

UCS Backup

Thema:	Dokumentation von Unidump und Remote Backup unter UCS.
Datum:	9. Dezember 2010
Seitenzahl:	41
Versionsnummer:	7397
Autoren:	Univention GmbH feedback@univention.de

Inhaltsverzeichnis

1	Sicherung auf Bandlaufwerken mit Unidump	3
1.1	Einführung	3
1.2	Hinweise zum Umgang mit Bändern	3
1.3	Labeln von Bändern	4
1.4	Backup von Festplatte auf Band	5
1.5	Regelmäßiges Backup	7
1.6	Nachholen eines Backups	7
1.7	Durchführung zusätzlicher Backups	8
1.8	Backup von Teilbereichen	9
1.9	Wiederherstellung einzelner Dateien	9
1.10	Wiederherstellung des Systems	12
1.11	Die Konfigurationsdatei für <code>unidump</code>	17
1.12	Sicherungsstrategien	22
1.13	Häufig gestellte Fragen	23
1.14	Kurzreferenz	26
2	Sicherung entfernter Systeme mit Unidump	27
2.1	Einführung	27
2.2	Konfiguration	27
3	Bacula und UCS	29
3.1	Einführung	29
3.2	Installation	30
3.3	Konfiguration	32
3.4	Beispiel-Konfiguration	34
3.5	Administration über die Bacula Console	38
3.6	Sicherung der Catalog-Datenbank	39
3.7	Weiterführende Informationen	40

Achtung:

Die Backup-Software **Bacula** ist seit UCS 2.1 Bestandteil der Distribution und wird mit UCS 3.0 **Unidump** als Backup-Lösung ablösen.



1 Sicherung auf Bandlaufwerken mit Unidump

1.1 Einführung

Mit dem Programm `unidump` können Datensicherungen, so genannte Backups, durchgeführt werden. Bei einem Backup wird eine Sicherheitskopie der Daten auf einem Band angelegt. Sollte ein Datenträger beschädigt sein oder eine Datei versehentlich gelöscht worden sein, so können die Daten von dem Band zurückgesichert werden. Außerdem besteht die Möglichkeit, einzelne Dateien wiederherzustellen, wenn diese z.B. aus Versehen gelöscht wurden. In begrenztem Umfang können auch ältere Versionen einer Datei wiederhergestellt werden (siehe Seite 25).

Bei entsprechender Konfiguration von `unidump` wird das Backup weitgehend automatisch durchgeführt, sodass lediglich das passende Band in das Bandlaufwerk (auch Streamer genannt) eingelegt werden muss. Der Administrator wird per Email benachrichtigt, ob das Backup erfolgreich war.

`unidump` kann so eingerichtet werden, dass das Backup auf einen speziellen Bereich der Festplatte geschrieben wird, wenn es nicht auf Band geschrieben werden kann. In der E-Mail findet sich dann den Hinweis "dump to holdingdisk". Wie das Backup nachträglich von der Festplatte auf ein Band kopiert werden kann, wird in Kapitel 1.4 erklärt.

Neben automatischen Backups in festen Intervallen sind auch Backups "außer der Reihe" und Backups von Datenteilmengen möglich. So können Backups nachgeholt werden, wenn beim automatischen Backup Probleme aufgetreten sind oder z.B. vergessen wurde, das Band zu wechseln.

Am Ende dieser Dokumentation werden häufig gestellte Fragen beantwortet und alle Befehle des Programms `unidump` zusammengefasst.

1.2 Hinweise zum Umgang mit Bändern

Um die gesicherten Daten nicht zu gefährden, sollte mit den Bändern sorgfältig umgegangen werden. Das heißt:

- Bänder dürfen nicht zu oft verwendet werden. In der Regel sollte ein Band durch ein neues ersetzt werden, wenn die vom Hersteller empfohlene Anzahl an Sicherungen damit durchgeführt wurde. Die Hinweise des Herstellers sind zu beachten.
- Die Hinweise des Herstellers hinsichtlich der Lagerbedingungen wie Temperatur und Luftfeuchtigkeit sollten beachtet werden.

- Die Bänder sollten an einem sicheren Ort gelagert werden.
- Es ist empfehlenswert, die Bänder in einem anderen Gebäude zu lagern als die Rechner, von denen Daten gesichert werden.
- Bänder sind nicht für die Langzeitarchivierung konzeptioniert. Werden sie trotzdem dazu verwendet, sollten sie von Zeit zu Zeit umgespult werden, um Datenverluste zu vermeiden.

1.3 Labeln von Bändern

Damit das Programm `unidump` ein Band verwenden kann, muss das Band unter einer Nummer und einem Namen registriert werden. Das Erzeugen von Bandname und -nummer nennt man "labeln". Der Name sollte auf dem Band notiert werden.

Die Namen der Bänder sind nicht frei wählbar. In Abhängigkeit von der verwendeten Strategie müssen Namenskonventionen eingehalten werden. Die verwendbaren Namen können der Dokumentation des Perl-Moduls **`Unidump::Strategy`** entnommen werden (erreichbar über den Befehl `man Unidump::Strategy`).

Das zu labelnde Band ist in das Bandlaufwerk einzulegen.

Achtung:

Es ist darauf zu achten, dass sich keine Daten, die noch benötigt werden, auf dem Band befinden. Durch das Labeln werden auf dem Band befindliche Daten unbrauchbar! Häufig werden auch die Daten auf anderen Bändern ohne diese Daten unbrauchbar! Leider gibt es keine einfache, allgemein gültige Methode festzustellen, ob und welche Daten sich auf dem Band befinden. Ob ein Band bereits mit Unidump verwendet wurde, kann, wie auf Seite 23 beschrieben, überprüft werden.



Der folgende Befehl schreibt ein Label auf das eingelegte Band:

```
unilabel -force -label <Bandname>
```

Dabei ist **<Bandname>** durch den gewünschten Namen zu ersetzen. Prinzipiell sind alle Namen außer ***gfs***, ***simple*** und ***distrib*** zulässig. Standardmäßig sollten nur die Buchstaben a - z in Groß- und Kleinschreibung, die Ziffern 0 - 9 und der Unterstrich "_" im Namen verwendet werden. Wird z.B. ***gfs*** verwendet, so werden die Bänder nach den Wochentagen (Monday, Tuesday, Wednesday, Thursday und Friday_week1 bis Friday_week5) benannt, damit Unidump sie zuordnen kann. Von Montag bis Donnerstag wird in diesem Fall jeden Tag eine inkrementelle Sicherung erzeugt und jeden Freitag eine Vollsicherung.

Damit die automatische Sicherung funktionieren kann, dürfen die Namen der Bänder nicht verändert werden. Wenn ein Band für die automatische Sicherung durch ein neues ersetzt werden muss, ist darauf zu achten, dass der Name des neuen Bandes exakt dem Namen des alten Bandes entspricht! (Siehe auch "Wie heißt das Band? Wie oft wurde es bereits verwendet?" auf Seite 23.) Steht auf dem alten Band z.B. ***Tuesday***, ist der Befehl

```
unilabel -force -label Tuesday
```

auszuführen.

1.4 Backup von Festplatte auf Band

Unidump kann so eingerichtet werden, dass das Backup auf einen speziellen Bereich der Festplatte geschrieben wird, wenn sich kein passendes Band im Laufwerk befindet. Dieser Bereich wird **Holdindisk** genannt. In der E-Mail über die automatische Durchführung des Backups ist in einem solchen Fall ein Hinweis zu finden, der diesem Beispiel ähnelt:

```
WARNING some dumps went to holdingdisk, flush them to tape soon!

boot   OK --> dump to holdingdisk: /var/lib/unidump/hd/3eb 868d7-02ed.dump
                                             (OK)
system OK --> dump to holdingdisk: /var/lib/unidump/hd/3eb 868dc-02ed.dump
                                             (OK)
home   OK --> dump to holdingdisk: /var/lib/unidump/hd/3eb 86919-02ed.dump
                                             (OK)
```

Ein auf der Holdingdisk gespeichertes Backup sollte möglichst bald auf Band kopiert werden. Dafür steht der Befehl

```
uniflush
```

zur Verfügung. Wenn kein Band im Laufwerk liegt, erscheint daraufhin eine Meldung, die folgender ähnelt:

```
ERROR: Tape not ready;
       please insert one of the following tapes in drive and restart
       tape labeled "Tuesday"
```

Wenn ein Band im Laufwerk liegt, das jetzt nicht benötigt wird, wird eine Meldung der folgenden Art ausgegeben:

```
there is currently no dump to flush on this tape
please insert one of the following tapes in drive and restart
       tape labeled "Tuesday"
```

Dann ist das Band mit dem Namen **Tuesday** in das Bandlaufwerk einzulegen. Anschließend kann der Befehl

```
uniflush
```

erneut ausgeführt werden. Der Rechner zeigt einen erfolgreichen Kopiervorgang folgendermaßen an:

```
flushing dumps to tape labeled "Tuesday"
       flushing dump 3eb86919-02ed on tape "Tuesday" file 1
       flush done, ok
```

Dieser Vorgang sollte so lange wiederholt werden bis die Meldung

```
no dumps to flush
```

zu sehen ist.

Nachdem die Backups erfolgreich kopiert wurden, sollten sie von der Festplatte gelöscht werden, damit gegebenenfalls neue Backups auf der Festplatte Platz finden. Mit dem Befehl

```
unistat | grep "'holdingdisk'"
```

kann der Verzeichnisname der Holdingdisk angezeigt werden. Es erscheint eine Ausgabe der folgenden Art:

```
'holdingdisk' => '/var/lib/unidump/hd'
```

Mit dem folgenden Befehl können die Backup-Dateien entfernt werden:

```
rm -rf <Verzeichnisname>/*.dump
```

<Verzeichnisname> ist dabei durch das angegebene Verzeichnis zu ersetzen. In diesem Beispiel müsste der folgende Befehl ausgeführt werden:

```
rm -rf /var/lib/unidump/hd/*.dump
```

Achtung:

Mit diesem Befehl werden die Dateien sofort und ohne Sicherheitsabfrage gelöscht. Wenn der Befehl falsch oder im falschen Verzeichnis ausgeführt wird, kann im Extremfall der gesamte Verzeichnisbaum gelöscht werden.



Wurde ein Dump von der Holdingdisk gelöscht, ohne auf ein Band gesichert zu werden, gibt `uniflush` beim nächsten Start eine Meldung aus, die der folgenden ähnelt:

```
flushing dumps to tape labeled "testband"
  flushing dump 3fb0f089-1628 on tape "testband" file 1
  flush () FAILED with error 256: Illegal seek
  command was:
dd bs=64k if=/var/lib/unidump/hd/3fb0f089-1628.dump of=/dev/nst0 \
2> /dev/null
```

Der nicht mehr vorhandene Dump kann natürlich auch nicht mehr gesichert werden. Wird erneut ein Dump auf die Holdingdisk geschrieben, wird `uniflush` beim nächsten Aufruf zwar wieder korrekt arbeiten, allerdings sind die Einträge der nicht gesicherten Holdingdisk-Dumps noch in der History-Datei **history.txt** von `unidump` vorhanden.

Um die History-Datei mit dem Stand der Holdingdisk abzugleichen, ist das Skript `holdingproper.sh` wie folgt aufzurufen:

```
/var/lib/unidump/scripts/holdingproper.sh
```

Dadurch werden die überflüssigen Einträge aus der History-Datei entfernt, sofern sie sich nicht mehr auf der Holdingdisk befinden. Dieses Skript sollte sicherheitshalber immer nach dem Löschen von Dumps von der Holdingdisk ausgeführt werden.

1.5 Regelmäßiges Backup

Um eine automatische Datensicherung vorzunehmen, muss auf dem Backup-Server ein Cron-Job eingerichtet werden. Ohne einen Cron-Job muss der Aufruf von Unidump manuell ausgeführt werden.

