Technologia i rozwiązania

Skrypty powłoki systemu Linux
Receptury

Najlepsze przepisy na smakowite skrypty!

Helion

Sarath Lakshman

[PACKT PUBLISHING]

Tytuł oryginału: Linux Shell Scripting Cookbook

Tłumaczenie: Piotr Pilch

ISBN: 978-83-246-3886-4

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: http://helion.pl (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/sposyl
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

6

ROZDZIAŁ

Plan tworzenia kopii zapasowych

Ten rozdział zawiera następujące podrozdziały:

- Archiwizowanie za pomocą programu tar
- Archiwizowanie za pomocą programu cpio
- Kompresowanie za pomocą programu gunzip (gzip)
- Kompresowanie za pomocą programu bunzip (bzip)
- Kompresowanie za pomocą programu lzma
- Archiwizowanie i kompresowanie za pomocą programu zip
- squashfs — system plików o wysokim stopniu kompresji
- Narzędzia kryptograficzne i wartości mieszające
- Tworzenie migawek kopii zapasowych za pomocą programu rsync
- Tworzenie kopii zapasowych za pomocą narzędzia Git na podstawie kontroli wersji
- Klonowanie dysku twardego i innych dysków za pomocą programu dd

Wprowadzenie

Tworzenie migawek i kopii zapasowych danych to regularnie wykonywane zadania. W przypadku serwera lub dużych systemów przechowywania danych ważne jest systematyczne sporządzanie kopii zapasowych. Możliwe jest zautomatyzowanie tego zadania za pośrednictwem skryptów powłoki. Archiwizowanie i kompresowanie wydają się mieć zastosowanie w codziennej pracy administratora systemu lub zwykłego użytkownika. Istnieją różne formaty kompresji, które mogą być wykorzystane na różne sposoby w celu uzyskania jak najlepszych rezultatów.

Szyfrowanie to kolejne zadanie, które jest często realizowane na potrzeby ochrony danych. Aby zmniejszyć wielkość zaszyfrowanych danych, zwykle pliki są archiwizowane i kompresowane przed zaszyfrowaniem. Dostępnych jest wiele standardowych algorytmów szyfrowania, które mogą być obsługiwane za pomocą programów narzędziowych powłoki. W tym rozdziale zaprezentowano różne receptury objaśniające tworzenie archiwów plików lub katalogów, a także utrzymywanie ich, formaty kompresji oraz techniki szyfrowania za pomocą powłoki. Poznaj kolejne receptury.

Archiwizowanie za pomocą programu tar

Polecenie `tar` może posłużyć do archiwizowania plików. Pierwotnie stworzono je do przechowywania danych w archiwach taśmowych (`tar` — ang. *tape archives*). Polecenie to umożliwia przechowywanie wielu plików i katalogów jako jednego pliku. Może ono zachować wszystkie atrybuty plików, takie jak właściciel, uprawnienia itp. Plik utworzony przez polecenie `tar` często jest nazywany archiwum narzędzia `tar` (ang. *tarball*).

Wprowadzenie

Domyślnie polecenie `tar` jest dołączone do wszystkich uniksowych systemów operacyjnych. Polecenie to ma prostą składnię i oferuje przenośny format plików. Dowiedz się, jak z niego skorzystać.

Polecenie `tar` zapewnia listę argumentów: `A`, `c`, `d`, `r`, `t`, `u`, `x`, `f` i `v`. Każda z tych liter może być użyta niezależnie do zrealizowania różnych odpowiednich celów.

Jak to zrobić

Aby zarchiwizować pliki za pomocą polecenia `tar`, użyj następującej składni:

```
$ tar -cf output.tar [DANE_ŹRÓDŁOWE]
```

Oto przykład:

```
$ tar -cf output.tar plik1 plik2 plik3 katalog1 ..
```

W tym poleceniu opcja `-c` powoduje utworzenie pliku, a opcja `-f` umożliwia określenie nazwy pliku.

W miejsce łańcucha *`DANE_ŹRÓDŁOWE`* możesz określić katalogi i pliki. W celu określenia danych źródłowych możesz użyć listy nazw plików lub symboli wieloznacznych (np. `*.txt`).

Polecenie dokona archiwizacji plików źródłowych w pliku o nazwie *output.tar*.

Nazwa pliku musi pojawić się bezpośrednio po opcji `-f`, która powinna być ostatnią opcją w grupie argumentów (np. `-cvvf nazwa_pliku.tar` i `-tvvf nazwa_pliku.tar`).

Z powodu limitu polecenia `tar` nie jest możliwe przekazanie jako argumentów wiersza poleceń setek plików lub katalogów. Z tego powodu, jeśli ma zostać zarchiwizowanych wiele plików, bezpieczniejsze jest zastosowanie opcji dołączania.

To nie wszystko

Zaznajom się z dodatkowymi opcjami, które są dostępne w przypadku polecenia `tar`.

Dołączanie plików do archiwum

Czasem może być konieczne dodanie plików do już istniejącego archiwum (przykładem zastosowania jest sytuacja, w której tysiące plików mają zostać zarchiwizowane, a nie mogą one być określone w jednym wierszu jako argumenty wiersza poleceń).

Oto opcja dołączania: `-r`.

Aby dołączyć plik do już istniejącego archiwum, użyj polecenia:

```
$ tar -rvf oryginalny.tar nowy_plik
```

W następujący sposób wyświetl listę plików znajdujących się w archiwum:

```
$ tar -tf archiwum.tar
yy/lib64/
yy/lib64/libfakeroot/
yy/sbin/
```

Aby wyświetlić więcej szczegółów podczas archiwizowania lub generowania listy, użyj flag `-v` lub `-vv`. Flagi te są nazywane flagami trybu szczegółowego. Umożliwiają one zaprezentowanie w oknie terminalu więcej szczegółów. Na przykład przy użyciu tych flag możesz wyświetlić takie dodatkowe informacje, jak: uprawnienia plików, grupa właściciela, data modyfikacji itp.

Oto przykład:

```
$ tar -tvvf archiwum.tar
drwxr-xr-x slynux/slynux     0 2010-08-06 09:31 yy/
drwxr-xr-x slynux/slynux     0 2010-08-06 09:39 yy/usr/
drwxr-xr-x slynux/slynux     0 2010-08-06 09:31 yy/usr/lib64/
```

Wyodrębnianie plików i katalogów z archiwum

Następujące polecenie wyodrębnia zawartość archiwum w bieżącym katalogu:

```
$ tar -xf archiwum.tar
```

Opcja `-x` powoduje operację wyodrębniania.

W przypadku zastosowania tej opcji polecenie `tar` wyodrębni zawartość archiwum w bieżącym katalogu. Używając opcji `-C`, możesz również określić katalog, w którym zostaną umieszczone wyodrębnione pliki:

```
$ tar -xf archiwum.tar -C /ścieżka/katalogu_wyodrębniania
```

Polecenie to wyodrębnia zawartość archiwum w określonym katalogu. Operacja dotyczy zawartości całego archiwum.

Możliwe jest też wyodrębnienie tylko kilku plików, przez określenie ich jako argumentów polecenia:

```
$ tar -xvf plik.tar plik1 plik4
```

To polecenie wyodrębnia tylko pliki `plik1` i `plik4`. Ignorowane są pozostałe pliki w archiwum.

Użycie standardowego wejścia i wyjścia w przypadku polecenia tar

Podczas archiwizowania możesz określić standardowe wyjście `stdout` jako plik wyjściowy, aby inne polecenie wstawione za znakiem potoku mogło wczytać ten plik jako standardowe wejście `stdin`, a następnie zrealizować dany proces lub wyodrębnić dane z archiwum.

Jest to pomocne w przypadku przesyłania danych za pośrednictwem aktywnego połączenia sieciowego SSH (*Secure Shell*). Oto przykład:

```
$ mkdir ~/miejsce_docelowe
$ tar -cf - plik1 plik2 plik3 | tar -xvf -  -C ~/miejsce_docelowe
```

W tym przykładzie pliki *`plik1`*, *`plik2`* i *`plik3`* są łączone do postaci archiwum programu `tar`, a następnie wyodrębniane w katalogu *`~/miejsce_docelowe`*. W przypadku powyższego polecenia:

- opcja `-f` określa standardowe wyjście `stdout` jako plik na potrzeby archiwizowania (gdy użyto opcji `-c`);
- opcja `-f` określa standardowe wejście `stdin` jako plik na potrzeby wyodrębniania (gdy użyto opcji `-x`).

Łączenie dwóch archiwów

Za pomocą opcji `-A` z łatwością możesz scalić wiele plików programu `tar`.

