Wenn der Mini-GAU kommt
Kleine Programme zur Sicherung einzelner Rechner

Einzelne Rechner oder kleine Netze lassen sich bequem mit kleinen, einfachen Programmen sichern. Dieser Artikel stellt einige alte und auch neuere Bekannte vor.

von **Stefan Schumacher**

Und überhaupt: so groß zu sein
Ist unmanierlich und nicht fein!

Hermann Löns, Die Zwerge

In der letzten Uptimes 2012-3 gab ein Artikel Tipps für eine Datensicherungsstrategie und ihre Umsetzung [1]. Doch nicht jeder Rechner benötigt eine umfangreiche Datensicherungsstrategie und entsprechend komplexe Programme. Kleine Helfer, die direkt zur Hand sind, stellt daher dieser Artikel vor. Als Grundlage dient das Betriebssystem *NetBSD*, was insbesondere bei *dump* im Hinterkopf zu behalten ist.

dump und *restore*

dump und *restore* sind historisch gewachsen die bedeutendsten Backup-Programme unter Unix. Sie arbeiten auf Blockebene, also unterhalb der Dateischicht. Das heißt, dass Dump einen Verzeichnisbaum oder ein Dateisystem komplett mit allen Eigenschaften sichert, und nicht nur auf Dateiebene Dateien kopiert. Allerdings bleibt Dump dabei in der angegebenen Partition, folgt also Mountpoints nicht. Dennoch lassen sich so ganze Partitionen spiegeln und zurücksichern oder auf andere Rechner übertragen.

Da Dump an das Dateisystem angepasst ist, kann *dump(8)* nur FFS sichern, für LFS gibt es hingegen *dump_lfs(8)*. Die beiden Programme unterscheiden sich aber nicht in der Handhabung. Restore wiederum entpackt einen Dump unabhängig vom zu Grunde liegenden Dateisystem, es kann also ein Archiv auch auf einem NFS-Verzeichnis entpacken.

Dump unterstützt inkrementelle Backups durch die Verwendung sogenannter Dumplevels. Das Tool trägt bei jedem Backup in der Datei */etc/dumpdates* das entsprechende Datum ein und kann sie so später auslesen und aufbereiten. Ein Dump mit Dumplevel 0 erzeugt ein Komplettbackup, während ein Level größer als 0 die Änderungen seit dem letzten Backup inkrementell sichert. Level 2 sichert also alle Daten, die seit dem letzten Dump mit Level 1 hinzukamen oder geändert wurden. In */etc/dumpdates* wandern die Datumsangaben in ein menschenlesbares Format, wenn der Admin die Option `-u` übergibt und Dump auf eine ganze Partition ansetzt – nicht aber auf einzelne Dateien oder Verzeichnisse. Mit der Option `-T Datum` lässt sich alternativ ein Wunschdatum angeben, das Dump dann als aktuelle Systemzeit betrachtet. Dies ist nützlich für automatisierte Skripte. Die Option `-T Datum` schließt die Option `-u` aus.

Da Dump für gewöhnlich dazu dient, komplette Partitionen zu sichern, verfügt es über keine Ein- oder Ausschlusslisten, wie dies einige andere Programme bieten. Stattdessen prüft Dump, ob das Fileflag *NODUMP* gesetzt ist. Mit diesem Flag also lassen sich einzelne Dateien oder Verzeichnisse doch vom Dump ausschließen. Allerdings ist zu beachten, dass Dump mit Level 0 Fileflags ignoriert. Wer trotzdem die markierten Dateien nicht sichern möchte, übergibt die Option `-h0`.

```
01 $ chflags nodump .signature
02 $ ls -lo .signature
03 -rw-r--r--   1   stefan   stefan   nodump   \
           369   Feb 18 21:09   .signature
```

