB/EB/MB PIIX4 IDE [8086:7111]
(rev 01)
00:07.3 Bridge [0680]: Intel Corporation 82371AB/EB/MB PIIX4 ACPI [8086:7113] (rev 08)
00:07.7 System peripheral [0880]: VMware Virtual Machine Communication Interface
[15ad:0740] (rev 10)
00:0f.0 VGA compatible controller [0300]: VMware SVGA II Adapter [15ad:0405]
03:00.0 Serial Attached SCSI controller [0107]: VMware PVSCSI SCSI Controller
[15ad:07c0] (rev 02)
0b:00.0 Ethernet controller [0200]: VMware VMXNET3 Ethernet Controller
[15ad:07b0] (rev 01)

```

Выделенный раздел [15ad:07b0] называется секцией [vendor: device].

Использование информации [vendor:device] может быть полезно для отображения подробной информации об определенном устройстве. Используя опцию `vendor:device`, можно выбрать просмотр информации только об одном устройстве.

Также можно просмотреть более подробную информацию, используя опцию `-v`, `-vv` или `-vvv`. Чем больше символов `v`, тем более подробным будет вывод. Например:

```

sysadmin@localhost:~$ lspci -d 15ad:07b0 -vvv
0b:00.0 Ethernet controller: VMware VMXNET3 Ethernet Controller (rev 01)
    Subsystem: VMware VMXNET3 Ethernet Controller
    Physical Slot: 192
    Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Step
ping- SERR- FastB2B- DisINTx+
    Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort-
<MAbort- >SERR- <PERR- INTx-
    Latency: 0, Cache Line Size: 32 bytes
    Interrupt: pin A routed to IRQ 19
    Region 0: Memory at fd4fb000 (32-bit, non-prefetchable) [size=4K]
    Region 1: Memory at fd4fc000 (32-bit, non-prefetchable) [size=4K]
    Region 2: Memory at fd4fe000 (32-bit, non-prefetchable) [size=8K]
    Region 3: I/O ports at 5000 [size=16]
    [virtual] Expansion ROM at fd400000 [disabled] [size=64K]
    Capabilities: <access denied>
    Kernel driver in use: vmxnet3
    Kernel modules: vmxnet3
sysadmin@localhost:~$

```

Команда `lspci` показывает подробную информацию об устройствах, подключенных к системе через шину PCI. Эта информация может быть полезна для определения того, поддерживается ли устройство системой, как указано в драйвере ядра, или модуле ядра, как показано в последних парах строк выше.

7.2.4. Устройства универсальной последовательной шины (USB)

Хотя шина PCI используется для многих внутренних устройств, таких как звуковые и сетевые карты, многие внешние устройства (или периферийные устройства) подключаются к компьютеру через интерфейс USB. Устройства, подключенные к плате внутри, обычно являются «холодным» подключением. Это означает, для подключения или отключения такого устройства требуется выключение компьютера. USB-устройства являются «горячим» подключением, то есть они могут быть подключены или отключены во время работы системы.

Чтобы отобразить устройства, подключенные к системе через USB, следует выполнить команду `lsusb`:

```
sysadmin@localhost:~$ lsusb
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
sysadmin@localhost:~$
```

Вариант `-v` для команды `lsusb` показывает больше информации о каждом устройстве:

```
sysadmin@localhost:~$ lsusb -v

Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Couldn't open device, some information will be missing
Device Descriptor:
  bLength                18
  bDescriptorType         1
  bcdUSB                  1.10
  bDeviceClass             9 Hub
  bDeviceSubClass          0 Unused
  bDeviceProtocol          0 Full speed (or root) hub
  bMaxPacketSize0          64
  idVendor           0x1d6b Linux Foundation

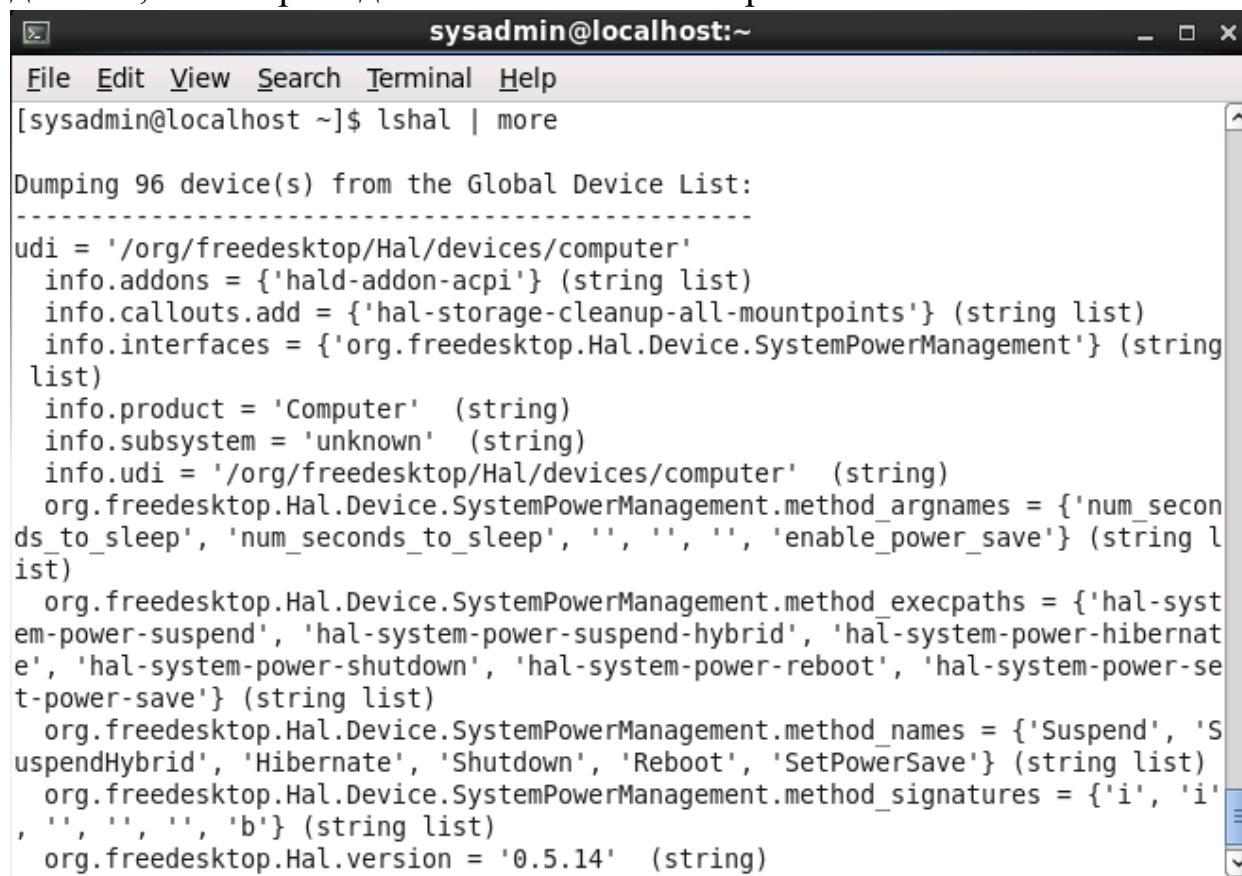
  idProduct           0x0001 1.1 Linux Foundation
  bcDevice             2.06
  iManufacturer          3
  iProduct               2
  iSerial                1
  ...
```

7.3. Слой аппаратных абстракций (HAL)

Hardware Abstraction Layer (HAL) – это слой аппаратных абстракций. HAL позволяет программистам абстрагироваться от конкретного аппаратного обеспечения. Образ для HAL в Linux – это `hald`, процесс, который собирает информацию обо всех устройствах, подключенных к системе. Когда происходят события, которые изменяют состояние

подключенных устройств, например, когда USB-устройство подключено к системе, `hal`d передает эту новую информацию всем процессам, которые зарегистрировались для уведомления о новых событиях.

Команда `lshal` позволяет просматривать устройства, обнаруженные HAL. Эта команда производит огромное количество выходных данных; ниже приводится небольшая выборка:



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ lshal | more  
Dumping 96 device(s) from the Global Device List:  
-----  
udi = '/org/freedesktop/Hal/devices/computer'  
  info.addons = {'hal-addon-acpi'} (string list)  
  info.callouts.add = {'hal-storage-cleanup-all-mountpoints'} (string list)  
  info.interfaces = {'org.freedesktop.Hal.Device.SystemPowerManagement'} (string list)  
  info.product = 'Computer' (string)  
  info.subsystem = 'unknown' (string)  
  info.udi = '/org/freedesktop/Hal/devices/computer' (string)  
  org.freedesktop.Hal.Device.SystemPowerManagement.method_argnames = {'num_seconds_to_sleep', 'num_seconds_to_sleep', '', '', '', 'enable_power_save'} (string list)  
  org.freedesktop.Hal.Device.SystemPowerManagement.method_execpaths = {'hal-system-power-suspend', 'hal-system-power-suspend-hybrid', 'hal-system-power-hibernate', 'hal-system-power-shutdown', 'hal-system-power-reboot', 'hal-system-power-set-power-save'} (string list)  
  org.freedesktop.Hal.Device.SystemPowerManagement.method_names = {'Suspend', 'SuspendHybrid', 'Hibernate', 'Shutdown', 'Reboot', 'SetPowerSave'} (string list)  
  org.freedesktop.Hal.Device.SystemPowerManagement.method_signatures = {'i', 'i', '', '', '', 'b'} (string list)  
  org.freedesktop.Hal.version = '0.5.14' (string)
```

7.4. Дисковые устройства

Дисковые устройства (или жесткие диски) могут быть присоединены к системе несколькими способами; контроллер может быть интегрирован в материнскую плату, на шину PCI, PCI-Express или USB-устройство.

Жесткие диски делятся на разделы. Раздел представляет собой логическое разделение жесткого диска, предназначенное для размещения файлов в пределах ограниченного пространства для хранения. В Microsoft Windows часто имеется один раздел для каждого жесткого диска, в дистрибутивах Linux распространено использовать несколько разделов на жесткий диск.

Существует несколько видов деления дисков на разделы. На более старых жестких дисках используется технология деления, называемая Master Boot Record (MBR). На новые диски рекомендуемым способом деления на разделы является GUID Partition Table (GPT). Тип деления MBR использовался с первых дней существования персонального компьютера (ПК), GPT с 2000 года.

Старый термин, используемый для описания внутреннего жесткого диска, – это «фиксированный диск», поскольку диск зафиксирован внутри корпуса компьютера (не извлекается во время работы компьютера). Этот термин породил несколько имен команд: `fdisk`, `cfdisk` и `sfdisk`, которые являются инструментами для работы с дисками MBR.

На GPT-дисках используется более новый тип разбиения на разделы, что позволяет пользователю разделить диск на большее количество разделов, чем то, что поддерживает MBR. GPT также позволяет иметь разделы, размер которых может быть больше двух терабайт (это ограничение для MBR). Инструменты для управления GPT-дисками называются `gdisk`, `cgdisk` и `sgdisk`.

Существует также множество инструментов, которые пытаются поддерживать диски MBR и GPT. Этот набор инструментов включает в себя команду `parted` и графический инструмент `gparted`.

Жесткие диски связаны с именами файлов (называемыми файлами устройств), которые хранятся в каталоге `/dev`. Для разных типов жестких дисков даются разные имена: `hd` для жестких дисков IDE (Intelligent Drive Electronics) и `sd` для USB, SATA и SCSI.

Каждому жесткому диску назначается буква, например, на первом жестком диске SATA будет указано имя файла устройства `/dev/sda`, а второй жесткий диск SATA должен быть связан с файлом устройства `/dev/sdb`.

Разделам присваиваются уникальные номера для каждого устройства. Например, если жесткий диск `sda` имеет два раздела, они могут быть связаны с файлами устройств `/dev/sda1` и `/dev/sda2`.

На следующем выводе можно увидеть, что система имеет три устройства `sd`: `/dev/sda`, `/dev/sdb` и `/dev/sdc`. Кроме того,

видно, что на первом устройстве есть два раздела (о чем свидетельствуют файлы /dev/sda1 и /dev/sda2) и один раздел на втором устройстве (о чем свидетельствует файл /dev/sdb1):

```
root@localhost:~$ ls /dev/sd*
/dev/sda /dev/sda1 /dev/sda2 /dev/sdb /dev/sdb1 /dev/sdc
root@localhost:~$
```

В следующем примере команда `fdisk` используется для отображения информации о разделе на первом sd-устройстве. Команда требует доступа администратора `root`.

```
root@localhost:~# fdisk -l /dev/sda
Disk /dev/sda: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders, total 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000571a2
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sda1	*	2048	39845887	19921920	83	Linux
/dev/sda2		39847934	41940991	1046529	5	Extended
/dev/sda5		39847936	41940991	1046528	82	Linux swap / Solaris

```
root@localhost:~#
```

Создание и изменение разделов выходит за рамки этого материала.

7.5. Оптические диски

Оптические диски, часто называемые CD-ROM, DVD или Blue-Ray, являются съемными носителями. Некоторые устройства, используемые с оптическими дисками, доступны только для чтения, другие могут записывать диски при использовании записываемого типа диска. Существуют различные стандарты для записываемых и перезаписываемых дисков, таких как CD-R, CD + R, DVD + RW и DVD-RW.

Если съемные диски монтируются в файловой системе, это важно для администратора Linux. Современные дистрибутивы часто монтируют диски в папке `/media`, а старые дистрибутивы обычно монтируют их в папке `/mnt`.

После установки диска большинство графических интерфейсов будут предлагать пользователю предпринять действия, например, открыть содержимое диска в файловом менеджере или запустить медиа-программу. Когда пользователь закончит работу с диском, разумно отключить его с помощью меню или команды извлечения. При нажатии

кнопки извлечения открывается лоток для диска, но некоторые программы «не понимают», что диск больше не монтируется в файловой системе.

7.6. Устройства отображения видео

Чтобы отображать видео (вывод графической информации на экран), компьютерная система должна иметь видеокарту (видеоадаптер) и монитор. Видеоустройства часто подключаются непосредственно к материнской плате, хотя они также могут быть подключены через слоты шины PCI-Express на материнской плате.

К сожалению, с первых дней работы персональных компьютеров стандарт видео не одобрен крупными поставщиками, поэтому для каждого устройства видеодисплея обычно требуется проприетарный драйвер, предоставляемый поставщиком. Драйверы – это программные продукты, которые позволяют операционной системе взаимодействовать с устройством.

Драйверы должны быть написаны для конкретной операционной системы, что обычно делается для Microsoft Windows, но не всегда для Linux. К счастью, крупнейшие производители видеочипов теперь предоставляют по крайней мере некоторый уровень поддержки Linux.

Для того чтобы мониторы работали правильно с устройствами видеоадаптера, они должны иметь возможность поддерживать одинаковое разрешение. Как правило, программное обеспечение, управляющее устройством видеокарты (обычно сервером X.org), может автоматически определять максимальное разрешение, которое может отображать видеокарта и монитор, и устанавливать разрешение экрана на это значение.

Для изменения разрешения, а также максимального количества цветов, которые могут отображаться (так называемая глубина цвета) с используемым дистрибутивом Linux обычно предоставляются специализированные графические инструменты. Для дистрибутивов на основе сервера X.org (почти все дистрибутивы отображающие графический рабочий стол) для изменения разрешения, глубины цвета и других параметров можно использовать файл `/etc/X11/xorg.conf`.

7.7. Управление устройствами

Для того чтобы устройство использовалось в Linux, может потребоваться несколько различных видов программного обеспечения. Прежде всего, есть драйвер. Драйвер может быть скомпилирован как часть ядра Linux, загружен в ядро в виде модуля или загружен пользовательской командой или приложением. Большинство устройств имеют драйвер, либо встроенный в ядро, либо загруженный в ядро, поскольку драйвер может потребовать низкоуровневый вид доступа, который имеет ядро с устройствами.

Внешние устройства, такие как сканеры и принтеры, обычно имеют свои драйверы, загруженные в виде приложения, и эти драйверы, в свою очередь, обмениваются с устройством через интерфейс, такой как USB.

Лучше всего при выборе дистрибутива Linux проверить, сертифицированы ли используемые устройства для работы с этим дистрибутивом. В коммерческих дистрибутивах, таких как Red Hat и SUSE, есть веб-страницы, посвященные перечню оборудования, которое сертифицировано или одобрено для работы.

Перед покупкой новых устройств можно обратиться к документации производителя, чтобы узнать, поддерживают ли они Linux.

Контрольные вопросы

1. Какие псевдофайловые системы монтирует ядро Linux для обеспечения доступа к информации об аппаратных устройствах, подключенных к системе?
2. Из какого типичного набора аппаратных средств состоит персональный компьютер?
3. Какими командами можно отобразить информацию об аппаратных средствах компьютера?
4. Что такое драйвер устройства?

Глава 8. УПРАВЛЕНИЕ ПАКЕТАМИ И ПРОЦЕССАМИ

8.1. Управление пакетами

Управление пакетами – это система, посредством которой программное обеспечение может быть установлено, обновлено, запрошено или удалено из файловой системы. В Linux существует множество различных систем управления пакетами программного обеспечения, но двумя наиболее популярными являются те, что от Debian и Red Hat.

8.1.1. Управление пакетами Debian

Дистрибутив Debian и его производные, такие как Ubuntu, Mint, Astra Linux используют систему управления пакетами Debian. В основе решения дистрибутивов Debian лежат пакеты программного обеспечения, которые распространяются как файлы, заканчивающиеся на «.deb».

Инструментом самого низкого уровня для управления этими файлами является команда `dpkg`. Эта команда может быть сложной для начинающих пользователей Linux, поэтому был разработан инструмент Advanced Package Tool (`apt`), интерфейсная программа – надстройка для инструмента `dpkg`, упрощает управление пакетами. Существуют другие инструменты командной строки, которые служат в качестве интерфейсов для `dpkg`, таких как `aptitude`, а также интерфейсы GUI, такие как `synaptic` и `software-center`.

8.1.1.1. Debian – Добавление пакетов

Репозитории Debian – серверы, содержащие десятки тысяч различных пакетов программного обеспечения. Чтобы получить обновленный список из репозитория, можно выполнить команду `sudo apt update`. Список используемых репозитория зависит от дистрибутива, но может быть изменен.

Чтобы искать ключевые слова в этих пакетах, можно использовать команду поиска `sudo apt search keyword`.

После того, как найден желаемый пакет, его можно установить с помощью команды `sudo apt install package`.

Репозитории по умолчанию располагаются на серверах в сети Интернет. Администратор может создать в локальной сети свою собственную копию репозитория и настроить компьютеры с установленной операционной системой Linux направив на этот сервер. В последнем случае соединение с сетью Интернет не требуется.

8.1.1.2. Debian – Обновление пакетов

Для того чтобы обновить отдельный пакет, следует выполнить команду для этого пакета: `sudo apt update имя_программы`

Если устаревшая версия пакета уже установлена, она будет обновлена. В противном случае произойдет новая установка.

Чтобы обновить все возможные пакеты, после команды `sudo apt update` следует выполнить команду `sudo apt upgrade`.

Пользователи, которые входят в систему с графическим интерфейсом, могут получить сообщение в области уведомлений, указывающее, что доступны обновления, как показано на рисунке 6.

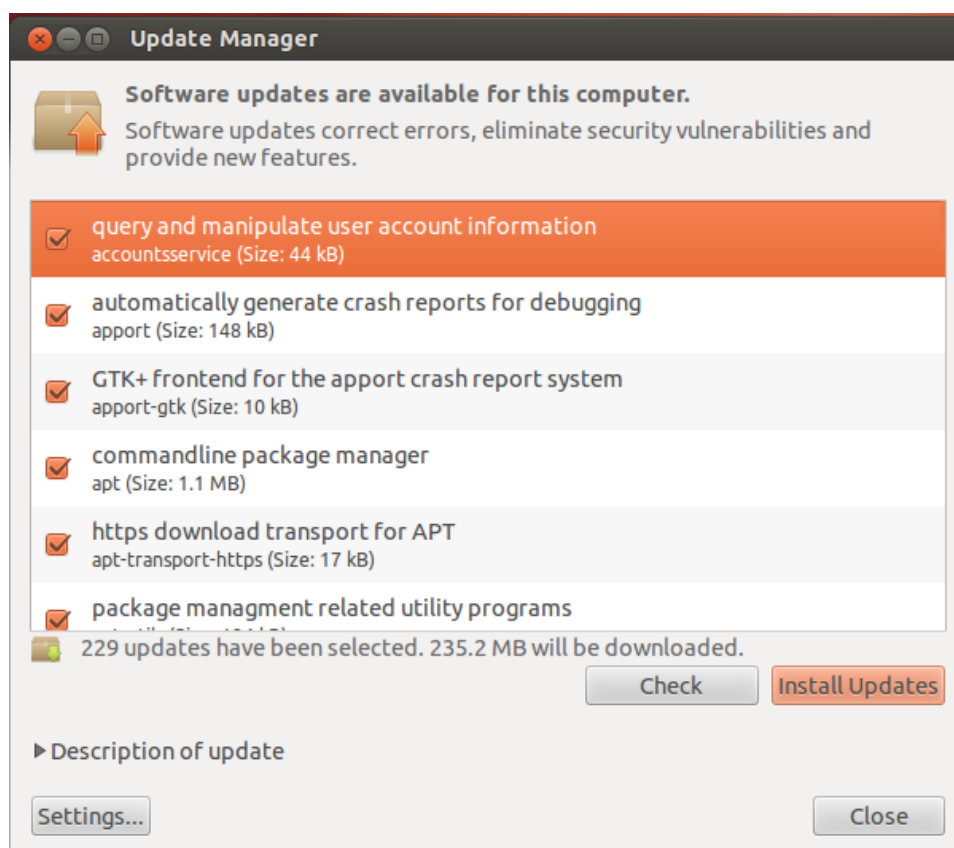


Рис. 6. Менеджер обновлений

8.1.1.3. Debian – Удаление пакетов

Следует помнить, что удаление одного пакета программного обеспечения может привести к удалению других пакетов. Из-за зависимостей между пакетами при удалении пакета все пакеты, которые нуждаются в нем или зависят от него, также будут удалены.

Для того чтобы удалить все файлы программного пакета, за исключением файлов конфигурации, следует выполнить команду `sudo apt remove имя_пакета`.

Чтобы удалить все файлы программного пакета, включая файлы конфигурации, следует выполнить команду `sudo apt --purge remove имя_пакета`.

Иногда требуется сохранить файлы конфигурации – в том случае, если планируется позднее установить пакет программного обеспечения заново.

8.1.1.4. Debian – Запрос пакетов

Существует несколько различных запросов, которые должны использовать администраторы. Чтобы получить список всех пакетов, которые в настоящее время установлены в системе, можно выполнить команду `dpkg -l`.

Чтобы просмотреть файлы, содержащие конкретный пакет, можно выполнить команду `dpkg -L имя_пакета`.

Чтобы запросить информацию или состояние пакета, следует использовать команду `dpkg -s имя_пакета`.

Чтобы определить, был ли конкретный файл помещен в файловую систему в результате установки пакета, следует использовать команду `dpkg -S /путь/к/файлу`. Например:

