Изучаем UNIX: Эффективное использование rsync

Мартин Стрейчер (Martin Streicher)
независимый web-разработчик
IBM

11.02.2010

Синхронизация множества компьютеров может оказаться очень сложной процедурой. К счастью, существует мощный инструмент, который может облегчить эту задачу: `rsync`.

Больше статей из этой серии

За последние 20 лет применение компьютерных сетей стало чрезвычайно широким. Это произошло главным образом благодаря развитию Интернет, инвестициям в национальную и международную сетевую инфраструктуру и падению цен на сетевое и компьютерное оборудование. Сегодня сети являются повсеместными, и всё новые приложения увеличивают требования к масштабируемости и скорости сетей. Когда-то Интернет начинался с нескольких небольших рабочих станций, но сейчас он и его частные аналоги соединяют бесчисленное количество компьютеров.

Часто используемые сокращения

- **FTP**: File Transfer Protocol (протокол передачи файлов)
- **WebDAV**: Web-based Distributed Authoring and Versioning (основанный на Web, распределенный протокол с поддержкой авторства и версионности файлов)

На протяжении этого же периода UNIX® также рос и предлагал для использования все более мощное ПО. Протокол FTP был одним из первых инструментов для обмена файлами между системами, он широко распространен и в наши дни. Команда `rcp` (сокращение от "remote copy" – удаленное копирование) была шагом вперед по сравнению с FTP, так как она не только предоставляла возможности стандартной утилиты `cp`, но и копировала файлы с одной машины на другую. `rdist`, основанная на `rcp`, автоматически распространяла файлы с одной машины на множество систем.

Сегодня все эти инструменты устарели, например, `rcp` и `rdist` не обеспечивают безопасность при передаче файлов. Теперь их место занимает `scp`. Хотя FTP все так же широко распространен, но везде, по возможности, следует использовать SFTP (Secure FTP), безопасную версию FTP. Есть также и другие возможности для обмена файлами, например WebDAV и BitTorrent™. Конечно, чем больше машин у вас есть, тем сложнее поддерживать их в синхронизованном или хотя бы известном состоянии. При работе с `scp` и WebDAV для этого необходимо написать свой собственный сценарий, выполняющий синхронизацию.

Идеальным инструментом для распределения файлов является `rsync`. Он умеет возобновлять передачу файлов после разрыва соединения, передает только те куски файла, которые различаются в исходном файле и его копии назначения, а также может выполнять полное или инкрементное резервное копирование. Кроме того, он доступен на всех разновидностях UNIX, в том числе Mac OS X, благодаря чему с его помощью можно легко связать практически любые версии UNIX.

Для знакомства с `rsync` сначала рассмотрим типичные варианты его использования, а затем перейдем к более сложным возможностям его применения. Для демонстрации работы `rsync` я буду использовать Mac OS X версии 10.5, Leopard (разновидность FreeBSD) и Ubuntu Linux® версии 8. Если вы используете другую операционную систему, то есть возможность перенести большинство примеров и на нее; обращайтесь к странице руководства (man) `rsync` на вашей машине чтобы узнать, поддерживаются ли у вас используемые здесь операции, и при необходимости попытаться найти аналог.

Знакомимся с rsync

Так же как `cp`, `rsync` копирует файлы из одного места в другое. В отличие от `cp`, `rsync` может осуществлять как локальное, так и удаленное копирование. Например, команда, приведенная в листинге 1, копирует директорию /tmp/photos со всем ее содержимым в домашнюю директорию.

Листинг 1. Копируем директорию и ее содержимое

```
$  rsync -n -av /tmp/photos ~
building file list ... done
photos/
photos/Photo 2.jpg
photos/Photo 3.jpg
photos/Photo 6.jpg
photos/Photo 9.jpg

sent 218 bytes  received 56 bytes  548.00 bytes/sec
total size is 375409  speedup is 1370.11
```

Параметр `-v` включает вывод подробных сообщений. Параметр `-a` (здесь a обозначает архив), является краткой формой записи параметров `-rlptgoD`, обозначающих, что нужно выполнять рекурсивное (recurse) копирование, копируя символические ссылки как символические ссылки (links), сохраняя у всех файлов права доступа (permissions), время создания (times), группу(group) и владельца файла (owner), а также сохраняя файлы устройств и специальные файлы (devices). Обычно ключ `-a` создает зеркальную копию файлов за исключением случаев, когда система, на которую выполняется копирование, не поддерживает какие-либо атрибуты копируемых файлов. Например, при копировании директории с UNIX на Windows® не всегда удается идеально отобразить атрибуты. Ниже приводятся несколько предложений для работы в нестандартных ситуациях.

