Linux в действии

Дэвид Клинтон

MANNING

Дэвид Клинтон

Linux
В ДЕЙСТВИИ

ПИТЕР Санкт-Петербург · Москва · Екатеринбург · Воронеж
Нижний Новгород · Ростов-на-Дону · Самара · Минск

2019

Linux in Action

DAVID CLINTON

MANNING
SHELTER ISLAND

ББК 32.973.2-018.2
УДК 004.451
К49

Клинтон Дэвид

К49 Linux в действии. — СПб.: Питер, 2019. — 416 с.: ил. — (Серия «Библиотека программиста»).

ISBN 978-5-4461-1199-2

Без практики ничему нельзя научиться, и Linux не исключение. Книга «Linux в действии» поможет приобрести навыки защиты файлов, папок и серверов, безопасной установки патчей и приложений, а также управления сетью.

В книге описываются 12 реальных проектов, в том числе автоматизация системы резервного копирования и восстановления, настройка личного файлового облака в стиле Dropbox и создание собственного сервера MediaWiki. На интересных примерах вы изучите виртуализацию, аварийное восстановление, обеспечение безопасности, резервное копирование, внедрение DevOps и устранение неполадок системы. Каждая глава заканчивается обзором практических рекомендаций, глоссарием новых терминов и упражнениями.

16+ (В соответствии с Федеральным законом от 29 декабря 2010 г. № 436-ФЗ.)

ББК 32.973.2-018.2
УДК 004.451

ISBN 978-1617294938 англ.
ISBN 978-5-4461-1199-2

Краткое содержание

Оглавление

Предисловие

Независимо от того, чем вы занимаетесь и как долго работаете в сфере информационных технологий или программирования, вы должны постоянно учиться новому. Дело не только в том, что платформы и парадигмы все время меняются. Новые бизнес-требования также вынуждают нестандартно мыслить. Хакеры постоянно придумывают новые способы атаковать ваши серверы. По этим и многим другим причинам нельзя переставать учиться.

Я очень хочу, чтобы вы прочитали хотя бы одну главу из этой книги и почувствовали себя достаточно уверенно, чтобы взяться за что-то более сложное — то, что раньше даже не рассматривали. Если вы прочитаете книгу до конца, то освоите важнейшие современные технологии, обеспечивающие виртуализацию, восстановление после сбоев, безопасность инфраструктуры, резервное копирование данных, безотказную работу веб-серверов, реализацию DevOps и устранение неполадок системы.

Но почему Linux? Потому что на базе Linux функционирует большая часть Интернета, научных организаций и коммерческих предприятий — фактически большинство серверов в мире. Эти серверы нужно готовить, запускать в работу, необходимо обеспечивать их безопасность. Ими должны эффективно управлять умные и хорошо обученные люди. Убежден, что с умом у вас все в порядке, и думаю, что смогу помочь с качественным обучением.

Не уверены, что достаточно знаете о Linux, чтобы начать такой амбициозный проект? Глава 1 быстро заполнит пробелы в знаниях. Прочитав ее, пристегните ремень безопасности и приготовьтесь к серьезному обучению.

Благодарности

Невозможно достичь конца длинного и иногда выматывающего пути по подготовке издания, не задумываясь о том, какие силы понадобились для этого. Что касается этой книги, как и других, написанных мной, успех предприятия был бы невозможен без таланта и самоотверженности каждого сотрудника издательства Manning.

Фрэнсис Лефковиц, редактор-консультант по аудитории, помогала мне более доходчиво донести материал, неуклонно удерживая меня на правильном пути. Река Хорвас и Джон Гасри кропотливо проверяли все примеры для книги и давали ценные советы. Фрэнсис Буран, литературный редактор, кажется, никогда не бывает полностью довольна текстом, по крайней мере моим. Но грамотность каждого предложения и изящество текста книги в его нынешнем виде явно указывают на качество ее работы.

В роли руководителя проекта Дейрдре Хиам эффективно провела нас до последнего шага, успешно синхронизируя все шестеренки механизма. Каждый из научных редакторов книги оставил важный след — все их ценные наблюдения были тщательно записаны, проанализированы и по возможности учтены. Поэтому большое спасибо Анджело Косто, Кристоферу Филлипсу, Дарио Виктору Дюрану, Флайолу Фредерику, Фостеру Хейнсу, Джорджу Л. Гейнсу, Густаво Патино, Хавьеру Колладо, Йенсу Кристиану Мэдсену, Джонасу Медина де лос Рейесу, Мацею Юрковски, Майеру Патилу, Мохсену Мостафе Джокару и Тиму Кейну.

Эта книга не только о навыках администрирования Linux. В ней я пытаюсь привить успешным администраторам большее чувство ответственности за серверы и системы, находящиеся под их опекой. Мне повезло, что в начале моей карьеры в качестве системного администратора Linux у меня был потрясающий наставник. Внимание Петра Федорова как к мелким операционным деталям, так и к глобальной картине делает его особенно эффективным администратором. Он вовлек меня в мир виртуализации Linux задолго до того, как контейнеры стали настолько популярны. Теперь по крайней мере часть наставлений Петра, без сомнения, отражена здесь.

И наконец, ни один из моих профессиональных (или частных) проектов не был бы реализован без поддержки и помощи моей дорогой жены. Мы вместе делаем всю тяжелую работу, но мои успехи — это в основном ее заслуга.

Об этой книге

Хотите научиться администрировать компьютеры под управлением операционной системы Linux? Прекрасный выбор. В то время как Linux еще пытается занять свое место в настольных системах, она абсолютно доминирует в мире серверов, особенно виртуальных и облачных. Поскольку самое серьезное администрирование серверов в наши дни выполняется удаленно, работа с графическим интерфейсом того или иного рода просто приводит к лишнему потреблению ресурсов. Если вы хотите управлять серверами и сетевыми архитектурами, которые в настоящее время привлекают всеобщее внимание, вам придется пользоваться оболочкой командной строки Linux.

Хорошей новостью является то, что основной набор команд Linux применим практически везде, где пересекаются компьютеры и бизнес. Лучшая новость заключается в том, что навыки работы в Linux всегда будут востребованы. Поскольку это зрелая и стабильная операционная система, большинство инструментов, использовавшихся четверть века назад, по-прежнему эффективны, а большинство инструментов, применяемых сегодня, вероятно, и через четверть века все еще будут востребованы. Другими словами, изучение Linux — это инвестиция на всю жизнь.

Да, вы наверняка заняты и у вас дедлайны. Я не могу обещать вам, что освоить Linux будет так же просто, как научиться завязывать шнурки. Но я могу помочь вам сфокусироваться на главном.

Как я собираюсь сделать это? Я не буду ориентироваться на какие-либо методики обучения. В то время как другие книги, курсы и онлайн-ресурсы организуют свой контент по темам («Хорошо, мальчики и девочки, все достают свои тетрадки и ручки. Сегодня мы поговорим о файловых системах Linux»), я собираюсь использовать для обучения реальные проекты.

Так, например, я мог бы написать целую главу (или две) о файловых системах Linux. Но вместо этого вы узнаете, как создавать корпоративные файловые серверы, диски восстановления системы и сценарии для репликации архивов критических данных. В процессе в качестве бесплатного бонуса вы получите знания о файловой системе.

Не думайте, что я собираюсь охватить все инструменты администрирования Linux. Это невозможно — их реально тысячи. Но не волнуйтесь. Основные навыки

и функциональные возможности, необходимые в течение первых лет карьеры в администрировании Linux, будут подробно описаны, но только если они требуются для реального, критически важного проекта. Когда вы закончите чтение, вы получите не меньше информации, чем из любого традиционного источника, но при этом еще узнаете, как реализовать более дюжины крупных административных проектов и будете готовы внедрить десятки других.

Вы в деле? Я так и думал.

Кому следует прочитать эту книгу

Книга призвана помочь вам приобрести множество навыков администрирования Linux. Возможно, вы разработчик, который хочет более тесно работать с серверной средой, где будут «жить» ваши приложения. Или, может быть, вы хотите попасть в мир администрирования серверов или DevOps. В любом случае эта книга для вас.

Что желательно знать? По крайней мере, вы должны без проблем работать с файлами, сетями и базовыми ресурсами современной операционной системы. Опыт работы с системным администрированием, сетевым управлением и языками программирования определенно не помешает, но не обязателен. Прежде всего, вы не должны бояться изучать новые среды и с энтузиазмом экспериментировать с новыми инструментами. Еще кое-что: вы должны знать, как выполнить стандартную установку операционной системы Linux.

Структура издания: дорожная карта

Несколько слов о том, как структурирована книга. Каждая глава охватывает один или два практических проекта (за исключением главы 1). Поскольку глава 1 предназначена для заполнения пробелов в ваших базовых знаниях о Linux, она будет отличаться от остальных. Вам не нужны основы? Я абсолютно уверен, что вы почерпнете много интересных новых идей, чтобы использовать их в главе 2.

По мере рассмотрения проектов я расскажу о навыках и инструментах, которые вам понадобятся. Кроме того, для работы над проектом каждой главы вам потребуются знания, полученные ранее в этой книге. Чтобы продемонстрировать вам, что я имею в виду, приведу полный список глав и основных проектов, областей знаний и инструментов, с которыми вы познакомитесь на протяжении всей книги.

Глава	Область знаний	Инструменты
1. Добро пожаловать в Linux	Оболочки, разделы и файловые системы	Bash, man
2. Виртуализация Linux: создание безопасной и простой рабочей среды	Виртуализация, файловые системы	VirtualBox, LXC, apt, yum/dnf
3. Удаленное подключение: безопасный доступ к машинам по сети	Безопасность, удаленные подключения	ssh, scp, systemctl, ps, grep

Глава	Область знаний	Инструменты
4. Управление архивами: создание резервных копий или копирование целых файловых систем	Разделы и файловые системы, текстовые потоки	tar, dd, redirects, rsync, locate, split, chmod, chown
5. Автоматизированное администрирование: настройка автоматического резервного копирования	Сценарии, управление системными процессами, безопасность	Сценарии, cron, anacron, таймеры systemd
6. Инструменты для критических ситуаций: создание устройства для восстановления системы	Разделы и файловые системы, управление устройствами	parted, GRUB, mount, chroot
7. Веб-серверы: создание сервера MediaWiki	Базы данных, сетевое взаимодействие, управление пакетами	PHP, MySQL (MariaDB), веб-сервер Apache, зависимости пакетов
8. Совместное использование файлов в сети: создание сервера для совместного использования файлов Nextcloud	Управление пакетами, сетевое взаимодействие, безопасность	snapd, файловые системы, шифрование
9. Защита вашего веб-сервера	Сетевое взаимодействие, безопасность, мониторинг системы	Apache, Iptables, /etc/group, SELinux, apt, yum/dnf, chmod, chown, Let's Encrypt
10. Защита сетевых соединений: создание VPN или DMZ	Сетевое взаимодействие, безопасность	Брандмауэры, ssh, Apache, OpenVPN, sysctl, easy rsa
11. Мониторинг системы: работа с файлами журналов	Мониторинг системы, текстовые потоки, безопасность	grep, sed, journal ctl, rsyslogd, /var/log/, Tripwire
12. Совместное использование данных в частной сети	Сетевое взаимодействие, разделы, файловые системы	nfs, smb, ln, /etc/fstab
13. Устранение проблем производительности системы	Мониторинг системы, управление системными процессами, сетевое взаимодействие	top, free, nice, nmon, tc, iftop, df, kill, killall, uptime
14. Устранение неполадок в сети	Сетевое взаимодействие	ip, dhclient, dmesg, ping, nmap, traceroute, netstat, netcat(nc)
15. Устранение неполадок с периферийными устройствами	Управление устройствами	lshw, lspci, lsusb, modprobe, CUPS
16. Инструменты DevOps: развертывание серверной среды с использованием Ansible	Сценарии, виртуализация	Ansible, YAML, apt

О коде

В этой книге содержится много примеров исходного кода: в виде многочисленных листингов и прямо в тексте. В обоих случаях исходный код выделен **таким моноширинным шрифтом**, что позволяет отличать его от обычного текста. Иногда код также выделен жирным шрифтом, чтобы подчеркнуть обсуждаемую тему.

В некоторых местах первоначальный исходный код переформатирован; мы добавили разрывы строк и изменили отступы так, чтобы наилучшим образом разместить код на странице. Но иногда даже этого было недостаточно, и тогда мы вставили в листинги маркеры переноса строки (➥). Кроме того, комментарии в исходном коде часто удалены из листингов, так как описание кода содержится в тексте главы. Многие листинги снабжены аннотациями к коду, в которых описываются важные понятия.

Дистрибутивы Linux

В настоящее время существуют десятки активно поддерживаемых дистрибутивов Linux. Хотя большинство основных версий Linux являются общими для всех дистрибутивов, всегда будут мелочи, которые будут работать «здесь», но не «там». Из соображений практичности я сосредоточусь в основном на двух дистрибутивах: Ubuntu и CentOS. Почему именно эти два? Потому что каждый представляет целое семейство дистрибутивов. Ubuntu является основой для Debian, Mint, Kali Linux и др., а CentOS совместим с Red Hat Enterprise Linux (RHEL) и Fedora.

Это не значит, что я сбрасываю со счетов другие дистрибутивы, такие как Arch Linux, SUSE и Gentoo. И это также не значит, что изученное в данной книге не поможет вам работать с указанными системами. Но полностью охватить семейства Ubuntu и CentOS означает получить самое полное представление о Linux, которое только возможно при использовании двух дистрибутивов.

Форум книги

Покупка книги «Linux в действии» включает бесплатный доступ к частному веб-форуму, организованному издательством Manning Publications, где можно оставлять комментарии о книге, задавать технические вопросы, а также получать помощь от автора и других пользователей. Чтобы получить доступ к форуму, перейдите на https://forums.manning.com/forums/linux-in-action. Узнать больше о других форумах на сайте издательства Manning и познакомиться с правилами вы сможете на странице https://forums.manning.com/forums/about.

Издательство Manning обязуется предоставить своим читателям место встречи, на которой может состояться содержательный диалог между отдельными читателями и между читателями и автором. Однако со стороны автора отсутствуют какие-либо обязательства уделять форуму внимание в каком-то определенном

объеме — его присутствие на форуме остается добровольным (и неоплачиваемым). Мы предлагаем задавать автору неожиданные вопросы, чтобы его интерес не угасал! Форум и архивы предыдущих обсуждений будут доступны на сайте издательства, пока книга находится в печати.

Другие интернет-ресурсы

Зашли в тупик? Поиск в Интернете — ваш лучший друг. В Сети вы найдете множество существующих руководств по Linux и с вами поделятся опытом устранения неполадок. Но вы не должны забывать о сайтах типа StackExchange и, в частности, о ресурсе serverfault.com. Если что-то не ладится с какой-либо конфигурацией системы или не работает сетевое подключение, то высока вероятность того, что кто-то другой уже столкнулся с той же проблемой, спросил об этом на сайте ServerFault и получил ответ. Ответа нет? Тогда сами задайте свой вопрос. Ресурсы LinuxQuestions.org и ubuntuforums.org также могут быть полезны.

А те, кто любит видеообучение, найдут хороший выбор курсов по Linux на сайте Pluralsight.com, в том числе более десятка моих собственных курсов.

От издательства

Ваши замечания, предложения, вопросы отправляйте по адресу comp@piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

На сайте издательства www.piter.com вы найдете подробную информацию о наших книгах.

Об авторе

Дэвид Клинтон — системный администратор, преподаватель и писатель. Он администрировал, писал об этом и создавал учебные материалы для многих важных технических дисциплин, включая системы Linux, облачные вычисления (в частности AWS) и контейнерные технологии, такие как Docker. Он написал книгу *Learn Amazon Web Services in a Month of Lunches* (Manning, 2017). Многие из его учебных видеокурсов можно найти на сайте Pluralsight.com, а ссылки на другие его книги (по администрированию Linux и виртуализации серверов) доступны по адресу bootstrap-it.com.

Об иллюстрации на обложке

Рисунок на обложке «Linux в действии» подписан как «Внешний вид уроженца Армении в Персии в 1700 году». Эта иллюстрация взята из четырехтомника Томаса Джефферис *A Collection of the Dresses of Different Nations, Ancient and Modern* («Коллекция платья разных народов, старинного и современного»), изданного в Лондоне в 1757–1772 годах. На титульном листе утверждается, что это гравюра ручной раскраски с применением гуммиарабика. Томаса Джефферис (1719–1771) называли географом при короле Георге III. Это английский картограф и ведущий поставщик карт своего времени. Он чертил и печатал карты для правительства и других государственных органов. Занимаясь картографией в разных уголках мира, он проявлял интерес к местным традиционным нарядам, которые впоследствии были блестяще переданы в данной коллекции.

В конце XVIII века заморские путешествия для собственного удовольствия были относительно новым явлением, поэтому подобные коллекции пользовались популярностью. Разнообразие рисунков в томах Джефферис ярко говорит об уникальности и индивидуальности одеяния народов мира примерно 200 лет назад. С тех пор стиль одежды сильно изменился и исчезло разнообразие, свойственное различным областям и странам. Теперь трудно различить по одежде даже жителей разных континентов. Если взглянуть на это с оптимистичной точки зрения, мы пожертвовали культурным и внешним разнообразием в угоду более насыщенной личной жизни или более разнообразной и интересной интеллектуальной и технической деятельности.

В наше время, когда трудно отличить одну компьютерную книгу от другой, издательство Manning проявляет инициативу и деловую сметку, украшая обложки книг изображениями, которые показывают богатое разнообразие жизни в регионах два века назад.

1 Добро пожаловать в Linux

В этой главе

- Отличия Linux от других систем.
- Основные сведения.
- Получение справки.

В этой книге предусмотрена не совсем стандартная методика обучения. Несмотря на то что другие книги, курсы и онлайн-ресурсы обучают в первую очередь *навыкам*, я собираюсь использовать в качестве образовательных инструментов реальные *проекты*. Каждый из основных навыков работы с Linux и каждая функциональная возможность системы будут тщательнейшим образом рассмотрены, но только тогда, когда это потребуется для проекта. Когда вы закончите чтение, вы овладеете необходимой теорией, получите опыт выполнения более десятка жизненно важных и сложных административных задач и сможете спокойно решить множество различных проблем.

Первые три главы познакомят вас с миром серверов Linux. После этого вы проработаете и адаптируете несколько реальных прикладных проектов, не отвлекаясь на изучение лишней информации. Благодаря этому вы не только изучите список команд и получите новые навыки. Будьте готовы к более глубокому погружению в задачу и в конечном счете к поиску решений для ваших собственных бизнес-проектов.

Ни один автор не сможет предусмотреть все проблемы, с которыми вы столкнетесь на протяжении своей карьеры. Но, продемонстрировав, как решать их с по-

мощью реальных примеров и инструментов, я научу вас эффективно использовать огромные ресурсы, доступные как через встроенную документацию, так и через Интернет. Если у вас совсем нет опыта работы с Linux, эта глава познакомит вас с некоторыми базовыми жизненно важными навыками работы с командной строкой и укажет, куда можно обратиться за помощью.

ПРИМЕЧАНИЕ

Далее вы увидите, что *командная строка* — это интерфейс, предоставляемый операционной системой (ОС). Он позволяет вводить текстовые команды для управления ОС или запрашивать данные, которыми она управляет.

Должен отметить, что в этой и каждой последующей главе я настоятельно буду рекомендовать попробовать все самостоятельно. Нет лучшего способа по-настоящему понять суть IT-навыка, чем все сделать самому, разобраться, почему что-то работает не так, как вы ожидали, и экспериментировать до тех пор, пока не освоите прием досконально. Удачи и приятного времяпрепровождения!

1.1. Что отличает Linux от других операционных систем

Linux — бесплатная ОС, а это значит, что ее намного проще установить, чем другие ОС, для любых целей, которые только можно себе представить. Отсутствие необходимости приобретать лицензию и соглашаться с цифровыми правами существенно упрощает тестирование всех видов аппаратных комбинаций и конфигураций серверов.

Linux позволяет реализовывать различные действительно полезные и креативные идеи. Например, вы можете развернуть *живой образ* Linux на USB-накопитель, запустить с него ПК, внутренний жесткий диск которого поврежден, а затем устранить неполадки и решить основную проблему. (Как это сделать, вы узнаете в главе 6.) Поскольку Linux — настоящая многопользовательская ОС, целые команды пользователей могут одновременно авторизовываться в системе для локальной или удаленной работы, будучи уверенными в конфиденциальности и стабильности системы.

Linux создавалась с использованием той же технологии и поставляется с большинством тех же инструментов, что и всем известная операционная система UNIX. Это говорит о серьезной стабильности и безопасности. Дистрибутивы Linux представляют собой сложные системы управления пакетами программного обеспечения, которые надежно устанавливают и поддерживают любые из тысяч бесплатных приложений, доступных в онлайн-репозиториях.

Но помимо того, что Linux бесплатна, у нее *открытый исходный код*, а это означает, что любой пользователь может взять кодовую базу и как угодно модифицировать ее.

Фактически это породило обширную экосистему специализированных дистрибутивов Linux. *Дистрибутив* (иногда сокращается до *дистр*) — это специализированный набор программного обеспечения, который поставляется вместе с ядром Linux и распространяется с инструментами для установки рабочей версии Linux на компьютерах пользователей. В табл. 1.1 приведен неполный список дистрибутивов, чтобы проиллюстрировать доступные разновидности.

Таблица 1.1. Некоторые дистрибутивы Linux

Предназначение	Дистрибутив
Безопасность и проверка на проникновение	Kali Linux
	Parrot
Частные пользователи	Mint
	Elementary OS
Легковесные (старое оборудование, диагностика)	Puppy Linux
	LXLE
Администрирование устройств Интернета вещей	Snappy Ubuntu Core
Корпоративная серверная система	CentOS (поддерживаемая сообществом открытая версия RHEL)
	OpenSUSE (поддерживаемая сообществом версия SUSE)
Облачные вычисления	Amazon Linux (AWS AMI)
	Ubuntu Server (AWS AMI)
Универсальное (исключая легковесные)	Ubuntu

Не можете найти дистрибутив, который вам идеально подходит? Создайте свой. Нужна помощь? В Интернете есть большое и активное сообщество: если кто-то еще не решил вашу проблему, то он обязательно будет знать, с чего начать, чтобы ее решить. Я бы сказал, что именно интернет-ресурсы сообщества линуксоидов делают Linux настолько мощным.

1.2. Основные сведения

Прежде чем приступить к реализации коммерческих проектов, которые составляют основную часть книги, нужно убедиться, что мы имеем единую отправную точку. В этой главе рассматриваются основы Linux: стандарт иерархии файловой системы UNIX (включая псевдофайловые системы), навигация (`ls`, `pwd` и `cd`), инструменты управления файлами (`cat`, `less`, `touch`, `mkdir`, `rmdir`, `rm`, `cp` и `mv`), некоторые хитрости (например, автозавершение ввода и универсализация файлов), `sudo` и места, куда обращаться за помощью (`man`, `info` и `journalctl`).

Возможно, у вас уже достаточно опыта и вам не понадобится указанный материал. Тогда можете пропустить эту главу.

Установка Linux

Я не собираюсь тратить время на рассказ о том, как установить Linux на свой ПК. И это не потому, что установка смехотворно проста: иногда она может быть довольно сложной. А скорее потому, что выбранный вами подход зависит от ваших конкретных обстоятельств. Описание одной или даже десятка возможностей ничего не даст, кроме того, что будет раздражать 75 % тех, для кого эти сценарии не сработают.

Нужна помощь с установкой? Прочтите книгу наподобие *Learn Linux in a Month of Lunches* (Manning, 2016). Возникла конкретная проблема при установке? Потратьте минуту, чтобы набросать краткое, но подробное описание, а затем используйте его для поиска помощи в Интернете. Ищете ноутбук или персональный компьютер с предустановленной ОС Linux? Поищите в Интернете «компьютер с предустановленной Linux». У вас есть неиспользуемый компьютер и USB-накопитель? Ищите «установить Linux с флешки». Предпочитаете установить Linux на виртуальную машину? Мудрый шаг. Не пропустите главу 2.

1.2.1. Файловая система Linux

Нередко говорят, что все в Linux работает на простых текстовых файлах, поэтому, вероятно, имеет смысл начать с разбора файловой системы Linux. Но прежде, чем мы перейдем к Linux, разберемся, что такое *файловая система*. Вы можете представить ее себе как таблицу данных (или *индекс*), которая создает очевидные связи между отдельными файлами и группами файлов и идентифицирует их местоположения на диске. Рисунок 1.1 поможет вам представить, как данные, распределенные по разделам диска, могут быть представлены пользователям системы в виде структуры каталогов.

Зачем нам нужен индекс? Цифровое запоминающее устройство, такое как жесткий диск или USB-устройство, физически не делится на части, которые можно использовать в качестве упорядочивающих *папок* (или *каталогов*, как их называют в Linux). Один конкретный файл может располагаться на реальном носителе в месте, которое находится на большом расстоянии от другого, почти идентичного файла, созданного с интервалом несколько минут или секунд, при этом все части одного файла могут быть несмежными. Мало того, географическое местоположение файла на диске не обязательно будет оставаться неизменным на протяжении всего времени.

Если вы хотите, чтобы ваши данные хорошо извлекались, вам понадобится какой-то индекс, который может последовательно указывать вам на искомые ресурсы. Файловая система использует такой индекс, чтобы обеспечить организованный набор каталогов и файлов в пределах одного сектора диска, известного как *раздел*.

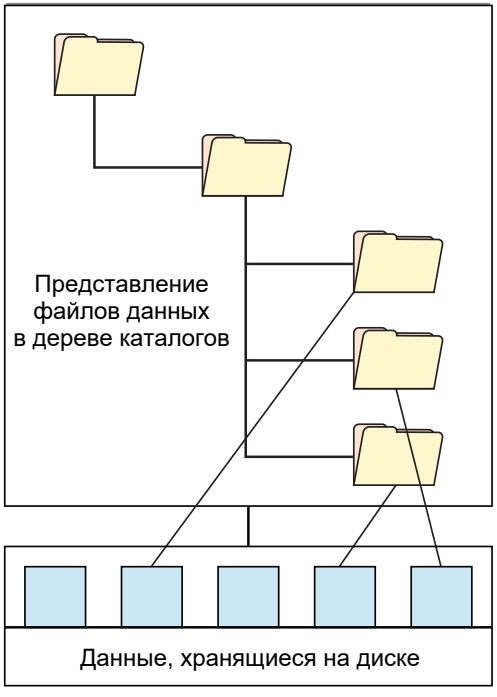


Рис. 1.1. Необработанные данные на устройствах хранения могут быть визуально представлены в ОС в виде организованных иерархий каталогов

ПРИМЕЧАНИЕ

Обратите внимание, что на сегодняшний день наиболее часто используемой файловой системой Linux является ext4. Но Linux также может работать с накопителями, отформатированными с применением других файловых систем, таких как FAT32 и NTFS.

Все файлы в разделе диска хранятся в каталогах ниже *корневого каталога*, который представлен символом / (прямой слеш). Способ расположения этих каталогов в значительной степени определяется стандартом иерархии файловой системы UNIX (FHS). Независимо от того, используете вы дистрибутив Linux, UNIX или даже macOS, вы увидите почти одинаковую базовую разметку. На рис. 1.2 показаны некоторые из наиболее часто используемых *каталогов верхнего уровня.*

Каталоги верхнего уровня — это папки, расположенные непосредственно в корневом каталоге, включая `/etc/`, которая содержит файлы конфигурации, определяющие способ функционирования отдельных программ и сервисов, и `/var/`, которая содержит *изменчивые* файлы, относящиеся к системе или отдельным приложениям, чье содержимое часто меняется в ходе обычной работы системы. Обратите также

внимание на каталог /home/, в котором разным пользователям по отдельности предоставляются папки для их личных файлов.

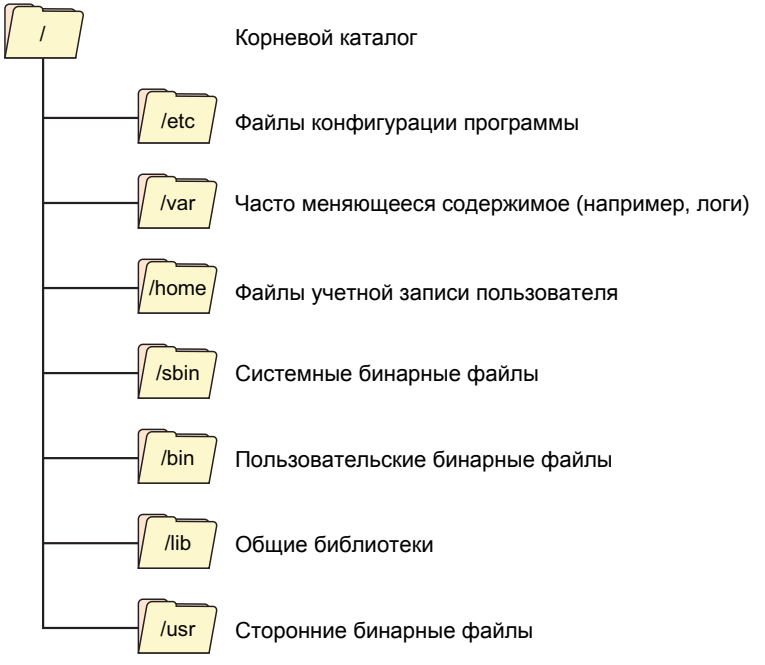


Рис. 1.2. Общие каталоги верхнего уровня, определенные в UNIX FHS

1.2.2. Начало работы: инструменты навигации в Linux

Здесь вы изучите пять основных и обязательных команд навигации в Linux (`ls`, `pwd`, `cd`, `cat` и `less`). Оболочка командной строки не является визуальной средой, поэтому, независимо от того, что вы попытаетесь сделать, для навигации вы будете использовать эти пять инструментов.

ПРИМЕЧАНИЕ

Я надеюсь, вам очевидно, что вы должны попробовать на своем компьютере каждый из этих инструментов. Это единственный способ научиться.

Далее в книге для выполнения примеров мы будем задействовать оболочку командной строки того или иного рода. К сожалению, нет единого способа открыть окно оболочки командной строки, которое будет работать во всех дистрибутивах Linux. Например, местоположение оболочки командной строки в системе меню Ubuntu не обязательно будет совпадать с расположением в Fedora или Mint. А в самом Ubuntu это еще зависит от того, какую версию вы используете.

Сочетание клавиш Ctrl+Alt+T должно работать по крайней мере в большинстве систем, так же как и при просмотре меню приложения, в котором выполняется поиск элемента с именем `terminal`. По умолчанию, как только ваша оболочка командной строки откроется, вы очутитесь в домашнем каталоге (`/home/`*`ваше_имя`*`/`).

ls (вывод содержимого каталога)

Нет смысла работать в оболочке командной строки, если вы не видите, что там находится. Вы можете вывести список имен файлов и подкаталогов в текущем каталоге, используя команду `ls`. Она выводит не только список имен объектов, но и их права доступа к файлам, имя владельца, группу, размер файла и метку времени. При добавлении имени каталога, например `/var/`, отображается содержимое этого каталога:

```
$ ls -l /var
total 40
drwxr-xr-x  2 root root   4096 May  3 06:25 backups
drwxr-xr-x 11 root root   4096 Jan 17 21:16 cache
drwxr-xr-x 39 root root   4096 Jan 17 21:16 lib
drwxrwsr-x  2 root staff  4096 Apr 12  2016 local
lrwxrwxrwx  1 root root      9 Aug 12  2016 lock -> /run/lock
drwxrwxr-x  7 root syslog 4096 May  3 06:25 log
drwxrwsr-x  2 root mail   4096 Aug 12  2016 mail
drwxr-xr-x  2 root root   4096 Aug 12  2016 opt
lrwxrwxrwx  1 root root      4 Aug 12  2016 run -> /run
drwxr-xr-x  5 root root   4096 Jan 17 21:16 spool
drwxrwxrwt  2 root root   4096 Nov  7  2016 tmp
drwxr-xr-x  3 root root   4096 Sep 11  2016 www
```

Если добавить к команде `ls` аргумент `h`, будут отображены размеры файлов в удобочитаемом формате, а не так, как предусмотрено по умолчанию (включает в себя множество трудноподсчитываемых цифр). То есть будут выводиться не байты, а килобайты, мегабайты и гигабайты:

```
$ ls -lh /var/log
total 18M
-rw-r--r-- 1 root   root    0 May  3 06:25 alternatives.log
drwxr-xr-x 2 root   root 4.0K May  3 06:25 apt
-rw-r----- 1 syslog adm  265K Jun  9 00:25 auth.log
-rw-r--r-- 1 root   root 312K Aug 12  2016 bootstrap.log
-rw------- 1 root   utmp    0 May  3 06:25 btmp
-rw-r----- 1 root   adm    31 Aug 12  2016 dmesg
-rw-r--r-- 1 root   root  836 May 21 14:15 dpkg.log
-rw-r--r-- 1 root   root  32K Nov  7  2016 faillog
drwxr-xr-x 2 root   root 4.0K Aug 12  2016 fsck
-rw-r----- 1 syslog adm  128K Jun  8 20:49 kern.log
-rw-rw-r-- 1 root   utmp 287K Jun  9 00:25 lastlog
-rw-r----- 1 syslog adm  1.7M Jun  9 00:17 syslog
-rw-rw-r-- 1 root   utmp 243K Jun  9 00:25 wtmp
```

Общее дисковое пространство (в мегабайтах), используемое файлами в этом каталоге

ПРИМЕЧАНИЕ

Как правило, вы добавляете аргументы в команды Linux одним из двух способов: вводите дефис, за которым следует одна буква (например, h, которая модифицирует вывод команды ls), или два дефиса, а затем указываете развернутую версию аргумента. В этом примере ls --human-readable генерирует точно такие же выходные данные, что и ls -h. Почти все команды Linux комплектуются полной документацией, которую мы рассмотрим позже в этой главе.

Хотите знать, что находится внутри текущего каталога? Добавление прописной буквы `R` в качестве аргумента команды `ls` приводит к отображению подкаталогов с файлами и вложенными подкаталогами, независимо от глубины вложения. Чтобы понять, как можно использовать этот аргумент, и увидеть результат работы, выполните команду `ls -R` для вывода содержимого папки `/etc/`:

```
$ ls -R /etc
```

pwd (текущий рабочий каталог)

Чаще всего ваше текущее местоположение в файловой системе будет отображаться слева от командной строки. В этом примере я нахожусь в сетевом каталоге, который находится чуть глубже `/etc/`:

```
ubuntu@base:/etc/network$
```

Поскольку вы, скорее всего, будете работать в системах, где эта подсказка не отображается, вам иногда может понадобиться возможность быстро уточнить свое местонахождение. Выполнив команду `pwd`, вы узнаете свой текущий рабочий каталог:

```
$ pwd
/etc/network
```

cd (смена каталога)

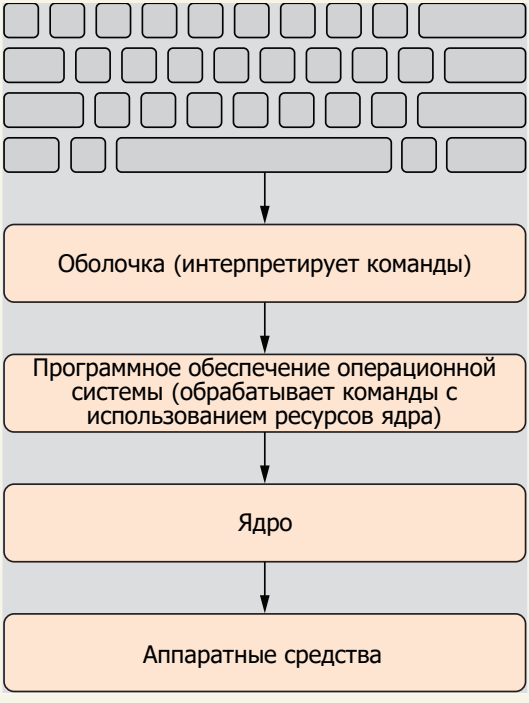
Получив представление о том, где вы находитесь и что доступно в текущем каталоге, вы должны узнать, как сменить свое местоположение. Если вы введете команду `cd`, оболочка командной строки (обычно это Bash) переместит вас в указанный каталог. Когда вы впервые открываете сеанс командной строки (часто называемый *оболочкой*), по умолчанию вы попадаете в домашний каталог своей учетной записи. Если вы выполните команду `pwd`, то, вероятно, увидите нечто подобное:

```
$ pwd
/home/ваше_имя
```

Что такое Bash

Bash, вероятно, самая популярная оболочка UNIX. Отлично! Но что такое оболочка? Оболочка командной строки — это любой пользовательский интерфейс, который интерпретирует команды пользователя либо через интерфейс командной строки (CLI),

либо через графический интерфейс пользователя (GUI). Вы можете представить оболочку (см. рисунок) как программный уровень, предназначенный для выполнения всех надлежащим образом отформатированных команд с использованием базового ядра и аппаратных системных ресурсов. Другими словами, это средство общения с компьютером.



Оболочка интерпретирует выполнение пользовательских команд ввода

Теперь вернемся в корневой каталог, введя `cd` и слеш:

```
cd /
```

Запустите команду `ls` еще раз, чтобы просмотреть содержимое каталога. (Вы увидите каталоги, показанные на рис. 1.2.) Обратите внимание на домашний каталог, из которого вы можете получить доступ к каталогу *ваше_имя*. Чтобы перейти в любой из перечисленных там подкаталогов, введите `cd`, а затем имя каталога, который хотите посетить. Поскольку путь, который вы здесь указываете, относится к вашему текущему местоположению, не нужно вводить имя каталога перед слешем. Команда `cd..` поднимет вас на один уровень в иерархии каталогов, например из `/home/`*ваше_имя*`/` в `/home/`.

Однако, если у вас есть более амбициозные планы перемещения и вы хотите перейти в пункт назначения далеко за пределами вашего текущего каталога, при-

дется использовать абсолютный путь. Это означает, что вы укажете путь, который начинается с корневого каталога (представленного слешем). Чтобы вернуться в свой домашний каталог из другого места в системе, вы вводите слеш, затем слово `home` (которое, как вы помните, относится к папке в корневом каталоге), а потом ваше имя пользователя. Попробуйте:

```
$ cd/home/ваше_имя
```

Тем не менее ввод команды `cd` без каких-либо аргументов вернет вас в домашний каталог текущего вошедшего в систему пользователя.

cat (вывод содержимого файла целиком)

Доступ к содержимому текстовых файлов в оболочке командной строки немного сложнее. Команда `cat` выводит содержимое файла на экран, где его можно прочитать, но не отредактировать. Это приемлемо для небольших документов, таких как файл `fstab` в папке `/etc/`. В следующем примере используется абсолютный путь, чтобы файл можно было открыть независимо от того, где в файловой системе вы находитесь в данный момент.

```
$ cat /etc/fstab
```

ПРИМЕЧАНИЕ

> Название cat на самом деле является сокращением слова concatenate — «конкатенация», что обозначает операцию объединения нескольких строк или файлов в один текстовый поток.

Предположим, что файл, который вы хотите прочесть, содержит больше строк, чем помещается на экране. Попробуйте просмотреть файл `/etc/group`:

```
cat /etc/group
```

Окажется, что первые строки прокручиваются вверх и исчезают с экрана слишком быстро и вы не успеваете их прочитать. Что хорошего в обычном текстовом файле, если вы не можете его прочесть? Конечно, как вы вскоре увидите, в Linux есть множество текстовых редакторов для активного управления контентом, но было бы неплохо иметь возможность просматривать длинный файл по страницам.

less (вывод содержимого файла по страницам)

Познакомьтесь с `less`! Эта команда имеет такое название, предположительно, потому, что может отображать меньше (по англ. less), чем полное содержимое файла (или, вероятно, чтобы можно было отличить ее от уже устаревшей команды `more`). Вы запускаете `less`, указав рядом имя файла:

```
less /etc/services
```

Используя `less`, вы можете прокручивать содержимое файла вверх и вниз с помощью клавиш со стрелками Page Up, Page Down и Пробел. Когда закончите, нажмите клавишу Q, чтобы закрыть файл.

1.2.3. Начало работы: инструменты управления файлами Linux

Поскольку у вас есть файлы и каталоги, вам нужно знать, как создавать, уничтожать, перемещать и копировать их. Файлы часто создаются автоматически каким-либо внешним процессом, например, при установке программного обеспечения, или автоматическом журналировании, или, скажем, в результате сохранения документа в офисном редакторе, таком как LibreOffice. Нет необходимости обсуждать все это здесь. Однако я отмечу, что вы можете быстро создать пустой файл с помощью команды `touch`. Рядом с ней нужно указать имя, которое вы хотели бы присвоить файлу:

```
$ touch myfile
```

После этого файл, расположенный в текущем каталоге, можно посмотреть с помощью команды `ls`. Попытка вывести его содержимое на экран с помощью команды `cat`, конечно, вообще ни к чему не приведет, потому что вы только что создали файл и он пуст.

```
$ ls
myfile
$ cat myfile
```

«Прикосновение» (так переводится название этой команды) к существующему файлу с помощью команды `touch` обновляет его отметку времени без внесения каких-либо изменений в сам файл. Это может быть полезно, если по какой-то причине вы хотите изменить метаданные, которые выводят различные команды наподобие `ls`. Это также может быть полезно, если вы хотите, чтобы ваш начальник подумал, что вы усердно работали над файлом данных, который фактически не открывался неделями.

Конечно, в этом быстро меняющемся мире вы многого не добьетесь, просто создав каталоги, заполненные пустыми файлами. В конце концов, вам нужно будет заполнить их контентом, а также отредактировать уже имеющуюся информацию. Для этого предлагаю вам познакомиться с надежным текстовым редактором.

Да, я знаю, что многие люди испытывают сильные чувства к тем текстовым редакторам, в которых они привыкли работать. Вы когда-нибудь вежливо намекали пользователю программы Vim, что его почтенный редактор может быть не таким полезным и важным, как раньше? Конечно, нет. Вы бы не смогли физически прочитать эту книгу, если бы сделали что-то подобное.

Я определенно не буду говорить вам, какой текстовый редактор необходимо использовать. Однако я скажу вам следующее: *никогда* не используйте для работы по администрированию Linux полнофункциональные текстовые процессоры, такие как LibreOffice и MS Word. Эти приложения добавят в ваши документы все виды скрытого форматирования и метаданных, что приведет к повреждению файлов системного уровня. В общих чертах есть три категории редакторов, которые стоит использовать.

- Если вы предпочитаете работать с документами в графическом интерфейсе, тогда вам пригодится *простой текстовый редактор*, такой как gedit. В нем доступны различные инструменты подсветки синтаксиса, которые делают верстку кода более продуктивной. Кроме того, вы можете быть уверены, что эта программа не сохранит ничего лишнего: только тот текст, который вы видите.
- В тех случаях, когда вам нужно отредактировать файл в оболочке командной строки, вам поможет *консольный редактор*, такой как nano (или Pico) с его интуитивно понятным интерфейсом.
- И наконец, есть *Vim* (или его первоначальная итерация vi). Ах, Vim! Если вы готовы потратить несколько месяцев своей жизни на изучение неинтуитивного интерфейса, то будете вознаграждены повышением производительности. Это так просто.

ПРИМЕЧАНИЕ

Все свои книги, статьи и документы я писал в gedit. Почему? Мне так нравится.

Почему бы нам не потратить пару минут прямо сейчас и не внести кое-какие изменения в только что созданный документ `myfile`, используя каждый из трех упомянутых текстовых редакторов? Например:

```
$ nano myfile
$ vi myfile
```

Для Vim войдите в режим вставки, нажмите клавишу `I`, а затем наберите текст. Сохранить свою работу можно так: нажмите клавишу `Esc`, затем введите `w`, после чего выйдите, введя `q`.

Создание и удаление каталогов

Каждый объект в файловой системе Linux представлен уникальной коллекцией метаданных, называемой *inode*. Я полагаю, вы скажете, что это индексный дескриптор файловой системы, о котором говорилось ранее, и он построен на основе метаданных, связанных со всеми индексами на диске. Чтобы вывести на экран больше информации о файле, который вы только что создали с помощью команды `touch`, включая информацию об inode, можете использовать команду `stat`:

```
$ stat myfile
  File: 'myfile'
  Size: 0               Blocks: 0          IO Block: 4096 regular empty file
Device: 802h/2050d      Inode: 55185258    Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/  ubuntu)
                           Gid: ( 1000/  ubuntu)
Access: 2017-06-09 13:21:00.191819194 +0000
Modify: 2017-06-09 13:21:00.191819194 +0000
Change: 2017-06-09 13:21:00.191819194 +0000
 Birth: -
```

Идентификатор файла inode

Права доступа и статус владельца файла

Как видите, результат включает в себя такие данные, как имя файла, атрибуты и метки времени. Но команда также сообщает вам свой идентификационный номер. Важно знать, что при перемещении, копировании или удалении файла/каталога вы на самом деле только изменяете его атрибуты *inode*, а не идентификатор. Кстати, inode — это объект, используемый системами UNIX для определения местоположения диска и атрибутов файлов в файловой системе (как показано на рис. 1.2). Обычно для каждого файла или каталога существует только один inode.

Предположим, сейчас вы находитесь в своем домашнем каталоге. Почему бы не создать в нем новый каталог, который вы сможете использовать для своих экспериментов? Для этого воспользуйтесь командой `mkdir`.

```
$ mkdir myplace
```

Теперь перейдите в новый каталог и создайте в нем файл:

```
$ cd myplace
$ touch newfile
$ ls
newfile
```

Чтобы увидеть, как выполняется удаление объектов, вернитесь обратно в родительский каталог (используя команду `cd ..`) и удалите только что созданный каталог. Как ни странно, предопределенная команда для удаления каталогов `rmdir` в этом конкретном случае не будет работать. Попробуйте сами:

```
$ cd ..
$ rmdir myplace
rmdir: failed to remove 'myplace': Directory not empty
```

Каталог не пустой? Что это? Это встроенная проверка, предотвращающая случайное удаление каталогов, заполненных важными файлами и подкаталогами, о которых вы, возможно, забыли. Есть несколько вариантов обойти проблему.

В первом случае потребуется добавить аргумент `--ignore-fail-on-non-empty` в команду `rmdir`, но придется напечатать слишком много букв. Второй подход заключается в том, чтобы вручную поработать с каждым подкаталогом и удалить каждый найденный объект. Иногда этот способ бывает намного хуже первого. В тех случаях, когда вы на 100 % уверены, что в каталоге нет абсолютно ничего, самый быстрый способ — добавить аргумент `-r` (что означает «рекурсивный») в команду `rm`:

```
$ rm -r myplace
```

Вероятно, сейчас самое время рассказать вам об одном очень важном различии между графическим интерфейсом настольных приложений и оболочкой командной строки: в командной строке нет корзины. Если вы удалите что-либо с помощью команды `rm` (или `rmdir`), а затем передумаете, то, по большому счету, у вас не будет возможности вернуть все обратно. Но вспомните о дисковом пространстве, которое вы теперь освободили!

Копирование и перемещение файлов

Следующим шагом создайте несколько файлов и новый каталог:

```
$ touch file1 file2 file3
$ mkdir newdir
```

Вы можете сделать идентичную копию объекта, используя команду `cp`. В этом примере создается копия файла `file1` в каталоге `newdir`:

```
$ cp file1 newdir
```

Кстати, команда `cp` достаточно «умна», чтобы распознать `newdir` как каталог, а не как файл. Если бы в текущем местоположении не было каталога с именем `newdir`, команда `cp` вместо этого сделала бы новую копию файла `file1` с именем `newdir`. Если вы случайно ошибетесь при вводе команды, то в результате получите новый файл, а не искомый каталог. В любом случае проверьте введенную команду, чтобы убедиться, что все работает так, как предполагалось.

В отличие от `cp`, команда `mv` не копирует, а перемещает объект из одного места в другое. Следовательно, если вы переместите файл из вашего домашнего каталога в подкаталог `newdir`, то в исходном месте файл уже будет недоступен:

```
$ mv file2 newdir
```

Опять же проверьте результат самостоятельно. Вы можете копировать, перемещать или удалять каталоги, используя те же команды, что и для файлов, при необходимости добавляя аргумент `-r`. Помните, что вы перемещаете не только каталог, имя которого видите: все вложенные папки и файлы также будут перемещены.

Подстановка имен файлов

Теперь рассмотрим применение подстановочных знаков к именам файлов, которыми вы управляете с помощью команд в оболочке командной строки.

```
$ mv* /some/other/directory
```

Чтобы переместить только файлы с именами, частично совпадающими с указанной последовательностью, попробуйте выполнить следующую команду:

```
$ mv file* /some/other/directory
```

Символ * заменяет *любое* количество символов. Указанная команда перемещает все файлы, имена которых начинаются со слова `file`, а все остальные оставляет в исходном месте. Если у вас есть файлы с именем `file1`, `file2` и т. д. до `file15` и вы хотите переместить только те файлы, имена которых находятся в диапазоне `file1` — `file9`, используйте вместо * знак вопроса (`?`).

```
$ mv file?/some/other/directory
```

Знак вопроса заменяет только *один* символ и применяет операцию только к тем файлам, имена которых состоят из слова `file` и одной цифры. Файлы `file10` — `file15` останутся в исходном каталоге.

Удаление файлов

Как вы уже знаете, объекты можно удалить с помощью команды `rm`. Но имейте в виду, что эти операции практически необратимы. Если вы хотите удалить файл `file1` из каталога, выполните следующую команду:

```
$ rm file1
```

Подстановочные знаки в именах файлов можно использовать в структуре команды `rm` так же, как вместе с `cp` или `mv`, и с той же эффективностью. Так, например, команда:

```
$ rm file*
```

удаляет в текущем каталоге все файлы, имена которых начинаются со слова `file`. Добавление к команде удаления аргумента `-r` сделает действие рекурсивным и удалит содержимое всех подкаталогов по указанному пути:

```
$ rm -r*
```

На самом деле это очень опасное сочетание. Более того, когда вы работаете с полномочиями администратора, вам также предоставляется власть над всеми системными файлами. Хорошенько подумайте, прежде чем выполнять такие опасные команды, как `rm`.

1.2.4. Управление с клавиатуры

Я сомневаюсь, что есть люди, которые с большим удовольствием печатают на клавиатуре. И я подозреваю, что большинство из них были бы очень признательны, если бы им сказали, что они могут делать, скажем, на 40 % меньше нажатий клавиш. Далее я расскажу, как при работе на клавиатуре можно сэкономить довольно значительное время.

Вырезание и вставка

Прежде всего, можно копировать и вставлять текст в оболочке командной строки. Стоит сказать, что знакомые вам сочетания клавиш Ctrl+C (копировать) и Ctrl+V (вставить) не будут работать в оболочке Bash. Вместо них используйте сочетания клавиш Shift+Ctrl+C и Shift+Ctrl+V. Вы также можете вырезать и вставлять текст, щелкнув правой кнопкой мыши и выбрав соответствующую операцию в меню. Поверьте, это может быть очень удобно. Представьте, что вы столкнулись с действительно длинной последовательностью команд из некоего онлайн-источника, которая выглядит примерно так:

```
$ find ./ -name \*.html -printf '%CD\t%p\n' | grep "09/10/17"
➥ | awk '{print $2}' | xargs -t -i mv {} temp/
```

Вы хотите набирать все это вручную? Я — нет. Здесь на помощь приходят команды вырезания и вставки.

Автозавершение ввода

Программа Bash отслеживает ваше местоположение и среду и наблюдает за тем, как вы пишете команду. Если вводимые вами символы с учетом файлов и каталогов текущей среды содержат какие-либо подсказки о том, что вы хотите сделать, нажатие клавиши **Tab** позволит Bash подставить наиболее релевантный вариант в оболочке командной строки. Если вы удовлетворены предложением, нажмите клавишу **Enter**, и программа автоматически завершит команду.

Приведу один пример. Предположим, вы загрузили архивный файл программного обеспечения с глубокомысленным названием вроде `foo-matic-plus_0.9.1-3_amd64.deb`. Вы хотите скопировать его в рабочий каталог, где его можно будет извлечь. Для этого нужно набирать такую команду:

```
$ sudo cp foo-matic-plus_0.9.1-3_amd64.deb /usr/bin/foo-matic/
```

Но если файл находится в вашем текущем каталоге, где он единственный файл, имя которого начинается с букв *foo*, то вам нужно всего лишь набрать `cp foo` и нажать клавишу **Tab**. Программа Bash заполнит остальную часть имени файла за вас. Конечно, поскольку Bash не может читать ваши мысли, вам все равно придется набрать хотя бы часть целевого пути, чтобы программа смогла автоматически завершить команду.

Попробуйте сами. С помощью команды `touch` создайте файл с каким-либо длинным именем, а затем попробуйте удалить или скопировать его, используя автозавершение ввода. Вот что я придумал:

```
$ touch my-very-silly-filename_66-b.txt
$ rm my-<tab>
```

1.2.5. Псевдофайловые системы

Обычный файл представляет собой набор данных, к которым можно многократно получать надежный доступ даже после перезагрузки системы. В отличие от такого файла содержимое псевдо- (или виртуального) файла Linux, как и файлов, размещенных в каталогах `/sys/` и `/proc/`, в действительности не существует в привычном смысле слова. Содержимое псевдофайла динамически генерируется самой ОС для представления определенных значений.

Например, вам может быть любопытно узнать, сколько всего места на одном из ваших жестких дисков. Позвольте мне заверить вас, что Linux будет только рада сообщить вам об этом. Давайте используем консольную утилиту `cat` для вывода информации о том, какое количество байтов занимает файл на диске, обозначаемом системой как `sda`:

```
$ cat /sys/block/sda/size
1937389568
```

ПРИМЕЧАНИЕ

Если первое устройство хранения в системе называется /dev/sda, то, как можно догадаться, второе устройство будет называться /dev/sdb, а третье — /dev/sdc. Первоначально аббревиатура sda, вероятно, расшифровывалась как SCSI Device A, но я считаю, что расшифровка Storage Device A (устройство хранения A) лучше подходит. Вы также можете встретить обозначения устройств типа /dev/hda (жесткий диск), /dev/sr0 (дисковод DVD), /dev/cdrom (верно, это дисковод CD-ROM) или даже /dev/fd0 (дисковод для гибких дисков).

Для получения такой информации есть гораздо более простые способы. Например, вы можете щелкнуть правой кнопкой мыши на значке диска в графическом интерфейсе диспетчера файлов, но псевдофайлы в `/sys/` находятся в общем источнике, с которым связаны все системные процессы.

Не знаете, как у вас обозначен диск? Не проблема. Зная, что Linux организует подключенное хранилище как *блочное устройство*, вы можете перейти в каталог `/sys/block/` и просмотреть его содержимое. Среди содержимого будет каталог с названием `sda/`. (Помните, что sda расшифровывается как Storage Drive A.) Это первый диск, используемый вашей системой при загрузке:

```
$ cd /sys/block
$ ls
loop0  loop1  loop2  sda  sr0
```

Все доступные на данный момент блочные устройства. Устройство цикла — это псевдоустройство, которое позволяет использовать файл так, как если бы он был реальным физическим устройством

Перейдите в каталог `sda/` и выполните команду `ls`. Среди содержимого вы, вероятно, увидите файлы с такими именами, как `sda1`, `sda2` и `sda5`. Каждый из них представляет собой один из разделов, созданных в Linux для лучшей организации данных на диске.

```
$ cd sda
$ ls
alignment_offset  discard_alignment  holders    range      sda3     trace
bdi               events             inflight   removable  size     uevent
capability        events_async       integrity  ro         slaves
dev               events_poll_msecs  power      sda1       stat
device            ext_range          queue      sda2       subsystem
```

1.2.6. Покажите, кто в доме хозяин: sudo

По практическим соображениям использование учетной записи операционной системы, обладающей полными административными полномочиями, для повседневной вычислительной деятельности — неоправданно рискованное дело. С другой стороны, если полностью ограничить себя учетной записью без прав администрирования, становится практически невозможным выполнение каких-либо действий.

Многие системы из семейства Linux решают эту проблему, предоставляя отдельным учетным записям полномочия администратора, которые в большинстве случаев

носят чисто формальный характер, но при необходимости их можно получить, предварительно добавив в команду слово sudo. Как только вы подтвердите свой доступ к аккаунту, указав свой пароль, такая команда будет обрабатываться так, как если бы она была инициирована администратором.

```
$ cat /etc/shadow
cat: /etc/shadow: Permission denied
$ sudo cat /etc/shadow
[sudo] password for ubuntu:
```

Файл /etc/shadow не будет отображаться без полномочий sudo

ПРИМЕЧАНИЕ

По умолчанию пользователь, созданный во время первоначальной установки Linux, будет иметь полномочия администратора.

На скриншотах из оболочки командной строки в этой книге я использую строку с символом $ в начале для команд, которые не требуют прав администратора, и # вместо $ sudo — для тех команд, которым необходимы права. Таким образом, команда без прав администратора будет выглядеть так:

```
$ ls
```

А команда с полномочиями sudo будет выглядеть так:

```
# nano /etc/group
```

1.3. Получение справки

Так или иначе, работа над IT-проектами всегда доставляет определенные хлопоты. Возможно, придется устранять неполадки в приложении, с которым вы работаете впервые, или решать задачу, с которой вы давно не сталкивались. Кроме того, не каждый помнит точный синтаксис той или иной команды. В любом случае вам может понадобиться помощь. Вот несколько хороших мест для поиска информации.

1.3.1. Man-файлы

По общепринятому соглашению люди, которые создают и поддерживают консольное программное обеспечение в Linux, параллельно пишут сложноструктурированные руководства, называемые *man-файлами*. Когда устанавливается программа для Linux, ее man-файл почти всегда устанавливается вместе с ней и его можно просмотреть в оболочке командной строки, набрав слово man, а затем имя команды. Даже у команды man есть man-файл, поэтому мы начнем с него:

```
$ man man
```

Выполнив эту команду на своем компьютере, вы увидите, что в разделе NAME содержится краткое введение, SYNOPSIS предлагает подробный обзор синтаксиса,

а **DESCRIPTION** предоставляет более подробное описание программы, которое обычно включает в себя список аргументов командной строки и флаги. Если вам повезет, вы также найдете несколько полезных примеров.

Иногда man-файлы могут быть довольно объемными, поэтому просмотр документа в поисках одной конкретной детали не всегда практичное решение. По различным причинам сочетание Ctrl+F, нажатие которого запускает локальные операции поиска в более современных приложениях, таких как браузеры и текстовые редакторы, здесь недоступно. Вместо этого нажмите клавишу /: в нижней части экрана откроется поле ввода текста, где вы можете ввести свой поисковый запрос. Если первый выделенный результат не соответствует желаемому, нажмите клавишу N (столько раз, сколько необходимо), чтобы выполнять поиск в документе, пока не найдете то, что ищете.

1.3.2. Команда info

Man-файлы великолепны, но только если вы знаете название команды или программы, которую ищете. Но предположим, что вы забыли название команды. Введите слово `info` в оболочке командной строки, и вы перенесетесь в среду, которая по стандартам Bash является полностью интерактивной (рис. 1.3):

```
$ info
```

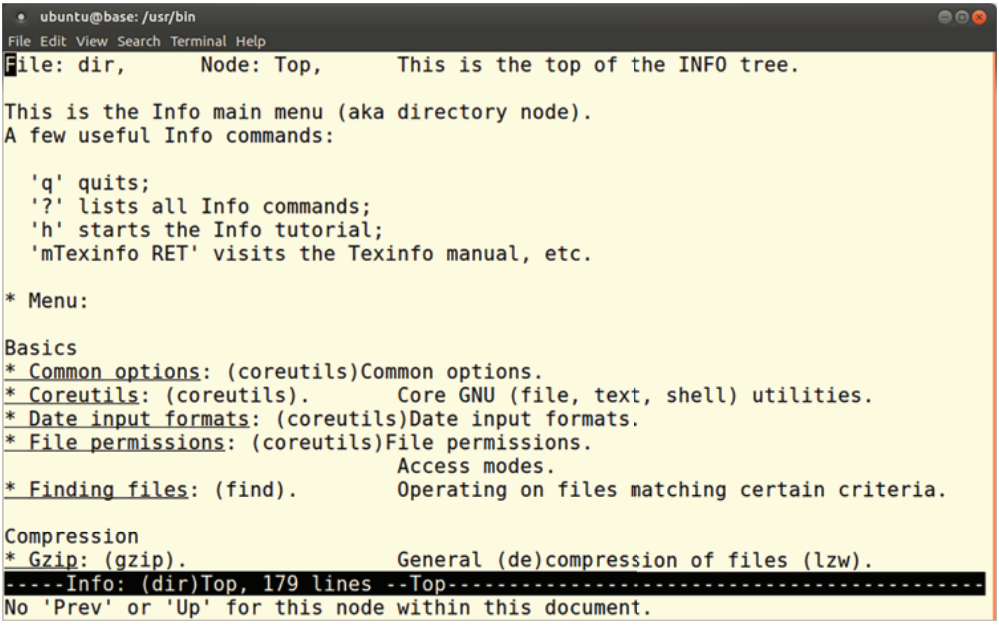


Рис. 1.3. Первый экран главного меню программы Info. Его вид может отличаться в вашей системе в зависимости от того, какое программное обеспечение вы установили

Как видно из рис. 1.3, содержимое структурировано в алфавитном порядке по темам с такими заголовками, как **Basics** и **Compression**. Вы можете использовать клавиши ↑ и ↓ для прокрутки содержимого экрана. Когда доберетесь до интересующей вас темы, нажмите клавишу **Enter**, чтобы перейти на страницу выбранной темы.

Предположим, вы хотите больше узнать о правах доступа к файлам. Прокрутите вниз раздел **Basics**, пока не доберетесь до пункта **File Permissions**, и нажмите клавишу **Enter**. Раздел **Menu** на этой странице указывает на то, что следующие строки являются ссылками на другие страницы, расположенные ниже. Нажатие клавиши **U** вернет вас на один уровень вверх, а нажатие клавиши **Q** приведет к выходу из программы Info.

У меня есть ощущение, что программа Info не так популярна у пользователей, как следовало бы. Надо признать, что у меня самого есть страшная тайна, связанная с Info, — я работал с Linux почти десять лет, прежде чем обратил внимание на эту программу!

По умолчанию программа Info может быть не установлена в некоторых дистрибутивах Linux. Если ввод команды `info` в оболочке командной строки не приносит результатов, вы можете установить Info (в системах Ubuntu/Debian) с помощью команды `sudo apt install info`.

1.3.3. Всемирная паутина

Независимо от того, насколько умными вы себя считаете, я могу заверить вас, что тысячи администраторов Linux с опытом любого уровня сталкивались с однотипными проблемами и решали их. Многие из этих решений были результатом обращения за помощью на форумах интернет-сообществ, например **serverfault.com** или **linuxquestions.org/questions**.

Конечно, вы всегда можете разместить свои вопросы на этих сайтах, но зачем? Поисковые системы в Интернете отлично справляются с поиском ответов на вопросы, которые были заданы ранее. Правильно сформированный поисковый запрос позволит вам быстрее найти решение, чем пришлось бы дожидаться ответа на собственный вопрос на таком сайте.

Фишка заключается в умении искать правильно. Если вы наберете в поисковике **мой сервер упал** в надежде на лучшее, это, вероятно, не принесет результатов. Очевидно, нужно больше подробностей. Прекрасно. Укажите, какой это сервер: веб-сервер Apache? Появились ли какие-либо сообщения об ошибках в вашем браузере? Произошел ли сбой каких-либо записей в журнале? Вероятно, было бы неплохо это выяснить.

Получение информации об ошибках из системных журналов

Почти во всех современных дистрибутивах Linux (за исключением Ubuntu 14.04) вы можете получить доступ ко всем системным журналам с помощью команды `journalctl`:

```
# journalctl
```

Выполнив команду `journalctl` без каких-либо аргументов, вы утонете в потоке данных. Вам нужно будет найти какой-то способ отфильтровать информацию, которую вы ищете. Позвольте мне представить вам ключевое слово `grep`:

```
# journalctl | grep filename.php
```

Символ | (пайп) использует выходные данные одной команды (например, journalctl) как ввод для следующей (grep)

В этом примере я использую вертикальную линию (`|`), которая вводится на американской раскладке клавиатуры нажатием сочетания клавиш Shift+\. Результат выполнения команды `journalctl` направляется в фильтр `grep`, выводящий на экран только те строки, которые содержат выражение `filename.php`. Разумеется, я предполагаю, что на вашем веб-сервере существует файл с именем `filename.php`. Не то чтобы я никогда так не делал, но обычно я даю своим файлам гораздо более описательные и полезные имена, например `stuff.php`.

Вы можете использовать `grep` последовательно, чтобы сузить результаты. Предположим, существует слишком много записей в журнале для `filename.php` и вы поняли, что вам нужны только те, которые также содержат слово *error*. Вы можете передать результаты первой операции второй команде `grep`, отфильтровав по *error*:

```
# journalctl | grep filename.php | grep error
```

Если вы предпочитаете видеть только те строки, в которых нет слова *error*, добавьте `-v` (для обращения запроса):

```
# journalctl | grep filename.php | grep -v error
```

Поиск в Интернете

Теперь представьте, что вывод, который вы получили из `journalctl`, включает такой текст:

```
[Fri Oct 12 02:19:44 2017] [error] [client 54.208.59.72]
➥ Client sent malformed Host header
```

Это может быть полезно. Нет смысла искать в Интернете дату или этот конкретный IP-адрес, но держу пари: кто-то уже сталкивался с тем, что клиент отправил искаженный заголовок хоста.

Чтобы сократить количество ложных результатов, вы можете заключать слова в кавычки — тогда ваша поисковая система будет подбирать только те результаты, что соответствуют именно этой фразе. Другой способ уменьшения ложных результатов — указать поисковой системе игнорировать страницы, содержащие определенную строку.

В следующем довольно смешном примере вы ищете в Интернете достойный мануал по написанию сценариев для Linux. Основываясь на большинстве результатов, выданных поисковой системой, вы обнаружите, что для написания сценариев вам предпочтительнее жить в Голливуде. Вы можете решить эту проблему, исключив страницы, содержащие слово *movie*:

```
writing scripts -movie
```

Резюме

- Почти любая консольная операция в Linux будет использовать некоторые или все пять основных инструментов: `ls`, `pwd`, `cd`, `cat` и `less`.
- Linux использует псевдофайловые системы для предоставления данных об аппаратной среде процессам и пользователям.
- Авторизованные пользователи могут указывать ключевое слово `sudo`, чтобы получить права администратора для отдельных команд.
- Существует много документации и другой справочной информации, доступной в Интернете, а также при использовании man-файлов и программы Info.

Ключевые понятия

- *Файловая система* состоит из файлов данных, индексированных таким образом, что позволяет воспринимать их как организованные на основе каталогов.
- *Процесс* является активным экземпляром работающей программы.
- *Дисковый раздел* — это логический отдел на физическом устройстве хранения данных, который может работать так же, как автономное устройство. Разделы являются общими организационными инструментами для всех современных операционных систем.
- *Bash* — это пользовательский консольный интерфейс для выполнения системных задач.
- *Простой текст*, который можно использовать для административных целей, представляет собой текст, состоящий из ограниченного набора символов и не содержащий постороннего кода форматирования.
- *Подстановочные знаки* применяются для отсылки на несколько файлов с помощью одной команды.
- *Для автозавершения команд* предназначена клавиша Tab.
- *Псевдофайловые системы* — это каталоги, содержащие файлы с динамическими данными, автоматически генерируемыми в процессе или после загрузки системы.

Рекомендации по безопасности

Старайтесь не работать на своем Linux-компьютере от имени администратора (root-пользователя). Вместо этого задействуйте обычную учетную запись пользователя и, когда вам нужно выполнить задачи администрирования, используйте `sudo`.

Обзор команд

- `ls -lh/var/log` перечисляет содержимое каталога и выводит полные, удобные для чтения сведения о папке `/var/log/`.

- `cd` возвращает вас в ваш домашний каталог.
- `cp file1 newdir` копирует файл с именем `file1` в каталог с именем `newdir`.
- `mvfile? /some/other/directory/` перемещает в указанное место все файлы, содержащие в имени буквы *file* и еще один символ.
- `rm -r *` удаляет все файлы и каталоги в текущем местоположении. **Использовать с большой осторожностью!**
- `man sudo` открывает man-файл со сведениями о команде `sudo`.

Самотестирование

1. Какой из следующих дистрибутивов Linux лучше всего подходит для безопасной работы:
   а) OpenSUSE;
   б) CentOS;
   в) Kali Linux;
   г) LXLE?
2. Какой из следующих инструментов позволяет редактировать текст в оболочке командной строки:
   а) nano;
   б) gedit;
   в) touch;
   г) LibreOffice?
3. Что делает команда `ls` с аргументом `-l`:
   а) выводит подробные сведения о файле;
   б) выводит информацию в удобочитаемом формате;
   в) отображает только имена файлов;
   г) отображает подкаталоги рекурсивно?
4. Какая из следующих команд отобразит ваше текущее местоположение в файловой системе:
   а) `touch`;
   б) `pwd`;
   в) `ls -c`;
   г) `cd`?
5. Для чего нужна команда `cat /etc/group`:
   а) отображает содержимое файла `/etc/group` с возможностью навигации;
   б) копирует файл `/etc/group` в новое указанное место;
   в) обновляет значение последнего доступа к файлу `/etc/group`;

г) выводит содержимое файла `/etc/group` на экран (с прокруткой содержимого на экране)?

6. Какая из этих команд удалит каталог `myfulldirectory`, содержащий файлы и подкаталоги:
   а) `rmdir myfulldirectory`;
   б) `sudo rmdir myfulldirectory`;
   в) `rm -r myfulldirectory`;
   г) `rm myfulldirectory`?
7. Допустим, что нет каталога с именем `mynewfile`. Что будет делать команда `mv myfile mynewfile`:
   а) создаст копию файла `myfile` с именем `mynewfile`;
   б) создаст пустой каталог с именем `mynewfile`;
   в) создаст пустой каталог с именем `mynewfile` и переместит в него файл `myfile`;
   г) изменит имя файла `myfile` на `mynewfile`?
8. Какая команда из нижеследующих удалит все файлы со словом *file* и с любым количеством символов в его имени:
   а) `rm file*`;
   б) `rm file?`;
   в) `r file.`;
   г) `rm file???`

Ответы
1 — в; 2 — а; 3 — а; 4 — б; 5 — г; 6 — в; 7 — г; 8 — а.

2 Виртуализация Linux: создание безопасной и простой рабочей среды

В этой главе

- Определение правильной технологии виртуализации.
- Использование менеджеров репозитория Linux.
- Создание эффективных сред с использованием VirtualBox.
- Построение контейнеров с помощью LXC.
- Профессиональное управление виртуальными машинами.

Виртуализация — это самая важная технология, стоящая за почти всеми недавними улучшениями в предоставлении сервисов и продуктов. Она сделала не только возможными, но и необходимыми целые отрасли промышленности: от облачных вычислений до самоуправляемых автомобилей. Заинтересовались? Вот два факта о виртуализации, которые вам нужно знать с самого начала.

- Linux абсолютно доминирует в виртуальном пространстве.
- Виртуализация облегчает изучение любой технологии.

Эта глава дает представление о доминирующих технологиях виртуализации уровня предприятия, используемых в настоящее время. Но что еще важнее, здесь вы также узнаете, как задействовать виртуальную среду, чтобы безопасно получать

в ней навыки администрирования Linux. Почему эта довольно сложная технология так рано появляется в книге? Потому что так вам будет намного легче проработать остальные главы.

Нужна свежая, чистая операционная система (ОС), чтобы попробовать что-то новое? Создайте ее за несколько секунд. Сделали ошибку в конфигурации и она заблокировала вам доступ к машине? Нет проблем. Уничтожьте эту конфигурацию и запустите новую. Вы узнаете, как использовать диспетчеры пакетов Linux для загрузки и установки всего необходимого программного обеспечения (например, VirtualBox и LXC) и управления им.

2.1. Что такое виртуализация

Когда вам понадобился новый сервер, чтобы запустить веб-сервер или хранилище документов с совместным доступом для вашей компании или ее клиентов, вам пришлось провести исследование, запросить одобрение бюджета, согласовать, заказать, безопасно разместить, обеспечить и затем запустить совершенно новую машину. Этот процесс от начала до конца мог занять месяцы (поверьте мне, я с таким сталкивался). И если бы растущие потребности сервиса грозили превысить возможности сервера, вам пришлось бы повторить все сначала в надежде правильно угадать соотношение «производительность/требования».

Стандартный сценарий предполагает, что компания предоставляет несколько взаимозависимых сервисов, каждый из которых работает на собственном оборудовании. Представьте себе фронтенд-веб-сервер, развернутый вместе с базой данных в бэкенде. Однако спустя время вы сталкиваетесь с ситуацией, когда один сервер используется недостаточно эффективно, а другой (как правило, рядом с первым в стойке) не справляется с нагрузкой. Но представьте, что вы можете безопасно разделить память, хранилище и сетевые ресурсы одного высокопроизводительного сервера между несколькими сервисами. Вообразите себе возможность выделять экземпляры виртуальных серверов из этого физического сервера, назначая им только необходимый уровень ресурсов, а затем мгновенно настраивая ресурсы для удовлетворения меняющихся потребностей.

А теперь представьте, что вы можете эффективно упаковать дюжины этих виртуальных компьютеров, работающих под управлением нескольких операционных систем, на одном сервере с «голым железом» так, чтобы абсолютно ничего не простаивало. Представьте себе, что эти виртуальные машины (ВМ) могут автоматически распространяться на другие физические серверы при заполнении первых. Подумайте о том, как удобна возможность уничтожить виртуальную машину, которая вышла из строя или нуждается в обновлении, и заменить ее так быстро, что пользователи даже не поймут, что что-то изменилось. Представили себе это (надеюсь, получилось что-то вроде рис. 2.1)? Это и есть *виртуализация*.

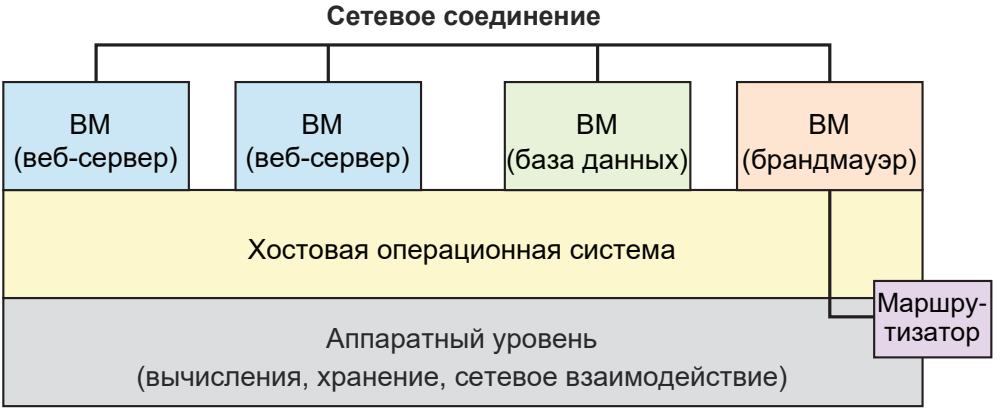


Рис. 2.1. Виртуальные машины — клиенты аппаратного хоста с возможностью их соединения между собой и с большей сетью через внешний маршрутизатор

Этот подход настолько привлекателен, что теперь доминирует в мире корпоративных вычислений. Я сомневаюсь, что на данный момент осталось много локальных или облачных серверных приложений, которые не работают с какой-либо технологией виртуализации. И Linux — та операционная система, на которой работает большинство этих виртуальных рабочих нагрузок.

Amazon Web Services (AWS), кстати, позволяют пользователям арендовать мощности на (Linux-) серверах, где размещены миллионы виртуальных машин, на которых, в свою очередь, работает бесчисленное множество полезных программ, включая большинство наиболее популярных онлайн-сервисов. На рис. 2.2 показано, как экземпляр ВМ AWS Elastic Compute Cloud (EC2) служит центром для целого комплекса инструментов хранения, баз данных и сетевого взаимодействия.

Не волнуйтесь, если некоторые из этих деталей AWS немного непонятны — мы не будем изучать их в этой книге. Но если вам захочется больше узнать о Amazon Web Services, вы всегда можете прочитать мою книгу *Learn Amazon Web Services in a Month of Lunches* (Manning, 2017). А что насчет виртуализации? У меня есть книга и на эту тему: *Teach Yourself Linux Virtualization and High Availability* (LULU Press, 2017).

Следующий короткий раздел может показаться немного трудным, но он обеспечит некоторой информацией тех из вас, кто хочет понимать, как все работает, так сказать, за кадром. При успешной виртуализации используется какое-то изолированное пространство на физическом компьютере, куда можно установить гостевую ОС, а затем «убедить» ее, что она одна на своем собственном компьютере. Гостевые операционные системы могут совместно использовать сетевые подключения, чтобы их администраторы могли удаленно входить в систему (о чем я расскажу в главе 3)

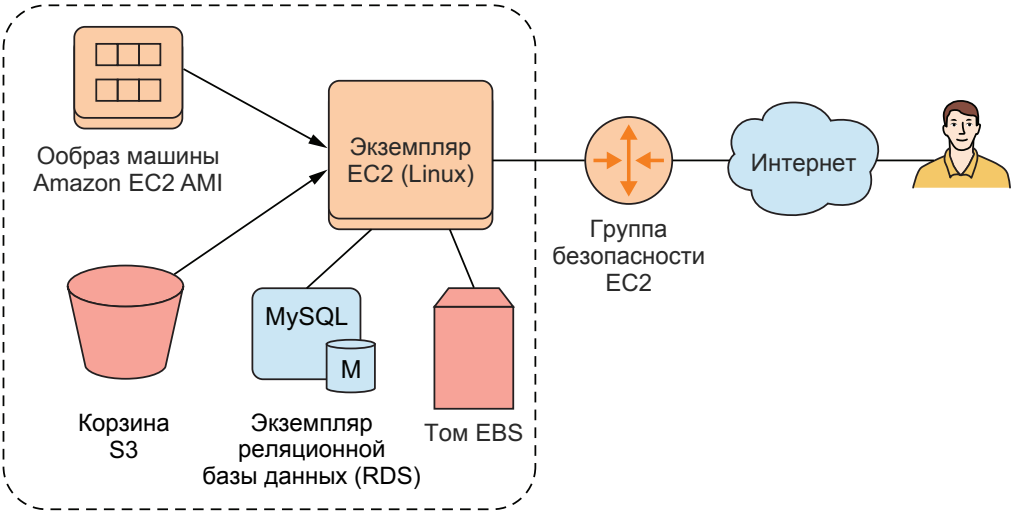


Рис. 2.2. Типичная рабочая схема при облачных вычислениях, сосредоточенная вокруг экземпляров виртуальных машин AWS Elastic Cloud Compute (EC2) на Amazon Web Services

и выполнять свою работу точно так же, как на традиционных машинах. Те же общие сетевые подключения позволяют вам использовать виртуальные машины для предоставления публичных сервисов, таких как сайты. Вообще говоря, в настоящее время существует два средства виртуализации.

- *Гипервизоры* — в той или иной степени управляют оборудованием хост-системы, предоставляя каждой гостевой ОС необходимые ей ресурсы (рис. 2.3). Гостевые машины запускаются как системные процессы, но с виртуализированным доступом к аппаратным ресурсам. Например, серверы AWS давно построены на технологии гипервизора Xen с открытым исходным кодом (хотя недавно они начали переключать некоторые из своих серверов на платформу KVM все с тем же открытым исходным кодом). Другие важные гипервизорные платформы включают VMware ESXi, KVM и Microsoft Hyper-V.
- *Контейнеры* — чрезвычайно легкие виртуальные серверы, которые вместо того, чтобы работать как полноценные операционные системы, совместно используют ядро своей хостовой машины (рис. 2.4). Контейнеры могут быть построены на базе текстовых сценариев, созданы и запущены за считаные секунды, а также легко доступны для совместного использования в сетях. Самая известная контейнерная технология сейчас, вероятно, Docker. Проект Linux Container (LXC), с которым мы будем работать в этой главе, послужил изначальным источником вдохновения для Docker.

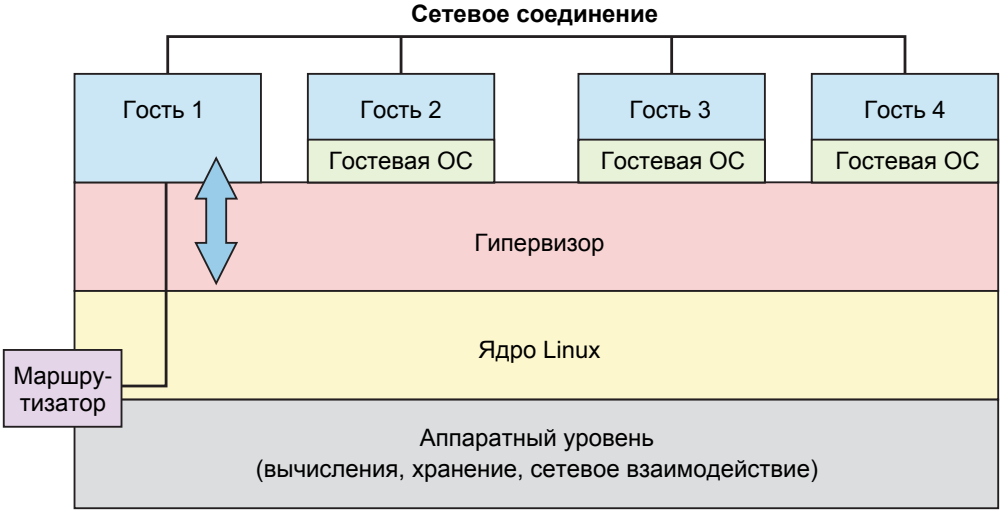


Рис. 2.3. Архитектура гипервизора типа 2, демонстрирующая целые операционные системы, установленные на каждом госте, с некоторыми специальными административными обязанностями, делегированными гостю 1

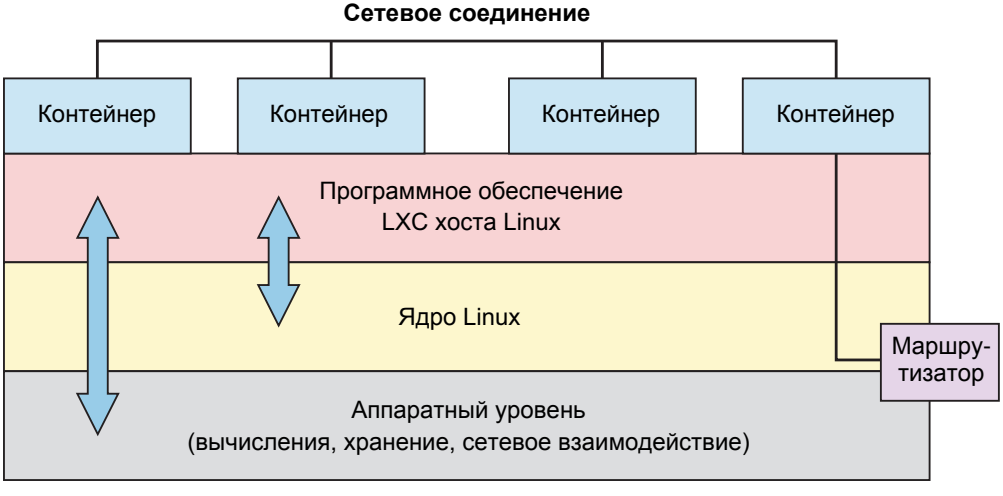


Рис. 2.4. Архитектура LXC, иллюстрирующая возможность доступа между средой LXC и как ядром Linux, так и аппаратным уровнем под ним

Ни одна технология не подходит для всех проектов идеально. Но если вы решите прочитать эту главу до конца, то узнаете, как и зачем использовать две технологии виртуализации: VirtualBox (гипервизор типа 2) и, как я упоминал ранее, LXC (менеджер контейнеров).

Проектные соображения

Я бы не хотел, чтобы вы закончили читать эту книгу, не получив некоторых базовых рекомендаций по выбору технологий виртуализации как минимум, поэтому вот несколько соображений.

- Полномасштабные гипервизоры, такие как Xen и KVM (через интерфейс управления, такой как Libvirt), обычно используются для развертываний на уровне предприятия, так как имеют большой парк виртуальных машин Linux.
- VirtualBox (и VMware Player) идеально подходит для тестирования работающих операционных систем и экспериментов с ними, по одной или две за раз, без необходимости их установки на настоящие ПК. При этом относительно высокие накладные расходы делают их непривлекательными для большинства производственных сред.
- Контейнерные технологии, такие как LXC и Docker, попроще и могут быть подготовлены и запущены в считаные секунды. Контейнеры LXC особенно хорошо подходят для обкатки новых технологий и безопасного создания стеков программного обеспечения ОС. В настоящее время Docker — это технология, управляющая бесчисленными динамическими, интегрированными парками контейнеров в рамках обширной архитектуры микросервисов. (Я немного больше расскажу о микросервисах в главе 9.)

2.2. Работа с VirtualBox

Вы можете многое сделать с помощью программного обеспечения с открытым исходным кодом VirtualBox компании Oracle. Вы можете установить его на любой ОС (включая Windows), работающей на любом портативном компьютере или ноутбуке, или использовать как ВМ практически для любой ОС.

Установка VirtualBox в среде Windows

Хотите попробовать все это на ПК с Windows? Зайдите на сайт VirtualBox (www.virtualbox.org/wiki/Downloads) и загрузите исполняемый архив. Щелкните кнопкой мыши на файле, который скачали, и выполните несколько шагов по настройке (можно оставить значения по умолчанию). Наконец, программа установки спросит вас, можно ли сбросить сетевые интерфейсы и хотите ли вы установить VirtualBox. Ответьте утвердительно.

VirtualBox предоставляет среду, в которой вы можете запустить столько виртуальных компьютеров, сколько в состоянии обрабатывать ваша физическая система. И это особенно полезный инструмент для безопасного тестирования и получения новых навыков администрирования, что и является нашей основной целью. Но, прежде чем это произойдет, вам нужно знать, как работает загрузка и установка программного обеспечения в Linux.

2.2.1. Работа с менеджерами пакетов Linux

Успешно установить VirtualBox на машине под управлением Ubuntu просто. Для этого потребуются две команды:

```
# apt update
# apt install virtualbox
```

ПРИМЕЧАНИЕ

Как вы помните, приглашение # означает, что для этой команды требуются права администратора, которые обычно можно получить, предварительно указав команду с помощью sudo.

Что происходит в нашем примере? Все вращается вокруг менеджера пакетов программного обеспечения, который называется Advanced Package Tool (APT, более известный как apt). В мире Linux менеджеры пакетов подключают компьютеры к огромным онлайн-хранилищам тысяч программных приложений, большинство из которых являются бесплатными или имеют открытый исходный код. У менеджера, который по умолчанию устанавливается с Linux, есть несколько заданий.

- Поддерживать локальный индекс для отслеживания репозиториев и их содержимого.
- Отслеживать состояние всего программного обеспечения, установленного на вашей локальной машине.
- Гарантировать, что все доступные обновления применяются к установленному программному обеспечению.
- Гарантировать, что программные зависимости (другие программные пакеты или параметры конфигурации, требуемые пакетом, который вы устанавливаете) соблюдаются для новых приложений перед их установкой.
- Поддерживать установку и удаление пакетов программного обеспечения.

На рис. 2.5 показаны некоторые элементы текущих отношений между онлайн-хранилищем программного обеспечения и менеджером пакетов, работающим на компьютере с Linux.

Система работает отлично, и по различным причинам за пределами мира Linux ничего подобного не наблюдается. Дело в том, что используемый вами менеджер будет зависеть от вашего конкретного дистрибутива Linux. В общем, если ваш дистрибутив относится к семейству Debian/Ubuntu, то вам стоит использовать APT. Члены семейства Red Hat будут использовать менеджер RPM и Yum (или его новую замену DNF). В табл. 2.1 перечислены варианты операционных систем для каждого из менеджеров.

Рис. 2.5. Отношения между основными репозиториями программного обеспечения, зеркальными серверами загрузки и Linux, запущенной на компьютере конечного пользователя

Таблица 2.1. Менеджеры пакетов и дистрибутивы

Менеджер пакетов	Дистрибутив
APT	Debian
	Ubuntu
	Mint
	Kali Linux
RPM	Red Hat Enterprise Linux
	CentOS
	Fedora
YaST	SUSE Linux
	OpenSUSE

Помимо использования менеджера пакетов для установки программного обеспечения из удаленных репозиториев, вам может потребоваться загрузить программное обеспечение с сайта. Часто можно найти пакеты, которые были отформатированы их разработчиками для работы с APT или Yum после установки из командной строки с помощью бэкенд-утилит. Например, вы хотите использовать Skype. Перейдя

на страницу загрузки (рис. 2.6), вы сможете загрузить файл DEB или RPM пакета Skype для Linux. Выбирайте DEB, если используете дистрибутив на основе Debian и APT, или RPM, если работаете в системе RedHat с менеджером Yum.

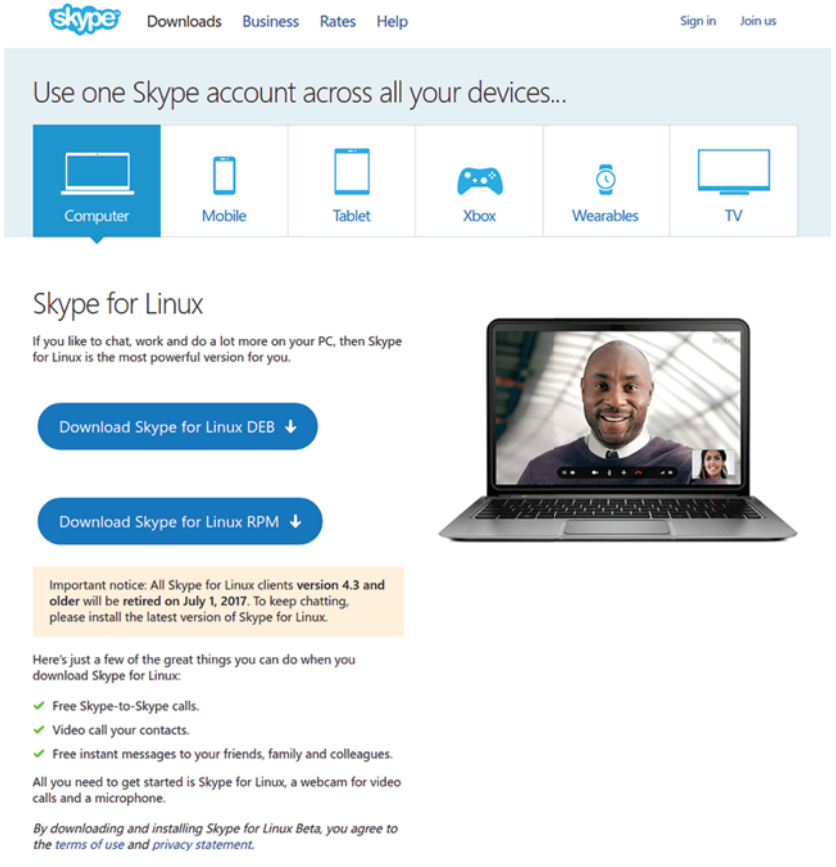


Рис. 2.6. Страница загрузки Skype для Linux. Обратите внимание на разные ссылки для менеджеров пакетов Debian (APT) и RPM (Yum)

Работа с менеджером пакетов Debian

Как только вы загрузили файл, можно установить его из оболочки командной строки, используя инструмент `dpkg`. Укажите флаг `-i` (для установки). Вам нужно убедиться, что вы запускаете команду `dpkg` из каталога, в котором находится файл `skypeforlinux-64`. В этом примере предполагается, что вы сохранили пакет в каталоге `Downloads` вашей учетной записи пользователя:

```
$ cd /home/<username>/Downloads
# dpkg -i skypeforlinux-64.deb
```

Команда dpkg должна позаботиться обо всех программных зависимостях. Но если этого не случилось, ее вывод обычно содержит достаточно информации, чтобы понять, что происходит.

Что значит «-64»

Linux, как и другие операционные системы для процессорной архитектуры x86, выпускается как в 64-разрядной, так и в 32-разрядной версии. Подавляющее большинство компьютеров, произведенных и проданных за последнее десятилетие, используют более быстрые 64-разрядные процессоры. Поскольку все еще существует частично устаревшее и ориентированное на такую разработку оборудование, вам иногда потребуется запускать 32-разрядную версию и вы захотите, чтобы установленное вами программное обеспечение работало с ней.

Вы можете проверить разрядность процессора самостоятельно, запустив команду `arch` в оболочке командной строки. Вряд ли вы другим образом узнаете, что работаете на старом оборудовании (это Linux делает особенно хорошо).

Установка VirtualBox для менеджера пакетов RPM

Немного раньше я упоминал короткие команды `apt update` и `apt install virtualbox`. Что они сделали? Для пояснения я установлю то же самое программное обеспечение VirtualBox на машину, на которой запущен дистрибутив Fedora Linux. Поскольку я буду использовать менеджер пакетов DNF от Red Hat, потребуется несколько дополнительных действий, и это хорошо, потому что так мы сможем увидеть, как выполняется этот процесс. Сам процесс немного сложнее, поэтому в табл. 2.2 перечислены нужные шаги.

Таблица 2.2. Процесс установки VirtualBox на Fedora

Задание	Команда
Добавить репозиторий	wget http://download.virtualbox.org/virtualbox/rpm/fedora/virtualbox.repo
Обновить индекс	dnf update
Установить зависимости	dnf install patch kernel-devel dkms
Установить пакет	dnf install VirtualBox-5.1

ПРИМЕЧАНИЕ

Эти шаги были протестированы в версии Fedora 25 и отлично иллюстрируют процесс управления пакетами. Однако все это может работать проще в более поздних релизах Fedora.

Возвращаясь к Ubuntu: APT распознал слово `virtualbox` в команде `install`, потому что пакет VirtualBox является частью онлайн-репозитория, с которым APT уже знаком. Однако оказывается, что RedHat и ее дочерние системы (такие как CentOS и Fedora) не настолько «мудры», по крайней мере не при первичной установке, поэтому мне понадобилось вручную добавить репозиторий `virtualbox` в Yum.

Из предыдущей главы вы узнали, что файлы конфигурации стороннего программного обеспечения часто хранятся в структуре каталогов `/etc/`, и в этом отношении yum/DNF ничем не различаются. Информация о репозитории хранится по адресу `/etc/yum.repos.d/`, поэтому вы должны перейти в этот каталог. Там вы используете программу `wget` (обычно устанавливаемую по умолчанию) для загрузки файла `.repo`. Вот как это сделать:

```
$ cd /etc/yum.repos.d/
# wget http://download.virtualbox.org/virtualbox/rpm/fedora/virtualbox.repo
```

Установка программного обеспечения в Linux

Конкретные инструкции по установке программного обеспечения в системе Linux, включая такие подробности, как точный URL-адрес, который я использовал ранее, почти всегда доступны в Интернете. Вы можете найти их либо на собственных сайтах разработчиков программного обеспечения, либо в тематических статьях. Интернет — ваш главный помощник.

При поиске убедитесь, что указали дистрибутив Linux, релиз и архитектуру в поисковой системе. Я нашел информацию о конкретных пакетах, необходимых для этого проекта, в моей любимой поисковой системе, что советую сделать и вам.

Наличие файла `.repo` в нужном каталоге не принесет большой пользы, пока вы не сообщите RPM, что произошло. Это можно сделать, выполнив команду `update`. Она также проверяет индекс локального репозитория по своим онлайн-аналогам, чтобы узнать, есть ли что-то новое, о чем вам следует знать. Независимо от того, каким менеджером вы пользуетесь, всегда полезно обновить информацию о репозитории перед установкой нового программного обеспечения:

```
# dnf update
Importing GPG key 0x98AB5139:
 Userid     : "Oracle Corporation (VirtualBox archive signing key)
➥ <info@virtualbox.org>"
 Fingerprint: 7B0F AB3A 13B9 0743 5925 D9C9 5442 2A4B 98AB 5139
 From       : https://www.virtualbox.org/download/oracle_vbox.asc
Is this ok [y/N]: y
Fedora 25 - x86_64 - VirtualBox          120 kB/s |  33 kB     00:00
Dependencies resolved.
Nothing to do.
Complete!
```

Все транзакции с репозиториями шифруются с использованием ключей GPG

Ссылки VirtualBox говорят о том, что я использую этот хост Fedora как виртуальную машину в VirtualBox

Следующий шаг — установка всех программных зависимостей, которые VirtualBox должен будет правильно использовать. *Зависимость* — это программное обеспечение, которое должно быть уже установлено на вашем компьютере для работы нового пакета. Возвращаясь к Ubuntu, следует сказать, что APT негласно заботится о соблюдении зависимостей; Yum также решает многие фоновые задачи. Но когда возникает необходимость в настройке зависимостей вручную, нужную справочную информацию легко найти в Интернете на сайтах, которые обсуждались ранее. Вот сокращенный вариант того, как может выглядеть процесс установки зависимостей:

```
# dnf install patch kernel-devel dkms
Last metadata expiration check: 0:43:23 ago on Tue Jun 13 12:56:16 2017.
[...]
Dependencies resolved.

================================================================================
 Package         Arch    Version                              Repository  Size
================================================================================
Installing:
dkms             noarch  2.3-5.20170523git8c3065c.fc25        updates     81 k
 kernel-devel    x86_64  4.11.3-202.fc25                      updates     11 M
 patch           x86_64  2.7.5-3.fc24                         fedora     125 k
Transaction Summary
================================================================================
Install  3 Packages
Total download size: 12 M
Installed size: 43 M
Is this ok [y/N]: y
Downloading Packages:
(1/3): dkms-2.3-5.20170523git8c3065c.fc25.noarc 382 kB/s |  81 kB     00:00
(2/3): patch-2.7.5-3.fc24.x86_64.rpm            341 kB/s | 125 kB     00:00
(3/3): kernel-devel-4.11.3-202.fc25.x86_64.rpm  2.4 MB/s |  11 MB     00:04
--------------------------------------------------------------------------------
Total                                           1.8 MB/s |  12 MB     00:06
[...]
Running transaction
  Installing  : kernel-devel-4.11.3-202.fc25.x86_64                       1/3
  Installing  : dkms-2.3-5.20170523git8c3065c.fc25.noarch                 2/3
  Installing  : patch-2.7.5-3.fc24.x86_64                                 3/3
  Verifying   : patch-2.7.5-3.fc24.x86_64                                 1/3
  Verifying   : kernel-devel-4.11.3-202.fc25.x86_64                       2/3
  Verifying   : dkms-2.3-5.20170523git8c3065c.fc25.noarch                 3/3

Installed:
  dkms.noarch 2.3-5.20170523git8c.fc25 kernel-devel.x86_64 4.11.3-202.fc25
  patch.x86_64 2.7.5-3.fc24
Complete!
```

Подтвердите операцию, введя y перед запуском

Краткий обзор успешной операции

Это лишь малая часть всего процесса установки, но вы наконец готовы установить VirtualBox на свою машину под управлением операционной системы RedHat, CentOS или Fedora. Номер версии, который указан в этом примере, взят

из онлайн-руководства, использованного ранее. Естественно, к тому времени, когда вы попробуете повторить мои действия, версия может стать уже далеко не 5.1.

DNF явно устраивают ранее установленные зависимости

```
# dnf install VirtualBox-5.1
Last metadata expiration check: 0:00:31 ago on Tue Jun 13 13:43:31 2017.
Dependencies resolved.
================================================================================
 Package             Arch    Version                     Repository      Size
================================================================================
Installing:
 SDL                 x86_64  1.2.15-21.fc24              fedora         213 k
 VirtualBox-5.1      x86_64  5.1.22_115126_fedora25-1    virtualbox      68 M
 python-libs         x86_64  2.7.13-2.fc25               updates        6.2 M
 qt5-qtx11extras     x86_64  5.7.1-2.fc25                updates         30 k
Transaction Summary
================================================================================
Install  4 Packages
[...]
Is this ok [y/N]: y
[...]
Creating group 'vboxusers'. VM users must be member
➥ of that group!
[...]
Installed:
  SDL.x86_64 1.2.15-21.fc24
  VirtualBox-5.1.x86_64 5.1.22_115126_fedora25-1
  python-libs.x86_64 2.7.13-2.fc25
  qt5-qtx11extras.x86_64 5.7.1-2.fc25
Complete!
```

Возвращает список всех пакетов, которые будут установлены во время этой транзакции

Обратите внимание, что процесс создал новую системную группу. Я расскажу о группах в главе 9

Дополнения VirtualBox

Вы должны знать, что Oracle предоставляет пакет расширений для VirtualBox. Этот пакет добавляет такие возможности, как поддержка USB-устройств и шифрование диска, а также предлагает некоторые альтернативы существующим параметрам загрузки. Не забывайте об этих инструментах на тот случай, если у вас возникнут трудности после установки того или иного стандартного пакета.

Вы также можете добавить дополнительную интеграцию файловой системы и устройств между виртуальными машинами VirtualBox и их хостом через образ CD-ROM VBox Guest Additions. Так вы получите возможность пользоваться общим буфером обмена и выполнять перетаскивание. Если дополнения Vbox еще недоступны на вашем хосте, установите пакет расширений в Ubuntu с помощью следующей команды:

```
sudo apt install virtualbox-guest-additions-iso
```

Затем добавьте CD-ROM в качестве виртуального оптического привода на работающую виртуальную машину. Ищите в Интернете документацию, касающуюся любых дополнительных пакетов, которые могут понадобиться для работы в вашей операционной системе.

Прежде чем вы по-настоящему перейдете к использованию инструментов виртуализации, таких как VirtualBox, я должен оставить вам хотя бы одну или две подсказки для отслеживания других пакетов репозитория, которые могут вам понадобиться. APT-системы позволяют напрямую искать доступные пакеты с помощью команды `apt search`.

В этом примере выполняется поиск пакетов, которые могут помочь вам контролировать состояние системы, а затем используется команда `apt show` для отображения полной информации о пакете:

```
$ apt search sensors
$ apt show lm-sensors
```

Если программа `aptitude` установлена, она выступает частично графической оболочкой, в которой вы можете отслеживать как доступные, так и уже установленные пакеты, а также управлять ими. Если вы не можете жить без мыши, обратите внимание на Synaptic — полноценный менеджер пакетов с графическим интерфейсом для ПК. В Yum также полностью реализованы инструменты поиска:

```
$ yum search sensors
$ yum info lm_sensors
```

2.2.2. Определение виртуальной машины (ВМ)

Я не уверен, что вы когда-нибудь собирали настоящий компьютер из комплектующих, но это могло бы помочь. Настройка новой виртуальной машины в VirtualBox выполняется примерно так же. Единственное существенное отличие состоит в том, что вместо того, чтобы, зажав между зубами фонарик, вручную вставлять планки ОЗУ и жесткий диск в компьютер, вы щелчком кнопкой мыши предоставляете возможность VirtualBox определять «аппаратные» характеристики виртуальной машины.

Выбрав пункт **New** (Создать) в интерфейсе VirtualBox, вы сообщите виртуальной машине, что готовы дать описательное имя. Как видно на рис. 2.7, программное обеспечение должно правильно автоматически заполнить поля **Type** (Тип) и **Version** (Версия). Тип и версия, которые вы здесь указываете, относятся не к настоящей, хостовой операционной системе, а используются для соответствующих настроек эмулируемого оборудования.

В следующем окне выделите оперативную память для своей виртуальной машины. Если не планируете что-то особенно требовательное, как, например, размещение ряда контейнеров или использование загруженного веб-сервера, то подойдет размер по умолчанию (768 Мбайт). Вы, конечно, можете выделить больше оперативной памяти, если это необходимо, но не забудьте оставить достаточно для вашего хост-компьютера и любых других виртуальных машин, которые уже могут в ней быть. Если на вашем хосте только 4 Гбайт физической памяти, вы, вероятно, не захотите отдавать половину этого объема своей виртуальной машине.

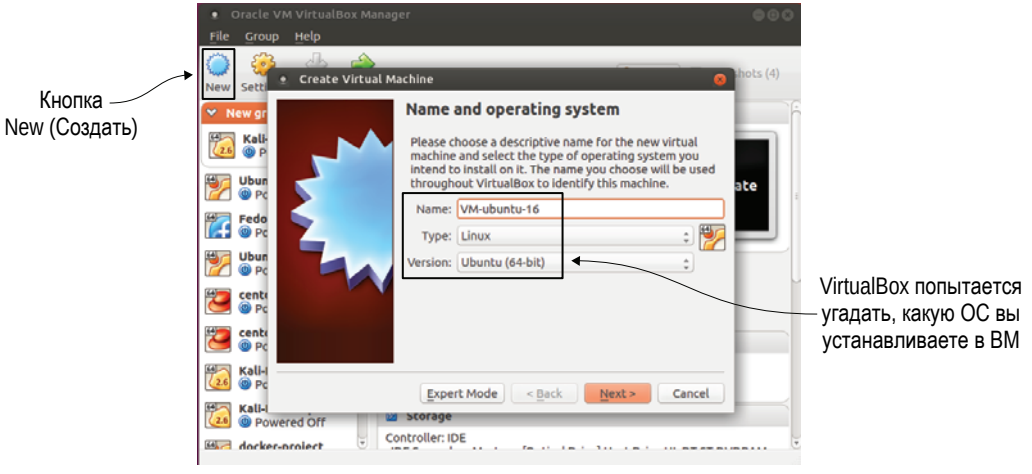


Рис. 2.7. Диалоговое окно Create Virtual Machine (Создание виртуальной машины): VirtualBox попытается угадать вашу ОС и ее версию, чтобы позже предложить правильный выбор по умолчанию

Помните об этих ограничениях, если решите запустить несколько виртуальных машин одновременно, что будет полезно для парочки проектов, о которых вы узнаете позже в этой книге. Даже если каждая виртуальная машина использует только объем памяти по умолчанию, две или три запущенные одновременно могут использовать всю оперативную память, необходимую для нормальной работы хоста.

Теперь мастер установки VirtualBox запрашивает, хотите вы создать новый виртуальный диск для своей виртуальной машины или использовать уже существующий (рис. 2.8). Что за компьютер без жесткого диска? Возможно, вы захотите разделить один диск между двумя виртуальными машинами, но в этом упражнении, я полагаю, вы предпочтете начать с нуля. Установите переключатель в положение **Create a virtual hard disk now** (Создать виртуальный жесткий диск сейчас).

В следующем окне (рис. 2.9) вы можете выбрать формат файла на жестком диске для вашей ВМ. Если не планируете в конечном итоге экспортировать диск для использования с какой-либо другой средой виртуализации, то формат VirtualBox Disk Image (VDI), указанный по умолчанию, будет нормально работать.

Я никогда не сожалел о том, что выбрал параметр **Dynamically allocated** (Динамически распределяемый) (рис. 2.10), чтобы указать, как виртуальный диск будет занимать пространство на хосте. Здесь *динамическое* распределение означает, что пространство на диске хоста будет выделяться виртуальной машине только по мере необходимости. Если виртуальная машина незначительно использует диск, то на хосте будет выделено меньше места.

Диск фиксированного размера, с другой стороны, сразу займет весь объем, независимо от того, сколько он фактически использует. Единственное преимущество фиксированного размера — производительность. Обычно я использую виртуальные машины VirtualBox только для тестирования и экспериментов, поэтому я обхожусь динамическим диском.

Рис. 2.8. Окно настройки жесткого диска. Обратите внимание на то, что в данном случае переключатель установлен в положение Use an existing virtual hard disk file (Использовать существующий файл виртуального жесткого диска)

Рис. 2.9. Виртуальные жесткие диски могут быть созданы с применением нескольких форматов. VDI подходит для виртуальных машин, которые будут использоваться только в VirtualBox

Рис. 2.10. Динамические виртуальные диски будут занимать столько места на устройствах хоста, сколько им нужно

Поскольку эмулятор знает, что вы используете Linux, и поскольку Linux эффективно задействует пространство для хранения, VirtualBox, вероятно, предложит вам только 8 Гбайт общего размера диска (рис. 2.11). Если у вас нет глобальных планов насчет ВМ (например, вы не собираетесь выполнять трудоемкие операции с крупными базами данных), этого, вероятно, будет достаточно. С другой стороны, если бы вы выбрали Windows в качестве виртуальной ОС, то по умолчанию было бы предложено 25 Гбайт — и на то есть веская причина: Windows не стесняется в требованиях к ресурсам. Это отличная иллюстрация того, почему Linux так хорошо подходит для виртуальных сред.

ПРИМЕЧАНИЕ

При желании вы также можете изменить название и расположение, которое VirtualBox будет использовать для вашего диска, в окне File location and size (Местоположение и размер файла).

Когда вы закончите, нажмите Create (Создать), и новая виртуальная машина появится в списке виртуальных машин в левой части менеджера VirtualBox. Но вы еще не закончили: вы создали только машину. Теперь вам понадобится ОС, чтобы она начала подавать признаки жизни.

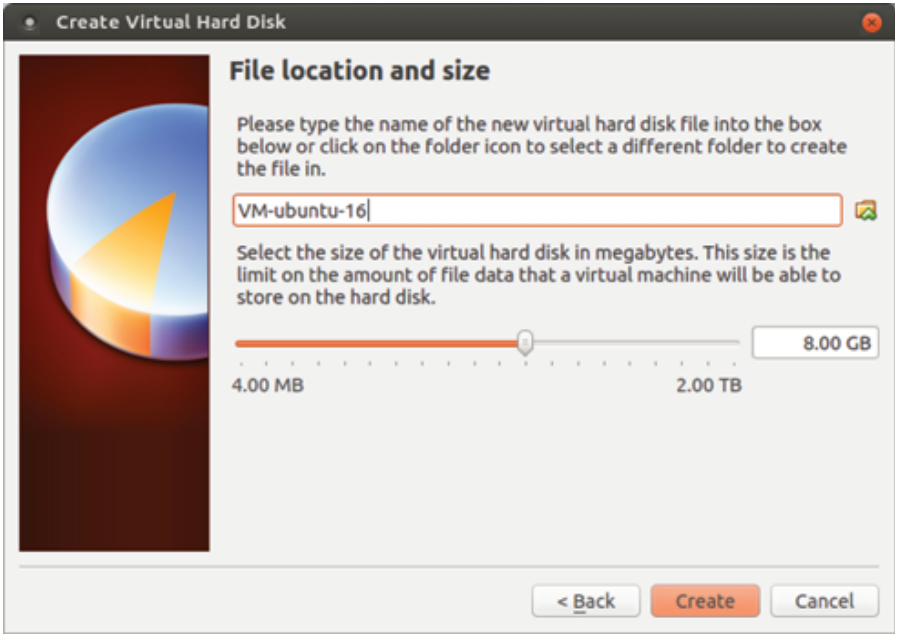


Рис. 2.11. При необходимости размер вашего виртуального диска можно расширить до 2 Тбайт или даже всего свободного пространства на хосте

2.2.3. Установка операционной системы

Теперь, когда вы определили профиль виртуального оборудования своей новой виртуальной машины, вам нужно сделать еще кое-что.

1. Скачать файл (в формате ISO) образа дистрибутива Linux, который вы хотите использовать.
2. Загрузить новую виртуальную машину, указав виртуальный привод DVD, в который «вставлен» скачанный ISO-образ.
3. Пройти стандартный процесс установки ОС.
4. Загрузить виртуальную машину и запустить установленную ранее ОС.

После того как вы выберете дистрибутив, вам нужно скачать файл в формате ISO, содержащий файлы ОС и программу установки. Подходящий файл обычно можно найти в Интернете по названию дистрибутива, добавив слово `скачать`. В случае с Ubuntu вы также можете открыть страницу **https://ubuntu.com/** и перейти на вкладку **Downloads** (Загрузки), как показано на рис. 2.12. Обратите внимание, что доступны различные типы Ubuntu. Если вы собираетесь использовать эту виртуальную машину для задач администрирования, то небольшая и быстрая версия **Server**, вероятно, будет лучшим выбором, чем **Desktop**.

Рис. 2.12. Меню Downloads (Загрузки) на сайте ubuntu.com. Обратите внимание на набор версий, которые предлагает Ubuntu.

Файлы большого размера при скачивании иногда могут повреждаться. Если хотя бы один байт в вашем ISO-файле был изменен, есть вероятность, что установка не пройдет должным образом. Вы наверняка не хотите тратить время и силы на обнаружение проблемы со скачиванием в процессе установки, поэтому советую сразу посчитать контрольную сумму (или *хеш*) загруженного файла ISO, чтобы подтвердить, что скачанный файл идентичен оригиналу. Для этого вам необходимо получить соответствующую контрольную сумму SHA или MD5, которая представляет собой длинную строку, выглядящую примерно так:

```
4375b73e3a1aa305a36320ffd7484682922262b3
```

Вы можете получить эту строку там же, откуда скачали свой файл ISO. В случае с Ubuntu нужно перейти на веб-страницу по адресу http://releases.ubuntu.com/, выбрать каталог, соответствующий скачанной версии, а затем щелкнуть на одной из ссылок на контрольную сумму (например, SHA1SUMS). Вы должны сравнить соответствующую строку на данной странице с результатами команды, запущенной в каталоге, куда вы скачали ISO-файл. Это выглядит следующим образом:

```
$ shasum ubuntu-16.04.2-server-amd64.iso
```

Если они совпадают, все хорошо. Если не совпадают (и вы дважды проверили, чтобы убедиться, что используете правильную версию), то вам, возможно, придется заново загрузить файл ISO.

ПРИМЕЧАНИЕ

Имейте в виду, что хеши бывают разных видов. В течение многих лет алгоритм MD5SUM был доминирующим, но SHA256 (с более длинными 256-битными хешами) набирает популярность. На практике для больших файлов образов ОС оба варианта, вероятно, равноценны.

Как только ваш файл ISO будет готов, вернитесь в VirtualBox. Только что созданная вами ВМ должна быть выделена на левой панели. Нажмите зеленую кнопку **Start** (Пуск) в верхней части приложения. Вам будет предложено выбрать файл ISO в вашей файловой системе для использования в качестве виртуального DVD-привода. Естественно, выбирайте тот файл, который только что скачали. Новая ВМ прочитает этот DVD и запустит установку ОС.

В основном процесс установки должен пройти нормально; однако поиск решений для каждой из множества возможных мелких проблем потребует еще пары полных глав. Если у вас возникли сложности, можете обратиться к документации и руководствам, которые доступны для любого дистрибутива Linux, или задать свой вопрос на форуме Manning в ветке книги *Linux in Action* и запросить помощь у сообщества.

Даже если установка прошла успешно, все еще может быть несколько задач, которые нужно выполнить, прежде чем вы сможете нормально загрузиться в свою виртуальную машину. Выделив виртуальную машину, щелкните кнопкой мыши на желтом значке **Settings** (Настройки). Здесь вы можете поэкспериментировать со средой вашей виртуальной машины и настройками оборудования. Например, выбрав раздел **Network** (Сеть), вы можете определить сетевое подключение. Если вы хотите, чтобы у вашей виртуальной машины был полный доступ к Интернету через сетевой интерфейс хост-компьютера, выберите пункт **Bridged Adapter** (Мост) в раскрывающемся списке **Attached to** (Присоединен к), как показано на рис. 2.13, а затем укажите название адаптера вашего хоста.

