

# METHODEN ZUR DATENSICHERUNG

STRATEGIEN UND TECHNIKEN FÜR NETBSD  
INKLUSIVE EINES KAPITELS ZU POSTGRESQL

STEFAN SCHUMACHER



# METHODEN ZUR DATENSICHERUNG

STRATEGIEN UND TECHNIKEN FÜR NETBSD  
INKLUSIVE EINES KAPITELS ZU POSTGRESQL

STEFAN SCHUMACHER

[www.net-tex.de](http://www.net-tex.de)

Magdeburg, den 2. März 2006

Copyright © 2002, 2003, 2004, 2005, 2006 Stefan Schumacher

`$Id: backup.tex,v 1.1 2003/02/28 09:54:15 stefan Exp $`

`$Id: backup.tex,v 1.116 2006/03/02 15:07:46 stefan Exp $`

`/home/stefan/daten/dokumente/ausarbeitungen/datensicherung/backup.tex`

Dies ist Version 1.116 mit [134](#) Seiten, kompiliert am 2. März 2006 mit pdfL<sup>A</sup>T<sub>E</sub>X  
3.141592-1.21a-2.2 (Web2C 7.5.4).

# Inhaltsverzeichnis

<b>1. Vorwort</b>	<b>13</b>
1.1. Danksagung . . . . .	14
1.2. Vorträge . . . . .	14
1.3. Warnung . . . . .	14
1.4. Urheberrechtshinweis . . . . .	15
<b>I. Strategie und Organisation</b>	<b>17</b>
<b>2. Strategie und Organisation</b>	<b>19</b>
2.1. Definitionen . . . . .	19
2.2. Vorbereitung der Strategie . . . . .	19
2.3. Inventur . . . . .	20
2.4. Bedrohungsszenarien analysieren . . . . .	20
2.5. Wichtige Daten finden und organisieren . . . . .	21
2.6. Sicherung der Sicherungssysteme . . . . .	23
2.7. Organisation der Sicherung . . . . .	23
2.7.1. Einweisung der Benutzer . . . . .	23
2.7.2. Cronjob vs. Batchjob . . . . .	23
2.7.3. Homogene Systeme . . . . .	24
2.7.4. Dokumentation der Administration . . . . .	24
2.7.5. Shellskripte statt Handarbeit . . . . .	25
2.7.6. Logdateien erstellen und auswerten . . . . .	25
2.7.7. Verifikation der Daten . . . . .	25
2.7.8. Den GAU erwarten . . . . .	25
2.7.9. Alles sichern . . . . .	26
2.7.10. Testsysteme und -läufe . . . . .	26
2.7.11. Verschiedene Sicherungskreise . . . . .	27
2.7.12. Maximale Archivierungsdauer . . . . .	27
2.7.13. Wochenende und Feiertage beachten . . . . .	27
2.8. Dokumentation . . . . .	28
2.9. Testen, Testen, Testen . . . . .	28
2.10. Verminderung von Ausfällen und Ausfallzeiten . . . . .	29
2.11. Datenrettung . . . . .	30

<b>3. Sicherungsvarianten</b>	<b>31</b>
3.1. Kompletbackup	31
3.2. Differentielles Backup	31
3.3. Inkrementelles Backup	31
3.4. Dump-Level	31
3.5. Beispielstrategie	32
3.6. Komprimierung und Verschlüsselung	34
3.7. Medien	35
3.7.1. Organisation und Lagerung der Medien	37
3.8. Prüfliste zur Strategie	38
<b>II. Sicherung einzelner Systeme</b>	<b>41</b>
<b>4. Programme automatisieren</b>	<b>43</b>
4.1. Zeitgesteuert	43
4.2. Ablaufgesteuert	45
<b>5. Sicherung einzelner Systeme</b>	<b>47</b>
5.1. Tägliche Sicherungsmechanismen im Basissystem	47
5.2. Das Basissystem sichern	47
5.3. Filesystem-Snapshots	50
5.3.1. Praktischer Einsatz	50
5.3.2. Sicherung eines Snapshots	50
5.4. dump(8)/dump_lfs(8) und restore(8)	51
5.4.1. restore	52
5.4.2. Dump im Detail	54
5.5. tar(1)	57
5.6. star	57
5.7. cpio(1)	58
5.8. afio(1)	58
5.9. pax(1)	59
5.10. Verzeichnisse synchronisieren mit rsync(1)	59
5.10.1. MS Windows als Client	61
5.11. rdiff-backup	61
5.12. dd(1)	62
5.13. Ghost for Unix (g4u)	63
<b>III. Sicherung verteilter Systeme</b>	<b>65</b>
<b>6. Sicherung verteilter Systeme in Netzwerken</b>	<b>67</b>
6.1. Amanda	67
6.1.1. Einrichtung des Servers	68

6.1.2.	Einrichtung der Clients	71
6.1.3.	Programme im Amanda-Paket	71
6.1.4.	Praktischer Einsatz	72
6.2.	Bacula	73
6.2.1.	Installation	75
6.2.2.	Konfiguration	75
6.2.3.	Rücksicherung	78
<b>IV. PostgreSQL</b>		<b>85</b>
<b>7. Sicherung eines PostgreSQL-Servers</b>		<b>87</b>
7.1.	Den Datenbankcluster sichern	87
7.2.	Point-in-Time-Recovery	87
7.3.	pg_dump, pg_dumpall und pg_restore	90
7.4.	Replikation	90
7.4.1.	Synchrone Replikation mit Pgpool	91
<b>V. Sonstige Programme</b>		<b>95</b>
<b>8. Sonstige Programme</b>		<b>97</b>
8.1.	Magnetbänder ansteuern mit mt(1)	97
8.2.	Datenströme puffern	97
8.3.	Dateien mit split(1) zerlegen	99
8.4.	Dateien mit find(1) finden	99
8.5.	Das kryptographische Dateisystem CFS	100
8.6.	Das kryptographische Pseudogerät CGD	100
8.7.	symmetrische Verschlüsselung mit mdecrypt	101
8.8.	Versionsverwaltung mit CVS	101
8.9.	Dateisystemintegrität prüfen	102
8.10.	Prüfsummen einzelner Dateien erstellen	103
<b>VI. Der Test</b>		<b>105</b>
<b>9. Der Test</b>		<b>107</b>
9.1.	Problemfelder	107
9.1.1.	Datumsprobleme	107
9.1.2.	Dateigrößen	107
9.1.3.	Verschiedene Dateitypen	107
9.1.4.	Zeichensätze	108
9.1.5.	Löcher in Dateien	108
9.1.6.	lange Pfadnamen	108
9.1.7.	Zugriffsrechte	109

9.1.8. Testgestellung und Testdurchführung . . . . .	109
9.2. Testergebnisse . . . . .	110
9.3. Auswertung . . . . .	115
9.3.1. dump . . . . .	115
9.3.2. Pax . . . . .	115
9.3.3. Tar . . . . .	115
9.3.4. Cpio . . . . .	115
9.3.5. Afio . . . . .	116
9.3.6. Star . . . . .	116
9.3.7. Amanda . . . . .	116
9.3.8. Bacula . . . . .	116
9.3.9. Rsync und Rdiff-Backup . . . . .	116
9.4. Fazit . . . . .	116

<b>VII. Anhang</b>	<b>119</b>
<b>A. Einen einzelnen Rechner sichern</b>	<b>121</b>
<b>B. Prüfliste zur Strategie</b>	<b>125</b>
<b>Literaturverzeichnis</b>	<b>127</b>
<b>Index</b>	<b>131</b>



## Listings:

2-1. Daten verifizieren . . . . .	26
3-1. Archive komprimieren und verschlüsseln . . . . .	35
4-1. Cronjobs um PostgreSQL zu sichern . . . . .	44
4-2. Anacrontab-Beispiel . . . . .	44
4-3. Jobs zur späteren Ausführung vorbereiten . . . . .	45
5-1. Betriebssystem mit g4u sichern . . . . .	48
5-2. Konfigurationsdateien sichern . . . . .	49
5-3. fssconfig-Optionen . . . . .	50
5-4. Snapshot mit Dump sichern . . . . .	51
5-5. NODUMP setzen . . . . .	52
5-6. Sicherung mit dump . . . . .	52
5-7. dump-Befehl für mehrere Volumes . . . . .	53
5-8. Dump im Netzwerk . . . . .	53
5-9. Zwei Dumps mit restore zurückspielen . . . . .	54
5-10. einzelne Dateien mit restore zurückspielen . . . . .	54
5-11. Protokoll einer Dump-Sicherung . . . . .	55
5-12. Dateien mit tar archivieren und entpacken . . . . .	57
5-13. Dateien mit cpio sichern . . . . .	58
5-14. Sicherung und Rücksicherung mit pax . . . . .	59
5-15. rsync-Beispiele . . . . .	59
5-16. /etc/powerd/scripts/power_button . . . . .	60
5-17. cwRsync auf Windows einsetzen . . . . .	61
5-18. Windows-Programme zeitgesteuert ausführen . . . . .	61
5-19. rdiff-backup . . . . .	62
5-20. Daten mit dd schreiben . . . . .	63
5-21. Festplatten-Images mit g4u erstellen . . . . .	63
6-1. Spoolplatten für Amanda definieren . . . . .	70

6-2. dumptypes in amanda.conf definieren . . . . .	70
6-3. Zu sichernden Pfade in der disklist festlegen . . . . .	71
6-4. Amanda-Server-Dienste starten . . . . .	71
6-5. Amanda-Client-Dienste starten . . . . .	71
6-6. Zugriffsrechte auf dem Client konfigurieren . . . . .	72
6-7. Amanda Header vom Band restaurieren . . . . .	73
6-8. PostgreSQL für Bacula einrichten . . . . .	75
6-9. Bacula per rc.d starten . . . . .	78
6-10. Baculas bconsole starten . . . . .	79
6-11. Beispiel einer Konfiguration für Bacula . . . . .	81
6-12. Daten mit Bacula rücksichern (1) . . . . .	82
7-1. PostgreSQL-Cluster mit dump sichern . . . . .	88
7-2. postgresql.conf-Konfig zur Sicherung der Logs . . . . .	89
7-3. PostgreSQL-Datenbankcluster und WALs sichern . . . . .	89
7-4. PostgreSQL-Datenbanken sichern . . . . .	90
7-5. Rückspielen von PostgreSQL-Sicherungen . . . . .	90
7-6. Pgpool-Konfiguration für die Replikation . . . . .	93
8-1. Bänder mit mt manipulieren . . . . .	97
8-2. mt(1)-Optionen . . . . .	98
8-3. Datenströme puffern . . . . .	98
8-4. Dateien zerlegen und zusammensetzen . . . . .	99
8-5. Dateien mit find(1) finden . . . . .	100
8-6. eine CGD-Partition verschlüsselt dumpen . . . . .	101
8-7. symmetrische Verschlüsselung mit mcript . . . . .	101
8-8. Datei mit mcript entschlüsseln . . . . .	102
8-9. Fingerabdruck eines Systems mitmtree erstellen . . . . .	102
8-10. Beispiel einesmtree-Reports . . . . .	103
8-11. Dateisystem mit Fingerabdruck vergleichen . . . . .	103
8-12. Ergebnisse eines Dateisystemvergleichs . . . . .	104
8-13. Dateien mit SHA1 vergleichen . . . . .	104
9-1. Testgestellung für den Belastungstest . . . . .	109
A-1. Shellskript für Dump (1/3) . . . . .	121
A-2. Shellskript für Dump (2/3) . . . . .	122
A-3. Shellskript für Dump (3/3) . . . . .	123

# Tabellenverzeichnis

3.1. Wochensicherung mit Leveln . . . . .	32
3.2. Datenmenge einer Komplettsicherung . . . . .	32
3.3. Datenmenge einer Inkrementellsicherung . . . . .	33
3.4. Datenmenge einer Differentiellsicherung . . . . .	33
3.5. Türme-von-Hanoi-Algorithmus . . . . .	33
3.6. Datenverteilung im Hanoi-Algorithmus . . . . .	34
3.7. Vergleich von Sicherungsmedien . . . . .	37
3.8. Prüfliste zur Strategie . . . . .	39
5.1. Dumplevel für einen Einzelrechner . . . . .	49
6.1. Baculas PostgreSQL-Datenbank . . . . .	76
6.2. Baculas Table „public.file“ . . . . .	77
6.3. Baculas Table „public.filename“ . . . . .	77
6.4. Baculas bconsole-Befehle . . . . .	80
6.5. Daten mit Bacula rücksichern (2) . . . . .	83
9.1. Auswertung von dump und tar . . . . .	111
9.2. Auswertung von pax und cpio . . . . .	112
9.3. Auswertung von Rsync und Rdiff-Backup . . . . .	113
9.4. Auswertung von star und Bacula . . . . .	114



# 1. Vorwort

Dieses Dokument stellt Strategien und Konzepte zur Datensicherung von einzelnen Rechnern und verteilten Systemen vor. Es werden mögliche Sicherungskonzepte eingeführt und diskutiert. Verfügbare Programme im Basissystem und Lösungen von Drittanbietern werden vorgestellt und einige davon mit einem Zuverlässigkeitstest überprüft. Der Einsatz geeigneter Programme wird an Beispielen gezeigt. Weiterhin werden Sicherungsmöglichkeiten und Replikationsmethoden für PostgreSQL vorgestellt.

Im ersten Teil wird die Strategie und Organisation der Datensicherung beleuchtet. Was muss wie organisiert werden, damit meine Datensicherung auch wirklich klappt? Hierbei geht es darum eine Strategie zu erstellen, in dem man Bedrohungsszenarien analysiert, zu sichernde Daten bestimmt und die beste Sicherungsvariante festlegt um die Sicherungsstrategie korrekt zu implementieren.

Der zweite Teil stellt Programme vor, mit denen einzelne Rechner oder auch kleinere Netze bspw. von Kleinen und Mittleren Unternehmen, Arbeitsgruppen oder Privatpersonen, gesichert werden können. Insbesondere die Sicherungsprogramme des Basissystems werden vorgestellt und besonders dump(8) näher untersucht und diskutiert.

Der dritte Teil stellt Programme vor, die speziell für größere Netze entwickelt wurden. Dabei handelt es sich um Amanda und Bacula.

Der vierte Teil behandelt PostgreSQL bezüglich Datensicherung mit den mitgelieferten Werkzeugen zur logischen Datensicherung, den Einsatz der Write-Ahead-Logs für Point-In-Time-Recovery und Replikationsmethoden mit Pgpool um die Ausfallsicherheit eines PostgreSQL-Servers zu erhöhen.

Im fünften Teil werden Programme vorgestellt, die nicht direkt der Datensicherung dienen, aber nützlich sind.

Im sechsten Teil werden die Ergebnisse eines Zuverlässigkeitstest für Sicherungsprogramme vorgestellt und analysiert. Elizabeth D. Zwicky hat bereits 1991 und wiederholt 2003 Sicherungsprogramme auf ihre Zuverlässigkeit und Robustheit getestet. Ich habe diesen Test auf NetBSD mit verschiedenen Programmen aus dem Basissystem und weiteren Anwendungen, die in dieser Anleitung vorgestellt werden, wiederholt. Es wird die Testgestaltung und -durchführung beschrieben, die Ergebnisse werden in tabellarischer Form präsentiert und anschließend bewertet.

Im siebenten und letzten Teil befinden sich die Anhänge, wie Skripte, Index und Bibliographie.

## 1.1. Danksagung

Dank für Korrekturen und Anregungen geht an:

- Peter Eisentraut, <PeterE@PostgreSQL.org>
- Hubert Feyrer, <HubertF@NetBSD.org>
- Juergen Hannken-Illjes, <Hannken@NetBSD.org>
- Matthias Scheler, <Tron@NetBSD.org>
- #netbsd im IRCnet
- news://de.comp.os.unix.bsd

## 1.2. Vorträge

Dieser Artikel ist die Grundlage verschiedener Vorträge:

- Magdeburger Linux-User-Group, Themenabend am 28.02.2006 in Magdeburg
- Chemnitzer Linux-Tage, 04. und 05. März 2006 in Chemnitz
- Frühjahrsfachgespräch 2006 der German Unix User Group, 23. und 24. März 2006 in Osnabrück. Zu dieser Veranstaltung wird es ein Tagungsband geben, in dem eine stark gekürzte Fassung dieses Artikel enthalten ist.

Auf meiner Webseite<sup>1</sup> sind die passenden Versionen des Artikels, der verwendeten Folien und der aktuellsten Version des Artikels verfügbar.

## 1.3. Warnung

Ich habe diesen Artikel sorgfältig recherchiert und weiß im Allgemeinen wovon ich rede. Trotzdem gewähre ich diese Informationen nur auf eigene Gefahr, da das Kapitel der Datensicherung sehr komplex ist und ich weder alle Fälle und Besonderheiten kennen noch berücksichtigen kann.

Wenn ein professionelles Datensicherungsverfahren notwendig ist, Sie aber nicht wissen wie es fehlerfrei umgesetzt werden kann, besorgen Sie sich professionelle Hilfe. Das ist in der Regel billiger als ein Datenverlust.