Ein Cron-Job für Unidump ist einfach zu erstellen. Die Konfiguration wird in der Datei `/etc/unidump.conf` gespeichert, die bei jedem Aufruf ausgewertet wird. Es muss sichergestellt sein, dass Unidump entsprechend der Strategie häufig genug ausgeführt wird, z.B. täglich bei der Strategie *archiv*.

Bei den Cron-Skripten unter `/etc/cron.d/` handelt es sich um Shell-Skripte, die mit weiteren Programmen kombiniert werden können. So können beispielsweise automatisch E-Mails versandt, Statistiken erzeugt oder vor dem Backup Verzeichnisse von temporären Dateien gereinigt werden.

Nachfolgend ein Beispiel für die tägliche Ausführung von Unidump. Die Ausgaben werden als E-Mail an **root** und den Backup-Administrator versandt und nach der Ausführung von Unidump Web-Seiten mit den verschiedenen Auswertungswerkzeugen erstellt. Der komplette Name der Datei lautet `/etc/cron.d/backup`:

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin

# 22:30 Uhr: Backup auf Band
30 22 * * 1-5    root    /usr/bin/unidump | \
/usr/bin/mail -s "Backup-Meldung von unidump" root backup@domain.de
# 03:00 Uhr: Backup-Auswertung erzeugen
00 03 * * *      root    /usr/bin/unisched -n 21 -html \
> /var/www/unidump/unisched.html
05 03 * * *      root    /usr/bin/unilib \
-html > /var/www/unidump/unilib.html
10 03 * * *      root    /usr/bin/unihist \
-html > /var/www/unidump/unihist.html
15 03 * * 1-5    root    /usr/bin/unisched | /usr/bin/mail \
-s "unidump erwartet heute folgendes Band:" backup@domain.de
```

1.6 Nachholen eines Backups

Ein ausgelassenes Backup – z.B. weil das Band nicht gewechselt wurde – kann nachgeholt werden. Der Befehl

```
uniresume -human-readable -date <Datum>
```

zeigt an, welche Backups für das angegebene Datum nachgeholt werden können. Ist kein Datum angegeben, wird automatisch das aktuelle Datum eingesetzt.

Wenn die Meldung der folgenden ähnelt, wobei anstelle von "Mon Apr 28 14:11:28 2007" das angegebene Datum steht, so sind alle Backups wie geplant durchgeführt worden.

```
date of dump to resume: Mon Apr 28 14:11:28 2007
```

Eine Meldung der folgenden Art zeigt dagegen an, dass die Backups der Teilbereiche (siehe Seite 9) **boot**, **system** und **home** für das Datum **2007-04-22** nachgeholt werden können.

```
date of dump to resume: Tue Apr 22 00:00:00 2007
dump: "boot" level 1 tapelabel "Tuesday"
dump: "system" level 1 tapelabel "Tuesday"
dump: "home" level 1 tapelabel "Tuesday"
```

Sollen alle Backups der Liste nachgeholt werden, kann der folgende Befehl verwendet werden:

```
unidump -starttime $(uniresume)
```

Das Programm fordert anschließend nacheinander die benötigten Bänder an.

Soll das Backup für einen bestimmten Tag nachgeholt werden, ist das entsprechende Band für diesen Tag in das Bandlaufwerk einzulegen. Der folgende Befehl startet das nachträgliche Backup:

```
unidump -starttime $(date -d <Datum> +%s)
```

<Datum> ist dabei durch das gewünschte Datum, für das das Backup nachgeholt werden soll, zu ersetzen. Wenn z.B. das Backup für den 22. April 2007 nachgeholt werden soll, ist der folgende Befehl auszuführen:

```
unidump -starttime $(date -d 2007-04-22 +%s)
```

Achtung:

Es wird zwar das Band für den 22. April 2007 erstellt, jedoch die aktuellen Daten auf das Backup geschrieben. Deswegen ist es in der Regel sinnvoller, ein Backup des aktuellen Datums durchzuführen, wie in Kapitel 1.7 beschrieben.



1.7 Durchführung zusätzlicher Backups

Neben den automatischen Backups können auch Backups von Hand durchgeführt werden. Dafür wird ein eigenes Band benötigt, das nicht für automatische Backups verwendet wird.

Das Band ist in das Bandlaufwerk einzulegen und wie in Kapitel 1.3 beschrieben zu labeln.

Das Backup kann anschließend mit dem Befehl

```
unidump -strategy <Bandname>
```

gestartet werden, wobei **<Bandname>** durch den tatsächlichen Bandnamen ersetzt werden muss.

Wenn das Backup bei der Wiederherstellung von Dateien oder dem ganzen System nicht benutzt werden soll, z.B. weil das Band an einem schwer zugänglichen Ort wie einem Banktresor aufbewahrt wird, so sollte der Befehl um die Option `-noregulardump` ergänzt werden.

1.8 Backup von Teilbereichen

Mit dem Befehl

```
unistat | grep ^% | grep -v GLOBAL
```

kann festgestellt werden, aus welchen Teilbereichen, so genannten Disks, das Backup normalerweise besteht. Die Ausgabe ähnelt in etwa der folgenden:

```
%boot = (  
%home = (  
%system = (
```

In dem Beispiel stehen also die Disks **boot**, **home** und **system** zur Wahl.

Soll nur eine bestimmte Disk gesichert werden, ist diese im Befehl mit anzugeben:

```
unidump -strategy <Bandname> <Diskname>
```

Beispiel:

Um den Teilbereich **home** auf dem Band **archiv_home** zu sichern und das Band bei einer Wiederherstellung nicht zu benötigen (dafür dient die Option `-noregulardump`), kann der Befehl:

```
unidump -noregulardump -strategy archiv_home home
```

verwendet werden.

1.9 Wiederherstellung einzelner Dateien

Die Wiederherstellung einzelner Dateien erfolgt in zwei Schritten: Erst wird das Backup gesucht, auf dem sich die entsprechende Datei befindet und danach die Datei wiederhergestellt.

1.9.1 Dateien suchen

Der Befehl `unilist` gibt eine Liste der gesicherten Dateien aus. Dabei muss das gewünschte Verzeichnis angegeben werden, in dem sich die gesuchte Datei befindet. Die zur Wahl stehenden Verzeichnisse können mit dem folgenden Befehl ausgegeben werden:

```
unistat | grep directory
```

Die Bildschirmausgabe ähnelt in etwa der folgenden:

```
'directory' => '/boot'  
'directory' => '/home'  
'directory' => '/',
```

In diesem Beispiel gibt es also die Verzeichnisse **/boot**, **/home** und **/**.

Standardmäßig zeigt `unilist` die Liste des letzten Backups. Soll ein älterer Zustand wiederhergestellt werden (siehe Kapitel 1.13.5), ist das gewünschte Datum mit anzugeben. Werden z.B. Daten aus dem Verzeichnis `/home` vom 20. März 2007 benötigt, kann der folgende Befehl verwendet werden:

```
unilist -dir /home -date 2007-03-20
```

Um die Liste weiterverarbeiten zu können, empfiehlt es sich, sie in einer Datei abzuspeichern:

```
unilist -dir <Verzeichnis> -date <Datum> > <Dateiname>
```

<Verzeichnis> ist durch das gewünschte Verzeichnis- und **<Dateiname>** durch den gewünschten Dateinamen, mit Pfadangabe, zu ersetzen. Falls gewünscht kann zusätzlich ein Datum angegeben werden. Soll z.B. die Liste für das letzte Backup des Verzeichnisses `/home` in der Datei `/tmp/liste` gespeichert werden, so ist der folgende Befehl auszuführen:

```
unilist -dir /home > /tmp/liste
```

Diese Listen-Datei kann anschließend mit einem Editor bearbeitet werden. Mit der Suchfunktion können die Dateien in der Liste ausgemacht werden, die wiederhergestellt werden sollen. Alle anderen Zeilen sind aus der Datei zu entfernen.

Soll das gesamte Verzeichnis wiederhergestellt werden, kann die Listen-Datei unverändert bleiben.

1.9.2 Dateien wiederherstellen

Mit dem Befehl

```
unirest <Listen-Datei>
```

werden die Dateien wiederhergestellt, **<Listen-Datei>** ist dabei durch den Pfad und Dateinamen der Listen-Datei zu ersetzen. In diesem Beispiel würde also der folgende Befehl ausgeführt:

```
unirest /tmp/liste
```

Anschließend erscheint eine Meldung, die der folgenden ähnelt:

```
To restore I will need the following dump(s) in that order:
dump 3c9610b5-1281 on tape DEBDED72-1DD1-11B2-A37F-D46F35FFE6B9
label "archiv_home"
the tape in drive is: 3ca41ab8-0959, label "Wednesday"
please insert now tape DEBDED72-1DD1-11B2-A37F-D46F35FFE6B9
label "archiv_home" and restart with
unirest -sessionfile /tmp/fileNRRXEH
```

Die Zeile **"*please insert now tape ... label '<Bandname>'...*"** gibt an, welches Band in das Bandlaufwerk eingelegt werden muss. Der Bandname sollte auf dem Band notiert sein. (Siehe auch "Wie heißt das Band? Wie oft wurde es bereits verwendet?" auf Seite [23](#).)

Die Zeile **"*restart with...*"** zeigt, mit welchem Befehl die eigentliche Rücksicherung gestartet werden kann. In diesem Beispiel ist das Band ***archiv_home*** einzulegen und anschließend der folgende Befehl auszuführen:

```
unirest -sessionfile /tmp/fileNRRXEH
```

Wenn der Computer die Dateien von dem Band kopiert hat und ein anderes Band benötigt, erscheint wieder eine Meldung wie oben. Dann ist das angeforderte Band in das Laufwerk einzulegen und der genannte Befehl auszuführen.

Mit dem Befehl

```
unistat | grep restoredir
```

kann festgestellt werden, in welchem Verzeichnis der Festplatte die wiederhergestellten Dateien liegen. Die Ausgabe ähnelt in etwa der folgenden:

```
'restoredir' => '/tmp/unirestore',
```

In diesem Beispiel liegen die wiederhergestellten Dateien also in dem Verzeichnis [/tmp/unirestore](#). Die Dateien werden also nicht sofort an ihren ursprünglichen Ort im Dateisystem zurückgeschrieben. Das hat den Vorteil, dass keine Dateien überschrieben werden, die möglicherweise noch vorhanden sind. Die Dateien in dem Wiederherstellungs-Verzeichnis können in Ruhe kontrolliert und nach Wunsch in beliebige Verzeichnisse kopiert beziehungsweise verschoben werden.

Um Speicherplatz beziehungsweise Inodes zu sparen, können die Dateien bei der Wiederherstellung in eine Archivdatei geschrieben werden. Dafür stehen die Programme `zip`, `tar` und `star` zur Verfügung. Mit den Befehlen

```
unirest -archiv zip
unirest -archiv tar
```

beziehungsweise

```
unirest -archiv star
```

kann die Rücksicherung in eine Archivdatei gestartet werden. Das Programm `star` speichert dabei zusätzlich Access Control Lists (ACLs).

Natürlich können Standard-GNU/Linux-Verfahren wie reguläre Ausdrücke, pipes etc. verwendet werden. Einsatzbeispiele sind

```
unilist -dir /home -date 2002-03-20 "^./rrobert/.*\.ps"
```

um nur bestimmte wiederherstellbare Dateien auflisten zu lassen, oder

```
unilist -dir /home | grep home/meier | grep .sxw | unirest
```

um die OpenOffice.org Writer-Dokumente der Mitarbeiterin Meier zu restaurieren, oder

```
unilist -dir /home | grep meier/texte/lebenslauf.txt | \ unirest
```

um die Datei(en) `meier/texte/lebenslauf.txt` zu restaurieren. Hierunter fallen auch Dateien wie `hans/meier/texte/lebenslauf.txt` oder `obermeier/texte/lebenslauf.txt`.

1.10 Wiederherstellung des Systems

Achtung:

Soll das System auf eine bereits verwendete Festplatte wiederhergestellt werden, ist unbedingt sicherzustellen, dass sich keine Daten mehr auf dieser befinden. In der Regel wird ein Backup so eingerichtet, dass es nur die wichtigsten Daten des Systems sichert. Außerdem ist nie auszuschließen, dass ein Band unerwartet nicht funktioniert.



Soll das System dagegen auf einer neuen Festplatte wiederhergestellt werden, kann bei der im Folgenden beschriebenen Vorgehensweise kein weiterer Schaden angerichtet werden.

Die vollständige Wiederherstellung eines Systems – auch Recovery genannt – läuft in folgenden Schritten ab:

1. Booten des Systems von der aktuellen UCS-Recovery-CD.
2. Als Bootvariante sollte die Variante mit der zum System passenden Kernel-Version gewählt werden.

Nach dem Bootvorgang des Recovery-Systems startet die Hardwareerkennung und listet die zu ladenden Module auf. Falls sich das Modul **st** (**SCSI tape**) nicht in dieser Liste befindet, sollte es über die Schaltfläche **[Modul hinzufügen]** hinzugefügt werden.

Durch Drücken der Taste **F12** oder der Schaltfläche **[F12-Recovery Shell starten]** werden die aufgelisteten Module in den Kernel geladen und anschließend eine Shell gestartet.

3. Der folgende Befehl wechselt das aktuelle Arbeitsverzeichnis nach `/tmp`.