ПРИМЕЧАНИЕ

Не всегда нужно выбирать вариант Bridged Adapter (Мост). Иногда это может представлять угрозу безопасности. Фактически вариант NAT Network (Сеть NAT) предлагает более распространенный способ предоставления виртуальной машине доступа в Интернет. Поскольку во многих упражнениях этой книги требуется соединение нескольких виртуальных машин (что сложно реализовать с помощью NAT), я пока продолжу с мостом.

Возможно, вам никогда не понадобится последующая информация, но предупрежден — значит вооружен. В некоторых случаях для правильной загрузки виртуальной машины необходимо извлечь DVD из дисковода так же, как вы делаете при реальной физической установке. Для этого нужно перейти в раздел **Storage** (Хранилище). Щелкните кнопкой мыши на первом диске в списке, а затем на значке **Remove Disk** (Удалить диск) внизу (рис. 2.14). Убедитесь, что случайно не удалили свой жесткий диск!

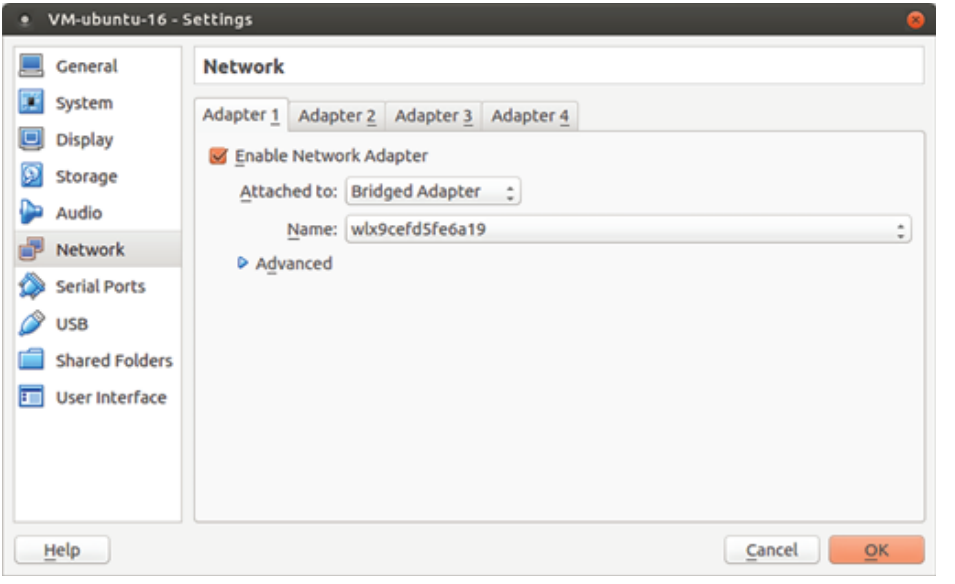


Рис. 2.13. Вкладка Network (Сеть) диалогового окна Settings (Настройки), где вы можете определить, какой тип сетевого интерфейса (или интерфейсов) использовать для своей виртуальной машины

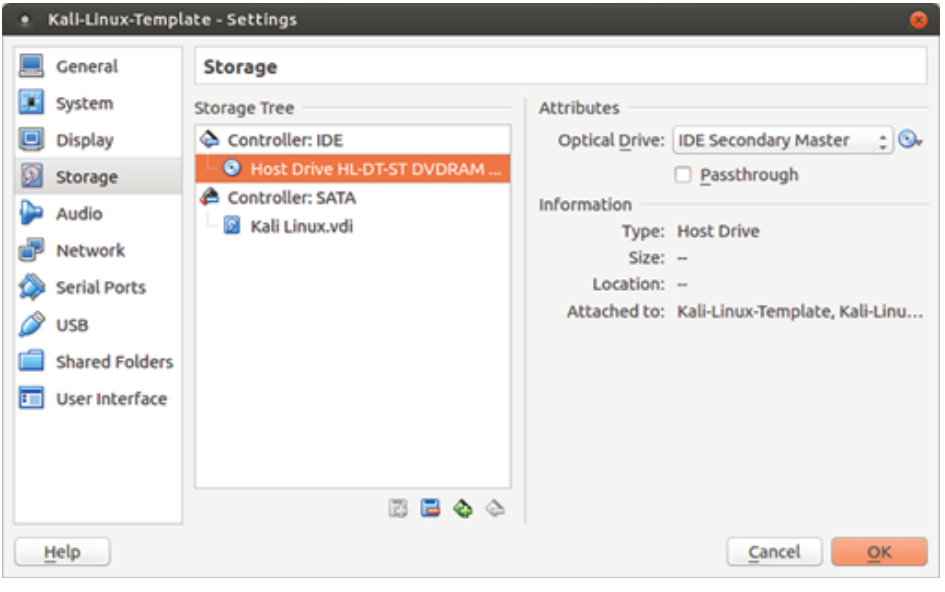


Рис. 2.14. Удалите виртуальный диск, щелкнув правой кнопкой мыши на его ссылке и выбрав пункт Delete (Удалить). Возможно, вам придется сделать это, чтобы убедиться, что виртуальная машина загружается на правильный диск

Возможно, вам иногда потребуется смонтировать DVD (или файл ISO), чтобы VirtualBox мог его распознать. Щелкнув на значке + при выделенной строке Controller: IDE (Контроллер: IDE), вы можете выбрать файл, который будет использоваться в качестве виртуального оптического привода.

2.2.4. Клонирование и совместное использование виртуальной машины VirtualBox

Информация из этого раздела не относится напрямую к теме главы, но кому не нравятся бесплатные бонусы? Я расскажу вам о двух связанных между собой хитростях: как клонировать виртуальные машины VirtualBox, чтобы ускорить запуск новых копий, и как использовать командную строку для совместного применения виртуальных машин в сети.

Клонирование виртуальных машин для быстрого запуска

Одним из наиболее очевидных преимуществ работы с виртуальными машинами является возможность быстрого доступа к свежей, чистой среде ОС. Но если для доступа к этой среде требуется пройти полный процесс установки, то не будет особой выгоды в скорости, пока вы не научитесь клонировать ВМ. Почему бы не сохранить исходную виртуальную машину в чистом состоянии после установки и создавать идентичный клон всякий раз, когда вы хотите выполнить какую-то работу?

Это легко. Посмотрите еще раз на приложение VirtualBox. Выберите (остановленную) виртуальную машину, которую хотите использовать в качестве главной копии, откройте меню Machine (Машина) и выберите команду Clone (Клонировать). Подтвердите название, которое хотите дать своему клону. Затем, после нажатия кнопки Next (Далее), выберите, хотите вы создать Full Clone (Полный клон) (то есть для новой виртуальной машины будут созданы совершенно новые копии файлов) или Linked Clone (Связанный клон) (то есть новая ВМ будет использовать все базовые файлы хозяина, при этом поддерживая ваш новый проект отдельно).

ПРИМЕЧАНИЕ

> Если вы выберете вариант Linked Clone (Связанный клон), процесс будет происходить намного быстрее и машина займет гораздо меньше места на жестком диске. Единственным недостатком будет то, что вы не сможете позже перенести этот конкретный клон на другой компьютер. Таков выбор.

Теперь нажмите кнопку Clone (Клонировать), и новая виртуальная машина появится на панели программы. Запустите ее как обычно, а затем войдите в систему, используя те же учетные данные, которые определили на исходной ВМ.

Управление виртуальными машинами из командной строки

VirtualBox поставляется с собственной оболочкой командной строки, которая вызывается с помощью команды `vboxmanage`. Зачем нам оболочка командной строки? Помимо прочего, она позволяет работать на удаленных серверах, что может значительно увеличить возможности потенциальных проектов. Чтобы увидеть, как работает команда `vboxmanage`, используйте инструкцию `list vms` для отображения всех виртуальных машин, доступных в настоящее время в вашей системе:

```
$ vboxmanage list vms
"Ubuntu-16.04-template" {c00d3b2b-6c77-4919-85e2-6f6f28c63d56}
"centos-7-template" {e2613f6d-1d0d-489c-8d9f-21a36b2ed6e7}
"Kali-Linux-template" {b7a3aea2-0cfb-4763-9ca9-096f587b2b20}
"website-project" {2387a5ab-a65e-4a1d-8e2c-25ee81bc7203}
"Ubuntu-16-lxd" {62bb89f8-7b45-4df6-a8ea-3d4265dfcc2f}
```

Команда `vboxmanage clonevm` выполнит такое же клонирование, как описанное раньше для графического интерфейса. Здесь я делаю клон шаблона виртуальной машины Kali Linux, называя копию `newkali`:

```
$ vboxmanage clonevm --register Kali-Linux-template --name newkali
```

Вы можете убедиться, что это сработало, еще раз запустив команду `vboxmanage list vms`.

Это подходит, пока мне нужно использовать новую виртуальную машину только здесь, на моем локальном компьютере. Но предположим, что я хочу, чтобы другие члены моей команды могли запустить точную копию этой виртуальной машины, возможно, для того, чтобы протестировать то, над чем я работал. Для этого мне нужно преобразовать виртуальную машину в какой-то стандартизированный файловый формат. Вот как я могу экспортировать локальную виртуальную машину в файл, используя открытый формат виртуализации (Open Virtualization Format):

```
$ vboxmanage export website-project -o website.ova
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Successfully exported 1 machine(s).
```

Флаг `-o` указывает выходное имя файла, в данном случае `website.ova`

Теперь вам необходимо скопировать файл OVA на компьютер вашего коллеги. Имейте в виду, что файл по всем параметрам не будет маленьким и удобным. Если вам не хватает пропускной способности сети для передачи нескольких гигабайт, рассмотрите возможность его копирования через USB-накопитель. Но если вы выберете копирование через сеть, лучшим инструментом для работы будет Secure Copy (`scp`). Вот как это может работать:

```
$ scp website.ova username@192.168.0.34:/home/username
```

Если весь этот код команды `scp` кажется немного странным, не беспокойтесь: помощь уже в пути. Команда `scp` будет полностью рассмотрена в главе 3, в части, посвященной OpenSSH. Тем временем команда `scp` будет работать только в том случае, если оболочка OpenSSH установлена на обоих компьютерах, у вас есть

авторизованный доступ к учетной записи на удаленном компьютере и она доступна с вашего локального компьютера.

Как только передача завершена, остается только импортировать с удаленного компьютера виртуальную машину в VirtualBox этой машины. Команда проста:

```
$ vboxmanage import website.ova
```

Убедитесь, что операция импорта успешно завершена, для чего выполните команду `list vms`. И попробуйте запустить виртуальную машину:

```
$ vboxmanage list vms
"website" {30ec7f7d-912b-40a9-8cc1-f9283f4edc61}
```

Если вам не нужен удаленный доступ, вы также можете расшарить виртуальную машину из графического интерфейса. Выделив в VirtualBox ВМ, которой вы хотите поделиться, выберите пункт меню **File ▸ Export Appliance** (Файл ▸ Экспортировать оборудование).

2.3. Работа с контейнерами Linux (LXC)

VirtualBox отлично подходит для выполнения операций, когда требуется доступ к ядру Linux (например, как если бы вы использовали функции безопасности наподобие SELinux, как вы увидите в главе 9), когда нужны сеансы рабочего стола с графическим интерфейсом пользователя или во время тестирования таких операционных систем, как Windows. Но если вам нужен быстрый доступ к чистой среде Linux и вы не ищете какой-либо специальной версии релиза, то лучше всего подойдет LXC.

ПРИМЕЧАНИЕ

Как и любая сложная система, LXC может не поддерживать некоторые аппаратные архитектуры. Если у вас возникли проблемы с запуском контейнера, вероятно, это проблема с совместимостью. В Интернете, как всегда, можно получить более точную информацию.

Насколько быстро работают контейнеры LXC? Вы скоро увидите сами. Но, поскольку они умело разделяют многие системные ресурсы как с хостом, так и с другими контейнерами, они работают как настоящие отдельные серверы, используя только минимальное место для хранения и память.

2.3.1. Начало работы с LXC

Установить LXC на рабочую станцию Ubuntu? Легко, вот рецепт:

```
# apt update
# apt install lxc
```

А как насчет CentOS? Все в общем понятно, но нужно проделать немного бóльшую работу. Как бы там ни было, попробуйте CentOS, но я не гарантирую

успеха. Сначала нужно добавить новый репозиторий Extra Packages for Enterprise Linux (EPEL), а затем установить LXC вместе с некоторыми зависимостями:

```
# yum install epel-release
# yum install lxc lxc-templates libcap-devel \
libcgroup busybox wget bridge-utils lxc-extra libvirt
```

Символ обратной косой черты (\) может использоваться для удобного разбиения длинной команды на несколько строк в оболочке командной строки

Теперь все готово и вы можете приступить к делу. Базовые принципы работы с LXC на самом деле довольно просты. Я собираюсь показать вам три или четыре команды, которые вам понадобятся, чтобы все это заработало, а затем открою один секрет, который поразит вас, как только вы поймете, как организована LXC.

2.3.2. Создание вашего первого контейнера

Почему бы не продолжить и не создать свой первый контейнер? Ключ `-n` задает имя, которое вы будете использовать для контейнера, а `-t` указывает LXC построить контейнер по шаблону Ubuntu:

```
# lxc-create -n myContainer -t ubuntu
```

Процесс создания может занять несколько минут, но вы увидите подробный вывод и, в конце концов, уведомление об успехе

ПРИМЕЧАНИЕ

Возможно, будут ссылки на альтернативный набор команд LXC, связанных с относительно новым диспетчером контейнеров LXD. LXD все еще использует неявно инструменты LXC, но с помощью немного другого интерфейса. Например, с применением LXD предыдущая команда будет выглядеть следующим образом: lxc launch ubuntu:16.04 myContainer. Допустимы оба набора команд.

На самом деле доступно довольно много шаблонов, как вы можете видеть в коде, относящемся к каталогу `/usr/share/lxc/templates/`:

```
$ ls /usr/share/lxc/templates/
lxc-alpine     lxc-centos   lxc-fedora       lxc-oracle     lxc-sshd
lxc-altlinux   lxc-cirros   lxc-gentoo       lxc-plamo      lxc-ubuntu
lxc-archlinux  lxc-debian   lxc-openmandriva lxc-slackware  lxc-ubuntu-cloud
lxc-busybox    lxc-download lxc-opensuse     lxc-sparclinux
```

ВНИМАНИЕ

Не все эти шаблоны гарантированно работают в исходном состоянии. Некоторые предоставляются как экспериментальные или находятся *в процессе разработки*. Использование шаблона Ubuntu на хосте Ubuntu, вероятно, окажется безопасным выбором. Как я уже отмечал, с самого начала набор LXC всегда работал лучше всего на хостах Ubuntu. Ваш опыт может быть другим, если вы имеете дело с иными дистрибутивами.

Если вы решили создать, скажем, контейнер CentOS, то следует записать последние несколько строк вывода, поскольку они содержат информацию о пароле, который вы используете для входа в систему:

```
# lxc-create -n centos_lxc -t centos
[...]
The temporary root password is stored in:
        '/var/lib/lxc/centos_lxc/tmp_root_pass'
```

В этом примере я вызвал контейнер centos_lxc и использовал шаблон centos

Пароль администратора находится в каталоге, названном в честь контейнера

Вы войдете в систему, используя логин *администратора* и пароль, содержащийся в файле `tmp_root_pass`. Если, с другой стороны, ваш контейнер задействует шаблон Ubuntu, то вы будете указывать `ubuntu` как имя пользователя и пароль. Естественно, если вы планируете применять этот контейнер для чего-либо серьезного, то лучше изменить этот пароль:

```
$ passwd
Changing password for ubuntu.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Кстати, это действительно команда `passwd`, а не `password`. Предполагаю, что создатель программы `passwd` не любил печатать. Теперь введите `lxc-ls --fancy` для проверки состояния вашего контейнера:

```
# lxc-ls --fancy
NAME    STATE   AUTOSTART GROUPS IPV4   IPV6
myContainer  STOPPED 0         -      -      -
```

Контейнер создан, и теперь, пожалуй, пора начать. Как и прежде, `-n` указывает имя контейнера, который вы хотите запустить; `-d` означает «отсоединение», то есть вы не хотите автоматически запускать интерактивную сессию при запуске контейнера. В интерактивных сессиях нет ничего плохого. Но в этом случае выполнение команды `lxc-start` без `-d` будет означать, что единственный способ выйти из сессии — закрыть контейнер, а это может быть не совсем то, чего вы хотите:

```
# lxc-start -d -n myContainer
```

Список ваших контейнеров теперь должен выглядеть подобно этому:

```
# lxc-ls --fancy
NAME    STATE   AUTOSTART GROUPS IPV4       IPV6
myContainer  RUNNING 0      -    10.0.3.142  -
```

Состояние контейнера теперь RUNNING

На этот раз контейнер работает и получил IP-адрес (10.0.3.142). Вы можете использовать данный адрес для входа в систему в сеансе защищенной оболочки,

но сначала прочитайте главу 3. На данный момент вы можете запустить сеанс оболочки администратора в работающем контейнере с помощью команды `lxc-attach`:

```
# lxc-attach -n myContainer
root@myContainer:/#
```

Обратите внимание на информацию в новой командной строке

Вы можете потратить пару минут на осмотр окружения. Например, `ip addr` выведет список сетевых интерфейсов контейнера. Здесь интерфейсу `eth0` был присвоен IP-адрес 10.0.3.142, который соответствует значению IPV4, полученному при выполнении `lxc-ls --fancy` ранее:

```
root@myContainer:/# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue
➥   state UNKNOWN group default qlen 1
    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
10: eth0@if11: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
➥   state UP group default qlen 1000
    link/ether 00:16:3e:ab:11:a5 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.3.142/24 brd 10.0.3.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::216:3eff:feab:11a5/64 scope link
       valid_lft forever preferred_lft forever
```

В данном случае eth0 — это обозначение основного сетевого интерфейса контейнера

IP-адрес контейнера (10.0.3.142) и маска сети CIDR (/24)

Когда вы закончите осматривать свой новый контейнер, можете либо набрать `exit`, чтобы выйти, оставив контейнер работающим:

```
root@myContainer:/# exit
exit
```

либо завершить его работу, используя команду `shutdown -h now`. Но, прежде чем сделать это, выясним, насколько быстрыми являются контейнеры LXC. Флаг `-h`, который я добавил к команде `shutdown`, означает *остановку*. Если бы я вместо этого указал `r`, а не полностью завершил работу, то контейнер перезагрузился бы. Запустим `reboot`, а затем попробуем снова войти в систему, чтобы узнать, сколько времени потребуется, чтобы контейнер «поднялся на ноги»:

```
root@myContainer:/# shutdown -r now
# lxc-attach -n myContainer
```

Как все прошло? Могу поспорить, что к тому времени, когда вы повторно введете команду `lxc-attach`, контейнер `myContainer` уже проснется и будет готов к работе. Знаете ли вы, что нажатие клавиши ↑ в Bash приведет к вызову предыдущей команды? Использование этого трюка ускорит запрос на вход в систему. В моем случае заметной задержки не было. Контейнер закрывается и полностью перезагружается менее чем за две секунды!

ПРИМЕЧАНИЕ

Контейнеры LXC также удобны в отношении использования системных ресурсов. В отличие от моего опыта работы с виртуальными машинами VirtualBox, когда запуск одновременно трех машин уже начинает серьезно влиять на производительность моей рабочей станции с 8 Гбайт ОЗУ, я могу запускать все виды контейнеров LXC без замедления работы.

Что вы говорите? Как насчет того секрета, который я обещал раскрыть? Отлично. Я вижу, вы внимательны. Итак, вернемся к оболочке командной строки на хост-компьютере (в отличие от контейнера): вам нужно будет войти в оболочку администратора с помощью команды `sudo su`. С этого момента, пока вы не введете команду `exit`, вы будете работать в `sudo`:

```
$ sudo su
[sudo] password for username:
#
```

Теперь перейдите в каталог `/var/lib/lxc/` и просмотрите его содержимое. Вы должны увидеть каталог с именем вашего контейнера. Если у вас есть другие контейнеры в системе, у них также будут собственные каталоги:

```
# cd /var/lib/lxc
# ls
myContainer
```

Перейдите в каталог контейнера и просмотрите его содержимое. Там будет файл с именем `config` и каталог с именем `rootfs` (`fs` означает файловую систему):

```
# cd myContainer
# ls
config rootfs
```

Не стесняйтесь взглянуть на `config`: именно здесь определены основные значения среды для контейнера. Как только вам станет удобнее работать с LXC, вы, вероятно, захотите использовать этот файл для настройки поведения ваших контейнеров. Но вот что представляет собой каталог `rootfs`, который я действительно хотел бы показать вам:

```
# cd rootfs
# ls
Bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
```

Все эти подкаталоги, которые заполняют `rootfs`, вам знакомы? Все они являются частью стандарта иерархии файловых систем Linux (FHS). Это корневой (`/`) каталог контейнера, но внутри файловой системы хоста. Пока у вас есть права администратора на хосте, вы сможете просматривать эти каталоги и редактировать любые файлы, которые вам нужны, даже если контейнер не работает.

Обладая таким доступом, вы сможете делать все что угодно, но есть одна возможность, которая вполне может однажды спасти вашу (профессиональную) жизнь.

Предположим, вы в контейнере. Теперь ничто не мешает вам перемещаться по файловой системе, исправлять испорченные файлы конфигурации и возвращаться к работе. Вот скажите мне, что это не круто! Но будет еще лучше.

Это правда, что экосистема Docker получила дополнительную функциональность и стала более сложной с тех пор, как несколько лет назад эта технология вышла из-под эгиды LXC. Однако с точки зрения внутреннего устройства Docker все еще построена на базовой структурной парадигме, которую сразу опознают все, кто знаком с LXC. Это означает, что, если у вас нет проблем с самой быстро развивающейся технологией виртуализации десятилетия, у вас уже есть джокер в игре.

Резюме

- Гипервизоры, такие как VirtualBox, предоставляют среду, в которой виртуальные операционные системы могут безопасно получать доступ к аппаратным ресурсам, тогда как облегченные контейнеры совместно используют ядро программного обеспечения своего хоста.
- Менеджеры пакетов Linux, такие как APT и RPM (Yum), наблюдают за установкой и администрированием программного обеспечения из курируемых онлайн-репозиториев, используя регулярно обновляемый индекс, который отражает состояние удаленного репозитория.
- Для запуска виртуальной машины в VirtualBox необходимо определить ее виртуальную аппаратную среду, загрузить образ ОС и установить ОС на свою виртуальную машину.
- Вы можете легко клонировать, совместно использовать и администрировать виртуальные машины VirtualBox из командной строки.
- Контейнеры LXC созданы на базе предопределенных, распространяемых шаблонов.
- Данные LXC хранятся в файловой системе хоста, что упрощает администрирование контейнеров.

Ключевые понятия

- *Виртуализация* — это совместное использование несколькими процессами вычислительных, сетевых ресурсов и ресурсов хранения, что позволяет запускать каждый из них так, как если бы это был отдельный физический компьютер.
- *Гипервизор* — это программное обеспечение, которое работает на хост-компьютере и предоставляет системные ресурсы гостевому уровню, позволяя запускать и администрировать полнофункциональные гостевые виртуальные машины.
- *Контейнер* — это виртуальная машина, которая запускается поверх ядра операционной системы хоста (и совместно использует его). Контейнеры чрезвычайно легко запускать и уничтожать в зависимости от краткосрочной необходимости.

- *Динамически распределяемый виртуальный диск* в VirtualBox занимает столько места на ваших физических дисках, сколько фактически использует виртуальная машина. *Диск фиксированного размера*, напротив, занимает указанное пространство независимо от объема фактических данных.
- *Репозиторий программного обеспечения* — это место, где могут храниться цифровые ресурсы. Хранилища особенно полезны для совместной работы и распространения пакетов программного обеспечения.

Рекомендации по безопасности

- Разрешить официальному менеджеру пакетов устанавливать и поддерживать программное обеспечение в вашей системе Linux предпочтительнее, чем делать это вручную. Онлайн-хранилища намного более безопасны, а загружаемая информация надежно зашифрована.
- Всегда проверяйте контрольные хеш-суммы загруженных файлов на правильность не только потому, что пакеты могут быть повреждены во время загрузки, но и потому, что они также иногда могут быть созданы в результате атаки посредника[1].

Обзор команд

- `apt install virtualbox` использует APT для установки программного пакета из удаленного хранилища.
- `dpkg -i skypeforlinux-64.deb` напрямую устанавливает загруженный пакет Debian на компьютер с Ubuntu.
- `wget https://example.com/document-to-download` использует консольную программу `wget` для загрузки файла.
- `dnf update`, `yum update` или `apt update` синхронизирует локальный программный индекс с тем, что доступно в онлайн-хранилищах.
- `shasum ubuntu-16.04.2-server-amd64.iso` вычисляет контрольную сумму для загруженного файла, чтобы подтвердить, что она соответствует данному значению. Совпадение означает, что содержимое не было повреждено при транспортировке.
- `vboxmanage clonevm Kali-Linux-template --name newkali` использует инструмент `vboxmanage` для клонирования существующей виртуальной машины.
- `lxc-start -d -n myContainer` запускает существующий контейнер LXC.
- `ip addr` отображает информацию о каждом из сетевых интерфейсов системы (включая их IP-адреса).
- `exit` — выход из сеанса оболочки без выключения машины.

---

[1] Атака посредника, или атака «человек посередине», — вид атаки, когда злоумышленник тайно ретранслирует и при необходимости изменяет связь между двумя сторонами, которые считают, что они непосредственно общаются друг с другом. Источник: «Википедия».

Самотестирование

1. Какое качество присуще как контейнерам, так и гипервизорам:
   а) оба позволяют виртуальным машинам работать независимо от хостовой ОС;
   б) оба полагаются на ядро хоста для своих основных операций;
   в) оба обеспечивают очень легкие виртуальные машины;
   г) оба позволяют чрезвычайно эффективно использовать аппаратные ресурсы?
2. Что из нижеперечисленного не входит в обязанности менеджера пакетов Linux:
   а) синхронизация локального индекса с удаленными репозиториями;
   б) сканирование установленного программного обеспечения на наличие вредоносных программ;
   в) обновление установленного программного обеспечения;
   г) проверка того, что установлены все зависимости пакетов?
3. Какую из следующих команд вы бы использовали для непосредственной установки загруженного программного пакета в систему Ubuntu:
   а) `dpkg -i`;
   б) `dnf --install`;
   в) `apt install`;
   г) `yum -i`?
4. При создании виртуальной машины в VirtualBox какой из шагов будет первым:
   а) выбрать тип файла на жестком диске;
   б) выбрать между динамическим и фиксированным размером диска;
   в) извлечь виртуальный DVD из дисковода;
   г) настроить сетевой интерфейс?
5. Какой из следующих форматов можно использовать для образов ОС:
   а) VDI;
   б) VMI;
   в) ISO;
   г) VMDK?
6. Какую из следующих команд вы бы использовали для сохранения ВМ в файл в формате OVA:
   а) `vboxmanage export`;
   б) `vboxmanage clonevm`;
   в) `vboxmanage import`;
   г) `vboxmanage clone-ova`?

7. Какие из следующих флагов командной строки LXC запустят контейнер без автоматического открытия нового сеанса оболочки:
   а) `lxc-start -t`;
   б) `lxc-start -a`;
   в) `lxc-start -d`;
   г) `lxc-start -n`?
8. По умолчанию в каком из следующих каталогов вы найдете файловую систему контейнера:
   а) `/usr/share/lxc/`;
   б) `/etc/share/lxc/`;
   в) `/usr/lib/lxc/`;
   г) `/var/lib/lxc/`?

Ответы

1 – г; 2 – б; 3 – а; 4 – а; 5 – в; 6 – а; 7 – в; 8 – г.

3 Удаленное подключение: безопасный доступ к машинам по сети

В этой главе

- Шифрование и безопасные удаленные соединения.
- Управление системными процессами в Linux с помощью systemd .
- Особо безопасный и удобный доступ по SSH без пароля.
- Безопасное копирование файлов между удаленными хостами с помощью SCP.
- Использование удаленных графических программ через соединения SSH.

Думаю, эта глава покажется вам одной из самых интересных и веселых. Что ж, когда дело доходит до работы в мире распределенных вычислений, невозможность получить доступ к вашим серверам и удаленным ресурсам в значительной степени снижает удовольствие от работы. Поскольку большая часть рабочей нагрузки в наши дни возложена на виртуальные машины, которые мы рассмотрели в предыдущей главе, и поскольку вы не можете просто подойти к виртуальному серверу, нажать кнопку питания и войти в систему, нужен какой-то другой способ доступа. Добро пожаловать в мир Secure Shell (SSH)!

3.1. Важность шифрования

В начале был Telnet для подключения через сеть при любой скорости обмена данными. Протокол Telnet был быстрым и надежным, а в только зародившемся сетевом мире, состоящем из небольших и простых сетей, его было легко обслуживать. В те времена не имел большого значения тот факт, что инструменты Telnet отправляли свои пакеты данных без шифрования.

Как оказалось, за последние несколько десятилетий ситуация немного изменилась. Теперь этот Интернет — игрушка, в которую играют все дети и взрослые, — стал больше, чем раньше, а сетевые администраторы уже не знают друг друга по именам. По-видимому, безопасность стала предметом оживленной дискуссии. Другими словами, если вы используете Telnet для передачи личных данных, которые включают пароли и личную информацию, в виде открытого текста по незащищенным сетям, то считайте, что информация больше не приватна. Фактически любой человек в сети, использующий свободно доступное программное обеспечение для перехвата пакетов, такое как Wireshark, может легко прочитать все, что вы отправляете и получаете.

Поскольку все регулярно отсылают конфиденциальные данные через публичные сети, что делать бедному администратору? Решение есть — нужно шифровать передаваемые данные. Но что такое шифрование?

Чтобы обеспечить конфиденциальность данных, даже если они попадут в чужие руки, программное обеспечение по безопасности может использовать так называемый *ключ шифрования*. Это небольшой файл, содержащий случайную последовательность символов. Как показано на рис. 3.1, ключ можно применить как часть алгоритма шифрования, чтобы преобразовать обычные текстовые данные, которые могут быть прочитаны, в какую-то тарабарщину. По крайней мере, так это будет выглядеть до того, как ключ будет применен обратно тем же алгоритмом. Использование ключа в зашифрованной версии файла преобразует тарабарщину в исходную форму. Пока вы и ваши доверенные друзья — единственные люди, владеющие ключом, никто другой не сможет понять данные, даже если они были перехвачены.

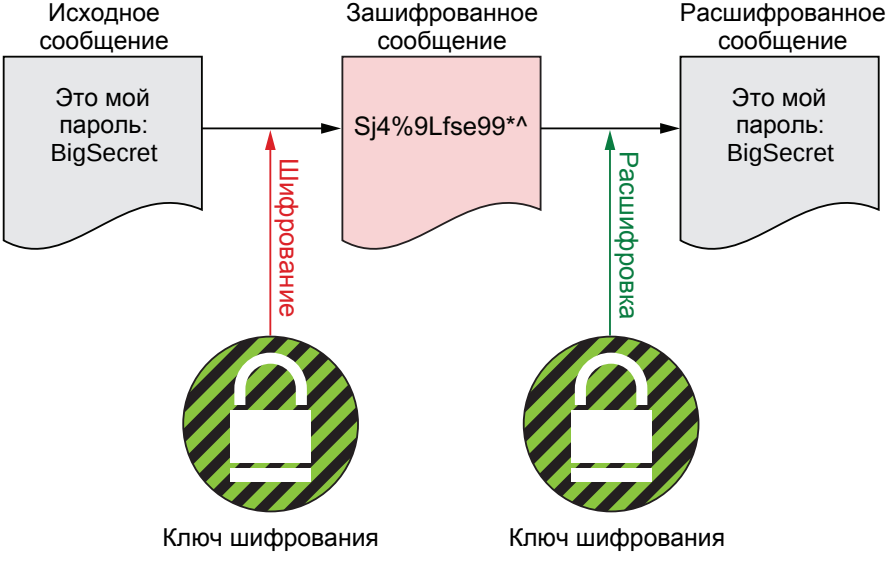


Рис. 3.1. Пара закрытых/открытых ключей для шифрования и дешифрования содержимого простого текстового сообщения. Здесь показан симметричный способ шифрования

Когда вы входите на удаленный сервер, происходит только передача между двумя компьютерами туда и обратно пакетов данных, содержащих информацию о сеансе. Хитрость *защищенного* обмена данными заключается в том, чтобы быстро зашифровать каждый из этих пакетов перед его передачей, а затем так же быстро расшифровать их на другом конце. Сетевой протокол SSH делает это так быстро и незаметно, что кто-то уже привыкший подключаться через сеансы Telnet не увидит никакой разницы.

Протокол SSH был разработан в 1990-х годах как простой способ для UNIX-подобных операционных систем безопасно шифровать данные, передаваемые при удаленном входе в систему. Реализация протокола OpenSSH сейчас настолько популярна, что корпорация Microsoft недавно сделала его частью Windows.

3.2. Начало работы с OpenSSH

В этом разделе вы узнаете, установлен ли протокол OpenSSH и активен ли он на вашей машине. При необходимости вы установите его. Поскольку тестирование рабочего состояния пакета требует понимания того, как современные дистрибутивы Linux управляют процессами, вы также погрузитесь в мир systemd. После этого вы будете использовать OpenSSH для установления сеанса связи с удаленным сервером.

Если пакет еще не установлен, выполните команду `apt install openssh-server` на компьютере Ubuntu или Debian — вам будет предоставлено все необходимое программное обеспечение. Многие версии дистрибутивов Linux поставляются с минимальной функциональностью SSH по умолчанию. Чтобы найти то, что у вас уже есть (по крайней мере на машинах с Debian/Ubuntu), вы можете использовать менеджер пакетов dpkg.

Консольный инструмент dpkg управляет пакетами и запрашивает те части программного обеспечения, которые входят в систему Advanced Package Tool (APT). Команда `dpkg` с флагом `-s` и именем пакета возвращает текущее состояние установки и статус обновления. Если пакет уже установлен (как в случае `gedit`), вывод будет выглядеть примерно так:

```
$ dpkg -s gedit
Package: gedit
Status: install ok installed
Priority: optional
Section: gnome
Installed-Size: 1732
Maintainer: Ubuntu Desktop Team <ubuntu-desktop@lists.ubuntu.com>
Architecture: amd64
Version: 3.18.3-0ubuntu4
Replaces: gedit-common (<< 3.18.1-1ubuntu1)
Depends: python3:any (>= 3.3.2-2~), libatk1.0-0 (>= 1.12.4)
[...]
```

Пример вывода dpkg -s для пакета gedit

Статус пакета

Два из множества пакетов-зависимостей

ПРИМЕЧАНИЕ

В главе 2 я говорил, что вы можете искать доступные пакеты, которые еще не установлены, используя команду apt search packagename.

Как показано на рис. 3.2, когда вы входите в систему на удаленном компьютере, ваш локальный ПК действует как клиент для удаленного сервера, поэтому вы используете пакет openssh-client. Однако операционная система (ОС) на удаленном сервере, на который вы входите, выступает в роли хоста для удаленного сеанса, поэтому должна использовать пакет openssh-server.

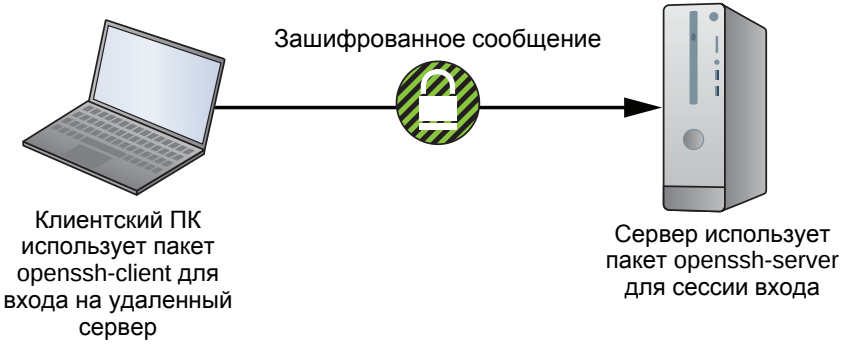


Рис. 3.2. Вход на удаленный сервер через зашифрованное соединение SSH

Вы можете запустить команду `dpkg -s openssh-client` или `dpkg -s openssh-server`, чтобы убедиться, что на вашей машине установлен правильный пакет. Поскольку они созданы для обработки сеансов с удаленных машин, в контейнерах Linux всегда будет задан полный пакет по умолчанию.

На сервере также есть все инструменты, которые вы найдете в пакете клиента. Это означает, что кто-то работающий на машине с установленным пакетом openssh-server также сможет войти через SSH на другие серверы. Поэтому, если клиентский пакет еще не установлен на вашем компьютере, установка серверного пакета обеспечит все, что может понадобиться вам впоследствии.

С другой стороны, лучшие методы обеспечения безопасности учат нас ограничивать маршруты доступа в нашу инфраструктуру только теми, которые абсолютно необходимы. Если вы не хотите входить в систему на своем компьютере или ноутбуке, установите только пакет openssh-client:

```
# apt install openssh-client
```

Тот факт, что пакет установлен правильно, не означает, что вы сможете сразу его использовать. Иногда файлы конфигурации по умолчанию отключены. В процессе работы с этой книгой вы увидите множество примеров конфигурации при установке, а чуть позже в главе вы увидите файлы конфигурации OpenSSH. Но есть и другая популярная причина, по которой программа для Linux может не работать, — она

не запущена. Вы можете ввести команду `systemctl status`, чтобы узнать, работает ли протокол SSH на вашем компьютере:

```
$ systemctl status ssh
? ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service;
       enabled; vendor preset: enabled)
   Active: active (running) since Mon 2017-05-15 12:37:18
      UTC; 4h 47min ago
Main PID: 280 (sshd)
   Tasks: 8
  Memory: 10.1M
     CPU: 1.322s
  CGroup: /system.slice/ssh.service
          |-- 280 /usr/sbin/sshd -D
          |-- 894 sshd: ubuntu [priv]
          |-- 903 sshd: ubuntu@pts/4
          |-- 904 -bash
          |-- 1612 bash
          |-- 1628 sudo systemctl status ssh
          |-- 1629 systemctl status ssh
[...]
```

SSH работает

Идентификатор процесса (PID), назначенный SSH (280 в этом примере)

Как видно из строки `Active`, все хорошо. Если же протокол не запущен, следует выполнить команду `systemctl` еще раз, но на этот раз с ключевым словом `start` вместо `status`.

Надо завершить работу протокола? Команда `systemctl stop` аккуратно выполнит эту задачу:

```
# systemctl stop ssh
```

Вы можете заставить процесс (например, SSH) автоматически загружаться при запуске системы (используйте команду `systemctl enable ssh`) или не загружаться при запуске (введите команду `systemctl disable ssh`). Следующий фрагмент кода включает SSH:

```
# systemctl enable ssh
```

Данный пример с `systemctl` кажется достаточно интересным, но он предоставляет лишь каплю информации о команде. Сейчас нас ждет протокол OpenSSH, а управление процессами я подробнее объясню в конце этой главы.

3.3. Вход на удаленный сервер по SSH

Запуск удаленных сессий намного проще, чем вы думаете. Найдите второй компьютер с запущенным пакетом openssh-server, к которому у вас есть доступ по сети. Например, вы можете запустить контейнер LXC, как это делалось в предыдущей главе.

Теперь узнайте IP-адрес этого компьютера. Если вы используете контейнер LXC, он выведет все, что вам нужно, по команде `lxc-ls --fancy`. Вот пример, показыва-

ющий один контейнер с именем test, который не запущен, и второй — с именем base, который работает по IP-адресу 10.0.3.144:

```
# lxc-ls --fancy
[sudo] password for ubuntu:
NAME    STATE   AUTOSTART GROUPS IPV4        IPV6
test    STOPPED 0         -      -           -
base    RUNNING 1         -      10.0.3.144  -
```

Команда для вывода списка контейнеров LXC и их статуса

Заголовки колонок

В качестве альтернативы, если вы вошли в систему на своем сервере, можете получить его публичный IP-адрес с помощью команды ip addr. Результат будет представлять собой довольно непонятную путаницу символов со списком всех локальных сетевых интерфейсов. Это выглядит примерно так:

```
$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue
    state UNKNOWN group default qlen 1
    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
8: eth0@if9: <BROADCAST,MULTICAST,UP,LOWER_UP>
        mtu 1500
    qdisc noqueue state UP group default qlen 1000
    link/ether 00:16:3e:ab:11:a5 brd
        ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.3.144/24 brd 10.0.3.255 scope
        global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::216:3eff:feab:11a5/64 scope link
       valid_lft forever preferred_lft forever
```

Общий сетевой интерфейс (в данном примере eth0)

Строка inet, показывающая публичный IP-адрес интерфейса

В этом случае строка inet с номером 8 — то, что нас интересует. В ней указан IP-адрес 10.0.3.144.

Имея эту информацию, для подключения вы можете запустить команду ssh с именем учетной записи на сервере, которую будете использовать для входа в систему, и IP-адресом. Если вы впервые обращаетесь к серверу с вашего компьютера, вас попросят подтвердить подлинность информации, отправленной программой OpenSSH вашего сервера, путем ввода yes. (Кстати, именно yes, а не только буквы y.) Наконец, вы введете пароль указанной вами учетной записи на сервере (в моем случае ubuntu) и тогда попадете в оболочку:

```
$ ssh ubuntu@10.0.3.144
The authenticity of host '10.0.3.144 (10.0.3.144)' can't be established.
ECDSA key fingerprint is SHA256:BPwiWLii7e+wPhFeLxJbYDjW53SgiBvZermGT9Hqck.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.3.144' (ECDSA) to the list of known hosts.
ubuntu@10.0.3.144's password:
```

Запрос на подтверждение

Введите пароль для вашей учетной записи на удаленном сервере

Получилось не так, как вы ожидали? Похоже, вам нужна еще кое-какая информация! Наиболее распространенная проблема, с которой вы, вероятно, столкнетесь, связана с сетевым подключением, так почему бы не заглянуть в главу 14? Пока используйте команду `ping`, чтобы проверить, могут ли два ваших компьютера видеть друг друга и разговаривать друг с другом. Если предположить, что вы работаете на локальном ПК и тестируете его подключение к удаленному серверу с IP-адресом 10.0.3.144, успешный пинг будет выглядеть так:

```
$ ping 10.0.3.144
PING 10.0.3.144 (10.0.3.144) 56(84) bytes of data.
64 bytes from 10.0.3.144: icmp_seq=1 ttl=64 time=0.063 ms
64 bytes from 10.0.3.144: icmp_seq=2 ttl=64 time=0.068 ms
64 bytes from 10.0.3.144: icmp_seq=3 ttl=64 time=0.072 ms
64 bytes from 10.0.3.144: icmp_seq=4 ttl=64 time=0.070 ms
```

Запись об успешном ответе на запрос ping

Вы можете остановить пинг и вернуться в командную строку, нажав сочетание клавиш Ctrl+C

Неудачный вывод будет выглядеть следующим образом. Для иллюстрации я пропинговал неиспользуемый IP-адрес:

```
$ ping 10.0.3.145
PING 10.0.3.145 (10.0.3.145) 56(84) bytes of data.
From 10.0.3.1 icmp_seq=1
   Destination Host Unreachable
From 10.0.3.1 icmp_seq=1 Destination Host Unreachable
```

Запись о неудаче ответа на пинг

3.4. Беспарольный доступ по SSH

Пароли почти никогда не используются должным образом, и это удручает. Либо они слишком короткие и/или легко угадываются, либо их просто устанавливают одинаковыми для нескольких учетных записей. И люди, кажется, постоянно забывают их. Если ваши данные защищены только паролем, то есть вероятность, что они не так уж хорошо защищены.

Именно поэтому ведущие компании отрасли, пользующиеся наибольшим доверием в области безопасности, например Amazon Web Services (AWS), по умолчанию полностью отключают аутентификацию по паролю для своих облачных экземпляров. Если вы обеспокоены риском несанкционированного доступа к вашим серверам, можете рассмотреть такую практику и последовать их примеру. Вот как выглядит этот параметр в файле `/etc/ssh/sshd_config` для экземпляра Amazon Linux на сервисе EC2:

```
# EC2 uses keys for remote access
PasswordAuthentication no
```

Файлы конфигурации OpenSSH

Как и все остальное в Linux, поведение OpenSSH на компьютере во многом зависит от настроек в его текстовых конфигурационных файлах. И, как и для большинства других программ, эти файлы конфигурации можно найти в структуре каталогов `/etc/`. В данном случае они находятся по адресу `/etc/ssh/`.

Файл конфигурации, в котором указывается, как удаленные клиенты смогут войти на ваш компьютер, называется `/etc/ssh/sshd_config`. Настройки, касающиеся того, как пользователи компьютера будут входить на удаленные хосты в качестве клиента, в свою очередь, задаются в файле `/etc/ssh/ssh_config`. Помимо ограничения людям доступа в ваши системы по SSH, настройки в этих файлах могут отвечать за управление всеми видами поведения, включая, как вы увидите чуть позже в этой главе, возможность удаленного доступа графического интерфейса к локальным программам.

Альтернативой аутентификации по паролю SSH является создание специальной пары ключей, а затем копирование открытой половины этой пары на удаленный хост, то есть компьютер, в который вы в конечном итоге хотите войти. С ключами шифрования, доступными на обоих концах соединения, OpenSSH, работающий на хосте, теперь будет иметь возможность узнать, кто вы, не запрашивая пароль. Нельзя сказать, что пароли не играют никакой роли в обеспечении безопасности инфраструктуры. Вскоре вы убедитесь, насколько важна их роль. В идеале вы должны создать так называемую *парольную фразу* и применять ее для локальной аутентификации перед использованием пары ключей.

ПРИМЕЧАНИЕ

Парольная фраза, как и пароль, представляет собой выбранную вами тайную текстовую строку. Но парольная фраза часто также включает пробелы и содержит реальные слова. Такой пароль, как 3Kjsi&*сп@PO, довольно хорош, но пароль вроде fully tired cares mound может быть даже лучше из-за его длины и относительной легкости для запоминания.

3.4.1. Генерация новой пары ключей

Для генерации новой пары ключей есть несколько способов. Но так как все хорошие системные администраторы ленивы по своей природе, я использую подход, который требует минимального количества нажатий клавиш. Непреднамеренным, но счастливым последствием этого выбора станет то, что я познакомлю вас с гораздо более сложным использованием символа «пайп» (|).

Вы начнете с создания новой пары открытого и закрытого ключей на клиентском компьютере с помощью программы ssh-keygen. Вам будет предложено ввести имя пары ключей, но, поскольку еще нет пары с именем `id_rsa`, следует просто нажать клавишу Enter и использовать название по умолчанию. Как вы видели ранее, обычно

лучше создавать парольную фразу по запросу, особенно если вы работаете на компьютере совместно с другими пользователями. Помните: если вы решите добавить парольную фразу, вам нужно будет вводить ее каждый раз при использовании ключа. Вот как все это будет происходить:

```
ubuntu@base:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_rsa.
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:1wQzEnybUSOFpYEvmbxVPZjy1uGAV6Tnn5m1w0qD5T8 ubuntu@base
The key's randomart image is:
+---[RSA 2048]----+
|        .oo**=*o  |
|         o.*B*o+  |
|        . =.Oo+.o |
|         = =oooo  |
|         S+..... .|
|         ..  + ..*|
|            . + B.|
|             . +E.|
|              . ..|
+----[SHA256]-----+
```

Отображает имя моего локального клиента, base

Местоположение и имя вашей новой пары ключей

Помогает предотвратить атаки «человек посередине»

Визуальная подсказка randomart для ключа также помогает предотвратить атаки «человек посередине»

Что ж, теперь вы являетесь счастливым обладателем новой прекрасной пары ключей на основе шифрования RSA. Продолжайте и используйте команду `ls -l` для отображения длинного списка содержимого вашего каталога `.ssh/`. Обратите внимание, что есть два файла с именем `id_rsa`, но только один из них имеет расширение `.pub`. Данный файл является публичной половиной пары, и именно его вы в конечном итоге скопируете на удаленный компьютер, с которым установите сеанс связи:

```
ubuntu@base:~$ ls -l .ssh
total 12
-rw------- 1 ubuntu ubuntu 1675 Jun  5 22:47 id_rsa
-rw-r--r-- 1 ubuntu ubuntu  393 Jun  5 22:47 id_rsa.pub
```

Какой алгоритм лучше использовать

Помимо RSA (аббревиатура, созданная из фамилий трех исследователей, которые впервые описали ее: Ron Rivest, Adi Shamir и Leonard Adleman), OpenSSH также поддерживает алгоритмы подписи ECDSA и ED25519. Между RSA, ECDSA и ED2551 9 есть незначительные технические различия, преимущество которых состоит в том, что они основаны на эллиптических кривых. Но все они считаются достаточно безопасными. При использовании ECDSA и ED25519 следует иметь в виду, что они могут еще не полностью поддерживаться некоторыми более старыми реализациями.

На сегодняшний день DSA поддерживается не всеми реализациями OpenSSH. Из-за подозрений относительно его происхождения DSA вовсе стараются не использовать.

3.4.2. Копирование открытого ключа по сети

Беспарольный доступ по SSH не будет работать, пока вы не скопируете открытый ключ на тот хост, с которым хотите связаться. Как видно из рис. 3.3, пара ключей обычно создается на клиентском компьютере. Это потому, что закрытый ключ должен быть именно таким: личным. Насколько это возможно, избегайте его копирования и скрывайте его от посторонних глаз.

После создания вы можете переместить открытый ключ в файл `.ssh/authorized_keys` на компьютере. Таким образом, программное обеспечение OpenSSH, работающее на хосте, сможет проверить подлинность криптографического сообщения, созданного с помощью закрытого ключа на клиенте. Как только сообщение окажется проверено, сеанс SSH будет разрешен.

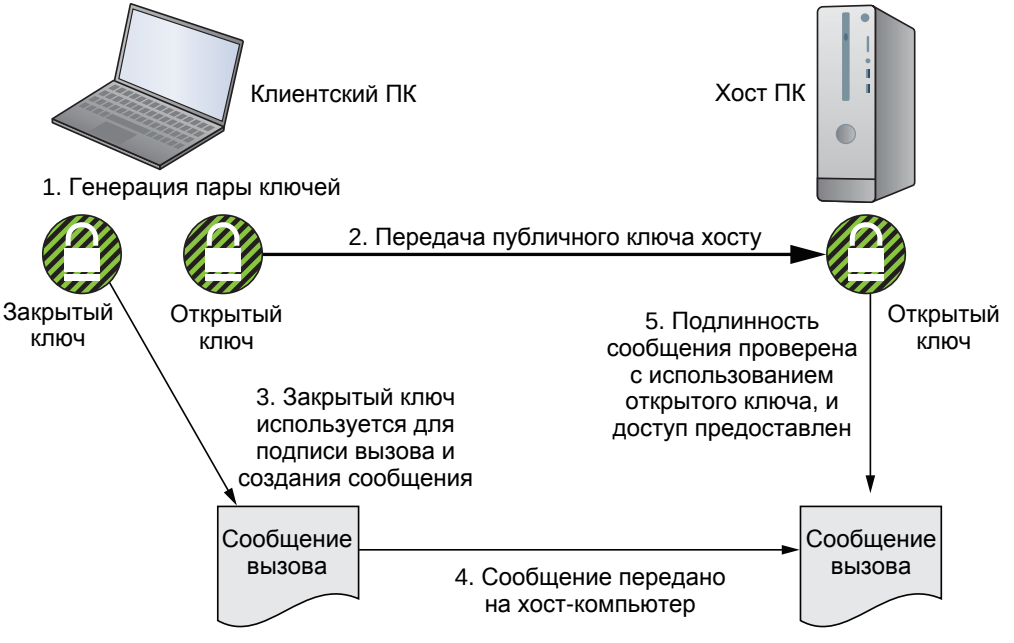


Рис. 3.3. Открытый ключ пары ключей должен быть помещен на хост-компьютер, а закрытый ключ остается на клиенте

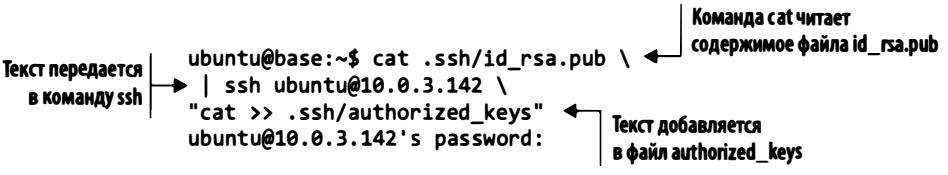
В первую очередь вам нужно выяснить, какую учетную запись на хосте вы будете использовать для входа. В моем случае это будет учетная запись **ubuntu**. Ключ необходимо скопировать в каталог с именем `.ssh/`, который находится по адресу `/home/ubuntu/`. Если его еще нет, вы должны создать его сейчас, используя команду `mkdir`.

Однако сначала я открою вам небольшой секрет: чтобы запустить одну команду, вам не нужно фактически открывать полный сеанс SSH на удаленном хосте. Вместо этого вы можете добавить свою команду к обычному синтаксису `ssh` следующим образом:

```
ubuntu@base:~$ ssh ubuntu@10.0.3.142 mkdir -p .ssh
ubuntu@10.0.3.142's password:
```

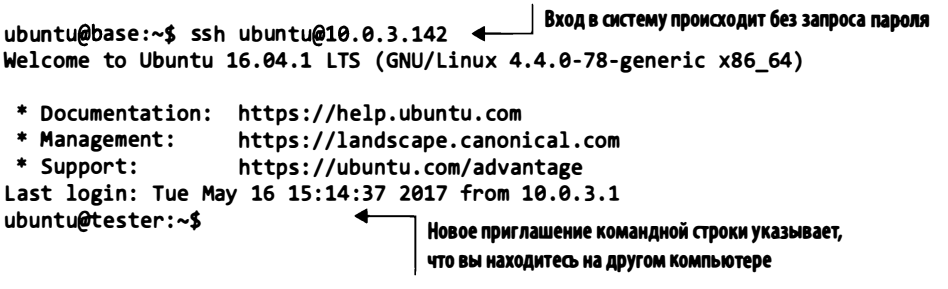
Вам все еще придется ввести пароль для удаленного хоста. Но как только вы это сделаете, у вас появится каталог `.ssh/` внутри `/home/ubuntu/` на этом хосте.

Чтобы было легче понять, я разделил следующую команду на три строки, используя символ обратной косой черты (\), который говорит Bash прочитать следующую строку как часть текущей строки, то есть объединяет строки в одну. Убедитесь, что после обратной косой черты нет символов (даже пробела), иначе это точно приведет к неприятностям:



Здесь используется команда `cat` для чтения всего текста в файле `id_rsa.pub` и сохранения его в памяти. Затем она передает данный текст через протокол SSH на удаленный хост. Наконец, она снова читает текст, на этот раз на хосте, и добавляет его в файл с именем `authorized_keys`. Если файла еще не существует, `>>` (инструмент добавления) создаст его. Если файл с таким именем уже есть, текст будет добавлен к текущему содержимому файла.

Ну вот и все. Вы готовы к работе. На этот раз, когда вы запускаете ту же старую команду `ssh`, нет необходимости вводить пароль:



3.4.3. Работа с несколькими ключами шифрования

В некоторых случаях (например, при входе в экземпляр виртуальной машины, работающей на сервисе Amazon EC2) вам придется указать, какую пару ключей использовать для данного сеанса. Это обязательно произойдет, как только вы начнете создавать набор ключей, используемых для разных хостов. Чтобы сообщить OpenSSH, какой ключ вам нужен, вы добавляете флаг `-i`, а затем указываете полное имя и местоположение файла закрытого ключа:

```
ssh -i .ssh/mykey.pem ubuntu@10.0.3.142
```

Обратите внимание на расширение файла `.pem` в этом примере. Оно означает, что ключ сохраняется в формате, который обычно используется для доступа ко всем видам виртуальных машин, включая экземпляры Amazon EC2.

3.5. Безопасное копирование файлов с помощью SCP

Уверен, вы помните, как команда `cp` копирует файлы и каталоги в файловой системе. По крайней мере теоретически нет никаких причин, почему бы это не могло так же хорошо работать для копирования файлов по сети. За исключением того, что это будет совершенно глупо — содержимое файла будет открыто любому, кто случайно окажется в сети в этот момент, или любому, кто случайно прочитает журнал с данными, переданными по сети в дальнейшем.

Даже не пытайтесь это делать! Если только не добавите букву `s` в качестве уровня защиты к имени команды `cp`. Программа SCP копирует файлы любого типа, используя протокол SSH для передачи файлов и все те же ключи, пароли и парольные фразы. Допустим, вы знаете, что на удаленном хосте, с которым вы работали ранее, уже есть каталог `.ssh/`. Вот как вы могли бы передать открытый ключ (`id_rsa.pub`) на удаленный хост, переименовав его в `authorized_keys`:

```
ubuntu@base:~$ scp .ssh/id_rsa.pub \
  ubuntu@10.0.3.142:/home/ubuntu/.ssh/authorized_keys
```

ВНИМАНИЕ

Если в данном каталоге уже есть файл authorized_keys, эта операция перезапишет его, уничтожив любое предыдущее содержимое. Вы можете копировать или сохранять файлы только в том случае, если используемые вами учетные записи имеют соответствующие разрешения. Поэтому не пытайтесь сохранить файл, скажем, в каталог /etc/ на удаленной машине, если у вашего пользователя нет привилегий администратора. Прежде всего, вход в сеанс SSH от имени администратора, как правило, запрещен согласно политикам безопасности.

Кстати, вы можете скопировать удаленные файлы на свой локальный компьютер. В этом примере выполняется копирование файла из экземпляра AWS EC2 (с вымышленным IP-адресом) в указанный локальный каталог:

```
$ scp -i mykey.pem mylogin@54.7.61.201:/home/mylogin/backup-file.tar.gz \
./backups/january/
```

Сохраняет файл в текущий каталог

В командах, которые вы использовали до этого момента, были задействованы некоторые важные инструменты. Но я должен отметить, что существует третий (и официальный) способ безопасного копирования вашего ключа на удаленный хост — можно воспользоваться специально созданной программой под названием ssh-copy-id:

```
$ ssh-copy-id -i .ssh/id_rsa.pub ubuntu@10.0.3.142
```

Автоматически копирует открытый ключ в соответствующее место на удаленном хосте

Достоинство сессий SSH состоит в том, что без графического интерфейса они быстры и эффективны. Но это может стать проблемой, если программе, которую вам нужно запустить на удаленном хосте, требуется графический интерфейс. В следующем разделе я расскажу, как решить эту проблему.

3.6. Использование удаленных графических программ через соединения SSH

Предположим, вы пытаетесь помочь удаленному пользователю, который сообщает о проблеме такого программного обеспечения для ПК, как LibreOffice. В этом случае можно запустить программу, чтобы помочь диагностировать и найти ошибку, выполнив это в графическом сеансе (с оконным менеджером Linux X) по SSH.

При всем этом не ожидайте чудес. Запуск `ssh` с флагом `-X` с использованием так называемого *перенаправления X11* позволит вам загрузить программу на хост-машине на **Рабочий стол** вашего клиента. В зависимости от таких факторов, как качество вашего сетевого соединения, ваши результаты могут не соответствовать ожиданиям. Это особенно актуально для ресурсоемких программ наподобие LibreOffice. Тем не менее данный прием всегда стоит попробовать. Из-за небольшой пропускной способности все равно можно часа два ждать доступа к офису клиента.

Еще одно замечание: не пытайтесь сделать это на сервере. В большинстве случаев операционная система, установленная на сервере или виртуальной машине (например, LXC или Docker-контейнер), графически не особо функциональна или не имеет такой функциональности вообще. Если вам точно необходимо выполнить вышеописанный прием, можете установить пакеты **Рабочего стола** и обновить его. На машине с Ubuntu это будет выглядеть так:

```
# apt update
# apt install ubuntu-desktop
```

Полагаю, вы еще не поняли, как все это работает, поэтому пришло время попрактиковаться. Прежде всего откройте файл `sshd_config` на хосте (на том, чью программу вы хотите запустить). Вам нужно убедиться, что строка `X11Forwarding` имеет значение `yes` (хотя из соображений безопасности, вероятно, стоит сразу же отключать перенаправление X11, как только оно перестает быть нужным):

```
# nano /etc/ssh/sshd_config
    X11Forwarding yes
```

Отредактируйте строку таким образом, чтобы она выглядела именно так

Аналогичная строка в файле `ssh_config` на клиентском компьютере также должна быть правильно настроена:

```
# nano /etc/ssh/ssh_config
    ForwardX11 yes
```

Отредактируйте строку таким образом, чтобы она выглядела именно так

Поскольку вы отредактировали файлы конфигурации, потребуется перезапустить SSH на обеих машинах, чтобы убедиться, что ваши изменения вступили в силу:

```
# systemctl restart ssh
```

Теперь вы готовы к работе. Чтобы начать сеанс с поддержкой графики, добавьте флаг `-X` к команде `ssh`:

```
$ ssh -X ubuntu@10.0.3.142
```

Вы увидите обычную командную строку, но теперь сможете вызвать команду, которая запустит графическую программу. Попробуйте что-нибудь простое. Вот, например, это должно сработать в системе Ubuntu:

```
$ gnome-mines
```

Удивительно! Вы успешно запускаете удаленную программу из окна на локальном Рабочем столе.

Протокол OpenSSH имеет гораздо больше возможностей, чем те основные функции, которые вы уже видели. После того как вы настроили работающее соединение SSH, можете использовать все возможные приемы. Попробуйте подключить локальную файловую систему или каталог на удаленном компьютере, чтобы удаленные пользователи могли беспрепятственно получать доступ к вашим файлам. Или с помощью возможностей SSH-туннелирования используйте переадресацию портов, чтобы разрешить безопасное закрытое применение удаленных HTTP-сервисов.

3.7. Управление процессами в Linux

Как и обещал, теперь я собираюсь вернуться к управлению процессами в Linux, чтобы вы могли понять, как работают, например, такие программы, как OpenSSH. Зная, как это происходит, в дальнейшем вы сможете намного более эффективно выполнять общее администрирование и устранять неполадки. Но если у вас нет желания сейчас погружаться в такую сложную тему, можете смело пропустить оставшуюся часть этой главы. У вас не должно возникнуть никаких проблем при чтении остальной части книги.

Что такое systemctl и для чего он нужен? Чтобы правильно ответить на эти вопросы, вам придется немного задуматься о том, как Linux управляет системными процессами в целом. И поскольку всегда интересно узнать что-то новое, вы также познакомитесь с парочкой инструментов отслеживания процессов, которые упростят понимание того, как все работает.

Я уверен, вы знаете, что *программное обеспечение* — это программный код, содержащий инструкции по управлению компьютерным оборудованием от имени пользователей. *Процесс* — экземпляр работающей программы. *Операционная система* — инструмент для организации и управления этими экземплярами/процессами для эффективного использования аппаратных ресурсов компьютера.

Организация процессов и управление ими для сложной многопроцессорной, многопользовательской операционной среды — непростая задача. Чтобы все это

заработало, вам понадобится какой-то регулировщик, который будет жестко контролировать множество составляющих (рис. 3.4). Позвольте мне представить вам systemctl.

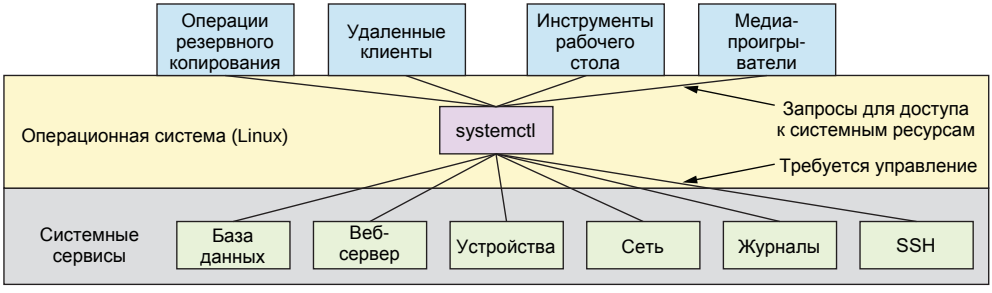


Рис. 3.4. Доступностью и временем ответа многих системных служб управляет менеджер процессов systemd — systemctl

3.7.1. Просмотр процессов с помощью команды ps

Давайте достанем электронный микроскоп и посмотрим, сможем ли мы обнаружить процесс в его естественной среде обитания. Введите следующую команду в оболочке командной строки. Она запустится (`sleep`) в фоновом режиме (`&`) на десять секунд, не выполняя никаких действий, а затем прекратит работу. Пока она запущена, введите `ps`:

```
    $ for i in {1..10}; do sleep 1; done &
[1] 19829
    $ ps
      PID TTY          TIME CMD
    19522 pts/17   00:00:00 bash
    19829 pts/17   00:00:00 bash
    19832 pts/17   00:00:00 sleep
    19833 pts/17   00:00:00 ps
```

Идентификатор команды, работающей в фоновом режиме

Процесс сна (sleep), запущенный первоначальной командой

Команда ps для вывода списка запущенных процессов

Вы получите сведения о двух запущенных процессах, порожденных этой командой, вместе с их PID: `19829` и `19832` в моем случае. Если вы снова запустите `ps` через десять секунд, вы увидите, что эти два процесса больше не существуют. Вы также должны получить отчет об успешном завершении команды `sleep`:

```
$ ps
  PID TTY          TIME CMD
19522 pts/17   00:00:00 bash
20549 pts/17   00:00:00 ps
[1]+  Done                    for i in {1..10};
do
    sleep 1;
done
```

Если вы наберете просто `ps` и запустите ее, то, вероятно, получите только два результата. Будет указан процесс, называемый `bash`, который представляет интерпретатор команд Bash, используемый вашим текущим сеансом оболочки, и самая последняя команда (конечно, это была `ps`). Но, взглянув на PID, назначенный для `bash` (`7447` в следующем примере), вы поймете, что в вашей системе уже работает множество других процессов. Они порождены родительскими оболочками, восходя к самому процессу `init`:

```
$ ps
  PID TTY          TIME CMD
7447 pts/3    00:00:00 bash
8041 pts/3    00:00:00 ps
```

На машине с Ubuntu первый процесс, который начинает работать и запускает все остальное при загрузке компьютера с Linux, называется `init`. Как вы вскоре узнаете, это имя может вводить в заблуждение, поэтому первый процесс в CentOS носит другое имя. Вы можете убедиться, что `init` — начало всего, выполнив следующую команду `ps` в точности так, как она показана здесь. Далее я объясню ее подробнее:

```
$ ps -ef | grep init
root           1      0  0 12:36 ?        00:00:00 /sbin/init
ubuntu      1406    904  0 16:26 pts/4    00:00:00 grep --color=auto init
```

Файл, ответственный за процесс 1

В крайней правой колонке в выводе (`/sbin/init` в первой строке) указано местоположение и имя файла, откуда начался сам процесс. В данном случае это файл с именем `init`, который находится в каталоге `/sbin/`. Крайняя левая колонка в первой строке содержит слово `root`, что говорит о том, что владельцем этого процесса является администратор (root-пользователь). Дополнительная информация, которая может быть нам интересна в текущий момент, — это номер `1`, который является PID процесса `init`. Единственный способ получить PID = `1` — начать работу раньше всех.

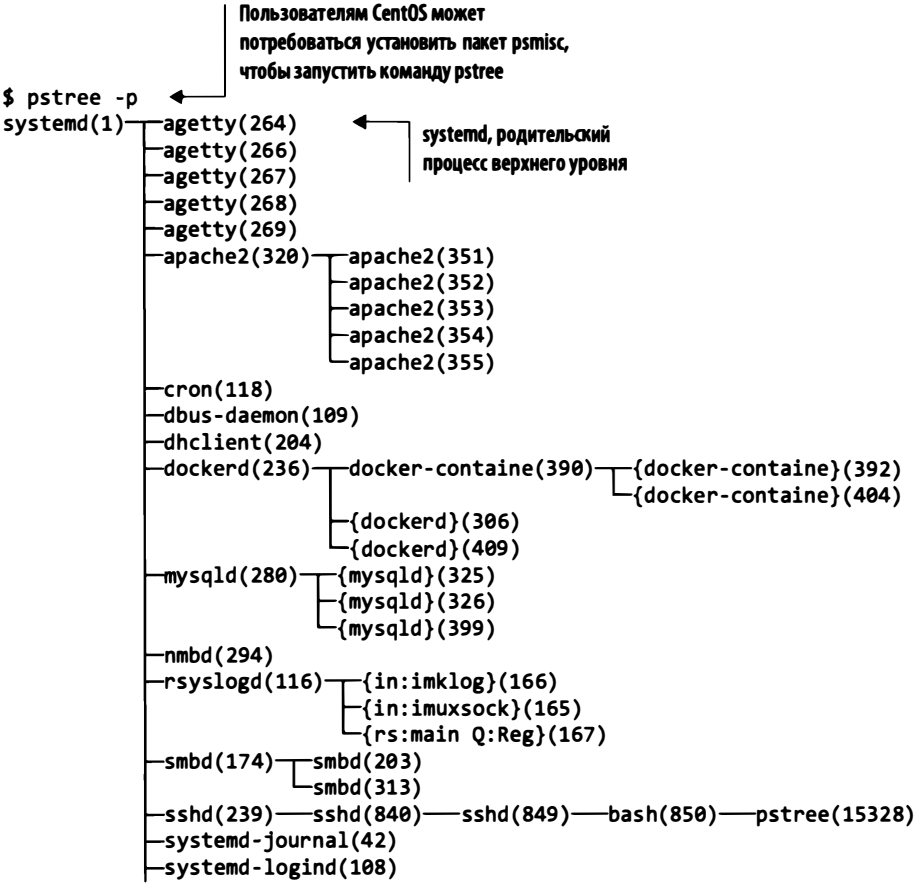
Прежде чем двигаться дальше, стоит потратить немного больше времени на команду `ps`. Как вы видели, `ps` отображает информацию об активных процессах. Часто важно иметь доступ к информации о процессе для правильного планирования и устранения неполадок в работе системы. Команда `ps` — частый помощник в этом деле.

Добавление аргумента `-e` к команде `ps` позволит вывести информацию не только о процессах, запущенных в вашей текущей дочерней оболочке, но и обо всех процессах всех родительских оболочек вплоть до `init`.

ПРИМЕЧАНИЕ

Родительская оболочка — это среда оболочки, из которой впоследствии могут быть запущены новые (дочерние) оболочки и через которую выполняются программы. Вы можете рассматривать ваш Рабочий стол с графическим интерфейсом как оболочку, а терминал, который вы открываете, чтобы получить командную строку, — как дочерний элемент. Оболочка верхнего уровня (прародитель) — та, которая запускается первой при загрузке Linux.

Если вы хотите визуализировать родительские и дочерние оболочки/процессы, используйте команду `pstree` (добавив аргумент `-p` для отображения PID каждого процесса). Обратите внимание, что первым процессом (с назначенным PID = `1`) является `systemd`. В более старых версиях Linux (например, Ubuntu 14.04 и раньше) он назывался `init`:



Попробуйте выполнить все эти команды на своем компьютере. Даже в простой системе вы, вероятно, увидите десятки процессов; на загруженном же ПК или сервере легко могут быть сотни и даже тысячи процессов.

3.7.2. Работа с systemd

В файле `/sbin/init`, который вы только что видели, есть кое-что интересное: `file` — старинная UNIX-программа, которая предоставляет внутреннюю информацию о файле. Если вы запустите команду `file` с `/sbin/init` в качестве аргумента, то увидите, что файл `init` на самом деле не программа, а *символическая ссылка* на

программу с именем systemd. Больше о символических ссылках мы поговорим в главе 12, а здесь предлагаю вам познакомиться с systemd:

```
$ file /sbin/init
/sbin/init: symbolic link to /lib/systemd/systemd
```

Потребовались годы, но теперь почти во всех дистрибутивах Linux используется один и тот же диспетчер процессов: systemd. Он полностью заменяет процесс `init`, который долгое время был самым первым процессом, запускаемым при загрузке всех операционных систем на основе UNIX. Под *«полной заменой»* я подразумеваю, что, даже если способ достижения цели может быть совершенно другим, для обычного наблюдателя systemd работает так же, как всегда работал `init`. Вот почему файл `/sbin/init` теперь является не чем иным, как ссылкой на программу systemd.

Данная информация чисто теоретическая, так как вы, вероятно, никогда не вызовете саму программу systemd по имени: ни напрямую, ни по ее ссылке `/sbin/init`. Это потому, что, как вы уже видели, ключевые задачи администрирования для systemd выполняются с помощью systemctl.

Технически основная задача systemd — контролировать то, как отдельные процессы рождаются, живут своей жизнью, а затем умирают. Команда `systemctl`, которую вы использовали ранее, является инструментом выбора для этих задач. При этом довольно удивительно, что разработчики systemd расширили функциональность далеко за рамки традиционного управления процессами, чтобы взять под контроль различные системные службы. Теперь systemd включает такие инструменты, как менеджер журналов (journald), менеджер сети (networkd) и менеджер устройств (как вы уже догадались, udevd). Любопытно? Здесь *d* означает *демон*, фоновый системный процесс.

В этой книге мы еще поговорим по крайней мере о некоторых из этих системных инструментов. В следующей же главе мы рассмотрим, как создавать резервные копии файловых систем и архивов и управлять ими.

Резюме

- Шифрование соединений очень важно для всех сетевых взаимодействий, а SSH в значительной степени является отраслевым стандартом.
- Вы можете включить беспарольный доступ по SSH, скопировав открытый ключ из пары ключей.
- Пакет OpenSSH также обеспечивает безопасное копирование файлов и возможность удаленных графических сеансов.
- В большинстве современных дистрибутивов Linux процессы управляются программой systemd с помощью инструмента systemctl.
- Вы можете передавать данные между командами, используя символ | («пайп»), и фильтровать потоковые данные с помощью утилиты grep.

Ключевые понятия

- *Пароль* — это строка из обычных символов, в то время как *парольная фраза* может содержать пробелы и знаки препинания.
- *RSA* — популярный алгоритм шифрования.
- *Перенаправление X11* позволяет запускать графические программы через удаленное соединение.
- *Процесс* Linux — это все текущие действия, связанные с одной запущенной программой.
- *Оболочка* — консольная среда, которая предоставляет интерпретатор командной строки (например, Bash), позволяющий пользователю выполнять команды. Когда вы работаете с настольного ПК или ноутбука под управлением операционной системы Linux, вы обычно получаете доступ к оболочке, открывая консольную программу (например, GNOME Terminal).
- *Родительская оболочка* — это начальная среда. Из нее впоследствии могут быть запущены новые дочерние оболочки, через которые запускаются программы. Оболочка также является процессом.

Рекомендации по безопасности

- Всегда шифруйте сеансы удаленного входа в систему через общедоступную сеть.
- Не полагайтесь только на пароли — они подвержены ошибкам.
- Беспарольные сеансы SSH на основе ключей предпочтительнее простых паролей.
- Никогда не передавайте через публичные сети файлы в виде простого текста.

Обзор команд

- `dpkg -s openssh-client` проверяет состояние пакета программного обеспечения на основе APT.
- `systemctl status ssh` проверяет состояние системного процесса (через `systemd`).
- `systemctl start ssh` запускает службу.
- `ip addr` показывает все сетевые интерфейсы на компьютере.
- `ssh-keygen` генерирует новую пару ключей SSH.
- `$ cat .ssh/id_rsa.pub | ssh ubuntu@10.0.3.142 "cat >> .ssh/authorized_keys"` считывает локальный ключ и копирует его на удаленный компьютер.
- `ssh-copy-id -i .ssh/id_rsa.pub ubuntu@10.0.3.142` безопасно копирует ключи шифрования (рекомендовано и является стандартом).
- `ssh -i .ssh/mykey.pem ubuntu@10.0.3.142` определяет конкретную пару ключей.
- `scp myfile ubuntu@10.0.3.142:/home/ubuntu/myfile` безопасно копирует локальный файл на удаленный компьютер.

- `ssh -X ubuntu@10.0.3.142` позволяет войти на удаленный хост для запуска графического сеанса.
- `ps -ef | grep init` отображает все запущенные в данный момент системные процессы и отфильтровывает результаты, содержащие строку `init`.
- `pstree -p` отображает все запущенные в данный момент системные процессы в виде дерева.

Самотестирование

1. Целью ключа шифрования является:
   а) установление безопасного сетевого соединения;
   б) шифрование и дешифровка пакетов данных;
   в) завуалирование конфиденциальных данных в пути;
   г) обеспечение надежности передачи данных.
2. Какой командой вы можете проверить состояние службы:
   а) `dpkg -s <имя_службы>`;
   б) `systemd status <имя_службы>`;
   в) `systemctl status <имя_службы>`;
   г) `systemctl <имя_службы> status`?
3. Какие из этих пакетов должны быть установлены для того, чтобы хост-сервер мог принимать удаленные входы по SSH:
   а) openssh-server;
   б) ssh-server;
   в) openssh-client;
   г) ssh-client?
4. В дистрибутиве Linux, использующем systemd, какая из этих программ выполняет работу `init`:
   а) `/lib/systemd/systemd`;
   б) `/bin/systemd`;
   в) `/sbin/init`;
   г) `/bin/init`?
5. Какая из следующих служб *не* является systemd:
   а) networkd;
   б) journald;
   в) processd;
   г) udevd?

6. Где должны быть размещены ключи для SSH-соединений без пароля:
   а) открытый и закрытый ключи на хосте, закрытый ключ на клиенте;
   б) открытый и закрытый ключи на хосте, открытый ключ на клиенте;
   в) закрытый ключ на хосте, открытый ключ на клиенте;
   г) открытый ключ на хосте, закрытый ключ на клиенте?
7. Какова цель парольной фразы в сессиях SSH:
   а) подтвердить свою идентичность в удаленной программе OpenSSH;
   б) подтвердить свою идентичность в локальной программе OpenSSH;
   в) определить, какую пару ключей вы хотите использовать;
   г) аутентифицировать статус пары ключей?
8. Что из нижеперечисленного скопирует удаленный файл в текущий каталог на вашем локальном компьютере (при условии, что удаленный каталог и файл существуют):
   а) `scp mylogin@10.0.3.142:/home/`*`ваш_логин/имя_файла`*;
   б) `scp mylogin@10.0.3.142/home/`*`ваш_логин/имя_файла`*;
   в) `scp mylogin@10.0.3.142:/home/`*`ваш_логин/имя_файла`*;
   г) `scp mylogin@10.0.3.142:/home/`*`ваш_логин/имя_файла`*`./home/myname/Documents`?

Ответы
1 — б; 2 — в; 3 — а; 4 — а; 5 — в; 6 — г; 7 — б; 8 — а.

4 Управление архивами: создание резервных копий или копирование целых файловых систем

В этой главе

- Зачем, что и где архивировать.
- Архивирование файлов и файловых систем с помощью инструмента `tar`.
- Поиск системных файлов.
- Защита файлов путем установки разрешений для объекта и владельца.
- Архивирование целых разделов с помощью инструмента `dd`.
- Синхронизация удаленных архивов с помощью инструмента `rsync`.

Из первых глав книги вы узнали много нового о том, как безопасно и эффективно работать в среде Linux. Вы также научились создавать базовые рабочие среды, используя чудеса виртуализации. С этого момента я сосредоточусь на создании и поддержании элементов инфраструктуры, которые вам понадобятся в реальном мире.

Строить IT-инфраструктуру без хорошего протокола резервного копирования все равно, что закладывать свой дом, инвестируя в бизнес своего шурина, «который может прогореть». Всегда есть вероятность, что дело плохо закончится. Но прежде, чем вы сможете правильно создавать резервные копии файловых систем и разделов, вам необходимо точно понять, как работают сами файловые системы и разделы. Какие инструменты предусмотрены? Когда каждый из них лучше задействовать и как снова восстановить все при возникновении проблем? Приступим.

4.1. Зачем архивировать

Прежде чем мы перейдем к вопросу *«зачем»*, разберемся, *что* такое архив. Это всего лишь единый файл, содержащий группу других объектов: файлы, каталоги или и то и другое. Объединение объектов в одном файле (рис. 4.1) иногда облегчает перемещение, совместное использование или хранение нескольких объектов, которые в противном случае могли бы быть громоздкими и неорганизованными.

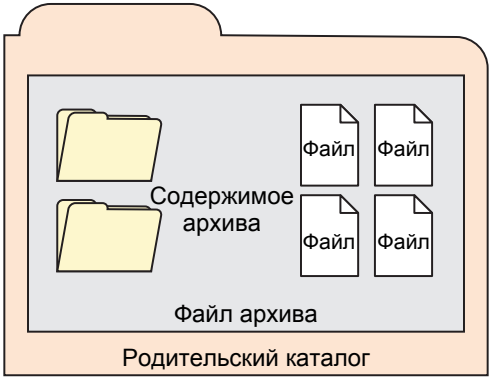


Рис. 4.1. Файлы и каталоги можно объединить в архивный файл и сохранить в файловой системе

Представьте себе, что вы пытаетесь скопировать несколько тысяч файлов, разбросанных по дюжине каталогов и подкаталогов, чтобы ваш коллега по сети тоже мог их увидеть. Конечно, с помощью надлежащих консольных инструментов все можно сделать. (Помните программу `cp` из главы 1? И ключ `-r`?) Но удостовериться, что вы копируете только те файлы, которые вам нужны, и ничего случайно не пропускаете, может быть проблематично. Конечно, вам все равно нужно будет указать все эти файлы и/или каталоги хотя бы один раз при создании архива. Но как только вы соберете все в один архивный файл, это будет намного легче отследить. Вот что такое архивы.

Архивы бывают разные. Какой выбрать? Это зависит от типа файлов, которые вы хотите архивировать, и от того, что вы планируете с ними делать. Возможно, вам нужно делать копии каталогов и их содержимого, чтобы легко обмениваться ими по сети или восстановить при необходимости. Для этого лучше всего подойдет программа `tar`. Однако если вам нужна точная копия раздела или даже всего жесткого диска, то понадобится инструмент `dd`. А если вам требуется решение для регулярного резервного копирования системы, попробуйте инструмент `rsync`.

В остальной части этой главы мы поговорим о том, как использовать эти три инструмента, и, что более важно, проанализируем реальные задачи, которые они могут решить для вас. Попутно немного обсудим, как защитить права доступа и атрибуты владения для файлов в архиве при их перемещении по жизненному циклу архива. Наконец, мы посмотрим, почему Linux в первую очередь использует права доступа к файлам и право владения файлами.

4.1.1. Сжатие

Еще одно замечание, прежде чем мы начнем. Хотя архивирование и сжатие часто используются вместе, не путайте эти понятия. *Сжатие*, как показано на рис. 4.2, представляет собой процесс, когда к файлу или архиву применяется умный алгоритм, чтобы уменьшить объем занимаемого дискового пространства. Конечно, сжатые файлы не читаются, поэтому алгоритм также нужно применить в обратном направлении, чтобы распаковать их.

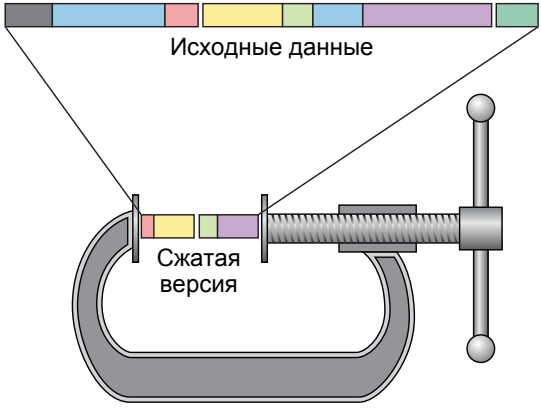


Рис. 4.2. Сжатие объектов путем устранения статистической избыточности и/или удаления менее важных частей файла

Как вы скоро увидите, сжать архив TAR просто, что особенно полезно, если вы планируете передавать большие архивы по сети. Сжатие может значительно уменьшить время передачи.

4.1.2. Архивы: некоторые важные соображения

Две основные цели, которые мы преследуем при архивировании, — создание надежных образов файловой системы и создание эффективных резервных копий данных.

Образы

Что такое образ? Помните файлы ISO, которые вы использовали для установки Linux на виртуальной машине в главе 2? Эти файлы представляли собой образы целых операционных систем, специально организованных для того, чтобы упростить копирование включенных файлов на нужный компьютер.

Образы также могут создаваться со всей или частью работающей операционной системы (ОС), поэтому вы можете копировать и клонировать содержимое на другой компьютер. Это фактически делает вторую систему (копию) точным клоном первой в ее текущем состоянии. Я часто делал образы в попытке спасти сложную систему на неисправном жестком диске, чтобы заново не устанавливать все с нуля

на новом диске. Это также очень удобно, когда вы хотите быстро задать идентичные настройки системы для нескольких пользователей, например для рабочих станций учеников в классе.

ПРИМЕЧАНИЕ

Даже не думайте о том, чтобы провернуть что-то подобное с Windows. В сущности, архитектура реестра Windows делает невозможным отделение установленной ОС от ее исходного оборудования.

Хотя в оставшейся части этой главы мы будем говорить о резервных копиях, а не об образах, не беспокойтесь: инструменты, которыми мы воспользуемся для создания и восстановления образов, практически одинаковы, так что в любом случае все будет в порядке.

Резервное копирование данных

Резервные копии должны занимать особое место в вашей работе. На самом деле если вы никогда не беспокоитесь о состоянии своих данных, то либо вы дзен-мастер, либо просто неправильно делаете свою работу. Может случиться так много страшных вещей.

- Аппаратное обеспечение может и будет отказывать. И скорее всего, это произойдет прямо перед тем, как вы решите сделать эту большую резервную копию. В самом деле.
- Толстые пальцы (я имею в виду неуклюжих людей) и клавиатуры могут испортить файлы конфигурации, полностью блокируя зашифрованную систему. Наличие резервной копии позволит спасти вашу работу и, возможно, вашу карьеру.
- Данные, которые небезопасно хранятся у провайдеров облачных инфраструктур, таких как Amazon Web Services (AWS), можно внезапно и непредсказуемо потерять. В 2014 году это произошло с компанией под названием Code Spaces. Неправильно настроенная консоль учетной записи AWS компании была взломана, и злоумышленники удалили бóльшую часть ее данных. Как восстанавливалась Code Spaces? Ну, когда вы в последний раз слышали что-нибудь о Code Spaces?
- Возможно, самое страшное из всего: вы можете стать жертвой вымогателей, которые зашифруют или заблокируют все ваши файлы, если вы не заплатите большой выкуп. Получили надежную и свежую резервную копию? Не стесняйтесь говорить злоумышленникам именно то, что вы о них думаете.

Прежде чем двигаться дальше, я должен упомянуть, что непроверенные резервные копии данных на самом деле могут не работать. Есть основания полагать, что почти половина из них неработоспособна. В чем проблема? Многое может пойти не так: на вашем устройстве резервного копирования могут быть неполадки, файл архива может быть поврежден или, возможно, сама утилита резервного копирования неправильно сохранила все ваши файлы.

Ведение и мониторинг журнальных сообщений поможет вам обнаружить проблемы, но единственный способ быть уверенными в резервном копировании — запустить пробное восстановление на соответствующем оборудовании. Это потребует энергии, времени и денег. Но такая процедура, безусловно, нужна. Кажется, все лучшие системные администраторы, которых я знаю, сходятся во мнении, что «паранойя — это только начало».

4.2. Что архивировать

Если у вас не очень много файлов для резервного копирования и они не очень велики, можете просто скопировать их в резервное хранилище как есть. Используйте что-то наподобие программы SCP, с которой вы познакомились в главе 3. В следующем примере программа SCP копирует содержимое моего открытого ключа шифрования в файл с именем `authorized_keys` на удаленной машине:

```
ubuntu@base:~$ scp .ssh/id_rsa.pub \
  ubuntu@10.0.3.142:/home/ubuntu/.ssh/authorized_keys
```

Перезаписывает текущее содержимое файла authorized_keys в удаленном хранилище

Но если вы хотите выполнить резервное копирование большого количества файлов, распределенных по нескольким каталогам (например, сложный проект с исходными кодами) или даже целым разделам (например, операционной системы, в которой вы сейчас работаете), вам понадобится что-то более мощное.

Мы немного говорили о разделах диска и псевдофайлах в главе 1, но, если вы хотите разработать какую-то интеллектуальную политику резервного копирования, вам нужно досконально разобраться, что они собой представляют. Предположим, вы планируете создать резервную копию раздела, содержащего большую бухгалтерскую базу данных вашей компании. Вам, вероятно, нужно знать, сколько места занимает этот раздел и как его найти.

Начнем с команды `df`, отображающей каждый раздел, который в настоящее время смонтирован в системе Linux, а также показывающей использование им диска и расположение в файловой системе. Добавление флага `-h` преобразует размеры разделов в удобочитаемые форматы, например гигабайты или мегабайты вместо байтов:

```
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda2       910G  178G  686G  21% /
none            492K     0  492K   0% /dev
tmpfs           3.6G     0  3.6G   0% /dev/shm
tmpfs           3.6G  8.4M  3.6G   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           3.6G     0  3.6G   0% /sys/fs/cgroup
```

Корневой раздел: единственный нормальный раздел в этой системе

Обратите внимание на 0 байт на диске. Это обычно указывает на псевдофайловую систему

Каталог /run содержит файлы с данными, сгенерированными во время загрузки

Первый раздел обозначен как `/dev/sda2`, что означает, что это второй раздел на устройстве хранения A (Storage Device A) и что он представлен как системный ресурс через каталог псевдофайловой системы `/dev/`. В данном случае это основной раздел ОС. Все устройства, связанные с системой, будут представлены файлом в каталоге `/dev/`. (Раздел, используемый вашим бухгалтерским программным обеспечением, появится где-то в этом списке, возможно, обозначенный как `/dev/sdb1`.)

ПРИМЕЧАНИЕ

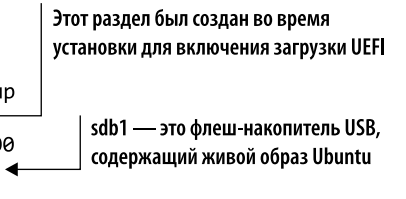
При запуске программы df в контейнере LXC отображаются разделы, связанные с хостом LXC.

Важно различать *реальную* и *псевдо*файловую системы (системы, файлы которых на самом деле не сохраняются на диске, а находятся в энергозависимой памяти и удаляются при выключении компьютера). В конце концов, нет смысла создавать резервные копии файлов, представляющих кратковременный аппаратный профиль, ведь в любом случае они будут автоматически заменены ОС всякий раз, когда будет загружена реальная файловая система.

Определить, какие разделы используются для псевдофайлов, довольно просто: если файл относится к временному хранилищу tmpfs, а число байтов, указанное в столбце `Used`, равно `0`, то, скорее всего, вы видите временную, а не нормальную файловую систему.

Кстати, в данном случае инструмент `df` был запущен в контейнере LXC, поэтому существует только один реальный раздел, `/`. Посмотрим, что программа `df` покажет при запуске на реальном компьютере:

```
df -h
Filesystem     Size  Used Avail Use% Mounted on
udev           3.5G     0  3.5G   0% /dev
tmpfs          724M  1.5M  722M   1% /run
/dev/sda2      910G  178G  686G  21% /
tmpfs          3.6G  549M  3.0G  16% /dev/shm
tmpfs          5.0M  4.0K  5.0M   1% /run/lock
tmpfs          3.6G     0  3.6G   0% /sys/fs/cgroup
/dev/sda1      511M  3.4M  508M   1% /boot/efi
tmpfs          724M   92K  724M   1% /run/user/1000
/dev/sdb1      1.5G  1.5G     0 100% /mnt/UB-16
```



Обратите внимание на раздел `/dev/sda1`, смонтированный в `/boot/efi`. Он был создан во время первоначальной установки Linux, чтобы разрешить загрузку системы, управляемую микропрограммой UEFI. Стандарт UEFI теперь в значительной степени заменил старый интерфейс BIOS, который использовался для инициализации оборудования во время загрузки системы. Программное обеспечение, установленное в этом разделе, позволяет интегрировать UEFI с системой Linux, а `/dev/sdb1` — это флеш-накопитель USB, который был подключен к USB-порту моей машины.

Когда вы работаете с серверами в производственной среде, вы часто видите отдельные разделы для таких каталогов, как `/var/` и `/usr/`. Так делается для того, чтобы упростить поддержание целостности и безопасности конфиденциальных данных или защитить остальную часть системы от переполнения из-за разрастания файлов, скажем журналов (логов) в `/var/log/`. Какой бы ни была причина, вам придется принимать обоснованные решения о том, что нужно резервировать, а что — нет.

Иногда вы можете встретить каталог `/boot/` с собственным разделом. Я думаю, что это плохая идея, и у меня есть причины так считать. Проблема в том, что новые образы ядра записываются в `/boot/`, и, когда ваша система обновляется до новых выпусков ядра Linux, дисковое пространство, необходимое для хранения всех этих образов, увеличивается. Если обычно вы назначаете только 500 Мбайт загрузочному разделу, у вас будет полгода или около того, прежде чем он заполнится, — после этого обновления будут невозможны. Вероятно, вы не сможете полностью загрузиться в Linux, пока вручную не удалите некоторые старые файлы, а затем обновите меню GRUB. Если это не очень сложно, сохраните каталог `/boot/` в самом большом разделе.

4.3. Где создавать резервную копию

С точки зрения операционной системы не имеет значения, где вы храните свои архивы. Вы можете выбирать между устаревшими ленточными накопителями, USB-накопителями SATA, сетевым хранилищем (NAS), сетями хранения данных (SAN) или облачным хранилищем. Подробнее об этом читайте в моей книге *Learn Amazon Web Services in a Month of Lunches* (Manning, 2017).

Какой бы путь вы ни выбрали, обязательно внимательно следите за передовыми методиками. В любом случае все ваши резервные копии должны быть:

- *надежными* — используйте только те носители данных, которые с достаточной вероятностью сохранят свою целостность за планируемое вами время работы;
- *протестированными* — проведите как можно больше тестов на восстановление архива в смоделированных производственных средах;
- *имеющими возможность замены и сохранения архивов* — сохраните хотя бы несколько архивов старше текущей резервной копии на случай, если последняя из них каким-то образом выйдет из строя;
- *распределенными* — убедитесь, что по крайней мере некоторые из ваших архивов хранятся в физически удаленном месте. Тогда в случае пожара или другого стихийного бедствия ваши данные не исчезнут вместе с офисом;
- *безопасными* — никогда не доверяйте свои данные небезопасным сетям или хранилищам в процессе архивирования;
- *совместимыми* — всегда соблюдайте все соответствующие нормативные и отраслевые стандарты;

- *актуальными* — нет смысла хранить архивы, которые на несколько недель или месяцев отстают от текущей версии;
- *сценарии* — в таких задачах никогда не полагайтесь на человека. Автоматизируйте работу (прочитайте главу 5).

4.4. Архивирование файлов и файловых систем с помощью инструмента tar

Чтобы успешно создать свой архив, нужно сделать следующее.

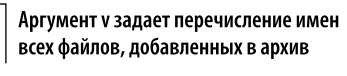
1. Найдите и обозначьте файлы, которые хотите архивировать.
2. Определите место на диске, которое хотите использовать для архива.
3. Добавьте файлы в архив и сохраните его в своем хранилище.

Хотите совместить все три шага в одном? Используйте инструмент `tar`. Назовите меня безнадежным романтиком, но я восхищаюсь хорошо продуманной командой `tar`: единственная, тщательно сбалансированная строка такого многофункционального кода — это поистине прекрасно.

4.4.1. Примеры простого архива и сжатия

В этом примере копируются все файлы и каталоги внутри текущего каталога и создается архивный файл, который я назвал `archivename.tar`. Здесь я использую три аргумента после имени команды `tar`: `c` сообщает `tar` о создании нового архива, `v` гарантирует подробный вывод на экран, а `f` указывает на имя файла, которое я хотел бы получить:

```
$ tar cvf archivename.tar *
file1
file2
file3
```



ПРИМЕЧАНИЕ

Команда tar никогда не перемещает и не удаляет какие бы то ни было исходные каталоги и файлы, которые вы указываете; она делает только архивные копии. Следует также отметить, что использование точки (.) вместо звездочки (*) в предыдущей команде позволит включить в архив даже скрытые файлы (имена которых начинаются с точки).

Если вы попробуете выполнить эту команду на своем собственном компьютере (что точно стоит сделать), то увидите новый файл с именем `archivename.tar`. Рас-

ширение имени файла `.tar` указывать не обязательно, но всегда полезно четко обозначить, что это за файл, как можно большим количеством способов.

Вам не всегда нужно включать в свой архив все файлы в дереве каталогов. Предположим, вы смонтировали несколько видео, но в настоящее время оригиналы хранятся в каталогах вместе со всеми исходными графическими, аудио- и текстовыми файлами (содержащими ваши заметки). Единственные файлы, которые вам следует включить в резервную копию, — готовые видеоклипы с расширением `.mp4`. Вот как это сделать:

```
$ tar cvf archivename.tar *.mp4
```

Это прекрасно. Но эти видеофайлы огромны. Было бы неплохо сделать архив немного меньше, сжав его. Есть решение! Просто запустите предыдущую команду с аргументом `z` (zip). Программа получит указание сжать архив с помощью инструмента gzip. Можно также добавить расширение `.gz` в дополнение к уже существующему `.tar`. Помните о ясности. Вот как это может выглядеть:

```
$ tar czvf archivename.tar.gz *.mp4
```

Если вы опробуете это на собственных файлах `.mp4`, а затем запустите команду `ls -l` в каталоге, содержащем новые архивы, то заметите, что файл `.tar.gz` ненамного меньше, чем файл `.tar`: возможно, на 10 % или около того. Почему? Что ж, формат файла `.mp4` сам по себе предполагает сжатие, поэтому у gzip гораздо меньше возможностей.

Поскольку инструмент `tar` полностью осведомлен об окружении в Linux, вы можете использовать его для выбора файлов и каталогов, которые находятся за пределами вашего текущего рабочего каталога. В этом примере все файлы `.mp4` добавляются в каталог `/home/myuser/Videos/`:

```
$ tar czvf archivename.tar.gz /home/myuser/Videos/*.mp4
```

Поскольку архивные файлы могут увеличиваться, иногда имеет смысл разбить их на несколько меньших файлов, перенести их на новое место и затем там заново собрать исходный файл. Для этого предусмотрен инструмент split.

В этом примере аргумент `-b` инструктирует Linux разделить файл `archivename.tar.gz` на части размером по 1 Гбайт (здесь *`archivename`* — это любое имя, которое вы хотите присвоить файлу). Затем каждая из частей автоматически получает имя — `archivename.tar.gz.partaa`, `archivename.tar.gz.partab`, `archivename.tar.gz.partac` и т. д.:

```
$ split -b 1G archivename.tar.gz "archivename.tar.gz.part"
```

Обратно вы воссоздаете архив, считывая каждую часть по порядку (`cat archivename.tar.gz.part*`), а затем перенаправляете вывод в новый файл с именем `archivename.tar.gz`:

```
$ cat archivename.tar.gz.part* > archivename.tar.gz
```

4.4.2. Потоковая архивация файловой системы

Вот где начинается красота. Я собираюсь показать вам, как создать архивный образ работающей установки Linux и передать его в удаленное хранилище — и все это одной командой (рис. 4.3).

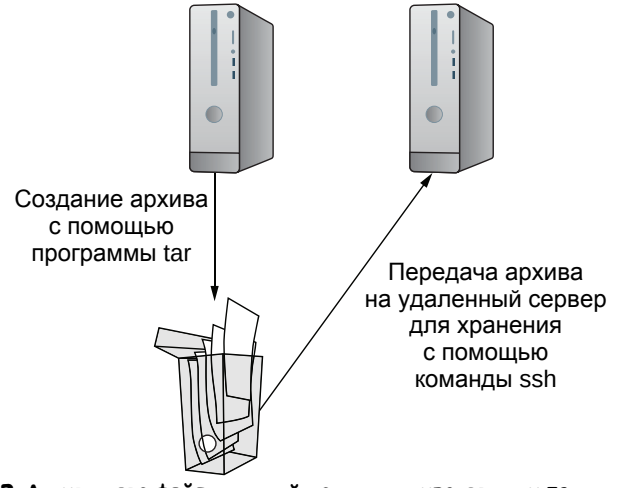


Рис. 4.3. Архив — это файл, который можно скопировать или переместить с помощью обычных инструментов Bash

Вот сама команда:

```
# tar czvf - --one-file-system / /usr /var \
  --exclude=/home/andy/ | ssh username@10.0.3.141 \
  "cat > /home/username/workstation-backup-Apr-10.tar.gz"\
```

Вместо того чтобы пытаться объяснить все это сразу, я буду приводить маленькие примеры, разбирая команду на части. Создадим архив содержимого каталога с именем `importantstuff`, который заполнен очень важными документами:

```
$ tar czvf - importantstuff/ | ssh username@10.0.3.141 \
<linearrow />   "cat > /home/username/myfiles.tar.gz"
importantstuff/filename1
importantstuff/filename2
[...]
username@10.0.3.141's password:
```

Вам нужно будет ввести пароль для вашей учетной записи на удаленном хосте

Позвольте мне объяснить этот пример. Вместо того чтобы вводить имя архива сразу после аргументов команды (как вы делали до сих пор), я добавил дефис (`czvf -`). Это позволяет вывести информацию в виде стандартного вывода, а также перенести сведения об имени файла архива обратно в конец команды и указать программе `tar` вместо этого ожидать исходный контент для архива. Затем я передал (`|`) безымянный сжатый архив в логин `ssh` на удаленном сервере, где меня попросили ввести пароль. В кавычки заключена команда `cat` для потока архивных данных,

которая записывает содержимое потока в файл `myfiles.tar.gz` в моем домашнем каталоге на удаленном хосте.

Как видно из рис. 4.4, одним из преимуществ создания архивов таким образом является то, что вы не тратите ресурсы на промежуточные действия. Нет необходимости даже временно сохранять копию архива на локальном компьютере. Представьте себе резервное копирование установки объемом 110 Гбайт из 128 Гбайт всего доступного пространства. Разве можно будет локально сохранить архив?

Это был просто каталог файлов. Допустим, вам нужно создать резервную копию активной установки операционной системы Linux на USB-накопителе, чтобы вы могли перенести ее на другой компьютер и развернуть на его основном диске. Предполагая, что свежая установка той же самой версии Linux на втором компьютере уже проведена, если вы выполните следующую операцию копирования/вставки, то создадите точную реплику первого.

Рис. 4.4. Потоковая передача архива в том виде, в котором он был создан, избавляет от необходимости сначала сохранять его на локальном диске

ПРИМЕЧАНИЕ

Система не будет работать на целевом диске, на котором еще не установлена файловая система Linux. Чтобы справиться с этой ситуацией, как вы вскоре увидите, вам потребуется инструмент dd.

В следующем примере создается сжатый архив на USB-накопителе, известном как `/dev/sdc1`. Аргумент `--one-file-system` исключает все данные из любой файловой системы, кроме текущей. Это означает, что псевдоразделы, такие как `/sys/` и `/dev/`, не будут добавлены в архив. Если есть другие разделы, которые вы хотите включить (как вы сделаете для `/usr/` и `/var/` в этом примере), то они должны быть явно добавлены. Наконец, вы можете исключить данные из текущей файловой системы, указав аргумент `--exclude`:

При построении архива исключаются данные из других разделов

```
# tar czvf /dev/sdc1/workstation-backup-Apr-10.tar.gz \
  --one-file-system \
  / /usr /var \
  --exclude=/home/andy/
```

При необходимости исключаются каталоги или файлы в выбранной файловой системе (извини, Энди)

Явно указываются разделы /usr и /var

Теперь вернемся к примеру полной команды. Опираясь на полученные знания, заархивируйте все важные каталоги файловой системы и скопируйте файл архива на USB-накопитель. Теперь это должно иметь смысл:

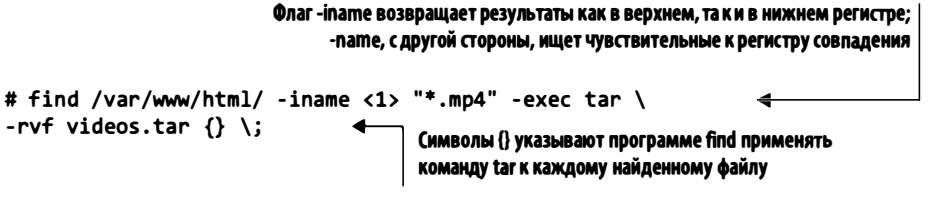
```
# tar czvf - --one-file-system / /usr /var \
  --exclude=/home/andy/ | ssh username@10.0.3.141 \
  "cat > /home/username/workstation-backup-Apr-10.tar.gz"
```

Все это хорошо, если файлы, которые нужно архивировать (и только эти файлы), находятся в одном месте единой структуры каталогов. Но что, если там есть другие файлы, которые вы не хотите включать? Есть ли способ собрать только определенные файлы и не получить при этом смесь с самими исходными файлами? Пришло время узнать о команде `find`.

4.4.3. Сбор файлов с помощью инструмента find

Команда `find` ищет в файловой системе объекты, отвечающие указанным вами правилам. Поиск выводит имена и местоположения обнаруживаемых файлов в так называемый *стандартный вывод* (stdout), который обычно отображается на экране. Но этот вывод можно так же легко перенаправить другой команде, например `tar`, которая затем скопирует файлы в архив.

Вот описание. На вашем сервере размещен сайт, который предоставляет множество видеофайлов с расширением `.mp4`. Файлы распределены по многим каталогам в `/var/www/html/`, поэтому идентифицировать их по отдельности будет непросто. Ниже показана команда, которая будет искать в структуре `/var/www/html/` файлы с расширением `.mp4`. Когда файл будет найден, программа `tar` выполнится с аргументом `-r` для добавления (в отличие от перезаписи) видеофайла в архив с именем `videos.tar`:



В данном случае будет хорошей идеей запустить `find` как `sudo`. Поскольку вы ищете файлы в системных каталогах, возможно, что некоторые из них имеют ограничивающие разрешения, способные помешать программе `find` читать их и, следовательно, сообщать о них.

И, поскольку мы говорим о команде `find`, я должен также рассказать вам о похожем инструменте под названием `locate`, который вы часто будете использовать, особенно при спешке. По умолчанию команда `locate` позволяет искать по всей системе файлы, соответствующие указанной вами строке. В примере ниже `locate` будет искать файлы, имена которых заканчиваются строкой `video.mp4` (даже если они имеют какой-либо префикс):

```
$ locate *video.mp4
```

Если вы запустите и сравните инструменты `locate` и `find`, то убедитесь, что `locate` почти всегда намного быстрее возвращает результаты. В чем секрет? Команда `locate` на самом деле не запускает поиск в самой файловой системе, а просто активизирует строку поиска для записей в существующем индексе. Суть в том, что, если индекс устаревает, поиск становится все менее и менее точным. Обычно индекс обновляется каждый раз при загрузке системы, но вы также можете выполнить обновление вручную, запустив команду `updatedb`:

```
# updatedb
```

4.4.4. Сохранение разрешений и прав собственности... и извлечение архивов

Я что-то упустил? Собственно, то, как извлечь файлы и каталоги из архива `tar`, чтобы вы могли снова их использовать. Но прежде, чем перейти к этому, я, как и обещал, займусь еще одним делом: позабочусь о том, чтобы ваши операции архивирования не повредили атрибуты прав доступа к файлам и владения файлами.

Разрешения

Как вы уже видели, команда `ls -l` выводит содержимое каталога в расширенной форме, показывая вам (справа налево) имя, срок службы и размер файла. Но она также дважды выводит имя (в данном примере `root`) и довольно загадочную строку, состоящую из букв `r`, `w` и `x`:

```
$ ls -l /bin | grep zcat
-rwxr-xr-x 1 root root 1937 Oct 27 2014 zcat
```

Сейчас я расшифрую эти два крайних левых раздела (рис. 4.5). Десять символов слева разделены на четыре отдельные секции. Первый дефис ❶ означает, что указанный объект является файлом. Он был бы заменен символом `d`, если бы это был каталог. Следующие три символа ❷ представляют собой разрешения файла, которые применяются к его владельцу, следующие три ❸ — это разрешения, применяемые к его группе, а последние три ❹ — разрешения для всех остальных пользователей.

Рис. 4.5. Анализ данных, отображаемых командой ls -l

В этом примере владелец файла обладает полными правами, включая права на чтение (`r`), запись (`w`) и выполнение (`x`). Члены группы и не только могут читать файл и выполнять его, но не перезаписывать (изменять) его.

Но что все это на самом деле означает? Здесь файл `zcat` — консольный сценарий, который считывает сжатые файлы. Разрешения говорят вам, что любой имеет право читать сам сценарий и выполнять его (что-то вроде `zcat myfile.zip`), но только владелец может редактировать (`w`) файл. Если другой пользователь попытается изменить файл, он получит предупреждение «Нет разрешения на запись».

Если вы хотите изменить права доступа к файлу, используйте инструмент изменения режима (`chmod`):

```
# chmod o-r /bin/zcat
# chmod g+w /bin/zcat
```

В этом примере убирается право читать файл для остальных (`o`) и добавляются разрешения на запись для группы (`g`). Владелец файла обозначается буквой `u` (пользователь).

Что такое группа

Вы можете рассматривать группу как обычную учетную запись пользователя: операции, которые они могут или не могут выполнять, а также их права доступа определяются правами доступа к файлам. Разница в том, что никто не может войти в систему Linux как группа. Тогда зачем создавать группы и какая у них цель? Вот объяснение.

Группы предоставляют мощный и очень эффективный способ организации ресурсов. Приведу простой пример. Рассмотрим компанию с несколькими десятками сотрудников, которым нужен некий доступ к серверу, но не обязательно к одним и тем же ресурсам. Вы можете создать пару групп с названием dev и IT, к примеру. Когда пользователи изначально станут получать свои учетные записи, все разработчики будут добавляться в группу dev, а все системные администраторы окажутся добавлены в группу IT. Теперь предположим, что используется файл конфигурации системы: вместо утомительного добавления прав доступа к файлам для каждого из 10 или 15 администраторов вы можете предоставить групповой доступ только группе IT. Все члены IT-группы будут автоматически добавлены, а все разработчики останутся исключенными.

Каждый пользователь системы вместе со многими приложениями автоматически получает свои собственные группы. Это объясняет, почему файлы, которые вы создаете, обычно принадлежат пользователю *`ваше_имя`* и входят в группу *`ваше_имя`*. Больше примеров применения групп будет в главе 9.

Есть еще два способа описания разрешений в Linux: числовой и с использованием маски. Разговор о маске сейчас будет немного неуместен, и в любом случае она не так часто применяется. Но вам нужно понимать числовую систему, где каждая возможная комбинация разрешений может быть представлена числом от 0 до 7.

В пошаговых руководствах и документации по командам часто сказано примерно следующее: дать файлу разрешения 644 (или что-то подобное) для успешного выполнения операции. Например, вызов приватной части пары ключей шифрования не будет работать, если у файла нет разрешений 400 или 600. Вам нужно знать, как это работает.

Разрешение на чтение всегда дается числом 4; разрешение на запись — числом 2; на выполнение — числом 1. Пользователь со всеми тремя разрешениями описывается числом 7 (4 + 2 + 1 = 7). Разрешения на чтение и запись, но не на выполнение, равны 6; чтение и выполнение, но не запись — 5; никаких разрешений вообще — 0.

Чтобы изменить разрешения для объекта, вы должны ввести итоговые числа для каждой категории пользователей (то есть владельца, группы и остальных). Например, для обновления исходного состояния файла `zcat` ранее вы использовали `chmod g+w` и `o-r`, что значит 755 (7 для владельца, затем 5 для группы и остальных). Удаление разрешения на чтение для остальных изменит это число на 751, а добавление разрешения на запись в группу снова изменит число на 771. Вот как использовать `chmod` для применения этого значения:

```
# chmod 771 /bin/zcat
```

Следующая таблица поможет вам запомнить все эти детали:

Разрешение	Буква	Число
Чтение	r	4
Запись	w	2
Выполнение	x	1

Права владения

Как насчет значений для владения файлами? Это те значения, которые определяют владельца файла (`u`) и группу (`g`). Проверьте сами. В своем домашнем каталоге создайте новый файл и затем выведите содержимое каталога в длинной форме. Вы увидите, что значения владельца и группы соответствуют вашему имени пользователя. В данном примере это `username`:

```
$ cd
$ touch newfile
$ ls -l
-rw-rw-r-- 1 username username 0 Jun 20 20:14 newfile
```

Я постоянно беспокоюсь о владении файлами. Предположим, один из моих пользователей запрашивает файл. Файл может быть слишком большим для отправки по электронной почте или содержать конфиденциальные данные, которые не следует отправлять по почте. Если я нахожусь на том же сервере, то очевидным решением будет скопировать его. Если я на другом сервере, то всегда могу использовать команду `scp` для передачи файла, а затем скопировать файл в домашний

каталог пользователя. Но в любом случае мне нужно будет ввести sudo для копирования файла в каталог пользователя, что означает, что его владельцем будет администратор.

Не верите? Попробуйте создать файл с помощью sudo:

```
$ sudo touch newerfile
[sudo] password for username:
$ ls -l
-rw-r--r-- 1 root root 0 Jun 20 20:37 newerfile
```

Обратите внимание, что владелец и группа для этого файла — администратор (root)

Что ж, теперь это будет настоящей проблемой, если моему пользователю когда-нибудь понадобится редактировать файл, который я любезно отправил. Нужно будет поменять владельца файла с помощью команды chown, во многом похожей на команду chmod, которую вы видели ранее. В этом примере предполагается, что имя учетной записи другого пользователя — otheruser. Создайте такую учетную запись с помощью команды sudo useradd otheruser:

```
$ sudo chown otheruser:otheruser newerfile
$ ls -l
-rw-r--r-- 1 otheruser otheruser 0 Jun 20 20:37 newerfile
```

Обратите внимание на нового владельца файла и группу

Мы говорим о разрешении и праве владения. Но какое отношение это имеет к извлечению ваших архивов? Вы бы расстроились, если бы я сказал вам, что есть большая вероятность, что все файлы и каталоги, восстановленные после катастрофического сбоя системы, будут иметь неправильные разрешения? Я так и думал. Подумайте об этом: вы перестраиваете свою систему и приглашаете всех пользователей снова войти в нее, но они сразу же начинают жаловаться, что не могут редактировать свои собственные файлы!

Я думаю, вам будет полезно опробовать это на практике. Вы можете работать с приведенными примерами самостоятельно, создать новый каталог и заполнить его несколькими пустыми файлами, а затем, если в вашей системе еще нет учетных записей других пользователей, создать новую:

```
$ mkdir tempdir && cd tempdir
$ touch file1
$ touch file2
$ touch file3
# useradd newuser
```

Символы && будут выполнять вторую команду, только если первая была успешной

В настоящий момент все три файла принадлежат вам. Используйте команду chown, чтобы изменить владельца одного из этих файлов на нового пользователя, а затем выполните команду ls -l, чтобы убедиться, что один из файлов теперь принадлежит новому пользователю:

```
# chown newuser:newuser file3
$ ls -l
-rw-rw-r-- 1 username username 0 Jun 20 11:31 file1
-rw-rw-r-- 1 username username 0 Jun 20 11:31 file2
-rw-rw-r-- 1 newuser  newuser  0 Jun 20 11:31 file3
```

Теперь создайте архив tar, включающий все файлы в текущем каталоге, как делали это раньше:

```
$ tar cvf stuff.tar *
file1
file2
file3
```

Чтобы извлечь архив, введите команду tar с именем архива, но на этот раз с аргументом x (для извлечения), а не c:

```
$ tar xvf stuff.tar
```

ВНИМАНИЕ

При извлечении архива любые файлы с одинаковыми именами в текущем каталоге перезаписываются без предупреждения. В нашем примере это хорошо, но на практике так не пойдет.

Запустив команду ls -l, вы снова увидите то, что не хотели бы видеть. Все три файла теперь принадлежат вам... даже file3:

```
$ ls -l
-rw-rw-r-- 1 username username 0 Jun 20 11:31 file1
-rw-rw-r-- 1 username username 0 Jun 20 11:31 file2
-rw-rw-r-- 1 username username 0 Jun 20 11:31 file3
```

Это плохо, и я уверен, что наш друг newuser тоже будет не рад. Какое решение? Во-первых, попробуем выяснить, в чем именно заключается проблема.

Как правило, только пользователи с полномочиями администратора могут работать с ресурсами в учетных записях других пользователей. Если бы я хотел, скажем, передать право владения одним из моих файлов коллеге, я бы не смог этого сделать, потому что пришлось бы перейти в чужую учетную запись. Щедрость имеет свои пределы. Поэтому, когда я пытаюсь восстановить файлы из архива, невозможно сохранить их в собственности других пользователей. Восстановление файлов с их исходными разрешениями представляет аналогичную (хотя и не идентичную) проблему. Решение состоит в том, чтобы выполнять эти операции от имени администратора, используя sudo. Теперь вы это знаете.

4.5. Архивирование разделов с помощью инструмента dd

Если вы достаточно хорошо узнаете инструмент dd, то с его помощью сможете делать практически все, но он особенно незаменим в работе с разделами. Ранее вы использовали инструмент tar для репликации целых файловых систем, копируя файлы с одного компьютера, а затем распаковывая их как есть, поверх свежей установки Linux на другом компьютере. Но, поскольку эти архивы файловой системы

не были полными образами, на хосте им требовалась работающая ОС, которая служила бы основой.

Использование команды dd, с другой стороны, позволяет сделать идеальные побайтные образы всего, что имеет цифровую природу. Но прежде, чем начать перебрасывать разделы с одного края земли на другой, я должен упомянуть, что в этой старой шутке администратора UNIX есть доля правды: *dd* означает *disk destroyer* («разрушитель диска»). Если в команде dd вы введете хотя бы один неправильный символ, вы можете мгновенно и навсегда уничтожить ценные данные на диске. И да, орфография имеет значение.

ПРИМЕЧАНИЕ

Правило работы с инструментом dd: сделайте паузу и тщательно подумайте, прежде чем нажимать клавишу Enter!

4.5.1. Работа с инструментом dd

Теперь, когда вы предупреждены, начнем с чего-то простого. Предположим, вы хотите создать точный образ всего диска с данными, обозначенный как /dev/sda. Вы подключили пустой диск (в идеале он должен иметь ту же емкость, что и ваша система /dev/sdb). Синтаксис прост: if= определяет исходный диск, а of= указывает файл или расположение, где вы хотите сохранить ваши данные:

```
# dd if=/dev/sda of=/dev/sdb
```

В следующем примере создается архив с расширением .img диска /dev/sda, который будет сохранен в домашнем каталоге вашей учетной записи пользователя:

```
# dd if=/dev/sda of=/home/username/sdadisk.img
```

Эти команды привели к созданию образов целых дисков. Вы также можете использовать один раздел диска. Следующий пример служит как раз для этого. В нем также используется аргумент bs, определяющий количество байтов для копирования за один раз (в данном случае 4096). Эксперименты со значением bs могут повлиять на общую скорость работы команды dd, хотя идеальная настройка будет зависеть от аппаратного обеспечения и других факторов:

```
# dd if=/dev/sda2 of=/home/username/partition2.img bs=4096
```

Восстановление выполняется очень просто: вы лишь меняете местами значения if и of. В этом случае аргумент if= указывает на образ, который вы хотите восстановить, а of= — на целевой диск, на который вы хотите записать образ:

```
# dd if=sdadisk.img of=/dev/sdb
```

Вам следует всегда проверять свои архивы, чтобы удостовериться, что они работают. Если это загрузочный диск, который вы создали, вставьте его в компьютер

и посмотрите, загрузится ли он, как ожидается. Если это обычный раздел данных, смонтируйте его, чтобы убедиться, что файл существует и доступен.

4.5.2. Стирание дисков с помощью инструмента dd

Несколько лет назад у меня был друг, который отвечал за безопасность в посольствах его правительства. Однажды он сказал мне, что каждому посольству под его наблюдением был предоставлен официальный правительственный молоток. Зачем? В случае если есть риск, что объект может попасть к недругам, этим молотком нужно разбить все их жесткие диски.

Что это значит? Почему бы просто не удалить данные? Вы шутите, правда? Всем известно, что при удалении файлов, содержащих конфиденциальные данные, с устройств хранения они фактически не удаляются. При наличии достаточного количества времени и мотивации практически все может быть извлечено почти с любого цифрового носителя, за исключением, возможно, тех, которые хорошо и должным образом повредили физически.

Однако есть одна команда, которую вы можете использовать, чтобы плохим парням было намного сложнее получить доступ к вашим старым данным. Это `dd`. Следующая команда потратит некоторое время на запись миллионов и миллионов нулей в каждый закоулок раздела `/dev/sda1`:

```
# dd if=/dev/zero of=/dev/sda1
```

Но можно поступить еще лучше. Используя файл `/dev/urandom` в качестве источника, вы можете наполнить диск случайными символами:

```
# dd if=/dev/urandom of=/dev/sda1
```

4.6. Синхронизация архивов с помощью инструмента rsync

Вы уже знаете, что для максимальной эффективности резервирование обязательно проводить регулярно. Одна из проблем заключается в том, что ежедневные передачи огромных архивов могут сильно снизить скорость работы вашей сети. Разве не лучше было бы передавать только несколько файлов, которые были созданы или обновлены с момента последнего использования, а не всю файловую систему? Это возможно. Познакомьтесь с командой `rsync`.

Я собираюсь показать вам, как создать удаленную копию каталога, содержащего файлы, и поддерживать актуальность копии даже после изменения локальных файлов. (Сначала необходимо убедиться, что пакет rsync установлен и на клиентском компьютере, и на хосте, с которыми вы будете работать.) Чтобы проиллюстрировать

то, что происходит между вашим собственным локальным компьютером и удаленным сервером (возможно, контейнером LXC, который вы используете), создайте каталог и заполните его несколькими пустыми файлами:

```
$ mkdir mynewdir && cd mynewdir
$ touch file{1..10}
```

Создает десять файлов с именами от file1 до file10

Теперь используйте команду `ssh` для создания нового каталога на вашем удаленном сервере, куда будут помещаться скопированные файлы, а затем запустите команду `rsync` с аргументами `-av`. Аргумент `v` указывает команде `rsync` отображать подробный список всего, что она делает. Аргумент `a` немного сложнее, но и намного важнее. Если указать супераргумент `-a`, команда `rsync` будет рекурсивно синхронизироваться (это означает, что подкаталоги и их содержимое также будут включены), а также сохранит специальные файлы, время модификации и (что крайне важно) атрибуты прав владения и доступа. Могу поспорить, что вы за `-a`. Вот пример:

```
$ ssh username@10.0.3.141 "mkdir syncdirectory"
$ rsync -av * username@10.0.3.141:syncdirectory
username@10.0.3.141's password:
sending incremental file list
file1
file10
file2
file3
file4
file5
file6
file7
file8
file9

sent 567 bytes  received 206 bytes  1,546.00 bytes/sec
total size is 0  speedup is 0.00
```

Укажите удаленный целевой каталог после двоеточия (:)

В подробном выводе отображены файлы, которые были скопированы

Если все прошло как надо, зайдите на свой удаленный сервер и просмотрите содержимое каталога `/syncdirectory/`. Там должно быть десять пустых файлов.

Чтобы обеспечить правильное тестовое выполнение команды `rsync`, можете добавить новый файл в локальный каталог `mynewdir` и использовать команду `nano`, скажем, для добавления нескольких слов в один из существующих файлов. Затем выполните ту же команду `rsync`, что и раньше. Когда это будет сделано, посмотрите, попали ли новый файл и обновленная версия старого файла на удаленный сервер:

```
$ touch newfile
$ nano file3
$ rsync -av * username@10.0.3.141:syncdirectory
username@10.0.3.141's password:
sending incremental file list
file3
newfile
```

В выводе перечислены только новые/обновленные файлы

Существует намного больше возможностей для средства резервного копирования `rsync`, которые вы можете использовать. Но, как и все другие инструменты, которые я обсуждаю в этой книге, у вас теперь есть основные знания. Куда двинетесь дальше — зависит от вас. В следующей главе вы узнаете об автоматизации резервного копирования с помощью системных планировщиков. Пока что я хотел бы поделиться еще одной мыслью по поводу резервных копий.

4.7. Вопросы планирования

Тщательный анализ поможет определить, сколько денег и усилий следует вложить в резервное копирование. Чем ценнее ваши данные для вас, тем надежнее они должны быть. Цель состоит в том, чтобы измерить ценность ваших данных по следующим пунктам.

- Как часто вы должны создавать новые архивы и как долго вы будете хранить старые копии?
- Сколько уровней проверки вы встроите в процесс резервного копирования?
- Сколько конкурентных копий ваших данных вы будете хранить?
- Насколько важно поддерживать географически удаленные архивы?

Еще один не менее важный вопрос: следует ли вам создавать инкрементные или дифференциальные резервные копии. Хотя вы, вероятно, захотите использовать команду `rsync` в любом случае, способ упорядочения резервных копий может повлиять как на ресурсы, которые они потребляют, так и на доступность архивов, которые создаются.

Используя *дифференциальную* систему, вы можете запускать полное резервное копирование один раз в неделю (понедельник), а также менее развернутое и более быстрое дифференциальное резервное копирование в каждый из следующих шести дней. Резервное копирование во вторник будет затрагивать только файлы, измененные после резервного копирования в понедельник. Каждое резервное копирование в среду, четверг и пятницу будет касаться всех файлов, измененных с понедельника. Пятничное резервное копирование, очевидно, займет больше времени и места, чем за вторник. С другой стороны, для восстановления дифференциального архива требуется только последняя полная резервная копия и самая последняя дифференциальная резервная копия.

Инкрементная система может также выполнять полное резервное копирование только по понедельникам и проводить резервное копирование, охватывающее только файлы, измененные во вторник. Резервное копирование в среду, в отличие от дифференциального подхода, будет касаться только файлов, добавленных или измененных со вторника, а в четверг — только тех, что были изменены со среды. Инкрементные резервные копии будут быстрыми и эффективными; но, поскольку обновленные данные распределены по большему количеству файлов, восстановление инкрементных архивов займет много времени и будет сложнее. Это показано на рис. 4.6.

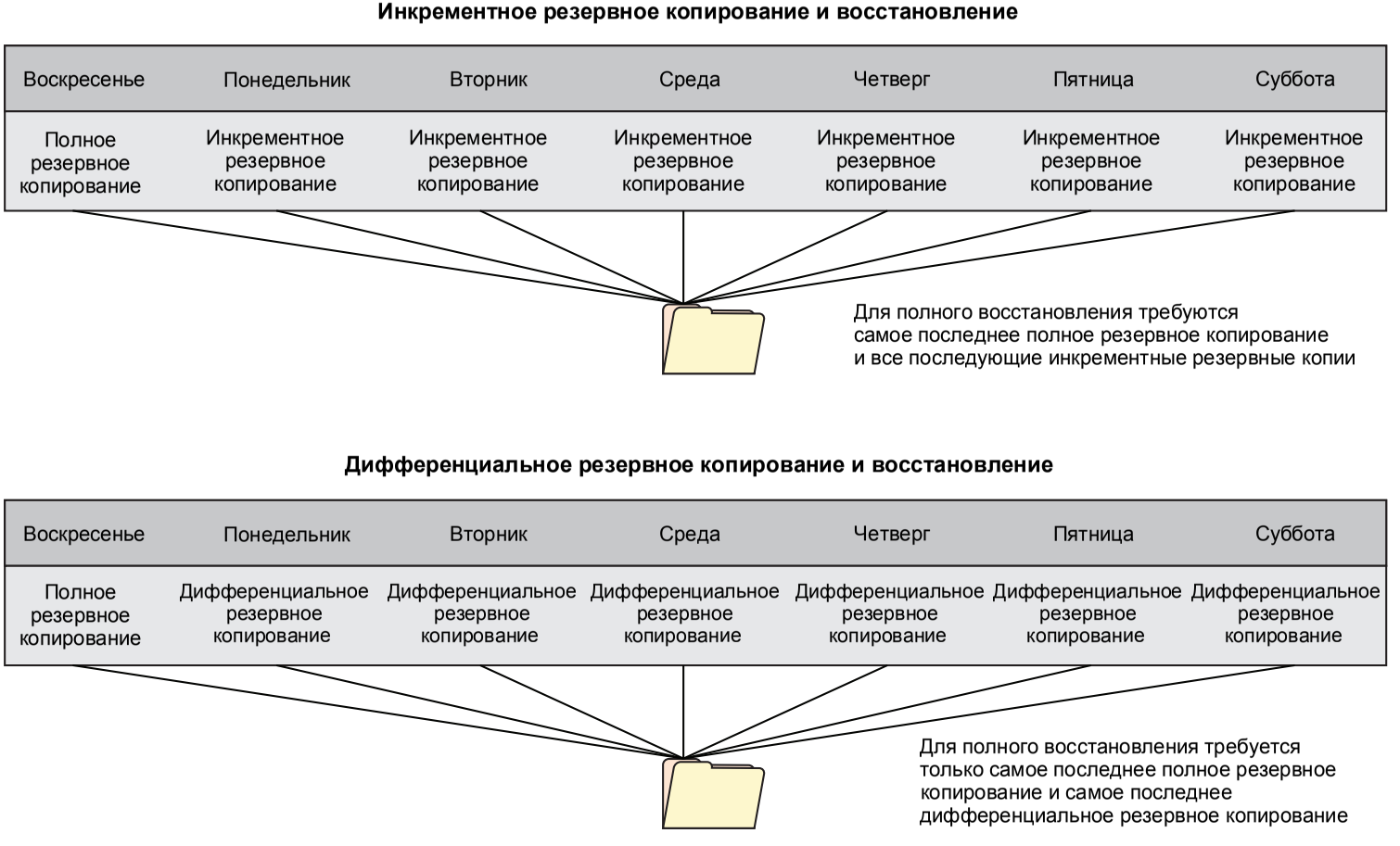


Рис. 4.6. Различия между инкрементной и дифференциальной системами резервного копирования

Резюме

- Отсутствие хороших резервных копий может испортить вашу карму.
- Команда `tar` обычно используется для полного или частичного архивирования файловых систем, тогда как команда `dd` больше подходит для создания образов разделов.
- Сжатие архива не только экономит место на накопителях, но и увеличивает пропускную способность во время передачи по сети.
- Каталоги, содержащие псевдофайловые системы, обычно не нуждаются в резервном копировании.
- Вы можете включить передачу архива в ту же команду, которая его генерирует. При желании можно обойтись без локального сохранения архива.
- Можно — и желательно — сохранить атрибуты владения и прав доступа к объектам, восстановленным из архива.
- Вы можете использовать инструмент `dd` для (достаточно) безопасного стирания старых дисков.
- Вы можете постепенно синхронизировать архивы, используя команду `rsync`, что значительно сокращает время и сетевые ресурсы, необходимые для текущих резервных копий.

Ключевые понятия

- *Архив* — это специально отформатированный файл, в который помещаются объекты файловой системы.
- *Сжатие* — процесс уменьшения дискового пространства, занимаемого файлом, с применением алгоритма сжатия.
- *Образ* — это архив, содержащий файлы и структуру каталогов, необходимые для воссоздания исходной файловой системы на новом месте.
- *Разрешения* — атрибуты, назначенные объекту, которые определяют, кто может его использовать и как.
- *Владение* — это владелец и группа, которым принадлежит объект.
- *Группа* — учетная запись, применяемая для управления разрешениями для нескольких пользователей.

Рекомендации по безопасности

- Создайте автоматизированный, надежный, проверенный и безопасный повторяющийся процесс для резервного копирования всех ваших важных данных.
- При необходимости разделите файловые системы с конфиденциальными данными, поместив их в свои собственные разделы, и подключайте их к файловой системе во время загрузки.

- Всегда проверяйте правильность прав доступа к файлам и предоставляйте только минимально необходимый доступ.
- Никогда не думайте, что данные на старом накопителе действительно стерты.

Обзор команд

- `df -h` выводит все разделы, активные в настоящий момент, в удобном для чтения формате.
- `tar czvf archivename.tar.gz /home/myuser/Videos/*.mp4` создает сжатый архив видеофайлов в указанном каталоге.
- `split -b 1G archivename.tar.gz archivename.tar.gz.part` разделяет большой файл на группу меньших по указанному размеру.
- `find /var/www/ -iname "*.mp4" -exec tar -rvf videos.tar {} \;` находит файлы по заданному критерию и передает их имена `tar` для включения в архив.
- `chmod o-r /bin/zcat` удаляет права на чтение для остальных.
- `dd if=/dev/sda2 of=/home/username/partition2.img` создает образ раздела `sda2` и сохраняет его в домашнем каталоге.
- `dd if=/dev/urandom of=/dev/sda1` перезаписывает раздел случайными данными, чтобы уничтожить старые данные.

Самотестирование

1. Какой из этих аргументов указывает программе `tar` сжать архив:
   а) `-a`;
   б) `-v`;
   в) `-z`;
   г) `-c`?
2. Какие из этих разделов вы бы не захотели включать в резервный архив:
   а) `/var`;
   б) `/run`;
   в) `/`;
   г) `/home`?
3. Как обычно указывается второй раздел на первом диске:
   а) `/dev/sdb2`;
   б) `/dev/srb0`;
   в) `/dev/sda2`;
   г) `/dev/sdb1`?

4. Какая из следующих команд создаст сжатый архив из всех файлов `.mp4` в каталоге:
   а) `tar cvf archivename.tar.gz *.mp4`;
   б) `tar cvf *.mp4 archivename.tar.gz`;
   в) `tar czvf archivename.tar.gz *.mp4`;
   г) `tar *.mp4 czvf archivename.tar`?
5. Какой из следующих инструментов поможет вам собрать части файла вместе:
   а) `cat`;
   б) `split`;
   в) `|`;
   г) `part`?
6. С помощью какой команды можно найти все файлы `.mp4` в определенных каталогах и добавить их в архив:
   а) `find /var/www/ -iname "*" -exec tar -rvf videos.tar {} \;`;
   б) `find /var/www/ -iname "*.mp4" -exec tar -vf videos.tar {} \;`;
   в) `find /var/www/ -iname "*.mp4" | tar -rvf videos.tar {} \;`;
   г) `find /var/www/ -iname "*.mp4" -exec tar -rvf videos.tar {} \;`?
7. Какая из следующих команд даст полные права владельцу файла, его группе — права на чтение и выполнение, а всем остальным — только права на выполнение:
   а) `chmod 752`;
   б) `chmod 751`;
   в) `chmod 651`;
   г) `chmod 744`?
8. Как работает команда `dd if=sdadisk.img of=/dev/sdb`:
   а) копирует содержимое диска `/dev/sdb` в файл под названием `sdadisk.img`;
   б) уничтожает все данные в сети;
   в) копирует образ под названием `sdadisk.img` на диск `/dev/sdb`;
   г) форматирует диск `/dev/sdb` и затем переносит на него `sdadisk.img`?

Ответы

1 — в; 2 — б; 3 — в; 4 — в; 5 — а; 6 — г; 7 — б; 8 — в.

5 Автоматизированное администрирование: настроика автоматического резервного копирования

В этой главе

- Автоматизация административных задач с помощью сценариев.
- Повышение безопасности и эффективности системы.
- Резервное копирование локальных данных.
- Планирование автоматизированных задач.

Есть одна вещь, которую, надеюсь, в предыдущей главе я объяснил достаточно ясно, — то, что регулярно необходимо делать надежные резервные копии системы. Но гарантировать *постоянство* очень непросто. Быть в курсе важных задач и немедленно реагировать на их результаты достаточно трудно, а помнить о необходимости скучного ежедневного или еженедельного резервного копирования нереально сложно.

Ни для кого не секрет: наилучшее решение проблемы — настроить автоматический планировщик для выполнения задачи за вас и забыть об этом. До недавнего времени планировщик, который вы использовали в Linux, почти наверняка представлял собой некоторую вариацию программной утилиты cron, и на самом деле это отличный выбор. Но менеджер процессов systemd, о котором вы узнали в главе 3, обзавелся таймерами systemd.

Я собираюсь рассмотреть в этой главе оба подхода, а также покажу вам, как запрограммировать создание резервных копий и любые другие задачи администри-

рования в виде сценариев, которые помещаются в автоматизированное расписание. Чтобы продемонстрировать, как все это работает в реальности, я создам команду для резервного копирования некоторых данных в корзину AWS Simple Storage Solution (S3), а затем использую команду для создания планировщиков, применяющих таймеры cron и systemd.

5.1. Сценарии с Bash

Сценарий Linux представляет собой простой текстовый файл, содержащий одну или несколько команд, совместимых с Bash (или другим интерпретатором оболочки). Возможность объединения нескольких команд в одном файле позволяет создавать исполняемые подпрограммы, которые по своей сложности и универсальности могут конкурировать с языками программирования.

5.1.1. Пример сценария резервного копирования системных файлов

Чтобы проиллюстрировать, как может выглядеть рабочий сценарий, позвольте мне показать вам короткий, хорошо написанный пример, который, вероятно, уже работает на вашем компьютере. Как только мы досконально проработаем сценарий, я расскажу вам, как он связан с этой главой. И не забудьте о другом важном выводе из этого упражнения: если вы умеете читать сценарии, вы можете их писать. Это точно такой же набор навыков.

Следующий сценарий использует ряд мощных инструментов, чтобы выполнить довольно простые действия: создать безопасные резервные копии четырех важных системных файлов для обеспечения удобной замены в случае, если оригиналы каким-либо образом оказались повреждены. Рисунок 5.1 иллюстрирует действия сценария в виде блок-схемы. Обратите внимание, что `$FILE` — переменная, используемая для представления набора имен файлов, обрабатываемых сценарием.

Перейдите в каталог `/etc/cron.daily/` и просмотрите его содержимое. Вы, вероятно, найдете файл с именем `passwd`. Выведите файл на экран, выполнив команду `less` (либо `cat`, `nano` или `vim` — на ваш вкус). Если файла в каталоге нет, то запрос выглядит так:

```
#!/bin/sh

cd /var/backups || exit 0

for FILE in passwd group shadow gshadow; do
        test -f /etc/$FILE              || continue
        cmp -s $FILE.bak /etc/$FILE     && continue
        cp -p /etc/$FILE $FILE.bak && chmod 600 $FILE.bak
done
```

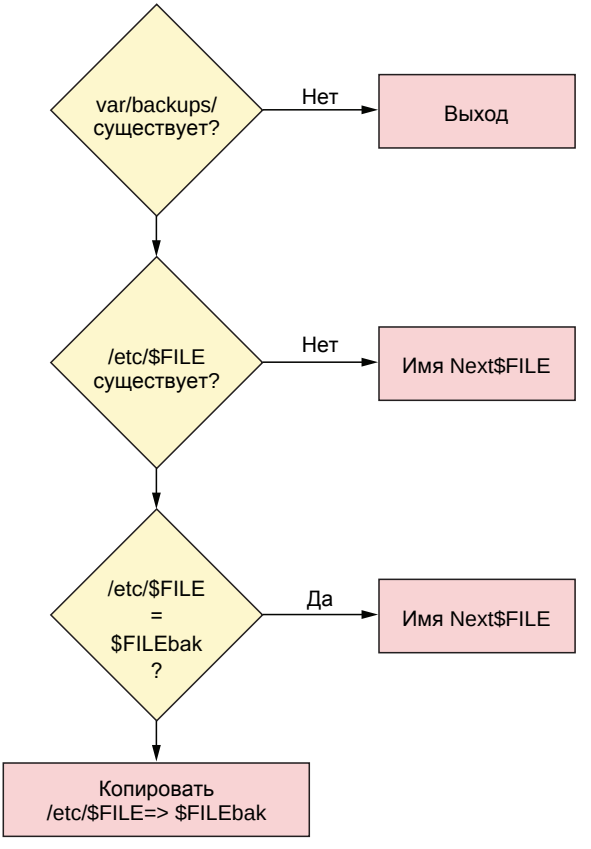


Рис. 5.1. Алгоритм принятия решений, отслеживаемый сценарием passwd

Обычно символ # обозначает комментарий, который интерпретатор игнорирует. В этом конкретном случае, где есть два символа — и #, и !, — Linux прочитает комментарий и использует его значение (`/bin/sh`) в качестве активной оболочки. Это обозначение широко известно под названием *«строка шебанга»*, но не спрашивайте меня почему. Хотя `sh` является альтернативой `bash`, для наших целей прямо сейчас между ними нет никаких практических различий.

Следующая строка в сценарии меняет текущий каталог на `/var/backups/`. Если такого каталога не существует, работа сценария завершается и выдается код состояния выхода — `0`. Это означает, что команда была выполнена успешно:

```
cd /var/backups || exit 0
```

Последовательность `||` (иногда называется двойным «пайпом») может быть прочитана как «или». Таким образом, эта строка означает: либо изменить каталог на `/var/backups/`, либо завершить работу сценария. Если все идет по плану, последующие операции сценария будут выполняться в каталоге `/var/backups/`.

ПРИМЕЧАНИЕ

Коды выхода передаются после завершения команды Linux. Ноль будет указывать на успех, в то время как другие числа можно настроить для обозначения какой-либо ошибки.

В следующей части кода строка со словом `for` начинает цикл. Те из вас, кто имеет опыт программирования, без труда поймут, что здесь происходит: сценарий по очереди назначит каждую из следующих четырех строк (`passwd`, `group` и т. д.) в качестве значения переменной `FILE`. Затем он выполнит блок кода между словами `do` и `done`:

```
for FILE in passwd group shadow gshadow; do
```

Приведу краткое определение некоторых понятий.

- *Цикл* — последовательность действий, которые ограничены зарезервированными словами и должны повторяться до тех пор, пока выполняется указанное условие.
- *Строка* — непрерывная последовательность символов.
- *Переменная* — значение, которое может изменяться и динамически внедряться в действия сценария.
- *Зарезервированное слово* — термин, интерпретируемый в соответствии с предопределенным значением в оболочке командной строки.

Вернемся к сценарию: первая из последующих строк будет проверять наличие файла в каталоге `/etc/`, имя которого соответствует текущему значению переменной `$FILE`. Если в `/etc/` нет файла с таким именем, сценарий продолжит работу, присвоив переменной `$FILE` следующую строку (следующее имя файла) и проверив его наличие:

```
test -f /etc/$FILE || continue
```

Если такой файл есть в `/etc/`, то сценарий будет сравнивать (`cmp`) содержимое файла с содержимым файла с таким же именем и расширением `.bak` в текущем каталоге (`/var/backups/`). Если операция сравнения (`&&`) прошла успешно, оболочка продолжит цикл `for` и проверит следующую строку. А если содержимое этих двух файлов не будет совпадать, то программа просто переместится на следующую строку:

```
cmp -s $FILE.bak /etc/$FILE && continue
```

Затем сценарий наконец выполнит свое предназначение: скопирует текущую версию из каталога `/etc/` в каталог `/var/backups/`, добавит к имени расширение `.bak` и ограничит права доступа к файлу, чтобы предотвратить чтение неавторизованными пользователями. Во время операции будут перезаписаны любые существующие файлы с таким же именем. Флаг `-p` в примере сохраняет исходные атрибуты владельца исходного файла и временную отметку:

```
cp -p /etc/$FILE $FILE.bak && chmod 600 $FILE.bak
```

Что делает этот сценарий? Он предназначен для создания копий указанных файлов конфигурации, которые были обновлены с момента их последнего резервного

копирования. Вот как это работает: если файлы с указанными именами существуют в активном каталоге `/etc/` и их содержимое отличается от содержимого файлов с аналогичным именем в каталоге `/var/backups/`, то файлы из `/etc/` будут скопированы в `/var/backups/`, соответствующим образом переименованы и защищены.

А как обстоят дела с оставшимися четырьмя файлами (`passwd`, `group`, `shadow` и `gshadow`)? Это файлы, содержимое которых определяет, каким образом отдельные пользователи и группы смогут получить доступ к конкретным ресурсам. Например, если вы посмотрите на содержимое `/etc/passwd`, то увидите отдельную строку для каждой существующей учетной записи. В следующем фрагменте вы можете видеть, что обычным учетным записям пользователей назначаются идентификаторы пользователей и групп (`1000` в случае `ubuntu`), домашний каталог (`/home/ubuntu/`) и оболочка по умолчанию (`bash`). У некоторых системных пользователей, таких как `syslog`, по умолчанию также есть оболочка, которая, как ни странно, установлена в `/bin/false`. Это способ предотвратить вход пользователя в систему с применением этой учетной записи, что было бы небезопасно:

```
$ cat /etc/passwd
[...]
syslog:x:104:108::/home/syslog:/bin/false
_apt:x:105:65534::/nonexistent:/bin/false
sshd:x:106:65534::/var/run/sshd:/usr/sbin/nologin
ubuntu:x:1000:1000::/home/ubuntu:/bin/bash
mysql:x:107:111:MySQL Server,,,:/nonexistent:/
  bin/false
  bind:x:108:112::/var/cache/bind:/bin/false
  newuser:x:1002:1002:,,,:/home/newuser:/bin/bash
  messagebus:x:109:114::/var/run/dbus:/bin/false
```

Идентификатор пользователя ubuntu (1000), домашний каталог (/home/ubuntu) и оболочка по умолчанию (bash)

Учетные записи других пользователей не должны использоваться для входа в систему (/bin/false)

При добавлении нового пользователя в систему с помощью:

```
# useradd -m alan
```

новые строки будут добавлены в каждый из файлов `passwd`, `shadow` и `group`. Фактически все связанные между собой операции администрирования пользователей можно выполнять из оболочки командной строки (или с помощью сценариев) без необходимости непосредственного редактирования этих файлов.

ПРИМЕЧАНИЕ

Система Ubuntu предпочитает команду adduser user-name вместо useradd user-name, хотя оба варианта применимы. Одним из преимуществ команды adduser является то, что домашний каталог создается автоматически, тогда как useradd требует аргумента -m. Команда adduser запрашивает пароль для нового пользователя. А если вы задействуете useradd, придется выполнить команду sudo passwd new-user-name отдельно, чтобы установить пароль.

Давным-давно здесь была бы подключена зашифрованная версия пароля каждого пользователя. Из практических соображений, поскольку файл `passwd` должен оставаться доступным для чтения любому пользователю системы, решили, что даже зашифрованные пароли — явление неразумное. Было принятно перенести эти пароли в `/etc/shadow`. Используя разрешения `sudo`, вы должны взглянуть на этот файл с зашифрованными паролями в своей собственной системе. Вот таким образом:

```
$ sudo cat /etc/shadow
```

Файл `/etc/group` содержит основную информацию обо всех существующих в данный момент системах и группах пользователей. Вы можете вручную редактировать его, чтобы управлять данными в нем. Например, вы можете предоставить административные права новым пользователям, присоединяющимся к вашей команде, добавив их имена в группу `sudo`. Такая строка будет выглядеть следующим образом:

```
sudo:x:27:steve,newuser,neweruser
```

Не надо добавлять пробелы между именами и запятыми. Это приведет к немедленному сбою.

И последнее: файл `/etc/gshadow` содержит зашифрованные версии групповых паролей на тот случай, если вы когда-либо захотите разрешить доступ к групповым ресурсам сторонним пользователям.

Как вы, возможно, уже догадались, данный сценарий стал отличным примером для этой главы из-за того, где он находится, а именно: в каталоге `/etc/cron.daily/`. Сценарии, сохраненные в каталоге `/cron.daily/`, выполняются каждый день. Мы скоро ко всему этому вернемся. А пока в качестве другого простого примера вот вам файл сценария с именем `upgrade.sh`, позволяющего автоматически обновлять все установленное программное обеспечение:

```
#!/bin/bash
# Script to automate regular software upgrades

apt update
apt upgrade -y
```

Как вы, несомненно, помните, команда `apt update` синхронизируется с индексами в онлайн-репозиториях, гарантируя, что APT будет в курсе всех самых последних доступных пакетов и версий. Команда `apt upgrade` загрузит и установит все соответствующие обновления. Аргумент `-y` автоматически ответит «Да», когда потребуется подтвердить операцию.

Но вы пока не совсем готовы к запуску своего сценария. Поскольку вы собираетесь запускать его как программу, необходимо изменить атрибуты файла, чтобы сделать его исполняемым. Следует сделать это так: набрать команду `chmod +x`, затем указать имя файла:

```
$ chmod +x upgrade.sh
```

Вот и все. Теперь вы можете свободно копировать файл в каталог `/etc/cron.daily/`, где он может присоединиться к `passwd` и другим, поскольку они запускаются ежедневно:

```
# cp upgrade.sh /etc/cron.daily/
```

Поскольку запускается программа `apt`, новый сценарий потребует прав администратора, но нет необходимости включать ключевое слово `sudo` в саму команду. Cron по умолчанию всегда будет работать от имени администратора. Если вы действительно хотите запустить сценарий непосредственно из оболочки командной строки, то добавьте слово `sudo` и имя файла, перед которым пропишите точку и косую черту, чтобы сообщить Linux, что команда, на которую вы ссылаетесь, находится в текущем каталоге:

```
$ sudo ./upgrade.sh
```

5.1.2. Пример сценария для изменения имен файлов

Позвольте мне добавить еще пару «сценарных инструментов». Вы, вероятно, уже сталкивались с тем фактом, что оболочка Linux иногда может неправильно интерпретировать имена файлов, включающие пробелы. Вот как это будет выглядеть, если вы попытаетесь найти содержимое файла с именем `big name`:

```
$ cat big name
cat: big: No such file or directory
cat: name: No such file or directory
```

Есть простое решение: заключить имя файла в одинарные или двойные кавычки, например:

```
$ cat 'big name'
Hello world
```

Но такой вариант не всегда применим. Тогда вы можете попробовать автоматизировать процесс для преобразования пробелов в именах файлов, например, в символ нижнего подчеркивания. Затем, спустя некоторое время, когда вы наткнетесь на каталог, содержащий множество проблемных файловых имен, вы сможете написать сценарий для быстрого исправления ситуации. Что ж, вот он:

```
#!/bin/bash
echo "which directory would you like to check?"
read directory
   find $directory -type f | while read file; do
    if [[ "$file" = *[[:space:]]* ]]; then
    mv "$file" `echo $file | tr ' ' '_'`
    fi;
   done
```

Строка `echo` выводит свой текст на экран и ожидает действий пользователя. Пользователь вводит допустимый каталог, например `/home/ubuntu/files/`, который будет назначен в качестве значения переменной `directory`. Команда `find` активизирует возврат всех файловых объектов (`-type f`) в указанный каталог. Набор имен файлов из `find` будет считываться по одному в цикле `while`, при этом каждое из них будет проверяться на наличие пробела. Если пробел найден, то любые пробелы (`' '`) в имени файла будут изменены (`mv`) на нижнее подчеркивание (`'_'`). Остается команда `fi`: останавливает цикл, когда в каталоге больше нет имен файлов.

```
$ ls
file name   file - name
```

теперь будет выглядеть так:

```
$ ls
file_name   file_-_name
```

ПРИМЕЧАНИЕ

Тщательно продумайте каждый шаг сценария и убедитесь, что точно понимаете, что происходит.

Давайте остановимся и вспомним, где именно мы находимся. Это всегда хороший способ убедиться, что вы не просто смотрите на деревья, а видите весь лес. Вот что происходило до этого момента.

- Глава была посвящена использованию сценариев для создания автоматических резервных копий.
- Вы исследовали сценарий для резервного копирования пользовательских файлов администратора из каталога `/etc/` в `/var/backup/`.
- Вы узнали о сохранности пользовательских файлов администратора.
- Вы написали свой собственный простой сценарий.

Что еще предстоит сделать.

- Вы создадите резервную копию своих собственных данных в корзине AWS S3.
- Вы будете использовать инструменты cron и anacron для планирования регулярного резервного копирования.
- Вы узнаете, как это сделать с помощью системных таймеров.

Сценарии можно использовать не только для резервного копирования и переименования файлов. Ввиду постоянно растущих требований к серверам и сетевой среде, в которой иногда требуются сотни или даже тысячи динамически генерируемых виртуальных микросервисов, ручное администрирование практически невозможно. В вашей карьере системного администратора вам, вероятно, придется создавать сценарии для подготовки и запуска множества виртуальных машин, а также для мониторинга массивных и постоянно меняющихся сред.

5.2. Резервное копирование данных в системе AWS S3

Есть две причины, по которым я выбрал платформу Amazon AWS S3 для примера резервного копирования.

- Крайне важно всегда хранить копии значимых данных в дополнительном надежном месте.
- Архивирование на S3 сейчас очень популярно и безумно просто.

Вот и все. Тот факт, что это была бы отличная возможность для хитроумного продвижения моей книги *Learn Amazon Web Services in a Month of Lunches* (Manning, 2017), в которой полно всего, что вам, возможно, следует знать о AWS, не повлияло на мой выбор. Правда. Ну, может быть, совсем немного.

В любом случае, если у вас еще нет собственного аккаунта на сайте AWS, вы все равно можете переходить к следующему разделу, а там заменить собственный сценарий резервного копирования тем, который я укажу. Кроме того, вы также можете посетить сайт https://aws.amazon.com/ru/ и зарегистрировать свою учетную запись. Открытие учетной записи не будет вам ничего стоить, и в рамках бесплатного использования многие услуги (включая 5 Гбайт хранилища) будут доступны в течение первого года.

Кстати, даже после того, как срок бесплатного пользования AWS истечет, хранилище будет стоить всего лишь 0,023 доллара за 1 Гбайт в месяц. Это достаточно дешево, чтобы вы в целом смогли пересмотреть свой взгляд на сторонние платформы.

5.2.1. Установка интерфейса командной строки AWS (CLI)

Существует много способов администрирования AWS из браузерной консоли AWS, но настоящие администраторы Linux выполняют свою работу по-другому. Если вы собираетесь включить свои резервные копии на S3 в сценарий, это должно быть что-то, что будет работать в оболочке командной строки. Для этого нет ничего лучше, чем собственный инструментарий AWS CLI компании Amazon. Поскольку он работает в среде Python, вам необходимо запустить как минимум Python 2 (версия 2.6.5) или Python 3 (версия 3.3). Кроме того, вам понадобится pip-менеджер пакетов Python для управления установкой. На момент написания этой главы Ubuntu пытается сделать Python 3 единственной допустимой версией, хотя другие дистрибутивы по-прежнему смогут работать с Python 2.

Если данная команда установки не работает, то вам совершенно точно нужно установить менеджер pip. (Используйте либо команду `apt install python-pip`, либо `apt install python3-pip`.) Вот сверхсекретный, инсайдерский «я-могу-рассказать-

вам-это-но-потом-мне-придется-вас-убить» скрытый код, который сообщит, какая установка pip вам понадобится. Если это сработает, то все верно. Если не сработает, попробуйте другой:

Менеджер pip хорошо отображает детали прогресса в реальном времени

```
$ pip3 install --upgrade --user awscli
Collecting awscli
  Downloading awscli-1.11.112-py2.py3-none-any.whl (1.2MB)
    100% |################################| 1.2MB 946kB/s
Collecting PyYAML<=3.12,>=3.10 (from awscli)
  Downloading PyYAML-3.12.tar.gz (253kB)
    100% |################################| 256kB 2.2MB/s
[...]
Collecting jmespath<1.0.0,>=0.7.1 (from botocore==1.5.75->awscli)
  Downloading jmespath-0.9.3-py2.py3-none-any.whl
Collecting six>=1.5 (from python-dateutil<3.0.0,>=2.1->
    botocore==1.5.75->awscli)
  Downloading six-1.10.0-py2.py3-none-any.whl
Building wheels for collected packages: PyYAML
  Running setup.py bdist_wheel for PyYAML ... done
  Stored in directory: /home/ubuntu/.cache/pip/wheels
    /2c/f7/79/13f3a12cd723892437c0cfbde1230ab4d82947ff7b3839a4fc
Successfully built PyYAML
Installing collected packages: PyYAML, pyasn1, rsa, colorama, six,
    python-dateutil, docutils, jmespath, botocore, s3transfer, awscli
Successfully installed PyYAML awscli botocore colorama docutils
    jmespath pyasn1 python-dateutil rsa s3transfer six
```

Сводка всех пакетов, установленных pip

5.2.2. Настройка аккаунта AWS

Теперь пришло время связать локальный интерфейс командной строки AWS со своим аккаунтом AWS. Для этого вам нужно получить некоторые ключи доступа. На любой странице консоли щелкните кнопкой мыши на строке меню с именем аккаунта (в верхнем правом углу страницы), а затем выберите пункт My Security Credentials (Мои учетные данные безопасности) (рис. 5.2).

Перейдя на страницу My Security Credentials (Мои учетные данные безопасности), щелкните кнопкой мыши на разделе Access Keys (Access Key ID and Secret Access Key) (Ключи доступа (Идентификатор ключа доступа и секретный ключ доступа)), чтобы развернуть его, и обратите внимание на предупреждение, которое может появиться: «Существующие корневые ключи не могут быть отображены». Затем нажмите кнопку Create New Access Key (Создать новый ключ доступа). Вам будет показан новый идентификатор ключа доступа и сопутствующий ему секретный ключ доступа. Первый выполняет ту же функцию, что и логин для входа, а второй действует как его пароль. Вы можете либо выгрузить ключ доступа и сохранить его в безопасном месте на своем компьютере, либо выбрать, скопировать и вставить его куда необходимо.

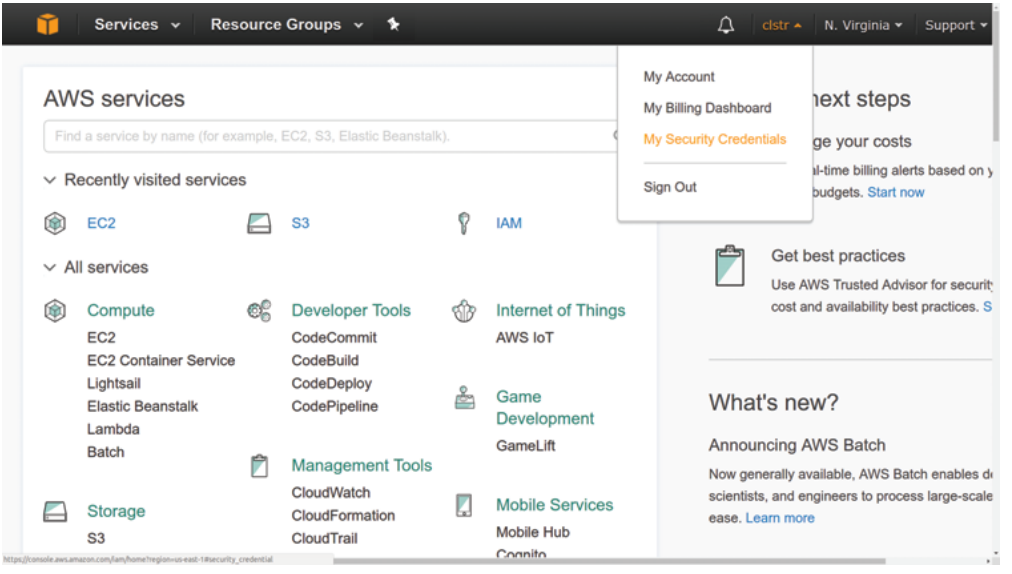


Рис. 5.2. Консоль AWS с прямыми ссылками на десятки сервисов AWS. Выделен пункт меню My Security Credentials (Мои учетные данные безопасности)

Для настройки откройте оболочку командной строки на вашем компьютере и выполните в ней команду `aws configure`. Вам будет предложено указать идентификатор ключа доступа, секретный ключ доступа, регион AWS, который вы решите установить по умолчанию, и формат, который следует использовать для вывода. Последние два поля при желании можно оставить пустыми. В следующем примере документации AWS приводятся поддельные учетные данные (вы никогда не должны публично показывать реальный набор ключей):

```
$ aws configure
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: us-east-1
Default output format [None]:
```

Определяет географию AWS центра, из которого будут запускаться ресурсы

Можно выбрать текстовый (по умолчанию), JSON или табличный формат

ПРИМЕЧАНИЕ

Выбирая регион по умолчанию, имейте в виду, что местные нормативные режимы могут ограничивать резервное копирование некоторых данных на офшорные серверы.

Теперь у вас все готово к работе. Следующая команда:

```
$ aws s3 ls
```

перенесет все корзины S3 в вашу учетную запись. *Корзина (bucket)* — это термин, который AWS использует для обозначения каталогов.

5.2.3. Создание корзины AWS

Предположим, у нас есть новая учетная запись и ничего в ней не отображается. Вы должны создать новую корзину с помощью команды `mb`. При выборе имени корзины следует помнить, что оно должно быть уникальным во всей системе S3. Из примера ниже видно, что имя вроде `mybucket` не будет принято системой:

```
$ aws s3 mb s3://mybucket
make_bucket failed: s3://mybucket/ An error occurred (BucketAlreadyExists)
  when calling the CreateBucket operation: The requested bucket name is
  not available. The bucket namespace is shared by all users of the system.
  Please select a different name and try again.
```

Если использовать менее распространенные слова и добавить к ним несколько чисел, будет, вероятно, намного лучше:

```
$ aws s3 mb s3://linux-bucket3040
```

Еще шаг, и мы проделаем всю работу по резервному копированию на внешний носитель. Предположим, что файлы, для которых необходимо выполнить резервное копирование, находятся в каталоге `/dir2backup`, расположенном в вашем домашнем каталоге. Вот как это будет происходить:

```
aws s3 sync /home/username/dir2backup s3://linux-bucket3040
```

Команда `s3 sync` работает во многом так же, как и инструмент `rsync`, о котором мы говорили в главе 4. При первом запуске все содержимое исходного каталога загружается в вашу корзину S3, куда впоследствии будут загружаться только новые или измененные файлы. Создать сценарий для запуска этой команды синхронизации довольно просто. Вот как он будет выглядеть:

```
#!/bin/bash
/usr/local/bin/aws s3 sync \
  /home/username/dir2backup s3://linux-bucket3040
```

Обратите внимание, как я прописал полный путь в команде `aws` (`/usr/local/bin/aws`). Это сделано для того, чтобы Bash знал, где в системе найти команду. Вы можете подтвердить местоположение AWS, используя команду `whereis`:

```
$ whereis aws
aws: /usr/local/bin/aws
```

Наличие отличного инструмента резервного копирования еще не означает, что вы будете его использовать. Для этого лучше задействовать планировщики задач. О некоторых из них мы поговорим ниже.

5.3. Планирование регулярного резервного копирования с помощью инструмента cron

Инструмент cron имеет несколько вариантов реализации. Поскольку их больше одного, можно ожидать, что существует несколько способов выполнения определенной задачи. Чтобы разобраться с вариантами, отфильтруйте объекты в каталоге `/etc/`, которые имеют в имени сочетание букв *cron*:

```
$ ls /etc | grep cron
anacrontab
cron.d
cron.daily
cron.hourly
cron.monthly
crontab
cron.weekly
```

Из них только `anacrontab` и `crontab` являются файлами, остальные — каталоги. Начнем с того, что разберемся, как работают каталоги.

Если у вас есть, например, сценарий резервного копирования файловой системы в исполняемом файле, который вы хотите запускать с заданным интервалом, вы копируете его в соответствующий каталог: `cron.hourly/` для повторения каждый час, `cron.daily/` — для запуска ежедневно и т. д. Каталог `cron.d/` немного отличается. Он предназначен для файлов, содержимое которых определяет время выполнения команд.

Предположим, вы хотите запускать обновление программного обеспечения, о котором я писал выше, один раз в понедельник, но без сценария. Вы можете создать файл в каталоге `/etc/cron.d` со следующим содержимым:

```
21 5 * * 1 root apt update && apt upgrade
```

Этот пример будет запускать обновление в 05:21 каждое утро понедельника. Как это работает? Обратите внимание на первые символы: `21` и `5`. Первое число (`21`) указывает минуты, когда вы хотите выполнить команду. В следующем поле вы указываете время дня (здесь число `5` означает 5 утра). Следующие две звездочки и следующее за ними число `1` указывают на то, что вы хотите, чтобы расписание выполнялось каждый день каждого месяца этого года и каждый раз по понедельникам (`1` означает понедельник). Чтобы избежать путаницы, для обозначения воскресенья можно использовать `0` или `7`. Аргумент `root` означает, что команда будет выполняться от имени администратора.

Почему так рано? Видимо, потому, что спрос на пропускную способность сети станет выше, когда работники придут в офис. Почему 05:21, а не 05:00? Вы же не хотите привыкнуть планировать запуск всех своих сценариев точно на начало часа (или на любое другое конкретное время), ведь это может в конечном итоге привести к загруженности сети. Лучше равномерно распределить работу.

Вы можете добавить эту строку непосредственно в файл `/etc/crontab`, и она будет работать точно так же, без необходимости создавать отдельный файл. Но я бы не стал так делать. Конечно, это не повлечет за собой зомби-апокалипсис, но все равно идея не очень хорошая. Видите ли, файл `crontab`, скорее всего, будет перезаписан во время обновления системы и ваши пользовательские команды окажутся утеряны.

Тогда в чем назначение файла `crontab`? Как видно из содержимого файла (рис. 5.3), это планировщик, который выполняет сценарии в каталогах `/cron`. Потратьте пару минут, чтобы просмотреть каждую команду.

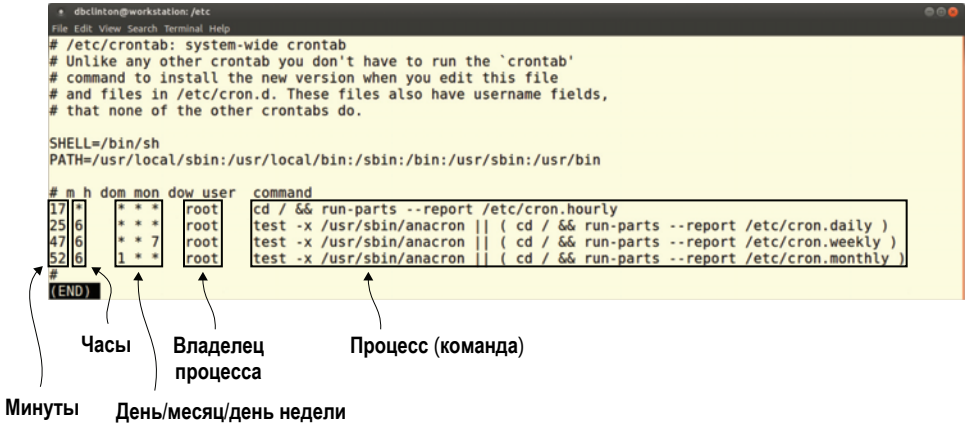


Рис. 5.3. Хорошо задокументированный файл /etc/crontab, показывающий четыре задания, каждое из которых предназначено для запуска содержимого каталога /etc/cron.?

Команда `test -x` ведет к последним трем командам на рис. 5.3, которые подтверждают существование бинарной программы под названием `anacron` и тот факт, что она является исполняемой. Если это не соответствует действительности, сценарии будут запущены в каталогах `/cron.?`. Команда `text -x` может быть полезна для работы сценариев, когда вам необходимо подтвердить состояние объекта перед запуском связанной операции.

Вы определенно захотите оставить файл `/etc/crontab` на усмотрение профессионалов. Но, поскольку Linux заботится о вас, разработчики дали вам на пробу свой собственный `crontab`. Он будет запускать команды от вашего имени пользователя, а не от имени администратора и только с доступными вам разрешениями и не будет перезаписываться во время обновлений.

Хотите посмотреть, что у вас уже запланировано? Выполните команду `crontab -l`. Если вы еще не добавили никаких команд, то получите сообщение `no crontab for` *`ваше_имя`*, показанное в примере ниже:

```
$ crontab -l
no crontab for ubuntu
```

Вы можете редактировать свой crontab, используя команду crontab -e. При первом редактировании вам будет предложено выбрать текстовый редактор. Если вы уже знакомы с Nano, выбирайте его, так как он самый простой из трех перечисленных ниже вариантов:

```
$ crontab -e
no crontab for ubuntu - using an empty one
Select an editor. To change, run 'select-editor'.
  1. /bin/nano
  2. /usr/bin/vim.basic
  3. /usr/bin/vim.tiny

Choose 1-3 [1]:
```

Проверка наличия заданий

Выбор текстового редактора

Выбор редактора Nano

ПРИМЕЧАНИЕ

По умолчанию в некоторых системах пользователь не может создавать задания crontab, если не создан файл /etc/cron.allow, содержащий соответствующее имя пользователя. Однако Debian/Ubuntu разрешают выполнение отдельных заданий crontab прямо из каталога.

Эти инструменты на основе файла cron хорошо выполняются на компьютерах (например, на коммерческих серверах), которые, вероятно, работают постоянно. А как насчет выполнения важных заданий, скажем, на вашем ноутбуке, который периодически выключается? Конечно, неплохо было бы сказать cron (или cron.daily и др.), что нужно сделать резервную копию ваших файлов в 05:21 в понедельник утром, но какова вероятность, что вы не забудете вовремя встать, чтобы загрузить свой ноутбук? Если быть откровенным, вряд ли вообще такое произойдет. Поэтому пришло время поговорить об anacron.

5.4. Планирование нерегулярного резервного копирования с помощью инструмента anacron

Существует еще один файл cron, который мы еще не обсуждали, — это anacrontab, в котором мы будем запускать операции в определенное время после каждой загрузки системы. Если вы хотите сделать резервную копию файлов на своем ноутбуке, но не можете гарантировать, что он будет включен в нужное время в течение дня, можно добавить соответствующую строку в anacrontab.

В файле anacrontab следует обратить ваше внимание на то, что записи имеют только два столбца для управления временем, а не пять, как в cron. Это потому, что anacron работает не в абсолютном времени, а относительно самой последней загрузки системы. А вот и файл во всей красе:

Каталоги в разделе PATH файла anacrontab. На файлы в них можно ссылаться без полного дерева каталогов

```
# /etc/anacrontab: configuration file for anacron
# See anacron(8) and anacrontab(5) for details.
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
HOME=/root
LOGNAME=root

# These replace cron's entries
1 5 cron.daily run-parts --report /etc/cron.daily
7 10 cron.weekly      run-parts --report /etc/cron.weekly
@monthly 15 cron.monthly run-parts --report /etc/cron.monthly
```

Это задание выполняет любые сценарии в каталоге /etc/cron.daily/ один раз в день

Например, строка `cron.daily` в файле `anacrontab` запускается через день, ровно через пять минут после загрузки компьютера. Команда `cron.weekly` выполняется с семидневным интервалом (то есть когда с момента последнего запуска прошло не менее семи дней) и через десять минут после загрузки.

А что насчет команд `cron.?`; не были ли они ранее запущены через файл `/etc/crontab`? Да, но только если `anacron` не активен в системе. Как видно из файла `crontab`, `anacron` имеет приоритет над `cron`. Помня про это, сценарий резервного копирования для вашего ноутбука можно запустить так:

```
1   10   myBackupJob  /home/myname/backup.sh
```

Эта команда запускает сценарий `backup.sh` не чаще одного раза в день, через десять минут после загрузки системы. Идентификатор задания — `myBackupJob`, и файл журнала с таким именем и информацией о состоянии задания будет сохранен в каталоге `/var/spool/anacron/`.

5.4.1. Запуск задания синхронизации S3

Теперь, узнав все необходимое о сценариях — AWS S3, `cron` и `ancron`, — вы наконец готовы сделать правильный выбор и решить, как лучше спланировать резервное копирование? И, как всегда, правильный ответ зависит от контекста.

Если вы хотите убедиться, что резервное копирование будет выполнено даже на машине, которая не всегда работает, можете добавить следующую строку в файл `anacrontab` (при условии, что у вас уже есть исполняемый файл сценария с таким именем в `/etc/`):

```
1   10   myDailyBackup  /etc/s3script.sh
```

В данном примере сценарий запускается каждый день (`1`) через десять минут (`10`) после загрузки системы. Для ваших серверов с режимом работы 24/7 специфичные для синтаксиса директивы могут быть добавлены в `crontab` вашего пользователя или файл каталога `/etc/cron.d/` (хотя вам нужно в версию `cron.d` добавить имя пользователя, например `root`).

Следующий пример запускает сценарий в 05:47 каждое утро каждый день недели каждого месяца (***):

```
47 5 * * * /home/myusername/s3script.sh
```

Синтаксис crontab для запуска сценария каждый день в 5:47

Поскольку мы говорим о простом сценарии, в данном случае его можно реализовать эффективнее, вставив команду `aws` в файл `cron`, например `anacrontab`:

```
47 5 * * * username /usr/local/bin/aws s3 sync
  /home/username/dir2backup s3://linux-bucket3040
```

Определенную команду, подобную этой операции резервного копирования, можно вставить непосредственно в файл `crontab`. Обратите внимание на имя пользователя и точное расположение `aws`. Они четко устанавливают как владельца процесса, который вы хотите запустить, так и структуру файловой системы исполняемого файла.

5.5. Планирование регулярного резервного копирования с помощью таймеров systemd

Если кто-то критикует новый системный таймер, альтернативный `cron`, то только потому, что он заметно сложнее и требует выполнения ряда шагов для настройки. Вместо того чтобы просто создать свой сценарий и затем скопировать файл в нужный каталог `cron`, вам необходимо создать и сохранить два отдельных файла в неизвестном каталоге (это пока не является стандартным для всех дистрибутивов Linux), а затем запустить обе команды `systemctl`.

Сложно? Пожалуй, да. Но точно не непреодолимо. Таймеры systemd имеют ряд существенных преимуществ, в том числе более глубокую интеграцию с другими системными службами (включая журналы) и возможность выполнять команды, основанные на изменениях состояния системы (например, когда кто-то подключил USB-устройство), а не просто установить время.

Углубленное изучение функций, которые вас заинтересуют, я оставлю вам для самостоятельного изучения в свободное время. А сейчас проведу вас по пути создания простой резервной копии. Что от вас требуется? Создайте архив `tar` с динамическим именем на сайте с поддержкой Apache, работающем на вашем сервере. Для этого примера вы можете выбрать любые каталоги для резервного копирования, и вам не нужно настраивать и запускать сайт.

Прежде всего, взгляните на таймеры, если они у вас уже есть. Мои отображаются на рис. 5.4 с помощью команды `systemctl list-timers --all`.

Эти таймеры были созданы системой автоматически. Чтобы создать свой собственный, начните со сценария резервного копирования. Поскольку я в душе очень креативный и творческий человек, я назову свой файл сценария `site-backup.sh`. Вот как это выглядит:

```
#!/bin/bash
NOW=$(date +"%m_%d_%Y")
tar czvf /var/backups/site-backup-$NOW.tar.gz /var/www
```

Назначает системную дату переменной окружения $NOW

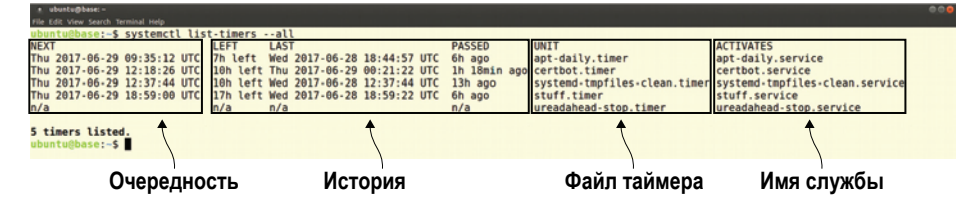


Рис. 5.4. Команда systemctl list-timers --all предоставляет развернутую информацию истории данных для всех существующих заданий таймера systemd

Было бы намного проще идентифицировать архивы, если бы имена моих архивных файлов включали дату их создания. Для этого я назначил текущую системную дату в качестве значения переменной **$NOW** и включил ее в имя файла нового архива. Вот как может выглядеть полученное имя файла:

```
site-backup-11_28_2017.tar.gz
```

Не забудьте сделать исполняемым свой файл сценария (`chmod +x site-backup-11_28_2017.tar.gz`). На самом деле никогда не забывайте делать исполняемым файл сценария!

```
$ chmod +x site-backup.sh
```

Теперь вам нужно создать файлы `.service` и `.timer`. Как я уже писал ранее, нет единого расположения, где хранятся все служебные файлы, но `/lib/systemd/system/` и `/etc/systemd/system/` могут выполнять эту функцию. При наличии выбора я предпочту `/etc/systemd/system/`, потому что для меня это легко запоминающееся и логичное место. Вы можете выбрать свой вариант.

Я начну с файла `.service`, который назову `site-backup.service`. «Служебные» файлы являются общими для всех операций systemd. Они предназначены для того, чтобы определенным, предсказуемым образом описывать и определять системные службы. Значение `Description` должно содержать текст, которого, по вашему мнению, будет достаточно для описания службы, а строка `ExecStart` станет указывать на местоположение исполняемого ресурса: в данном случае сценария. Это все, что нужно systemd для выяснения ваших желаний:

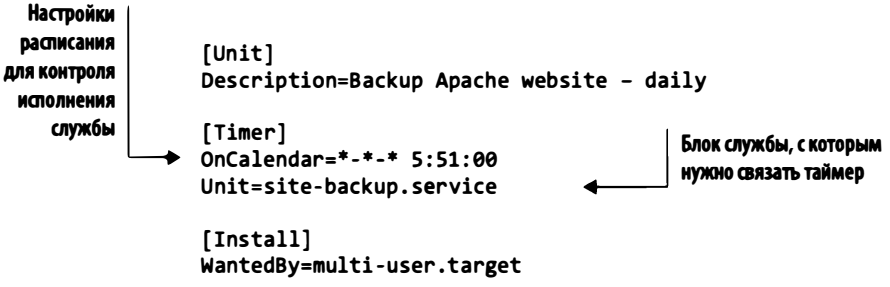
```
[Unit]
Description=Backup Apache website

[Service]
Type=simple
ExecStart=/home/username/site-backup.sh

[Install]
WantedBy=multi-user.target
```

Исполняемый ресурс, который будет запущен службой

Файл `.timer`, относящийся к таймерам systemd, сообщает systemd, когда необходимо запустить соответствующую службу. Связь устанавливается через строку `Unit` в разделе `[Timer]`, которая в данном случае указывает на мой файл `site-backup.service`. Обратите внимание, что и здесь значение `OnCalendar` устанавливается на ежедневное выполнение (`*-*-*`) в 05:51 утра, а значением `Unit` является файл `site-backup.service`:



С помощью этих файлов вы запускаете службу, используя команду `systemctl start`. Кроме того, вы устанавливаете его для автоматической загрузки каждый раз, когда система запускается с помощью команды `systemctl enable`:

```
# systemctl start site-backup.timer
# systemctl enable site-backup.timer
```

Хотите узнать больше о состоянии вашей службы? Начните с использования команд `is-enabled` и `is-active`:

```
# systemctl is-enabled backup.timer
enabled
# systemctl is-active backup.timer
active
```

Наконец, когда вы отредактируете свой файл `.timer`, потребуется обновить систему. Когда вы будете прорабатывать на практике то, что узнали из этой главы, вам, вероятно, придется кое-что редактировать. Естественно, вы хотите знать, как это сделать. Так вот:

```
# systemctl daemon-reload
```

Резюме

- Грамотно написанные сценарии Bash позволяют эффективно и надежно автоматизировать как сложные, так и простые административные задачи.
- Linux хранит информацию об учетной записи пользователя и об аутентификации в текстовых файлах (с именами `passwd`, `group`, `shadow` и `gshadow`) в каталоге `/etc/`.
- Можно создать резервную копию локальных данных в корзине S3 и управлять ими в течение всего жизненного цикла непосредственно из оболочки командной строки.

- Копирование исполняемого сценария в один из каталогов `/etc/cron.?` заставляет его запускаться с соответствующим интервалом.
- Указания по выполнению команд, добавленные в файл `anacrontab`, относятся к времени загрузки системы, а не к абсолютному моменту времени.
- Таймеры systemd могут быть установлены относительно абсолютного времени и реакции на системные события, такие как изменение состояния оборудования.

Ключевые понятия

- Все команды Linux выводят код завершения после своего завершения: `0` означает успешное выполнение. Однако все положительные целые числа могут быть задействованы программой для обозначения различных состояний отказа.
- Консольный доступ к ресурсам AWS обеспечивается с помощью ключей доступа (идентификатора ключа доступа и секретного ключа доступа).
- *Корзина* — это объект AWS, который работает почти так же, как каталог в операционной системе.

Рекомендации по безопасности

- Заблокируйте свои системные учетные записи (например, системный журнал и в идеале даже административную запись), чтобы предотвратить их удаленное использование.
- Включите резервное копирование во внешнее хранилище. Это добавит вам еще один уровень надежности данных.
- Всегда скрывайте от публичного доступа свои (AWS) ключи, не говоря уже о паролях и ключах шифрования.

Обзор команд

- `#!/bin/bash` (так называемая строка шебанга) сообщает Linux, какой интерпретатор оболочки вы собираетесь использовать для сценария.
- `||` добавляет в сценарий условие выбора. Читается как «команда слева успешна» или «выполнить команду справа».
- `&&` вставляет в сценарий условие добавления. Читается как «если команда слева успешна» и «выполнить команду справа».
- `test -f /etc/filename` проверяет наличие указанного файла или каталога.
- `chmod + x upgrade.sh` делает файл сценария готовым к работе.
- `pip3 install --upgrade --user awscli` устанавливает интерфейс командной строки AWS, используя менеджер пакетов Python.
- `aws s3 sync /home/username/dir2backup s3://linux-bucket3040` синхронизирует содержимое локального каталога с указанной корзиной S3.

- `21 5 * * 1 root apt update && apt upgrade` (директива `cron`) выполняет обе команды `apt` в 05:21 каждое утро.
- `NOW=$(date +"%m_%d_%Y")` присваивает переменной сценария текущую дату.
- `systemctl start site-backup.timer` активизирует системный таймер systemd.

Самотестирование

1. Какой символ используется для написания комментариев в сценарии Linux:
   а) `!`;
   б) `//`;
   в) `#`;
   г) `^`?
2. Какова задача символов `||` в сценарии Linux:
   а) выбор;
   б) добавление;
   в) условие;
   г) комментарий?
3. Какой тип данных хранится в файле `/etc/shadow`:
   а) данные группы учетных записей и оболочки;
   б) данные о членстве в группе;
   в) зашифрованные пароли аккаунтов;
   г) зашифрованные групповые пароли?
4. Какая команда создаст новый каталог в аккаунте AWS S3:
   а) `s3 mb s3://mybucket`;
   б) `aws s3 mb s3://mybucket`;
   в) `aws s3 cb s3://mybucket`;
   г) `aws s3 sync mb s3://mybucket`?
5. Какая из следующих команд позволит ввести директиву `cron`, которая будет запускаться от вашего имени:
   а) `nano anacrontab`;
   б) `crontab -l`;
   в) `nano /etc/crontab`;
   г) `crontab -e`?
6. Что из следующего будет запускать резервное копирование каждое утро понедельника:
   а) `21 * 1 ** root apt update && apt upgrade`;
   б) `21 5 ** 1 root apt update && apt upgrade`;

в) `21 1 ** 0 root apt update && apt upgrade`;

г) `21 5 * 4 * root apt update && apt upgrade`?

7. Что из перечисленного не подходит для компьютеров, которые периодически выключаются:

   а) `crontab`;

   б) `anacron`;

   в) таймер systemd;

   г) `anacrontab`?

8. Какова цель команды `systemctl enable site-backup.timer`:

   а) загружает таймер резервного копирования сайта вручную;

   б) настраивает таймер резервного копирования сайта при загрузке системы;

   в) отображает текущее состояние таймера резервного копирования сайта;

   г) запускает принудительный таймер резервного копирования сайта до того, как компьютер может сломаться?

Ответы

1 — в; 2 — а; 3 — в; 4 — б; 5 — г; 6 — б; 7 — а; 8 — б.

6 Инструменты для критических ситуаций: создание устройства для восстановления системы

В этой главе

- Восстановление поврежденных систем Linux.
- Управление ресурсами с помощью загрузочных дисков Linux.
- Восстановление данных с поврежденного носителя.
- Управление недоступной файловой системой.

Linux еще заставит вас поплакать, и не пытайтесь убедить себя в обратном. Вы можете забыть синтаксис команд (вот почему вы всегда должны держать экземпляр этой книги под рукой). Вы (или пользователи, которых вы поддерживаете) ненароком введете команды с ошибками и окончательно уничтожите документы. Или вы будете испытывать бессилие, когда поймете, что какой-то важный компонент оборудования или программного обеспечения вышел из строя. Вот такая благодарность за все, что вы сделали для него за все эти годы. Правильное резервное копирование, как показали последние несколько глав, означает, что вы можете отказаться от нефункционирующей операционной системы (ОС) или компьютера и воссоздать все это где-нибудь еще. Но это всегда будет планом Б. План А — восстановить!

В этой главе я познакомлю вас с ключевыми инструментами из пакета восстановления Linux. Вы узнаете, как можно использовать загрузочный диск для установки новой копии Linux, монтировать диск, который доставляет вам массу проблем, и либо обновить поврежденные файлы конфигурации, чтобы вы могли снова нормально загрузиться, либо восстановить любые данные, которые можно восстановить перед перепрофилированием, либо полностью уничтожить поврежденный диск.

Вы увидите, как файлы нефункциональной системы можно запустить и оживить в их собственной виртуальной среде, и вы даже сможете выполнить такую операцию, как, например, изменение забытого пароля пользователя.

Есть много чего, что может не заладиться при попытке собрать аппаратное и программное обеспечение, — не всегда компоненты будут хорошо работать вместе. Я собираюсь сосредоточиться на самых критических ситуациях, подобных следующим.

- Ваш компьютер загружается, жесткий диск работает, но Linux не запускается.
- Ваш компьютер загружается (насколько вы можете судить об этом), но вы не совсем уверены, полностью ли функционирует жесткий диск.
- Все работает, но проблема с программным обеспечением или утерянный пароль не позволяют войти в Linux.

Конкретная проблема, с которой вы столкнетесь, определит ваш дальнейший план действий по возвращению к работе. На рис. 6.1 показано несколько вариантов диагностики и восстановления, большинство из которых я рассмотрю позже в этой главе.

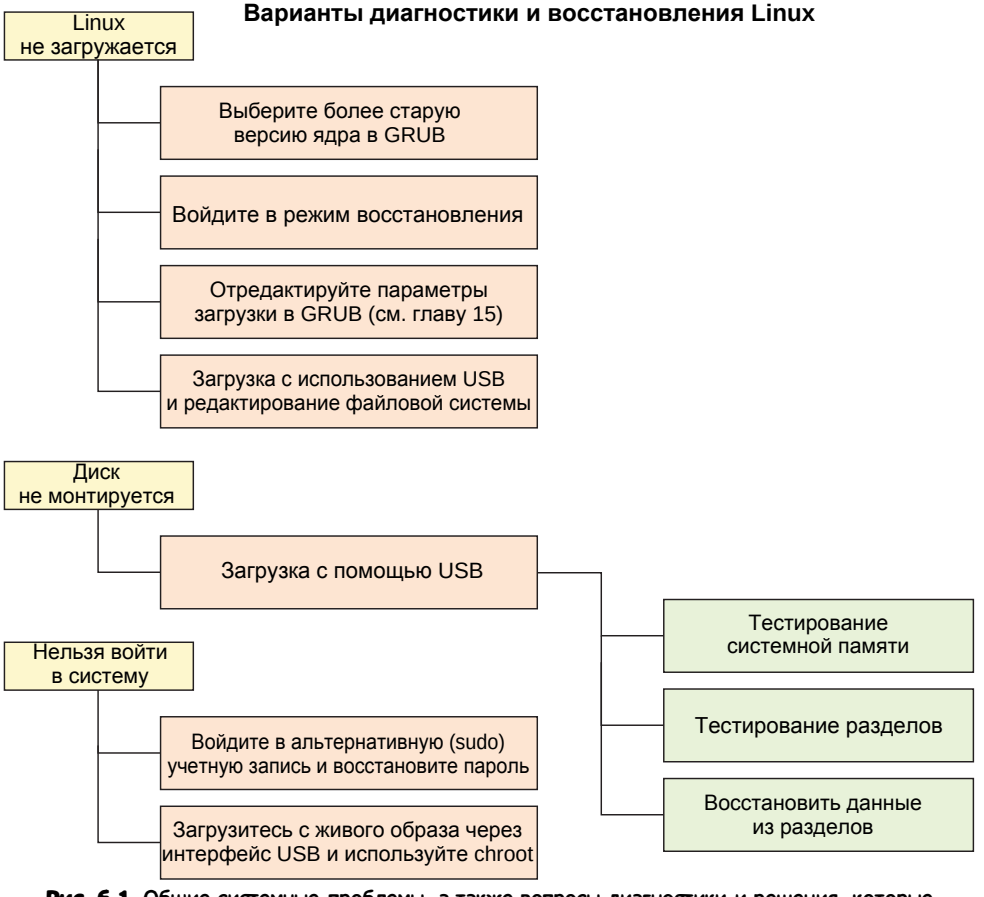


Рис. 6.1. Общие системные проблемы, а также вопросы диагностики и решения, которые мы рассмотрим в этой главе

6.1. Работа в режиме восстановления

Операционная система Linux не позволяет вам привычным способом войти в систему? Возможно, процесс загрузки неожиданно остановился перед тем, как отобразить экран входа в систему. Вам понадобятся некоторые базовые инструменты системного администрирования.

Но подождите: если Linux не загружается, как вы собираетесь запустить эти инструменты? Что ж, даже если Linux не загружается до появления обычной командной строки, вы наверняка попадете в меню GRUB. Там (как показано на рис. 6.2) вы сможете использовать клавиши ↑, ↓ и Enter, чтобы выбрать ядро Linux, работающее в режиме восстановления. Это, как вы скоро увидите, предоставляет целый ряд хитростей.

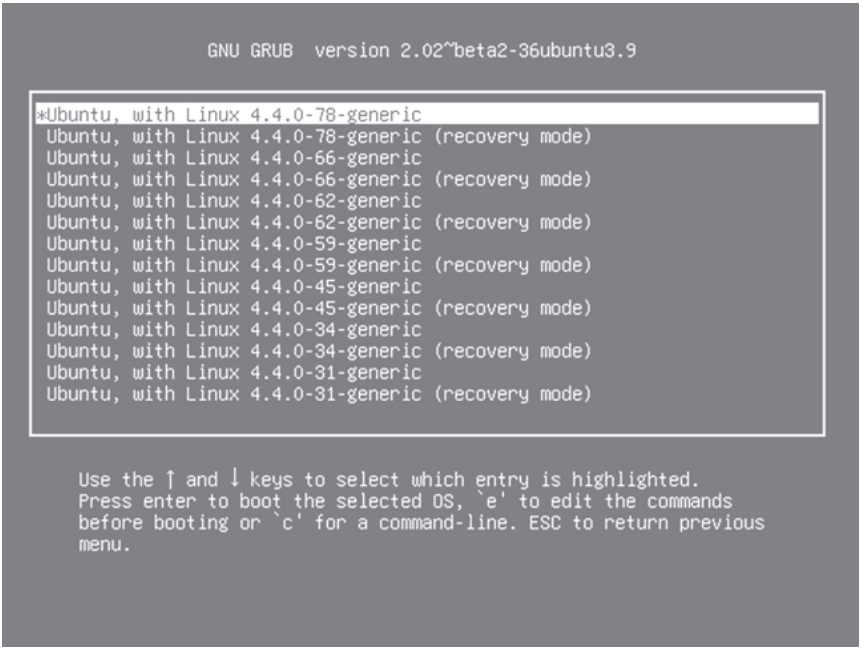


Рис. 6.2. Меню дополнительных параметров GRUB для установки Ubuntu, содержащее ссылки как на текущую, так и на более старую версию ядра, а также на параметры запуска в режиме восстановления

Однако, прежде чем вы сможете в полной мере воспользоваться этими инструментами, вам сначала необходимо понять, что такое GRUB и как он работает. В следующих подразделах я собираюсь объяснить, что делает GRUB, как его можно использовать для доступа к инструментам восстановления и как вы можете задействовать эти инструменты, чтобы справиться с некоторыми неприятностями.

6.1.1. Системный загрузчик GRUB

Что такое *GRUB*? Аббревиатура расшифровывается как GNU GRand Unified Bootloader. Хорошо, что такое *bootloader*? Это код, который ОС использует, чтобы «оживить» себя при включении. Рисунок 6.3 иллюстрирует процесс.

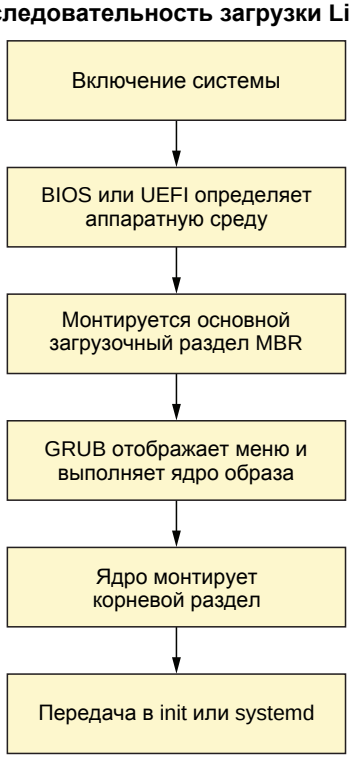


Рис. 6.3. Основные шаги в процессе загрузки компьютера под управлением операционной системы Linux

При включении компьютера инструкции по встроенному программному обеспечению, встроенные в базовое аппаратное оборудование, идентифицируют доступные сетевые ресурсы, ресурсы хранилища и памяти. Это делалось с помощью системы BIOS на старых компьютерах, а в последнее время делается с использованием UEFI (оба варианта кратко рассматривались в главе 4).

Как только система находит раздел жесткого диска, содержащий основную загрузочную запись (MBR), она загружает содержимое в активную память. В системах Linux раздел MBR включает несколько файлов, которые при запуске представляют собой одну или несколько загружаемых конфигураций загрузки образа ядра. Вы можете загрузить любую из этих конфигураций из меню загрузчика GRUB.

ПРИМЕЧАНИЕ

Зачастую GRUB загружает образ по умолчанию автоматически, не спрашивая вашего мнения, если предыдущая сессия не удалась. Если вы захотите принудительно вызвать меню GRUB, нажмите правую клавишу Shift при загрузке компьютера.

6.1.2. Использование режима восстановления в Ubuntu

Как видно на рис. 6.4, после загрузки Ubuntu в режиме восстановления вам будет показано меню инструментов, позволяющих устранить некоторые распространенные проблемы с загрузкой. Стоит попробовать каждый из них по очереди: вдруг один из них в конечном итоге решит вашу главную проблему. Например, утилита clean удаляет неиспользуемые файлы, если у вас есть подозрение, что проблема связана с нехваткой пространства на жестком диске. А утилита dpkg попытается исправить любые вышедшие из строя программные пакеты на основе APT, которые могут все запутать. (Инструменту dpkg сначала может потребоваться сетевое подключение.)

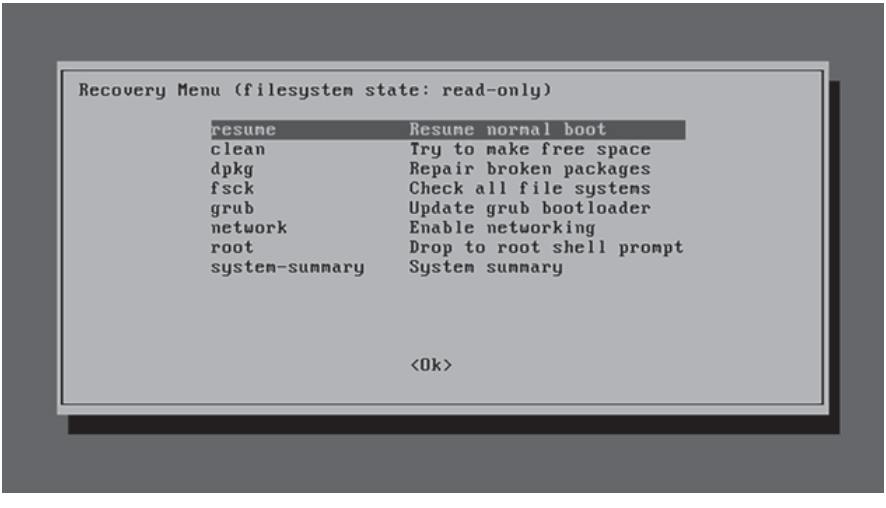


Рис. 6.4. Меню режима восстановления на Ubuntu со ссылками на некоторые базовые инструменты диагностики и восстановления, а также с возможностью открытия сеанса оболочки от имени администратора

Утилита root открывает сеанс командной строки от имени администратора, где в вашем распоряжении будет Bash. В целом, имеет смысл использовать простой сеанс оболочки для восстановления, а не полный Рабочий стол с графическим интерфейсом. Это потому, что чем меньше сложных служб вы используете, тем выше вероятность того, что вы по крайней мере сможете запустить свою систему.

Как только вам удастся получить рабочую консоль, вы сможете начать поиски возможностей для обнаружения и устранения проблемы. По крайней мере, вы будете выглядеть очень круто, выполняя эти шаги.

6.1.3. Использование режима восстановления в CentOS

Меню GRUB в CentOS предлагает загрузить аварийное ядро, а не режим восстановления. Это ядро не включает в себя меню инструментов, как в Ubuntu, но оно аналогичным образом приведет вас к однопользовательской командной строке от имени администратора. На рис. 6.5 показан режим аварийной загрузки в CentOS GRUB.

Рис. 6.5. CentOS Linux предлагает аварийное ядро для загрузки непосредственно в однопользовательский консольный сеанс для устранения неполадок в поврежденной системе

Что будет после того, как вы выберете режим восстановления на вашем компьютере CentOS? В оставшейся части этой главы вы познакомитесь с некоторыми полезными инструментами. Но сначала почему бы вам не запустить режим восстановления Ubuntu и вручную не попробовать применить некоторые инструменты? «Легче сказать, чем сделать», — ответите вы. Как же работают эти инструменты? Далее я собираюсь показать вам, как применять ваши навыки по Bash и сценариям, чтобы во всем разобраться.

6.1.4. Поиск средств восстановления из командной строки

У вас есть машина, работающая под управлением операционной системы Ubuntu (она не будет работать под Debian)? Если код запускает меню, он должен находиться

где-то в файловой системе Ubuntu. Посмотрите сами. Используйте команду `locate`, чтобы найти его:

```
$ locate recovery-mode
/lib/recovery-mode
/lib/recovery-mode/l10n.sh
/lib/recovery-mode/options
/lib/recovery-mode/recovery-menu
/lib/recovery-mode/options/apt-snapshots
/lib/recovery-mode/options/clean
/lib/recovery-mode/options/dpkg
/lib/recovery-mode/options/failsafeX
/lib/recovery-mode/options/fsck
/lib/recovery-mode/options/grub
/lib/recovery-mode/options/network
/lib/recovery-mode/options/root
/lib/recovery-mode/options/system-summary
```

Сценарий l10n.sh устанавливает соответствующие переменные среды для меню

Файл сценария из меню восстановления

Сценарий для очистки диска

Если вы перейдете в каталог `/lib/recovery-mode/`, то увидите, что файл меню восстановления — это сценарий, который отображает интерфейс меню, показанный на рис. 6.3. Каталог `options/` содержит файлы для выполнения каждого из пунктов меню. Например, сценарий `fsck` проверит и по возможности исправит все поврежденные файловые системы.

Поскольку вы теперь опытный эксперт в области сценариев Bash, почему бы не просмотреть каждый сценарий в каталоге `options/`, чтобы узнать, сможете ли вы понять, как они работают? Ниже показано содержимое сценария `fsck`, с которого вы можете начать. Обратите внимание на то, как он хорошо задокументирован (с использованием символа `#`), чтобы помочь вам понять, что происходит:

```
$ cat /lib/recovery-mode/options/fsck
#!/bin/sh

. /lib/recovery-mode/l10n.sh

if [ "$1" = "test" ]; then
  echo $(eval_gettext "Check all file systems")
  exit 0
fi

# Actual code is in recovery-menu itself
exit 0
```

Сценарий l10n.sh вызывается для установки среды

Этот комментарий говорит вам, где происходит событие

Кое-что вы можете попробовать самостоятельно. Вручную запустите сценарий очистки диска на компьютере с Ubuntu. Что получилось? Затем попробуйте аккуратно отредактировать сценарий `/lib/recovery-mode/recovery-menu` (сначала сделайте его резервную копию). Измените что-нибудь простое, например заголовок меню или одно из описаний сценариев. Затем перезагрузите компьютер и в меню GRUB перейдите в режим восстановления, чтобы посмотреть, как выглядит среда восстановления. С некоторыми изменениями и исключениями вы сможете использовать эти сценарии и в других системах, в том числе в CentOS.

6.2. Создание загрузочного диска восстановления

Как вы, вероятно, уже знаете, образы операционной системы в формате ISO, которые вы использовали для своих виртуальных машин VirtualBox в главе 2, также могут быть записаны на CD или USB-накопители и задействованы для загрузки *текущего* сеанса ОС. Такие загрузочные диски/устройства позволяют загружать полнофункциональные сеансы Linux без необходимости устанавливать что-либо на жесткий диск. Многие люди используют такие диски/устройства, чтобы убедиться, что определенный дистрибутив Linux будет успешно работать на их оборудовании, прежде чем пытаться его установить. Другие проводят такие сеансы в режиме реального времени — это безопасный способ сохранения конфиденциальности при проведении таких важных операций, как онлайн-банкинг.

Оказывается, эти загрузочные диски также являются отличным инструментом для спасения и восстановления системы. Помните наш второй аварийный сценарий ранее в этой главе?

> *Ваш компьютер загружается (насколько вы можете об этом судить), но вы не совсем уверены, полнофункционален ли жесткий диск.*

Подключение загрузочного диска к проблемному компьютеру и запуск Linux со всеми его инструментами администрирования могут помочь вам понять, что на самом деле происходит, и дать вам инструменты для исправления многих проблем. Я покажу вам, как создать USB-накопитель с динамической загрузкой и как его использовать. Но сначала кратко рассмотрим некоторые из самых полезных образов, доступных в настоящее время.

6.2.1. Образы аварийного восстановления системы

Если у вас уже есть DVD или USB-накопитель с полноценной системой Linux (например, Ubuntu), то это будет самым простым вариантом, потому что большая часть необходимого вам программного обеспечения, которое вам понадобится, уже окажется предустановлена. Если у вас есть сетевое подключение, вы всегда сможете установить другие пакеты во время сеанса связи. Другими словами, в следующих пунктах описываются некоторые специальные образы.

Образ Boot-Repair

Если вы пытаетесь спасти систему Ubuntu, попробуйте образ Boot-Repair. Небольшой и быстрый дистрибутив Boot-Repair проверяет ваши настройки GRUB и при необходимости восстанавливает их. Как показано на рис. 6.6, он также может выполнять другие полезные задачи администрирования. Он очень часто спасал меня.

Образ Boot-Repair также можно установить на уже запущенный сеанс. На сайте help.ubuntu.com/community/Boot-Repair есть подробные инструкции.

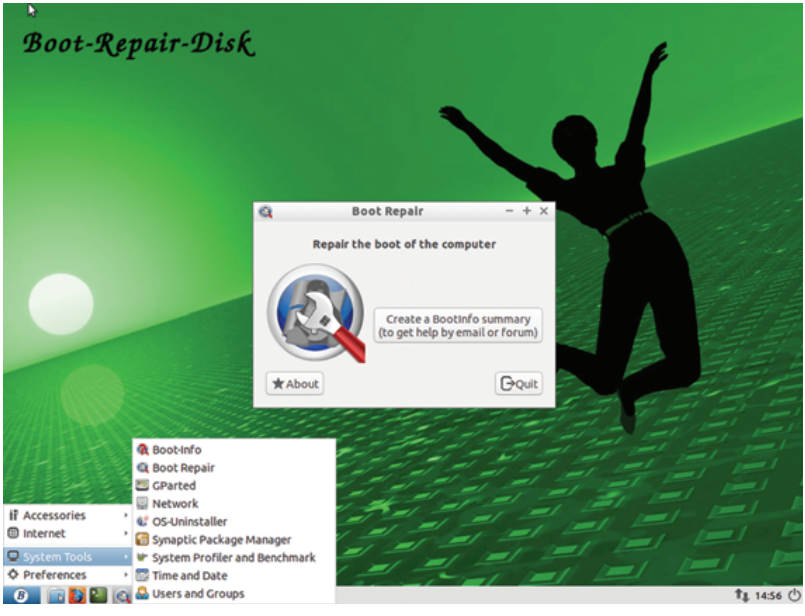


Рис. 6.6. Рабочий стол системы Boot-Repair с отображенным меню системных средств

Образ GParted Live

Если проблема связана с поврежденными разделами, которые могут нарушить ваши данные или помешать успешной загрузке, то образ GParted Live предоставит полнофункциональный и мощный редактор разделов GParted на CD или USB-накопителе. Как и Boot-Repair, образ GParted можно установить и использовать из любого обычного сеанса Linux. Если вы точно знаете, что проблема связана с разделами, полноценная загрузка живого образа может быть самым быстрым и прямым способом ее устранить. В отличие от Boot-Repair образ GParted создан для использования в любом дистрибутиве Linux, а не только в Ubuntu.

Образ SystemRescueCd

Другая альтернатива, SystemRescueCd, представляет собой облегченный образ, созданный на основе дистрибутива Gentoo Linux. Поскольку SystemRescueCd поставляется с множеством полезных инструментов для восстановления системы, он отлично подходит для USB-носителей, которые можно носить с собой. Компакт-диск также годится, хотя он не так легко помещается в задний карман и садиться нужно аккуратно.

6.2.2. Запись образов на загрузочные USB-накопители

Прежде чем записать любой ISO-образ, вам необходимо его скачать. Я могу смело предположить, что вы в состоянии найти нужную страницу на сайте дистрибутива, которая вам требуется для этого. Но вы можете не знать, что большие файлы иногда могут повреждаться во время их перемещения по Интернету. Хуже того: они могут быть заражены вредоносным ПО. Подобное уже случалось с некоторыми людьми, скачивавшими Mint Linux некоторое время назад.

Помимо проверки того, что сайт, на котором вы находитесь, должным образом зашифрован (ваш браузер должен показывать значок блокировки в адресной строке), вам обязательно нужно создать хеш для загруженного файла и сравнить его с хешем, предоставленным сайтом. (Хеши обсуждались еще в главе 2.)

Хеши иногда отображаются прямо на странице загрузки образов, но некоторые дистрибутивы затрудняют их поиск. Существует страница со ссылками на все его хеши Ubuntu (**help.ubuntu.com/community/UbuntuHashes**). На рис. 6.7 показано, как будут выглядеть хеши для Ubuntu 17.04.

Как только вы найдете опубликованные хеши для своего образа, вы вычислите хеш для загруженного файла и сравните два значения. В этом примере создается хеш SHA256 для образа SystemRescueCd из того же каталога, в который он был загружен:

```
$ cd Downloads
$ ls | grep systemrescue
systemrescuecd-x86-5.0.2.iso
$ sha256sum systemrescuecd-x86-5.0.2.iso
a2abdaf5750b09886cedcc5233d91ad3d1083e10380e555c7ca508
   49befbf487  systemrescuecd-x86-5.0.2.iso
```

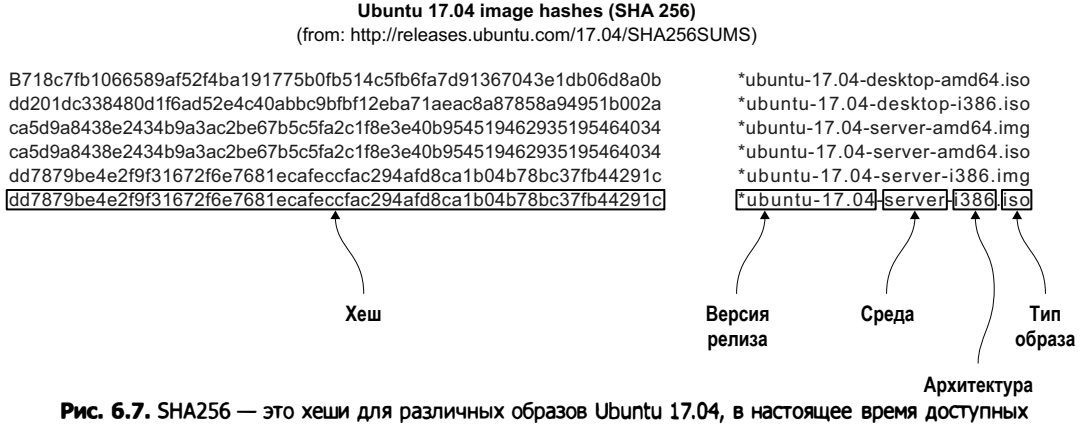


Рис. 6.7. SHA256 — это хеши для различных образов Ubuntu 17.04, в настоящее время доступных для скачивания

Если тот или иной образ надежно загружен, можно переходить к созданию загрузочного диска для восстановления. Если ваш текущий компьютер и восстанавливающий

USB-накопитель, который вы создадите, будет работать на Debian, Ubuntu или его производном, то проще всего использовать инструмент Ubuntu Startup Disk Creator.

Программа Ubuntu Startup Disk Creator доступна в меню графического интерфейса и, как видно из рис. 6.8, проста в использовании. Выберите на своем жестком диске ISO-образ и целевой накопитель (USB или CD), куда образ должен быть записан. Программа Ubuntu Startup Disk Creator позаботится обо всем остальном.

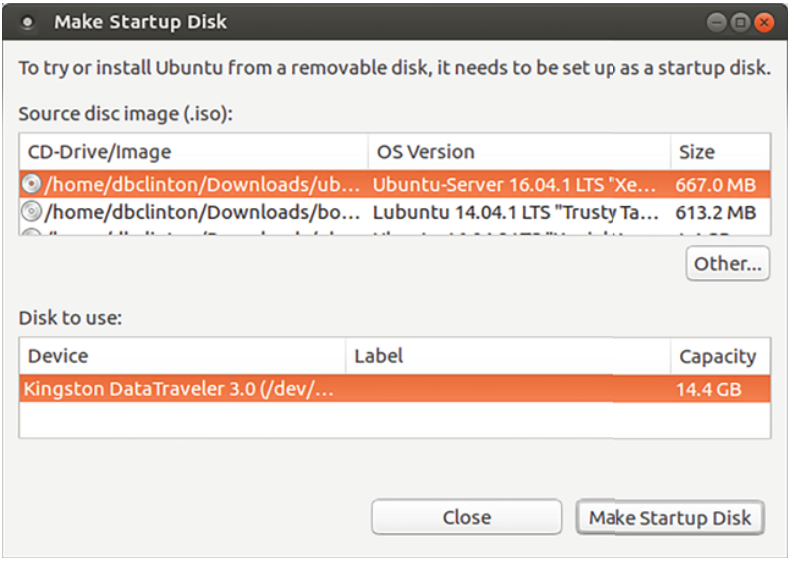


Рис. 6.8. Программа Ubuntu Startup Disk Creator с выбранным образом Ubuntu Server 16.04 и целевым USB-накопителем

С другой стороны, если вам требуется загрузочное устройство, созданное из другого дистрибутива (если вы пытаетесь спасти машину, работавшую под управлением операционной системы CentOS, и хотите использовать для этого инструменты CentOS), то вам придется обратиться к старой доброй команде `dd`. Если вам понадобится загрузочный диск на CD или DVD, запустите `dd` на любом другом хосте Linux. Но если образ будет записан на USB-накопитель и при этом вы работаете на хосте Ubuntu, вам сначала потребуется изменить образ, добавив MBR в ISO-образ, чтобы прошивки BIOS и UEFI знали, что делать дальше. Рисунок 6.9 иллюстрирует этот процесс.

На хостах Ubuntu для этой модификации образа вы используете утилиту isohybrid. Пакет apt, содержащий isohybrid, называется syslinux-utils. После установки перейдите в каталог, содержащий загруженный образ, и выполните команду `isohybrid`, указав имя файла образа в качестве единственного аргумента:

```
# apt update
# apt install syslinux-utils
$ cd ~/Downloads
$ isohybrid systemrescuecd-x86-5.0.2.iso
```

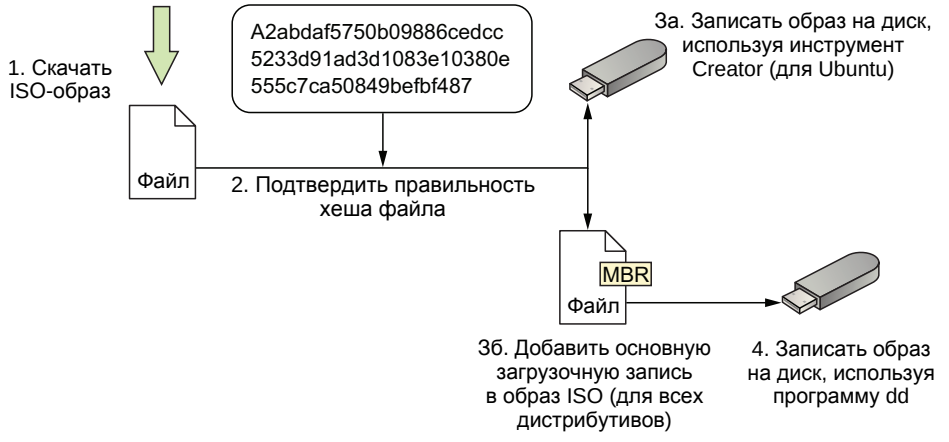


Рис. 6.9. Шаги, необходимые для записи рабочего образа на загрузочный USB-накопитель

Следующие шаги актуальны независимо от того, какой дистрибутив вы используете. Тщательно определите системное обозначение вашего целевого устройства. Как вы помните, команда `df` выводит все распознанные в данный момент файловые системы вместе с их обозначениями:

```
$ df -h
Filesystem Size Used Avail Use% Mounted on
udev       3.5G    0 3.5G   0% /dev
tmpfs      724M 1.5M 722M   1% /run
/dev/sda2  910G 183G 681G 22% /
tmpfs      3.6G 214M 3.4G   6% /dev/shm
tmpfs      5.0M 4.0K 5.0M   1% /run/lock
tmpfs      3.6G    0 3.6G   0% /sys/fs/cgroup
/dev/sda1  511M 3.4M 508M   1% /boot/efi
tmpfs      724M  92K 724M   1% /run/user/1000
/dev/sdb1   15G   16K  15G   1% /media/myname/KINGSTON
```

Это моя корневая файловая система. Я точно не хочу ее переписывать!

Файловая система на моем съемном USB-накопителе

В этом примере на моем USB-устройстве Kingston смонтирована файловая система с именем `/dev/sdb1`. Это говорит о том, что само устройство известно как `/dev/sdb`.

Если вы планируете записать образ на оптический CD или DVD, то получите его обозначение с помощью команды `lsblk`, расшифровываемой как *список блочных устройств*. Сам диск должен быть доступен для записи. В листинге ниже мой DVD-привод обозначен как `sr0`:

```
$ lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0 931.5G  0 disk
├─sda1   8:1    0   512M  0 part /boot/efi
├─sda2   8:2    0 923.8G  0 part /
└─sda3   8:3    0   7.2G  0 part [SWAP]
sdb      8:16   1  14.4G  0 disk
└─sdb1   8:17   1  14.4G  0 part /media/clinton/KINGSTON
sr0      11:0   1  1024M  0 rom
```

DVD-ROM с возможностью записи

Момент истины. Больше нельзя его откладывать. Пришло время выполнить команду dd в вашей плохой, беззащитной файловой системе и записать образ на USB-накопитель.

Сначала отмонтируйте сам диск, чтобы dd могла получить полный доступ. Затем запишите архив. В этом примере я использовал образ systemrescuecd-x86-5.0.2.iso и записал его на диск по адресу /dev/sdb.

Осторожно! Ввод sda вместо sdb (в данном конкретном случае) безвозвратно перезапишет вашу хостовую файловую систему и разрушит не только вашу сегодняшнюю работу, но и всю карьеру. Естественно, вы также должны убедиться, что на USB-накопителе нет ничего важного, так как все будет удалено:

```
# umount /dev/sdb
# dd bs=4M if=systemrescuecd-x86-5.0.2.iso \
    of=/dev/sdb && sync
```

Добавленная команда синхронизации гарантирует, что все кэшированные данные будут немедленно записаны на целевой диск

Команде dd может потребоваться некоторое время, чтобы завершить запись образа на USB-устройство, но, когда все будет сделано, вы сможете подключить диск к компьютеру, запустить его и начать сеанс в реальном времени. Предполагается, что ваш компьютер настроен для загрузки с USB-накопителя. Если это не так, вы один раз сможете принудительно загрузить выбранное устройство, войдя в меню Boot (Загрузка) во время процесса загрузки. Каждый производитель материнской платы назначает свои собственные сочетания клавиш для настройки загрузки (часто выводя нужное сочетание клавиш в процессе запуска компьютера). Но, как правило, для перехода в меню настройки загрузки предназначена клавиша F1, F9 или F12.

Вы также можете настроить системную прошивку (BIOS или UEFI), чтобы определить порядок загрузки устройств. Доступ к утилите настройки можно также получить при нажатии клавиш F2, F10, Delete или даже Enter.

6.3. Запуск загрузочного диска для работы

С помощью созданного вами накопителя с карманной версией Linux вы можете сделать многое. В следующих подразделах описывается несколько распространенных проблем и способы их решения.

6.3.1. Тестирование системной памяти

Если вы столкнулись с внезапными сбоями системы, то, вероятно, в них виновата память (ОЗУ). Как и все оборудование, оперативная память в конечном итоге выходит из строя. Проблема в том, что вы не сможете должным образом проверить ее на наличие ошибок, пока она используется работающей ОС. Это нужно делать до

загрузки ОС. К счастью, будучи счастливым обладателем загрузочного образа Linux, вы уже имеете все необходимое для этой процедуры. Как показано на рис. 6.10, одним из пунктов меню, которое будет отображаться после загрузки Ubuntu с диска, является Test memory (Проверка памяти).

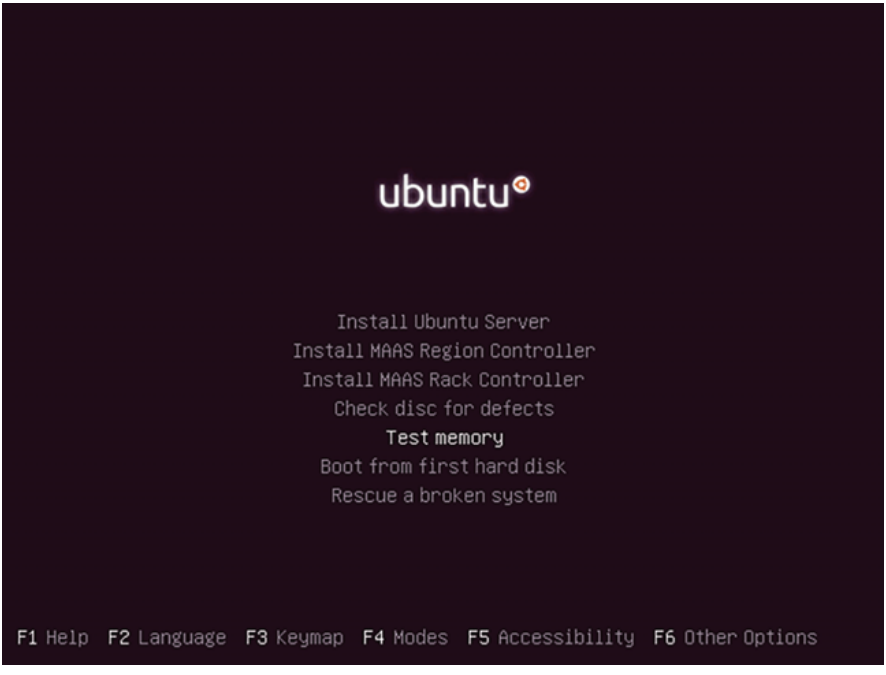


Рис. 6.10. Главное меню процесса установки Ubuntu с выделенным пунктом Test memory (Проверка памяти)

Выбор пункта Test memory (Проверка памяти) (нажатием клавиш ↑ и ↓ и затем Enter) откроет программу Memtest86+ (рис. 6.11). Эта программа выполняет многократное сканирование оперативной памяти, отображая все найденные ошибки. Честно говоря, я не уверен, что процесс сканирования когда-нибудь остановится сам по себе: по крайней мере я никогда не мог этого дождаться. Но если после многочисленных попыток запуска сканирования Memtest86+ не выдает никаких ошибок, то, вероятно, оперативная память не является источником ваших проблем. С одной стороны это хорошая новость, с другой — плохая... вы ведь так и не нашли причину сбоя.

ПРИМЕЧАНИЕ

Помимо проверки памяти, пункт Rescue a broken system (Восстановление системы после сбоя) в главном меню процесса установки Ubuntu предоставит вам рабочую оболочку Bash.

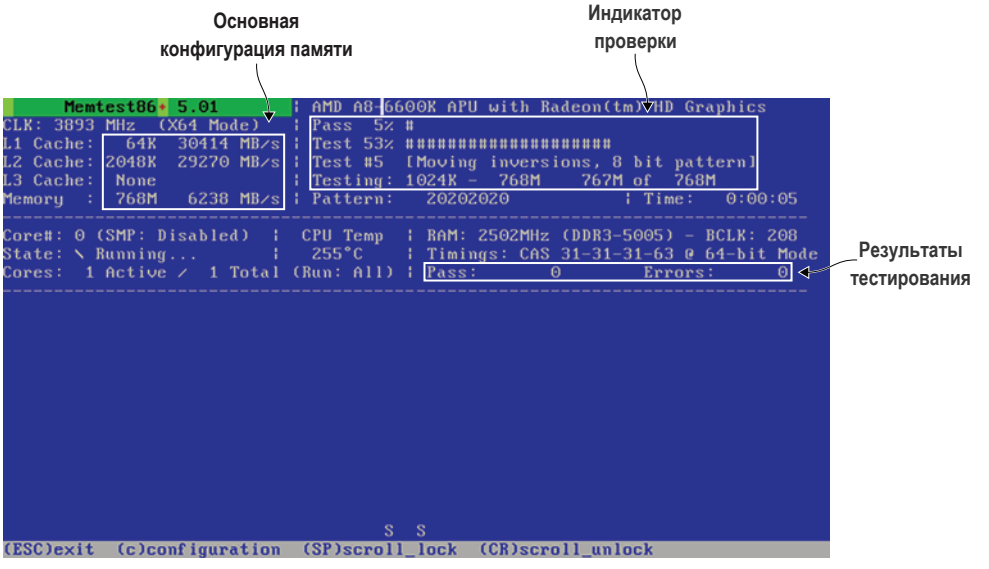


Рис. 6.11. Программа Memtest86+ показывает местоположение и вид любых ошибок вашей оперативной памяти. Все чисто (на данный момент)

Установочные диски CentOS имеют собственную ссылку на Memtest86+ в меню поиска и устранения неисправностей (рис. 6.12).

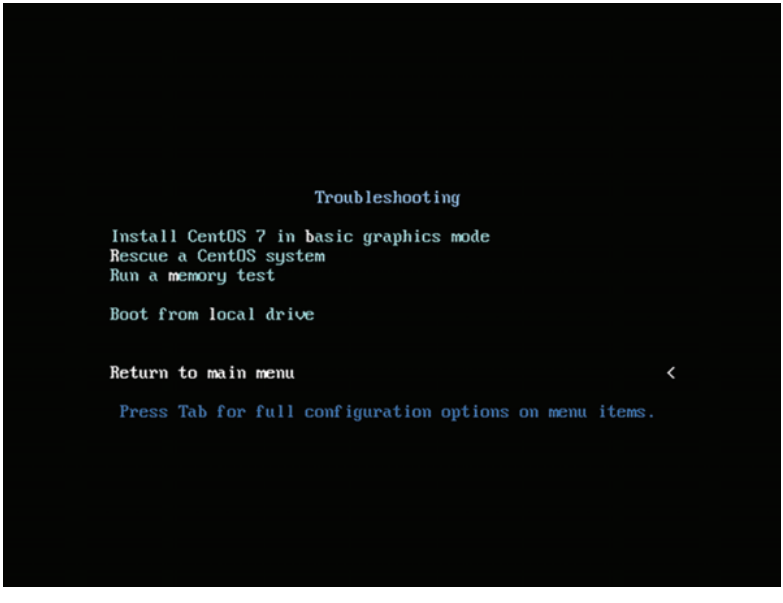


Рис. 6.12. Меню устранения неполадок, связанное с главным меню на установочном диске CentOS

6.3.2. Поврежденные разделы

Раздел представляет собой метаданные, указывающие на области на физическом диске, занимаемые файловой системой. Эти разделы — очень хрупкая вещь, и кажется, что их очень легко разрушить. Если данные на диске каким-либо образом повреждены, а точные адреса начальной и конечной точек раздела изменены или утеряны, файловая система всего раздела станет недоступна. А если файловая система недоступна, то данные в файловой системе можно считать утерянными.

Не удается получить доступ к разделу? Пришло время загрузить ваш новый диск SystemRescue. SystemRescue — это небольшой пакет, поэтому не ожидайте от него тех же действий, что и от полнофункционального дистрибутива. Могут давать сбои такие операции, как функционирование тачпада ноутбука и автоматическое подключение к Wi-Fi. Однако это быстрый способ ввести в игру несколько мощных спасательных инструментов. Как видно на рис. 6.13, параметр загрузки по умолчанию откроет специальную оболочку с командами.

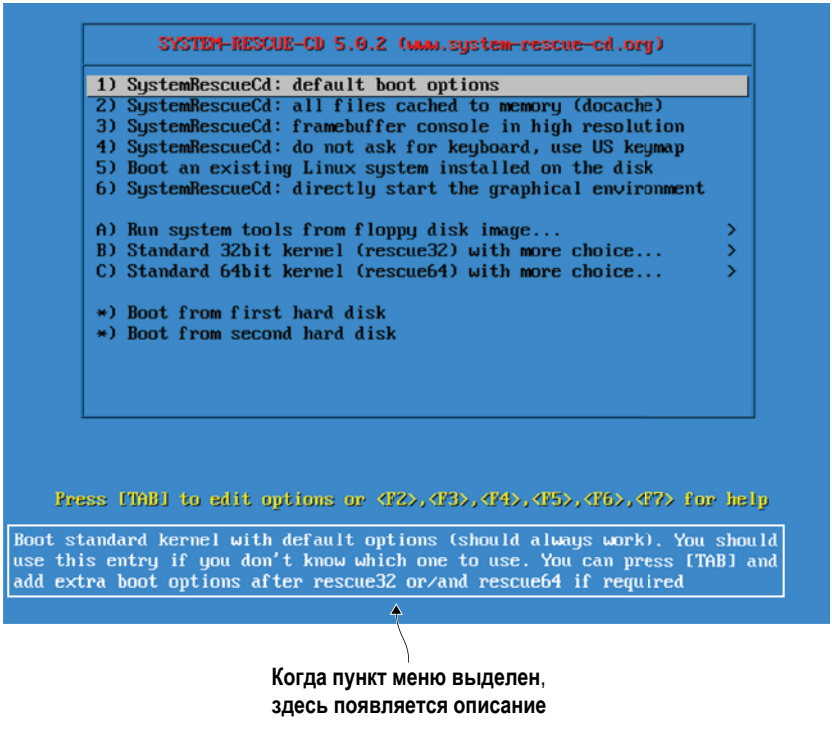


Рис. 6.13. Меню параметров загрузки, отображаемое при запуске SystemRescue. Обратите внимание на подробные объяснения для каждого параметра внизу страницы

По мере того как вы готовитесь к реабилитации поврежденного раздела (или его данных), SystemRescue вместе с командной строкой показывает различную полезную

информацию для работы по разным направлениям, включая основы работы с сетью и редактирование текста (рис. 6.14). Не беспокойтесь: Vim и Nano доступны из командной строки. А ввод команды `startx` позволит загрузить быстрый и легкий графический интерфейс Рабочего стола.

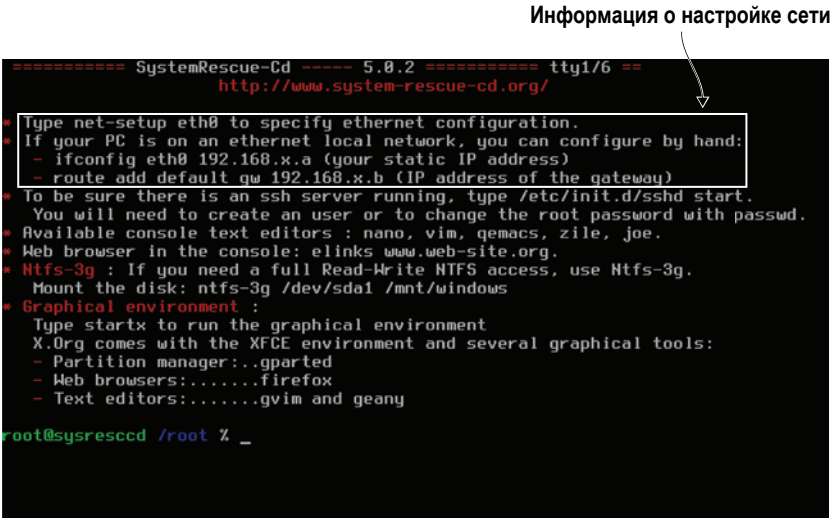


Рис. 6.14. Оболочка SystemRescue. Обратите внимание на текстовые редакторы, доступные по умолчанию (включая Nano), а также на то, что при вводе startx запускается графический интерфейс Рабочего стола

Если вам понадобится доступ к сети для загрузки или установки дополнительных инструментов или для передачи данных, введите в оболочке командной строки команду `net-setup`, выберите нужный интерфейс и укажите, является ли ваша сеть проводной или беспроводной. Если сеть беспроводная, введите SSID вашей сети и пароль (в котором, вероятнее всего, будет использоваться кодировка ASCII, а не шестнадцатеричные символы). В большинстве случаев необходимо разрешить протоколу DHCP автоматически искать сеть.

Если у вас проблемы с поврежденным разделом, ваша главная задача сейчас — восстановление. Если существует вероятность выхода из строя жесткого диска, тогда вашей первостепенной задачей должно быть обеспечение безопасности его данных. Для этого я бы использовал команду `dd` в оболочке командной строки SystemRescue, чтобы создать идеальную копию раздела в его текущем состоянии и сохранить ее на исправном диске с таким же или бóльшим объемом памяти. После выполнения команды `lsblk` для подтверждения обоих разделов операция копирования может выглядеть примерно так (поврежденный диск — `/dev/sda`, а пустой раздел на новом диске — `/dev/sdc1`):

```
# dd if=/dev/sda of=/dev/sdc1
```

После этого вы можете сами проверить, сможете ли сохранить оригинальную копию. Введите команду `test-disk`. Вам будет задан вопрос о том, как вы хотите регистрировать события сеанса, какой диск хотите восстановить и какой тип раздела найти. Чаще всего программа TestDisk распознает правильный вариант и выделяет его как значение по умолчанию.

Затем перед вами будет экран, похожий на показанный на рис. 6.15, где вы сможете попросить TestDisk просканировать ваш диск в поисках существующих разделов. Обнаружив и соответствующим образом отметив поврежденные разделы, вы сможете записать изменения на диск и успешно загрузить компьютер.

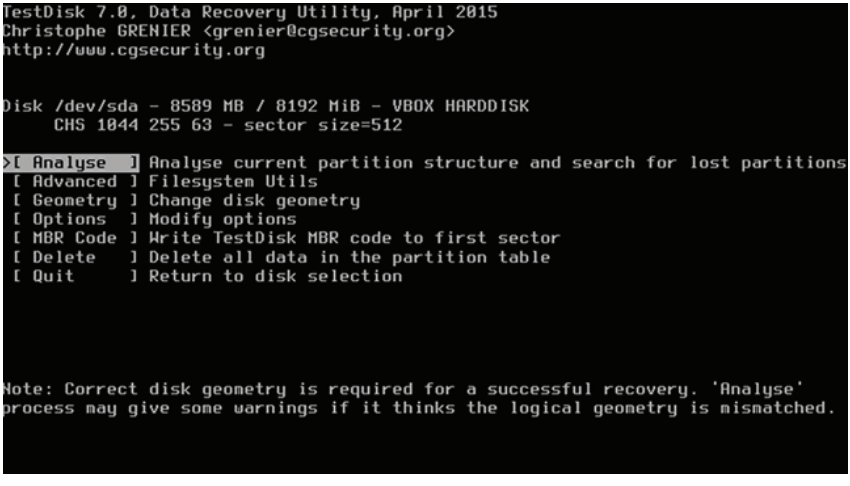


Рис. 6.15. Страница восстановления разделов TestDisk, на которой разделы, обнаруженные с помощью команды Analyze, можно редактировать и восстанавливать, используя различные инструменты

Подождите... и это все? А как насчет нюансов? Охватывает ли сложный процесс «обнаруженные и должным образом отмеченные поврежденные разделы» так, чтобы это было вообще полезно? Нет. Но тогда почему вы читаете эту главу? Я бы сказал, либо потому, что вы пытаетесь подготовиться к аварии в будущем, либо потому, что будущее уже наступило и вы восстанавливаете свой неработающий раздел.

Если вас просто беспокоит ваше будущее и наблюдение за мной во время работы над одним или двумя гипотетическими решениями, вряд ли вам пригодится эта информация. Кроме того, я не собираюсь советовать вам намеренно портить реальный раздел, чтобы вы могли проверить инструмент TestDisk в деле. А если у вас есть реальная проблема, то маловероятно, что тот пример, который я выберу, будет вам полезен. В таком случае лучшее, что я могу сделать, — сообщить вам, что существует подходящее программное обеспечение, и научить его запускать. Ах да, есть еще довольно хорошая документация, доступная по адресу cgsecurity.org/testdisk.pdf.

6.3.3. Восстановление файлов из поврежденной файловой системы

Если окажется, что вы не можете полностью исправить свой диск, возможно, вам удастся восстановить некоторые разделы и обеспечить доступ к файлам, хотя этого и недостаточно для надежной загрузки с диска. Тогда ваш новый приоритет — извлечь как можно больше важных файлов.

Самый простой подход — использовать обычные инструменты управления файловой системой Linux из любого загрузочного сеанса. Если вы не уверены в обозначении раздела, выполните команду `lsblk`, чтобы быстро в этом разобраться.

Возможно, ваш раздел пока не будет доступен в виде файловой системы, поскольку он еще не смонтирован. На самом деле, если он не отображается в выводе команды `df`, вам придется *монтировать* его вручную, то есть назначить ему местоположение в файловой системе, где к нему можно получить доступ. Вы можете исправить это достаточно быстро, создав новый каталог, к которому будет примонтирован ваш раздел, а затем использовав команду `mount`. Я решил создать свой каталог временных точек монтирования в `/run/`, но подойдет и любое другое незадействованное расположение (например, `/media/` или `/mnt/`). Предположим, как и прежде, что раздел называется `/dev/sdc1`:

```
# mkdir /run/temp-directory
# mount /dev/sdc1 /run/temp-directory
```

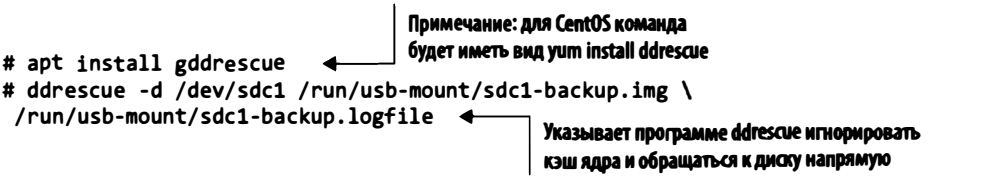
С этого момента любые исправные файлы из поврежденной файловой системы будут доступны для копирования или редактирования из временного каталога. Возможно, они также автоматически появятся в любом графическом менеджере файлов, который вы захотите использовать.

Восстановление с помощью инструмента ddrescue

Все равно ничего не получается? Пора воспользоваться тяжелой артиллерией. Инструмент восстановления данных ddrescue копирует файлы между файловыми системами. Но в то же время он делает кое-то еще, что наверняка порадует вас: анализирует ваши файлы и пытается восстановить любые повреждения. Ни один инструмент не гарантирует, что исправит все, что повреждено, но репутация программы ddrescue не дает усомниться в ее способностях.

Если инструмент ddrescue не установлен на используемом вами загрузочном диске, знайте, что он по умолчанию есть в SystemRescue. Извлеките его из хранилища. Затем определите проблемный раздел (в данном примере `/dev/sdc1`), в котором вы хотите сохранить образ, а также имя и местоположение файла журнала, в который можно внести запись событий операции. Наличие файла журнала позволяет программе ddrescue продолжить остановленную операцию, а не начинать процесс вос-

становления с нуля. В моем примере используется внешний диск (большей емкости, чем исходный), примонтированный к каталогу `/run/usb-mount`:



Можно протестировать восстановление, используя команду `dd` для записи образа резервной копии на новый пустой диск (в моем примере он называется `/dev/sdd/`), а затем загрузить систему на новый диск:

```
# dd if=backup.img of=/dev/sdd
```

Даже если вы обнаружите, что по-прежнему не можете получить доступ к разделу, то попробуйте обратиться к его важным отдельным файлам. В любом случае вы уже намного продвинулись в решении своих проблем.

Восстановление файлов с помощью инструмента PhotoRec

PhotoRec — еще один инструмент, который поможет вам извлечь файлы с поврежденного диска. Только взглянув на его интерфейс, вы поймете, что у PhotoRec и TestDisk много общего. Да, фактически оба созданы и поддерживаются Кристофом Гренье из компании CGSecurity.

Чтобы загрузить программу, введите:

```
# photorec
```

После обнаружения поврежденной файловой системы, ее файлов и места, в котором вы хотите их сохранить, все файлы будут сохранены в пронумерованных каталогах с использованием префикса `recup_dir.?`:

```
$ ls recup_dir.12
f1092680.elf
f0668624.png
f0853304.xml
f0859464.xml
f0867192.txt
f0977016.gz
f0982184.xml
[...]
```

Что нужно помнить о PhotoRec: всем файлам присваиваются новые пронумерованные имена при сохранении их исходных расширений. Иногда это может затруднить поиск отдельных файлов, но такой вариант, безусловно, лучше, чем полная потеря информации.

6.4. Восстановление пароля: монтирование файловой системы с помощью инструмента chroot

Пароли, выбранные сотрудниками, машины которых вы обслуживаете, вероятнее всего, недостаточно надежны для защиты всей инфраструктуры от серьезных атак. Даже при наличии нескольких исключений из общего правила эти пароли, возможно, повторно используются при доступе к нескольким серверам и учетным записям. Ты их умоляешь и ворчишь, умоляешь и ворчишь, но людям как об стенку горох. Однако еще не все потеряно.

Проблема отслеживания достаточно сложных паролей может быть в значительной степени упрощена при наличии хорошего хранилища паролей, такого как KeePass2 или LastPass. А проблема постоянного использования повторяющихся паролей может быть решена путем внедрения решения с единым входом, такого как Kerberos. Тем не менее ворчать тоже иногда придется.

Что происходит с пользователями, заботящимися о том, чтобы придумать хорошие и надежные пароли для каждого сервера, к которому они обращаются? Время от времени они забывают свои пароли. Это не проблема, если есть другой администратор с полномочиями `sudo`, который может войти на сервер и запустить команду `passwd`, чтобы создать новый пароль для пользователя:

```
# passwd username
[sudo] password for yourname:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Но если ваш незадачливый и забывчивый пользователь был единственным администратором с учетной записью на данном компьютере, у вас проблемы. Хотя нет: прадед всех виртуализаций Linux — инструмент `chroot` — спасет ваш день. Вот один рабочий способ.