`rsync` имеет множество параметров. Если вы подозреваете, что параметры команды, описание источника или места назначения копирования неверны, то можно с помощью `-n` выполнить тестовый запуск. В ходе тестового запуска rsync покажет вам, что будет

сделано с каждым файлом, не перемещая в действительности ни одного байта. После этого, убедившись в правильности всех параметров, `-n` можно убрать, и изменения будут выполнены.

В листинге 2 приведен пример, в котором параметр `-n` может оказаться очень полезным. Команды, показанные в листинге 1 и в этом примере, приводят к различным результатам.

Листинг 2.Копируем содержимое указанной директории

```
$ rsync -av /tmp/photos/ ~
./
Photo 2.jpg
Photo 3.jpg
Photo 6.jpg
Photo 9.jpg

sent 210 bytes  received 56 bytes  532.00 bytes/sec
total size is 375409  speedup is 1411.31
```

В чем разница между командами? Разница в косой черте, завершающей аргумент источника копирования. Если аргумент завершается косой чертой, копируется *содержимое* указанной директории, но не сама директория. Наличие косой черты в конце адреса назначения не играет никакой роли.

В листинге 3 приведен пример копирования той же самой директории на удаленную систему.

Листинг 3. Копируем директорию на удаленную систему

```
$ rsync -av /tmp/photos example.com:album
created directory album
Photo 2.jpg
Photo 3.jpg
Photo 6.jpg
Photo 9.jpg

sent 210 bytes  received 56 bytes  21.28 bytes/sec
total size is 375409  speedup is 1411.31
```

Предполагая, что у вас на удаленной машине имеется учетная запись с тем же именем, `rsync` предлагает вам ввести пароль и затем, при условии успешной авторизации и наличии прав записи, создает директорию *album* и копирует в нее все фотографии. По умолчанию в `rsync` в качестве транспортного механизма используется Secure Shell (SSH), поэтому при работе с ним вы можете использовать уже имеющиеся псевдонимы (алиасы) машин и открытые ключи.

Режимы rsync

В листинге 2 и листинге 3 демонстрируются два из четырех режимов работы `rsync`. В листинге 2 был показан *режим оболочки*, также известный как локальный режим. В листинге 3 показан *режим удаленной оболочки*, названный так потому, что в нем соединение и передача данных осуществляются с помощью SSH. `rsync` имеет еще два режима. Один из них – это *режим списка*, который работает как команда ls, т.е. выводит содержимое источника копирования, как показано в листинге 4.

Листинг 4. Выводим список с содержимым источника копирования

```
$
drwxr-xr-x         238 2009/08/22 18:49:50 photos
-rw-r--r--        6148 2008/07/03 01:36:18 photos/.DS_Store
-rw-r--r--       71202 2008/06/18 04:51:36 photos/Photo 2.jpg
-rw-r--r--       69632 2008/06/18 04:51:45 photos/Photo 3.jpg
-rw-r--r--       61046 2008/07/14 00:31:17 photos/Photo 6.jpg
-rw-r--r--      167381 2008/07/14 00:31:56 photos/Photo 9.jpg
```

Четвертый режим – это *серверный режим*. Здесь демон `rsync` постоянно работает на машине, принимая запросы на прием или посылку файлов. Серверный режим является идеальным для создания центрального сервера резервного копирования или репозитория проекта.

Чтобы различать режим удаленной оболочки и серверный режим, в последнем в адресах источника и назначения используются два двоеточия (::). Следующая команда копирует файлы с удаленного сервера (здесь whatever.example.com) на локальную машину:

```
$ rsync -av whatever.example.com::src /tmp
```

Что такое `src`? Это модуль `rsync`, который определяется и конфигурируется на машине, где работает демон. Модуль имеет имя, путь к файлам и несколько других параметров, например `read only`, который защищает содержимое от модификации.