Załóżmy, że istnieją dwa pliki archiwum programu `tar`: *plik1.tar* i *plik2.tar*. W następujący sposób możesz scalić zawartość pliku *plik2.tar* z zawartością pliku *plik1.tar*:

```
$ tar -Af plik1.tar plik2.tar
```

Sprawdź wynik operacji przez wyświetlenie zawartości pliku *plik1.tar*:

```
$ tar -tvf plik1.tar
```

Aktualizowanie plików w archiwum uwzględniające sprawdzenie znacznika czasu

Opcja dołączania umożliwia dodanie dowolnego pliku do archiwum. Jeśli w archiwum znajduje się już plik, który ma zostać dołączony, efektem operacji będzie pojawienie się w archiwum duplikatu pliku. Przy użyciu opcji aktualizowania `-u` możesz określić, że zostaną dołączone tylko te pliki, które są nowsze od plików o tej samej nazwie obecnych w archiwum.

```
$ tar -tf archiwum.tar
plika
plikb
plikc
```

Powyższe polecenie wyświetla listę plików archiwum.

Aby dołączyć plik *plika* tylko wtedy, gdy ma późniejszą datę modyfikacji niż plik o tej samej nazwie znajdujący się w archiwum *archiwum.tar*, użyj polecenia:

```
$ tar -uvvf archiwum.tar plika
```

Jeśli wersja pliku *plika* poza archiwum i plik *plika* wewnątrz archiwum *archiwum.tar* mają ten sam znacznik czasu, nie będzie miało miejsce żadne zdarzenie.

Użyj polecenia `touch` do zmodyfikowania znacznika czasu pliku, a następnie ponownie spróbuj wykonać polecenie `tar`:

```
$ tar -uvvf archiwum.tar plika
-rw-r--r-- slynux/slynux     0 2010-08-14 17:53 plika
```

Plik jest dołączany, ponieważ jego znacznik czasu jest aktualniejszy od znacznika pliku wewnątrz archiwum.

Porównywanie plików w archiwum i systemie plików

Czasem przydatne jest stwierdzenie, czy plik w archiwum oraz plik o identycznej nazwie w systemie plików są takie same, czy zawierają jakiekolwiek różnice. Flaga `-d` może posłużyć do wyświetlenia różnic:

```
$ tar -df archiwum.tar nazwa_pliku1 nazwa_pliku2 ...
```

Oto przykład:

```
$ tar -df archiwum.tar plika plikb
plika: Czas modyfikacji się różni
plika: Rozmiar się różni
```

Usuwanie plików z archiwum

Używając opcji `-delete`, możesz usunąć pliki z danego archiwum. Oto przykład:

```
$ tar -f archiwum.tar --delete plik1 plik2 ..
```

Oto kolejny przykład:

```
$ tar -tf archiwum.tar
plika
plikb
plikc
```

Możesz też zastosować następującą składnię:

```
$ tar --delete --file archiwum.tar [LISTA PLIKÓW]
```

Oto przykład:

```
$ tar --delete --file archiwum.tar plika
$ tar -tf archiwum.tar
plikb
plikc
```

Kompresowanie archiwum programu tar

Polecenie `tar` archiwizuje pliki, lecz nie kompresuje ich. Z tego powodu większość osób podczas pracy z archiwami programu `tar` zwykle dodaje określonego rodzaju kompresję. Dzięki temu znacznie zmniejsza się wielkość plików. Archiwa są często kompresowane przy użyciu jednego z następujących formatów:

- *plik.tar.gz*,
- *plik.tar.bz2*,
- *plik.tar.lzma*,
- *plik.tar.lzo*,

Różne flagi polecenia `tar` są używane do określenia różnych formatów kompresji:

- `-j` (dotyczy formatu *bunzip2*),
- `-z` (dotyczy formatu *gzip*),
- `--lzma` (dotyczy formatu *lzma*).

Formaty objaśniono w zamieszczonych dalej recepturach poświęconych kompresowaniu.

Możliwe jest zastosowanie formatów kompresji bez jawnego określania specjalnych opcji przedstawionych powyżej. Polecenie `tar` może przeprowadzić kompresję przez sprawdzenie danego rozszerzenia nazw plików wyjściowych lub wejściowych. Aby polecenie `tar` automatycznie obsługiwało kompresję przez określanie rozszerzeń, użyj opcji `-a` lub `--auto-compress`.

Wykluczanie zestawu plików z procesu archiwizowania

Istnieje możliwość wykluczenia zestawu plików z procesu archiwizowania przez określenie wzorców. Zastosuj opcję `--exclude [WZORZEC]` w celu wykluczenia plików dopasowanych przez wzorce w postaci symboli wieloznacznych.

Aby na przykład wykluczyć z archiwizowania wszystkie pliki *.txt*, użyj polecenia:

```
$ tar -cf arch.tar * --exclude "*.txt"
```

> Zauważ, że wzorzec powinien być ujęty w cudzysłów.

W następujący sposób za pomocą flagi `-X` możliwe jest też wykluczenie listy plików zawartej w pliku:

```
$ cat lista
plika
plikb

$ tar -cf arch.tar * -X lista
```

Polecenie to spowoduje wykluczenie plików *plika* i *plikb* z procesu archiwizowania.

Wykluczanie katalogów kontroli wersji

Zwykle archiwa programu `tar` są używane do dystrybucji kodu źródłowego. Większość kodu jest utrzymywana za pomocą systemów kontroli wersji, takich jak: Subversion, Git, Mercurial, CVS itp. Katalogi z kodem objęte kontrolą wersji będą zawierać specjalne katalogi używane do zarządzania wersjami (np. *.svn* lub *.git*). Jednakże te katalogi nie są wymagane przez kod, dlatego należy je wykluczyć z archiwum programu `tar` z kodem źródłowym.

Aby podczas archiwizowania wykluczyć pliki i katalogi powiązane z kontrolą wersji, użyj opcji `--exclude-vcs` polecenia `tar`. Oto przykład:

```
$ tar --exclude-vcs -czvvf kod_zrodlowy.tar.gz eye_of_gnome_svn
```

Wyświetlanie sumy bajtów

Czasem przydatna jest możliwość wyświetlenia sumy bajtów skopiowanych do archiwum. W następujący sposób za pomocą opcji `--totals` wyświetl całkowitą liczbę bajtów skopiowanych po zakończeniu archiwizacji:

```
$ tar -cf arc.tar * --exclude "*.txt" --totals
Liczba zapisanych bajtów: 20480 (20KiB, 12MiB/s)
```

Zobacz również

- W podrozdziale „Kompresowanie za pomocą programu gunzip (gzip)” objaśniono polecenie `gzip`.
- W podrozdziale „Kompresowanie za pomocą programu bunzip2 (bzip2)” objaśniono polecenie `bzip2`.
- W podrozdziale „Kompresowanie za pomocą programu lzma” objaśniono polecenie `lzma`.

Archiwizowanie za pomocą programu cpio

cpio to inny format archiwizowania, podobny do formatu programu `tar`. Służy on do przechowywania plików i katalogów w pliku z takimi atrybutami, jak uprawnienia, prawo właściciela itp. Format programu `cpio` nie jest jednak tak powszechny jak format programu `tar`. Jednak program `cpio` bywa używany w przypadku archiwów pakietów RPM, plików systemu plików *initramfs* dla jądra systemu Linux itp. W tej recepturze zaprezentowano najprostsze przykłady użycia programu `cpio`.

Jak to zrobić

Program `cpio` pobiera nazwy plików wejściowych za pośrednictwem standardowego wejścia `stdin` i zapisuje archiwum w standardowym wyjściu `stdout`. W celu otrzymania pliku z danymi wyjściowymi programu `cpio` konieczne jest przekierowanie standardowego wyjścia `stdout` do pliku w poniższy sposób.

Utwórz pliki testowe:

```
$ touch plik1 plik2 plik3
```

Pliki testowe możesz zarchiwizować przy użyciu polecenia:

```
$ echo plik1 plik2 plik3 | cpio -ov > archiwum.cpio
```

W tym poleceniu:

- opcja `-o` określa dane wyjściowe;
- opcja `-v` służy do wyświetlenia listy zarchiwizowanych plików.

> Używając programu `cpio`, możesz też archiwizować pliki z wykorzystaniem ścieżek bezwzględnych. */usr/katalog* to ścieżka bezwzględna, ponieważ stanowi pełną ścieżkę, począwszy od katalogu głównego (/).
>
> Ścieżka względna, zamiast znakiem /, rozpoczyna się na poziomie bieżącego katalogu. Na przykład ścieżka *test/plik* oznacza, że jest katalog *test*, w którym istnieje plik *plik*.
>
> Podczas wyodrębniania program `cpio` używa ścieżki bezwzględnej. Jednakże w przypadku polecenia `tar` program `cpio` usuwa znak / ze ścieżki bezwzględnej i zamienia ją na ścieżkę względną.