Используйте загрузочный диск для включения заблокированного сервера, выполните команду `lsblk` для определения корневого раздела на жестком диске сервера и примонтируйте корневой раздел к временному каталогу:

```
# mkdir /run/mountdir/
# mount /dev/sdb1 /run/mountdir/
```

Затем прошепчите волшебные слова, и вы попадете в:

```
# chroot /run/mountdir/
root@ubuntu:/#
```

Это все, что от вас требуется. На данном этапе вы можете выполнять команды, как если бы работали над запущенной версией жесткого диска. Используйте команду `passwd`, чтобы задать вашему администратору новый пароль для замены утерянного. После ввода команды `exit`, чтобы завершить сеанс `chroot`, перезагрузите компьютер (уже без загрузочного USB-накопителя). Теперь все должно быть хорошо.

Шифровать или нет?

Шифрование данных на накопителях с помощью таких инструментов, как ecryptfs или dm-crypt, значительно снижает вероятность того, что ваши данные будут скомпрометированы. Но, с другой стороны, многие операции по спасению и восстановлению данных, описанные в этой главе, не будут работать на зашифрованном диске.

Сложно найти баланс между безопасностью и доступностью. Многие администраторы оставляют локальные серверы и рабочие компьютеры незапароленными, поскольку они как минимум защищены запертыми дверями офиса, однако в то же время настаивают на шифровании мобильных устройств.

Резюме

- Режимы восстановления Linux предоставляют доступ к инструментам администрирования, которые необходимы для восстановления поврежденных систем.
- Загрузочные диски позволяют загружать выбранные вами дистрибутивы Linux независимо от файловой системы на жестком диске компьютера.
- Дистрибутивы специального назначения, такие как SystemRescueCd, представляют собой облегченные версии Linux, которые поставляются с предустановленным полным набором инструментов для решения проблем.
- Поврежденные разделы иногда можно восстановить с помощью таких инструментов, как TestDisk.
- Данные из поврежденных разделов иногда можно восстановить с помощью таких инструментов, как ddrescue и PhotoRec.
- Файловые системы можно монтировать и администрировать с помощью виртуального процесса `chroot`.

Ключевые понятия

- *GRUB* — это загрузчик, который управляет образами, используемыми в процессе загрузки Linux.
- *Хеш (контрольная сумма)* — криптографически сгенерированное значение, которое можно проверить по основной копии, чтобы подтвердить подлинность образа.
- *Основная загрузочная запись раздела (MBR)* для CD/DVD будет отличаться от таковой для USB, что требует особого внимания при создании загрузочного диска на USB-накопителе.
- Инструмент `chroot` дает возможность применять виртуальные оболочки от имени администратора в примонтированных файловых системах.

Рекомендации по безопасности

- Всегда проверяйте подлинность загруженных образов, просматривая их хеши. И избегайте загрузки с незашифрованных (использующих протокол HTTP, а не HTTPS) сайтов.
- Тщательно обдумывайте, следует ли шифровать неиспользуемые данные на системных дисках, балансируя между доступностью к ним и безопасностью.
- Обязательно задействуйте хранилища паролей и службы единого входа для пользователей вашей инфраструктуры.

Обзор команд

- `sha256sum systemrescuecd-x86-5.0.2.iso` вычисляет контрольную сумму SHA256 файла в формате ISO.
- `isohybrid systemrescuecd-x86-5.0.2.iso` добавляет запись MBR с поддержкой USB-носителей к образу загрузочного диска.
- `dd bs=4M if=systemrescuecd-x86-5.0.2.iso of=/dev/sdb && sync` записывает образ загрузочного диска на пустой носитель.
- `mount /dev/sdc1 /run/temp-directory` монтирует раздел в каталог в загрузочной файловой системе.
- `ddrescue -d /dev/sdc1 /run/usb-mount/sdc1-backup.img /run/usb-mount/ sdc1-backup.logfile` сохраняет файлы из поврежденного раздела в образ с именем `sdc1-backup.img` и записывает события в файл журнала.
- `chroot /run/mountdir/` открывает оболочку от имени администратора в файловой системе.

Самотестирование

1. Что из нижеперечисленного позволит вам получить доступ к режиму восстановления поврежденной машины Linux:
   а) SystemRecovery;
   б) GRUB;
   в) нажатие сочетания клавиш Ctrl+Shift во время загрузки;
   г) chkdsk?
2. Какая учетная запись пользователя применяется по умолчанию для сеанса режима восстановления Linux:
   а) root;
   б) начальная учетная запись, созданная во время установки;
   в) admin;
   г) guest?

3. Какова цель пункта Clean (Очистить) в меню режима восстановления Ubuntu:
   а) поиск и удаление вирусов;
   б) удаление неиспользуемых программных файлов;
   в) удаление неиспользуемых учетных записей;
   г) очистка сеанса и завершение работы?
4. К чему из нижеперечисленного приведет загрузочный диск Linux:
   а) запуск среды восстановления системы;
   б) запуск среды восстановления файлов;
   в) запуск сеанса Linux с использованием разделов хост-компьютера;
   г) запуск сеанса Linux с использованием только раздела с начальной загрузкой?
5. Какой инструмент, необходимый для создания образа загрузочного диска, можно использовать для загрузки с USB-устройства:
   а) ddrescue;
   б) GRUB;
   в) isohybrid;
   г) ecryptfs?
6. Для каких целей предназначена программа sha256sum:
   а) открытие сеанса виртуальной оболочки в смонтированной файловой системе;
   б) управление образами Linux при запуске системы;
   в) создание хешей для проверки подлинности файлов;
   г) управление восстановлением файлов с поврежденных дисков?
7. Какая команда запишет образ SystemRescue на второе блочное устройство в вашей системе:
   а) `dd if=systemrescuecd-x86-5.0.2.iso of=/dev/sdb`;
   б) `dd if=/dev/sdb of=systemrescuecd-x86-5.0.2.iso`;
   в) `dd if=/dev/sdc of=systemrescuecd-x86-5.0.2.iso`;
   г) `dd if=systemrescuecd-x86-5.0.2.iso of=/dev/sdb2`?
8. Что делает команда `chroot`:
   а) создает образ для загрузочного диска, который можно использовать для загрузки с USB-устройства;
   б) открывает сеанс виртуальной оболочки в смонтированной файловой системе;
   в) генерирует хеши для проверки целостности файла;
   г) удаляет неиспользуемые программные файлы?

Ответы

1 — б; 2 — а; 3 — б; 4 — г; 5 — в; 6 — в; 7 — а; 8 — б.

7 Веб-серверы: создание сервера MediaWiki

В этой главе

- Создание динамических веб-серверов Apache.
- Управление данными серверных приложений с помощью баз данных SQL.
- Выявление и устранение зависимостей пакета приложений.
- Установка и настройка движка MediaWiki.

Допустим, вы ведете небольшой корпоративный блог для публикации технических или корпоративных данных за последние 30 лет, насчитывающих 100 тысяч страниц. Вам просто необходима некая система управления контентом (CMS). Думаю, вам известно, что CMS — это программное обеспечение, разработанное в качестве основы для создания и администрирования цифрового контента. К одним из самых известных систем CMS, с которыми вы могли столкнуться, относятся WordPress и Joomla.

Вики-проекты предоставляют особенно эффективный способ управления большими сообществами участников и авторов. Это своего рода система CMS, архитектура которой преднамеренно децентрализована, что позволяет пользователям свободно работать не только над самим контентом, но и над всей структурой сбора данных. *Вики-движок* — это платформа, на которой создаются вики-проекты с использованием простого и интуитивно понятного языка разметки. MediaWiki — популярный пример вики-движка с открытым исходным кодом, а Atlassian Confluence — уже устоявшаяся коммерческая альтернатива.

Сам по себе движок MediaWiki сейчас может быть не так уж важен для нас. Я решил сосредоточиться только на его установке, потому что это отличный способ проиллюстрировать процесс создания веб-сервера (часто называемого *сервером LAMP*) в Linux. Это важно еще потому, что сегодня два из трех веб-серверов в Интернете работают под управлением операционной системы Linux.

Не поймите меня неправильно: у движка MediaWiki нет недостатков. В конце концов, это система CMS, которую изначально создали для поддержки десятков миллионов статей, входящих в «Википедию» и другие проекты Фонда «Викимедиа». Если вы ищете надежный способ управления большим количеством медиаконтента, будь то частные коллекции, задокументированные процессы вашей компании или общедоступные справочные страницы, вы не найдете ничего лучше MediaWiki. Но эти конкретные случаи использования не настолько распространены, чтобы подробно писать о движке MediaWiki в книге о базовых навыках работы с Linux.

Однако, поработав с движком MediaWiki, вы узнаете о пакетах программного обеспечения, которые составляют веб-сервер Linux, и о том, как они сочетаются друг с другом, чтобы поддерживать в рабочем состоянии бóльшую часть сайтов. Будучи администратором Linux, вы наверняка будете создавать веб-серверы для поддержки всех видов приложений, поэтому вы же заинтересованы в том, чтобы уметь это делать, верно?

7.1. Создание сервера LAMP

Если у вас или вашей компании есть некая интересная информация, приложения или услуги, высока вероятность того, что вы захотите сделать их доступными для других пользователей. *Веб-сервер* — это программное обеспечение, работающее на компьютере и позволяющее посетителям сайтов просматривать и использовать локально размещенные ресурсы. Чтобы было еще понятнее, термин «веб-сервер» также часто используется для описания компьютера, на котором размещено программное обеспечение веб-сервера.

Статические или динамические сайты?

Данный раздел поможет вам в процессе создания *динамического* сайта. Это сайт, страницы которого создаются с помощью серверных операций. Также возможно создание *статических* сайтов, которые по большей части представляют собой простые HTML-файлы, делегирующие всю работу клиентскому браузеру.

Статический вариант лучше использовать для более простых сайтов. Но, поскольку сейчас речь пойдет уже не о Linux, я закрою эту тему. С другой стороны, в главе 6 моей книги *Learn Linux in a Month of Lunches* (Manning, 2017) есть замечательный пример использования AWS S3 для размещения статического сайта.

Как показано на рис. 7.1, большинство веб-серверов Linux построены на базе четырех принципов так называемого сервера LAMP. Аббревиатура *LAMP* подразумевает четыре компонента: Linux, программу администрирования веб-сервера Apache, ядро базы данных MySQL или MariaDB, а также язык сценариев на стороне сервера PHP (варианты: Perl или Python). Помимо Linux, с которой, я надеюсь, вы уже знакомы, в этой главе мы обсудим оставшиеся компоненты.

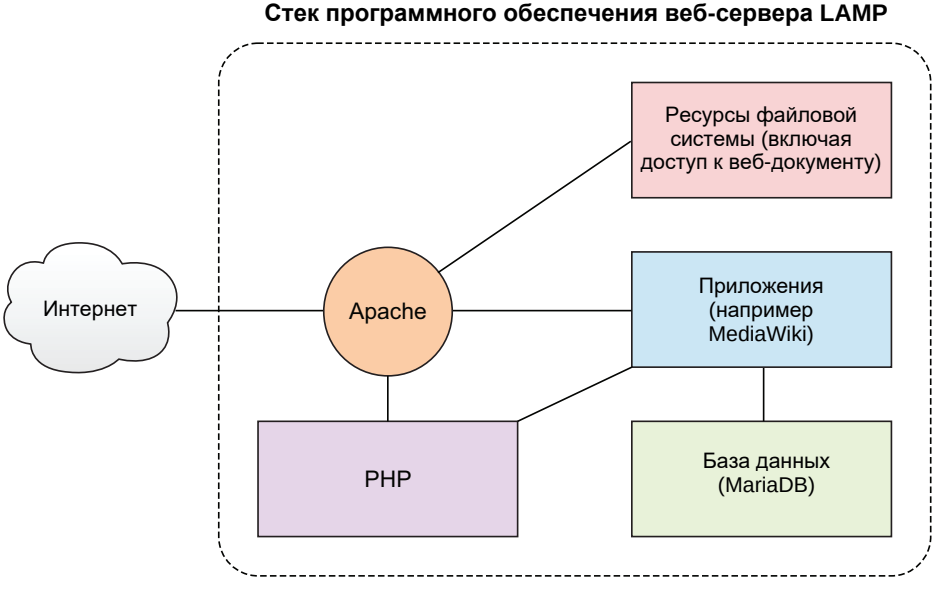


Рис. 7.1. Программное обеспечение веб-сервера Apache предоставляет внешние ресурсы на основе HTTP и координирует внутренние службы

Сервер LAMP — настолько распространенная конфигурация Linux, что даже Ubuntu имеет собственный установочный метапакет. Знак вставки (^) в конце следующего примера определяет цель как специальный пакет, собранный воедино, чтобы упростить установку общих программных стеков:

```
# apt install lamp-server^
```

Эта команда запрашивает создание пароля для базы данных и автоматически создает работающий веб-сервер для вашей системы, а вам остается лишь подготовить контент для сайта. Перейдя в браузере по IP-адресу сервера, вы увидите страницу приветствия, созданную при установке Apache.

Но автоматизация не всегда лучшее решение. В какой-то момент вам захочется настроить свой стек программного обеспечения, указав конкретные версии выпуска для обеспечения совместимости приложений или заменив один пакет другим (например, вскоре будет пример MariaDB поверх MySQL). Ручная настройка будет особенно полезна для того, чтобы лучше понять, как работает каждый бит. Именно

такой подход я буду применять в этой главе. Вот перечень действий, которые нужно выполнить, чтобы вы достигли своей цели.

1. Установить Apache.
2. Добавить одну или две веб-страницы в корневой каталог веб-документа.
3. Установить движок SQL (в данном случае MariaDB).
4. Установить серверный язык сценариев PHP.
5. Установить и настроить MediaWiki.

7.2. Настройка веб-сервера Apache вручную

У программного обеспечения веб-сервера есть одна основная задача — направлять посетителей сайта к нужным каталогам и файлам на хосте сервера, поэтому должны быть доступны соответствующие ресурсы сайта. Иначе говоря, ввод единого указателя ресурсов (URL) в адресной строке браузера — это запрос программного обеспечения веб-сервера, запущенного на удаленном хосте сайта, для извлечения веб-страницы, видео или другого ресурса из файловой системы хоста и загрузки его в вашем браузере. Программное обеспечение веб-сервера, как правило, интегрировано с другими системами на главном сервере, такими как сетевые инструменты, средства безопасности и инструменты файловой системы, так что доступ к локальным ресурсам хорошо управляем.

HTTP-сервер Apache с открытым исходным кодом, как правило, доминирует на этом нестабильном рынке веб-серверов на всех платформах. Несмотря на то что у Apache есть серьезные конкуренты, например Nginx (также кросс-платформенный) и IIS Microsoft (работает исключительно на серверах Windows), я собираюсь придерживаться Apache, потому что он очень популярен. (Да и вообще, вы действительно думаете, что я бы посвятил IIS целую главу?)

7.2.1. Установка веб-сервера Apache на Ubuntu

Установка Apache сама по себе проста. В среде Debian/Ubuntu это делается с помощью команды `apt intall apache2`. Если вы всегда вовремя обновляете компьютер с Ubuntu, то после установки Apache ничто не помешает вам открыть браузер и сразу же посетить ваш действующий сайт. Вы увидите вводную страницу (рис. 7.2).

ПРИМЕЧАНИЕ

URL-адрес, который вы будете использовать для доступа к сайту Apache, работающему на вашей рабочей станции, называется localhost. Если вместо этого вы решили поработать через контейнер LXC или с виртуальной машиной VirtualBox, то в качестве URL указывайте IP-адрес компьютера. Чтобы проверить наличие сетевого доступа к сайтам, работающим на вашей виртуальной машине VirtualBox, убедитесь, что он настроен на использование сетевого моста (как мы делали это в главе 2).

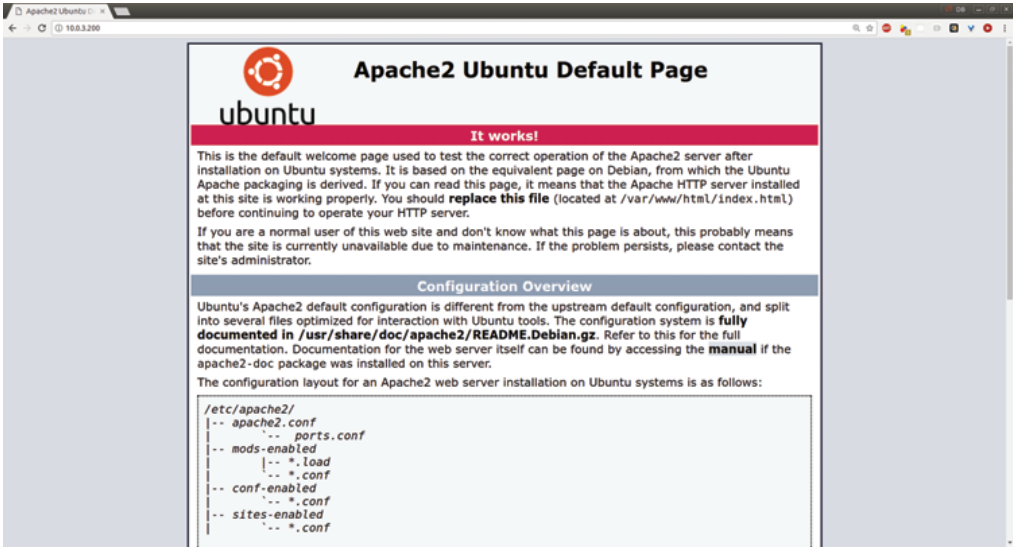


Рис. 7.2. Страница Apache по умолчанию, отображается в браузерах и указывает путь на URL-адрес или IP-адрес вашего сервера, содержит важные основные сведения о конфигурации и навигации

Ubuntu облегчает жизнь. Какой от этого плюс? Еще раз: изучение того, как это работает на CentOS, помогает понять, что происходит «под капотом». Даже если у вас на данный момент нет виртуальной машины или контейнера с CentOS, Fedora или Red Hat, я бы порекомендовал вам хотя бы ознакомиться с методами работы CentOS.

Процессы CentOS, которые мы рассмотрим в этой главе, отличаются от процессов в Ubuntu. Для удобства я решил разделить их, поместив все направления CentOS в отдельный раздел: 7.6. Если вы хотите установить Apache на CentOS, перейдите в него за подробной информацией.

7.2.2. Заполнение корневого каталога документов сайта

Фантастика! У вас есть рабочий сайт. Если учесть, что на сайте нет ничего, кроме страницы приветствия Apache, не ожидайте похвалы (или получения большого дохода). Теперь вам потребуется добавить контент. И, чтобы это сделать, вам нужно знать, откуда он появляется.

Расположение содержимого на сайте контролируется настройкой `DocumentRoot` в файле конфигурации Apache. В системах CentOS конфигурация задается в файле `httpd.conf` в каталоге `/etc/httpd/conf/`. Пользователи Ubuntu найдут его в файле с именем `000-default.conf` в каталоге `/etc/apache2/sites-available/`. В любом

случае поиск файла конфигурации для `DocumentRoot`, вероятно, приведет к следующему результату:

```
DocumentRoot "/var/www/html"
```

Это означает, что Apache будет направлять все входящие запросы браузера в каталог `/var/www/html/`. Вы можете изменить это значение, чтобы оно указывало на необходимое место в вашей файловой системе. Хоть это и не тема данной книги, заострю внимание на следующем: если вы планируете разместить на своем сервере несколько сайтов, можете указать несколько местоположений файловой системы.

Если вы никогда не делали этого раньше, почему бы не потратить пару минут прямо сейчас и не создать собственный простой сайт? Создайте текстовый файл `index.html` в корне вашего документа. (Этот файл создаст страницу приветствия Apache с таким же именем.) Можно добавить в него определенный текст, а также ссылку на второй HTML-файл и вставить графическое изображение. Обязательно создайте второй файл вместе с изображением. Файл `index.html` теперь будет выглядеть так:

```
<h2>Welcome!</h2>
Take a look at our <a href="info.html">company history</a>.
<br>
And how about a look at our new company logo: <img src="logo.png">
```

7.3. Установка базы данных SQL

Взгляните на страницу с профессиональными прогнозами для системных администраторов компьютерных систем из справочника бюро статистики труда (BLS) министерства труда США (**mng.bz/kHN3**) (рис. 7.3). Учитывая весь контент, отображаемый на каждой из девяти вкладок страницы, текста довольно много. И я подозреваю, что вручную добавляли очень малую часть этой информации.

Вероятнее всего, база данных на сервере бюро содержит терабайты необработанных данных и в ее пределах можно найти структурированную информацию, относящуюся к каждой из тысяч описанных специальностей. Эти данные, вероятно, затем организовываются по категориям информации (сводка, рабочая среда и т. д.). Когда я запрашивал эту страницу из меню BLS (или через интернет-поисковик), вебсервер BLS запрашивал соответствующие необработанные данные из базы данных и динамически упорядочивал их на странице так, как вы видите на рис. 7.3.

Это очень хороший пример, но существует множество других способов использования сайтом динамического доступа к подсистеме хранения (или движку базы данных) (database engine), установленной в серверной части. Тип СУБД, наиболее вероятно используемый для подобного проекта бюро (или для нашего сайта MediaWiki), называется *реляционной базой данных*. Это инструмент для организации данных в таблицы, состоящие из столбцов и строк. Данные, содержащиеся в отдельной строке, называются *записью*. Запись идентифицируется значением идентификатора, известным как *ключ*, который может использоваться для ссылки на записи между таблицами.

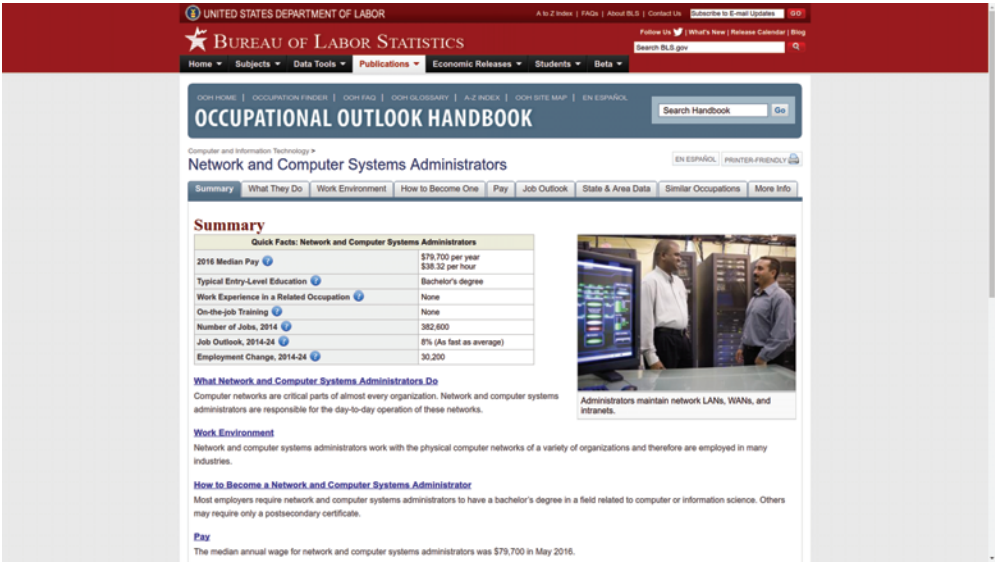


Рис. 7.3. Страница из бюро статистики труда. Вероятнее всего, заголовок What Network and Computer Systems Administrators Do изначально был чем-то вроде этого: What $ selected_occupationDo

Язык структурированных запросов (SQL) — это стандартизированный синтаксис для управления данными в реляционных базах данных. Движок базы данных (БД) — это программное обеспечение для управления данными реляционной БД и предоставления их администраторам и автоматизированным процессам с использованием синтаксиса SQL.

Вскоре я покажу вам, как создать и отобразить простую таблицу базы данных. Но сначала вам нужно установить собственный движок БД, чтобы вы могли все делать сами. Поскольку нашей долгосрочной целью является создание полноценного LAMP-сервера, имеет смысл установить его на тот же компьютер/виртуальную машину/контейнер, где вы создали свой веб-сервер Apache. (Информацию о CentOS можно найти в конце главы.)

```
# apt update
# apt install mariadb-server
```

Установка MariaDB на сервере

Почему я выбрал MariaDB вместо MySQL? Они же обе работают с одинаковыми стандартами MySQL. На самом деле обе изначально были созданы одними и теми же людьми. Обе великолепны, но на данный момент MariaDB находится в приоритете, потому что активнее разрабатывается и поддерживается. Помимо этих двух, в IT-сфере широко используются другие важные подсистемы хранения SQL, в том числе Oracle, PostgreSQL и Amazon Aurora, разработанные специально для рабочих нагрузок AWS.

Почему бы нам не проверить состояние базы данных, которую вы только что установили? Вы можете получить подтверждение о работе БД, используя команду `systemctl`:

```
# systemctl status mysql
? mysql.service – MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service;
       enabled; vendor preset: enabled)
   Active: active (running) since Wed 2018-05-02 12:26:47 UTC; 6h ago
  Process: 396 ExecStartPost=/usr/share/mysql/mysql-systemd-start post
       (code=exited, status=0/SUCCESS)
  Process: 318 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre
       (code=exited, status=0/SUCCESS)
 Main PID: 395 (mysqld)
    Tasks: 28
   Memory: 126.3M
    CPU: 20.413s
   CGroup: /system.slice/mysql.service
           ??395 /usr/sbin/mysqld

May 02 12:26:29 base systemd[1]: Starting MySQL Community Server...
May 02 12:26:47 base systemd[1]: Started MySQL Community Server.
```

Независимо от того, установлена ли у вас MySQL или MariaDB, Linux обращается к БД с помощью команды mysql

7.3.1. Усиление защиты SQL

После установки MariaDB неплохо было бы улучшить безопасность вашей базы данных. Для этого нужно запустить инструмент `mysql_secure_installation`. Если вам не было предложено создать пароль MariaDB во время процесса установки (что довольно распространенное явление), потребуется запустить `mysql_secure_installation`, поскольку именно так вы и настроите свою аутентификацию. При запуске этого инструмента отображается следующий интерактивный диалог:

```
# mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MariaDB
root user without the proper authorisation.

Set root password? [Y/n]
```

Обратите внимание, что эта команда может требовать права sudo, что вызовет проблемы позже. Будьте внимательны

Введите пароль root-пользователя базы данных, а не суперпользователя Linux

Значения, рекомендуемые `mysql_secure_installation`, предназначены для ограничения доступа анонимных и удаленных пользователей к вашим данным. Если вы планируете пользоваться этой базой данных только для тестирования и она не будет содержать важные или конфиденциальные данные, то, очевидно, вы можете оставить эти значения по умолчанию.

ПРИМЕЧАНИЕ

Если инструмент mysql_secure_installation работает только при запуске с использованием sudo, все равно создавайте свой пароль. Но имейте в виду, что это приведет к проблемам, которые позже придется исправить. Как их решить, вы узнаете в следующем разделе.

7.3.2. Администрирование SQL

Теперь, как и обещал, я покажу вам несколько простых команд по администрированию базы данных. Вам, возможно, никогда не придется выполнять какие-либо из них напрямую, поскольку доступ к большинству баз данных осуществляется из кода приложения, а не из командной строки. Учитывая, насколько неудобно было бы вручную управлять тысячами или даже миллионами записей данных, обычно включенных в базы данных SQL, это имеет большое значение. Чуть позже в этой главе, когда мы наконец приступим к установке и настройке MediaWiki, вы увидите прекрасный пример автоматизированных отношений между приложением и базой данных.

Тем не менее вам может понадобиться навык создания собственной базы данных. Возможно, пока вы создаете новое приложение, вам потребуются для работы тестовые данные. Или, может быть, ваш новый бизнес совсем медленно развивается и вместо того, чтобы инвестировать в новое приложение, сейчас имеет смысл управлять своими клиентами вручную. В любом случае вы должны хотя бы знать, как это делается.

Можете спокойно переходить к разделу 7.4, если чувствуете, что не готовы сейчас погрузиться в изучение баз данных. Но есть еще одна важная деталь, о которой я должен упомянуть.

По умолчанию с помощью root-пользователя вы будете получать доступ к базам данных и администрировать их при установке MariaDB или MySQL. Это изначально плохая идея. По соображениям безопасности отдельные базы данных должны принадлежать обычным пользователям баз данных и управляться ими. Этим пользователям предоставлены только те полномочия, которые им необходимы для выполнения конкретной работы. Тем не менее для наглядности я пренебрегу мерами предосторожности и буду работать с root-пользователем. Позже, когда я стану показывать, как настроить базу данных MediaWiki, я уже сделаю все правильно, создав пользователя без root-полномочий.

Доступ к базе данных

Вы установили MariaDB или MySQL и теперь входите в свою оболочку, используя команду `mysql`, за которой следует `-u root`. Это сообщает базе данных, что вы хотите аутентифицироваться как пользователь root. `-p` означает, что вам будет предложено ввести пароль MariaDB:

```
$ mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 10
Server version: 5.5.52-MariaDB MariaDB Server

Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type \c to clear the current input statement

MariaDB [(none)]>
```

Войдите, указав пользователя, и запросите пароль пользователя

Значение [(none)] изменится на имя активной базы данных

Вот где появляется проблема, о которой я говорил ранее. MariaDB может не позволить вам войти, если вы не запустите команду `mysql` как `sudo`. Если это произошло, войдите в систему с помощью `sudo` и введите созданный вами пароль MariaDB. Затем выполните следующие три команды в MySQL-запросах (указав вместо `your-password` свой пароль):

```
> SET PASSWORD = PASSWORD('your-password');
> update mysql.user set plugin = 'mysql_native_password' where User='root';
> FLUSH PRIVILEGES;
```

При последующих авторизациях вам больше не потребуется указывать `sudo`, и, что еще более важно, MediaWiki должна работать корректно. Избавившись от этой небольшой проблемы, взгляните на свою среду SQL. Вот как можно создать новую базу данных:

```
MariaDB> CREATE DATABASE companydb;
```

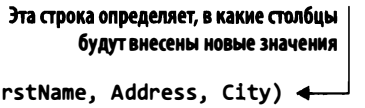
Большинство команд MySQL должны заканчиваться точкой с запятой (;). Забыв об этом, вы не сможете выполнить команду

Предположим, вашей компании необходимо хранить контактную информацию клиентов. Создать новую таблицу для контактов в базе данных вы сможете следующим образом:

```
MariaDB> use companydb
MariaDB> CREATE TABLE Contacts (
    ID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```

Это последняя часть одной команды, для ясности растянутой на семь строк

У вас появился первый клиент? Поздравляю! Вот как вы введете новую информацию о нем:




```
MariaDB> INSERT INTO Contacts (ID, LastName, FirstName, Address, City)
VALUES ('001', 'Torvalds', 'Linus', '123 Any St.', 'Newtown');
```

Хотите посмотреть на результаты проделанной работы? Чтобы отобразить все данные в новой таблице контактов, используйте команду `select *`:

```
MariaDB> select * from Contacts;
+------+----------+-----------+-------------+---------+
| ID   | LastName | FirstName | Address     | City    |
+------+----------+-----------+-------------+---------+
|    1 | Torvalds | Linus     | 123 Any St. | Newtown |
+------+----------+-----------+-------------+---------+
1 row in set (0.00 sec)
```

Обратите внимание на значение `ID` — его можно использовать в качестве ключа для ваших записей. Когда все будет готово, вы можете закрыть оболочку MariaDB, введя команду `exit`.

Я говорил ранее, что вам, возможно, никогда не потребуется выполнять какие-либо из подобных задач вручную, зачем тогда здесь эта информация? Ответ прост: наверняка в один прекрасный день вам потребуется подключить вариации синтаксиса MySQL в скрипты и код приложения для интеграции автоматизированных операций с базами данных. Даже самые простые данные интернет-магазинов хранятся отдельно. Возможно, вы не занимаетесь программированием, а это делает кто-то, кого вы знаете или любите, и ему может понадобиться ваша помощь в установлении соединения с базой данных. Неутомимый системный администратор Linux сделает все за один день.

Создание пользователя базы данных MediaWiki

Есть еще кое-что важное в администрировании баз данных. Как вы теперь знаете, MariaDB сразу поставляется с активным пользователем **root**. Но поскольку у этого пользователя есть полные права администратора для всех таблиц в системе, не рекомендуется применять **root** для повседневных операций. Вместо этого с точки зрения безопасности вам лучше создавать уникальных пользователей для каждого пользователя базы данных и предоставлять им только необходимые права доступа.

Давайте еще раз войдем в MariaDB и создадим новую базу под названием `wikidb` для использования MediaWiki. Затем создадим пользователя, которого назовем **mw-admin**. Команда `FLUSH PRIVILEGES` имеет другие настройки и предоставляет пользователю **mw-admin** полный контроль над базой данных `wikidb`:

```
mysql> CREATE DATABASE wikidb;
Query OK, 1 row affected (0.01 sec)
```

```
mysql> CREATE USER 'mw-admin'@'localhost' IDENTIFIED BY 'mypassword';
Query OK, 0 rows affected (0.00 sec)
mysql> GRANT ALL PRIVILEGES ON wikidb.* TO 'mw-admin'@'localhost'
   IDENTIFIED BY 'mypassword';
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
mysql> exit
```

7.4. Установка PHP

Последний компонент LAMP — это язык сценариев PHP. PHP — это инструмент, который можно использовать для написания собственных веб-приложений. Готовые PHP-приложения часто применяются сторонними приложениями, такими как MediaWiki, для доступа и обработки системных ресурсов. Поэтому можно с уверенностью сказать, что буква P в аббревиатуре сервера LAMP — очень важный компонент.

7.4.1. Установка PHP в Ubuntu

Несмотря на примеры, которые вы встречали в книге ранее, установка на Ubuntu не всегда проще, чем на CentOS. Но с данной задачей Ubuntu справится быстрее. Хотите последнюю версию PHP? Команда `apt install php` — все, что нужно для Ubuntu, чтобы исполнить ваше желание. Поскольку вы хотите, чтобы PHP хорошо работал с Apache, вам также понадобится следующее расширение:

```
# apt install php
# apt install libapache2-mod-php
```

У вас должно войти в привычку перезапускать Apache каждый раз после внесения изменений в конфигурацию системы веб-сервера. Вот так:

```
# systemctl restart apache2
```

Это все, теперь PHP установлен.

7.4.2. Тестирование установки PHP

Чтобы убедиться, что установка PHP прошла успешно (и узнать о локальной среде PHP и интеграции ресурсов), создайте новый файл в корневом каталоге веб-документа Apache с расширением `.php`. Затем сохраните в этом файле следующий код:

```
# nano /var/www/html/testmyphp.php
<?php
phpinfo();
?>
```

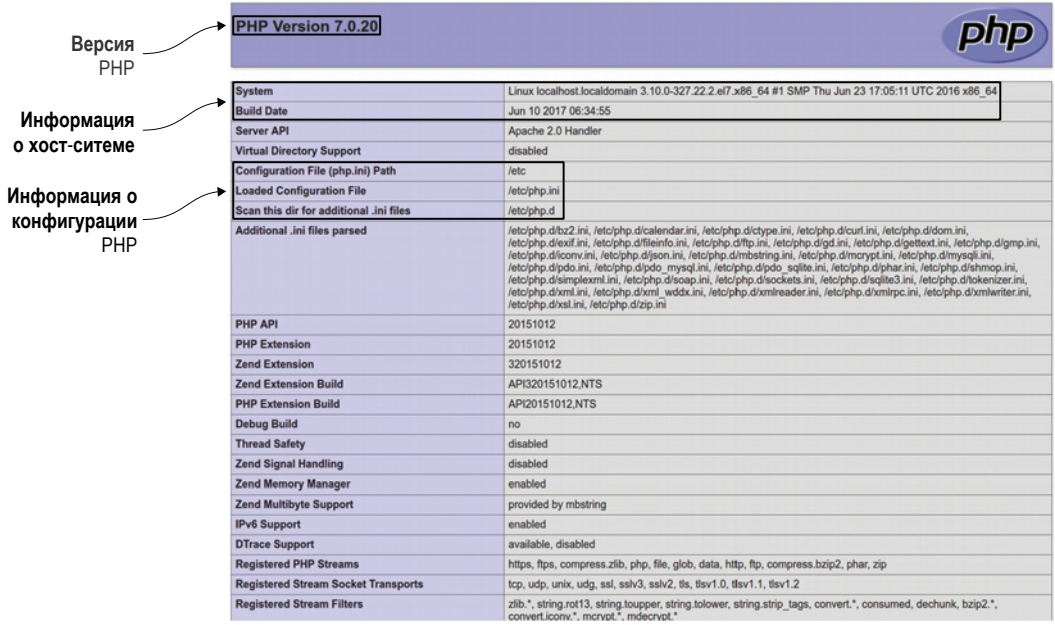
Теперь перейдите в браузер, введите IP-адрес компьютера, на котором работает PHP (или localhost, если это Рабочий стол, на котором вы работаете), и имя созданного файла:

```
10.0.3.184/testmyphp.php
```

Отобразится длинная веб-страница (рис. 7.4) с множеством разделов, в которых описываются ваш компьютер и способы взаимодействия PHP с ним.

ПРИМЕЧАНИЕ

Когда вы закончите работу, не забудьте удалить или ограничить доступ к файлу testmyphp.php. Доступ к такой информации о вашей системе является серьезным нарушением безопасности.



System	Linux localhost.localdomain 3.10.0-327.22.2.el7.x86_64 #1 SMP Thu Jun 23 17:05:11 UTC 2016 x86_64
Build Date	Jun 10 2017 06:34:55
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
Additional .ini files parsed	/etc/php.d/bz2.ini, /etc/php.d/calendar.ini, /etc/php.d/ctype.ini, /etc/php.d/curl.ini, /etc/php.d/dom.ini, /etc/php.d/exif.ini, /etc/php.d/fileinfo.ini, /etc/php.d/ftp.ini, /etc/php.d/gd.ini, /etc/php.d/gettext.ini, /etc/php.d/gmp.ini, /etc/php.d/iconv.ini, /etc/php.d/json.ini, /etc/php.d/mbstring.ini, /etc/php.d/mcrypt.ini, /etc/php.d/mysqli.ini, /etc/php.d/pdo.ini, /etc/php.d/pdo_mysql.ini, /etc/php.d/pdo_sqlite.ini, /etc/php.d/phar.ini, /etc/php.d/shmop.ini, /etc/php.d/simplexml.ini, /etc/php.d/soap.ini, /etc/php.d/sockets.ini, /etc/php.d/sqlite3.ini, /etc/php.d/tokenizer.ini, /etc/php.d/xml.ini, /etc/php.d/xml_wddx.ini, /etc/php.d/xmlreader.ini, /etc/php.d/xmlrpc.ini, /etc/php.d/xmlwriter.ini, /etc/php.d/xsl.ini, /etc/php.d/zip.ini
PHP API	20151012
PHP Extension	20151012
Zend Extension	320151012
Zend Extension Build	API320151012,NTS
PHP Extension Build	API20151012,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, compress.bzip2, phar, zip
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, sslv2, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, bzip2.*, convert.iconv.*, mcrypt.*, mdecrypt.*

Рис. 7.4. Небольшой пример данных конфигурации и среды, отображаемых командой phpinfo

Далее все это будет продемонстрировано на CentOS. А пока предоставим MediaWiki свою собственную базу данных.

7.5. Установка и настройка MediaWiki

Есть документы и информация, которой можно поделиться (как мы обсуждали в начале этой главы несколько страниц назад)? Вы дошли до нужного места. Вот как будет выглядеть этот процесс.

1. Скачайте и распакуйте архивный пакет MediaWiki.
2. Выберите и установите необходимые расширения программного обеспечения.
3. Подключите MediaWiki к вашей базе данных MariaDB.
4. Запустите и проверьте установку.

Перейдите на страницу загрузки MediaWiki (www.mediawiki.org/wiki/Download) и щелкните на ссылке Download (Скачать), чтобы получить последнюю версию пакета. Если вы хотите перетащить файл сразу на сервер с помощью командной строки, щелкните правой кнопкой мыши на ссылке Download (Скачать), выберите пункт Copy Link Address (Копировать адрес ссылки) и вставьте адрес в окно оболочки командной строки вместе с командой `wget`:

```
$ wget https://releases.wikimedia.org/mediawiki/1.30/\
    mediawiki-1.30.0.tar.gz
```

Точный адрес (и версия), скорее всего, изменится к тому времени, как вы прочитаете это

ПРИМЕЧАНИЕ

Если при запуске предыдущей команды вы получаете ошибку -bash: wget: Command Not Found, вам необходимо установить программу wget.

Запуск команды `tar` для загруженного архива создает новый каталог, содержащий все извлеченные файлы и каталоги. Необходимо скопировать всю эту структуру каталогов в файловую систему, где она будет выполнять свою работу. Если MediaWiki будет единственным веб-приложением, размещенным на вашем сервере, это, вероятно, будет ваш корневой веб-каталог:

```
$ tar xzvf mediawiki-1.30.0.tar.gz
$ ls
mediawiki-1.30.0  mediawiki-1.30.0.tar.gz
# cp -r mediawiki-1.30.0/* /var/www/html/
```

Если MediaWiki будет одним из нескольких приложений, то можете создать подкаталог в корне документа, который будет предоставлять сервис обычным способом. Например, размещение файлов в каталоге `/var/www/html/mediawiki/` означает, что ваши пользователи найдут MediaWiki по адресу www.example.com/mediawiki, при условии что вы используете example.com в качестве своего публичного домена.

С этого момента интерфейс браузера MediaWiki выходит на передний план. Запустите файл `index.php` в каталоге `MediaWiki` по IP-адресу вашего сервера (или на localhost, если вы запустили все это на своем компьютере). Если вы скопировали файлы в каталог `/var/www/html/root`, то команда будет выглядеть примерно так:

```
10.0.3.184/index.php
```

Если вместо этого вы создали подкаталог для MediaWiki, команда будет выглядеть следующим образом:

```
10.0.3.184/mediawiki/index.php
```

Обычно, когда я первый раз устанавливаю приложение, я следую инструкциям хорошего онлайн-руководства, желательно того, которое было создано и поддерживается самими разработчиками проекта. Чаще всего это руководство дает мне длинный список зависимостей пакетов, который я быстро прочитываю (иногда слишком быстро), а затем копирую и вставляю в команду `apt install`.

На этот раз, однако, я перевернул процесс с ног на голову и установил только основные пакеты Apache, MariaDB и PHP. Я знаю, что этого недостаточно для полноценной работы MediaWiki, но именно этого я и добивался. Это лучший способ узнать о том, как все сложные биты взаимодействуют друг с другом.

7.5.1. Диагностика недостающих расширений

Люди, которые разрабатывали процесс установки MediaWiki, понимали, что не всегда все будет проходить гладко. Если выяснится, что в вашей конфигурации чего-то не хватает, перед вами сразу отобразится страница, содержащая полезную информацию об ошибке. В данном случае, как показано на рис. 7.5, мне не хватает двух расширений PHP: *mbstring* и *xml*.

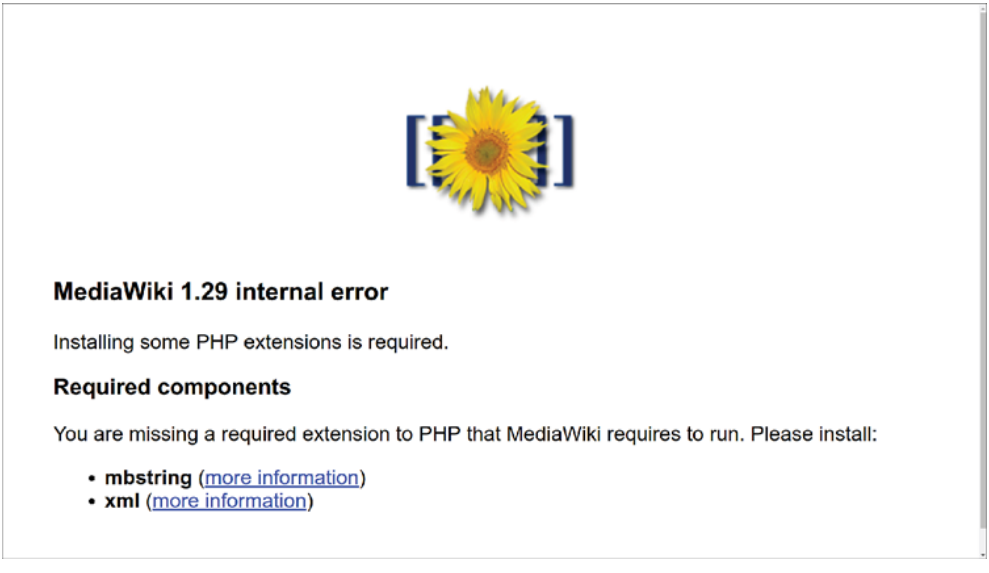


Рис. 7.5. Полезная страница об ошибках, сообщающая, что в моей системе отсутствуют два расширения. Здесь же прикреплены ссылки на соответствующие страницы документации PHP

Я использую команду `apt search`, чтобы увидеть, какие пакеты относятся к mbstring. Поскольку я работаю с PHP 7, пакет php7.0-mbstring больше всего подходит для того, чтобы вернуться к цветку на странице MediaWiki:

```
$ apt search mbstring
Sorting... Done
Full Text Search... Done
php-mbstring/xenial 1:7.0+35ubuntu6 all
  MBSTRING module for PHP [default]

php-patchwork-utf8/xenial 1.3.0-1build1 all
  UTF-8 strings handling for PHP

php7.0-mbstring/xenial-updates 7.0.18-0ubuntu0.16.04.1 amd64
  MBSTRING module for PHP
```

Обеспечивает многобайтовое кодирование строк для PHP 7

Похожий поиск по `xml` и `php` (`apt search xml | grep php`) указал мне на пакет php7.0-xml, который, вероятно, удовлетворяет требованиям MediaWiki по XML. Я установлю оба пакета, а затем введу команду `systemctl` для перезапуска Apache:

```
# apt install php7.0-mbstring php7.0-xml
# systemctl restart apache2
```

Самое прекрасное — ни вам, ни мне не нужно разбираться в том, что такое многобайтовое строковое кодирование или даже XML и почему MediaWiki не будет без них функционировать. На данный момент мы можем доверять разработчикам приложений, и нет ничего плохого в том, чтобы следовать их инструкциям. Это подразумевает, что мы можем доверять разработчикам в том, что нам зачастую трудно оценить. Философия, лежащая в основе всей системы управления пакетами Linux, построена на том, что менеджеры лицензионных репозиториев уже сделали для нас работу по проверке. Мы должны быть уверены в ее надежности.

Вернемся к нашему примеру. Если все идет по плану, обновление страницы браузера вернет вас к начальной приветственной странице MediaWiki. Когда страница загрузится, вы увидите предупреждение об отсутствующем файле `LocalSettings.php` и ссылку для настройки вики. Когда вы щелкнете на ссылке, сможете выбрать язык, а затем страницу MediaWiki Environmental Checks и... получите еще больше проблем!

Самая большая проблема — отсутствие *драйвера базы данных*. Это программное обеспечение, используемое для сообщения между PHP и в нашем случае MariaDB. Нет установленного драйвера? Это смерти подобно. Несмотря на предложенный пакет php5-mysql, показанный на рис. 7.6, `apt search` сообщает нам, что мы с большей вероятностью добьемся успеха с пакетом php-mysql.

Добавьте также предложенные PHP-расширения для APCu (часть фреймворка для кэширования и оптимизации промежуточного кода PHP) и ImageMagick (инструмент для обработки изображений). Еще один перезапуск для Apache, обновление окна браузера — и все должно быть готово:

```
# apt install php-mysql php-apcu php-imagick
# systemctl restart apache2
```

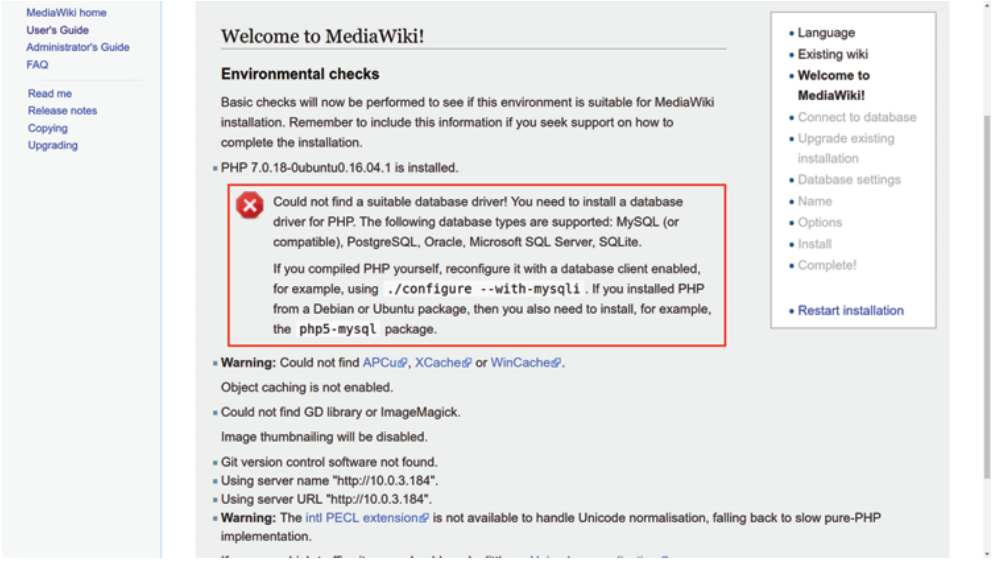


Рис. 7.6. Красной рамкой выделена важная ошибка в конфигурации; другие примечания представляют собой менее строгие предупреждения

Вы увидите кнопку **Continue** (Продолжить) внизу страницы. Нажмите ее.

7.5.2. Подключение MediaWiki к базе данных

Информация, которую вы предоставляете через страницу подключения к базе данных, сообщается MediaWiki.

- Какая база данных установлена в системе (в данном случае MySQL или совместимая).
- Где база данных находится (она может быть удаленной или даже облачной, но наша на локальном сервере, поэтому **localhost** — правильный вариант).
- Название, которое вы хотели бы дать базе данных, используемой MediaWiki (я буду работать с базой данных `wikidb`, которую создал ранее).
- Название существующей учетной записи пользователя базы данных (в нашем случае **mw-admin**).
- Пароль текущей учетной записи базы данных (это позволит MediaWiki получать доступ к MariaDB, создавать и администрировать свою базу данных; рис. 7.7).

ПРИМЕЧАНИЕ

Не работает? Если MediaWiki не может подключиться к вашей базе данных, убедитесь, что указали правильный пароль, а также удостоверьтесь, что можете войти в оболочку MariaDB из командной строки.

Если все пойдет успешно, вы пройдете через ряд диалоговых окон, в которых потребуется ввести детали конфигурации: настройки базы данных, имя вашей вики (что-то вроде документации компании в этом примере) и имя пользователя, пароль и контактный адрес электронной почты, адрес для учетной записи администратора вики. Эта учетная запись не связана с учетными записями, которые у вас уже есть на хосте Linux или MariaDB.

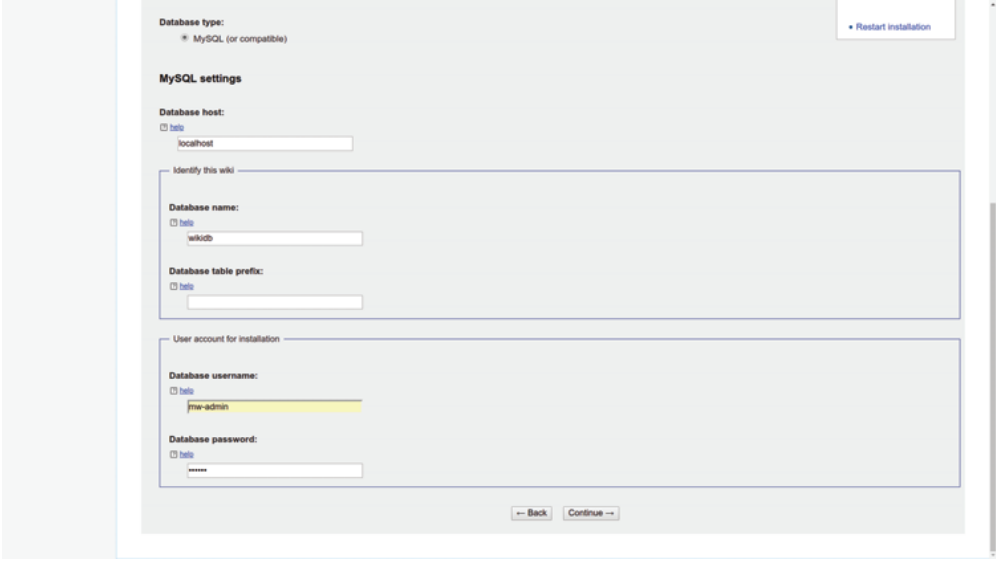


Рис. 7.7. Часть страницы настроек MySQL, где вы определяете в MediaWiki параметры для подключения к базе данных

Дополнительные вопросы по настройке позволят вам установить предпочтения по правам пользователя, авторским правам для содержимого вики, почтового сервера для отправки уведомлений по электронной почте и дополнительных программных расширений для таких компонентов, как WikiEditor в браузере или программное обеспечение против спама. Везде, кроме адреса электронной почты, можно оставить настройки по умолчанию.

Когда вы закончите, MediaWiki начнет процесс установки. По завершении вам будет предложено скачать файл с именем `LocalSettings.php` и сохранить его в корневом каталоге MediaWiki (в примере это `/var/www/html/`). Можно использовать команду `scp` для копирования сохраненного файла из домашнего каталога пользователя:

```
$ scp LocalSettings.php ubuntu@10.0.3.184:/home/ubuntu/
```

А затем из командной строки сервера переместить его в корневой каталог документа:

```
# cp /home/ubuntu/LocalSettings.php /var/www/html/
```

Когда все будет готово, вернитесь на страницу по тому же адресу, который указывали ранее (здесь `10.0.3.184/index.php`). На этот раз, как вы можете видеть на рис. 7.8, вы окажетесь на главной странице своей новой вики. С этого момента начинается ваша работа по наполнению вики плодами вашего труда.

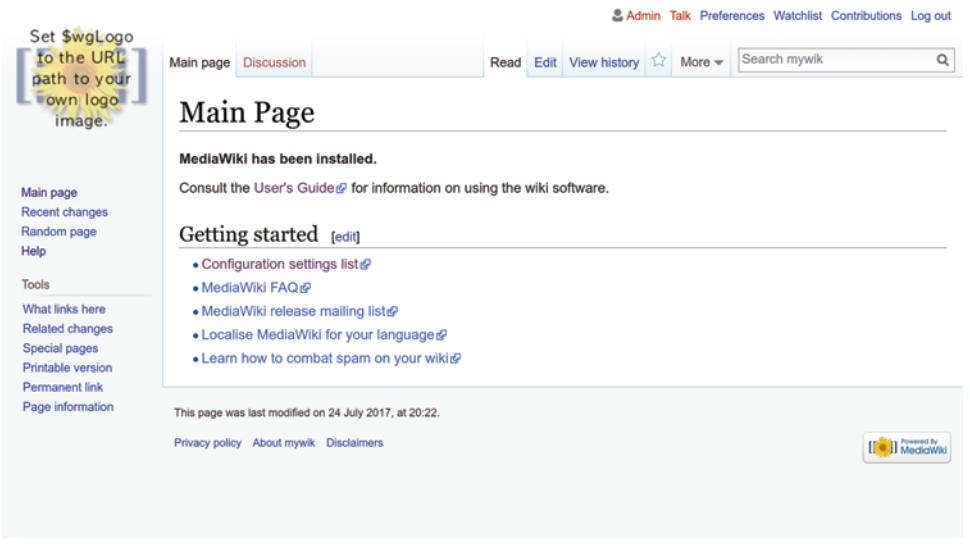


Рис. 7.8. Можно добавлять и редактировать ссылки и содержимое страницы, выбрав вверху вкладку Edit (Редактировать)

Потратьте некоторое время на изучение руководства пользователя MediaWiki (**www.mediawiki.org/wiki/Help:Contents**). Оно подскажет, как работать с файлами и писать код на простом языке разметки.

7.6. Установка веб-сервера Apache на CentOS

Если хотите избежать проблем, с которыми мы столкнулись в процессе установки на Ubuntu, подумайте над тем, чтобы провести некоторые исследования для выяснения того, какие именно версии вам понадобятся для каждой части головоломки LAMP. Краткий обзор страницы требований к установке MediaWiki (**mng.bz/4Bp1**) предоставит вам всю информацию, необходимую для установки веб-сервера Apache (или httpd, известного в CentOS). Как только вы все сделаете, получить программное обеспечение будет просто:

```
# yum install httpd
```

По умолчанию Apache не работает. Помните, как это исправить с помощью systemd? Не подглядывайте! Я надеюсь, что вы догадались:

```
# systemctl start httpd
# systemctl enable httpd
```

Команда `systemctl start` запускает систему, а `enable` вызывает ее запуск при каждой загрузке компьютера. Чтобы убедиться, что Apache работает, вы можете использовать команду `curl` для отображения веб-страницы в терминале, указав `localhost`, который сообщит `curl`, что после запуска страницы по умолчанию вы обслуживаетесь локально. Возможно, вам придется установить `curl`. Я уверен, что с этим проблем не возникнет. Если `curl` отображает информацию, начинающуюся с текста, как в примере, то очевидно, что Apache выполняет свою работу:

```
                                                    Атрибут title для страницы
$ curl localhost                                          Apache index.html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"><html><head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
     <title>Apache HTTP Server Test Page
         powered by CentOS</title>
     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
[...]
```

Но вот что странно: команда `curl` показывает, что Apache работает. Сетевое подключение между вашим компьютером и виртуальной машиной CentOS проходит через мостовой адаптер. (Надеюсь, вы в курсе этого? Если нет, глава 14 поможет с этим разобраться.) Но у вас все равно не получается загрузить страницу из GUI браузера. Скорее всего, отобразится ошибка «Этот сайт не может быть отображен». В чем же проблема?

Эта ошибка иллюстрирует еще одно интересное различие философий Ubuntu и CentOS. По умолчанию Ubuntu поставляется без какого-либо брандмауэра (поскольку за пределами базовой инфраструктуры нет открытых сетевых служб, это не так странно, как кажется). Apache готов принимать входящий внешний трафик сразу после установки. Естественно, есть много способов закрыть доступ к сети в соответствии с вашими потребностями, но изначально вы открыты для всего мира. CentOS, наоборот, поставляется с надежно закрытыми портами. Если вы хотите, чтобы ваш веб-сервер получал входящие HTTP-запросы, вам сначала нужно открыть HTTP-порт (по договоренности это порт 80).

7.6.1. Общие сведения о сетевых портах

Итак, что такое *сетевой порт*? Это не что иное, как способ идентификации определенного ресурса сервера для пользователей сети. Представьте, что на вашем сервере размещены два отдельных приложения. Посетители могут получить доступ к вашему серверу, используя его публичный IP-адрес или соответствующее имя домена DNS (например, 172.217.1.174 для google.com). Но как браузер узнает, какое из двух приложений вы хотите загрузить?

Когда заранее задан определенный порт, приложениям может быть предписано отслеживать трафик, поступающий на сервер. Таким образом, одно приложение может использовать порт 50501, а другое — порт 50502. Как видно из рис. 7.9, первое приложение будет отвечать на входящие запросы, используя 192.168.2.10:50501 (при

условии, что 192.168.2.10 — это IP-адрес вашего сервера), а второе приложение будет ждать трафик, используя 192.168.2.10:50502.

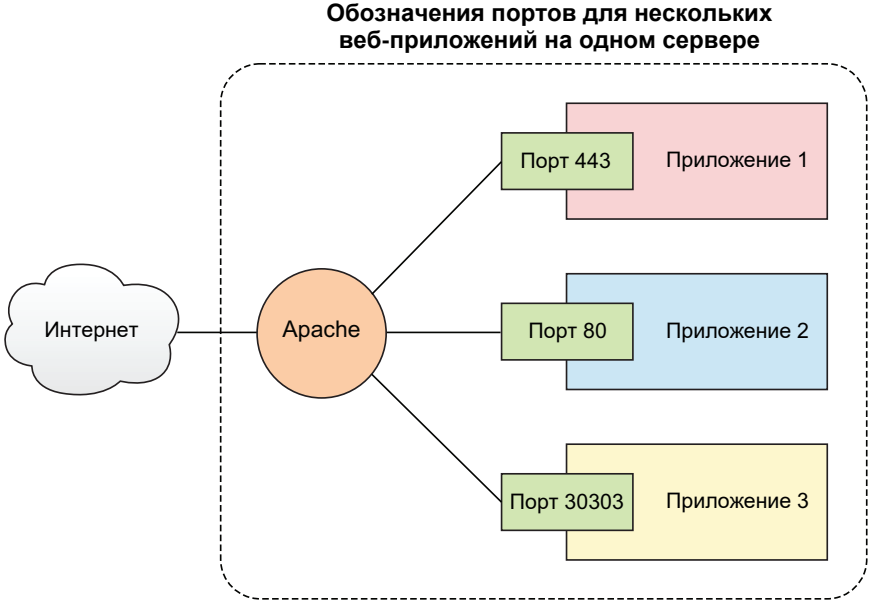


Рис. 7.9. Приложения, настроенные на отслеживание через отдельные сетевые порты (80 = небезопасный HTTP; 443 = безопасный HTTPS; 30303 = пользовательское приложение)

В главе 9 и 14 мы подробнее рассмотрим порты. Но на данный момент вам нужно понимать, что именно программное обеспечение вашего веб-сервера (например, Apache) будет выполнять бóльшую часть тяжелой работы для серверов, на которых размещены несколько сайтов, и автоматически переводить порты в локальные местоположения файловой системы. Если вы планируете открыть свой веб-сервер для нескольких сайтов, то вам следует начать с изучения руководств по настройке программного обеспечения (например, mng.bz/w6h4).

7.6.2. Управление сетевым трафиком

Как вы будете контролировать доступ к сети? Один из способов — использовать *правила брандмауэра*. На CentOS это осуществляется через службу firewalld и ее инструмент firewall-cmd. В данном случае необходимо добавить службу http и с помощью команды `--permanent` убедиться, что правило будет активно при каждом перезапуске службы или загрузке компьютера (в главе 9 будет гораздо больше информации о брандмауэрах). Чтобы применить изменения, перезапустите службу:

```
# firewall-cmd --add-service=http -permanent
success
# systemctl restart firewalld
```

После этого вы сможете успешно загрузить на CentOS версию тестовой страницы Apache (рис. 7.10).

Рис. 7.10. Страница по умолчанию на веб-сервере Apache с CentOS

7.6.3. Установка MariaDB на CentOS

Опять же сама установка проще простого. Однако вам нужно будет вручную запустить MariaDB, а затем ввести `enable`, чтобы настроить базу для запуска при загрузке системы:

```
# yum install mariadb-server
# systemctl start mariadb
# systemctl enable mariadb
```

С этого момента вы можете выполнять все шаги по установке MariaDB, описанные ранее, как часть процесса установки Ubuntu.

7.6.4. Установка PHP на CentOS

На момент написания книги через репозиторий CentOS по умолчанию была доступна версия PHP 5.4, которая значительно отстает от последней версии, доступной в репозитории Ubuntu. Проблема в том, что текущая версия MediaWiki (1.30) не будет работать с версиями PHP старше 5.5.9. К тому времени, когда вы прочтете это, проблема, возможно, решится. Если нет, вам нужно будет воспользоваться веб-поиском, чтобы найти репозитории, содержащие более современные версии PHP для установки.

Я говорю об этом не потому, что CentOS на данный момент отстает по версиям, а потому, что подобные вещи случаются постоянно и кое-что нужно изучить дополнительно. Урок первый — всегда проверять версию пакета перед его установкой. Все менеджеры пакетов Linux предоставляют необходимую информацию, и, прежде чем приступать к работе, ее надо изучить. Привыкайте читать. Ниже приводится сводная информация о программном обеспечении для установочного пакета PHP:

Установка не начнется, пока вы не подтвердите ее

Обратите внимание, что версия PHP dnf устанавливается по умолчанию

```
# dnf install php
Using metadata from Sun Jul 23 10:37:40 2017
Dependencies resolved.
================================================================================
 Package         Arch         Version                  Repository     Size
================================================================================
Installing:
 libzip          x86_64       0.10.1-8.el7             base           48 k
 php             x86_64       5.4.16-42.el7            base          1.4 M
 php-cli         x86_64       5.4.16-42.el7            base          2.7 M
 php-common      x86_64       5.4.16-42.el7            base          564 k

Transaction Summary
================================================================================
Install 4 Packages

Total download size: 4.7 M
Installed size: 17 M
Is this ok [y/N]:
```

dnf отображает сводную информацию об установке перед запуском указанного действия

Важно также помнить, что старые и более новые версии программных пакетов могут сосуществовать в системе хранилища одновременно. Помните, что у вас есть выбор, но имейте в виду, что самая последняя версия не всегда вам подходит. Это может быть связано с тем, что другое приложение, с которым вы работаете, не поддерживает последнюю версию этого пакета или более новая версия недостаточно стабильна или безопасна для вашего проекта.

ПРИМЕЧАНИЕ

Есть большая вероятность, что однажды вам понадобится установить пакет, который не является частью официального репозитория Linux. Это не преступление и не аморальный поступок, поэтому нет причин чувствовать себя виноватым. Но вы должны понимать, что теперь только вы сами будете нести полную ответственность за поддержание совместимости с другими сервисами и приложениями, а также за то, что ваш пакет и нерепозиторная конфигурация выдержат обновления системы. И конечно же, вы должны быть особенно осторожны в том, чтобы сам пакет не содержал вредоносных программ.

Независимо от того, используете вы лицензионную версию репозитория CentOS PHP или нет, вам, вероятно, придется вручную добавлять отдельные модули для

поддержки предполагаемых операций. Руководство по настройке, опубликованное в Интернете, поможет вам в этом. Вы также можете запустить `yum search php-` для проверки доступных модулей, а затем ввести `yum info` и название понравившегося модуля, чтобы больше о нем узнать:

```
# yum search php- | grep mysql
php-mysql.x86_64 : A module for PHP applications that use
  MySQL databases
php-mysqlnd.x86_64 : A module for PHP applications that use
   MySQL databases
php-pear-MDB2-Driver-mysql.noarch : MySQL MDB2 driver
php-pear-MDB2-Driver-mysqli.noarch : MySQL Improved MDB2 driver
[...]
# yum info php-mysql.x86_64
Available Packages
Name        : php-mysql
Arch        : x86_64
Version     : 5.4.16
Release     : 42.el7
Size        : 101 k
Repo        : base/7/x86_64
Summary     : A module for PHP applications that use MySQL databases
URL         : http://www.php.net/
License     : PHP
Description : The php-mysql package contains a dynamic shared object
            : that will add MySQL database support to PHP. MySQL is
            : an object-relational databasemanagement system. PHP is
            : an HTML-embeddable scripting language. Ifyou need MySQL
            : support for PHP applications, you will need to install
        : this package and the php package.
```

Пакет php-mysql для 64-битной архитектуры

Поскольку установлена база данных, похожая на MySQL, имеет смысл поискать схожие модули

Используйте команду yum info, чтобы больше узнать об интересующем пакете

Это имя пакета, которое вы будете использовать с yum для установки модуля

После установки PHP и других дополнительных модулей PHP перезапустите Apache, чтобы убедиться, что Apache смог включить новые модули в свою службу:

```
# systemctl restart httpd.service
```

Как уже говорилось в этой главе, в процессе установки Ubuntu используйте `phpinfo`, чтобы убедиться, что PHP установлен правильно.

Резюме

- Пакеты веб-сервера, такие как Apache, координируют взаимодействие между системными ресурсами (базами данных и файлами) и предоставляют ресурсы сайта клиентам.
- Доступные версии программного пакета могут различаться в зависимости от дистрибутива Linux, а требуемая версия релиза может зависеть от вашего конкретного проекта.

- Требования пакетов могут быть выполнены с помощью инструментов поиска и метаданных, предоставляемых системами управления пакетами (такими как APT и Yum).
- Приложениям часто требуется доступ к ядру базы данных, поэтому им должна быть предоставлена достоверная информация для аутентификации.

Ключевые понятия

- *Вики* — это инструмент для создания и управления распределенными совместными проектами.
- *Система управления контентом (CMS)* — приложение, разработанное для упрощения создания, обмена и редактирования цифрового контента.
- *Веб-сервер* — это программное обеспечение, предназначенное для безопасного и надежного предоставления ресурсов сервера удаленным клиентам.
- *DocumentRoot* — параметр Apache, определяющий, где в файловой системе веб-сервер будет искать файлы сайта.
- *Язык структурированных запросов (SQL)* — это синтаксис для управления данными в реляционных базах данных.
- *Зависимости пакетов* — программы или расширения, необходимые для правильной работы установленных приложений.

Рекомендации по безопасности

- Убедитесь, что настройки брандмауэра вашей системы дают клиенту соответствующий доступ к системным ресурсам и блокируют все остальные запросы.
- В большинстве случаев не стоит разрешать удаленный доступ к вашей базе данных.
- Никогда не оставляйте файл под управлением команды `phpinfo` на общедоступном сайте.

Обзор команд

- `apt install lamp-server^` — единственная команда Ubuntu, устанавливающая все элементы сервера LAMP.
- `systemctl enable httpd` запускает Apache на CentOS при каждой загрузке системы.
- `firewall-cmd --add-service = http --permanent` разрешает трафик HTTP-браузера в системе CentOS.
- `mysql_secure_installation` сбрасывает ваш административный пароль и повышает безопасность базы данных.

- `mysql -u root -p` позволяет войти в MySQL (или MariaDB) в качестве root-пользователя.
- `CREATE DATABASE newdbname;` создает новую базу данных в MySQL (или MariaDB).
- `yum search php- | grep mysql` ищет доступные пакеты, связанные с PHP, на компьютере с CentOS.
- `apt search mbstring` выполняет поиск доступных пакетов, связанных с многобайтовым кодированием строк.

Самотестирование

1. Какой из пакетов нужно установить, чтобы предоставить базу данных для веб-сервера LAMP:
   а) mariabd;
   б) httpd;
   в) mariadb-client;
   г) mariadb-server?
2. Какая из платформ обеспечивает сетевое взаимодействие между клиентами браузера и ресурсами данных на сервере:
   а) Apache;
   б) Perl;
   в) PHP;
   г) systemctl?
3. localhost — это обозначение, используемое для вызова:
   а) `DocumentRoot` удаленного клиента;
   б) `DocumentRoot` сервера, с которого запускается команда;
   в) `.conf`-файла на веб-сервере;
   г) порта HTTP по умолчанию на хост-сервере.
4. Где по умолчанию находится `DocumentRoot` веб-сервера Apache:
   а) `/var/www/html/`;
   б) `/var/html/www/`;
   в) `/ect/apache2/`;
   г) `/home/username/www/html`?
5. Какая из команд разрешит трафик браузера на веб-сервер CentOS через порт 80:
   а) `firewalld --add-service = http`;
   б) `firewall-cmd --add-service = https`;
   в) `firewall --add-service = https`;
   г) `firewall-cmd --add-service = http`?

6. Какая из команд позволит вам успешно войти в оболочку MariaDB:
   а) `mariadb -u root`;
   б) `mysql root -p`;
   в) `mariadb -r -p`;
   г) `mysql -u root –p`?
7. Какая из следующих команд `yum` отобразит контекстные данные об указанном пакете:
   а) `yum search php-mysql.x86_64`;
   б) `yum describe php-mysql.x86_64`;
   в) `yum info php-mysql.x86_64`;
   г) `yum data php-mysql.x86_64`?
8. Какой из следующих инструментов является необязательным для запуска MediaWiki на сервере LAMP:
   а) php-mysql;
   б) mbstring;
   в) php-imagick;
   г) php7.0-xml?

Ответы
1 — г; 2 — а; 3 — б; 4 — а; 5 — г; 6 — г; 7 — в; 8 — в.

8 Совместное использование файлов в сети: создание сервера для совместного использования файлов Nextcloud

В этой главе

- Установка программных пакетов и управлениеим и с использованием мгновенного снимка.
- Настройка Apache для управления несколькими сайтами на одном сервер.
- Обеспечение и администрирование корпоративного сайта для совместного использования файлов.
- Настройка Nextcloud для работы с облачным хранилищем (AWS S3).

После вашего знакомства с веб-сервером LAMP и основами управления текстовым содержимым в предыдущей главе я уверен, что вы начали думать о других технологиях, которые сможете использовать. Возможно, вы не сразу решили вернуться к проблеме корпоративного обмена файлами, но об этом стоит поговорить, чтобы больше узнать о том, как Linux предоставляет веб-сервисы.

Обмен файлами, о котором я говорю, не имеет ничего общего с незаконным распространением фильмов и книг, защищенных авторским правом. Я также не имею в виду протоколы однорангового обмена, такие как BitTorrent, хотя они могут использоваться в совершенно законных целях. Скорее я говорю о компаниях, выпускающих

большое количество документов и других медиафайлов, которые должны быть доступны и в то же время должны надежно и безопасно поддерживаться.

Представьте себе проект с участием десятков разработчиков, разделенных на несколько команд и распределенных по трем или четырем физическим точкам. В дополнение к своему коду, который может находиться в частных репозиториях, каждая команда выпускает множество проектных документов, видеороликов, предложений по финансированию проектов и записей видеочатов, не говоря уже о контрактах и технических заданиях в формате PDF. Чтобы система такого типа работала, вам необходимо обеспечить безопасный доступ к ней в небезопасной сети, но таким образом, чтобы тщательно контролировать этот доступ и чтобы только нужные люди могли видеть определенные ресурсы.

Попробуем установить сервер Nextcloud, чтобы проиллюстрировать, как наладить такой обмен файлами, используя некоторые важные навыки администрирования Linux. Вы увидите новый менеджер пакетов snapd в Ubuntu, узнаете о расширении ограничений конфигурации Apache, поймете, как приложения управляют своей конфигурацией, и увидите, как хранилище данных может «фотографировать» несколько устройств.

8.1. Корпоративный файлообменник и Nextcloud

Nextcloud — это набор программ с открытым исходным кодом, который может пользоваться памятью для сохранения, редактирования и применения широкого спектра типов документов и предоставления таких услуг, как хостинг аудио- и видеозвонков. Nextcloud также предоставляет клиентские приложения, которые позволяют пользователям, работающим в Linux, Windows, MacOS, взаимодействовать с медиаресурсами.

С помощью Nextcloud вы можете создать свою особую версию Dropbox или Google Drive, но на своих условиях и не беспокоиться о неожиданных изменениях в соглашениях о доступности или в обслуживании/конфиденциальности.

ПРИМЕЧАНИЕ

> Nextcloud — это ответвление старого проекта ownCloud. В то время как оба проекта одинаково хорошо работают, у Nextcloud, похоже, есть более сильная команда активных разработчиков и более богатый набор функций.

Отлично. Nextcloud имеет некоторые реальные преимущества. Но если вы будете делать все самостоятельно, значит, вы готовы к затратам и сложностям размещения, репликации и резервного копирования данных. Стоит ли Nextcloud всех хлопот и затрат, когда вы можете получить достаточное хранилище за небольшую плату или даже бесплатно, используя Dropboxes и Google Drives всего мира?

Хорошие новости: вы можете обеспечить хранение обоими способами. Особо конфиденциальные данные вы можете хранить внутри компании. Но, как вы вскоре увидите, вы также можете создать сервер Nextcloud в качестве своего интерфейса, чтобы точно контролировать, как пользователи взаимодействуют с вашими медиафайлами, и автоматически сохранять данные в безопасных и надежных сторонних сервисах, в том числе Dropbox, Google Drive и Amazon S3. Если в конце концов вы обнаружите, что вам нужно перенести данные из, скажем, Dropbox, то сможете сделать это, причем ваши пользователи ничего не заметят.

Но достаточно ли все это касается Linux для того, чтобы оправдать написание целой главы в этой книге? Я думаю, что да. Конечно, Nextcloud — это сторонняя технология, которая работает только поверх самой Linux, но тем не менее она тесно связана с ОС Linux. Более того, переход к процессу развертывания позволит вам познакомиться с навыками администрирования Linux, перечисленными ранее.

8.2. Установка Nextcloud с помощью моментальных снимков

На данный момент я готов поспорить, что вы уже немного расстроены конфликтами между менеджерами пакетов, используемыми в дистрибутивах Linux. Как я уже говорил, это не APT против Yum. Есть также менеджеры, используемые конкретными дистрибутивами, такими как Arch Linux и SUSE.

Разве было бы не удобно, если бы единый набор приложений Linux можно было одинаково установить во всем диапазоне дистрибутивов и управлять им с помощью одного инструмента? Почему люди не могут просто обойтись без лишних сложностей? У меня есть решение по крайней мере одной из этих проблем: моментальные снимки Ubuntu. Я должен отметить, что ни один из проектов в этой книге не основан на снимках, поэтому, если данная тема вас не интересует, ничто не мешает вам пропустить этот раздел и присоединиться к нам в разделе 8.3 через несколько страниц. Никаких обид.

О чем это я говорил? Ах да. *Snaps* — это пакеты программного обеспечения, которые по своей природе полностью автономны. Вам потребуется лишь дистрибутив Linux, совместимый с мгновенными снимками, и название пакета. (На момент написания этой книги большинство дистрибутивов, в том числе CentOS и OpenSUSE, либо уже включают такую функциональность, либо близки к этому.) Если вы запустите следующую команду из терминала Linux, она автоматически установит полный пакет создания Blender 3D вместе со всеми зависимостями и расширениями среды, которые ему нужны. Обновления и патчи также будут обрабатываться неявно.

```
# snap install blender
```

Blender — только пример. Поскольку это графическая программа, имеет смысл установить ее лишь на настольную версию Linux. Вы не получите хорошего результата на сервере.

Snap все еще находится в стадии разработки. Поскольку существует множество приложений, которые пока не поддерживают мгновенные снимки, APT и ей подобные в ближайшем будущем не рассматриваются. Но у технологии есть значительные преимущества, и к ней присоединяется все больше крупных игроков отрасли. Вы можете получить выгоду, просто рекламируя себя.

ПРИМЕЧАНИЕ

Ubuntu Core — это специальный облегченный дистрибутив, созданный в первую очередь для работы с устройствами Интернета вещей (IoT), такими как интеллектуальные устройства и автомобили с сетевыми возможностями, а также со множеством контейнеров на базе Docker. По своей сути, Core использует только мгновенные снимки для управления пакетами.

Система snap больше, чем менеджер пакетов. Как видно из рис. 8.1, мгновенные снимки сами по себе являются изолированными «песочницами» с ограниченным доступом к другим системным ресурсам.

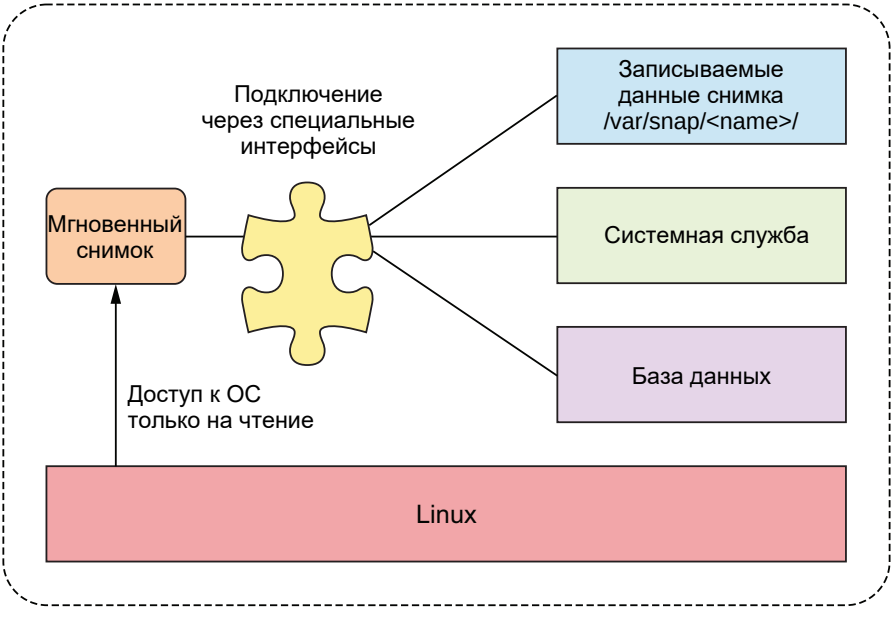


Рис. 8.1. Структура моментальных снимков: обратите внимание на то, как тщательно контролируется обмен данными между моментальными снимками и системными ресурсами, чтобы обеспечить изоляцию для безопасности

Как показано на рисунке, связь с другими снимками достигается через систему интерфейсов, которые подключаются только там, где это необходимо. Снимки получают доступ к специальным местам в файловой системе (в `/var/snap/<snapname>/`),

куда они могут записывать свои данные. Если snap еще не установлен на вашем компьютере, вам понадобится пакет snapd:

```
# apt install snapd
```

Пакет snapd может быть еще недоступен для CentOS 7, но команда dnf install snapd будет работать в Fedora 25

Разработчики приложений уже создали сотни моментальных снимков, управление которыми осуществляется с помощью нескольких источников, включая Ubuntu Store (snapcraft.io). Вы можете поискать доступные снимки из командной строки, используя `snap find` и ключевое слово, описывающее то, что вы ищете. Вот некоторые результаты поиска моментальных снимков, содержащих ключевое слово `server`:

```
$ snap find server
Name                       Version   Developer     Summary
minecraft-server-jdstrand  16.04.10  jdstrand      Minecraft server packaging
                                                   for Ubuntu Core
thinger-maker-server       1.3.0     thinger       Thinger.io Internet Of
                                                   Things Server
rocketchat-server          0.57.2    rocketchat    Group chat server for 100s,
                                                   installed in seconds.
tika-server                1.16      magicaltrout  Tika Server for metadata
                                                   discovery and extraction
kurento-media-server       6.6.1     surjohn       kurento-media-server on
                                                   snappy
nats-server                0.9.4     nats-io       High-Performance server
                                                   for NATS
kms-serverproxy-demo       6.6.0     surjohn       kurento service server
                                                   side proxy demo
lxd-demo-server            git       stgraber      Online software demo
                                                   sessions using LXD
[...]
```

Оказывается, Nextcloud был одним из первых крупных проектов приложений, которые добавили свой пакет в качестве мгновенного снимка. Выполнение команды `snap install nextcloud` на совершенно новой и чистой виртуальной машине Ubuntu 17.04 VirtualBox предоставит вам полнофункциональный сервер Nextcloud без необходимости предварительно устанавливать все элементы LAMP вручную:

```
# snap install nextcloud
2017-07-25T21:07:41-04:00 INFO cannot auto connect core:core-support-plug
  to core:core-support: (slot auto-connection), existing connection state
  "core:core-support-plug core:core-support" in the
  way
nextcloud 11.0.3snap7 from 'nextcloud' installed
```

Это сообщение об ошибке не повлияет на установку; выход из системы и повторный вход перед установкой полностью компенсируют ошибку

Когда снимок установлен, используйте браузер для перехода в корень вашей виртуальной машины (указав IP-адрес виртуальной машины, который, как вы помните, можно получить с помощью команды `ip addr`). После загрузки страницы вас попросят создать учетную запись администратора — и вы в деле.

Почему Ubuntu 17.04 и почему VirtualBox?

Кажется, что интеграция технологии моментальных снимков в среду LXC немного медленнее, чем ожидалось, и она еще недоступна для контейнеров LXC до Ubuntu 16.10. Поскольку я сейчас использую Ubuntu 16.04 на своей рабочей станции, запуск виртуальной машины VirtualBox, созданной на образе Ubuntu 17.04 ISO, является самым быстрым и простым решением.

Это еще одна иллюстрация ценности обучения использованию нескольких технологий виртуализации: если одна из них не работает, вероятно, сработает другая.

8.3. Установка Nextcloud вручную

Если установка мгновенных снимков Nextcloud настолько проста, зачем вам делать это старомодным ручным способом? Вот несколько причин. Во-первых, снимки бывают слишком простыми и не допускают большой гибкости в разработке и конфигурации. Во-вторых (по крайней мере когда я пишу это), нельзя добавлять и полностью администрировать приложения с помощью мгновенной версии. Вы скоро увидите, насколько это может быть необходимо. А ручная установка ваших приложений дает больше знаний о внутренней работе вашей системы Linux, а также об определении и преодолении проблем.

8.3.1. Предварительные требования к оборудованию

Всегда полезно посмотреть документацию приложения, чтобы убедиться, что у вас достаточно аппаратных и программных средств для выбранной нагрузки. На рис. 8.2 показана веб-страница системных требований Nextcloud. Если вы планируете разместить простой, мало используемый сервер, обеспечивающий работу нескольких десятков пользователей, то обнаружите, что Nextcloud довольно легко справится, не требуя ничего, что не может быть обработано готовым контейнером.

Любая минимальная конфигурация старого оборудования будет хорошо работать для нашего примера технологического тестирования, но я бы не стал полагаться на один контейнер LXC, работающий на старом ПК, при обслуживании десятков тысяч пользователей и терабайт данных.

Планируете развертывания в масштабах предприятия? Nextcloud предоставляет полезное многоуровневое руководство с рекомендациями по развертыванию на полнофункциональных платформах (mng.bz/8ZAO). Вот, например, что Nextcloud рекомендует для небольшой рабочей группы до 150 пользователей, которые имеют доступ к данным объемом до 10 Тбайт.

- Один сервер с двухъядерным процессором.
- 16 Гбайт ОЗУ.

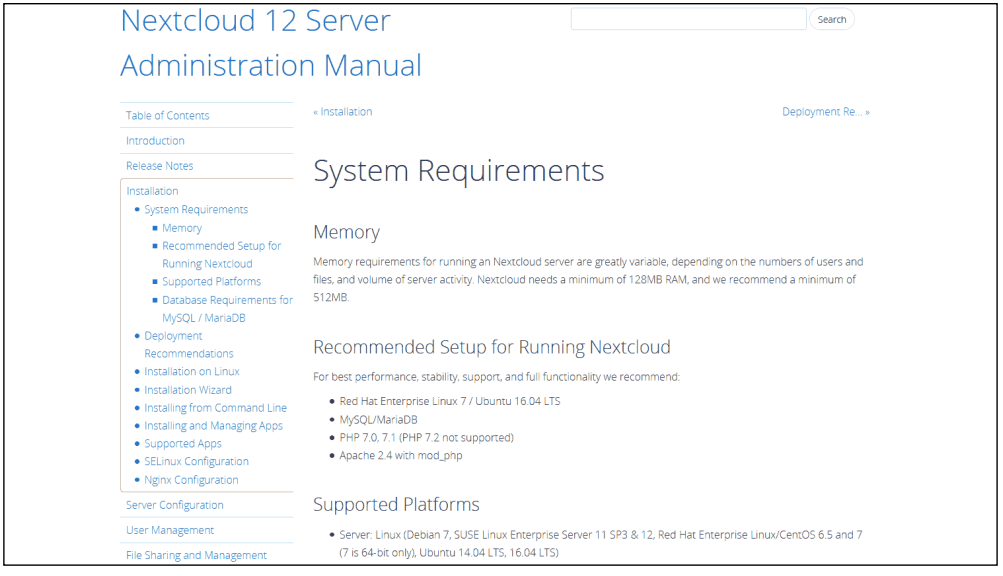


Рис. 8.2. Требования к аппаратному и программному обеспечению для рекомендуемых и минимальных установок Nextcloud

- Аутентификация с применением облегченного протокола доступа к каталогам (LDAP, широко используемый протокол распределенной информации).
- Red Hat Enterprise Linux или Ubuntu 16.04 с поддержкой поставщиков.
- Apache с сертификатом шифрования TLS/SSL.
- База данных MySQL или MariaDB.
- Файловая система Btrfs, смонтированная с параметром `nodatacow` для разделов данных Nextcloud, чтобы обеспечить *нулевое* время простоя резервного копирования.
- Кэширование с помощью утилиты memcache для ускорения доступа.

ПРИМЕЧАНИЕ

Btrfs — это файловая система для Linux, которая менее распространена, чем ext4 (кратко упоминается в главе 1). Она была разработана для обеспечения исключительной производительности и надежности данных в больших масштабах. Но начиная с выпуска 7.4 (август 2017 г.) Red Hat Enterprise Linux не поддерживает Btrfs.

8.3.2. Построение сервера LAMP

Следующий шаг на данный момент должен быть достаточно простым. Я бы порекомендовал запустить новый контейнер или виртуальную машину, а не использовать тот, что остался от вашего проекта MediaWiki, хотя бы по той причине, что это будет

хорошей практикой. Вот все пакеты, которые вам понадобятся для вашего сервера в одной команде. Я добавил wget и nano на тот случай, если они еще не установлены:

```
# apt install apache2 mariadb-server libapache2-mod-php7.0 \
  php7.0-gd php7.0-json php7.0-mysql php7.0-curl php7.0-mbstring \
  php7.0-intl php7.0-mcrypt php-imagick php7.0-xml php7.0-zip \
  wget nano
```

Чтобы базовый образ был как можно меньшим, такие пакеты, как nano, часто не устанавливаются по умолчанию в новые контейнеры LXC

Если вы не против использовать MySQL, а не MariaDB и находитесь на сервере Ubuntu, то можете сэкономить время на наборе и задействовать метапакет LAMP-сервера, о котором я упоминал в предыдущей главе. Опять же не забывайте добавлять символ ^ в конце имени пакета:

```
# apt install lamp-server^
```

После установки не забудьте запустить средство безопасной инсталляции MySQL:

```
# mysql_secure_installation
```

Если вы выбрали MariaDB и обнаружили, что вам нужно использовать с этой командой `sudo`, вспомните короткий путь, который я показал вам в главе 7:

```
MariaDB [(none)]> SET PASSWORD = PASSWORD('your-password');
MariaDB [(none)]> update mysql.user set plugin = 'mysql_native_password'
   where User='root';
MariaDB [(none)]> FLUSH PRIVILEGES;
```

Установив программное обеспечение LAMP и имея работающую базу данных, вы готовы настроить Apache для работы с вашим приложением.

8.3.3. Конфигурирование Apache

Чтобы гарантировать, что Apache сможет взаимодействовать с Nextcloud, необходимо выполнить несколько относительно простых настроек. Во-первых, вы должны включить пару модулей Apache с помощью инструмента a2enmod. Модуль rewrite используется для перезаписи URL-адресов в режиме реального времени по мере того, как они перемещаются между клиентом и сервером. Модуль headers выполняет аналогичную функцию для заголовков HTTP.