```
cd /tmp
```

4. Das Band, auf das das letzte erfolgreiche Backup geschrieben wurde, ist in das Bandlaufwerk einzulegen. Von dem Band werden Restore-Skripte und eine History-Datei in das aktuelle Verzeichnis kopiert. Dazu wird das Skript `unihrest` verwendet

```
/usr/bin/unihrest -all -f <Bandlaufwerk>
```

In der Regel lautet die Angabe für das Bandlaufwerk `/dev/nst0` und der Befehl damit

```
/usr/bin/unihrest -all -f /dev/nst0
```

5. Zur Wiederherstellung des Root-Dateisystems wird das Skript `restore_root.sh` ausgeführt. Dieses fragt interaktiv ab, ob die Festplatte, auf der sich die Root-Partition befand, neu partitioniert werden soll.

Achtung:

Alle Daten, die sich möglicherweise noch auf der Festplatte befinden, gehen durch die



Partitionierung verloren! Die folgenden Schritte sollten nur ausgeführt werden, wenn sichergestellt ist, dass sich keine benötigten Daten mehr auf der Festplatte befinden.

Die Partitionierung kann automatisch erfolgen. Falls eine andere Partitionierung gewählt werden soll oder die neue Festplatte eine andere Geometrie als die Originalplatte hat, muss die Partitionierung von Hand mit `cfdisk` oder `fdisk` erfolgen.

Der Befehl

```
./restore_root.sh
```

startet das Restore-Skript. Es folgt ein Dialog, der dem hier abgedruckten ähnelt:

```
Do you want to rebuild the root-filesystem (the fs for '/')? [yN] >y
Do you want to write a new partition table on disk /dev/sda? [yN] >y
Do you want to create a new filesystem on partition /dev/sda2? [N] >y
+++++
+++ WARNING +++ WARNING +++ WARNING +++ WARNING +++ WARNING +++
+++++

THIS WILL DESTROY ALL YOUR DATA ON DISK /dev/sda

PLEASE ABORT WITH ^C IF YOU DO NOT WANT TO DO THAT

+++++
+++ WARNING +++ WARNING +++ WARNING +++ WARNING +++ WARNING +++
+++++
Do you want to proceed and destroy all your data on /dev/sda? [yN] >y
this partition table will be written to /dev/sda
unit: sectors

/dev/sda1 : start=      63, size=   32067, Id=83, bootable
/dev/sda2 : start=   32130, size=16209585, Id=83
/dev/sda3 : start=16241715, size= 1606500, Id=82
/dev/sda4 : start=      0, size=      0, Id= 0
Do you want to write this partition table to disk /dev/sda? [yN] >y
+++++
+++ WARNING +++ WARNING +++ WARNING +++ WARNING +++ WARNING +++
+++++

THIS WILL DESTROY ALL YOUR DATA ON PARTITION /dev/sda2

PLEASE ABORT WITH ^C IF YOU DO NOT WANT TO DO THAT

+++++
+++ WARNING +++ WARNING +++ WARNING +++ WARNING +++ WARNING +++
+++++

Do you want to proceed and destroy all data on /dev/sda2? [yN] >y

Do you want to mount your target filesystem /dev/sda2 on /mnt
(necessary to restore data)? [yN] >y

Do you want to restore the data of your root-filesystem? [yN] >y
```

```

please insert tape 3f1fc718-3efd (label Friday) and press ENTER
Do you want to create a new filesystem on partition /dev/sda1? [yN]>y
+++++
+++ WARNING +++ WARNING +++ WARNING +++ WARNING +++ WARNING +++
+++++

THIS WILL DESTROY ALL YOUR DATA ON PARTITION /dev/sda1

PLEASE ABORT WITH ^C IF YOU DO NOT WANT TO DO THAT

+++++
+++ WARNING +++ WARNING +++ WARNING +++ WARNING +++ WARNING +++
+++++
Do you want to mount your target filesystem /dev/sda1 on /mnt/boot
(necessary to restore data)? [yN] >y

Do you want to restore the data of your boot-filesystem? [yN] >y
please insert tape 3f1fc718-3efd (label Friday) and press ENTER
Do you want to make the target filesystem bootable? [yN] >y

```

Das Skript partitioniert die Platte, erzeugt ein Dateisystem auf der Root-Partition (d.h. es formatiert sie) und kopiert die gesicherten Daten der Root-Partition vom Band auf das gerade erzeugte Dateisystem. Sollte das Verzeichnis `/boot` auf einer eigenen Partition liegen, so wird diese Partition ebenfalls von diesem Skript wiederhergestellt. Anschließend werden die noch benötigten Skripte in das neue Root-Dateisystem kopiert und die Partition, auf der sich das Verzeichnis `/boot` befindet, als bootbar markiert.

6. Nun ist die CD aus dem Laufwerk zu entfernen und der Rechner neu zu starten. Der Rechner sollte jetzt von der wiederhergestellten Root-Partition in den Single-User-Mode booten. Dort kann sich lediglich der Benutzer **root** anmelden. Nach der Anmeldung sollte mit dem Befehl

```
cd /
```

in das Wurzelverzeichnis gewechselt werden. Dort liegen die nun benötigten Skripte.

7. Falls mehrere Platten im System eingebaut sind (und waren), können diese mit dem Skript `part_nonroot.sh` neu partitioniert werden.

Der Befehl

```
./part_nonroot.sh
```

startet das entsprechende Skript. Es folgt ein Dialog, der dem hier abgedruckten ähnelt:

```

Do you want to write a new partition table on disk /dev/sdb? [yN] >y
+++++
+++ WARNING +++ WARNING +++ WARNING +++ WARNING +++ WARNING +++
+++++

THIS WILL DESTROY ALL YOUR DATA ON DISK /dev/sdb

PLEASE ABORT WITH ^C IF YOU DO NOT WANT TO DO THAT

+++++
+++ WARNING +++ WARNING +++ WARNING +++ WARNING +++ WARNING +++

```

```
+++++
Do you want to proceed and destroy all your data on /dev/sdb? [yN] >y
this partition table will be written to /dev/sdb
unit: sectors

/dev/sdb1 : start=      63, size=35840952, Id=83
/dev/sdb2 : start=      0, size=      0, Id= 0
/dev/sdb3 : start=      0, size=      0, Id= 0
/dev/sdb4 : start=      0, size=      0, Id= 0
Do you want to write this partition table to disk /dev/sdb? [yN] >y
```

8. Das Logical-Volume-Management wird mit dem Skript `restore_lvm.sh` wiederhergestellt.

```
./restore_lvm.sh
```

9. Die Swap-Partition(en) wird (werden) mit dem Skript `restore_swap.sh` wiederhergestellt.

```
./restore_swap.sh
```

Es folgt eine Benutzerabfrage, die der hier abgedruckten ähnelt:

```
Do you want to create a swap partition on /dev/sdb3? [yN] >y
```

10. Nun können die restlichen Daten wiederhergestellt werden, wozu das Skript `restore_nonroot.sh` dient.

```
./restore_nonroot.sh
```

Zuerst durchläuft das Skript in einer Schleife alle Partitionen und erfragt, ob ein neues Dateisystem auf die Partition geschrieben werden soll.

Beispiel:

```
Do you want to write a new filesystem on /dev/sda4 (was /extra)?
                                                    [yN] >y
Do you want to write a new filesystem on /dev/sdb1 (was /home)?
                                                    [yN] >y
```

Nun werden alle Verzeichnisse, welche auf dem ursprünglichen System als eigene Partitionen vorhanden waren, in einer Schleife durchlaufen.

Nun wird abgefragt, ob das betreffende Verzeichnis wiederhergestellt werden soll. Wird diese Frage mit **y** beantwortet, so erfragt das Skript, ob das Verzeichnis in den Verzeichnisbaum eingehängt (gemountet) werden soll.

Im nächsten Schritt erfragt das Skript, ob das Verzeichnis wiederhergestellt werden soll. Hier ist zu beachten, dass das Verzeichnis auf jeden Fall gemountet sein muss, damit die Daten zurückgeschrieben werden können.

Wird diese Frage mit **y** beantwortet, erscheint eine Aufforderung, das betreffende Band einzulegen.

Ist das Verzeichnis wiederhergestellt, so erfragt das Skript im letzten Schritt, ob es wieder ausgehängt ("Do you want to umount...") werden soll. Im Zweifelsfall ist diese Frage mit **n** zu beantworten, da in bestimmten Fällen die eingehängten Verzeichnisse (z.B. `/var`) zu einem späteren Zeitpunkt von dem Skript benötigt werden.

Beispiel:

```
Do you want to restore /home? [yN] >y

Do you want to mount /home? [yN] >y
please insert tape 3f1fc718-3efd (label Friday) and press ENTER
Do you want to umount /home? [yN] >n

Do you want to restore /extra? [yN] >y

Do you want to mount /extra? [yN] >y
please insert tape 3f1fc718-3efd (label Friday) and press ENTER
Do you want to umount /extra? [yN] >n
History zurückkopiert.
```

Sind alle Verzeichnisse wiederhergestellt, so kopiert das Skript auch die History-Datei zurück in das Unidump-Verzeichnis.

Hinweis:

Falls nicht das standardmäßige Unidump-Verzeichnis verwendet wird, kann mit dem Befehl `unistat | grep unidir` der Pfad zum Verzeichnis festgestellt werden..

11. Der Rechner kann nun neu gestartet werden.

Infobox Recovery ohne UCS-CD

Sollte z.Z. keine UCS-Installations-CD zur Verfügung stehen, kann das System auch mit einer üblichen Linux-Rettungs-Diskette/CD gebootet werden. Bei ausreichender Erfahrung kann das System mit den folgenden Hinweisen wiederhergestellt werden:

Falls `unihrest` nicht zur Verfügung steht, können die Dateien auch von Hand zurückgespielt werden. Sie befinden sich in einem nicht-komprimierten GNU-tar-Archiv am Ende jedes Unidump-Bandes. Man findet dieses Archiv wie folgt:

```
mt -f <Bandlaufwerk> eod
mt -f <Bandlaufwerk> status
```

```
SCSI 2 tape drive:
File number=3, block number=0, partition=0.
Tape block size 0 bytes. Density code 0x13 (DDS (61000 bpi)).
Soft error count since last status=0
General status bits on (89010000):
EOF EOD ONLINE IM_REP_EN
```

Das letzte Kommando liefert bei End-Of-Data **File number=3**, das heißt, es gibt zwei Dateien auf dem Band. Damit ist die gesuchte Datei die zweite auf dem Band.