Aby wyświetlić listę plików w archiwum programu `cpio`, użyj następującego polecenia:

```
$ cpio -it < archiwum.cpio
```

Polecenie to wyświetli listę wszystkich plików w danym archiwum programu `cpio`. Wczytuje ono pliki ze standardowego wejścia `stdin`. W przypadku tego polecenia:

- opcja `-i` służy do określenia danych wejściowych;
- opcja `-t` umożliwia wygenerowanie listy.

W celu wyodrębnienia plików z archiwum programu `cpio` użyj polecenia:

```
$ cpio -id < archiwum.cpio
```

W poleceniu tym opcja `-d` powoduje wyodrębnianie.

Polecenie nadpisuje pliki bez żądania potwierdzenia operacji. Jeśli w archiwum znajdują się pliki ze ścieżką bezwzględną, polecenie zastąpi te pliki. W przeciwieństwie do polecenia `tar`, polecenie `cpio` nie wyodrębni plików w bieżącym katalogu.

Kompresowanie za pomocą programu gunzip (gzip)

gzip to format kompresji powszechnie używany w przypadku platform GNU/Linux. Dostępne są programy narzędziowe, takie jak: `gzip`, `gunzip` i `zcat`, które obsługują typy plików o formacie kompresji *gzip*. Program `gzip` może być zastosowany tylko dla pliku. Nie umożliwia on archiwizowania katalogów i wielu plików. A zatem używane jest archiwum programu `tar`, które jest kompresowane za pomocą programu `gzip`. Jeśli na wejściu określono wiele plików, program `gzip` wygeneruje kilka osobnych skompresowanych plików z rozszerzeniem *.gz*. Dowiedz się, jak korzystać z tego programu.

Jak to zrobić

Aby skompresować plik za pomocą programu `gzip`, użyj następującego polecenia:

```
$ gzip nazwa_pliku
$ ls
nazwa_pliku.gz
```

Polecenie to usunie plik i utworzy skompresowany plik o nazwie *nazwa_pliku.gz*.

W następujący sposób wyodrębnij plik skompresowany przy użyciu programu `gzip`:

```
$ gunzip nazwa_pliku.gz
```

Polecenie usunie plik *nazwa_pliku.gz* i utworzy jego wersję bez kompresji.

Aby wyświetlić listę właściwości skompresowanego pliku, użyj polecenia:

```
$ gzip -l test.txt.gz
compressed        uncompressed  ratio uncompressed_name
  35                     6        -33.3% test.txt
```

Polecenie `gzip` może wczytać plik ze standardowego wejścia `stdin`, a także zapisać skompresowany plik w standardowym wyjściu `stdout`.

W następujący sposób możesz odczytać standardowe wejście `stdin` oraz zwrócić dane jako standardowe wyjście `stdout`:

```
$ cat plik | gzip -c > plik.gz
```

Opcja `-c` służy do określenia danych wyjściowych w standardowym wyjściu `stdout`.

Możliwe jest określenie poziomu kompresji dla programu `gzip`. Użyj opcji `--fast` lub `--best`, aby zapewnić odpowiednio niski i wysoki współczynnik kompresji.

To nie wszystko

Polecenie `gzip` jest często stosowane z innymi poleceniami. Oferuje ono też zaawansowane opcje przeznaczone do określania współczynnika kompresji. Dowiedz się, jak korzystać z tych opcji.

Zastosowanie programu gzip dla archiwum programu tar

Zwykle w przypadku archiwów programu `tar` jest używany program `gzip`. Takie archiwa mogą zostać skompresowane z wykorzystaniem opcji `-z` przekazywanej poleceniu `tar` podczas archiwizowania i wyodrębniania.

Korzystając z następujących metod, możesz tworzyć archiwa programu `tar` skompresowane przez program `gzip`:

- **Metoda 1.**

  ```
  $ tar -czvvf archiwum.tar.gz [PLIKI]
  ```

  lub:

  ```
  $ tar -cavvf archiwum.tar.gz [PLIKI]
  ```

  Opcja `-a` określa, że format kompresji powinien być automatycznie wykryty na podstawie rozszerzenia.

- **Metoda 2.**

  Najpierw utwórz archiwum programu `tar`:

  ```
  $ tar -cvvf archiwum.tar [PLIKI]
  ```

  W następujący sposób skompresuj utworzone archiwum:

  ```
  $ gzip archiwum.tar
  ```

Jeśli w archiwum programu `tar` ma zostać umieszczonych wiele plików (nawet kilkaset), a ponadto archiwum wymaga skompresowania, zostanie użyta druga metoda z kilkoma modyfikacjami. Problem z podawaniem wielu plików jako argumentów polecenia `tar` polega na tym, że z poziomu wiersza poleceń może ono zaakceptować tylko ograniczoną liczbę plików. Aby roz-

wiązać ten problem, możesz utworzyć plik programu `tar`, dodając pliki kolejno za pomocą pętli z opcją dołączania (`-r`):

```
FILE_LIST="plik1  plik2  plik3  plik4  plik5"
for f in $FILE_LIST;
do
tar -rvf archiwum.tar $f
done
gzip archiwum.tar
```

Aby wyodrębnić archiwum programu `tar` skompresowane przez program `gzip`, użyj następujących opcji:

- opcja `-x` dotyczy wyodrębniania;
- opcja `-z` dotyczy specyfikacji formatu *gzip*.

Możesz też wykonać polecenie:

```
$ tar -xavvf archiwum.tar.gz -C katalog_wyodrębniania
```

W powyższym poleceniu opcja `-a` powoduje automatyczne wykrycie formatu kompresji.

Program zcat — odczytywanie plików formatu gzip bez wyodrębniania

`zcat` to polecenie, które może być użyte do umieszczenia pliku wyodrębnionego z pliku *.gz* w standardowym wyjściu `stdout` bez ręcznego wykonywania operacji wyodrębniania. Plik *.gz* pozostaje w niezmienionej postaci, lecz wyodrębniony plik zostanie umieszczony w standardowym wyjściu `stdout` w następujący sposób:

```
$ ls
test.gz

$ zcat test.gz
Plik testowy
# Plik testowy zawiera wiersz "Plik testowy".

$ ls
test.gz
```

Współczynnik kompresji

Możliwe jest określenie współczynnika kompresji z zakresu od 1 do 9, w przypadku którego:

- 1 to najgorsza, lecz najszybsza kompresja;
- 9 to najlepsza, lecz najwolniejsza kompresja.

W następujący sposób możesz też określić inne współczynniki kompresji:

```
$ gzip -9 test.img
```

Polecenie maksymalnie skompresuje plik.

Zobacz również

- W podrozdziale „Archiwizowanie za pomocą programu tar” objaśniono polecenie `tar`.

Kompresowanie za pomocą programu bunzip2 (bzip2)

bunzip2 to następny format kompresji, który bardzo przypomina format programu `gzip`. Program `bzip2` zwykle generuje mniejsze (bardziej skompresowane) pliki niż program `gzip`. Program `bzip2` wchodzi w skład wszystkich dystrybucji systemu Linux. Dowiedz się, jak z niego korzystać.

Jak to zrobić

Aby za pomocą programu `bzip2` dokonać kompresji, użyj polecenia:

```
$ bzip2 nazwa_pliku
$ ls
nazwa_pliku.bz2
```

Polecenie to usunie plik i utworzy skompresowany plik o nazwie *nazwa_pliku.bz2*.

W następujący sposób wyodrębnij plik *.bz2*:

```
$ bunzip2 nazwa_pliku.bz2
```

Polecenie usunie plik *nazwa_pliku.bz2* i utworzy wersję pliku *nazwa_pliku* bez kompresji.

Program `bzip2` umożliwia wczytanie pliku ze standardowego wejścia `stdin`, a także zapisanie skompresowanego pliku w standardowym wyjściu `stdout`.

Aby odczytać standardowe wejście `stdin` oraz pobrać dane ze standardowego wyjścia `stdout`, użyj polecenia:

```
$ cat plik | bzip2 -c > plik.tar.bz2
```

Opcja `-c` służy do skierowania danych wyjściowych do standardowego wyjścia `stdout`.

Zwykle program `bzip2` jest używany w przypadku archiwów programu `tar`. Takie archiwa mogą zostać skompresowane z wykorzystaniem opcji `-j` przekazywanej poleceniu `tar` podczas archiwizowania i wyodrębniania.