```
# a2enmod rewrite
# a2enmod headers
```

Если вы не планируете использовать этот сервер для каких-либо других целей, то размещение файлов приложения Nextcloud в корне документа Apache будет работать. Поскольку значение записи `DocumentRoot` в файле `000-default.conf` в вашем каталоге `/etc/apache2/sites-available/` уже указывает на `/var/www/html/`, делать больше нечего. Но размещение файлов данных Nextcloud в корневом каталоге документов по умолчанию несет потенциальную угрозу безопасности.

Вы, вероятно, захотите поместить Nextcloud в какое-то другое место в вашей файловой системе.

Есть два способа сообщить Apache, как найти файлы сайта, которые не находятся в корне документа. В Ubuntu предлагается добавить новый раздел в существующий файл `000-default.conf`, который содержит всю необходимую информацию. Большинство предпочитают создать новый файл `.conf` в каталоге `/etc/apache2/sites-available/` для каждого нового сервиса. Оба метода отлично работают, но вот как должен выглядеть отдельный файл, если вы поместили приложение в `/var/www/`, а не в корень документа (листинг 8.1).

Листинг 8.1. Содержимое файла /etc/apache2/sites-available/nextcloud.conf

```
Alias /nextcloud "/var/www/nextcloud/"

<Directory /var/www/nextcloud/>
  Options +FollowSymlinks
  AllowOverride All

 <IfModule mod_dav.c>
 Dav off
 </IfModule>

 SetEnv HOME /var/www/nextcloud
 SetEnv HTTP_HOME /var/www/nextcloud

</Directory>
```

Связывает содержимое каталога /var/www/nextcloud/ с хостом Nextcloud (или сайтом)

Задает переменные среды, которые определяют, как работает приложение Nextcloud

Схожая директива по методу Ubuntu предполагает добавление раздела в файл `000-default.conf`. Код может выглядеть примерно так, как в листинге 8.2.

Листинг 8.2. Пример директивы Nextcloud в установочном файле Apache 000-default.conf

```
<VirtualHost *:443>
    ServerName bootstrap-it.com
    DocumentRoot /var/www/nextcloud
    ServerAlias bootstrap-it.com/nextcloud
</VirtualHost>
```

Обратите внимание, как в этом примере используется безопасный порт 443 HTTP (https)

Перенаправляет трафик, нацеленный на адрес bootstrap-it.com/nextcloud, в пользовательский корень документа

Как вы можете видеть на рис. 8.3, когда Apache считывает этот файл конфигурации, он перенаправляет весь входящий трафик, адресованный на example.com/nextcloud, в файлы приложения в `/var/www/`, предполагая, что ваш домен — example.com. IP-адрес будет работать так же хорошо, как и раньше.

Наконец, вам нужно создать символическую ссылку в каталоге `/etc/apache2/sites-enabled/`, указывающую на файл `nextcloud.conf`, который вы создали в `/etc/apache2/sites-available/`:

```
# ln -s /etc/apache2/sites-available/nextcloud.conf \
 /etc/apache2/sites-enabled/nextcloud.conf
```

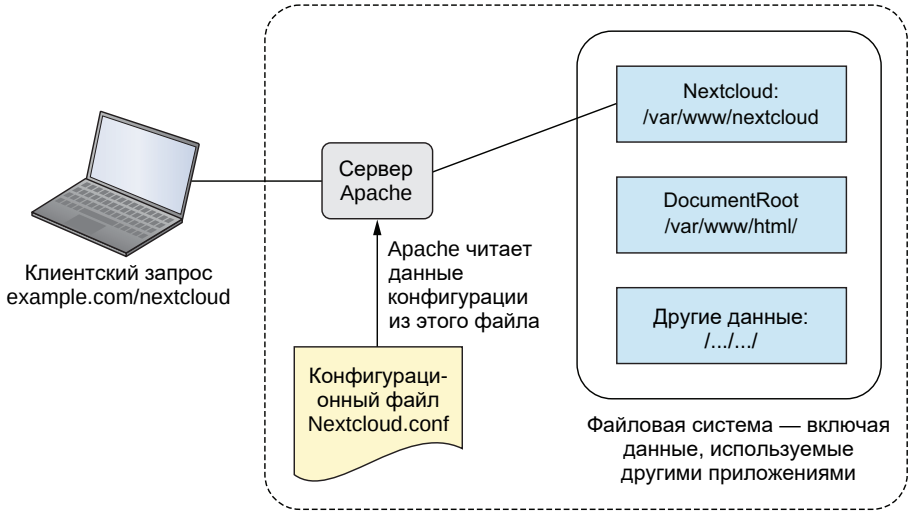


Рис. 8.3. Apache считывает файлы конфигурации в /etc/apache2/sites-enabled/ и использует их настройки для перенаправления запросов

Но зачем? И что такое символическая ссылка? При запуске Apache считывает содержимое `/etc/apache2/sites-enabled/` и ищет конфигурации сайта для загрузки. Эти конфигурации на самом деле существуют не в `/etc/apache2/sites-enabled/`, но там будут символические ссылки на реальные файлы в `/etc/apache2/sites-available/`.

Почему бы не велеть Apache сначала прочитать `/etc/apache2/sites-available/` напрямую? Потому что, используя для этого символические ссылки, можно легко и удобно быстро отключить сайт, а затем, когда вы закончили вносить изменения, снова включить его. Вместо того чтобы удалять и перезаписывать настоящий файл, вам нужно лишь использовать простую в управлении ссылку на него.

Что такое *символические ссылки*? Это объекты, которые представляют файлы или каталоги, находящиеся в других местах файловой системы. Они позволяют пользователю выполнять или просматривать ресурс в одном месте, даже если он находится в другом месте. Больше о символических ссылках вы узнаете, когда дойдете до главы 12.

8.3.4. Скачивание и распаковка Nextcloud

Вы можете загрузить самый последний пакет Nextcloud со страницы установки Nextcloud (nextcloud.com/install). Если вы устанавливаете в контейнер или виртуальную машину либо с сервера без установленного графического интерфейса, то наиболее удобный способ — получить URL-адрес загрузки пакета и скачать пакет, используя командную строку.

Один из быстрых способов получить этот URL-адрес с сайта Nextcloud с помощью обычного сеанса на вашем компьютере — щелкнуть на вкладке **Download** (Скачать) в разделе **Get Nextcloud Server** (Получить сервер Nextcloud), а затем нажать кнопку **Details and Download options** (Подробности и параметры загрузки) (рис. 8.4). Щелкните правой кнопкой мыши на ссылке **.tar.bz2** и выберите в меню пункт **Copy Link Address** (Копировать адрес ссылки).

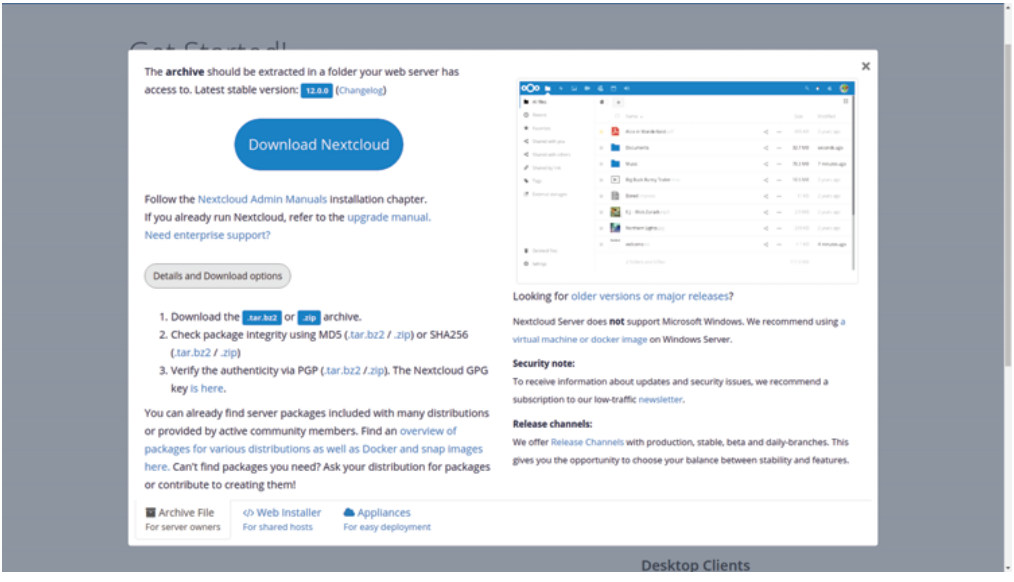


Рис. 8.4. Ссылки на загрузочные архивы Nextcloud: будет доступен формат .tar.bz2 или .zip

Вы можете скопировать URL файла `.tar.bz2` в команду `wget`, щелкнув правой кнопкой мыши в терминале и выбрав пункт **Insert** (Вставить) или нажав сочетание клавиш **Shift+Ctrl+V**. Вот как будет выглядеть команда:

```
$ wget https://download.nextcloud.com/server/releases/nextcloud-12.0.0.tar.bz2
```

Не забудьте выбрать ссылки на хеш MD5 или SHA256, отображаемые при нажатии кнопки **Details and Download options** (Подробности и параметры загрузки). Затем убедитесь, что эти значения идентичны хешам, которые вы сгенерируете для загруженного архива.

Вы заметите, что скачанный вами файл имеет расширение `.tar.bz2`, а не `.tar.gz`, как в главе 4. Архив `.gz` создается с использованием другого алгоритма сжатия, нежели архив `.bz2`. Какой из них лучше? Сжатие архивов GZIP занимает меньше времени (и ресурсов), но дает файлы большего размера по сравнению с BZ2. Таким образом, ваш выбор будет зависеть от того, что для вас более важно — скорость или объем архива.

Для распаковки архива `.tar.bz2` используется аргумент `xjf`, а не `xzf`, который вы указывали бы для `.gz`:

```
$ tar xjf nextcloud-12.0.0.tar.bz2
```

Аргумент j используется для распаковки сжатого BZ2 архива

Следующий шаг — копирование распакованных файлов и каталогов в их новый дом, который согласно рекомендациям, приведенным ранее, будет находиться в `/var/www/`, то есть вне корня документа.

Добавление `-r` к команде копирования позволяет рекурсивно скопировать файлы, включая подкаталоги и их содержимое:

```
# cp -r nextcloud /var/www/
```

Еще два маленьких шага — и все готово. Для работы Apache потребуется полный доступ ко всем файлам в каталогах Nextcloud. Вы могли бы работать как пользователь **root**, но это означает, что вам необходимо предоставить посетителям полномочия **root** для доступа к этим файлам. Как вы догадываетесь, предоставление такого рода доступа к вашим файлам в Интернете небезопасно.

Многие веб-серверы работают от имени специального системного пользователя под именем **www-data**. Следующая команда, `chown` (с которой вы уже встречались в главе 4), позволяет передать право владения всеми этими файлами группе пользователей **www-data** веб-сервера. Использование заглавных букв `-R` (подобно строчным `-r`, которые вы указывали вместе с `cp`) говорит о том, что команда будет рекурсивно применяться ко всем файлам и каталогам в структуре каталогов:

```
# chown -R www-data:www-data /var/www/nextcloud/
```

Apache не имеет представления о том, чем мы занимались, пока не перечитает конфигурацию. Вам лучше перезапустить сервис:

```
# systemctl restart apache2
```

Если в этот раз перезапуск не был успешным, отметьте для себя все сообщения об ошибках и посмотрите, есть ли что-нибудь, что можно исправить. Вы также можете немного внимательнее изучить журналы, просмотрев последние десять записей с помощью команды `tail`. Например, в файле `nextcloud.conf` может быть ссылка на определенную строку:

```
# journalctl | tail
```

Но если все прошло хорошо, введите в браузере IP-адрес вашего контейнера, а затем `nextcloud`. Вы попадете на страницу, где вас попросят создать новую учетную запись администратора и предоставить работающие учетные данные для входа в базу данных MariaDB. Если вы не создали для этой цели другую учетную запись пользователя базы данных, вы будете использовать **root** и пароль, который указали ранее:

```
10.0.3.36/nextcloud
```

ПРИМЕЧАНИЕ

Ассоциирование ресурсов базы данных с пользователями без полномочий root разумно не только в плане безопасности, но и в плане логистики. Если у вас есть несколько приложений, использующих ресурсы одной базы данных, рекомендуется разделять приложения, создавая несколько изолированных учетных записей.

Как только ваша информация будет обработана, вам будут предоставлены ссылки на клиентские приложения Nextcloud, а затем вы попадете в консоль администрирования, как можно увидеть на рис. 8.5. Здесь вы можете загружать и просматривать документы и медиафайлы, а также обмениваться ими.

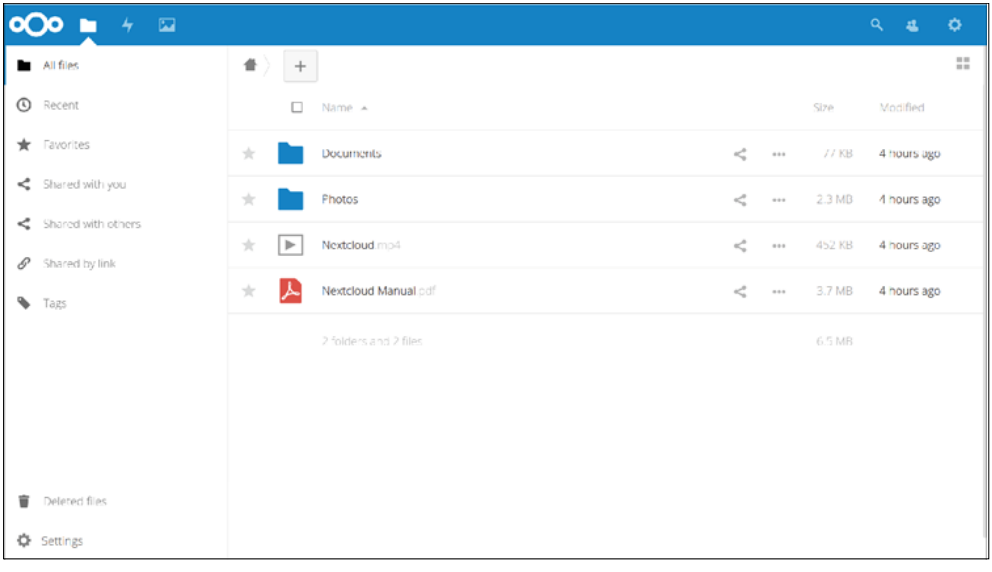


Рис. 8.5. Основная консоль Nextcloud с примером папок и файлов. Вы можете работать здесь с объектами так же, как в файловом менеджере ОС

Как администратор сайта, вы также можете создавать группы и пользователей, назначать разрешения и квоты и управлять работой сайта. Далее мы посмотрим, как администрировать Nextcloud.

8.4. Администрирование Nextcloud

По мере усложнения сайта Nextcloud и изменения ваших потребностей (например, возможно, более эффективное управление растущими ресурсами лучше осуществлять с помощью сценариев) вам иногда придется работать с административными ресурсами на уровне командной строки. Было бы хорошо знать, где что находится и как выглядит.

Вместо того чтобы использовать каталог `/etc/` так, как это делают многие приложения Linux, Nextcloud сохраняет свой основной файл конфигурации `config.php` в каталоге `/var/www/nextcloud/config/`. Файл содержит информацию об аутентификации и среде, а также сведения о ключевых подключениях к базе данных. Вот пример содержимого этого файла:

```
<?php
$CONFIG = array (
  'instanceid' => 'ociu535bqczx',
  'passwordsalt' => '',
  'secret' => '',
  'trusted_domains' =>
  array (
    0 => '10.0.3.36',
  ),
  'datadirectory' => '/var/www/nextcloud/data',
  'overwrite.cli.url' => 'http://10.0.3.36/nextcloud',
  'dbtype' => 'mysql',
  'version' => '12.0.0.29',
  'dbname' => 'nextcloud',
  'dbhost' => 'localhost',
  'dbport' => '', 'dbtableprefix' => 'oc_',
  'dbuser' => 'oc_admin',
  'dbpassword' => '',
  'installed' => true,
);
```

Все зашифрованные коды аутентификации были отредактированы по соображениям безопасности

Публичный домен сервера или в данном случае IP-адрес

Расположение данных об учетной записи и документе в файловой системе

Как и любое хорошее приложение для Linux, Nextcloud имеет полнофункциональную оболочку командной строки, которая называется `occ command`. Чтобы использовать ее, вам нужно добавить действительно подробную строку для запуска команд от имени пользователя `www-data`, работающего с PHP. Ввод `sudo -u www-data php occ -h` отображает справку по основному синтаксису, а `list` выводит полный список доступных команд:

```
$ cd /var/www/nextcloud
$ sudo -u www-data php occ -h
$ sudo -u www-data php occ list
```

Пока не будем разбирать эти инструменты командной строки и вернемся к консоли браузера. Как вы можете видеть на рис. 8.6, при щелчке на значке шестеренки в правом верхнем углу отображаются ссылки на страницу **Users** (Пользователи) (где вы управляете пользователями и группами), страницу **Apps** (Приложения) (с меню доступных приложений, которые можно включить) и страницу **Admin** (Администрирование) (где вы управляете настройками безопасности, службами и приложениями).

Если вы перейдете на страницу администратора, то, вероятно, увидите предупреждения о безопасности и настройке. Как видно из рис. 8.7, вызывает беспокойство тот факт, что мой сайт не настроен на использование зашифрованного протокола HTTPS для передачи данных. Nextcloud определенно знает толк в данном вопросе, и мы еще поговорим об этом в главе 9.

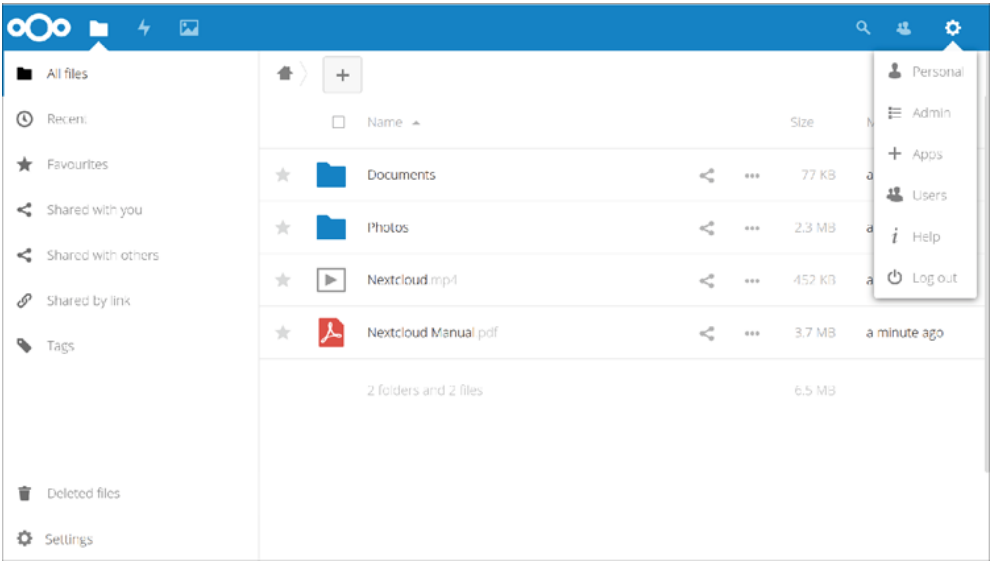


Рис. 8.6. При щелчке на значке шестеренки открывается меню со ссылками на ресурсы, соответствующие вашей учетной записи

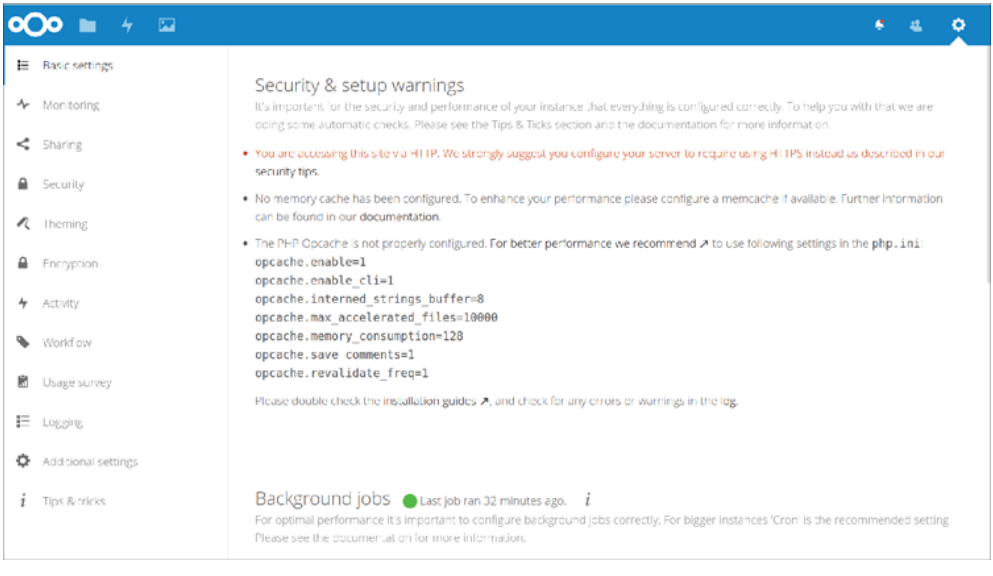


Рис. 8.7. Страница администратора со ссылками на различные административные приложения *(слева)*, а также предупреждениями и информацией о состоянии *(посередине)*

HTTPS предоставляет полное шифрование данных, когда они перемещаются между сервером (таким как тот, на котором сейчас работает Nextcloud) и браузерами на компьютерах ваших пользователей. Это очень важно, ведь вы не хотите,

чтобы ваши данные передавались кому-либо другому в сети. Но вы также должны заботиться о данных, когда они просто сохраняются.

Шифрование файлов на вашем сервере не позволит никому, кто каким-либо образом получит доступ к вашим дискам, смонтировать их и прочитать их содержимое. Кто это может быть? Представьте, что кто-то взломал вашу сеть. Или кто-то физически похищает ваш сервер. Если файлы зашифрованы, похитители никак не воспользуются ими.

Чтобы включить шифрование, щелкните на ссылке **Encryption** (Шифрование) на левой панели, а затем установите флажок **Enable Server-side Encryption** (Включить шифрование на стороне сервера). Как показано на рис. 8.8, будет выведен список последствий, которые следует учитывать. Прочитав все это, нажмите кнопку **Enable Encryption** (Разрешить шифрование).

Теперь нет пути назад. Только вперед...

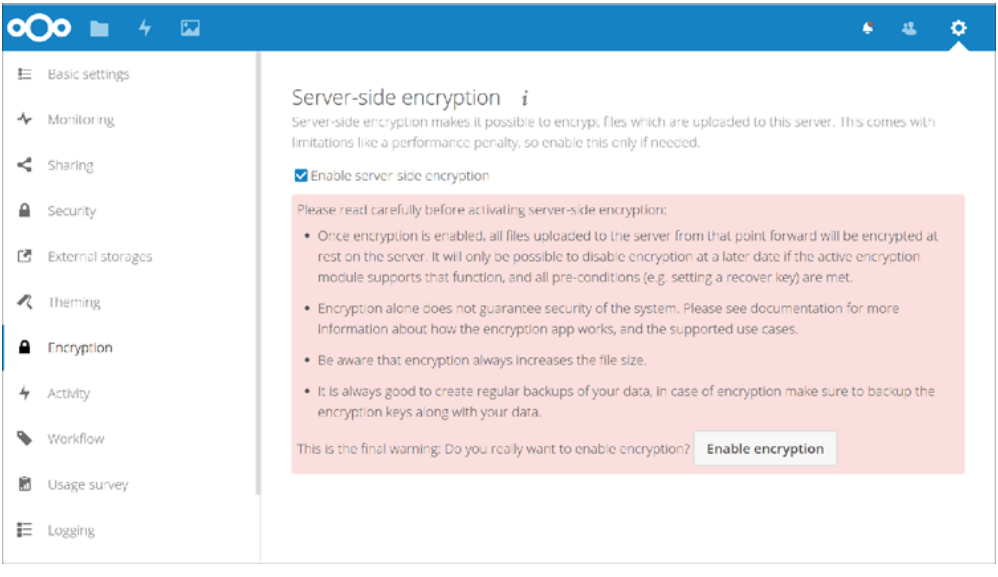


Рис. 8.8. Включение шифрования на стороне сервера в Nextcloud — простой процесс, который можно запустить одним нажатием, но вы должны помнить о последствиях

8.5. Использование AWS S3 в качестве основного хранилища Nextcloud

В хранении самое главное — найти место для всех данных. И поскольку все устройства хранения в конечном итоге без предупреждения выходят из строя, вам потребуется несколько копий для каждого устройства. Выяснение того, как подготовить, подключить и поддерживать такие массивы хранения, отнимает много времени, а поддерживать их в рабочем состоянии относительно дорого. Но есть и хорошая новость.

Облачное хранилище обходится сравнительно дешево, и, как вы можете прочитать в моей книге *Learn Amazon Web Services in a Month of Lunches book* (Manning, 2017), его просто настроить. Поскольку крупные облачные провайдеры вкладывают огромные средства в обеспечение безопасности и восстановления данных, их услуги в значительной степени более надежны, чем все, что вы могли бы организовать сами. Поэтому использование облачных данных в качестве бэкенда для локально размещенного сайта Nextcloud — действительно грамотный подход. Вот как это работает.

Сначала вам нужно включить пакет приложений External Storage Support. Щелкните на значке шестеренки в правом верхнем углу, а затем выберите страницу **Apps** (Приложения) и ссылку **Disabled Apps** (Отключенные приложения) на левой панели. Как показано на рис. 8.9, в списке появится опция поддержки внешнего хранилища. Нажмите кнопку **Enable** (Включить).

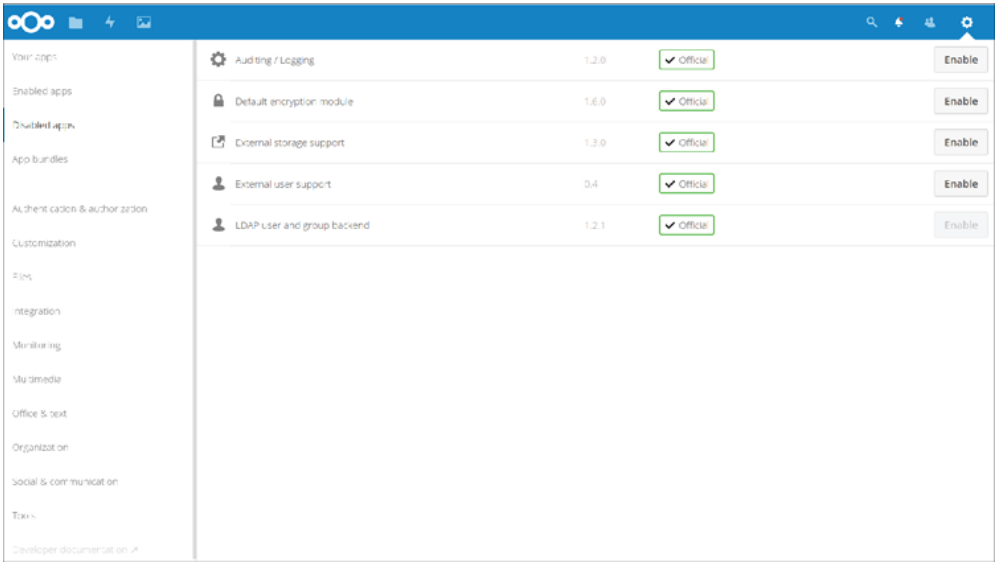


Рис. 8.9. Список доступных на данный момент приложений, включая поддержку внешнего хранилища

Из командной строки на любом компьютере с CLI AWS, установленным и настроенным для вашей учетной записи AWS (как вы делали это в главе 5), создайте новый сегмент памяти с глобально уникальным именем:

```
$ aws s3 mb s3://nextcloud32327
```

Опять же как вы это делали в главе 5, получите набор ключей доступа к учетной записи со страницы **Your Security Credentials** (Ваши учетные данные безопасности) в консоли AWS. Вы также можете использовать существующий набор ключей, если он у вас есть.

Теперь вернитесь к консоли Nextcloud, выберите страницу **Admin** (Администрирование) в раскрывающемся списке под шестеренкой, а затем щелкните на ссылке **External Storage** (Внешнее хранилище), которая должна быть на левой панели. Откроется страница **External Storage** (Внешнее хранилище), где вы можете щелкнуть на раскрывающемся списке **Add Storage** (Добавить хранилище) и выбрать в нем **Amazon S3**. (В список также входят Dropbox и Google Drive.)

Вам будет предложено ввести сегмент S3, который вы хотите использовать, а также ключ доступа и секретный ключ. Все остальные поля конфигурации для применения нестандартных портов или шифрования SSL являются необязательными. Когда вы закончите, щелкните на флажке справа, чтобы сохранить настройки и отключить Nextcloud для аутентификации в AWS.

Если все в порядке, вы увидите зеленый кружок слева, как показано на рис. 8.10. Если что-то не работает, то, скорее всего, вы каким-то образом использовали неверные ключи аутентификации. Не повредит убедиться в наличии сетевого подключения к Интернету и, в частности, к AWS.

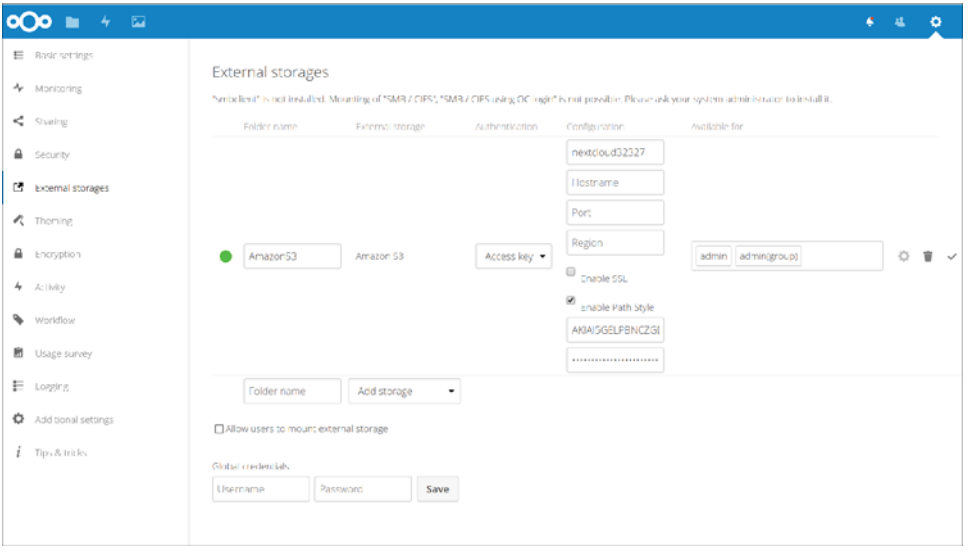


Рис. 8.10. На странице External Storages (Внешние хранилища) для Amazon S3 отображено успешное подключение к сегменту S3

Вы можете проверить новую конфигурацию хранилища, скопировав и вставив файл со своего компьютера в папку в консоли Nextcloud. Затем из интерфейса командной строки AWS просмотрите содержимое вашего сегмента:

```
$ aws s3 ls s3://nextcloud32327
testfile.pdf
```

Конечно, вам нужно будет проверить это и другим способом. Скопируйте локальный файл в сегмент из командной строки:

```
$ aws s3 cp test.txt s3://nextcloud32327
```

Файл `test.txt` должен появиться в вашей консоли. Великолепная многоплатформенная интеграция хранилищ!

Резюме

- Планирование стратегии хранения данных требует нахождения баланса между стоимостью и простотой использования, с одной стороны, и безопасностью, и контролем — с другой.
- Мгновенные снимки Ubuntu — это система управления пакетами, не зависящая от дистрибутива, которая также обеспечивает безопасную изолированную рабочую среду только для чтения.
- Чтобы узнать, как маршрутизировать входящий HTTP-трафик, Apache считывает либо несколько определений хоста в одном файле `.conf`, либо несколько файлов конфигурации хоста.
- Веб-серверы обычно предоставляют ресурсы посетителям, использующим специальную учетную запись пользователя веб-сервера, чтобы сбалансировать доступ и безопасность системы.
- Nextcloud хранит свои конфигурационные файлы в каталоге `/<DocumentRoot>/nextcloud/config`, в частности в файле `config.php`.
- Аутентификация Nextcloud в сегменте AWS S3 заключается в предоставлении имени сегмента и ключей доступа.

Ключевые термины

- *Моментальный снимок (snap)* — это пакетное приложение, которое можно установить автономно в большинстве современных дистрибутивов Linux с помощью команды `snapd`.
- *Символическая ссылка* — объект файловой системы, представляющий файл или каталог в другом месте. Символические ссылки в каталоге `/etc/apache2/sites-enabled/` указывают на файлы в `/etc/apache2/sites-available/`, которые Apache считывает при запуске.
- *Архивы* можно сжимать с использованием алгоритма BZ2 в качестве альтернативы gzip или zip.

Рекомендации по безопасности

- Избегайте хранения файлов данных Nextcloud в корне документа.
- Убедитесь, что все общедоступные файлы с вашего сайта принадлежат пользователю **www-data**, а не **root**.
- Храните изолированно ресурсы базы данных, используемые отдельными приложениями, и создавайте отдельные учетные записи пользователей.

Обзор команд

- `a2enmod rewrite` подключает модуль перезаписи, поэтому Apache может редактировать URL-адреса при их перемещении между клиентом и сервером.
- `nano /etc/apache2/sites-available/nextcloud.conf` создает или редактирует файл конфигурации хоста Apache для Nextcloud.
- `chown -R www-data: www-data /var/www/nextcloud/` меняет владельца и группу всех файлов сайта на пользователя www-data.
- `sudo -u www-data php occ list` использует CLI Nextcloud для вывода списка доступных команд.
- `aws s3 ls s3://nextcloud32327` выводит содержимое сегмента S3.

Самотестирование

1. Что из нижеперечисленного представляет собой мгновенный снимок:
   а) драйвер Интернета вещей для управления устройствами;
   б) изолированная среда для безопасно работающих приложений;
   в) менеджер пакетов на основе Ubuntu;
   г) приложение, упакованное для использования в качестве автономного ресурса?
2. Какая из следующих команд выведет список всех пакетов, установленных в системе на данный момент:
   а) `snap list`;
   б) `snap find`;
   в) `snap ls`;
   г) `snapd ls`?
3. Что из нижеперечисленного является обоснованием для установки файлов Nextcloud не в корневой каталог документа:
   а) файлы в корне документа по умолчанию являются общедоступными;
   б) если есть другие хосты, обслуживаемые Apache, это может вызвать конфликты;
   в) конфигурация Nextcloud распознает только файлы в каталоге `/var/www/nextcloud/`;
   г) файлам в корне документа нельзя присвоить атрибуты владельца и группы, совместимые с Nextcloud?
4. Какая команда из нижеперечисленных создаст символическую ссылку на файл `nextcloud.conf`, который будет прочитан Apache:
   а) `ln -s /etc/apache2/sites-enabled/nextcloud.conf /etc/apache2/sitesavailable/nextcloud.conf`;
   б) `ln -s /etc/apache2/sites-available/nextcloud.conf /etc/apache2/sites-enabled/nextcloud.conf`;

в) `ln /etc/apache2/sites-enabled/nextcloud.conf /etc/apache2/sitesavailable/nextcloud.conf`;

г) `ln -h /etc/apache2/sites-enabled/nextcloud.conf /etc/apache2/sitesavailable/nextcloud.conf`?

5. Какая из следующих команд распакует архив `nextcloud-12.0.0.tar.bz2`:

а) `tar xcf nextcloud-12.0.0.tar.bz2`;

б) `tar xzf nextcloud-12.0.0.tar.bz2`;

в) `tar jf nextcloud-12.0.0.tar.bz2`;

г) `tar xjf nextcloud-12.0.0.tar.bz2`?

Ответы

1 — г; 2 — а; 3 — а; 4 — б; 5 — г.

9 Защита вашего веб-сервера

В этой главе

- Защита вашей инфраструктуры.
- Управление доступом к вашему серверу с помощью брандмауэров.
- Использование шифрования для защиты ваших данных.
- Ужесточение процесса аутентификации.
- Управление программным обеспечением и процессами.

Частичка *«веб»* в термине *«веб-сервер»* вводит в заблуждение. В конце концов, большинство инструментов безопасности, которые я собираюсь обсудить в этой главе, важны независимо от того, какой сервер вы используете. На самом деле *сервер* также является своего рода дополнительным уровнем защиты, поскольку все компьютеры нуждаются в защите. Тем не менее, поскольку по определению веб-серверы предназначены для передачи значительного количества внешнего трафика, их безопасность должна быть особенно приоритетной. И лучший способ проверить то, о чем вы узнаете в этой главе, — запустить веб-сервер Apache. Подумайте о том, чтобы сделать это прямо сейчас: `apt install apache2`.

В контексте IT *безопасность* — это защита оборудования, программного обеспечения, данных и цифровых услуг от несанкционированного доступа и повреждения. Принимая во внимание, что сетевые ресурсы компьютеров предназначены для того, чтобы быть доступными разным пользователям, сложно обеспечить такую работу, когда только правильные клиенты могут выполнять исключительно отведенные им операции.

Вы можете думать о безопасности как об искусном балансе между полезностью и риском. Когда вы посмотрите, сколько видов угроз безопасности уже существует

и как часто появляются новые, вы, вероятно, поймете, что баланс никогда не будет идеальным. Это соотношение наверняка придется часто пересматривать.

Нет единого инструмента или практики, которые могут охватить все аспекты безопасности. Хотя неплохо было бы создать себе контрольный список ключевых задач безопасности, но и этого недостаточно. Все самые успешные администраторы, с которыми я был знаком, были очень опытными и знающими и, казалось, также разделяли мое мнение о том, что ни одному программному обеспечению, поставщику, правительственному агентству, коллеге или даже близкому другу никогда нельзя полностью доверять. Очень легко ошибиться и оставить открытым для атаки важное окно. Везде должна использоваться вторая пара глаз и двойная проверка.

Что вы можете сделать, чтобы защитить свои серверы? Дело в мелочах. Нужно учитывать много-много мелких деталей. На самом деле их так много, что парочка из них перейдет в следующую главу. Однако в этой главе мы затронем основы, а затем углубимся в использование брандмауэров для контроля доступа к сети, защиты передачи данных сайта с помощью шифрования SSL/TLS и ограничения того, что можно сделать с помощью ресурсов сервера, посредством стратегического использования таких инструментов, как SELinux (Linux с улучшенной безопасностью) и системные группы.

9.1. Очевидные вещи

Начнем с низко висящих фруктов. В первую очередь при обеспечении безопасности нужно опираться на здравый смысл. Кроме того, вспомните о множестве лучших практик, о которых уже прочитали в этой книге. И, как бы просто это ни было, не стоит игнорировать основы.

- Делайте резервную копию ваших данных. Сегодня и каждый день.

  Неважно, что плохие парни сделают с вашим сервером, — если вы можете восстановить его из надежной резервной копии, значит, вы все еще в игре. Просмотрите еще раз главы 4 и 5, а затем составьте себе сценарий регулярного, автоматизированного, всеобъемлющего и проверяемого резервного копирования, которое позволит сберечь все ценное. Убедитесь, что доступно более одной архивной версии и что хотя бы один архив хранится вне сайта.

- Применяйте все обновления программного обеспечения вашей системы. Без исключения.

  О, всегда есть оправдания: вы боитесь, что обновления могут сломать то, от чего зависит ваше приложение, или может потребоваться перезагрузка, которая нарушит работу. Но сделайте это в любом случае. Не поймите меня неправильно: я понимаю, что это реальные проблемы. И что может быть хуже. Вот вам дружеское напоминание об обновлении вашей системы:

  ```
  # yum update
  ```

  Или для Ubuntu:

  ```
  # apt update
  # apt upgrade
  ```

ПРИМЕЧАНИЕ

Не забывайте, что менеджеры пакетов обновляют только те пакеты, которые были установлены через управляемые репозитории. Любые приложения, которые вы добавили вручную, останутся без патчей (и окажутся потенциально небезопасными), пока вы не примените патчи вручную или не отключите их.

Вы могли бы избежать большинства рисков в плане сбоев, создав тестовые (или *промежуточные*) среды (рис. 9.1), где запускаются зеркальные образы ваших приложений, которые надежно защищены от общедоступных сетей. Применение обновлений и исправлений к вашей промежуточной инфраструктуре должно дать вам отличное представление о том, как это будет работать в реальном мире.

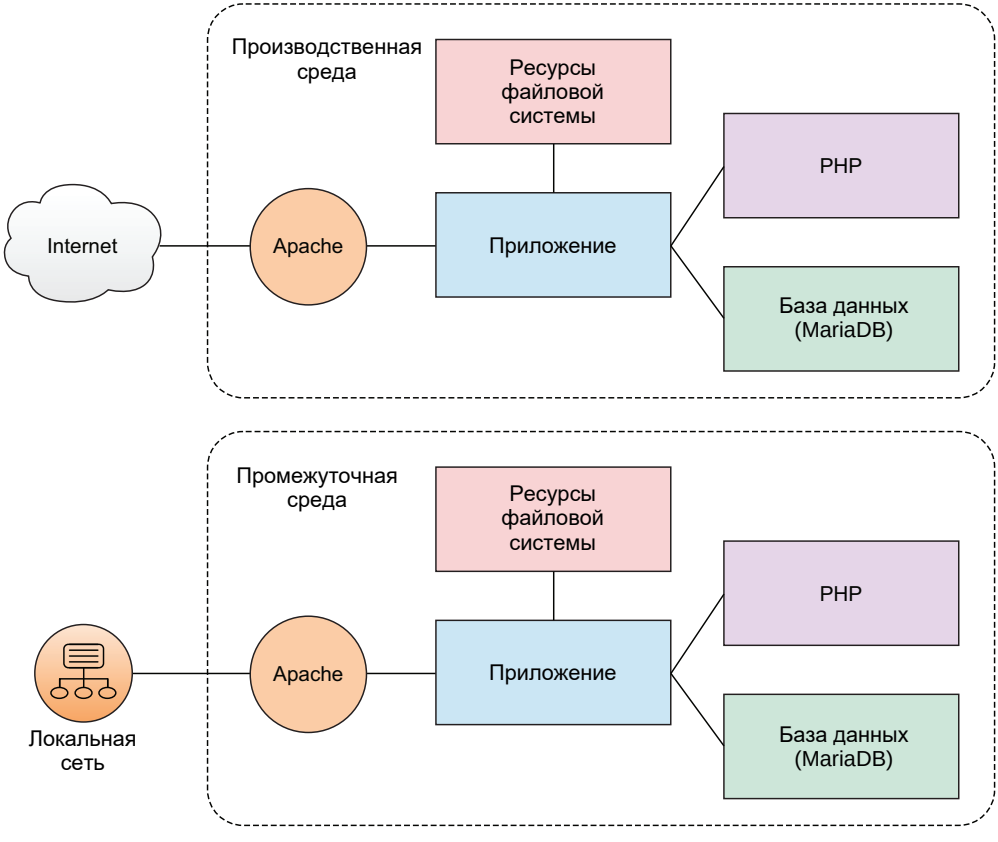


Рис. 9.1. Вы можете реплицировать инфраструктуру сервера в защищенную промежуточную среду, где ее можно безопасно обслуживать

Более того, вы можете использовать инфраструктуру в качестве программного обеспечения для управления конфигурацией кода (см. главу 16) для автоматизации всего процесса развертывания. Таким образом, как только вы подтвердите, что ис-

правленная промежуточная среда работает правильно, она может стать вашей производственной средой. Но пока отложим это.

9.2. Контролирование доступа к сети

Подумайте о подключении вашего сервера к сети и о «большом плохом» Интернете за ней как о первой линии обороны. Сетевые протоколы разработаны так, чтобы быть гибкими и помочь вам точно контролировать, какой трафик проходит. Хитрость заключается в том, чтобы понять, как работают протоколы, а затем правильно использовать эти знания для качественной настройки.

9.2.1. Настройка брандмауэра

Брандмауэр (в другой интерпретации — *файрвол*) — это набор правил. Когда пакет данных перемещается в защищенное сетевое пространство или поступает из него, его содержимое (в частности, информация о его источнике, его цели и протоколе, который он планирует использовать) проверяется на соответствие правилам брандмауэра, чтобы определить, разрешать ли его передачу. Простой пример проиллюстрирован на рис. 9.2.

Допустим, веб-сервер вашей компании должен быть открыт для входящего вебтрафика из любой точки мира, с использованием небезопасного протокола HTTP или безопасного протокола HTTPS. Поскольку ваши разработчики и администраторы должны время от времени заходить в бэкенд, чтобы сделать свою работу, вы также захотите разрешить трафик SSH, но только для тех людей, которым он понадобится. Запросы на любые другие услуги должны автоматически отклоняться. Посмотрим, как это делается.

Машина с Linux может быть настроена на применение правил брандмауэра на уровне ядра с помощью программы под названием *iptables*. Создавать правила iptables не так уж сложно — синтаксис можно понять без особых проблем. Но в интересах упрощения вашей жизни многие дистрибутивы Linux добавили свои высокоуровневые инструменты для абстрагирования такой работы. В этом разделе вы познакомитесь с программами firewalld в CentOS и UncomplicatedFirewall в Ubuntu (ufw).

Функциональность брандмауэра также предоставляют аппаратные устройства, выпускаемые такими компаниями, как Juniper и Cisco. Эти устройства частных компаний работают на собственных операционных системах с уникальным синтаксисом и дизайном. Для крупных корпоративных развертываний, в которых используются сотни серверов, распределенных по нескольким сетям, такие инструменты имеют большой смысл, но вы можете достичь той же цели на любом старом компьютере с Linux при минимальных затратах. В этом разделе я познакомлю вас только с небольшим подмножеством возможностей брандмауэров Linux. Если вы хотите узнать больше, прочитайте немного внимательнее главу 10 и обратитесь к таким «источникам мудрости» Linux, как man-файлы и онлайн-руководства.

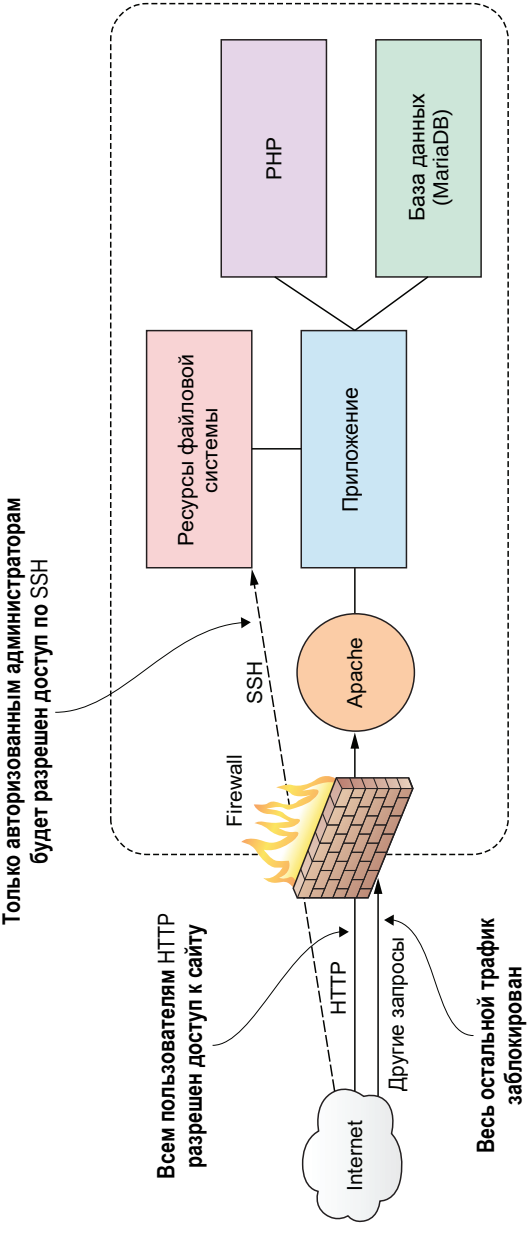


Рис. 9.2. Брандмауэр может фильтровать запросы на основе протокола или целевых правил

firewalld

Как вы уже догадались из его названия, firewalld является частью семейства systemd. Он может быть установлен на машинах с Debian/Ubuntu и он есть по умолчанию в Red Hat и CentOS. Если вы хотите использовать именно firewalld и ничто иное, то вот как его можно установить и запустить на Ubuntu:

```
# apt update
# apt install firewalld
```

Чтобы убедиться, что брандмауэр работает, попробуйте зайти в корневой каталог вашего сервера. Если сайт недоступен, то firewalld делает свою работу.

Вы будете использовать команду `firewall-cmd` для управления настройками firewalld из командной строки. Добавление аргумента `--state` позволяет получить текущее состояние брандмауэра:

```
# firewall-cmd --state
running
```

Несколько важных терминов

Чтобы быть уверенным, что вы меня понимаете, приведу несколько важных терминов. *Hypertext Transfer Protocol (HTTP)* координирует обмен данными между веб-клиентами и веб-серверами по сети. Браузер может, например, запросить веб-страницу, написанную на языке *Hypertext Markup Language (HTML)*, на что сервер может ответить, передав содержимое страницы. *Метаданные* (контекстная информация, прикрепленная к пакету) содержат информацию о состоянии сеанса, генерируются с каждым событием передачи данных и позже используются администраторами, пытающимися выяснить, что пошло не так. Версия протокола HTTPS обеспечивает надежное шифрование передачи данных с использованием технологии *Transport Layer Security (TLS)*.

Пакет — это небольшой блок данных, который мог быть вырезан из файла или архива большего размера. После передачи пакеты можно собрать в их первоначальной последовательности. Когда для передачи данных по сети используется *Transmission Control Protocol (TCP)*, пакеты, передаваемые по сети, проверяются на наличие ошибок при получении и при необходимости повторно отправляются. Передачи с использованием протокола *User Datagram Protocol (UDP)* будут выполняться быстрее, чем с помощью TCP, но поскольку они не включают исправление ошибок, то подходят только для операций, которые очень лояльны к ошибкам.

По умолчанию firewalld будет активен и станет отклонять весь входящий трафик за несколькими исключениями, такими как SSH. Это означает, что на вашем сайте будет не слишком много посетителей, что, безусловно, сэкономит вам много средств на передаче данных. Поскольку это, вероятно, не то, о чем вы мечтали для своего веб-сервера, вы захотите открыть порты HTTP и HTTPS, которые, как правило, обозначены как 80 и 443 соответственно. firewalld предлагает два способа сделать это. Один из них подразумевает добавление аргумента `--add-port`, который указывает номер порта вместе с используемым сетевым протоколом (в данном случае TCP).

Аргумент `--permanent` указывает firewalld опираться на это правило каждый раз при загрузке сервера:

```
# firewall-cmd --permanent --add-port=80/tcp
# firewall-cmd --permanent --add-port=443/tcp
```

Аргумент `--reload` применит эти правила к текущей сессии:

```
# firewall-cmd --reload
```

Такой подход будет работать для любой сложной или пользовательской конфигурации, которую вы можете придумать. Но если вы непритязательны в запросах, то можете использовать одно из предопределенных значений firewalld для многих наиболее часто используемых служб. Эти значения взяты из файла `/etc/services`.

Когда аргумент `--add-service` ссылается на ваши службы HTTP и HTTPS, он открывает порты 80 и 443. Сейчас это может показаться не таким уж важным делом, но когда наступит цейтнот, вы точно вспомните, что порт MySQL по умолчанию 3306? Не проще ли будет просто набрать `mysql`?

```
# firewall-cmd --permanent --add-service=http
# firewall-cmd --permanent --add-service=https
```

Хотите узнать текущие настройки своего брандмауэра? Используйте аргумент `--list-services`:

```
# firewall-cmd --list-services
dhcpv6-client http https ssh
```

Если вы добавили доступ для браузеров, как описано выше, то все порты HTTP, HTTPS и SSH будут открыты вместе с пакетом dhcpv6-client, который позволяет Linux запрашивать IP-адрес IPv6 с локального сервера DHCP. Вы узнаете больше об этом в главе 14.

Вы, конечно же, не хотите, чтобы кто-нибудь получил SSH-доступ к вашему серверу, поэтому настроим firewalld для его защиты. Вы ограничите доступ по SSH, чтобы разрешались только сеансы, исходящие с определенного IP-адреса. Для этого я покажу вам, как отключить весь доступ по SSH, а затем открыть его только для одного IP-адреса.

ПРИМЕЧАНИЕ

> Должен предупредить вас, что играть с брандмауэрами во время входа в сеанс SSH немного опасно. Вы можете оказаться заблокированными на вашем собственном сервере. Далее в этой главе будут описаны несколько хитростей, которые позволят вам вернуться обратно. В любом случае, если вы используете одноразовый контейнер LXC или виртуальную машину, нет причины беспокоиться об одном или другом способе: если что-то сломается, уничтожьте экземпляр и запустите новый.

Чтобы закрыть существующий доступ по SSH, добавьте аргумент `--remove-service`, а затем перезагрузите firewalld (`--remove-port` будет работать так же, если вы укажете номер порта):

```
# firewall-cmd --permanent --remove-service=ssh
success
# firewall-cmd --reload
```

Протестируйте новую конфигурацию, чтобы убедиться, что она работает. Откройте новый терминал на любом другом компьютере с доступом к сети и попробуйте войти на свой сервер с помощью SSH. Ваша попытка провалится:

```
$ ssh root@192.168.1.22
ssh: connect to host 192.168.1.22 port 22: No route to host
```

Разрешение входа в систему через SSH не лучшая идея, поэтому оно может быть запрещено в файле /etc/ssh/sshd.conf, если параметр PermitRootLogin имеет значение по

Теперь, вернувшись на свой компьютер с firewalld, добавьте новое правило, которое будет принимать TCP-трафик через порт 22 (порт SSH по умолчанию), но только от клиентов, использующих IP-адрес 192.168.1.5 (или любой другой IP-адрес вашего клиентского компьютера). Аргумент `--add-rich-rule` сообщает `firewall-cmd`, что эта команда использует набор *Rich Language* — высокоуровневый синтаксис, предназначенный для упрощения создания сложных правил брандмауэра (подробнее см. mng.bz/872B):

```
# firewall-cmd --add-rich-rule='rule family="ipv4" \
 source address="192.168.1.5" port protocol="tcp" port="22" accept'
success
```

Теперь попробуйте снова войти в систему с терминала по указанному IP-адресу. Должно работать. Поскольку вы не обозначили это правило как постоянное, все должно вернуться к обычному состоянию при следующей загрузке.

UncomlicatedFirewall (ufw)

Посмотрим, можно ли так же контролировать доступ по SSH на машине с Ubuntu, используя ufw. Программу ufw можно не устанавливать в новых инсталляциях Ubuntu, и в любом случае она будет отключена по умолчанию, поэтому нужно ее установить:

```
# apt install ufw
```

Поскольку по умолчанию ufw закрывает все порты, открыть новый сеанс SSH не удастся. Любые существующие сеансы не должны быть затронуты, но, вероятно, будет хорошей идеей добавить правило, разрешающее SSH, даже до включения ufw:

Используйте команду ufw deny ssh, чтобы отключить SSH

```
# ufw allow ssh
Rules updated
# ufw enable
Command may disrupt existing ssh connections.
Proceed with operation (y|n)?
```

Запускает брандмауэр. При необходимости используйте команду ufw disable для отключения ufw

Предупреждение о том, что это действие может повлиять на существующие или новые удаленные подключения

Если вы запускаете ufw в контейнере LXC, эти команды, возможно, не будут работать. Вместо этого отобразится сообщение об ошибке:

```
ERROR: initcaps
[Errno 2] modprobe: ERROR: ../libkmod/libkmod.c:586 kmod_search_moddep()
 could not open moddep file '/lib/modules/4.4.0-87-generic/modules.dep.bin'
modprobe: FATAL: Module ip6_tables not found in directory
  /lib/modules/4.4.0-87-generic
ip6tables v1.6.0: can't initialize ip6tables table `filter':
  Table does not exist (do you need to insmod?)
Perhaps ip6tables or your kernel needs to be upgraded.
```

Если в системе хоста отключена поддержка Ipv6, может появиться это сообщение об ошибке

Это связано с тем, что в контейнерах LXC не всегда включена поддержка IPv6 по умолчанию. Исправить это может быть сложно, учитывая, что контейнеры не имеют полного доступа к ядру своего хоста. Если вы не планируете включать IPv6 в конфигурацию сети (которая подходит в большинстве случаев), то проще всего отключить поддержку IPv6 в файле конфигурации `/etc/default/ufw`, изменив строку `IPV6 = yes` на `IPV6 = no` (листинг 9.1).

Листинг 9.1. Часть файла конфигурации /etc/default/ufw

```
# /etc/default/ufw
#

# Set to yes to apply rules to support IPv6 (no means only IPv6 on loopback
# accepted). You will need to 'disable' and then 'enable' the firewall for
# the changes to take affect.
IPV6=no
```

Измените значение IPV6 с yes на no, чтобы отключить поддержку IPv6 и избежать ошибки ufw

```
# Set the default input policy to ACCEPT, DROP, or REJECT. Please note that
# if you change this you'll most likely want to adjust your rules.
DEFAULT_INPUT_POLICY="DROP"

# Set the default output policy to ACCEPT, DROP, or REJECT. Please note that
# if you change this you'll most likely want to adjust your rules.
DEFAULT_OUTPUT_POLICY="ACCEPT"
[...]
```

Включение ufw, добавление правила для SSH и запуск команды `ufw enable` теперь должны работать:

```
# ufw enable
Command may disrupt existing ssh connections.
Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
# ufw allow ssh
Rules updated
```

Как и firewalld, ufw позволяет создавать правила с использованием номеров портов или имен служб (например, `ufw allow ssh`, как вы только что делали). Следующие две команды откроют доступ по HTTP и HTTPS для вашего веб-сервера:

```
# ufw allow 80
# ufw allow 443
```

Команда `ufw status` показывает, что служба запущена и что три требуемых правила теперь активны. Продолжайте и попробуйте выполнить это на своем веб-сервере:

```
# ufw status
Status: active
To                         Action      From
--                         ------      ----
80                         ALLOW       Anywhere
22                         ALLOW       Anywhere
443                        ALLOW       Anywhere
```

ПРИМЕЧАНИЕ

Чтобы действительно проверить доступ к веб-серверу через брандмауэр, нужно не забывать, что ваш браузер кэширует данные. Это означает, что браузер может показать страницу, на которой он побывал ранее, даже при наличии правила брандмауэра, которое должно сделать такой процесс невозможным. Чтобы убедиться, что вы тестируете действительное состояние своего сайта, очистите кэш браузера или обновите страницу.

Еще одна тонкая настройка ограничит доступ по SSH членам вашей команды по определенному IP-адресу. Если это безопасно (то есть ваш веб-сервер сейчас недоступен в Интернете), рекомендуется отключить ufw перед внесением таких изменений. Затем удалите правило allow-SSH, используя аргумент `delete 2` (который удаляет второе правило в списке ufw), и снова откройте его только для трафика, поступающего с 10.0.3.1. (В моем случае, поскольку я входил в контейнер LXC со своего хоста LXC, это будет IP-адрес, который я буду использовать; у вас может быть по-другому.) Наконец, перезапустите ufw и проверьте ее состояние теперь:

```
# ufw disable
Firewall stopped and disabled on system startup
#
# ufw delete 2
Rules updated
#
# ufw allow from 10.0.3.1 to any port 22
Rules updated
#
# ufw enable
Command may disrupt existing ssh connections.
Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
#
# ufw status
Status: active

To                         Action      From
--                         ------      ----
80                         ALLOW       Anywhere
443                        ALLOW       Anywhere
22                         ALLOW       10.0.3.1
```

Удаляет второе правило брандмауэра, отображаемое ufw status

Разрешает трафик SSH только с указанного IP и ниоткуда больше

Новое правило, разрешающее SSH трафик только с указанного IP

Вы можете проверить свою конфигурацию, войдя в систему как с компьютера с использованием разрешенного IP-адреса, так и с любого другого. Разрешенный адрес должен работать, а запрещенный — нет!

Теперь вы знаете, как применять firewalld и ufw для настройки безопасного доступа к простому веб-серверу. Хотя брандмауэры могут контролировать трафик, используя любой протокол или порт, мы рассмотрели только HTTP, HTTPS и SSH. Стоит также отметить, что вы можете использовать нестандартные сетевые порты для своих приложений.

Восстановление заблокированной виртуальной машины

Если вам удастся заблокировать себя из контейнера LXC, можете использовать команду `chroot` (как делали это в главе 6), чтобы отключить или даже перенастроить брандмауэр. Прежде всего, остановите контейнер и затем запустите `chroot` для каталога `rootfs`, который находится в структуре каталогов, используемой вашим контейнером LXC (`/var/lib/lxc/your-container-name/`). Командная строка, которую вы получите, позволит вам выполнять команды, как если бы контейнер действительно работал. Теперь отключите ufw или, если хотите, запустите необходимые команды, чтобы разрешить все, что нужно, а затем выйдите из оболочки `chroot`. Когда вы снова запустите контейнер, у вас должен быть доступ по SSH:

```
# lxc-stop -n your-container-name
# chroot /var/lib/lxc/your-container-name/rootfs/
# ufw disable
# exit
# lxc-start -d -n your-container-name
```

Останавливает работающий контейнер LXC

Монтирует файловую систему вашего контейнера как chroot

Закрывает сеанс оболочки chroot

Что, если вас заблокировала виртуальная машина VirtualBox? Все очень просто: войдите в систему через начальный терминал, который открылся, когда вы впервые запустили виртуальную машину. Это равносильно тому, чтобы сидеть за клавиатурой, которая подключена к физическому серверу и которой не требуется сеть для доступа.

9.2.2. Использование нестандартных портов

Одно из преимуществ указания сетевых портов по номеру состоит в том, что это позволяет настраивать приложения на использование нестандартных портов. Вы можете, например, установить для SSH порт 53987, а не 22. Преимущество нестандартных портов в том, что они позволяют реализовать *безопасность через затенение.*

Позвольте мне объяснить. Сам по себе порт 53987 не более безопасен, чем порт 22: его использование — это просто способ дать SSH-клиенту новые настройки. Тем не менее это может повысить уровень защиты.

Представьте, что в вашей инфраструктуре есть хакер, пытающийся найти уязвимость. Возможно, этот человек обнаружил, что один из ваших администраторов имеет дурную привычку повторно использовать один и тот же пароль для несколь-

ких учетных записей и одна из этих учетных записей уже взломана. Хакер получает много ценной информации от этого взлома: IP-адрес вашего сервера (он часто совпадает с тем, который используется вашим сайтом), имя пользователя и пароль вашего администратора. Предполагая, что вы разрешаете вход с паролем в свои учетные записи SSH (что, как вы знаете из главы 3, не очень хорошая идея), хакер может спокойно войти в систему и внести в вашу жизнь некоторый хаос. За исключением того, что никто не сказал ему, что порт 22 закрыт и доступ по SSH доступен только через какой-то непонятный непривилегированный порт (например, 53987). Поскольку вы убрали порт по умолчанию, стало немного сложнее прорвать вашу оборону, и однажды это может сыграть значительную роль.

Как это работает? Во-первых, вам нужно отредактировать конфигурационный файл `/etc/ssh/sshd_conf` на вашем сервере (компьютере, отвечающем за сеансы SSH). Файл будет содержать строку, которая по умолчанию равна `Port 22`. Отредактируйте ее, чтобы использовать тот порт, который хотите (листинг 9.2).

Листинг 9.2. Строка настройки порта из файла sshd_conf

```
# What ports, IPs, and protocols we listen for
Port 22
```

Измените это значение на номер порта, который вы хотите использовать

Когда вы закончите и будете уверены, что сможете войти на свой сервер, притом что ваш текущая сессия SSH завершится, перезапустите службу SSH. Если запущен брандмауэр, вам нужно разрешить доступ к вашему новому порту... Это скоро произойдет:

```
# systemctl restart ssh
```

Теперь, когда вы снова захотите войти с удаленного компьютера, добавьте аргумент `-p`, а затем новый номер порта. Ваш SSH-клиент сможет запросить сеанс по новому порту:

```
$ ssh -p53987 username@remote_IP_or_domain
```

Если вы входите в систему с помощью другого SSH-клиента (например, PuTTY), вам необходимо аналогичным образом сообщить клиенту о нестандартном номере порта. Рассмотрим это.

Настройка UFW FIREWALL для разрешения трафика по нестандартному порту

Открыть порт по номеру довольно просто, но вам нужно явно указать протокол, который вы будете применять (TCP или UDP). В этом примере используется протокол TCP:

```
# ufw allow 53987/tcp
```

Вы также можете открыть диапазон портов одной командой, добавив символ двоеточия (`:`). Это может быть полезно при планирования инфраструктуры, когда, скажем, вы знаете, что ваши разработчики будут создавать новые приложения и им потребуется доступ к нескольким портам. Предоставление им диапазона для

тестирования сейчас может сэкономить время. Этот конкретный пример открывает все порты между 52900 и 53000:

```
# ufw allow 52900:53000/tcp
```

Сетевые порты

Все 65 535 доступных сетевых портов делятся на три категории.

- Порты между 1 и 1023 обозначены как общеизвестные и отведены для таких служб, как SSH (22) и HTTP (80). Вы никогда не должны использовать хорошо известный номер порта для своих собственных приложений, так как это, скорее всего, спровоцирует конфликт.
- Порты между 1024 и 49151 *зарегистрированы*, а это означает, что компании и организации запросили, чтобы определенные порты в этом диапазоне были отведены для их приложений, даже если они не получили всеобщего признания. К числу таких портов относятся 1812, который используется для протокола аутентификации RADIUS, и 3306, выделенный порт MySQL.
- Порты от 49152 и до 65535 являются *незарегистрированными* и считаются динамическими (или частными). Эти порты доступны для любого временного или специального использования, особенно в частных сетях. Вы можете быть уверены, что они не будут конфликтовать с известными приложениями или сервисами.

Выбираем нестандартный номер порта

Какой номер порта выбрать? Что ж, прежде всего учтите: никогда не позволяйте посторонним (таким как я) влиять на ваше решение! Но, чтобы избежать возможных конфликтов с активными сетевыми приложениями, вы должны придерживаться значений в незарегистрированном диапазоне между 49152 и 65535. Этого должно быть достаточно для работы.

Нестандартные порты предназначены, конечно, не только для SSH. Вы должны рассмотреть такую возможность для любого приложения, которое вы написали сами либо которым можете управлять с помощью файлов конфигурации. И помните: как и большинство инструментов в данной главе, отдельно этот прием не будет настолько эффективным, как если бы использовался как часть более широкого набора протоколов безопасности.

9.3. Шифрование данных при передаче

Шифрование сайтов очень важно как минимум по двум причинам.

- Незашифрованные сайты подвергают свои данные и своих пользователей значительному риску.
- Незашифрованные сайты значительно менее привлекательны для коммерческой деятельности.

Первая проблема связана с тем, что незашифрованные сайты отображают и обрабатывают все в виде простого текста. Иными словами, все переводы с использованием паролей и личной и финансовой информации (например, кредитных карт) видны любому человеку, имеющему доступ к сети. Это явно не то, что нужно.

Вторая проблема — результат принятого в Google еще в январе 2017 года решения. Компания Google решила «оштрафовать» незашифрованные сайты, поместив их ниже в результатах поиска в Интернете. В результате пользователям стало намного труднее находить небезопасный контент.

Почему Google (наряду с другими влиятельными интернет-компаниями) должна заботиться об этом? И почему это должно волновать вас? Потому что Интернет и все, чем мы в нем пользуемся, не сможет существовать, если мы не будем доверять его контенту и методам, которыми сайты обрабатывают нашу личную информацию. Даже если ваш сайт не обрабатывает покупки по кредитным картам, тот факт, что он не зашифрован, означает, что вероятность его взлома гораздо выше, поскольку его ресурсы используются для атак зомби на другие сайты. Любой слабый сайт ослабляет весь Интернет.

Если вы хотите обезопасить свой сайт (о чем и рассказывается в этой главе), то помните, что шифрование является важной частью процесса. Не думайте, что оно *гарантирует* безопасность ваших данных — оно просто усложняет задачу для правонарушителей. Вам понадобится *сертификат*, который представляет собой файл с информацией, идентифицирующей домен, владельца, ключ и надежную цифровую подпись.

Получив сертификат, браузеры могут аутентифицировать безопасность вашего сайта и в течение сеанса обмениваться только зашифрованными данными. Все популярные современные браузеры поставляются с предварительно установленными общедоступными корневыми сертификатами, поэтому могут проверять подлинность соединений с любым сайтом по частному сертификату центра сертификации (ЦС). Вот как это работает.

1. Клиентский браузер запрашивает идентификацию сервера, чтобы они могли выполнить *рукопожатие*.
2. Сервер отвечает, отправив копию сертификата, полученного от ЦС.
3. Браузер сопоставляет сертификат со своим списком корневых сертификатов и подтверждает, что его срок действия не истек и сертификат не аннулирован.
4. Если все прошло нормально, браузер шифрует симметричный сеансовый ключ, используя открытый ключ, отправленный вашим сервером, и передает ключ на сервер.
5. Все передачи будут зашифрованы с использованием ключа сеанса.

Процесс проиллюстрирован на рис. 9.3.

До 2016 года генерация и последующая установка сертификатов шифрования из доверенных ЦС с использованием стандарта SSL/TLS отнимала время и стоила денег. В Linux вы должны были использовать инструмент командной строки OpenSSL для генерации пары ключей, а затем собрать специально отформатированный пакет

запроса на подпись сертификата (Certificate Signing Request, CSR), содержащий открытую половину пары вместе с информацией о профиле сайта.

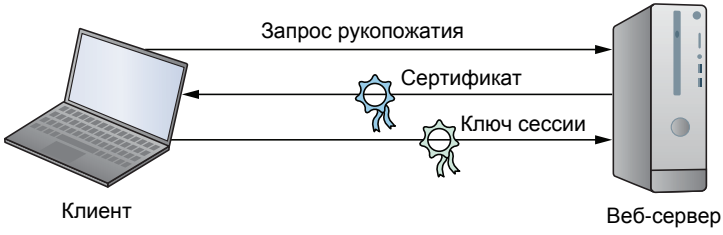


Рис. 9.3. Обмен идентификационными данными, сертификатами и сеансовыми ключами для сеанса браузера с шифрованием TLS

Затем CSR отправлялся в ЦС. Если запрос был одобрен, ЦС отправлял вам сертификат для установки в файловой системе. Вам также необходимо было обновить файлы конфигурации веб-сервера (`/etc/apache2/sites-available/default-ssl.conf` в случае Apache в Ubuntu), чтобы сообщить программе, где в вашей файловой системе хранится сертификат. Это было давно.

С 2016 года Let's Encrypt бесплатно выдает сертификаты, выступая центром сертификации. Спонсором Let's Encrypt (letsencrypt.org) является фонд Electronic Frontier вместе с большим количеством корпоративных партнеров, в том числе Cisco, Chrome, Shopify и Digital Ocean. Его задача — продвигать шифрование сайта, делая его дешевле и, что не менее важно, проще.

Забудьте о настройке файлов и использовании OpenSSL для генерации CSR: пусть клиент Encrypt Certbot ACME сделает почти все за вас. Сертификаты Let's Encrypt действительны в течение 90 дней и могут быть настроены для автоматического продления.

9.3.1. Подготовка домена вашего сайта

Прежде чем вы сможете установить сертификат для шифрования домена вашего сайта, придется получить домен. Это потребует покупки имени у регистратора домена, такого как GoDaddy или Amazon Route 53. Подробнее о том, как это работает, вы можете прочитать в главе 5 моей книги *Learn Amazon Web Services in a Month of Lunches* (Manning, 2017).

Поскольку вы хотите, чтобы Apache обрабатывал специфичные для домена запросы от внешних клиентов, вам также необходимо добавить раздел в файл `/etc/apache2/sites-available/000-default.conf` (на компьютере с CentOS эти настройки находятся в файле `/etc/httpd/conf/httpd.conf`, который вы будете редактировать). Вот как это может выглядеть на моем сервере bootstrap-it.com. Обратите внимание, что на данный момент он настроен только на прием трафика через небезопасный порт HTTP 80 (листинг 9.3).

Листинг 9.3. Пример раздела домена в файле конфигурации Apache

```
<VirtualHost *:80>
    ServerName bootstrap-it.com
    DocumentRoot /var/www/html
    ServerAlias www.bootstrap-it.com
</VirtualHost>
```

В этой конфигурации прослушивается только порт 80

Ваше доменное имя используется в качестве значения для ServerName

Эта строка ServerAlias добавляет www в качестве префикса к домену

9.3.2. Генерация сертификатов с использованием Let's Encrypt

С этого момента все довольно просто. Перейдите на страницу Getting Started на сайте Certbot Electronic Frontier Foundation (certbot.eff.org) и укажите программное обеспечение веб-сервера и используемую вами ОС (рис. 9.4).

С домашней страницы Certbot вы будете перенаправлены на страницу с краткими инструкциями. Для Apache в Ubuntu 16.04, который включает команды для установки менеджера общих репозиториев ПО, добавьте репозиторий Certbot в свой список APT, а затем установите программное обеспечение Certbot на основе Python для Apache:

```
# apt update
# apt install software-properties-common
# add-apt-repository ppa:certbot/certbot
# apt update
# apt install python-certbot-apache
```

Рис. 9.4. После выбора программного обеспечения и ОС веб-сервера на домашней странице Certbot вам будут предоставлены инструкции по установке

Наконец, вы запустите программу Certbot с правами администратора (в моем случае в качестве аргумента используется `--apache`). Certbot прочитает файлы конфигурации вашего веб-сервера, чтобы определить домены, которые вы, вероятно, захотите зарегистрировать:

```
# certbot --apache
```

После ответа на несколько вопросов о контактной информации и условиях предоставления услуг Let's Encrypt вам будет представлен список возможных доменных имен, который может выглядеть следующим образом:

```
Which names would you like to activate HTTPS for?
-------------------------
1: bootstrap-it.com
2: www.bootstrap-it.com
-------------------------
Select the appropriate numbers separated by commas and/or spaces,
  or leave input blank to select all options shown (Enter 'c' to cancel):
```

После того как вы ответите, бот попытается подтвердить, что выбранные вами домены существуют и зарегистрированы на общедоступном DNS-сервере. Сервер сертификатов наконец попробует подключиться к вашему сайту. В случае успеха сертификат Let's Encrypt будет установлен автоматически, а все необходимые дополнительные разделы будут добавлены в ваши файлы конфигурации.

Если что-то пойдет не так, Certbot выведет содержательные сообщения об ошибках, которые вы сможете использовать для поиска решения. Кроме того, в Let's Encrypt поддерживается активный форум для помощи сообществу, куда пользователи всех уровней могут смело обращаться за советами: community.letsencrypt.org.

На текущий момент вы узнали, как повысить безопасность сайта, постоянно обновляя свои приложения, используя правила брандмауэра для контроля доступа к своей сети, повышая безопасность и шифруя данные при их передаче между сайтом и его посетителями. Но мы еще не закончили с безопасностью.

Вот что еще впереди: ужесточение протоколов входа в систему, использование модуля ядра SELinux и групп для более тщательного управления проблемами, с которыми могут столкнуться пользователи, и отслеживание запущенных процессов.

9.4. Усиление процесса аутентификации

Применение некоторых решений для безопасного подключения, особенно SSH — это прекрасно. Но также неплохо было бы обратить внимание на то, как члены вашей команды используют SSH. Вот несколько советов по улучшению безопасности удаленного доступа. Возможно, нецелесообразно применять их в каждой среде (особенно в процессе настройки), но они по крайней мере должны быть вам понятны.

Избегайте входа на серверы от имени пользователя root. Всегда лучше использовать команду sudo, когда необходимы права администратора. Фактически вы можете полностью запретить входы root по SSH, отредактировав строку PermitRootLogin в файле /etc/ssh/sshd_conf:

```
PermitRootLogin no
```

Строка root-login-control в /etc/ssh/sshd_config

Вы также можете рекомендовать своим администраторам использовать только беспарольный доступ по SSH с помощью пары ключей (как было показано в главе 3). Это также можно настроить в файле sshd_conf, на этот раз в строке PasswordAuthentication. Без аутентификации по паролю пользователи будут вынуждены применять пары ключей:

```
PasswordAuthentication no
```

Строка управления проверкой пароля в /etc/ssh/sshd_config

После каждого из этих изменений обязательно перезагрузите SSH; в противном случае новые настройки не вступят в силу до следующего включения:

```
# systemctl restart sshd
```

Это в любом случае важные шаги. Но если для развертывания требуется некоторая изоляция на промышленном уровне, рассмотрите возможность включения SELinux.

9.4.1. Контроль за объектами файловой системы с помощью SELinux

Помните, как мы обсуждали права доступа к объектам еще в главе 4? На тот момент нужно было убедиться, что пользователи могут получить доступ к своим файлам и редактировать их. Но оборотная сторона этой монеты в том, что «неправильные» пользователи не смогут получить доступ к файлам других людей.

Как уже говорилось, общий профиль разрешений для объекта может предоставить владельцу полные полномочия на чтение-запись-выполнение, но дает группе объекта и другим только право на чтение. Это будет 744 в нашей числовой записи или rwx r-- r-- в другом случае.

Предоставление вашим пользователям полной власти над их собственными ресурсами иногда характеризуется таким понятием, как *дискреционный контроль доступа (DAC)*. DAC имеет смысл устанавливать, если вы хотите повысить эффективность работы ваших пользователей, но он имеет свою цену: полный контроль влечет за собой риск того, что люди используют его, не зная обо всех последствиях.

Приведу практический пример того, что я имею в виду. Предположим, есть пара разработчиков, которые усердно трудятся над решением проблемы в вашей

компании: локальное тестирование их программного обеспечения всегда заканчивается неудачей при попытке записи данных в файл. Отладка показывает, что проблема в разрешениях и она вызвана тем, что приложение запускает один пользователь, а файл с данными принадлежит другому.

Поскольку это происходило несколько раз и с несколькими файлами данных (или файлами ключа SSH), разработчики выбирают быстрый и легкий путь: они предоставляют разрешения в отношении файлов данных и всех файлов в этих каталогах — 777 — полный доступ для всех. Теперь это серьезная проблема безопасности. Кроме того, есть отличный шанс, что приложение, над которым они работают, в конечном итоге будет перенесено в производственную среду с теми же настройками системы. Именно такая ошибка лежит в основе многих главных нарушений хранения данных, о которых вы время от времени слышите.

SELinux относится еще к одной из тех сложных тем, которая, хотя и является критически важной для многих приложений Linux, не должна играть главную роль в этой книге. Еще раз: не стесняйтесь перейти прямо к обсуждению в системных группах, если требуется.

После установки и активации модуль ядра SELinux применяет *обязательное управление доступом* (mandatory access control, MAC) к объектам файловой системы независимо от владельца конкретного объекта. По сути, как показано на рис. 9.5, он накладывает тщательно определенные общесистемные ограничения на то, что может делать пользователь, делая невозможными опасные по своей природе конфигурации.

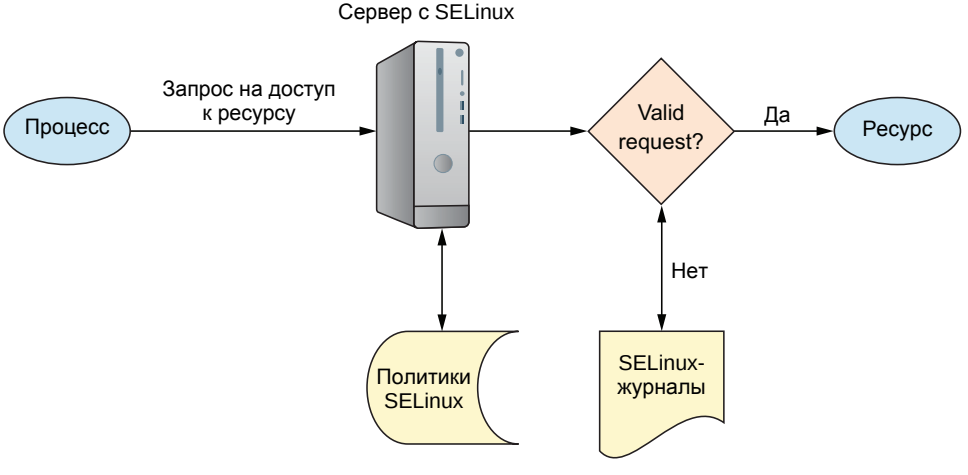


Рис. 9.5. Поток, показывающий, что следует за запросом на доступ к ресурсу через фильтр политик SELinux

Если бы SELinux был активен, два наших разработчика могли бы применять разрешения 777 к своим файлам данных снова и снова, но это бы не помогло. Вместо

этого им пришлось бы искать более подходящее и эффективное решение. Они могли бы, например, рассмотреть вопрос о создании системной группы с правами на данные, а затем добавить в группу соответствующие учетные записи пользователей. Об этом вы узнаете чуть позже в данной главе. Звучит здорово. Но что не так с безопасностью?

Ну, есть кое-какие проблемы. У SELinux непонятные отношения с совместимостью приложений — он не может заставить их работать. Проблема заключается в том, что все приложения создаются как в локальном, так и в пользовательском пространстве SELinux, что часто приводит к сбою.

У меня есть все основания полагать, что эти решения невозможно внедрить без лучшего понимания принципов проектирования файловых систем и обеспечения безопасности. В частности, вы должны помнить о *принципах предоставления привилегий*, которые должны предоставить всем пользователям. В любом случае вам придется разобраться в SELinux, поэтому следующие подразделы посвящены именно ему.

9.4.2. Установка и активация SELinux

SELinux был разработан для Red Hat Linux (и CentOS), и, вероятно, поэтому в данных системах он устанавливается и активируется по умолчанию. Запуск его на других дистрибутивах, включая Ubuntu, определенно возможен (хотя AppArmor является более распространенным выбором для Ubuntu), но никто не гарантирует, что установка всегда будет проходить гладко. (Даже не думайте попробовать его в контейнере LXC; вместо этого используйте для тестирования VirtualBox.). В Ubuntu вам понадобятся три пакета: selinux, setools и policycoreutils. Вот как это будет выглядеть:

```
# apt install setools policycoreutils selinux
```

После того как пакеты будут правильно настроены, перезагрузите Ubuntu и запустите команду `sestatus` для получения снимка текущего состояния SELinux, а также знакомства с важными положениями файловой системы и политикой. Если повезет (!), вы должны увидеть что-то вроде этого:

```
# sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   permissive
Mode from config file:          permissive
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Max kernel policy version:      30
```

Текущий статус SELinux — включен

Используется политика по умолчанию

В ряде случаев вам может понадобиться выполнить команду `selinux-activate` для включения настроек SELinux в процесс загрузки:

```
# selinux-activate
Activating SE Linux
Generating grub configuration file ...
Warning: Setting GRUB_TIMEOUT to a non-zero value when GRUB_HIDDEN_TIMEOUT
   is set is no longer supported.
Found linux image: /boot/vmlinuz-4.4.0-89-generic
Found initrd image: /boot/initrd.img-4.4.0-89-generic
Found linux image: /boot/vmlinuz-4.4.0-87-generic
Found initrd image: /boot/initrd.img-4.4.0-87-generic
Found linux image: /boot/vmlinuz-4.4.0-83-generic
Found initrd image: /boot/initrd.img-4.4.0-83-generic
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
done
SE Linux is activated. You may need to reboot now.
```

В соответствии с названием, selinux-activate делает SELinux активным при следующей загрузке

Флаги для SELinux добавляются в команду загрузки для каждого образа, управляемого GRUB

Поскольку SELinux зависит от настроек уровня ядра, изменения часто требуют перезагрузки

Вам может потребоваться перезагрузка, чтобы изменения вступили в силу.

Вы можете контролировать поведение SELinux через конфигурационный файл в `/etc/selinux/`. Файл содержит две настройки: состояние и тип SELinux. Таблица 9.1 дает краткий обзор возможных значений.

Таблица 9.1. Настройки конфигурации для SELinux в /etc/selinux/config

Категория	Значение	Описание	Использование
Состояние	disabled	SELinux не работает	
	enforcing	Применяется политика безопасности	
	permissive	Нарушения политики приводят только к предупреждению в журнале	Полезно для тестирования конфигурации
Тип политики	targeted	Включает домен, процессы которого не определены ограничениями SELinux	Полезно для смешанных систем, где не все процессы требуют ограничений
	minimum	SELinux ограничивает только минимальные процессы	Более тонкая настройка для экспериментальных систем
	mls	Политики применяются на основе уровня чувствительности и существующих возможностей	

Помимо файла конфигурации, вы также можете установить состояние SELinux из командной строки, используя `setenforce`, где `setenforce 1` задает состояние `enforcing`, а `setenforce 0` устанавливает SELinux в разрешающее состояние. В раз-

решающем состоянии нарушения правил допускаются, но регистрируются. Это хороший способ устранять неполадки или тестировать конфигурацию:

```
# setenforce 1
```

Как насчет примера SELinux для иллюстрации того, как вы можете контролировать доступ к отдельному файлу? Считайте, что это сделано. Вы должны обязательно попробовать выполнить пример из следующего подраздела (или что-то подобное) самостоятельно.

9.4.3. Применение политик SELinux

Допустим, вы системный администратор, ответственный за тех двух ленивых разработчиков, о которых мы говорили ранее. Исходя из своего опыта, вы подозреваете, что у них может возникнуть соблазн открыть доступ к файлу данных слишком многим посторонним. Вы можете защитить свои данные независимо от того, что делают разработчики.

Можно использовать SELinux для управления доступом к любому файлу или процессу, но для простоты поработаем на машине с установленным Apache (или httpd) и файлом `index.html` в корневом каталоге документа по адресу `/var/www/html/`. По умолчанию файл будет доступен по крайней мере для локальных запросов (через `wget localhost` из командной строки сервера). Вот как это обычно выглядит:

```
$ wget locahost
--2017-08-02 10:24:25-- http://localhost/
Resolving localhost (localhost)... ::1, 127.0.0.1
Connecting to localhost (localhost)|::1|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 11 [text/html]
Saving to: 'index.html'

100%[======================================>] 11 --.-K/s in 0s
```

wget успешно сохранила файл index.html в локальном каталоге

Теперь проверьте разрешения на файл `index.html` с помощью команды `ls -Z` (`-Z` даст сведения о контексте безопасности файла):

```
# ls -Z /var/www/html/
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0 index.html
```

Во-первых, обратите внимание на обычные права доступа (`-rw-r--r--`), которые делают файл читаемым (`r`) для любого пользователя. Это стандартно для ресурсов сайта. Состояние SELinux для файла отображается как `unconfined_u:object_r:httpd_sys_content_t:s0`. Вы можете использовать команду `chcon -t`, чтобы изменить тип контекста файла. Она заменяет тип Apache `httpd_sys_content_t` на связанный с Samba тип `samba_share_t`. Я не уверен, что вам когда-нибудь придется делать это в реальной жизни, но пример должен наглядно продемонстрировать, как можно

уравновесить полномочия, которые вы предоставляете своим пользователям, с их способностью испортить ситуацию:

```
# chcon -t samba_share_t /var/www/html/index.html
```

Повторное выполнение `ls -Z` показывает, что файл теперь связан с типом `samba_share_t`:

```
# ls -Z /var/www/html/
-rw-r--r--. root root unconfined_u:object_r:samba_share_t:s0
➥ /var/www/html/index.html
```

Как теперь `wget localhost` будет обрабатывать новый контекст SELinux?

```
$ wget localhost
--2017-08-02 10:27:30-- http://localhost/
Resolving localhost (localhost)... ::1, 127.0.0.1
Connecting to localhost (localhost)|::1|:80... connected.
HTTP request sent, awaiting response... 403 Forbidden
2017-08-02 10:27:30 ERROR 403: Forbidden.
```

Apache отвечает на запрос сообщением о неудаче из-за ошибки 403: Forbidden (Запрещено)

Доступ запрещен, и команда не будет выполнена. Apache вынужден разочаровать вас (или, скорее, разработчиков), поскольку не имеет власти над файлом в его текущем контексте. Это верно, несмотря на то что атрибуты файла включают разрешения на чтение для всех пользователей. Независимо от того, насколько отчаянно ваши разработчики захотят открыть доступ к защищенному файлу, они ничего не смогут сделать.

9.4.4. Системные группы и принцип наименьших привилегий

Два наших разработчика наконец получили сообщение. Они понимают, что им запретили предоставлять слишком широкий доступ. Но теперь они просят вас помочь им решить исходную проблему: как сделать файлы с конфиденциальными данными доступными для нескольких учетных записей, не открывая их для всех.

Ответ: использовать группы. *Группа* — это системный объект, почти такой же, как пользователь, за исключением того, что никто не может войти в систему в качестве группы. Мощь групп в том, что их, как и пользователей, можно назначать файлам или каталогам, что позволяет любому члену группы разделять ее полномочия. Это показано на рис. 9.6.

Попробуйте сами: используйте команду `nano` для создания нового файла. Добавьте в него текст `Hello World`, чтобы вы могли легко определить, когда к нему есть доступ. Теперь отредактируйте права доступа к файлу с помощью `chmod 770`, чтобы его владелец и группа имели полные права на файл, но все остальные не могли его прочитать:

```
$ nano datafile.txt
$ chmod 770 datafile.txt
```

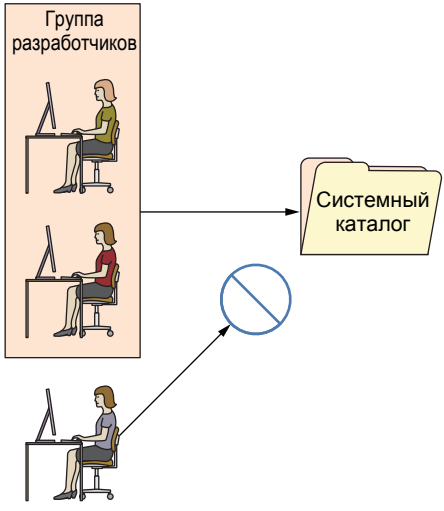


Рис. 9.6. Разработчикам, которые являются членами группы, может быть предоставлен доступ к определенному каталогу, в отличие от всех остальных, кто не входит в эту группу

Если в вашей системе еще нет других учетных записей, кроме вашей, создайте их, используя команду `adduser` (способ Debian/Ubuntu) или `useradd` (если вы работаете в CentOS). Обратите внимание, что `useradd` также будет работать в Ubuntu:

```
# useradd otheruser
# passwd otheruser
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Команда useradd (в отличие от команды Debian adduser) требует, чтобы вы задали пароль пользователя отдельно

Введите команду `su`, чтобы войти в учетную запись нового пользователя. Как только вы укажете пароль пользователя, все ваши команды будут выполняться от имени этого пользователя. Вы будете работать только с его полномочиями: не больше и не меньше. Если вы попытаетесь прочитать файл `datafile.txt` (используя команду `cat`), вам не удастся этого сделать, потому что, как вы помните, остальным пользователям было отказано в разрешении на чтение. Когда вы закончите, введите команду `exit`, чтобы выйти из пользовательской оболочки нового пользователя и вернуться к исходной оболочке:

```
$ su otheruser
Password:
$ cat /home/ubuntu/datafile.txt
cat: /home/ubuntu/datafile.txt: Permission denied
$ exit
```

Все это ожидаемо. И, как вы видели, неспособность прочитать файл, принадлежащий другому читателю, иногда может быть проблемой. Посмотрим, что можно сделать, если связать файл с группой, а затем правильно настроить права доступа к файлу.

Создайте новую группу, которую можете использовать для управления данными вашего приложения, а затем отредактируйте свойства вашего файла данных с помощью команды `chown`. Аргумент `ubuntu:app-data-group` оставляет владельца файла в руках пользователя `ubuntu`, но меняет группу владения на новую: `app-data-group`:

```
# groupadd app-data-group
# chown ubuntu:app-data-group datafile.txt
```

Запустите команду `ls` с длинным выводом (`-l`) для файла, чтобы просмотреть его новые разрешения и статус. Обратите внимание, что, как и ожидалось, `ubuntu` является владельцем файла, а `app-data-group` — его группой:

```
$ ls -l | grep datafile.txt
-rwxrwx--- 1 ubuntu app-data-group     6 Aug 9 22:43 datafile.txt
```

Вы можете использовать команду `usermod`, чтобы добавить своего пользователя в группу `app-data-group`, а затем опять `su`, чтобы переключиться в оболочку другого пользователя. На этот раз, несмотря на то что права доступа к файлу блокируют других и вы определенно действуете как «другой» прямо сейчас, вы сможете прочитать его... благодаря членству в группе:

```
# usermod -aG app-data-group otheruser
$ su otheruser
$ cat datafile.txt
Hello World
```

Используйте команду su для переключения между учетными записями пользователей

Это содержимое моего файла datafile.txt

Такая организация позволяет правильно и эффективно решать многие сложные проблемы с разрешениями, которые могут возникнуть в многопользовательской системе. Фактически этот подход применяется не только для предоставления индивидуальным пользователям необходимого им доступа: многие системные процессы не могут выполнять свою работу без специального членства в группах. Просмотрите файл `/etc/group` и обратите внимание, сколько системных процессов имеют свои собственные группы (листинг 9.4).

Листинг 9.4. Фрагмент содержимого файла /etc/group

```
$ cat /etc/group
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:syslog
tty:x:5:
disk:x:6:
lp:x:7:
mail:x:8:
news:x:9:
uucp:x:10:
man:x:12:
proxy:x:13:
[...]
```

Я закончу главу описанием нескольких кратких, но жизненно важных операций, которые вы можете включить в свои методы обеспечения безопасности.

9.4.5. Изоляция процессов в контейнерах

Обеспокоены тем, что несколько сервисов у вас запущено на одном сервере, и не знаете, будут ли они все подвержены риску при взломе одного сервиса? Один из способов ограничить ущерб, который могут нанести неосторожные пользователи или злоумышленники, — изолировать системные ресурсы и процессы. Таким образом, даже если кто-то захочет расширить свои возможности за установленные пределы, у него не будет физического доступа.

Старый подход к проблеме заключался в предоставлении отдельной физической машины для каждого сервиса. Но виртуализация может значительно упростить и сделать доступным создание *изолированной* инфраструктуры. Такую архитектуру часто соотносят с *микросервисами*. С ней вы можете запускать несколько контейнеров, один из которых, возможно, работает только с базой данных, другой — с Apache, а третий — с медиафайлами, которые могут быть встроены в ваши веб-страницы. В дополнение к многочисленным преимуществам производительности и эффективности, связанным с архитектурами микросервисов, их применение может значительно снизить риск взлома для каждого отдельного компонента.

ПРИМЕЧАНИЕ

Под контейнерами я не обязательно подразумеваю контейнеры LXC. В наши дни для такого рода развертывания гораздо удобнее применять контейнеры Docker. Если вы заинтересованы в получении дополнительной информации, посмотрите книги Microservices in Action (Морган Брюс и Пауло А. Перейра, Manning, 2018), Microservice Patterns[1] (Крис Ричардсон, Manning, 2018) или Docker in Practice, 2-е изд. (Ян Миелл и Эйдан Хобсон Сайерс, Manning, 2018).

9.4.6. Сканирование на наличие опасных идентификаторов пользователей

В то время как любой пользователь с правами администратора может временно получить полномочия root с помощью sudo, на самом деле только у настоящего root будут все его права. Как вы уже видели, выполнять обычные функции от имени root небезопасно. Но это может произойти, например, в результате несчастного случая или злонамеренного вмешательства, и обычный пользователь сможет получить полные права администратора.

Хорошая новость заключается в том, что самозванцев легко обнаружить: их идентификационные номера пользователей и/или групп будут, как и у root, равны нулю (0). Посмотрите на файл `passwd` в `/etc/`. Он содержит запись для каждой

[1] *Ричардсон К.* Микросервисы. Паттерны разработки и рефакторинга. — СПб.: Питер, 2019.

учетной записи обычного и системного пользователя, которые существуют в настоящее время. Первое поле хранит имя учетной записи (в данном случае root и ubuntu), а второе может содержать x вместо пароля (который, если он существует, будет отображаться в зашифрованном виде в файле /etc/shadow). Но следующие два поля содержат идентификаторы пользователя и группы. В случае ubuntu в этом примере оба идентификатора равны 1000. И, как вы можете видеть, у root они равны нулям:

```
$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
[...]
ubuntu:x:1000:1000::/home/ubuntu:/bin/bash
```

Если вы когда-нибудь увидите обычного пользователя с идентификатором пользователя или группы 0, то знайте, что происходит что-то нехорошее и вам следует насторожиться. Быстрый и простой способ обнаружить проблему — запустить команду awk для файла passwd. Она напечатает любую строку, третье поле которой содержит значение 0. В моем случае, к моему большому облегчению, единственным результатом был root. Вы можете запустить команду еще раз, подставив $4 вместо $3, чтобы выбрать поле идентификатора группы:

```
$ awk -F: '($3 == "0") {print}' /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

Команда awk более подробно обсуждается в главе 11

9.5. Аудит системных ресурсов

Чем больше приложений у вас запущено, тем больше шансов, что что-то сломается, поэтому имеет смысл отслеживать, что именно работает. Это относится к сетевым портам (если они открыты, то по определению должен быть доступ к компьютеру), сервисам (если они активны, то люди могут их запускать) и установленному программному обеспечению (если оно установлено, то может быть запущено).

Чтобы аудиты приносили пользу, их нужно время от времени проводить. Поскольку вы наверняка снова забудете о них, гораздо проще включить свои инструменты аудита в сценарий, который не только регулярно выполняется, но и в идеале интерпретирует результаты, чтобы сделать их более читабельными. В этом разделе я расскажу вам о трех ключевых инструментах аудита, которые помогут вам сканировать открытые порты, активные сервисы и ненужные пакеты программного обеспечения.

9.5.1. Сканирование на наличие открытых портов

Порт считается *открытым*, если на хосте запущен какой-то процесс, который прослушивает на этом порте запросы. Следя за открытыми портами, вы можете быть в курсе того, что происходит с вашим сервером.

Вы уже знаете, что на обычном веб-сервере, вероятно, будут открыты порты HTTP (80) и SSH (22), и это не должно вас удивлять. Таким образом, лучше сосредоточиться на других, неожиданных результатах. Команда netstat перечисляет открытые порты вместе с подробной информацией о том, как они используются.

В этом примере на довольно типичном многоцелевом сервере `-n` указывает `netstat` выводить порты и адреса в числовом формате, `-l` включает только прослушивающие сокеты, а `-p` добавляет к выводу идентификатор процесса прослушивающей программы:

```
# netstat -npl                                                  Процесс MySQL работает на порте 3306
Active Internet connections (only servers)
Proto Local Address      Foreign Address   State       PID/Program name
tcp   127.0.0.1:3306     0.0.0.0:*         LISTEN      403/mysqld
tcp   0.0.0.0:139        0.0.0.0:*         LISTEN      270/smbd
tcp   0.0.0.0:22         0.0.0.0:*         LISTEN      333/sshd
tcp   0.0.0.0:445        0.0.0.0:*         LISTEN      270/smbd
tcp6  :::80              :::*              LISTEN      417/apache2   Идентификатор
[...]                                                                процесса SSH — 333
```

В последние годы вместо команды `netstat` для многих целей используется `ss`. Если вы однажды окажетесь на вечеринке и кто-то спросит вас о `ss`, вспомните этот пример (в котором перечислены все установленные SSH-соединения) — он должен дать вам достаточно информации, чтобы вы смогли ответить:

```
$ ss -o state established                          Отображает все сокеты TCP
   '( dport = :ssh or sport = :ssh )'
Netid  Recv-Q Send-Q    Local Address:Port      Peer Address:Port
tcp    0      0         10.0.3.1:39874          10.0.3.96:ssh
timer:(keepalive,18min,0)
```

9.5.2. Сканирование на предмет активных служб

Получение мгновенного снимка служб, управляемых systemd и в настоящее время включенных на вашем компьютере, также может помочь вам обнаружить действия, которых вы не выполняли. Команда `systemctl` выведет список всех существующих служб, который затем можно сузить до тех результатов, чьи описания содержат `enabled`. Этот код возвращает только активные службы:

```
# systemctl list-unit-files --type=service --state=enabled
autovt@.service                              enabled
bind9.service                                enabled
cron.service                                 enabled
dbus-org.freedesktop.thermald.service        enabled
docker.service                               enabled
getty@.service                               enabled
haveged.service                              enabled
mysql.service                                enabled
networking.service                           enabled
resolvconf.service                           enabled
rsyslog.service                              enabled
ssh.service                                  enabled
sshd.service                                 enabled
syslog.service                               enabled
systemd-timesyncd.service                    enabled
thermald.service                             enabled
unattended-upgrades.service                  enabled
ureadahead.service                           enabled
```

sshd — сервер SSH;
ssh — это клиентское программное обеспечение

Если вы найдете что-то, чего не должно быть, можете использовать команду `systemctl`, чтобы остановить службу и убедиться, что она не запустится снова при следующей загрузке:

```
# systemctl stop haveged
# systemctl disable haveged
```

На самом деле в службе, которую я остановил в этом примере, нет ничего страшного. Это небольшой инструмент, который я часто устанавливаю, чтобы генерировать случайную фоновую активность системы, когда я создаю ключи шифрования.

9.5.3. Поиск установленного программного обеспечения

Может ли кто-то или что-то установить программное обеспечение в вашей системе без вашего ведома? Как вы узнаете, если не посмотрите? Чтобы получить весь список, используйте команду `yum list installed` или, в Debian/Ubuntu, `dpkg --list`. Чтобы удалить все пакеты, которых не должно быть в списке, введите `remove` *<имя_пакета>*:

```
# yum list installed
# yum remove packageName
```

Вот как это происходит в Ubuntu:

```
# dpkg --list
# apt-get remove packageName
```

Выводит длинный список пакетов, который вам придется как можно быстрее визуально просмотреть. Я не знаю легких путей

Полезно также быть в курсе изменений в конфигурационных файлах вашей системы. Об этом вы узнаете в главе 11.

Резюме

- Используя брандмауэры, вы контролируете сетевой трафик по протоколу, порту и источнику или назначению.
- Сконфигурируйте приложения для прослушивания на нестандартных сетевых портах, чтобы повысить безопасность, используя для инфраструктуры принцип *«безопасность через неясность»* (security through obscurity).
- Используя сертификаты, полученные от ЦС, зашифруйте сеансы браузера на хосте клиента — это значительно снизит вероятность получения доступа к передаваемым данным.
- Глобальный контроль в многопользовательской файловой системе осуществляется с помощью SELinux.
- Выборочный доступ пользователей и процессов к ресурсам возможен при использовании групп.

- Регулярные (по сценарию) проверки запущенных процессов, установленного программного обеспечения и открытых портов имеют решающее значение для обеспечения безопасности сервера.

Ключевые термины

- Вы можете администрировать правила брандмауэра в Linux, используя *iptables* или более простые инструменты.
- *Протокол передачи гипертекста (HTTP)* управляет передачей данных для браузера через сеть.
- *Протокол безопасности транспортного уровня (TLS)* обеспечивает принудительное шифрование данных для их передачи по сети.
- *Дискреционные системы контроля доступа (DAC)* позволяют контролировать расход пользователями ресурсов файловой системы.
- Контроль над ресурсами в *системах обязательного контроля доступа (MAC)* в конечном итоге управляется общесистемными политиками.
- *Микросервисы* — небольшие компьютерные сервисы, запускаемые из отдельных контейнеров, как часть более крупной инфраструктуры приложений, охватывающей несколько контейнеров.

Обзор команд

- `firewall-cmd --permanent --add-port=80/tcp` открывает порт 80 для входящего трафика HTTP и сохраняет эту конфигурацию для использования при загрузке.
- `firewall-cmd --list-services` выводит список активных в настоящий момент правил системы firewalld.
- `ufw allow ssh` открывает порт 22 для SSH-трафика, используя UncomplicatedFirewall (ufw) в Ubuntu.
- `ufw delete 2` удаляет второе правило ufw, что можно проверить командой `ufw status`.
- `ssh -p53987 username@remote_IP_or_domain` открывает сессию SSH по нестандартному порту.
- `certbot --apache` конфигурирует веб-сервер Apache для использования сертификатов шифрования Let's Encrypt.
- `selinux-activate` активирует SELinux на машине Ubuntu.
- `setenforce 1` устанавливает принудительный режим в конфигурации SELinux.
- `ls -Z /var/www/html/` отображает контекст безопасности файлов в указанном каталоге.
- `usermod -aG app-data-group otheruser` добавляет пользователя `otheruser` в системную группу `app-data-group`.
- `netstat -npl` сканирует открытые (прослушиваемые) сетевые порты на сервере.

Самотестирование

1. Вы переживаете, что хакеры могли получить доступ к вашему серверу, и хотите убедиться, что они не могут повысить свои права доступа до полномочий root. Какая из следующих команд может помочь:
   а) `firewall-cmd --list-services`;
   б) `netstat -npl`;
   в) `certbot --apache`;
   г) `awk -F: '($3 == "0") {print}' /etc/passwd`?
2. Вы заметили, что на вашем сервере открыты сетевые порты, появление которых вы не можете объяснить. Какой из следующих инструментов можно использовать для их закрытия:
   а) `firewalld`;
   б) `netstat`;
   в) `certbot --apache`;
   г) `awk`?
3. Какие преимущества в безопасности можно получить при разделении служб одного приложения между несколькими контейнерами:
   а) отказ одной из них не обязательно повлияет на работу других;
   б) уязвимость в одной службе не обязательно распространится на другие;
   в) такая конструкция сделает аутентификацию менее связанной с серверами;
   г) такая конструкция повысит видимость процесса?
4. Какая из следующих команд разрешит SSH доступ к серверу только с одного IP-адреса:
   а) `firewall-cmd allow from 10.0.3.1 to any port 22`;
   б) `ufw allow from 10.0.3.1 to port 22`;
   в) `ufw allow from 10.0.3.1 to any port 22`;
   г) `firewall-cmd --allow from 10.0.3.1 to any port 22`?
5. Запрос сертификата TLS в ЦС позволяет вам:
   а) запретить неавторизованным пользователям доступ к серверной части вашего веб-сервера;
   б) обезопасить данные при хранении на веб-сервере;
   в) защитить данные при передаче между веб-сервером и клиентами;
   г) разрешить SSH-доступ без пароля к серверной части вашего веб-сервера.
6. Какие из следующих параметров в файле `/etc/ssh/sshd_config` заставят клиентов SSH использовать пары ключей:
   а) `PermitRootLogin no`;
   б) `PermitRootLogin yes`;

в) `#PasswordAuthentication no`;

г) `PasswordAuthentication no`?

7. Какая из следующих команд переведет SELinux в разрешающий режим:

   а) `setenforce 0`;

   б) `chcon -t samba_share_t /var/www/html/index.html`;

   в) `setenforce 1`;

   г) `selinux-activate`?

8. Какая из следующих команд сделает `app-data-group` группой для файла `datafile.txt`:

   а) `chown app-data-group,ubuntu datafile.txt`;

   б) `chown app-data-group datafile.txt`;

   в) `chown app-data-group:ubuntu datafile.txt`;

   г) `chown ubuntu:app-data-group datafile.txt`?

Ответы

1 — г; 2 — а; 3 — б; 4 — в; 5 — в; 6 — г; 7 — а; 8 — г.

10 Защита сетевых соединений: создание VPN или DMZ

В этой главе

- Внедрение конфигураций безопасности сервера.
- Развертывание туннеля OpenVPN для защиты удаленных соединений.
- Использование брандмауэров для управления доступом между сегментами.
- Использование iptables и Shorewall для создания сети на основе DMZ.
- Тестирование сетевых решений с применением виртуальных сред.

Говорят, что мы живем в гипермобильном мире. Например, я редко покидаю свой домашний офис. Я получаю удовольствие от работы дома, потому что все серверные ресурсы, которые мне могут понадобиться, доступны удаленно. Видимо, я не одинок.

Почти каждый человек, чья работа связана с IT, будет время от времени получать доступ удаленно к своим профессиональным инструментам. И, учитывая, что публичные сети, через которые вы получаете доступ ко всем удаленным терминалам, по своей природе небезопасны, необходимо тщательно контролировать эти соединения.

В предыдущей главе основное внимание уделялось обеспечению того, чтобы данные, используемые вашими удаленными клиентами, надежно передавались и были невидимы для тех, кто может скрываться в подключенной сети. В этой главе, напротив, основное внимание я уделю тому, чтобы данные, используемые удаленными клиентами, надежно передавались и были невидимы для тех, кто может скрываться в подключенной сети. Увидели разницу? И я нет.

На самом деле существует множество технологий, предназначенных для защиты сетевых коммуникаций, и *принцип глубокой защиты* учит нас никогда не полагаться

на них. Здесь вы узнаете о добавлении новых уровней защиты для ваших удаленных действий.

В этой главе вы вернетесь к старым друзьям. Вы будете работать с виртуальными машинами VirtualBox и использовать шифрование для создания туннеля виртуальной частной сети (VPN), чтобы обеспечить безопасные и невидимые удаленные подключения. А в отдельном проекте вы будете разрабатывать более сложные архитектуры брандмауэров, чтобы стратегически разделить вашу сеть на изолированные сегменты. Наконец, вы создадите виртуальную сетевую среду в VirtualBox, чтобы иметь возможность протестировать свои настройки.

10.1. Создание туннеля OpenVPN

В этой книге я уже немало рассказал о шифровании. SSH и SCP могут защитить данные, передаваемые через удаленные соединения (глава 3), шифрование файлов позволяет защитить данные при хранении на сервере (глава 8), а сертификаты TLS/SSL способны защитить данные при передаче между сайтами и клиентскими браузерами (глава 9). Но иногда ваши данные требуют защиты в более широком спектре соединений. Например, возможно, некоторым членам вашей команды приходится работать в дороге, подключаясь к сети по Wi-Fi через общедоступные точки доступа. Определенно не стоит полагать, что все такие точки доступа безопасны, но вашим людям действительно нужен способ подключения к ресурсам компании — и в этом случае поможет VPN.

Правильно спроектированный VPN-туннель обеспечивает прямое соединение между удаленными клиентами и сервером таким образом, чтобы скрывать данные при их передаче по небезопасной сети. Ну и что? Вы уже видели много инструментов, которые могут сделать это с помощью шифрования. Реальная ценность VPN состоит в том, что, открыв туннель, можно подключать удаленные сети, как если бы они были все вместе локальными. В некотором смысле вы используете обход.

Используя такую расширенную сеть, администраторы могут выполнять свою работу на своих серверах из любого места. Но, что более важно, компания с ресурсами, распределенными по нескольким филиалам, может сделать их все видимыми и доступными для всех групп, которые в них нуждаются, где бы они ни находились (рис. 10.1).

Сам по себе туннель не гарантирует безопасность. Но один из стандартов шифрования может быть включен в структуру сети, что существенно повышает уровень безопасности. Туннели, созданные с применением пакета OpenVPN с открытым исходным кодом, используют то же шифрование TLS/SSL, о котором вы уже читали. OpenVPN не единственный доступный вариант для туннелирования, но один из самых известных. Считается, что он немного быстрее и безопаснее, чем альтернативный туннельный протокол уровня 2, использующий шифрование IPsec.

Вы хотите, чтобы в вашей команде все безопасно общались друг с другом, находясь в дороге или работая в разных зданиях? Для этого необходимо создать сервер OpenVPN, чтобы разрешить совместное использование приложений и доступ

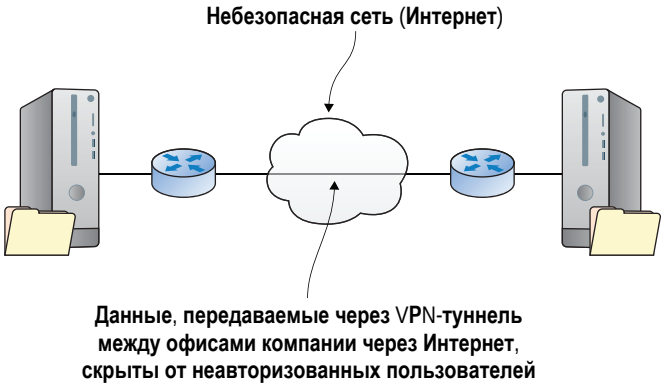


Рис. 10.1. Туннель, соединяющий удаленные частные соединения через публичную сеть

к локальной сетевой среде сервера. Чтобы это работало, достаточно запустить две виртуальные машины или два контейнера: одну для выполнения роли сервера/хоста, а другую — для клиента. Создание VPN — непростой процесс, поэтому, вероятно, стоит потратить несколько минут на то, чтобы представить себе общую картину.

10.1.1. Конфигурирование сервера OpenVPN

Прежде чем начать, дам вам полезный совет. Если вы собираетесь сделать все самостоятельно (а я настоятельно рекомендую вам это), то, вероятно, обнаружите, что работаете с несколькими терминальными окнами, открытыми на Рабочем столе, каждое из которых подключено к своей машине. Есть риск, что в какой-то момент вы введете команду не то в окно. Во избежание этого можно использовать команду `hostname`, чтобы изменить имя машины, отображаемое в командной строке, на что-то, что будет четко говорить вам о том, где вы находитесь. Как только сделаете это, вам нужно будет выйти с сервера и зайти снова, чтобы новые настройки вступили в силу. Вот как это выглядит:

```
ubuntu@ubuntu:~# hostname OpenVPN-Server
ubuntu@ubuntu:~$ exit
<Host Workstation>$ ssh ubuntu@10.0.3.134
ubuntu@OpenVPN-Server:~#
```

Активизируйте новое имя хоста, выйдя из оболочки и снова войдя в систему

Придерживаясь этого подхода и присваивая соответствующие имена каждой из машин, с которыми вы работаете, вы сможете легко отслеживать, где находитесь.

ПРИМЕЧАНИЕ

После использования hostname вы можете столкнуться с раздражающими сообщениями Unable to Resolve Host OpenVPN-Server при выполнении последующих команд. Обновление файла /etc/hosts с соответствующим новым именем хоста должно решить проблему.

Подготовка вашего сервера для OpenVPN

Для установки OpenVPN на вашем сервере требуется два пакета: openvpn и easy-rsa (для управления процессом генерации ключа шифрования). Пользователи CentOS должны при необходимости сначала установить репозиторий epel-release, как вы делали это в главе 2. Чтобы иметь возможность проверить доступ к серверному приложению, вы также можете установить веб-сервер Apache (apache2 для Ubuntu и httpd на CentOS).

Пока вы настраиваете сервер, советую активировать брандмауэр, который блокирует все порты, кроме 22 (SSH) и 1194 (порт OpenVPN по умолчанию). Этот пример иллюстрирует, как будет работать ufw в Ubuntu, но я уверен, что вы все еще помните программу firewalld CentOS из главы 9:

```
# ufw enable
# ufw allow 22
# ufw allow 1194
```

Чтобы разрешить внутреннюю маршрутизацию между сетевыми интерфейсами на сервере, вам нужно раскомментировать одну строку (`net.ipv4.ip_forward = 1`) в файле `/etc/sysctl.conf`. Это позволит перенаправлять удаленных клиентов по мере необходимости после их подключения. Чтобы новый параметр заработал, запустите `sysctl -p`:

```
# nano /etc/sysctl.conf
# sysctl -p
```

Теперь серверная среда полностью настроена, но нужно сделать еще кое-что, прежде чем вы будете готовы: придется выполнить следующие шаги (мы подробно рассмотрим их далее).

1. Создайте на сервере набор ключей для шифрования инфраструктуры открытых ключей (PKI) с помощью сценариев, поставляемых с пакетом easy-rsa. По сути, сервер OpenVPN также действует как собственный центр сертификации (ЦС).
2. Подготовьте соответствующие ключи для клиента.
3. Сконфигурируйте файл `server.conf` для сервера.
4. Настройте ваш клиент OpenVPN.
5. Проверьте свой VPN.

Генерирование ключей шифрования

Чтобы не усложнять жизнь, можно настроить свою ключевую инфраструктуру на той же машине, где работает сервер OpenVPN. Однако в рекомендациях по безопасности обычно предлагается использовать отдельный сервер ЦС для развертываний в производственной среде. Процесс генерации и распределения ресурсов ключа шифрования для использования в OpenVPN проиллюстрирован на рис. 10.2.

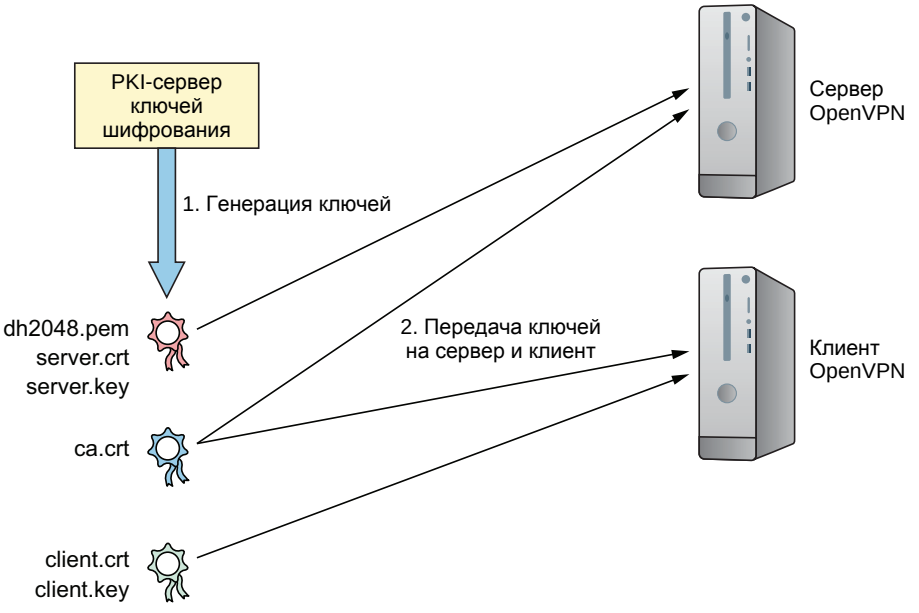


Рис. 10.2. Файлы, которые будут созданы вашим сервером PKI, и места, куда они будут распространяться

Когда вы устанавливали OpenVPN, автоматически был создан каталог `/etc/openvpn/`, но в нем пока ничего нет. Пакеты openvpn и easy-rsa поставляются с примерами файлов шаблонов, которые вы можете использовать как основу для своей конфигурации. Чтобы запустить процесс сертификации, скопируйте каталог шаблона easy-rsa из `/usr/share/` в `/etc/openvpn` и перейдите в каталог `easy-rsa/`:

```
# cp -r /usr/share/easy-rsa/ /etc/openvpn
$ cd /etc/openvpn/easy-rsa
```

Каталог `easy-rsa` теперь будет содержать довольно много сценариев. В табл. 10.1 перечислены инструменты, которые вы будете использовать для создания ключей.

Таблица 10.1. Ключевые сценарии easy-rsa и их функции

Название сценария	Функция
clean-all	Удаляет старые файлы ключей для подготовки к генерации нового ключа
pkitool	Фронтенд для OpenSSL (выполняет большую часть тяжелой работы по генерации ключей)
build-ca	Использует сценарий pkitool для генерации корневого сертификата
build-key-server server	Использует сценарий pkitool для генерации пары ключей и сертификата
build-dh	Устанавливает параметры аутентификации Диффи — Хеллмана

ПРИМЕЧАНИЕ

Перечисленные операции требуют полномочий root, поэтому через sudo su вам нужно стать root.

Первый файл, с которым вы будете работать, называется `vars` и содержит переменные окружения, которые `easy-rsa` использует при генерации ключей. Вам нужно отредактировать файл, чтобы использовать собственные значения вместо значений по умолчанию, которые уже есть. Вот как будет выглядеть мой файл (листинг 10.1).

Листинг 10.1. Основные фрагменты файла /etc/openvpn/easy-rsa/vars

```
export KEY_COUNTRY="CA"
export KEY_PROVINCE="ON"
export KEY_CITY="Toronto"
export KEY_ORG="Bootstrap IT"
export KEY_EMAIL="info@bootstrap-it.com"
export KEY_OU="IT"
```

Запуск файла `vars` позволит передать его значения в среду оболочки, откуда они будут включены в содержимое ваших новых ключей. Почему команда `sudo` сама по себе не работает? Потому что на первом этапе мы редактируем сценарий с именем `vars`, а затем применяем его. *Применение* и означает, что файл `vars` передает свои значения в среду оболочки, откуда они будут включены в содержимое ваших новых ключей.

Обязательно запустите файл повторно, используя новую оболочку, чтобы завершить незаконченный процесс. Когда это будет сделано, сценарий предложит вам запустить другой сценарий, `clean-all`, для удаления любого содержимого в каталоге `/etc/openvpn/easy-rsa/keys/`:

```
$ cd /etc/openvpn/easy-rsa/
# . ./vars                    <-- Эта команда требует привилегий sudo
NOTE: If you run ./clean-all,
  I will be doing a rm -rf on /etc/openvpn/easy-rsa/keys
```

Естественно, следующим шагом будет запуск сценария `clean-all`, за которым следует `build-ca`, который использует сценарий `pkitool` для создания корневого сертификата. Вас попросят подтвердить настройки идентификации, предоставленные `vars`:

```
# ./clean-all
# ./build-ca
Generating a 2048 bit RSA private key
```

Далее идет скрипт `build-key-server`. Поскольку он использует тот же сценарий `pkitool` вместе с новым корневым сертификатом, вы увидите те же вопросы для подтверждения создания пары ключей. Ключам будут присвоены имена на основе передаваемых вами аргументов, которые, если только вы не запускаете несколько VPN на этом компьютере, обычно будут `server`, как в примере:

```
# ./build-key-server server
[...]
```

```
Certificate is to be certified until Aug 15 23:52:34 2027 GMT (3650 days)
Sign the certificate? [y/n]:y
1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

OpenVPN использует параметры, сгенерированные с помощью алгоритма Диффи — Хеллмана (с применением `build-dh`), чтобы согласовать аутентификацию для новых соединений. Созданный здесь файл не должен быть секретным, но должен быть сгенерирован с использованием сценария `build-dh` для ключей RSA, которые в данный момент активны. Если вы создадите новые ключи RSA в будущем, вам также потребуется обновить файл на основе алгоритма Диффи — Хеллмана:

```
# ./build-dh
```

Ваши ключи на стороне сервера теперь попадут в каталог `/etc/openvpn/easy-rsa/keys/`, но OpenVPN этого не знает. По умолчанию OpenVPN будет искать ключи в `/etc/openvpn/`, поэтому скопируйте их:

```
# cp /etc/openvpn/easy-rsa/keys/server* /etc/openvpn
# cp /etc/openvpn/easy-rsa/keys/dh2048.pem /etc/openvpn
# cp /etc/openvpn/easy-rsa/keys/ca.crt /etc/openvpn
```

Подготовка ключей шифрования клиента

Как вы уже видели, в TLS-шифровании используются пары соответствующих ключей: один установлен на сервере, а другой — на удаленном клиенте. Это означает, что вам понадобятся ключи клиента. Наш старый друг pkitool — именно то, что нужно для этого. В данном примере, запуская программу в каталоге `/etc/openvpn/easy-rsa/`, мы передаем ей аргумент `client` для генерации файлов под названием `client.crt` и `client.key`:

```
# ./pkitool client
```

Два клиентских файла вместе с исходным файлом `ca.crt`, который все еще находится в каталоге `keys/`, теперь должны быть безопасно переданы вашему клиенту. Из-за их принадлежности и прав доступа это может оказаться не таким легким делом. Самый простой подход — вручную скопировать содержимое исходного файла (и ничего, кроме этого содержимого) в терминал, работающий на **Рабочем столе** вашего ПК (выделите текст, щелкните на нем правой кнопкой мыши и выберите в меню пункт **Copy** (Копировать)). Затем вставьте это в новый файл с тем же именем, которое вы создаете во втором терминале, подключенном к вашему клиенту.

Но ведь любой может вырезать и вставить. Вы же вместо этого думайте как администратор, потому что у вас не всегда будет доступ к GUI, где возможна операция вырезания/вставки. Скопируйте файлы в домашний каталог вашего пользователя (чтобы удаленная операция `scp` могла получить к ним доступ), а затем с помощью `chown` измените владельца файлов с **root** на обычного пользователя без полномочий **root**, чтобы можно было выполнить удаленное действие `scp`. Убедитесь, что все ваши

файлы на данный момент установлены и доступны. На клиент вы переместите их чуть позже:

```
# cp /etc/openvpn/easy-rsa/keys/client.key /home/ubuntu/
# cp /etc/openvpn/easy-rsa/keys/ca.crt /home/ubuntu/
# cp /etc/openvpn/easy-rsa/keys/client.crt /home/ubuntu/
# chown ubuntu:ubuntu /home/ubuntu/client.key
# chown ubuntu:ubuntu /home/ubuntu/client.crt
# chown ubuntu:ubuntu /home/ubuntu/ca.crt
```

С полным набором ключей шифрования, готовых к действию, вам нужно сообщить серверу, как вы хотите создать VPN. Это делается с помощью файла `server.conf`.

Уменьшаем количество нажатий клавиш

Слишком много приходится печатать? Расширение со скобками поможет уменьшить эти шесть команд до двух. Я уверен, что вы сможете изучить эти два примера и понять, что происходит. Что еще более важно, вы сможете понять, как применять данные принципы к операциям, включающим десятки или даже сотни элементов:

```
# cp /etc/openvpn/easy-rsa/keys/{ca.crt,client.{key,crt}} /home/ubuntu/
# chown ubuntu:ubuntu /home/ubuntu/{ca.crt,client.{key,crt}}
```

Настройка файла server.conf

Откуда вы можете знать, как должен выглядеть файл `server.conf`? Помните шаблон каталога `easy-rsa`, который вы скопировали из `/usr/share/`? При установке OpenVPN остался сжатый файл шаблона конфигурации, который вы можете скопировать в `/etc/openvpn/`. Я буду опираться на тот факт, что шаблон заархивирован, и познакомлю вас с полезным инструментом: `zcat`.

Вы уже знаете о выводе текстового содержимого файла на экран с помощью команды `cat`, но что, если файл сжат с помощью gzip? Вы всегда можете распаковать файл, и тогда `cat` с удовольствием его выведет, но это на один или два шага больше, чем нужно. Вместо этого, как вы уже, наверное, догадались, можно ввести команду `zcat` для загрузки распакованного текста в память за один шаг. В следующем примере вместо того, чтобы печатать текст на экране, вы перенаправите его в новый файл с именем `server.conf`:

```
# zcat \
 /usr/share/doc/openvpn/examples/sample-config-files/server.conf.gz \
 > /etc/openvpn/server.conf
$ cd /etc/openvpn
```

Оставим в стороне обширную и полезную документацию, прилагаемую к файлу, и посмотрим, как он может выглядеть, когда вы закончите редактирование. Обратите внимание, что точка с запятой (`;`) указывает OpenVPN не читать и не выполнять следующую строку (листинг 10.2).

Листинг 10.2. Активные настройки из файла /etc/openvpn/server.conf

```
port 1194
# TCP or UDP server?
proto tcp
;proto udp
;dev tap
dev tun
ca ca.crt
cert server.crt
key server.key # This file should be kept secret
dh dh2048.pem
server 10.8.0.0 255.255.255.0
ifconfig-pool-persist ipp.txt
push "route 10.0.3.0 255.255.255.0"
keepalive 10 120
comp-lzo
port-share localhost 80
user nobody
group nogroup
persist-key
persist-tun
status openvpn-status.log
log openvpn.log
;log-append openvpn.log
verb 3
```

Минимизирует привилегированный доступ к системе

Записывает журналы сеансов в /etc/openvpn/openvpn.log

Задает уровень детализации вывода, значение может доходить до 9

Пройдемся по некоторым из этих настроек.

- По умолчанию OpenVPN работает через порт 1194. Вы можете изменить это, например, чтобы еще больше скрыть свои действия или избежать конфликтов с другими активными туннелями. Поскольку 1194 требует минимальной координации с клиентами, лучше всего сделать именно так.
- OpenVPN применяет для передачи данных либо протокол управления передачей (TCP), либо протокол пользовательских датаграмм (UDP). TCP может быть немного медленнее, но он более надежен и с большей вероятностью понятен приложениям, работающим на обоих концах туннеля.
- Вы можете указать `dev tun`, когда хотите создать более простой и эффективный IP-туннель, который передает содержимое данных и ничего больше. Если, с другой стороны, вам потребуется подключить несколько сетевых интерфейсов (и сетей, которые они представляют), создав мост Ethernet, то придется выбрать `dev tap`. Если вы не понимаете, что все это значит, используйте аргумент `tun`.
- Следующие четыре строки передают OpenVPN имена трех файлов аутентификации на сервере и файл параметров `dh2048`, который вы создали ранее.
- Строка `server` устанавливает диапазон и маску подсети, которые будут использоваться для присваивания IP-адресов клиентам при входе в систему.
- Необязательный параметр `push "route 10.0.3.0 255.255.255.0"` позволяет удаленным клиентам получать доступ к частным подсетям за сервером. Для вы-

полнения этой работы также требуется настройка сети на самом сервере, чтобы частная подсеть знала о подсети OpenVPN (10.8.0.0).

- Строка `port-share localhost 80` позволяет перенаправлять клиентский трафик, поступающий через порт 1194, на локальный веб-сервер, прослушивающий порт 80. (Будет полезно в том случае, если вы собираетесь задействовать веб-сервер для тестирования вашей VPN.) Это работает только тогда, когда выбран протокол tcp.
- Строки `user nobody` и `group nogroup` должны быть активны — для этого нужно удалить точки с запятой (`;`). Принудительная работа удаленных клиентов под `nobody` и `nogroup` гарантирует, что сеансы на сервере будут непривилегированными.
- `log` указывает, что текущие записи в журнале будут перезаписывать старые записи при каждом запуске OpenVPN, тогда как `log-append` добавляет новые записи в существующий файл журнала. Сам файл `openvpn.log` записывается в каталог `/etc/openvpn/`.

Кроме того, в файл конфигурации также часто добавляют значение `client-to-client`, чтобы несколько клиентов могли видеть друг друга в дополнение к серверу OpenVPN. Если вы удовлетворены своей конфигурацией, то можно запускать сервер OpenVPN:

```
# systemctl start openvpn
```

ПРИМЕЧАНИЕ

Из-за меняющегося характера отношений между OpenVPN и systemd для запуска службы иногда может потребоваться такой синтаксис: systemctl start openvpn@server.

Запуск `ip addr` для отображения списка сетевых интерфейсов вашего сервера теперь должен вывести ссылку на новый интерфейс с именем tun0. OpenVPN создаст его для обслуживания входящих клиентов:

```
$ ip addr
[...]
4: tun0: mtu 1500 qdisc [...]
    link/none
    inet 10.8.0.1 peer 10.8.0.2/32 scope global tun0
       valid_lft forever preferred_lft forever
```

Возможно, вам потребуется перезагрузить сервер, прежде чем все начнет полностью работать. Следующая остановка — клиентский компьютер.

10.1.2. Конфигурирование клиента OpenVPN

Традиционно туннели строятся как минимум с двумя выходами (иначе мы бы называли их пещерами). Правильно настроенный OpenVPN на сервере направляет трафик в туннель и из него с одной стороны. Но вам также потребуется какое-то

программное обеспечение, работающее на стороне клиента, то есть на другом конце туннеля.

