Оглавление

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.

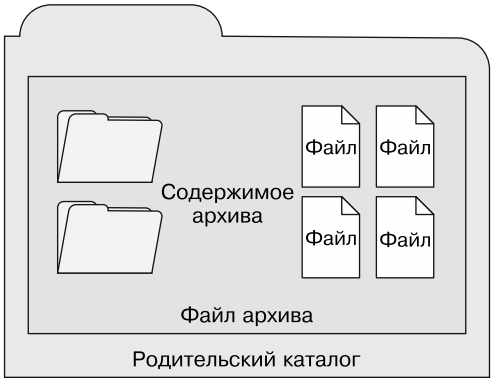

Рис. 4.1. Файлы и каталоги можно объединить в архивный файл и сохранить в файловой системе

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.

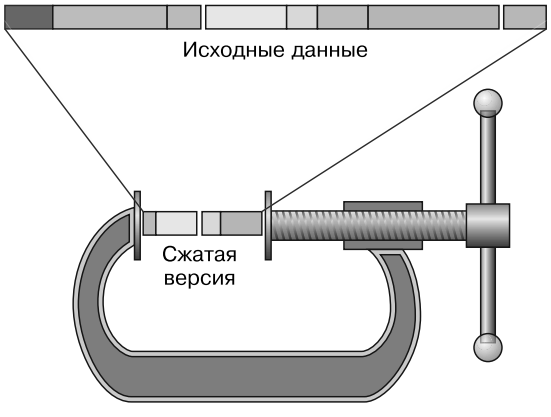

Рис. 4.2. Сжатие объектов путем устранения статистической избыточности и/или удаления менее важных частей файла

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
```

Этот раздел был создан во время установки для включения загрузки UEFI

sdb1 — это флеш-накопитель USB, содержащий живой образ Ubuntu

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
```

Аргумент v задает перечисление имен всех файлов, добавленных в архив

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
```