Korzystając z następujących metod, możesz tworzyć archiwa programu `tar` skompresowane przez program `bzip2`:

- **Metoda 1.**

  ```
  $ tar -cjvvf archiwum.tar.bz2 [PLIKI]
  ```

  lub:

  ```
  $ tar -cavvf archiwum.tar.bz2 [PLIKI]
  ```

  Opcja `-a` określa, że format kompresji zostanie automatycznie wykryty na podstawie rozszerzenia.

- **Metoda 2.**

  Najpierw utwórz archiwum programu `tar`:

  ```
  $ tar -cvvf archiwum.tar [PLIKI]
  ```

  Skompresuj utworzone archiwum:

  ```
  $ bzip2 archiwum.tar
  ```

Jeśli w archiwum mają zostać umieszczone setki plików, powyższe polecenia mogą być nieprzydatne. W celu poradzenia sobie z tym problemem użyj pętli do dołączenia kolejno plików do archiwum za pomocą opcji `-r`. Przejdź do podobnego punktu zamieszczonego w podrozdziale „Kompresowanie za pomocą programu gunzip (gzip)".

W następujący sposób wyodrębnij archiwum programu `tar` skompresowane przez program `bzip2`:

```
$ tar -xjvvf archiwum.tar.bz2 -C katalog_wyodrębniania
```

W tym poleceniu:

- opcja `-x` powoduje wyodrębnianie;
- opcja `-z` określa format *bzip2*;
- opcja `-C` służy do określenia katalogu, w którym zostaną wyodrębnione pliki.

Możesz też użyć następującego polecenia:

```
$ tar -xavvf archiwum.tar.bz2 -C katalog_wyodrębniania
```

Opcja `-a` automatycznie wykryje format kompresji.

To nie wszystko

Program `bunzip2` oferuje kilka dodatkowych opcji do realizowania różnych funkcji. Poznaj kilka z nich.

Zachowywanie plików wejściowych

Program `bzip2` (`bunzip2`) usuwa pliki wejściowe i tworzy skompresowane pliki wyjściowe. Aby zapobiec usuwaniu plików wejściowych, użyj opcji `-k`.

Oto przykład:

```
$ bunzip2 test.bz2 -k
$ ls
test test.bz2
```

Współczynnik kompresji

Możliwe jest określenie współczynnika kompresji z zakresu od 1 do 9, w przypadku którego 1 to najgorsza, lecz najszybsza kompresja, a 9 to najwyższa możliwa, lecz znacznie wolniejsza kompresja.

Oto przykład:

```
$ bzip2 -9 test.img
```

Powyższe polecenie zapewnia maksymalną kompresję.

Zobacz również

- W podrozdziale „Archiwizowanie za pomocą programu tar” objaśniono polecenie `tar`.

Kompresowanie za pomocą programu lzma

lzma to stosunkowo nowy format w porównaniu z formatami programów `gzip` lub `bzip2`. Program `lzma` oferuje lepsze współczynniki kompresji niż tamte programy. Ponieważ program `lzma` nie jest domyślnie instalowany w większości dystrybucji systemu Linux, może być konieczne wykonanie tej operacji za pomocą menedżera pakietów.

Jak to zrobić

Aby dokonać kompresji za pomocą programu `lzma`, użyj następującego polecenia:

```
$ lzma nazwa_pliku
$ ls
nazwa_pliku.lzma
```

Polecenie to spowoduje usunięcie pliku i utworzenie skompresowanego pliku o nazwie *nazwa_pliku.lzma*.

W celu wyodrębnienia pliku programu `lzma` zastosuj polecenie:

```
$ unlzma nazwa_pliku.lzma
```

Polecenie usunie plik *nazwa_pliku.lzma* i utworzy wersję pliku bez kompresji.

Polecenie `lzma` umożliwia wczytanie pliku ze standardowego wejścia `stdin`, a także zapisanie skompresowanego pliku w standardowym wyjściu `stdout`.

Aby odczytać standardowe wejście `stdin` oraz pobrać dane ze standardowego wyjścia `stdout`, użyj polecenia:

```
$ cat plik | lzma -c > plik.lzma
```

Opcja `-c` służy do skierowania danych wyjściowych do standardowego wyjścia `stdout`.

Zwykle program `lzma` jest używany w przypadku archiwów programu `tar`. Takie archiwa mogą zostać skompresowane za pomocą opcji `--lzma` przekazywanej poleceniu `tar` podczas archiwizowania i wyodrębniania.

Istnieją następujące dwie metody tworzenia archiwum programu `tar` skompresowanego przez program `lzma`:

- **Metoda 1.**

  ```
  $ tar -cvvf --lzma archiwum.tar.lzma [PLIKI]
  ```

  lub:

  ```
  $ tar -cavvf archiwum.tar.lzma [PLIKI]
  ```

  Opcja `-a` określa, że format kompresji zostanie automatycznie wykryty na podstawie rozszerzenia.

- **Metoda 2.**

  Najpierw utwórz archiwum programu `tar`:

  ```
  $ tar -cvvf archiwum.tar [PLIKI]
  ```

  Skompresuj utworzone archiwum:

  ```
  $ lzma archiwum.tar
  ```

Jeśli w archiwum mają zostać umieszczone setki plików, powyższe polecenia mogą być nieprzydatne. W celu poradzenia sobie z tym problemem użyj pętli do dołączenia kolejno plików do archiwum za pomocą opcji `-r`. Przejdź do podobnego punktu zamieszczonego w podrozdziale „Kompresowanie za pomocą programu gunzip (gzip)”.

To nie wszystko

Poznaj dodatkowe opcje związane z programem narzędziowym `lzma`.

Wyodrębnianie archiwum programu tar skompresowanego przez program lzma

Aby wyodrębnić archiwum programu `tar` skompresowane przez program `lzma` w określonym katalogu, użyj następującego polecenia:

```
$ tar -xvvf --lzma archiwum.tar.lzma -C katalog_wyodrębniania
```

W poleceniu opcja `-x` powoduje wyodrębnianie. Opcja `--lzma` określa, że do dekompresji pliku wynikowego zostanie zastosowany program `lzma`.

Możesz też użyć polecenia:

```
$ tar -xavvf archiwum.tar.lzma -C katalog_wyodrębniania
```

Opcja `-a` powoduje automatyczne wykrycie formatu kompresji na podstawie rozszerzenia.

Zachowywanie plików wejściowych

Programy `lzma` lub `unlzma` usuwają pliki wejściowe i tworzą skompresowane pliki wyjściowe. Można jednak zapobiec usunięciu plików wejściowych i zachować je przy użyciu opcji `-k`. Oto przykład:

```
$ lzma test.bz2 -k
$ ls
test.bz2.lzma
```

Współczynnik kompresji

Możliwe jest określenie współczynnika kompresji z zakresu od 1 do 9, w przypadku którego 1 to najgorsza, lecz najszybsza kompresja, a 9 to najwyższa możliwa, lecz znacznie wolniejsza kompresja.

W następujący sposób możesz też określić inny współczynnik:

```
$ lzma -9 test.img
```

Powyższe polecenie zapewnia maksymalną kompresję pliku.

Zobacz również

- W podrozdziale „Archiwizowanie za pomocą programu tar” objaśniono polecenie `tar`.

Archiwizowanie i kompresowanie za pomocą programu zip

ZIP to popularny format kompresji używany na wielu platformach. Choć nie jest tak powszechnie stosowany na platformach z systemem Linux, jak formaty programów `gzip` lub `bzip2`, pliki pochodzące z internetu często są zapisywane właśnie w formacie ZIP.

Jak to zrobić

W celu zarchiwizowania w formacie ZIP użyj następującej składni:

```
$ zip nazwa_archiwum.zip [PLIKI ŹRÓDŁOWE/KATALOGI]
```

Oto przykład:

```
$ zip plik.zip plik
```

W tym przypadku zostanie wygenerowany plik *plik.zip*.

W następujący sposób rekurencyjnie zarchiwizuj katalogi i pliki:

```
$ zip -r archiwum.zip katalog1 plik2
```

W tym poleceniu opcja `-r` służy do określenia rekurencji.

W przeciwieństwie do programów `lzma`, `gzip` lub `bzip2`, program `zip` nie usunie pliku źródłowego po zakończeniu archiwizowania. Choć pod tym względem program `zip` przypomina program `tar`, może on kompresować pliki, natomiast program `tar` tego nie umożliwia.

Aby wyodrębnić pliki i katalogi z pliku ZIP, użyj polecenia:

```
$ unzip plik.zip
```

Polecenie to spowoduje wyodrębnienie plików bez usuwania pliku *plik.zip* (w przeciwieństwie do programów `unlzma` lub `gunzip`).

