SelfLinux-0.10.0

Backups

Autor: Johnny Graber (*selflinux@jgraber.ch*)
Formatierung: Alexander Fischer (*Selflinux@tbanus.org*)
Lizenz: GFDL

Das Anlegen eines Backups sollte man nicht so lange hinauszögern, bis man es brauchen könnte. In dem Fall ist es zu spät und man hat nichts mehr, das sich sichern lässt. Dieser Text soll einige grundlegende Begriffe erklären und einem Ermöglichen zu entscheiden, was gesichert werden muss und was nicht.

Inhaltsverzeichnis

1 Begriffe

1.1 Backup

Mit Backup wird die Datensicherung bezeichnet. Der Sinn dieser Sicherung ist es, in möglichst kurzer Zeit wieder über all seine Daten zu verfügen, sollten diese durch einen äusseren Einfluss, einen Bedienungs- oder Programmfehler zerstört worden sein.
Man muss sich schon vorher Gedanken über die Datensicherung machen, denn wenn man das Backup brauchen könnte, ist es bekanntlich zu spät für eine Sicherung.

1.2 Vollständiges Backup

Bei einem vollständigen Backup werden alle zu sichernden Daten auf einmal gesichert. Der Vorteil liegt darin, das all diese Daten in einem Durchgang zurückgespielt werden können, sollte dies notwendig werden.
Bei den heutigen Standardinstallationen der Distributionen wird man aber ein Problem haben, sein ganzes System auf ein Medium zu schreiben. Einen Streamer haben die wenigsten Heimanwender und auf eine CD passt ein komplettes System selten.

Zudem dauert ein ganzes Backup ziemlich lange. Da man meist ungeduldig ist, kommt die lange Dauer des Backups als Ausrede gerade recht. Wieso sollte man auch täglich unzählige Dateien speichern, die man schon seit Wochen nicht mehr angefasst hat?

1.3 Differentielles Backup

Bei einem differentiellen Backup werden nur die seit dem letzten vollständigen Backup geänderten Dateien gesichert. Dies spart sowohl Zeit als auch Speicherplatz.

Bei einer Wiederherstellung der Daten muss man aber mit mehreren Medien arbeiten. Einerseits das vollständige Backup einspielen und danach noch das differentielle Backup.

1.4 Inkrementelles Backup

Beim inkrementellen Backup werden nur die Dateien gespeichert, die sich seit dem letzten Backup, geändert haben. Daher eignet sich das inkrementelle Backup besonders für die tägliche Sicherung.

1.5 Backup-Medien

Als Backup-Medium kann man alles verwenden, auf das man Daten speichern kann. Dies reicht von der *Floppy-Diskette* über *Zips*, *CDs*, *DVD*, *Magnetbänder* bis hin zu *Festplatten*. Schafft man sich ein neues Backup System an, sollte man sich vorher Gedanken über die notwendige Kapazität machen. Dadurch erspart man sich einen Fehlkauf und hat auch beim Betrieb weniger Ärger. Jedes mal für ein paar Dateien extra noch ein weiteres Medium anfangen nervt einen sehr bald.

2 Backup-Strategien

2.1 Was soll gesichert werden?

Schlussendlich will man bei einem Datenverlust sein System möglichst schnell wieder herstellen können. Daher ergibt es sich, das man **alle persönlichen Daten und Einstellungen** sichern muss. Arbeitet man an **Projekten** oder mit **Datenbanken**, muss man diese auch ins Backup mit einbeziehen.

2.2 Was braucht man nicht zu sichern?

Es muss nichts gesichert werden, das man schon auf anderen CDs hat. Es macht wenig Sinn, bei jedem vollen Backup auch noch alle Programme, die mit der Distribution mitgeliefert wurden, zu sichern.
Braucht man für seine Arbeit aber zusätzliche Programme oder hat diese selber modifiziert, sollte man diese neben dem Backup auf eine Spezielle CD sichern. Diese Programme ändern sich ja kaum und wenn sie einmal gesichert sind, kann man diese wieder herstellen.