```
mt -f <Bandlaufwerk> asf 2
tar -x -v -f <Bandlaufwerk>
```

Gegebenenfalls muss noch eine Blocksize angegeben werden. Falls diese unbekannt ist, kann sie wie folgt ermittelt werden:

```
mt -f <Bandlaufwerk> asf 2
dd if=<Bandlaufwerk> of=testfile bs=256k count=1
ls -l testfile
```

Die Größe des testfile entspricht der Blocksize. Das Verfahren beruht darauf, dass die Blocksize **sicher** kleiner als 256k ist. Ergibt sich eine Blocksize von z.B. 10240, so wird dieser Wert noch durch 512 geteilt und dann als Option an `tar` übergeben.

```
mt -f <Bandlaufwerk> asf 2
tar -x -v -f <Bandlaufwerk> -b <Blocksize>
```

1.11 Die Konfigurationsdatei für `unidump`

Es folgen die technischen Details zur Konfigurationsdatei von `unidump`. Die `unidump`-Konfigurationsdatei liegt standardmäßig unter `/etc/unidump.conf`.

1.11.1 Der globale Teil der Konfigurationsdatei

Der globale Teil der Konfigurationsdatei beinhaltet die Parameter, welche `unidump` immer verwendet, unabhängig von der zu sichernden Partition.

```
unidir /var/lib/unidump
```

unidir gibt den Pfad zu den `unidump`-Administrationsdateien an.

```
ntapedevice /dev/nst0
```

ntapedevice gibt das zu verwendete Tapedevice an. Es muss das nonrewinding Device angegeben werden. Manche Tapes stellen unterschiedliche Devices mit oder ohne Hardware-Komprimierung zur Verfügung. In diesem Fall wird, falls Hardware-Komprimierung gewünscht ist, das entsprechende Device angegeben. Der Parameter **hwcompression** wird dann auf **0** gesetzt. Dies trifft z.B. auf Floppy-Tapes zu (Tapedevice mit Hardware-Komprimierung ist z.B. `/dev/nzqft0`). SCSI-DAT-Laufwerke stellen keine speziellen komprimierenden Devices zur Verfügung. Bei diesen muss stattdessen der Parameter **hwcompression** auf **1** gesetzt werden.

```
magicfile = /var/lib/unidump/magic
```

magicfile ist eine Textdatei, welche dem System hilft, bestimmte Dateien zu Programmen zuzuordnen. Nähere Informationen dazu bietet die betreffende man page mit

```
man magic
```

```
logfile = /var/lib/unidump/unidump.log
```

logfile ist die globale Logdatei, welche von `unidump` verwendet wird, sofern der Parameter **useunilog** auf **1** gesetzt ist.

```
logfiledir = /var/lib/unidump/log
```

logfiledir definiert das Unterverzeichnis, in welchem die einzelnen Logdateien der Backups gespeichert werden. In diesen Logdateien sind Informationen über die einzelnen Backups zu finden.

```
restoredir = /tmp/unirestore
```

restoredir gibt das Verzeichnis an, in das bei einer Wiederherstellung die Daten geschrieben werden. Generell werden nicht wie bei anderen Systemen die bestehenden Daten überschrieben, sondern die Daten vom Band werden in das unter **restoredir** angegebene Verzeichnis geschrieben und können von dort durch den Administrator an ihren eigentlichen Platz zurückgespeichert werden.

```
strategy = archiv
```

strategy spezifiziert die Backup-Strategie. Weitere Informationen können über den Befehl

```
man unidump::strategy
```

und aus Kapitel 1.12 bezogen werden.

```
holdingdisk = /var/holdingdisk
```

holdingdisk gibt das Verzeichnis an, in das gesichert wird, wenn eine Bandsicherung nicht möglich ist.

Achtung:

Bei der Konfiguration von `unidump` muss darauf geachtet werden, dass die Holdingdisk nicht auf ein Netzlaufwerk und nicht in einem der Verzeichnisse liegt, die gesichert werden. Letzteres würde dazu führen, dass die Partition, auf der die Holdingdisk angelegt ist, nach wenigen Durchläufen von `unidump` restlos gefüllt wird.

```
holdingdisksize = 1G
```

holdingdisksize gibt die maximale Größe an, die die Holdingdisk auf der Festplatte einnehmen darf. Steht dieser Parameter auf **0**, so ist die Holdingdisk 0 Byte groß, eine Sicherung auf der Holdingdisk ist also nicht möglich. Gibt man den Parameter mit vorangestelltem Minus-Zeichen (-) an, z.B. **-100M**, so wird der gesamte freie Platz auf der Partition bis auf den angegebenen Wert verwendet.

Beispiel:

Die Holdingdisksize ist 1 G. Die Partition, auf der die Holdingdisk liegt, ist 10 G groß. 9,5 G der Partition sind belegt. Wenn jetzt ein 750 MB großer Dump auf die Holdingdisk geschrieben werden soll, wird die Partition komplett gefüllt. Dies kann zu Problemen führen, wenn die Partition von anderen Programmen benötigt wird. Sollen Probleme dieser Art ausgeschlossen und außerdem sichergestellt werden, dass für die Holdingdisk immer genug Platz zur Verfügung steht, sollte eine eigene Partition für die Holdingdisk angelegt werden.

```
blocksize = 64k
```

blocksize spezifiziert die zu verwendenden Blockgröße. Manche Tapes benötigen bestimmte Blockgrößen, z.B. arbeitet der `zftape`-Treiber standardmäßig nur mit Vielfachen von 10k. DAT-Laufwerke arbeiten in der Regel mit beliebigen Blockgrößen, jedoch verbessern große Blöcke



die Performance. Die Blockgröße darf aber zumindest bei der Verwendung von `dump` maximal 64k betragen. Weitere Informationen bietet die man page von `dump`:

```
man dump
```

```
hwcompression = 1
```

Dieser Parameter schaltet die Hardware-Komprimierung bei SCSI-Tapes ein. Er kann auf die Werte **0** oder **1** gesetzt werden.

Achtung:

Bei einigen Bandlaufwerken führt das Aktivieren der Hardware-Komprimierung in der `unidump`-Konfiguration zu Problemen. Dort ist die Hardware-Komprimierung evtl. per `mt` mit der Option `datcompression` (1 oder 2) zu aktivieren. In diesem Fall muss die Softwarekomprimierung jedoch deaktiviert werden.

```
softcompression = 1
```

Dieser Parameter schaltet die Software-Komprimierung ein. Wie diese realisiert wird, ist vom verwendeten Backend abhängig. `GNU-tar`, `Star` und `dump` komprimieren intern (mit Hilfe der **libz** oder `gzip`), bei `xfsdump` muss extern mit `gzip` komprimiert werden. Dies führt zu höheren Zugriffszeiten auf einzelne Dateien eines komprimierten `xfs-Dump`. Dieser Parameter ist auch im Disk-spezifischen Teil zulässig. Der Parameter kann auf die Werte **0** oder **1** gesetzt werden.

```
mt = mt  
dd = dd
```

Hier kann für die jeweiligen Programme ein anderes gewählt werden. In der Regel sollten hier keine Veränderungen vorgenommen werden.

```
ext2dump = /var/lib/unidump/scripts/dump_wrapper
```

ext2dump gibt das verwendete Sicherungs-Programm an, das bei `ext2/ext3`-Partitionen benutzt wird.

```
ext2restore = /var/lib/unidump/scripts/restore_wrapper
```

ext2restore gibt das verwendete Recovery-Programm an, das bei `ext2/ext3`-Partitionen benutzt wird.

```
xfsdump = xfsdump
```

xfsdump definiert das Sicherungsprogramm für XFS-Partitionen.

```
xfsrestore = xfsrestore
```

xfsrestore definiert das Wiederherstellungsprogramm für XFS-Partitionen.

```
zip = zip  
gtar = tar  
star = star
```



Hier können vom Standard abweichende Komprimierungsprogramme gewählt werden.

Achtung:

Es kann zu Komplikationen bei der Wiederherstellung kommen, wenn bereits Sicherungen durchgeführt wurden und nachträglich das Komprimierungsformat geändert wird.



```
precommand =
```

Hier kann ein Befehl angegeben werden, der unmittelbar vor dem Start eines Backups ausgeführt wird.

```
postcommand =
```

Hier kann ein Befehl angegeben werden, der unmittelbar nach der Durchführung eines Backups ausgeführt wird.

```
eotcommand =
```

Hier kann ein Befehl angegeben werden, der ausgeführt wird, wenn das Ende des Bandes erreicht wird. Dies kann zum Beispiel gewünscht sein, wenn ein Bandwechselsystem eingesetzt wird.

```
writeflags = "Option1 Option2 Option3"
```

Hier können zusätzliche Optionen, die dem Dump-Befehl übergeben werden, angegeben werden. Die Optionen sind in Anführungszeichen einzuschließen und durch Leerzeichen zu trennen.

```
readflags = "Option1 Option2 Option3"
```

Hier können zusätzliche Optionen, die dem Restore-Befehl übergeben werden, angegeben werden. Die Optionen sind in Anführungszeichen einzuschließen und durch Leerzeichen zu trennen.

```
diskmode = 0
```

Wird **diskmode = 1** gesetzt, werden die Backups nicht auf ein Band gesichert, sondern nur auf die Holdingdisk.

```
regulardump = 1
```

Dieser Parameter ist in der Regel in der Konfigurationsdatei auf **1** gesetzt. Um später inkrementelle Backups erstellen zu können, muss das System den Inhalt der absoluten Backups kennen. Aus dem Stand der letzten Vollsicherung und dem aktuellen System wird eine Differenzmenge berechnet. Diese Differenzmenge stellt das zu speichernde inkrementelle Backup dar. Um jedoch absolute Backups außerhalb der geplanten Reihenfolge durchzuführen, kann **regulardump = 0** gesetzt werden. Damit werden die abgespeicherten Systemstände nicht verändert und die regelmäßige Sicherung kann ohne Rücksicht auf die zusätzliche Vollsicherung wie gewohnt weiter erfolgen. Bei darauf folgenden inkrementellen Sicherungen bleibt die zusätzliche Vollsicherung unberücksichtigt. Es ist besser, den Parameter **regulardump = 1** zu belassen und bei einer zusätzlichen Sicherung den Parameter als Kommandozeilenoption mitzugeben, wie in diesem Beispiel:

```
unidump -noregulardump
```

Dieser Parameter ist auch im Disk-spezifischen Teil zulässig.

```
useunilog = 1
```

Protokollierung erfolgt in ein eigenes Logfile (`unidump.log`).

```
logfiledir = /var/lib/unidump/unidump.log
```

Verzeichnis, in dem die Logfiles der einzelnen Backups abgelegt werden.

```
usesyslog = 0
```

Wird hier **1** gewählt, erfolgt die Protokollierung über den syslogd.

```
usestderr = 0
```

Wird hier hier **1** gewählt, werden die (Fehler-)Meldungen nicht in der Logdatei gespeichert, sondern auf der Konsole ausgegeben.

```
debug = 0
```

Der Debug-Modus bewirkt eine ausführlichere Fehlerbeschreibung. Dies ist im Normalbetrieb nicht notwendig, hilft aber in Problemsituationen, Fehler einzugrenzen.