W celu aktualizowania plików w archiwum przy użyciu nowszych plików systemu plików użyj flagi `-u`:

```
$ zip plik.zip -u nowy_plik
```

Usuń plik z archiwum ZIP, korzystając z opcji `-d`:

```
$ zip -d arc.zip plik.txt
```

Aby wyświetlić listę plików w archiwum, wykonaj polecenie:

```
$ unzip -l archiwum.zip
```

squashfs — system plików o wysokim stopniu kompresji

squashfs to system plików tylko do odczytu o wysokim stopniu kompresji, który umożliwia skompresowanie danych o pojemności od 2 do 3 GB w pliku o wielkości 700 MB. Czy kiedykolwiek zastanawiałeś się, jak działają dyski Live CD systemu Linux? Gdy rozruch jest przeprowadzany

z dysku Live CD, ładuje się kompletne środowisko systemu Linux. Tego rodzaju dyski korzystają ze skompresowanego systemu plików tylko do odczytu o nazwie `squashfs`. W tym przypadku główny system plików znajduje się w skompresowanym pliku systemu plików. System plików `squashfs` może być podłączony w trybie pętli zwrotnej, a ponadto możliwe jest uzyskanie dostępu do plików. A zatem, gdy jakiś proces wymaga użycia określonych plików, są one dekompresowane i ładowane do pamięci RAM, a następnie używane. Znajomość systemu plików `squashfs` może być przydatna podczas tworzenia niestandardowego dysku uruchomieniowego systemu operacyjnego lub w sytuacji, gdy konieczne jest utrzymanie wysokiej kompresji plików i uzyskiwanie do nich dostępu bez całkowitego wyodrębniania plików. Wyodrębnianie dużego skompresowanego pliku może zająć wiele czasu. Jeśli jednak plik podłączono w trybie pętli zwrotnej, dostęp do niego będzie bardzo szybki, ponieważ wymagana część skompresowanych plików jest dekompresowana tylko wtedy, kiedy pojawi się żądanie dotyczące plików. W przypadku zwykłej dekompresji operacji tej są poddawane wszystkie dane. Dowiedz się, jak używać systemu plików `squashfs`.

Wprowadzenie

Jeśli dysponujesz dyskiem CD systemu Ubuntu, po prostu znajdź plik *.squashfs* w katalogu *CDRom ROOT/casper/filesystem.squashfs*. Wewnętrznie system plików `squashfs` używa algorytmów kompresji, takich jak `gzip` i `lzma`. Ten system plików jest obsługiwany przez wszystkie najnowsze dystrybucje systemu Linux. Jednakże do utworzenia plików systemu plików `squashfs` niezbędne będzie zainstalowanie za pomocą menedżera pakietów dodatkowego pakietu **squashfs-tools**.

Jak to zrobić

Aby utworzyć plik systemu plików `squashfs` przez dodanie katalogów i plików źródłowych, użyj polecenia:

```
$ mksquashfs DANE_ŹRÓDŁOWE compressedfs.squashfs
```

Dane źródłowe mogą być określone za pomocą symboli wieloznacznych bądź jako plik lub ścieżki katalogów.

Oto przykład:

```
$ sudo mksquashfs /etc test.squashfs
Parallel mksquashfs: Using 2 processors
Creating 4.0 filesystem on test.squashfs, block size 131072.
[=========================================] 1867/1867 100%
# W oknie terminalu zostaną wyświetlone dodatkowe szczegóły. Tutaj pominięto je w celu
# zaoszczędzenia miejsca.
```

W celu podłączenia pliku systemu plików `squashfs` do punktu podłączenia w następujący sposób użyj podłączenia w trybie pętli zwrotnej:

```
# mkdir /mnt/squash
# mount -o loop compressedfs.squashfs /mnt/squashfs
```

Zawartość możesz skopiować, uzyskując dostęp do punktu podłączenia */mnt/squashfs*.

To nie wszystko

System plików `squashfs` może być utworzony przez określenie dodatkowych parametrów. Poznaj inne opcje.

Wykluczanie plików podczas tworzenia pliku systemu plików squashfs

W trakcie tworzenia pliku systemu plików `squashfs` możesz wykluczyć listę plików lub określić wzorzec plików przy użyciu symboli wieloznacznych.

Wyklucz listę plików określoną jako argumenty wiersza poleceń za pomocą opcji `-e`. Oto przykład:

```
$ sudo mksquashfs /etc test.squashfs -e /etc/passwd /etc/shadow
```

Opcja `-e` służy do wykluczania plików *passwd* i *shadow*.

Przy użyciu opcji `-ef` możliwe jest również określenie listy wykluczonych plików podanej w pliku:

```
$ cat excludelist
/etc/passwd
/etc/shadow

$ sudo mksquashfs /etc test.squashfs -ef excludelist
```

Aby w listach wykluczeń były obsługiwane symbole wieloznaczne, jako argumentu użyj opcji `-wildcard`.

Narzędzia kryptograficzne i wartości mieszające

Techniki szyfrowania są stosowane głównie do ochrony danych przed nieautoryzowanym dostępem. Istnieje wiele algorytmów. Użytkownicy korzystają z ogólnego zestawu standardowych algorytmów. W środowisku systemu Linux udostępniono kilka narzędzi służących do szyfrowania i rozszyfrowywania. Czasem do sprawdzenia integralności danych są używane wartości mieszające algorytmy szyfrowania. W tej recepturze zaprezentowano kilka powszechnie wykorzystywanych narzędzi kryptograficznych oraz ogólny zestaw algorytmów, które te narzędzia mogą obsługiwać.

Jak to zrobić

Dowiedz się, jak używać narzędzi takich jak: `crypt`, `gpg`, `base64`, `md5sum`, `sha1sum` i `openssl`.

- **crypt**

  Polecenie `crypt` to prosty kryptograficzny program narzędziowy, który pobiera plik ze standardowego wejścia `stdin` oraz frazę kodującą, a zwraca zaszyfrowane dane umieszczone w standardowym wyjściu `stdout`:

  ```
  $ crypt <plik_wejściowy> plik_wyjściowy
  Enter passphrase:
  ```

  Polecenie w sposób interaktywny zażąda frazy kodującej. Możliwe jest również przekazanie frazy za pośrednictwem argumentów wiersza poleceń:

  ```
  $ crypt FRAZA_KODUJĄCA < plik_wejściowy > zaszyfrowany_plik
  ```

  Aby rozszyfrować plik, użyj polecenia:

  ```
  $ crypt FRAZA_KODUJĄCA -d < zaszyfrowany_plik > plik_wyjściowy
  ```

- **gpg** (**ang.** ***GNU privacy guard***)

  `gpg` to powszechnie używany schemat szyfrowania służący do ochrony plików za pomocą technik opartych na podpisywaniu kluczem. Schemat ten umożliwia dostęp do danych tylko wiarygodnemu odbiorcy. Sygnatury schematu `gpg` są bardzo często spotykane. Omawianie szczegółów schematu wykracza poza zakres tej książki. Poniżej wyjaśniono, jak szyfrować i rozszyfrowywać plik.

  Aby zaszyfrować plik za pomocą schematu `gpg`, wykonaj polecenie:

  ```
  $ gpg -c nazwa_pliku
  ```

  Polecenie to wczytuje w trybie interaktywnym frazę kodującą i generuje plik *nazwa_pliku.gpg*.

  W celu rozszyfrowania pliku *.gpg* użyj polecenia:

  ```
  $ gpg -c nazwa_pliku.gpg
  ```

  Wczytuje ono frazę kodującą i rozszyfrowuje plik.

- **Base64**

  `Base64` to grupa podobnych schematów kodowania, które reprezentują dane binarne w formacie łańcucha ASCII przez przekształcenie go do postaci kodu `radix-64`. Polecenie `base64` może być użyte do kodowania i dekodowania łańcucha `Base64`.

  Aby zakodować plik binarny do formatu `Base64`, wykonaj polecenie:

  ```
  $ base64 nazwa_pliku > plik_wyjściowy
  ```

  lub:

  ```
  $ cat plik | base64 > plik_wyjściowy
  ```

  Polecenie to może wczytać zawartość standardowego wejścia `stdin`.