2.3 Wie geht man vor? Ein Beispiel

Wir nehmen an, dass wir 5 Dateien haben (a, b, c, d und e) und uns 8 CD-RWs zur Verfügung stehen.

Am **Montag** machen wir ein volles Backup und sichern dabei alle Dateien auf unsere CD1.
Am **Dienstag** ändern wir die Dateien b, c und d. Da unsere Arbeit sehr wichtig ist, machen wir ein differentielles Backup und sichern dabei die Dateien b, c und d auf die CD2.
Am **Mittwoch** ändern wir die Dateien d und e. Würden wir nun ein weiteres differentielles Backup machen, müssten wir die Dateien b, c, d und e sichern, da sich diese seit dem vollständigen Backup geändert haben. Machen wir stattdessen ein inkrementelles Backup, sichern wir nur die Differenzen zum letzten Backup - also die Dateien d und e auf CD3.

Mit dem inkrementellen Backup fahren wir nun täglich weiter und fangen jeden Tag eine neue CD an, bis wir am **nächsten Montag** wieder ein volles Backup machen. Wichtig ist dabei, das wir dies auf die CD8 und nicht auf die CD1 machen. Geht beim Backup etwas schief, ist sonst unser volles Backup zerstört und alle unsere anderen CDs nutzen uns nichts mehr.

Sobald die CD8 korrekt geschrieben wurde, sind die CDs 1-7 nicht mehr notwendig um das System wieder aufzusetzen. CD1 sollte man vorerst bei Seite legen und erst wieder für das nächste volle Backup verwenden. CD 2-7 können für die inkrementellen Backups der nächsten Tage verwendet werden und helfen dabei beim Einsparen von Kosten.

So machen wir an den ersten 8 Tagen 8 Backups und brauchen 8 CDs. Dies mag auf den ersten Blick übertrieben erscheinen. Aber wie viel kostet ein Rohling? Wie wichtig ist die Arbeit und wie viel Zeit steckt dahinter?
Berechnet man die Kosten eines Datenverlustes (Erstellungszeit * Stundensatz), überwiegt diese Summe um Längen den Preis von 8 CD-RWs. Zumal diese 8 CDs ja wiederverwendet werden können.

2.4 komprimierte Archive

Man stellt sich bald einmal die Frage, ob man die Archive nicht komprimieren könnte. Dadurch spart man zwar Platz, doch wird das Archiv anfälliger für Fehler.
Werden beim Schreiben des komprimierten Archivs nur wenige Bits falsch kopiert, kann das ganze Archiv unbrauchbar werden. Daher sollte man sicher sein, das beim Schreiben keine Fehler passieren.

3 tar

Mit `tar` können einzelne Dateien oder ganze Verzeichnisse in ein ein Archiv gepackt werden. Da `tar` für grössere Systeme geschrieben wurde, versucht es standardmässig auf ein Bandlaufwerk zu schreiben.
Dies stellt kein grosses Problem dar, da man mit der Option `-f` den Datenstrom in eine Datei (`f` für File) umleiten kann. Daher wird im Verlauf des Textes `tar` immer auch mit `-f` aufgerufen.

`tar` ist zwar ein älteres Programm, doch ist es sehr weit verbreitet und unter vielen Systemen verfügbar. So kann man davon ausgehen, `tar` immer zur Verfügung zu haben.

3.1 Vollständiges Backup mit tar

```
user@linux / # tar [Optionen] [Ziel] [Dateien und Verzeichnisse]
```

Will man eine neue Datei erstellen, genügen die Optionen `-cf`. Dabei werden aber die vorangestellten `/` der Pfadangaben entfernt. Will man die Dateien im Rahmen eines Restore zurückschreiben, braucht man aber die ganzen Pfade. Diese müssen beim Restore entweder angegeben werden oder beim Erstellen durch die Option `-P` explizit angefordert werden:

```
user@linux / # tar -Pcf /mnt/disk2/backup-1.tar /home/
```

Mit diesem Aufruf werden alle Dateien unter `/home` in die Datei `/mnt/disk2/backup-1.tar` gespeichert. Die gesamten Pfade bleiben erhalten, doch werden diese relativ von dem Ort aus eingetragen, von dem man `tar` aufruft.