Fragen werde ich in der Regel beantworten, es bietet sich aber auch an die NetBSD-Mailinglisten oder das Usenet (de.alt.comp.os.unix.bsd) zu benutzen.

---

<sup>1</sup><http://www.net-tex.de/netbsd/backup.html>

## **1.4. Urheberrechtshinweis**

Sämtliche Rechte am Werk verbleiben beim Urheber. Weitergabe des unveränderten Artikels ist gestattet und erwünscht, Wiedergabe in kommerziellen Medien bedarf der Genehmigung des Autors. Zitate gemäß den üblichen Regeln sind erwünscht.





**Teil I.**

# **Strategie und Organisation**



## 2. Strategie und Organisation

*„Niemand will Backup. Alle wollen Restore.“*  
Kristian Köhntopp zitiert einen Vertriebler

### 2.1. Definitionen

Ein Backup ist die Sicherung relevanter Daten um diese vor Verlust oder Beschädigung zu schützen. Es ist ein Sicherungssystem für den Fall des Verlusts der Daten. Im allgemeinen Falle ist eine Datensicherung nicht mit der Archivierung der Daten gleichzusetzen, da in der Regel nur eine Momentaufnahme des Datenbestandes gesichert wird. Ist eine Archivierung des Datenbestandes notwendig<sup>1</sup>, kann dies über spezielle Archivierungsprogramme erfolgen, oder in dem die Datensicherungen selbst archiviert werden. Eine weitere Möglichkeit Daten vor Verlust zu schützen ist die Spiegelung dieser auf andere Systeme.

### 2.2. Vorbereitung der Strategie

Da die Organisation einer Sicherungsstrategie äußerst komplex ist, ist es notwendig eine umfassende Anforderungsanalyse vorzunehmen und die gewonnenen Erkenntnisse umzusetzen. Hierzu gilt es einen Notfallplan<sup>2</sup> zu erstellen, der das Fundament für die Sicherungsstrategie bildet.

Die einzelnen Schritte umfassen:

1. Inventur
2. Bedrohungsszenarien analysieren
3. Wichtige Daten finden

---

<sup>1</sup>bspw. §239 Abs. 4 Satz 2 HGB *„Die Handelsbücher und die sonst erforderlichen Aufzeichnungen können auch in der geordneten Ablage von Belegen bestehen oder auf Datenträgern geführt werden, soweit diese Formen der Buchführung einschließlich des dabei angewandten Verfahrens den Grundsätzen ordnungsmäßiger Buchführung entsprechen. Bei der Führung der Handelsbücher und der sonst erforderlichen Aufzeichnungen auf Datenträgern muß insbesondere sichergestellt sein, daß die Daten während der Dauer der Aufbewahrungsfrist verfügbar sind und jederzeit innerhalb angemessener Frist lesbar gemacht werden können. Absätze 1 bis 3 gelten sinngemäß.“*

<sup>2</sup>Neudeutsch auch gerne als *„Disaster Recovery Plan“* bezeichnet.

4. Sicherungssysteme sichern
5. Dokumentation
6. Testen

Am einfachsten ist es, den Plan so aufzustellen, wie es im Idealfall (also ohne technische Einschränkungen) möglich wäre. Anschließend passt man den Idealplan an die reale Situation an, in dem man bspw. die Sicherungszyklen an die verfügbaren Zeitfenster anpasst. Steht der Plan und seine Implementierung, gilt es ihn zu testen, zu testen, zu testen und zu warten. Denn die zugrundeliegende Umgebung verändert sich und ebenso wie Sicherheitsstrategien, muss der Plan stetig an die neue Situation angepasst werden. Daher empfiehlt es sich, in der Regel alle sechs Monate, den Plan zu überprüfen und gegebenenfalls anzupassen.

### 2.3. Inventur

Ziel der Inventur ist es, einen Überblick über die eingesetzten Systeme zu bekommen. Dabei ist es notwendig zu katalogisieren, welche Hardware, Betriebssysteme, Anwendungsprogramme und Dienste auf welchen Maschinen laufen. Anhand dieser Inventur können Anforderungen an die Sicherungssoftware festgelegt werden.

Existiert schon ein Inventar, idealerweise in einer Datenbank oder einem speziellen Inventurprogramm, kann dieses um die Informationen bezgl. der Sicherung erweitert werden. Falls noch kein Inventar besteht, sollte beim Erstellen eines Neuen auf die Erfassung der notwendigen Daten zur Organisation der Sicherung geachtet werden.

### 2.4. Bedrohungsszenarien analysieren

„Wodurch werden meine Daten bedroht?“ und „Wie kann ich mich vor diesen Bedrohungen schützen?“ lauten die Fragestellungen zu den Bedrohungen.

In der Regel gibt es bestimmte Fehlertypen / Bedrohungen, die jeweils eine eigene Sicherungsstrategie erfordern:

1. Benutzerfehler  
Der Benutzer löscht oder verfälscht Dateien. Hier muss die letzte vorhandene Version zurückgespielt werden. Dieser häufige Fehler erfordert eine Archivierung der Daten oder Sicherungsbänder.
2. Administratorenfehler  
Ist in der Regel eher selten, wenn er aber vorkommt, dann richtig. Schließlich kann sich `root` nicht nur einfach ins Knie schießen, sondern beide Beine wegblasen. Hier hilft meist nur die Rücksicherung des kompletten Systems, daher sollte man das *gesamte* System bspw. mit Snapshots sichern.

### 3. Systemfehler

#### a) Festplatte defekt

Leider auch recht häufig, reißt dies meist alle Daten (und oft das gesamte System) in den Tod. Hier helfen Sicherungen auf Wechselmedien, andere Rechner oder RAID-Systeme.

#### b) Dateisystemkorruption

Hier helfen nur *archivierte* Sicherungen, da sich ein Fehler auf die Spiegel bzw. Sicherungen fortpflanzt.

### 4. elektronischer Einbruch / Vandalismus / Diebstahl

Einbruch auf elektronischem Wege und daraus resultierende Vernichtung der Daten<sup>3</sup> erfordert eine Rücksicherung der Daten. Im Falle schleichender Korruption<sup>4</sup> sind auch hier wieder Archive notwendig.

### 5. herkömmlicher Einbruch / Vandalismus / Diebstahl

Was ist wenn jemand einbricht und alle Rechner klaut? Hier hilft die letzte Datensicherung weiter, aber wurde die auch gestohlen oder vernichtet?

### 6. Naturkatastrophen / Höhere Gewalt

Glücklicherweise wird Deutschland nicht von Wirbelstürmen oder Erdbeben heimgesucht, aber Fluten sind an gewissen Standorten schon eine Gefahr. Weiterhin können auch andere Gefahren lauern, wie Flugzeugabstürze / -einschläge, Erdbeben, Großbrände oder Unglücke in benachbarten Unternehmungen (Chemiewerk, E-Werk, Raffinerie).

Was passiert also wenn unser Gebäude/Stadtteil/Stadt ausradiert wird und unser Unternehmen die Daten noch benötigt? Existieren Sicherungen an anderen Orten? Hier sollte entsprechend dem Gefahrenpotential Sicherungsbänder ausgelagert werden, bspw. in andere Filialen. Ebenso möglich ist eine Spiegelung/Replikation der Datenbestände auf entfernte Systeme.

## 2.5. Wichtige Daten finden und organisieren

Als erster Schritt ist es notwendig, alle zu sichernden Daten zu finden. Dies sind in der Regel alle Daten, deren Verlust nicht akzeptabel ist.

Dies hat drei Gründe:

- nicht wiederherstellbare Daten

Die Daten sind bei Verlust nicht wiederherstellbar, da sie bspw. an einen bestimmten Zeitpunkt gebunden sind<sup>5</sup> oder die beteiligten Programme / Bearbeiter nicht mehr verfügbar sind

---

<sup>3</sup>Die Daten müssen nicht unbedingt gelöscht worden sein, aber sind sie noch integer?

<sup>4</sup>Wurden Daten manipuliert? Wenn ja, wann?

<sup>5</sup>z.B. Patientendaten, Logdateien, Versuchsprotokolle

- wirtschaftlicher Totalschaden  
Die Daten sind zwar rekonstruierbar, der Aufwand dafür, oder die Nichtverfügbarkeit, übersteigt monetär den Aufwand der Datensicherung.
- Systemdaten  
Auf einem System liegen nicht nur einfach Daten vor, sondern es existiert mindestens ein Betriebssystem und in der Regel auch noch weitere Programme bzw. Dienste. Es empfiehlt sich auch diese Daten zu sichern. Moderne Sicherungsmedien stellen genügend Platz für Betriebssystemdateien zur Verfügung. Kann, oder möchte, man Betriebssystem und Anwendungen nicht sichern, sollte man zumindest deren Konfigurationsdateien sichern.

Eine finanzielle Bewertung der Daten hilft auch festzustellen, ob der Aufwand für die Datensicherung den Wert der Daten übersteigt.

Im Allgemeinen bietet es sich an Datenbestände aus Anwendungsprogrammen zu sichern, bspw. Datenbanken oder CAD-Systeme. Desweiteren sollten die Homeverzeichnisse der Benutzer regelmäßig gesichert werden. Konfigurationsdateien oder modifizierte Programme sollten einmalig bzw. nach Veränderungen gesichert werden.

Um die Datensicherung zu erleichtern, sollten die erzeugten bzw. verwendeten Dateien systematisch abgelegt werden und in Mehrbenutzerumgebungen / Netzwerken die Benutzer in die Struktur der Verzeichnisse (Was wird gesichert?) eingewiesen werden.

Bei der Organisation der Daten sollte man auch deren Sicherung im Blick haben. Hat man Daten, die nicht oft verändert werden, sollte man diese so in Verzeichnissen ablegen, daß die Sicherung vereinfacht wird. Meist umfasst dies Bilder, Fotos, Audiodateien oder Filme, die lediglich ein- oder zweimal auf CD oder DVD gebrannt werden müssen und nicht im regelmäßigen Sicherungszyklus erfasst werden müssen.

Einige Programme legen Daten in Systempfaden und/oder im Homeverzeichnis des Benutzers ab. Werden nur die Homeverzeichnisse gesichert, sollte man Änderungen oder neue Dateien nur im Homeverzeichnis ablegen. Ein Beispiel dafür ist  $\LaTeX$ , das standardmäßig die Pakete unter `/usr/pkg/share/texmf` ablegt. Zusätzlich kann man auch für jeden Benutzer ein eigenes Verzeichnis unter `~/texmf` anlegen. Werden die Homeverzeichnisse regelmäßig gesichert, bietet es sich an die eigenen  $\TeX$ -Pakete im Homeverzeichnis abzulegen.

Weiterhin ist es nicht notwendig Dateien zu sichern, die sich aus anderen Dateien erzeugen lassen. Dies umfasst beispielsweise PDF-Dateien, die mit  $\LaTeX$  erzeugt werden. Es genügt die  $\TeX$ -Dateien zu sichern. Oder Binaries, deren Quellcode gesichert wird, Fotos die für eine Webseite verkleinert und aufbereitet wurden etc. pp.

## 2.6. Sicherung der Sicherungssysteme

Einige Sicherungssysteme, insbesondere solche für Netzwerksysteme wie Amanda oder Bacula, verwenden einen Index. In diesem Index<sup>6</sup> werden alle Metadaten zur Sicherung abgelegt. Also Daten zu „wann wurde welche Datei von welchem Client in welcher Version auf welches Band gesichert“. Dieser Index ist also wichtig wenn man eine Rücksicherung einer bestimmten Version durchführen muss. Ohne den Index kann man zwar in der Regel noch die Bänder von Hand zurückspielen, dies ist aber bei einer großen Anzahl von Bändern recht mühselig. Daher sollte zwingend auch das Sicherungssystem selbst gegen Ausfälle und Verlust gesichert werden. Welche Daten hierbei zu sichern sind, hängt vom verwendeten Softwarepaket ab. Verwendet man eigene Skripte, die bspw. einen Katalog der gesicherten Dateien erstellen, sollte dieser ebenfalls gesichert werden.

## 2.7. Organisation der Sicherung

Hat man ein Netzwerk aus heterogenen Rechnern zu sichern, sollte man bereits bei der Einrichtung bzw. der Administration einige Punkte beachten.

### 2.7.1. Einweisung der Benutzer

Insbesondere in Netzwerken, in denen nur Teile der Clients gesichert werden, ist es wichtig die Benutzer der Systeme über die Strategie zu unterrichten. Dabei ist darauf zu achten daß Anwenderdaten von den Benutzern in Verzeichnissen abgespeichert werden, die auch gesichert werden. Gegebenenfalls muss man hier mit sozialen Maßnahmen (Richtlinien, Abmahnungen, LARTs etc.) nachhelfen.

Dürfen die Benutzer auf den Clients selbständig Programme installieren, muss bereits im Vorfeld dafür gesorgt werden, daß deren Anwendungsdaten ebenfalls mitgesichert werden. Ist dies nicht möglich, muss ein Prozedere installiert werden, daß in solchem Falle die Sicherungsmechanismen anpasst.

### 2.7.2. Cronjob vs. Batchjob

Wenn man einzelne Systeme sichern möchte, kann man dies bspw. nachts per Cron erledigen. Dabei sollte man aber beachten, daß die entsprechenden Skripte robust sind und Toleranzen einplanen. Kommt es nämlich dazu, das sich die Laufzeit der Skripte verschiebt, kann unter Umständen der gesamte Sicherungsalgorithmus fehlschlagen.

Ein Beispiel aus der Praxis:

---

<sup>6</sup>Logdateien bei Amanda, eine Datenbank bei Bacula

In einem Unternehmen sollten vier kleinere Arbeitsgruppenserver mit verschiedenen Unixvarianten gesichert werden. Dazu stand ein älterer Rechner mit Spoolingplatte und Streamer zur Verfügung. Nächtlich sollten die vier Server ihre Daten per dump und SSH-Tunnel auf den Spoolserver schieben, der diese danach auf ein Streamerband schreibt. Da alle Rechner im selben Raum standen, wurden sie über einen einfachen Switch extra für die Sicherung mit TBase100 verkabelt. Ein Cronjob auf den vier Servern startete mit einer Stunde Abstand den Dump-Lauf und auf dem Spoolingserver die endgültige Sicherung auf Band. Da das Backupnetzwerk eben nur für das Backup genutzt wurde, bemerkte niemand in der Firma daß ein Port am Switch massive Hardwareprobleme entwickelte und daher die Sicherung über das Netzwerk extrem langsam wurde. Zu dem Zeitpunkt als der Spoolingserver per Cron begann die Daten auf Band zu schreiben, lieferte der dritte Server seine Sicherung immer noch auf die Spoolingplatte ein. Der Spoolingserver verwendete zwar dump um das Band zu schreiben, so daß die anderen Sicherungen fehlerfrei waren, das Archiv vom dritten Server war aber defekt. Das ganze fiel erst auf, als ein Admin früher als gewöhnlich in der Firma auftauchte und bemerkte daß der Sicherungsprozess noch läuft.

Nachdem der Netzwerkfehler behoben war, wurde das Skript abgeändert. Und zwar so, daß nächtlich per Cron auf dem Spoolingserver ein Shellskript anrief, das nacheinander die Sicherungsläufe auf den Servern anwarf und erst nach erfolgreichem Abschluss dieser Sicherungen die Archive auf Band schrieb. Abschließend wurden die Logdateien aller Dump-Läufe per Mail an die Administratoren weitergeleitet – und von diesen auch gelesen.

### **2.7.3. Homogene Systeme**

Meist ist es zwar nicht möglich alle Systeme homogen aufzusetzen, man sollte sich aber bemühen uniforme Systeme einzurichten und zu benutzen. Dazu gehört auch, daß man bspw. die Systemdienste gleich einrichtet, Benutzerdaten möglichst zentral verwaltet und gleiche Hardware einsetzt.

Dies vereinfacht die Administration des Netzes und damit auch die Sicherung bzw. Rücksicherung im Falle eines Falles.

### **2.7.4. Dokumentation der Administration**

Die normalen administrativen Aufgaben sollten ebenso dokumentiert werden wie das Sicherungssystem selbst. Es ist natürlich nützlich den Arbeitsablauf zu dokumentieren, da man so auch später noch nachvollziehen kann, was wann warum wie gemacht wurde. Aber besonders nach einem Systemausfall und der Systemrestaurierung ist es nützlich, Aufzeichnungen über das System zu haben, da man bestimmt das ein oder andere Detail vergessen und die Daten dazu nicht gesichert hat. Dies ist insbesondere dann wichtig, wenn mehrere Administratoren



in der EDV arbeiten. Man kann beispielsweise Blog-Systeme oder Wikis verwenden, um die Arbeit in Gruppen aufzuzeichnen. Dann darf man aber nicht vergessen, den Datenserver zu sichern :-)

### 2.7.5. Shellskripte statt Handarbeit

Meist kommt es vor, daß man einzelne Handgriffe wirklich manuell erledigt – sei es bspw. das Anlegen neuer Benutzer oder das Warten des Datenbankservers. Meist kann man die Befehle dazu auswändig und gibt sie von Hand ein. Praktischer ist es aber, diese Befehle in ein Shellskript zu gießen und ggf. per Cron ausführen zu lassen. Dies hat den Vorteil, daß die Skripte mitgesichert werden, also nach einem Systemausfall wiederhergestellt werden können. Außerdem kann man ein Skript leicht kommentieren und somit dokumentieren. Im Zweifelsfalle wissen also auch die Kollegen was auf den Maschinen abläuft.