  W następujący sposób zdekoduj dane `Base64`:

```
$ base64 -d plik > plik_wyjściowy
```

lub:

```
$ cat plik_base64 | base64 -d > plik_wyjściowy
```

- **md5sum i sha1sum**

  `md5sum` i **`sha1sum`** to jednokierunkowe algorytmy mieszające, których działanie nie może być odwrócone w celu utworzenia oryginalnych danych. Algorytmów tych zwykle używa się do sprawdzania integralności danych lub generowania unikalnego klucza dla określonych danych. Unikalny klucz jest generowany dla każdego pliku przez analizę jego zawartości:

```
$ md5sum plik
8503063d5488c3080d4800ff50850dc9  plik

$ sha1sum plik
1ba02b66e2e557fede8f61b7df282cd0a27b816b  plik
```

  Tego typu wartości mieszające idealnie nadają się do przechowywania haseł. Hasła są składowane w postaci ich wartości mieszających. Gdy użytkownik zamierza dokonać uwierzytelnienia, hasło jest wczytywane i przekształcane w wartość mieszającą. Wartość ta jest następnie porównywana z już przechowywaną wartością. Jeśli wartości są takie same, hasło jest uwierzytelniane, a dostęp zapewniany. W przeciwnym razie ma miejsce odmowa dostępu. Przechowywanie oryginalnych łańcuchów haseł jest ryzykowne i stwarza zagrożenie dotyczące zabezpieczeń, które polega na ujawnieniu hasła.

- **wartość mieszająca przypominająca wartość w pliku *shadow* (generowana za pomocą ciągu zaburzającego)**

  Dowiedz się, jak przy użyciu ciągu zaburzającego (ang. *salt*) wygenerować dla haseł wartości mieszające podobne do tych zawartych w pliku *shadow*.

  W systemie Linux hasła użytkowników są przechowywane jako ich wartości mieszające w pliku */etc/shadow*. Typowy wiersz w tym pliku wygląda następująco:

```
test:$6$fG4eWdUi$ohTKOlEUzNk77.4S8MrYeO7NTRV4M3LrJnZP9p.qc1bR5c.
EcOruzPXfEu1uloBFUa18ENRH7F7Ozhodas3cR.:14790:0:99999:7:::
```

  W tym wierszu ciąg `$6$fG4eWdUi$ohTKOlEUzNk77.4S8MrYeO7NTRV4M3LrJnZP9p.qc1bR5c.EcOruzPXfEu1uloBFUa18ENRH7F7Ozhodas3cR` stanowi wartość mieszającą przesłaniania, która odpowiada hasłu.

  W określonych sytuacjach może być konieczne napisanie skryptów do realizowania kluczowych zadań administracyjnych, które mogą wymagać ręcznego edytowania haseł lub dodawania użytkowników za pomocą skryptu powłoki. W tym przypadku musisz wygenerować łańcuch hasła przesłanianego i umieścić w pliku *shadow* wiersz podobny do powyższego. Dowiedz się, jak wygenerować hasło przesłaniane przy użyciu programu `openssl`.

  Hasła przesłaniane są zwykle tworzone za pomocą ciągu zaburzającego, który jest dodatkowym łańcuchem służącym do zaciemniania i poprawy szyfrowania. Ciąg zaburzający składa się z losowych bitów stosowanych jako jedno z wejść funkcji

KDF (ang. *Key Derivation Function*), która dla hasła generuje wartość mieszającą z ciągiem zaburzającym.

Więcej informacji o ciągu zaburzającym zamieszczono na stronie serwisu Wikipedia pod adresem: *http://en.wikipedia.org/wiki/Salt_(cryptography)*.

```
$ openssl passwd -1 -salt ŁAŃCUCH_CIĄGU_ZABURZAJĄCEGO HASŁO
$1$ŁAŃCUCH_CIĄGU_ZABURZAJĄCEGO$323VkWkSLHuhbt1zkSsUG.
```

W miejsce łańcucha *ŁAŃCUCH_CIĄGU_ZABURZAJĄCEGO* wstaw losowy łańcuch, a zamiast łańcucha *HASŁO* użyj żądanego hasła.

Tworzenie migawek kopii zapasowych za pomocą programu rsync

Sporządzanie kopii zapasowych danych to coś, co musi być regularnie wykonywane przez większość administratorów systemów. Może być konieczne utworzenie kopii zapasowej danych serwera WWW lub danych położonych w lokalizacjach zdalnych. `rsync` to polecenie, które może posłużyć do synchronizacji plików i katalogów zlokalizowanych w dwóch miejscach. Dzięki określaniu różnic w plikach i stosowaniu kompresji odbywa się to przy minimalnym transferze danych. W porównaniu z poleceniem `cp`, zaletą polecenia `rsync` jest to, że korzysta ono z silnych algorytmów różnicowych. Ponadto obsługuje przesyłanie danych między sieciami. Podczas tworzenia kopii polecenie `rsync` porównuje pliki w lokalizacjach oryginalnej i docelowej. W efekcie zostaną skopiowane wyłącznie nowsze pliki. Polecenie to obsługuje też kompresję, szyfrowanie i znacznie więcej rzeczy. Dowiedz się, jak pracować z programem `rsync`.

Jak to zrobić

Aby skopiować katalog źródłowy do miejsca docelowego (w celu utworzenia kopii lustrzanej), użyj polecenia:

```
$ rsync -av ścieżka_źródłowa ścieżka_docelowa
```

W tym poleceniu:

- opcja `-a` powoduje archiwizowanie;
- opcja `-v` powoduje wyświetlenie w standardowym wyjściu `stdout` szczegółów lub danych o postępie operacji.

Powyższe polecenie w sposób rekurencyjny skopiuje wszystkie pliki ze ścieżki źródłowej do docelowej. Ścieżki możesz określić jako ścieżki zdalne lub ścieżki lokalnego hosta.

Przykładowy format ścieżki: */home/slynux/data*, *slynux@192.168.0.6:/home/backups/data*.

Ścieżka */home/slynux/data* reprezentuje ścieżkę bezwzględną w przypadku komputera, na którym jest wykonywane polecenie `rsync`. Ścieżka *slynux@192.168.0.6:/home/backups/data* identyfikuje katalog *home/backups/data* na komputerze o adresie IP `192.168.0.6`, na którym jest zalogowany użytkownik *slynux*.

Aby utworzyć kopię zapasową danych na zdalnym serwerze lub hoście, użyj polecenia:

```
$ rsync -av katalog_źródłowy nazwa_użytkownika@host:ŚCIEŻKA
```

W celu zachowania kopii lustrzanej w miejscu docelowym zaplanuj uruchamianie tego samego polecenia `rsync` w regularnych odstępach czasu. Polecenie to skopiuje w miejsce docelowe tylko zmodyfikowane pliki.

W następujący sposób przywróć dane ze zdalnego hosta do hosta lokalnego:

```
$ rsync -av nazwa_użytkownika@host:ŚCIEŻKA miejsce_docelowe
```

Polecenie `rsync` używa protokołu SSH do nawiązania połączenia ze zdalnym komputerem. Zapewnij adres zdalnego komputera w formacie *`użytkownik@host`*, gdzie łańcuch *`użytkownik`* reprezentuje nazwę użytkownika, a łańcuch `host` — adres IP lub nazwę domeny zdalnego komputera. Łańcuch *`ŚCIEŻKA`* identyfikuje ścieżkę bezwzględną miejsca, w które należy skopiować dane. Jak to zwykle bywa w przypadku protokołu SSH, polecenie `rsync` zażąda podania hasła użytkownika. Zadanie może zostać automatyzowane (eliminuje to sprawdzanie hasła użytkownika) za pomocą kluczy SSH.

Upewnij się, że na zdalnym komputerze zainstalowano i uruchomiono oprogramowanie OpenSSH.

Kompresowanie danych podczas przesyłania ich w sieci może znacznie zoptymalizować szybkość transferu. W celu określenia operacji kompresowania danych w trakcie przesyłania ich w sieci użyj opcji `-z` polecenia `rsync`. Oto przykład:

```
$ rsync -avz miejsce_źródłowe miejsce_docelowe
```

> Jeśli w formacie łańcucha *`ŚCIEŻKA`* na końcu ścieżki docelowej zostanie użyty znak `/`, polecenie `rsync` skopiuje w miejsce docelowe zawartość katalogu określonego przez ten znak w ścieżce źródłowej.
>
> Jeśli na końcu ścieżki źródłowej nie wstawiono znaku `/`, polecenie `rsync` skopiuje w miejsce docelowe tylko katalog określony przez ten znak.

Na przykład następujące polecenie kopiuje zawartość katalogu *test*:

```
$ rsync -av /home/test/ /home/backups
```

Następujące polecenie kopiuje katalog *test* w miejsce docelowe:

```
$ rsync -av /home/test /home/backups
```

> Jeśli w łańcuchu *ścieżka_docelowa* na końcu umieszczono znak /, polecenie rsync skopiuje do katalogu docelowego dane źródłowe.
>
> Jeśli na końcu ścieżki docelowej nie wstawiono znaku /, zamiast niego polecenie rsync umieści w ścieżce katalog o nazwie podobnej do nazwy katalogu źródłowego i skopiuje do niego dane źródłowe.