```
sysadmin@localhost:~$ dpkg -S /usr/bin/who
coreutils: /usr/bin/who
```

В примере показан запрос информации о файле `/usr/bin/who`, который является частью пакета `coreutils`.

8.1.2. Управление пакетами RPM

База стандартов Linux, которая является проектом Linux Foundation, предназначена для определения (посредством консенсуса) набора стандартов, повышающих совместимость между соответствующими системами Linux. В соответствии с базой стандартов Linux стандартная система управления пакетами – это RPM.

RPM использует файл `.rpm` для каждого пакета программного обеспечения. Эту систему используют дистрибутивы Red Hat (например, Red Hat, CentOS и Fedora) для управления программным обеспечением. Кроме того, несколько других дистрибутивов, не входящих в Red Hat (например, SUSE, OpenSUSE и Mandriva), также используют RPM.

Как и система Debian, системы управления пакетами RPM отслеживают зависимости между пакетами. Отслеживание зависимостей гарантирует, что при установке пакета система также будет устанавливать любые пакеты, необходимые для этого пакета, для правильной работы. Зависимости также обеспечивают правильное выполнение обновлений и удаления программного обеспечения.

Инструментом, наиболее часто используемым для управления пакетами RPM, является команда `rpm`. Хотя команда `rpm` может устанавливать, обновлять, запрашивать и удалять пакеты, средства командной строки, такие как `yum` и `urpdate`, автоматизируют процесс решения проблем с зависимостями и упрощают работу.

Кроме того, есть интерфейсные инструменты на основе графического интерфейса, такие как `yumex` и `gpk-application`, которые также упрощают управление пакетами RPM.

Следует отметить, что для многих из следующих команд потребуются привилегии супер-администратора `root`. Эмпирическое правило состоит в том, что если команда влияет на состояние пакета, необходимо иметь административный доступ. Другими словами, обычный пользователь может выполнить запрос или поиск, но для добавления, обновления или удаления пакета требуется, чтобы команда выполнялась от лица пользователя `root`.

8.1.2.1. RPM – Добавление пакетов

Чтобы выполнить поиск пакета из сконфигурированных репозиториев, следует выполнить команду `yum` с ключевым словом.

Чтобы установить пакет вместе со своими зависимостями, следует выполнить команду `yum install имя_пакета`.

8.1.2.2. RPM – Обновление пакетов

Для того чтобы обновить отдельный программный пакет, можно выполнить команду `yum update имя_пакета`.

Чтобы обновить все пакеты, следует выполнить команду `yum update`.

Если обновления доступны и пользователь использует графический интерфейс, тогда на экран могут отображаться сообщения в области уведомлений, указывающие, что доступны обновления. Пример уведомления показан на рисунке 7.

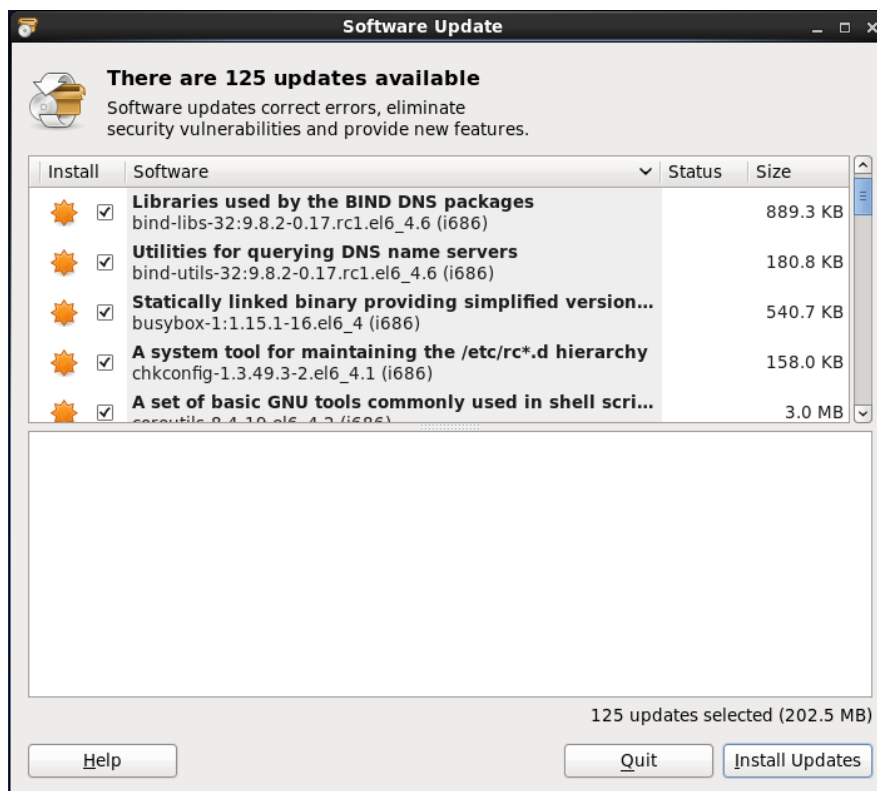


Рис. 7. Доступные обновления

8.1.2.3. RPM – Удаление пакетов

Как и в случае любой системы управления пакетами, которая отслеживает зависимости, если нужно удалить один пакет, тогда можно удалить несколько из-за зависимостей. Самый простой способ автоматического разрешения проблем с зависимостями – использовать команду `yum`:

```
yum remove имя_пакета
```

Можно удалить программные пакеты с помощью команды `rpm`, но она не будет автоматически удалять пакеты зависимостей.

8.1.2.4. RPM – Запрос пакетов

Управление пакетами Red Hat аналогично управлению пакетами Debian, когда дело доходит до выполнения запросов. Лучше всего использовать `rpm`, вместо `yum`. Хотя интерфейсные инструменты могут выполнять некоторые из этих запросов, производительность страдает, потому что эти команды обычно подключаются к нескольким репозиториям по сети при выполнении любой команды. Команда `rpm` выпол-

няет свои запросы, подключаясь к базе данных, которая является локальной для машины, и не подключается по сети к каким-либо репозиториям.

Чтобы получить список всех пакетов, которые в настоящее время установлены в системе, можно выполнить команду `rpm -qa`.

Чтобы просмотреть файлы, содержащие конкретный пакет, можно выполнить команду `rpm -ql package`.

Чтобы запросить информацию о пакете или его состоянии, можно выполнить команду `rpm -qi package`.

Чтобы определить, был ли конкретный файл помещен в файловую систему в результате установки пакета, можно выполнить команду `rpm -qf /путь/к/файлу`.

8.2. Ядро Linux

Когда большинство людей говорят о Linux, они ссылаются на GNU / Linux, который определяет операционную систему. Часть Gnu's Not Unix (GNU) этой комбинации предоставляется проектом Free Software Foundation. GNU – это то, что обеспечивает эквиваленты с открытым исходным кодом многих распространенных команд UNIX, основная часть необходимых команд командной строки. Часть Linux этой комбинации – это ядро Linux, которое является ядром операционной системы. Ядро загружается во время загрузки и остается загруженным для управления всеми аспектами работающей системы.

Реализация ядра Linux включает в себя множество подсистем, которые являются или частью самого ядра и или внешними, которые могут быть загружены модульным способом, когда это необходимо. Некоторые из ключевых функций ядра Linux включают в себя интерфейс системного вызова, управление процессами, управление памятью, виртуальную файловую систему, сетевые устройства и драйверы устройств.

Вкратце, ядро принимает команды от пользователя и управляет процессами, выполняющими эти команды, предоставляя им доступ к таким устройствам, как память, диски, сетевые интерфейсы, клавиатуры, мыши, мониторы и т.д.

Ядро предоставляет доступ к информации о запуске процессов через псевдо-файловую систему, которая видна в каталоге `/proc`. Аппаратные устройства доступны через специальные файлы в каталоге `/dev`, а информацию об этих устройствах можно найти в другой псевдо-файловой системе в каталоге `/sys`.

Каталог `/proc` содержит информацию не только о запущенных процессах, но также об аппаратном обеспечении системы и текущей конфигурации ядра. Пример вывода ниже:

```
sysadmin@localhost:~$ ls /proc
1          cpuinfo      irq          modules     sys
128        crypto        kallsyms     mounts      sysrq-trigger
17         devices     kcore        mtrr        sysvipc
21         diskstats   key-users    net          thread-self
23         dma         keys         pagetypeinfo timer_list
39         driver      kmsg         partitions  timer_stats
60         execdomains kpagecgroup  sched_debug  tty
72         fb          kpagecount   schedstat    uptime
acpi       filesystems kpageflags   scsi         version
buddyinfo  fs              loadavg      self         version_signature
bus        interrupts    locks        slabinfo     vmallocinfo
cgroups    iomem          mdstat       softirqs     vmstat
cmdline    ioports        meminfo      stat         zoneinfo
consoles   ipmi           misc         swaps
```

Результат выполнения `ls /proc` показывает нумерованные каталоги. Для каждого запущенного процесса в системе есть нумерованный каталог, где имя каталога соответствует идентификатору PID (идентификатор процесса) для текущего процесса.

Поскольку процесс `/sbin/init` всегда является первым процессом, он имеет PID 1, а информация о процессе `/sbin/init` может быть найдена в каталоге `/proc/1`. Есть несколько команд, которые позволяют просматривать информацию о запущенных процессах, поэтому пользователям крайне редко приходится просматривать файлы для каждого запущенного процесса напрямую.

Также можно увидеть, что существует ряд обычных файлов в директории `/proc`, таких как `/proc/cmdline`, `/proc/meminfo` и `/proc/modules`. Эти файлы предоставляют информацию о запущенном сервере:

- `/proc/cmdline` файл содержит всю информацию, которая была передана ядру при первом запуске.
- `/proc/meminfo` файл содержит информацию об использовании памяти ядром

- `/proc/modules` файл содержит список модулей, загруженных в ядро для добавления дополнительных функций.

Опять же, редко приходится просматривать эти файлы напрямую, так как есть команды, предлагающие более «удобный» вывод и альтернативный способ просмотра этой информации.

Хотя большинство «файлов» под каталогом `/proc` не могут быть изменены, даже пользователем `root`, «файлы» под каталогом `/proc/sys` могут быть изменены пользователем `root`. Изменение этих файлов изменит поведение ядра Linux.

Прямая модификация этих файлов вызывает только временные изменения ядра. Чтобы сделать изменения постоянными, записи могут быть добавлены в файл `/etc/sysctl.conf`.

Например, каталог `/proc/sys/net/ipv4` содержит файл с именем `icmp_echo_ignore_all`. Если этот файл содержит ноль 0, как обычно, то система будет отвечать на запросы `icmp`. Если этот файл содержит один 1, система не будет отвечать на запросы `icmp`:

```
sysadmin@localhost:~$ su -
Password:
root@localhost:~# cat /proc/sys/net/ipv4/icmp_echo_ignore_all
0
root@localhost:~# ping -c1 localhost
PING localhost.localdomain (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost.localdomain (127.0.0.1): icmp_seq=1 ttl=64 time=0.026 ms

--- localhost.localdomain ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.026/0.026/0.026/0.000 ms
root@localhost:~# echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all
root@localhost:~# ping -c1 localhost
PING localhost.localdomain (127.0.0.1) 56(84) bytes of data.

--- localhost.localdomain ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 1000
```

8.3. Иерархия процессов

Когда ядро завершает операции во время процедуры загрузки, оно запускает процесс `/sbin/init` и присваивает ему идентификатор процесса (PID) 1. Затем этот процесс запускает другие системные процессы, и каждому процессу назначается PID в последовательном порядке.

Поскольку процесс `/sbin/init` запускает другие процессы, они, в свою очередь, могут запускать процессы, которые могут запускать другие процессы, и далее. Когда один процесс запускает другой процесс, процесс, который выполняет запуск, называется родительским процессом, и процесс, который запускается, называется дочерним процессом. При просмотре процессов родительский PID будет помечен как PPID.

Когда система работает в течение длительного времени, она в конечном итоге достигнет максимального значения PID, которое можно просмотреть и настроить через файл `/proc/sys/kernel/pid_max`. После того, как используется самый большой PID, система «перевернется» и возобновится, назначая значения PID, доступные в нижней части диапазона.

Можно «отобразить» процессы в виде генеалогического дерева родительских и дочерних связей командой `ps tree`:

```
sysadmin@localhost:~$ ps tree
init--+-cron
      |-login---bash---ps tree
      |-named---18*[{named}]
      |-rsyslogd---2*[{rsyslogd}]
      `--sshd
```

Если необходимо изучить отношения родительского и дочернего процессов, используя вывод предыдущей команды, можно полагать это таким, как показано на рисунке 8:

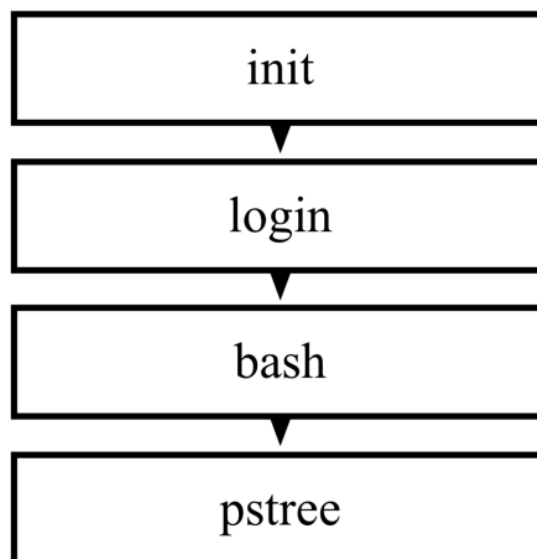


Рис. 8. Последовательность запуска процессов

8.4. Команда ps (Process)

Другим способом просмотра процессов является команда `ps`. По умолчанию команда `ps` покажет только текущие процессы, запущенные в текущей оболочке:

```
sysadmin@localhost:~$ ps
  PID TTY          TIME CMD
 6054 ?            00:00:00 bash
 6070 ?            00:00:01 xeyes
 6090 ?            00:00:01 firefox
 6146 ?            00:00:00 ps
sysadmin@localhost:~$
```

Подобно команде `pstree`, если запустить `ps` с параметром `--forest`, она отобразит строки, указывающие родительские и дочерние отношения:

```
sysadmin@localhost:~$ ps --forest
  PID TTY          TIME CMD
 6054 ?            00:00:00 bash
 6090 ?            00:00:02  \_ firefox
 6180 ?            00:00:00   \_ dash
 6181 ?            00:00:00    \_ xeyes
 6188 ?            00:00:00     \_ ps
sysadmin@localhost:~$
```

Чтобы иметь возможность просматривать все процессы в системе, можно выполнить команду `ps aux` или команду `ps -ef`:

```
sysadmin@localhost:~$ ps aux | head
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1   0.0   0.0  17872  2892 ?        Ss   08:06   0:00 /sbin?? /ini
syslog    17   0.0   0.0 175744  2768 ?        Sl   08:06   0:00 /usr/sbin/rsyslogd
-c5
root      21   0.0   0.0  19124  2092 ?        Ss   08:06   0:00 /usr/sbin/cron
root      23   0.0   0.0  50048  3460 ?        Ss   08:06   0:00 /usr/sbin/sshd
bind      39   0.0   0.0 385988 19888 ?        Ssl  08:06   0:00 /usr/sbin/named -u
bind
root      48   0.0   0.0  54464  2680 ?        S    08:06   0:00 /bin/login -f
sysadmin  60   0.0   0.0  18088  3260 ?        S    08:06   0:00 -bash
sysadmin 122   0.0   0.0  15288  2164 ?        R+   16:26   0:00 ps aux
sysadmin 123   0.0   0.0  18088   496 ?        D+   16:26   0:00 -bash
sysadmin@localhost:~$ ps -ef | head
UID        PID  PPID  C  STIME TTY          TIME CMD
root         1    0  0 08:06 ?           00:00:00 /sbin?? /init
syslog    17    1  0 08:06 ?           00:00:00 /usr/sbin/rsyslogd -c5
root      21    1  0 08:06 ?           00:00:00 /usr/sbin/cron
root      23    1  0 08:06 ?           00:00:00 /usr/sbin/sshd
bind      39    1  0 08:06 ?           00:00:00 /usr/sbin/named -u bind
root      48    1  0 08:06 ?           00:00:00 /bin/login -f
sysadmin   60   48  0 08:06 ?           00:00:00 -bash
sysadmin  124   60  0 16:46 ?           00:00:00 ps -ef
sysadmin  125   60  0 16:46 ?           00:00:00 head
sysadmin@localhost:~$
```

Вывод всех процессов, работающих в системе, определенно может быть очень большим. В приведенном примере вывод команды `ps` был отфильтрован головной командой, поэтому были показаны только первые десять процессов.

Общим способом запуска команды `ps` является использование команды `grep` для фильтрации выходных строк отображения, соответствующих ключевому слову, например, имени процесса. Чтобы просмотреть информацию об браузере `firefox`, можно выполнить команду типа:

```
sysadmin@localhost:~$ ps -e | grep firefox
6090 pts/0    00:00:07 firefox
```

Пользователю `root` следует быть более обеспокоенным процессами другого пользователя. Из-за нескольких стилей опций, поддерживаемых командой `ps`, существуют разные способы просмотра процессов отдельного пользователя. Используя традиционную опцию UNIX, чтобы просмотреть процессы пользователя `sysadmin`, можно выполнить следующую команду:

```
[root@localhost ~]# ps -u username
```

Или в стиле BSD-опций:

```
[root@localhost ~]# ps u U username
```

8.5. Команда `top`

По умолчанию вывод команды `top` выводит информацию о процессах сортируя %-ом от времени процессора, которое в настоящее время использует каждый процесс. Процессы, которые значительно нагружают центральный процессор перечислены первыми:

```
top - 16:58:13 up 26 days, 19:15,  1 user,  load average: 0.60, 0.74, 0.60
Tasks:  8 total,   1 running,   7 sleeping,   0 stopped,   0 zombie
Cpu(s):  6.0%us,  2.5%sy,  0.0%ni, 90.2%id,  0.0%wa,  1.1%hi,  0.2%si,  0.0%st
Mem:  32953528k total, 28126272k used,  4827256k free,    4136k buffers
Swap:      0k total,        0k used,        0k free, 22941192k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	17872	2892	2640	S	0	0.0	0:00.02	init
17	syslog	20	0	171m	2768	2392	S	0	0.0	0:00.20	rsyslogd
21	root	20	0	19124	2092	1884	S	0	0.0	0:00.02	cron
23	root	20	0	50048	3460	2852	S	0	0.0	0:00.00	sshd
39	bind	20	0	376m	19m	6100	S	0	0.1	0:00.12	named
48	root	20	0	54464	2680	2268	S	0	0.0	0:00.00	login
60	sysadmin	20	0	18088	3260	2764	S	0	0.0	0:00.01	bash
127	sysadmin	20	0	17216	2308	2072	R	0	0.0	0:00.01	top

Существует обширный список команд, которые могут выполняться из текущей программы `top`, как показано в таблице 15.

Таблица 15

Клавиша	Значение
h или ?	Помощь
L	Переключить статистику загрузки
T	Переключить статистику времени
M	Переключить статистику использования памяти
<	Переместить отсортированный столбец влево
>	Переместить отсортированный столбец вправо
F	Выбрать отсортированное поле
R	Переключить направление сортировки
P	Отсортировать по % CPU
M	Отсортировать по % использования памяти
K	Убить процесс (или отправить ему сигнал)
R	Уточнить приоритет процесса

Одним из преимуществ команды `top` является то, что ее можно оставить работать, чтобы оставаться на «вершине» процессов для целей мониторинга. Если процесс начинает доминировать или бесконечно загружать систему, то он по умолчанию будет отображаться в верхней части списка, представленного командой `top`. Администратор, который запускает команду `top`, может выполнить одно из двух действий:

- Завершить процесс: нажатие клавиши `k` во время работы команды `top` предложит пользователю указать PID, а затем номер сигнала. Отправка сигнала по умолчанию потребует завершения процесса, но передача сигнала `9`, сигнал `KILL`, заставит процесс завершить работу.
- Изменить приоритет процесса: нажатие клавиши `r` во время работы команды `top` приведет к тому, что пользователь введет новое значение для приоритета. Значения могут варьироваться от -20 до 19 и влиять на приоритет.

Другим преимуществом команды `top` является то, что она может дать общее представление о том, насколько занята система в настоящее время и вывести общий тренд с течением времени. Средние значения нагрузки, показанные в первой строке вывода команды `top`, показывают, насколько занята система в течение последней минуты, пяти и пятнадцати минут. Эта информация также может быть просмотрена

путем выполнения команды `uptime` или непосредственно путем отображения содержимого файла `/proc/loadavg`:

```
sysadmin@localhost:~$ cat /proc/loadavg
0.12 0.46 0.25 1/254 3052
```

Первые три цифры в этом файле показывают среднее значение нагрузки за последнюю минуту, пять и пятнадцать минут. Четвертое значение представляет собой долю, которая показывает количество процессов, выполняемых в настоящее время в ЦП 1, и общее количество процессов 254. Пятое значение является последним значением PID, выполняющим код на ЦП.

Число, сообщаемое как среднее значение нагрузки, пропорционально количеству ядер ЦП, способных выполнять процессы. На одном центральном процессоре значение одного означает, что система полностью загружена. На четырехъядерном процессоре значение одного означает, что система загружается только на 1/4 или 25%.

Другая причина, по которой администраторы любят команду `top`, – это возможность отслеживать использование памяти в режиме реального времени. В командах `top` и `free` отображаются статистические данные о том, как используется общая память.

Команда `top` также имеет возможность отображать процент памяти, используемый каждым процессом, поэтому процесс, который потребляет чрезмерное количество памяти, может быть быстро идентифицирован.

8.6. Команда `free`

Выполнение команды `free` без каких-либо параметров обеспечивает моментальный снимок информации об используемой в этот момент памяти.

Для отслеживания использования памяти с течением времени с помощью команды `free`, можно выполнить ее с опцией `-s` и указать количество секунд. Например, выполнение `free -s 10` будет обновлять вывод каждые десять секунд.

Чтобы упростить интерпретацию того, что выводит команда `free` используются параметры `-m` или `-g`, показывающие информацию в мегабайтах или гигабайтах соответственно. Без этих параметров вывод отображается в байтах:

```

sysadmin@localhost:~$ free
              total        used        free      shared    buffers     cached
Mem:          32953528    26171772    6781756           0        4136    22660364
-/+ buffers/cache:    3507272    29446256
Swap:           0           0           0
sysadmin@localhost:~$

```

При чтении вывода команды `free`:

- Первая строка – описательный заголовок.
- Вторая строка с надписью `Mem` – статистика для физической памяти системы.
- Третья строка представляет количество физической памяти после корректировки этих значений, не принимая во внимание память, которая используется ядром для буферов и кешей. Технически эта «используемая» память может быть «восстановлена», если это необходимо.
- Четвертая строка вывода относится к памяти `Swap`, также известной как виртуальная память или пространство подкачки. Это пространство на жестком диске, которое используется как физическая память, когда количество физической памяти становится низким. Фактически, это заставляет казаться, что система имеет больше памяти, чем есть на самом деле, но использование пространства подкачки также может замедлить работу системы.