Чтобы запустить демон `rsync`, наберите:

```
$ sudo rsync --daemon
```

Вообще говоря, необязательно запускать демон `rsync` от имени супепользователя root, но имейте в виду, что такая практика защищает другие файлы вашей машины. При запуске от имени пользователя root используется команда `chroot`, благодаря которой `rsync` ограничивает себя иерархией директорий модуля. После команды `chroot` все остальные файлы и директории становятся для него невидимыми. Если вы будете запускать демон `rsync` от своего имени, то выберите неиспользуемый сокет и убедитесь, что модули `rsync` имеют достаточные права для чтения и записи пересылаемых файлов. В листинге 5 показана минимальная конфигурация для обмена файлами из своей домашней директории, которая не использует sudo. Конфигурация хранится в файле rsyncd.conf.

Листинг 5. Простая конфигурация для обмена файлами

```
motd file = /home/strike/rsyncd/rsync.motd_file
pid file = /home/strike/rsyncd/rsyncd.pid
port = 7777
use chroot = no

[demo]
path = /home/strike
comment = Martin home directory
list = no

[dropbox]
path = /home/strike/public/dropbox
comment = A place to leave things for Martin
read only = no

[pickup]
path = /home/strike/public/pickup
comment = Get your files here!
```

Этот файл имеет два сегмента. В первом сегменте – в данном случае это первые четыре строки – конфигурируется работа демона `rsync` (имеются также и другие параметры). В первой строке указывается путь к файлу с сообщением приветствия, помогающим идентифицировать сервер. Во второй строке указан другой файл, в который записывается ID процесса сервера. Это может быть удобно, если вам нужно вручную послать сигнал `kill` демону `rsync`:

```
kill -INT `cat /home/strike/rsyncd/rsyncd.pid`
```

Эти два файла находятся в домашней директории, так как в этом примере сервер запускается без привилегий суперпользователя. Также в первом сегменте указывается порт, на котором будет работать демон. Порты с номерами больше 1000 можно использовать для любых приложений. В четвертой строке выключается `chroot`.

Второй сегмент разделен на маленькие секции, по одной секции на модуль. Каждая секция, в свою очередь, имеет заголовок и список пар ключ–значение, задающих параметры каждого модуля. По умолчанию все модули доступны только для чтения; для разрешения записи выставьте read `only = no`. Также по умолчанию все модули указываются в каталоге модулей; выставьте `list = no`, если вы хотите скрыть этот модуль.

Для запуска демона выполните:

```
$ rsync --daemon --config=rsyncd.conf
```

Теперь, соединимся с демоном с другой машины, не указывая при этом имя модуля. Мы увидим следующее:

```
rsync --port=7777 mymachine.example.com::
Hello! Welcome to Martin's rsync server.

dropbox         A place to leave things for Martin
pickup          Get your files here!
```

Если после двойного двоеточия (`::`) не указывать имя модуля, то демон покажет список всех доступных модулей. Если указать имя модуля, но не указать имя конкретного файла или директории внутри него, демон вернет каталог содержимого модуля, как показано в листинге 6.

Листинг 6. Выводим содержимое модуля

```
rsync --port=7777 mymachine.example.com::pickup
Hello! Welcome to Martin's rsync server.

drwxr-xr-x        4096 2009/08/23 08:56:19 .
-rw-r--r--           0 2009/08/23 08:56:19 article21.html
-rw-r--r--           0 2009/08/23 08:56:19 design.txt
-rw-r--r--           0 2009/08/23 08:56:19 figure1.png
```

А если указать имя модуля и имя файла или каталога, то демон скопирует его на локальную машину, как показано в листинге 7.

Листинг 7. Копируем файлы на локальную машину

```
rsync --port=7777 mymachine.example.com::pickup/
Hello! Welcome to Martin's rsync server.

drwxr-xr-x        4096 2009/08/23 08:56:19 .
-rw-r--r--           0 2009/08/23 08:56:19 article21.html
-rw-r--r--           0 2009/08/23 08:56:19 design.txt
-rw-r--r--           0 2009/08/23 08:56:19 figure1.png
```

Поменяв местами адреса источника и назначения, можно записать в модуль файл(ы) с локальной машины, как показано в листинге 8.

Листинг 8. Меняем местами директории источника и назначения

```
$ rsync -v --port=7777 application.js mymachine.example.com::dropbox
Hello! Welcome to Martin's rsync server.

application.js

sent 245 bytes  received 38 bytes  113.20 bytes/sec
total size is 164  speedup is 0.58
```

Это был беглый, но довольно полный обзор возможностей `rsync`. Теперь давайте посмотрим, как можно применять этот пакет для повседневных задач. `rsync` особенно полезен для резервного копирования. А поскольку он умеет синхронизировать локальные и удаленные файлы или даже файловые системы, то он является идеальным инструментом для управления большими кластерами машин, которые должны быть (по крайней мере, частично) идентичными.

