HOWTO zur Datensicherung

(C) 2006-2025 T.Birnthaler/H.Gottschalk <howtos(at)ostc.de>
OSTC Open Source Training and Consulting GmbH
http://www.ostc.de

$Id: unix-backup-HOWTO.txt,v 1.28 2025/02/23 20:14:55 tsbirn Exp $

Dieses Dokument beschreibt die Datensicherung (Backup) und die
Datenwiederherstellung (Restore) mittels UNIX-Bordmitteln.

INHALTSVERZEICHNIS

1) Einführung
2) Lokale Datensicherung
2.1) "tar" (tape archiver)
2.2) "cpio" (copy input output)
2.3) "pax" (Portable Archive Exchange bzw. "Frieden" zw. "tar" und "cpio")
3) Datensicherung via Netzwerk
3.1) "ssh" (secure shell)
3.2) "netcat" (auch "nc", network concat)
4) Fazit
5) Weitere Datensicherungstools
6) Weiterführende Links

---

1) Einführung

Dieses HOWTO beschreibt die Datensicherung mittels der UNIX-Bordmittel "cpio", "tar" und "pax" in Verbindung mit "gzip" und "bzip2". Es werden Methoden zur lokalen Datensicherung und zur Datensicherung via Netzwerktransfer mittels "ssh" und "netcat" erläutert sowie auf Unterschiede der Tools auf verschiedenen Plattformen (Linux, OpenBSD) eingegangen. Allerdings werden nur recht spezielle Punkte beleuchtet, die u.E. für die praktische Anwendung der Tools sehr wichtig sind.

2) Lokale Datensicherung

Prinzipiell unterscheidet UNIX zwischen dem Erstellen des Archivs mittels "tar", "cpio" oder "pax "und dem Komprimieren desselben mittels "compress", "gzip" oder "bzip2". Allerdings können die Archivierungstools in den aktuellen Versionen den Komprimierungsschritt gleich mit durchführen.

2.1) "tar" (tape archiver)

Größter Vorteil von "tar": Es erzeugt Archive, die von WinZip gelesen werden können.

Größter Nachteil von "tar": Ein defektes Archiv ist ab dem Punkt des Fehlers verloren. Was dahinter noch an Daten käme, kann von "tar" nicht mehr gelesen werden.

Es ist eigentlich nicht nachzuvollziehen, warum "tar" in der Linuxwelt eine so herausragende Bedeutung beim Sichern von Daten erlangt hat.

"tar" liest ab einem oder mehreren Dateibaum-Startverzeichnissen rekursiv den gesamten Dateibaum und packt diesen zu einem Archiv. Die Startpunkte werden auf der Kommandozeile als Parameter übergeben. "tar" funktioniert NICHT als unixüblicher Filter. Lediglich die Ausgabe eines Archivs auf STDOUT und das Einlesen eines Archivs von STDIN kann erzwungen werden. Die Liste der Startpunkte kommt IMMER von der Kommandozeile.

"tar" sichert Besitzverhältnisse, Zeitstempel und Zugriffsrechte mit; ebenso kann es symbolische Links und Hardlinks mitsichern. Allerdings können keine Devices, Sockets und Named Pipes gesichert werden.

"tar" wandelt absolute Pfade immer in relative Pfade um (durch entfernen des "/" am Pfadanfang). So können die gesicherten Daten an eine beliebige wählbare Stelle zurückgeschrieben werden.

"tar" packt im aktuellen Verzeichnis aus, also vorher dort hinein wechseln. Oder (besser!) mit "-C PFAD" die Startposition für die Rückspielen festlegen, ohne dort hinein zu wechseln.