### 2.7.6. Logdateien erstellen und auswerten

Für gewöhnlich können die meisten Sicherungsprogramme Logdateien erstellen. Dies sollte man auch aktivieren – und die erzeugten Protokolle unbedingt lesen. Denn nur so ist sichergestellt, daß alle Sicherungsläufe auch wie erwartet funktionieren.

### 2.7.7. Verifikation der Daten

Wann immer Daten übertragen oder bearbeitet werden, kann dabei etwas schief gehen. Daher sollte jedes Programm, das Daten verifizieren kann, dies auch tun. Die Kompressionsprogramme Bzip2 und Gzip unterstützen mit der Option `-t` die Verifikation der komprimierten Daten.

Ansonsten kann man Daten mit Prüfsummen wie MD5 oder SHA1 (siehe Kap. 8.10) nach der Übertragung testen. Komplette Dateisysteme lassen sich mit `mtree(8)` (Siehe Kap. 8.9) überprüfen und vergleichen.

### 2.7.8. Den GAU erwarten

Im Endeffekt zielt die gesamte Organisation der Datensicherung nur auf einen Tag ab – den des Super-GAUs. Es wird der Tag kommen, an dem Mond und Sonne verschlungen werden, der Weltenbrand beginnt und das Schicksal der Götter besiegelt ist. Alle Rechner werden in Flammen stehen und ihre Daten verlieren – und nur ein kleiner, unbedeutender Systemadministrator mit einem Schrank voll Sicherungsbänder wird das Schicksal wenden können.

Und wehe Ihm er hat nicht alle benötigten Werkzeuge bereitliegen.

---

```
$ gzip netbsd.gz
$ root@balmung {48} gzip -t netbsd.gz ; echo $?
0
$ root@balmung {49} date >> netbsd.gz
$ root@balmung {50} gzip -t netbsd.gz ; echo $?
gzip: input not gzipped (MAGIC0)
gzip: netbsd.gz: uncompress failed
1
$
$ md5 dump.0 > MD5
$ scp dump.0 stefan@backup:/home/backup/
$ ssh stefan@backup md5 /home/backup/dump.0 > MD5.1
$ diff MD5 MD5.1
```

---

Listing 2-1: Daten verifizieren

Wichtig ist es die Konfiguration aller Rechner und deren Software zu kennen. Am besten nicht (nur) als Datei, sondern auch als Ausdruck.

Dann benötigt man alle Installationsmedien für das Betriebssystem und eine Live-CD oder eine Festplatte mit einem eingerichteten NetBSD.

Eine Live-CD kann man mit `pkgsrc/sysutils/mklivecd` selbst erstellen, man sollte dazu einen Kernel verwenden, der alle benötigten Geräte (Laufwerke, Netzwerkkarten, CGD, RAID, LVM, etc. pp.) unterstützt und alle benötigten Pakete (z.B. `mccrypt`, `Amanda`) bereitstellen. Alternativ kann man auch eine kleinere Festplatte mit NetBSD einrichten und alle benötigten Programme einbinden.

### 2.7.9. Alles sichern

Das Sicherungssystem muss nach dem Prinzip „Erstmal alles sichern und den Rest ausschließen“ angelegt sein. In der Praxis heißt dies bspw. daß alle Platten, die nach `/home/` gemountet werden, gesichert werden sollen. Dies ist besonders praktisch, wenn bspw. ein Kollege eine weitere Platte für eine Arbeitsgruppe als `/home/ag/` einbindet, dann aber vergisst diese Platte im Backupskript nachzutragen. Sichert das Skript allerdings alle Partitionen die in der `/etc/fstab` nach `/home/` gemountet werden, wird diese neue Partition automatisch mitgesichert.

### 2.7.10. Testsysteme und -läufe

Es ist allgemein bekannt, das man nicht am Produktivsystem herumschrauben sollte, wenn sich dies vermeiden lässt. Ähnlich wie bei wichtigen Servern sollte

auch für die Sicherungsanlage ein Testsystem bereitgestellt werden, an dem Testläufe und Versuche durchgeführt werden können.

Außerdem sollte regelmäßig eine Inventur der Medienbestände durchgeführt werden, in der neben deren Vollständigkeit auch die Funktionsfähigkeit<sup>7</sup> der Laufwerke und Medien getestet wird.

### **2.7.11. Verschiedene Sicherungskreise**

Aus Datenschutz- oder rechtlichen Gründen oder technischen Gegebenheiten kann es notwendig werden das Netzwerk in verschiedene Sicherungskreise zu unterteilen, die entsprechend behandelt werden.

### **2.7.12. Maximale Archivierungsdauer**

So wie der Gesetzgeber für einige Daten eine Mindestarchivierung vorschreibt, legt er auch bestimmte Obergrenzen fest. So sind beispielsweise Eintragungen in der Personalakte über Disziplinarmaßnahmen nach Eintritt des Verwertungsverbots<sup>8</sup> von Amts wegen zu entfernen und zu vernichten. Das heißt natürlich auch, dass in dieser Frist angelegte Sicherungskopien der Akte zu vernichten sind. Daher kann es sinnvoll sein bspw. die Personalabteilung in einen eigenen Sicherungskreis zu gliedern und auch entsprechend zu archivieren, da man sonst erfolgreich juristisch belangt werden kann.

### **2.7.13. Wochenende und Feiertage beachten**

In Unternehmungen, in denen nicht rund um die Uhr, oder zumindest jeden Tag, gearbeitet wird, wird oftmals nur an den regulären Werktagen gesichert. Dies kann problematisch werden, wenn doch einmal an einem Feiertag oder Wochenende gearbeitet wird. Denn wie es Murphy's Law will, wird es dann garantiert zu einem Datenverlust kommen.

Sind die entsprechenden Rechner auch außerhalb der Arbeitszeiten an, können sie über das Netzwerk gesichert werden. Falls nicht, sollte in den Netzwerkrichtlinien und Benutzerordnungen auf die fehlende Sicherung hingewiesen werden, und Maßnahmen ergriffen werden, die den Benutzern eine Sicherung der Daten ermöglicht. Dies kann bspw. über CD-Brenner, ZIP-Disketten oder USB-Sticks erfolgen. Falls Wechseldatenträger verboten sind, kann man bspw. einen Fileserver bereitstellen, auf den die Mitarbeiter Kopien der zu sichernden Dateien ablegen können.

---

<sup>7</sup>Bänder sollten bspw. mindestens einmal pro Jahr einmal durchgespult werden.

<sup>8</sup>Meist zwei Jahre.

## 2.8. Dokumentation

Es muss sichergestellt werden, daß das gesamte Prozedere auch von anderen Administratoren als dem Implementierenden ausgeführt werden kann. Daher ist es lebenswichtig, eine funktionierende Dokumentation des gesamten Systems zu erstellen. Idealerweise sollte hier der gesamte Entwurf und die gesamte Implementierung des Systems formal dargelegt werden – zusätzlich aber auch eine einfache und leicht verständliche Schritt-für-Schritt-Anleitung zur Rücksicherung von Daten.

Anhand der Dokumentation sollte ein Administrator, der bisher noch nicht mit dem Sicherungssystem gearbeitet hat, in der Lage sein eine Rücksicherung durchzuführen. Es ist unerlässlich mehr als eine Person im Sicherungssystem zu unterweisen, denn schließlich kann diese nicht rund um die Uhr verfügbar<sup>9</sup> sein.

Nützlich ist es, die Einweisung in das Sicherungssystem mit der Dokumentation als Testlauf anzusetzen. Hierbei sollten weitere Administratoren anhand der Dokumentation eine Rücksicherung auf ein Testsystem durchführen, wobei der Entwickler des Systems lediglich beobachtet und nicht eingreift. So kann bequem die Qualität der Dokumentation (Versteht man was dort beschrieben wurde? Fehlen Teile?) und die Funktionstüchtigkeit des Sicherungssystems (Gelingt die Rücksicherung?) überprüfen. Nach Abschluss der Rücksicherung wird eine Manöverkritik durchgeführt, in der gemeinsam Schwächen und Fehler des Systems und der Dokumentation durchgesprochen und abgestellt werden.

Wie hoffentlich schon ersichtlich wurde, ist es wichtig nicht nur einen Administrator in das Sicherungssystem einzuweisen, sondern auch hier mehrere Personen auszubilden. Insbesondere in kleinen bis mittleren Unternehmungen, die keine eigentliche EDV-Abteilung sondern nur einen „Teilzeit-Admin“ haben, ist dies oftmals ein Problem. Lösungen lassen sich dann evtl. über externe Berater finden, die im System eingewiesen und vertraglich entsprechend in den Ablauf eingebunden werden.

## 2.9. Testen, Testen, Testen

Ein Sicherungssystem, das nicht unter realen Bedingungen getestet wurde, kann und darf nicht als funktionierend betrachtet werden. Dies hat mehrere Gründe:

- Der Administrator muss in der Lage sein die Rücksicherung durchzuführen. In der Regel erzeugt ein Systemausfall mit Datenverlust Stress. Wenn dann zusätzlich der Chef das Problem am besten gestern gelöst sehen will, wird der Administrator zusätzlich unter Druck gesetzt. In solch einer Situation muss man aber verständlicherweise einen kühlen Kopf bewahren um die Situation nicht weiter eskalieren zu lassen. Daher ist es unerlässlich,

---

<sup>9</sup>Freigang auf dem Hof, Fütterungszeiten u. Ä.

derartige Situationen und Verfahren in Ruhe durchzuspielen, um Sicherheit im Ablauf zu bekommen. Ist man an den Ablauf der Prozedur gewöhnt, verringert dies den auftretenden Stress und somit Fehlerquellen

- Es ist nicht sichergestellt, daß die Sicherungsstrategie auch wirklich funktioniert. Man kann zwar versuchen im Entwurfsprozess der Strategie Fehlerquellen zu minimieren, kann dies aber, wie in jedem Entwurfsprozess überhaupt, nicht ausschließen. Nur durch Testläufe lässt sich sicherstellen, das die Prozedur funktioniert und auch wirklich die gewünschten Daten gesichert werden. Daher sollte man verschiedene Testläufe durchführen, in denen mögliche Fehler/Situationen betrachtet werden:
  - Rücksicherung von  $n$  einzelnen Dateien
  - Rücksicherung von  $n$  Versionen verschiedener Dateien
  - Rücksicherung eines kompletten Dateisystems (eines Clients) und Vergleich der Rücksicherung mit dem echten System
  - ein komplettes System (Betriebssystem mit Konfiguration plus Anwendungssoftware und Daten) wieder aufsetzen
  - Rücksicherung einer Archivversion zu einem bestimmten Zeitpunkt in der Vergangenheit (*Point-in-Time-Recovery*)
  - Rücksicherung eines Systems obwohl der Backup-Server ausgefallen ist

## 2.10. Verminderung von Ausfällen und Ausfallzeiten

Neben der eigentlichen Datensicherung ist es natürlich erstrebenswert Ausfälle erst gar nicht auftreten zu lassen. Hierzu existieren ebenfalls verschiedene Strategien, wie bspw. „hochverfügbare Systeme“ (*HV* oder engl. *HA*) oder Replikation. Wünscht man hochverfügbare Systeme, muss man in entsprechende Technik investieren. Dies setzt unter anderem redundante Bauteile (Netzteile, Laufwerke), zuverlässige Hardware, RAID-Systeme, entsprechende Stromversorgung und im Betrieb austauschbare Hardware voraus. Auch wenn man so unter Einsatz entsprechender finanzieller Mittel ein hochverfügbares System installiert, ist immer noch eine Datensicherung erforderlich.

Selbst wenn eine Verfügbarkeit von 99,99999% (das heißt maximal 5,3 Minuten Ausfallzeit pro Jahr) erreicht werden kann, ist das System nicht vor Benutzer- und Bedienfehlern oder höherer Gewalt sicher.

Bei einer Spiegelung werden Daten idealerweise sofort (also direkt beim Schreiben) auf ein anderes System übertragen. Dies ist bspw. bei RAID-5 Systemen der Fall, wo Daten auf mindestens drei Festplatten verteilt werden und so bei Ausfall einer Platte die Daten integer bleiben.

Abzuraten ist hierbei dringendst von RAID-0, dem sogenannten „Striping“. Hierbei werden zur Geschwindigkeitssteigerung mindestens zwei Festplatten zu

einer Logischen zusammengefasst. Bei diesem System multipliziert sich auch die Verfügbarkeit<sup>10</sup> beider Festplatten. Da die Verfügbarkeit immer kleiner Null ist, wird die Gesamtverfügbarkeit logischerweise mit jeder weiteren Platte kleiner.

Eine ideale Form der Spiegelung ist die Replikation eines laufenden Systems auf ein identisches, bereitstehendes Ersatzsystem („hot standby“) das bei Ausfall des Originals sofort einspringen kann. Diese Variante kostet natürlich mehr als das Doppelte eines Einzelsystems, außerdem ist sie nur sinnvoll wenn der Nutzen den Aufwand rechtfertigt, man also auf entsprechende Hochverfügbarkeit angewiesen ist. Wichtig ist hierbei vor allem auch eine räumliche Trennung (mindestens eine Brandschutzschleuse) der Systeme. Da beide Systeme in der Regel rund um die Uhr laufen sind sie den üblichen elektrischen Gefährdungen (Stromausfall, Blitzschlag, Kurzschluss) ausgesetzt und müssen dagegen gesichert werden. Diese Variante hat den Vorteil, das bei ausreichendem Schutz des Ersatzsystems dieses das ausgefallene System mit minimalem Zeitaufwand ersetzen kann. Nachteilig sind hierbei natürlich die Kosten.

Besonders im Datenbankenbereich sind schon seit Jahren erfolgreich Replikationsmechanismen etabliert. Für PostgreSQL existieren mehrere Programme, die synchrone oder asynchrone Replikation ermöglichen. Mit einem synchronen Replikationssystem, wie beispielsweise Pgpool (siehe Kap. 7.4.1, Seite 91), werden alle SQL-Anforderungen auf mehrere verteilte Systeme repliziert, so daß der Datenbestand auf mehreren unabhängigen Rechnern vorliegt. Da diese Rechner natürlich räumlich getrennt aufgestellt werden können, lassen sich die Daten vor einem Totalverlust schützen.

## 2.11. Datenrettung

Ist es notwendig Daten nach einem Headcrash oder einem Brand wiederherzustellen, muss man die Daten von den beschädigten Festplatten retten. Hierzu sollte man unbedingt, insbesondere im professionellen Bereich, spezialisierte Datenrettungsunternehmen beauftragen. Diese verfügen über die notwendige Erfahrung und Technologien um beschädigte Medien wiederherzustellen.

Sind Daten versehentlich gelöscht wurden (rm, Partitionierung/Formatierung) können diese u.U. mit Datenrettungsprogrammen (bspw. pkgsrc/sysutils/gpart) wiederhergestellt werden. Dabei sollte man grundsätzlich nicht auf dem betroffenen Medium arbeiten, insbesondere Schreibzugriffe sind zu verhindern. Alle Datenrettungsversuche dürfen nur auf einer 1:1-Kopie (siehe Kap. 5.12, S. 62) des Mediums erfolgen.

Sollen die Daten (trotz der recht hohen Kosten) von einem Datenrettungsunternehmen wiederhergestellt werden, sollte man jeglichen Zugriff auf die Medien unterlassen und die Profis arbeiten lassen.

<sup>10</sup> $Verfügbarkeit = \frac{meantime\ between\ failure}{meantime\ between\ failure * meantime\ to\ repair}$

## 3. Sicherungsvarianten

### 3.1. Kompletbackup

Ein Kompletbackup sichert alle relevanten Daten. Dies ist in der Regel sehr einfach zu bewerkstelligen, außerdem ist das Zurückspielen der Daten ebenfalls recht einfach. Nachteile sind dabei vor allem die Dauer der Sicherung und der benötigte Platz, da auch Daten gesichert werden, die seit dem letzten Backup nicht verändert wurden.

### 3.2. Differentielles Backup

In Anlehnung an das mathematische  $\delta$  auch Delta-Methode genannt. Um Platz und Zeit zu sparen werden hierbei nur Daten gesichert, die seit dem letzten Kompletbackup geändert wurden. Bei der Rücksicherung muss allerdings die Reihenfolge der Bänder beachtet werden, im allgemeinen benötigt man hierzu die letzte Komplettsicherung<sup>1</sup> und das letzte Tagesband.

### 3.3. Inkrementelles Backup

Hier werden nur die Dateien gesichert, die sich seit dem letzten Inkrementalbackup geändert haben. Bei der Rücksicherung muss ebenfalls die Reihenfolge beachtet werden, dafür ist bei der Sicherung noch weniger Platz und Zeit als beim Differentialbackup erforderlich.