Если объем доступной памяти, включая пространство подкачки становится очень малым, система начнет автоматически завершать процессы. Это одна из причин, по которой важно контролировать использование памяти системы. Администратор, который отмечает, что система становится бедной на свободную память, может использовать команду `top` или `kill` для прекращения процессов по своему выбору.

8.7. Лог-файлы

Когда ядро и различные процессы запускаются в системе, они создают выходные данные, описывающие, как они работают. Некоторые из этих результатов отображаются в окне терминала, где процесс был выполнен, но некоторые из этих данных не отправляются на экран, а вместо этого записываются в различные файлы. Это называется «данные журнала» или «сообщения журнала».

Эти файлы журналов очень важны по ряду причин; они могут быть полезны при устранении проблем, и их можно использовать для

определения того, были ли предприняты попытки несанкционированного доступа.

Некоторые процессы могут «записывать» свои собственные данные в эти файлы, другие процессы полагаются на сторонний процесс («демон», служба в терминологии Microsoft Windows) для обработки файлов журнала.

Демоны регистрации могут варьироваться от одного дистрибутива к другому. Например, в некоторых дистрибутивах демоны, выполняемые в фоновом режиме для ведения журнала, называются `syslogd` и `klogd`. В других дистрибутивах эта функция регистрации может служить одним демоном, таким как `rsyslogd` в Red Hat и Centos или `systemd-journald` в Fedora.

Независимо от того, как назван процесс демона, сами файлы журнала почти всегда помещаются в структуру каталога `/var/log`. Хотя некоторые имена файлов могут различаться, некоторые из наиболее распространенных файлов, которые можно найти в этом каталоге показаны в таблице 16.

Таблица 16

Файл	Содержимое
boot.log	Сообщения, созданные как службы, запускаются во время запуска системы.
cron	Сообщения, генерируемые демоном <code>crond</code> для заданий, которые будут выполняться на постоянной основе.
dmesg	Сообщения, сгенерированные ядром во время загрузки системы.
maillog	Сообщения, отправленные почтовым демоном для отправленных или полученных сообщений электронной почты
messages	Сообщения из ядра и другие процессы, которые не принадлежат ничему. Иногда называется <code>syslog</code> после демона, который записывает этот файл.
secure	Сообщения от процессов, требующих авторизации или аутентификации (например, процесс входа в систему).
Xorg.0.log	Сообщения с сервера X-Window (GUI).

Файлы журнала ротируются, что означает, что старые файлы журналов переименовываются и заменяются новыми файлами журнала. Имена файлов, отображаемые в приведенной выше таблице, могут содержать добавленный к ним числовой или датированный суффикс, например: `secure.0` или `secure-20221103`.

Ротация файла журнала обычно происходит на регулярной основе, например, один раз в неделю. Когда файл журнала ротируется, система перестает записываться в файл журнала и добавляет к нему суффикс. Затем создается новый файл с исходным именем, и процесс ведения журнала продолжается с использованием этого нового файла.

Хотя большинство файлов журналов содержат текст в качестве своего содержимого, которое можно безопасно просматривать при помощи многих инструментов, другие файлы, такие как `/var/log/btmp` и `/var/log/wtmp`, содержат двоичные данные. С помощью команды `file` можно проверить тип содержимого файла перед его просмотром, чтобы убедиться, что он безопасен для просмотра.

Для файлов, содержащих двоичные данные, обычно доступны специальные команды для интерпретации данных и вывода их для человека в виде текста.

По соображениям безопасности большинство файлов журналов не читаются обычными пользователями, поэтому выполнять команды, которые взаимодействуют с этими файлами, необходимо с привилегиями суперадминистратора `root`.

8.8. Команда `dmesg`

Файл `/var/log/dmesg` содержит сообщения ядра, которые были созданы во время запуска системы. Файл `/var/log/messages` содержит сообщения ядра, которые создаются при запуске системы, но эти сообщения будут смешиваться с другими сообщениями от демонов или процессов.

Хотя ядро не имеет собственного файла журнала, их обычно можно настроить путем изменения файла `/etc/syslog.conf` или `/etc/rsyslog.conf`. Кроме того, команда `dmesg` может использоваться для просмотра кольцевого буфера ядра, который будет содержать большое количество сообщений, генерируемых ядром.

В активной системе или в случае нескольких ошибок ядра емкость этого буфера может быть превышена, и некоторые сообщения могут быть потеряны. Размер этого буфера устанавливается во время компилирования ядра.

Выполнение команды `dmesg` может генерировать до 512 килобайт текста, поэтому рекомендуется фильтровать команду с помощью

less или grep. Например, если возникли проблемы с USB-устройством, поиск текста «USB» с командой grep может оказаться полезным:

```
sysadmin@localhost:~$ dmesg | grep -i usb
usbcore: registered new interface driver usbfs
usbcore: registered new interface driver hub
usbcore: registered new device driver usb
ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI) Driver
ohci_hcd: USB 1.1 'Open' Host Controller (OHCI) Driver
ohci_hcd 0000:00:06.0: new USB bus registered, assigned bus number 1
usb usb1: New USB device found, idVendor=1d6b, idProduct=0001
usb usb1: New USB device strings: Mfr=3, Product=2, SerialNumber=1
```

Контрольные вопросы

1. Какой идентификатор процесса (PID) у процесса инициализации (init или systemd)?
2. О чем выводит статистику команда free?
3. Каково предназначение менеджера пакетов?
4. Какой менеджер пакетов используется в дистрибутивах на основе RedHat?
5. Какой менеджер пакетов используется в дистрибутивах на основе Debian?
6. Какой каталог содержит информацию о запущенных процессах и текущей конфигурации ядра?
7. Какими командами можно вывести дерево запущенных процессов?
8. Что такое log-файлы?
9. Какой командой выводится информация о системе, аналогичная выводу диспетчера задач в системе Windows?

Глава 9. КОНФИГУРАЦИЯ СЕТИ

9.1. Базовая сетевая терминология

Доступ к сети является ключевой особенностью большинства Linux-систем. Пользователи хотят пользоваться ресурсами сети, отправлять и получать электронную почту и передавать файлы другим пользователям и т.д.

Обычно программы, выполняющие эти функции (веб-браузеры, почтовые клиенты и т.д.), просты в использовании. Тем не менее, все они полагаются на важную функцию: способность компьютера к взаимодействию с другим компьютером. Чтобы иметь эту способность, необходимо знать, как настраивать сеть системы.

Host (Хост): Хост – это оконечное устройство, в основном компьютер. Однако многие люди имеют ограниченное представление о том, что такое компьютер (например, настольный компьютер или ноутбук). Многие другие устройства, подключаемые к сети, такие как сотовые телефоны, планшеты, современные телевизоры, IP камеры, принтеры тоже являются оконечными устройствами. В сетевых терминах хост – это любое устройство, которое взаимодействует с другим устройством.

Network (Сеть): Сеть представляет собой набор из двух или более хостов (компьютеров), которые могут общаться друг с другом. Эта связь может осуществляться через проводное или беспроводное соединение.

Internet (Интернет): Интернет является примером сети. Интернет состоит из общедоступной сети, которая соединяет миллионы хостов по всему миру. Многие люди используют Интернет для просмотра веб-страниц и отправки / получения электронной почты, но в Интернете есть много дополнительных возможностей, помимо этих действий.

Wi-Fi: Термин Wi-Fi относится к беспроводным сетям.

Server (Сервер): Хост, предоставляющий услугу другому хосту или клиенту, называется сервером. Например, веб-сервер хранит, обрабатывает и доставляет веб-страницы. Сервер электронной почты получает входящую почту и отправляет исходящую почту.

Service (Служба): Особенностью, предоставляемой хостом, является служба. Примером службы может быть, когда хост предоставляет веб-страницы другому хосту.

Client (Клиент): Клиент – это хост, который обращается к серверу. Когда пользователь работает на компьютере, занимающемся серфингом в Интернете, он считается находящимся на клиентском хосте.

Router (Маршрутизатор): Маршрутизатор, называемый также шлюзом, является машиной, которая соединяет хосты из одной сети в другую. Например, если пользователь работает в офисной среде, компьютеры внутри компании могут взаимодействовать через локальную сеть, созданную администраторами. Для доступа в Интернет компьютеры должны будут взаимодействовать с маршрутизатором, который будет использоваться для пересылки сетевых сообщений в Интернет. Обычно, когда пользователи общаются в большой сети (например, в Интернете), существует несколько маршрутизаторов, которые используются до того, как сообщение пользователя достигнет конечного адресата.

На рисунке 9 диаграмма визуализирует часть терминов, рассмотренных выше.

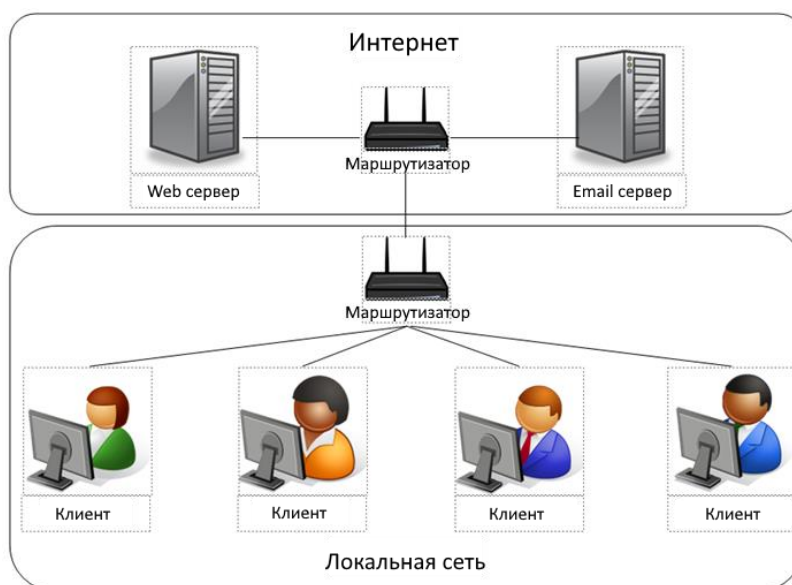


Рис. 9. Сетевые устройства

9.2. Терминология сетевых функций

Network packet (Сетевой пакет): Сетевой пакет – неделимая часть сообщения, используемая для передачи данных между хостами.

Деление сообщений на более мелкие куски (пакеты) делает способ доставки данных намного эффективнее. Пакет помимо передаваемой информации имеет заголовок, в котором указываются адреса отправителя и получателя, а также другая служебная информация.

IP address (IP-адрес): IP-адрес - это уникальный номер, назначенный хосту в сети. Хосты используют эти номера в качестве адреса отправителя и получателя пакетов.

Network mask (Маска сети): называемая сетевой маской. Маска – это числовая система, которая может использоваться для определения границ отдельной сети. По маске определяется какие IP-адреса считаются находящимися в локальной сети, а какие во внешней.

Hostname (Имя хоста): Каждый хост в сети может иметь свое собственное имя. Это облегчает людям обращение к сетевым ресурсам на других хостах, потому что людям легче запоминать имена устройств, а не их номера. Имена преобразуются в IP-адреса до того, как сетевой пакет будет отправлен в сеть.

DHCP: Хостам могут быть автоматически назначены имена, IP-адреса и другая связанная с сетью информация сервером DHCP (Dynamic Host Configuration Protocol). В мире компьютеров протокол – это четко определенный набор правил. DHCP определяет, как сетевая информация назначается хостам клиента, а DHCP-сервер – это машина, которая предоставляет эту информацию.

DNS: Имена хостов транслируются в IP-адреса, прежде чем сетевой пакет будет отправлен в сеть. Это означает, что хост должен знать IP-адрес всех других хостов, с которыми общается пользователь. При работе в большой сети (например, в Интернете) это может создать проблему, так как существует так много хостов. Сервер DNS (Domain Name Server) предоставляет услугу перевода доменных имен в IP-адреса.

Ethernet: В проводной сетевой среде Ethernet является наиболее распространенным способом физического подключения хостов к сети. Кабели Ethernet подключаются к сетевым картам, поддерживающим Ethernet-соединения. Кабели и устройства Ethernet (например, маршрутизаторы) специально разработаны для поддержки различных скоростей связи, причем самый низкий – 10 Мбит/с (10 мегабит в секунду), а самый высокий – 100 Гбит/с (100 гигабит в секунду). Наиболее распространенные скорости – 100 Мбит/с и 1 Гбит/с.

TCP/IP: The Transmission Control Protocol/Internet Protocol (Протокол управления передачей / Интернет-протокол) является сочетанием имён для набора протоколов, которые используются при осуществлении сетевой связи между хостами в современных сетях. Хотя это не единственный набор протоколов, используемых для определения сетевой связи, он используется наиболее часто. Например, TCP/IP включает определение того, как работают IP-адреса и сетевые маски.

9.3. IP-адреса

Хосты «адресуют» сетевые пакеты, используя IP-адрес конечного компьютера. Сетевой пакет также включает «обратный адрес», IP-адрес отправляющей машины.

Существуют два разных типа IP-адресов: IPv4 и IPv6. Чтобы понять, почему существуют два разных типа, нужно понять краткую историю IP-адресов.

На протяжении многих лет технология IP-адресации, используемая всеми компьютерами, была IPv4 (IP-версия 4). В адресе IPv4 для определения адреса используются четыре 8-битных (8-разрядных = числа от 0 до 255) чисел. Например: 192.168.10.120. Примечание. Это считается 32-разрядным адресом ($4 \times 8 \text{ бит} = 32$).

Каждый хост в Интернете должен иметь уникальный IP-адрес. В среде IPv4 существует технический предел в размере около 4,3 миллиарда IP-адресов. Однако многие из этих IP-адресов не могут использоваться по разным причинам. Кроме того, IP-адреса были назначены организациям, которые еще не полностью использовали все доступные им IP-адреса.

Хотя кажется, что должно существовать много IP-адресов, различные факторы (увеличение количества хостов в Интернете, зарезервированные частные IP-адреса и т.д.) привели к возникновению проблемы: в Интернете закончилось использование IP-адресов.

Это, в частности, способствовало разработке IPv6. IPv6 был официально «создан» в 1998 году. В сети IPv6 адреса намного больше, 128-разрядные адреса выглядят следующим образом: 2001:0db8:85a3:0042:1000:8a2e:0370:7334. По сути, это обеспечивает гораздо больший пул адресов, настолько большой, что

исчерпание адресов в любое время в ближайшем будущем очень маловероятно.

Важно отметить, что разница между IPv4 и IPv6 – это не просто «больше IP-адресов». IPv6 имеет множество других дополнительных функций, которые затрагивают некоторые из ограничений IPv4, включая лучшую скорость, более совершенное управление пакетами и более эффективную передачу данных.

Учитывая все преимущества, можно подумать, что к настоящему времени все хосты будут использовать IPv6. Это совсем не так. Большинство сетевых устройств в мире по-прежнему используют IPv4 (около 98-99% всех устройств). Итак, почему мир не перешёл на превосходящую технологию IPv6?

Есть две основные причины:

- Изобретение NAT: изобретенное для преодоления возможности исчерпания IP-адресов в среде IPv4, Net Address Translation (NAT) использовало технику, обеспечивающую больший доступ к Интернету. В двух словах: группа хостов помещается в частную сеть без прямого доступа к Интернету; специальный маршрутизатор обеспечивает доступ в Интернет, и только одному маршрутизатору нужен IP-адрес для связи в Интернете. Другими словами, группа хостов использует один IP-адрес, что означает, что к Интернету может подключаться гораздо больше компьютеров. Эта функция означает, что необходимость перехода на IPv6 менее критична, чем до изобретения NAT.
- Проблемы с переносом: перенос с одной технологии на другую. У IPv6 есть много замечательных новых функций, но все хосты должны иметь возможность использовать эти функции. Привлечение всех пользователей Интернета (или даже некоторых из них) к этим изменениям создает проблему.

Большинство экспертов согласны с тем, что IPv6 в конечном итоге заменит IPv4, поэтому понимание основ обоих важно для тех, кто работает в ИТ-индустрии.

9.4. Конфигурация сетевых устройств

Когда настраиваются сетевые устройства, есть два начальных вопроса:

- Проводное или беспроводное? Настройка беспроводного устройства будет немного отличаться от проводного устройства из-за некоторых дополнительных функций, обычно встречающихся на беспроводных устройствах (таких как безопасность).
- DHCP или статический адрес? DHCP-сервер предоставляет сетевую информацию, такую как IP-адрес и маску подсети. Если не используется DHCP-сервер, необходимо вручную предоставить эту информацию хосту. Это называется использованием статического IP-адреса.

Чаще всего десктопная машина будет использовать проводную сеть, а ноутбук будет использовать беспроводную связь. Почти во всех случаях и проводные и беспроводные машины используют DHCP. Статическую адресацию используют с серверами или другими специальными ресурсами.

9.4.1. Настройка сети с помощью GUI

Если имеется доступ к среде графического интерфейса пользователя (GUI), скорее всего, также будет доступ к графическому интерфейсу, который позволит настроить сеть. Эти инструменты варьируются от одного дистрибутива к другому. Следующие примеры были выполнены на машине CentOS.

Чтобы запустить инструмент настройки сети, надо нажать **System** в строке меню, затем **Preferences >** и затем **Network Connections**, что продемонстрировано на рисунке 10.

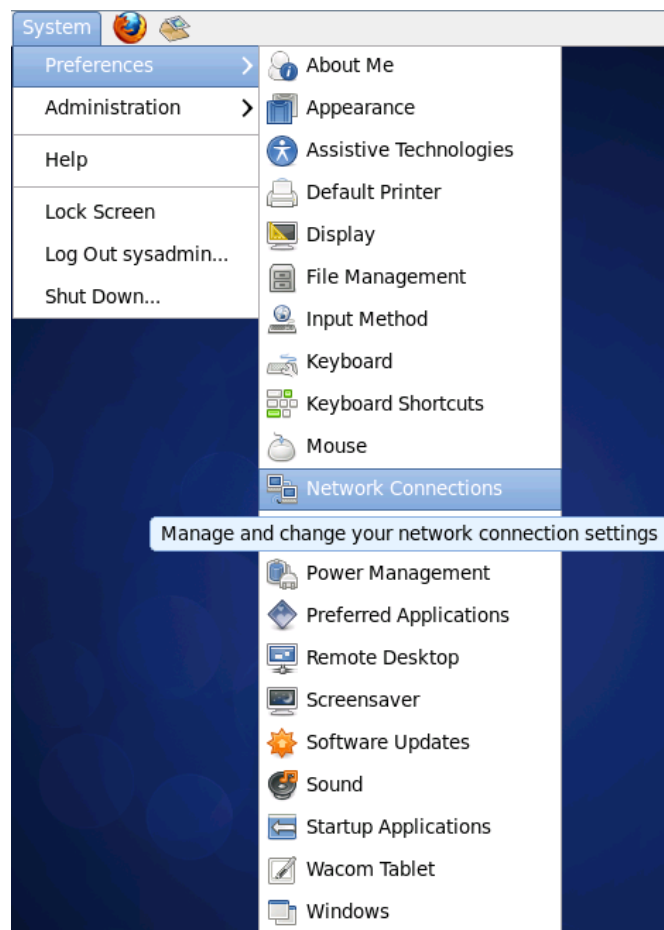


Рис. 10. Вызов программы настройки сети

Инструмент сначала отображает все текущие сетевые устройства. В приведенном ниже примере на рисунке 11 отображается только проводное устройство:

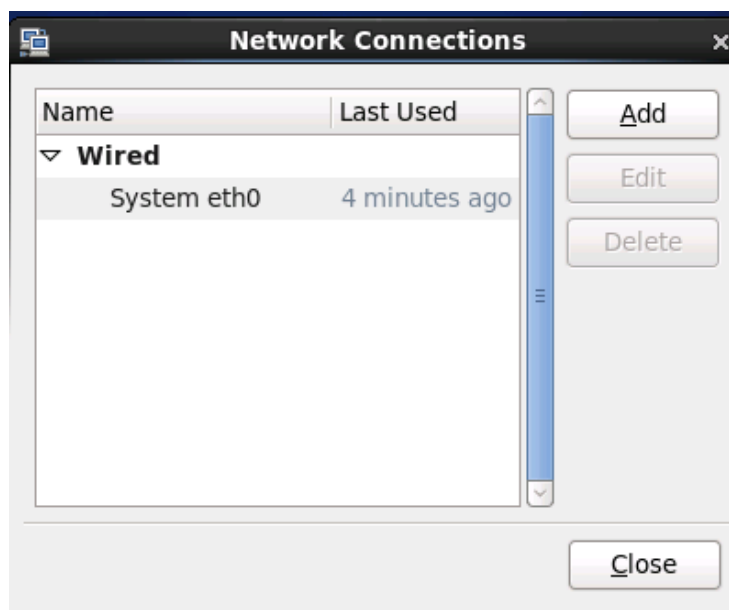


Рис. 11. Отображение текущих соединений

Самым сетевым устройством является eth0. Сетевые устройства называются eth0, eth1 и т.д. Чтобы изменить настройки сетевого устройства, нужно щелкнуть на имя устройства и нажать кнопку **Edit**.

Если нажать на закладку **IPv4 Settings**, отобразятся настройки IPv4. Пример настроек показан на рисунке 12.

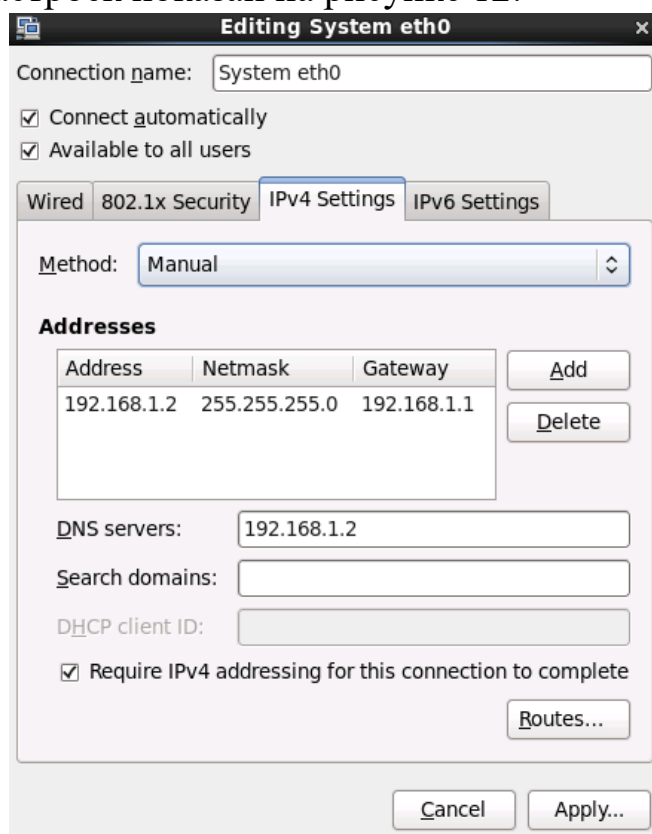


Рис. 12. Настройка параметров IPv4

Можно назначить статический IP-адрес или использовать DHCP-сервер (если он доступен). Это изменение можно сделать, нажав на раскрывающийся список рядом с **Method**.