Hinweis:
Durch die Verwendung von `-P` kann man die Dateien nur an ihren ursprünglichen Ort wiederherstellen. Will man sich alle Möglichkeiten offen halten, sollte man daher auf `-P` verzichten.

3.2 Inkrementelles Backup mit tar

`tar` wurde vor allem für ganze Backups gemacht. Es gibt aber einige mehr oder weniger einfache Möglichkeiten, inkrementelle Backups zu machen.

Die einfachste Form ist wohl die Verwendung der Option `-g` beim ersten vollen Backup.
Dabei werden Informationen über die gesicherten Dateien in eine externe Datei zeitstempel geschrieben:

```
user@linux / # tar -vcf /mnt/disk2/backup-1.tar -g zeitstempel /home/
```

Beim inkrementellen Backup lautet der Aufruf bis auf den Archivnamen gleich, doch sichert `tar` diesmal nur die Dateien, die sich seit dem vollen Backup geändert haben.

Man kann `tar` auch mitteilen, das es nur Dateien sichern soll, die sich seit einem bestimmten Datum geändert haben. Bei diesem Aufruf werden die seit dem 1. November modifizierten Dateien getart:

```
user@linux / # tar -N 2002-11-01 -Pcf /mnt/disk2/backup-1.tar /home/
```

Will man ein eigenes kleines Backupscript schreiben, das einem alle Dateien sichert, die neuer als 5 Tage sind, kann man `date` zur Mitarbeit bewegen:

```
user@linux / # tar -N $(date -d "now 5 days ago" +%Y-%b-%d) -Pcf
/mnt/disk2/backup-1.tar /home/
```

3.3 komprimierte Backups mit tar

Man kann ein nach dem zuvor erklärten Verfahren erstelltes Archiv auch komprimieren. Dazu genügt der Aufruf von `gzip`:

```
user@linux / # gzip backup-1.tar
```

Danach befindet sich im Verzeichnis eine Datei mit dem Namen *backup-1.tar.gz*. Um die Datei zu dekomprimieren, ruft man an der Stelle von `gzip` das Tool `gunzip` auf:

```
user@linux / # gunzip backup-1.tar.gz
```

Es gibt bei `tar` aber auch eine integrierte Methode. Mit der Option `-z` wird `gzip` durch `tar` aufgerufen und es gibt keinen Umweg über einen zweiten Befehl. Neben `gzip` kann auch `bzip2` für die Komprimierung verwendet werden. Dafür dient die Option `-j`.

```
user@linux / # tar -zcf /mnt/disk2/backup-1.tar /home/
user@linux / # ls

backup-1.tar.gz
```

3.4 Überprüfen mit tar

Eine der wichtigsten Schritte beim Anlegen eines Backups ist dessen Überprüfung. Mit der Option `-d` geht dies sehr einfach, sofern man noch in dem Verzeichnis ist, in dem man tar gestartet hatte und die Option `-P` verwendete:

```
user@linux / # tar -df /mnt/disk2/backup-1.tar
```

Wenn alles stimmt, gibt `tar` keine Fehlermeldung aus. Hat man `-P` nicht verwendet, muss man `tar` den Pfad mitgeben:

```
user@linux / # tar -C /home -df /mnt/disk2/backup-1.tar
```

Diese Option geht ebenfalls, wenn man zwischenzeitlich das Verzeichnis gewechselt hat.
Will man wissen, was alles im Archiv ist, genügt ein `-t`

```
user@linux / # tar -tf /mnt/disk2/backup-1.tar

/home/jg/datei1
/home/jg/datei2
```

```
/home/jg/datei3
/home/jg/datei4
```