В этом разделе я собираюсь сосредоточиться на ручной настройке компьютера с Linux того или иного типа для работы в качестве клиента OpenVPN. Но это не единственный способ, с помощью которого доступна такая возможность. OpenVPN поддерживает клиентские приложения, которые можно устанавливать и использовать на настольных компьютерах и ноутбуках с Windows или macOS, а также на смартфонах и планшетах на базе Android и iOS. Подробности см. на сайте openvpn.net.

Пакет OpenVPN необходимо будет установить на клиентском компьютере, как он был установлен на сервере, хотя здесь нет необходимости в `easy-rsa`, поскольку используемые вами ключи уже существуют. Вам нужно скопировать файл шаблона `client.conf` в каталог `/etc/openvpn/`, который только что был создан. На этот раз файл не будет заархивирован, поэтому обычная команда `cp` отлично справится с данной задачей:

```
# apt install openvpn
# cp /usr/share/doc/openvpn/examples/sample-config-files/client.conf \
 /etc/openvpn/
```

Большинство настроек в вашем файле `client.conf` будут довольно понятными: они должны соответствовать значениям на сервере. Как видно из следующего примера файла, уникальным параметром является `remote 192.168.1.23 1194`, который указывает клиенту IP-адрес сервера. Опять же убедитесь, что это адрес вашего сервера. Вы также должны заставить компьютер клиента проверять подлинность сертификата сервера, чтобы предотвратить возможную атаку «человек посередине». Один из способов сделать это — добавить строку `remote-cert-tls server` (листинг 10.3).

Листинг 10.3. Активные настройки в файле /etc/openvpn/client.conf VPN-клиента

```
client
;dev tap
dev tun
proto tcp
remote 192.168.1.23 1194
resolv-retry infinite
nobind
user nobody
group nogroup
persist-key
persist-tun
ca ca.crt
cert client.crt
key client.key
comp-lzo
verb 3
remote-cert-tls server
```

Определяет компьютер как VPN-клиент, конфигурация которого будет основана на удаленном сервере

Адрес и сетевой порт, используемый для доступа к серверуVPN

Заставляет проверять сертификат сервера

Теперь вы можете перейти в каталог `/etc/openvpn/` и извлечь ключи сертификации с сервера. Замените IP-адрес сервера или имя домена в примере своими значениями:

```
$ cd /etc/openvpn
# scp ubuntu@192.168.1.23:/home/ubuntu/ca.crt .
# scp ubuntu@192.168.1.23:/home/ubuntu/client.crt .
# scp ubuntu@192.168.1.23:/home/ubuntu/client.key .
```

Точка (.) в конце указывает scp сохранить файл в текущем каталоге

Ничего захватывающего, скорее всего, не произойдет, пока вы не запустите OpenVPN на клиенте. Поскольку вам нужно передать пару аргументов, вы сделаете это из командной строки. Аргумент `--tls-client` сообщает OpenVPN, что вы будете действовать как клиент и подключаться через шифрование TLS, а `--config` указывает на ваш файл конфигурации:

```
# openvpn --tls-client --config /etc/openvpn/client.conf
```

Внимательно прочитайте вывод команды, чтобы убедиться, что вы правильно подключены. Если в первый раз что-то пойдет не так, возможно, это связано с несоответствием настроек между файлами конфигурации сервера и клиента или с проблемой сетевого подключения/брандмауэра. Вот несколько советов, как устранить неполадки.

- Внимательно прочитайте вывод операции OpenVPN на клиенте. Он часто содержит ценные советы о том, что именно не может быть выполнено и почему.
- Проверьте сообщения об ошибках в файлах `openvpn.log` и `openvpn-status.log` в каталоге `/etc/openvpn/` на сервере.
- Проверьте связанные с OpenVPN и подходящие по времени сообщения в системных журналах на сервере и клиенте. (`journalctl -ce` выведет на экран самые последние записи.)
- Убедитесь, что у вас есть активное сетевое соединение между сервером и клиентом (подробнее — в главе 14).

10.1.3. Тестирование вашего VPN

Если при загрузке на клиенте OpenVPN выдал нормальную информацию, вам следует перейти к тестированию туннеля, чтобы убедиться, что он работает и защищает ваше соединение. Запуск команды `curl` с клиента по адресу сервера с использованием порта OpenVPN должен вернуть файл `index.html` в корневой веб-каталог (`/var/www/html/`):

```
curl 192.168.1.23:1194
```

Проверьте настройки, удалив номер порта и двоеточие перед ним. Если ваш брандмауэр использует ранее заданные настройки, `curl` не сработает.

На самом деле это не такой и полезный тест. В конце концов, цель VPN состоит в том, чтобы защитить вашу сессию от других пользователей в сети, поэтому

возможность загрузки страницы ничего не доказывает. Не вдаваясь в подробности, в следующих разделах я предлагаю несколько способов (без определенного порядка), чтобы подтвердить, что ваша VPN работает должным образом.

Подслушивание в сети

Настройте в качестве клиента обычный настольный ПК с графическим интерфейсом и используйте браузер, чтобы открыть серверное приложение, которое потребует ввода данных. Для приложения вы можете взять простую форму HTML, как показано в листинге 10.4.

Листинг 10.4. Простая HTML-страница с формой ввода данных в браузере

```
<!DOCTYPE html>
<html>
<body>
<form action="/action_page.php">
  First name:<br>
  <input type="text" name="firstname" value="Your">
  <br>
  Last name:<br>
  <input type="text" name="lastname" value="Name">
  <br><br>
  <input type="submit" value="Submit">
</form>
</body>
</html>
```

Сохраните файл в корневом каталоге веб-документа сервера, например, как `form.html`. Затем вы получите доступ к странице по URL-адресу 192.168.1.23/form.html (замените IP-адресом вашего сервера).

Чтобы выяснить, была ли введенная вами информация видна неавторизованным пользователям, можете воспользоваться программным обеспечением для отслеживания сетевых пакетов, например Wireshark, предназначенным для сбора и анализа данных. Если бы туннель работал для шифрования сеанса, то пакеты с введенными вами именами были бы зашифрованы.

К сожалению, действительно полезное руководство по настройке и использованию Wireshark потребует целой главы. Поскольку это будет существенным отклонением от нашей основной темы — Linux, — такой главе придется искать себе место в другой книге.

Сетевая инфраструктура

Другой подход к тестированию VPN заключается в настройке ресурсов (серверов, маршрутизаторов и т. д.) в локальной сети сервера, куда вы направляете клиентов по настройкам из файла `server.conf`. Может быть непросто построить для этого инфраструктуру того или иного вида. Но если вы справитесь с такой задачей и ваш

клиент сможет получить доступ к этим ресурсам, то вы будете знать, что все работает. В конце этой главы я добавил руководство по использованию VirtualBox для создания подобной инфраструктуры.

10.2. Построение сетей, защищенных от вторжений

VPN благодаря волшебству шифрования отлично подходят для защиты данных сессии. А брандмауэры могут прекрасно контролировать входящий и исходящий трафик по информации о порте, протоколе или маршрутизации. Теоретически этого должно быть достаточно для защиты ваших серверных сетей. В конце концов, если вы заблокировали все, кроме нескольких подступов к своей сети, и усилили блокировки для немногих доступов, которые все еще нужны (используя, например, аутентификацию пары ключей без пароля для SSH), то вы должны быть в безопасности. Но если бы все было так просто...

Независимо от того, насколько хорошо вы все продумали, нет такого понятия, как 100%-ная уверенность, когда дело доходит до IT-безопасности. Вы будете делать все возможное, чтобы получать информацию о новых видах угроз, исправлять свое программное обеспечение, проверять свои процессы и настраивать брандмауэр, но какой-то злоумышленник всегда найдет лазейку. Сегодня это лишь вопрос времени — когда машины, способные взломать алгоритмы шифрования, станут широкодоступными.

Фактически решение состоит в том, чтобы обеспечить максимальную безопасность, добавив как можно больше уровней в свой профиль безопасности. Тогда, даже если плохие парни проберутся через одни ворота, перед ними все равно будут еще две или три закрытые двери. (Ранее это называлось защитой в глубину.) Подумав, вы можете решить, что один из таких уровней подразумевает разделение ваших ресурсов на несколько изолированных сетей в надежде на то, что, если один из ресурсов окажется взломан, другие все же могут быть защищены.

Основная цель состоит в том, чтобы безопасно открыть все службы, которые должны быть видны, и защищать все остальное. Допустим, вы используете веб-сервер, на котором размещено общедоступное приложение. Браузеры пользователей загружают свои страницы с веб-сервера, а пользовательская информация и данные приложений обрабатываются вторым сервером, где расположена база данных. Вы захотите предоставить всем на Земле (а также всем, кто может наслаждаться видом на околоземной орбите) доступ к вашему веб-серверу, но при этом закрыть общий доступ к серверу с базой данных. Обсудим, как провернуть этот трюк.

10.2.1. Демилитаризованные зоны (DMZ)

Одна из популярных архитектур изоляции известна как *DMZ* (сокращение от выражения, используемого для описания географического буфера между двумя недоверчивыми странами: демилитаризованная зона). Идея состоит в том, чтобы

разделить ваши серверы, рабочие станции, Wi-Fi-маршрутизаторы и другие ресурсы на отдельные сети. Чтобы это работало, каждое из ваших устройств должно быть физически подключено к отдельному сетевому интерфейсу.

Простая реализация DMZ заключается в применении одного сервера в качестве маршрутизатора для перенаправления трафика между Интернетом и двумя внутренними сетями (рис. 10.3). Одна из сетей может охватывать серверные базы данных или рабочие компьютеры, расположенные в вашем офисе. Эта сеть будет тщательно защищена строгими правилами доступа. У другой сети будет довольно прямой и легкий доступ к внешнему миру, и она сможет охватывать общедоступные ресурсы, такие как веб-серверы.

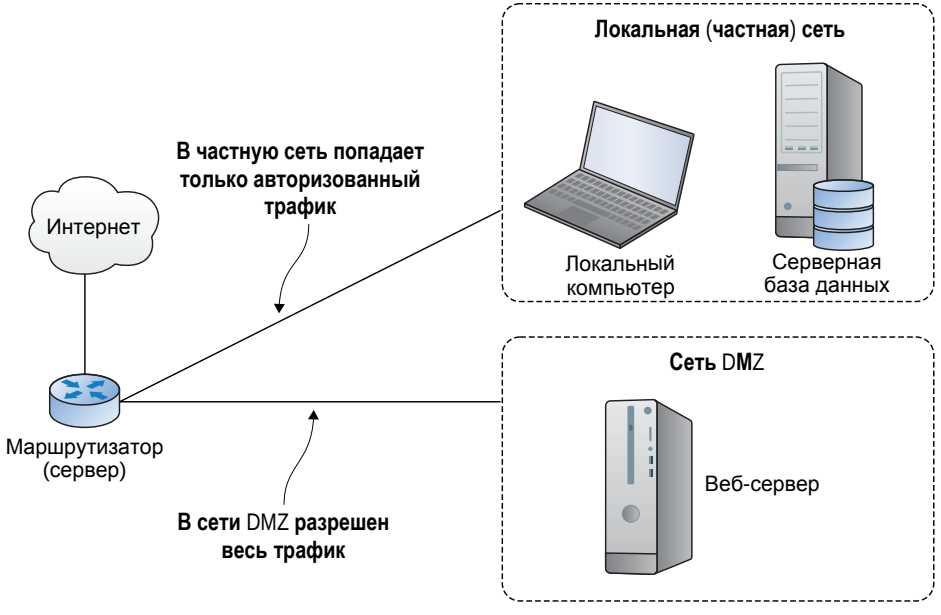


Рис. 10.3. Простая архитектура DMZ с тремя интерфейсами

Прежде чем перейти к созданию собственной среды DMZ, вы должны знать по крайней мере пару альтернатив этой модели. Я кратко расскажу о некоторых из них далее.

Сети, защищенные от вторжения: конструктивные соображения

В дополнение к DMZ, которые являются надежными и достаточно гибкими, чтобы подходить для широкого диапазона вариантов использования, существуют другие модели конфигурации защищенной сети, которые вы могли бы рассмотреть.

Сервер перехода (Jump server) (иногда называемый *jumpbox* или, возможно, узел-бастион) — это легкий сервер с подключением к Интернету, который выполняет только одну функцию: позволяет удаленным SSH-клиентам входить в систему, а затем «переходить» к другим серверам, работающим в частной сети сервера перехода. Как показано на рис. 10.4, частные серверы в сети будут иметь определенные средства управления доступом, настроенные для разрешения доступа только к удаленным учетным записям, поступающим с сервера перехода.

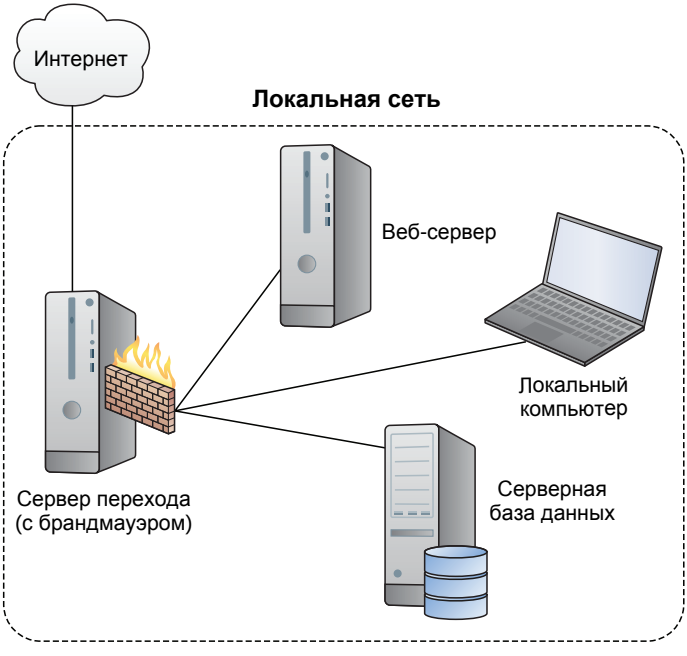


Рис. 10.4. Типичная архитектура сервера перехода

Серверы перехода (по определению представляющие собой возможную единственную точку отказа и дополнительный уровень сложности) сегодня не так популярны, как когда-то. Как упоминалось в главе 9, для управления связностью и безопасностью при крупных корпоративных развертываниях могут использоваться частные аппаратные решения, предоставляемые такими компаниями, как Juniper и Cisco. Но затраты на такое оборудование и на серьезное обучение, которое потребуется пройти, прежде чем вы сможете грамотно использовать все это, могут перечеркнуть все преимущества.

Ни одна из этих систем не будет иметь особого смысла для гибридных облачных решений, где инфраструктура компании размещается как локально, так и на удаленных облачных платформах. Но основные инструменты и концепции дизайна всегда будут полезны.

Зачем использовать DMZ

Далее я собираюсь обсудить создание DMZ с помощью двух широко используемых пакетов программных брандмауэров: iptables и Shorewall. Однако прежде, чем я это сделаю, должен сказать вам, что оба предоставляют почти одну и ту же функциональность. Хотя Shorewall — более удобный инструмент, который только манипулирует правилами iptables без вашего ведома. Вот две причины, по которым я хотел бы, чтобы вы познакомились с обоими подходами.

- Всегда есть некие практические соображения, по которым выбираются конкретные инструменты, поэтому наличие нескольких уже готовых вариантов станет для вас преимуществом.
- Даже если вы в конечном итоге используете более простое решение наподобие Shorewall, хорошо хотя бы просто посмотреть на iptables в его естественной среде обитания. Это поможет вам лучше понять брандмауэры Linux в целом.

Итак, начнем. Предупреждаю: впереди сложная тема. Перейдите к концу главы, если хотите. Но не забудьте вернуться сюда, когда вас попросят построить такую инфраструктуру.

10.2.2. Использование iptables

iptables — это инструмент для управления правилами брандмауэра на компьютере с Linux. iptables? Но как насчет наборов команд ufw и firewalld, о которых вы узнали в предыдущей главе? Возможно, вам не нравится выбирать? И не испорчу ли я вам день, если скажу, что существует четвертый инструмент под названием nftables?

Признаю, что все это немного забавно, поэтому позвольте мне объяснить. Все начинается с *netfilter*, который контролирует доступ к сетевому стеку и из него на уровне модуля ядра Linux. В течение десятилетий основным инструментом командной строки для управления сетевым фильтром netfilter был набор правил iptables.

Поскольку синтаксис, необходимый для вызова этих правил, может показаться немного запутанным, были созданы различные удобные для пользователя реализации, такие как ufw и firewalld. Они представляют собой интерпретаторы сетевого фильтра более высокого уровня. При этом в первую очередь ufw и firewalld предназначены для решения проблем, с которыми сталкиваются отдельные компьютеры. Создание полномасштабных сетевых решений часто требует дополнительных возможностей iptables или его замены (с 2014 года) nftables (с помощью инструмента командной строки nft).

iptables никуда не делся и до сих пор очень популярен. На самом деле вы должны иметь в виду, что, работая администратором, на протяжении многих лет будете сталкиваться с сетями, защищенными iptables. Но nftables привнес некоторые важные новые функциональные возможности, добавив их к классическому набору инструментов netfilter. Тем не менее для простоты я собираюсь использовать iptables, чтобы быстро показать вам, как создавать DMZ из командной строки. Сделав это, мы перейдем к более интуитивной реализации Shorewall.

10.2.3. Создание DMZ с помощью iptables

iptables должен быть установлен по умолчанию. Чтобы убедиться в этом, запустите `iptables -L` — так вы просмотрите все текущие правила. Если вы еще не работали с этим инструментом, то должны увидеть полностью открытый (ACCEPT) брандмауэр. Обратите внимание, что цепочки INPUT, FORWARD и OUTPUT по умолчанию являются частью таблицы Filter. Таблица NAT, которая также существует по умолчанию, может использоваться для изменения (или трансляции) информации о маршрутизации пакета, чтобы его можно было успешно перемещать между сетями. Мы вернемся к NAT чуть позже:

Вывод (в данном случае это часть таблицы Filter) показывает правила в цепочке: INPUT в этом первом примере

```
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source       destination
ACCEPT     tcp  --  anywhere     anywhere      tcp dpt:domain
ACCEPT     udp  --  anywhere     anywhere      udp dpt:domain
ACCEPT     tcp  --  anywhere     anywhere      tcp dpt:bootps
ACCEPT     udp  --  anywhere     anywhere      udp dpt:bootps

Chain FORWARD (policy ACCEPT)
target     prot opt source       destination
ACCEPT     all  --  anywhere     anywhere
ACCEPT     all  --  anywhere     anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source       destination
```

После того как вы получили общий план, который в данном случае касается разделения серверов по типам, которые вы задействуете с помощью DMZ, вы должны установить строгую политику по умолчанию.

СОВЕТ

Правило администратора брандмауэра № 1: всегда знайте, чем должно закончиться действие, прежде чем начать его выполнение.

Эти правила формируют базовый уровень брандмауэра, блокируя (DROP) весь входящий и перенаправляющийся трафик для всех интерфейсов, но разрешая исходящий:

```
# iptables -P INPUT DROP
# iptables -P FORWARD DROP
# iptables -P OUTPUT ACCEPT
```

Поскольку подключение к Интернету осуществляется через маршрутизирующий сервер (возможно, через сетевой коммутатор), все устройства в ваших сетях автоматически подчиняются этим правилам. Проверьте сами. Опять же не стесняйтесь использовать инфраструктуру виртуальной сети, описанную ниже. Мы предполагаем,

что три сетевых интерфейса, подключенных к вашему серверу с брандмауэром, названы так, как описано в табл. 10.2.

Таблица 10.2. Обозначения сетевых интерфейсов, используемые в следующих примерах

Назначение	Цель
eth0	Подключен к Интернету
eth1	Подключен к DMZ
eth2	Подключен к локальной частной сети

Соглашения об именах Linux для сетевых интерфейсов

Когда-то давно можно было предположить, что ваш дистрибутив Linux будет назначать простые и понятные имена для физических сетевых интерфейсов, подключенных к вашему компьютеру. Первым распознаваемым интерфейсом Ethernet был бы eth0, вторым — eth1 и т. д.; беспроводные интерфейсы были бы wlan0 и wlan1 и т. д. (Как вы видели в главе 9, виртуальным устройствам, таким как туннели и мосты, можно присваивать имена, используя некие принятые соглашения.) Дело в том, что даже если вы не знаете точного имени интерфейса, то обычно можете его довольно быстро угадать. Но запустите `ip addr` на вашей современной машине с systemd, и вы столкнетесь с чем-то вроде enp1s0. Или как насчет этого ужаса: wlx9cefd5fe4a18?

Вот что происходит. По многим веским причинам системы, использующие предсказуемые имена интерфейсов, более просты и безопасны в управлении, даже если есть некоторые вопросы относительно удобства. Интерфейс Ethernet может состоять из *en* для Ethernet, *p1* для обозначения шины 1 и *s0* для обозначения первого физического слота. А другое устройство может получить *wl* (беспроводная локальная сеть) и *x9cefd5fe4a18*, где *x* указывает, что шестнадцатеричное число, следующее за ним, является MAC-адресом устройства.

Если вы знаете способ подключения устройства и/или его MAC-адрес, то можете легко предположить имя интерфейса. Тем не менее для простоты в примерах этой главы я буду использовать более старые соглашения.

Вы хотите, чтобы устройства в вашей локальной сети могли обмениваться данными с публичными серверами, включая сам маршрутизатор, в демилитаризованной зоне. Следующие два правила, добавленные в цепочку `FORWARD` в таблице `Filter`, позволят пакетам данных перемещаться (`FORWARD`) между сетями, находящимися за интерфейсами eth1 и eth2. `-A` указывает, что все остальное должно быть добавлено в качестве нового правила. Это позволяет вашему веб-серверу в DMZ обмениваться данными с сервером базы данных в частной сети:

```
iptables -A FORWARD -i eth1 -o eth2 -m state \
 --state NEW,ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -i eth2 -o eth1 -m state \
 --state ESTABLISHED,RELATED -j ACCEPT
```

Данное правило будет добавлено в NAT-таблицу (`-t nat`). Здесь используется протокол TCP для обработки всего трафика, поступающего через интерфейс eth0, который нацелен на (вымышленный) публичный IP-адрес вашего приложения (54.4.32.10). Кроме того, указывается, что нужно применять порт HTTP 80. Любой трафик, отвечающий всем этим условиям, будет перенаправлен на IP-адрес вашего веб-сервера в DMZ (192.168.1.20):

```
iptables -t nat -A PREROUTING -p tcp -i eth0 -d 54.4.32.10 \
 --dport 80 -j DNAT --to-destination 192.168.1.20
```

Поскольку вы не открыли доступ к Интернету через интерфейс eth0 ни к DMZ, ни к частной сети, ничто из внешних источников не может достичь локальных серверов.

Вы можете решить, что уже все сделали, только это не так. Подождите примерно полчаса и наверняка начнете получать гневные электронные письма от разработчиков из вашей команды, задающихся вопросом, почему они не могут загружать критические обновления программного обеспечения или смешные видео с котиками. Будьте готовы настроить параметры брандмауэра в соответствии с непредусмотренными особыми потребностями (такими как доступ к удаленным репозиториям программного обеспечения или облачным ресурсам и т. д.). В идеале вы должны предусмотреть все возможные запросы вплоть до создания брандмауэра, но старый подход «спусти курок и посмотри, что получится» тоже работает.

Как я упоминал ранее, iptables больше не в авангарде технологии брандмауэров, и этот пример был очень простым, но я думаю, что он по крайней мере помог проиллюстрировать, как такие инструменты работают. За получением дополнительной информации обратитесь к официальному руководству по работе с nftables, которое доступно по адресу mng.bz/b0DM.

10.2.4. Создание DMZ с помощью Shorewall

Я уверен, вы помните, что наш первый шаг — получить программное обеспечение. Пользователи Ubuntu могут обратиться к APT для установки пакета Shorewall, хотя пользователям CentOS потребуется получить его через репозиторий epel-release:

```
# yum install epel-release
# yum install shorewall
```

В то время как iptables управляется из командной строки, работа с Shorewall осуществляется через файлы конфигурации. Есть что-то в четком визуальном представлении сложных настроек, заключенных в конфигурационные файлы, что я лично считаю обнадеживающим. Как только вы разберетесь в том, как настройки Shorewall распределены по файлам, вы, вероятно, найдете работу с синтаксисом Shorewall гораздо менее сложной, чем с iptables.

Параметры запуска Shorewall задаются файлом `/etc/default/shorewall`, но сама конфигурация брандмауэра обычно определяется несколькими файлами в каталоге `/etc/shorewall/`. `/etc/shorewall/shorewall.conf` устанавливает среду оболочки, в которой будет работать Shorewall. Возможно, для простых проектов

вам даже не придется трогать этот файл или файлы `params` и `conntrack`, которые вы там обнаружите. Однако вам потребуется создать парочку других файлов, шаблоны которых есть в каталоге `/usr/share/doc/shorewall/examples/` (или `/usr/share/doc/shorewall-5.0.14.1/Samples` в CentOS, где 5.0.x — это номер версии). Если вы просмотрите содержимое каталога `examples/`, то увидите четыре подкаталога, которые, в свою очередь, хранят примеры файлов конфигурации для нескольких распространенных сценариев:

```
# ls /usr/share/doc/shorewall/examples/
LICENSE.gz      README.txt        two-interfaces
one-interface   three-interfaces  Universal
```

Хотя пример с тремя интерфейсами хорошо подходит для нашей задачи, мы соберем все, что нам нужно, с нуля, чтобы вы могли четко увидеть, как происходит весь процесс. В табл. 10.3 приведено краткое описание файлов из каталога `/etc/shorewall/`, которые вы можете использовать.

Таблица 10.3. Файлы конфигурации Shorewall, хранящиеся в каталоге /etc/shorewall/

Имя файла	Назначение	Обязателен или нет
zones	Объявляет сетевые зоны, которые нужно создать	Да
interfaces	Определяет, какие сетевые интерфейсы будут использоваться для указанных зон	Да
policy	Устанавливает правила высокого уровня, контролирующие трафик между зонами	Да
rules	Определяет исключения из правил в файле политики	Нет
masq	Задает динамические настройки NAT	Нет
stoppedrules	Определяет поток трафика, когда Shorewall остановлен	Нет
params	Устанавливает переменные оболочки для Shorewall	Нет
conntrack	Исключает указанный трафик из отслеживания соединений с помощью Netfilter	Нет

С помощью команды `man` можно получить много хорошей документации. Например, наберите `man shorewall-` и имя конкретного файла, который вы ищете:

```
$ man shorewall-rules
```

Файл zones

Для создания файла зон воспользуемся текстовым редактором. Первая строка определяет сервер Shorewall как тип `firewall`. Каждая из трех активных зон, которые вы создадите, будет использовать адресацию типа `ipv4` (в отличие от `IPv6`). Зона `net` представляет общедоступную сеть (которую все продвинутые дети называют Интернетом), `dmz` будет общедоступной зоной в вашей инфраструктуре, а `loc` — частной локальной сетью для ваших внутренних серверов (листинг 10.5).

Листинг 10.5. /etc/shorewall/zones

```
fw firewall
net ipv4
dmz ipv4
loc ipv4
```

Файл interfaces

Теперь вам нужно создать файл под названием `interfaces`, где вы будете ассоциировать каждую из ваших новых зон с одним из трех сетевых интерфейсов, которые подключили к серверу Shorewall. `detect` сообщает Shorewall, что настройки сети должны быть автоматически обнаружены; `dhcp` означает, что IP-адреса должны автоматически назначаться вашим интерфейсам DHCP-серверами. А `nosmurfs,routefilter,logmartians` на интернет-интерфейсе будут фильтровать подозрительные пакеты и исходные домены, а также фиксировать события, которые стоит записать в журнал (листинг 10.6).

Листинг 10.6. Файл /etc/shorewall/interfaces

```
net eth0 detect dhcp,nosmurfs,routefilter,logmartians
dmz eth1 detect dhcp
loc eth2 detect dhcp
```

Сопоставляет интерфейс eth0 с зоной Интернета (сети) и устанавливает протоколы поведения

Файл policy

Файл политики устанавливает желаемое поведение по умолчанию. Первая строка в этом примере говорит о том, что будет отбрасываться весь трафик, поступающий из Интернета (`net`), направленный в любой пункт назначения. Исходящий трафик из частной зоны (`loc`) в Интернет разрешен, что позволяет локальным компьютерам получать обновления программного обеспечения. В третьей строке указывается, что трафик, исходящий от брандмауэра, должен приниматься везде. Команды в последней строке указывают отклонять любые пакеты, не охватываемые другими правилами (листинг 10.7).

Листинг 10.7. Файл /etc/shorewall/policy

```
net all DROP
loc net ACCEPT
fw all ACCEPT
all all REJECT
```

DROP удалит отфильтрованные пакеты, но REJECT не только удалит, но и отправит уведомление отправителю

Эта политика должна быть последней, поскольку охватывает весь трафик, который не подходит под предыдущие правила

Файл rules

Единственный оставшийся файл, который вам необходимо создать для этой простой конфигурации, будет хранить правила. Файл правил со временем может стать довольно длинным и сложным, так как требования могут меняться. Однако сейчас

вам понадобится всего несколько строк. Основная цель файла правил — точная настройка исключений из файла политики.

Например, поскольку вы будете работать с веб-сервером в зоне DMZ, вы захотите разрешить весь трафик, использующий протокол TCP, для доступа по порту 80 (небезопасный HTTP) или по порту 443 (безопасный HTTP). Вы также захотите открыть SSH-доступ с локальных серверов к компьютерам (включая сам сервер брандмауэра Shorewall) в DMZ. Как рабочие станции администратора в локальной сети смогут администрировать брандмауэр без SSH?

При необходимости для удаленного доступа по SSH вы также можете открыть порт 22 из сети `net` на брандмауэр. Правило `Web(DNAT)` позволяет перенаправлять порты из Интернета на ваш веб-сервер в DMZ. Хотя это не относится к нашему примеру, если вы в конечном итоге запустите DNS-сервер на своем компьютере с брандмауэром, то также откроете доступ для порта 53 из вашей локальной сети на компьютер с брандмауэром (листинг 10.8).

Листинг 10.8. Файл /etc/shorewall/rules

```
ACCEPT all dmz tcp 80,443
ACCEPT net dmz tcp 22
ACCEPT loc dmz tcp 22
ACCEPT loc fw udp 53
Web(DNAT) net dmz:10.0.1.4
```

Осталось только запустить Shorewall на компьютере с брандмауэром:

```
# systemctl start shorewall
```

ПРИМЕЧАНИЕ

Не забудьте перезапустить Shorewall после внесения изменений в файлы конфигурации. И не думайте, что в дальнейшем вам не придется вносить изменения в конфигурационные файлы, раз вы изо всех сил пытаетесь настроить все так, как положено.

10.3. Построение виртуальной сети для тестирования инфраструктуры

Как и я, вы знаете, что нужно самому попробовать инструмент, прежде чем получится понять, как он работает. Но для тестирования развертываний, обсуждаемых в этой главе, потребуется более одного или двух виртуальных серверов для тестирования. Теперь для продуктивной работы вам нужно настроить несколько сетей, доступ ко входу и выходу из которых можно полностью контролировать.

Если у вас есть несколько ненужных компьютеров, несколько кабелей и пара коммутаторов, то отлично — закатайте рукава и сделайте это. Я, например, очень скучаю по экспериментам с серверным и сетевым оборудованием. Но что я могу

сделать? Я теперь практически полностью в виртуальном мире: я даже не помню, когда в последний раз запускал свою рабочую станцию (рабочий компьютер, предназначенный для настройки сети).

Если, как и я, вы застряли в сетях, которые можете построить для себя прямо из командной строки, вот что лучше всего делать. Воспользуйтесь VirtualBox для создания пары виртуальных сетей NAT, предоставления сетевых интерфейсов для каждой виртуальной машины, которую вы запускаете, и вручную свяжите интерфейсы с соответствующими сетями.

Помните, что количество виртуальных машин VirtualBox, которые вы сможете запустить, ограничено объемом свободной физической памяти и мощностью процессора, которые у вас есть на хост-компьютере VirtualBox. С хостом Linux, работающим на 8 Гбайт ОЗУ, вы можете получить одновременно три виртуальные машины. И даже не думайте пробовать это на компьютере с Windows, если у вас не менее 16 Гбайт ОЗУ. Рассмотрите возможность разместить одну или две виртуальные машины на отдельном ПК в вашей сети.

Поначалу вы будете использовать инструмент командной строки `vboxmanage` (создание сетей для конфигурации DMZ). Насколько я вижу, это невозможно сделать из графического интерфейса VirtualBox. Команда `natnetwork add` сообщает VirtualBox, что вы хотите добавить новую сеть NAT. В первом примере сетевое имя будет dmz, а подсеть — 10.0.1.0/24. Вторая сеть будет называться loc (для локальной сети), а ее подсеть — 10.0.2.0/24. Когда сети будут созданы, вы запустите их командой `natnetwork start`:

```
$ vboxmanage natnetwork add --netname dmz \
 --network "10.0.1.0/24" --enable --dhcp on
$ vboxmanage natnetwork add --netname loc \
 --network "10.0.2.0/24" --enable --dhcp on
$ vboxmanage natnetwork start --netname dmz
$ vboxmanage natnetwork start --netname loc
```

Поскольку это только виртуальные сети, для их создания не требуются права администратора

ПРИМЕЧАНИЕ

Из-за известной ошибки может быть невозможно создать и запустить сети NAT в версиях VirtualBox до 5.1.

Следующим шагом будет создание (или лучше клонирование) виртуальных машин, которые вам понадобятся. (Убедитесь, что у вас достаточно системной памяти и мощности ЦП на хост-компьютере для работы всех виртуальных машин, которые вы планируете запускать.) Прежде чем запускать виртуальные машины, щелкните на первом имени в списке VirtualBox, затем нажмите кнопку **Settings** (Настройки), а затем выберите раздел **Network** (Сеть) с левой стороны. Если эта виртуальная машина будет, скажем, сервером Shorewall, то подключите ее первый сетевой адаптер к любому интерфейсу, который обычно используете для предоставления вашим виртуальным машинам полного доступа в Интернет.

Теперь перейдите на вкладку **Adapter 2** (Адаптер 2), а затем в раскрывающемся списке **Attached to** (Присоединено к) выберите настройку сети NAT. Поскольку вы

уже создали пару сетей NAT, обе должны быть доступны в качестве параметров (рис. 10.5). Выберите один из них для адаптера 2, а затем другой для адаптера 3.

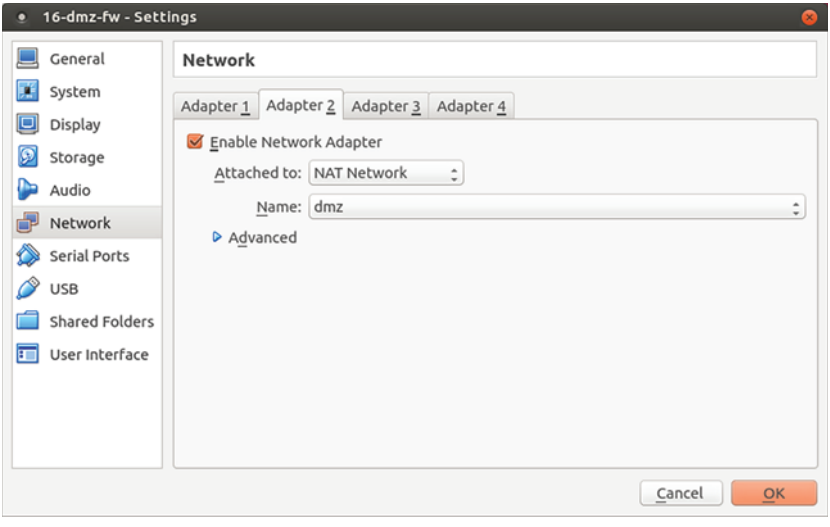


Рис. 10.5. Ассоциирование сетевого адаптера, подключенного к сети NAT, с виртуальной машиной VirtualBox

Чтобы закончить с тестовой средой Shorewall, проработайте раздел **Settings ▸ Network** (Настройки ▸ Сеть) ваших остальных двух машин и привяжите одну к NAT-сети dmz, а другую — к loc. Эти две виртуальные машины имеют только по одному интерфейсу.

После этого вы можете запустить три виртуальные машины и войти в систему. Запустите `ip addr` на каждой, чтобы убедиться, что интерфейсы были распознаны и подключены к своим сетям. Если вы видите правильный вариант IP-адреса в строке inet (например, 10.0.1.5/24), то все работает:

```
$ ip addr
2: enp0s3:: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc [...]
    inet 10.0.1.5/24 brd 10.0.1.255/ scope global enp0s3
       valid_lft forever preferred_lft forever
```

Этот интерфейс правильно подключен к вашей сети dmz NAT

Но если нет inet-адреса, вам нужно присвоить его вручную. Возьмите имя интерфейса (в примере enp0s3) и используйте его в качестве аргумента для `ifconfig up`. Это скажет интерфейсу, что ему пора проснуться и приступить к работе. Команда `dhclient` запросит IP-адрес в сети dmz с DHCP-сервера. (Помните, как вы установили `--dhcp` при создании сети NAT? На то была причина.)

```
# ifconfig enp0s3 up
# dhclient enp0s3
```

Если эта операция завершится неудачно, возможно, что-то не так с вашим сетевым оборудованием или конфигурацией

Наконец, запустите `ip addr` еще раз, чтобы убедиться, что все в порядке. У вас есть тестовая среда. Если вы не уверены, что достаточно хорошо разбираетесь в подсетях и сетях NAT, еще не все потеряно: мы поговорим об этом в главе 14.

Резюме

- VPN-туннели могут использоваться для защиты удаленных соединений, а также для безопасного доступа к сетевой инфраструктуре между удаленными точками.
- Пакет Easy RSA содержит сценарии, которые можно применять с целью создания полной инфраструктуры открытых ключей (PKI) для приложений шифрования TLS.
- Брандмауэры и сетевая архитектура могут использоваться совместно для создания защищенных сред подсетей для ресурсов, которые не должны быть доступны в общественных сетях.
- В Shorewall Firewall предусмотрены различные уровни сложности и все правила хранятся в простых текстовых файлах, тогда как реализацией межсетевого экрана iptables можно управлять из командной строки.
- Структура сети DMZ использует конфигурации брандмауэра для размещения уязвимых ресурсов в защищенных частных подсетях, а общедоступных серверов — в отдельных подсетях, что упрощает удаленный доступ.
- iptables и nftables — гибкие средства командной строки для управления работой брандмауэра netfilters для ядер Linux.

Ключевые термины

- *Виртуальная частная сеть (VPN)* — метод подключения удаленных сетей и безопасного разделения ресурсов между ними.
- *VPN-туннель* описывает, как сетевые протоколы могут использоваться для маскировки VPN-трафика при его прохождении по незащищенным сетям.
- *VPN-клиент* — это устройство, подключающееся к VPN-серверу или через него. В случае OpenVPN клиент может быть смартфоном или ПК, на котором запущено клиентское приложение OpenVPN GUI, или компьютером с Linux, настроенным для использования OpenVPN.
- *Анализатор пакетов*, такой как WireShark, может подслушивать и анализировать пакеты данных при их перемещении по сети, показывая информацию об их содержимом, происхождении и назначении.
- *Сеть NAT* предоставляет один внешний IP-адрес общедоступной сети, одновременно управляя трафиком между частными устройствами. Будет более подробно обсуждаться в главе 14.

Обзор команд

- `hostname OpenVPN-Server` задает приглашение командной строки, чтобы упростить отслеживание того, на какой сервер вы зашли.
- `cp -r /usr/share/easy-rsa/ /etc/openvpn` копирует сценарии Easy RSA и файлы конфигурации среды в рабочий каталог OpenVPN.
- `./build-key-server server` генерирует пару ключей RSA с именем `server`.
- `./pkitool client` генерирует клиентский набор ключей из существующей инфраструктуры ключей RSA.
- `openvpn --tls-client --config /etc/openvpn/client.conf` запускает OpenVPN на клиенте Linux, используя настройки из файла `client.conf`.
- `iptables -A FORWARD -i eth1 -o eth2 -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT` позволяет передавать данные между сетевыми интерфейсами eth1 и eth2.
- `man shorewall-rules` отображает документацию по файлу правил, используемому Shorewall.
- `systemctl start shorewall` запускает брандмауэр Shorewall.
- `vboxmanage natnetwork add --netname dmz --network "10.0.1.0/24" -enable --dhcp on` использует CLI VirtualBox с целью создания и настройки виртуальной сети NAT с DHCP для виртуальных машин VirtualBox.
- `vboxmanage natnetwork start --netname dmz` запускает виртуальную сеть NAT.
- `dhclient enp0s3` запрашивает IP-адрес для интерфейса enp0s3 у сервера DHCP.

Самотестирование

1. Правильно настроенный туннель OpenVPN может улучшить безопасность:
   а) применением правил брандмауэра для управления доступом между сетями;
   б) изоляцией сетевых устройств с помощью подсетей;
   в) добавлением шифрования к сетевым подключениям;
   г) затенением трафика, проходящего через общедоступную сеть.
2. Чтобы включить внутреннюю маршрутизацию на сервере, какую строку в файле `/etc/sysctl.conf` необходимо раскомментировать:
   а) `net.ipv4.ip_forward=1`;
   б) `net.ipv4.tcp_syncookies=1`;
   в) `net.ipv6.conf.all.accept_redirects = 0`;
   г) `net.ipv4.conf.all.accept_source_route = 0`?
3. Установив easy-rsa, где вы найдете сценарии, чтобы сгенерировать ключи:
   а) `/usr/share/easy-rsa/`;
   б) `/usr/share/easy-rsa/scripts/`;

в) `/usr/share/easy-rsa/examples/`;

г) `/usr/share/docs/easy-rsa/`?

4. Какой из следующих сценариев будет выполнять бóльшую часть работы по созданию ценариев RSA:

   а) `vars`;

   б) `build-key-server`;

   в) `build.ca`;

   г) `pkitool`?

5. Установив OpenVPN, где вы найдете шаблоны файлов конфигурации:

   а) `/usr/share/doc/openvpn/examples/sample-config-files/server.conf/`;

   б) `/usr/share/doc/openvpn/sample-config-files/server.conf.gz`;

   в) `/usr/share/doc/openvpn/examples/sample-config-files/server.conf.gz`;

   г) `/usr/share/openvpn/examples/sample-config-files/server/`?

6. Какие из следующих значений можно добавить в файл `/etc/openvpn/server.conf` для перенаправления клиентов на веб-сервер:

   а) `port-share localhost 80`;

   б) `proto tcp`;

   в) `client-to-client`;

   г) `push "route 10.0.3.0 255.255.255.0"`?

7. Какая из следующих команд `iptables` будет незаметно блокировать весь трафик, отправляемый на интерфейс:

   а) `iptables -P OUTPUT DROP`;

   б) `iptables -P INPUT DROP`;

   в) `iptables -P INPUT REJECT`;

   г) `iptables -P FORWARD DROP`?

8. Какой из следующих файлов Shorewall используется для установки профиля доступа по умолчанию для вашего брандмауэра:

   а) `params`;

   б) `interfaces`;

   в) `rules`;

   г) `policy`?

Ответы

1 — в; 2 — а; 3 — а; 4 — г; 5 — в; 6 — а; 7 — б; 8 — г.

11 Мониторинг системы: работа с файлами журналов

В этой главе

- Фильтрация журнальных записей для поддержания работоспособности системы.
- Обслуживание системы журналирования Linux.
- Фильтрация текстовых потоков с помощью инструментов grep, awk и sed.
- Применение систем обнаружения вторжений.

Если бы в своей работе вам приходилось делать только то, о чем вы узнали из этой книги, я бы сказал, что вы готовы создать довольно хороший сервер. Он будет подключен, все будет автоматизировано, будет выполняться резервное копирование, сервер будет открыт для удаленных клиентов, запрашивающих данные или сервисы, и по крайней мере все будет достаточно безопасно. Все удобства.

Можно расслабиться? Пока нет. Ваш сервер может быть правильно настроен, но вам также нужно следить за тем, как он справляется с работой, когда вводится в производственную среду. Как это осуществить? Как вы скоро увидите, системный мониторинг Linux в основном представляет собой анализ журнальных файлов.

Запись в журнале — это текстовая запись какого-либо системного события. Когда пользователь вводит учетные данные аутентификации, удаленный клиент запрашивает данные с веб-сервера, происходит сбой приложения или подключается новое аппаратное устройство, в один или несколько журнальных файлов добавляется запись.

Даже нечасто используемая система в промежутке между загрузкой и завершением работы может генерировать тысячи строк в файлах журналов, а загруженные приложения могут легко создавать миллионы строк в день. Поскольку файлы журналов имеют тенденцию быть длинными и скучными, вы, вероятно, захотите передать их чтение программе, которая способна фильтровать только срочные записи, такие как предупреждения о неизбежных сбоях, и уведомлять вас лишь в случае крайней необходимости. Чем тщательнее вы будете настраивать поведение журналов вашей системы и управлять постоянно увеличивающимися файлами журналов, тем лучше поймете сильные и слабые стороны вашей системы — и тем более надежной она станет.

Для тех, кто будет их читать, файлы журналов — сундук с сокровищами в виде ценных знаний. Эти файлы могут рассказать вам о недостатках вашей защиты и несанкционированных вторжениях. Записи журнала могут помочь вам предвидеть проблемы с безопасностью и производительностью вашей системы и диагностировать их, когда уже все накрылось.

Можно сказать, что в этой главе перечислены ресурсы, находящиеся в ваших журналах, и описаны наилучшие подходы к настройке и использованию журналов, а также управлению ими. Вы также узнаете об инструментах обнаружения вторжений, которые можно настроить для регулярного сканирования вашего сервера и сетевых сред в поисках явных признаков подозрительной активности.

Но на самом деле глава посвящена защите вашей системы от взломов и провалов в производительности. Если вы администратор, ответственный за критически важный общедоступный сервер, то эта глава для вас.

11.1. Работа с системными журналами

В течение десятилетий за ведение журналов в Linux отвечал демон *syslogd*. Он собирает сообщения, которые системные процессы и приложения отправляют на псевдоустройство `/dev/log`. Затем он направляет их в соответствующие текстовые файлы журнала в каталоге `/var/log/`. syslogd знает, куда отправлять сообщения, поскольку каждое из них включает заголовки, содержащие поля метаданных (включая метку времени, а также источник и приоритет сообщения) (рис. 11.1).

После безжалостного и безоговорочного завоевания мира программой systemd ведение журналов в Linux теперь также ведется с помощью journald. Я говорю *«также»*, потому что syslogd никуда не делся и вы все еще можете найти большинство его традиционных файлов журналов в каталоге `/var/log/`. Но вы должны знать, что в городе есть новый шериф, которого (в командной строке) зовут *journalctl*.

Переход на journald был спорным. Никто не будет отрицать ценность новой функциональности journald (рис. 11.2). Вскоре вы сами увидите, что в нем сочетаются мощная фильтрация и поиск. Но тот факт, что journald хранит данные журнала в бинарном виде, а не в виде простого текста, определенно является проблемой.

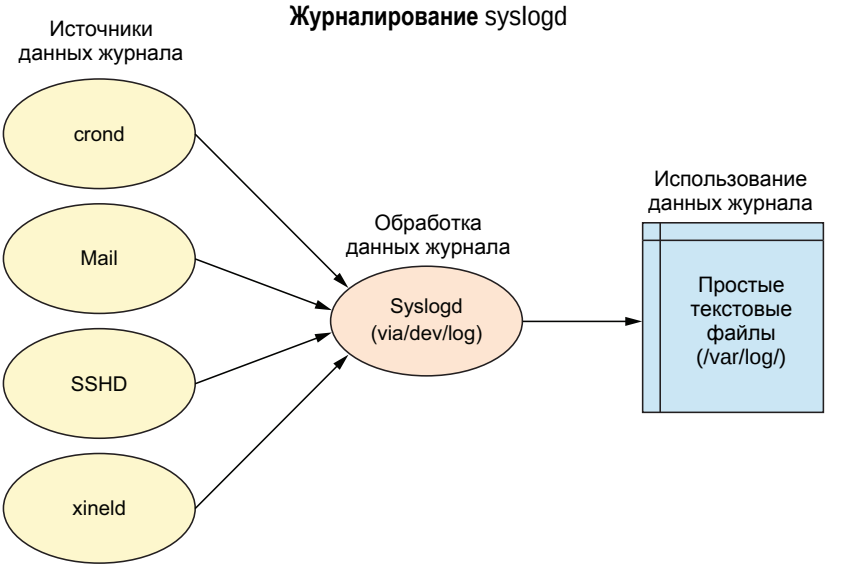


Рис. 11.1. Поток журнальных данных из возможных источников через демон syslogd

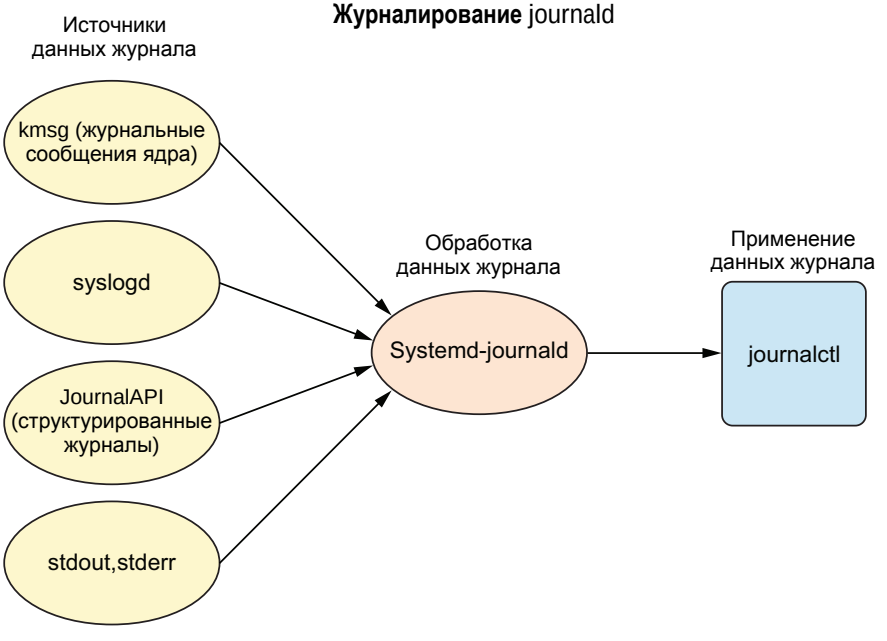


Рис. 11.2. Журналы journald (в том числе созданные syslogd) используются с помощью инструмента командной строки journalctl

На самом деле, поскольку журналы бинарные, иногда может быть трудно или даже невозможно получить к ним доступ. Подумайте об этом: когда journalctl вам будет больше всего нужен (например, вы восстанавливаетесь после сбоя системы), он, вероятно, не будет работать. Журналы syslogd, с другой стороны, всегда будут вам доступны, пока вы можете примонтировать диск, на котором они находятся. Это может быть очень удобно, когда вам потребуется узнать, что сломалось в вашей системе и что нужно сделать, чтобы восстановить ее. Хорошей новостью является то, что в ближайшем будущем оба подхода будут продолжать сосуществовать.

11.1.1. Журналирование с помощью journald

На прошлой неделе один взволнованный разработчик принес мне свой ноутбук с Ubuntu, жалуясь, что тот умирает. Загрузка фактически остановилась, и на черном экране появилось единственное сообщение, которое выглядело примерно так:

```
/dev/sda1: clean, 127663/900212 files, 709879/3619856 blocks
```

Тот факт, что я получил ссылку на `/dev/sda1`, говорил о том, что жесткий диск был жив, процесс загрузки прошел GRUB и Linux была по крайней мере частично загружена. Фактически это конкретное сообщение вводило в заблуждение, поскольку описывало последний успешный этап процесса загрузки перед остановкой, и не имело никакого отношения к самой проблеме.

После некоторого количества проб и ошибок, включая загрузку ноутбука с USB-накопителя в режиме реального времени и установку диска ноутбука (как вы видели в главе 6), оказалось, что Linux полностью загружена, но не удалось запустить графический интерфейс — Рабочий стол. Как я узнал это? Вернувшись к неудачной загрузке (с отображением `/dev/sda1:...` на экране), я нажал сочетание клавиш Alt+F1 и был перенаправлен в виртуальную консоль. Оттуда у меня был полный доступ к командной строке.

СОВЕТ

> Оказывается, вы можете открыть несколько виртуальных консолей, нажав сочетание клавиш Alt+F2, Alt+F3 и т. д. Вы также можете переключаться между ними с помощью сочетания Alt+<соответствующая F-клавиша>. Имейте в виду, что к хост-оболочке (обычно к сеансу GUI) можно вернуться, нажав Alt+F7.

Почему графический интерфейс Рабочего стола не загружается? И почему я вообще рассказываю вам обо всем этом? Ответ на оба вопроса, как вы уже догадались, можно найти в журналах ноутбука.

Чтобы понять все это, совершим небольшой обход и ближе познакомимся с journalctl, а затем вернемся к этому бедному ноутбуку. Вызов команды `journalctl` сам по себе выводит самые старые записи журнала, которые в настоящее время

находятся в системе. Однако в первой строке отображается дата начала и окончания доступных записей. Далее говорится о почти девяти месяцах записей в журнале:

```
# journalctl
-- Logs begin at Thu 2016-12-15 08:46:17 EST,
➥ end at Mon 2017-09-04 21:14:54 EDT. --
```

Поскольку вас, скорее всего, интересует более свежая информация, воспользуйтесь аргументом `-n 20` — он будет отображать только последние 20 записей:

```
# journalctl -n 20
```

Полезно проверять ваши журналы время от времени. Я выполнил эту команду на своей машине, чтобы убедиться, что она работает, как ожидалось, и обнаружил потерянного клиента OpenVPN. Благослови его Господь. Каждые пять секунд клиент добросовестно пытался подключиться к VPN-серверу, которого, к сожалению, больше не существует, а затем отправлял новые записи, чтобы сообщить мне об этом. Для решения этой проблемы я использовал systemctl, чтобы сначала остановить, а затем отключить службу OpenVPN.

Как я писал ранее, чем больше вы можете сузить результаты, тем быстрее получите информацию, которую ищете. Один из полезных способов фильтрации журналов — отображение результатов по приоритету. Например, добавление аргумента `-p emerg` выводит только те записи журнала, которые относятся к категории чрезвычайных ситуаций. Если вы знаете, что в вашей системе что-то сломалось, но не можете изолировать проблему, лучше всего сначала указать `emerg`:

```
# journalctl -p emerg
```

Кроме того, вы можете фильтровать сообщения отладки, информационные сообщения, уведомления, предупреждения, сведения об ошибках, критические замечания и предупреждения (см. уровни приоритетов syslogd в следующем разделе). При добавлении флага `-f` (для отслеживания) отображаются десять самых последних записей и любые последующие записи по мере их создания. Это позволяет вам наблюдать за событиями в реальном времени по мере их возникновения:

```
# journalctl -f
```

Вы также можете фильтровать записи журналов по дате и времени. И вот здесь мы возвращаемся к ноутбуку без графического интерфейса. Если вы знаете, что процесс загрузки остановился, то можете сузить область поиска, чтобы она возвращала только события из этого периода времени. К счастью, аргументы `--since` и `--until` принимают даты и время. Если вы не укажете дату, то получите самое последнее время, отвечающее этим критериям.

В моем случае я мог бы указать двухминутный интервал, в течение которого, как я полагаю, произошел сбой. За две минуты, особенно во время загрузки системы, можно внести еще много записей в журнал, но такое количество будет намного легче просматривать, чем записанное за 20 минут:

```
# journalctl --since 15:50:00 --until 15:52:00
```

Если бы я выполнил эту команду на ноутбуке, я бы сэкономил себе время. Однако так получилось, что вместо этого я использовал старые файлы системного журнала. Посмотрим, как это работает.

11.1.2. Журналирование с помощью syslogd

Все записи в журналы, генерируемые событиями в системе syslogd, добавляются в файл `/var/log/syslog`. Но в зависимости от их характеристик они также могут быть отправлены в один или несколько других файлов в том же каталоге.

При использовании syslogd способ распределения сообщений определяется содержимым файла `50-default.conf`, который находится в каталоге `/etc/rsyslog.d/`. Во фрагменте файла `50-default.conf` ниже показано, как сообщения журнала, помеченные как связанные с `cron`, будут записываться в файл `cron.log`. В этом случае звездочка (`*`) указывает syslogd отправлять записи с любым уровнем приоритета (в отличие от конкретного уровня наподобие `emerg` или `err`):

```
cron.* /var/log/cron.log
```

Для работы с файлами журналов syslogd не требуется никаких специальных инструментов, таких как journalctl. Но желательно знать, какая информация хранится в каждом из стандартных файлов журнала. В табл. 11.1 перечислены наиболее распространенные файлы журналов syslogd и их назначение. (Чтобы познакомиться с остальными, посмотрите их справочные страницы. Например, по команде `man lastlog`.)

Таблица 11.1. Популярные средства syslogd

Имя файла	Назначение
auth.log	Системная аутентификация и события безопасности
boot.log	Запись событий, связанных с загрузкой
dmesg	События уровня ядра, связанные с драйверами устройств
dpkg.log	События управления пакетами программ
kern.log	События ядра Linux
syslog	Сборка из всех журналов
wtmp	Отслеживание сеансов пользователей (можно применять команды who и last)

Кроме того, отдельные приложения будут иногда записывать данные в свои файлы журналов. Вы также сможете увидеть целые каталоги, такие как `/var/log/apache2/` или `/var/log/mysql/`, созданные для сообщений от приложения. Для журнала можно указывать любой из восьми уровней приоритета, в дополнение к символу `*`, который вы видели ранее. В табл. 11.2 перечислены уровни приоритетов syslogd.

Таблица 11.2. Уровни приоритета syslogd

Уровень	Описание
debug	Полезно для отладки
info	Информационный
notice	Нормальное состояние
warn	Предупреждение
err	Ошибка
crit	Критический уровень
alert	Требуется немедленное вмешательство
emerg	Систему нельзя использовать

Итак, что же я нашел на том ноутбуке? Ну, это не совсем относится к данной главе, но я все равно расскажу. Следующая строка появилась именно в файле системного журнала:

```
xinit: unable to connect to X server: connection refused
```

Сервер X — система Linux, которая обрабатывает графические интерфейсы **Рабочего стола**. Если пользователь не смог подключиться, это означает одно из двух:

- система сервера X была повреждена;
- появилась какая-то проблема с аутентификацией.

Это был не первый случай, и к тому времени я уже успешно вошел в сеанс **Рабочего стола**, используя другую учетную запись. Но было трудно понять, как владелец ноутбука мог проходить аутентификацию в командной строке и все равно получить отказ в доступе к **Рабочему столу**. Тем не менее проблема, похоже, заслуживает дальнейшего рассмотрения, поэтому я открыл файл `auth.log`, и увидел следующее (вместо ***username*** здесь используется имя разработчика):

```
lightdm: pam_succeed_if(lightdm:auth): requirement
➥   "user ingroup nopasswdlogin" not met by user "username"
```

Я знаю, что lightdm — это менеджер **Рабочего стола**, используемый компьютерами с Ubuntu, а pam — модуль, который выполняет аутентификацию пользователей Linux. Но я признаю, что не понял всей значимости сообщения, пока не ввел его в строку поиска своей любимой поисковой системы в Интернете.

Оказалось, другие люди, столкнувшиеся с этим сообщением, обнаружили, что проблема связана с неправильными правами владения на файл `.Xauthority` из домашнего каталога пользователя. Очевидно, сервер X может загружать сеанс с графическим интерфейсом, только если файл `.Xauthority` принадлежит пользователю. Я проверил: файл `.Xauthority` этого пользователя на самом деле принадлежал

пользователю root. Устранить проблему было так же просто, как запустить chown, чтобы сменить владельца:

```
$ ls -al | grep Xauthority
-rw------- 1 root root 56 Sep 4 08:44 .Xauthority
# chown username:username .Xauthority
```

Как изначально возникла проблема с неправильным правом собственности? Кто знает? Но она появилась, а журнальные файлы помогли ее исправить.

11.2. Управление файлами журналов

Благодаря тому что сотни системных процессов ежечасно записывают в журнал тысячи сообщений, неуправляемая система журналов может быстро заполнить все доступное ей для хранения пространство. В этот момент запись в журнал прекратится, как и любые другие системные процессы, использующие то же пространство. Что делать?

11.2.1. Способ journald

journald решает эту проблему, автоматически ограничивая максимальное дисковое пространство, которое разрешено использовать системе journald. Как только предел достигнут, старые сообщения удаляются. Эта возможность контролируется параметрами SystemMaxUse= и RuntimeMaxUse= в файле /etc/systemd/journal.conf.

В чем разница между этими настройками? По умолчанию journald создает и поддерживает свой журнал в файле /run/log/journal, который является временным файлом и уничтожается при каждом завершении работы системы. Тем не менее вы можете настроить ведение журнала для сохранения файла постоянного журнала в /var/log/journal. Вы будете использовать тот или другой из этих двух параметров journal.conf в зависимости от настроек вашей системы. Преобразование в постоянный файл журнала требует только того, чтобы вы создали каталог /var/log/journal/ и ввели команду systemd-tmpfiles для правильного направления потока в журнал:

```
# mkdir -p /var/log/journal
# systemd-tmpfiles --create --prefix /var/log/journal
```

11.2.2. Способ syslogd

По умолчанию syslogd сама обрабатывает ротацию, сжатие и удаление логов без какой-либо помощи с вашей стороны. Но вы должны знать, как это делается, если у вас есть журналы, требующие особой настройки.

Какой вид специальной обработки может потребоваться? Предположим, ваша компания должна соответствовать правилам отчетности о транзакциях, связанным с нормативными или отраслевыми требованиями, такими как Sarbanes-Oxley или

PCI-DSS. Если журналы вашей IT-инфраструктуры должны оставаться доступными в течение длительных периодов времени, тогда вы определенно захотите узнать, как задействовать ключевые файлы.

Чтобы увидеть систему logrotate в действии, просмотрите часть содержимого каталога `/var/log/`. Например, файл `auth.log` представлен в трех разных форматах.

- `auth.log` — текущая активная версия, в которую записываются новые сообщения авторизации.
- `auth.log.1` — самый последний файл, который уже отключен. Он поддерживается в несжатом формате, чтобы можно было быстро его просмотреть при необходимости.
- `auth.log.2.gz` — более старая версия (как вы можете видеть из расширения файла `.gz` в листинге 11.1), которая из-за меньшей вероятности использования была сжата для экономии места.

Листинг 11.1. Содержимое каталога /var/log/

```
$ ls /var/log | grep auth
auth.log
auth.log.1
auth.log.2.gz
auth.log.3.gz
auth.log.4.gz
```

Текущая активная версия auth.log

Последняя вышедшая из оборота версия

Последующие версии сжимаются с помощью gzip

Когда через семь дней наступит следующая дата ротации, `auth.log.2.gz` будет переименован в `auth.log.3.gz`, `auth.log.1` будет сжат и переименован в `auth.log.2.gz`, `auth.log` станет `auth.log.1` и будет создан новый файл с именем `auth.log`. Цикл ротации журнала по умолчанию задан в файле `/etc/logrotate.conf`. Значения в этом файле задают действия по отношению к файлам по прошествии одной недели и указывают удалять старые файлы через четыре недели (листинг 11.2).

Листинг 11.2. Некоторые общие настройки из файла /etc/logrotate.conf

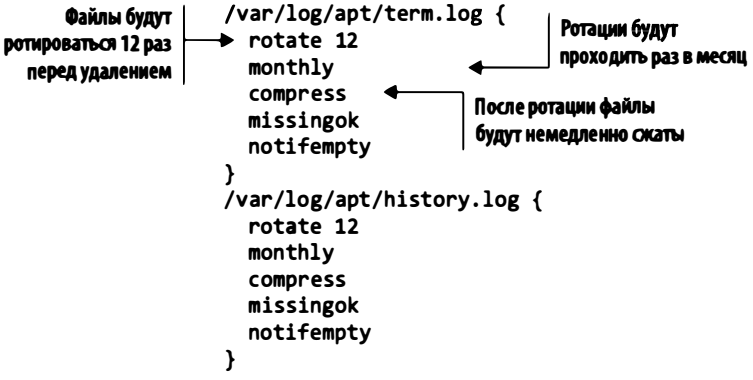
```
# ротация файлов журнала еженедельно
weekly
# хранить журналы за 4 недели
rotate 4
# создавать новые (пустые) файлы журнала после ротации старых
create
# пакеты добавляют информацию о ротации журнала в этот каталог
include /etc/logrotate.d
```

Каталог `/etc/logrotate.d/` также содержит настраиваемые файлы конфигурации для управления ротацией журналов отдельных сервисов или приложений. Посмотрим на файлы конфигурации из этого каталога:

```
$ ls /etc/logrotate.d/
apache2 apt dpkg mysql-server rsyslog samba unattended-upgrade
```

Вот как выглядит файл конфигурации apt в моей системе (листинг 11.3).

Листинг 11.3. Содержимое файла конфигурации /etc/logrotate.d/apt




```
/var/log/apt/term.log {
  rotate 12
  monthly
  compress
  missingok
  notifempty
}
/var/log/apt/history.log {
  rotate 12
  monthly
  compress
  missingok
  notifempty
}
```

СОВЕТ

Многие администраторы предпочитают перенаправлять записи журнала на специально созданные удаленные серверы журналов, где данные могут быть обработаны индивидуально, в зависимости от потребностей. Это позволяет освободить серверы приложений для их непосредственных задач и консолидировать данные журналов в едином легкодоступном централизованном месте.

11.3. Обработка больших файлов

Ни один человек не захочет тратить время на чтение миллионов строк записей в журнале. В следующих подразделах описываются три инструмента обработки текста, которые могут сделать это за вас — быстрее и лучше.

11.3.1. Использование grep

По моему опыту, самый универсальный и простой инструмент для фильтрации текста — наш старый друг grep. Вот очевидный пример, который будет искать в файле auth.log доказательства неудачных попыток входа в систему. Поиск слова *failure* вернет любую строку, содержащую фразу *authentication failure*. Проверяя это время от времени, вы можете обнаружить информацию о попытках взломать учетную запись. Любой пользователь может неправильно ввести пароль один или два раза, но слишком большое количество неудачных попыток должно насторожить вас:

```
$ cat /var/log/auth.log | grep 'Authentication failure'
Sep 6 09:22:21 workstation su[21153]: pam_authenticate: Authentication failure
```

Если вы из тех администраторов, которые никогда не ошибаются, то такой поиск может ничего не выдать. Но вы можете гарантировать себе хотя бы один результат,

вручную создав запись в журнале с помощью программы под названием logger. Попробуйте ввести что-то вроде следующего:

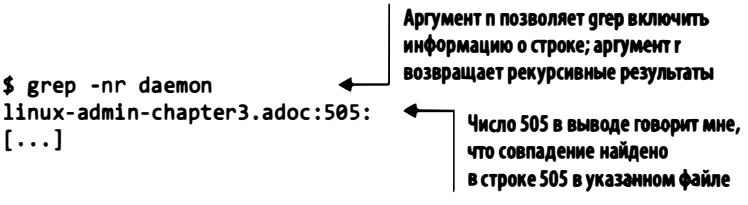
```
logger "Authentication failure"
```

Вы также можете заранее вызвать настоящую ошибку, войдя в учетную запись пользователя и введя неправильный пароль.

Да, grep сделал всю работу за вас, но по результатам вы можете видеть только то, что произошла ошибка аутентификации. Было бы полезно знать, о какой учетной записи идет речь. Вы можете расширить результаты grep, указав, чтобы они включали строки непосредственно до и после совпадения. Следующий пример выводит совпадение вместе со строками до и после него. Здесь видно, что кто-то использующий учетную запись david безуспешно пытался ввести команду su (переключение пользователя) для входа в учетную запись studio:

```
$ cat /var/log/auth.log | grep -B 1 -A 1 failure
Sep 6 09:22:19 workstation su[21153]: pam_unix(su:auth): authentication
   failure; logname= uid=1000 euid=0 tty=/dev/pts/4 ruser=david rhost=
   user=studio
Sep 6 09:22:21 workstation su[21153]: pam_authenticate:
   Authentication failure
Sep 6 09:22:21 workstation su[21153]: FAILED su for studio by david
```

Кстати, вы также можете использовать grep для поиска по нескольким файлам, что лично мне пригодилось при написании этой книги. Моя проблема заключается в том, что каждая глава сохраняется как отдельный файл в своем собственном каталоге, поэтому поиск, скажем, каждой ссылки на слово *daemon* может быть утомительным. Поскольку копии файлов глав в виде простого текста также сохраняются в одном каталоге, я могу перейти в этот каталог и искать по команде `grep -nr`. Вот, например, в главе 3:



11.3.2. Использование awk

Инструмент grep может многое сделать, но он не всемогущ. Возьмем эти записи из файла `/var/log/mysql/error.log` в качестве примера (листинг 11.4).

Листинг 11.4. Выборка записей из файла /var/log/mysql/error.log

```
2017-09-05T04:42:52.293846Z 0 [Note] Shutting down plugin 'sha256_password'
2017-09-05T04:42:52.293852Z 0 [Note]
➥ Shutting down plugin 'mysql_native_password'
2017-09-05T04:42:52.294032Z 0 [Note] Shutting down plugin 'binlog'
```

```
2017-09-05T04:42:52.294647Z 0 [Note] /usr/sbin/mysqld: Shutdown complete
2017-09-05T12:24:23.819058Z 0 [Warning] Changed limits:
➥ max_open_files: 1024 (requested 5000)
2017-09-05T12:24:23.819193Z 0 [Warning] Changed limits:
➥ table_open_cache: 431 (requested 2000)
```

Как видите, записи классифицируются по уровню приоритета, заключенному в скобки (например, `[Warning]`). Предположим, вы хотели бы узнать, сколько предупреждений было с момента последней ротации журнала. Возможно, вы регулярно с определенной частотой сравниваете цифры, чтобы узнать, когда у вас может возникнуть проблема, требующая вмешательства. Вы можете использовать grep каждый раз, когда появляется `[Warning]`, а затем направлять его в программу wc, которая будет считать строки, слова и символы в выводе:

```
# cat error.log | grep [Warning] | wc
   4219   37292   360409
```

Казалось бы, предупреждение `[Warning]` найдено 4219 раз. Кто-то пытается привлечь ваше внимание. Но попробуйте то же самое с помощью инструмента awk. Программа wc вернет только 204 строки! В чем дело?

```
# cat error.log | awk '$3 ~/[Warning]/' | wc
    204    3213   27846
```

Кажется, что grep использует скобки, чтобы обозначить список символов, и ищет в текстовом потоке любой из них. Поскольку в слове *Warning* есть шесть уникальных букв, подходит почти каждая строка в файле журнала. Вы всегда можете убрать квадратные скобки и искать именно слово *Warning*, но встречаются ситуации, когда при этом возвращается слишком много ложных совпадений.

Пока строка поиска обрамлена слешами (`/[Warning]/`), у awk не будет проблем со скобками. А сложный синтаксис awk имеет свои преимущества. Возьмем, к примеру, `$3` в предыдущем примере. По умолчанию awk разбивает строку текста на отдельные поля, каждое из которых разделено одним или несколькими пробелами. В этой записи журнала, например, семь полей:

```
2017-09-05T04:42:52.294032Z 0 [Note] Shutting down plugin binlog
```

Инструмент grep интерпретирует последние четыре слова (`Shutting down plugin binlog`) как четыре отдельных поля данных, поэтому сложно найти способ их поиска. Но третье поле определяет уровень приоритета записи, поэтому вы можете запустить awk с `$3`, чтобы вернуть только совпадающие результаты из этого поля.

11.3.3. Использование sed

Если использование wc для возврата количества строк в потоке вам не подходит, вы всегда можете обратиться к инструменту sed, который напечатает номер текущей строки (`-n` говорит команде `sed` не печатать сам текст):

```
# cat error.log | awk '$3 ~/[Warning]/' | sed -n '$='
204
```

Хорошо, но вам вряд ли когда-нибудь понадобится использовать такой крупный редактор потоков, как sed, только для того, чтобы сделать что-то простое, например посчитать строки. Я упомянул его только в качестве примера, чтобы познакомить вас с ним и, в частности, с его способностью делать сложные замены в потоках текста.

Вот простой пример, где sed получает текст *hello world* и велит заменить первый экземпляр слова *world* словом *fishtank*:

```
$ echo "hello world" | sed "s/world/fishtank/"
hello fishtank
```

Добавив g перед последней кавычкой, вы скажете sed заменить каждый экземпляр *world*, предполагая, что их больше одного.

Это хорошо, но не очень полезно. Где sed действительно пригодится, так это в редактировании текста. Возможно, сценарий Bash, над которым вы работаете, должен обрабатывать только определенную часть входящих данных. Или, возможно, вы редактируете текст, чтобы он был более читабельным для конечного пользователя.

Допустим, у вас есть файл numbers.txt, содержащий код с номерами строк. Вы должны удалить номера строк, не трогая сам код, строки которого сами по себе могут содержать номера. Вот текст, который вы можете использовать (скопируйте его в файл на своем компьютере, чтобы изучить) (листинг 11.5).

Листинг 11.5. Пример фрагмента кода с номерами нежелательных строк

```
4 <menuitem action='Item 1'/>
5 <menuitem action='Item 2'/>
6 <menuitem action='Item 3'/>
7 <menuitem action='Exit'/>
8   <separator/><
```

Теперь передайте файл numbers.txt в sed: используйте символ вставки (^), чтобы направить sed на начало каждой строки. Укажите любое число, и оно будет удалено при первом вхождении. Обратите внимание, как исчезли номера строк, но номера пунктов все еще остались:

```
$ sed "s/^ *[0-9]* //g" numbers.txt
<menuitem action='Item 1'/>
<menuitem action='Item 2'/>
<menuitem action='Item 3'/>
<menuitem action='Exit'/>
  <separator/><
```

Как вариант, вы можете перенаправить вывод в новый файл:

```
$ sed "s/^ *[0-9]* //" numbers.txt > new-numbers.txt
```

Наконец, sed может выборочно печатать только подкаталоги (а не отдельные файлы) из списка каталогов. Вот как выглядел бы нефильтрованный список:

```
$ ls -l
total 28
drwxrwxr-x 2 dbclinton dbclinton 4096 Sep 7 14:15 code
```

```
-rw-rw-r-- 1 dbclinton dbclinton   55 Sep 6 11:53 file
-rw-rw-r-- 1 dbclinton dbclinton   76 Sep 6 11:54 file2
-rw-rw-r-- 1 dbclinton dbclinton 4163 Sep 6 00:02 mysql.log
-rw-rw-r-- 1 dbclinton dbclinton  137 Sep 7 13:58 numbers
drwxrwxr-x 2 dbclinton dbclinton 4096 Sep 7 14:15 old-files
```

И вот как можно использовать `sed` с `^d` для вывода (`p`) только тех строк, которые начинаются с `d` (что, как вы знаете, обозначает каталог):

```
$ ls -l | sed -n '/^d/ p'
drwxrwxr-x 2 dbclinton dbclinton 4096 Sep 7 14:15 code
drwxrwxr-x 2 dbclinton dbclinton 4096 Sep 7 14:15 old-files
```

11.4. Мониторинг с обнаружением вторжений

Помимо ведения журналов, существует еще один способ контроля состояния и благополучия вашей системы: обнаружение вторжений. Идея состоит в том, чтобы создать базовый профиль состояния вашей системы, а затем периодически сканировать систему в поисках тех изменений, которые указывают на вторжение или другие неприятности.

Один из вариантов — внедрение сетевой системы обнаружения вторжений (NIDS), которая опирается на программное обеспечение (такое как Snort, Nmap и Wireshark) для регулярного «прослушивания» сетевого окружения в поисках устройств, открытых портов и хостов, которых не должно быть. Но в этой книге я не собираюсь тратить время на NIDS (хотя мой курс «Сетевая безопасность Linux» посвящен Pluralsight).

Вы также можете создать систему обнаружения вторжений на базе хоста (HIDS), чтобы постоянно следить за своим сервером. Именно это мы и сделаем, используя пакет с открытым исходным кодом под названием *Tripwire*.

Tripwire сканирует ваш сервер и добавляет ключевые атрибуты важных системных файлов (например, размер файла) в свою собственную базу данных. Если какой-либо из этих файлов будет отредактирован или удален либо новые файлы будут добавлены в контролируемые каталоги, эти атрибуты изменятся. Когда вы позже велите Tripwire проверить систему, она сравнит текущие значения со значениями, хранящимися в базе данных, и сообщит о любых расхождениях.

ПРИМЕЧАНИЕ

> Компания Tripwire также предоставляет коммерческую версию, которая предлагает централизованное управление множеством установок, соблюдение политик, поддержку и совместимость с Windows. Данная версия недешевая, но для использования в масштабах предприятия может стоить своих денег.

Чтобы запустить Tripwire, сначала установите простой почтовый сервер для отправки отчетов по электронной почте любым администраторам, которых вы укажете. Вы сами установите и настроите Tripwire, а затем отредактируете и зашифруете

его политику и файлы конфигурации (`tw.cfg` и `tw.pol` соответственно). Наконец, вы внесете изменение в систему, чтобы увидеть, как оно будет описано в отчете по электронной почте.

11.4.1. Настройка почтового сервера

Создать базовый почтовый сервер Linux намного проще, чем вы думаете. Установите пакет postfix (и mailutils для Ubuntu). Во время установки (по крайней мере для Ubuntu) выберите сайт при появлении запроса на тип конфигурации почты и localhost.localdomain в качестве имени вашей системы. Вы также должны убедиться, что для параметра `inet_interfaces` в файле `/etc/postfix/main.cf` установлено значение `localhost`. Осталось только удостовериться, что порт 25 открыт для SMTP на любом запущенном вами брандмауэре, и перезапустить postfix:

```
# systemctl restart postfix
```

Это было неплохо, правда? Теперь у вас должен быть активный почтовый сервер, который может отправлять электронные письма удаленным получателям. Он не готов обрабатывать входящую почту, но вам это и не требуется для данного проекта.

Как узнать, что ваш почтовый сервер работает?

Отправьте письмо на локальный адрес (например, Стиву), используя `sendmail steve`. Вы увидите пустую строку, в которую, как ожидается, требуется ввести сообщение, после чего нажать Enter, напечатать одну точку и еще раз нажать Enter. Стив может прочитать свою электронную почту, набрав `mail` в командной строке.

11.4.2. Установка Tripwire

Установка проста, хотя между Ubuntu и CentOS есть небольшие различия. Я расскажу о них отдельно.

Ubuntu

Когда apt установит пакет Tripwire, вам будет предложено создать новые парольные фразы для двух наборов ключей подписи. При этом ваша парольная фраза ненадолго останется незашифрованной и доступной для всех, кто случайно войдет в систему (рис. 11.3).

После создания парольных фраз для обоих наборов ключей вас спросят, хотите ли вы перестроить зашифрованный файл конфигурации Tripwire (рис. 11.4). Перестройка файлов конфигурации и политик требуется после любых изменений их исходных текстовых файлов. Позже вы узнаете, как это сделать вручную, но в первый раз при установке Tripwire сделает работу за вас.

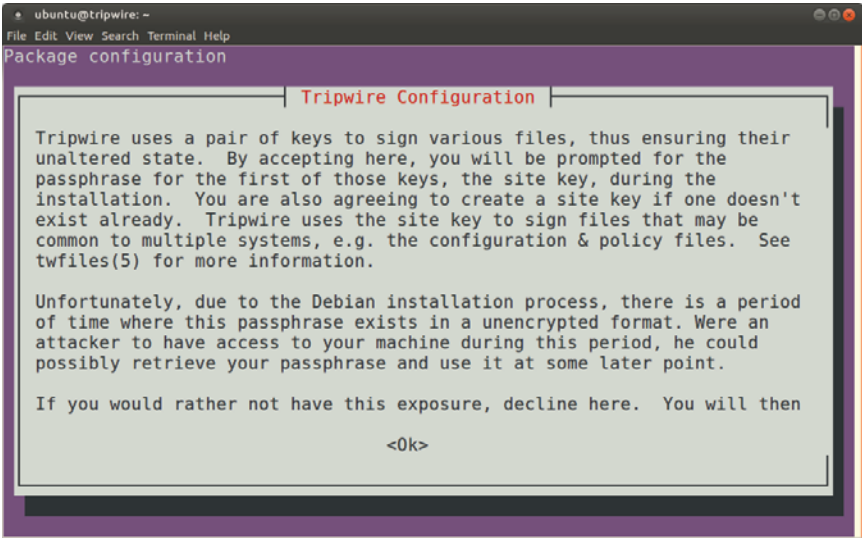


Рис. 11.3. Предупреждение в процессе установки Debian/Ubuntu для Tripwire

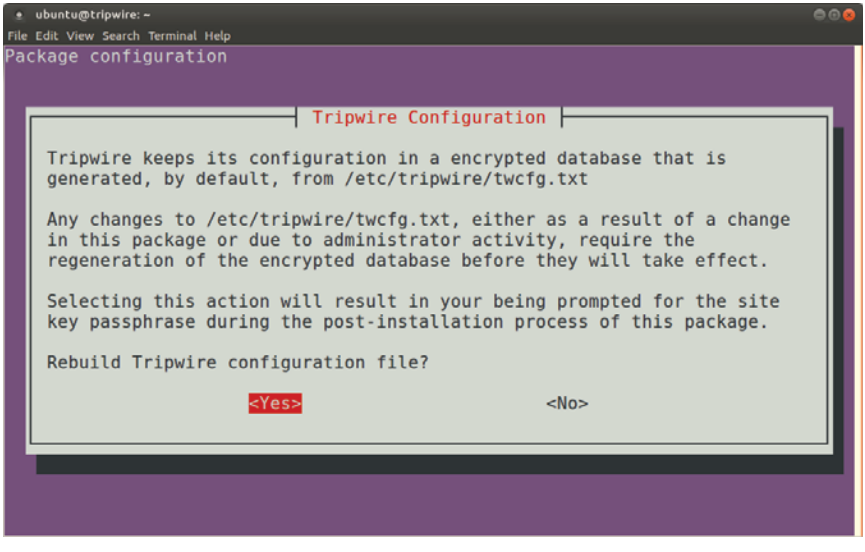


Рис. 11.4. Финальный экран перед тем, как Tripwire перестраивает зашифрованный файл конфигурации

После завершения установки вы увидите информацию о расположении ключевых файлов Tripwire (рис. 11.5). Запишите адреса и, в частности, обратите внимание на документацию в `/usr/share/`. Как и следовало ожидать, в каталоге `/etc/tripwire/` есть также файлы конфигурации.

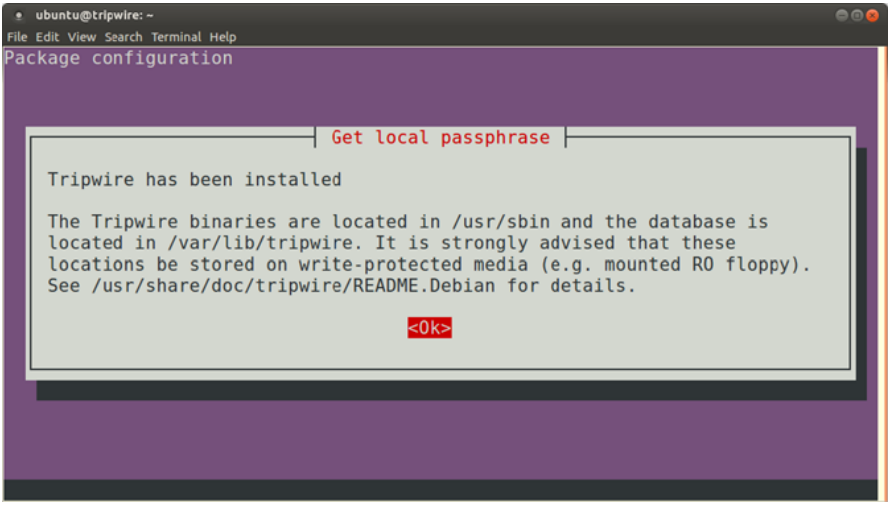


Рис. 11.5. Когда установка завершится, Tripwire скажет, где находятся самые важные файлы

CentOS

При установке в CentOS нет причудливой, ультрасовременной графики 1980-х годов, как при установке в Ubuntu. Фактически, если у вас уже есть установленный репозиторий epel-release (`install epel-release`), это вопрос установки пакета Tripwire и просмотра строк текста.

В CentOS ключи подписи создаются путем запуска программы tripwire-setup-keyfiles после основной установки. Там вам будет предложено создать две парольные фразы. После завершения установки (как для Ubuntu, так и для CentOS) вам потребуется запустить `tripwire --init` для инициализации базы данных:

```
[...]
Wrote database file: /var/lib/tripwire/tripwire.twd
The database was successfully generated.
```

Если вы настраиваете Tripwire в контейнере LXC, имейте в виду, что, вероятно, увидите много ошибок, подобных этим:

```
The object: "/proc/cpuinfo" is on a different file system...ignoring.
The object: "/proc/diskstats" is on a different file system...ignoring.
The object: "/proc/meminfo" is on a different file system...ignoring.
```

Так происходит потому, что некоторые системные файлы, обычно отслеживаемые Tripwire, используются вместе с хостом, на котором расположен контейнер. Это означает, что процессы, запущенные в контейнере, не всегда могут иметь полный доступ к данным файлам. Не беспокойтесь. Помимо отображения множества неприятных сообщений об ошибках, не будет никакого влияния на работу Tripwire.

11.4.3. Конфигурирование Tripwire

Вы управляете поведением Tripwire с помощью двух файлов, находящихся в каталоге `/etc/tripwire/`: `tw.cfg` и `tw.pol`. Дело в том, что они зашифрованы и их невозможно отредактировать. Кроме того, если у вас нет специальных навыков супергероя, вы даже не сможете их прочитать. Файлы созданы на основе информации из двух текстовых файлов: `twcfg.txt` и `twpol.txt`.

Файл `twcfg.txt` содержит основные переменные среды, которые вы можете изменить в соответствии со своими запросами. Большинство будет отлично работать сразу после установки. Но изменение некоторых местоположений файлов может повысить уровень безопасности, особенно если вы переживаете, что злоумышленники могут найти и отключить систему сигнализации, прежде чем приступят к выполнению своей работы. Если вы решите что-то изменить, нужно отредактировать именно файл `twcfg.txt` (листинг 11.6).

Листинг 11.6. Содержимое файла twcfg.txt по умолчанию

```
ROOT                    =/usr/sbin
POLFILE                 =/etc/tripwire/tw.pol
DBFILE                  =/var/lib/tripwire/$(HOSTNAME).twd
REPORTFILE              =/var/lib/tripwire/report/$(HOSTNAME)-$(DATE).twr
SITEKEYFILE             =/etc/tripwire/site.key
LOCALKEYFILE            =/etc/tripwire/$(HOSTNAME)-local.key
EDITOR                  =/bin/vi
LATEPROMPTING           =false
LOOSEDIRECTORYCHECKING  =false
MAILNOVIOLATIONS        =true
EMAILREPORTLEVEL        =3
REPORTLEVEL             =3
MAILMETHOD              =SENDMAIL
SYSLOGREPORTING         =false
MAILPROGRAM             =/usr/sbin/sendmail -oi -t
```

Обратите внимание, что имя файла отчета отражает имя хоста сервера для легкой идентификации

Вы можете изменить текстовый редактор по умолчанию на Nano, если вам так удобнее

Вы можете обнаружить, что изменение настройки уровня детализации на 1 облегчит чтение отчетов

Наверняка вы захотите добавить в этот файл адрес электронной почты (или адреса), на который будете отправлять отчеты. Вы можете сделать это, добавив строку `GLOBALEMAIL`, указывающую на ваш адрес:

```
GLOBALEMAIL =info@bootstrap-it.com
```

В файле `twpol.txt` описаны политики, которые Tripwire будет использовать для классификации и сканирования вашей файловой системы. Этот файл дает вам рабочую общую политику, но вам, скорее всего, понадобится выполнить хотя бы некоторые настройки для вашего конкретного сервера. Вы можете добавлять эти настройки постепенно, как только увидите ложные срабатывания в отчетах и поймете, чего не хватает Tripwire.

Вероятно, первые ложные срабатывания ваша система выдаст довольно быстро. В число вероятных виновников входят файлы идентификатора процесса (PID) и файлы конфигурации для приложений, над которыми вы активно работаете.

После того как вы запустите Tripwire пару раз и начнете распознавать ложные срабатывания, найдите в файле `twpol.txt` строку, ссылающуюся на поврежденный файл (например, строка для файла `bashrc` будет выглядеть как `/etc/bashrc -> $(SEC_CONFIG);`) и закомментируйте ее с помощью символа `#` (то есть `#/etc/bashrc -> $(SEC_CONFIG);`).

Просмотрите файл `twpol.txt`, который вы установили на своем сервере, и обратите внимание на то, как политики определяются правилами с именами, такими как Invariant Directories. Каждому правилу назначается уровень серьезности (например, `SIG_MED` для средней значимости в листинге 11.7). Обозначение `Invariant` означает, что вы не ожидаете, что такие объекты изменят разрешение или владельца, поэтому вы должны быть предупреждены, если подобные события произойдут.

Листинг 11.7. Пример правил политики Tripwire из файла twpol.txt

```
# Общедоступные каталоги, которые должны оставаться статичными
# в отношении владельца и группы.

(
  rulename = "Invariant Directories",
  severity = $(SIG_MED)
)
{
  /                       -> $(SEC_INVARIANT) (recurse = 0) ;
  /home                   -> $(SEC_INVARIANT) (recurse = 0) ;
  /etc                    -> $(SEC_INVARIANT) (recurse = 0) ;
}
```

Очевидно, что вы можете редактировать содержимое и значения любого из дюжины правил, составляющих политику Tripwire, в соответствии с вашими потребностями. Файл хорошо документирован, поэтому, потратив 10 или 15 минут на его чтение, вы должны четко разобраться в том, что можете сделать.

После редактирования текстовых файлов и при условии, что вы находитесь в том же каталоге, обновите зашифрованные версии, используя `twadmin --create-cfgfile` и `twadmin -create-polfile`:

```
# twadmin --create-cfgfile --site-keyfile site.key twcfg.txt
Please enter your site passphrase:
Wrote configuration file: /etc/tripwire/tw.cfg
#
# twadmin --create-polfile twpol.txt
Please enter your site passphrase:
Wrote policy file: /etc/tripwire/tw.pol
```

Поскольку исходные файлы — просто текст, вы должны удалить их, как только настройки будут успешно включены в их зашифрованные версии:

```
$ cd /etc/tripwire
# rm twcfg.txt
# rm twpol.txt
```

Если когда-нибудь в будущем вам понадобится обновить конфигурацию, можете восстановить исходные значения, набрав `twadmin --print-cfgfile` или `twadmin --print-polfile`. Эти команды открывают файл, в который разрешается внести любые необходимые изменения:

```
# twadmin --print-cfgfile > twcfg.txt
# twadmin --print-polfile > twpol.txt
```

Пришло время тест-драйва Tripwire. В командной строке `tripwire` принимает множество аргументов с префиксом `-m`, где `m` обозначает модуль, поэтому `-m c` будет загружать проверочный модуль. После проверки на экране появится отчет, который (вероятно) будет состоять в основном из десятков сообщений об ошибках файловой системы о файлах и каталогах, которых там нет:

```
# tripwire -m c
[...]
176. File system error.
     Filename: /proc/pci
     No such file or directory
------------------------------
*** End of report ***
Integrity check complete.
```

Теперь вы можете обновить базу данных Tripwire на основе результатов предыдущего сканирования:

```
# tripwire -m u
Please enter your local passphrase:
Wrote database file: /var/lib/tripwire/localhost.localdomain.twd
```

Возможно, Tripwire пожалуется, что не может открыть файл отчета. Если это произойдет, запустите `--update -r` для самого последнего файла, который в настоящее время находится в каталоге `/var/lib/tripwire/report/`. В текстовом редакторе вы увидите отчет. Выйдите из системы, и обновление продолжится (`:q!` должно сработать, если вы находитесь в редакторе vi):

```
tripwire -m u -r \ /var/lib/tripwire/report/\
  localhost.localdomain-20170907-102302.twr
```

Вам не нужно указывать полное имя файла, включая его дату/время, если название уникально

11.4.4. Генерация тестового отчета Tripwire

Создадим некоторые проблемы и посмотрим, заметит ли Tripwire. Добавление нового пользователя затронет всю систему. По крайней мере, файлы `passwd`, `shadow` и `group` в `/etc/` будут обновлены. Дайте вашему другу `max` учетную запись и пароль:

```
# useradd max
# passwd max
Changing password for user max.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

Теперь запустите `tripwire`, указав, что отчет по электронной почте следует отправлять с уровнем детализации 1:

```
tripwire --check --email-report --email-report-level 1
```

Если все работает, вы должны получить электронное письмо, содержащее такой текст:

```
Modified:          "/etc"
Modified:          "/etc/group"
Modified:          "/etc/group-"
Modified:          "/etc/gshadow"
Modified:          "/etc/gshadow-"
Modified:          "/etc/passwd-"
Modified:          "/etc/shadow-"
Modified:          "/etc/subgid"
Modified:          "/etc/subgid-"
Modified:          "/etc/subuid"
Modified:          "/etc/subuid-"
Modified:          "/etc/tripwire"
Modified:          "/etc/tripwire/tw.pol"
Modified:          "/etc/tripwire/tw.cfg"
Modified:          "/etc/passwd"
Modified:          "/etc/shadow"
```

Что осталось? Когда вы закончите настройку своих политик, можете добавить в cron команды `tripwire --check` и `tripwire -m u` для автоматизации процесса. Хорошей охоты!

Резюме

- Ведение журнала journald строится вокруг бинарного файла журнала, который можно точно проанализировать с помощью команды `journalctl` для определения конкретных совпадений.
- Файлы журнала быстро увеличиваются в размере и должны контролироваться ротацией с учетом технологических потребностей.
- Инструменты grep, awk и sed, предназначенные для фильтрации и форматирования текста, могут использоваться для управления большими объемами данных: grep — по совпадениям строк; awk — с разбиением строк на поля; sed — через подстановку строк.
- Tripwire может использоваться для текущего мониторинга сервера, чтобы предупредить администраторов о подозрительном поведении.

Ключевые понятия

- *journald* хранит свои данные журнала в бинарном файле, что обеспечивает более гибкий поиск, но требует наличия работающего демона.

- В системе *syslogd* все события журнала записываются в файл `/var/log/syslog`.
- *Ротация файлов журналов* syslogd контролируется через файл `/etc/logrotate.conf` и через отдельные файлы в каталоге `/etc/logrotate.d/`.
- *Фильтр текстового потока* (например, grep) позволяет искать в тексте совпадающие строки и, как правило, возвращать всю строку, содержащую совпадение.
- *Сетевая система обнаружения вторжений (NIDS)* сканирует частную сеть на наличие признаков проникновения.
- *Система обнаружения вторжений на хосте (HIDS)* отслеживает серверы на наличие вредоносных изменений в системных файлах.
- *Почтовый сервер* позволяет отправлять и получать электронную почту из приложений или на уровне командной строки.

Рекомендации по безопасности

- Регулярно проверяйте файл `auth.log` на наличие повторных неудачных попыток доступа к учетным записям пользователей.
- Сконфигурируйте программное обеспечение для сканирования, например систему обнаружения вторжений, чтобы отправлять предупреждения администраторам при возникновении потенциально опасных событий.
- Исходные текстовые файлы, применяемые для создания зашифрованных файлов конфигурации и политик Tripwire, следует удалять сразу после использования.

Обзор команд

- `Alt+F<n>` открывает виртуальную консоль из оболочки без графического интерфейса.
- `journalctl -n 20` отображает 20 последних записей журнала.
- `journalctl --since 15:50:00 --until 15:52:00` отображает только события, произошедшие от одного указанного времени до другого.
- `systemd-tmpfiles --create --prefix /var/log/journal` дает команду systemd создавать и поддерживать постоянный журнал, а не файл, который уничтожается при каждой загрузке.
- `cat /var/log/auth.log | grep -B 1 -A 1 failure` отображает совпадающие строки вместе со строками непосредственно перед и после.
- `cat /var/log/mysql/error.log | awk '$3 ~/[Warning]/' | wc` ищет в журнале ошибок MySQL события, классифицированные как предупреждение.
- `sed "s/^[0-9]//g" numbers.txt` удаляет числа в начале каждой строки файла.
- `tripwire --init` инициализирует базу данных при установке Tripwire.
- `twadmin --create-cfgfile --site-keyfile site.key twcfg.txt` создает новый зашифрованный файл `tw.cfg` для Tripwire.

Самотестирование

1. Псевдоустройство `/dev/log` используется для:
   а) хранения файла временного журнала;
   б) хранения файла постоянного журнала;
   в) сбора данных журнала событий для syslogd;
   г) хранения журнала данных, собранных syslogd.
2. Какая из следующих команд отобразит пять самых последних записей журнала в файле журнала:
   а) `journalctl -l 5`;
   б) `journalctl -n 5`;
   в) `journalctl -f 5`;
   г) `journalctl --since 5`?
3. Какая из следующих директив будет отправлять аварийные сообщения, связанные только с ядром, в существующий файл `kern.log`:
   а) `kern.emerg -/var/log/kern` в файле `/etc/rsyslog.d/50-default.conf`;
   б) `kern.* -/var/lib/kern.log` в файле `/etc/rsyslog.d/50-default.conf`;
   в) `*.emerg -/var/log/kern.log` в файле `/etc/rsyslog.d/30-default.conf`;
   г) `kern.emerg -/var/log/kern.log` в файле `/etc/rsyslog.d/50-default.conf`?
4. Какой файл конфигурации используется для управления политиками ротации журналов для файлов в `/var/log/`:
   а) `/etc/logrotate.conf`;
   б) `/etc/systemd/journal.conf`;
   в) `/etc/logrotate.d`;
   г) `/etc/rsyslog.conf`?
5. Какие аргументы заставят grep показать окружающие строки текста вместе с соответствующей строкой, которую он отображает:
   а) `cat /var/log/auth.log | grep --B 1 --A 1 failure`;
   б) `cat /var/log/auth.log | grep --since 1 --until 1 failure`;
   в) `cat /var/log/auth.log | grep -B 1 -A 1 failure`;
   г) `cat /var/log/auth.log | grep -b 1 -a 1 failure`?
6. Какая из следующих команд sed удалит номера строк из строк текста:
   а) `sed "s/^ [0-9] //g"`;
   б) `sed -n '/^d/ p'`;
   в) `sed -n '/^d/ [0-9] p'`;
   г) `sed "s/^ [0-9] //"`?

7. Какая из следующих команд подготовит базу данных Tripwire к работе:
   а) `tripwire-setup-keyfiles`;
   б) `tripwire --init`;
   в) `twadmin --create-polfile twpol.txt`;
   г) `tripwire -m c`?
8. Какая из этих команд имеет соответствующий синтаксис для шифрования и обновления файла конфигурации Tripwire:
   а) `wadmin --create-polfile --site-keyfile site.key twcfg.txt`;
   б) `wadmin --create-cfgfile --site-keyfile site.key twcfg.txt`;
   в) `wadmin --create-cfgfile --local-keyfile local.key twcfg.txt`;
   г) `wadmin --create-cfgfile --site-keyfile site.key twpol.txt`?

Ответы

1 — в; 2 — б; 3 — г; 4 — а; 5 — в; 6 — а; 7 — б; 8 — б.

12 Совместное использование данных в частной сети

В этой главе

- Совместное использование документов с помощью протокола сетевого доступа к файловой системе (NFS).
- Тонкая настройка ограниченного доступа к общему ресурсу NFS.
- Автоматизация удаленного доступа к файлам с помощью /etc/fstab.
- Защита и шифрование общих ресурсов NFS.
- Настройка Samba для обмена файлами с клиентам и Windows.
- Организация системных ресурсов с помощью символических и жестких ссылок.

Если вы прочитали главу 8, то уже знаете, как использовать Nextcloud для обмена файлами между клиентами по небезопасным сетям вроде Интернета. Но для доверенных локальных сетей предусмотрены более простые способы сотрудничества, которые глубже интегрированы в саму файловую систему Linux.

Я не предлагаю вам работать только локально. Можно полностью зашифровать и защитить инструменты общего доступа на основе файловой системы, чтобы безопасно использовать их через Интернет, но сделать это правильно будет непросто. И особые преимущества данных инструментов лучше всего использовать ближе к дому.

В этой главе вы узнаете, как предоставить тщательно отобранные части файловой системы сервера, чтобы удаленные доверенные клиенты могли совместно работать с файлами и документами. Вам не нужно будет создавать отдельное хранилище документов, как вы это делали для Nextcloud, поскольку ваши клиенты смогут получить

доступ к документам со своего рабочего места. Это может быть полезно в следующих распространенных ситуациях.

- Вы хотите, чтобы сотрудники могли входить на любую физическую рабочую станцию по всему зданию и иметь мгновенный доступ к файлам в своих собственных домашних каталогах.
- Вы хотите сделать конкретные коллекции данных доступными для членов соответствующих команд, где бы они ни находились.
- Вы хотите предоставить полный доступ к документу для чтения/записи некоторым удаленным работникам и доступ только для чтения всем остальным.

Чуть позже мы рассмотрим настройку Samba, которая является предпочтительным инструментом для обмена документами на основе Linux с Windows-клиентами. Но в основном мы сосредоточим наше внимание на использовании NFS для обеспечения интегрированного сотрудничества между системами Linux.

12.1. Обмен файлами с помощью протокола сетевого доступа к файловым системам (NFS)

Итак, пришло время что-нибудь сделать. Как люди получают полный доступ к документам в сети? Как показано на рис. 12.1, NFS работает, позволяя клиентам монтировать определенные каталоги, размещенные на удаленном сервере, как если бы они были локальными разделами. После подключения содержимое этих каталогов будет видно в клиентской системе как на уровне командной строки, так и из графического интерфейса **Рабочего стола**.

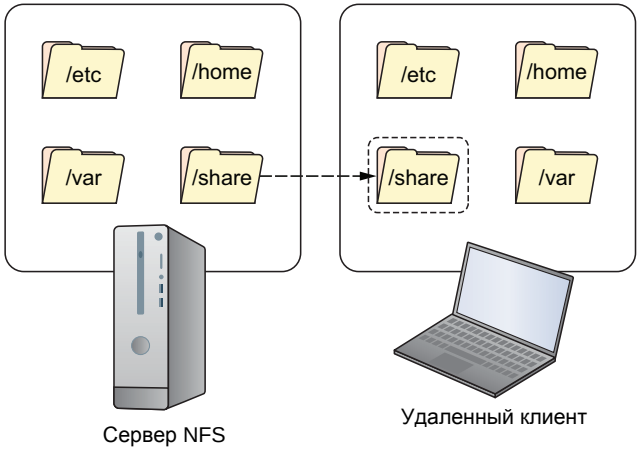


Рис. 12.1. Подключенный файловый ресурс будет выглядеть как локальный ресурс на удаленном клиенте

Еще в главе 6 вы узнали, как примонтировать периферийный носитель, чтобы получить доступ из файловой системы. Напомню:

```
# mkdir /media/mountdir/
# mount /dev/sdb1 /media/mountdir/
```

Здесь вы также будете использовать команду `mount`, но уже для монтирования каталогов с удаленного NFS-сервера в качестве локального раздела. Прежде чем мы туда доберемся, нужно кое-что сделать.

1. Установить NFS на сервере.
2. Определить клиентский доступ к ресурсам сервера в файле `/etc/exports`.
3. Обновить NFS на сервере.
4. Установить NFS на клиенте.
5. Монтировать общий ресурс NFS.
6. Конфигурировать общий ресурс NFS для монтирования при загрузке.
7. Открыть доступ через все запущенные вами брандмауэры (при необходимости).

12.1.1. Настройка NFS-сервера

Позвольте мне показать вам, как делиться домашними каталогами на вашем сервере, чтобы несколько пользователей могли иметь удаленный доступ к содержимому. Для сервера CentOS вам необходимо установить пакет nfs-utils. В Ubuntu это будет nfs-kernel-server.

ПРИМЕЧАНИЕ

> Может быть, это возможно — установить программное обеспечение NFS-сервера в контейнер LXC и заставить его думать, что он имеет достаточный доступ к ядру для правильной работы, но я подозреваю, что это не стоит вашего времени. Если вы хотите виртуализировать это упражнение, придерживайтесь VirtualBox.

В любом случае файл конфигурации, с которым вы собираетесь работать, называется `exports` и находится в каталоге `/etc/`. По крайней мере, в Ubuntu файл содержит несколько полезных примеров директив, каждая из которых отключена символом `#` (комментарий). В нашем простом примере (и при условии, что IP-адрес вашего клиентского компьютера — 192.168.1.11) это единственная активная строка, которая понадобится в файле:

```
/home 192.168.1.11(rw,sync)
```

Давайте разберемся с этим.

- `/home` сообщает NFS, что вы хотите открыть доступ к каталогу `/home` на сервере вместе со всеми его подкаталогами. До тех пор пока вы не делаете доступными

ненужные секретные данные системы или личные данные, вы можете свободно раскрывать любые каталоги, которые пожелаете.

- `192.168.1.11` обозначает IP-адрес клиента NFS, которому вы хотите предоставить доступ.
- `rw` назначает этому клиенту права на чтение и запись для файлов в открытых каталогах.
- `sync` поддерживает стабильную среду, записывая изменения на диск перед ответом на удаленные запросы.

Значения NFS по умолчанию установлены в `ro` (только для чтения, это означает, что операции записи заблокированы) и `root_squash` (пользователям удаленных клиентов не разрешено выполнять действия на сервере от имени пользователя **root**, независимо от того, какой у них статус в их собственных системах). Оба параметра обеспечивают защиту сервера и его файлов. Если вы пытаетесь открыть некоторые ресурсы для использования в качестве библиотеки знаний, то настройки по умолчанию подойдут лучше всего.

Если вы хотите переопределить `root_squash` по умолчанию и разрешить действия от имени **root** для удаленных пользователей, то должны добавить значение `no_root_squash`. Хотя это потенциально серьезная уязвимость безопасности, оно может понадобиться, когда нужно, чтобы пользователи вашего клиента выполняли административную работу с системными файлами. Вот как это будет выглядеть:

```
/home 192.168.1.11(rw,sync,no_root_squash)
```

Основываясь на сценарии использования, который обеспечивает единую сетевую иерархию домашних каталогов для всех пользователей в вашей компании, вы захотите открыть доступ к нескольким клиентам. В следующем примере мы позволяем любому из любой точки локальной сети подключить и использовать каталог `/home/` сервера. Предполагается, что никому постороннему вы не доверяете доступ к сети 192.168.1.0. Однако если эта сеть доступна через услугу Wi-Fi, которую вы предоставляете своим посетителям, то это может быть не очень хорошей идеей:

```
/home 192.168.1.0/255.255.255.0(rw,sync)
```

Если вы не понимаете, для чего нужен адрес 255.255.255.0 и как работают все эти сетевые структуры, обратитесь к разделу 14.1.

ПРИМЕЧАНИЕ

Повторюсь: в Linux и орфография, и пунктуация имеют значение. Помните, что 192.168.1.11 (rw, sync) означает, что клиент с 192.168.1.11 получит разрешения на чтение/запись. Но если добавить пробел так, чтобы директива выглядела как 192.168.1.11 (rw,sync), то любой человек получит права rw, а клиенты, входящие с IP-адреса 192.168.1.11, по умолчанию будут иметь права только на чтение!

Когда вы закончите редактировать файл `exports`, потребуется запустить `exportfs`, чтобы заставить NFS принять ваши новые настройки. Вы, вероятно, увидите уведомление о том, что проверка поддерева по умолчанию отключена. Это изменение поведения по умолчанию, потому что для большинства современных сценариев такая проверка обычно не требует усилий:

Флаг r указывает exportfs синхронизировать файловые системы, а флаг a применяет действие ко всем каталогам

```
# exportfs -ra
exportfs: /etc/exports [2]: Neither 'subtree_check' or 'no_subtree_check'
  specified for export "192.168.1.0/255.255.255.0:/home".
  Assuming default behaviour ('no_subtree_check').
  NOTE: this default has changed since nfs-utils version 1.0.x
```

Если вам интересно, то, когда включена проверка поддерева, она используется для того, чтобы убедиться, что использование файла соответствует политикам, управляющим как серверными файловыми системами, так и экспортируемыми деревьями.

Вы можете просмотреть любые файловые системы NFS, которые в настоящее время доступны клиентам, используя команду `exportfs`:

```
# exportfs
/home 192.168.1.0/255.255.255.0
```

Чтобы упростить тестирование общего ресурса со стороны клиента, создайте новый файл в своем домашнем каталоге на сервере и добавьте в него текст. Если вы используете NFS-сервер в CentOS, не забудьте открыть брандмауэр (который по умолчанию работает) и запустить NFS (который по умолчанию будет остановлен). Вот как это будет выглядеть:

```
# firewall-cmd --add-service=nfs
# firewall-cmd --reload
# systemctl start nfs-server
```

Теперь вы готовы настраивать NFS на клиентском компьютере. Об этом пойдет речь дальше.

12.1.2. Настройка клиента

Со стороны клиента все быстро и легко. Установите тот же пакет nfs-utils, который вы использовали для сервера в CentOS, и nfs-common для Ubuntu. Еще два шага. Сначала создайте новый каталог, в который вы будете монтировать удаленную файловую систему. Затем подключите его, указав IP-адрес сервера NFS и адрес файловой системы, который вы задали в файле конфигурации `/etc/export` сервера:

Флаг -p указывает Linux создавать любые каталоги по заданному пути, если они еще не существуют (например, /nfs/)

```
# mkdir -p /nfs/home/
# mount 192.168.1.23:/home /nfs/home/
```

На этом этапе вы должны иметь возможность открывать и редактировать общие файлы. Перейдите к точке монтирования, которую создали, там вы должны найти хотя бы один подкаталог, принадлежащий основному пользователю вашего сервера (в моем случае это `ubuntu`). Войдите в каталог и попробуйте открыть, отредактировать и сохранить файл, затем вернитесь на сервер, чтобы посмотреть, видна ли вам обновленная версия:

```
$ cd /nfs/home/
$ ls
ubuntu
```

Получилось? Поздравляю! Хотя это немного жутко, что вы разговариваете сами с собой. Я обещаю никому не говорить об этом.

Ничего не вышло? Вот несколько советов по устранению неполадок.

- Убедитесь, что ваш клиент и сервер имеют базовое сетевое соединение друг с другом и действительно могут общаться. Отправьте эхо-запрос на IP-адрес клиента с сервера и IP-адрес сервера с клиента (например, `ping 192.168.1.23`) и удостоверьтесь, что вы получаете соответствующий ответ. Помните, как вы делали это еще в главе 3?
- Убедитесь, что брандмауэр не блокирует трафик. По умолчанию для работы NFS необходим открытый порт TCP 2049. Если вы используете ufw, вам не нужно запоминать номер порта: `ufw allow nfs` сделает это (полный список доступных псевдонимов служб см. в файле `/etc/services`).
- Удостоверьтесь, что NFS правильно работает на сервере. Проверьте конфигурацию с помощью `exportfs`: чтобы убедиться, что NFS использует вашу последнюю версию, введите `exportfs -a`.
- Убедитесь, что IP-адрес вашего сервера не изменился. Это может легко произойти, если сервер получает свой IP динамически от сервера DHCP. В идеале ваш NFS-сервер должен использовать статический адрес (подробнее см. главу 14). В любом случае вы можете временно решить эту проблему, обновив директиву в `/etc/fstab` (см. следующий подраздел).

Если вы хотите удалить монтирование NFS с клиента, используйте команду `umount`:

```
# umount /nfs/home/
```

12.1.3. Монтирование общего ресурса NFS во время загрузки

Хотя сейчас вы можете получить доступ к этим удаленным файлам, я боюсь, что такая возможность не сохранится после очередного выключения системы. Вы можете запускать команду `mount` при каждой загрузке, но вряд ли вам захочется делать это постоянно. Вместо этого вы можете указать Linux автоматически монтировать

общий ресурс при каждой загрузке, если отредактируете файл `/etc/fstab` (листинг 12.1).

Листинг 12.1. Содержимое стандартного файла /etc/fstab

```
# /etc/fstab: static file system information.
#
# Use 'blkid' to print a universally unique identifier
# for a device; this may be used with UUID= as a more robust way to name
# devices that works even if disks are added and removed. See fstab(5).
#
# <file system>      <mount point> <type>  <options>      <dump>  <pass>
# / was on /dev/sda2 during installation
UUID=130fe070-9c44-4236-8227-6b1515baf270 /
   ext4     errors=remount-ro 0       1
# /boot was on /dev/sda1 during installation
UUID=e06b8003-255f-44b4-ab0f-291367d2928b /boot
   ext4     defaults          0       2
# /utility was on /dev/sda5 during installation
UUID=9cae6b93-963d-4995-8d33-5826da106426 /utility
   ext4     defaults          0       2
```

Инструкции по поиску идентификатора UUID для устройств, которые еще не перечислены

Как видите, активная строка `fstab` содержит шесть полей с информацией о каждом устройстве (табл. 12.1).

ПРИМЕЧАНИЕ

К числу опций fstab относятся: exec (или noexec) — для управления тем, будут ли бинарные файлы выполняться из файловой системы, ro — для ограничения доступа только для чтения, rw — для разрешения доступа для чтения и записи, а также defaults — для вызова настроек по умолчанию (rw, suid, dev, exec, auto, nouser и async).

Таблица 12.1. Поля в файле fstab

Поле	Назначение
file system	Идентифицирует устройство либо по его назначению при загрузке (/dev/sda1, которое иногда может меняться), либо — что предпочтительно — по более надежному UUID
mount point	Определяет местоположение в файловой системе, где в данный момент примонтировано устройство
type	Указывает тип файловой системы
options	Перечисляет параметры монтирования, назначенные устройству
dump	Сообщает (устаревшей) программе Dump, делать (1) или нет (0) резервное копирование устройства
pass	Сообщает программе fsck, какую файловую систему проверять первой во время загрузки. Корневой раздел должен быть первым (1)

Файл `fstab` изначально заполняется ссылками на подключенные аппаратные устройства во время установки ОС. Как администратор, вы имеете право добавлять свои собственные устройства, чтобы они также были примонтированы во время загрузки. Не забывайте ссылку на устройство. Добавление новой строки (в файле клиента `fstab`) для общего ресурса NFS может выглядеть следующим образом:

```
192.168.1.23:/home /nfs/home nfs
```

Перезагрузите свой клиент. На этот раз не монтируйте общий ресурс NFS вручную, а сразу перейдите в каталог, где он должен быть смонтирован (`/nfs/home/`), чтобы убедиться, что вы видите файлы на сервере. Если это так, то вы знаете, что `fstab` справился со своей задачей. Запустите команду `mount` самостоятельно; наряду с длинным списком других файловых систем вы должны увидеть свой новый общий ресурс NFS с параметрами, которые были ему назначены (листинг 12.2).

Листинг 12.2. Частичный вывод смонтированных файловых систем на клиенте