Если выбрать **Manual**, то можно изменить текущий адрес, щелкнув в области, где в данный момент указан адрес.

Если выбрать **Automatic (DHCP)**, то детальные настройки адресов будут недоступны.

Большинство инструментов, основанных на графическом интерфейсе, вносят изменения в силу сразу после их сохранения. Однако в некоторых случаях может потребоваться либо перезагрузить компьютер, либо запустить команду в качестве администратора, чтобы изменения вступили в силу. Следующий пример демонстрирует команду, которая должна быть выполнена в системе CentOS:

```
[sysadmin@localhost ~]$ su - root
Password:
[root@localhost ~]# service network restart
Shutting down interface eth0: Device state: 3 (disconnected)
[ OK ]
Shutting down loopback interface: [ OK ]
Bringing up loopback interface: [ OK ]
Bringing up interface eth0: Active connection state: activated
Active connection path: /org/freedesktop/NetworkManager/ActiveConnection/1
[ OK ]
```

9.4.2. Настройка сети с использованием файлов конфигурации

Бывают случаи, когда инструмент GUI недоступен. В таких случаях полезно знать файлы конфигурации, которые используются для хранения и изменения сетевых данных.

Эти файлы могут различаться в зависимости от того, в каком дистрибутиве идет работа. Настройки сетевых функций могут отличаться даже в разных версиях одних и тех же операционных систем. Такое можно встретить для операционной системы Ubuntu, для которой настройки сохраняются в разных файлах, в зависимости и версии системы. При поиске информации необходимо обращать на это внимание. Следующие примеры приведены для систем CentOS.

9.4.2.1. Первичный файл конфигурации IPv4

Первичным файлом конфигурации для сетевого интерфейса IPv4 является файл `/etc/sysconfig/network-scripts/ifcfg-eth0`. Ниже показано, как может выглядеть этот файл при настройке для статического IP-адреса:

```
root@localhost:~# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE="eth0"
BOOTPROTO=none
NM_CONTROLLED="yes"
ONBOOT=yes
TYPE="Ethernet"
UUID="98cf38bf-d91c-49b3-bb1b-f48ae7f2d3b5"
DEFROUTE=yes
IPV4_FAILURE_FATAL=yes
IPV6INIT=no
NAME="System eth0"
IPADDR=192.168.1.1
PREFIX=24
GATEWAY=192.168.1.1
DNS1=192.168.1.2
HWADDR=00:50:56:90:18:18
LAST_CONNECT=1376319928
root@localhost:~#
```

Если устройство настроено как клиент DHCP, значения IPADDR, GATEWAY и DNS1 не будут установлены. Кроме того, значение BOOTPROTO будет установлено на dhcp.

9.4.2.2. Первичный файл конфигурации IPv6

В системе CentOS основной файл конфигурации IPv6 – это тот же файл, в котором хранится конфигурация IPv4: файл /etc/sysconfig/network-scripts/ifcfg-eth0. Чтобы система имела статический IPv6-адрес, следует добавить в конфигурационный файл следующее:

```
IPV6INIT=yes
IPV6ADDR=<IPv6 IP Address>
IPV6_DEFAULTGW=<IPv6 IP Gateway Address>
```

Чтобы система была клиентом DHCP IPv6, следует добавить следующую настройку:

```
DHCPV6C=yes
```

Также необходимо добавить следующий параметр в файл /etc/sysconfig/network:

```
NETWORKING_IPV6=yes
```

9.4.2.3. Domain Name Service (DNS)

Когда компьютер запрашивает доступ к веб-сайту, например www.example.com, он изначально не знает, какой IP-адрес получателя использовать для отправки пакетов с сообщениями. Чтобы компьютер связал IP-адрес с запросом URL или именем хоста, компьютер полагается на службу DNS. Часто IP-адрес DNS-сервера устанавливается во время запроса DHCP, в то время как компьютер получает важную информацию адресации для связи в сети.

Адрес DNS-сервера хранится в файле /etc/resolv.conf. Типичный файл /etc/resolv.conf автоматически создается и выглядит следующим образом:

```
sysadmin@localhost:~$ cat /etc/resolv.conf
nameserver 127.0.0.1
sysadmin@localhost:~$
```

Параметр nameserver часто устанавливается на IP-адрес DNS-сервера. В следующем примере используется команда host, описанная

ниже в этой главе. В примере показано, что сервер example.com связан с IP-адресом 192.168.1.2 (DNS-сервером):

```
sysadmin@localhost:~$ host example.com
example.com has address 192.168.1.2
sysadmin@localhost:~$
```

9.4.2.4. Файл/etc/resolv.conf

Файл, который содержит расположение DNS-сервера, - это /etc/resolv.conf. Типичный файл /etc/resolv.conf выглядит следующим образом:

```
sysadmin@localhost:~$ cat /etc/resolv.conf
# Generated by NetworkManager
nameserver 192.168.1.2
sysadmin@localhost:~$
```

Параметр nameserver установлен на IP-адрес DNS-сервера. В настройках часто устанавливается несколько серверов DNS, на случай если один DNS-сервер не отвечает.

9.4.2.5. Дополнительные файлы конфигурации сети

В таблице 17 описаны дополнительные файлы конфигурации сети, о которых нужно знать.

Таблица 17

Команда	Объяснение
/etc/hosts	Этот файл содержит таблицу имен хостов для IP-адресов. Он может использоваться для дополнения DNS-сервера.
/etc/sysconfig/network	Этот файл имеет две настройки. Настройка NETWORK может определить, включена ли сеть (yes) или выключена (no). Параметр HOSTNAME определяет имя хоста локального компьютера.
/etc/nsswitch.conf	Этот файл можно использовать для изменения того, где происходит поиск хоста. Например, по умолчанию имена ищутся в файле /etc/hosts, а затем в DNS-сервере. Если поменять настройку, все будет наоборот.

9.4.2.6. Перезапуск сети

После изменения файла конфигурации сети (например, файла /etc/sysconfig/network-scripts/ifcfg-eth0 или файла /etc/resolv.conf) необходимо перезагрузить компьютер или запустить команду в роли администратора, чтобы изменения вступили в

силу. В следующем примере демонстрируется команда, которая должна быть выполнена в системе CentOS:

```
[sysadmin@localhost ~]$ su - root
Password:
[root@localhost ~]# service network restart
Shutting down interface eth0: Device state: 3 (disconnected)
[ OK ]
Shutting down loopback interface:
[ OK ]
Bringing up loopback interface:
[ OK ]
Bringing up interface eth0: Active connection state: activated
Active connection path: /org/freedesktop/NetworkManager/ActiveConnection/1
[ OK ]
```

9.5. Сетевые утилиты

Существует несколько команд, которые можно использовать для просмотра сетевой информации. Эти инструменты также могут быть полезны при устранении неполадок сети.

9.5.1. Команда ifconfig

Команда `ifconfig` показывает конфигурацию сетевого интерфейса и используется для отображения информации о конфигурации сети. На следующем рисунке важно отметить, что IP-адрес основного сетевого устройства (`eth0`) равен `192.168.1.2` и что устройство в настоящее время активно (UP):

```
root@localhost:~# ifconfig
eth0      Link encap:Ethernet  HWaddr b6:84:ab:e9:8f:0a
          inet addr:192.168.1.2  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::b484:abff:fee9:8f0a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:95 errors:0 dropped:4 overruns:0 frame:0
          TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:25306 (25.3 KB)  TX bytes:690 (690.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:460 (460.0 B)  TX bytes:460 (460.0 B)

root@localhost:~#
```

Устройство `lo` называется устройством кольцевой проверки (указывает на сам компьютер). Это специальное сетевое устройство, используемое системой при отправке данных на основе сети, но приводящее пакеты на тот же компьютер.

Команда `ifconfig` также может использоваться для временного изменения сетевых настроек. Обычно эти изменения должны быть постоянными, поэтому команду `ifconfig` используют для внесения таких изменений довольно редко.

В качестве альтернативы для вывода информации и сетевых интерфейсах можно использовать команду `ip`:

```
root@localhost:~# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
6476: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codelstate UP qlen 1000
    link/ether b6:84:ab:e9:8f:0a brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.2/24 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::b484:abff:fee9:8f0a/64 scope link
        valid_lft forever preferred_lft forever
root@localhost:~#
```

9.5.2. Команда `route`

Маршрутизатор (или шлюз) — это машина, которая позволяет хостам из одной сети взаимодействовать с другой сетью. Чтобы просмотреть таблицу, в которой описывается, куда отправляются сетевые пакеты, можно использовать команду `route`:

```
root@localhost:~# route
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0      *              255.255.255.0   U        0      0      0 eth0
default          192.168.1.1    0.0.0.0         UG       0      0      0 eth0
root@localhost:~#
```

Первое выделенное поле в приведенном выше примере указывает, что любой сетевой пакет, отправленный на машину в сети `192.168.1.0/24`, не отправляется на шлюз (* указывает «нет шлюза»). Второе выделение указывает, что все остальные сетевые пакеты отправляются на хост с IP-адресом `192.168.1.1` (маршрутизатор).

Некоторые пользователи предпочитают отображать эту информацию только с числовыми данными, используя параметр `-n` для команды `route`. Например, на следующем примере стоит сосредоточиться на том месте, где показан вывод маршрута, используемого по умолчанию:


```

root@localhost:~# route -n
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
192.168.1.0      0.0.0.0          255.255.255.0    U        0      0      0 eth0
0.0.0.0          192.168.1.1     0.0.0.0          UG        0      0      0 eth0
root@localhost:~#

```

0.0.0.0 относится к «всем другим машинам» или тому же, что и «по умолчанию».

Аналогичный вывод с помощью команды `ip`:

```

root@localhost:~# ip route show
default via 192.168.1.254 dev eth0 proto static
192.168.1.0/24 dev eth0 proto kernel scope link src 192.168.1.2
root@localhost:~#

```

9.5.3. Команда `ping`

Команда `ping` может использоваться для определения, доступна ли другая машина. Если команда `ping` может отправить сетевой пакет на другой компьютер и получить ответ, пользователь имеет возможность подключиться к этой машине.

По умолчанию команда `ping` будет отправлять пакеты бесконечно. Чтобы ограничить количество отправленных писем, используется параметр `-c`.

Если команда `ping` успешна, получится такой вывод, как показано ниже:

```

root@localhost:~# ping -c 4 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_req=1 ttl=64 time=0.051 ms
64 bytes from 192.168.1.2: icmp_req=2 ttl=64 time=0.064 ms
64 bytes from 192.168.1.2: icmp_req=3 ttl=64 time=0.050 ms
64 bytes from 192.168.1.2: icmp_req=4 ttl=64 time=0.043 ms

--- 192.168.1.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.043/0.052/0.064/0.007 ms
root@localhost:~#

```

Если команда `ping` не удастся, появится сообщение `Destination Host Unreachable`:

```

root@localhost:~# ping -c 4 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
From 192.168.1.2 icmp_seq=1 Destination Host Unreachable
From 192.168.1.2 icmp_seq=2 Destination Host Unreachable
From 192.168.1.2 icmp_seq=3 Destination Host Unreachable
From 192.168.1.2 icmp_seq=4 Destination Host Unreachable

--- 192.168.1.1 ping statistics ---
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 2999ms
pipe 4
root@localhost:~#

```

Важно отметить, что только потому, что команда `ping` не работает, нельзя сделать вывод, что удаленная система действительно недоступна. Некоторые администраторы настраивают свои компьютеры так, чтобы они не отвечали на запросы `ping`.

Это связано с тем, что на сервер можно применить атаку, называемую «denial of service attack» (DoS). При таком типе атаки сервер перегружен огромным количеством сетевых пакетов. Путем игнорирования запросов `ping` сервер менее уязвим.

Команда `ping` может быть полезна для проверки доступности локальных машин, но не всегда для машин вне собственной сети.

9.5.4. Команда `netstat`

Команда `netstat` – это мощный инструмент, предоставляющий много информации о сети. Она может использоваться для отображения информации о сетевых подключениях, а также отображения таблицы маршрутизации, аналогичной команде `route`.

Например, можно отображать статистику сетевого трафика. Это можно сделать, используя опцию `-i`:

```
root@localhost:~# 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      137    0      4 0       12     0      0     0 BMRU
lo      65536 0       18     0      0 0       18     0      0     0 LRU
root@localhost:~#
```

Наиболее важной статистикой из вышеприведенного вывода являются `TX-OK` и `TX-ERR`. Высокий процент `TX-ERR` может указывать на проблему в сети, такую как слишком большой сетевой трафик.

Чтобы использовать команду `netstat` для отображения информации о маршрутизации, используется параметр `-r`:

```
root@localhost:~# netstat -r
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window  irtt Iface
192.168.1.0 * 255.255.255.0 U 0 0 0 eth0
default 192.168.1.1 0.0.0.0 UG 0 0 0 eth0
root@localhost:~#
```

Команда `netstat` также обычно используется для отображения открытых портов. Порт – это уникальный номер, связанный с сервисом, предоставляемым хостом. Если порт открыт, то услуга доступна для других хостов.

Например, можно войти в хост с другого хоста с помощью службы SSH. Для службы SSH назначен порт № 22. Если порт № 22 открыт, то услуга доступна для других хостов.

Важно отметить, что хост также должен иметь сами службы; это означает, что программа, которая позволяет удаленным пользователям регистрироваться, должна быть запущена (как правило, для большинства дистрибутивов Linux).

Чтобы просмотреть список всех открытых портов, можно использовать следующую команду:

```
root@localhost:~# netstat -tln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 192.168.1.2:53          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN
tcp6       0      0 :::53                   :::*                    LISTEN
tcp6       0      0 :::22                    :::*                    LISTEN
tcp6       0      0 :::1:953                 :::*                    LISTEN
root@localhost:~#
```

Как видно из вышеприведенного вывода, порт № 22 является «LISTEN», что означает, что он открыт и находится в режиме прослушивания.

В предыдущем примере `-t` означает TCP, `-l` означает «listening» (какие порты прослушивают), а `-n` означает «show numbers, not names».

Иногда отображение имен может быть более полезным. Для этого надо убрать опцию `-n`:

```
root@localhost:~# netstat -tl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 cserver.example.:domain *:.*                     LISTEN
tcp        0      0 localhost:domain        *:.*                     LISTEN
tcp        0      0 *:ssh                   *:.*                     LISTEN
tcp        0      0 localhost:953            *:.*                     LISTEN
tcp6       0      0 [::]:domain             [::]:*                   LISTEN
tcp6       0      0 [::]:ssh                 [::]:*                   LISTEN
tcp6       0      0 localhost:953            [::]:*                   LISTEN
root@localhost:~#
```

В некоторых дистрибутивах можно увидеть следующее сообщение на странице man команды `netstat`:

```
Эта программа в основном устарела. Заменяет netstat команда ss. Заменяет
netstat -r команда ip route. Заменяет netstat -i команда ip -s link. Заменяет
netstat -g команда ip maddr.
```

9.5.5. Команда dig

Могут быть моменты, когда нужно проверить функциональность DNS-сервера, к которому обращается хост пользователя. Один из способов сделать это – использовать команду `dig`. Эта команда будет выполнять запросы на DNS-сервере, чтобы определить, доступна ли информация на сервере.

В следующем примере команда `dig` используется для определения IP-адреса узла `example.com`:

```
root@localhost:~# dig example.com
; <<>> DiG 9.8.1-P1 <<>> example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45155
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;example.com.                IN      A

;; ANSWER SECTION:
example.com.                 86400   IN      A      192.168.1.2
;; AUTHORITY SECTION:
example.com.                 86400   IN      NS      example.com.

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Tue Dec  8 17:54:41 2022
;; MSG SIZE rcvd: 59
root@localhost:~#
```

Следует обратить внимание, что ответ включал IP-адрес `192.168.1.2`, а это означает, что DNS-сервер имеет IP-адрес для передачи данных имени хоста в своей базе данных.

Если DNS-сервер пользователя не имеет запрашиваемой информации, он настроен на другие DNS-серверы. Если ни один из них не имеет запрошенной информации, отобразится сообщение об ошибке:

```
root@localhost:~# dig sample.com

; <<>> DiG 9.8.1-P1 <<>> sample.com
;; global options: +cmd
;; connection timed out; no servers could be reached
root@localhost:~#
```

9.5.6. Команда host

В самой простой форме команда `host` работает с DNS для связывания имени хоста с IP-адресом. Как в предыдущем примере, `example.com` связан с IP-адресом `192.168.1.2`:

```
root@localhost:~# host example.com
```

```
example.com has address 93.184.216.34
example.com has IPv6 address 2606:2800:220:1:248:1893:25c8:1946
example.com mail is handled by 0 .
root@localhost:~#
```

Команда `host` также может использоваться в обратном порядке, если известен IP-адрес, но не имя домена.

```
root@localhost:~# host 192.168.1.2
2.1.168.192.in-addr.arpa domain name pointer example.com.
2.1.168.192.in-addr.arpa domain name pointer cserver.example.com.
root@localhost:~#
```

Существуют и другие варианты запроса различных аспектов DNS, таких как CNAME (каноническое название `-alias`):

```
root@localhost:~# host -t CNAME example.com
example.com has no CNAME record
root@localhost:~#
```

Поскольку многие DNS-серверы хранят только копию информации об имени `example.com`, записи SOA (Start of Authority) указывают основной сервер для домена:

```
root@localhost:~# host -t SOA example.com
example.com has SOA record ns.icann.org. noc.dns.icann.org. 2022040403 7200 3600
1209600 3600
root@localhost:~#
```

Полный список DNS-информации, относящейся к URL, можно найти с помощью опции `-a` (`all`).

9.5.7. Команда `ssh`

Команда `ssh` позволяет подключиться к другой машине по сети, выполнить вход и выполнить задачи на удаленном компьютере.

Когда используется команда `ssh` и указывается только имя компьютера или IP-адрес для входа в систему, команда предполагает, что пользователь хочет войти в систему, используя то же имя пользователя, в котором он в настоящее время зарегистрирован. Чтобы использовать другое имя пользователя, можно использовать синтаксис `имя@название_или_адрес_узла`:

```
root@localhost:~# ssh bob@test
The authenticity of host 'test (127.0.0.1)' can't be established.
RSA key fingerprint is c2:0d:ff:27:4c:f8:69:a9:c6:3e:13:da:2f:47:e4:c9.
Are you sure you want to continue connection (yes/no)? yes
Warning: Permanently added 'test' (RSA) to the list of known hosts.
bob@test's password:
bob@test:~$
```

9.5.7.1. Отпечаток ключа RSA

В первом приглашении запрашивается идентификация машины, в которую пользователь входит. Можно проверить вход с администратором удаленной машины, чтобы убедиться, что отпечаток ключа RSA правильный. В целом запрос предназначен для будущих попыток входа в систему.

После того, как пользователь ответит «yes», отпечаток ключа RSA удаленного компьютера будет сохранен в локальной системе. Когда пользователь пытается выполнить `ssh` на этом же компьютере в будущем, отпечаток ключа RSA, предоставленный удаленной машиной, сравнивается с копией, хранящейся на локальном компьютере. Если они совпадают, появится приглашение имени пользователя. Если они не совпадают, появится ошибка, подобная следующей:

```
sysadmin@localhost:~$ ssh bob@test
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@  WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!  @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
c2:0d:ff:27:4c:f8:69:a9:c6:3e:13:da:2f:47:e4:c9.
Please contact your system administrator.

Add correct host key in /home/sysadmin/.ssh/known_hosts to get rid of this message.
Offending key in /home/sysadmin/.ssh/known_hosts:1
RSA host key for test has changed and you have requested strict checking.
Host key verification failed.
sysadmin@localhost:~$
```

Эта ошибка может указывать на то, что посторонний хост заменил правильный хост. Следует обратиться к администратору удаленной системы. Если система была недавно переустановлена, у нее будет новый ключ RSA, и это приведет к этой ошибке.

В случае если это сообщение об ошибке связано с переустановкой удаленной машины, можно удалить файл `~/.ssh/known_hosts` из локальной системы (или просто удалить запись для этой машины) и попытаться снова подключиться:

```
sysadmin@localhost:~$ cat ~/.ssh/known_hosts
test          ssh-rsa          AAAAB3NzaC1yc2EAAAQAAIwAAAEAAk-
lOUpkDHRfHY17SbrmTIp/RZ0V4DTxgq9wzd+ohy006SWDSGPA+nafz1HDPow7vdI4mZ5ew18KL4JW9jbhU-
FrvIqzM7x1ELEVF4h91FX5QVkbPppSrg0cda3Pbv7kOdJ/MTyBlWXFCRH+Cv3FXRitBqxiXlnKhXpHAZ-
sMciLq8V6RjsNAQwdsdMFvS1VK/7BA
t5FaiKoAfnCM1Q8x3+2V0Ww71/eIFmb1zuUFljHYTprX88XypNDvjYNby6vw/Pb0rwrpz/Tn
mZAW3UX+PnTPI89ZPmNBLuxyrD2cE86Z/il8b+gw3r3+1nJotmIkjn2sold01QraTlMqVVSbx
NrRFi9wrf+ghw==
```

```
sysadmin@localhost:~$ rm ~/.ssh/known_hosts
sysadmin@localhost:~$ ssh bob@test
The authenticity of host 'test (127.0.0.1)' can't be established.
RSA key fingerprint is c2:0d:ff:27:4c:f8:69:a9:c6:3e:13:da:2f:47:e4:c9.
Are you sure you want to continue connection (yes/no)? yes
Warning: Permanently added 'test' (RSA) to the list of known hosts.
bob@test's password:
Last login: Fri Oct 4 16:14:39 CDT 2020 from localhost
bob@test:~$
```

9.5.7.2. Возврат на локальную машину

Чтобы вернуться на локальный компьютер, можно использовать команду `exit`:

```
bob@test:~$ exit
logout
Connection to test closed.
sysadmin@localhost:~#
```

Если команда `exit` выполняется в консоли локальной машины, закрывается окно рабочего терминала.

Контрольные вопросы

1. Какие устройства являются оконечными для сети?
2. Кто иницирует сетевое взаимодействие клиент или сервер?
3. Что означает аббревиатура IP?
4. В чем отличие настройки сетевых параметров через службу DHCP или статически?
5. Для чего используется команда SSH?
6. С помощью какой команды можно проанализировать работу протокола DHCP?
7. Какой командой можно отобразить текущие сетевые подключения и открытые порты?