### 3.4. Dump-Level

Für gewöhnlich unterstützen ordentliche™ Sicherungsprogramme den Einsatz sogenannter Level, die mittels einer Zahl die zu sichernden Dateien angeben. Auf dem Level  $n$  werden alle Dateien gesichert, die seit der Sicherung mit  $n - 1$  verändert wurden.

Vorteil ist hierbei die Zeit- und Platzersparnis, Nachteil ist die aufwändigere Rücksicherung.

---

<sup>1</sup>meist montags oder freitags erstellt

Man kann hierbei aber auch die Methoden mischen, so ist es beispielsweise praktikabel montags eine Komplettsicherung mit Level 0 durchzuführen und werktags, also dienstags – freitags, mit Level 1 die Arbeitsdaten der Woche zu sichern. Da es auch am Wochenende zu Arbeitseinsätzen kommen kann<sup>2</sup>, können am Sonnabend und Sonntag mit Level 2 die Tagesdaten gesichert werden.

### 3.5. Beispielstrategie

Eine einfache Beispielstrategie für einen einzelnen Rechner mit Leveln:

- Am Montag existiert 1 GB an Daten
- es werden jeden Tag 100 MB Daten erzeugt
- am Sonntag liegen 1,6 GB Daten vor
- Sicherungen erfolgen in der Nacht, vor Arbeitsbeginn
- Sonntag Mittag sei eine Rücksicherung vom Band notwendig

Tabelle 3.1 zeigt die verwendeten Duplelevel, die Tabellen 3.2, 3.3 und 3.4 zeigen die anfallende und gesicherte Datenmenge.

	Mo	Di	Mi	Do	Fr	Sa	So
Komplett	0	0	0	0	0	0	0
Differentiell	0	1	1	1	1	1	1
Inkrementell	0	1	2	3	4	5	6

Tabelle 3.1.: Wochensicherung mit Leveln

	Mo	Di	Mi	Do	Fr	Sa	So
Komplett	0	0	0	0	0	0	0
Tagesration	1 GB	1,1 GB	1,2 GB	1,3 GB	1,4 GB	1,5 GB	1,6 GB
Gesamt	1 GB	2,1 GB	3,3 GB	4,6 GB	6,0 GB	7,5 GB	<b>9,1 GB</b>

Tabelle 3.2.: Datenmenge einer Komplettsicherung

Die Komplettsicherung sichert jeden Tag alle Daten. Die Rücksicherung erfolgt vom letzten Band, also dem Sonntagsband.

Differentiell sichert man jeden Montag alle Daten und dienstags bis freitags auf Level 1. Die Rücksicherung erfolgt erst mit dem letzten Montagsband und anschließend dem letzten Tagesband.

<sup>2</sup>Vorausgesetzt das Wochenende ist kein normaler Arbeitstag.



	Mo	Di	Mi	Do	Fr	Sa	So
Inkrementell	0	1	1	1	1	1	1
Tagesration	1 GB	0,1 GB	0,2 GB	0,3 GB	0,4 GB	0,5 GB	0,6 GB
Gesamt	1 GB	1,1 GB	1,3 GB	1,6 GB	2,0 GB	2,5 GB	<b>3,1 GB</b>

Tabelle 3.3.: Datenmenge einer Inkrementellsicherung

	Mo	Di	Mi	Do	Fr	Sa	So
Differentiell	0	1	2	3	4	5	6
Tagesration	1 GB	0,1 GB	0,1 GB	0,1 GB	0,1 GB	0,1 GB	0,1 GB
Gesamt	1 GB	1,1 GB	1,2 GB	1,3 GB	1,4 GB	1,5 GB	<b>1,6 GB</b>

Tabelle 3.4.: Datenmenge einer Differentiellsicherung

In der inkrementellen Sicherung sichert man jeden Montag alle Daten und wochentags inkrementell, also nur Veränderungen zum Vortag. Die Rücksicherung erfolgt hier in der Reihenfolge aller Bänder von Montag bis Sonntag. Nachteil der Strategie ist, daß Dateien in der Regel nur auf einem einzigen Band gesichert sind, was bei Verlust des Bandes dem Verlust aller Daten des Tages und der darauffolgenden Tage gleichkommt.

Daher gibt es noch eine andere, platzsparende Strategie, die an die „Türme von Hanoi“ angelehnt ist.

Diese Strategie (siehe Tab. 3.5) verteilt einzelne Dateien auf mehrere Bänder, wobei bei diesem Algorithmus aber am wenigsten Platz verschwendet wird. Für einen gesamten Monat, lässt sich der Algorithmus mit Level-1 Sicherungen an den folgenden Montagen erweitern. Vorteil der Strategie ist natürlich hierbei die

	Mo	Di	Mi	Do	Fr	Sa	So
	0	3	2	5	4	7	6
	1	3	2	5	4	7	6
	1	3	2	5	4	7	6
	1	3	2	5	4	7	6

Tabelle 3.5.: Türme-von-Hanoi-Algorithmus

Sicherung der Dateien auf mehr als ein Medium, ohne jedesmal alles sichern zu müssen.

Die Verteilung der Dateien auf die Bänder wird in Tabelle 3.6 illustriert:

Wochentag	Daten auf dem Tagesband
Montag	Montag
Dienstag	Montag & Dienstag
Mittwoch	Montag & Dienstag & Mittwoch
Donnerstag	Mittwoch & Donnerstag
Freitag	Mittwoch & Donnerstag & Freitag
Sonnabend	Freitag & Sonnabend
Sonntag	Freitag & Sonnabend & Sonntag
Montag	1. Montag - 2. Montag
Dienstag	2. Montag & 2. Dienstag
...	

Tabelle 3.6.: Datenverteilung im Hanoi-Algorithmus

### 3.6. Komprimierung und Verschlüsselung

Oftmals ist es nützlich ein Archiv zu komprimieren um Platz zu sparen. Jedoch sei hierzu angemerkt, daß bei Archivfehlern ein Dekomprimieren der gepackten Archive oftmals nicht mehr möglich ist. Daher sollte man die Kosten und Nutzen abwägen und die gesicherten Archive jedesmal verifizieren lassen. Ebenso ist es sinnvoll nur gängige Komprimierungsprogramme zu verwenden, um später wieder an die Daten heranzukommen.

Aktuelle Bandlaufwerke unterstützen in der Regel Hardwarekomprimierung. Das heißt daß das Laufwerk vor dem eigentlichen Schreiben der Daten diese in einem eigenen Chip komprimiert. Dieses Verfahren hat den Vorteil die sichernde Maschine nicht weiter zu belasten.

Ob man Komprimierungsverfahren einsetzen sollte, hängt von der entstehenden Last bzw. Einsparung (Netzwerk ./ CPU ./ Bandbedarf) und von der Zusammensetzung der Daten ab. Sind die Daten in der Mehrzahl (natürlichsprachige) Texte (Logdateien, Textverarbeitung, Datenbanken mit Texten) ist eine recht große Kompressionsrate gegeben. Bereits komprimierte (z. B. JPEG, MP3, AVI) oder verschlüsselte Datentypen hingegen lassen sich nicht weiter komprimieren.

Es ist sinnlos ein verschlüsselndes Dateisystem einzusetzen und unverschlüsselte Backups im Schrank liegen zu haben, daher sollten auch Backups verschlüsselt werden. Dies gilt insbesondere dann, wenn die Sicherungsbänder ausgelagert werden sollen.

Zur Komprimierung eignen sich die Standardprogramme gzip und bzip2<sup>3</sup>, da diese relativ weit verbreitet sind und bisher zuverlässig arbeiten.

<sup>3</sup>bzip2 erzeugt zwar im Allgemeinen eine bessere Komprimierungsrate, erzeugt dabei aber auch höhere Systemlast und Laufzeiten

Verschlüsseln kann man die Archive sicher mit OpenSSL, GnuPG oder dem symmetrischen Verschlüsselungsprogramm mdecrypt, welches verschiedene Algorithmen wie Rijndael, 3DES oder Panama unterstützt.

Werden die Archive nicht mittels einer Pipe direkt verschlüsselt auf Datenträger geschrieben, müssen die unverschlüsselten Dateien sicher gelöscht („gewipet“) werden. Dazu gibt es die Option `-P` für `rm`, die Dateien vor dem Löschen dreimal überschreibt. Diese Option genügt aber nicht den allgemein anerkannten Regeln der Technik. Daher sollte hierfür dringend ein Programm verwendet werden, das zumindest den US DoD 5220.22-M ECE oder Peter Gutmanns Standard implementiert. In `pkgsrc` befindet sich dazu `pkgsrc/security/destroy` oder `pkgsrc/sysutils/wipe`. Empfehlenswert ist es allerdings, die unverschlüsselten Archive in eine mit CGD verschlüsselte Partition zu schreiben und danach ebenfalls zu wipen. Hierzu kann man auf einzelnen Rechnern eine hinreichend große `/tmp`-Partition erstellen, die mit CGD verschlüsselt ist.

Verwendet man GnuPG zum Verschlüsseln der Archive, sollte man auf asymmetrische Verschlüsselung verzichten und Symmetrische einsetzen. Denn für die Entschlüsselung benötigt man sonst wieder den GnuPG-Schlüssel – was das Archiv unbrauchbar macht, wenn der nicht mehr existiert.

---

```
# dump -0a -f - /etc/ | bzip2 > dump.bz2
# gpg -a -c dump.bz2 && destroy -f -s 7 t dump.bz2
# dump -0a -f - /etc/ | bzip2 | openssl aes-256-ecb -out dump.bz2.enc -e -salt
# openssl aes-256-ecb -in dump.bz2.enc -d -salt | bunzip2 - | restore -r -f -
```

---

Listing 3-1: Archive komprimieren und verschlüsseln

Listing 3-1 erzeugt einen Dump, der sofort per `bzip2` komprimiert wird. Anschließend wird der dump mit GnuPG symmetrisch verschlüsselt und mit `destroy` sicher gelöscht. Zeile drei erzeugt einen Dump der sofort per Pipe an `bzip2` zum komprimieren und OpenSSL zum Verschlüsseln geschickt wird. Die vierte Zeile ist des Gegenstück zur Dritten, denn hier wird der verschlüsselte und komprimierte Dump entschlüsselt, dekomprimiert und an `restore` geschickt.

### 3.7. Medien

Als Medium eignet sich prinzipiell alles was am Markt erhältlich ist, jedoch sollte man einige Vorüberlegungen treffen.

#### 1. Verlässlichkeit

Die Medien müssen zuverlässig sein und auch längere Archivierungsperioden problemlos überstehen. Außerdem müssen Lesegeräte noch verfügbar sein, um die Sicherungsmedien einlesen zu können. Dies ist besonders wichtig,

wenn Medien über mehrere Jahre archiviert werden müssen. Hier sollte man regelmäßig (bspw. alle sechs Monate) zusammen mit der Inventur prüfen ob die ältesten Medien noch lesbar sind und ggf. die Medien austauschen. Im allgemeinen muss man bei der Archivierung spätestens alle Acht bis Zehn Jahre eine komplette Migration der Medien durchführen, da die Medien und Lesegeräte veraltet sind.

## 2. Geschwindigkeit

Das Sicherungssystem muss natürlich so schnell wie möglich sein. Dabei sollte man aber auch die Kapazitäten des Netzwerks und des Sicherungsservers beachten, denn es ist übertrieben, einen 50-MBit-Streamer einzusetzen, wenn die Sicherung in einem 10-MBit-Netz erfolgt.

Ein weiterer Faktor, der die Geschwindigkeit des Sicherungslaufwerks beeinflusst, ist ein Bandwechsel. Wenn ein Sicherungslauf auf mehrere Bänder verteilt werden muss, senkt die Wartezeit auf das neue Band den Gesamtdurchsatz des Systems.

## 3. Dauer der Rücksicherung (sog. *Time to Data*)

„Wie lange benötige ich um bei einer Rücksicherung an bestimmte Daten zu kommen?“

Hierbei spielen verschiedene Faktoren eine Rolle, wie Zugriffszeit der Medien aber auch die Organisation der Archivierung selbst.

Mehr als 90% aller Rücksicherungen betreffen nur einzelne Dateien, die in älteren Versionen restauriert werden müssen. Der gesamte Zeitaufwand der Operation besteht daraus, die richtigen Medien mit der gewünschten Version zu finden, einzulegen und die Dateien zurückzuspielen. Verwendet man hierzu ein automatisiertes System (Index, Bandwechsler) geht dies natürlich wesentlich schneller als wenn man die Daten von Hand zurückspielen muss. Ist man auf besonders schnelle Zugriffszeiten angewiesen, sollte man zur Sicherung hierarchische Speichersysteme<sup>4</sup> verwenden.

## 4. Kapazität

Die anfallenden Datenmengen müssen auf möglichst wenige Medien verteilt werden. Verwendet man einen einfachen Streamer, ist es besonders wichtig daß alle Daten auf ein Band passen, denn niemand möchte nachts um drei neben dem Streamer Wache schieben.

Beachten sollte man auch das eine steigende Kapazität die Zeit zur Rücksicherung anhebt, denn ein 10GB Band lässt sich schneller durchspulen als ein 100GB Band.

---

<sup>4</sup>Hierbei werden verschiedene Medientypen mit unterschiedlichen Zugriffszeiten eingesetzt. Häufig benötigte Dateien werden auf schnellen Medien gesichert (Festplatte, Magneto-Optische Medien), während ältere bzw. seltener benötigte Daten auf entsprechend langsameren aber größeren Medien (Magnetbänder) gesichert werden. Diese Methode war früher, als Festplatten noch in Megabyte vermessen wurden, sehr beliebt und wird heute noch in einigen sehr datenintensiven Umgebungen (Grafik-/Videobearbeitung) eingesetzt.

Aus betriebswirtschaftlicher Sicht ist eine Betrachtung der Kosten unerlässlich, denn es ist unnötig einen 100GB-Streamer mit entsprechend großen und teuren Medien anzuschaffen, wenn täglich nur 5GB Daten anfallen. Andererseits sollte man auch nicht zu kurz planen, da die Hardware wenigstens die vier Jahre bis zur Abschreibung im Einsatz bleiben sollte.

#### 5. Kosten

Dabei darf das System üblicherweise nichts Kosten. Denn wozu braucht man schon ein Backupsystem, wenn doch alles prima läuft?<sup>5</sup>

Disketten, ZIP-Medien, CD/DVD und Festplatten sind weit verbreitet, billig und von daher gut zur Rücksicherung geeignet. Nachteilig wirkt sich hierbei allerdings die relativ geringe Integrität und die kurze Lebensdauer der Datenträger aus. Relativ kostspielig und langsam, aber dennoch hervorragend geeignet aufgrund hoher Kapazitäten und exzellenter Integrität und Lebensdauer sind Magnetbänder.

Selbstverständlich bietet es sich auch hier wieder an verschiedene Medien zu nutzen und zu mischen, z. B. ZIP-Disketten oder CDRW/DVDRW für tägliche Sicherungen.

Festplatten können aufgrund der hohen Kapazität und Geschwindigkeit auch zur Datensicherung eingesetzt werden, sollten dann aber als Wechselpalte ausgeführt sein und außerhalb der Sicherungsläufe offline gelagert werden.

Medien	Verfügbarkeit	Geschwindigkeit	Haltbarkeit	€/GB
DVD+RW	++	+	-	0,3€
USB-Stick	++	+	+	70,00€
ZIP250	o	-	o	32,00€
Platten	++	++	-	0,7€
Bänder	+	o	++	1-5,00€

Tabelle 3.7.: Vergleich von Sicherungsmedien

### 3.7.1. Organisation und Lagerung der Medien

Bei großen Datenmengen und niedrigen Migrationszyklen, sowie der Archivierung der Medien, ist eine ausgefeilte Organisation der Lagerung notwendig.

Zuerst ist die Lagerung der Medien ihren physikalischen Bedürfnissen anzupassen. Für gewöhnlich befinden sich in deren Spezifikation Angaben zur besten Luftfeuchtigkeit, Umgebungstemperatur, Luftdruck und so weiter und sofort. Insbesondere magnetische Datenträger sollten vor starken

---

<sup>5</sup>man Zynismus

Temperaturschwankungen geschützt werden. Bänder müssen aufrecht stehend gelagert werden, da sich sonst das Magnetband lockern kann – und niemand will Bandsalat bei einer Rücksicherung erleben.

Physikalische Sicherungen (gegen Einbruch) und Brandschutzeinrichtungen sind ebenso obligatorisch wie eine restriktive Zugangspolitik.

Da die Anzahl der Medien mit der Zeit wächst und die Archivierung komplexer wird, bietet sich ein datenbankengestütztes Inventarisierungssystem an. Hierzu sollten mindestens folgende Attribute aufgenommen werden:

- Primärschlüssel
- Name
- Zweck
- Medientyp
- gesicherte Dateien inkl. Datum / Version, Eigentümer, Attribute
- Lagerort

Der Primärschlüssel dient der Identifikation des Mediums und kann bspw. eine laufende Nummer sein, sinnvoller ist es aber in ihm Informationen zu codieren (bspw. Datum, gesicherte Daten, IP-Adresse, Level oder ähnliches). Der Primärschlüssel muss mindestens auf jedem Medium und der entsprechenden Hülle angebracht werden. „Name“ kann ggf. verwendet werden, wenn man die Medien nicht über den Primärschlüssel ansprechen möchte. Zusätzlich muss ebenfalls der Zweck, Medientyp und der Lagerort der Medien katalogisiert werden.