Организуем резервное копирование своих данных с помощью rsync

Регулярное сохранение резервных копий является необычайно важной, но, как правило, игнорируемой рутинной работой. Ни длительность процедуры резервного копирования, ни потребность в наличии большого внешнего хранилища файлов, ни что-либо другое не могут

являться оправданием; копирование данных для обеспечения их сохранности должно быть ежедневной процедурой.

Чтобы сделать эту задачу безболезненной, используйте для резервного копирования `rsync` и удаленный сервер, возможно, предоставляемый вашим провайдером. Каждая из ваших UNIX-машин может использовать этот механизм, который является идеальным решением для безопасного хранения ваших данных.

Установите на удаленной машине ключи SSH, демон `rsync` и создайте модуль для резервного копирования, разрешающий запись. После этого запустите rsync и, как показано в сценарии из листинга 9, создавайте резервные копии, которые едва ли будут занимать много места.

Листинг 9. Создаем ежедневные резервные копии файлов

```
#!/bin/sh
# This script based on work by Michael Jakl (jakl.michael AT gmail DOTCOM) and used
# with express permission.
HOST=mymachine.example.com
SOURCE=$HOME
PATHTOBACKUP=home-backup

date=`date "+%Y-%m-%dT%H:%M:%S"`

rsync -az --link-dest=$PATHTOBACKUP/current $SOURCE $HOST:PATHTOBACKUP/back-$date

ssh $HOST "rm $PATHTOBACKUP/current && ln -s back-$date $PATHTOBACKUP/current"
```

Замените `HOST` именем вашего сервера резервного копирования, а `SOURCE` – директорией, которую вы хотите сохранять. Замените `PATHTOBACKUP` на имя модуля. (Также три последние строки сценария можно заключить в цикл и, изменяя переменную `SOURCE`, делать резервные копии множества директорий). Данный сценарий работает следующим образом.

- Сначала в переменную `date` помещается строка вида `2009-08-23T12:32:18`, содержащая текущую дату и время; эта строка будет уникально идентифицировать каждую резервную копию.
- Главную работу здесь выполняет команда rsync. Параметры `-az` сохраняют всю информацию о файлах и выполняют сжатие данных перед их передачей, а параметр `--link-dest=$PATHTOBACKUP/current` указывает, что если какой-либо файл не менялся, нужно не копировать его в новый экземпляр резервной копии, а создать жесткую ссылку, указывающую на этот файл в существующем архиве. Другими словами, новая резервная копия содержит только файлы, претерпевшие *изменения*, остальные файлы являются ссылками.
  Рассмотрим сценарий более подробно (и подставим вместо всех переменных их значения). Текущим архивом является `mymachine.example.com::home-backup/current`. Новый архив для каталога `/home/strike` будет находиться в каталоге `mymachine.example.com::home-backup/back-2009-08-23T12:32:18`. Если файл в `/home/strike` не был изменен, то файл в новом архиве будет представлен жесткой ссылкой на соответствующий файл в текущем архиве. В противном случае новый файл копируется в новый архив.

Если вы каждый день изменяете лишь небольшое количество файлов и директорий, то дополнительное место, необходимое для очередного экземпляра резервной копии, будет ничтожно мало. Более того, так как все резервные копии (за исключением самой первой) довольно малы, можно поддерживать в своем распоряжении длинную историю ваших файлов.
- В последнем шаге мы изменяем организацию резервных копий на удаленной машине, чтобы сделать вновь созданный архив текущим архивом и таким образом минимизировать различия, которые нужно будет записать во время следующего выполнения сценария. В последней команде удаляется текущий архив, (который является просто жесткой ссылкой) и создается символическая ссылка с тем же именем, указывающая на новый архив.

Помните, что жесткая ссылка на жесткую ссылку, указывающую на какой-либо файл, указывает на тот же самый файл. Жесткие ссылки очень удобны и позволяют легко поддерживать полную резервную копию, используя лишь инкрементную схему.