Глава 10. СИСТЕМНАЯ И ПОЛЬЗОВАТЕЛЬСКАЯ БЕЗОПАСНОСТЬ

10.1. Аккаунты пользователей

Аккаунты пользователей разработаны для обеспечения безопасности в операционной системе. Каждый пользователь должен войти в систему используя свой аккаунт, который обеспечивает доступ к определённым файлам и директориям или наоборот запрещает. Это достигается за счёт разрешения файлов. Эта тема рассматривается в следующей главе.

Аккаунты пользователей принадлежат группам, которые так же могут быть использованы для обеспечения доступа к файлам или директориям. Каждый пользователь принадлежит как минимум к одной группе (чаще нескольким) для обеспечения упрощённого общего доступа к данным.

Аккаунты пользователей и группы хранятся в файлах базы данных. Важно знать содержимое этих файлов, т.к. это позволит понять к каким файлам и директориям пользователь имеет доступ. Эти файлы базы данных так же содержат жизненно важную информацию, которая может влиять на способность пользователей входить в систему.

Существует несколько команд, которые предоставляют возможность видеть информацию об учетной записи пользователя и группы, а также позволяют переключаться с одной учетной записи пользователя на другую (при условии, что есть соответствующие полномочия для этого). Эти команды ценны для изучения системы, устранения неполадок системы и мониторинга несанкционированного доступа к системе.

10.2. Каталог /etc

В каталоге /etc имеется несколько текстовых файлов, содержащих данные учетных записей пользователей и групп, определенных в системе. Например, если требуется узнать, определена ли учетная запись пользователя в системе, то местом для проверки является файл /etc/passwd.

Каждая строка файла `/etc/passwd` относится к учетной записи пользователя. Ниже показаны первые десять строк обычного файла `/etc/passwd`

```
sysadmin@localhost:~$ head /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
sysadmin@localhost:~$
```

Каждая строка разделяется на поля символами двоеточия. Поля слева направо выглядят следующим образом:

имя: пароль: идентификатор пользователя: первичный идентификатор группы: комментарий: домашняя директория: оболочка

В таблице 18 подробно описывается каждое из этих полей, используя в качестве примера первую строку вывода (`root:x:0:0:root:/root:/bin/bash`):

Таблица 18

Поле	Пример	Описание
Имя	root	Это имя учетной записи. Это имя используется при входе в систему и при условии, что владельцу файла предоставляется команда <code>ls -l</code> . Как правило, система использует идентификатор пользователя, имя учетной записи предоставляется, чтобы облегчить обычным пользователям обращение к учетной записи. Учетная запись <code>root</code> обычно является специальной административной учетной записью. Однако важно отметить, что не все системы имеют учетную запись <code>root</code> . Идентификатор пользователя 0 (ноль), предоставляет административные привилегии в системе.
пароль	x	В свое время пароль для пользователя хранился в этом месте, но теперь он сохраняется в файле <code>/etc/shadow</code> . Символ “x” в поле пароля указывает системе, что пароль хранится не здесь, а в файле <code>/etc/shadow</code> .
идентификатор пользователя	0	Каждой учетной записи присваивается идентификатор пользователя (UID). UID - это то, что определяет учетную запись, поскольку имя пользователя обычно не используется напрямую системой. Например, файлы принадлежат UID, а не именам пользователей.

		<p>Некоторые UID являются особыми. Например, UID 0 предоставляет эту учетную запись с правами администратора.</p> <p>UID меньше 500 (в некоторых дистрибутивах Linux 1000) зарезервированы для системных учетных записей. Системные учетные записи будут рассмотрены более подробно ниже в этой главе.</p>
первичный идентификатор группы	0	<p>Группам присваиваются числовые идентификаторы, как и пользователям.</p> <p>Когда пользователь создает файл, файл принадлежит UID пользователя, а также принадлежит идентификатору группы (GID), основному GID пользователя. Это поле определяет, какой GID является основным GID пользователя.</p> <p>Кроме того, предоставляя групповое владение по умолчанию в этом файле, это поле также указывает, что пользователь является членом группы, а это означает, что у пользователя будут специальные разрешения на любой файл, принадлежащий этой группе.</p>
комментарий	root	<p>Это поле может содержать любую информацию о пользователе, включая реальное (полное) имя и другую полезную информацию.</p> <p>Это поле также называется полем GECOS (General Electric Complete Operating System). GECOS - это редко используемый предопределенный формат для этого поля, который определяет список элементов, разделенных запятыми, включая полное имя пользователя, местоположение офиса, номер телефона и дополнительную информацию.</p> <p>Администратор может изменять информацию GECOS с помощью команды <code>chfn</code>, и пользователи могут отображать эту информацию с помощью команды <code>finger</code>.</p>
домашняя директория	/root	<p>Это поле определяет местоположение домашнего каталога пользователя. Для обычных пользователей это обычно будет <code>/home/username</code>. Например, имя пользователя bob будет иметь домашний каталог <code>/home/bob</code>. Обычно пользователь root имеет другое место для домашнего каталога: <code>/root</code>.</p> <p>У системных учетных записей редко есть домашние каталоги, поскольку они обычно не используются для создания или хранения файлов.</p>

оболочка	/bin/bash	Это местоположение оболочки входа пользователя. По умолчанию пользователь «помещается» в эту оболочку всякий раз, когда пользователь входит в среду командной строки или открывает окно терминала. Затем пользователь может переключиться на другую оболочку, набрав имя оболочки, например: /bin/tcsh. Оболочка bash (/bin/bash) является наиболее распространенной оболочкой для пользователей Linux.
----------	-----------	---

Файл /etc/shadow содержит информацию об учетной записи, связанную с паролем пользователя. Типичный файл /etc/shadow будет выглядеть следующим образом:

```
root@localhost:~# head /etc/shadow
root:$6$4Yga95H9$8HbxqsMEIBTZ0YomlMffYCV9VE1SQ4T2H3SHXw41M02SQtfAddVE9mqGp2hr20q
.ZuncJpLyWkYwQdKlSJyS8.:16464:0:99999:7:::
daemon*:16463:0:99999:7:::
bin*:16463:0:99999:7:::
sys*:16463:0:99999:7:::
sync*:16463:0:99999:7:::
games*:16463:0:99999:7:::
man*:16463:0:99999:7:::
lp*:16463:0:99999:7:::
mail*:16463:0:99999:7:::
news*:16463:0:99999:7:::
root@localhost:~#
```

Поля файла /etc/shadow: имя; пароль; последнее изменение; мин; макс; предупреждение; неактивно; истекает; зарезервировано. Описание значений полей показана в таблице 19.

Таблица 19

Поле	Пример	Описание
имя	sysadmin	Это имя учетной записи, которая соответствует имени учетной записи в файле /etc/passwd.
пароль	\$6\$.....rI 1	Поле пароля содержит зашифрованный пароль для учетной записи. Эта очень длинная строка (усечена в примере) является односторонним шифром, а это означает, что ее исходный пароль не может быть «раскрыт». Системные учетные записи будут иметь символ * в этом поле.

Поле	Пример	Описание
последнее изменение	15020	<p>Это поле содержит номер, который представляет собой дату последнего раза, когда пароль был изменен. Число 15020 - это количество дней с 1 января 1970 года (называемое “Эпоха” - “Epoch”).</p> <p>Это значение генерируется автоматически при изменении пароля пользователя. Это значение важно, так как оно используется функциями устаревания паролей, предоставляемыми остальными полями этого файла.</p>
мин	5	<p>Это одно из полей устаревания паролей. Ненулевое значение в этом поле указывает, что после того, как пользователь изменит свой пароль, пароль не может быть снова изменен на указанное количество дней (5 дней в этом примере). Это поле важно, когда используется поле макс.</p> <p>Значение нуля в этом поле означает, что пользователь может изменить свой пароль в любой момент.</p>
макс	30	<p>Это поле используется, чтобы заставить пользователей менять свои пароли на регулярной основе. Значение 30 в этом поле означает, что пользователь должен изменить свой пароль не реже одного раза в 30 дней, чтобы избежать блокировки своей учетной записи.</p> <p>Если для поля мин установлено значение 0, пользователь может сразу установить свой пароль обратно на исходное значение, нарушив цель принуждения пользователя к изменению пароля каждые 30 дней. Таким образом, если задано максимальное поле, обычно устанавливается и поле мин. Например, мин: макс 5:60 означает, что пользователь должен менять свой пароль каждые 60 дней, и после изменения пользователь должен подождать 5 дней, прежде чем сможет снова изменить свой пароль.</p> <p>Если для максимального поля установлено значение 99999, максимально возможное значение, то пользователю, по существу, никогда не придется менять свой пароль (поскольку 99999 дней составляет около 274 лет).</p>

Поле	Пример	Описание
предупреждение	7	<p>Если задано максимальное поле, поле предупреждения указывает, что пользователь будет «предупрежден», когда приближается максимальный таймаут. Например, если для предупреждения установлено значение 7, то в течение 7 дней до максимального таймаута пользователь будет предупрежден об изменении пароля во время процессов входа в систему.</p> <p>Пользователь предупреждается только при входе в систему, поэтому некоторые администраторы взяли подход, чтобы установить поле предупреждения на более высокое значение, чтобы обеспечить больший шанс получить предупреждение.</p> <p>Если максимальный таймаут установлен в 99999, то поле предупреждения, по существу, бесполезно.</p>
неактивный	60	<p>Если пользователь игнорирует предупреждения, и они превышают максимальный таймаут, учетная запись будет заблокирована. В этом случае неактивное поле предоставляет пользователю «льготный» период, во время которого пароль может быть изменен, но только во время процесса входа в систему.</p> <p>Если для неактивного поля установлено значение 60, пользователь имеет 60 льготных дней, чтобы перейти на новый пароль. Если пользователь этого не сделает, тогда администратору потребуется сбросить пароль для пользователя.</p>
истекает	15050	<p>В этом поле указывается количество дней с 1 января 1970 года - день истечения срока действия учетной записи. Просроченная учетная запись будет заблокирована, а не удалена, то есть администратор может сбросить пароль, чтобы разблокировать учетную запись.</p> <p>Учетные записи с датами истечения срока действия обычно предоставляются временным сотрудникам или подрядчикам. Учетная запись автоматически истечет после последнего дня работы пользователя.</p>

Поле	Пример	Описание
		Когда администратор устанавливает это поле, используется конвертер для преобразования из реальной даты в дату «Эпоха». В интернете имеется несколько бесплатных конвертеров.
зарезервировано		В настоящее время не используется, это поле зарезервировано для будущего использования.

По соображениям безопасности обычные пользователи не могут просматривать содержимое файла `/etc/shadow`. Чтобы просмотреть содержимое этого файла, нужно войти в систему как администратор (учетная запись `root`).

Хорошим способом просмотра информации об учетной записи из файла `/etc/passwd` является использование команды `grep` для вывода только строки, содержащей интересующую учетную запись. Например, чтобы просмотреть информацию об учетной записи для имени пользователя с именем `sysadmin`, используется команда `grep sysadmin /etc/passwd`:

```
sysadmin@localhost:~$ grep sysadmin /etc/passwd
sysadmin:x:1001:1001:System Administrator,,,:/home/sysadmin:/bin/bash
sysadmin@localhost:~$
```

Другой способ получения информации пользователя, который обычно содержится в файлах `/etc/passwd` и `/etc/shadow`, заключается в использовании команды `getent`. Одним из преимуществ использования команды `getent` является то, что он может извлекать информацию об учетной записи, которая определяется локально (`/etc/passwd` и `/etc/shadow`) или на сервере сетевых каталогов.

Общий синтаксис команды `getent`: `getent database record`. Например, команда `getent passwd sysadmin` будет извлекать информацию учетной записи `passwd` для пользователя `sysadmin`:

```
sysadmin@localhost:~$ getent passwd sysadmin
sysadmin:x:1001:1001:System Administrator,,,:/home/sysadmin:/bin/bash
sysadmin@localhost:~$
```

Чтобы проверить свою личность (просмотреть учетную запись, которую пользователь сейчас использует), можно выполнить команду `id`.

Команда `id` сообщит о текущем идентификаторе, как по имени пользователя, так и по идентификатору пользователя. В дополнение к предоставлению информации учетной записи пользователя также отображается членство в группе. Без аргумента команда `id` отобразит личность самого пользователя. Учитывая имя пользователя в качестве аргумента, например `id root`, команда отображает информацию другой учетной записи:

```
sysadmin@localhost:~$ id
uid=1001(sysadmin) gid=1001(sysadmin) groups=1001(sysadmin),4(adm),27(sudo) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
sysadmin@localhost:~$
sysadmin@localhost:~$ id root
uid=0(root) gid=0(root) groups=0(root)
sysadmin@localhost:~$
```

Пользователи входят в систему, используя обычные учетные записи пользователей. Как правило, эти учетные записи имеют значения UID более 500 (на некоторых системах 1000).

Пользователь `root` имеет специальный доступ к системе. Как упоминалось ранее, этот специальный доступ фактически предоставляется учетной записи с UID 0.

Существуют дополнительные учетные записи, которые не предназначены для входа в систему. Эти учетные записи, как правило, от UID 1 до UID 499, называются системными учетными записями, и они предназначены для предоставления учетных записей для служб, работающих в системе.

У системных учетных записей есть несколько полей в файлах `/etc/passwd` и `/etc/shadow`, отличных от других учетных записей. Например, в файле `/etc/passwd` системные учетные записи будут иметь специальную не-регистрационную программу в поле «оболочка»:

```
bin:x:1:1:bin:/bin:/sbin:/sbin/nologin
```

В файле `/etc/shadow` системные учетные записи обычно будут иметь символ `*` вместо поля пароля:

```
bin*:15513:0:99999:7:::
```

Есть несколько важных вещей, которые нужно помнить о системных учетных записях:

- 1) Большинство из них необходимы для правильной работы системы.

2) Нельзя удалять системную учетную запись, если нет уверенности, что удаление учетной записи не вызовет проблем.

3) По мере того, как администратор приобретает больше опыта, он должен узнать, что делает каждая учетная запись системы. Системным администраторам поручено обеспечить безопасность в системе и обеспечить надлежащее обеспечение безопасности системных учетных записей.

10.3. Группы

Уровень доступа к системе определяется не только учетной записью пользователя. Каждый пользователь может быть членом одной или нескольких групп, что также может повлиять на уровень доступа к системе.

Традиционно UNIX-системы ограничивают пользователей не более чем шестнадцатью группами, но последние ядра Linux поддерживают пользователей с более чем 65000 членств в группах.

Как было показано ранее, файл `/etc/passwd` определяет первичное членство в группе для пользователя. Дополнительное членство в группах (или членство во вторичных группах), а также сами группы, определяется в файле `/etc/group`.

Чтобы просмотреть информацию о конкретной группе, можно использовать команды `grep` или `getent`. Например, следующие команды будут отображать информацию учетной записи группы `mail`:

```
sysadmin@localhost:~$ grep mail /etc/group
mail:x:12:mail,postfix
sysadmin@localhost:~$ getent group mail
mail:x:12:mail,postfix
```

Файл `/etc/group` представляет собой файл таблиц с разделителями двоеточия с указанием следующих полей:

имя группы: групповой пароль: идентификатор группы: список пользователей

Таблица 20 описывает поля файла `/etc/group` более подробно, используя в качестве примера строку, которая описывает типичную учетную запись группы:

```
mail:x:12:mail,postfix
```


Таблица 20

Поле	Пример	Описание
имя группы	mail	Это поле содержит имя группы. Как и имена пользователей, имена групп легче запоминать. Обычно система использует идентификаторы групп (GID), а не имена групп.
групповой пароль	x	Хотя есть пароли для групп, они редко используются в Linux. В случае, если администратор устанавливает групповой пароль, он будет храниться в другом файле (/etc/gshadow), потому что групповой пароль больше не сохраняется в файле /etc/group. X в этом поле используется, чтобы указать, что пароль не сохраняется в этом файле.
идентификатор группы	12	Каждая группа связана с уникальным идентификатором группы (GID), который помещается в это поле.
список пользователей	mail,postfix	Это последнее поле используется, чтобы указать, кто является членом группы. Хотя первичное членство в группе определяется в файле /etc/passwd, пользователи, которым назначены дополнительные группы, будут иметь свое имя пользователя в этом поле файла /etc/group. В этом примере пользователи mail и postfix являются вторичными членами группы mail.

Группы в основном используются для контроля доступа к файлам. По умолчанию любой новый файл, созданный пользователем, будет принадлежать первичной группе пользователя. Команда `id` может использоваться для проверки первичной группы пользователя:

```
sysadmin@localhost:~$ id -g
1001
```

Пользователи также могут обращаться к файлам и устройствам, если они являются членами вторичных групп. Команда `id` также может использоваться для проверки членства в дополнительных группах пользователей:

```
sysadmin@localhost:~$ id -G
1001 4 27
```

Без передачи каких-либо параметров команде `id` содержит как первичные, так и вторичные членства в группах:

```
sysadmin@localhost:~$ id
uid=1001(sysadmin) gid=1001(sysadmin) groups=1001(sysadmin),4(adm),27(sudo)
```

Имя пользователя может быть добавлено для определения групп определенного пользователя:

```
sysadmin@localhost:~$ id sysadmin
uid=1001(sysadmin) gid=1001(sysadmin) groups=1001(sysadmin),4(adm),27(sudo)
```

Для вывода вхождений пользователя во все вторичные группы можно воспользоваться командами `cat` и `grep` (в примере пользователь `sysadmin`):

```
sysadmin@localhost:~$ cat /etc/group | grep sysadmin
adm:x:4:sysadmin
sudo:x:27:sysadmin
sysadmin:x:1001:
```

Чтобы изменить группу-владельца существующего файла, можно использовать команду `chgrp group_name`. Пользователи могут изменять права собственности только на принадлежащие им файлы. Новая группа-владелец файла также должна быть группой, членом которой является пользователь:

```
sysadmin@localhost:~$ touch sample
sysadmin@localhost:~$ ls -l sample
-rw-rw-r-- 1 sysadmin sysadmin 0 Dec 10 00:44 sample
sysadmin@localhost:~$ chgrp games sample
-rw-rw-r-- 1 sysadmin games 0 Oct 23 22:12 sample
sysadmin@localhost:~$
```

Чтобы изменить групповое владение всеми файлами структуры каталога, используется параметр `-R` для команды `chgrp`. Например, команда `chgrp -R games test_dir` изменит группу-владельца в каталоге `test_dir` и всех файлах и подкаталогах каталога `test_dir`.

Существует также команда `chown`, которая может использоваться для изменения владельца пользователя файла или каталога. Однако эта команда может использоваться только пользователем `root`. Обычные пользователи не могут «отдавать» свои файлы другому пользователю.

10.4. Суперпользователь

Существует множество различных способов выполнения команды, требующей административных или корневых привилегий. Как уже упоминалось, вход в систему под пользователем `root` позволяет выполнять команды как администратор. Это потенциально опасно, потому что пользователь может забыть, что вошёл в систему под учетной записью `root` и может запустить команду, которая может вызвать проблемы в системе. В результате не рекомендуется входить в систему от имени пользователя `root` напрямую.

Использование учетной записи `root` потенциально опасно, но часто требуется выполнять команды с административными привилегиями как у пользователя `root`. Если учетная запись `root` отключена, как и в дистрибутиве Ubuntu, административные команды могут быть выполнены с использованием команды `sudo`. Если учетная запись `root` включена, то обычный пользователь может выполнить команду `su` для переключения учетных записей в корневую учетную запись (`root`).

Когда пользователь входит в систему напрямую с правами `root` для выполнения команд, то все в сеансе выполняется от лица пользователя `root`. Если использовать графическую среду, это особенно опасно, так как процесс графического входа состоит из множества различных исполняемых файлов (программ, которые запускаются во время входа в систему). Каждая программа, которая работает с правами пользователя `root`, представляет большую угрозу, чем процесс, выполняемый с правами стандартного пользователя, поскольку этим программам будет разрешено делать почти что угодно, тогда как стандартные пользовательские программы очень ограничены в том, что они могут делать.

Другая потенциальная опасность при входе в систему с правами `root` заключается в том, что человек, который делает это, может забыть выйти из системы, чтобы выполнить свою неадминистративную работу. Это означает, что такие программы, как браузеры, почтовые клиенты и т. д., будут выполняться с правами пользователя `root` без ограничений на то, что они могут сделать.

10.5. Команда SU

Команда `su` позволяет запускать оболочку от лица другого пользователя. По умолчанию, если учетная запись пользователя не указана, команда `su` откроет новую оболочку пользователя `root`. `su` также может переключить на других пользователей.

Один общий параметр, который используется с командой `su`, - это опция `-l`, которая приводит к тому, что новая оболочка является оболочкой входа. Использование команды `su` с параметром оболочки входа часто важно для обеспечения правильной работы любых команд, так как оболочка входа полностью настраивает новую оболочку с настройками нового пользователя. Оболочка без входа существенно изменяет `UID`, но не полностью регистрирует пользователя. Параметр `-l` может быть прописан как `-login`.

Поскольку учетная запись `root` используется по умолчанию с помощью команды `su`, следующие две команды являются эквивалентными способами запуска оболочки в качестве пользователя `root`:

```
su - root
su -
```

После нажатия `Enter` для выполнения любой из этих команд пользователь должен указать пароль пользователя `root`, чтобы запустить оболочку в качестве пользователя `root`.

После использования оболочки, запущенной командой `su` для выполнения необходимых административных задач, возврат к исходной оболочке (и исходной учетной записи пользователя) происходит с помощью команды `exit`.

```
sysadmin@localhost:~$ su -
Password:
root@localhost:~# id
uid=0(root) gid=0(root) groups=0(root)
root@localhost:~# exit
logout
sysadmin@localhost:~$ id
uid=1001(sysadmin) gid=1001(sysadmin) groups=1001(sysadmin),4(adm),27(sudo)
sysadmin@localhost:~$
```

10.6. Команда sudo

В дистрибутивах, которые не позволяют пользователю `root` входить напрямую или через команду `su`, процесс установки автоматиче-

ски настраивает одну учетную запись пользователя, чтобы иметь возможность использовать команду `sudo` для выполнения команд, как если бы они были выполнены пользователем `root`.

Как и команда `su`, команда `sudo` предполагает, что учетная запись пользователя `root` должна использоваться для выполнения команд. Для указания другой учетной записи пользователя используется параметр `-u`.

При использовании команды `sudo` для выполнения команды в качестве пользователя `root` команда запрашивает пароль пользователя (а не пароль пользователя `root`). Это функция безопасности, которая предотвратила бы несанкционированный доступ `root`, если бы пользователь оставил свой компьютер без присмотра. Запрос пароля не появится снова, пока пользователь продолжает выполнять команды `sudo` менее чем за пять минут.

Использование команды `sudo` для выполнения административной команды приведет к записи, помещенной в файл журнала. Эта запись будет включать имя пользователя, выполнившего команду, выполненную команду, дату и время выполнения, комментарии. Это позволяет повысить подотчетность по сравнению с системой, в которой многие пользователи могут знать пароль `root` и могут либо входить в систему как `root`, либо использовать команду `su` для выполнения команд в качестве пользователя `root`.