"tar"-Archiv erstellen:

```
  tar cf /tmp/archiv.tar startverz1 startverz2
  # c = create
  # f = Ausgabe auf File; "tar" ist dazu gedacht, auf ein Device auszugeben
  # v = verbose (Ausgabe der Namen der archivierten Files auf
  #               - STDOUT, wenn Archiv direkt in File geschrieben wird
  #               - STDERR, wenn Archiv auf STDOUT ausgegeben wird)
```

Vor Auspacken eines "tar"-Archivs IMMER erst dessen Inhalt inspizieren:

```
  tar tf /tmp/archiv.tar
  # t = Auflisten des Archivinhalts
  # f = Einlesen aus File ist gedacht, um aus einem Device zu lesen
  # v = verbose (Ausgabe der Namen der archivierten Files auf STDERR)
```

"tar"-Archiv auspacken:

```
  tar xf /tmp/archiv.tar
  # x = extract
  # f = Einlesen aus File ist gedacht, um aus einem Device zu lesen
  # v = verbose (Ausgabe der Namen der archivierten Files auf STDERR)
```

"tar"-Archiv erstellen und mit "bzip2" komprimieren:

```
  tar cf - startverz | bzip2 > /tmp/archiv.tbz2
  # Dateiname "-" bewirkt Ausgabe auf STDOUT (bzw. Einlesen von STDIN).
```

"tar"-Archiv erstellen und mit "gzip" komprimieren:

```
  tar cf - startverz | gzip -c > /tmp/archiv.tgz
```

Auf vielen Systemen kann "tar" die Komprimierung gleich mit erledigen:

```
  tar cfz /tmp/archiv.tgz startverz        # gzip
  tar cfj /tmp/archiv.tbz2 startverz       # bzip2 (nur unter Linux)
```

Um von einem Verzeichnisbaum eine exakte Kopie herzustellen, kann folgendes Kommando verwendet werden:

```
  ( cd SRCDIR; tar cpf - . ) | ( cd DESTDIR; tar xf - )
```

Natürlich kann dies auch mit "cp -a --preserve=all" erfolgen (-a = -dpR = --no-dereference --preserve=link,mode,ownership,timestamps --recursive).

Zwar archiviert "tar" immer einen gesamten Dateibaum, jedoch kann mittels Optionen die Auswahl der Files und Pfade in diesem Baum beeinflusst werden. Hierauf wird aber nicht näher eingegangen.

2.2) "cpio" (copy input output)

Im Gegensatz zu "tar" funktioniert "cpio" als KLASSISCHER FILTER. Ihm wird

auf STDIN eine Dateiliste übergeben, welche es zu einem Archiv packt und dieses auf STDOUT ausgibt. Wie diese Dateiliste generiert wird, ist irrelevant.

"cpio" kann diverse Formate, unter anderem das "tar"-Format. Aufgrund der Filtereigenschaften ist es insbesondere für Skripte deutlich besser geeignet als "tar" und kann auch WinZip-kompatible Archive erzeugen. Hält man sich nur im UNIX-Umfeld auf, ist das "tar"-Format nicht anzuraten.

Archiv erstellen:

```
  echo -e "/etc/passwd\n/etc/group" | cpio -o > /tmp/archiv.cpio
  # -o = cpio gibt Archiv auf STDOUT aus
```

Natürlich wird man kaum mittels "echo" die Dateiliste übergeben, aber bspw. aus einer Datei, in der zeilenweise Files eingetragen sind, ließe sich das Archiv erzeugen:

```
  cpio -o > /tmp/archiv.cpio < dateiliste.lst       # A
  cat dateiliste.lst | cpio -o > /tmp/archiv.cpio   # B
```

Geschickter ist natürlich, diese Dateiliste direkt zu erzeugen, bspw. mittels "find":

```
  find /etc -name "*.conf" | cpio -o > /tmp/archiv.cpio
```

Zu beachten ist, dass eventuell Sonderzeichen in Dateinamen vorkommen, die Probleme bereiten können. Unter Linux kann "cpio" das NULLBYTE als Filenamen-Trenner akzeptieren und "find" dieses Zeichen als Filenamen-Trenner erzeugen. Insofern bietet sich folgende Kommandokette an:

```
  find /etc -name "*.conf" -print0 | cpio -0 -o > /tmp/archiv.cpio
  # -0 (bzw. -print0) legt Nullbyte als Filenamen-Trenner fest (vgl. "xargs")
  # Leider ist Option "-0" (null) unter OpenBSD für "cpio" nicht verfügbar.
```

Ein "cpio"-Archiv lässt sich auch wieder komprimieren:

```
  find /etc -name "*.conf" | cpio -o | bzip2   > /tmp/archiv.cpio.bz2
  find /etc -name "*.conf" | cpio -o | gzip -c > /tmp/archiv.cpio.gz
```

Die "gzip"-Variante kann auch mittels des Schalters "-z" direkt von "cpio" erzeugt werden.

```
  find /etc -name "*.conf" | cpio -z -o > /tmp/archiv.cpio.gz
```

Entpacken lässt sich ein Archiv mit (-i=input):

```
  cpio -i < /tmp/archiv.cpio
```

Selbstverständlich gelten hier dieselben Regeln für das Komprimieren wie beim Einpacken.