Dump lässt sich sowohl auf Dateisysteme als auch auf Partitionen ansetzen. Dies ist ein Anwendungsbeispiel für einen Dump-Skript auf mehrere Volumes:

```
01 #!/bin/sh

02 DATUM=`date +%y%m%d`
03 QUELLE=/home/

04 # Mediengröße in kB, 102000=>99MB,
05 # 715000=>698MB, 2097152000=>2000MB
06 MEDIUM=102000

07 #Größe des Verzeichnis in kB
08 MEDIEN=`du -sk $QUELLE | awk '{print $1}'`
```

```
09 # Medienzahl kalkulieren,
10 # Nachkomma wird abgeschnitten!
11 MEDIEN=$(($MEDIEN/$MEDIUM))

12 # Medienzahl aufrunden
13 MEDIEN=$(($MEDIEN+1))

14 # Dateinamen iterieren
15 DAT=${DATUM}_1
16 i=1
17 while [ $i -lt $MEDIEN ]
18 do
19  i=`expr $i + 1`
20  DATEIEN=${DATUM}_$i
21  DAT=${DAT},${DATEIEN}
22 done

23 echo "dump -0 -B $MEDIUM -f $DAT $QUELLE"
```

Die Option `-a` in folgenden Kommandos veranlasst Dump, die Größe des Archivs selbst festzulegen, anstatt Dateien voreingestellter Größe zu schreiben:

```
01 # dump -0u -f dump-02-31-12 -a /home
02 # dump -0u -f dump-02-31-12 -a /dev/wd0e
```

Dump kann mit der Option `-W` auch eine Übersicht der bisher erstellten Dumps aus */etc/dumpdates* erstellen und mit der Option `-w` die noch zu sichernden Partitionen angeben. Wer Archive bestimmter Größe erzeugen möchte, zum Beispiel für Streamer oder um sie auf eine DVD zu brennen, verfügt mit *dump(8)* über einige, teils historische Optionen. Von Interesse dürfte heute aber nur noch `-B` sein, die als Angabe die Größe der zu sichernden Archive in Kilobyte erwartet. Dazu braucht das Kommando aber zwingend den Namen jeden zu erzeugenden Archivs, da Dump(8) sonst automatisch nacheinander in die zuletzt angegebene Datei schreibt. Gibt der Admin beispielsweise drei Dateinamen an, erzeugt aber fünf Dateien, schreibt das Tool die dritte, vierte und fünfte Datei in die dritte. Der unglückliche Admin hätte also drei Sicherungsdateien vorliegen, die aber nur die erste, zweite und fünfte Sicherung enthalten.

Dump selbst kann keine Archive komprimieren. Dies besorgt aber eine Pipe an *bzip2* oder *gzip*, oder ein kleines Shellskript. Möglich ist ebenfalls der Einsatz im Netzwerk mittels *rdump(8)* und *rrestore(8)*. Dies ist aber aufgrund mangelnder Sicherheit wie fehlender Verschlüsselung nur in gesicherten Netzwerken ratsam, sodass in anderen Fällen zusätzlich die Umgebungsvariable *RCMD_CMD* einen SSH-Tunnel aufrufen sollte. Noch besser ist eine Pipe an SSH, sodass die Ausgabe auf einem Rechner im Netzwerk landet:

```
01 # dump -0a -f - home | \
           ssh operator@192.168.1.1 \
           dd of=/usr/backup/dump.0
```

Dump im Detail

Da Dump das wichtigste Sicherungsprogramm ist, verdient es einen genaueren Blick. Dies ist ein exemplarischer Log eines Sicherungslaufes:

```
01 root@# dump -1au -h0  -f - /home |gzip > home.1
02   DUMP: Found /dev/rwd1a on /home in /etc/fstab
03   DUMP: Date of this level 1 dump: Sat Dec 24 12:17:56 2005
04   DUMP: Date of last level 0 dump: Sun Dec 18 23:20:19 2005
05   DUMP: Dumping /dev/rwd1a (/home) to standard output
06   DUMP: Label: none
07   DUMP: mapping (Pass I) [regular files]
08   DUMP: mapping (Pass II) [directories]
09   DUMP: mapping (Pass II) [directories]
10   DUMP: estimated 41438 tape blocks.
11   DUMP: Volume 1 started at: Sat Dec 24 12:18:18 2005
12   DUMP: dumping (Pass III) [directories]
13   DUMP: dumping (Pass IV) [regular files]
14   DUMP: 40541 tape blocks
15   DUMP: Volume 1 completed at: Sat Dec 24 12:18:40 2005
16   DUMP: Volume 1 took 0:00:22
17   DUMP: Volume 1 transfer rate: 1842 KB/s
18   DUMP: Date of this level 1 dump: Sat Dec 24 12:17:56 2005
19   DUMP: Date this dump completed:  Sat Dec 24 12:18:40 2005
20   DUMP: Average transfer rate: 1842 KB/s
21   DUMP: level 1 dump on Sat Dec 24 12:17:56 2005
22   DUMP: DUMP IS DONE
23 root@#
```