In der Datenbank werden auch die Metadaten der gesicherten Dateien erfasst. Dies ist notwendig, um im Falle einer Rücksicherung das passende Sicherungsband zu finden.

Für große Datenmengen existieren auch automatische Bandwechseleinheiten, die die Medien und Hüllen mit einem Strichcode etikettieren können. In Verbindung mit einem einfachen Handscanner ist dies ideal um größere Medienbestände zu inventarisieren und zu verwalten.

### **3.8. Prüfliste zur Strategie**

- \* **Wer** ist verantwortlich?
- \* **Wie** wird gesichert?
- \* **Wann** wird gesichert?
- \* **Was** wird gesichert?
- \* **Welche** Medien und Geräte werden verwendet?
- \* **Wo** werden die Medien gelagert?
- \* **Wie** lange werden die Medien gelagert? .
- \* **Wie/Wann** wird die Integrität der Sicherung geprüft?
- \* **Welche** Backupstrategie? (Wann welche Level?)
- \* **Wo** ist die Dokumentation zur Backupstrategie?

Tabelle 3.8.: Prüfliste zur Strategie





**Teil II.**

## **Sicherung einzelner Systeme**



## 4. Programme automatisieren

*„A computer program does what you tell it to do,  
not what you want it to do.“*

Greer's Third Law

### 4.1. Zeitgesteuert

NetBSD verfügt, wie jedes anständige Unix, über eine Cron-Implementierung. Cron fungiert sozusagen als Zeitschaltuhr und ermöglicht dem System Programme zeitgesteuert auszuführen. Jeder Benutzer verfügt über einen eigenen crontab(5), in dem die Programm und Startzeitpunkte konfiguriert werden können. um den crontab mit vi zu editieren genügt `crontab -e`.

cron erwartet die Einträge in einem einfachen Tabellenformat mit 6 Einträgen:

1. Minute (0-59)
2. Stunde (0-23)
3. Tag (1-31)
4. Monat (1-2)
5. Wochentag (0-7)<sup>1</sup>
6. Programm

Zahlenwerte können als Liste (0, 3, 5), Intervall (1 – 9) oder gemischt (0 – 9, 11, 13, 17 – 31) angegeben werden.

Der erste Eintrag in Listing 4-1 reinigt die PostgreSQL-Datenbank alle Viertelstunde, der Zweite sichert die Datenbank jeden Tag um 00:20 in eine Datei. Eintrag drei und vier sichern /home/ mittels `dump(8)` auf das Magnetband, wobei montags Level 0 und dienstags bis freitags Level 1 verwendet wird.

Problematisch ist bei cron, das absolute Zeitwerte angegeben werden müssen, der Rechner zu dem Zeitpunkt also laufen muss. Abhilfe schafft hier das Programm Anacron, welches aus `pkgsrc`<sup>2</sup> installiert werden kann.

Anacron verwendet anders als cron keine absoluten Zeitangaben sondern Frequenzen. Man bestimmt also nicht wann genau ein Programm laufen soll,

---

<sup>1</sup>0 und 7 sind Sonntag, 1 Montag ... 6 Sonnabend

<sup>2</sup>`pkgsrc/time/anacron`

---

```
0,15,30,45 * * * * /usr/pkg/bin/vacuumdb
                -Uoperator -f -z woerterbuch

20 0 * * * /usr/pkg/bin/pg_dump -Fc -Uoperator
        woerterbuch > /home/pg/wb`date +%y%m%d`

30 0 * * 1 /sbin/dump -0 -au /home/
30 0 * * 2-5 /sbin/dump -1 -au /home/
```

---

Listing 4-1: Cronjobs um PostgreSQL zu sichern

sondern an welchen Tagen und nach wieviel Minuten Laufzeit des Systems es gestartet wird.

Konfiguriert wird es nach der Installation über `/usr/pkg/etc/anacrontab` in folgender Weise:

1. Frequenz in Tagen
2. Verzögerung in Minuten
3. ID des Eintrags, ist Primärschlüssel für *alle* Einträge
4. auszuführendes Programm

---

```
SHELL=/bin/sh
PATH=/bin:/sbin:/usr/bin:/usr/sbin
HOME=/var/log

#days delay id command
1 5 daily /bin/sh /etc/daily
7 15 weekly /bin/sh /etc/weekly
30 30 monthly /bin/sh /etc/monthly
```

---

Listing 4-2: Anacrontab-Beispiel

Die ersten drei Zeilen definieren einige Variablen, die letzten drei die auszuführenden Programme. Der Eintrag „daily“ wird hier täglich ausgeführt, es wird aber fünf Minuten nach Start des Rechners gewartet. Die anderen beiden Einträge werden alle sieben bzw. 30 Tage, also wöchentlich bzw. monatlich ausgeführt.

## 4.2. Ablaufgesteuert

Programme lassen sich nicht nur zeit- sondern auch ablaufgesteuert aufrufen. Eine einfache Möglichkeit ist es `/etc/rc.local` dazu zu verwenden, Programme beim Systemstart aufzurufen. Dazu ist es lediglich notwendig die Befehle an `/etc/rc.local` anzufügen, oder für elaboriertere Shellskripte den `/etc/rc.d/-`Mechanismus zu verwenden.

Ebenfalls möglich ist es, beim Herunterfahren des Systems ein Skript ausführen zu lassen. Hierzu muss lediglich „KEYWORD: shutdown“ im betreffenden Skript gesetzt sein.

Um Jobs später ausführen zu lassen, kann man die Programme `at(1)` oder `batch(1)` verwenden. Soll ein Programm zu einer bestimmten Zeit ausgeführt werden, verwendet man `at` und übergibt einen Zeitstempel sowie die auszuführenden Programme. Soll das Programm in Abhängigkeit von der Systemlast gegebenenfalls verschoben werden, kann man `batch` einsetzen. Es erwartet genau wie `at` einen Zeitstempel und die auszuführenden Programm, führt diese aber erst aus, wenn die Systemlast unter 1,5<sup>3</sup> gesunken ist.

---

```
$ at 10am Jul 31
tar cf /root/etc.tar /etc
Job 1 will be executed using /bin/sh
$ batch 1pm tomorrow
dump 0a -f /root/etc.dump /etc
Job 2 will be executed using /bin/sh
$
```

---

Listing 4-3: Jobs zur späteren Ausführung vorbereiten

---

<sup>3</sup>Oder ein anderer Wert, der an `atrun(8)` übergeben wurde.



# 5. Sicherung einzelner Systeme

## 5.1. Tägliche Sicherungsmechanismen im Basissystem

NetBSD verfügt mit `/etc/daily` und `/etc/security` über zwei Shellskripte, die nächtlich von cron gestartet werden. Sie führen verschiedene Wartungsaufgaben aus und prüfen sicherheitsrelevante Systemeinstellungen. Konfiguriert werden die Skripte über `/etc/daily.conf` bzw. `/etc/security.conf`. Es empfiehlt sich diese Skripte ausführen zu lassen und über die Option `run_security` in `/etc/daily.conf` auch `/etc/security` aufzurufen.

In `/etc/security.conf` finden sich einige Optionen, die der Sicherheit des Systems dienen. Unter anderem sorgt `check_disklabels` dafür, das die Ausgaben von `disklabel` und `fdisk`<sup>1</sup> nach `/var/backups/work/` kopiert werden. `check_pkgs` sichert eine Liste der installierten Pakete nach `/var/backups/work/pkgs`, `check_changelist` vergleicht die in `/etc/changelist` definierte Liste an Dateien mit ihren jeweiligen Sicherungskopien in `/var/backups/`. Empfehlenswert ist es ebenfalls, die Option `backup_uses_rcs` zu setzen, da die Sicherungskopien somit im RCS vorgehalten werden und damit jede Version verfügbar ist.

Zusätzlich hält `/etc/security` in `/var/backups/etc` ein RCS-Repository verschiedener Konfigurationsdateien aus `/etc/` vor, die somit bei Verlust oder Beschädigung ersetzt werden können.

## 5.2. Das Basissystem sichern

Um einen einzelnen Rechner, bspw. einen Heimrechner, zu sichern, kann man einem einfachen Schema folgen:

1. NetBSD installieren
2. NetBSD konfigurieren
3. `pkgsrc` einrichten und Pakete installieren
4. Sicherung des Systems mit `dd(1)` oder Ghost 4 Unix (Siehe Listing 5-1)
5. Systemkonfiguration sichern (Siehe Listing 5-2)
6. Regelmäßige Sicherung der Daten und Konfigurationsdateien

---

<sup>1</sup>Sofern auf der Plattform verfügbar

7. erneute Sicherung des Systems nach System- oder Paketaktualisierungen  
(Wie Punkt 4)

Ob man das System erst sichert, nachdem alle Pakete installiert wurden, liegt im Ermessen des Administrators und den verfügbaren Ressourcen. Passt das Festplattenimage nicht mehr auf des Sicherungsmedium, sollte das System gesichert werden, bevor alle Pakete installiert sind.

Ob man die Pakete sichern muss, entscheidet sich in der Administration des Systems. Setzt man bspw. nur Stable-Releases für PCs ein, sind in der Regel Binärpakete auf ftp.netbsd.org verfügbar. Daher kann man hier auf die Sicherung der Pakete verzichten, da sie eben auf den NetBSD-Servern und Spiegeln verfügbar sind. Ebenso kann man eigene Binärpakete erstellen und diese getrennt vom Basissystem auf eigene Medien sichern. Diese Vorgehensweise bietet sich an, wenn man Current-Releases und/oder Current-Pkgsrc einsetzt. Man installiert mit `make package` die gewünschten Pakete und lässt sie danach als Binärpaket standardmäßig unter `pkgsrc/packages/All` ablegen. Dieses Verzeichnis kann man bspw. per NFS importieren oder die gesammelten Pakete auf DVD brennen, um sie nach der Restauration schnell wieder einspielen zu können.

Um nach der Einrichtung des Systems die Systemplatte mit dem Betriebssystem zu sichern, verwendet man `dd(1)` oder `g4u`, siehe Kapitel 5.13 bzw. Listing 5-1.

---

```
g4u# uploaddisk benutzer@ftp.server.local Image-wd0.gz wd0
```

---

Listing 5-1: Betriebssystem mit `g4u` sichern

Ist das System inklusive aller Pakete eingerichtet, muss die Sicherung der Anwenderdaten und der Konfigurationsdateien betrachtet werden. Im Allgemeinen kommt es vor, daß sich Konfigurationsdateien im System ändern. Es ist nun nicht notwendig, das gesamte System zu sichern, sondern es reicht meist die Konfigurationsdateien zu sichern. Da diese in der Regel nicht besonders groß sind, kann man sie mit den Anwendungsdaten zusammen sichern, indem man bspw. vor dem dump der Homeverzeichnisse eine Tarball mit allen wichtigen Systemverzeichnissen anlegt. Die Verzeichnisse umfassen:

- `/etc`
- `/usr/pkg/etc`
- `/var/cron`
- `/var/log`
- `/var/backups`
- `/var/db`
- `/var/yp`



- /root/

Außerdem sollte man noch folgende Informationen über das System sammeln (und idealerweise ausdrucken):

- dmesg
- fdisk für alle Platten
- disklabel für alle Platten
- ggf. Kernel-Konfiguration
- pkg\_info | sort > pkgs-list

Die Befehle in Listing 5-2 packt man am einfachsten in ein Shellskript, das man vor jedem Dumplauf ausführen lässt.

---

```
$ dmesg > /home/back/system/dmesg
$ pkg_info | sort > /home/back/system/pkgs-list
$ for i in wd0 w1 wd2 wd3 sd0 sd1 sd2 sd3 sd4
> do disklabel $i >> /home/back/system/disks
> fdisk $i >> /home/back/system/disks
> done
$ config -x /netbsd > /home/back/system/MYKERNEL
# tar cpf - /etc /usr/pkg/etc /var/cron /var/log /var/backups \
    /var/db /var/yp /root/ | gzip > konf.'date +%y%m%d'.tgz
# gzip -t konf.'date +%y%m%d'.tgz
```

---

Listing 5-2: Konfigurationsdateien sichern

Für die Sicherung der Daten verwendet man dump(8) mit einem Wochenschema (siehe Tab. 5.1) auf mindestens zwei Medienzyklen. Man kann diese bspw. in gerade und ungerade Kalenderwoche unterteilen, besser wären aber drei oder sogar vier Medienzyklen. Ist die Datenmenge nicht zu groß um ein Medium zu sprengen, was bei Heimrechnern oft der Fall ist, kann man auch auf verschiedene Dumplevel verzichten und täglich ein Level 0 Dump erzeugen. Dies vereinfacht die Rücksicherung und erhöht die Zuverlässigkeit der Sicherung, da nur ein Medium notwendig ist.

Woche	Mo	Di	Mi	Do	Fr	Sa	So
1.	0	1	1	1	1	2	2
2.	0	1	1	1	1	2	2

Tabelle 5.1.: Dumplevel für einen Einzelrechner

Im Anhang (A-1, A-2 und A-3) findet sich ein Shellskript, das ich einsetze um einen einzelnen Rechner zu dumpen.

## 5.3. Filesystem-Snapshots

Ein Snapshot erzeugt ein Abbild eines Dateisystems zu einem gegebenen Zeitpunkt. Dieses Abbild kann bspw. mit `dump` gesichert werden, während das eigentliche Dateisystem wieder im Produktiveinsatz ist. Somit werden Ausfallzeiten von Dateisystemen minimiert. Die Erstellung eines Snapshots dauert nicht einmal eine Sekunde, dabei werden alle datenverändernden Prozesse gestoppt, die Blöcke synchronisiert und der Snapshot erstellt. Anschließend werden die gestoppten Prozesse fortgesetzt. Das Dateisystem ist somit zur Erstellung des Snapshots so konsistent wie nach einem gewöhnlichen `umount`.

### 5.3.1. Praktischer Einsatz

Ab NetBSD 2.0 sind Snapshots im GENERIC-Kernel verfügbar.

Um einen Snapshot zu verwenden, muss mit `fssconfig(8)` das entsprechende Pseudogerät zum Dateisystem konfiguriert werden. Dies geschieht mit: `fssconfig -c fss0 /home home0.fss` wobei folgende Optionen gelten:

- \* **fss0** ist das zu verwendende Pseudogerät
- \* **/home** ist das Quellverzeichnis
- \* **home0.fss** ist eine Datei, in die alle Änderungen am Quellverzeichnis nach dem Snapshot geschrieben werden.

Listing 5-3: `fssconfig`-Optionen

Das Quellsystem ist nun ab dem Zeitpunkt des Snapshots als `/dev/fss0` wie ein normales, schreibgeschütztes Dateisystem verfügbar, kann also bspw. gemounted werden. Wird das Quellverzeichnis verändert, bspw. eine Datei hineinkopiert, wird diese Änderung nicht nach `/dev/fss0` durchgeschleift, sondern nach `home0.fss` geschrieben. Sollen die Änderungen nicht mitgeschrieben werden (was z. B. beim Einsatz zur Datensicherung der Fall wäre), wird die Option `-x` übergeben.

`fssconfig -v1` zeigt alle Pseudogeräte und ihre Konfiguration an. `fssconfig -u fss0` deaktiviert das Pseudogerät, wobei die Datei für die Änderungen zurückbleibt.

Ist `/dev/fss0` konfiguriert, kann es ganz normal (nur lesbar) eingebunden werden.

### 5.3.2. Sicherung eines Snapshots

Snapshots sind ein ideales Werkzeug um Dateisysteme mit `dump` zu sichern. Die Arbeitsschritte sind wie folgt:

1. Snapshot erstellen
2. Snapshot sichern
3. Snapshot abschalten

```
# fssconfig -x -c fss0 /home /usr/fss0.log
# dump -0au -X /dev/fss0
# fssconfig -u fss0
```

---

Listing 5-4: Snapshot mit Dump sichern

Möchte man andere Sicherungsprogramme, bspw. Amanda mit `gtar` verwenden, kann man stattdessen an zweiter Stelle den Snapshot mounten (`mount -r /dev/fss0 /mnt && tar -c -f foo.tar /mnt`) und als normales Dateisystem sichern.

## 5.4. `dump(8)/dump_lfs(8)` und `restore(8)`

Dump und Restore sind historisch gewachsen die bedeutendsten Backupprogramme unter Unix. Sie arbeiten hierbei auf Blockebene, also unterhalb der Dateisicht. Dies heisst das Dump einen Verzeichnisbaum oder ein Dateisystem<sup>2</sup> komplett mit allen Eigenschaften sichert und nicht auf Dateiebene Dateien kopiert. So ist es quasi möglich eine Partition zu spiegeln und zurückzusichern oder auch auf andere Rechner zu übertragen.