Oto przykład:

```
$ rsync -av /home/test /home/backups/
```

Powyższe polecenie kopiuje dane źródłowe (katalog */home/test*) do istniejącego katalogu o nazwie *backups*.

```
$ rsync -av /home/test /home/backups
```

Powyższe polecenie kopiuje dane źródłowe (katalog */home/test*) do stworzonego przez siebie katalogu o nazwie *backups*.

To nie wszystko

Polecenie rsync oferuje kilka dodatkowych funkcji, które mogą być określone za pomocą opcji wiersza poleceń. Zaznajom się z nimi.

Wykluczanie plików podczas archiwizowania przy użyciu programu rsync

Niektóre pliki nie wymagają aktualizowania podczas archiwizowania w zdalnej lokalizacji. Możliwe jest poinstruowanie polecenia rsync, aby wykluczyło określone pliki z bieżącej operacji. Pliki mogą być wykluczone za pomocą dwóch opcji. Pierwsza jest następująca:

```
--exclude WZORZEC
```

Możesz określić wieloznaczny wzorzec plików do wykluczenia. Oto przykład:

```
$ rsync -avz /home/code/some_code /mnt/disk/backup/code --exclude "*.txt"
```

Powyższe polecenie wyklucza pliki *.txt* z procesu tworzenia kopii zapasowej.

Możliwe jest też określenie listy plików do wykluczenia przez podanie nazwy pliku z listą.

W tym celu użyj opcji `--exclude-from` *ŚCIEŻKA_DO_PLIKU*.

Usuwanie nieistniejących plików podczas aktualizowania kopii zapasowej programu rsync

Pliki są archiwizowane jako archiwum programu tar, które jest transferowane do zdalnego miejsca składowania kopii zapasowych. Gdy konieczne jest zaktualizowanie danych kopii zapasowej, plik archiwum programu tar jest ponownie tworzony i przenoszony w miejsce prze-

chowywania kopii zapasowych. Domyślnie program `rsync` nie usuwa plików z miejsca docelowego, jeśli nie istnieją już w lokalizacji źródłowej. Aby z miejsca docelowego usunąć pliki, których nie ma w miejscu źródłowym, użyj opcji `--delete` polecenia `rsync`:

```
$ rsync -avz MIEJSCE_ŹRÓDŁOWE MIEJSCE_DOCELOWE --delete
```

Planowanie tworzenia kopii zapasowych w odstępach czasu

Możliwe jest utworzenie zadania programu `cron` w celu zaplanowania tworzenia kopii zapasowych w regularnych odstępach czasu.

Oto przykładowe polecenie:

```
$ crontab -e
```

Dodaj następujący wiersz:

```
0 */10 * * * rsync -avz /home/code użytkownik@ADRES_IP:/home/backups
```

Powyższy wpis programu `crontab` powoduje zaplanowanie uruchamiania polecenia `rsync` co 10 godzin.

Ciąg `*/10` określa pozycję godziny w składni polecenia `crontab`. Ciąg `/10` powoduje, że kopia zapasowa będzie tworzona co 10 godzin. Jeśli ciąg `*/10` umieszczono na pozycji minut, proces będzie wykonywany co 10 minut.

Aby dowiedzieć się, jak skonfigurować program `crontab`, przejdź do podrozdziału „Planowanie za pomocą programu cron” z rozdziału 9.

Tworzenie kopii zapasowych za pomocą narzędzia Git na podstawie kontroli wersji

Ludzie korzystają z różnych strategii tworzenia kopii zapasowej danych. Różnicowe kopie zapasowe są bardziej efektywne od tworzenia kopii całego katalogu źródłowego, gdy celem jest uzyskanie katalogu kopii zapasowych z numerem wersji zawierającym datę lub godzinę. Operacja kopiowania powoduje marnowanie przestrzeni dyskowej. Niezbędne jest jedynie skopiowanie zmian, które wystąpiły w plikach od chwili wykonania poprzednich kopii zapasowych. Taki proces jest nazywany tworzeniem przyrostowych kopii zapasowych. Możliwe jest ręczne tworzenie tego rodzaju kopii za pomocą takich narzędzi, jak np. `rsync`. Jednakże odtwarzanie tego rodzaju kopii zapasowych może być trudne. Najlepszym sposobem utrzymywania i odtwarzania zmian jest zastosowanie systemów kontroli wersji. Są one intensywnie wykorzystywane w przypadku tworzenia oprogramowania i zarządzania kodem, ponieważ z pisaniem kodu wiąże się częste wprowadzanie zmian. Git (*GNU it*) to najbardziej efektywny z dostępnych

systemów kontroli wersji. Może być używany do tworzenia kopii zapasowych zwykłych plików w kontekście niezwiązanym z programowaniem. System Git można zainstalować za pomocą menedżera pakietów dystrybucji. System ten został stworzony przez Linusa Torvaldsa.

Wprowadzenie

Oto opis problemu: istnieje katalog zawierający kilka plików i podkatalogów. Konieczne jest śledzenie zmian dokonywanych w treści katalogu i tworzenie dla nich kopii zapasowej. Jeśli dane ulegną uszkodzeniu lub zaginą, musi być możliwe odtworzenie ich poprzedniej kopii. Niezbędne jest archiwizowanie danych na komputerze zdalnym w regularnych odstępach czasu. Ponadto kopia zapasowa musi zostać umieszczona w różnych lokalizacjach na tym samym komputerze (host lokalny). Dowiedz się, jak to zrealizować za pomocą systemu Git.

Jak to zrobić

Dla katalogu, dla którego zostanie sporządzona kopia zapasowa, użyj polecenia:

```
$ cd /home/dane/źródłowe
```

Niech to będzie katalog źródłowy do śledzenia.

Utwórz i zainicjuj zdalny katalog kopii zapasowych. Na komputerze zdalnym utwórz docelowy katalog kopii zapasowych:

```
$ mkdir -p /home/backups/backup.git
$ cd /home/backups/backup.git
$ git init --bare
```

Na źródłowym komputerze hosta zostaną wykonane następujące kroki.

1. Do systemu Git na źródłowym komputerze hosta dodaj szczegóły dotyczące użytkownika:

   ```
   $ git config --global user.name "Jan Nowak"
   #ustawienie nazwy użytkownika "Jan Nowak"

   $ git config --global user.email slynux@slynux.com
   #  ustawienie adresu e-mail slynux@slynux.com
   ```

   Zainicjuj katalog źródłowy komputera hosta, dla którego zostanie sporządzona kopia zapasowa. Z poziomu tego katalogu, zawierającego pliki do zarchiwizowania, wykonaj następujące polecenia:

   ```
   $ git init
   Initialized empty Git repository in /home/backups/backup.git/
   #  inicjalizowanie repozytorium systemu Git

   $ git commit --allow-empty -am "Init"
   [master (root-commit) b595488] Init
   ```

2. Z poziomu katalogu źródłowego wykonaj następujące polecenie, aby dodać zdalny katalog systemu Git i zsynchronizować kopię zapasową:

```
$ git remote add origin użytkownik@zdalny_host:/home/backups/backup.git

$ git push origin master
Counting objects: 2, done.
Writing objects: 100% (2/2), 153 bytes, done.
Total 2 (delta 0), reused 0 (delta 0)
To użytkownik@zdalny_host:/home/backups/backup.git
 * [new branch]      master -> master
```

3. Dodaj lub usuń pliki w powiązaniu ze śledzeniem przez system Git.

   Następujące polecenie dodaje do listy archiwizowania wszystkie pliki i katalogi znajdujące się w bieżącym katalogu:

```
$ git add *
```

   W następujący sposób warunkowo do listy archiwizowania mogą zostać dodane tylko niektóre pliki:

```
$ git add *.txt
$ git add *.py
```

   Używając następującego polecenia, możesz usunąć pliki i katalogi, które nie wymagają śledzenia:

```
$ git rm plik
```

   W poleceniu możesz określić katalog, a nawet symbol wieloznaczny. Oto przykład:

```
$ git rm *.txt
```

4. Określ punkty kontrolne lub punkty archiwizacji.

   Przy użyciu następującego polecenia możesz określić punkty kontrolne dla kopii zapasowej z komunikatem:

```
$ git commit -m "Komunikat zatwierdzenia"
```

   Konieczne jest aktualizowanie w regularnych odstępach czasu kopii zapasowej w lokalizacji zdalnej. A zatem skonfiguruj zadanie programu `cron`, które na przykład tworzy kopie zapasowe co 5 godzin.