Можно использовать команду `sudo` для выполнения команды, требующей прав `root`. Например, нужно быть пользователем `root`, чтобы просмотреть файл `/etc/shadow`. Команда `sudo head /etc/shadow` запускает команду `head` как пользователь `root`:

```
sysadmin@localhost:~$ head /etc/shadow
head: cannot open `/etc/shadow' for reading: Permission denied
sysadmin@localhost:~$ sudo head /etc/shadow
[sudo] password for sysadmin:
root:$6$4Yga95H9$8HbxqsMEIBTZ0YomlMffYCV9VE1SQ4T2H3SHXw41M02SQtfAddVE9mqGp2hr20q
.ZuncJpLyWkYwQdKlSJyS8.:16464:0:99999:7:::
daemon*:16463:0:99999:7:::
bin*:16463:0:99999:7:::
sys*:16463:0:99999:7:::
sync*:16463:0:99999:7:::
games*:16463:0:99999:7:::
man*:16463:0:99999:7:::
lp*:16463:0:99999:7:::
mail*:16463:0:99999:7:::
news*:16463:0:99999:7:::
sysadmin@localhost:~$
```

Одним из больших преимуществ использования `sudo` для выполнения административных команд является то, что он снижает риск того, что пользователь случайно выполнит команду с правами `root`. Намерение выполнить команду ясно; команда выполняется как `root`, если префикс с командой `sudo`. В противном случае команда выполняется как обычный пользователь.

Если во время установки команда `sudo` не настроена автоматически, ее можно настроить для работы после установки. Первоначально для этого потребуется войти в систему как пользователь `root` (или использовать команду `su` для переключения на учетную запись `root`), но после того, как она будет настроена, указанные пользователи смогут запускать команды `sudo`. Установка доступа к команде `sudo` производится командой `visudo`.

Команда `visudo` переводит в программу редактора, по умолчанию редактор `vi` (или `vim`) на большинстве систем, что может быть сложным для начинающих пользователей Linux. Чтобы настроить другой редактор, надо изменить глобальную переменную `EDITOR` со значением желаемого редактора. Например, чтобы использовать редактор `gedit` вместо `vi` / `vim`, нужно выполнить команду `export EDITOR = gedit`, прежде чем выполнять команду `visudo`.

С помощью команды `visudo` откроется файл конфигурации `/etc/sudoers` для редактирования. Хотя с помощью этого файла возможны расширенные конфигурации команды `sudo`, они выходят за рамки этого материала. В основном в этот файл вводятся записи, чтобы указать, какой пользователь(и) на машине могут использовать команду `sudo`.

Эта запись по умолчанию

```
root    ALL=(ALL)    ALL
```

может быть прочитана как «пользователь `root` на всех компьютерах действует как ВСЕ пользователи для выполнения ВСЕХ команд». Чтобы позволить пользователю, например `sysadmin`, выполнять все команды, как и все пользователи, использующие команду `sudo`, можно добавить следующую запись:

```
sysadmin ALL=(ALL) ALL
```

10.7. Команда who

Команда `who` отображает список пользователей, которые в настоящее время вошли в систему, откуда они вошли в систему и когда они вошли в систему. Благодаря использованию параметров эта команда также может отображать информацию, такую как текущий уровень выполнения (функциональное состояние компьютера) и время загрузки системы.

Например:

```
sysadmin@localhost:~$ who
root          tty2      2021-10-11 10:00
sysadmin      tty1      2021-10-11 09:58 (:0)
sysadmin      pts/0     2021-10-11 09:59 (:0.0)
sysadmin      pts/1     2021-10-11 10:00 (example.com)
```

В таблице 21 описывается вывод команды `who`.

Таблица 21

Поле	Пример	Описание
Имя пользователя	root	В этом столбце указано имя пользователя, который вошел в систему. Под «вошедшим в систему» подразумевается «любой процесс входа в систему и любое открытое окно терминала».
терминал	tty2	В этом столбце указывается, в каком окне терминала работает пользователь. Если имя терминала начинается с <code>tty</code> , это указывает на локальный логин, поскольку это регулярный терминал командной строки. Если имя терминала начинается с <code>pts</code> , это означает, что пользователь использует псевдотерминал или выполняет процесс, который действует как терминал. Это может означать, что у пользователя есть терминальное приложение, работающее в X Windows, такое как <code>gnome-terminal</code> или <code>xterm</code> , или они могут использовать сетевой протокол для подключения к системе, например <code>ssh</code> или <code>telnet</code> .
дата	2021-10-11 10:00 (example.com)	Это указывает, когда пользователь вошел в систему. После даты и времени, когда пользователь вошел в систему, может появиться некоторая информация о местоположении.

Поле	Пример	Описание
		<p>Если информация о местоположении содержит имя хоста, доменное имя или IP-адрес, то пользователь удаленно зарегистрировался.</p> <p>Если есть двоеточие и число, то это означает, что они выполнили локальный графический вход.</p> <p>Если в последнем столбце не отображается информация о местоположении, это означает, что пользователь зарегистрировался через процесс локальной командной строки.</p>

Если пользователь хочет отобразить информацию о состоянии системы, то команда `who` может сделать это, используя несколько параметров. Например, параметр `-b` будет показывать последний раз, когда система была запущена (была загружена), а параметр `-r` покажет время, когда система достигла определенного уровня выполнения:

```
sysadmin@localhost:~$ who -b -r
      system boot      2021-10-11 09:54
      run-level 5      2021-10-11 09:54
```

10.8. Команда `w`

Команда `w` предоставляет более подробный список пользователей, находящихся в настоящее время в системе, по сравнению с командой `who`. Она также может выводить сводку состояния системы. Например:

```
sysadmin@localhost:~$ w
10:44:03 up 50 min,  4 users,  load average: 0.78, 0.44, 0.19
USER                TTY      FROM          LOGIN@      IDLE        JCPU       PCPU       WHAT
root                tty2      -             10:00       43:44       0.01s      0.01s      -bash
sysadmin            tty1      :0            09:58       50:02       5.68s      0.16s      pam: gdm-
password
sysadmin pts/0      :0.0         09:59       0.00s       0.14s       0.13s      ssh 192.168.1.2
sysadmin            pts/1     example.com 10:00       0.00s       0.03s      0.01s      w
```

Первая строка вывода из команды `w` идентична команде `uptime`. Она показывает текущее время, продолжительность работы системы, общее количество текущих пользователей и нагрузку на систему, усредненную за последние 1, 5 и 15 минут. Средняя загрузка - это загрузка процессора, когда значение 100 будет означать полное использование ЦП в течение этого периода времени.

В таблице 22 описывается остальная часть вывода команды `w`.

Таблица 22

Поле	Пример	Описание
USER	root	В этом столбце указано имя пользователя, который вошел в систему.
TTY	tty2	В этом столбце указывается, в каком окне терминала работает пользователь.
FROM	example.com	Откуда пользователь вошел в систему. Показывает адрес или имя компьютера.
LOGIN@	10:00	Когда пользователь вошел в систему.
IDLE	43:44	Как долго пользователь простаивает с момента последней команды.
JCPU	0.01s	Общее время процессора (s = секунды), используемое всеми процессами (программами), выполняется с момента входа в систему.
PCPU	0.01s	Общее время процессора для текущего процесса.
WHAT	-bash	Общее время процессора для текущего процесса. Текущий процесс, который выполняется пользователем.

10.9. Управление группами

В предыдущих разделах было описано то, что информация об учетной записи пользователя хранится в файле `/etc/passwd`, а информация об аутентификации пользователя (данные пароля) хранится в файле `/etc/shadow`. Создание нового пользователя может быть выполнено путем добавления новой строки к каждому из этих файлов вручную, но это обычно не рекомендуется. Используя соответствующую команду для добавления нового пользователя, эти файлы могут быть изменены автоматически и безопасно. Если указанные файлы изменяются вручную, то есть риск совершить ошибку, которая может помешать всем пользователям нормально войти в систему.

В некоторых дистрибутивах создание новой учетной записи пользователя также автоматически создает учетную запись группы для

пользователя, называемую пользовательской частной группой (User Private Group - UPG). В этих системах группа и имя пользователя будут одинаковыми, и единственным членом этой новой группы будет новый пользователь.

Для дистрибутивов, которые не создают UPG, новым пользователям обычно присваивается группа «users» в качестве основной группы. Администратор может вручную создавать учетные записи групп, которые являются частными для пользователя, но администратор чаще всего создает группы для нескольких пользователей, которым необходимо сотрудничать. Учетные записи пользователей могут быть изменены в любое время, чтобы добавить или удалить их из членства в групповой учетной записи, но пользователи должны принадлежать хотя бы одной группе.

Прежде чем начинать создавать пользователей, нужно спланировать к каким группам отнести нового пользователя. Пользователи могут быть созданы с членством в существующих группах или можно изменить существующих пользователей, чтобы сменить членство в группах.

Если уже запланировано, какие будут создаваться пользователи и группы, то правильной создать сначала группу затем пользователя. В противном случае, если сначала создать пользователей, а затем группы, нужно сделать дополнительный шаг, чтобы изменить пользователей, сделав их членами новых групп.

10.10. Создание группы

Наиболее распространенной причиной создания группы является предоставление пользователям возможности обмениваться файлами. Примером этого может быть то, что несколько человек работают вместе в одном проекте и должны иметь доступ к документам, хранящимся в файлах для проекта. В этом случае администратор может сделать этих людей членами общей группы, изменить владельца каталога на новую группу и установить разрешения в каталоге, которые позволят членам группы получать доступ к файлам.

После создания или изменения группы можно проверить изменения, просмотрев информацию о конфигурации группы в файле `/etc/group` командой `grep`. Если работа идет с сетевыми службами аутен-

тификации, команда `getent` может отображать как локальные, так и сетевые группы. Для локального использования эти команды показывают тот же результат, в данном случае для группы `root`:

```
root@localhost:~# grep root /etc/group
root:x:0:
root@localhost:~# getent group root
root:x:0:
```

Команда `groupadd` может быть выполнена пользователем `root` для создания новой группы. Команда требует только имя группы. Опцию `-g` можно использовать для указания идентификатора группы для новой группы:

```
root@localhost:~# groupadd -g 506 research
root@localhost:~# grep research /etc/group
research:x:506:
```

Если опция `-g` не указана, команда `groupadd` автоматически предоставит `GID` для новой группы. Для этого команда `groupadd` просматривает файл `/etc/group` и использует число, которое на одно значение больше текущего наивысшего номера `GID`. Выполнение следующих команд иллюстрирует это:

```
root@localhost:~# grep research /etc/group
research:x:506:
root@localhost:~# groupadd development
root@localhost:~# grep development /etc/group
development:x:507:
```

В некоторых дистрибутивах Linux, особенно на основе Red Hat, когда создается идентификатор пользователя (`UID`), пользовательская группа (`UPG`) также создается с этим пользователем в качестве единственного члена. В этих дистрибутивах `UID` пользователя и идентификатор частной группы должны соответствовать (быть одинаковым числом).

Поэтому следует избегать создания `GID` в тех же числовых диапазонах, где планируется создать идентификаторы пользователей (`UID`), чтобы избежать конфликта между созданным `GID` и номером `UPG`, созданным для соответствия `UID`.

`GID` менее 500 зарезервированы для использования в системе. Могут быть моменты, когда нужно назначить `GID` до 500. Для этого используется опция `-g`. Параметр `-g` назначит новой группе `GID`, который будет меньше минимального стандартного `UID`:

```
root@localhost:~# groupadd -r sales
root@localhost:~# getent group sales

sales:x:491:
```

Следуя этим рекомендациям для имен групп, можно выбрать имя группы, которое будет переносимым (правильно работать с другими системами или службами):

- Первым символом имени должен быть либо символ подчеркивания `_`, либо строчный алфавитный символ `a-z`.
- В большинстве дистрибутивов Linux допускается до 32 символов, но использование более 16 может быть проблематичным, поскольку некоторые дистрибутивы могут не принимать более 16.
- После первого символа остальные символы могут быть буквенно-цифровыми, тире - или подчеркиванием.
- Последний символ не должен быть дефисом.

К сожалению, эти рекомендации не всегда соблюдаются. Проблема заключается не в том, что команда `groupadd` обязательно завершится неудачей, а в том, что другие команды или системные службы могут работать некорректно.

10.11. Редактирование групп

Команда `groupmod` может использоваться для изменения имени группы (с параметром `-n`) или изменения GID (с опцией `-g`) для группы.

Изменение имени группы может вызвать путаницу для пользователей, которые знакомы со старым именем и не были проинформированы о новом имени. Однако изменение имени группы не вызовет проблем с доступом к файлам, поскольку файлы действительно принадлежат GID, а не именам групп. Например:

```
root@localhost:~# ls -l index.html
-rw-r-----. 1 root sales 0 Aug  1 13:21 index.html
root@localhost:~# groupmod -n clerks sales
root@localhost:~# ls -l index.html
-rw-r-----. 1 root clerks 0 Aug  1 13:21 index.html
```

После предыдущей команды `groupmod` файл `index.html` имеет другое имя владельца группы. Тем не менее, все пользователи, которые были в группе продаж, теперь входят в группу клерков, поэтому все эти пользователи могут получить доступ к файлу `index.html`. Опять же, это потому, что группа определяется GID, а не именем группы.

С другой стороны, если изменить GID для группы, то все файлы, которые были связаны с указанной группой, теряют эту связь. Фактически, все файлы, связанные с этой группой, больше не будут связаны с каким-либо именем группы. Вместо этого эти файлы будут принадлежать только GID, как показано ниже:

```
root@localhost:~# groupmod -g 10003 clerks
root@localhost:~# ls -l index.html
-rw-r----- 1 root 491 13370 Aug  1 13:21 index.html
```

Эти файлы без имени группы называются «потерянными» файлами. В качестве пользователя root вы можете искать все файлы, принадлежащие только GID (не связанные с именем группы). Это можно сделать с помощью опции `-nogroup` для команды `find`:

```
root@localhost:~# find / -nogroup
/root/index.html
```

10.12. Удаление групп

Если нужно удалить группу с помощью команды `groupdel`, нужно иметь в виду, что любые файлы, принадлежащие этой группе, станут «потерянными».

Только дополнительные группы могут быть удалены, поэтому, если группа является основной группой для любого пользователя, ее нельзя удалить. Администратор может изменить, какая группа является основной группой пользователя, поэтому группа, которая использовалась в качестве основной группы, может быть добавлена в дополнительную группу и затем может быть удалена.

Пока удаляемая группа не является основной группой пользователя, удаление группы выполняется с помощью команды `groupdel` вместе с именем группы:

```
root@localhost:~# groupdel clerks
```

10.13. Файл `/etc/default/useradd`

Прежде чем создавать пользователей для системы, нужно проверить или установить значения параметров, которые будут использоваться по умолчанию с помощью команды `useradd`. Это может быть выполнено путем изменения параметров в файлах конфигурации, которые используются командой `useradd`.

Обеспечение того, чтобы значения в этих конфигурационных файлах были правильными, прежде чем добавлять пользователей, поможет сэкономить время и предупредить проблемы с исправлением настроек учетной записи пользователя после его создания.

Параметр `-D` для команды `useradd` позволит просмотреть или изменить некоторые значения по умолчанию, используемые командой `useradd`. Значения, отображаемые `useradd -D`, также можно просмотреть или обновить, используя файл `/etc/default/useradd`:

```
root@localhost:~# useradd -D
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
CREATE_MAIL_SPOOL=yes
```

Чтобы изменить одно из значений по умолчанию, можно редактировать файл `/etc/default/useradd` с помощью текстового редактора. Другой (более безопасный) метод заключается в использовании команды `useradd -D`.

Например, если нужно, чтобы у пользователей были истекающие пароли, которые они могли бы использовать для входа и смены на новый пароль до тридцати дней, можно выполнить:

```
root@localhost:~# useradd -D -f 30
root@localhost:~# useradd -D
GROUP=100
HOME=/home
INACTIVE=30
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
CREATE_MAIL_SPOOL=yes
```

10.14. Файл `/etc/login.defs`

Файл `/etc/login.defs` также содержит значения, которые будут применяться по умолчанию для новых пользователей, созданных с помощью команды `useradd`. В отличие от `/etc/default/useradd`, который может быть обновлен командой `useradd -d`, файл `/etc/login.defs` обычно редактируется администратором в текстовом редакторе.

Этот файл содержит много комментариев и пустых строк, поэтому, если просто просматривать строки, которые не являются комментариями или пустыми строками (реальные настройки конфигурации), можно использовать команду `grep`:

```
root@localhost:~# grep -Ev '^#|^$' /etc/login.defs
MAIL_DIR /var/mail/spool
PASS_MAX_DAYS 99999
PASS_MIN_DAYS 0
PASS_MIN_LEN 5
PASS_WARN_AGE 7
UID_MIN 500
UID_MAX 60000
GID_MIN 500
GID_MAX 60000
CREATE_HOME yes
UMASK 077
USERGROUPS_ENAB yes
ENCRYPT_METHOD SHA512
MD5_CRYPT_ENAB no
```

10.15. Создание пользователя

Во время процесса установки большинство установщиков создадут нормального пользователя и предоставят этому пользователю разрешение на выполнение административных команд с помощью `sudo` или требуют, чтобы пароль учетной записи пользователя `root` был настроен как часть процесса установки. Это означает, что большинство Linux-систем будут настроены так, чтобы позволить одному непривилегированному (не `root`) пользователю входить в систему, а также иметь возможность эффективно выполнять команды как пользователь `root`, прямо или косвенно.

Если компьютер должен использоваться только одним человеком, достаточно иметь только одну обычную учетную запись пользователя. Однако, если компьютер должен быть разделен несколькими людьми, тогда желательно иметь отдельную учетную запись для каждого человека, который ее использует. Существует несколько преимуществ для отдельных лиц, имеющих собственные отдельные учетные записи:

- Учетные записи могут использоваться для предоставления выборочного доступа к файлам или службам. Например, пользователь каждой учетной записи имеет отдельный домашний каталог, который обычно недоступен для других пользователей.

- Команда `sudo` может быть сконфигурирована так, чтобы предоставить возможность выполнять отдельные административные команды. Если пользователи должны использовать команду `sudo` для выполнения административных команд, система будет регистрироваться при выполнении этими командами пользователей.
- Каждая учетная запись может иметь членство в группах и права, связанные с ней, что позволяет повысить гибкость управления.

Создание учетной записи пользователя для использования с системой Linux может потребовать сбора нескольких фрагментов информации. Хотя все, что может потребоваться, это имя учетной записи, можно планировать UID, основную группу, дополнительные группы, домашний каталог, каталог скелета и оболочку, которые будут использоваться. При планировании этих значений учитывается следующее:

Имя пользователя: единственным обязательным аргументом для команды `useradd` является имя, которое требуется для учетной записи. Это имя должно следовать тем же принципам, которые обсуждались ранее в этой главе для имен групп. Подводя итог, это должно быть 32 символа или меньше, начинаться со строчной буквы или подчеркивания, а затем содержать только строчные буквы, цифры, дефисы и символы подчеркивания. Если пользователю требуется доступ к нескольким системам, обычно рекомендуется, чтобы имя учетной записи было одинаковым в этих системах. Имя учетной записи должно быть уникальным для каждого пользователя.

Идентификатор пользователя (UID): после создания пользователя с определенным UID система, как правило, просто увеличивает на единицу UID для следующего создаваемого пользователя. Если есть подключение к сети с другими системами, можно убедиться, что этот UID одинаковый во всех системах, чтобы обеспечить постоянный доступ. Параметр `-u` для команды `useradd` позволяет указать номер UID. UID обычно могут варьироваться от нуля до более четырех миллиардов, но для максимальной совместимости со старыми системами максимальное рекомендуемое значение UID составляет 60 000.

Как обсуждалось ранее, у пользователя `root` есть идентификатор пользователя (UID), равный 0, что наделяет эту учетную запись особыми привилегиями. Любая учетная запись с UID=0 будет эффективно действовать как «администратор».

Системные учетные записи - это учетные записи, обычно используемые для запуска фоновых служб (называемых демонами). Не имея служб, запущенных как пользователь root, размер ущерба, который может быть нанесен с помощью скомпрометированной учетной записи службы, ограничен. Системные учетные записи, используемые службами, обычно используют UID, которые находятся в «зарезервированном» диапазоне. Одной системной учетной записью, являющейся исключением из этого правила, является пользователь nfsnobody, который имеет UID 65534.

Зарезервированный диапазон, используемый для учетных записей служб, со временем расширяется. Первоначально он был для UID между 1 и 99. Затем он расширился до 499. Текущая тенденция среди дистрибутивов заключается в том, что системными учетными записями будет любая учетная запись с UID между 1 и 999, но диапазон 1-499 также по-прежнему широко используется.

Если настраивается новая система, рекомендуется начинать UID не ниже 1000. Это также даст преимущество, заключающееся в том, что будет достаточно UID для многих системных служб и даст возможность создавать множество GID в «зарезервированный» диапазон.

Первичная группа: в дистрибутивах с функцией UPG эта группа будет создана автоматически с именем GID и группой, которая соответствует UID и имени пользователя только что созданной учетной записи пользователя. В дистрибутивах, не использующих UPG, основная группа обычно будет по умолчанию для группы «users» с GID 100. Чтобы указать основную группу с помощью команды useradd, используется параметр -g с именем или GID группы.

Дополнительная группа(ы): Если требуется сделать пользователя членом одной или нескольких дополнительных групп, параметр -G можно использовать для указания списка имен групп или номеров, разделенных запятыми.

home directory: по умолчанию большинство дистрибутивов создадут домашний каталог пользователя с тем же именем, что и учетная запись пользователя в папке /home. Например, при создании учетной записи пользователя с именем sam, новым домашним каталогом пользователя будет /home/sam. Существует несколько вариантов команды useradd, которые могут повлиять на создание домашнего каталога пользователя:

- Параметр `-b` позволяет указать другой каталог, в котором будет создан домашний каталог пользователя. Например: `-b /test` приведет к домашнему каталогу `/test/sam` для учетной записи пользователя с именем `sam`.
- Параметр `-d` позволяет указать либо существующий каталог, либо новый домашний каталог для создания для пользователя. Это должен быть полный путь для домашнего каталога пользователя. Например: `-d /home/sam`
- Параметр `-m` указывает `useradd` создать домашний каталог; обычно не требуется, так как это поведение по умолчанию для команды `useradd`. Однако при использовании опции `-k` (см. Ниже) для указания другого каталога скелета требуется опция `-m`.
- Параметр `-M` используется для указания команде `useradd`, что он не должен создавать домашний каталог

Каталог скелета (каркаса). По умолчанию содержимое каталога `/etc/skel` копируется в домашний каталог нового пользователя. Результирующие файлы также принадлежат новому пользователю. Используя параметр `-k` с помощью команды `useradd`, содержимое другого каталога можно использовать для заполнения домашнего каталога нового пользователя. При указании каталога скелета с опцией `-k` необходимо использовать параметр `-m`, иначе команда `useradd` завершится с ошибкой, указав «флаг `-k` разрешен только с флагом `-m`».