Um eine Dateien von einem Verzeichnis in ein anderes zu übertragen kann

```
  ( cd SRCDIR; find . ) | cpio -o | ( cd DESTDIR; cpio -i )
  ( cd SRCDIR; find . ) | cpio -p DESTDIR
```

verwendet werden. Dies kann auch mit der Option "-p" (-p=pass-through) direkt erreicht werden:

Vorsicht bei "cpio" ist geboten wenn:

* Das Filesystem, auf dem die zu archivierenden Daten stehen, sehr groß ist. Dann bekommt "cpio" Probleme mit den Inodes und es können Hardlink-Informationen verloren gehen. "cpio" warnt davor mit "truncated inode number". In solchen Fällen muss das new SVR4-Format zum Einsatz kommen, welches mit größeren Filesystemen zurecht

kommt. (Linux: -H newc)

* Files archiviert werden müssen, die größer als 4GB sind. Im "Old ASCII"-Format (Linux: -H odc) liegt diese Grenze bei 8GB, es kann aber wieder Schwierigkeiten mit den Inodes geben.

Diese Limitierungen sind wohl ein Grund für den Einsatz von "tar". Nachdem die aktuellen Versionen von "cpio" die tar-Formate beherrschen --- insbesondere "ustar" --- sollte einfach dieses mit "cpio" verwendet werden. Dann ist der Nachteil von "tar", nämlich defekte tar-Archive nicht bearbeiten zu können behoben und die Restriktionen der cpio-Formate sind aufgehoben.

2.3) "pax" (Portable Archive Exchange bzw. "Frieden" zw. "tar" und "cpio")

Die beste Alternative ist "pax" (kann alles), denn es beherrscht alle Formate und behebt die Probleme der anderen Tools, selbst wenn es deren Formate erzeugt! Standardmäßig verwendet "pax" das Format "ustar". Evtl. kann "pax" aufgrund dieses Sachverhalts keine Sockets kopieren, dann ist ein anderes Format zu verwenden (Option "-x" FORMAT).

"pax" kann sowohl von STDIN lesen als auch von Kommandozeile Fileparameter übernehmen. Hier wird nur auf die Version von STDIN eingegangen.

Archiv erstellen (-w=write):

```
find /etc -name "*.conf" -print0 | pax -0 -w -f /tmp/archiv.pax
```

Archiv entpacken (-r=read):

```
pax -r -f /tmp/archiv.pax
cat /tmp/archiv.pax | pax -r
```

Um Dateien von einem Verzeichnis in ein anderes zu übertragen kann

```
( cd SRCDIR; find . -print0 ) | pax -0 -w | ( cd DESTDIR; pax -0 -r )
```

verwendet werden. "pax" kann dies aber auch selbst (-rw=read+write):

```
cd SRCDIR; pax -rw . DESTDIR
```

"pax" kennt noch viele Funktionen, auf die hier nicht näher eingegangen wird.

3) Datensicherung via Netzwerk

Prinzipiell gelten für die Datensicherung via Netzwerk die selben Bemerkungen wie bei der lokalen Datensicherung. Diese werden nur um Netzwerkaspekte erweitert. Zum Einsatz kommen "ssh" und "netcat", um die Netzwerkverbindung zu realisieren.

3.1) "ssh" (secure shell)

"ssh" bietet sich an, wenn eine verschlüsselte Datenübertragung notwendig ist. Diese Verschlüsselung benötigt natürlich Rechenkapazität der beteiligten Rechner. Bei niedriger Bandbreite der Netzwerkverbindung ist dies aber nicht der Flaschenhals der Datensicherung. Soll im lokalen Netz eine größere Datenmenge transferiert werden, ist die Verwendung von "ssh" nicht anzuraten.