```
# mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
/dev/sda2 on / type
 ext4 (rw,relatime,errors=remount-ro,data=ordered)        <-- Корневой раздел
/dev/sda1 on /boot type ext4 (rw,relatime,data=ordered)
192.168.1.23:/home on /nfs/home type nfs
 (rw,relatime,vers=4.0,rsize=262144,wsize=262144,namlen=255,hard,
 proto=tcp,port=0,timeo=600,retrans=2,sec=sys,clientaddr=192.168.1.11,
 local_lock=none,addr=192.168.1.23)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,
 size=204828k,mode=700,uid=1000,gid=1000)
                                  Общий ресурс NFS
                                  и параметры его загрузки
```

12.1.4. Безопасность NFS

Посмотрите еще раз на заголовок этой главы («Совместное использование данных в частной сети») и отметьте слово *«частной»*. Как я упоминал ранее, NFS, если использовать его на своем собственном устройстве, не будет шифровать данные, которые он перебрасывает между хостами. Вы должны использовать NFS только в частных сетях.

Но насколько безопасны частные пространства? Мы называем сеть *частной*, когда к ней невозможно подключиться извне, используя обычные правила доступа. Однако хочу вас заверить: хакеры не стесняются использовать нестандартные правила доступа. Или, точнее, они могут создавать и развертывать свои собственные частные правила.

Если безопасность вашего общего NFS-сервера зависит от того, насколько вы доверяете сотрудникам офиса, то это плохая защита и могут потребоваться некоторые улучшения. Дело не в том, что вы никогда не должны доверять людям, с которыми вместе работаете, а в том, что вы не всегда можете быть уверены, что их компьютеры

и учетные записи не были взломаны посторонними. Вы также не можете быть уверены, что никто извне не прослушивает ваш сетевой трафик.

Ваш подход должен заключаться в том, чтобы следовать *принципу наименьших привилегий*: никогда не открывайте доступ шире, чем это абсолютно необходимо, и никогда не предоставляйте пользователям (или клиентам) больше доступа, чем им абсолютно необходимо для выполнения их работы. Например, если вашим клиентам не нужно редактировать файлы, обязательно укажите параметр `ro` (только для чтения) в файле `/etc/exports`. Или, если вашим клиентам нужен доступ только к некоторым файлам в структуре каталогов, заблокируйте все остальное.

Правила брандмауэра

Важной линией обороны будет ваш брандмауэр. Если в вашей доверенной частной сети есть отдельные компьютеры, которым не нужен доступ к вашему NFS-серверу, заблокируйте их:

```
# ufw deny to 192.168.1.10
```

На CentOS это выглядело бы так:

```
# firewall-cmd --add-rich-rule="rule family='ipv4'
     source address='192.168.1.10' reject"
# firewall-cmd --reload
```

Если говорить о блокировке, то после изменения правил брандмауэра на NFS-сервере вам может потребоваться перезагрузить клиентский компьютер, прежде чем общий ресурс NFS будет полностью заблокирован.

Иногда может потребоваться, чтобы на сервере размещались не только файлы NFS. Там могут быть и несвязанные хранилища данных и файлы конфигурации, необходимые для разработчиков или администраторов, работающих на своих компьютерах. Вот как можно заблокировать разработчику с IP-адресом 192.168.1.10 доступ к NFS на сервере, сохраняя при этом полный доступ к другим ресурсам сервера:

```
# ufw deny from 192.168.1.10 to any port 2049
```

Чтобы проверить это, рассмотрите возможность установки веб-сервера Apache в дополнение к NFS на вашем сервере NFS. Если ваша общая папка NFS не загружается на клиенте, но вы все равно можете получить доступ к веб-приложению на сервере, то считайте, что добились результата:

```
# ls /nfs/home/
# curl 192.168.1.3
Welcome to my web server on my NFS machine
```

curl извлекает содержимое веб-страницы, если она доступна

Шифрование

Как всегда, ограничение доступа к вашим ресурсам не означает, что данные, которые вы передаете между серверами и клиентами, безопасны — по крайней мере когда они не зашифрованы. Если вы только передаете данные в своей локальной частной

сети и уверены, что они действительно локальные и частные, то такое поведение простительно. Но любые данные, которые будут перемещаться по небезопасным сетям, требуют дополнительной защиты.

Вот краткий список способов, используя которые вы можете справиться с проблемой (помимо того, что вы закрываете глаза и надеетесь, что с вами случится только хорошее).

- Вы можете запустить общий ресурс NFS поверх VPN практически так же, как использовали VPN для шифрования перемещения данных в главе 10.
- Протокол IPSec защищает данные на уровне IP-сети, предотвращая подделку IP-адресов и пакетов. Он шифрует все данные, содержащиеся в IP-пакетах, с помощью ключей сеанса. Вы можете настроить IPSec как туннель, который в целом будет работать так же, как и туннель OpenVPN.
- NFS может быть настроен для работы через базовый сеанс SSH, который является другим видом туннеля. Всегда хорошо иметь несколько доступных вариантов, если вы сталкиваетесь с необычным сценарием использования. Обратите внимание, что это нестандартный вариант.
- Предположим, у вас уже есть сервер аутентификации Kerberos, работающий как часть вашей инфраструктуры, и вы используете по крайней мере NFS v.4. Тогда, добавив строку `sec=krb5p` в вашу конфигурацию NFS, вы отдадите управление проблемой в руки Kerberos.
- Наконец, альтернативой NFS является файловая система SSH (SSHFS). SSHFS аналогичным образом монтирует удаленные файловые системы как локальные тома, но работает через протокол SFTP с использованием программного интерфейса FUSE (файловая система в пользовательском пространстве).

12.2. Обмен файлами с пользователями Windows с помощью Samba

Если в вашей офисной сети есть пользователи, которым необходим доступ к Linux-файлам с их компьютеров с Windows, то стоит воспользоваться Samba. Это простое и одновременно чертовски сложное решение. Что я имею в виду? Настройка базовой связи, подобной той, которую я собираюсь продемонстрировать, не потребует большого количества работы. Но интеграция аутентификации с доменом Windows Active Directory или борьба с SELinux на сервере Linux может доставить вам гораздо больше хлопот. Не случайно за эти годы появилось множество книг по настройке Samba.

Для наших целей вы можете установить только пакеты samba и smbclient в Ubuntu. Для CentOS проще запустить команду `yum` с аргументом `samba*`, чтобы получить всю коллекцию инструментов, связанных с Samba (помните: символ * интерпретируется

как возвращающий все результаты, содержащие предыдущий текст). Вот как будет разворачиваться остальная часть процесса.

1. Создайте учетную запись пользователя Samba на сервере Linux.
2. Выберите каталог общего пользования.
3. Определите общий ресурс в файле `smb.conf`.
4. Протестируйте конфигурацию.
5. Подключитесь с Windows-клиента.

Вам потребуется программа smbpasswd, чтобы задать пароль пользователя Samba в качестве учетной записи, которую клиенты будут использовать для входа в систему. Но, поскольку полномочия Samba будут добавлены в существующую учетную запись, вам следует сначала создать новую учетную запись Linux. Я назвал своего пользователя **sambauser**, но вы можете выбрать любое другое имя:

```
# adduser sambauser
# smbpasswd -a sambauser
```

Далее вы можете создать каталог, в котором будет располагаться общий ресурс. Я собираюсь следовать той же схеме, которую использовал ранее для своей папки NFS. Чтобы позже упростить тестирование, я создам файл в новом каталоге. Поскольку с файлами в нем могут в конечном итоге работать несколько клиентов, вы можете избежать потенциальных проблем с разрешениями, используя команду `chmod` для открытия доступа к каталогу `777` (чтение, запись и выполнение для всех пользователей):

```
# mkdir -p /samba/sharehome
# touch /samba/sharehome/myfile
# chmod 777 /samba/sharehome
```

Это свежесозданная среда Samba, в которой все есть. Теперь вы можете добавить конфигурацию в файл `smb.conf` в каталоге `/etc/samba/`. Вы должны просмотреть файл конфигурации, чтобы понять, насколько возможна настройка и как задать конфигурацию для сложных действий:

```
# nano /etc/samba/smb.conf
```

Для ваших скромных целей потребуется добавить лишь один раздел, который опишет разделяемый ресурс. Я наконец расскажу, что нужно делать. Вы должны обязательно включить две записи. Первой будет путь, указывающий на каталог, который вы планируете использовать для ваших общих документов. Если же вы хотите, чтобы ваши клиенты могли создавать и редактировать файлы, то во второй записи нужно указать доступ: `writable = yes` (листинг 12.3). После этого сохраните файл и закройте текстовый редактор.

Листинг 12.3. Раздел конфигурации общего файлового ресурса из файла /etc/samba/smb.conf

```
[sharehome]
path = /samba/sharehome
writable = yes
```

Теперь используйте systemctl для запуска и включения демона Samba. Ubuntu знает Samba как `smbd`, но в CentOS это будет `smb` (без *d*):

```
# systemctl start smbd
# systemctl enable smbd
```

Обратите внимание, что smbd — это имя демона, которое systemctl знает в Ubuntu; на CentOS это будет smb

12.2.1. Тестирование вашей конфигурации Samba

Всегда полезно проверить свою конфигурацию, прежде чем углубляться дальше. Запуск команды `testparm` покажет вам, может ли добавленный вами раздел быть правильно прочитан службой Samba. В следующем выводе предполагается, что все в порядке:

```
# testparm
Load smb config files from /etc/samba/smb.conf
rlimit_max: increasing rlimit_max (1024) to minimum Windows limit (16384)
WARNING: The "syslog" option is deprecated
Processing section "[printers]"
Processing section "[print$]"
Processing section "[sharehome]"
Loaded services file OK.
Server role: ROLE_STANDALONE
Press Enter to see a dump of your service definitions
```

testparm тестирует и отображает вашу конфигурацию Samba

Подтверждение того, что ваш раздел sharehome распознан

Еще один тест перед тем, как пригласить своих друзей из Windows: вы можете использовать программу smbclient для входа на общий ресурс Samba с локального компьютера. Сначала вам нужно переключить пользователя (`su`) на учетную запись sambauser, которую вы ранее связали с Samba. Затем можете запустить smbclient для адреса и имени хоста общего ресурса (`//localhost/sharehome`):

```
$ su sambauser
Password:
# smbclient //localhost/sharehome
Enter sambauser's password:
Domain=[WORKGROUP] OS=[Windows 6.1] Server=[Samba 4.3.11-Ubuntu]
smb: \>
```

Обратите внимание на специальную командную строку Samba, сообщающую, что вы сейчас находитесь в сеансе оболочки Samba

В этом случае, как вы можете видеть из командной строки, вы попали в оболочку Samba. Запустите `ls`, чтобы отобразить содержимое, и вы должны увидеть созданный вами файл `myfile`:

```
smb: \> ls
  .        D        0  Thu Sep 14 20:54:32 2017
  ..       D        0  Thu Sep 14 20:36:52 2017
  myfile   N        0  Thu Sep 14 20:54:32 2017

    953363332 blocks of size 1024. 732660884 blocks available
```

Кстати, как вы уже знаете из главы 9, неопытные и/или нетерпеливые администраторы иногда испытывают желание отключить SELinux при первых признаках конфликта. Конечно, так делать не стоит. В большинстве случаев.

Если вы запускаете следующую команду в CentOS и не можете получить доступ к файлам в вашей общей папке, это может быть связано с тем, что мешает SELinux:

```
smb: \> ls
NT_STATUS_ACCESS_DENIED listing \*
```

Никогда не делайте то, что я вам дальше скажу. Даже для простой демонстрации на непроизводственном сервере. Никогда. Но если вы действительно хотите отключить SELinux на минуту или две, чтобы избежать необходимости читать документацию по файлу `smb.conf` в попытке выяснить, как правильно решить проблему, вам поможет запуск `setenforce 0`. Но учтите: я никогда не говорил вам этого.

Если у вас есть брандмауэр, защищающий ваш сервер, вам нужно открыть несколько портов, чтобы Samba могла передавать и принимать данные. Но конкретные порты, которые вам понадобятся, будут частично зависеть от того, что вы включите в свою конфигурацию Samba. Чаще всего в связке с Samba работают службы Kerberos, LDAP, DNS и NetBIOS.

Если вы введете команду `netstat` на сервере для поиска портов, используемых Samba (с добавлением термина `smbd`), то получите список ключевых портов в базовой системе. Этот пример возвращает 139 и 445, но их может быть гораздо больше:

```
# netstat -tulpn | egrep smbd
tcp   0  0 0.0.0.0:139  0.0.0.0:*  LISTEN 2423/smbd
tcp   0  0 0.0.0.0:445  0.0.0.0:*  LISTEN 2423/smbd
tcp6  0  0 :::139       :::*       LISTEN 2423/smbd
tcp6  0  0 :::445       :::*       LISTEN 2423/smbd
```

Если у вас на компьютере нет netstat, установите пакет net-tools

12.2.2. Доступ к серверу Samba из Windows

На клиентском компьютере под управлением Windows создайте ярлык на Рабочем столе (в Windows они тоже есть!) и укажите местоположение общего ресурса Samba на вашем сервере Linux. Если предположить, что IP-адрес сервера Linux — 192.168.1.23, а имя, которое вы дали разделяемому ресурсу Samba, было sharehome, то введенный вами адрес будет выглядеть следующим образом:

```
\\192.168.1.23\sharehome
```

Для подключения пользователь Windows должен пройти аутентификацию, указав имя пользователя и пароль Linux Samba, которые вы создали ранее, если только вы не решили интегрировать аутентификацию Samba с доменом Windows Active Directory. Это то, что может быть более эффективным в долгосрочной перспективе и более безопасным, чем простой пароль. Я покажу вам, как это делается, но сначала позвольте мне быстро взглянуть на обложку книги, чтобы подтвердить мои подозрения. Ага. Это книга о Linux. Извините, но это не руководство по настройке Windows.

12.3. Совместное использование файлов с помощью символических ссылок

Пользователи могут захотеть поделиться своими файлами друг с другом. Это можно сделать, создав символические ссылки (также известные как *symlinks*). Позвольте мне проиллюстрировать это. Ранее в главе вы использовали общий ресурс NFS, чтобы предоставить пользователям доступ к центральному каталогу `/home/`. Цель состояла в том, чтобы позволить любому пользователю сесть за любой компьютер в офисе и через созданный вами каталог `/nfs/home/` немедленно приступить к работе со своими собственными документами.

Но что, если не все ваши пользователи настолько знакомы с иерархией файловой системы Linux, не говоря уже о командной строке? Возможно, некоторые пользователи не знают, где в файловой системе находятся их файлы. Не стесняйтесь создавать символическую ссылку, например, на смонтированный общий каталог, который будет отображаться на Рабочем столе графического интерфейса пользователя.

ПРИМЕЧАНИЕ

> Учетным записям пользователей в системах с графическим интерфейсом Linux предоставляется набор подкаталогов (таких как Documents (Документы) и Pictures (Изображения)) в их каталоге /home/username/, где они могут удобно организовать свои документы. Один из этих подкаталогов называется Desktop (Рабочий стол), и любые сохраненные в нем файлы будут отображаться в виде значков на экране графического интерфейса, который пользователи увидят первым при входе в систему. Имейте в виду, что эти подкаталоги не будут созданы до тех пор, пока пользователь не войдет в систему в первый раз со своей новой учетной записью.

Вы создаете символическую ссылку с помощью команды `ln`, за которой следует аргумент `-s`, за ним — объект файловой системы, который вы хотите связать, и место, где вы хотите разместить ссылку:

```
# ln -s /nfs/home/ /home/username/Desktop/
```

Это все, что нужно сделать. На самом деле это так просто, что вы можете задаться вопросом, почему обсуждение заслуживает отдельного раздела. Все дело в аргументе `-s`. Символ `s` обозначает «символический», то есть может быть и другой вид ссылки. Действительно, по умолчанию `ln` создаст жесткую ссылку.

Но в чем между ними разница? *Символическая ссылка* указывает на отдельный объект файловой системы. Чтение, выполнение или редактирование символической ссылки приведет к чтению, выполнению или редактированию связанного объекта. Но если вы переместите или удалите оригинал, на который ссылается символическая ссылка, она будет оборвана.

Жесткая ссылка, напротив, является точной копией своего оригинала до такой степени, что эти два файла будут совместно использовать один индексный дескриптор

(inode). inode, как вы помните из главы 1, — это метаданные, описывающие атрибуты объекта, в частности его местоположение в файловой системе.

Посмотрим на все это в действии. Создайте два файла и добавьте в каждый из них немного текста. Теперь создайте жесткую ссылку для одного из файлов и символическую ссылку для другого:

```
$ nano file1
$ nano file2
$ ln file1 file1-hard
$ ln -s file2 file2-sym
```

Запустите `ls` с аргументом `i`, чтобы отобразить идентификаторы inode. Обратите внимание, что жестко связанные файлы имеют идентичный идентификационный номер inode. Обратите также внимание, как файл `file2-sym` указывает на `file2`:

```
$ ls -il
9569544 -rw-rw-r-- 2 ubuntu ubuntu  4 Sep 14 15:40 file1
9569544 -rw-rw-r-- 2 ubuntu ubuntu  4 Sep 14 15:40 file1-hard
9569545 -rw-rw-r-- 1 ubuntu ubuntu  5 Sep 14 15:40 file2
9569543 lrwxrwxrwx 1 ubuntu ubuntu  5 Sep 14 15:40 file2-sym -> file2
```

Если вы удалите, переименуете или переместите `file2`, его символическая ссылка будет разорвана и, в зависимости от цветовой схемы вашей оболочки, будет отображаться на ярко-красном фоне. Жесткая ссылка, с другой стороны, переживет все изменения в оригинале. Попробуйте сами: переименуйте файлы `file1` и `file2` и посмотрите, что происходит с их соответствующими ссылками:

```
$ mv file1 newname1
$ mv file2 newname2
```

Как вы можете заметить, когда вы повторно запускаете `ls -il`, даже после использования `mv` для переименования `file1`, ссылка `file1-hard` все еще там и, что более важно, все еще использует тот же дескриптор с `newname1`. Любые изменения в одном файле будут отражены в другом. К сожалению, `file2-sym` «осиротел».

Зачем все это? Почему бы просто не сделать копии? Вспомните ситуацию, которую наблюдали чуть раньше, когда создали ссылку на местоположение файловой системы на **Рабочем столе** пользователя. И как насчет связи между файлами `/sbin/init` и `/lib/systemd/systemd`, которые вы видели в главе 8, или содержимым `/etc/apache2/sites-available/` и `/etc/apache2/sites-enabled/` из главы 3? Все это примеры ссылок, используемых для улучшения функциональности системы.

Жесткие ссылки гораздо более популярны в Linux, но их назначение не всегда так очевидно. Даже если файлы хранятся в разных местах файловой системы, они будут связаны вместе. Это может упростить резервное копирование важных, но далеко расположенных друг от друга файлов конфигурации без необходимости регулярно делать новые копии. А поскольку жесткие ссылки не занимают места, вам не нужно беспокоиться о дополнительном использовании диска.

Резюме

- Разделяемые ресурсы NFS предоставляют определенные ресурсы файловой системы на удаленных компьютерах, обеспечивая удобный доступ к документам и совместную работу пользователей.
- Файловые системы и устройства можно монтировать (и использовать) либо вручную из командной строки, либо во время загрузки с помощью записей в файле `/etc/fstab`.
- Вы можете защитить NFS-ресурсы с помощью политик интеллектуальной конфигурации, правил брандмауэра и шифрования, предоставляемых сторонними решениями, такими как Kerberos и IPSec.
- Клиенты Windows могут работать с файлами Linux, используя пакет Samba и сгенерированную им учетную запись.
- Символьные ссылки указывают на объекты файловой системы, в то время как жесткие ссылки создают объекты, являющиеся точными копиями оригинала, не занимая места на диске.

Ключевые термины

- *Общий ресурс* — это набор документов или файловая система, которые можно использовать совместно с удаленным клиентом.
- *Конфигурацию сервера Samba* можно протестировать с помощью команды `testparm`.
- *Символьные и жесткие ссылки* позволяют предоставлять в нескольких местах объекты файловой системы Linux (файлы и каталоги).

Рекомендации по безопасности

- Разделяемые ресурсы NFS должны быть максимально конкретно описаны, включая только тех клиентов, которым необходим доступ, и только те права, которые требуются каждому клиенту. Это соответствует *принципу наименьших привилегий*.
- Там, где это возможно, политики обращения с разделяемыми ресурсами NFS должны всегда включать параметр `squash_root`, чтобы гарантировать, что удаленные клиенты не получат неограниченный root-доступ к серверу.
- Разделяемые ресурсы NFS (и Samba) по умолчанию не шифруются. Если ваши клиенты будут обращаться к ним через ненадежную сеть, вам следует использовать шифрование.

Обзор команд

- `/home 192.168.1.11(rw,sync)` (запись в файле сервера NFS `/etc/exports`) определяет разделяемый ресурс удаленного клиента.

- `firewall-cmd --add-service=nfs` открывает брандмауэр CentOS для клиентского доступа к вашему ресурсу NFS.
- `192.168.1.23:/home /nfs/home nfs` (типичная запись в файле `/etc/fstab` клиента NFS) монтирует ресурс NFS.
- `smbpasswd -a sambauser` добавляет возможность использовать Samba (и уникальный пароль) к существующей учетной записи Linux.
- `nano /etc/samba/smb.conf` редактирует конфигурационный файл Samba на сервере.
- `smbclient //localhost/sharehome` выполняет вход в систему для использования локального ресурса Samba с учетной записью Samba.
- `ln -s /nfs/home/ /home/username/Desktop/` создает символическую ссылку, позволяющую пользователю легко открыть ресурс NFS, щелкнув на ярлыке на Рабочем столе.

Самотестирование

1. Какая из следующих директив обеспечит доступ только для чтения к клиенту NFS с использованием IP-адреса 192.168.1.11:
   а) `/home 192.168.1.11(ro,sync)`;
   б) `192.168.1.11 /home(ro,sync)`;
   в) `/home 192.168.1.11(rw,sync)`;
   г) `192.168.1.11 /home(r,sync)`?
2. Какой файл конфигурации на стороне клиента необходимо отредактировать, чтобы обеспечить подключение общего ресурса NFS во время загрузки:
   а) `/etc/nfs`;
   б) `/etc/exports`;
   в) `/etc/nfs/exports`;
   г) `/etc/fstab`?
3. Что будет выполнять команда `exportfs -ra`:
   а) заставит NFS перезагрузить политики из файла конфигурации;
   б) выведет список текущих файловых систем NFS;
   в) проверит конфигурацию Samba;
   г) перезагрузит все подключенные устройства?
4. Что из нижеперечисленного полностью блокирует доступ одного удаленного клиента к серверу:
   а) `ufw deny ALL 192.168.1.10/UDP`;
   б) `ufw deny ALL 192.168.1.10`;
   в) `ufw deny to 192.168.1.10`;
   г) `ufw deny to 192.168.1.10/255.255.255.0`?

5. Какой из следующих инструментов безопасности не поможет защитить данные при передаче между сервером и клиентскими компьютерами:
   а) Kerberos;
   б) firewalld;
   в) SSHFS;
   г) IPSec?
6. Каким будет правильное имя и месторасположение файла конфигурации сервера Samba:
   а) `/etc/samba/smb.conf`;
   б) `/etc/smb.cfg`;
   в) `/etc/samba/smb.conf`;
   г) `/etc/samba.cfg`?
7. Какая из следующих команд запустит Samba на сервере Ubuntu:
   а) `systemctl start sambad`;
   б) `systemctl start smbd`;
   в) `systemctl start smb`;
   г) `systemctl start samba`?
8. Какая из следующих команд лучше всего подходит для создания ссылки на файл конфигурации, чтобы сделать обычные резервные копии более простыми и надежными:
   а) `ln -s /var/backups/important_config_file_backup.cfg /etc/important_config_file.cfg`;
   б) `ln /var/backups/important_config_file_backup.cfg /etc/important_config_file.cfg`;
   в) `ln -s /etc/important_config_file.cfg /var/backups/important_config_file_backup.cfg`;
   г) `ln /etc/important_config_file.cfg /var/backups/important_config_file_backup.cfg`?

Ответы
1 — а; 2 — г; 3 — а; 4 — в; 5 — б; 6 — в; 7 — б; 8 — г.

13 Устранение проблем производительности системы

В этой главе

- Получение информации о поведении вашей системы и сбор метрик.
- Контроль требований приложений и клиентов к системным ресурсам.
- Многоуровневые стратегии для решения проблемы нехватки ресурсов.
- Использование эффективных протоколов постоянного мониторинга.

«Каскадный хаос» и «обреченность» — так описываются ваши IT-операции прямо сейчас? Ваши серверы медленные и не отвечают? Жалуются ли ваши клиенты на низкую производительность приложений?

Даже если все не так плохо, жизнь не всегда будет спокойной. Тот факт, что по определению мы постоянно стараемся извлечь максимальную выгоду из наших инвестиций в IT, означает, что иногда мы слишком далеко заходим: перегруженные системы ломаются, а сложные элементы программного стека перестают слаженно работать.

Секрет долгой и счастливой жизни заключается в том, чтобы предвидеть неприятности, быстро выявлять симптомы и причины проблем и применять правильные решения в нужное время. Все это должно помочь в вашей работе в качестве IT-администратора.

Что я имею в виду под *системой*? Это аппаратная и программная среда, которую вы используете для предоставления своих услуг, будь то приложение, база данных, веб-сервер или простая, но надежная автономная рабочая станция. В этой главе вы сосредоточитесь на работоспособности четырех основных элементов вашей системы: центрального процессора (ЦП), памяти (как физической, так и виртуальной ОЗУ),

устройств хранения и управления сетевой нагрузкой. Вы узнаете, как обнаружить проблему, определить причину, а затем либо устранить основную проблему, либо при необходимости использовать больше оборудования.

Конечно, хотелось бы вообще избежать проблем. Один из способов сделать это — подвергнуть вашу систему стресс-тестированию, чтобы увидеть, как она работает. Чуть позже в этой главе я также кратко расскажу о *yes* — отличном инструменте, который может подвергнуть вашу инфраструктуру жестоким и необычным пыткам.

В любом случае я предполагаю, что один из серверов или компьютеров, за которые вы несете ответственность, дал сбой или замедлился в самое неподходящее время. Вы злобно смотрите на него в течение минуты или двух, но машина никак не реагирует. Проработаем такую ситуацию шаг за шагом.

13.1. Проблемы с загрузкой процессора

Процессор — это мозг вашего компьютера. Предполагается, что электронные схемы ЦП, независимо от количества ядер и ширины используемой шины, будут выполнять только одну задачу: ждать инструкций, переданных программным обеспечением, выполнять вычисления и выдавать ответы.

По большому счету, ваш процессор либо будет работать, либо не будет. Большинство проблем с производительностью, связанных с ЦП (например, большое время отклика или неожиданное отключение), спровоцированы превышением его физических возможностей. Задаваясь вопросом, может ли какая-то проблема с производительностью быть связана с процессором, вам в первую очередь нужно выяснить, не слишком ли грубо вы относитесь к бедняжке.

13.1.1. Измерение загрузки процессора

О состоянии процессора говорят два индикатора: загрузка процессора и его использование.

- *Загрузка ЦП* — это мера объема работы (то есть количества текущих активных процессов и процессов в очереди), выполняемой ЦП в процентах от общей производительности. Средние значения нагрузки представляют активность системы с течением времени и дают гораздо более точную картину состояния вашей системы, поэтому в данном случае являются лучшей метрикой.
- *Использование ЦП* — мера времени, в течение которого ЦП не простаивает (описывается как доля от общей производительности ЦП).

Оценка нагрузки 1 на одноядерном компьютере будет предоставлять полную мощность. Если ваша система имеет несколько ядер ЦП, скажем четыре, то полная мощность будет представлена числом 4. Пользователь, вероятно, будет замечать задержки (по крайней мере время от времени), когда показатель использования ЦП поднимется выше 75 %, что будет 0,75 для одного ядра и 3,0 для четырехъядерной системы.

Получить среднюю загрузку процессора легко. Команда `uptime` позволяет вернуть текущее время, время, прошедшее с момента последней загрузки системы, количество пользователей, вошедших в систему в данный момент, и, что наиболее важно для нас сейчас, средние значения загрузки за последнюю минуту, 5 и 15 минут:

```
$ uptime
10:08:02 up 82 days, 17:13, 1 user, load average: 0.12, 0.18, 0.27
```

Средняя загрузка 1,27 в системе с одним ЦП будет означать, что в среднем ЦП работает на полную мощность и еще 27 % процессов ждут своей очереди на ЦП. Напротив, средняя нагрузка 0,27 в системе с одним ЦП означала бы, что в среднем ЦП не использовался в течение 73 % времени. В четырехъядерной системе вы можете увидеть средние значения нагрузки в диапазоне 2,1, что составляет чуть более 50 % емкости (или не используется примерно 52 % времени).

Чтобы правильно понять эти цифры, вам нужно знать, сколько ядер работает в вашей системе. Если эта информация не указана на наклейке шасси и вы не хотите открывать корпус и искать, то можете запросить псевдофайл `cpuinfo`:

```
$ cat /proc/cpuinfo | grep processor
processor    : 0
processor    : 1
processor    : 2
processor    : 3
```

Похоже, в этой системе четыре ядра. Раз уж вы открыли файл, просмотрите и остальную его часть, чтобы понять, как Linux описывает ваш процессор. В частности, попробуйте разобраться в разделе флагов, где перечислены функции, поддерживаемые вашим оборудованием.

13.1.2. Управление загрузкой процессора

Стабильно недостаточная загрузка (на основе результатов, которые вы получаете от `uptime`)? Вы можете просто наслаждаться дополнительными возможностями. Или подумать о консолидации ресурсов, используя недостаточно загруженный компьютер для предоставления дополнительных услуг (вместо покупки дополнительных серверов), чтобы максимизировать окупаемость инвестиций.

Постоянно перегружен? Вам нужно будет либо перейти на более надежную аппаратную архитектуру с бóльшим количеством ядер ЦП, либо в виртуальной среде выделить больше виртуальных машин или контейнеров для выполнения дополнительной работы. Время от времени вы также можете внимательнее присматриваться к процессам, запущенным в вашей системе, чтобы увидеть, нет ли чего-то лишнего или даже того, что запускает вредоносное ПО без вашего ведома. Выполнение команды `top` обеспечивает отображение развернутой, самообновляющейся информации о процессе.

На рис. 13.1 приведен типичный пример вывода `top`. Обратите внимание, что в первой строке представлены те же сведения, которые вы получили бы с помощью команды `uptime`. Поскольку вы пытаетесь решить проблемы с производительностью, наиболее интересным столбцом данных является **`%CPU`** (процентная доля процессорной мощности, используемой в данный момент конкретным процессом), особенно для процессов, отображаемых вверху списка.

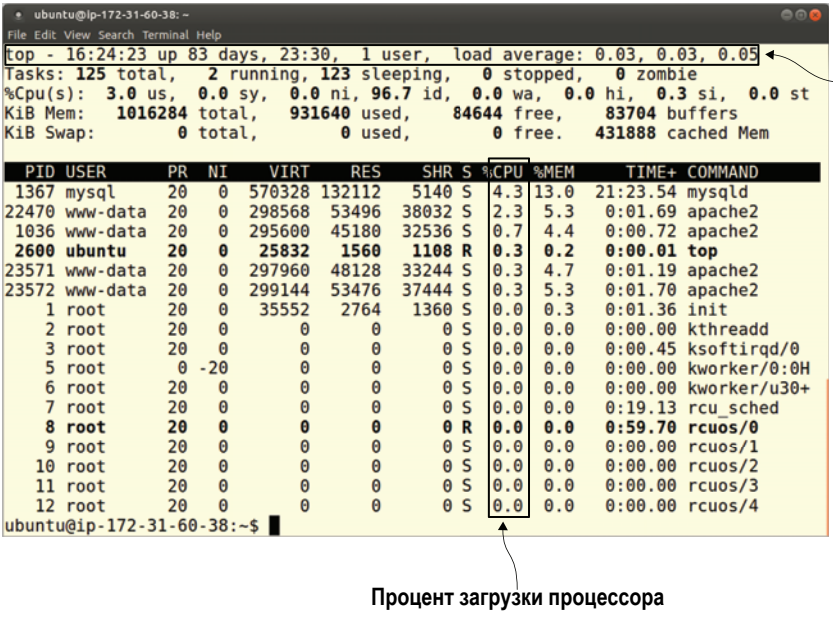


Рис. 13.1. Моментальный снимок данных по процессам, отображаемый командой top

Здесь вы можете видеть, что демон MySQL использует 4,3 % ЦП сервера и 13 % его памяти. Если вы посмотрите слева, то увидите, что идентификатор процесса равен 1367 и этот процесс принадлежит пользователю `mysql`. Вы могли бы прийти к заключению, что процесс требовал больше ресурсов, чем можно было бы оправдать, и им придется пожертвовать (для большего блага). Вы можете выйти из утилиты top, нажав клавишу Q.

Проанализировав информацию, выведенную top, вы смогли принять решение убить процесс. Поскольку MySQL — это сервис, управляемый systemd, вам нужно использовать команду `systemctl`, чтобы аккуратно остановить процесс, не подвергая риску какие-либо данные приложения:

```
# systemctl stop mysqld
```

Если процессом управляет не systemd или что-то пошло не так и выполнением `systemctl` не удалось остановить его, можете воспользоваться командой `kill` или

killall (некоторые системы требуют, чтобы вы установили killall как часть пакета psmisc). Вы передаете PID команде kill таким образом:

```
# kill 1367
```

С другой стороны, killall использует имя процесса, а не его идентификатор:

```
# killall mysqld
```

Что же выполнить: kill или killall? На самом деле ответ очевиден. kill завершит работу одного процесса, основанного на PID, тогда как killall уничтожит столько экземпляров конкретной программы, сколько было запущено. Если бы было два или три отдельных экземпляра MySQL, возможно принадлежащих отдельным пользователям, все они были бы остановлены.

ПРИМЕЧАНИЕ

Перед запуском killall убедитесь, что не запущены процессы с аналогичными именами, которые тоже могут пострадать.

Вам также придется использовать systemctl еще раз, чтобы убедиться, что процесс не перезапустится при следующей загрузке:

```
# systemctl disable mysqld
```

Расстановка приоритетов с помощью инструмента nice

Не всегда можно убить процесс — иногда он является необходимой частью критически важного сервиса. Но вы можете ограничить ресурсы процессора, которые он получает, воспользовавшись командой nice. По умолчанию новому процессу присваивается значение nice, равное 0, но вы можете изменить его на любое число в диапазоне от –20 до 19. Чем выше число, тем более покладистым будет процесс, когда дело доходит до отказа от ресурсов в пользу других. И наоборот, чем меньше число, тем больше ресурсов захватывает процесс, несмотря на другие.

Например, вы хотите запустить сценарий mybackup.sh, который выполняет большую работу по удаленному резервному копированию. Проблема в том, что у вас активный сервер, которому время от времени требуется много ресурсов для ответа на важные запросы клиентов. Если запрос поступит во время активного выполнения резервного копирования, производительность будет неприемлемой. С другой стороны, если резервное копирование выполняется от начала до конца, оно может никогда не завершиться.

Команда nice может помочь вам справиться с этим так, чтобы всем угодить. Добавьте перед сценарием (или любой другой командой) nice и числовое значение, которое вы выбрали для выполнения. В этом случае дефис (-), за которым следует 15, говорит Linux, что сценарий будет работать с очень хорошим приоритетом. Это означает, что, когда возникает конфликт из-за доступа к ресурсам, ваш сценарий будет отложен, но в противном случае он будет использовать все, что доступно:

```
# nice -15 /var/scripts/mybackup.sh
```

Если запуск вашего сценария является первоочередной операцией, которая должна завершиться как можно скорее, можете добавить второй дефис, чтобы назначить процессу отрицательное значение (`-15`), как в этом примере:

```
# nice --15 /var/scripts/mybackup.sh
```

В любом случае, если вы хотите увидеть это в действии, создайте сценарий и, пока он выполняется, войдите во второй терминал, где выполните `top`. Вы должны увидеть, что ваш процесс запущен, а его соответствующее значение `nice` должно появиться в столбце `NI`.

Для многих программ Linux вы также можете установить значение `nice` по умолчанию. `rsync`, по крайней мере в Ubuntu, позволяет вам явно указать значение `nice` с помощью параметра `RSYNC_NICE` в файле `/etc/default/rsync` (листинг 13.1).

Листинг 13.1. Возможная настройка nice в файле конфигурации /etc/default/rsync

```
RSYNC_NICE='10'
```

Вы также можете использовать `renice`, чтобы изменить поведение процесса даже после его запуска. Следующий пример при необходимости ограничит ресурсы, доступные процессу, которому в настоящее время присвоен PID 2145:

```
renice 15 -p 2145
```

Советы по поводу top

Если вам когда-либо понадобятся значения времени по ряду метрик ЦП, вы найдете их в третьей строке вывода `top` (в процентах) (см. рис. 13.1). Таблица 13.1 дает краткое объяснение сокращений, которые вы там увидите.

Таблица 13.1. Сокращения для метрик, связанных с процессором, отображаемых top

Метрика	Значение
us	Время выполнения высокоприоритетных (без nice) процессов
sy	Время выполнения процессов ядра
ni	Время выполнения низкоприоритетных (nice) процессов
id	Время на холостом ходу
wa	Время ожидания завершения событий ввода/вывода
hi	Время, потраченное на управление аппаратными прерываниями
si	Время, потраченное на управление программными прерываниями
st	Время, украденное у этой виртуальной машины ее гипервизором (хостом)

Обратите внимание, что вывод `top` можно настроить в режиме реального времени с клавиатуры. Нажмите клавишу `H`, чтобы узнать как.

13.1.3. Создание проблем (симуляция загрузки процессора)

Нужно попробовать парочку трюков, чтобы убедиться, что все идет гладко? Почему бы не смоделировать для себя кризисную перегрузку ЦП?

`yes` будет непрерывно производить (цифровой) шум, пока не получит команду остановиться. Следующая команда перенаправит этот шум в файл `/dev/null`. Символ амперсанда (`&`) означает, что процесс будет отправлен на задний план и вам будет предоставлена командная строка. Чтобы увеличить нагрузку, запустите команду еще несколько раз:

```
$ yes > /dev/null &
```

Это должно нагрузить систему. Пока все работает, используйте `top`, чтобы посмотреть, что происходит. Вы также можете попробовать запускать другие приложения, чтобы увидеть, сколько их потребуется, чтобы они начали работать с торможением. Когда вы закончите, запустите `killall`, чтобы сбросить сразу все сеансы `yes`:

```
$ killall yes
```

13.2. Проблемы с памятью

Несмотря на значительные достижения в области информационных технологий за последние годы, оперативная память (ОЗУ) используется так же, как и всегда. Компьютеры ускоряют вычисления, загружая ядра ОС и другой программный код в энергозависимые модули ОЗУ. Это обеспечивает быстрый доступ системы к часто запрашиваемым инструкциям программного обеспечения.

Самая большая проблема с памятью связана с ее ограничениями. В моем первом компьютере было 640 Кбайт ОЗУ (это меньше 1 Мбайт), и этого было недостаточно. Я оставлял свой компьютер включенным на ночь только для того, чтобы создать одно GIF-изображение размером 640 × 480 пикселов. Сейчас мой компьютер имеет 8 Гбайт оперативной памяти, и, как только я добавляю три виртуальные машины VirtualBox к своей обычной рабочей нагрузке, она тоже оказывается перегружена.

13.2.1. Оценка состояния памяти

Система с недостатком памяти не сможет выполнить нужные задачи или просто замедлится. Конечно, это может быть вызвано чем угодно, поэтому, прежде чем делать выводы, вам требуется подтверждение. Если хотите «посмотреть через электронный микроскоп» на модули памяти, подключенные к материнской плате, то воспользуйтесь командой `free`.

`free` анализирует файл `/proc/meminfo` и выводит общий объем доступной физической памяти (в данном примере — 7,1 Гбайт), а также информацию о том, как она

используется в настоящее время. `shared` — это память, используемая хранилищем tmpfs для поддержки различных псевдофайловых систем наподобие `/dev/` и `/sys/`. Буферы и кэш связаны с памятью, используемой ядром для операций ввода-вывода на уровне блоков (не беспокойтесь, если еще не понимаете, что это такое).

Любая память, используемая любым системным процессом, обозначается как `used`. `available` обозначает память, которая в настоящее время доступна для запуска новых приложений, даже если используется для кэширования на диске, без необходимости задействовать память подкачки (о которой мы поговорим). Вот пример:

```
$ free -h
              total        used        free      shared  buff/cache   available
Mem:           7.1G        2.8G        1.3G       372M        3.0G        3.5G
Swap:          7.2G        540K        7.2G
```

Как видите, я добавил аргумент `-h` к `free`, чтобы получить удобочитаемый вывод, где используются более простые для чтения большие числовые единицы (гигабайты, мегабайты и килобайты), а не байты.

Этот пример демонстрирует работоспособную систему с большими возможностями для роста. Но если значение `free` постоянно ближе к 0 и перенос нагрузки на своп (swap — «подкачка») не устраняет проблему, возможно, вам придется добавить память. Теперь об этом свопе...

13.2.2. Оценка состояния свопа

Поскольку модули ОЗУ, как правило, стоят дороже, чем дисковые хранилища, при многих установках ОС выделяется файл или раздел на диске для использования в качестве аварийного источника виртуальной оперативной памяти. Таким образом, даже если, строго говоря, у вас недостаточно оперативной памяти для всех запущенных процессов, перегруженная система не выйдет из строя, хотя будет работать заметно медленнее.

С помощью команды `vmstat` вы можете получить представление о том, как в вашей системе используется своп. Аргументы `30` и `4`, добавленные к этой команде, говорят программе возвращать четыре показания с 30-секундными интервалами между каждым чтением. В реальной ситуации вы, вероятно, захотите дольше проводить тестирование, чтобы повысить точность своих результатов. Два столбца, за которыми вы должны внимательно наблюдать, — это `si`, показывающий передачу данных из свопа в системную память, и `so`, где сообщается о переносах из системной памяти в своп. Как и было обещано, вот команда:

```
$ vmstat 30 4
procs -----------memory---------- ---swap-- -----io---- -system--  -----cpu--
r  b  swpd   free   buff  cache    si  so   bi   bo   in   cs us sy id wa st
0  0  540 1311400 373100 2779572    0   0   35   41  418  186  4  1 94  1  0
0  0  540 1311168 373104 2779540    0   0    0    9  671  881  0  0 99  0  0
0  0  540 1311216 373116 2779508    0   0    0   33  779 1052  1  1 98  0  0
0  0  540 1310564 373116 2779476    0   0    0    2  592  815  0  0 99  0  0
```

Если вы видите постоянное перемещение в своп и из него, вам следует подумать о добавлении физической оперативной памяти, если такая производительность представляет проблему для вашей работы.

13.3. Проблемы доступности запоминающего устройства

Если ваши прикладные программы регулярно записывают на накопители новые документы, данные или файлы журналов, нельзя игнорировать тот факт, что всегда есть ограничения на доступное пространство. То, что не может продолжаться вечно, в конце концов закончится. И эти записи данных прекратятся, когда на вашем диске не останется свободного места, сведя к нулю функциональность вашей системы.

Как можно определить, насколько близок предел? Легко. Вы уже встречали команду `df`. Поскольку устройства не используют фактическое дисковое пространство, вы можете игнорировать устройства, перечисленные в столбце `Use%` как использующие 0 % их максимального пространства. Вы уже знаете, что это псевдофайловые системы. Но нужно сосредоточиться на других и особенно на корневом разделе (`/`). В этом случае `root` по-прежнему владеет 686 Гбайт (почти 80 %) свободного пространства, поэтому прямо сейчас не стоит беспокоиться. Но, очевидно, это нужно регулярно проверять:

```
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            3.5G     0  3.5G   0% /dev
tmpfs           726M  1.5M  724M   1% /run
/dev/sda2       910G  178G  686G  21% /          <-- Запись о корневом разделе
tmpfs           3.6G   71M  3.5G   2% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           3.6G     0  3.6G   0% /sys/fs/cgroup  <-- Псевдофайловая система; обратите внимание, что использовано 0 байт
/dev/sda1       511M  3.4M  508M   1% /boot/efi
Tmpfs           726M   72K  726M   1% /run/user/1000
```

Здесь показан простой способ отследить использование дискового пространства. Трудности возникают, когда вы обнаруживаете, что не можете сохранить файл на диск, или входите в систему и получаете сообщение о том, что ваш сеанс доступен только для чтения.

Все мест о заполнено или ошибка диска?

Надо отметить, что не каждый сбой «только для чтения» является результатом заполненного накопителя. Это также может означать, что физическое устройство выходит из строя. В этом случае следует немедленно приступить к работе, сохраняя все важные данные на периферийных дисках или в учетных записях онлайн-хранилища, прежде чем диск полностью выйдет из строя. Как это можно распознать? Если ваша система достаточно работоспособна для запуска, должна помочь команда `df`, но если сомневаетесь — молитесь.

13.3.1. Ограничения inode

Физическое пространство не единственное ограничение для хранения данных в Linux. Вспомните наше обсуждение еще в главе 12, где мы отметили, что все объекты файловой системы Linux идентифицируются и управляются с помощью метаданных, содержащихся в уникальных inode. Оказывается, существует жесткое ограничение на количество inode, которые разрешены в системе, и возможно исчерпать это количество, даже если все еще достаточно свободного места физически.

ПРИМЕЧАНИЕ

Количество доступных inode задается при создании файловой системы. Учитывая, что сами inode занимают место, при создании файловой системы (например, с использованием такого инструмента, как mkfs.ext4) важно найти баланс, который позволит сохранять наибольшее количество файлов, расходуя наименьшее дисковое пространство.

Вот как выглядит запуск `df` в той же системе, что и раньше, но на этот раз с аргументом `-i` для отображения данных inode:

```
$ df -i
Filesystem     Inodes   IUsed     IFree  IUse% Mounted on
udev           914806     546    914260     1% /dev
tmpfs          928143     797    927346     1% /run
/dev/sda2    60547072  701615  59845457     2% /
tmpfs          928143     155    927988     1% /dev/shm
tmpfs          928143       5    928138     1% /run/lock
tmpfs          928143      16    928127     1% /sys/fs/cgroup
/dev/sda1           0       0         0     - /boot/efi
tmpfs          928143      31    928112     1% /run/user/1000
```

Это корневой раздел, тот, для которого статус inode является наиболее важным

С таким количеством свободных дескрипторов пока нет причины для тревоги, но вы определенно должны принять меры, если количество занятых будет 10 или 20 %. Один из моих серверов когда-то был перегружен inode, и потребовалось несколько минут, прежде чем я хотя бы понял, что произошло. Позвольте мне рассказать вам об этом подробнее.

Первый звоночек прозвенел, когда я попытался установить новое программное обеспечение, используя `apt`. Установка не удалась, появилось сообщение об ошибке «Нет свободного места на устройстве». Это было просто глупо, так как я знал, что были свободны гигабайты. Мне пришлось поискать информацию в Интернете, что побудило меня проверить свои уровни inode: конечно же, я был новичком в этом деле.

По логике, следующим шагом должен стать поиск каталогов, содержащих наибольшее количество файлов. В конце концов, концентрация inode, вероятно, будет больше в тех местах, где множество файлов. Вот как настроить поиск, чтобы получить такую информацию:

```
$ cd /
# find . -xdev -type f | cut -d "/" -f 2 | sort | uniq -c | sort -n
```

Таблица 13.2 объясняет, что все это значит.

Таблица 13.2. Синтаксис команды find

Синтаксис	Функция
.	Начать поиск с текущего каталога и ниже
-xdev	Оставаться в одной файловой системе
-type f	Поиск объектов типа file
cut -d "/"	Удалить текст, идентифицируемый разделителем (в данном случае символ /)
-f 2	Выбрать второе найденное поле
sort	Сортировать строки вывода и отправлять их в стандартный вывод (stout)
uniq -c	Посчитать количество строк, отправленных sort
sort -n	Вывести в числовом порядке

Это отличная команда. Проблема в том, что она не сработала, поскольку программа `find` временно сохраняет необработанные данные на диск. Но, поскольку у меня не было inode, спасти что-либо в настоящее время было невозможно. Потрясающе. Что теперь? Освободите немного места, найдя пару ненужных файлов и удалив их. После этого вот что показала мне `find`:

```
# find . -xdev -type f | cut -d "/" -f 2 | sort | uniq -c | sort -n
5 root
48 tmp
127 sbin
128 bin
377 boot
989 etc
2888 home
6578 var
15285 lib
372893 usr
```

Каталог /usr/ содержит больше всего файлов в одном дереве каталогов

ВНИМАНИЕ

Поскольку может потребоваться поиск по тысячам файлов и каталогов, приготовьтесь подождать некоторое время.

На данный момент наибольшее количество файлов было где-то в каталоге `/usr/`. Но где? Нет проблем, двигайтесь глубже и снова запускайте `find`:

```
$ cd usr
# find . -xdev -type f | cut -d "/" -f 2 | sort | uniq -c | sort -n
6 include
160 sbin
617 bin
7211 lib
16518 share
348381 src
```

Каталог, содержащий наибольшее количество файлов

На этот раз очевидным виновником является `/usr/src/`. Что происходит в `/usr/src/`? Оказывается, именно здесь хранятся заголовочные файлы ядра, включая те, что остались от предыдущих версий, установленных на вашем компьютере. Если вы тщательно рассмотрите дерево каталогов, то увидите, что файлов действительно много.

13.3.2. Решение

Чтобы освободить место, вам может потребоваться вручную удалить некоторые из старых каталогов. Затем, предполагая, что вы используете Ubuntu, позвольте `dpkg` безопасно удалить все, что не нужно, добавив параметр `--configure`:

```
# dpkg --configure -a
```

Чтобы безопасно удалить все старые заголовки ядра, запустите `autoremove`, и все вернется к оптимальному рабочему порядку:

```
# apt-get autoremove
```

В CentOS установите пакет yum-utils, а затем запустите `package-cleanup`. Добавление параметра `--count=2` позволит удалить все, кроме двух самых последних ядер:

```
# package-cleanup --oldkernels --count=2
```

СОВЕТ

Всегда полезно оставить хотя бы одно старое ядро на случай, если что-то будет не так с последним.

Что можно сделать для устранения ограничений по хранению? Наиболее очевидное действие — добавить больше пространства. Но вы также можете периодически проводить аудит системы, чтобы увидеть, что можно удалить или перенести на альтернативные и зачастую более дешевые решения для хранения данных. Amazon Glacier — отличное место для хранения больших объемов данных, к которым нечасто обращаются.

Кроме того, работайте над тем, чтобы уменьшить объем производимых данных. Как вы видели в главе 11, один из способов сделать это — убедиться, что ваши журналы регулярно ротируются и удаляются.

13.4. Проблемы с перегрузкой сети

Когда слова *«сеть»* и *«проблема»* объединяются в одном предложении, большинство людей, вероятно, в первую очередь думают о неудачных соединениях. Но об этом мы поговорим в следующей главе. Здесь же мы обсуждаем, как обращаться с активным и исправным соединением, которое перегружено.

Когда нужно начать подозревать, что нагрузка превышает возможности сети? На обычном настольном ПК вы, вероятно, увидите загрузки, которые занимают больше времени, чем должны, или вообще перестают работать; на общедоступном сервере ваши клиенты могут жаловаться на медленное обслуживание. Такие симптомы могут стать результатом сразу нескольких причин, поэтому стоит провести дальнейшие исследования, а затем попробовать возможные решения. Вот что вы узнаете дальше.

13.4.1. Измерение полосы пропускания

В Linux есть десятки инструментов для того, чтобы понять, как используется сеть. Два из них, которые я собираюсь вам показать, особенно полезны для быстрой идентификации ресурсов, наиболее требовательных к пропускной способности сети. Этот пример, в свою очередь, позволит вам понять основные проблемы.

Как и top, утилита iftop (полученная при установке пакета iftop по обычным каналам) отображает самообновляющуюся запись о самой большой сетевой активности, проходящей через сетевой интерфейс: например, `iftop -i eth0`.

Как видно из рис. 13.2, iftop показывает сетевые соединения между моим компьютером (рабочей станцией) и удаленными хостами, а также пропускную способность в байтах или килобайтах. Соединения перечислены по парам «вход/выход». Очевидно, что соединения с высоким потреблением нужно проверить и при необходимости удалить. Внизу экрана отображается совокупное и пиковое использование (как входящее, так и исходящее), а также средние показатели использования.

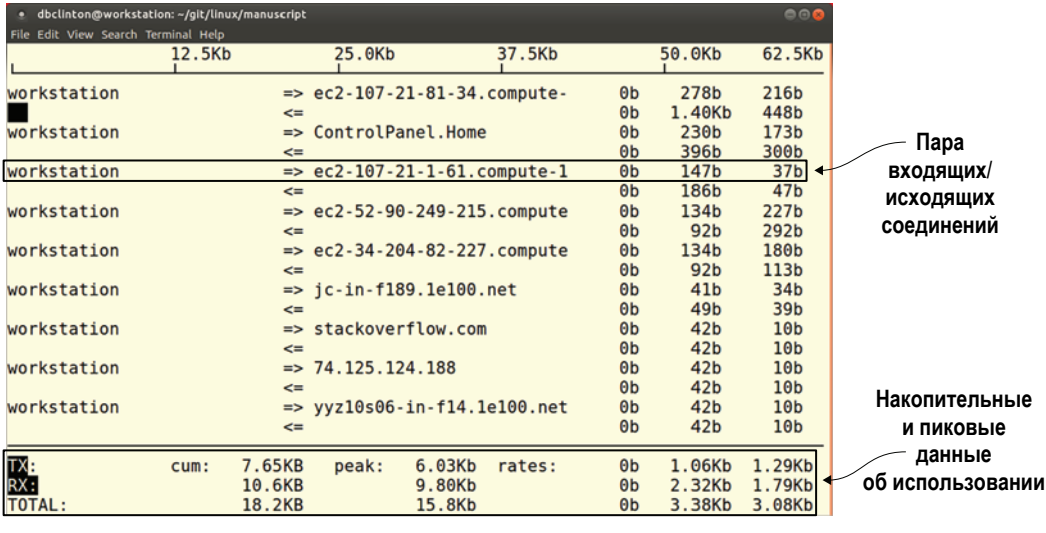


Рис. 13.2. Типичный вывод iftop сетевых подключений и использования ими полосы пропускания

Вы должны тщательно сканировать и анализировать удаленные хосты, которые идентифицирует iftop. Регулярный мониторинг позволит выявить неожиданные

потери ресурсов вашей сети и даже ранее неопознанное вредоносное ПО, которое iftop ловит «звонком домой». Звонки в Amazon или Google не должны насторожить вас, но странные и неясные URL-адреса нужно исследовать повторно.

iftop отлично подходит для уменьшения использования сети с точки зрения удаленного хоста (например, что будет полезно для устранения неполадок трафика браузера). Но иногда приходится управлять локальными процессами через их PID, и iftop не поможет с этим. NetHogs (установленный из пакета репозитория nethogs), с другой стороны, поможет. Запустите NetHogs из командной строки, указав сетевой интерфейс. Обратите внимание, что на этот раз вы не добавляете флаг `-i`:

```
# nethogs eth0
```

На рис. 13.3 показан типичный экран NetHogs на моей рабочей станции, включая PID для моего клиента Linux Slack и браузера Chrome. Если что-то пойдет не так, я смогу отследить это и контролировать через PID.

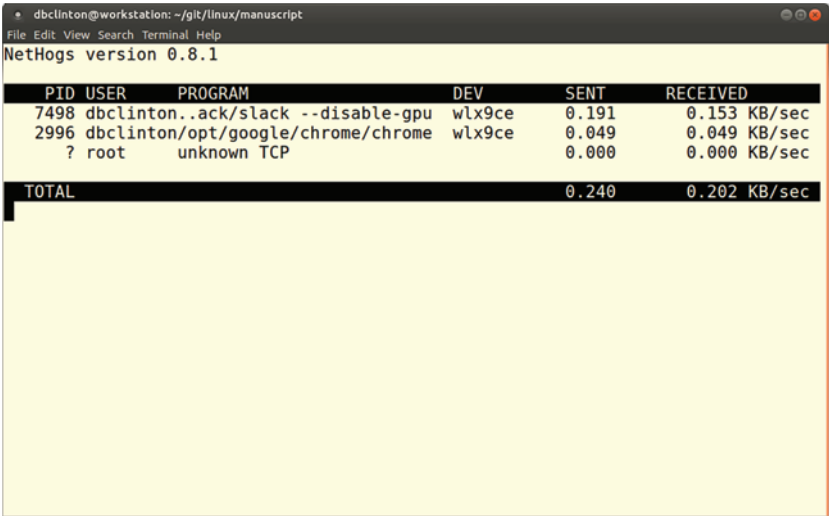


Рис. 13.3 Относительно тихий день на моей рабочей станции, как видно из NetHogs

13.4.2. Решения

После того как вы определили проблемный системный процесс, вам придется выяснить, как с ним бороться. Если это несущественный или мошеннический, вредоносный процесс, вы можете навсегда закрыть его, как делали ранее, используя `systemctl`, `kill` или `killall`. Но чаще всего это невозможно: вероятно, существуют причины, по которым большинство процессов запущены на ваших серверах.

Вы также можете рассмотреть возможность обновить вашу сетевую инфраструктуру. Вероятно, понадобится связаться с вашим интернет-провайдером для обсуждения повышения уровня предоставляемых услуг. В отношении локальных сетей вы также можете подумать о лучшем сетевом оборудовании. Если ваша

пропускная способность в настоящее время ограничена 100 Мбит/с (CAT 5), попробуйте обновить до 1000 Мбит/с (CAT 6). Помните, что недостаточно поменять кабели. Все маршрутизаторы, коммутаторы и интерфейсы, подключенные к вашим устройствам, также должны иметь эту пропускную способность, чтобы преимущества были ощутимы в полной мере. Переход на оптоволокно может обеспечить еще более высокую производительность, но за гораздо более высокую цену.

13.4.3. Формирование сетевого трафика с помощью команды tc

Более тонкое и изощренное решение проблем с нагрузкой — *формирование трафика*. Вместо того чтобы полностью закрывать определенные сервисы, вы можете, например, установить ограничение для пропускной способности процессов. В некотором смысле вы хотите управлять пропускной способностью так же, как ранее использовали `nice` для управления процессами. Это может позволить равномерно или стратегически распределить ограниченные ресурсы в вашей системе.

Например, можно ограничить пропускную способность, разрешенную для веб-клиентов, чтобы гарантировать, что другие процессы (скажем, обновления или резервные копии DNS) не будут ограничены пропускной способностью. В качестве быстрой иллюстрации я покажу вам, как установить вентиль на сетевом интерфейсе, имеющем доступ в Интернет с помощью инструмента Traffic Control (tc). Он обычно устанавливается в Linux по умолчанию.

Сначала отправьте пинг на удаленный сайт и запишите полученное время ответа. В этом примере время составляет в среднем около 37 миллисекунд:

```
$ ping duckduckgo.com
PING duckduckgo.com (107.21.1.61) 56(84) bytes of data.
64 bytes from duckduckgo.com (107.21.1.61):
      icmp_seq=1 ttl=43 time=35.6 ms
64 bytes from duckduckgo.com (107.21.1.61): icmp_seq=2 ttl=43 time=37.3 ms
64 bytes from duckduckgo.com (107.21.1.61): icmp_seq=3 ttl=43 time=37.7 ms
```

Значение времени представляет собой время, необходимое для одного прохода туда и обратно

Чтобы убедиться, что нет правил, связанных с вашим сетевым интерфейсом (в данном примере eth0), выведите все текущие правила:

```
$ tc -s qdisc ls dev eth0
qdisc noqueue 0: root refcnt 2
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
```

qdisc означает дисциплину в очереди — очередь, через которую должны проходить сетевые пакеты данных

Теперь добавьте правило, которое задерживает весь трафик на 100 миллисекунд. Это добавит 100 дополнительных миллисекунд к каждой передаче по сети, предоставляя другим процессам большую долю ресурсов, а также уменьшит сетевую активность, поэтому вы должны быть довольны результатами. Не волнуйтесь, через мгновение я покажу вам, как все это отменить:

```
# tc qdisc add dev eth0 root netem delay 100ms
```

Еще раз просмотрите правила tc и обратите внимание на новое правило, которое задерживает трафик на 100 миллисекунд:

```
$ tc -s qdisc ls dev eth0
qdisc netem 8001:
   root refcnt 2 limit 1000 delay 100.0ms
 Sent 514 bytes 3 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 102b 1p requeues 0
```

У qdisc теперь есть одно правило, задерживающее трафик на 100 мс

Проверьте свое правило. Запустите `ping` еще раз и просмотрите значения `time`. Теперь они должны быть больше примерно на 100 миллисекунд:

```
$ ping duckduckgo.com
PING duckduckgo.com (107.21.1.61) 56(84) bytes of data.
64 bytes from duckduckgo.com (107.21.1.61): icmp_seq=1 ttl=43 time=153 ms
64 bytes from duckduckgo.com (107.21.1.61): icmp_seq=2 ttl=43 time=141 ms
64 bytes from duckduckgo.com (107.21.1.61): icmp_seq=3 ttl=43 time=137 ms
```

Получилось так, как вы ожидали? Отлично. Возможно, вы захотите восстановить свою систему до ее исходного состояния, поэтому выполните эту команду, чтобы удалить правила, а затем снова протестируйте все, чтобы убедиться, что вы вернулись в нормальное состояние:

```
# tc qdisc del dev eth0 root
```

Утилита tc — сложная часть программного обеспечения, и полное руководство по ней может потребовать собственной книги. Но, я думаю, вы видели достаточно, чтобы почувствовать функциональность, которую она предлагает. Как и во всем остальном, моя задача — дать вам в руки инструменты, чтобы вы могли копать глубже и применять их в своих конкретных задачах.

13.5. Инструменты мониторинга

Как и в большинстве задач, связанных с администрированием, не забывайте запускать каждый из инструментов, обсуждаемых в этой главе. Некоторые задачи могут быть сценариями, управляться cron и программироваться для выдачи предупреждений при достижении заданных пороговых значений. Это здорово. Но ничто не заменит наблюдения за текущими данными хотя бы время от времени. Вот несколько идей, которые помогут вам создать эффективную систему мониторинга.

13.5.1. Агрегирование данных мониторинга

nmon — это многоцелевой инструмент для мониторинга и тестирования системы. Он предлагает вам настраиваемое единое представление различных сведений о состоянии всех компонентов системы. Вы можете приступить к работе, установив пакет nmon, а затем запустив команду `nmon` из командной строки. Первый экран, который вы увидите, будет выглядеть, как на рис. 13.4, и будет содержать подсказку по клавишам, используемым для переключения различных видов.

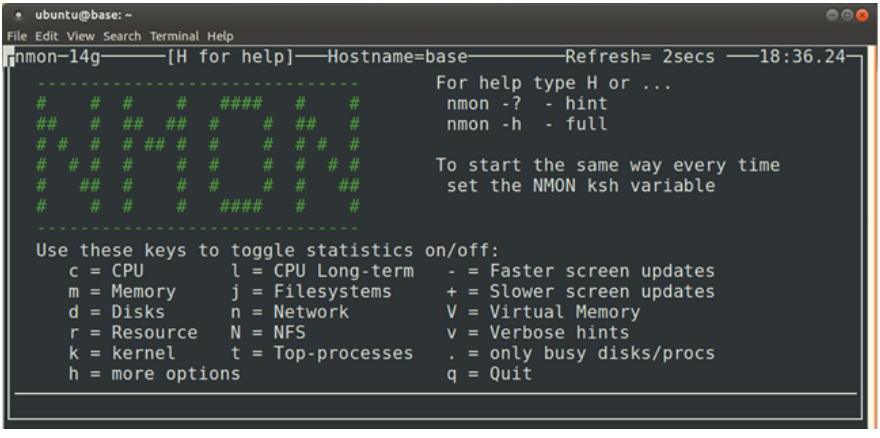


Рис. 13.4 Экран входа nmon с основными инструкциями по использованию

Ключевым моментом здесь является *переключение*, потому что, например, однократное нажатие клавиши **C** позволяет вывести информацию о процессоре, а повторное нажатие приводит к исчезновению этой информации. Нажатие **C**, **M** и **N** приведет к отображению данных о ЦП, памяти и сети одновременно, как вы можете видеть на рис. 13.5. Это отличный способ создать свой вид в едином окне, с помощью которого вы можете просматривать часто обновляемые параметры вашей системы. Кстати, нажатие **Q** приведет к выходу с экрана.

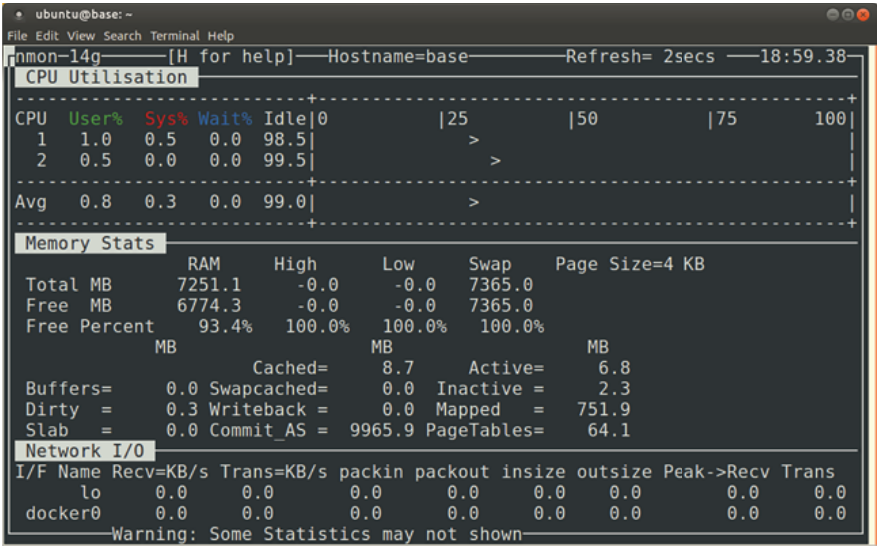


Рис. 13.5. Мониторинг нескольких сервисов на одном экране nmon

nmon охватывает только один сервер. Если вы отвечаете за всю группу машин, вам понадобится что-то более надежное, например Nagios или collectd, позволяющие отслеживать работоспособность и активность нескольких серверов, и Munin для отслеживания работы и предоставления анализа.

13.5.2. Визуализация ваших данных

Отлично. Но это все равно будет работать только тогда, когда вы действительно что-то ищете в реальном времени. Отправитесь на обед, и рискуете пропустить полномасштабную катастрофу. Удобно, что nmon позволяет записывать данные, которые он собирает с течением времени, в файл. В этом примере данные, собранные через каждые 30 секунд в течение часа (120 × 30 секунд), сохраняются в файле в текущем рабочем каталоге:

```
# nmon -f -s 30 -c 120
```

Если имя вашего хоста `ubuntu`, тогда имя файла по умолчанию будет состоять из `ubuntu`, метки даты и времени и расширения `.nmon`:

```
ubuntu_170918_1620.nmon
```

Если в вашей системе установлен веб-сервер, можете использовать инструмент nmonchart для преобразования файлов данных в более удобный для пользователя формат `.html`. Вам необходимо скачать инструмент с сайта nmon SourceForge (sourceforge.net/projects/nmon/files). Самый простой способ — щелкнуть правой кнопкой мыши на файле `nmonchartx.tar` (*x* будет номером версии) и скопировать URL-адрес. Из командной строки вашего сервера используйте wget для загрузки файла `tar`, а затем распакуйте архив обычным способом:

```
$ wget http://sourceforge.net/projects/nmon/files/nmonchart31.tar
$ tar xvf nmonchart31.tar
```

Вот как вы будете вызывать nmonchart для преобразования файла `.nmon` и его сохранения в корневом веб-каталоге:

```
# ./nmonchart ubuntu_170918_1620.nmon /var/www/html/datafile.html
```

Если вы столкнулись с ошибкой из-за отсутствия интерпретатора ksh, не стесняйтесь установить пакет ksh. Интерпретатор командной строки ksh является одной из альтернатив Bash:

```
-bash: ./nmonchart: /usr/bin/ksh: bad interpreter: No such file or directory
```

Когда все это будет сделано, можете указать в браузере IP-адрес вашего сервера, а затем имя файла, которое выбрали для nmonchart:

```
10.0.3.57/datafile.html
```

Вы должны увидеть страницу, похожую на рис. 13.6.

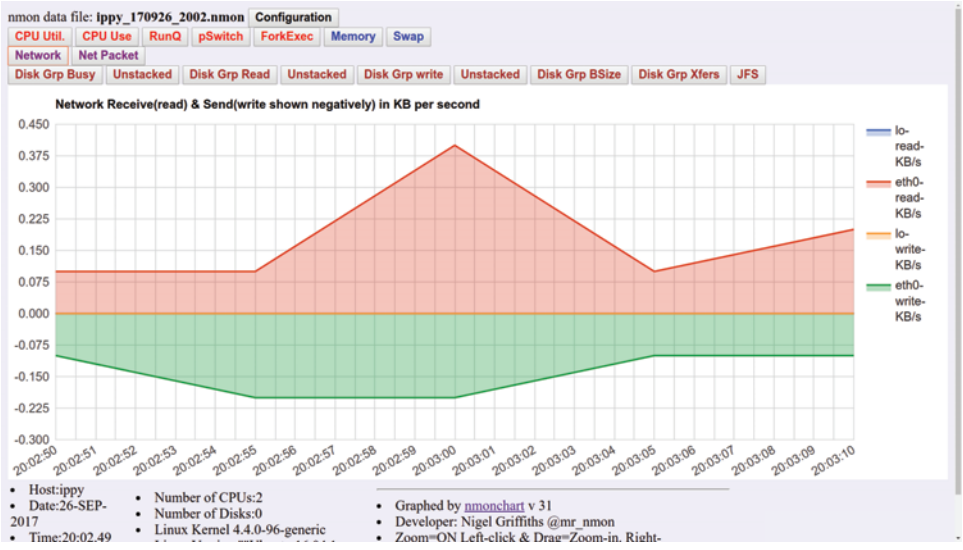


Рис. 13.6. Веб-страница, сгенерированная nmap

Вы можете поместить весь этот код в сценарий, включая код для генерации последовательных имен файлов `.html`. Так будет удобно отслеживать как новые, так и архивные события.

Резюме

- Инструмент `uptime` дает представление о средней загрузке ЦП с течением времени. Увидев в нем необычные результаты, вы сможете найти процессы с высокой нагрузкой и управлять ими.
- Команда `nice` позволяет тонко управлять тем, как процессы конкурируют друг с другом за ограниченные системные ресурсы.
- Фактическая память и память подкачки могут контролироваться с помощью таких инструментов, как `free` и `vmstat`.
- Ограничения для хранения определяются как доступным объемом физического диска, так и доступными индексными дескрипторами.
- iftop и NetHogs — два из десятков инструментов Linux для доступа к данным о сетевой нагрузке, а tc подойдет для контроля использования.
- Регулярный мониторинг возможен с применением инструментов командной строки, но включение их в сценарий или передача данных мониторинга в визуальном представлении в браузер гораздо эффективнее.

Ключевые термины

- *Загрузка ЦП* — это объем работы, выполняемой ЦП.
- *Использование ЦП* — доля используемой в настоящее время мощности ЦП.
- `nice` позволяет контролировать приоритет процесса при ограниченных ресурсах.
- *Оперативная память* содержит ядро ОС и другое важное программное обеспечение для возможности быстрого доступа к ключевым данным.
- *Память подкачки* — это пространство на жестком диске, обозначаемое как виртуальная память на тот случай, если у вас кончается реальная.
- *Индексные дескрипторы (inodes)* — метаданные, содержащие местоположение и другую информацию; связаны со всеми объектами файловой системы Linux.
- *Регулирование сетевого трафика* ограничивает полосу пропускания, разрешенную одному процессу, в пользу одного или нескольких процессов с более высоким приоритетом.

Рекомендации по безопасности

- Периодически сканируйте запущенные процессы, чтобы отследить вредоносное программное обеспечение, которое не было запущено аутентифицированными пользователями или вашей базовой системой.
- Запустив iftop, вы сможете увидеть удаленные сетевые хосты с подключениями к вашей системе. Если вы не распознаете соединение, вероятно, оно является несанкционированным и опасным.

Обзор команд

- `uptime` возвращает средние значения загрузки процессора за последние 1, 5 и 15 минут.
- `cat /proc/cpuinfo | grep processor` возвращает количество процессоров в системе.
- `top` отображает статистику в реальном времени для работающих процессов Linux.
- `killall yes` закрывает все запущенные экземпляры команды `yes`.
- `nice --15 /var/scripts/mybackup.sh` повышает приоритет потребления системных ресурсов для сценария `mybackup.sh`.
- `free -h` отображает общее и доступное пространство ОЗУ в системе.
- `df -i` отображает общее и доступное количество дескрипторов для каждой файловой системы.
- `find . -xdev -type f | cut -d "/" -f 2 | sort | uniq -c | sort -n` подсчитывает и выводит количество файлов в родительском каталоге.
- `apt-get autoremove` удаляет старые и неиспользуемые заголовки ядра.

- `nethogs eth0` отображает процессы и передает данные, связанные с сетевыми подключениями, использующими интерфейс eth0.
- `tc qdisc add dev eth0 root netem delay 100ms` замедляет все сетевые передачи через интерфейс eth0 на 100 миллисекунд.
- `nmon -f -s 30 -c 120` записывает в файл данные из серии сканирований `nmon`.

Самотестирование

1. Какой из следующих показателей загрузки ЦП может вызвать замедление работы двухъядерной системы:
   а) 1,7;
   б) 2,1;
   в) 0,17;
   г) 3,0?
2. Показатели нагрузки, отображаемые `uptime`, представляют собой средние значения за:
   а) 1, 10 и 25 минут;
   б) 1, 5 и 24 часа;
   в) 1, 5 и 15 минут;
   г) 10, 60 и 300 секунд.
3. Какой из следующих файлов будет содержать информацию о количестве процессоров в вашей системе:
   а) `/proc/uptime`;
   б) `/sys/cpuconf`;
   в) `/proc/cpuinfo`;
   г) `/etc/proc/envinfo`?
4. Какая из следующих команд завершит процесс mysqld с PID 4398:
   а) `kill mysqld`;
   б) `killall mysqld`;
   в) `killall 4398`;
   г) `kill mysqld:4398`?
5. Вы хотите уменьшить приоритет вашего сценария `mybackup.sh`. Какая из следующих команд понизит его приоритет больше всего:
   а) `nice -10 /var/scripts/mybackup.sh`;
   б) `nice -0 /var/scripts/mybackup.sh`;
   в) `nice --15 /var/scripts/mybackup.sh`;
   г) `nice -15 /var/scripts/mybackup.sh`?

6. Какой из следующих наборов симптомов скорее наводит на мысль о серьезной постоянной проблеме с памятью:
   а) низкая производительность приложений и высокий уровень передачи данных в своп и из него;
   б) низкая производительность приложений и низкий уровень передачи данных в своп и из него;
   в) низкая производительность приложений и высокая средняя загрузка ЦП;
   г) низкая производительность приложения и высокая доступность индексных дескрипторов?
7. Что из перечисленного предоставит PID вместе с данными о пропускной способности сети:
   а) `nmon -i eth0`;
   б) `nethogs eth0`;
   в) `nethogs -i eth0`;
   г) `iftop eth0`?
8. Как замедлить сетевой трафик через интерфейс eth0 на 100 миллисекунд:
   а) `tc qdisc add dev eth0 root netem delay 1000ms`;
   б) `tc qdisc add eth0 root netem delay 100ms`;
   в) `tc qdisc add dev eth0 root netem delay 100ms`;
   г) `tc qdisc add dev eth0 root netem -h delay 100ms`?

Ответы
1 — а; 2 — в; 3 — в; 4 — б; 5 — г; 6 — а; 7 — б; 8 — в.

14 Устранение неполадок в сети

В этой главе

- Использование сетевого взаимодействия TCP/IP для управления сетевыми проблемами.
- Устранение неполадок в сетях и сетевых интерфейсах.
- Управление подключением DHCP.
- Настройка DNS для трансляции адресов.
- Устранение неполадок при подключении к сети.

Когда я был мальчиком, приобретение нового программного обеспечения для ПК означало либо написать его самостоятельно, либо съездить в магазин и купить коробку с программой, хранящейся на одном или нескольких 5,25-дюймовых дискетах. Часто для удаленного сотрудничества требовались матричный принтер и почтовое отделение. Потоковое видео? Не смешите меня. Я не могу вспомнить, был ли на моем первом компьютере модем. Если и был, я, конечно, никогда не пользовался им.

В наши дни подключение к сети стало неотъемлемой частью вычислений, как клавиатура и крепкий кофе. И так как все больше используются голосовые интерфейсы, такие как Alexa компании Amazon, было бы неразумно вкладывать слишком большие средства в производство клавиатуры. (Тем не менее перспективы кофе выглядят все еще неплохо.) Суть в том, что вы и пользователи, которых вы поддерживаете, были бы довольно беспомощными без быстрого и надежного доступа к сети.

Чтобы обеспечить быстрый и надежный доступ, вам необходимо знать, как использовать сетевые инструменты и протоколы для установления соединения между

вашими сетевыми интерфейсами и внешним миром. Вам также нужно знать, как идентифицировать и подключать сетевые адаптеры к вашим компьютерам, чтобы у инструментов и протоколов было что-то, с чем можно работать. Мы еще поговорим об этом.

Но если вы планируете противостоять неприятным и непредвиденным сбоям, которые могут мешать вашему сетевому взаимодействию, вам сначала понадобятся глубокие практические знания об основах *стека интернет-протоколов*, часто называемых протоколом управления передачей (TCP) и межсетевым протоколом (IP) или TCP/IP для краткости. Технически TCP/IP вообще не является темой Linux, так как протоколы универсально используются всеми сетевыми устройствами, независимо от ОС. Поскольку работа, которую вы собираетесь выполнить в этой главе, не будет иметь большого смысла без учета TCP/IP, именно с него мы и начнем. Можете пропустить этот раздел, если уже владеете данной темой.

14.1. Понимание адресации TCP/IP

Основной единицей сети является простой IP-адрес, по крайней мере один из которых должен быть назначен каждому подключенному устройству. Каждый адрес должен быть уникальным во всей сети; в противном случае маршрутизация сообщений погрузится в хаос.

В течение десятилетий стандартный формат адресов следовал протоколу IPv4: по нему каждый адрес состоит из четырех восьмибитных октетов, что в сумме составляет 32 бита. (Не переживайте, если не понимаете, как считать в двоичном формате.) Каждый октет должен быть числом от 0 до 255. Вот типичный (ненастоящий) пример:

```
154.39.230.205
```

Максимальное теоретическое количество адресов, которые можно получить из пула IPv4, составляет более 4 миллиардов (256^4). Когда-то это казалось большим. Но так как Интернет разросся далеко за пределы любых ожиданий, очевидно, что в пуле IPv4 не будет достаточно уникальных адресов для всех бесчисленных устройств, которые хотят подключиться.

Четыре миллиарда возможных адресов кажется большим числом, но представьте, что в настоящее время используется более 1 миллиарда Android-смартфонов; это в дополнение ко всем миллионам серверов, маршрутизаторов, ПК и ноутбуков, не говоря уже о телефонах Apple. Есть большая вероятность, что у вашей машины, холодильника и камер для видеонаблюдения также есть собственные адреса, доступные по сети.

Было предложено два решения надвигающегося краха системы интернет-адресации (и конца обслуживания, как мы его знаем): IPv6 (совершенно новый протокол адресации) и механизм NAT (преобразование сетевых адресов). IPv6 предоставляет гораздо больший пул адресов, но, поскольку он все еще не так популярен, я остановлюсь на NAT.

14.1.1. Что такое адресация NAT

NAT основан на прекрасном принципе: вместо того чтобы назначать уникальный, понятный в сети адрес каждому из ваших устройств, почему бы не дать им всем один и тот же публичный адрес, который используется вашим маршрутизатором? Но как трафик будет поступать на ваши локальные устройства и с них? Благодаря использованию *частных* адресов. И если вы хотите разделить сетевые ресурсы на несколько подгрупп, как можно эффективно управлять всем? Через сегментацию сети. Ясно как день? Посмотрим, как работает NAT-адресация, чтобы лучше разобраться.

14.1.2. Работа с адресацией NAT

Когда браузер на одном из ноутбуков, подключенных к домашней сети Wi-Fi, посещает сайт, он делает это с общедоступного IP-адреса, назначенного модему/маршрутизатору DSL, предоставленному поставщиком услуг Интернета (ISP). Любые другие устройства, подключенные через ту же сеть Wi-Fi, используют тот же адрес для всех своих действий в Интернете (рис. 14.1).

В большинстве случаев маршрутизатор опирается на протокол динамической конфигурации хоста (DHCP) для назначения уникальных частных (NAT) адресов каждому локальному устройству, но они уникальны только в локальной среде. Таким образом, все локальные устройства могут пользоваться полной, надежной связью со своими локальными коллегами. Это работает так же хорошо и для крупных предприятий, многие из которых задействуют десятки тысяч IP-адресов NAT, и все они связаны с одним общедоступным IP-адресом.

Протокол NAT выделяет три диапазона адресов IPv4, которые могут использоваться только для частной адресации:

- от 10.0.0.0 до 10.255.255.255;
- от 172.16.0.0 до 172.31.255.255;
- от 192.168.0.0 до 192.168.255.255.

Администраторы локальных сетей могут свободно использовать любой из этих адресов (их более 17 миллионов) любым удобным для них способом. Но адреса обычно организованы в меньшие блоки сети (или *подсети*), хост-сеть которых идентифицируется октетами слева от адреса. Это оставляет октеты справа от адреса доступными для присвоения отдельным устройствам.

Например, вы можете выбрать для создания подсети адрес 192.168.1, что будет означать, что все адреса в этой подсети будут начинаться с 192.168.1 (сетевая часть адреса) и заканчиваться уникальным одноoктетным адресом устройства между 2 и 254. Таким образом, один ПК или ноутбук в этой подсети может получить адрес 192.168.1.4, а другой — 192.168.1.48.

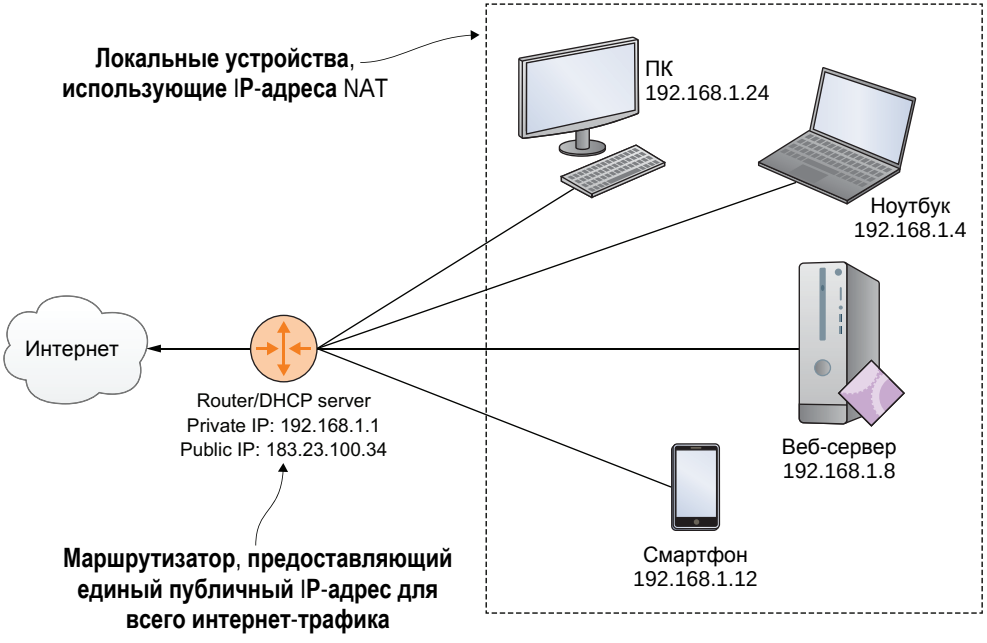


Рис. 14.1. Типичная конфигурация NAT, показывающая, как несколько локальных устройств, каждое из которых имеет собственный частный адрес, могут быть представлены одним общедоступным IP-адресом

ПРИМЕЧАНИЕ

Следуя соглашениям о сетевом взаимодействии, DHCP-серверы обычно не назначают сетевым устройствам номера 0, 1 и 255.

Продолжая в том же духе, вы можете в дальнейшем добавить следующую, но отдельную подсеть, используя 192.168.2. В этом случае 192.168.1.4 и 192.168.2.4 — два отдельных адреса, доступных для назначения двум различным устройствам. Кроме того, так как они находятся в разных сетях, они могут даже не иметь доступа друг к другу (рис. 14.2).

Обозначение подсети

Поскольку крайне важно убедиться, что системы знают, в какой подсети находится сетевой адрес, нам нужна стандартная запись, которая будет точно указывать, какие октеты являются частью сети, а какие доступны для устройств. Существует два широко используемых стандарта: *нотация бесклассовой междоменной маршрутизации (CIDR)* и *маска сети*. С использованием CIDR первая сеть в предыдущем примере будет представлена как 192.168.1.0/24. Часть /24 сообщает вам, что первые три октета (8 × 3 = 24)

составляют сетевую часть, оставляя лишь четвертый октет для адресов устройств. Вторая подсеть в CIDR будет описана как 192.168.2.0/24.

Эти же две сети также могут быть описаны с помощью сетевой маски 255.255.255.0. Такая запись означает, что все 8 бит каждого из первых трех октетов используются сетью, но из четвертого — ни один.

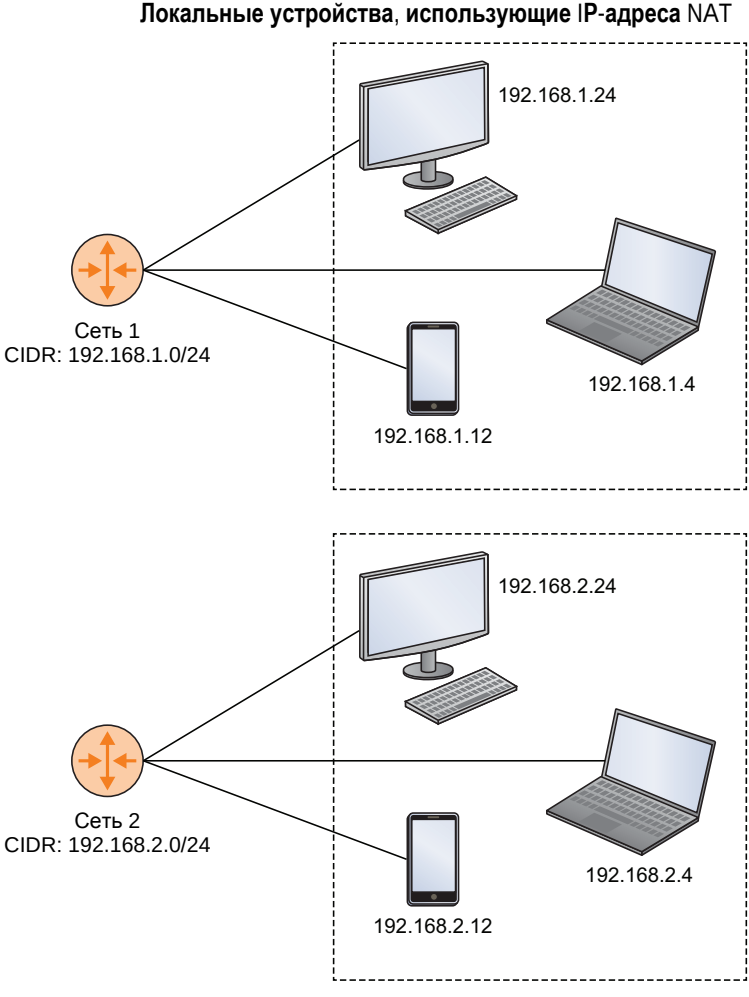


Рис. 14.2. Устройства, подключенные к двум отдельным подсетям NAT в диапазоне сети 192.168.x

Вам не нужно разбивать адресные блоки именно таким образом. Если вы знаете, что вряд ли когда-нибудь понадобится много подсетей в вашем домене, но ожидаете, что нужно будет подключить более 255 устройств, то можно указать только первые

два октета (192.168) в качестве сетевых адресов, оставляя все между 192.168.0.0 и 192.168.255.255 для устройств. В нотации CIDR это будет представлено как 192.168.0.0/16, а в виде маски сети — 255.255.0.0.

Кроме того, ваши сетевые части не должны использовать полные (восьмибитные) октеты. Часть диапазона, доступного в конкретном октете, может быть выделена для адресов, применяемых для целых сетей (например, 192.168.14.x), а оставшаяся часть оставлена для устройств (или хостов, как их чаще называют). Таким образом, вы можете выделить все адреса первых двух октетов подсети (192 и 168), а также некоторые адреса третьего октета (0) в качестве сетевых адресов. Это может быть представлено как 192.168.0.0/20, а в виде маски сети — 255.255.240.0.

Где я взял эти цифры? Большинство опытных администраторов в таких ситуациях вспоминают правила двоичных вычислений. Но в главе об общих проблемах с сетью мы не будем об этом говорить, так как эти сведения не нужны для стандартной работы. Тем не менее есть много онлайн-калькуляторов подсетей, которые все рассчитают за вас.

Почему вы хотите разделить свою сеть на подсети? Обычный сценарий подразумевает наличие групп ресурсов компании, которые должны быть доступны для одних команд (возможно, для разработчиков) и недоступны для других. Эффективный способ добиться этого — хранить их логически разделенными на собственные подсети.

14.2. Установление сетевого подключения

Все приходят на работу рано утром в понедельник. Коллеги обмениваются друг с другом короткими, но веселыми приветствиями, садятся за свои рабочие места, готовые к продуктивной работе, и обнаруживают, что Интернет недоступен. Источником недоступности сети может быть любой фактор из следующих.

- Сбой оборудования или операционной системы на локальном компьютере.
- Нарушение физических кабелей, маршрутизации или беспроводных соединений.
- Проблема с конфигурацией программного обеспечения локальной маршрутизации.
- Разбивка на уровне интернет-провайдера.
- Целая часть Интернета не работает.

В первую очередь нужно сузить поиск, исключив то, что не имеет значения. Проверку следует начать ближе всего к неисправности, чтобы удостовериться, что ошибка не в ваших собственных локальных системах, а затем постепенно расширяться дальше. Рисунок 14.3 иллюстрирует ход процесса.

Посмотрим, как все это может работать. Вы начнете с устранения проблем доступа с локальных компьютеров к внешним ресурсам, а затем рассмотрите проблемы с доступом к ресурсам на ваших серверах, которые могут испытывать внешние клиенты или пользователи.

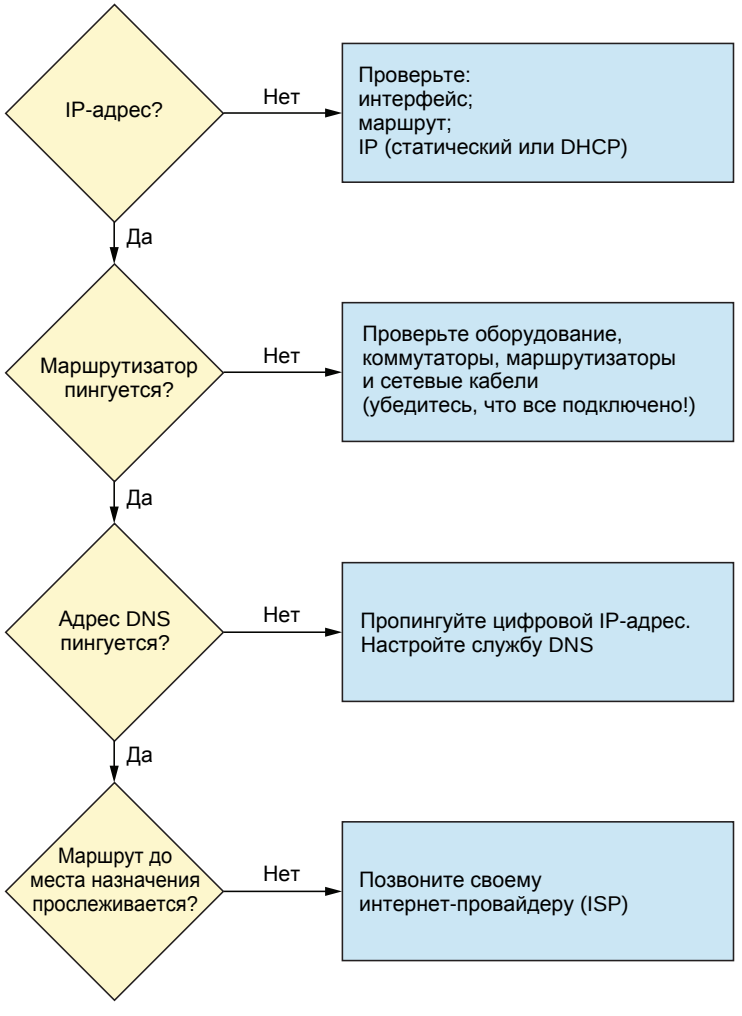
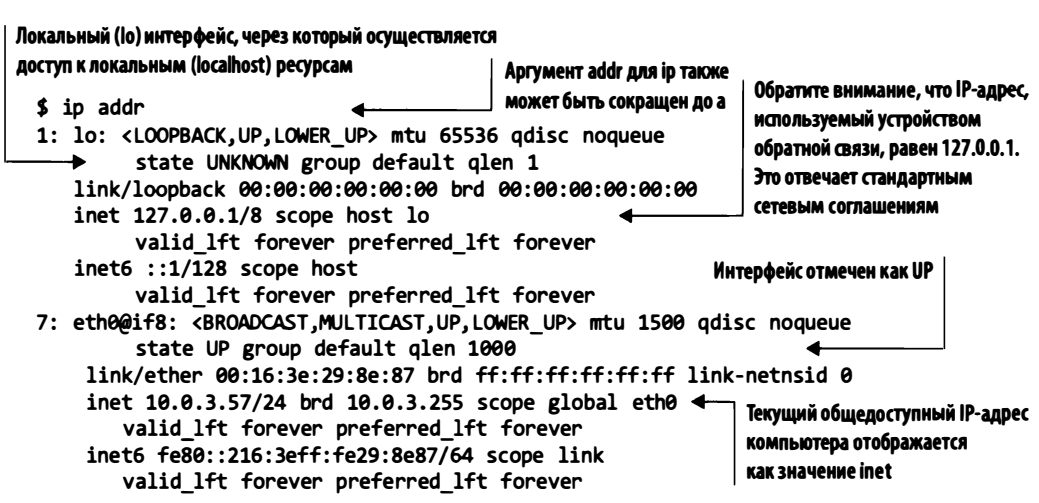


Рис. 14.3. Блок-схема, иллюстрирующая последовательность действий, на которую вы можете ориентироваться при устранении неполадок с исходящими подключениями

14.3. Устранение неполадок исходящего соединения

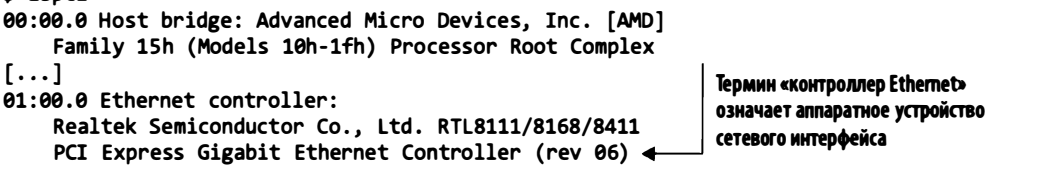
Вполне возможно, ваш компьютер так и не получил собственный IP-адрес, без которого невозможно успешно работать в сети. Запустите `ip`, чтобы отобразить устройства сетевых интерфейсов, а затем убедитесь, что у вас есть активное внешнее устройство и с ним связан действительный IP-адрес. Далее для интерфейса eth0 используется адрес 10.0.3.57:



Если в строке `inet` нет IP-адреса или в списке вообще нет сетевого интерфейса, то именно на этом вы и должны сосредоточить свое внимание.

14.3.1. Отслеживание статуса вашей сети

Сначала убедитесь, что на вашем компьютере физически установлен *сетевой адаптер* (также называемый *сетевой картой* или *сетевым интерфейсом*) и Linux видит его. Вы можете перечислить все оборудование PCI, установленное в данный момент, используя команду `lspci`. В следующем выводе обнаружен гигабитный Ethernet-контроллер PCI Express:



Если `lspci` не возвращает сетевые адаптеры, это может быть аппаратный сбой.

ПРИМЕЧАНИЕ

Peripheral Component Interconnect (PCI) — аппаратный стандарт, используемый для подключения периферийных устройств к микропроцессорам на материнских платах компьютеров через шину PCI. Существуют также различные новые стандарты, такие как PCI Express (PCIe), каждый из которых использует свой уникальный формфактор для физического подключения к материнской плате.

Помимо `lspci`, вы можете использовать такой инструмент, как `lshw`. Он выводит список сетевого оборудования, о котором знает ваша система. Сам по себе `lshw`

возвращает полный профиль оборудования, но `lshw -class network` покажет вам только подмножество устройств, которые относятся к сети. Попробуйте.

Положительный результат от `lspci` сам по себе много не даст, потому что он не говорит вам, как получить доступ к устройству из командной строки. Но он все же дает вам важную информацию. Возьмите, скажем, слово *Ethernet* из вывода `lspci` и используйте его с `grep` для поиска в выводе `dmesg`. Как вы помните из главы 11, `dmesg` — это записи событий, связанных с ядром и устройствами. После некоторого количества проб и ошибок я обнаружил, что этот конкретный поиск будет работать лучше всего, если включить две строки `dmesg`, следующие сразу за строкой, содержащей мою строку поиска (с использованием `-A 2`):

```
$ dmesg | grep -A 2 Ethernet
[    1.095265] r8169 Gigabit Ethernet driver 2.3LK-NAPI loaded
[    1.095840] r8169 0000:01:00.0 eth0 <1>: RTL8168evl/8111evl
        at 0xffffc90000cfa000, 74:d4:35:5d:4c:a5, XID 0c900800 IRQ 36
[    1.095842] r8169 0000:01:00.0
        eth0: jumbo features [frames: 9200 bytes, tx checksumming: ko]
```

Устройство eth0 отображается, как связанное с устройством Gigabit Ethernet

Получилось! Вы можете видеть, что устройству было присвоено обозначение eth0. Продолжайте. Не так быстро. Несмотря на то что eth0 изначально был задан устройству, потому что Linux теперь использует предсказуемые имена интерфейсов (см. главу 10), это может быть не то обозначение, которое фактически закреплено за интерфейсом. Просто для безопасности вам нужно еще раз выполнить поиск в выводе `dmesg`, чтобы увидеть, появляется ли eth0 где-нибудь еще:

```
$ dmesg | grep eth0
[    1.095840] r8169 0000:01:00.0
        eth0: RTL8168evl/8111evl at 0xffffc90000cfa000, 74:d4:35:5d:4c:a5,
        XID 0c900800 IRQ 36
[    1.095842] r8169 0000:01:00.0
        eth0: jumbo features [frames: 9200 bytes, tx checksumming: ko]
[    1.129735] r8169 0000:01:00.0 enp1s0:
        renamed from eth0
```

Обозначение eth0 было отброшено и заменено enp1s0

Ага! Похоже, что в какой-то момент процесса загрузки устройство было переименовано в enp1s0. Ладно. У вас есть правильно настроенный сетевой интерфейс, но по-прежнему нет IP-адреса и сетевого подключения, что дальше? Понадобится команда `dhclient`, но сначала немного предыстории.

14.3.2. Назначение IP-адресов

Сетевые устройства могут получить свои IP-адреса следующими способами.

- Кто-то вручную задает статический адрес, который (надеюсь) попадает в диапазон адресов локальной сети.
- DHCP-сервер автоматически дает устройству неиспользуемый адрес из пула.

Как обычно бывает, у каждого подхода есть свои недостатки. DHCP-серверы выполняют свою работу автоматически и незаметно и гарантируют, что два управляемых устройства никогда не будут пытаться занять один и тот же адрес. Но, с другой стороны, эти адреса являются динамическими, то есть адреса, которые они используют в один день, могут быть не теми, которые они получат на следующий. Например, если вы успешно использовали, скажем, 192.168.1.34 для SSH на удаленном сервере, будьте готовы к неожиданным изменениям.

И наоборот, установка IP-адресов вручную гарантирует, что они будут постоянно связаны с их устройствами. Но всегда есть вероятность, что возникнет конфликт — с непредсказуемыми результатами. Как правило, если у вас нет особой потребности в статическом адресе (возможно, вам нужен надежный удаленный доступ к ресурсу с использованием адреса), лучше выбрать DHCP.

Определение сетевого маршрута

Прежде чем искать адрес, вы должны убедиться, что Linux в первую очередь знает, как найти сеть. Если Linux уже может видеть свой путь к работающей сети, то команда `ip route` покажет вам таблицу маршрутизации вашего компьютера, включая локальную сеть и IP-адрес устройства, которое вы будете использовать в качестве шлюза-маршрутизатора:

```
$ ip route
default via 192.168.1.1
    dev enp0s3 proto static metric 100
192.168.1.0/24 dev enp0s3 proto kernel scope
    link src 192.168.1.22 metric 100
```

Адрес шлюза-маршрутизатора, через который локальный компьютер будет получать доступ к остальной сети

Сеть NAT (192.168.1.x) и маска сети (/24) локальной сети NAT

Если рабочего маршрута в списке нет, вам нужно его создать, но сначала придется определить диапазон подсетей вашей локальной сети. Если есть другие компьютеры, использующие ту же сеть, проверьте их IP-адреса. Если, скажем, один из этих компьютеров зарезервировал 192.168.1.34, то, скорее всего, адрес маршрутизатора будет 192.168.1.1. Аналогично, если IP-адрес этого подключенного компьютера — 10.0.0.45, то адрес маршрутизатора будет 10.0.0.1. Понимаете? Нам нужна команда `ip` для создания нового маршрута по умолчанию к вашему шлюзу:

```
# ip route add default via 192.168.1.1 dev eth0
```

ПРИМЕЧАНИЕ

Команды ip, обсуждаемые в этой главе, являются относительно новыми и предназначены для замены устаревших наборов команд, таких как ifconfig, route и ifupdown. Вы еще встретите множество практических руководств, ориентированных на эти старые команды, и по крайней мере на данный момент они по-прежнему будут работать, но постарайтесь привыкнуть к использованию ip.

Запрос динамического адреса

Лучший способ запросить адрес DHCP — использовать `dhclient` для поиска сервера DHCP в вашей сети, а затем запросить динамический адрес. Вот как это может выглядеть, если предположить, что ваш внешний сетевой интерфейс называется enp0s3:

```
# dhclient enp0s3
Listening on LPF/enp0s3/08:00:27:9c:1d:67
Sending on   LPF/enp0s3/08:00:27:9c:1d:67
Sending on   Socket/fallback
DHCPDISCOVER on enp0s3 to 255.255.255.255
   port 67 interval 3 (xid=0xf8aa3055)
DHCPREQUEST of 192.168.1.23 on enp0s3 to 255.255.255.255
   port 67 (xid=0x5530aaf8)
DHCPOFFER of 192.168.1.23 from 192.168.1.1
DHCPACK of 192.168.1.23 from 192.168.1.1
RTNETLINK answers: File exists
bound to 192.168.1.23 -- renewal in 34443 seconds.
```

Адрес DHCP-сервера в этом случае — 192.168.1.1

Новый адрес успешно дан вам в аренду на определенное время; продление будет автоматическим

Конфигурирование статического адреса

Вы можете временно присвоить интерфейсу статический IP-адрес из командной строки с помощью команды `ip`, но он будет действовать только до следующей загрузки системы. Вот как это делается:

```
# ip addr add 192.168.1.10/24 dev eth0
```

Это отлично подходит для быстрых одноразовых конфигураций, возможно, при попытке установить соединение в неисправной системе в случае устранения неполадок. Но, как правило, все предпочитают, чтобы изменения были постоянными. На компьютерах с Ubuntu нужно будет отредактировать файл `/etc/network/interfaces`. Он может уже содержать раздел, определяющий ваш интерфейс как DHCP, а не как статический (листинг 14.1).

Листинг 14.1. Раздел в файле /etc/network/interfaces

```
auto enp0s3
iface enp0s3 inet dhcp
```

Вы можете отредактировать этот раздел, изменив `dhcp` на `static`, введя желаемый IP-адрес, маску сети (в формате x.x.x.x) и IP-адрес сетевого шлюза (маршрутизатора), который будет использовать компьютер. Вот пример:

```
auto enp0s3
iface enp0s3 inet static
    address 192.168.1.10
    netmask 255.255.255.0
    gateway 192.168.1.1
```

В CentOS каждый интерфейс будет иметь собственный файл конфигурации в каталоге `/etc/sysconfig/network-scripts/`. Стандартный текст интерфейса с адресацией DHCP будет выглядеть так, как показано в листинге 14.2.

Листинг 14.2. Конфигурации в файле /etc/sysconfig/network-scripts/ifcfg-enp0s3

```
TYPE="Ethernet"
BOOTPROTO="dhcp"          <-- Инструктирует Linux запрашивать динамический IP для интерфейса
DEFROUTE="yes"
PEERDNS="yes"
PEERROUTES="yes"
IPV4_FAILURE_FATAL="no"
IPV6INIT="yes"
IPV6_AUTOCONF="yes"
IPV6_DEFROUTE="yes"
IPV6_PEERDNS="yes"
IPV6_PEERROUTES="yes"
IPV6_FAILURE_FATAL="no"
NAME="enp0s3"
UUID="007dbb43-7335-4571-b193-b057c980f8d0"
DEVICE="enp0s3"
ONBOOT="yes"
```

Листинг 14.3 показывает, как этот файл может выглядеть после того, как вы отредактируете его, чтобы разрешить статическую адресацию.

Листинг 14.3. Статическая версия файла конфигурации интерфейса CentOS

```
BOOTPROTO=none            <-- DHCP-адресация не будет использоваться
NETMASK=255.255.255.0
IPADDR=10.0.2.10          <-- Задает статический IP-адрес, который вы хотите использовать
USERCTL=no
DEFROUTE="yes"
PEERDNS="yes"
PEERROUTES="yes"
IPV4_FAILURE_FATAL="no"
IPV6INIT="yes"
IPV6_AUTOCONF="yes"
IPV6_DEFROUTE="yes"
IPV6_PEERDNS="yes"
IPV6_PEERROUTES="yes"
IPV6_FAILURE_FATAL="no"
NAME="enp0s3"
UUID="007dbb43-7335-4571-b193-b057c980f8d0"
DEVICE="enp0s3"
ONBOOT="yes"
```

Если вы хотите, чтобы ваши настройки вступили в силу немедленно, нужно будет перезагрузить сеть. В основном сетью в современных системах управляет служба systemd, NetworkManager. Вместо этого, по крайней мере в Ubuntu, запуск или остановка интерфейсов, которые определены в файле `/etc/network/interfaces`, обрабатывается сетевой службой. Поэтому, если вы хотите применить вновь

отредактированные настройки в файле интерфейсов, запустите команду `systemctl restart network`, а не `systemctl restart NetworkManager`. В качестве альтернативы вы можете использовать `ip`, чтобы включить (или выключить) только один интерфейс:

```
# ip link set dev enp0s3 up
```

Не помешает знать, что в некоторых местах вашей системы NetworkManager скрывает свои рабочие файлы. В каталоге `/etc/NetworkManager/` есть файл конфигурации с именем `NetworkManager.conf`, файлы конфигурации для каждого сетевого подключения, которые устанавливал ваш компьютер за все время работы, — в `/etc/NetworkManager/system-connections/`, данные, подробно описывающие подключения DHCP вашего компьютера, — в `/var/lib/NetworkManager/`. Почему бы не взглянуть на каждый из этих ресурсов?

14.3.3. Конфигурирование службы DNS

Если у вас есть действующий сетевой маршрут и IP-адрес, но проблема с подключением не исчезла, вам придется немного расширить поиск. Задумайтесь на минуту о том, что именно вы не можете сделать.

Ваш браузер не может загружать страницы? Возможно, у вас нет подключения. Это также может означать, что DNS не работает.

Что такое DNS

Можете мне не верить, но Всемирная паутина действительно основана на цифрах. Там нет места названиям типа manning.com или wikipedia.org. Скорее, это 35.166.24.88 и 208.80.154.224 соответственно. Программное обеспечение, которое выполняет всю работу по соединению нас с сайтами, которые мы знаем и любим, распознает только числовые IP-адреса.

Инструмент, который переводит текст то на язык, понятный людям, то на язык более ориентированных на цифры компьютеров, называется *системой доменных имен* (DNS). Слово *«домен»* часто используется для описания отдельной группы сетевых ресурсов, в частности ресурсов, идентифицируемых уникальным, понятным человеку именем. Как показано на рис. 14.4, при вводе текстового адреса в браузере будут запрашиваться службы DNS-сервера.

Как работает DNS

Первой остановкой обычно является локальный индекс имен и связанных с ними IP-адресов, которые хранятся в файле, автоматически создаваемом ОС на вашем компьютере. Если у этого локального индекса нет ответа на конкретный вопрос о трансляции имени в цифровой адрес, он перенаправляет запрос на назначенный общедоступный DNS-сервер, который поддерживает гораздо более полный индекс и может подключить вас к нужному сайту. К общедоступным DNS-серверам относятся те, которые предоставляются Google (а он использует очень простые адреса — 8.8.8.8 и 8.8.4.4) и OpenDNS.

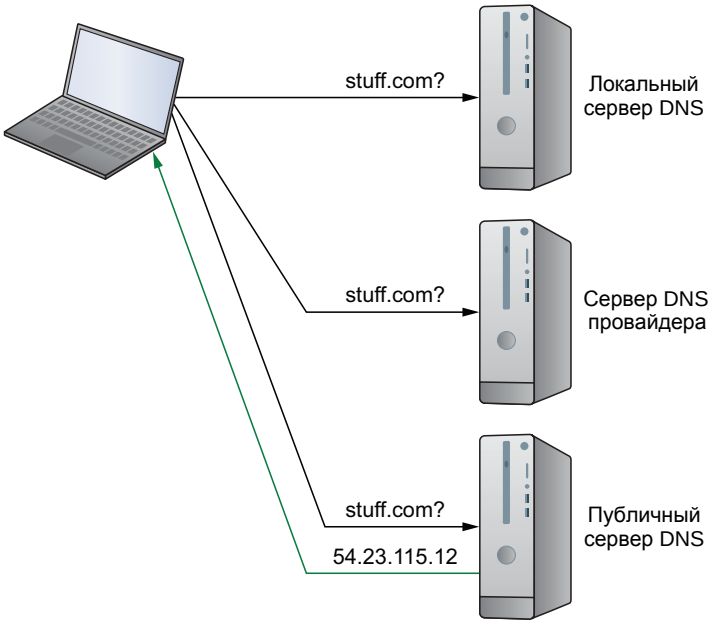


Рис. 14.4. Запрос адреса в DNS для stuff.com и ответ, содержащий (вымышленный) IP-адрес

Решение проблем с DNS

Пока что-то не сломается, вы вряд ли будете тратить много времени на размышления о DNS-серверах. Если же проблема появится, проверьте ее источник с помощью инструмента `ping`. Если проверка обычного URL-адреса сайта (например, manning.com) не работает, а с использованием IP-адреса работает, значит, вы нашли свою проблему. Вот как это может выглядеть:

```
$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=60 time=10.3 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=60 time=10.2 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=60 time=9.33 ms
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 9.339/10.002/10.378/0.470 ms
```

Этот символ говорит о том, что сочетание клавиш Ctrl+C было нажато для прерывания операции ping

Как исправить? Здесь много нюансов. В большинстве случаев отдельные компьютеры наследуют настройки DNS от маршрутизатора, через который они подключаются к более широкой сети. Я полагаю, следующие несколько минут вы потратите на поиск в ящиках стола, чтобы восстановить пароль для входа в ваш

маршрутизатор (подсказка: пароль по умолчанию часто указан на корпусе самого маршрутизатора).

После того как вам удастся войти в систему (обычно с подключенного ПК, на котором запущен браузер, указывающий на IP-адрес маршрутизатора), просмотрите меню операционной системы маршрутизатора, чтобы убедиться, что настройки DNS действительны. Вы также можете настроить параметры DNS на локальном компьютере, которые будут переопределять те, что диктует маршрутизатор. На компьютере с CentOS добавьте ссылки на пару общедоступных DNS-серверов в файл вашего интерфейса `/etc/sysconfig/network-scripts/ifcfg-enp0s3`. В этом примере используются два IP-адреса, связанные с DNS-серверами Google:

```
DNS1=8.8.8.8 DNS2=8.8.4.4
```

А в Ubuntu добавьте значения `dns-nameserver` в соответствующий интерфейс в файлах `/etc/network/interfaces`:

```
dns-nameserver 8.8.8.8
dns-nameserver 8.8.4.4
```

14.3.4. Обслуживание сети

Итак, настало время засучить рукава и сделать самую грязную работу. Если у вас есть рабочий интерфейс, маршрут, IP-адрес и служба DNS, но по-прежнему нет полной связи, значит, что-то блокирует поток.

В Linux есть такой отличный инструмент, как Traceroute, который отслеживает маршрут прохождения пакета через сеть на пути к цели. Если что-нибудь блокирует трафик в любом месте на линии, Traceroute по крайней мере покажет вам, где находится засор. Даже если у вас нет возможности проводить дальнейшие расследования, информация может оказаться очень ценной, поскольку ваш провайдер пытается все восстановить.

В этом примере показано успешное прохождение между моей домашней рабочей станцией и google.com (то есть 172.217.0.238). Если бы что-то пошло не так, показанные переходы прекратились бы до достижения цели. Строки вывода, не содержащие ничего, кроме звездочек (*), могут иногда соответствовать устройствам, которые не могут вернуть пакет. Полный сбой обычно сопровождается сообщениями об ошибках:

Первый переход — мой локальный маршрутизатор; отображаемое время перехода ~ 20 мс — немного медленно, но приемлемо

```
$ traceroute google.com
traceroute to google.com (172.217.0.238), 30 hops max, 60 byte packets
 1 ControlPanel.Home (192.168.1.1)
      21.173 ms 21.733 ms 23.081 ms
 2 dsl-173-206-64-1.tor.primus.ca (173.206.64.1)
      25.550 ms 27.360 ms 27.865 ms
 3 10.201.117.22 (10.201.117.22) 31.185 ms 32.027 ms 32.749 ms
```

Мой интернет-провайдер