Dieses exemplarische Log eines Sicherungslaufes gibt wertvolle Informationen wieder. Das zu sichernde Verzeichnis */home* ist als eigene Slice */dev/rwd1a* in */etc/fstab* aufgeführt (Zeilen 01 und 02). Dieser Sicherungslauf (Level 1) beginnt um 12:17 Uhr am 24.12.2005 (Zeile 03 und Zeile 21). Der letzte Sicherungslauf im Level 0 fand am 18.12. statt (Zeile 04). Der Dump wird durchgeführt (Zeile 05) und erfolgreich beendet (Zeile 22). Dazu wird eine Statistik unter anderem über Dauer, Platz und Übertragungsrate ausgegeben (Zeilen 06 bis 20).

Interessant sind die einzelnen Durchläufe *Pass I* bis *Pass IV* (Zeilen 07 bis 13):

- Aus dem aktuellen Datum, dem angegebenen Level (hier: 1) und den letzten relevanten Durchläufen – aufgezeichnet in */etc/dumpdates* – berechnet Dump die Variable *DUMP_SINCE*. Es traversiert alle Inodes im Dateisystem und vergleicht deren *modification time* (mtime) mit der Größe von *DUMP_SINCE*. Ist die *mtime* einer Datei gleich groß oder größer, wandert der Inode in die Liste zu sichernder Dateien. Nicht zugeordnete Inodes ignoriert Dump, sodass am Ende des ersten Durchlaufes drei Listen vorliegen: die Inodes der Dateien, die zu sichern sind; die Inodes aller Verzeichnisse; und die zugeordneten Inodes.
- Durchlauf II läuft in mehreren Stufen ab. Als erstes traversiert Dump erneut die in Durchlauf 1 erstellte Liste der Verzeichnisse und prüft, ob zu sichernde Dateien im Verzeichnis existieren. Wenn nicht, schmeißt es den entsprechenden Inode aus der Liste der zu sichernden

Verzeichnisse wieder hinaus. Als zweites überprüft es die Verzeichnisse erneut, indem es den Verzeichnisbaum von den Blättern zur Wurzel durchläuft. Dabei findet es im letzten Durchlauf frei gewordene Verzeichnisse. Im Beispiel fand es keine frei gewordenen Verzeichnisse, da sonst ein weiterer *Pass II* stattgefunden hätte.

- Vorbereitungen zu *Pass III*: Bevor Dump mit dem Schreiben der Daten beginnt, legt es deren Metainformationen ab. Hierzu schreibt es einen *Header*, der die nachfolgenden Daten beschreibt. Zusätzlich schreibt das Tool zwei Inode-Tabellen – eine im nativen Format, und eine aus Kompatibilitätsgründen im normalisierten alten BSD-Format.
- Jetzt entsteht ein Header, der den Inode des Verzeichnisses enthält, und die Datenblöcke für jedes Verzeichnis in der Liste der zu sichernden Verzeichnisse werden geschrieben (Zeile 12).
- Jetzt entsteht ein Header, der den Inode der Datei enthält, und die Datenblöcke für jede Datei in der zu-sichernde-Dateien-Liste werden geschrieben (Zeile 13).
- Nachbereitung von *Pass IV*: Der abschließende Header wird aufs Band geschrieben.