Weiterhin kann "ssh" komprimieren ("ssh -C") und erlangt dabei die Effizienz von "gzip". Wird also eine exakte Kopie der Daten auf dem Zielrechner DESTHOST gewünscht, kann auf Komprimieren beim Erstellen des Archivs verzichtet werden.

```
( cd SRCDIR; tar cf - . ) |
  ssh -C DESTHOST "( cd DESTDIR; tar xf - )"
```

```
  ( cd SRCDIR; find . ) | cpio -o |
    ssh -C DESTHOST "( cd DESTDIR; cpio -i )"
  ( cd SRCDIR; find . -print0 ) | pax -0 -w |
    ssh -C DESTHOST "( cd DESTDIR; pax -0 -r )"
```

Soll das Archiv auf dem Zielrechner zu liegen kommen, muss dort STDOUT mittels "cat" in ein Archivfile ausgegeben werden:

```
  ( cd SRCDIR; tar cf - . ) |
    ssh -C DESTHOST "( cd DESTDIR; cat > archiv.tar )"
  ( cd SRCDIR; tar czf - . ) |
    ssh DESTHOST "( cd DESTDIR; cat > archiv.tgz )"
  ( cd SRCDIR; find . ) | cpio -o |
    ssh -C DESTHOST "( cd DESTDIR; cat > archiv.cpio )"
  ( cd SRCDIR; find . ) | cpio -z -o |
    ssh DESTHOST "( cd DESTDIR; cat > archiv.cpio.gz )"
  ( cd SRCDIR; find . -print0 ) | pax -0  -w |
    ssh -C DESTHOST "( cd DESTDIR; cat > archiv.pax )"
  ( cd SRCDIR; find . -print0 ) | pax -0 -z -w |
    ssh DESTHOST "( cd DESTDIR; cat > archiv.pax.gz )"
```

3.2) "netcat" (auch "nc", network concat)

Werden im lokalen Netz Daten übertragen, sollte "netcat" zum Einsatz kommen und bei exakten Kopien auf Komprimierung verzichtet werden. Das Komprimieren dauert dort meist länger als die Übertragung selbst. Sollen die Daten als Archiv platzsparend abgelegt werden, dann sollte natürlich komprimiert werden.

Auf dem Zielrechner muss mittels "netcat" ein LISTENER auf einem frei wählbaren Port (größer 1023) aufgebaut werden, der auf einem bestimmten Port lauscht (hier 4321) und alles, was dort ankommt, auf STDOUT ausgibt und so an das Komprimierungstool übergibt (-l=listen):

Linux (-l=Listener):

```
  cd DESTDIR; netcat -l -p 4321 | tar xf -     # A) tar
  cd DESTDIR; netcat -l -p 4321 | cpio -0 -i   # B) cpio
  cd DESTDIR; netcat -l -p 4321 | pax -0 -r    # C) pax
```

OpenBSD (-l=Listener):

```
  cd DESTDIR; nc -l 4321 | tar xf -    # A) tar
  cd DESTDIR; nc -l 4321 | cpio -i     # B) cpio
  cd DESTDIR; nc -l 4321 | pax -0 -r   # C) pax
```

Auf dem sendenden Rechner muss das Archiv erzeugt und via "netcat" zum Zielrechner DESTHOST übermittelt werden. "netcat" nimmt dabei auf STDIN Daten entgegen und schickt diese unverändert an den Listener auf dem Zielrechner.

Linux (Sender):

```
  ( cd SRCDIR; tar cf - . )                  | netcat DESTHOST 4321   # A) tar
  ( cd SRCDIR; find . -print0 | cpio -0 -o ) | netcat DESTHOST 4321   # B) cpio
  ( cd SRCDIR; find . -print0 | pax -0 -w )  | netcat DESTHOST 4321   # C) pax
```

OpenBSD (Sender):

```
  ( cd SRCDIR; tar cf - . )         | nc DESTHOST 4321   # A) tar
  ( cd SRCDIR; find . ) | cpio -o   | nc DESTHOST 4321   # B) cpio
  ( cd SRCDIR; find . ) | pax -0 -w | nc DESTHOST 4321   # C) pax
```

Anmerkung: "nc" unter OpenBSD ist ein Rewrite und unterscheidet sich von "netcat" unter Linux stark. Insbesondere die OpenBSD-Optionen -x und -X sind sehr nützlich!