3.5 Restore mit tar

Nachdem das Archiv ordentlich geschrieben und geprüft wurde, kann man sich erst einmal ruhig zurücklehnen. Da aber erfahrungsgemäss immer etwas passieren kann, sollte man auch etwas über die Wiederherstellung wissen. Die dafür notwendige Option ist `-x` (extract = extrahieren). Damit man weiss was gerade passiert, gibt es `-v` (verbose = wortreich):

```
user@linux / # tar -xvzf /mnt/disk2/backup-1.tar.gz
```

oder mit zusätzlicher Pfadangabe und einem nicht komprimierten Archiv:

```
user@linux / # tar -C /home -xvf /mnt/disk2/backup-1.tar
```

4 cpio

`cpio` (copy in/out) ist `tar` sehr ähnlich, doch kann es im Gegensatz dazu mit beschädigten Archiven umgehen. So kann man den unbeschadeten Teil des Archivs meistens noch retten. Allerdings funktioniert `cpio` nur auf Festplatten mit dem Dateisystem *ext2*.

4.1 Vollständiges Backup mit cpio

Will man die zu sichernden Dateien an `cpio` übergeben, müssen diese nach einem speziellen Muster mitgeteilt werden. Pro übergebene Zeile erwartet `cpio` einen Dateinamen. Daher verwendet man am Besten eine Pipe:

```
user@linux / # ls *.tex | cpio -o > /mnt/disk2/backup-2
```

Bei dieser Befehlskette werden zuerst alle TeX-Dateien (Endung .tex) aufgelistet und an `cpio` übergeben. Mit der Option `-o` wird die Datei *backup-2* erstellt.

Der Nachteil beim Aufruf über `ls` ist, das man keine Unterverzeichnisse sichern kann. Verwendet man stattdessen `find`, kann man dieses Problem mit der Option `-depth` umgehen:

```
user@linux / # find /home/jg/ -maxdepth 2 -depth | cpio -o >
/mnt/disk2/backup-3
```

4.2 Inkrementelles Backup mit cpio

Mit `find` kann man sich nicht nur Verzeichnisebenen anzeigen lassen, sondern Dateien auch auf ihre letzte Modifikation prüfen. Daher genügt die Ergänzung des vorhin verwendeten Befehls um `-mtime -5` um alle Dateien zu erhalten, die in den letzten 5 Tagen geändert wurden:

```
user@linux / # find /home/jg/ -mtime -5 -maxdepth 2 -depth | cpio -o >
/mnt/disk2/backup-4
```

Für weitere Optionen von `find` empfielt sich ein Blick in `man find`.

4.3 Überprüfen mit cpio

Ein Nachteil von `cpio` ist die fehlende direkte Überprüfung der gesicherten Dateien. Um einen wirklichen Vergleich zu machen, müsste man die Dateien in ein anderes Verzeichnis entpacken und mit Hilfe von `diff` sicherstellen, das die Dateien korrekt sind.

Es gibt aber wenigstens eine Möglichkeit, sich die Dateien im Archiv anzeigen zu lassen:

```
user@linux / # cpio -itvI /mnt/disk2/backup-2

-rw-r--r--    1 jg       jg            100 Nov  5 20:03 /home/jg/test1.tex
-rw-r--r--    1 jg       jg             91 Nov  1 19:24 /home/jg/test2.tex
-rw-r--r--    1 jg       jg            212 Nov  4 17:05 /home/jg/test3.tex
-rw-r--r--    1 jg       jg             69 Nov  5 15:38 /home/jg/test4.tex
```

Da die Option `i` das Archiv auspacken würde, setzt man um dies zu Verhindern ein `t` davor. Über `I` teilt man das Verzeichnis mit und dank `v` werden die Dateirechte ebenfalls ausgegeben.

4.4 Zurückspielen mit cpio

```
user@linux / # cpio -id %lt; /mnt/disk2/backup-2
```

Damit werden die Dateien aus dem Archiv an ihren Platz zurück kopiert, sofern die dort vorhandenen Dateien nicht identisch oder älter sind.