Da Dump speziell an das Dateisystem angepasst ist, kann `dump(8)` nur FFS sichern, für LFS gibt es `dump_lfs(8)`. Die beiden Programme unterscheiden sich aber in der Bedienung nicht. Restore entpackt einen dump aber dateisystemunabhängig, kann also auch bspw. auf einem NFS-Verzeichnis arbeiten.

Dump unterstützt inkrementelle Backups durch die Verwendung sogenannter Dumplevels, dabei werden bei jedem Backup in der Datei `/etc/dumpdates` die entsprechenden Daten eingetragen und können so später ausgelesen und aufbereitet werden. Ein Dump mit Dumplevel 0 erzeugt ein Komplettbackup, während ein Level größer 0 die Änderungen seit dem letzten Backup inkrementell sichert. Level 2 sichert also alle Daten, die seit dem letzten Dump mit Level 1 erzeugt oder geändert wurden. Die Daten dazu werden in einem menschenlesbaren Format in `/etc/dumpdates` abgelegt, wenn man die Option `-u` übergibt und Dump auf eine Partition ansetzt, nicht aber auf einzelne Dateien oder Verzeichnisse.

Da Dump für gewöhnlich dazu eingesetzt wird, komplette Partitionen zu sichern, verfügt es über keine Ein-/Ausschlusslisten, wie dies einige andere Programm benutzen. Stattdessen prüft Dump, ob das Fileflag `NODUMP` gesetzt ist. Mit diesem Flag kann man einzelne Dateien oder Verzeichnisse vom Dump ausschließen. Allerdings muss man hierbei beachten, daß Dump mit Level 0

---

<sup>2</sup>allerdings bleibt dump dabei in der angegebenen Partition, folgt also Mountpoints nicht.

generell Fileflags ignoriert. Möchte man trotzdem markierte Dateien nicht sichern, muss man `-h0` übergeben.

---

```
$ chflags nodump .fetchmailrc .signature mail/ .gnupg/
$ ls -lo .signature
-rw-r--r-- 1 stefan stefan nodump 369 Feb 18 21:09 .signature
$ chflags dump mail/
```

---

#### Listing 5-5: NODUMP setzen

Dump lässt sich auf Dateisysteme und Partitionen ansetzen, dies geschieht am einfachsten mit Listing 5-6, die Option `-a` veranlasst dabei die Grösse des Archivs selbst festzulegen, anstatt nur Dateien voreingestellter Größe zu schreiben.

---

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

---

#### Listing 5-6: Sicherung mit dump

Dump kann mit der Option `-W` auch eine Übersicht der bisher erstellten Dumps aus `/etc/dumpdates` erstellen und mit `-w` die noch zu sichernden Partitionen angeben. Möchte man Archive bestimmter Größe erzeugen (bspw. für Streamer oder um sie auf CD/DVD zu brennen), verfügt dump über einige, teils historische, Optionen. Von Interesse dürfte heute aber nur noch `-B` sein, das die Größe der zu sichernden Archive in Kilobyte erwartet. Dazu muss aber zwingend der Name eines jeden zu erzeugenden Archivs übergeben werden, da dump sonst automatisch nacheinander in die zuletzt<sup>3</sup> angegebene Datei schreibt.

selber hat keine Möglichkeit Archive zu komprimieren, dies kann aber mittels Pipe an Bzip2 oder Gzip bzw. einem kleinen Shellskript gelöst werden. Möglich ist ebenfalls der Einsatz im Netzwerk mittels `rdump(8)` und `rrestore(8)`, dies ist aber aufgrund mangelnder Sicherheit bzw. fehlender Verschlüsselung nur in gesicherten Netzwerken möglich, so dass ein SSH-Tunnel mit der Umgebungsvariable `RCMD_CMD` aufgerufen werden sollte. Besser ist hier eine Pipe an `ssh`, der die Ausgabe auf einen Rechner im Netzwerk schreibt, wie in Listing 5-8 illustriert.

### 5.4.1. restore

Restore kann ein komplettes Archiv mit der Option `-r` zurückspielen (siehe Listing 5-9). Dabei erzeugt es eine Datei namens „restoresymtable“, in der die

---

<sup>3</sup>werden drei Dateinamen angegeben, aber fünf Dateien erzeugt, werden die dritte, vierte und fünfte in die dritte Datei geschrieben, man hat dann also drei Dateien mit der ersten, zweiten und fünften Sicherung vorliegen

```
#!/bin/sh

DATUM='date +%y%m%d'
QUELLE=/home/

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

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

# Medienzahl kalkulieren, Nachkomma wird abgeschnitten!
MEDIEN=$((MEDIEN/MEDIUM))

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

# Dateinamen iterieren
DAT=${DATUM}_1
i=1
while [ $i -lt $MEDIEN ]
do
    i='expr $i + 1'
    DATEIEN=${DATUM}_${i}
    DAT=${DAT},${DATEIEN}
done

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

---

Listing 5-7: dump-Befehl für mehrere Volumes

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

---

Listing 5-8: Dump im Netzwerk

Metainformationen zur Rücksicherung (Inodes, welche Bänder, etc.) abgelegt werden. Diese Datei kann nach der Rücksicherung *aller* Bänder entfernt werden.

---

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

---

Listing 5-9: Zwei Dumps mit restore zurückspielen

Möchte man nur bestimmte Pfade oder Dateien zurückspielen, kann man Teile des Verzeichnisses mit der Option `-x` übergeben. Leider unterstützt `restore` keine RegExes, aber man kann sich hierbei mit ein paar Backticks behelfen. Um zu prüfen ob ein Archiv erfolgreich geschrieben wurde, reicht es zu Testzwecken die allerletzte Datei zu restaurieren. Da `dump` alle vorigen Einträge zumindest traversieren muss, werden Fehler im Archiv erkannt und gemeldet. In der Dateiliste (`-t`) werden die Inodes und die korrespondierende Dateiname angegeben, man kann also den letzten Eintrag zurücksichern.

---

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

---

Listing 5-10: einzelne Dateien mit restore zurückspielen

Sucht man eine bestimmte Datei im gesicherten Verzeichnisbaum, kann man `restore -i` verwenden. Hier wird eine Shell aufgerufen, in der man mit `ls` und `cd` den Verzeichnisbaum traversieren kann. Hat man die gewünschten Dateien gefunden, kann man sie mit `add` zur Liste der zurückzusichernden Dateien hinzufügen. Mit `extract` werden die in der Liste spezifizierten Dateien im aktuellen Verzeichnis entpackt.

Um einen unterbrochenen Lauf fortzusetzen, kann man die Option `-R` verwenden.

### 5.4.2. Dump im Detail

Da das wichtigste Sicherungsprogramm ist, werde ich es näher untersuchen:

Das exemplarisch abgedruckte Log eines Sicherungslaufes gibt wertvolle Informationen wieder:

- das zu sichernde Verzeichnis `/home` ist als eigene Slice `/dev/rwd1a` in `/etc/fstab` aufgeführt
- Dieser Sicherungslauf (Level 1) beginnt 12:17 am 24.12.2005

---

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

---

Listing 5-11: Protokoll einer Dump-Sicherung

- Der letzte Sicherungslauf im Level 0 fand am 18.12. statt
- Der dump wird durchgeführt und erfolgreich beendet, dazu wird eine Statistik (Dauer, Platz, Übertragungsrate) ausgegeben.

Interessant sind hierbei insbesondere die einzelnen Durchläufe (*Pass I - IV*):

1. aus dem aktuellen Datum, dem angegebenen Level (hier: 1) und den letzten relevanten Durchläufen aufgezeichnet in `/etc/dumpdates` berechnet `dump` die Variable `DUMP_SINCE`. Alle Inodes im Dateisystem werden traversiert und deren *modification time* (*mtime*) wird mit `DUMP_SINCE` bezgl. Größe verglichen. Ist die *mtime* einer Datei größer oder gleich `DUMP_SINCE`, wird der Inode in die Liste zu sichernder Dateien übernommen. Nichtzugeordnete Inodes werden ignoriert, so das dump am Ende des ersten Durchlaufes drei Listen erzeugt hat:
  - a) Inodes der Dateien, die gesichert werden sollen
  - b) Inodes aller Verzeichnisse
  - c) Zugeordnete Inodes
2. Durchlauf II läuft in mehreren Stufen ab:
  - a) Die in Durchlauf 1 erstellte Liste der Verzeichnisse wird erneut traversiert und geprüft ob zu sichernde Dateien im Verzeichnis existieren. Wenn nicht, wird der entsprechende Inode aus der Liste zu sichernder Verzeichnisse entfernt.
  - b) Die Verzeichnisse werden erneut überprüft, in dem der Verzeichnisbaum von den Blättern zur Wurzel traversiert wird. Dabei sollen evtl. im letzten Durchlauf freigewordene Verzeichnisse gefunden werden.

Es wurden in diesem Lauf keine weiteren Verzeichnisse freigemacht, da sonst ein weiterer *Pass II* stattgefunden hätte.
3. Vorbereitung von Pass III:

Bevor `dump` mit dem Schreiben der Daten beginnt, werden deren Metainformationen abgelegt. Hierzu wird ein *Header* geschrieben, der die nachfolgenden Daten beschreibt. Zusätzlich werden zwei Inode-Tabellen geschrieben, eine im nativen Format und eine, aus Kompatibilitätsgründen, im normalisierten alten BSD-Format.
4. Ein Header, der den Inode des Verzeichnisses enthält, und die Datenblöcke für jedes Verzeichnis in der zu-sichernde-Verzeichnisse-Liste werden geschrieben.
5. Ein Header, der den Inode der Datei enthält, und die Datenblöcke für jede Datei in der zu-sichernde-Dateien-Liste werden geschrieben.
6. Nachbereitung von Pass IV :

Der abschließende Header wird aufs Band geschrieben.



Inkonsistenzen in der Sicherung können nur auftreten, wenn das Quelllaufwerk während der Sicherung beschrieben wird. Dabei ist es möglich, dass einzelne Dateien nicht, oder nur falsch, gesichert werden. Eine Korruption der *gesamten* Sicherung ist dabei aber äußerst unwahrscheinlich. Unmöglich werden diese Fehler, wenn das Dateisystem unmounted ist oder man Snapshots (siehe Kap. 5.3) verwendet.

## 5.5. tar(1)

Tar(1) ist das wohl bekannteste Sicherungsprogramm und vielseitig einsetzbar. Allerdings gibt es verschiedene Versionen und Implementierungen von tar, so dass man darauf achten sollte nur entsprechend kompatible Versionen einzusetzen bzw. bzw. kompatible Archive zu erzeugen.

Tar steht für Tape Archiver, es wurde also dazu entwickelt Bandarchive zu beschreiben. Es erzeugt dazu eine Datei, die alle zu sichernden Dateien enthält und schreibt diese auf ein Band – oder in eine einfache Datei.

---

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

---

Listing 5-12: Dateien mit tar archivieren und entpacken

Die Option `-f` erzeugt eine Datei, standardmäßig schreibt das Archiv sonst auf das erste Bandlaufwerk `/dev/nrst0`.

Zurückspielen kann man ein Backup mit `x`, die Option `p` sorgt dafür, dass die Dateirechte beibehalten werden.

Tar schreibt relativ zum jetzigen Pfad, d. h. das gesicherte `/home/`-Verzeichnis wird an aktueller Stelle entpackt. Tar kann mit der Option `-z` gzip- bzw. mit `-j` bzip2-komprimierte Archive erzeugen. Entpackt werden Archive mit `-x`.

Es sei hierbei nicht verschwiegen dass Tar grosse Probleme mit defekten Archiven hat, da bei einem Archiv ein Header erzeugt wird der Informationen über das gesamte Archiv beinhaltet. Taucht jetzt an einer Stelle ein Fehler auf, entpackt Tar das Archiv nur bis zu dieser Fehlerstelle.

## 5.6. star

Star ist eine von Jörg Schilling seit 1982 entwickelte Tar-Implementierung. Es ist die weltweit schnellste Tar-Variante und weist einige sehr nützliche Erweiterungen gegenüber Tar auf. Es unterstützt unter anderem Reguläre Ausdrücke, Levels a la Dump, keine Begrenzung der Pfadlänge, automatische Erkennung der Endianess

oder die Möglichkeit ein Dateisystem mit einem Tarball zu vergleichen. In der Standardvariante kann Star Tar-Archive lesen und umgekehrt. Star lässt sich aus `pkgsrc/archivers/star` installieren und sollte im Allgemeinen dem normalen Tar vorgezogen werden. Star unterstützt verschiedene Tar-Formate, die mit der Option `-H=` bestimmt werden können. Am leistungsfähigsten ist EXUSTAR.

## 5.7. *cpio(1)*

Cpio (copy in/out) ist von der Syntax her etwas eigen, da es als erste Option die Übergabe einer zu sichernden Datei erwartet und diese auf `STDOUT` schreibt. Dieses „Problem“ erweist sich allerdings mithilfe der Pipe als Vorteil, da so mit `find(1)` oder `ls(1)` die Dateiliste übergeben und die Ausgabe mit `>` bestimmt werden kann.

---

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

---

Listing 5-13: Dateien mit cpio sichern

Listing 5-13 zeigt wie man alle JPEGs im aktuellen Verzeichnis in die Datei `./Bilder.cpio` sichert. Die Ausgabe kann stattdessen auch auf das Bandlaufwerk `/dev/nrst0` umgeleitet werden. Um Unterverzeichnisse zu sichern oder inkrementelle Backups zu erzeugen nutzt man `find(1)`. Der zweite Befehl sichert alle Dateien die in den letzten sieben Tagen geändert wurden auf das erste Bandlaufwerk. Zurückspielen kann man die Sicherung mit dem dritten Befehl.

Leider unterstützt Cpio keine Überprüfung der Archive, so daß man dies auf Wunsch manuell machen muss, indem man das Archiv wieder entpackt und die Dateien mit `diff(1)` oder `mtree(8)` vergleicht.

## 5.8. *afio(1)*

Afio ist eine Neuauflage von Cpio und behebt einige Probleme, z. B. Probleme mit leeren Dateien oder auf 0 gesetzte Rechte. `afio` verwendet dabei die selbe Syntax wie Cpio `afio` ist via `pkgsrc/archivers/afio` installierbar und sollte Cpio vorgezogen werden.

## 5.9. pax(1)

Pax ist ein Standardisierungsversuch der IETF, da es zu viele Implementierungen von Tar und Cpio gibt. Pax ist in der Lage verschiedene Archivformate wie Cpio und Tar zu schreiben und zu lesen und kann so gerade in heterogenen Umgebungen nützlich sein. pax wird unter anderem auch auf den Installationsmedien von NetBSD verwendet.

---

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

---

Listing 5-14: Sicherung und Rücksicherung mit pax

Befehl Nummer eins in Listing 5-14 schreibt das /home/-Verzeichnis auf Band und zweitens gibt ein Inhaltsverzeichnis der Sicherung aus. Der letzte Befehl extrahiert alle Dateien in /usr/ im Archiv ./a.pax

## 5.10. Verzeichnisse synchronisieren mit rsync(1)

Rsync ist aus pkgsrc/net/rsync installierbar und ermöglicht es Dateihierarchien auf lokalen Platten oder im Netz zu synchronisieren. Dabei verwendet es den rsync-Algorithmus, der nur Daten überträgt, die nicht gleich sind. Dabei ist es verständlicherweise recht schnell und effizient, setzt aber, bei Verwendung im Netzwerk, auf beiden Maschinen ein installiertes rsync voraus.

Rsync kann verschiedenste Optionen übernehmen, unter anderem auch Exclude/Include-Anweisungen oder Optionen, ob es nicht mehr existierende Dateien auf dem Empfänger löschen soll.

---

```
1$ rsync -azc -e ssh --exclude-from=/home/rsync.excl \
    --delete-excluded --delete-after /home/ \
    stefan@192.168.2.2:/home/

2$ rsync -azc -e ssh --exclude-from=/home/rsync.excl \
    --delete-excluded --delete-after \
    /home/* /usr/home/
```

---

Listing 5-15: rsync-Beispiele

Der erste Befehl synchronisiert das lokale /home/ mit /home/ auf 192.168.2.2. Es verwendet dazu ssh(1) als Tunnel, schließt die Pfade, die in /home/rsync.excl

---

definiert sind, aus. `-delete-excluded` und `-delete-after` löscht Dateien die nicht synchronisiert werden sollen auf dem Empfänger, dies soll allerdings erst nach dem Transfer geschehen. Die Optionen `-azc` starten den Durchlauf rekursiv, als Archivierung, mit Komprimierung und Prüfsummenvergleich der Dateien. Der zweite Befehl führt genau das selbe durch, synchronisiert aber mit der lokalen Platte.

Rsync lässt sich einfach verwenden um Datenbestände auf verschiedenen Medien (Wechselplatten, tragbare Wechselmedien, ZIP-Disketten, NFS) oder verschiedenen Systemen (Dateiserver, Laptop) zu synchronisieren. Durch den *rsync*-Algorithmus ist *rsync* insbesondere bei vielen kleinen Dateien (z.B. CVS-Repository) sehr schnell.

Rsync lässt sich sehr gut verwenden um Daten auf eine andere Platte zu spiegeln. Das folgende Skript synchronisiert bei jedem Herunterfahren des Rechners per ACPI-Schalter ein CVS-Repository mit einer zweiten Platte. Dazu muss lediglich ein NetBSD-Kernel mit ACPI verwendet werden, `powerd=yes` in der `/etc/rc.local` gesetzt und `/etc/powerd/scripts/power_button` wie in Listing 5-16 angepasst werden.

---