```
useqfa = 0
```

Bewirkt die Erstellung von Quick-File-Access-Dateien, die den Zugriff auf einzelne Dateien des Archivs drastisch beschleunigen. Dieser Parameter ist auch im Disk-spezifischen Teil zulässig. QFA-Dateien werden bisher nur von `dump` unterstützt.

1.11.2 Der Disk-spezifische Teil der Konfigurationsdatei

Im unteren Teil der Konfigurationsdatei werden so genannte Disks spezifiziert. Eine Disk besteht aus einem Verzeichnis, das gesichert werden soll, und den dazugehörigen Spezifikationen. Das Verzeichnis muss ein Mountpoint sein (also eine eigene Partition), wenn `dump` oder `xfsdump` verwendet wird. Jede Disk erhält einen eindeutigen alphanumerischen Namen. Ein Disk-spezifischer Abschnitt in der Konfigurationsdatei beginnt mit dem Namen der Disk in eckigen Klammern. Es folgen Spezifikationen, die jeweils nur für diese Disk gelten. Die einzige notwendige Angabe ist **directory**, alle anderen Angaben sind optional.

```
[diskname]
```

Der frei wählbare Name leitet einen Disk-spezifischen Abschnitt ein. Eine Disk wird damit automatisch definiert. Der Name der Disk ist case-sensitive, die Groß- und Kleinschreibung ist also zu beachten!

```
directory /home
```

Das zu sichernde Verzeichnis, in der Regel ein Mountpoint einer Partition. Der Mountpoint muss in der Datei `/etc/fstab` eingetragen sein. Wenn nur `tar` zur Datensicherung verwendet wird, muss nicht notwendig ein Mountpoint gewählt werden.

```
dumper = xfsdump
```

Hier kann angegeben werden, welches Programm zum Dumpen dieser Disk verwendet werden soll (z.B. `dump` oder `xfsdump`).

```
exclude = /var/lib/unidump/hd
```

Liste von Dateien oder Verzeichnissen, die aus dem Backup ausgenommen werden sollen. Die Pfade werden relativ zum zu sichernden Verzeichnis ausgewertet (bspw. `tmp:usr/tmp:var/tmp:var/lib/unidump/hd`).

Achtung:

Der Parameter **exclude** wird von `xfsdump` ignoriert!

```
writeflags =
```

Zusätzliche Optionen, die dem Dump-Kommando übergeben werden.

```
readflags =
```

Zusätzliche Optionen, die dem Restore-Kommando übergeben werden.

```
precommand =
```

Kommando, das unmittelbar vor dem Start eines Backups ausgeführt wird.

```
postcommand =
```

Kommando, das unmittelbar nach dem Ende eines Backups ausgeführt wird.

1.12 Sicherungsstrategien

`unidump` kann die Datensicherung mit drei verschiedenen Strategien durchführen:

- `strategy = archiv`

Bei dieser Sicherung handelt es sich um die Standardeinstellung. Hierbei wird immer eine Vollsicherung auf das entsprechende Band geschrieben. Es werden im Normalfall fünf Bänder benötigt, welche von Montag bis Freitag gewechselt werden und in der darauf folgenden Woche rotieren und wieder überschrieben werden.

- `strategy = simple`

Diese Strategie führt Montag bis Donnerstag täglich eine inkrementelle Sicherung (level 1 dump) und am Freitag ein Vollsicherung (level 0 dump) durch. Dabei werden die Bänder von Montag bis Donnerstag wöchentlich überschrieben, während die Freitagsbänder monatlich überschrieben werden. Für diese Strategie werden also 9 Bänder (4 Wochenbänder und 5 Freitagsbänder) benötigt.



- strategy = distrib

Sollen mehrere Festplatten oder große Speicherbestände gesichert werden, so kann es sein, dass diese nicht auf ein Band passen. Für diesen Fall kann der Speicherbestand in mehrere Speichergruppen aufgeteilt werden, die dann an verschiedenen Wochentagen eine Vollsicherung erhalten.

Weitere Informationen gibt die man page, welche mit folgendem Befehl aufgerufen werden kann:

```
man unidump::strategy
```

1.13 Häufig gestellte Fragen

1.13.1 Wie heisst das Band?

1. Das entsprechende Band in das Bandlaufwerk einlegen.
2. An der Kommandozeile den folgenden Befehl ausführen:

```
unilabel
```

3. Auf dem Bildschirm erscheint eine Anzeige, die diesem Beispiel ähnelt:

```
tapelabel: Monday
tapeid:    3cadc87d-25ef
tapecycle: 22
```

4. Der Name des Bandes steht hinter **tapelabel**. In dem Beispiel heißt das Band also **Monday**.
5. Wie oft das Band mit `unidump` verwendet wurde, steht hinter **tapecycle**. Das Beispielband wurde also bereits 22 Mal verwendet.

Achtung:

`unilabel` kann nicht anzeigen, ob das Band mit einem anderen Programm als `unidump` verwendet wurde. Außerdem ist es möglich, den Wert für **tapecycle** mit einem Befehl zu ändern. Es kann also nicht ausgeschlossen werden, dass sich Daten auf dem Band befinden, obwohl **tapecycle** den Wert Null angibt.



1.13.2 Welche Bänder mit welchen Bandnamen gibt es?

Der Befehl

```
unilib
```

gibt eine Liste aller bekannten Bänder mit ihren Namen aus.

1.13.3 Welches Band befindet sich im Bandlaufwerk?

Der Befehl

```
unilabel
```

gibt den Name des Bandes, in der erscheinenden Meldung hinter "tapelabel:", aus.

1.13.4 Wurde das Backup korrekt durchgeführt?

Über automatisch durchgeführte Backups erhält der Benutzer **root** eine E-Mail-Nachricht.

Außerdem werden Erfolgs- und Fehlermeldungen in Log-Dateien gespeichert. Abhängig von der Einstellung von Unidump werden die LogDateien des Systems und/oder Unidump-spezifische LogDateien verwendet.

Der Befehl

```
unistat | grep "'logfile'"
```

zeigt mit einer ähnlichen Meldung wie der folgenden an, wo die Unidump-Protokolldatei zu finden ist.

```
'logfile' => '/var/lib/unidump/unidump.log',
```

In dem Beispiel befindet sich die Protokolldatei `unidump.log` also in dem Verzeichnis `/var/lib/unidump/`.

Wenn das Backup korrekt durchgeführt wurde, sind in der Protokolldatei, bei dem entsprechenden Datum, die Schlüsselworte **DUMP ... DONE** oder **DUMP ... DONE (OK)** zu finden.

```
[2007/05/05 17:23:01] DUMP[3eb68195-5509]: xfsdump -F -e -o -l 0 -L \  
3eb68195-5509 -c "/var/lib/unidump/Skripts/exit2" -m -b 65536 -f /dev/nst0  
/home > /var/lib/unidump/log/3eb68195-5509.dumplog  
[2007/05/05 18:58:13] DUMP[3eb68195-5509]: DONE (OK)  
[2007/05/05 19:50:03] DUMP[history_save]: tar -c -C /tmp/filec1e3PS \  
-f /dev/nst0 .  
[2007/05/05 19:50:07] DUMP[history_save]: DONE
```

Wenn Probleme aufgetreten sind, sind Worte wie **DUMP... ABORTED**, **ERROR** oder **FAILURE** zu finden.

Optional kann zur Bandkontrolle in der Konfigurationsdatei `/etc/unidump.conf` der Parameter **verify = 1** gesetzt werden. Dabei wird nach dem Backup der Datenbestand auf dem Band mit dem Datenbestand auf der Festplatte verglichen.

1.13.5 Wie lange können Dateien wiederhergestellt werden?

Die Bänder werden beim Backup überschrieben. Alte Versionen einer Datei können also so lange wiederhergestellt werden, wie das entsprechende Sicherungsband nicht wieder für ein Backup eingesetzt wurde. Wie schnell ein Band erneut eingesetzt wird, hängt von der Sicherungsstrategie ab. Wird z.B. die Strategie *simple* verwendet, wird in der Nacht von Freitag auf Samstag ein Backup des gesamten Systems erstellt und an den anderen Werktagen werden die Dateien gesichert, die sich seit Freitag verändert haben. So kann der Zustand jedes Werktages wiederhergestellt werden. Normalerweise werden die Bänder in der nächsten Woche wieder eingesetzt und mit dem neuen Zustand überschrieben. Nur das Band vom Freitag kommt erst einen Monat später wieder zum Einsatz, so dass der Zustand von jedem Freitag einen Monat lang wiederhergestellt werden kann.

Soll ein bestimmter Zustand dauerhaft gesichert werden, bietet es sich an, ein Backup außer der Reihe (siehe Kapitel 1.7) zu erzeugen. Bänder sind allerdings grundsätzlich nicht zur Dauersicherung gedacht. Sollen Daten über Monate oder gar Jahre aufbewahrt werden, empfehlen sich andere Speichermedien wie CD-ROMs oder DVDs.

1.13.6 Welche Backups wurden wann mit welchem Band erstellt?

Mit dem Befehl

```
unihist
```

kann eine Liste der bisher durchgeführten Backups ausgegeben werden. Aus der Liste geht hervor, welche Teilbereiche wann auf welches Band gesichert wurden. Es ist zu beachten, dass die Liste nach Verzeichnissen und Teilbereichen sortiert ist, erst dann nach Datum.

1.13.7 Wo liegt die Konfigurationsdatei?

Der Befehl

```
unistat | grep config
```

zeigt mit einer ähnlichen Meldung wie der folgenden an, wo die Konfigurationsdatei zu finden ist.

```
here is your unidump configuration
'config' => '/etc/unidump.conf',
```

Standardmäßig liegt sie in `/etc/unidump.conf`.

1.13.8 Wo finde ich weitere Informationen?

Für `unidump` und die zugehörigen Befehle gibt es so genannte man pages (von englisch manual pages = Handbuch-Seiten). man pages sind als Dateien auf dem Computer gespeichert

und werden in der Regel mit dem Befehl

```
man <Befehlsname>
```

aufgerufen. Zu `unidump` gibt es einige Übersichtsseiten, die durch Eingabe von

```
man unidump::config
man unidump::history
man unidump::logger
man unidump::strategy
man unidump::units
```

aufgerufen werden können.

1.14 Kurzreferenz

Hinweis:

`/var/lib/unidump` ist das Standardverzeichnis. In `/etc/unidump.conf` kann ein anderes Verzeichnis als Standardverzeichnis festgelegt werden.

unidump [-strategy <Bandname> oder gfs oder simple oder distrib] [-noregulardump] [-diskmode] [-level <Level 0-9>] [-starttime <Zeit in Sekunden seit 1.1.1970> oder \$(date -d yyyy-mm-dd +%s) oder \$(uniresume)] [<Diskname>] führt das Backup durch.

uniflush kopiert Backups von der Holdingdisk auf Band.

unihist [-list|box|html] gibt die Liste bisher durchgeführter Backups geordnet nach Disks und Datum (als Liste, Tabelle, html-Seite) aus, die in `/var/lib/unidump/history.txt` gespeichert ist.

unihrest -f <Bandlaufwerk> stellt beim Wiederherstellen des Systems die History-Datei wieder her.

unilabel [-label <Labelstring>] [-force] [-cycle <Rundenstempel>] [-id <ID-String>] gibt Labelinformationen aus und labelt Bänder. Das Label sollte der Strategie entsprechend gewählt werden. Rundenstempel und Tape-ID werden automatisch erzeugt. Die Tape-ID soll das Band eindeutig identifizieren, darf also nicht für andere Bänder wiederverwendet werden, auch wenn diese dasselbe Label bekommen!

unilib zeigt die Liste bisher gelabelter Bänder an, die in `/var/lib/unidump/tapelib.txt` gespeichert ist.

unilist -dir <Verzeichnis> [-date <yyyy-mm-dd>] gibt eine Liste der Dateien, die wiederhergestellt werden können, aus.

uniproper [-dryrun] [-force] entfernt überflüssige Einträge aus Holdingdisk, History und den Verzeichnissen der Protokoll-, toc- und qfa-Dateien.

unirest [-sessionfile <Dateiname>] [-archiv <tar|gtar|star|tgz|zip>] [<Listen-Datei>] stellt Dateien (aus der Listen-Datei) wieder her (und packt sie mit dem angegebenen Archivierungsprogramm).

uniresume [-human-readable] [-date <yyyy-mm-dd>] gibt (in menschenlesbarer Form) an, welche Backups heute, oder am angegebenen Datum, nachgeholt werden können.

unisched [-n <Anzahl der Tage>] [-list|box|html] gibt eine Liste der geplanten Backups aus, die an den nächsten <Anzahl> Tagen ausgeführt werden sollen. Ist die Anzahl negativ, liegen die Tage in der Vergangenheit.

unistat [-config] [-holdingdisk] zeigt Informationen über die Konfiguration und die Holdingdisk an.

unitoc erzeugt die Table-of-Contents-Dateien neu, die auch im Verzeichnis `/var/lib/unidump/toc` gespeichert sind.

holdingproper.sh vergleicht den Inhalt der History mit den Inhalten der Holdingdisk und löscht überflüssige Einträge.

2 Sicherung entfernter Systeme mit Unidump

2.1 Einführung

Mit der Software **univention-remote-backup** besteht die Möglichkeit, regelmäßig Verzeichnisse von im Netzwerk erreichbaren Systemen auf ein Sicherungssystem zu übertragen. Verfügt dieses Sicherungssystem über ein Bandlaufwerk, kann wie in Abschnitt 1 beschrieben **unidump** eingesetzt werden, um die Verzeichnisse auf Bänder zu sichern. Bei ausreichend großen Festplatten können die Sicherungskopien alternativ zur Sicherung auf Bändern im Dateisystem verbleiben.

Über eine Konfigurationsdatei kann angegeben werden, welche Verzeichnisse von welchen Rechnern übertragen werden sollen. Entfernte Systeme werden nach Server und Client Systemen unterschieden. Dabei geht **univention-remote-backup** davon aus, dass Client Systeme nicht immer erreichbar sein müssen.

Bei der Sicherung von Verzeichnissen werden nur Unterverzeichnisse und Dateien aus dem angegebenen Verzeichnis berücksichtigt. Über das Netzwerk eingebundene Freigaben oder symbolische Links werden nicht mitgesichert.

2.2 Konfiguration

In der Datei `/etc/remote-backup.conf` können globale Konfigurationsparameter für `univention-remote-backup` aufgenommen werden. Die Syntax der Optionen lautet