После того, будут выбраны значения по умолчанию, и собрана информация о пользователе, можно создать учетную запись пользователя. Пример команды `useradd`, использующий несколько параметров, будет выглядеть следующим образом:

```
root@localhost:~# useradd -u 1000 -g users -G wheel,research -c 'Jane Doe' jane
```

Этот пример команды `useradd` создает пользователя с идентификатором 1000, основной группой `users`, дополнительными членствами в группах `wheel` и `research`, комментарием «Jane Doe» и именем учетной записи `jane`.

Информация о учетной записи пользователя `jane` будет автоматически добавлена в файлы `/etc/passwd` и `/etc/shadow`, Информация о дополнительном групповом доступе будет автоматически добавлена в файлы `/etc/group` и `/etc/gshadow`. Например:

```

root@localhost:~# useradd -u 1000 -g users -G wheel,research -c "Jane Doe" jane
root@localhost:~# grep jane /etc/passwd
jane:x:1000:100:Jane Doe:/home/jane:/bin/bash
root@localhost:~# grep jane /etc/shadow
jane:!!:16003:0:99999:7:::
root@localhost:~# grep jane /etc/group
wheel:x:10:jane
research:x:2000:jane
root@localhost:~# grep jane /etc/gshadow
wheel:::jane
research:::jane
root@localhost:~#

```

Нужно обратить внимание, что учетная запись еще не имеет действительного пароля!

Кроме того, будет создан файл почты /var/spool/mail/jane, каталог /home/jane будет создан с разрешениями, разрешающими доступ только пользователю jane, а содержимое каталога /etc/skel будет скопировано в этот каталог:

```

root@localhost:~# ls /var/spool/mail
jane root rpc sysadmin
root@localhost:~# ls /home
jane sysadmin
root@localhost:~# ls -a /home/jane
.  ..  .bash_logout  .bashrc  .profile  .selected_editor
root@localhost:~# ls -a /etc/skel
.  ..  .bash_logout  .bashrc  .profile  .selected_editor
root@localhost:~#

```

10.16. Выбор пароля

Выбор хорошего пароля - непростая задача, но очень важно, чтобы это было сделано правильно или безопасность учетной записи (возможно, всей системы) может быть скомпрометирована. Выбор хорошего пароля - это только начало; нужно быть очень осторожным с паролем, чтобы он не был показан другим людям. Никогда не нужно рассказывать кому-либо свой пароль и никогда не позволять никому его видеть, в том числе при вводе. Если есть желание записать свой пароль, нужно безопасно его хранить в таком месте, как сейф.

Легко сделать плохой пароль! Если пользователь использует какую-либо информацию в своем пароле, которая связана с ним, эта информация также может быть известна или обнаружена другими, в результате чего пароль легко будет скомпрометирован.

Есть множество факторов, которые следует учитывать при попытке выбрать пароль для учетной записи:

Длина: файл `/etc/login.defs` позволяет администратору указывать минимальную длину пароля. Хотя некоторые считают, что чем дольше пароль, тем лучше, это не совсем правильно. Проблема со слишком длинными паролями заключается в том, что их сложно запомнить и, как результат, часто записываются в местах, где их легко найти и скомпрометировать.

Состав: хороший пароль должен состоять из комбинации алфавитных, числовых и символьных знаков.

Продолжительность жизни: количество времени, которое пароль может быть использован должно быть ограничено по нескольким причинам:

Если учетная запись скомпрометирована и время, когда пароль действителен, ограничен, злоумышленник в конечном итоге потеряет доступ, поскольку пароль в конечном итоге станет недействительным.

Если учетная запись не используется, ее можно автоматически отключить, если пароль больше недействителен.

Если злоумышленники пытаются атаковать «грубой силой», пытаясь использовать все возможные пароли, тогда пароль может быть изменен до того, как атака будет успешной.

Однако, требуя от пользователя слишком часто менять свой пароль, могут возникнуть проблемы с безопасностью, в том числе:

Качество пароля, которое пользователь выбирает, может пострадать.

Пользователь может начать писать свой пароль на бумаге, увеличивая вероятность того, что пароль может быть обнаружен.

Редко используемые учетные записи пользователей могут быть с истекшими паролями и требуют административного внимания при перезагрузке.

Мнения различаются о том, как часто пользователи вынуждены менять свои пароли. Для высокочувствительных учетных записей рекомендуется чаще менять пароли, например, каждые 30 дней. С другой стороны, для некритических учетных записей, не имеющих доступа к конфиденциальной информации, меньше требуется частых изменений. Для учетных записей с минимальным риском продолжительность 90 дней считается более разумной.

10.17. Установка пароля

Существует несколько способов изменения пароля пользователя: пользователь может выполнить команду `passwd`, администратор может выполнить команду `passwd`, предоставляющую имя пользователя в качестве аргумента, а также доступны графические инструменты.

Администратор может использовать команду `passwd` для установки начального пароля или изменения пароля для учетной записи. Например, если администратор создал учетную запись `jane`, тогда выполнение `passwd jane` предоставит администратору приглашение установить пароль для учетной записи `jane`. Если оно завершено успешно, файл `/etc/shadow` будет обновлен с новым паролем пользователя.

Хотя обычные пользователи должны следовать многим правилам паролей, пользователь `root` должен следовать только одному правилу: пароль не может быть пустым. Все другие правила пароля, которые нарушает пользователь `root`, просто приведут к тому, что на экран будет выведено предупреждение, а правило не будет принудительно выполнено:

```
root@localhost:~# passwd jane
Enter new UNIX password:
BAD PASSWORD: it is WAY to short
BAD PASSWORD: is too simple
Retype new UNIX password:
passwd: password updated successfully
root@localhost:~#
```

Предполагая, что администратор установил пароль для учетной записи пользователя, пользователь может войти в систему с именем учетной записи и паролем. После того, как пользователь открывает терминал, он может выполнить команду `passwd` без аргументов, чтобы изменить свой собственный пароль. Будет предложено ввести текущий пароль, а затем будет предложено ввести новый пароль дважды.

Для обычного пользователя может быть сложно установить действительный пароль, потому что должны соблюдаться все правила для пароля. Пользователю обычно разрешено три попытки предоставить действительный пароль до того, как команда `passwd` завершится с ошибкой.

Используя привилегии пользователя `root`, зашифрованные пароли и другую информацию, связанную с паролем, можно просмотреть в

файле `/etc/shadow`. Обычные пользователи не могут видеть содержимое этого файла.

10.18. Команда `chage`

Команда `chage` предоставляет множество опций для управления информацией о пароле, находящейся в файле `/etc/shadow`.

Хорошим примером команды `chage` было бы изменить максимальное количество дней, в течение которых пароль пользователя действителен, равный 60 дням:

```
root@localhost:~# chage -M 60 jane
```

10.19. Изменение пользователя

Прежде чем вносить изменения в учетную запись пользователя, нужно знать, что некоторые команды не смогут успешно изменить учетную запись пользователя, если пользователь в настоящий момент зарегистрирован (например, при изменении имени пользователя).

Другие изменения, которые можно сделать, не будут эффективны, если пользователь уже вошёл в систему, но вступят в силу, как только пользователь выйдет из системы, а затем снова войдет в неё. Например, если изменяется членство в группах, то новые членства будут недоступны для пользователя до следующего входа пользователя в систему.

В любом случае полезно знать, как использовать команды `who`, `w` и `last`, чтобы знать, кто и когда входит в систему, поскольку это может повлиять на изменения, которые производятся для пользователя.

Команды `who` и `w` обсуждались в предыдущей главе. Обе команды позволяют видеть, кто в настоящее время входит в систему. Команда `w` на самом деле более сложная из двух, так как она показывает информацию о времени работы и загрузки системы, а также о том, какой процесс выполняется каждым пользователем.

Команда `last` немного отличается от команд `who` и `w`. По умолчанию он также показывает имя пользователя, терминал и местоположение входа, при этом не только текущие сеансы входа в систему, но и предыдущие сеансы. В отличие от команд `who` и `w`, команда `last` отображает дату/время, когда пользователь был зарегистрирован в системе. Если пользователь вышел из системы, тогда отобразится общее время,

в течение которого пользователь находился в системе, или отобразит «все еще зарегистрированный».

Команда `last` будет считывать всю историю входа из файла `/var/log/wtmp` и отображать все записи и перезагружать записи по умолчанию. Интересной деталью записей перезагрузки является то, что она отображает версию ядра Linux, которая была загружена, а не место входа в систему.

Предоставляя имя пользователя или имя `tty` (terminal) в качестве аргумента, команда будет отображать только записи, соответствующие этому имени. Если нужно выяснить, кто был зарегистрирован на определенную дату и время, последняя команда может это отобразить, если используется опция `-t`, чтобы указать эту дату и время.

Команда `usermod` предлагает множество вариантов изменения существующей учетной записи пользователя. Большинство этих параметров также доступны с помощью команды `useradd` при создании учетной записи.

Некоторые из этих вариантов заслуживают обсуждения из-за того, как они влияют на управление пользователями. Очень сложно изменить `UID` пользователя с параметром `-u`, так как любые файлы, принадлежащие пользователю, будут потеряны. С другой стороны, указание нового имени пользователя для пользователя с `-l` не приводит к тому, что файлы становятся “сиротами”.

Удаление пользователя с помощью команды `userdel` (см. Следующий раздел) может привести к “сироте” или удалению файлов пользователя в системе. Вместо удаления учетной записи другой вариант - заблокировать учетную запись с помощью опции `-L` для команды `usermod`. Блокировка учетной записи запрещает использование учетной записи, но сохраняется право собственности на файлы.

Есть несколько важных вещей, которые нужно знать об управлении дополнительными группами. Если использовать опцию `-G` без опции `-a`, нужно перечислить все группы, к которым будет принадлежать пользователь. Использование опции `-G` может привести к случайному удалению пользователя из всех прежних дополнительных групп, к которым принадлежал пользователь.

Если используется параметр `-a` с `-G`, нужно только указать новые группы, к которым будет принадлежать пользователь. Например, если пользователь `jane` в настоящее время принадлежит группам `wheel` и

research, то для добавления ее учетной записи в группу development нужно выполнить следующую команду:

```
root@localhost:~# usermod -aG development jane
```

10.20. Удаление пользователя

Когда удаляется учетная запись пользователя, необходимо решить, следует ли удалять домашний каталог пользователя. Файлы пользователя могут быть важны для организации, и могут быть даже юридические требования для хранения данных на определенное время, поэтому нужно быть осторожными, чтобы не принимать это решение опрометчиво. Кроме того, если не сделали резервные копии данных, как только выполнили команду для удаления пользователя и его файлов, это приведет к необратимым последствиям.

Чтобы удалить пользователя jane без удаления домашнего каталога пользователя (/home/jane), можно выполнить:

```
root@localhost:~# userdel jane
```

Нужно помнить, что удаление пользователя без удаления его домашнего каталога означает, что файлы домашнего каталога пользователя теперь будут потеряны, и эти файлы будут принадлежать только прежним UID и GID.

Чтобы удалить пользователя jane и удалить домашний каталог /home/jane, используется параметр -r:

```
root@localhost:~# userdel -r jane
```

Вышеупомянутая команда будет удалять только файлы пользователя в их домашнем каталоге и их почтовой папке. Если пользователь владеет другими файлами вне своего домашнего каталога, эти файлы будут продолжать существовать как потерянные файлы.

Контрольные вопросы

1. Какой файл содержит зашифрованную информацию о пароле пользователя?
2. Какая команда отобразит UID, GID и группы, к которым принадлежит текущий пользователь?
3. Какие файлы содержат информацию об учетной записи пользователя?

4. Какой пользователь по умолчанию используется для команды `su`?
5. Какая команда позволит пользователю выполнять одиночную команду как от пользователя `root`?
6. Для каких пользователей обычно зарезервированы UID 1–499?
7. Какую команду можно использовать для изменения параметров учетной записи пользователя?
8. Какая команда отобразит пользователей, которые в данный момент вошли в систему?
9. Какой пользователь может просматривать файл `/etc/shadow`?
10. Что произойдет с файлами и каталогами пользователя, учетная запись которого была удалена из системы?
11. Когда стартовала «Эпоха»?

Глава 11. ПРАВА ДОСТУПА

11.1. Владелец файла

Владение файлом имеет решающее значение для безопасности файлов. Каждый файл имеет владельца и владельца группы.

По умолчанию пользователи будут владеть файлами, которые они создают. Хотя это право собственности может быть изменено, для этой функции требуются административные привилегии. Хотя большинство команд обычно показывают владельца пользователя в качестве имени, операционная система фактически связывает право пользователя с UID для этого имени пользователя.

У каждого файла также есть владелец группы. В предыдущей главе о создании пользователей и групп обсуждалась основная группа пользователя. По умолчанию основной группой пользователя, создавшего файл, будет владелец группы любых новых файлов. Пользователям разрешено изменять владельца группы файла в любой группе, к которой они принадлежат. Подобно пользовательской собственности, объединение файла с группой фактически выполняется внутри операционной системы не по имени, а по GID группы.

Поскольку право собственности фактически определяется идентификаторами UID и GID, связанными с файлом, изменение UID пользователя (или удаление пользователя) приводит к тому, что файл, который изначально принадлежал этому пользователю, не имеет реального пользователя. Если в файле `/etc/passwd` нет UID, который соответствует UID владельца файла, тогда UID (номер) будет отображаться вместо имени владельца. То же самое происходит для групп.

Команда `id` может быть полезна для проверки того, какую учетную запись пользователь использует и в какие группы входит. Просмотрев вывод этой команды, можно увидеть идентификационную информацию пользователя, выраженную как в числовой, так и символьной форме в виде имен.

Результат команды `id` отображает UID и имя учетной записи текущего пользователя, за которым следуют GID и имя группы первичной группы, а также идентификаторы GID и имена групп, в которые входит пользователь:

```
sysadmin@localhost:~$ id
uid=500(sysadmin) gid=500(sysadmin) groups=500(sysadmin),10001(research),10002(development) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

В приведенном выше примере показано, что у пользователя есть UID 500 и имя учетной записи sysadmin. Основная группа для этого пользователя имеет GID 500 для группы sysadmin. Поскольку учетная запись пользователя и первичная групповая учетная запись имеют одинаковый числовой идентификатор и одно и то же имя, это означает, что этот пользователь находится в пользовательской частной группе (UPG). Кроме того, пользователь принадлежит к двум дополнительным группам: research с GID 10001 и development с GID 10002.

Когда файл создается, он будет принадлежать текущему пользователю и его первичной группе. Если пользователь из приведенного выше примера должен был выполнить команду, например touch, чтобы создать файл, а затем просмотреть сведения о файле, результат будет выглядеть следующим образом:

```
sysadmin@localhost:~$ touch /tmp/filetest1
sysadmin@localhost:~$ ls -l /tmp/filetest1
-rw-rw-r--. 1 sysadmin sysadmin 0 Oct 21 10:18 /tmp/filetest1
```

Владельцем и группой владельца является sysadmin.

11.2. Команды newgrp и groups

Если создаваемый файл, должен принадлежать группе, отличной от текущей основной группы, то можно использовать команду newgrp для изменения текущей основной группы пользователя.

Как показано ранее, команда id будет перечислять идентификационную информацию, включая членство в группе. Если интересует только то, к каким группам пользователь принадлежит, можно выполнить команду groups:

```
sysadmin@localhost:~$ groups
sysadmin research development
```

Вывод команды groups может быть не таким подробным, как вывод команды id, но если все, что нужно знать, это то, по каким группам можно переключаться с помощью команды newgrp, команда groups предоставляет необходимую информацию. Вывод команды id показывает текущую основную группу, поэтому полезно проверить, что команда newgrp выполнена успешно.

Например, если пользователь `sysadmin` планировал иметь файл, принадлежащий группе `research`, но это не была основная группа пользователя, пользователь мог использовать команду `newgrp`, а затем проверить правильную основную группу с помощью команды `id` перед созданием нового файла:

```
sysadmin@localhost:~$ id
uid=502(sysadmin) gid=503(sysadmin) groups=503(sysadmin),10001(research),10002(development) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
sysadmin@localhost:~$ newgrp research
sysadmin@localhost:~$ id
uid=502(sysadmin) gid=10001(research) groups=503(sysadmin),10001(research),10002(development) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Из вывода предыдущих команд видно, что сначала GID пользователя равен 503 (`sysadmin`), после выполняется команда `newgrp research`, а затем идентификатор основной группы пользователя становится равным 10001 (`research`). После выполнения этих команд, если пользователь должен был создать новый файл и просмотреть его данные, новый файл будет принадлежать группе `research`:

```
sysadmin@localhost:~$ touch /tmp/filetest2
sysadmin@localhost:~$ ls -l /tmp/filetest2
-rw-r--r--. 1 sysadmin research 0 Oct 21 10:53 /tmp/filetest2
```

Команда `newgrp` открывает новую оболочку; пока пользователь остается в этой оболочке, первичная группа не изменится. Чтобы переключить основную группу на исходную, пользователь может покинуть новую оболочку, выполнив команду `exit`. Например:

```
sysadmin@localhost:~$ id
uid=502(sysadmin) gid=10001(research) groups=503(sysadmin),10001(research),10002(development) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
sysadmin@localhost:~$ exit
exit
sysadmin@localhost:~$ id
uid=502(sysadmin) gid=503(sysadmin) groups=503(sysadmin),10001(research),10002(development) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Административные привилегии необходимы для постоянного изменения основной группы пользователя. Для этого пользователь `root` должен выполнить команду `usermod -g имя_пользователя`.

11.3. Команды `chgrp` и `stat`

Если нужно изменить принадлежность группы к существующему файлу, можно использовать команду `chgrp`. От лица пользователя без административных привилегий команда `chgrp` может использоваться

только для изменения владельца группы файла в группу, в которой пользователь уже является членом. В качестве пользователя root команда `chgrp` может использоваться для изменения владельца группы любого файла в любой группе.

Хотя можно просмотреть право собственности на файл командой `ls` с параметром `-l`, система предоставляет другую команду, которая полезна при просмотре прав доступа к файлам и директориям. Команда `stat` отображает более подробную информацию о файле, включая предоставление группового владения как по имени группы, так и по номеру GID:

```
sysadmin@localhost:~$ stat /tmp/filetest1
  File: `/tmp/filetest1'
  Size: 0                Blocks: 0                IO Block: 4096   regular empty file
Device: fd00h/64768d    Inode: 31477             Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 502/sysadmin)   Gid: ( 503/sysadmin)
Access: 2020-10-21 10:18:02.809118163 -0700
Modify: 2020-10-21 10:18:02.809118163 -0700
Change: 2020-10-21 10:18:02.809118163 -0700
```

На следующем рисунке показано, как пользователь `sysadmin` меняет принадлежность группы к файлу:

```
sysadmin@localhost:~$ chgrp development /tmp/filetest1
sysadmin@localhost:~$ stat /tmp/filetest1
  File:
`/tmp/filetest1'
  Size: 0                Blocks: 0                IO Block: 4096   regular empty file
Device: fd00h/64768d    Inode: 528677           Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 500/sysadmin)   Gid: ( 502/development)
Access: 2020-10-21 10:18:02.809118163 -0500
Modify: 2020-10-21 10:18:02.809118163 -0500
Change: 2020-10-21 10:18:02.809118163 -0500
sysadmin@localhost:~$
```

Если пользователь пытается изменить принадлежность группы к файлу, которым пользователь не владеет, он получит сообщение об ошибке:

```
sysadmin @ localhost: ~ $ chgrp development /etc/passwd
chgrp: изменение группы «/etc/passwd»: операция не разрешена
```

Иногда нужно иметь возможность не только изменять файлы в текущем каталоге, но и файлы в подкаталогах. При выполнении с параметром `-R` (рекурсивный) команда `chgrp` будет работать не только в текущем каталоге, но и во всех каталогах, которые могут быть вложены под указанным каталогом. Эта операция также затронет все файлы в подкаталогах, а не только сами каталоги.

В следующем примере показано использование опции `-R`:

```

sysadmin@localhost:~$ cp -r /etc/sound .
sysadmin@localhost:~$ ls -ld sound
drwxr-xr-x 1 sysadmin sysadmin 0 Dec 11 02:02 sound
sysadmin@localhost:~$ ls -lR sound
sound:
total 4
drwxr-xr-x. 2 sysadmin sysadmin 4096 Oct 28 13:06 events

sound/events:
total 48
-rw-r--r--. 1 sysadmin sysadmin 27223 Oct 28 13:06 gnome-2.soundlist
-rw-r--r--. 1 sysadmin sysadmin 18097 Oct 28 13:06 gtk-events-2.soundlist
sysadmin@localhost:~$ chgrp -R development sound
sysadmin@localhost:~$ ls -ld sound
drwxr-xr-x. 3 sysadmin development 4096 Oct 28 13:06 sound
ls -lR sound
sound:
total 4
drwxr-xr-x. 2 sysadmin development 4096 Oct 28 13:06 events

sound/events:
-rw-r--r--. 1 sysadmin development 27223 Oct 28 13:06 gnome-2.soundlist
-rw-r--r--. 1 sysadmin development 18097 Oct 28 13:06 gtk-events-2.soundlist
sysadmin@localhost:~$

```

11.4. Команда chown

Команда `chown` позволяет пользователю `root` изменять права пользователя на файлы и каталоги. Обычный пользователь не может использовать эту команду для изменения владельца файла, даже для передачи права собственности на один из своих файлов другому пользователю. Однако команда `chown` также позволяет изменять групповое владение, которое может быть выполнено либо `root`, либо владельцем файла.

Существует три разных способа выполнения команды `chown`. Первый метод используется для изменения только владельца пользователя файла. Например, если пользователь `root` хотел изменить права пользователя на файл `abc.txt` на пользователя `ted`, тогда может быть выполнена следующая команда:

```
root@localhost:~# chown ted abc.txt
```

Второй способ - изменить как пользователя, так и группу; это также требует привилегий `root`. Чтобы выполнить это, нужно разделить пользователя и группу символом двоеточия или точкой. Например:

```

root@localhost:~# chown user:group /path/to/file
root@localhost:~# chown user.group /path/to/file

```

Если у пользователя нет привилегий root, можно использовать третий метод для изменения группы владельцев файла так же, как командой `chgrp`. Чтобы использовать `chown` для изменения группового владения файлом, используется двоеточие или точка в качестве префикса для имени группы:

```
root@localhost:~# chown :group /path/to/file
root@localhost:~# chown .group /path/to/file
```

11.5. Разрешения

Когда выполняется команда `ls -l`, итоговый вывод отображает десять символов в начале каждой строки, которые указывают тип файла и разрешения файла:

- Первый символ указывает тип файла.
- Символы 2-4 указывают разрешения для пользователя, которому принадлежит файл.
- Символы 5-7 указывают разрешения для группы, которая владеет файлом.
- Символы 8-10 указывают разрешения для «других» или то, что иногда называют разрешениями в мире. Это будет включать всех пользователей, которые не являются владельцем файла или членом группы владельцев файла.

Например, рассмотрим вывод следующей команды:

```
root@localhost:~# ls -l /etc/passwd
-rw-r--r--. 1 root root 4135 May 27 21:08 /etc/passwd
```

Обычный пользователь никогда не столкнется с чем-либо, кроме обычных файлов каталогов и ссылок, если не начнет изучать каталог `/dev`.

Символы в разрешающей части вывода имеют следующие значения:

- `r` означает разрешение на чтение
- `w` означает разрешение на запись
- `x` означает разрешение на выполнение

Применение прав для файлов и директорий показано в таблице 23.

Таблица 23

Права	Файл	Директория
Read	Прочитать содержимое	Отобразить список содержимого
Write	Изменить содержимое	Разрешить создание, удаление файлов или задание прав на файлы
Execute	Запустить как программу	Переход в каталог

Для файлов: r - право на чтение из файла; w - разрешает запись в файл (в частности перезапись или изменение); x - позволяет исполнить файл.

Для каталогов, флаги r w x имеют несколько отличный смысл: r - позволяет читать только имена файлов в каталоге; x - позволяет иметь доступ к самим файлам и их атрибутам (но не именам); w имеет смысл только в сочетании с x, и позволяет (в дополнение к x) манипулировать с файлами в каталоге (создавать, удалять и переименовывать). w без x - не имеет никакого эффекта.

Разрешения, установленные для файлов, определяют уровень доступа, который пользователь будет иметь к файлу. Когда пользователь запускает программу, и программа обращается к файлу, то разрешения проверяются, чтобы определить, имеет ли пользователь правильные права доступа к файлу.

Если пользователь является владельцем файла или директории, то для определения доступа к этому файлу / директории используются только разрешения владельца.

Если пользователь не является владельцем, но является членом группы, которая владеет файлом / директорией, то для определения доступа к этому файлу / директории используются только права группы владельцев.

Если пользователь не является владельцем и не является членом группы владельцев файлов / каталогов, его разрешения будут «другими».

Команда `chmod` используется для изменения разрешений. Существует две техники, которые могут использоваться с этой командой: символическая и числовая. Оба метода используют следующий базовый синтаксис:

```
chmod новые_разрешения file_name
```


Если нужно изменить некоторые из существующих разрешений, символический метод, скорее всего, будет проще в использовании. С помощью этого метода указывается, какие разрешения нужно изменить в файле, а остальные разрешения остаются такими, какие они есть.

При указании разрешений параметр `новые_разрешения` начинается с использования одного из следующих символов, чтобы указать, какой набор разрешений изменить:

- `u` = изменить права владельца (**u**ser)
- `g` = изменить права группы владельцев (**g**roup)
- `o` = изменить «другие» разрешения (**o**thers)
- `a` = применить изменения ко всем наборам разрешений (**a**ll, владелец, группа владельцев и другие)

Затем указывается символ `+` для добавления разрешения или символ `-` для удаления разрешения. В конце указывается `r` для чтения, `w` для записи и `x` для выполнения.

Например, чтобы предоставить пользователю права на чтение для файла с именем `abc.txt`, можно использовать следующую команду:

```
root@localhost:~# chmod u+r abc.txt
```

Можно комбинировать значения, чтобы сделать несколько изменений в разрешениях файла. Например, следующая команда добавит права на чтение владельцу и группе при удалении разрешения на запись для «других»:

```
root@localhost:~# chmod ug+r,o-w abc.txt
```

Можно использовать символ `=` вместо `-` или `+`, чтобы точно указать разрешения, необходимые для набора разрешений:

```
root@localhost:~# chmod u=r-x abc.txt
```

Когда для изменения разрешений используется числовой метод, необходимо указать все девять разрешений. Из-за этого символический метод, как правило, проще для изменения нескольких разрешений, в то время как числовой метод лучше подходит для более радикальных изменений.

Цифровой формат кодирует символ `r` цифрой 4 (100 в двоичном формате), символ `w` — цифрой 2 (010 в двоичном формате), а символ `x` — цифрой 1 (001 в двоичном формате). Для получения комбинации `r,w,x` надо суммировать значения.

$rw\bar{x}=111_2=7_8$, $rw\bar{-}=110_8=6_8$, $r\bar{-}x=101_2=5_8$, $r\bar{-}\bar{-}=100_2=4_8$, $\bar{-}wx=011_2=3_8$, $\bar{-}w\bar{-}=010_2=2_8$, $\bar{-}\bar{-}x=001_2=1_8$.

Например, $rw\bar{-} r\bar{-} r\bar{-} = 110\ 100\ 100_2 = 644$, $rw\bar{x} r\bar{-}x r\bar{-}x = 111\ 101\ 101_2 = 755$

В таблице 24 показаны все возможные комбинации прав.

Таблица 24

Восьмеричный	Бинарный	Символьный	Права на файл	Права на каталог
0	000	- - -	отсутствие прав	отсутствие прав
1	001	- - x	права на выполнение	доступ к файлам и их атрибутам
2	010	- w -	права на запись	отсутствие прав
3	011	- w x	права на запись и выполнение	все, кроме доступа к именам файлов
4	100	r - -	права на чтение	только чтение имен файлов
5	101	r - x	права на чтение и выполнение	чтение имен файлов и доступ файлам и их атрибутам
6	110	r w -	права на чтение и запись	только чтение имен файлов
7	111	r w x	полные права	все права

Чтобы установить права доступа к файлу с именем abc.txt как $rw\bar{x}r\bar{-}x\bar{-}$, можно использовать следующую команду:

```
root@localhost:~]# chmod 754 abc.txt
```

11.6. umask

Команда `umask` — это функция, которая используется для определения разрешений по умолчанию, которые устанавливаются при создании файла или каталога. Разрешения по умолчанию определяются, когда значение `umask` вычитается из максимально допустимых разрешений по умолчанию.

Разрешения, которые первоначально устанавливаются в файле при его создании, не могут превышать $rw\bar{-}rw\bar{-}rw\bar{-}$ (666) — это максимально допустимые разрешения по умолчанию для файла. Для директории максимум по умолчанию 777. Чтобы иметь разрешение на выполнение в файле, сначала необходимо создать файл, а затем изменить

разрешения. Если при этом установлена `umask = 002`, то надо побитно вычесть из 666 маску 002 и получатся права вновь создаваемого файла 664. Если `umask = 026`, то файл получит разрешения 640.

Команда `umask` может использоваться для отображения текущего значения `umask`:

```
sysadmin@localhost:~$ umask
0002
```

Разные пользователи могут иметь разные маски `umask`. Обычно пользователь `root` будет иметь более ограничительную `umask`, чем обычные учетные записи пользователей:

```
root@localhost:~# umask
0022
```

`Umask 027` означает, что по умолчанию новые файлы получают разрешения 640 или `rw-r-----`, как показано ниже:

```
sysadmin@localhost:~$ umask 027
sysadmin@localhost:~$ touch sample
sysadmin@localhost:~$ ls -l sample
-rw-r-----. 1 sysadmin sysadmin 0 Oct 28 20:14 sample
```

Маска 027 означает, что по умолчанию файлы каталогов будут получать разрешения 750 или `rwxr-x---`, как показано ниже:

```
sysadmin@localhost:~$ umask 027
sysadmin@localhost:~$ mkdir test-dir
sysadmin@localhost:~$ ls -ld test-dir
drwxr-x---. 1 sysadmin sysadmin 4096 Oct 28 20:25 test-dir
```

Новое значение `umask` будет применяться только к файлам и каталогам, созданным в течение текущего сеанса. Когда запускается новая оболочка, снова будет введен `umask` по умолчанию.

Постоянное изменение пользовательской `umask` требует изменения файла `.bashrc`, находящегося в домашнем каталоге этого пользователя.

11.7. Параметр `setuid`

Когда для исполняемого двоичного файла (программы) устанавливается разрешение `setuid`, файл «запускается от лица» владельца файла, а не от лица пользователя, который его выполняет. Это разрешение настроено на несколько системных утилит, чтобы их могли за-

пускать обычные пользователи, но выполнялись с правами root, обеспечивая доступ к системным файлам, к которым обычный пользователь доступа не имеет.

Рассмотрим следующий сценарий, в котором пользователь sysadmin пытается просмотреть содержимое файла /etc/shadow:

```
sysadmin@localhost:~$ more /etc/shadow
/etc/shadow: Permission denied
sysadmin@localhost:~$ ls -l /etc/shadow
-rw-r----- 1 root root 5195 Oct 21 19:57 /etc/shadow
```

Как можно видеть, файл /etc/shadow не может быть просмотрен (или изменен) обычными пользователями, поскольку разрешения файла: -rw-r-----. Файл принадлежит пользователю root, поэтому системный администратор может временно изменить разрешения в случае, если пользователи хотят просмотреть или изменить этот файл.

Теперь рассмотрим команду passwd. Когда эта команда запускается, она изменяет файл /etc/shadow. Это кажется невозможным, потому что другие команды, которые запускает пользователь sysadmin, пытающиеся получить доступ к этому файлу, не работают. Итак, почему пользователь sysadmin может изменить файл /etc/shadow при запуске команды passwd, когда обычно у этого пользователя нет доступа к файлу?

Команда passwd имеет специальное разрешение setuid. Когда команда passwd запускается, и команда обращается к файлу /etc/shadow, система действует так, как будто пользователь, обращающийся к файлу, является владельцем команды passwd (пользователь root), а не пользователем, который фактически выполняет команду.

Этот набор разрешений можно увидеть, запустив команду ls -l:

```
sysadmin@localhost:~$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 31768 Jan 28 2010 /usr/bin/passwd
```

Разрешение setuid представлено символом s в разрешениях владельца, где обычно будет представлено разрешение на выполнение.

Подобно разрешениям чтения, записи и выполнения, специальные разрешения могут быть установлены с помощью команды chmod, используя либо символический, либо восьмеричный методы.

Когда предоставляется трехзначный код, команда chmod предполагает, что первая цифра перед трехзначным кодом равна 0. Только если указаны четыре цифры, будет установлено специальное разрешение.

Если при изменении разрешения на файл, который уже имеет специальный набор разрешений, указаны три цифры, первая цифра будет установлена в 0, и специальное разрешение будет удалено из файла.

11.8. setgid разрешение для файла

Разрешение setgid аналогично setuid, но оно использует разрешения владельца группы. Существуют две формы разрешений: setgid для файла и setgid для каталога. Как работает setgid, зависит от того, установлен ли он в файле или каталоге.

Разрешение setgid в файле очень похоже на setuid. Он позволяет пользователю запускать исполняемый двоичный файл таким образом, который предоставляет ему дополнительный (временный) доступ к группе. Система позволит пользователю, выполняющему команду, эффективно принадлежать группе, которая владеет файлом, но только в программе с установленным setgid. Поскольку команда выполняет и обращается к файлам, это дополнительное членство в группе может предоставлять доступ к файлам.

Хорошим примером разрешения setgid для исполняемого файла является команда /usr/bin/wall:

```
sysadmin@localhost:~$ ls -l /usr/bin/wall
-rwxr-sr-x. 1 root tty 10996 Jul 19 2020 /usr/bin/wall
```

Можно увидеть, что для этого файла установлен символ s в позиции выполнения для группы. Из-за этого исполняемого файла, принадлежащего группе tty, когда пользователь выполняет эту команду, команда будет иметь доступ к файлам, которые принадлежат группе tty.

Этот доступ важен, потому что команда /usr/bin/wall отправляет сообщения на «терминалы». Это достигается путем записи данных в файлы следующим образом:

```
sysadmin@localhost:~$ ls -l /dev/tty?
crw--w----. 1 root tty 4, 0 Mar 29 2020 /dev/tty0
crw--w----. 1 root tty 4, 1 Oct 21 19:57 /dev/tty1
```

Группа tty имеет разрешение на запись в файлы выше, тогда как пользователи, не входящие в группу tty («другие»), не имеют прав на эти файлы. Без разрешения setgid команда /usr/bin/wall завершится с ошибкой.

11.9. setgid разрешение для директории

Разрешение `setgid` также может быть установлено в каталоге. Когда установлено в каталоге, `setgid` заставляет файлы, созданные в каталоге, автоматически принадлежать группе, которая владеет каталогом. Это противоречит тому, как обычно будет функционировать новое владение файловой группой, так как по умолчанию новые файлы принадлежат группе первичной группы пользователя, создавшего файл.

Кроме того, любые каталоги, созданные в каталоге, который имеет установленный `setgid`-разрешение, будут принадлежать не только группе, которая владеет каталогом `setgid`, но и в новом каталоге автоматически устанавливается `setgid`. Другими словами, если каталог `setgid`, то любые каталоги, созданные в этом каталоге, наследуют разрешение `setgid`.

Следующее демонстрирует функцию разрешения `setgid` в каталоге. Пользователь `sysadmin` создает файл в каталоге `/tmp/data`. Группа-владелец нового файла не группа `sysadmin`, а группа `demo`:

```
sysadmin@localhost:~$ ls -ld /tmp/data
drwxrwsrwx. 2 root demo 4096 Oct 30 23:20 /tmp/data
sysadmin@localhost:~$ id
uid=500(sysadmin) gid=500(sysadmin)
groups=500(sysadmin),10001(research),10002(development)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
sysadmin@localhost:~$ touch /tmp/data/file.txt
sysadmin@localhost:~$ ls -ld /tmp/data/file.txt
-rw-rw-r--. 1 sysadmin demo 0 Oct 30 23:21 /tmp/data/file.txt
```

Итак, почему администратор настраивает каталог с параметром `setgid`? Для примера рассмотрим следующие учетные записи пользователей: `ivan`, `alex` и `tim`. Этим трем пользователям необходимо работать над совместным проектом. Они обращаются к администратору, чтобы создать общий каталог, в котором они могут работать вместе. Администратор делает следующее:

- Создает новую группу, называемую `team`.
- Добавляет `ivan`, `alex` и `tim` к группе `team`.
- Создает новый каталог `/home/team`.
- Делает группу владельца каталога `/home/team` группой `team`.
- Предоставляет директории `/home/team` следующие разрешения: `rw-rwx---`

В результате, ivan, alex и tim могут обращаться к директории /home/team и добавлять файлы. Однако существует потенциальная проблема: когда ivan создает файл в каталоге /home/team, новые данные файла выглядят следующим образом:

```
-rw-r-----. 1 ivan ivan 100 Oct 30 23:21 /home/team/file.txt
```

К сожалению, в то время как alex и tim могут обращаться к директории /home/team, они ничего не могут сделать с файлом Ивана. Их разрешения для этого файла — это разрешение «других» (---).

Если администратор установил разрешение setgid в каталог /home/team, когда ivan создает файл, он выглядит следующим образом:

```
-rw-r-----. 1 ivan team 100 Oct 30 23:21 /home/team/file.txt
```

И, как результат, alex и tim имели бы групповой доступ к файлу (r--).

Разумеется, ivan может изменить владельца группы после создания файла (или изменить «другие» разрешения), но это стало бы утомительным, если бы появилось много новых файлов. Разрешение setgid облегчает эту ситуацию.

11.10. Настройка разрешений setgid

Используется следующий синтаксис для добавления разрешения setgid символически:

```
chmod g+s <file|directory>
```

Чтобы добавить разрешение setgid численно, нужно добавить 2000 к существующим разрешениям файла (предположим в следующем примере, что изначально у каталога было разрешение 775):

```
chmod 2775 <file|directory>
```

Чтобы удалить разрешение setgid символически, нужно выполнить:

```
chmod g-s <file|directory>
```

Чтобы удалить разрешение setgid численно, нужно вычесть 2000 из существующих разрешений файла:

```
chmod 0775 <file|directory>
```

11.11. Липкий бит

Разрешение липкого бита (sticky bit) используется, чтобы другие пользователи не могли удалять файлы, которыми они не владеют в общем каталоге. Любой пользователь с разрешением на запись в каталоге может создавать файлы в этом каталоге, а также удалять любые файлы в каталоге, даже если он не владеет файлом!

Доступ с липким битом позволяет передавать файлы другим пользователям, изменяя права на запись в каталоге, чтобы пользователи могли добавлять и удалять файлы в каталоге, но файлы могут быть удалены только владельцем файла или пользователем root.

Чтобы символически установить разрешение липкого бита, нужно выполнить команду, подобную следующей:

```
chmod o+t <directory>
```

Чтобы установить разрешение липкого бита численно, нужно добавить 1000 к существующим разрешениям каталога (предположим, что каталог в следующем примере изначально имел разрешения 775):

```
chmod 1775 <file|directory>
```

Чтобы символически удалить липкое разрешение, нужно запустить:

```
chmod o-t <directory>
```

Чтобы удалить разрешение липкого бита численно, нужно вычесть 1000 из существующих разрешений каталога:

```
chmod 0775 <directory>
```

Вывод команды `ls -l` отображает липкий бит на `t` в позиции выполнения «другие»: `drwxrwxrwt`.

Строчные буквы `t` означают, что и липкий бит, и разрешения выполнения установлены для «других». Заглавная буква `T` означает, что задано только разрешение с липким битом: `drwxrwxrwtT`.

Хорошими примерами использования каталогов с липкими битами будут каталоги `/tmp` и `/var/tmp`. Эти каталоги разработаны как места, где любой пользователь может создать временный файл.

Поскольку эти каталоги предназначены для записи всеми пользователями, они настроены на использование липкого бита. Без этого специального разрешения любой пользователь сможет удалять временные файлы другого пользователя.

Данные по дополнительным правам сведены в таблицу 25.

Таблица 25

Права	Чис- ловое значе- ние	Символь- ное зна- чение	Действие на файлы	Действие на каталоги
setuid	4	u+s	Файл выполняется с разрешениями владельца	Нет смысла
setgid	2	g+s	Файл выполняется с разрешениями группы владельцев	Создаваемые в директории файлы, получают общую группу владельцев
sticky bit	1	o+t	Нет смысла	Запрещает пользователям не владельцам удалять файлы

Контрольные вопросы

1. Кто может использовать команду `chmod` для изменения свойств файла?
2. Кто может использовать команду `chown` для изменения пользователя-владельца файла?
3. Что позволяет разрешение на выполнение для каталога?
4. Для чего используется команда `umask`?
5. Как сочетаются восьмеричная, бинарная и символьная формы записи прав на файл или каталог?
6. С помощью каких команд можно увидеть права доступа файла?

ЗАКЛЮЧЕНИЕ

В учебном пособии рассматривались базовые основы применения операционных систем Linux. Материал пособия может быть использован при изучении вопросов администрирования вычислительных систем на основе различных операционных систем Unix и Linux.

В настоящее время наметился значительный вектор перехода с использования операционных систем семейства Microsoft Windows на операционные системы Linux. Для большинства конечных пользователей переход на новую операционную систему означает лишь смену графической оболочки и замену некоторых программ на аналоги. Для профессионалов, использующих, настраивающих и поддерживающих операционные системы, существует гораздо больше вопросов. В пособии показаны первичные навыки, необходимые для профессионального использования.

Рассмотрены как общие вопросы выбора, установки и настройки системы, так и более детально вопросы использования различных команд консоли. Пособие не является всеобъемлющим, но дает возможность первично ознакомиться с обозначенной темой.

Издание является дополнением к материалам лекционного курса по дисциплине «Администрирование вычислительных систем» и выступает в роли информационной поддержки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Администрирование в информационных системах. Учебное пособие для вузов. - 2-е изд., испр. и доп. - М.: Горячая линия-Телеком, 2018. - 408 с., ил. - ISBN 978-5-9912-0418-7

2. Петцке, К. LINUX. От понимания к применению / Петцке К. – М. : ДМК Пресс. - 576 с. - ISBN 5-93700-004-8

3. Зубков, С. В. Linux. Русские версии / Зубков С. В. – М. : ДМК Пресс. - 347 с. - ISBN 5-94074-013-8

4. Курячий, Г. В. Операционная система Linux / Курячий Г. В. , Маслинский К. А. - Москва : Национальный Открытый Университет "ИНТУИТ", 2016. (Основы информационных технологий) - ISBN 5-9556-0029-9. - Текст : электронный // ЭБС "Консультант студента" : [Сайт]. - URL : <https://www.studentlibrary.ru/book/ISBN5955600299.html> (дата обращения: 21.09.2022). - Режим доступа : по подписке.

5. Гончарук, С. В. Администрирование ОС Linux / Гончарук С. В. - Москва : Национальный Открытый Университет "ИНТУИТ", 2016. - Текст : электронный // ЭБС "Консультант студента" : [Сайт]. - URL : <https://www.studentlibrary.ru/book/intuit023.html> (дата обращения: 21.09.2022). - Режим доступа : по подписке.

6. Костромин, В. А. Основы работы в ОС Linux / Костромин В. А. - Москва : Национальный Открытый Университет "ИНТУИТ", 2016. - Текст : электронный // ЭБС "Консультант студента" : [Сайт]. - URL : https://www.studentlibrary.ru/book/intuit_209.html (дата обращения: 21.09.2022). - Режим доступа : по подписке.

7. Бражук, А. И. Сетевые средства Linux / Бражук А. И. - Москва : Национальный Открытый Университет "ИНТУИТ", 2016. - Текст : электронный // ЭБС "Консультант студента" : [Сайт]. - URL : https://www.studentlibrary.ru/book/intuit_360.html (дата обращения: 21.09.2022). - Режим доступа : по подписке.

8. Гунько, А. В. Системное программирование в среде Linux : учебное пособие / А. В. Гунько. - Новосибирск : НГТУ, 2020. - 235 с. - ISBN 978-5-7782-4160-2. - Текст : электронный // ЭБС "Консультант студента" : [Сайт]. - URL :

<https://www.studentlibrary.ru/book/ISBN9785778241602.html> (дата обращения: 21.09.2022). - Режим доступа : по подписке.

9. Арпачи-Дюссо Р. Х., Арпачи-Дюссо А. К. Операционные системы: Три простых элемента / пер. с англ. А. А. Слинкина. – М.: ДМК Пресс, 2021. - ISBN 978-5-97060-932-3

10. Уорд Б. У. Внутреннее устройство Linux. — СПб.: Питер, 2016. — 384 с.: ил. — (Серия «Для профессионалов»). ISBN 978-5-496-01952-1

11. Кетов Д.В. Внутреннее устройство Linux.- 2-е изд., перераб и доп. – СПб.: БХВ-Петербург, 2021. – 400 с.: ил. — ISBN 978-5-9775-6630-8

Учебное электронное издание

КУЛИКОВ Константин Владимирович

ОСНОВЫ ПРИМЕНЕНИЯ ОПЕРАЦИОННЫХ СИСТЕМ LINUX

Учебное пособие

Издается в авторской редакции

Системные требования: Intel от 1,3 ГГц; Windows XP/7/8/10; Adobe Reader; дисковод CD-ROM.

Тираж 25 экз.

Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых
Изд-во ВлГУ
rio.vlsu@vlsu.ru

Институт информационных технологий и радиоэлектроники
kulikov@vlsu.ru