```
case "${2}" in
  pressed)
    /usr/pkg/bin/pg_dumpall -Upgsql > \
      /usr/home/back/pgdump.‘date +%y%m%d‘
    /usr/pkg/bin/pg_ctl stop -D /usr/pkg/pgsql/
    /usr/pkg/bin/rsync -a /home/cvs/ /usr/home/back/
    echo "System wird heruntergefahren."
    /sbin/shutdown -p now "power button pressed"
  exit 0
;;
*)
```

---

Listing 5-16: `/etc/powerd/scripts/power_button`

Zuerst wird ein `pg_dump` erzeugt und PostgreSQL heruntergefahren. Anschließend wird mit Rsync das lokale CVS-Repository (`/home/daten/cvs/`) mit der zweiten Platte synchronisiert.

Rsync unterstützt mit der Option `-c` die Prüfung übertragener Dateien mit einer MD4-Prüfsumme. Anders als `cp(1)` kann es so verifizieren, ob die übertragenen Dateien korrekt sind. Daher sollte man Rsync mit Prüfsummenvergleich verwenden, wenn man sicherstellen will, daß Daten korrekt übertragen wurden.

### 5.10.1. MS Windows als Client

Möchte man einen Windowsclient mit einem NetBSD-Server synchronisieren, bietet sich cwRsync<sup>4</sup> an.

Es ermöglicht den kompletten Betrieb als Rsync-Client und/oder Server, da es alle benötigten Komponenten (SSH, Cygwin) integriert.

Nach der Installation kann man eine Batchdatei erstellen, die bspw. bei jedem Systemstart oder Login aufgerufen wird und bestimmte Verzeichnisse mit dem Server synchronisiert.

---

```
c:\CWRSYNC\rsync -e C:\CWRSYNC\ssh -av --delete
--delete-excluded
--exclude "[Tt]emp*" --exclude "privat"
--exclude 'Temporary Internet Files'
"Eigene Dateien", "Lokale Daten"
uni@192.168.2.1:/home/uni
```

---

Listing 5-17: cwRsync auf Windows einsetzen

Am einfachsten konfiguriert man SSH für die Benutzer per Schlüssellogin, indem man die Schlüsseldateien nach `c:\Eigene Dateien\ssh` kopiert. Aus Sicherheitsgründen sollte man auf dem NetBSD-Server einen eigenen `sshd` auf einem anderen Port als 22 starten und den Zugriff darauf per `sshd.conf` und `ipf` begrenzen.

Die Batchdatei kann nun explizit, oder über `schtasks`, der Windows XP Variante von `cron`, aufgerufen werden:

---

```
schtasks /create /SC BEI ANMELDUNG /TN taeglbackup
/TR c:\CWRSYNC\backup.bat /SD 14/12/2004
```

---

Listing 5-18: Windows-Programme zeitgesteuert ausführen

## 5.11. rdiff-backup

Synchronisationsmechanismen haben ein Problem - auch Fehler werden synchronisiert. Stürzt bspw. eine Anwendung ab und vernichtet dabei den Inhalt einer Datei, wird diese fehlerhafte Datei bei der Synchronisation übertragen. Um dieses Problem zu umgehen, ist eine Archivierung der synchronisierten Daten

---

<sup>4</sup>[http://sourceforge.net/project/showfiles.php?group\\_id=69227&package\\_id=68081](http://sourceforge.net/project/showfiles.php?group_id=69227&package_id=68081)

erforderlich. Dies kann mit einer nächtlichen Sicherung des Spiegels geschehen - oder aber vom Sicherungsprogramm selbst erledigt werden.

Basierend auf der Rsync-Bibliothek wurde Rdiff-backup entwickelt. Es erstellt wieder eine Spiegelung der Daten, hält aber zusätzlich eine Reihe an Metadaten und Diffs der Dateien vor. Somit wird ab Beginn der Sicherung jede Version der Sicherung vorgehalten und kann wiederhergestellt werden.

Werden Dateien in der Quelle geändert und Rdiff-backup erneut aufgerufen, legt es im Zielverzeichnis rdiff-backup-data/increments/ an, in dem die verschiedenen Diffs der Versionen abgelegt sind. Man kann jede dieser Versionen durch Angabe des Zeitpunkts wiederherstellen lassen.

---

```
1$ rdiff-backup /etc/ /usr/back/etc/
2$ rdiff-backup -l /usr/back/etc/
Found 6 increments:
  increments.2006-01-01T12:35:24+01:00.dir  Sun Jan  1 12:35:24 2006
  increments.2006-01-04T12:35:37+01:00.dir  Wed Jan  4 12:35:37 2006
  increments.2006-01-28T12:35:31+01:00.dir  Sat Jan 28 12:35:31 2006
  increments.2006-02-03T12:35:28+01:00.dir  Fri Feb  3 12:35:28 2006
  increments.2006-02-04T12:35:50+01:00.dir  Sat Feb  4 12:35:50 2006
  increments.2006-02-06T12:35:27+01:00.dir  Mon Feb  6 12:35:27 2006
Current mirror: Sun Feb 26 12:35:29 2006
3$ rdiff-backup -r 2B --force /usr/back/etc/ /etc/
```

---

Listing 5-19: rdiff-backup

Der erste Befehl sichert alle Dateien aus /etc/ nach /usr/back/etc/. Der zweite Befehl zeigt die inkrementellen Sicherung und deren Daten an.

Der dritte Befehl stellt die Sicherung wieder her, und zwar mit -r 2B den Zustand der zweitletzten inkrementellen Sicherung. -force erlaubt dabei das Dateien im Zielverzeichnis überschrieben werden dürfen.

## 5.12. dd(1)

Dd(1) ist kein eigentliches Sicherungsprogramm, sondern es dient dazu Dateien zu konvertieren. Dd kann jedoch auch auf Raw Devices schreiben und von ihnen lesen, dies ermöglicht es, echte 1:1-Kopien von Geräten zu erstellen und zu schreiben, so kann man z. B. mit den Befehlen in Listing 5-20 ein Image einer Floppydiskette erstellen, die Partition /dev/wd1a 1:1 auf /dev/wd2a kopieren oder den MBR der ersten Platte sichern. Somit eignet es sich hervorragend um an Rohdaten, bspw. von Magnetbändern, heranzukommen oder komplette Systemplatten zu sichern. Da dd die gesamten Blöcke einer Partition einliest,

wird das erzeugte Image genau so groß wie die Partition werden. Möchte man das Image komprimieren, sollte man vorher im laufenden System den leeren Festplattenplatz mit Nullen überschreiben.

Dd ist ebenfalls in der Lage Dateien zu konvertieren, so kann man mit der Option `conv=swab` die Byte-Order vertauschen lassen.

---

```
# dd if=/dev/zero of=/nullen && rm /nullen
# dd if=/dev/fd0a of=floppy.iso
# dd if=/dev/wd1a of=/dev/wd2a bs=2m | split -b 695M
# dd if=/dev/wd0d of=mbr.dd count=1
```

---

Listing 5-20: Daten mit dd schreiben

### 5.13. Ghost for Unix (g4u)

Ghost for Unix<sup>5</sup> (g4u) ist eine NetBSD-Diskettenvariante, die von Hubert Feyrer entwickelt wurde um Festplatten zu spiegeln und im Netz via FTP zu verteilen. Es verwendet dd (siehe Kap. 5.12) um auf Platten zuzugreifen und kann daher jedes Datei-/Betriebssystem spiegeln. Es verwendet zum Betrieb einige Shellskripte, so dass es auch ein ungeübter NetBSD-Anwender einfach anwenden kann. Es eignet sich insbesondere hervorragend um mehrere gleich aufgebaute Systeme zu verteilen, so dass man bspw. in Laboren, Klassenräumen oder Clustern (siehe Regensburger Marathon-Cluster<sup>6</sup>) ein System nur einmal aufsetzt und dieses dann auf mehrere Rechner verteilt.

Es wird lediglich das Disketten- oder CD-Image benötigt um ein bootbares Medium zu erstellen. Davon wird g4u gebootet und anschließend können Festplatten und/oder Partitionen gesichert und restauriert werden.

---

```
# uploaddisk benutzer@ftp.server.local Imagename.gz wd1
# slurpdisk benutzer@ftp.server.local Imagename.gz wd0
# copydisk sd0 sd1
```

---

Listing 5-21: Festplatten-Images mit g4u erstellen

In Listing 5-21 wird die zweite IDE-Platte (wd1) als ./Imagename.gz auf ftp.server.local gesichert. Der zweite Befehl installiert das Image auf der ersten IDE-Platte, der dritte Befehl kopiert lokal die erste SCSI auf die zweite SCSI-Platte.

---

<sup>5</sup><http://www.feyrer.de/g4u/>

<sup>6</sup><http://www.feyrer.de/marathon-cluster/>

G4u ist ein ausgezeichnetes System um komplette Platten zu sichern und zu verteilen. Für Datenbackups eignet es sich nur bedingt, da immer komplette Platten oder Partitionen mit dd gesichert werden.

Auf der Homepage des Systems findet sich eine ausführliche Anleitung.



**Teil III.**

# **Sicherung verteilter Systeme**



## 6. Sicherung verteilter Systeme in Netzwerken

### 6.1. Amanda

[Amanda](#) steht für „Advanced Maryland Network Disk Archiver“ und eignet sich, wie der Name schon sagt, hervorragend zur Sicherung in Netzen. Es verwendet dazu ein Client/Server-Modell um Unix-Clients und SMB-Shares<sup>1</sup> auf Bandlaufwerk zu sichern. Inzwischen verfügt es auch über die Fähigkeit die Sicherung auf Platten abzulegen oder Platten als Spoolaufwerk einzusetzen. Zur Sicherung eines Unixclients kann es GNU Tar oder das native dump des Clients verwenden. Um Daten von MS-Windows Rechnern zu sichern, müssen diese als Freigabe ins Netzwerk exportiert werden, um dann vom Server aus mit smbtar kopiert werden zu können. Für Unixclients kann Amanda auf den Clients selbst die Daten vor der Übertragung komprimieren oder diese nach der Übertragung auf dem Server komprimieren. Je nachdem ob die Auslastung der Clients oder die des Netzes wichtiger ist, kann man eine passende Strategie wählen. Da Amanda ein eigenes Netzprotokoll zur Übertragung der Daten verwendet, kann es an verschiedenen Punkten Daten abgreifen oder verändern (bspw. komprimieren, verschlüsseln oder auf Viren prüfen).

Die Zyklen der Sicherungsstrategie (also wann welches Level eingesetzt wird) berechnet Amanda selbständig aus den Statistiken der vorangegangenen Läufe, so das ein Optimum an Platzersparnis und Sicherungsaufwand errechnet wird. Bei manuellen Sicherungen lautete die Devise im allgemeinen „Freitags ein Komplettbackup, das archiviert wird.“. Mit Amandas automatischer Strategieberechnung lautet diese Devise: „Gib mir ein Komplettbackup in 7 Tagen.“. Durch diese Definition des *dumpcycle* verteilt Amanda die Sicherungen selbständig auf die Bänder – was dazu führt das den Logdateien erhebliche Bedeutung zukommt. Die Berechnung der Sicherungsstrategie erfolgt aus mehreren Daten, zum einen dem Dumpcycle, also wieviel Tage für einen Zyklus verwendet werden sollen. Ist ein Dumpcycle sieben Tage lang, werden pro Tag ein Siebentel der Daten gesichert. Zusätzlich sichert Amanda aber noch jene Daten inkrementell, die seit der ersten Sicherung verändert wurden.

Wie schon bereits im Kapitel 3.7 erwähnt, sind Festplatten heutzutage in der Regel hinreichend groß und auch sehr billig, so das der Einsatz einer

---

<sup>1</sup>Windows-Freigaben

Spoolingplatte als Zwischenspeicher wärmstens empfohlen werden kann. Sichert Amanda direkt auf Band und die Daten kommen nicht schnell genug heran, fängt ein Streamer an „Schuhe zu putzen“, also immer wieder vor- und zurückzuspulen. Das tut der Mechanik und dem Band nicht wirklich gut und vermindert den weiteren Durchsatz dramatisch. Amanda unterstützt eine Vielzahl an Bandlaufwerken und Wechseinheiten, die über Skripte angesteuert werden können.

Seit geraumer Zeit verfügt Amanda über RAIT - dem Redundant Array of Inexpensives Tapes. Ähnlich dem RAID für Festplatten werden hier einfach mehrere Bänder eingesetzt und die Kapazität und Ausfallsicherheit zu erhöhen, indem man bsp. einfach fünf Bänder verwendet – vier für Daten und eines für eine XOR-Prüfsumme. So kann man Partitionen sichern, die viermal so groß sind wie ein einzelnes Band und zusätzlich noch die Integrität der Sicherung erhöhen. Da ein RAIT nicht einfach nacheinander die Sicherungen auf Bänder schreibt, sondern parallel, benötigt es in diesem Falle auch fünf Bandlaufwerke. Wenn das Sicherungssystem entsprechende Durchsatzraten erzeugt, erhöht dieses Verfahren natürlich auch den Datendurchsatz.

Allerdings kann Amanda einen dump-Lauf nicht auf mehrere Bänder verteilen. Ist der dump einer Partition größer als das Band, kann dump zum sichern dieser Partition nicht verwendet werden. Stattdessen kann man GNU Tar verwenden und mittels Include/Exclude-Liste<sup>2</sup> die Partition „stücken“ und so auf mehrere Bänder verteilen.

### 6.1.1. Einrichtung des Servers

Amanda wird komplett per Konsole bzw. per dotconf-Dateien konfiguriert. Nach der Installation aus `pkgsrc/misc/amanda` für den Server werden in `/usr/pkg/etc/amanda` die Standard-Konfigurationsdateien angelegt. Amanda erwartet nun für jeden Konfigurationssatz ein eigenes Verzeichnis der Art `/usr/pkg/etc/amanda/konfigname`.

in jeder Konfiguration erwartet Amanda einige Dateien:

**.amandahosts** Konfiguriert den Netzwerkzugriff ähnlich `/etc/hosts`

**amanda.conf** Allgemeine Konfiguration, Geräte, Zyklus etc.

**disklist** zu sichernde Platten/Verzeichnisse (DLE = Disk List Entry)

**tapelist** Liste der Bänder, von Amanda selbst verwaltet

**/etc/amandates** Amandas Äquivalent zu `/etc/dumpdates`, sollte mit `touch(1)` angelegt werden

**/etc/inetd.conf** Amandas Dienste müssen über den `inetd` gestartet werden

---

<sup>2</sup><http://www.amanda.org/docs/topten.html#id2552399>

Die Datei `amanda.conf` enthält folgende Parameter:

**org** Betreffzeile für Berichtmails

**mailto** Empfänger der Berichtmails

**dumpuser** Benutzer der die Dumps durchführt. Standard ist `[backup]`

**inparallel** Maximale Anzahl parallel laufender Dumps. (0-63)

**dumporder** Priorisierungskriterium der Dumpreihenfolge. Möglich sind Größe, Bandbreite und Zeit, jeweils auf- oder absteigend.

**netusage** Maximal belegbare Bandbreite.

**dumpcycle** Anzahl der Tage/Wochen für einen Sicherungszyklus.

**runspercycle** Durchläufe pro Zyklus, zum Beispiel (Wochen des `dumpcycle`) mal Werkzeuge

**tapecycle** Anzahl der Bänder Amanda benötigt, entspricht `runspercycle` + Sicherheitszuschlag

**runtapes** Bänder je einzeltem Amandalauf

**tpchanger** Skript für den Bandwechsler

**tapedev** Bandlaufwerk oder Verzeichnis, wenn auf Platte gesichert werden soll (z.B: `file:/mnt/amanda`)

**tapetype** Bandtyp, z.B. HP-DAT, DLT oder harddisk. Beispiele sind bereits in der `amanda.conf` enthalten.

**reserve** Reserviert  $n\%$  der Spoolplatte für inkrementelle Backups

**labelstr** Ein RegEx der das Label für die Bänder festlegt.

Wie bereits erläutert, sollte Amanda unbedingt mit einer Spoolplatte betrieben werden. Diese kann ebenfalls in der `amanda.conf` definiert werden. Ist eine Spoolplatte konfiguriert, benutzt Amanda diese um erst Clientdaten zu sammeln und dann von der Platte auf Band zu schreiben. Definiert man mehrere Spoolplatten, benutzt Amanda diese nach eigenem Gutdünken. Ist ein dump zu groß für die Spoolplatte, wird er direkt aufs Band geschrieben. Wenn ein Band nicht beschrieben werden kann, weil es bspw. defekt oder schreibgeschützt ist, wird ein sogenannter *degraded mode* erzeugt, die Sicherung also nur auf die Spoolplatte geschrieben und nicht auf das Band. Hat man den Bandfehler behoben, kann man mit `amflush` die Spooldatei auf ein Band schreiben. Eine Spoolplatte wird beispielhaft in Listing 6-1 definiert.

Die Konfiguration der Sicherungsparameter findet in den sogenannten *dumptypes* (Listing 6-2) statt. Hier wird bspw. festgelegt ob beim Sicherungslauf komprimiert werden soll

Die eigentlichen zu sichernden Pfade legt man nun in der `./disklist` (Listing 6-3) an:

Zu guter Letzt muss man noch die Amandadienste via `/etc/inetd.conf` (Listing 6-4) starten