```
OPTION=WERT
```

Folgende Parameter können gesetzt werden:

BACKUPROOT=<PFAD> Das lokale Verzeichnis, in welchem die einzelnen Sicherungen abgelegt werden. Die Anzahl der Sicherungen ist vom Parameter MAXHISTORY abhängig, als Name der Sicherungsverzeichnisse wird eine Ordnungszahl verwendet. 0 ist der aktuelle Sicherungsstand, höhere Zahlen sind in aufsteigender Folge die vorigen Sicherungszustände.

MAXHISTORY=<ANZAHL> Die Anzahl der aufzubewahrenden Sicherungszustände. `univention-remote-backup` rotiert die Sicherungszustände, die älteste Version wird gelöscht. Soll nur die aktuelle Version vorgehalten werden, etwa weil das Verzeichnis BACKUPROOT regelmäßig auf ein Band gesichert wird, sollte MAXHISTORY auf 0 gesetzt werden.

Entfernte Rechner und zu sichernde Verzeichnisse müssen je nach Funktion des Rechners in eine von zwei Konfigurationsdateien eingetragen werden. In `/etc/remote-backup.servers` werden Server-Systeme eingetragen. Dabei wird angenommen, dass ein Server-System immer erreichbar sein muss. Ist das System nicht erreichbar, wird eine Fehlermeldung ausgegeben. In `/etc/remote-backup.clients` werden Client-Systeme aufgenommen. Client-Systeme können ausgeschaltet sein, sie werden dann bei der Sicherung übersprungen. Die Syntax beider Dateien ist identisch, sie lautet:

```
<Rechnername>:<Pfad1>,<Pfad2>,<Pfad3>...
```

Standardmäßig wird der Benutzer **root** verwendet, um die Verzeichnisse auf dem entfernten System zu lesen. Soll ein alternativer Benutzer verwendet werden, ist dieser in der folgenden Schreibweise in die Datei `/etc/remote-backup.servers` oder `/etc/remote-backup.clients` einzutragen:

```
<Benutzername>@<Rechnername>:<Pfad1>,<Pfad2>,<Pfad3>...
```

`univention-remote-backup` verwendet `rsync` zur Übertragung von Verzeichnissen. Daraus ergibt sich, dass `univention-remote-backup` eine Benutzeridentität verwenden muss, die auf dem entfernten System Leseberechtigungen für die ausgewählten Verzeichnisse hat. Soll ein alternativer Benutzer verwendet werden, ist dieser im Univention Directory Manager anzulegen. Es wird empfohlen, diesem Benutzer so wenig Rechte wie möglich einzuräumen und ihn ausschließlich für `univention-remote-backup` zu verwenden. Der erstellte Sicherungsbeneutzer muss allerdings ein eigenes Benutzerverzeichnis besitzen. Auf den entfernten Systemen muss der Benutzer Leseberechtigung für die zu sichernden Verzeichnisse und alle Unterverzeichnisse erhalten, damit die Sicherung erfolgen kann. Am einfachsten lassen sich diese Leseberechtigungen parallel zu den ursprünglichen Dateirechten setzen, wenn das zu sichernde Verzeichnis in einem Dateisystem liegt, das ACLs unterstützt (z.B. XFS). Mit dem Befehl `setfacl` können die entsprechenden Berechtigungen gesetzt werden. Nähere Informationen enthält die man page zu `setfacl`.

Der zugrunde liegende `rsync`-Mechanismus verwendet SSH als Protokoll zur Verschlüsselung und Benutzerauthentisierung. Da `univention-remote-backup` als Cron-Job ausgeführt wird, kann kein Benutzerpasswort angegeben werden, um den Benutzer zu authentisieren. Es muss ein SSH-Schlüsselpaar für den Benutzer erzeugt werden.

Als Benutzer **root** muss auf dem System, auf welchem `univention-remote-backup` installiert ist, der folgende Befehl ausgeführt werden

```
ssh-keygen -t rsa
```

Es wird ein SSH-Schlüsselpaar erzeugt. Die Abfragen können mittels der Return-Taste bestätigt werden. Für den `univention-remote-backup`-Mechanismus ist es außerdem notwendig, dass der Schlüssel ohne Passphrase, also ohne Passwort, erstellt wird. Dies ist zwar ein Sicherheitsrisiko, allerdings für ein automatisches Backup unvermeidbar. Das erzeugte Schlüsselpaar wird anschließend im Benutzerverzeichnis unter `.ssh/` in den Dateien `id_rsa` und `id_rsa.pub` gespeichert.

Hinweis:

Die folgenden Schritte müssen auf allen Systemen vorgenommen werden, von denen Verzeichnisse durch `univention-remote-backup` übertragen werden sollen.

Kopieren des öffentlichen Teils des im letzten Schritt erzeugten Schlüsselpaares in das Benutzerverzeichnis des Sicherungsbenutzers:

```
scp .ssh/id_rsa.pub <sicherungsbenutzer>@<entfernter Rechner>:
```

Nach der Anmeldung als Sicherungsbenutzer an dem entfernten System muss das Verzeichnis `.ssh` im Benutzerverzeichnis erzeugt und anschließend der folgende Befehl ausgeführt werden. Dieser erstellt, falls nötig, die Datei `authorized_keys` und fügt den übertragenen öffentlichen Schlüssel an das Ende an.

```
cat ~/id_rsa.pub >> .ssh/authorized_keys
```

Nach der Abmeldung sind alle Einstellungen vorgenommen, damit `univention-remote-backup` ausgeführt werden kann.

Bei der Installation von `univention-remote-backup` wird ein Cron-Job erzeugt, der die Sicherung der Client-Systeme von Montag bis Freitag um 13:00 Uhr und die Sicherung der Server-Systeme von Dienstag bis Samstag um 00:10 Uhr veranlasst. Sollen die Ausführungszeiten angepasst werden, kann dies in der Datei `/etc/cron.d/univention-remote-backup` erfolgen. Die Datei ist als Benutzer **root** zu öffnen, das Format ist in der `man 5 crontab` dokumentiert (`man 5 crontab`).

3 Bacula und UCS

3.1 Einführung

Bacula ist ein netzwerkfähiges Datensicherungsprogramm mit einer Client/Server-Architektur. Es ist in der Lage Daten von Clients in einem heterogenen Netzwerk über einen zentralen Server auf Backupmedien zu sichern bzw. wiederherzustellen.

Bacula selbst besteht aus einer Reihe von einzelnen Diensten und Programmen, die die verschiedenen Aspekte der Datensicherung kontrollieren:

Der **Director Daemon** ist die zentrale Steuereinheit, in dem die meisten Einstellungen zum Backup und Restore gespeichert sind. Im Director werden die übrigen Bacula-Dienste konfiguriert.

Der **Storage Daemon** kontrolliert den Zugriff auf die Backupmedien (z.B. eine Tape Library oder Festplatten) und nimmt die Anweisungen des **Directors** entgegen, von welchen Systemen gesichert oder zurückgesichert werden soll.

Der **File Daemon** ist auf den Clients installiert und nimmt die Anweisungen des **Directors** entgegen, welche Dateien über welchen **Storage Daemon** gesichert oder zurückgesichert werden sollen.

Der **Catalog** speichert alle Sicherungen katalogartig in einer externen Datenbank und ermöglicht das Rücksichern einzelner Dateien oder Verzeichnisse.

Die **Bacula Console** ist das zentrale Benutzerinterface für den **Director Daemon**. Von dort können **Backup/Restore Jobs** gestartet werden. Auch administrative Aufgaben - wie das Einbinden von Backupmedien - oder die Abfrage von Statusinformationen werden darüber realisiert.

Das **Bacula Administration Tool** ist eine grafische Version der **Bacula Console**.

Die Backup-Einstellungen (zu sichernde Daten, Backup-Modus- und -zeiten) werden also im **Director Daemon** konfiguriert und das Backup automatisch oder über die **Bacula Console** gestartet. Der **File Daemon** gibt dann die zu sichernden Daten an den **Storage Daemon** weiter, der für die Speicherung der Daten auf physikalischen Medien sorgt. Zusätzlich werden Meta-Information zu den Backups über den **Catalog** in einer Datenbank gesichert.

In dieser Dokumentation wird davon ausgegangen, dass sich der **Director Daemon**, **Storage Daemon** und **Catalog** auf einem System befinden. Der **File Daemon** muss auf allen System, von denen Daten gesichert werden sollen, installiert sein. Als Datenbank wird PostgreSQL vorausgesetzt.

3.2 Installation

3.2.1 Bacula Server

Ab UCS 2.3 steht Bacula in der Version 3.0.2 zu Verfügung. Der „Bacula Server“ (also Storage Daemon, Director Daemon, Catalog) kann über die Univention Management Console installiert werden (Paketname **univention-bacula**).

Während der Installation wird die für den **Catalog** notwendige PostgreSQL-Datenbank angelegt und eingerichtet. Die Zugriffsinformationen dieser Datenbank (Datenbankname, Name/Passwort des Datenbankbenutzers) stehen anschließend in der Datei `/etc/dbconfig-common/bacula-director-pgsql.conf`. Für den Zugriff auf die Datenbank muss noch das PostgreSQL-Konfigurationsdatei-Template `/etc/univention/templates/files/etc/postgresql/8.3/main/pg_hba.conf.d/10-pg_hba.conf` angepasst werden

Achtung:

Bei Updates wird das Template nicht überschrieben, die Änderung aus dem neuen Template



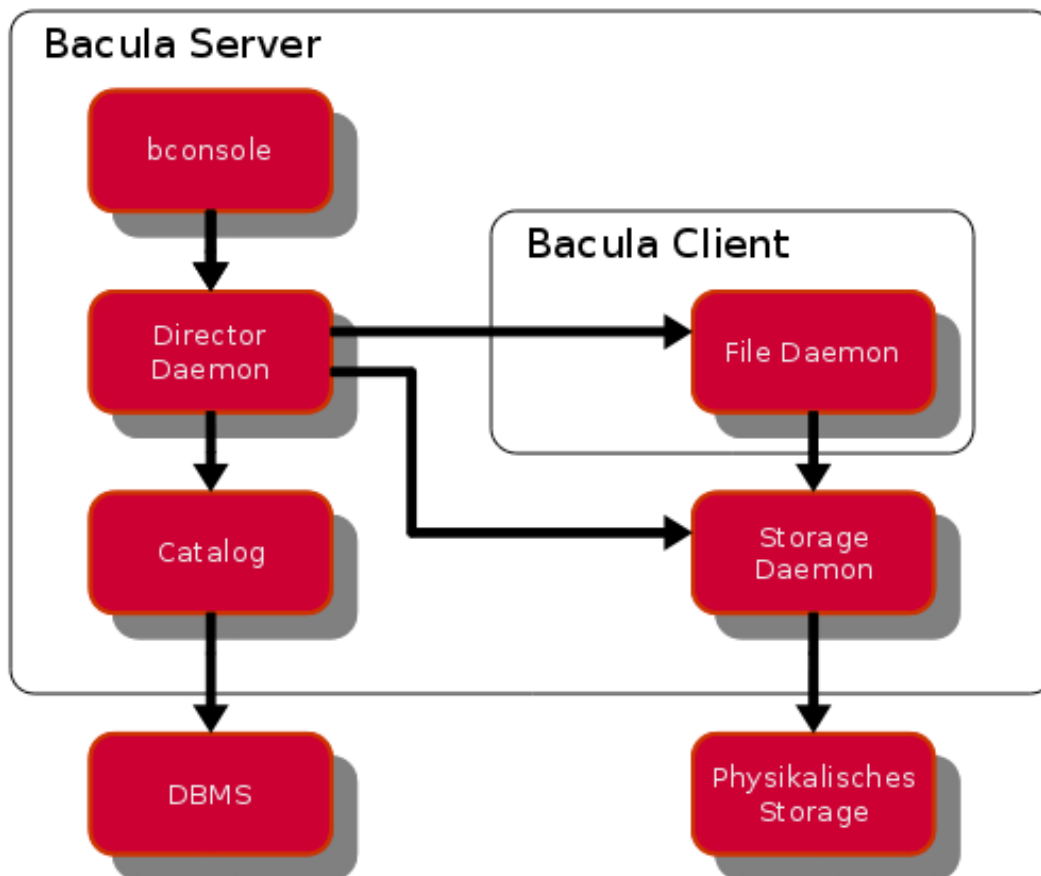


Abbildung 1: Bacula Schema

müssen dann manuell übernommen werden

Hier muss der entsprechende Datenbankbenutzer ergänzt werden, z.B.