```
4 74.125.48.46 (74.125.48.46) 26.546 ms 28.613 ms 28.947 ms
5 108.170.250.241 (108.170.250.241) 29.820 ms 30.235 ms 33.190 ms
6 108.170.226.217 (108.170.226.217)
      33.905 ms 108.170.226.219 (108.170.226.219) 10.716 ms 11.156 ms
7 yyz10s03-in-f14.1e100.net (172.217.0.238) 12.364 ms * 6.315 ms
```

Все еще ничего? Похоже, пришло время позвонить вашему провайдеру.

14.4. Устранение неполадок при входящем соединении

В вашей инфраструктуре всегда будут объекты, которые вы захотите сделать доступными в режиме 24/7, будь то сайт вашей компании, API, поддерживающий ваше приложение, или внутренняя вики-документация. В итоге входящие подключения могут быть так же важны для вашего бизнеса или организации, как и исходящий трафик, который мы только что обсуждали.

Если ваши удаленные клиенты не могут подключиться к вашим сервисам или их соединения слишком медленные, ваш бизнес пострадает. Поэтому важно регулярно убеждаться, что ваше приложение исправно и слушает входящие запросы, что эти запросы доходят туда, куда необходимо, и что пропускная способность достаточна для обработки всего трафика. В этом могут помочь команды `netstat` и `netcat`.

14.4.1 Сканирование внутреннего соединения: netstat

Запуск `netstat` на сервере позволяет получить множество сведений о сети и интерфейсе. Однако больше всего вас заинтересует список служб, которые прослушивают сетевые запросы.

`netstat -l` покажет вам все открытые сокеты. Если вы работаете с сайтом, можете сузить результаты, отфильтровав их по http. В этом случае оба порта, 80 (http) и 443 (https), кажутся активными:

```
$ netstat -l | grep http
tcp6 0  0 [::]:http   [::]:*  LISTEN
tcp6 0  0 [::]:https  [::]:*  LISTEN
```

Протокол показан как tcp6: предполагается, что это исключительно сервис Ipv6. На самом деле он охватывает как IPv6, так и IPv4

Что такое сетевой сокет на самом деле

Если честно, я не уверен на 100 %, что смогу правильно это описать. То, что это будет означать для программиста на C, может показаться странным для обычного системного администратора, такого как ваш покорный слуга. Тем не менее я рискну упростить и скажу, что конечная точка службы определяется IP-адресом сервера и портом (например, 192.168.1.23:80). Данная комбинация идентифицирует сетевой сокет. Соединение создается во время сеанса с участием двух конечных точек/сокетов (клиент и сервер).

`netstat -i` выведет список ваших сетевых интерфейсов. На первый взгляд, это не кажется таким уж большим делом; в конце концов, `ip addr` делает то же самое, верно? Но `netstat` также покажет вам, сколько пакетов данных было получено (`RX`) и передано (`TX`). `OK` означает безошибочную передачу; `ERR` — поврежденные пакеты; `DRP` — пакеты, которые были отброшены. Эта статистика может быть полезна, если вы не уверены, что служба активна:

```
$ netstat -i
Kernel Interface table
Iface   MTU Met RX-OK RX-ERR  RX-DRP RX-OVR TX-OK TX-ERR Flg
enp1s0     1500 0       0       0       0 0      0      0 BMU
lo        65536 0   16062       0       0 0  16062      0 LRU
wlx9cefd5fe6a19  1500 0 1001876 0       0 0 623247      0 BMRU
```

В примере демонстрируется работоспособный и занятый беспроводной интерфейс (с неудачным именем `wlx9cefd5fe6a19`) и неиспользуемый интерфейс enp1s0. В чем дело? Это ПК с неиспользуемым портом Ethernet, который получает доступ в Интернет через Wi-Fi. В коде `lo` — интерфейс localhost, также известный как 127.0.0.1. Это отличный способ оценить ситуацию изнутри сервера, но как все выглядит со стороны?

14.4.2. Сканирование внешнего соединения: netcat

Вы использовали `cat` для потоковой передачи текстовых файлов и `zcat` для потоковой передачи содержимого сжатых архивов. Теперь пришло время познакомиться с другим членом семейства (кошачьих): `netcat` (часто называемым `nc`). Как вы можете догадаться по его названию, `netcat` может использоваться для потоковой передачи файлов по сети или даже в качестве простого приложения для двустороннего чата.

Но сейчас вас больше интересует состояние вашего сервера и, в частности, то, как его увидит клиент. При запуске по удаленному адресу `nc` сообщает, удалось ли установить соединение. `-z` ограничивает вывод `netcat` результатами сканирования для прослушивания демонов (вместо того чтобы пытаться установить соединение), а `-v` добавляет подробности к выводу. Вам нужно указать порт или порты, которые вы хотите сканировать. Вот пример:

```
$ nc -z -v bootstrap-it.com 443 80
Connection to bootstrap-it.com 443 port [tcp/https] succeeded!
Connection to bootstrap-it.com 80 port [tcp/http] succeeded!
```

Если одна или обе эти службы (HTTP и HTTPS) недоступны, сканирование завершится неудачно. Это может быть связано с тем, что служба не запущена на сервере (возможно, ваш веб-сервер Apache остановился) или существует слишком строгое правило брандмауэра, блокирующее доступ. Вот как будет выглядеть неудачное сканирование:

```
$ nc -z -v bootstrap-it.com 80
nc: connect to bootstrap-it.com port 80 (tcp) failed: Connection timed out
```

Однако это Linux, так что вы можете быть уверены, что точно есть несколько хороших способов выполнить такую работу. Поэтому имейте в виду, что для аналогичного сканирования можно использовать `nmap`:

```
$nmap -sT -p80 bootstrap-it.com
Nmap scan report for bootstrap-it.com (52.3.203.146)
Host is up (0.036s latency).
PORT   STATE SERVICE
80/tcp open  http
Nmap done: 1 IP address (1 host up) scanned in 0.37 seconds
```

Команда `nmap` будет сканировать любые открытые порты между портами 1 и 1023, а это отличный способ быстро проверить вашу систему, чтобы убедиться, что не открыт никакой лишний порт:

```
$ nmap -sT -p1-1023 bootstrap-it.com
Nmap scan report for bootstrap-it.com (52.3.203.146)
Host is up (0.038s latency).
Not shown: 1020 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open https
Nmap done: 1 IP address (1 host up) scanned in 4.69 seconds
```

Какие порты «должны быть» открыты? Это зависит от программного обеспечения, которое вы используете на сервере. Если `nmap` сообщает о незнакомых открытых портах, поищите в Интернете, чтобы выяснить, какое программное обеспечение задействует эти порты, а затем спросите себя, разумно ли, чтобы оно работало на вашем сервере.

Резюме

- Linux определяет сетевые интерфейсы и маршруты и управляет ими в контексте сетевых протоколов NAT.
- Linux должна распознавать подключенные аппаратные периферийные устройства, такие как сетевые интерфейсы, а также назначать метки устройств (например, eth0), прежде чем их можно будет использовать.
- Пользовательские статические IP-адреса могут быть назначены устройству как путем редактирования файлов конфигурации, так и из командной строки (с помощью `ip`). Динамический адрес может автоматически запрашиваться с DHCP-серверов, но вы не можете управлять полученными адресами.
- Для подтверждения того, что соответствующие локальные службы доступны для удаленных клиентов, необходимо выполнить сканирование на наличие открытых сокетов и портов.

Ключевые понятия

- *TCP/IP* — протокол управления передачей и межсетевой протокол, которые определяют администрирование поведения сети.
- *Общедоступные IP-адреса* должны быть глобально уникальными, тогда как *NAT-адреса* должны быть уникальными только в пределах их локальной сети. *Протокол динамической конфигурации хоста (DHCP)* обычно используется для управления динамическим (непостоянным) назначением адресов.
- *Сетевой маршрут* — это адрес шлюзового маршрутизатора, через который компьютер получает доступ к сети.
- *Система доменных имен (DNS)* обеспечивает трансляцию между числовыми IP-адресами и понятными для человека URL-адресами, что обеспечивает удобную навигацию по интернет-ресурсам.
- *Сетевой сокет* — это представление IP-адреса и порта, через которые можно активизировать сетевое соединение.

Рекомендации по безопасности

Полезно периодически использовать такой инструмент, как `nmap` для аудита вашей системы на предмет непонятно открытых портов.

Обзор команд

- `ip addr` показывает активные интерфейсы в системе Linux. Вы можете сократить команду до `ip a` или удлинить до `ip address`. Как хотите.
- `lspci` перечисляет устройства PCI, которые в данный момент подключены к вашему компьютеру.
- `dmesg | grep -A 2 Ethernet` ищет в журналах `dmesg` совпадения со строкой *Ethernet* и отображает найденное вместе с последующими двумя строками вывода.
- `ip route add default via 192.168.1.1 dev eth0` вручную устанавливает новый сетевой маршрут для компьютера.
- `dhclient enp0s3` запрашивает динамический (DHCP) IP-адрес для интерфейса enp0s3.
- `ip addr add 192.168.1.10/24 dev eth0` назначает статический IP-адрес интерфейсу eth0, который не сохранится после перезапуска системы.
- `ip link set dev enp0s3 up` запускает интерфейс enp0s3 (полезно после редактирования конфигурации).
- `netstat -l | grep http` сканирует локальный компьютер на наличие веб-службы, прослушивающей порт 80.
- `nc -z -v bootstrap-it.com 443 80` сканирует удаленный сайт на наличие служб, прослушивающих порты 443 или 80.

Самотестирование

1. Что из перечисленного является действительным IP-адресом NAT:
   а) 11.0.0.23;
   б) 72.10.4.9;
   в) 192.168.240.98;
   г) 198.162.240.98?
2. Как бы вы описали подсети IPv4, используя два октета для сетевых адресов, как с CIDR, так и в нотации сетевых масок:
   а) x.x.x.x/16 или 255.255.0.0;
   б) x.x.x.x/24 или 255.255.255.0;
   в) x.x.x.x/16 или 255.0.0.0;
   г) x.x.x.x/16 или 255.255.240.0?
3. Какая из следующих команд поможет вам определить обозначение, данное Linux сетевому интерфейсу:
   а) `dmesg`;
   б) `lspci`;
   в) `lshw -class network`;
   г) `dhclient`?
4. Вы устанавливаете компьютер в своем офисе и хотите, чтобы он имел надежное подключение к сети. Какой из следующих профилей будет работать лучше:
   а) динамический IP-адрес, подключенный напрямую к Интернету;
   б) статический IP-адрес, который является частью сети NAT;
   в) статический IP-адрес, подключенный напрямую к Интернету;
   г) динамический IP-адрес, который является частью сети NAT?
5. Какая из следующих команд используется для запроса динамического IP-адреса:
   а) `ip route`;
   б) `dhclient enp0s3`;
   в) `ip client enp0s3`;
   г) `ip client localhost`?
6. Какой файл вы бы отредактировали для настройки сетевого интерфейса enp0s3 на компьютере с CentOS:
   а) `/etc/sysconfig/networking/ipcfg-enp0s3`;
   б) `/etc/sysconfig/network-scripts/ipcfg-enp0s3`;
   в) `/etc/sysconfig/network-scripts/enp0s3`;
   г) `/etc/sysconfig/network-scripts/ifcfg-enp0s3`?

7. Какую строку вы бы добавили в раздел конфигурации сетевого интерфейса файла `/etc/network interfaces` на компьютере с Ubuntu, чтобы заставить интерфейс использовать сервер DNS Google:
   а) `DNS1=8.8.8.8`;
   б) `dns-nameserver 8.8.8.8`;
   в) `nameserver 8.8.8.8`;
   г) `dns-nameserver1 8.8.8.8`?
8. Что из нижеперечисленного будет сканировать известные TCP-порты на удаленном сервере на наличие доступных слушающих служб:
   а) `nmap -s -p1-1023 bootstrap-it.com`;
   б) `nmap -sU -p80 bootstrap-it.com`;
   в) `nmap -sT -p1-1023 bootstrap-it.com`;
   г) `nc -z -v bootstrap-it.com`?

Ответы

1 — в; 2 — а; 3 — а; 4 — г; 5 — б; 6 — г; 7 — г; 8 — в.

15 Устранение неполадок с периферийными устройствами

В этой главе

- Анализ системных профилей оборудования.
- Управление модулям и ядра для администрирования аппаратных устройств.
- Управление настройкам и ядра для разрешения конфликтов оборудования при загрузке.
- Настройка DNS для трансляции адресов.
- Использование CUPS для управления принтерами и устранения неполадок с ними.

Связь между щелчком кнопкой мыши и тем, что происходит на вашем экране, сложна. Проще говоря, вам нужен какой-то программный процесс, который будет переносить данные между мышью и компьютером, между компьютером и запущенным на нем программным обеспечением, а также между программным обеспечением и экраном.

Это больше чем просто перемещение данных: нужен особый способ передачи данных между мышью, которая знакома только с ковриком, на котором лежит, и программным обеспечением, которое распознает лишь нули и единицы. Умножьте это на тысячи моделей устройств и добавьте множество типов подключений (PCI, SATA, USB, последовательный), и все это только на вашем компьютере.

Учитывая всю сложность, удивительно, что эти устройства так слаженно работают. Из данной главы вы узнаете, как справляться со многими трудностями... например, когда маркетинговая команда ждет, что вы настроите веб-камеру, чтобы начать их виртуальную встречу. Или когда их Wi-Fi не позволяет им присоединиться к беседе. Чтобы сделать все это, вам нужно понять, как Linux видит ваши периферийные устройства и как вы можете заставить ядро Linux обратить внимание на конкретное устройство.

15.1. Идентификация подключенных устройств

Веб-камера подключена, но не транслирует ваше улыбающееся лицо через Интернет? Принтер не печатает? Wi-Fi-адаптер не адаптируется (или что он там делает)? Прежде чем тратить слишком много времени и энергии на активизацию аппаратных устройств, сначала вы должны принять то, что, как это ни печально, операционная система (ОС) может иногда даже не распознать подключенное оборудование. Если новое подключенное устройство не работает, то в первую очередь вы должны подтвердить, что Linux знает о нем. Мы займемся этим на следующих страницах. Если вы провели диагностику, о которой я собираюсь рассказать, и до сих пор нет признаков жизни, рассмотрите возможность того, что:

- устройство несовместимо с вашим оборудованием или с Linux;
- устройство повреждено или неисправно;
- аппаратный интерфейс или кабель поврежден или неисправен;
- система нуждается в перезагрузке;
- у вас плохой день.

Как только устройство и система будут общаться друг с другом, я покажу вам, как использовать модули ядра, чтобы Linux и устройство могли объединиться и выполнить для вас некоторую работу. Начнем с рассмотрения вашего оборудования глазами Linux. На самом деле весь этот процесс «выяснить, распознает ли Linux устройство, которое вы только что подключили», не совсем новый. Вспомните, как вы использовали `lsblk` в главе 6 для обнаружения подключенных блочных устройств. У `lsblk` есть несколько родственников: `lsusb` перечисляет все USB-устройства, о которых знает Linux, и, как вы видели в предыдущей главе, `lspci` делает то же самое для PCI-устройств. Вот пример:

```
$ lsusb
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 008: ID 04f9:0249
    Brother Industries, Ltd      <-- Лазерный принтер Brother
Bus 001 Device 007: ID 413c:2005 Dell Computer Corp. RT7D50 Keyboard
Bus 001 Device 006: ID 046d:081a Logitech, Inc.              <-- Веб-камера
Bus 001 Device 005: ID b58e:9e84 Blue Microphones Yeti Stereo Microphone
Bus 001 Device 004: ID 1a40:0101
    Terminus Technology Inc. Hub  <-- Многопортовый USB-концентратор
Bus 001 Device 002: ID 148f:5372
    Ralink Technology, Corp. RT5372 Wireless Adapter  <-- USB Wi-Fi-адаптер
Bus 003 Device 002: ID 093a:2510 Pixart Imaging, Inc. Optical Mouse
```

В главе 14 вы также увидели великого патриарха семьи `ls` — `lshw`. При запуске от имени **root** `lshw` выводит полный профиль оборудования вашей системы. Эта команда может очень многое рассказать о каждом элементе вашего оборудования. Удобнее всего будет преобразовать вывод в простой для чтения HTML-файл, который вы

сможете просмотреть в своем браузере. Для этого предназначен аргумент `-html`. Если щелкнуть на имени файла в файловом менеджере с графическим интерфейсом, например Nautilus, он загрузится в браузер по умолчанию. Вот как:

```
# lshw -html > lshw-output.html
```

Помните, как мы использовали `lshw -class network` в прошлой главе, чтобы ограничить вывод только сетевым контентом? Этот прием будет работать и для других подмножеств данных. Например, `lshw -c memory` отображает подробную информацию обо всех типах памяти, используемой вашей системой (включая RAM, прошивку BIOS и кэш); `-c`, как вы уже догадались, работает как более быстрая альтернатива `-class`. В дополнение к этому `lshw -c storage` выводит информацию об интерфейсах SATA и SCSI; `-c multimedia` охватывает аудио- и видеоустройства, а команда `-c cpu` расскажет вам все, что вы когда-либо хотели знать о процессоре, подключенном к материнской плате. Вот как удобно *получить* всю информацию. Но как это *использовать*?

Опишу обычный сценарий. Предположим, вы рассматриваете возможность добавить дополнительную оперативную память в систему — вероятно, метрики, которые вы собрали (см. главу 13), говорят о том, что вам ее не хватает. Вам нужно знать, сколько у вас памяти на данный момент, какой тип ОЗУ, не говоря уже о том, какую материнскую плату вы используете, чтобы понять, какие слоты доступны для ОЗУ и какова их максимальная емкость.

В данном случае ОЗУ — периферийное устройство, которое мы рассматриваем в примере возможного обнаружения аппаратного обеспечения. А обнаружение оборудования всегда должно быть первым шагом при устранении проблем с оборудованием.

В качестве иллюстрации `lshw` показывает, что на моей материнской плате четыре слота оперативной памяти, два из которых в настоящее время заняты модулями памяти A-Data DDR3 1600 размером 4 Гбайт. Поскольку следует избегать установки неодинаковых модулей памяти в одной системе, я точно знаю, какой тип ОЗУ должен купить, чтобы заполнить эти два пустых слота и удвоить емкость.

Надо отметить, что у меня нет никаких краткосрочных планов по обновлению моего рабочего места. А зачем мне это? Скромное аппаратное обеспечение, которое уже установлено, позволяет мне запускать несколько виртуальных машин при редактировании и/или кодировании небольшого количества видео (с помощью Kdenlive), и при этом хотя бы в одном браузере открыто более десятка вкладок. И компьютер, который я создал с нуля менее чем за 300 долларов, работает значительно лучше, чем те, которые стоят больше 1000 долларов и которыми пользуются многие мои коллеги. Какая разница? Эти бедняги кормят свои ресурсоемкие операционные системы Windows и macOS, в то время как я работаю в быстрой и эффективной Linux. Попробуйте.

Что, если ваше устройство распознается Linux, но все еще неактивно? Возможно, нужно загрузить соответствующий модуль ядра.

15.2. Управление периферийными устройствами с помощью модулей ядра Linux

Linux управляет аппаратной периферией, используя модули ядра. Вот как это происходит.

Работающее ядро Linux — один из тех компонентов, с настройкой которого вы точно не хотите напутать. В конце концов, ядро — это программное обеспечение, которое управляет всем, что делает ваш компьютер. Учитывая, сколько деталей необходимо одновременно обрабатывать в работающей системе, лучше не трогать непосредственно настройки, чтобы выполнять свою работу с как можно меньшим количеством отвлекающих факторов. Но если невозможно внести даже небольшие изменения в вычислительную среду без перезагрузки всей системы, то подключение новой веб-камеры или принтера может привести к серьезному нарушению всего рабочего процесса. Необходимость перезагрузки при каждом добавлении устройства, чтобы система распознала его, едва ли эффективна.

Чтобы достичь баланса между стабильностью и удобством использования, Linux изолирует само ядро, но позволяет добавлять специальные функции через *загружаемые модули ядра* (LKM). Глядя на рис. 15.1, вы можете представить модуль как часть программного обеспечения, которая сообщает ядру, где найти устройство и что с ним делать. В свою очередь, ядро делает устройство доступным для пользователей и процессов и контролирует его работу.

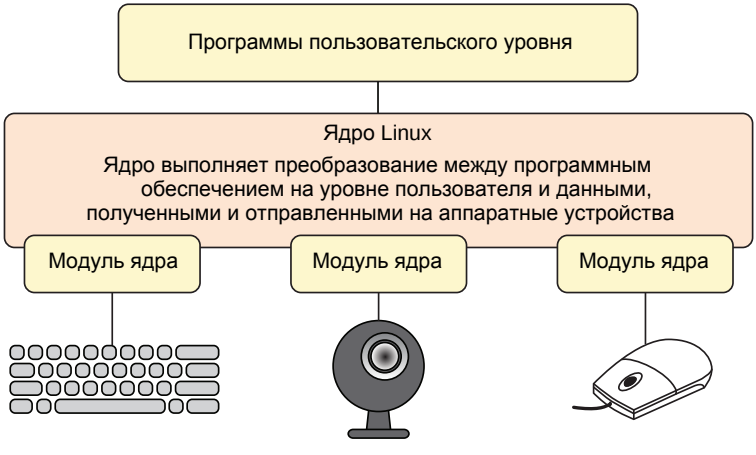


Рис. 15.1. Модули ядра действуют как трансляторы между устройствами и ядром Linux

Ничто не мешает вам написать собственный модуль для поддержки устройства именно так, как вам нравится, но зачем? Библиотека модулей Linux уже настолько надежна, что, как правило, нет необходимости разрабатывать свою собственную. И в большинстве случаев Linux автоматически загружает модуль нового устройства, даже если вы об этом не знаете.

Тем не менее бывает, что по какой-то причине этого не происходит. Чтобы понять, как действовать в такой ситуации, вам следует немного больше узнать о модулях ядра и, в частности, о том, как найти тот самый модуль, который будет работать с вашим периферийным устройством, а затем как вручную активизировать его.

15.2.1. Поиск модулей ядра

По общепринятому соглашению модули — это файлы с расширением `.ko` *(объект ядра)*, которые находятся в каталоге `/lib/modules/`. Однако прежде, чем перейти к ним, вам, вероятно, придется сделать выбор. Поскольку вам предоставляется возможность загружать лишь один релиз из списка, программное обеспечение, необходимое для поддержки вашего выбора (включая модули ядра), должно где-то существовать. Одним из этих мест является `/lib/modules/`. Здесь вы найдете каталоги, заполненные модулями для каждого доступного релиза ядра Linux, например:

```
$ ls /lib/modules
4.4.0-101-generic
4.4.0-103-generic
4.4.0-104-generic
```

В моем случае активное ядро — это версия с наибольшим номером релиза (`4.4.0-104-generic`), но нет никакой гарантии, что она будет такой же для вас (ядра часто обновляются). Если вы собираетесь поработать с модулями, которые хотели бы использовать в реальной системе, то должны быть уверены, что у вас есть правильное дерево каталогов.

Хорошие новости: есть проверенный метод. Вместо того чтобы определять каталог по имени и надеяться, что вы получите правильный, используйте системную переменную, которая всегда указывает на имя активного ядра. Вы можете вызвать ее, введя команду `uname -r` (`-r` велит показать номер выпуска ядра из всей системной информации, которая обычно отображается):

```
$ uname -r
4.4.0-104-generic
```

Получив эту информацию, вы можете включить `uname` в ссылки на файловую систему, используя процесс, известный как *подстановка команд*. Например, чтобы перейти к нужному каталогу, вы должны добавить его в `/lib/modules`. Чтобы сообщить Linux, что само слово `uname` не является местоположением файловой системы, заключите его в обратные кавычки, например:

```
$ ls /lib/modules/`uname -r`
build   modules.alias        modules.dep       modules.softdep
initrd  modules.alias.bin    modules.dep.bin   modules.symbols
kernel  modules.builtin      modules.devname   modules.symbols.bin
misc    modules.builtin.bin  modules.order     vdso
```

Вы увидите, что большинство модулей сами организованы в собственных подкаталогах в каталоге `kernel/`. Потратьте несколько минут, чтобы просмотреть эти

каталоги и понять, как все устроено и что доступно. Имена файлов обычно дают хорошее представление об их назначении:

```
$ ls /lib/modules/`uname -r`/kernel
arch  crypto  drivers  fs  kernel  lib  mm
    net  sound  ubuntu  virt  zfs
```

Самым загруженным из перечисленных здесь подкаталогов является kernel, в котором вы можете найти модули для сотен устройств

Я привел один из способов найти модули ядра, но на самом деле это не самый оптимальный вариант. Если вы хотите получить полный список, можете перечислить все загруженные в настоящее время модули вместе с некоторой базовой информацией, используя команду `lsmod`. Первый столбец — имя модуля, за ним размер и номер файла, а затем имена других модулей, от которых зависит текущий:

```
$ lsmod
[...]
vboxdrv               454656   3 vboxnetadp,vboxnetflt,vboxpci
rt2x00usb              24576   1 rt2800usb
rt2800lib              94208   1 rt2800usb
[...]
```

Эта небольшая выборка результатов показывает модули, связанные с VirtualBox и моим адаптером USB Wi-Fi

Слишком много — это сколько? Хорошо, давайте снова запустим `lsmod`, но на этот раз передадим вывод в `wc -l`, чтобы подсчитать количество строк:

```
$ lsmod | wc -l
113
```

Это загруженные модули. А сколько всего доступно? Запуск `modprobe -c` и подсчет строк даст нам это значение:

```
$ modprobe -c | wc -l
33350
```

Есть 33 350 доступных модулей? Похоже, что кто-то много лет работал над тем, чтобы создать программное обеспечение для работы наших физических устройств.

ПРИМЕЧАНИЕ

В некоторых системах вы можете встретить пользовательские модули, на которые ссылаются либо с их собственными уникальными записями в файле /etc/modules, либо в виде файла конфигурации, сохраненного в /etc/modules-load.d/. Скорее всего, такие модули являются продуктом локальных технических проектов, возможно новаторских. В любом случае хорошо иметь представление о том, на что ты смотришь.

Итак, вы поняли, как находить модули. Ваша следующая задача — выяснить, как вручную загрузить неактивный модуль, если по какой-то причине это не произошло автоматически.

15.2.2. Загрузка модулей ядра вручную

Прежде чем вы сможете загрузить модуль ядра, следует удостовериться в его существовании. А для этого вам нужно знать, как он называется. Получение данной информации иногда может потребовать большого везения, а также помощи от авторов онлайн-документации.

Я проиллюстрирую этот процесс, описав проблему, с которой столкнулся какое-то время назад. В один прекрасный день по причине, которая мне все еще неясна, интерфейс Wi-Fi на ноутбуке перестал работать. Просто так. Возможно, виновато обновление программного обеспечения. Кто знает? Я запустил `lshw -c network` и получил очень странную информацию:

```
network UNCLAIMED
    AR9485 Wireless Network Adapter
```

Linux распознала интерфейс (Atheros AR9485), но показала его как невостребованный. Как говорится: «Когда есть проблемы, надо искать в Интернете». Я запустил поиск `atheros ar9 linux module` и, просеяв множество страниц пяти- и даже десятилетних результатов, где мне советовали написать собственный модуль или просто сдаться, я наконец-то обнаружил, что по крайней мере в Ubuntu 16.04 существует рабочий модуль. Его имя ath9k.

Да! Битва хороша, когда выиграна! Добавить модуль в ядро намного проще, чем кажется. Чтобы дважды проверить, что он доступен, вы можете запустить `find` для дерева каталогов модулей, указать `-type f`, чтобы сообщить Linux, что вы ищете файл, а затем добавить строку `ath9k` и звездочку, чтобы включить все имена файлов, которые начинаются с вашей строки:

```
$ find /lib/modules/$(uname -r) -type f -name ath9k*
/lib/modules/4.4.0-97-generic/kernel/drivers/net/wireless/ath/
➥ ath9k/ath9k_common.ko
/lib/modules/4.4.0-97-generic/kernel/drivers/net/wireless/ath/ath9k/ath9k.ko
/lib/modules/4.4.0-97-generic/kernel/drivers/net/wireless/ath/ath9k/ath9k_htc.ko
/lib/modules/4.4.0-97-generic/kernel/drivers/net/wireless/ath/ath9k/ath9k_hw.ko
```

Еще один шаг, загрузка модуля:

```
# modprobe ath9k
```

Это все. Без перезагрузки. Без суеты.

Приведу еще один пример, чтобы показать вам, как работать с активными модулями, которые были повреждены. Было время, когда моя веб-камера Logitech с определенным программным обеспечением оказывалась недоступной для любых других программ до следующей загрузки системы. Иногда мне приходилось открывать камеру в другом приложении, и у меня не было времени, чтобы выключиться и запуститься снова. (Я запускаю много приложений, и их загрузка после перезагрузки системы занимает некоторое время.)

Поскольку этот модуль предположительно был активен, использование `lsmod` для поиска слова *video* дало мне подсказку о названии соответствующего модуля. Фактически это было лучше, чем подсказка, — единственным модулем, описанным со словом *video*, был uvcvideo, как вы можете видеть из следующего вывода:

```
$ lsmod | grep video
uvcvideo            90112  0
videobuf2_vmalloc   16384  1 uvcvideo
videobuf2_v4l2      28672  1 uvcvideo
videobuf2_core      36864  2 uvcvideo,videobuf2_v4l2
videodev           176128  4 uvcvideo,v4l2_common,videobuf2_core,
➥ videobuf2_v4l2
media               24576  2 uvcvideo,videodev
```

Возможно, я мог что-то сделать с тем, что послужило причиной сбоя, и порыться немного глубже, чтобы увидеть, нельзя ли переопределить настройки. Но вы знаете, как это бывает: иногда просто хочется, чтобы устройство работало. Поэтому я использовал команду `rmmod`, чтобы выгрузить модуль uvcvideo, и `modprobe`, чтобы снова запустить его, красивым и свеженьким:

```
# rmmod uvcvideo
# modprobe uvcvideo
```

Опять же нет перезагрузок. Нет пятен крови.

15.3. Ручное управление параметрами ядра во время загрузки

Поскольку в любом случае мы говорим о ядре, сейчас самое время поговорить о его параметрах. Это тема, которую мы откладывали на потом, потому что «параметры ядра» звучит страшно. Что ж, они пугают: если их неправильно понять, компьютер может по крайней мере временно не загружаться.

Зачем этим вообще заниматься? Затем, что иногда конфигурация загрузки вашего ядра по умолчанию для вашей задачи не работает и единственный способ исправить это — изменить способ загрузки ядра.

Есть два способа передать пользовательские параметры ядру во время загрузки. Один включает редактирование пункта меню GRUB во время процесса загрузки, а другой — изменение файла конфигурации `/etc/default/grub` в работающей системе, чтобы правки вступили в силу при следующем запуске. Чтобы проиллюстрировать каждый из этих подходов, я опишу два практических варианта использования.

15.3.1. Передача параметров во время загрузки

Я не уверен, что это распространенная проблема, но ее обсуждение послужит хорошим примером для обучения. У некоторых неудачников не получается правильно выключить или перезагрузить Linux — каждый раз система зависает. Иногда проблему можно решить добавлением простого параметра ядра. Вот как это делается.

Выберите в меню GRUB релиз Linux, который хотите загрузить (рис. 15.2), нажмите клавишу E, и откроется экран редактирования. Здесь вы сможете редактировать содержимое с клавиатуры.

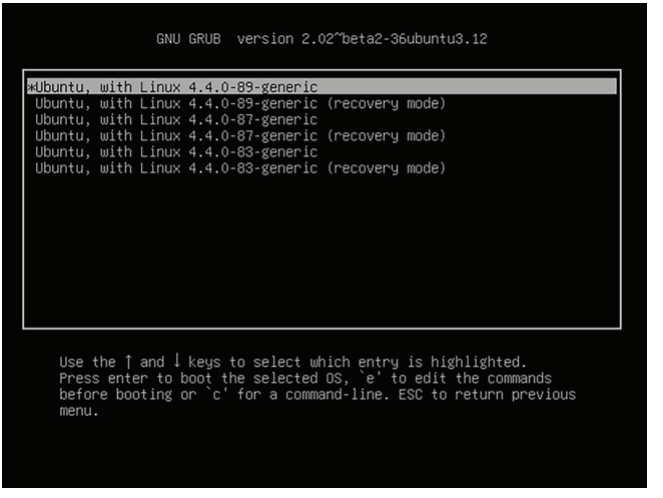


Рис. 15.2. Главное меню GRUB, отображающее несколько ядер Linux, доступных для загрузки

Прокрутите вниз, пока не дойдете до записи Linux, выделенной на рис. 15.3. В этом примере после переноса на следующую строку запись заканчивается на *ro*. (Не беспокойтесь, если у вас все по-другому.) Добавьте `reboot=bios` в конец строки и нажмите Ctrl+X, чтобы принять изменения и загрузиться. Если это не решает проблему с отключением, можете попробовать снова, используя `reboot=pci` вместо `reboot=bios`.

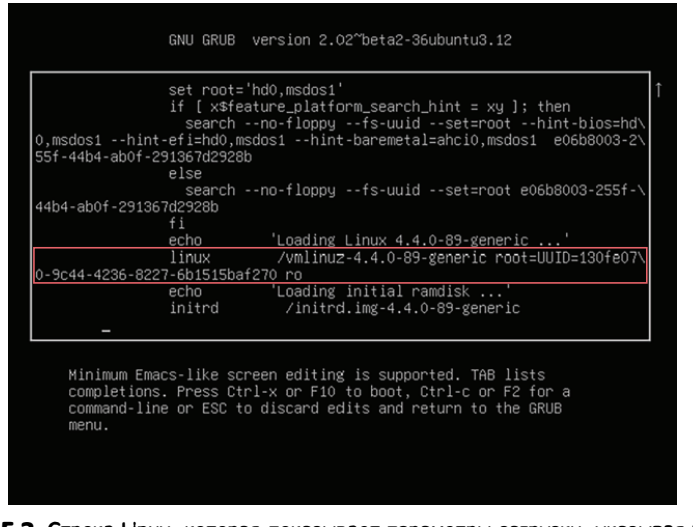


Рис. 15.3. Строка Linux, которая показывает параметры загрузки, указывая GRUB на местоположение образа Linux

Имейте в виду, что эти настройки не будут храниться постоянно. После следующей загрузки настройками GRUB снова будут управлять файлы конфигурации в файловой системе. Как вносить изменения, которые будут сохраняться при загрузке, вы узнаете чуть позже.

15.3.2. Передача параметров через файловую систему

Возможны ситуации, когда вы захотите загрузить настольный компьютер без графического интерфейса. Вероятно, некоторые элементы самого графического интерфейса не загружаются должным образом и вам требуется надежный сеанс оболочки для устранения неполадок. Вы знаете, что можете установить уровень запуска по умолчанию на 3 (многопользовательский неграфический режим) через GRUB.

ПРИМЕЧАНИЕ

> Уровень выполнения — это параметр, который определяет состояние системы Linux для конкретного сеанса. Выбор между уровнями выполнения 0–6 определяет, какие службы должны быть доступны, начиная от полной графической, многопользовательской системы и заканчивая отсутствием служб (то есть отключение).

Откройте файл `/etc/default/grub` и найдите строку `GRUB_CMDLINE_LINUX_DEFAULT`. Обычно она содержит пару параметров и выглядит примерно так:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
```

Добавьте код `systemd.unit=runlevel3.target` в конец строки, чтобы она выглядела следующим образом. (Строка `"quiet splash"` нас не интересует; это то, что вы видите на экране во время загрузки.)

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash systemd.unit=runlevel3.target"
```

Запустите `update-grub` в Ubuntu или `grub2-mkconfig` в CentOS, чтобы изменения вступили в силу. В следующий раз, когда вы включите свой компьютер, вы попадете в оболочку командной строки. Закончив с устранением неполадок, можете удалить `systemd.unit=runlevel3.target` из `/etc/default/grub`, снова обновить GRUB и перезагрузиться.

15.4. Управление принтерами

Хотите удивиться? Присядьте. Я собираюсь посоветовать вам отказаться от совершенного консольного инструмента в пользу его графического эквивалента. Инструмент командной строки? `lp`. В комплекте с обновленным движком `lp`, безусловно, может делать очень интересные вещи. Но, поверьте мне, если из сети исчезнет один

из поддерживаемых вами офисных принтеров, вы не станете открывать командную строку для устранения неполадок. В наши дни за это отвечает единая система печати UNIX (CUPS). Но прежде, чем мы до нее доберемся, я приведу пару команд `lp`, которые могут пригодиться.

15.4.1. Основы lp

Предположим, на удаленном компьютере есть файл, который нужно распечатать. Вы знаете, что запускать LibreOffice через удаленный сеанс X не так уж и весело, верно? Разве было бы плохо сделать это с помощью простой, быстрой и надежной оболочки SSH? Больше ни слова. Используйте `lpq` для просмотра списка доступных принтеров (вместе с текущей очередью заданий):

```
$ lpq
Brother-DCP-7060D is ready
no entries
```

Затем введите `lp`, чтобы распечатать файл. Если в системе несколько принтеров, вам также необходимо указать принтер, который вы хотите использовать. Вот пример:

```
$ lp -d Brother-DCP-7060D /home/user/myfile.pdf
```

Не хотите печатать прямо сейчас? Запланируйте это на потом. Параметр расписания `-H` всегда опирается на время UTC, а не на местное:

```
$ lp -H 11:30 -d Brother-DCP-7060D /home/user/myfile.pdf
```

15.4.2. Управление принтерами с помощью CUPS

Давным-давно, прежде чем покупать принтер для использования с системой Linux, нужно было провести тщательное и трудоемкое исследование, чтобы убедиться в их совместимости. Приходилось загружать и устанавливать соответствующий драйвер, а затем вручную устанавливать принтер через ОС. Если все работало, это было настоящим праздником. За последние годы произошло три события, которые положительно повлияли на печать в Linux.

- Модульная система печати CUPS была принята многими, если не всеми, дистрибутивами Linux в целях управления принтерами и печатью. Верьте или нет, CUPS управляется от имени сообщества Apple. Как вы вскоре увидите, интерфейс CUPS значительно упрощает администрирование и устранение неполадок и является достаточно надежным.
- Крупные производители принтеров теперь обычно предоставляют драйверы для Linux. Они не всегда идеальны, но работоспособны. Это означает, что в наши дни практически любой современный принтер может работать на компьютерах

с Linux. Тем не менее вам стоит быстро взглянуть на онлайн-ресурс help.ubuntu.com/community/Printers.

- Начиная с выпуска 17.04, Ubuntu предлагает печать без драйверов. Это значит, что любые доступные локальные или сетевые принтеры будут автоматически добавлены в CUPS без необходимости какой-либо настройки.

В любом случае вы получаете доступ к интерфейсу CUPS через браузер, указывая его порт — 631 — на своем собственном компьютере (localhost:631). Вкладка Administration (Администрирование) (рис. 15.4) содержит прямые ссылки для управления поиском, защитой, планированием и отслеживанием всех доступных принтеров.

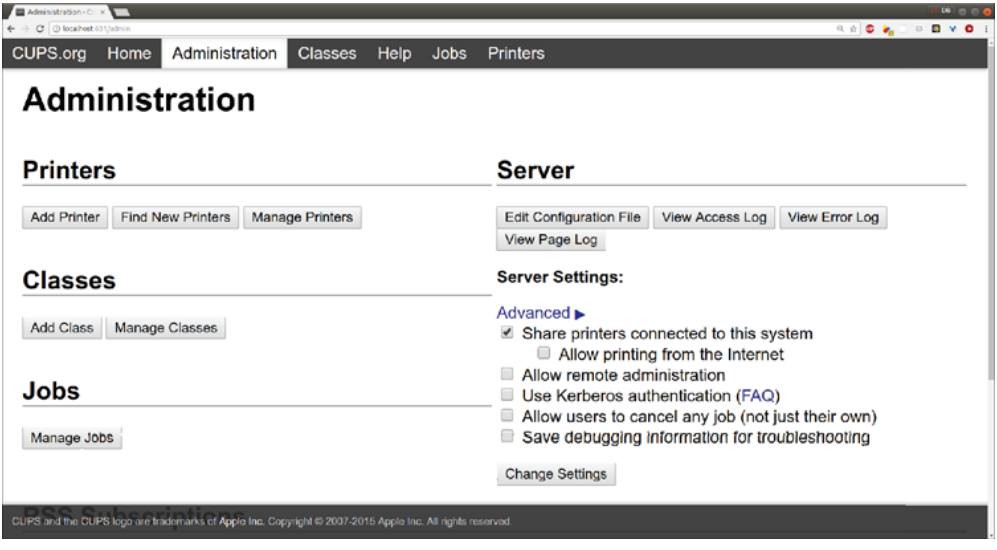


Рис. 15.4. Интерфейс браузера CUPS, который по умолчанию доступен через порт 631 на любом компьютере с Linux, где работает CUPS

Вы даже можете управлять группами принтеров, чтобы обеспечить эффективное использование нескольких устройств. Это отличный способ организации ресурсов, позволяющий, например, печатать в цвете только при высокоприоритетных заданиях, а черновики документов печатать на более дешевых и менее качественных принтерах.

CUPS неявно обрабатывает все настройки администрирования. Если вам нужно отключить определенный принтер от одного компьютера (или сетевой подсети, если это сетевой принтер) и подключить к другому, понадобится только внести соответствующие изменения в интерфейс CUPS на хосте. Информация о маршруте будет автоматически обновлена по сети в течение минуты или двух.

CUPS знает, какие принтеры доступны, потому что по умолчанию принтеры, подключенные к любому компьютеру, на котором работает CUPS, будут передавать данные о своем существовании всем остальным в сети. Этим, наряду со многими

другими параметрами конфигурации, можно управлять из файла `/etc/cups/cupsd.conf` (листинг 15.1).

Листинг 15.1. Раздел файла конфигурации /etc/cups/cupsd.conf

```
LogLevel warn
PageLogFormat
MaxLogSize 0
# Allow remote access
Port 631
Listen /var/run/cups/cups.sock
# Share local printers on the local network.
Browsing On
BrowseLocalProtocols dnssd
DefaultAuthType Basic
WebInterface Yes
[...]
```

Вы можете изменить сетевой порт, используемый CUPS

Вы можете полностью заблокировать доступ к веб-интерфейсу, если этого требуют соображения безопасности

Если CUPS не распознает один из ваших принтеров, можно попробовать несколько операций еще до поиска в Интернете (используя имя вашего принтера и слово *linux*).

- Проверьте сообщения об ошибках. Запуск `systemctl status cups` позволит просмотреть самые последние предупреждения.
- Запустите `lsusb` (при условии, что это USB-принтер) и/или `lpinfo -v` из командной строки, чтобы убедиться, что система видит ваш принтер.
- Убедитесь, что в файле `/etc/cups/printers.conf` нет повторяющихся или устаревших записей. Если есть, остановите CUPS (`systemctl stop cups`), сохраните копию исходного файла, а затем удалите все старые записи. Снова запустите CUPS и попробуйте добавить принтер из интерфейса браузера.
- Убедитесь, что в разделе `<Policy default>` файла `/etc/cups/cupsd.conf` отсутствуют чрезмерно ограничительные параметры, которые могут блокировать допустимые запросы.

Резюме

- При любых решениях по обновлению и ремонту оборудования используйте такие инструменты Linux, как `lshw`, который предоставляет информацию о профиле вашего оборудования.
- Ядро Linux изолировано от системной активности, чтобы защитить его от дестабилизирующих изменений, но модули ядра могут безопасно предоставлять аппаратным устройствам динамический доступ к ресурсам ядра.
- Ядро Linux можно изменить, добавив во время загрузки параметры, отредактировав файлы конфигурации либо воспользовавшись меню GRUB.
- CUPS предоставляет интерфейс для администрирования принтеров в сети.

Ключевые понятия

- *Модуль ядра* — это программное обеспечение, которое определяет атрибуты аппаратного устройства для ядра Linux.
- *Параметр ядра* — аргумент, который добавляется в ядро во время выполнения для управления поведением системы.

Рекомендации по безопасности

Используйте файл `/etc/cups/cupsd.conf` для управления сетевым доступом к вашим принтерам.

Обзор команд

- `lshw -c memory` (или `lshw -class memory`) отображает раздел памяти аппаратного профиля системы.
- `ls /lib/modules/`uname -r`` показывает модули для вашего текущего активного ядра в каталоге `/lib/modules/`.
- `lsmod` перечисляет все активные модули.
- `modprobe -c` перечисляет все доступные модули.
- `find /lib/modules/$(uname -r) -type f -name ath9k*` ищет файл среди доступных модулей ядра с именем, начинающимся с ath9k.
- `modprobe ath9k` загружает указанный модуль в ядро.
- `GRUB_CMDLINE_LINUX_DEFAULT="systemd.unit=runlevel3.target"` (в файле `/etc/default/grub`) загружает Linux в многопользовательском неграфическом режиме.
- `lp -H 11:30 -d Brother-DCP-7060D /home/user/myfile.pdf` ставит задание на печать в очередь на принтер Brother в 11:30 UTC.

Самотестирование

1. Как лучше всего легко визуализировать полный профиль оборудования вашего компьютера:
   а) `lsmod`;
   б) `lshw -class memory`;
   в) `lshw -html > lshw-output.html`;
   г) `modprobe -C`?
2. Как лучше всего ссылаться на местоположение в файловой системе, где содержатся активные модули ядра:
   а) `/lib/kernel/uname -a`;
   б) `/lib/modules/name -r`;

в) `/usr/modules/uname -r`;

г) `/lib/modules/uname -r`?

3. Какая из следующих команд деактивирует модуль ядра:

   а) `delmod uvcvideo`;

   б) `rmmod uvcvideo`;

   в) `modprobe -d uvcvideo`;

   г) `rmmod -r uvcvideo`?

4. Вам нужно передать параметр ядру Linux, который будет задействован немедленно и навсегда. Какие шаги следует предпринять:

   а) отредактировать строку `linux` в разделе **Edit** меню GRUB во время загрузки, сохранить файл нажатием **Ctrl+X** и загрузиться;

   б) добавить параметр в файл `/etc/default/grub`, обновить GRUB и перезагрузить компьютер;

   в) обновить GRUB из командной строки, перезагрузить компьютер, добавить параметр в строку `linux` в разделе **Edit** меню GRUB во время загрузки, сохранить файл нажатием **Ctrl+X** и загрузиться;

   г) перезагрузиться, обновить GRUB из командной строки и добавить параметр в файл `/etc/default/grub`?

5. Какая из следующих команд запланирует задание на печать в 10:30:

   а) `lpd 10:30 -d Brother-DCP-7060D /home/user/myfile.pdf`;

   б) `lpq -h 10:30 -d Brother-DCP-7060D /home/user/myfile.pdf`;

   в) `lp -T 10:30 -d Brother-DCP-7060D /home/user/myfile.pdf`;

   г) `lp -H 10:30 -d Brother-DCP-7060D /home/user/myfile.pdf`?

Ответы

1 — в; 2 — г; 3 — б; 4 — б; 5 — г.

16 Инструменты DevOps: развертывание серверной среды с использованием Ansible

В этой главе

- Использование инструментов оркестровки для автоматизации многоуровневых развертываний Linux.
- Управление серверами Linux с помощью сценариев Ansible playbook.
- Организация данных, связанных с развертыванием, в модульной архитектуре.

Вы видели, как сценарии могут автоматизировать сложные и скучные процессы и гарантировать их регулярное и правильное выполнение. Вы писали сценарии, определяя задания резервного копирования, в главе 5. В главе 2 вы видели, как за считаные секунды можно подготовить и запустить виртуальные серверы Linux. Есть ли какая-то причина, по которой вы не сможете объединить эти инструменты и автоматизировать создание инфраструктуры виртуальной среды? Нет. Абсолютно.

А вы сами этого хотите? Если вы и ваша команда вовлечены в IT-проект с участием нескольких разработчиков, регулярно размещающих версии программного обеспечения на нескольких серверах, то вам, вероятно, следует серьезно подумать об этом, особенно если у вас есть планы принять какой-то вариант DevOps-подхода для управления проектом (рис. 16.1).

Что такое DevOps? Это способ организации рабочего процесса, используемый технологическими компаниями и отличающийся тесным сотрудничеством между командами разработки, обеспечения качества (QA) и системного администрирования проекта. Цель состоит в том, чтобы использовать шаблоны *(инфраструктура*

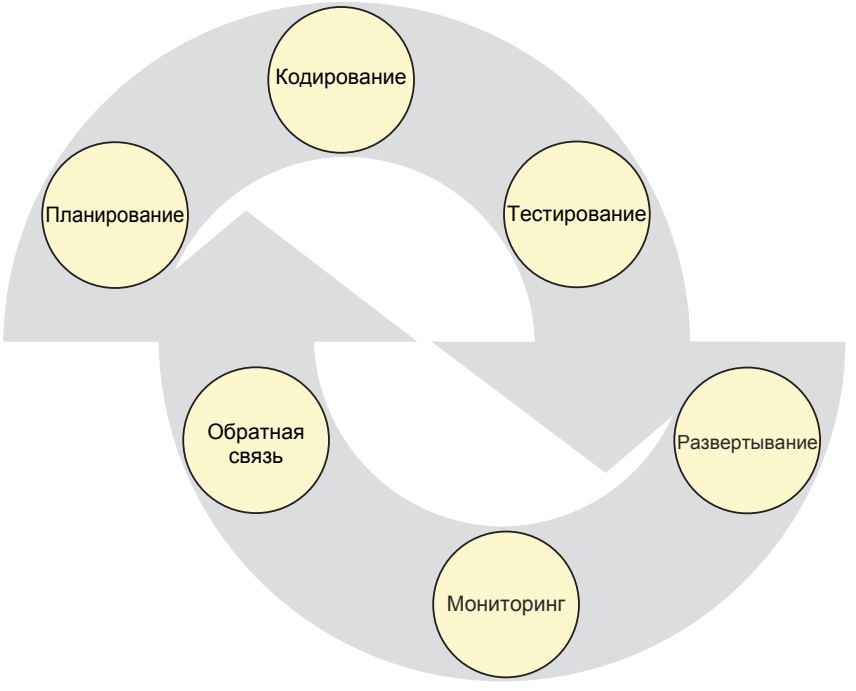


Рис. 16.1. Классический цикл DevOps включает в себя обратную связь, тестирование и мониторинг в процессе разработки

как код) для ускорения циклов развертывания и обновления программного обеспечения, а также для повышения уровня процессов автоматизации и мониторинга.

Внедрение таких управляющих инструментов, как Ansible, принесет множество дивидентов. Безусловно, ускорит рабочий процесс и даст возможность подключить новый или обновленный код к своего рода виртуальной сборочной линии со всей базовой инфраструктурой и решением проблем совместимости, о которых позаботились заранее. Кроме того, это позволяет значительно улучшить качество и уменьшить количество ошибок.

Поскольку бóльшая часть действий DevOps связана с инфраструктурой Linux, а системные администраторы так же важны для процесса, как и разработчики, есть большая вероятность, что рано или поздно ваша карьера в Linux перерастет DevOps. Прежде чем закончить эту книгу, было бы неплохо немного поговорить о мире DevOps и управляющих инструментах.

Представьте, что вы несете ответственность за сложную платформу, подобную той, что показана на рис. 16.2. Она включает в себя отдельные серверы приложений, баз данных и аутентификации, все дублируется в среде разработки, эксплуатации и резервного копирования. Серверы разработки предоставляют вам безопасное место для тестирования кода перед его отправкой в эксплуатацию, а резервные серверы могут быть задействованы в случае сбоя эксплуатируемых серверов. Ваши разработчики постоянно трудятся над добавлением функций и исправлением ошибок.

И они регулярно добавляют свой новый код в производственный цикл. Кроме того, количество серверов, которые вы используете, постоянно меняется, чтобы удовлетворить растущий или снижающийся пользовательский спрос.

С таким большим количеством кода вам понадобится некоторая помощь, чтобы все слаженно работало.

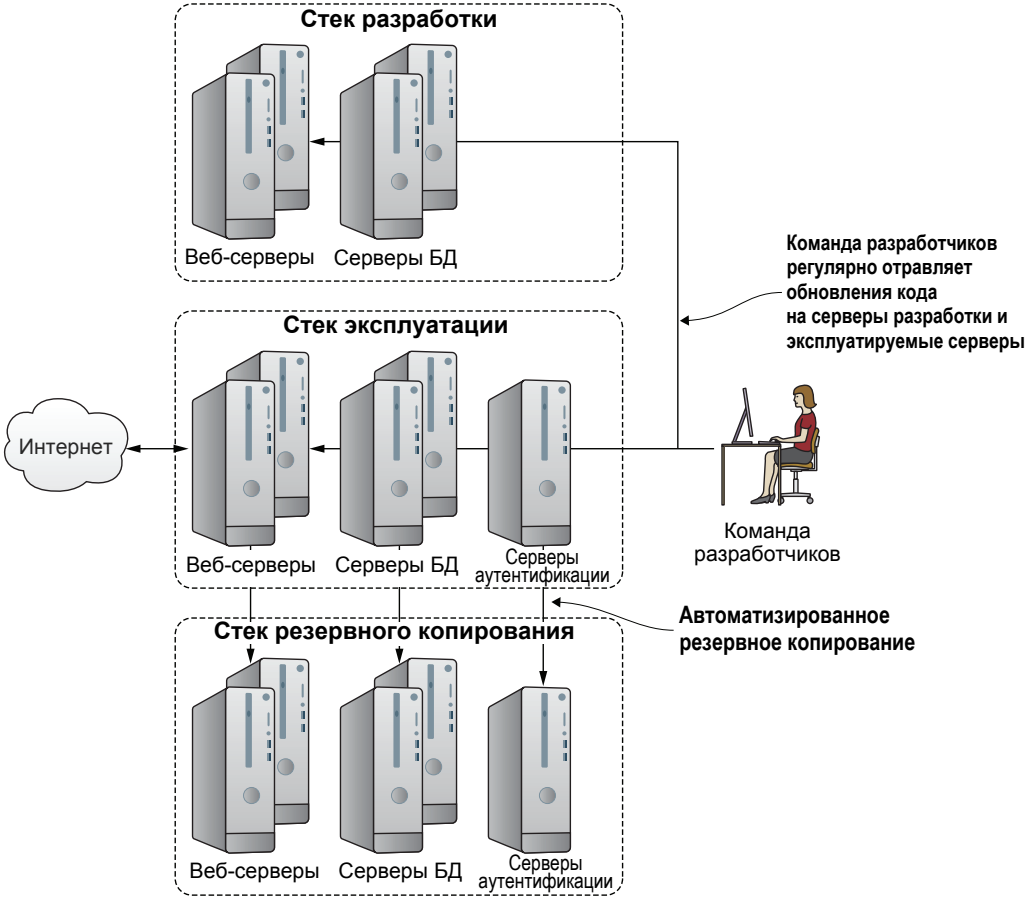


Рис. 16.2. Стандартная среда разработки приложений, с постоянными обновлениями программного обеспечения от команды разработчиков

16.1. Чем полезна оркестровка развертывания

Управляющие инструменты развертывания будут прекрасно работать на устаревших серверах с открытым исходным кодом, но, только подключив окрестратор к виртуализированному развертыванию, вы сможете по полной насладиться их мощностью. Учитывая, как легко писать сценарии для создания виртуальных серверов хоть на вашем собственном оборудовании, хоть с использованием ресурсов облачного

провайдера наподобие AWS, возможность автоматизировать создание стеков программного обеспечения для ваших виртуальных машин только увеличит скорость и эффективность.

Идея в том, что вы создаете один или несколько текстовых файлов. Их содержимое описывает точное состояние, которое вы хотите создать для всей системы и прикладного программного обеспечения на указанном компьютере (обычно его называют *«хост»*). При запуске оркестратор прочитает эти файлы, войдет в систему на соответствующем хосте или хостах и выполнит все команды, необходимые для достижения желаемого состояния. Вместо того чтобы вручную выполнять утомительный и чреватый ошибками процесс на каждом из запускаемых хостов, вы даете команду оркестратору сделать все за вас. Когда ваша инфраструктура вырастет до десятков или даже тысяч хостов, подобный способ автоматизации будет не просто удобен, а необходим.

Но если дело только в автоматизации действий файловой системы, почему бы не использовать уже имеющиеся у вас навыки написания сценариев Bash? Вы можете попробовать, но как только дело дойдет до подключения в эти сценарии таких компонентов, как удаленная аутентификация и конфликтующие программные стеки, ваш оптимизм быстро угаснет.

Оркестраторы будут безопасно и надежно управлять переменными и паролями за вас и применять их по мере необходимости. Вам не нужно самостоятельно отслеживать все детали. Поскольку существует множество видов инструментов оркестровки, то ваш выбор будет зависеть от специфики вашего проекта, организации и опыта. Вам придется задать себе несколько основных вопросов, например: «Большинство вовлеченных людей — разработчики или системные администраторы?» или «Будете ли вы использовать методологию непрерывной интеграции?». В табл. 16.1 приведены профили четырех основных вариантов.

Таблица 16.1. Популярные оркестраторы развертывания

Инструмент	Особенности
Puppet	Широкая поддержка. Рекомендуются навыки написания кода. Расширяется с помощью Ruby. Требует установки агентов на всех клиентах
Chef	Интегрирован с Git. Рекомендуются навыки написания кода. Расширяется с помощью Ruby. Сложность в изучении. Широкая поддержка. Требует установки chef-агентов на всех клиентах
Ansible	Удобен для системных администраторов. На основе Python. Не требуется код, нет агентов на хосте. Простые, быстрые соединения работают через SSH. Запуск через текстовые файлы (под названием playbook). Прост в изучении

Продолжение ⇙

Таблица 16.1 *(продолжение)*

Инструмент	Особенности
Salt	Работает через агенты (называемые миньонами). Высокомасштабируемый. Удобен для системных администраторов

Мне, как системному администратору, Ansible подходит лучше всего, поэтому на нем мы и сосредоточимся далее. Но цели и запросы вашего конкретного проекта могут различаться.

16.2. Ansible: установка и настройка

Перед запуском вам понадобится обновить Python до последней версии на сервере Ansible и на всех машинах, которые вы планируете использовать в качестве хостов. Для этого подойдут команды `apt install python` либо `yum install python`. Какую бы версию Python вы ни использовали (имеется в виду Python 2 или 3), убедитесь, что `python --version` запускается из командной строки.

ПРИМЕЧАНИЕ

На момент написания этой статьи Ansible лучше работает с более старой версией Python — 2.7. Очевидно, что это не будет длиться вечно.

Чтобы установить Ansible на сервер (или управляющий компьютер), вам нужно включить репозиторий EPEL (для CentOS 6), репозиторий Extras (для CentOS 7) или репозиторий ppa:ansible (для Ubuntu). Однако, прежде чем вы сможете включить этот репозиторий на Ubuntu с помощью команды `add-apt-repository`, вам может потребоваться установить пакет software-properties-common. Это будет выглядеть так:

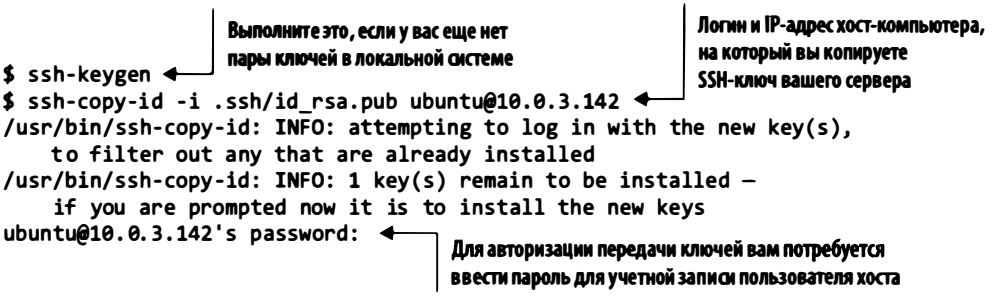
```
# apt install software-properties-common
# add-apt-repository ppa:ansible/ansible
# apt update
# apt install ansible
```

В конце останется лишь запустить два или три контейнера LXC, готовых к Python. Они будут служить хостами (или узлами), которые выполняют всю работу. Нет необходимости устанавливать Ansible ни на одном из хостов, достаточно только на управляющем компьютере.

16.2.1. Настройка беспарольного доступа к хостам

Рассмотрим, как настроить беспарольный доступ к хостам. Ansible предпочитает выполнять свою работу через соединения SSH. Аутентификацию можно обрабатывать из командной строки, но гораздо лучше отправлять SSH-ключи, чтобы обеспечить

беспарольный доступ с хостами. Как это работает, вы должны помнить из главы 3, но на всякий случай повторю:



Теперь, когда Ansible установлен и правильно подключен к вашим хостам, пришло время настройки среды.

16.2.2. Организация Ansible-хостов

Из файла инвентаризации под названием `hosts`, который находится в каталоге `/etc/ansible/`, Ansible получает информацию о том, какими хостами управлять. Файл может состоять из списка IP-адресов или доменных имен и их комбинаций (листинг 16.1).

Листинг 16.1. Простой пример файла /etc/ansible/hosts

```
10.0.3.45
192.168.2.78
database.mydomain.com
```

Но по мере роста количества хостов, которые будет администрировать Ansible, а также сложности общей среды вы захотите организовать все немного лучше. Один из способов сделать это — разделить ваши хосты на группы, которые затем можно использовать для точных действий Ansible (листинг 16.2).

Листинг 16.2. Пример файла /etc/ansible/hosts с организацией в группы хостов

```
[webservers]
10.0.3.45
192.168.2.78

[databases]
database1.mydomain.com
```

Используя подобные группы, задачи Ansible можно настраивать для запуска только с четко определенным подмножеством хостов. Например, отправлять обновленные общедоступные веб-страницы только на веб-серверы, а новые файлы конфигурации — в базы данных (как показано на рис. 16.3).

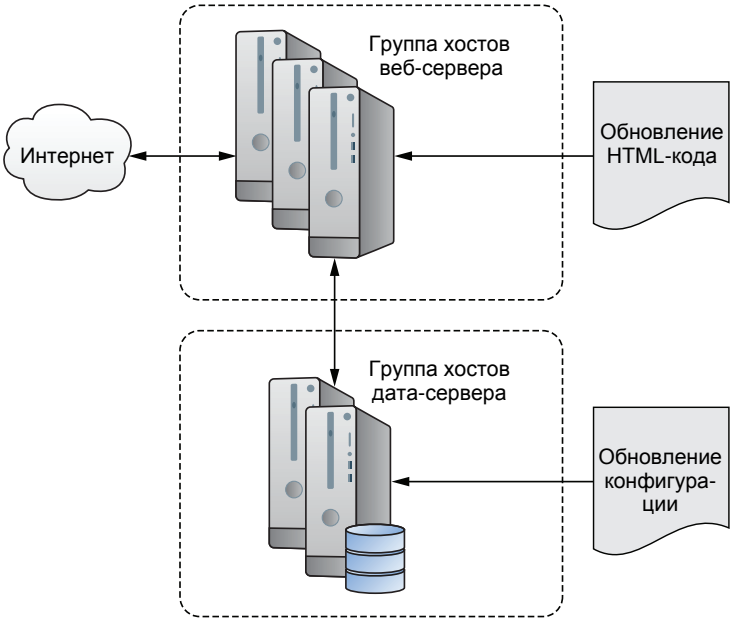


Рис. 16.3. Обновления для конкретных задач отправляются на серверы, организованные в группы хостов

Существует гораздо больше вариантов, которые можно применить к файлу `hosts`. Вы обнаружите, что файл хостов по умолчанию, созданный в `/etc/ansible/` во время установки, уже будет иметь большой выбор синтаксических предложений, например, вы сможете ссылаться на несколько имен хостов в одной строке: `www[001:006].example.com`.

ПРИМЕЧАНИЕ

Таким образом, вы сможете следовать предложениям из этой главы и добавлять IP-адреса LXC хостов (или других), совместимых с Python, в файл hosts.

16.2.3. Тестирование подключения

Чтобы проверить, что все настроено правильно, Ansible может попытаться связаться с хостами, указанными в файле `hosts`. Следующая команда запускает Ansible из командной строки в так называемом режиме *ad hoc*. Параметр `-m` позволяет Ansible загрузить и запустить модуль `ping` для отправки простого запроса *Are You There?* всем хостам, перечисленным в файле `hosts`:

```
$ ansible all -m ping
10.0.3.103 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Условие `all` в этой команде означает, что вы хотите, чтобы это действие выполнялось на всех хостах, перечисленных в файле `hosts`. Если вы хотите запустить команду только для определенной группы хостов, то должны указать имя группы вместо `all`:

```
$ ansible webservers -m ping
```

Теперь, когда подключение прошло успешно, вы можете запускать простые команды удаленно. В этом примере файл `/etc/group` копируется в домашний каталог каждого хоста. Помните, что вы можете сделать это без прохождения аутентификации только потому, что ранее использовали команду `ssh-keygen` для сохранения вашего SSH-ключа на удаленном хосте:

```
$ ansible all -a "cp /etc/group /home/ubuntu"
```

Вы можете подтвердить работу этой операции, запустив команду `ls` через SSH:

```
$ ssh ubuntu@10.0.3.103 "ls /home/ubuntu"
```

Если имя пользователя учетной записи, в которую вы вошли на своем Ansible-сервере, не совпадает с именами пользователей на ваших хостах, вам нужно сообщить об этом Ansible. Вы можете сделать это из командной строки, используя аргумент `--user`, который знает, что имена пользователей хоста относятся к Ubuntu и они будут выглядеть так: `ansible —-userubuntuall -mping`.

16.3. Аутентификация

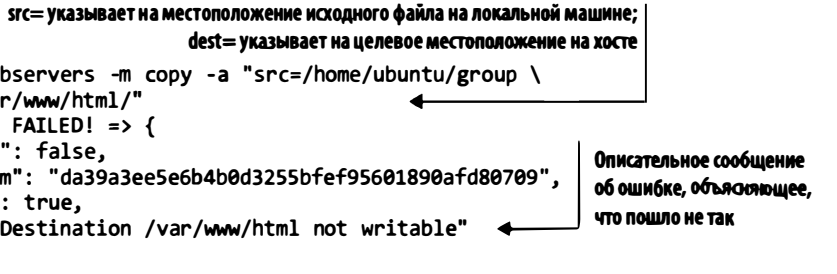
Теперь предположим, что вам нужно выполнить команду на удаленных хостах, которая требует полномочий `sudo`. Представьте, что вы хотите отправить обновленный файл `.html` на десятки веб-серверов, неустанно выполняющих запросы балансировщика нагрузки. Конечно, имеет смысл сделать это за раз, а не повторять операцию отдельно для каждого хоста.

Что такое балансировщик нагрузки

Если вам интересно, балансировщик нагрузки — это сервер или сетевой маршрутизатор, который получает запросы на доступ к сервису и перенаправляет эти запросы на несколько серверов приложений. Балансировщики нагрузки хорошо распределяют запросы между серверами, гарантируя, что ни один из них не будет перегружен, а также перенаправляя запросы от сломанных или недоступных серверов. уществует два широко известных пакета Linux с открытым исходным кодом для балансировки нагрузки — это HAProxy и, в дополнение к функциям веб-сервера, nginx.

Почему бы не попробовать проделать это самому? Посмотрите, что происходит, когда вы пытаетесь использовать модуль копирования для копирования файла из локального домашнего каталога (возможно, файла группы, который вы скопировали ранее) в каталог `/var/www/html/` на удаленном хосте. Если у вашего хоста

уже нет каталога `/var/www/html/`, вы можете получить такой же эффект, заменив любой системный каталог (например, `/etc/`), который не принадлежит вашему пользователю:




```
$ ansible webservers -m copy -a "src=/home/ubuntu/group \
    dest=/var/www/html/"
10.0.3.103 | FAILED! => {
    "changed": false,
    "checksum": "da39a3ee5e6b4b0d3255bfef95601890afd80709",
    "failed": true,
    "msg": "Destination /var/www/html not writable"
}
```

Ой. Судя по сообщению `Destination /var/www/html not write`, у нас проблема с доступом. Похоже, вам придется найти способ расширять свои полномочия. Лучший способ сделать это — отредактировать настройки в файле `/etc/ansible/ansible.cfg`. Как видно из листинга 16.3, я отредактировал раздел `[privilege_escalation]` в `ansible.cfg`, раскомментировав четыре строчки.

Листинг 16.3. Измененные настройки в /etc/ansible/ansible.cfg

```
[privilege_escalation]
become=True
become_method=sudo
become_user=root
become_ask_pass=True
```

Когда вы снова запустите операцию копирования, добавив аргумент `--ask-become-pass`, Ansible считает обновленный файл конфигурации и запросит пароль sudo удаленного пользователя ubuntu. На этот раз все получится:

```
$ ansible --ask-become-pass webservers -m copy -a "src=/home/ubuntu/group \
    dest=/var/www/html/"
SUDO password:
10.0.3.103 | SUCCESS => {
    "changed": true,
    "checksum": "da39a3ee5e6b4b0d3255bfef95601890afd80709",
    "dest": "/var/www/html/stuff.html",
    "gid": 0,
    "group": "root",
    "md5sum": "d41d8cd98f00b204e9800998ecf8427e",
    "mode": "0644",
    "owner": "root",
    "size": 0,
    "src": "/home/ubuntu/.ansible/tmp/
        ansible-tmp-1509549729.02-40979945256057/source",
    "state": "file",
    "uid": 0
}
```

Войдите на удаленный сервер, чтобы убедиться, что файл скопирован. Кстати, с точки зрения безопасности было бы глупо оставить копию файла в корневом каталоге. Мы сделали это просто для примера. Пожалуйста, в будущем не оставляйте его там.

16.4. Сценарии Ansible playbook

Как вы уже заметили, к выполнению некоторых основных заданий Ansible может приступать через минуту или две после установки. Но разнообразием такие задачи отличаться не будут. Если вы хотите использовать всю мощь инструмента Ansible, чтобы он мог организовать автоматизированную многоуровневую инфраструктуру, которую я описал во введении к главе, вам нужно научиться пользоваться сценариями playbook. Playbook предоставляет способ тщательного определения политики и действий, которые должен запускать Ansible. Кроме того, это простой способ обмениваться рабочими профилями конфигурации. Playbook можно использовать:

- как простой, автономный скрипт;
- как ссылку, указывающую на ресурсы, распределенные по специально структурированному дереву каталогов (для более сложных сред).

16.4.1. Написание простого playbook

Разберемся, как создать простой playbook, который может за один раз обеспечить относительно простой веб-сервер всем необходимым. Для этого вы будете использовать модули (например, модуль копирования, который видели ранее), задачи для запуска системных действий Linux и обработчики для динамического реагирования на системные события. Во-первых, убедитесь, что файл `hosts` в `/etc/ansible/` не требует обновления (листинг 16.4).

Листинг 16.4. Простой файл /etc/ansible/hosts

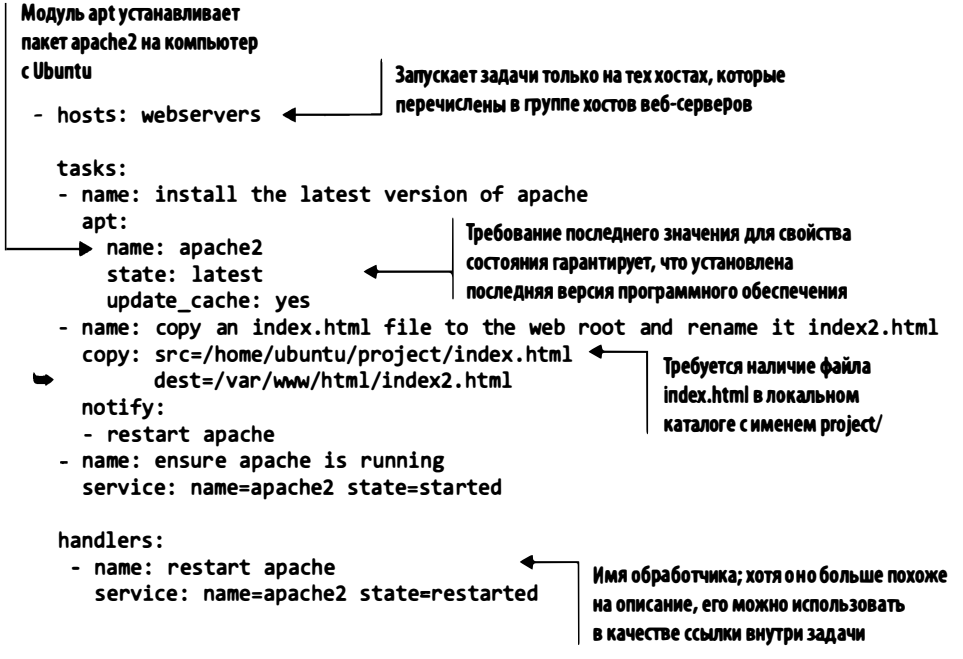
```
10.0.3.103
10.0.3.96
```

Затем вам нужно создать файл в формате YAML с названием `site.yml`. YAML — это язык форматирования текста, связанный с более широко используемой нотацией объектов JavaScript (JSON). Несмотря на то что вы должны быть внимательны, чтобы получить правильные отступы, формат YAML создает профили конфигурации, которые легко читать, понимать и редактировать.

После запуска строки, содержащей три дефиса (`---`), ваш файл будет состоять из трех разделов: `hosts`, `tasks` и `handlers`. В этом случае раздел `hosts` сообщает Ansible о применении действия из playbook ко всем адресам из группы веб-серверов в файле

hosts. Раздел tasks (с отступом на столько же пробелов, что и hosts) вводит три задачи (или модуля): apt для установки веб-сервера Apache, copy для копирования локального файла в корневой каталог веб-документа и service, очень похожий на systemctl в среде systemd, который нужен, чтобы убедиться, что Apache работает (листинг 16.5).

Листинг 16.5. Простой Ansible playbook под названием site.yml




```
- hosts: webservers

  tasks:
  - name: install the latest version of apache
    apt:
      name: apache2
      state: latest
      update_cache: yes
  - name: copy an index.html file to the web root and rename it index2.html
    copy: src=/home/ubuntu/project/index.html
    ➥     dest=/var/www/html/index2.html
    notify:
    - restart apache
  - name: ensure apache is running
    service: name=apache2 state=started

  handlers:
   - name: restart apache
     service: name=apache2 state=restarted
```

Чтобы проверить все самостоятельно, вы можете создать простой файл index.html и сохранить его в каталоге на сервере Ansible (для своей лаборатории Ansible я использовал контейнер LXC). Убедитесь, что вы правильно указали расположение файла в playbook (как это было в предыдущем примере — copy: src=). Если хотите, файл может содержать только фразу Hello World. Это нужно лишь для того, чтобы подтвердить, что playbook работает. После запуска playbook копия этого файла должна появиться в корневом каталоге веб-документа.

Обратите также внимание на строку notify: в задаче копирования в предыдущем примере. Как только задача копирования завершена, notify запускает обработчик с именем restart apache, который, в свою очередь, гарантирует, что Apache перезапущен и работает правильно.

Когда вы создаете собственные сценарии playbook, вам определенно потребуется больше информации о синтаксисе и функциях. В этом вам поможет запуск ansible-doc и имя конкретного модуля:

```
$ ansible-doc apt
```

Запуск Playbook

Предположим, что файл `/etc/ansible/ansible.cfg` правильно настроен для обработки аутентификации хоста и вы готовы использовать команду `ansible-playbook` для запуска playbook. По умолчанию команда будет задействовать хосты, перечисленные в `/etc/ansible/hosts`, а также вы можете указать `-i`, чтобы выбрать другой файл. Вот пример:

```
$ ansible-playbook site.yml
SUDO password:
PLAY ***************************************************
TASK [setup] *******************************************
ok: [10.0.3.96]
TASK [ensure apache is at the latest version] ***
changed: [10.0.3.96]
TASK [copy an index.html file to the root directory] ****
changed: [10.0.3.96]
TASK [ensure apache is running] ************************
ok: [10.0.3.96]
RUNNING HANDLER [restart apache] ***********************
changed: [10.0.3.96]
PLAY RECAP *********************************************
10.0.3.96 : ok=5 changed=3 unreachable=0 failed=0
```

Краткое изложение цели задачи

Сообщение ok показывает, что задача успешно завершена

Сводка результатов запуска playbook

Это успех! С помощью единственной команды вы создали веб-сервер, работающий на всех хостах, перечисленных в файле `hosts`. Не верите? Укажите в браузере URL-адрес, который должен использоваться скопированным файлом `index2.html` (в моем случае это `10.0.3.96/index2.html`). В итоге вы должны увидеть свой файл `index2.html`.

16.4.2. Создание многоуровневых ролевых сценариев playbook

Как только инфраструктура, управляемая Ansible, утяжеляется слоями элементов, каждый из которых имеет свои подробные параметры, хранить их все в одном сценарии playbook нецелесообразно. Попробуйте представить себе, каково это — управлять платформой, изображенной ранее на рис. 16.2.

Распределение задач, обработчиков и других типов данных в отдельные каталоги и файлы облегчит вам жизнь. Такая модульная организация также позволяет создавать новые playbook без необходимости снова изобретать колесо: у вас всегда будет полный и легкий доступ ко всему, что вы создали.

Ansible организует свои модульные элементы в роли и даже предоставляет собственный инструмент командной строки для управления существующими ролями и создания необходимой структуры файловой системы для запуска новых ролей. Этот инструмент называется `ansible-galaxy`. На рис. 16.4 показана базовая топология Ansible.

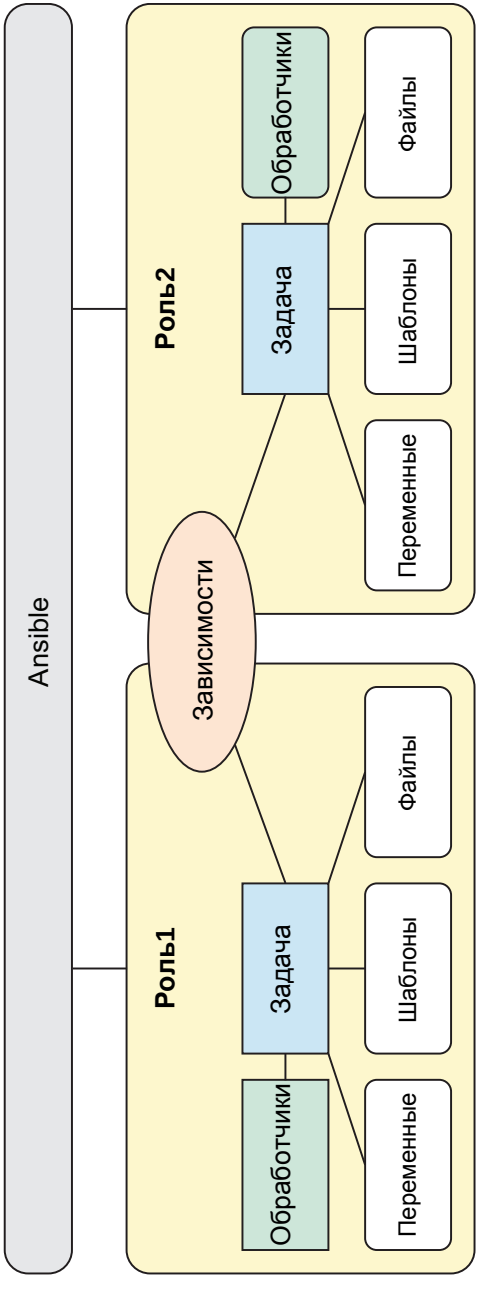


Рис. 16.4. Роли Ansible показаны в качестве отдельных ресурсных групп, включая доступ к системе зависимостей

Создание ролей Ansible

Выберите каталог для использования в качестве корневого для Ansible. Если вы работаете с контейнером или виртуальной машиной, цель которых — действовать как Ansible-сервер, тогда это также можно использовать как корневой каталог документа вашего основного пользователя (`/home/username/`). В корне Ansible вы создадите каталог с именем `role`, а затем перейдете в новый каталог.

После этого инициализируйте каталог с помощью `ansible-galaxy init`, а затем пропишите имя, которое хотите использовать для ролей:

```
$ mkdir roles
$ cd roles
$ ansible-galaxy init web-app
- web-app was created successfully
```

Создается новый каталог под названием `web-app`. Запустите `ls -R`, чтобы рекурсивно вывести список новых подкаталогов и их содержимого, созданных для вас ansible-galaxy:

```
$ cd web-app
$ ls -R
.:
defaults  files  handlers  meta  README.md  tasks  templates  tests  vars
./defaults:
main.yml
./files:
./handlers:
main.yml
./meta:
main.yml
./tasks:
main.yml
./templates:
./tests:
inventory test.yml
./vars:
main.yml
```

Каждый подкаталог в веб-приложении отображается начиная с ./

Некоторые каталоги уже заполнены собственными пустыми файлами playbook

Как Ansible будет использовать данные, содержащиеся в этих каталогах? Здесь есть на что посмотреть: значения переменных и параметры, заданные в этих файлах, часто будут управлять тем, как Ansible управляет ресурсами, запущенными с применением этой конкретной роли.

Прочитайте это все еще один или два раза. Готово? Хорошо, есть две вещи, которые нужно выделить: постоянный контроль (но не всегда же?) и конкретная выбранная роль (вы спросите: у меня что, могут быть другие?).

Все верно. Параметры, добавленные в файлы под каталогом ролей веб-приложения, могут вызываться либо из playbook верхнего уровня, либо с помощью специального действия командной строки. Например, вы можете определить корневое местоположение веб-документа в файле `role/web-app/defaults/main.yml` как

`webroot_location:/var/www/myroot/`. Вызов переменной `webroot_location` вернет значение `/var/www/myroot/`.

Но есть случаи, когда все проходит не так гладко. Вы должны понять, что Ansible был разработан для сред, охватывающих несколько проектов. У вас может быть отдельный playbook для каждого отдельного приложения и для других внутренних служб компании. Ничто не мешает вам управлять более чем одним приложением с одного Ansible-сервера. Вероятнее всего, это будет значить, что вы хотите, чтобы конкретная переменная означала одно для приложения x, а другое — для приложения y.

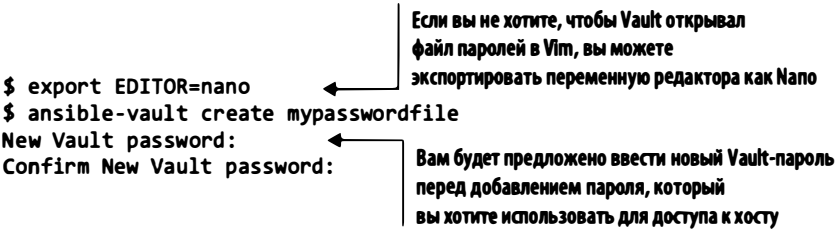
Это подводит нас ко второму важному моменту: каждое приложение или сервис могут быть представлены своей собственной ролью Ansible. Но для того, чтобы несколько ролей могли успешно сосуществовать в одной системе, вам потребуется расставить приоритеты для их перекрывающихся переменных. Ansible сделает это все довольно запутанно, а я могу подвести итог, лишь сказав, что значения, найденные в каталоге ролей `vars/`, переопределяют значения из `/defaults`, а значения, заданные с помощью `-e` (или `--extra-vars=`), превосходят все остальное.

Что может входить в каждые ваши роль/веб-приложение/каталог? Вот краткий список.

- Каталог `vars/` может содержать информацию о расположении файловой системы для ключей шифрования.
- Каталог `templates/` будет содержать шаблоны, предназначенные для установки, например, в виде файлов конфигурации Apache в формате Python `.j2`.
- В каталоге `files/` могут содержаться другие файлы (например, HTML-файл, который вы копировали в предыдущем примере), которые используются для данных хоста.

16.4.3. Управление паролями в Ansible

Вероятнее всего, вам потребуется включить пароли хостов в инфраструктуру Ansible, вы никогда не должны хранить их в виде простых текстовых документов. Никогда! Тем более Ansible предоставляет инструмент под названием Vault, хранящий конфиденциальные данные в зашифрованных файлах, которые при необходимости playbook может безопасно вызвать. Этот фрагмент открывает редактор, в который вы можете ввести новый пароль Vault:



Предположим, что все ваши хосты используют только один пароль, это сработает путем добавления аргумента `--ask-vault-pass` к команде `ansible-playbook`:

```
$ ansible-playbook site.yml --ask-vault-pass
Vault password:
```

К сведению, начиная с версии 2.3, в Ansible также можно использовать то, что называют *переменной с переменным доступом*, которая, по сути, является зашифрованным паролем, хранящимся в текстовом YAML-файле. Это позволяет управлять несколькими паролями.

Резюме

- Инструменты оркестровки позволяют автоматизировать серверную инфраструктуру в масштабе одного или тысячи хостов. Выбор инструмента будет зависеть от навыков вашей команды, потребностей проекта и культуры компании.
- Ansible не требует кодирования, работает по SSH и занимает мало места.
- Сценарии playbook управляют ресурсами через роли — это эффективный и действенный способ управлять безопасностью и ресурсами.

Ключевые понятия

- *DevOps* — это организационная структура проекта, которая помогает командам разработчиков и администраторов ускорить и автоматизировать циклы разработки продуктов.
- *Инструменты оркестровки* позволяют точно составлять сценарий поведения инфраструктуры для достижения желаемых состояний за счет автоматизации.
- *Группа хостов* — это способ организации хостов. Ansible может быть направлен на управление четко определенными подмножествами вашего хост-парка.
- В Ansible playbook *модули* представляют собой предопределенные последовательности команд, выполняемые в хост-системе, *обработчики* — это действия, инициируемые событиями, а *роли* — связанные ресурсы, организованные для обслуживания одного проекта.

Рекомендации по безопасности

- Направление Ansible для доступа к вашим хостам с использованием SSH без пароля, основанного на парах ключей, предпочтительнее, чем ввод паролей в командной строке для каждой операции.
- Никогда не включайте пароли в Ansible playbok или другие текстовые сценарии. Вместо этого используйте Ansible Vault.

Обзор команд

- `add-apt-repository ppa: ansible/ansible` добавляет программный репозиторий Debian Ansible, чтобы инструмент `apt` смог установить Ansible на компьютере с Ubuntu/Debian.
- `ansible webservers -m ping` в группе хостов веб-серверов проверяет все хосты на сетевое подключение.
- `ansible webservers -m copy -a "src=/home/ubuntu/stuff.html dest=/var/www/html/"` копирует локальный файл в указанное место на всех хостах в группе веб-серверов.
- `ansible-doc apt` отображает синтаксис и информацию об использовании модуля `apt`.
- `ansible-playbook site.yml` запускает операцию на основе `playbook site.yml`.
- `ansible-playbook site.yml --ask-vault-pass` использует пароль Vault для аутентификации и выполнения playbook-операций.

Самотестирование

1. Какие из перечисленных инструментов оркестровки лучше всего подойдут для команды разработчиков с небольшим опытом в DevOps, которые строят большую и сложную платформу:
   а) Ansible;
   б) Chef;
   в) Puppet;
   г) Salt?
2. Какие из перечисленных пакетов должны быть установлены на каждом хосте, чтобы Ansible мог работать:
   а) Ansible;
   б) Python;
   в) software-properties-common;
   г) Ansible и Python?
3. Что из перечисленного является в первую очередь проблемой безопасности:
   а) организация хостов в группы хостов;
   б) планирование регулярного тестирования подключения для всех хостов;
   в) разделение переменных среды;
   г) хранение данных в Ansible Vault?
4. Какая команда указывает Ansible автоматически заполнять корневой веб-документ локальным файлом по умолчанию только на тех хостах, где работает Apache:
   а) `ansible all -i copy -a "src =/var/www/html/ dest=/home/ubuntu/stuff.html"`;
   б) `ansible all webservers -m copy -a "/home/ubuntu/stuff.html /var/www/html/"`;

в) `ansible webservers -m copy -a "src=/home/ubuntu/stuff.html dest=/var/www/html/"`;

г) `ansible webservers -m copy -a src=/home/ubuntu/stuff.html dest=/var/www/html/`?

5. Какая из следующих команд создаст каталоги и файлы, необходимые для новой роли Ansible:

   а) `ansible-root-directory/roles/ansible-galaxy init rolename`;

   б) `ansible-root-directory/ansible-galaxy rolename`;

   в) `ansible-root-directory/roles/ansible-init rolename`;

   г) `ansible-root-directory/roles/ansible init rolename`?

Ответы

1 — б; 2 — б; 3 — г; 4 — в; 5 — а.

Заключение

Ну что ж, вот и все. Наше совместное путешествие почти закончено. Если я выполнил свою часть, а вы выполнили свою, то к настоящему времени вы должны были приобрести серьезные навыки работы с Linux. На самом деле, если вы достаточно практиковались, то должны чувствовать себя комфортно, принимая на себя ответственность за многие распространенные задачи по администрированию сервера, и работодатели могут спокойно нанимать вас для выполнения этих задач на своих серверах. Но прежде, чем мы расстанемся, потратим пару минут на то, чтобы пройтись по тому, что вы узнали, и выяснить, куда это может привести вас дальше.

Что вы узнали

Усвоить все многочисленные методы, которые вы рассмотрели в «Linux в действии», будет непросто. Чтобы сделать этот обзор более полезным, я рассортирую большую часть прочитанного в несколько тем:

- виртуализация;
- связь;
- шифрование;
- сетевое взаимодействие;
- управление образами;
- системный мониторинг.

Виртуализация

Работая с виртуальными машинами и контейнерами в главе 2, вы использовали виртуализацию для создания «песочниц» компьютерных сред, где вы можете безопасно экспериментировать с новыми инструментами и технологиями. В главах 6 и 9

вы запустили сеансы `chroot` для восстановления поврежденных конфигураций и файловых систем или для сброса пароля аутентификации. С вашим пониманием технологий виртуализации, а также после знакомства с инфраструктурой управления из главы 16 вы в шаге от глубокого погружения в мир корпоративных облачных и контейнерных вычислений.

Связь

Удаленное подключение играло важную роль почти во всех разделах книги: от удаленного управления сервером (глава 3) до резервного копирования в архивы (глава 4), от предоставления веб-серверов (глава 7) и общего доступа к файлам (глава 8) до мониторинга системы (глава 11). Существует не так много критических административных задач, которые были бы возможны без наличия безопасных и надежных соединений. Теперь у вас есть основные инструменты Linux, необходимые для создания этих соединений, управления ими и устранения в них неполадок.

Шифрование

Бизнес-процессы, которые в значительной степени зависят от открытых сетей, таких как Интернет, будут нуждаться в способах защиты своих данных на любом этапе развития. На данный момент вы готовы решить эту проблему с помощью инструментов шифрования. Вы узнали о шифровании сеансов SSH из главы 3, о совместном использовании зашифрованных сетевых файлов — из главы 8, шифровании сайтов и сертификатах TLS — из главы 9 и VPN-сетях — из главы 10.

Сетевое взаимодействие

В сложном мире вашей организации потребуются непростые сетевые решения, чтобы все работало. Linux определенно играет свою роль в области сетевых технологий, и из этой книги вы узнали, как использовать Linux в качестве платформы для создания сложных решений для подключения таких сетей, как VPN и DMZ (глава 10). Вы также увидели, как можно обмениваться данными по частной сети с использованием NFS, в главе 12 и познакомились с целым стеком инструментов оптимизации производительности сети в главе 14.

Управление образами

Вы видели, как полные образы файловой системы можно использовать для резервного копирования и восстановления данных. Как вы узнали, прочитав о клонировании и совместном использовании виртуальных машин VirtualBox в главе 2, образы также важны для управления сервером. Это особенно актуально и полезно в мире управления облачной инфраструктурой.

Системный мониторинг

В главе 13 вы изучили работу текущего мониторинга системы. Вы использовали инструменты мониторинга для предотвращения проблем безопасности и повышения производительности системы, которые связаны с четырьмя основными элементами компьютера: процессором, памятью, устройством хранения и сетью. Вспоминаете?

Что дальше

Если вам нужна эта книга для своих собственных конкретных целей, то, пожалуйста, не позволяйте мне останавливать вас. Добейтесь своего. Но если вы ищете идеи для приключений на следующем уровне, подумайте вот о чем.

- Номер один (два и три): получить реальный практический опыт. Начните работать в командной строке и не прекращайте. Используйте ее для управления чем угодно — от простой передачи файлов до просмотра видео на ноутбуке, подключенном к Wi-Fi. В качестве дополнительного бонуса, когда вы запускаете видеоприложение VLC из командной строки, вам будут показаны все виды вывода процесса. Изучив результаты, вы сможете выяснить, почему любимый фильм неожиданно испортился в середине просмотра.
- Вытащите пустой USB-накопитель и создайте себе аварийный инструментарий. Загрузите его с образом загрузки Linux, как рассказывалось в главе 6, и добавьте некоторые из ваших любимых инструментов для устранения неполадок и восстановления. Проверьте его хотя бы один раз, прежде чем убрать в дальний ящик.
- Копайте глубже в направлении сценариев Bash. Находите примеры сценариев, в которых есть обработка событий, переменные, циклы и входные данные. Рассмотрите возможность создания сценариев для автоматизации любых задач администрирования, которые вы обычно выполняете. Полное руководство Vivek Gite по написанию сценариев Bash — хорошее место для начала: **bash.cyberciti.biz/guide/Main_Page**.
- Исследуйте мир зашифрованных накопителей, используя такие инструменты, как eCryptfs и cryptsetup. Если вы носите с собой конфиденциальные данные на своем ноутбуке или USB-накопителе, вам следует серьезно подумать о том, что может произойти, если ваши устройства попадут в чужие руки.
- Если у вас есть навыки кодирования, вы можете взять ядро Linux и добавить свои собственные настройки. Конечно, вам нужно привлечь Линуса Торвальдса на свою сторону, прежде чем ваши изменения будут приняты в официальное ядро, но вы можете развернуть свою версию для личного использования. Так вы узнаете, как Linux работает «под капотом».
- Примените свои навыки работы с Linux-сервером в облачном мире. Вы будете гораздо эффективнее работать с AWS или даже Azure, если поймете, что движет облачной инфраструктурой. Не знаете, с чего начать? Мэннинг определенно поможет. Вот моя книга *Learn Amazon Web Services in a Month of Lunches* (2017),

Amazon Web Services in Action братьев Виттинг, 2-е изд. (2018), *Azure in Action* (Крис Хэй и Брайан Х. Принс, 2010) и *Learn Azure in a Month of Lunches* (Айан Фоулдс, 2018). Выбор за вами.

- Используйте контейнеры с помощью таких технологий, как Docker и Kubernetes. Учитывая скорость и масштабируемость контейнеров, они будущее корпоративных вычислений. *Kubernetes in Action* (Марко Лукша, 2017), *Docker in Action* (Джефф Николофф, 2016) и *Docker in Practice* (Ян Миелл и Эйдан Хобсон Сайерс, 2016) — все это великолепные ресурсы.

Ресурсы

Что? Вы хотите больше? Отлично. Но только один раз.

Вам, безусловно, стоит посетить мой собственный сайт **https://bootstrap-it.com**, где вы найдете оригинальный контент и информацию о других моих книгах и курсах, все из которых сертифицированы как для Linux, так и для облачных сред. Не стесняйтесь следить за мной в «Твиттере» (**@davidbclinton**) и быть среди первых, кто узнает о моих очередных мрачных планах по завоеванию мира.

Посетите форум Manning по книге *«Linux в действии»* (**forums.manning.com/forums/linux-in-action**). Если у вас возникли проблемы с каким-либо из проектов или вы просто хотите рассказать о своих последних успехах, поделитесь ими с сообществом. Я внимательно слежу за этим форумом.

Используйте онлайн-ресурсы. Замечательные люди десятилетиями предоставляют отличную документацию и руководства по работе с Linux. Существует слишком много форумов, блогов, IRC-каналов, групп Slack и сайтов документации, чтобы перечислять их здесь. Но **www.tldp.org/FAQ/Linux-FAQ/online-resources.html** — отличное место для начала.

Ваш любимый интернет-поисковик должен быть первым помощником. И не забывайте, что (удивительно) `man` работает в Интернете почти так же, как в командной строке. Первый результат, который вы получите при поиске `man selinux` в вашем браузере, должен быть тем же документом `man`, который вы получите в оболочке.

Я желаю вам больших успехов в изучении Linux. Пока, и оставайтесь на связи!

Дэвид Клинтон

Приложение. Обзор команд по главам

Глава 1. Добро пожаловать в Linux

- `ls -lh/var/log` перечисляет содержимое каталога и выводит полные, удобные для чтения сведения о папке `/var/log/`.
- `cd` возвращает вас в ваш домашний каталог.
- `cp file1 newdir` копирует файл с именем `file1` в каталог с именем `newdir`.
- `mvfile? /some/other/directory/` перемещает все файлы, содержащие в имени буквы `file` и еще один символ в целевой локации.
- `rm -r *` удаляет все файлы и каталоги в текущем расположении. Использовать с большой осторожностью.
- `man sudo` открывает man-файл документации со сведениями о команде `sudo`.

Глава 2. Виртуализация Linux: создание безопасной и простой рабочей среды

- `apt install virtualbox` использует APT для установки программного пакета из удаленного хранилища.
- `dpkg -i skypeforlinux-64.deb` напрямую устанавливает загруженный пакет Debian на компьютер с Ubuntu.
- `wget https://example.com/document-to-download` использует консольную программу `wget` для загрузки файла.
- `dnf update`, `yum update` или `apt update` синхронизирует локальный программный индекс с тем, что доступно в онлайн-хранилищах.
- `shasum ubuntu-16.04.2-server-amd64.iso` вычисляет контрольную сумму для загруженного файла, чтобы подтвердить, что она соответствует данному значению. Это означает, что содержимое не было повреждено при транспортировке.

- `vboxmanage clonevm Kali-Linux-template --name newkali` использует инструмент `vboxmanage` для клонирования существующей виртуальной машины.
- `lxc-start -d -n myContainer` запускает существующий контейнер LXC.
- `ip addr` отображает информацию о каждом из сетевых интерфейсов системы (включая их IP-адреса).
- `exit`: выход из сеанса оболочки без выключения машины.

Глава 3. Удаленное подключение: безопасный доступ к машинам по сети

- `dpkg -s openssh-client` проверяет состояние пакета программного обеспечения на основе APT.
- `systemctl status ssh` проверяет состояние системного процесса (через `systemd`).
- `systemctl start ssh` запускает системный процесс.
- `ip addr` показывает все сетевые интерфейсы на компьютере.
- `ssh-keygen` генерирует новую пару ключей SSH.
- `$ cat .ssh/id_rsa.pub | ssh ubuntu@10.0.3.142 "cat >> .ssh/authorized_keys"` считывает локальный ключ и копирует его на удаленный компьютер.
- `ssh-copy-id -i .ssh/id_rsa.pub ubuntu@10.0.3.142` безопасно копирует ключи шифрования (рекомендовано и является стандартом).
- `ssh -i .ssh/mykey.pem ubuntu@10.0.3.142` определяет конкретную пару ключей.
- `scp myfile ubuntu@10.0.3.142:/home/ubuntu/myfile` безопасно копирует локальный файл на удаленный компьютер.
- `ssh -X ubuntu@10.0.3.142` позволяет войти на удаленный хост для графического сеанса.
- `ps -ef | grep init` отображает все запущенные в данный момент системные процессы и отфильтровывает результаты, содержащие строку `init`.
- `pstree -p` отображает все запущенные в данный момент системные процессы в виде дерева.

Глава 4. Управление архивами: создание резервных копий или копирование целых файловых систем

- `df -h` выводит все разделы, активные в настоящий момент, в удобном для чтения формате.
- `tar czvf archivename.tar.gz /home/myuser/Videos/*.mp4` создает сжатый архив видеофайлов в указанном каталоге.

- `split -b 1G archivename.tar.gz archivename.tar.gz.part` разделяет большой файл на группу меньших по указанному размеру.
- `find /var/www/ -iname "*.mp4" -exec tar -rvf videos.tar {} \;` находит файлы по заданному критерию и передает их имена `tar` для включения в архив.
- `chmod o-r /bin/zcat` удаляет права на чтение для остальных.
- `dd if=/dev/sda2 of=/home/username/partition2.img` создает образ раздела `sda2` и сохраняет его в домашнем каталоге.
- `dd if=/dev/urandom of=/dev/sda1` перезаписывает раздел случайными данными, чтобы уничтожить старые данные.

Глава 5. Автоматизированное администрирование: настройка автоматического резервного копирования

- `#!/bin/bash` (так называемая строка шебанга) сообщает Linux, какой интерпретатор оболочки вы собираетесь использовать для сценария.
- `||` добавляет условие выбора в сценарий. Читается как «команда слева успешна» или «выполнить команду справа».
- `&&` — вставляет в сценарий условие добавления. Читается как «если команда слева успешна» и «выполнить команду справа».
- `test -f /etc/filename` проверяет наличие указанного файла или каталога.
- `chmod + x upgrade.sh` делает файл сценария готовым к работе.
- `pip3 install --upgrade --user awscli` устанавливает интерфейс командной строки AWS, используя менеджер пакетов Python.
- `aws s3 sync /home/username/dir2backup s3://linux-bucket3040` синхронизирует содержимое локального каталога с указанной корзиной S3.
- `21 5 * * 1 root apt update && apt upgrade` (директива `cron`) выполняет обе команды `apt` в 5:21 каждое утро.
- `NOW=$(date +"%m_%d_%Y")` присваивает переменной сценария текущую дату.
- `systemctl start site-backup.timer` активирует системный таймер systemd.

Глава 6. Инструменты для критических ситуаций: создание устройства для восстановления системы

- `sha256sum systemrescuecd-x86-5.0.2.iso` вычисляет контрольную сумму SHA256 файла в формате ISO.
- `isohybrid systemrescuecd-x86-5.0.2.iso` добавляет запись MBR с поддержкой USB-носителей к образу загрузочного диска.

- `dd bs=4M if=systemrescuecd-x86-5.0.2.iso of=/dev/sdb && sync` записывает образ загрузочного диска на пустой носитель.
- `mount /dev/sdc1 /run/temp-directory` монтирует раздел в каталог в загрузочной файловой системе.
- `ddrescue -d /dev/sdc1 /run/usb-mount/sdc1-backup.img /run/usb-mount/ sdc1-backup.logfile` сохраняет файлы из поврежденного раздела в образ с именем `sdc1-backup.img` и записывает события в файл журнала.
- `chroot /run/mountdir/` открывает оболочку от имени администратора в файловой системе.

Глава 7. Веб-серверы: создание сервера MediaWiki

- `apt install lamp-server^` — единственная команда Ubuntu, устанавливающая все элементы сервера LAMP.
- `systemctl enable httpd` запускает Apache на CentOS при каждой загрузке системы.
- `firewall-cmd --add-service = http --permanent` разрешает HTTP-трафик браузера в системе CentOS.
- `mysql_secure_installation` сбрасывает ваш административный пароль и повышает безопасность базы данных.
- `mysql -u root -p` позволяет войти в MySQL (или MariaDB) в качестве root-пользователя.
- `CREATE DATABASE newdbname;` создает новую базу данных в MySQL (или MariaDB).
- `yum search php- | grep mysql` ищет доступные пакеты, связанные с PHP, на компьютере с CentOS.
- `apt search mbstring` выполняет поиск доступных пакетов, связанных с многобайтовым кодированием строк.

Глава 8. Совместное использование файлов в сети: создание сервера для совместного использования файлов Nextcloud

- `a2enmod rewrite` подключает модуль перезаписи, поэтому Apache может редактировать URL-адреса при их перемещении между клиентом и сервером.
- `nano /etc/apache2/sites-available/nextcloud.conf` создает или редактирует файл конфигурации хоста Apache для Nextcloud.
- `chown -R www-data: www-data /var/www/nextcloud/` меняет владельца и группу всех файлов сайта на пользователя www-data.

- `sudo -u www-data php occ list` использует CLI Nextcloud для вывода списка доступных команд.
- `aws s3 ls s3://nextcloud32327` выводит содержимое раздела S3.

Глава 9. Защита вашего веб-сервера

- `firewall-cmd --permanent --add-port=80/tcp` открывает порт 80 для входящего трафика HTTP и сохраняет эту конфигурацию для использования при загрузке.
- `firewall-cmd --list-services` выводит список активных в настоящий момент правил системы firewalld.
- `ufw allow ssh` открывает порт 22 для SSH-трафика, используя UncomplicatedFirewall (ufw) в Ubuntu.
- `ufw delete 2` удаляет второе правило ufw, как показано командой `ufw status`.
- `ssh -p53987 username@remote_IP_or_domain` открывает сессию SSH по нестандартному порту.
- `certbot --apache` конфигурирует веб-сервер Apache для использования сертификатов шифрования Let's Encrypt.
- `selinux-activate` активирует SELinux на машине Ubuntu.
- `setenforce 1` переключает принудительный режим в конфигурации SELinux.
- `ls -Z /var/www/html/` отображает контекст безопасности файлов в указанном каталоге.
- `usermod -aG app-data-group otheruser` добавляет пользователя `otheruser` в системную группу `app-data-group`.
- `netstat -npl` сканирует открытые (прослушиваемые) сетевые порты на сервере.

Глава 10. Защита сетевых соединений: создание VPN или DMZ

- `hostname OpenVPN-Server` задает приглашение командной строки, чтобы упростить отслеживание того, на какой сервер вы зашли.
- `cp -r /usr/share/easy-rsa/ /etc/openvpn` копирует сценарии Easy RSA и файлы конфигурации среды в рабочий каталог OpenVPN.
- `./build-key-server server` генерирует пару ключей RSA с именем `server`.
- `./pkitool client` генерирует клиентский набор ключей из существующей инфраструктуры ключей RSA.
- `openvpn --tls-client --config /etc/openvpn/client.conf` запускает OpenVPN на клиенте Linux, используя настройки из файла `client.conf`.

- `iptables -A FORWARD -i eth1 -o eth2 -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT` позволяет передавать данные между сетевыми интерфейсами eth1 и eth2.
- `man shorewall-rules` отображает документацию по файлу правил, используемому Shorewall.
- `systemctl start shorewall` запускает брандмауэр Shorewall.
- `vboxmanage natnetwork add --netname dmz --network "10.0.1.0/24" -enable --dhcp on` использует CLI VirtualBox для создания и настройки виртуальной сети NAT с DHCP для виртуальных машин VirtualBox.
- `vboxmanage natnetwork start --netname dmz` запускает виртуальную сеть NAT.
- `dhclient enp0s3` запрашивает IP-адрес для интерфейса enp0s3 у сервера DHCP.

Глава 11. Мониторинг системы: работа с файлами журналов

- Alt+F<n> открывает виртуальную консоль из оболочки без графического интерфейса.
- `journalctl -n 20` отображает 20 самых последних записей журнала.
- `journalctl --since 15:50:00 --until 15:52:00` отображает только события, произошедшие от одного указанного времени до другого.
- `systemd-tmpfiles --create --prefix /var/log/journal` дает команду systemd создавать и поддерживать постоянный журнал, а не файл, который уничтожается при каждой загрузке.
- `cat /var/log/auth.log | grep -B 1 -A 1 failure` отображает совпадающие строки вместе со строками непосредственно перед и после.
- `cat /var/log/mysql/error.log | awk '$3 ~/[Warning]/' | wc` ищет в журнале ошибок MySQL события, классифицированные как предупреждение.
- `sed "s/^[0-9]//g" numbers.txt` удаляет числа в начале каждой строки файла.
- `tripwire --init` инициализирует базу данных при установке Tripwire.
- `twadmin --create-cfgfile --site-keyfile site.key twcfg.txt` создает новый зашифрованный файл `tw.cfg` для Tripwire.

Глава 12. Совместное использование данных в частной сети

- `/home 192.168.1.11(rw,sync)` (запись в файле сервера NFS `/etc/exports`) определяет разделяемый ресурс удаленного клиента.
- `firewall-cmd --add-service=nfs` открывает брандмауэр CentOS для клиентского доступа к вашему ресурсу NFS.

- `192.168.1.23:/home /nfs/home nfs` (типичная запись в файле `/etc/fstab` клиента NFS) монтирует ресурс NFS.
- `smbpasswd -a sambauser` добавляет возможность использовать Samba (и уникальный пароль) к существующей учетной записи Linux.
- `nano /etc/samba/smb.conf` редактирует конфигурационный файл Samba на сервере.
- `smbclient //localhost/sharehome` выполняет вход в систему для использования локального ресурса Samba с учетной записью Samba.
- `ln -s /nfs/home/ /home/username/Desktop/` создает символическую ссылку, позволяющую пользователю легко открыть ресурс NFS, щелкнув на ярлыке на Рабочем столе.

Глава 13. Устранение проблем производительности системы

- `uptime` возвращает средние значения загрузки процессора за последние 1, 5 и 15 минут.
- `cat /proc/cpuinfo | grep processor` возвращает количество процессоров в системе.
- `top` отображает статистику в реальном времени для работающих процессов Linux.
- `killall yes` закрывает все запущенные экземпляры команды `yes`.
- `nice --15 /var/scripts/mybackup.sh` повышает приоритет потребления системных ресурсов для сценария `mybackup.sh`.
- `free -h` отображает общее и доступное пространство ОЗУ в системе.
- `df -i` отображает общее и доступное количество инодов для каждой файловой системы.
- `find . -xdev -type f | cut -d "/" -f 2 | sort | uniq -c | sort -n` подсчитывает и выводит количество файлов в родительском каталоге.
- `apt-get autoremove` удаляет старые и неиспользуемые заголовки ядра.
- `nethogs eth0` отображает процессы и передает данные, связанные с сетевыми подключениями, использующими интерфейс eth0.
- `tc qdisc add dev eth0 root netem delay 100ms` замедляет все сетевые передачи через интерфейс eth0 на 100 миллисекунд.
- `nmon -f -s 30 -c 120` записывает в файл данные из серии сканирований `nmon`.

Глава 14. Устранение неполадок в сети

- `ip addr` показывает активные интерфейсы в системе Linux. Вы можете сократить команду до `ip a` или удлинить до `ip address`. Как хотите.
- `lspci` перечисляет устройства PCI, которые в данный момент подключены к вашему компьютеру.

- `dmesg | grep -A 2 Ethernet` ищет в журналах `dmesg` совпадения со строкой *Ethernet* и отображает найденное вместе с последующими двумя строками вывода.
- `ip route add default` через `192.168.1.1 dev eth0` вручную устанавливает новый сетевой маршрут для компьютера.
- `dhclient enp0s3` запрашивает динамический (DHCP) IP-адрес для интерфейса enp0s3.
- `ip addr add 192.168.1.10/24 dev eth0` назначает статический IP-адрес интерфейсу eth0, который не сохранится после следующего перезапуска системы.
- `ip link set dev enp0s3 up` запускает интерфейс enp0s3 (полезно после редактирования конфигурации).
- `netstat -l | grep http` сканирует локальный компьютер на наличие веб-службы, прослушивающей порт 80.
- `nc -z -v bootstrap-it.com 443 80` сканирует удаленный сайт на наличие служб, прослушивающих порты 443 или 80.

Глава 15. Устранение неполадок с периферийными устройствами

- `lshw -c memory` (или `lshw -class memory`) отображает раздел памяти профиля оборудования системы.
- `ls /lib/modules/`uname -r`` показывает модули для вашего текущего активного ядра в каталоге `/lib/modules/`.
- `lsmod` перечисляет все активные модули.
- `modprobe -c` перечисляет все доступные модули.
- `find /lib/modules/$(uname -r) -type f -name ath9k*` ищет файл среди доступных модулей ядра с именем, начинающимся с ath9k.
- `modprobe ath9k` загружает указанный модуль в ядро.
- `GRUB_CMDLINE_LINUX_DEFAULT="systemd.unit=runlevel3.target"` (в файле `/etc/default grub`) загружает Linux в многопользовательском неграфическом режиме.
- `lp -H 11:30 -d Brother-DCP-7060D /home/user/myfile.pdf` ставит задание на печать в очередь на принтер Brother в 11:30 UTC.

Глава 16. Инструменты DevOps: развертывание серверной среды с использованием Ansible

- `add-apt-repository ppa: ansible/ansible` добавляет программный репозиторий Debian Ansible, чтобы инструмент apt смог установить Ansible на компьютере с Ubuntu/Debian.
- `ansible webservers -m ping` в группе хостов веб-серверов проверяет все хосты на наличие сетевого подключения.

- `ansible webservers -m copy -a "src=/home/ubuntu/stuff.html dest=/var/www/html/"` копирует локальный файл в указанное место на всех хостах в группе веб-серверов.
- `ansible-doc apt` отображает синтаксис и информацию об использовании модуля apt.
- `ansible-playbook site.yml` запускает операцию на основе `playbook site.yml`.
- `ansible-playbook site.yml --ask-vault-pass` использует пароль Vault для аутентификации и выполнения playbook-операций.

Дэвид Клинтон

Linux в действии

Перевел с английского С. Черников

Заведующая редакцией	*Ю. Сергиенко*
Руководитель проекта	*С. Давид*
Ведущий редактор	*Н. Гринчик*
Художественный редактор	*С. Заматевская*
Корректоры	*О. Андриевич, Е. Павлович*
Верстка	*Г. Блинов*

Изготовлено в России. Изготовитель: ООО «Прогресс книга».
Место нахождения и фактический адрес: 194044, Россия, г. Санкт-Петербург,
Б. Сампсониевский пр., д. 29А, пом. 52. Тел.: +78127037373.

Дата изготовления: 06.2019. Наименование: книжная продукция. Срок годности: не ограничен.

Налоговая льгота — общероссийский классификатор продукции ОК 034-2014, 58.11.12 — Книги печатные профессиональные, технические и научные.

Импортер в Беларусь: ООО «ПИТЕР М», 220020, РБ, г. Минск, ул. Тимирязева, д. 121/3, к. 214, тел./факс: 208 80 01.

Подписано в печать 28.05.19. Формат 70×100/16. Бумага офсетная. Усл. п. л. 33,540. Тираж 1000. Заказ 4615.

Отпечатано в АО «Первая Образцовая типография». Филиал «Чеховский Печатный Двор»
142300, Московская область, г. Чехов, ул. Полиграфистов, 1
Сайт: www.chpd.ru, E-mail: sales@chpd.ru
тел: 8(499) 270-73-59

Linux
В ДЕЙСТВИИ

Без практики ничему нельзя научиться, и Linux не исключение. Книга «Linux в действии» поможет приобрести навыки защиты файлов, папок и серверов, безопасной установки патчей и приложений, а также управления сетью. Дэвид Клинтон предлагает решения каждодневных задач, стоящих перед профессиональными разработчиками, администраторами и специалистами DevOps.

В книге описываются 12 реальных проектов, в том числе автоматизация системы резервного копирования и восстановления, настройка личного файлового облака в стиле Dropbox и создание собственного сервера MediaWiki. На интересных примерах вы изучите виртуализацию, аварийное восстановление, обеспечение безопасности, резервное копирование, внедрение DevOps и устранение неполадок системы. Каждая глава заканчивается обзором практических рекомендаций, глоссарием новых терминов и упражнениями.

В этой книге:

- Настройка безопасной среды Linux.
- Организация защищенных удаленных подключений.
- Создание средства восстановления системы.
- Исправление и обновление вашей системы.

Предварительный опыт администрирования Linux не требуется.

Дэвид Клинтон
сертифицированный администратор Linux, опытный преподаватель. Автор множества видеокурсов, книг, включая бестселлер Learn Amazon Web Services in a Month of Lunches.

Все, что вам понадобится, чтобы начать администрировать Linux. Речь идет не о том, как использовать систему Linux, а о том, как позаботиться о ней.

Мацей Юрковский,
Grupa Pracuj

Важное руководство по Linux с множеством реальных примеров.

Дарио Виктор Дуран,
HiQ Стокгольм

Содержит описание множества функций Linux, которые сделают вашу жизнь намного проще.

Дженс Кристиан Б. Мэдсен,
IT Relation

ПИТЕР®

Заказ книг:
тел.: (812) 703-73-74
books@piter.com

WWW.PITER.COM
каталог книг и интернет-магазин

instagram.com/**piterbooks**
youtube.com/**ThePiterBooks**
vk.com/**piterbooks**
facebook.com/**piterbooks**

ISBN 978-5-4461-1199-2