Inkonsistenzen in der Sicherung treten auf, wenn das Quelllaufwerk während der Sicherung beschrieben wird. In diesem Fall würden einzelne Dateien nicht oder falsch gesichert. Eine Korrumpierung der gesamten Sicherung ist aber unwahrscheinlich. Vermeiden lassen sich diese Inkonsistenzen, wenn der Backup-Admin das Dateisystem unmountet, oder wenn er Snapshots verwendet.

restore

Restore spielt mittels der Option `-r` ein komplettes Archiv zurück. Dabei erzeugt es eine Datei namens *restoresymtable*, in der es Meta-Informationen zur Rücksicherung ablegt – also Inodes, oder welche Bänder. Diese Datei kann der Admin nach der Rücksicherung aller (!) Archive entfernen:

```
01 # restore -r -f 02-11-01 && \
          restore -r -f 02-11-08
02 # rm restoresymtable
```

Wer nur bestimmte Pfade oder Dateien zurückspielen möchte, übergibt Teile des Verzeichnisses mit der Option `-x`. Leider unterstützt Restore keine regulären Ausdrücke, aber man kann sich mit ein paar Backticks behelfen. Um zu prüfen ob ein Archiv erfolgreich geschrieben wurde, reicht es zu Testzwecken, die allerletzte Datei zu restaurieren. Da Dump alle vorigen Einträge zumindest traversieren muss, erkennt es Fehler im Archiv und meldet sie. In der Dateiliste (Option `-t`) stehen die Inodes und die korrespondierende Dateinamen, man kann also den letzten Eintrag zurücksichern:

```
01 $ restore -x stefan/ www/
02 $ restore -t > dateiliste
03 $ restore -x `grep -i 'bz2$' dateiliste | \
           awk '{print $2}'`
04 $ restore -x `tail -n 1 dateiliste | \
           awk '{print $2}'`
```

Ist eine bestimmte Datei im gesicherten Verzeichnisbaum gewünscht, ruft `restore -i` eine Shell auf, in der sich mit `ls` und `cd` der Verzeichnisbaum durchkämmen lässt. Ist die gewünschte Datei gefunden, fügt `add` sie zur Liste der zurückzusichernden Dateien hinzu. Mit `extract` werden wiederum die in der Liste spezifizierten Dateien im aktuellen Verzeichnis entpackt. Um einen unterbrochenen Lauf fortzusetzen, dazu dient die Option `-R`.

tar(1)

tar(1) ist das wohl bekannteste Sicherungsprogramm. Allerdings gibt es verschiedene Versionen und Implementierungen, sodass man darauf achten sollte, nur kompatible Versionen einzusetzen und mithin kompatible Archive zu erzeugen.

Tar steht für „Tape Archiver“, es entstand also zu dem Zweck, Bandarchive zu beschreiben. Es erzeugt dazu eine Datei, die alle zu sichernden Dateien enthält, und schreibt diese auf ein Band – oder in eine einfache Datei. Die Option `f` erzeugt eine Datei – standardmäßig würde Tar das Archiv sonst auf das erste Bandlaufwerk `/dev/nrst0` schreiben.

```
01 $ tar cpfz  \
        operator@backup:/home/backup/backup.tgz \
       /home/
02 $ tar xpfz backup.tgz
```

Tar schreibt relativ zum jetzigen Pfad, das im Beispiel gesicherte Home-Verzeichnis wird also an aktueller Stelle entpackt. Die Option `z` erzeugt ein Gzip-Archiv. Mit `-j` ließe sich auch ein Bzip2-komprimiertes Archiv erzeugen. Zurückspielen lässt sich das Backup mit `x`. Die Option `p` sorgt dafür, dass die Dateirechte bestehen bleiben:

Es ist wichtig zu wissen, dass Tar große Probleme mit defekten Archiven hat, da bei einem Archiv ein Header erzeugt wird, der Informationen über das gesamte Archiv enthält. Taucht an einer Stelle ein Fehler auf, entpackt Tar das Archiv nur bis zu dieser Fehlerstelle.

star

star ist eine von Jörg Schilling seit 1982 entwickelte Tar-Implementierung. Es ist die weltweit schnellste Tar-Variante und besitzt einige nützliche Erweiterungen. Es unterstützt unter anderem reguläre Ausdrücke, Levels (so wie Dump), begrenzt die Pfadlänge nicht, erkennt automatische die Endianness und ermöglicht, ein Dateisystem mit einem Tarball zu vergleichen. In der Standardvariante kann Star Tar-Archive lesen und umgekehrt.