4) Fazit

Wie man am besten die zu archivierenden Daten sichert, hängt von folgenden Faktoren ab:

* Datenaufbewahrung (komprimiertes Archiv, "live"-Dateien)?
* Netzwerkbandbreite?
* Ist aus Sicherheitsgründen eine verschlüsselte Übertragung notwendig?
* Wie groß sind die zu sichernden Dateien / Dateisysteme?
* Sollen auch Spezialdateien (z.B. Geräte) gesichert werden?

ACHTUNG: Generell gilt, dass:

* REKURSIVE Aufrufe (also ein Archiv im gerade zu archivierenden Teil des Dateibaums abzulegen) unvorhersehbare Folgen haben können. Das Zielverz. DESTDIR darf also nie Teil des zu archivierenden Quellbaums sein!

* Falls Dateien auf dem Zielsystem/-verzeichnis nicht als Archiv, sondern "echt" abgelegt werden, dort ROOTRECHT benötigt wird, um die Dateien anderen Usern zuweisen zu können.

5) Weitere Datensicherungstools

* http://freecode.com/projects/afio	afio
* http://members.chello.nl/~k.holtman/afio.html	afio
* http://www.amanda.org/	Amanda
* http://www.areca-backup.org/	Areca-Backup
* http://www.arkeia.com/	Arkeia
* http://backintime.le-web.org/	BackInTime
* http://backuppc.sourceforge.net/	BackupPC
* http://www.bacula.org/	Bacula
* http://www.clonezilla.org/	Clonezilla
* http://dar.linux.free.fr/	dar (Disk ARchiver)
* http://www.gnu.org/software/coreutils/	dd
* http://www.garloff.de/kurt/linux/ddrescue/	ddrescue
* http://www.dirvish.org/	Dirvish
* http://dump.sourceforge.net/	dump/restore
* http://duplicity.nongnu.org/	Duplicity
* http://faubackup.sourceforge.net/	faubackup
* http://flyback-project.org/	FlyBack (Apple TimeMachine under Linux)
* http://www.fsarchiver.org/	FSArchiver (Systemdisk sichern+wiederherst.)
* http://luckybackup.sourceforge.net/	luckybackup
* http://mkcdrec.sourceforge.net/	mkCDrec (Linux Desaster Recovery; Nachfolger: ReaR)
* http://www.mondorescure.org/	MondoRescue
* http://www.partimage.org/	partimage
* http://www.nongnu.org/rdiff-backup	rdiff-backup
* http://www.relax-and-recover.org/	ReaR (Relax and Recover; Vorgänger: mkCDrec)
* http://www.remastersys.com/	Remastersys (Debian/Ubuntu full system backup)
* http://www.rsnapshot.org/	Rsnapshot
* http://samba.org/rsync/	rsync
* http://launchpad.net/sbackup	sbackup (simple backup für GNOME Desktop)
* http://launchpad.net/timevault	TimeVault (GNOME Backup/Snapshot System)
* http://www.zmanda.com/	Zmanda

6) Weiterführende Links

* http://www.freebsd.org/doc/de/books/handbook/backup-basics.html Backup Basics (FreeBSD Handbuch)
* https://codeberg.org/schilytools/schilytools/src/branch/master/star Star --- unique standard tape archiver
* https://sourceforge.net/projects/s-tar/ Star --- unique standard tape archiver
* https://www.mankier.com/5/star Star --- unique standard tape archiver
* http://www.heise.de/kiosk/archiv/ix/02/12/134_Packesel Packesel --- Dateien archivieren mit tar (iX 12/2002)
* http://www.heise.de/ix/artikel/2003/01/140/default.shtml Pax vobiscum --- Werkzeuge zur Archivierung mit Eigenheiten: cpio und pax (iX 10/2003)
* http://coredump.c-base.org/coredump/TaoDesBackup TaoDesBackup (deu)
* http://www.taobackup.com/ The Tao of Backup (eng)
* http://www.coredumps.de/doc/dump/zwicky/testdump.doc.html Torture-testing Backup and Archive Programs
* http://www.heise.de/ix/artikel/2002/01/139/ Zauberhaft --- Identifizierung von Datei- und MIME-Typen mit file (iX 1/2001)