```
# bacula
local bacula bacula password
```

Die Konfigurationsdatei wird dann mit

```
univention-config-registry commit /etc/postgresql/8.3/main/pg_hba.conf
```

neu geschrieben. Anschließend wird PostgreSQL neu gestartet:

```
/etc/init.d/postgresql-8.3 restart
```

3.2.2 Bacula Client

Auf Rechnern, von denen Daten gesichert werden sollen, muss lediglich das Paket ***bacula-client*** installiert werden.

3.3 Konfiguration

Die grundlegende Konfiguration der Bacula-Dienste erfolgt über verschiedene Konfigurationsdateien.

/etc/bacula/bacula-dir.conf (Server): Hier wird im Abschnitt ***Director*** der ***Director Daemon*** konfiguriert. Die Standardwerte können beibehalten werden, lediglich die Option ***DirAddress*** sollte von ***127.0.0.1***, also localhost, auf die IP-Adresse des Bacula-Servers geändert werden:

```
...
Director {
  Name = master-dir
  DIRport = 9101
  DirAddress = 10.200.7.22
  QueryFile = "/etc/bacula/scripts/query.sql"
  WorkingDirectory = "/var/lib/bacula"
  PidDirectory = "/var/run/bacula"
  Maximum Concurrent Jobs = 1
  Password = "master-dir-password"
  Messages = Daemon
}
```

Außerdem wird hier die Verbindung zur ***Catalog***-Datenbank definiert. Im Abschnitt ***Catalog*** muss das Passwort des Bacula-Datenbankbenutzers eingetragen werden. Dieses wurde bei der Erstinstallation automatisch gesetzt und ist unter ***dbc_dbpass*** in der Datei [/etc/dbconfig-common/bacula-director-pgsql.conf](#) zu finden.

```
...
# Generic catalog service
Catalog {
  Name = MyCatalog
  # Uncomment the following line if you want the dbi driver
  # dbdriver = "dbi:sqlite3"; dbaddress = 127.0.0.1; dbport =
  dbname = "bacula"; dbuser = "bacula"; dbpassword = "DBPASSWORD"
}
...
```

/etc/bacula/bacula-sd.conf (Server): Im Abschnitt ***Storage*** wird der ***Storage Daemon*** konfiguriert. Auch hier können die Vorgabewerte weitgehend beibehalten werden; nur die Option ***SDAddress*** sollte auf die IP-Adresse des ***Storage Daemons*** angepasst werden. Im Abschnitt ***Director*** wird auf den Bacula-Server verwiesen und ein Passwort gesetzt, das dieser beim Zugriff verwenden muss.

```
...
Storage {
```

```
Name = master-sd
SDPort = 9103
WorkingDirectory = "/var/lib/bacula"
Pid Directory = "/var/run/bacula"
Maximum Concurrent Jobs = 20
SDAddress = 10.200.7.22
}
...
Director {
  Name = master-dir
  Password = "master-storage-password"
}
...
```

/etc/bacula/bacula-fd.conf (Server und Client): Der **File Daemon**, also der Bacula-Client, wird hier konfiguriert. Im Abschnitt **Director** muss die Option **Name** auf den Namen des **Directors** gesetzt werden. Die Option **Password** ist frei wählbar und muss später auf dem Bacula-Server hinterlegt werden, damit dieser Zugriff auf den Client erhält. Außerdem muss hier die Option **FDAddress** im Abschnitt **FileDaemon** auf die IP-Adresse des Systems gesetzt werden.

```
...
Director {
  Name = master-dir
  Password = "client-password"
}
FileDaemon {
  Name = client-fd
  FDport = 9102
  WorkingDirectory = /var/lib/bacula
  Pid Directory = /var/run/bacula
  Maximum Concurrent Jobs = 20
  FDAddress = 10.200.7.22
}
...
```

/etc/bacula/bconsole.conf (Server und Client): Die Konfiguration für die **Bacula Console**. Hier muss im Abschnitt **Director** die Adresse des Rechners, auf dem **Director Daemon** läuft und dessen Passwort angegeben werden.

```
...
Director {
  Name = master-dir
  DIRport = 9101
  address = master
  Password = "master-dir-password"
}
...
```

In den einzelnen Konfigurationsdateien werden weiterhin bestimmte **Ressourcen** definiert, die in einem **Job** zusammengefasst, eine bestimmte Aktion, wie das Backup der Daten X vom Rechner Y auf das Medium Z, repräsentieren. Es gibt u.a. folgende Ressourcen:

- Der Zugriff auf physikalische Backupmedien wird in einem **Device** definiert, z.B. der Gerätetyp und wie es angeschlossen wurde.
- Die verschiedenen Backupmedien (z.B. Bänder oder Festplatten) werden als **Volume** bezeichnet. Volumes können manuell, aber auch direkt vom Director erzeugt werden. Bacula versieht die Volumes dabei mit Software-Labeln zur Identifizierung.
- Bacula verwaltet die Volumes in **Pools**. Dort sind beliebig viele Volumes zusammengeschlossen und deren Eigenschaften definiert. Backups erfolgen ausschließlich auf Pools. Bacula verwaltet dabei die Auslastung der Volumes und überwacht, wann Volumes wieder überschrieben werden dürfen.
- In einem **Schedule** wird definiert, wann eine Aktion ausgeführt wird. Hier können zusätzlich weitere Optionen für eine Aktion gesetzt oder überschrieben werden.
- Ein **FileSet** definiert, welche Dateien oder Verzeichnisse gesichert werden sollen, ob diese komprimiert werden und welche Metainformationen (z.B. ACLs) gesichert werden.
- Jeder Rechner, von dem Daten gesichert werden sollen, wird in Bacula als **Client** behandelt. **Client**-Jobs definieren, um welchen Rechner es sich handelt und wie auf den **File Daemon** des Clients zugegriffen werden kann (z.B. Passwort).

Jobs: Ein Job führt alle die oben genannten Informationen zusammen. Jobs sind entweder vom Typ Restore oder Backup. Außerdem wird hier das Sicherungsverfahren der Backup-Läufe (inkrementelle, volle oder differentielle Sicherung) definiert.

Messages: Hier wird definiert, wie mit Bacula-Statusnachrichten umgegangen werden soll. Meldungen können u.a. in Log-Dateien geschrieben, auf der Konsole angezeigt oder per Email verschickt werden.

3.4 Beispiel-Konfiguration

Im folgenden soll an einem Beispiel die Konfiguration verdeutlicht werden. Dabei werden Daten von einem Client auf Tapes in einem Tape-Streamer gespeichert. Montags bis donnerstags wird ein inkrementelles Backup, freitags ein volles Backup von dem Client durchgeführt. Tapes dürfen nach zwei Wochen überschrieben werden.

3.4.1 Bacula Server

Zunächst muss der **Storage Daemon** in der Datei `bacula-sd.conf` konfiguriert werden. Hier wird in der Ressource **Device** der Tape-Streamer eingerichtet (meist als `/dev/nst0`) und einige Optionen gesetzt (siehe den Abschnitt **Device Resource** in der Bacula-Dokumentation).

```
/etc/bacula/bacula-sd.conf:
...
Device {
    Name = LTO-2                # Name der Ressource (frei wählbar)
    Media Type = LTO-2         # Typ des Mediums
```

```
Archive Device = /dev/nst0 # Gerätedatei
AutomaticMount = yes      # Neue Medien einbinden
AlwaysOpen = yes          # Gerät wird von Bacula immer eingebunden
RemovableMedia = yes      # Wechselmedium
RandomAccess = no        # Zugriff ist nicht beliebig
Maximum File Size = 3GB   # Maximale Größe von Dateien
LabelMedia = yes          # Bacula labelt die Medien
}
...
```

Zusätzlich muss in der Konfigurationsdatei des **Director Daemons** (`bacula-dir.conf`) das konfigurierte Gerät hinterlegt werden.

```
/etc/bacula/bacula-dir.conf:
...
Storage {
  Name = Default           # Name der Ressource
  SDPort = 9103            # Port des Storage Daemon
  Address = master.domain  # FQDN des Storage Daemon
  Password = "master-storage-password" # Passwort des Storage Daemon
  Device = LTO-2           # Der Name des Device
                          # des Storage Daemon
  Media Type = LTO-2       # Der Media Type des Device
                          # des Storage Daemon
}
...
```

Nun muss ein Pool eingerichtet werden der die Volumes (also die Tapes) verwaltet. Dies geschieht über die Ressource **Pool** in der Datei `bacula-dir.conf`. Die Pools müssen aber nicht nur konfiguriert, sondern auch über die **Bacula Console** angelegt werden. Standardmäßig legt Bacula die Pools **Scratch** und **Default** an (Pools können aber auch über das Kommando **create** in der **Bacula Console** angelegt werden). Im folgenden wird der Pool **Default** konfiguriert.