   Za pomocą następującego wiersza utwórz wpis w pliku programu `crontab`:

```
0 */5 * * *  /home/data/backup.sh
```

   Utwórz skrypt */home/data/backup.sh*:

```
#!/bin/ bash
cd /home/dane/źródłowe
git add .
git commit -am "Zatwierdzenie - @ $(date)"
git push
```

   W ten sposób przygotowano system tworzenia kopii zapasowych.

5. Odtwórz dane za pomocą systemu Git.

   Aby wyświetlić wszystkie wersje kopii zapasowej, użyj polecenia:

```
$ git log
```

Zaktualizuj zawartość bieżącego katalogu przy użyciu ostatniej kopii zapasowej, ignorując wszelkie ostatnie zmiany.

- W celu przywrócenia dowolnego wcześniejszego stanu lub wersji sprawdź identyfikator zatwierdzenia, który jest 32-znakowym łańcuchem szesnastkowym. Tego identyfikatora użyj z poleceniem `git checkout`.
- W przypadku identyfikatora zatwierdzenia `3131f9661ec1739f72c213ec5769bc0abefa85a9` polecenie będzie miało następującą postać:

```
$ git checkout 3131f9661ec1739f72c213ec5769bc0abefa85a9

$ git commit -am "Odtwarzanie - @ $(date) Identyfikator zatwierdzenia:
3131f9661ec1739f72c213ec5769bc0abefa85a9"
$ git push
```

- W celu ponownego wyświetlenia szczegółów dotyczących wersji użyj polecenia:

```
$ git log
```

Jeśli z jakichś powodów zawartość bieżącego katalogu uległa uszkodzeniu, konieczne jest odtworzenie jej przy użyciu kopii zapasowej znajdującej się w zdalnej lokalizacji.

Aby zrealizować to zadanie, wykonaj następujące polecenie:

```
$ git clone użytkownik@zdalny_host:/home/backups/backup.git
```

Polecenie to spowoduje odtworzenie całej zawartości katalogu z kopii zapasowej.

Klonowanie dysku twardego i innych dysków za pomocą programu dd

Podczas pracy z dyskami twardymi i partycjami może być konieczne utworzenie kopii dysku lub sporządzenie kopii zapasowych kompletnych partycji zamiast kopiowania całej ich zawartości (proces tworzenia kopii obejmuje nie tylko partycje dysku twardego, ale też kopię całej jego zawartości wraz z takimi danymi, jak rekord rozruchu, tabela partycji itp.). W tej sytuacji można użyć polecenia `dd`. Może ono posłużyć do klonowania dowolnego typu dysków (np.: dysków twardych, dysków Flash, dysków CD, dysków DVD, dyskietek).

Wprowadzenie

Nazwa polecenia `dd` stanowi skrót od *Data Definition*. Ponieważ niewłaściwe użycie polecenia prowadzi do utraty danych, spotykane jest też rozwinięcie skrótu `dd` jako *Data Destroyer* (niszczyciel danych). Ważna jest kolejność użycia argumentów. Niepoprawne argumenty mogą spo-

wodować utratę wszystkich danych lub sprawić, że te dane staną się bezużyteczne. Polecenie `dd` to w zasadzie duplikator strumienia bitów, który zapisuje cały strumień z dysku w pliku lub zachowuje plik na dysku. Dowiedz się, jak korzystać z polecenia `dd`.

Jak to zrobić

Składnia polecenia `dd` jest następująca:

```
$ dd if=MIEJSCE_ŹRÓDŁOWE of=MIEJSCE_DOCELOWE bs=WIELKOŚĆ_BLOKU count=LICZBA
```

W tym poleceniu:

- argument `if` odpowiada ścieżce urządzenia lub pliku wejściowego;
- argument `of` odpowiada ścieżce urządzenia lub pliku wyjściowego;
- argument `bs` reprezentuje wielkość bloku (zwykle jest to wynik podniesienia wartości do drugiej potęgi, na przykład: 512, 1024, 2048 itd.). Łańcuch *`LICZBA`* identyfikuje liczbę bloków do skopiowania (wartość całkowita).

  Całkowita liczba skopiowanych bajtów = *`WIELKOŚĆ_BLOKU`* * *`LICZBA`*.

  Argumenty `bs` i `count` są opcjonalne.

Określając wartość łańcucha *`LICZBA`*, możesz ograniczyć liczbę bajtów do skopiowania z pliku wejściowego do docelowego. Jeśli nie podano tej wartości, polecenie `dd` będzie kopiować dane z pliku wejściowego do momentu osiągnięcia znaku końca pliku `EOF`.

Aby skopiować partycję do pliku, wykonaj polecenie:

```
# dd if=/dev/sda1 of=partycja_sda1.img
```

W tym przypadku `/dev/sda1` jest ścieżką urządzenia partycji.

W następujący sposób odtwórz partycję przy użyciu kopii zapasowej:

```
# dd if=partycja_sda1.img of=/dev/sda1
```

W przypadku argumentów `if` i `of` należy zachować ostrożność. Niewłaściwe ich użycie może spowodować utratę danych.

Zmieniając ścieżkę urządzenia `/dev/sda1` na właściwą, możesz skopiować lub odtworzyć dowolny dysk.

Jeśli chcesz trwale usunąć wszystkie dane partycji, musisz sprawić, że program `dd` zapisze partycję zerami. Oto odpowiednie polecenie:

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

Ścieżka `/dev/zero` reprezentuje urządzenie znakowe, które zawsze zwraca nieskończoną liczbę znaków `\0`.

W następujący sposób sklonuj dysk twardy na innym dysku twardym o takiej samej wielkości:

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

Ścieżka `/dev/sdb` reprezentuje drugi dysk twardy.

Aby utworzyć obraz dysku CD-ROM (plik ISO), użyj polecenia:

```
# dd if=/dev/cdrom of=cdrom.iso
```

To nie wszystko

Po utworzeniu systemu plików w pliku generowanym przez program `dd` możesz go podłączyć za pomocą punktu podłączania. Dowiedz się, jak pracować z tym plikiem.

Podłączanie plików obrazów

Dowolny plik obrazu utworzony za pomocą polecenia `dd` może być podłączony przy użyciu metody pętli zwrotnej. Wraz z poleceniem `mount` zastosuj argument `-o loop`:

```
# mkdir /mnt/punkt_podłączenia
# mount -o loop plik.img /mnt/punkt_podłączenia
```

Za pośrednictwem katalogu */mnt/punkt_podłączenia* możesz uzyskać dostęp do zawartości plików obrazów.

Zobacz również

- W podrozdziale „Tworzenie plików ISO (hybrydowe pliki ISO)” z rozdziału 3. objaśniono, jak za pomocą polecenia `dd` utworzyć plik ISO dla dysku CD.

Skorowidz

R

S

Ś

T

U

W

Z

Ż

Skrypty powłoki systemu Linux

Receptury

GNU/Linux oferuje kompletne środowisko programistyczne — stabilne, niezawodne, z wyjątkowymi możliwościami. Powłoka z interfejsem przeznaczonym do komunikacji z systemem operacyjnym umożliwia całościowe kontrolowanie tego systemu. Zrozumienie skryptów powłoki ułatwi Ci więc zorientowanie się w systemie operacyjnym, a dodanie zaledwie kilku wierszy skryptu pozwoli zautomatyzować większość ręcznie wykonywanych zadań. Dzięki temu zaoszczędzisz ogromną ilość czasu!

Ta książka w całości jest poświęcona skryptom powłoki systemu operacyjnego Linux. Przedstawia rozwiązania w postaci sprawdzonych receptur. Pomogą Ci one błyskawicznie zrobić kopię offline stron internetowych, dokonać modyfikacji w plikach oraz przygotować kopię bezpieczeństwa. Kilka wierszy kodu wystarczy, by uzyskać rozwiązania pozwalające zrealizować wiele złożonych zadań obsługiwanych przez skrypty powłoki systemu Linux. Taką wydajność zapewni Ci właściwe użycie poleceń powłoki w odniesieniu do praktycznych zastosowań. Książka ta ma jeszcze jedną ogromną zaletę: w czasie lektury z pewnością wpadniesz na pomysły ulepszenia swojego systemu operacyjnego oraz poznasz możliwoś ci takich narzędzi, jak sed, awk, grep. Na co czekasz?

Otwórz tę książkę i sprawdź:

- jak stosować wyrażenia regularne
- jak stworzyć archiwum
- po co Ci kopia bezpieczeństwa
- jak monitorować pracę użytkowników
- jaki jest poziom wykorzystania dysków twardych

Zautomatyzuj codzienne zadania administratora!

ISBN 978-83-246-3886-4

9 788324 638864

Cena: 59,00 zł

Informatyka w najlepszym wydaniu