Другие приемы и советы

Начав работать с удаленным `rsync` в повседневных задачах, вам, вероятно, понадобится, чтобы демон был всегда в рабочем состоянии. Для Linux- и UNIX-машин имеется загрузочный сценарий `rsync`, который обычно находится по адресу `/etc/init.d/rsync`. Воспользовавшись этим сценарием и утилитой вашей операционной системы, управляющей включением и выключением компонентов, можно организовать запуск `rsync` при загрузке системы. Если же вы запускаете демон `rsync` без привилегий суперпользователя или у вас нет доступа к загрузочным сценариям, то вы можете запускать `rsync` с помощью `cron`:

```
@reboot /usr/bin/rsync --daemon --port=7777 --config=/home/strike/rsyncd/rsyncd.conf
```

Эта команда запускает демон каждый раз при перезагрузке машины. Поместите эту строку в файл crontab и сохраните его.

Вы уже видели, как можно заранее обнаружить проблему, используя предварительный просмотр с помощью `-n`. Также можно отслеживать состояние задач `rsync` с помощью двух параметров: `--progress` и `--stats`. Первый из этих параметров отображает шкалу хода выполнения задания. Второй показывает статистику сжатия и передачи данных. С помощью `--compress` можно ускорить передачу данных между машинами. Вместо пересылки данных в изначальном виде отправитель выполняет сжатие перед отправкой, а получатель их распаковывает, и в результате меньшее количество байтов передается за меньшее время.

По умолчанию `rsync` копирует все файлы из источника данных в место назначения. Это называется дублированием. Если вы хотите организовать зеркалирование данных, т.е. чтобы локальные и удаленные данные в точности совпадали, следует использовать параметр `--delete`. Например, если в источнике имеются файлы A, B и C, то по умолчанию `rsync` создаст на удаленной машине копии всех трех файлов. Однако если удалить из источника, например, файл B и выполнить дублирование еще раз, то на удаленной машине файл B останется, т.е. удаленная копия перестанет быть точной копией локальных данных.

Команда `--delete` обеспечивает зеркалирование данных, убирая из удаленной копии файлы, которые уже не существуют в исходных данных.

Зачастую имеются файлы, которые вы не хотели бы помещать в архив или резервную копию. Это могут быть вспомогательные файлы, создаваемые редакторами (их имена обычно заканчиваются тильдой [`~`]) и другими утилитами, а также множество не имеющих для вас ценности файлов в вашей домашней директории, таких как MP3-файлы, которые при необходимости можно будет восстановить. В таком случае можно указать `rsync` шаблоны, по которым он будет исключать файлы из обработки. Можно указать в командной строке шаблон или же текстовый файл, содержащий список шаблонов. Также шаблоны можно использовать совместно с командой `--delete-excluded`, чтобы удалить подобные файлы из удаленной копии.

Чтобы исключить файлы, соответствующие определенному шаблону, используйте команду `--exclude`. Помните, что если какие-либо символы в шаблоне имеют для оболочки особое значение, например `*`, то шаблон следует заключить в одиночные кавычки:

```
$ rsync -a --exclude='*~' /home/strike/data example.com::data
```

Допустим, что файл /home/strike/excludes содержит следующий список шаблонов:

```
*~
*.old
*.mp3
tmp
```

Тогда скопировать все файлы за исключением тех, которые соответствуют какому-либо из этих шаблонов, можно с помощью следующей команды:

```
$ rsync -a --exclude-from=/home/strike/excludes /home/strike/data example.com::data
```

Синхронизируй это

Теперь, когда вы знакомы с `rsync`, у вас не осталось никаких причин не выполнять регулярное резервное копирование. Что случилось? Ваша собака разгрызла жесткий диск? (Бывает и такое!) Примите меры заранее, и тогда ваши данные останутся в полном порядке. Ведь теперь все ваши ценные файлы хранятся *в*FIDOnet.

Ресурсы

- Speaking UNIX: Advanced applications of rsync - оригинал статьи.
- Посетите домашнюю страницу rsync, чтобы узнать больше о проекте и почитать новости о его разработке
- Руководство по rsync: здесь можно найти множество рецептов работы с rsync
- Прочтите man-страницу rsync
- Оболочки UNIX: узнайте больше об оболочках UNIX из этой статьи Википедии.
- Посетите электронный магазин технической литературы Safari, чтобы найти интересующую вас информацию.

Об авторе

Мартин Стрейчер (Martin Streicher)

Мартин Стрейчер (Martin Streicher) - независимый web-разработчик и бывший главный редактор *Linux Magazine*. Он имеет степень магистра компьютерных наук Университета Пардью (Purdue University) и занимается программированием в UNIX-подобных операционных системах с 1986 года. Он коллекционирует предметы искусства и игрушки.