```
/etc/bacula/bacula-dir.conf:
...
# Default pool definition
Pool {
  Name = "Default"        # Name der Ressource
  Pool Type = Backup      # Typ
  Storage = "Default"     # Storage Ressource
  Recycle = yes           # Tapes wiederverwenden
  AutoPrune = yes         # Metadaten von Volumes, die recycelt
                          # werden, aus dem Catalog löschen
  Volume Retention = 28 days # Ablaufzeit für Volumes
}
...
```

Anschließend müssen der **Storage Daemon** und der **Director Daemon** neu gestartet werden:

```
/etc/init.d/bacula-sd restart
/etc/init.d/bacula-director restart
```

Nun müssen die Volumes zum Pool **Default** hinzugefügt werden. Dies geschieht interaktiv über die **Bacula Console**. Auf dem Bacula-Server wird sie mit `bconsole` gestartet.

Mit dem Kommando **help** kann die Hilfe aufgerufen werden. Um die Volumes hinzuzufügen, muss ein Tape in den Streamer gelegt werden. Mit dem Kommando **label** wird dieses Tape als Volume eingebunden. Dabei wird ein Name für das Tape gesetzt und als Pool **Default** angegeben. Alle Tapes, die verwendet werden sollen, müssen auf diese Weise eingebunden werden. Sollte Bacula im laufenden Betrieb kein beschreibbares Volume finden, verlangt es nach einem neuen Volume.

Es gibt nun einen Pool mit verschiedenen Volumes. Dieser Pool kann für Backup-Läufe benutzt werden. Bacula sucht dann automatisch ein benutzbares Volume und verwendet dieses bis kein Speicherplatz mehr vorhanden ist. In diesem Fall muss das Tape im Streamer gewechselt werden. Wenn alle Volumes im Pool erschöpft sind, löscht Bacula die Volumes deren Daten älter als zwei Wochen sind.

Nun müssen noch die eigentlichen Jobs definiert werden. Zunächst wird in `bacula-dir.conf` die Ressource **FileSet** definiert:

```
/etc/bacula/bacula-dir.conf:
...
FileSet {
  Name = "Default"           # Name der Ressource
  Include {                  # Diese Daten werden eingebunden
    Options {               # Optionen
      signature = MD5
      aclsupport=yes
      Compression=GZIP
    }
    File = /etc
    File = /root
    File = /home
  }
  Exclude {                  # Diese Daten werden ausgelassen
    File = /.journal
    File = /.fsck
  }
}
...
```

Es sollen also alle Daten in den Verzeichnissen `/etc`, `/root` und `/home` mit Ausnahme der Dateien `.journal` und `.fsck` gesichert werden. Die Daten werden mit `gzip` komprimiert und inklusive der ACLs gesichert. Über **Exclude** können Dateien von der Sicherung ausgeschlossen werden.

Der Startzeitpunkt der Backups wird dann über die Ressource **Schedule** definiert:

```
/etc/bacula/bacula-dir.conf:
...
Schedule {
  Name = "Default"
  Run = Level=Incremental monday-thursday at 23:00
  Run = Level=Full friday at 23:00
}
...
```

Backup-Jobs mit diesem **Schedule** werden montags bis freitags um 23 Uhr gestartet, wobei montags bis donnerstags ein inkrementelles Backup durchgeführt und freitags ein Vollbackup angestoßen wird.

Zusätzlich wird über die Ressource **Messages** definiert, dass alle Meldungen auf die Konsole und in eine Log-Datei geschrieben werden:

```
/etc/bacula/bacula-dir.conf:
...
Messages {
  Name = "Default"
  console = all, !skipped, !saved
  append = "/var/lib/bacula/log" = all, !skipped
  catalog = all
}
...
```

Das eigentliche System, von dem Daten gesichert werden sollen, wird in der Ressource **Client** definiert:

```
/etc/bacula/bacula-dir.conf:
...
Client {
  Name = master # Name der Ressource
  Address = 10.200.7.22 # Netzwerkadresse
  FDPort = 9102 # Port
  Catalog = MyCatalog # In welchen Catalog die Metadaten
  # gespeichert werden.
  Password = "client-password" # Das Passwort des Client, aus dessen
  # bacula-fd.conf
  File Retention = 30 days # Metadaten zu gelöschten Dateien werden
  # im Catalog 30 Tage aufgehoben
  Job Retention = 6 months # Ablaufzeit für Metadaten von Jobs
  AutoPrune = yes # Bacula darf Metadaten von Volumes
  # löschen, falls eine Volume benötigt
  # wird
}
...
```

Abschließend werden alle Ressourcen in der Ressource **Job** zu einem Backup-Job zusammengefasst.

```
/etc/bacula/bacula-dir.conf:
...
Job {
  Name = "master" # Name der Ressource
  Type = Backup # Type
  Level = Full # Default Level, wird durch Schedule überschrieben
  Client = master-fd # Für welches System ist dieser Job
  FileSet = "Default" # Verweis auf FileSet Ressource
  Storage = "Default" # Verweis auf Device Ressource
  Schedule = "Default" # Verweis auf Schedule Ressource
  Pool = "Default" # Verweis auf Pool Ressource
  Priority = 10 # Priorität
}
```

```
Messages = "Default" # Verweis auf Ressource Messages, dort
                    # wird definiert, wie Meldungen behandelt
                    # werden (standardmäßig alles an bconsole)
}
...
```

Zusätzlich kann ein **Restore Job** eingerichtet werden, der als Vorlage für das interaktive Zurrücksichern von Daten verwendet wird:

```
/etc/bacula/bacula-dir.conf:
...
Job {
  Name = "RestoreFiles"
  Type = Restore
  Client = master-fd
  FileSet = "Default"
  Storage = "Default"
  Pool = "Default"
  Messages = "Default"
  Where = /opt          # Wohin Daten zurückgespielt werden
}
...
```

Der **Director Daemon** muss nun neu gestartet werden:

```
/etc/init.d/bacula-director restart
```

Nun ist ein Backup Job konfiguriert, der vom Client **10.200.7.22** die Verzeichnisse `/etc`, `/root` und `/home` komprimiert inklusiv ihrer ACLs, auf Volumes (Tapes) im Pool **Default** sichert. Montags bis donnerstags wird ein inkrementelles Backup, freitags ein Vollbackup um 23:00 Uhr gestartet.

3.4.2 Bacula Client

Der Bacula Client, also der **File Daemon** wird über die Datei `/etc/bacula/bacula-fd.conf` konfiguriert. Hier müssen lediglich, die Optionen **Name** und **Password** im Abschnitt **Director** mit den Werten aus dem `/etc/bacula/bacula-dir.conf` des **Director Daemon** abgestimmt werden.

Danach wird der **File Daemon** neu gestartet:

```
/etc/init.d/bacula-fd restart
```

3.5 Administration über die Bacula Console

Mit der **Bacula Console** können Informationen über den Status von Bacula ausgelesen, Backup-Jobs gestartet oder Daten zurückgesichert werden. Gestartet wird die Konsole mit dem Befehl `bconsole`.

Das Kommando **status** zeigt Status-Informationen an. Es wird z.B. eine Liste der anstehenden, laufenden und beendeten Jobs des Directors ausgegeben.

In der Beispielkonfiguration wurde ein Backup-Job definiert, der automatisch an jedem Wochentag gestartet wird. Backups und Rücksicherungen können aber auch interaktiv über die **Bacula Console** gestartet werden.

Mit dem Kommando **run** kann ein Job gestartet werden. Es wird daraufhin eine Liste der verfügbaren Jobs angezeigt, aus denen der gewünschte Job ausgewählt werden muss. Mit dem Kommando **mod** können Optionen wie der Sicherungstyp für den Job gesetzt bzw. geändert werden. Nach Bestätigung durch **yes** wird der Job gestartet.

Mit dem Kommando **restore** können Daten zurückgesichert werden. Nun kann mit **3 (Enter list of comma separated Joblds to select)** ein Backup-Job ausgewählt werden, von dem Daten zurückgesichert werden sollen. Dann erscheint ein Dateibrowser, in dem mit den Standardkommandos `cd` und `ls` navigiert werden kann. Hier können mittels **mark FILE** bzw. **mark -r DIR** Dateien bzw. Verzeichnisse für die Rücksicherung markiert werden. Sind alle gewünschten Daten markiert, wird der Dateibrowser mit **done** beendet. Nach der Angabe des Clients und der Bestätigung einiger Optionen für den Restore-Job (z.B. wohin die Daten kopiert werden sollen) kann der Restore-Job mit **yes** gestartet werden. Nach Abschluß befinden sich die ausgewählten Daten im konfigurierten Rücksicherungsverzeichnis.

Falls für ein Backup oder Restore ein Tape benötigt wird, das sich nicht im Laufwerk befindet, fordert Bacula dieses Tape explizit an.

Weitere Informationen über die **Bacula Console** können der Bacula-Dokumentation bzw. dem Kommando **help** in der Bacula Konsole entnommen werden.

3.6 Sicherung der Catalog-Datenbank

Die Metadaten der Sicherung werden in so genannten **Catalog** gespeichert. Standardmäßig wird der Catalog in einer PostgreSQL-Datenbank gespeichert, die ebenfalls gesichert werden sollte. Dies erfolgt über einen Backup-Job, der einen SQL-Dump der Datenbank sichert.

```
/etc/bacula/bacula-dir.conf:
...
Job {
  Name = "BackupCatalog"
  Type = Backup
  Level = Full
  FileSet="Catalog"
  Schedule = "WeeklyCycleAfterBackup"
  # make_catalog_backup <database-name> <user-name> <password> <host>
  RunBeforeJob = "/etc/bacula/scripts/make_catalog_backup \
    bacula bacula dbpassword"
  RunAfterJob = "/etc/bacula/scripts/delete_catalog_backup"
  Write Bootstrap = "/var/lib/bacula/BackupCatalog.bsr"
  Priority = 11
  Pool = "Default"
  Storage = "Default"
```

```
Client = master-fd
Messages = "Default"
}
...
Schedule {
  Name = "WeeklyCycleAfterBackup"
  Run = Full monday-saturday at 23:10
}
...
FileSet {
  Name = "Catalog"
  Include {
    Options {
      signature = MD5
      Compression=GZIP
    }
    File = /var/lib/bacula/bacula.sql
  }
}
...
```

Über die Anweisungen **RunBeforeJob** und **RunAfterJob** werden vor bzw. nach der eigentlichen Sicherung Skripte ausgeführt. Im Falle des Catalogs wird mit `make_catalog_backup` vor der Sicherung ein SQL-Dump der Catalog-Datenbank erzeugt und unter `/var/lib/bacula/bacula.sql` gespeichert. Nach erfolgter Sicherung wird diese Datei wieder entfernt. Wichtig ist hier, dass dem Skript die Benutzerdaten der Bacula Datenbank, also Name und Passwort, übergeben werden.

Zusätzlich wird für das Backup des Catalogs mit **Write Bootstrap** eine Bootstrap-Datei erzeugt. In dieser Datei wird protokolliert, wie die Daten wiederhergestellt werden können, d.h. auf welchem Volume sie gespeichert sind und wo auf dem Volume sie sich befinden. Normalerweise übernimmt dies der Catalog selbst, für den Fall der Rücksicherung der Catalog-Datenbank wird jedoch die Bootstrap-Datei benötigt. Sie sollte unabhängig von Bacula zusätzlich gesichert werden.

Der Backup-Job des Catalogs, mit dazugehörigem **FileSet** und **Schedule**, ist als Vorlage bereits in der Konfiguration des **Director Daemon** enthalten und muss lediglich angepasst werden.

3.7 Weiterführende Informationen

Weitere Informationen zur Bacula-Einrichtung sind unter anderem auf den folgenden Webseiten zu finden:

- <http://www.bacula.org>
- <http://wiki.bacula.org/doku.php>
- <http://www.online-tutorials.net/backup/bacula/tutorials-t-125-302.html>
- <http://de.wikipedia.org/wiki/Bacula>

- <http://www.bacula.org/de/dev-manual/Kurzanleitung.html>