Star lässt sich aus *pkgsrc/archivers/star* installieren und ist im Allgemeinen dem normalen Tar der Vorzug zu geben. Star unterstützt verschiedene Tar-Formate, die mit der Option `-H=` bestimmt werden können. Am leistungsfähigsten ist *EXUSTAR*.

cpio(1)

cpio(1) (von „copy in/out“) ist von der Syntax her etwas eigen, da es als erste Option die Übergabe einer zu sichernden Datei erwartet und diese auf *STDOUT* schreibt. Das erweist sich allerdings mit Hilfe der Pipe als Vorteil, da der Kommandeur so mit *find(1)* oder *ls(1)* die Dateiliste übergeben und die Ausgabe mit > umleiten kann. Dateien mit Cpio zu sichern, sieht folgendermaßen aus:

```
01 $ ls *.jpg | cpio -0 > /home/Bilder.cpio
02 $ find /home -mtime 7 | cpio -o > /dev/nrst0
03 $ cpio -i < /dev/nrst0
```

Dieses Listing zeigt in Zeile 01, wie man alle JPGs im aktuellen Verzeichnis in die Datei *Bilder.cpio* sichert. Die Ausgabe kann stattdessen auch auf das Bandlaufwerk */dev/nrst0* umleiten. Um Unterverzeichnisse zu sichern oder inkrementelle Backups zu erzeugen, nutzt man *find(1)*. Zeile 02 sichert alle Dateien, die in den letzten sieben Tagen geändert wurden, auf das erste Bandlaufwerk. Zeile 03 spielt schließlich die Sicherung zurück.

Leider unterstützt Cpio keine Überprüfung der Archive, sodass dies manuell zu erledigen ist. Dazu entpackt man das Archiv wieder und vergleicht die Dateien mit *diff* oder *mtree*.

afio(1)

afio(1) ist eine Neuauflage von Cpio und behebt einige Probleme, etwa mit leeren Dateien, oder it auf 0 gesetzten Rechten. Es verwendet dieselbe Syntax wie Cpio. Das Tool ist via *pkgsrc/archivers/afio* installierbar und sollte Cpio vorgezogen werden, ebenso wie Star bei Tar.

pax(1)

pax(1) ist ein Standardisierungsversuch der *IETF*, da es zu viele Implementierungen von Tar und Cpio gibt. Pax ist in der Lage, verschiedene Archivformate wie Cpio und Tar zu schreiben oder zu lesen. Daher ist es in heterogenen Umgebungen nützlich. Das Tool wird unter anderem auch auf den Installationsmedien von NetBSD verwendet. So sieht eine Sicherung und Rücksicherung mit Pax aus:

```
01 # pax -w -f /dev/nrst0 /home
02 # pax -v -f /dev/nrst0
03 $ pax -r -s ',^/*usr/* »' -f a.pax
```

Zeile 01 schreibt das Home-Verzeichnis auf Band. Zeile 02 gibt ein Inhaltsverzeichnis der Sicherung aus. Der letzte Befehl packt alle Dateien in */usr* in das Archiv *./a.pax*.

Literatur

[1] Stefan Schumacher: Wenn der GAU kommt. Datensicherungsstrategie erarbeiten und umsetzen, Uptimes 2012-3, S. 21-29: http://www.guug.de/uptimes/2012-3/index.html

Über Stefan



Stefan Schumacher ist geschäftsführender Direktor des Magdeburger Instituts für Sicherheitsforschung und gibt zusammen mit Jan W. Meine das Magdeburger Journal zur Sicherheitsforschung heraus. Er befasst sich seit knapp 20 Jahren als Hacker mit Fragen der Informations- und Unternehmenssicherheit und erforscht Sicherheitsfragen aus pädagogisch-psychologischer Sicht. Seine Forschungsergebnisse stellt er auf Fachkongressen und in Publikationen vor. Seine Schwerpunkte liegen auf Social Engineering, Security Awareness, Organisationssicherheit, internationale Cyber-Security und Mensch-Maschine-Interaktion.