```
holdingdisk hd1 { # Name der Platte
  comment "erste SCSI als Spoolplatte"
  directory "/mnt/sd0a/" # Mountpoint der Platte
  use 5500 MB # Maximal verfügbarer Platz
  #use -100 MB # Nutze gesamte Platte bis auf 100MB
  chunksize 1Gb # Größe einzelner Dateien, wenn der dump
                  # auf mehrere Dateien verteilt werden soll
```

---

Listing 6-1: Spoolplatten für Amanda definieren

---

```
define dumptype always-full { # Name der Konf.
  global # dumptype global einbinden
  comment "Full dump of this filesystem always"
  compress none
  priority high
  dumpcycle 0
}

define dumptype stantar {
  global # dumptype "global" einbinden
  program "GNUTAR" #gtar verwenden
  comment "standardtar"
  compress client fast # auf Client schnell komprimieren
  holdingdisk yes # Spoolplatte verwenden
  index # Inhaltsverzeichnis erstellen
  priority # high
  dumpcycle 7 # eine Woche Zyklus
  exclude list "/mnt/amanda.exclude"
}

define dumptype root-tar {
  global
  program "GNUTAR"
  comment "root partitions dumped with tar"
  compress none
  index
  exclude list "/usr/amanda/exclude.gtar"
  priority low
}
```

---

Listing 6-2: dumptypes in amanda.conf definieren

---

---

```
balmung.net-tex.de /mnt/source stantar
balmung.net-tex.de /home always-full
kusanagi.net-tex.de /home always-full
```

---

Listing 6-3: Zu sichernden Pfade in der disklist festlegen

---

```
amanda dgram udp wait amanda /usr/pkg/libexec/amandad amandad
amandaidx stream tcp nowait amanda /usr/pkg/libexec/amindexd amindexd
amidxtape stream tcp nowait amanda /usr/pkg/libexec/amidxtaped amidxtaped
```

---

Listing 6-4: Amanda-Server-Dienste starten

### 6.1.2. Einrichtung der Clients

Auf den Clients muss das Amanda-client Paket (`pkgsrc/sysutils/amanda-client`) installiert werden. Anschließend wird in der `/etc/inetd.conf` (Listing 6-5) der Amanda-Server gestartet und in der `/usr/pkg/etc/amanda/.amandahosts` (Listing 6-6) die Zugriffsrechte ähnlich einer `.rhosts` vergeben.

---

```
amanda dgram udp wait amanda /usr/pkg/libexec/amandad amandad
```

---

Listing 6-5: Amanda-Client-Dienste starten

### 6.1.3. Programme im Amanda-Paket

Amanda verfügt über mehrere Programme, die mit dem Paket installiert werden:

**amadmin** Verwaltungsprogramm für die Amandadaten. Kann bspw. Bänder freigeben, löschen oder suchen und die Datenbank im-/exportieren.

**amcheck** Überprüft die Konfigurationsdateien auf Fehler. Sollte zu den Bürozeiten durchgeführt werden.

**amcheckdb** Überprüft die Banddatenbank auf Konsistenz, also ob alle aufgeführten Bänder in der Bandliste sind.

**amcleanup** Bereinigt den Server nach einem Absturz. Normalerweise automatisch ausgeführt wird es nur nach einem Systemfehler benötigt.

**amdd** Amandas dd.

**amdump** Startet den Sicherungslauf mit allen DLEs einer Konfiguration. Das Programm für den nächtlichen Cronlauf.

```
localhost backup
localhost root
```

---

Listing 6-6: Zugriffsrechte auf dem Client konfigurieren

**amflush** Schreibt dumps von der Spoolplatte auf Band.

**amgetconf** Gibt Parameter der Konfigurationsdateien wieder.

**amlabel** Labelt ein Band. Amanda überschreibt kein gelabeltes Band und benutzt keines ohne Label.

**ammt** Amandas mt.

**amoverview** Listet eine Tabelle auf in der angezeigt wird wann welche Partition mit welchem Level gesichert wurde. (Ähnlich `dump -W`)

**amrecover** Durchsucht die Amanda Indizes um wiederherzustellende Dateien auszuwählen. Dateien können bspw. nach Datum gesucht werden.

**amrestore** Spielt ein Backup vom Band zurück, so das es native zurückgesichert werden kann.

**amrmtape** Entfernt ein existierendes Band aus der Datenbank. Nur notwendig wenn das Band zerstört wurde.

**amstatus** Zustand der laufenden Sicherung.

**amtape** Zugriff auf Bandwechsler, bspw. Band auswerfen oder das Laufwerk reinigen.

**amtoc** Perlskript das ein Inhaltsverzeichnis aus einer Logdatei erzeugt. Kann idealerweise mit `amdump` in einem Cronjob kombiniert werden.

**amverify** Überprüft ein Band auf Fehler.

**amverifyrun** Überprüft alle Bänder des letzten Laufs auf Fehler.

#### 6.1.4. Praktischer Einsatz

Zuerst sollte man mit `amcheck(8)` die Konfiguration auf Fehler überprüfen. Man kann dies auch automatisiert per Cron zu den üblichen Bürozeiten ausführen, so das rechtzeitig auftretende Probleme erkannt werden können. Anschließend sollten alle Bänder, die verwendet werden soll, gelabelt werden. Dies kann man mit `amlabel(8)` tun, indem man es ggf. in ein Shellskript einbindet (nützlich für Bandwechsler oder wenn man auf Festplatte sichert).

Mit `amdump(8)` kann man vorerst von Hand den Sicherungslauf anstoßen und die Benachrichtigungsemail studieren. Nach einem erfolgreichen Sicherungslauf sollte das Band mit `amverify(8)` oder `amverifyrun(8)` überprüft werden. Mit `amtoc(8)` kann man nun aus dem Inhaltsverzeichnis des Sicherungsbandes eine



menschenlesbare ASCII-Datei machen, was recht nützlich ist, falls man nicht auf den Katalog zugreifen kann.

Hat das System von Hand funktioniert, sollte man es in ein Shellskript gießen (dokumentieren und sichern nicht vergessen) und von Cron aufrufen lassen.

## Rücksicherung

Einzelne Dateien, Verzeichnisstrukturen oder ganze Platten lassen sich mit `amrestore` zurückspielen. Als Optionen erwartet `amrestore` optional mehrere Hostnamen, Plattennamen und Zeitstempel. Wird eine Option nicht gesetzt, werden alle Dateien getroffen und zurückgespielt, übergibt man also gar keine der Optionen, werden alle verfügbaren Sicherungen zurückgespielt.

Hat man den Amanda-Index nicht mehr zur Verfügung, kann man Bänder auch mit `amrecover` auslesen. Normalerweise spielt `amrecover` die Dateien der Sicherungen nicht zurück an ihren alten Platz, sondern spielt nur die Archivdateien von Band auf Festplatte. Möchte man die Archivdateien gleich entpacken und die Sicherungen so zurückspielen, übergibt die Option `-p` (wie Pipe) und in einer Pipe das entsprechende Restoreprogramm, also GNU Tar oder `restore` für `dump`. Neben `-p` kann man auch `-h` angeben, um die Header des Bandes auslesen zu können, oder `-r` um die Archive im Rohformat zu Debuggingzwecken vom Band lesen zu können.

Hat man nicht einmal mehr `amrecover` zur Verfügung, ist dies wohl der GAU der Datensicherung. Solange die Bänder aber unbeschädigt sind, kann man die Images auch mit `dd(1)` vom Band herunterziehen. Dabei hilft Amandas Headerdatei (Listing 6-7) am Anfang des Bandes, denn dort wird im Klartext angezeigt, wie man mit `dd(1)` und GNU Tar oder `restore(8)` die Daten rücksichert.

---

```
# mt rewind
# dd if=/dev/nrst0 bs=32k count=1
```

---

Listing 6-7: Amanda Header vom Band restaurieren

## 6.2. Bacula

Bacula ist ein System zur Datensicherung, das speziell für verteilte Systeme entwickelt wurde. Es besteht aus verschiedenen Dæmons, die relativ einfach den Client-Server-Betrieb ermöglichen.

Bacula besteht aus folgenden Dæmons:

- Director (`bacula-dir`)

- Storage (bacula-sd)
- File (bacula-fd)
- Catalog (PostgreSQL)
- Console (bconsole)

Der „Director“ läuft als Steuerungsprozess im Hintergrund und koordiniert alle Sicherungsaktivitäten. Er wird eingesetzt um Daten zu sichern und wiederherzustellen und um Aufgaben zu planen.

Der „Storage“-Dæmon übernimmt das Schreiben der Daten auf die Bänder<sup>3</sup> und ggf. das Wiedereinlesen der Bänder.

Der „File“-Dæmon liefert die Daten des zu sichernden Rechners auf Befehl des Directors an den Storage-Dæmon. Der File-Dæmon ist betriebssystemabhängig. Für NetBSD kann er via pkgsrc installiert werden, für andere Unix-artige steht er ebenfalls zur Verfügung. Weiterhin existiert ein Client-Programm für MS-Windows Rechner.

Der „Catalog“ speichert alle Metadaten zu den Sicherungsläufen, also Daten wie: „welche Datei wurde in welcher Version wann auf welches Band gesichert und hat dabei folgenden Prüfsumme?“. Mit ihm ist es möglich die Sicherungsbänder zu organisieren und ggf. im Falle einer Rücksicherung das gewünschte Band zu finden.

Da hierzu mehr als eine einfache Textdatei notwendig ist, kann sich Bacula an SQLite, PostgreSQL oder MySQL binden. Ich beschränke mich hierbei auf PostgreSQL<sup>4</sup>. Der Catalog ist wichtig, wenn man Dateien zur Rücksicherung suchen will. Daher sollte er zwingend gesichert werden, das Bacula-Handbuch zeigt wie man den Katalog sichert und für eine Notfall-CD aufbereitet. Außerdem kann man natürlich PostgreSQL-eigene Funktionen verwenden, um den Katalog zu sichern.

Die „Console“ ermöglicht den Zugriff auf die Bacula-Dienste. Neben einer einfachen Shell-basierten Textkonsole existieren auch klicki-bunte Varianten für Gnome, Java oder mit wxWidgets.

Alle Komponenten kommunizieren über das Netz miteinander und können so auf verteilten Systemen eingesetzt werden. Zur Authentifizierung verwenden sie CRAM-MD5 und zur Verschlüsselung kann TLS eingesetzt werden.

Bacula unterstützt Bandlaufwerke, Bandwechsler und Sicherungen auf Festplatte. Tägliche Bandrotation wird durch multiple Pools ermöglicht.

Um die Konfiguration zu vereinfachen, wird sie in verschiedene Direktiven zerlegt, das FileSet definiert was gesichert wird, der Client welcher Rechner gesichert wird, der Schedule wann gesichert wird und der Pool wohin gesichert wird.

---

<sup>3</sup>oder auch CDs, DVDs, Festplatte, 8"-Disketten

<sup>4</sup>**Niemand** will seine Datensicherung [MySQL](#) anvertrauen.

SQLite ist hinreichen, wenn man keinen Wert auf umfangreiche Datenbankfunktionen legt.

Bacula verwendet keine Level wie `dump`, stattdessen wird im Schedule ein Zeitpunkt und die Sicherungsart (*Full*, *Incremental* und *Differential*) festgelegt.

### 6.2.1. Installation

Bacula ist als `pkgsrc/sysutils/bacula` in `pkgsrc` erhältlich. Ein Windows-Client existiert ebenfalls, er ist auf der Bacula-Seite erhältlich und wird wie ein `Unix-bacula-fd` konfiguriert.

Bacula wird standardmäßig mit SQLite als Katalog kompiliert. Um PostgreSQL anzubinden, muss man die Paketoptionen in `/etc/mk.conf` auf `PKG_OPTIONS.bacula= -catalog-sqlite catalog-pgsql` setzen.

Mit dem üblichen `make install clean` wird bacula nun installiert.

Für den Einsatz auf Clients, kann man `pkgsrc/sysutils/bacula-clientonly` verwenden.

Um die Datenbank einzurichten, liegen in `/usr/pkg/libexec/bacula/` verschiedene Shellskripte bereit. PostgreSQL wird gemäß Listing 6-8 eingerichtet.

---

```
# mkdir /usr/postgres && chown -R postgres.postgres /usr/postgres
# su - postgres
postgres$ initdb /usr/postgres
[...]
postgres$ /usr/pkg/libexec/bacula/create_postgresql_database
postgres$ /usr/pkg/libexec/bacula/make_postgresql_tables
postgres$ /usr/pkg/libexec/bacula/grant_postgresql_privileges
[...]
CREATE USER
postgres$ psql bacula
Welcome to psql 8.0.4, the PostgreSQL interactive terminal
bacula=#
```

---

Listing 6-8: PostgreSQL für Bacula einrichten

### 6.2.2. Konfiguration

Zur Konfiguration verwendet Bacula verschiedene Direktiven in den Dæmons. Bevor man sich an die Konfiguration macht, sollte man ein vorhandenes Bandlaufwerk auf Kompatibilität mit Bacula testen. Dazu gibt [[Bacula User's Manual](#), Kap. „Testing Your Tape Drive With Bacula“] eine ausführliche Schritt-für-Schritt-Anleitung. Um PostgreSQL zu starten, kann man `/usr/bin/su -m postgres -c '/usr/pkg/bin/pg_ctl -D /usr/postgres/ -l /usr/postgres/logfile start&'` in `/etc/rc.local` einfügen. Für Baculas

Tabelle 6.1.: Baculas PostgreSQL-Datenbank

Schema	Name	Type	Owner
public	basefiles	table	bacula
public	basefiles_baseid_seq	sequence	bacula
public	cdimages	table	bacula
public	client	table	bacula
public	client_clientid_seq	sequence	bacula
public	counters	table	bacula
public	device	table	bacula
public	device_deviceid_seq	sequence	bacula
public	file	table	bacula
public	file_fileid_seq	sequence	bacula
public	filename	table	bacula
public	filename_filenameid_seq	sequence	bacula
public	fileset	table	bacula
public	fileset_filesetid_seq	sequence	bacula
public	job	table	bacula
public	job_jobid_seq	sequence	bacula
public	jobmedia	table	bacula
public	jobmedia_jobmediaid_seq	sequence	bacula
public	media	table	bacula
public	media_mediaid_seq	sequence	bacula
public	mediatype	table	bacula
public	mediatype_mediatypeid_seq	sequence	bacula
public	path	table	bacula
public	path_pathid_seq	sequence	bacula
public	pool	table	bacula
public	pool_poolid_seq	sequence	bacula
public	status	table	bacula
public	storage	table	bacula
public	storage_storageid_seq	sequence	bacula
public	unsavedfiles	table	bacula
public	version	table	bacula

Tabelle 6.2.: Baculas Table „public.file“

Column	Type	Modifiers
fileid	integer	not null default nextval('public.file_fileid_seq'::text)
fileindex	integer	not null default 0
jobid	integer	not null
pathid	integer	not null
filenameid	integer	not null
markid	integer	not null default 0
lstat	text	not null
md5	text	not null

## Indexes:

```
"file_pkey" PRIMARY KEY, btree (fileid)
"file_fp_idx" btree (filenameid, pathid)
"file_jobid_idx" btree (jobid)
```

Tabelle 6.3.: Baculas Table „public.filename“

Column	Type	Modifiers
filenameid	integer	not null default nextval('public.filename_filenameid_seq'::text)
name	text	not null

## Indexes:

```
"filename_pkey" PRIMARY KEY, btree (filenameid)
"filename_name_idx" btree (name)
```

Dæmons existieren insgesamt drei rc-Skripte, die man nach `/etc/rc.d` kopieren und in `/etc/rc.conf` starten kann, siehe Listing 6-9.

```
#cp /usr/pkg/share/examples/rc.d/bacula /etc/rc.d/  
# echo 'bacula=yes' >> /etc/rc.conf  
#cp /usr/pkg/share/examples/rc.d/bacula-dir /etc/rc.d/  
# echo 'baculadir=yes' >> /etc/rc.conf  
#cp /usr/pkg/share/examples/rc.d/bacula-sd /etc/rc.d/  
# echo 'baculasd=yes' >> /etc/rc.conf  
#cp /usr/pkg/share/examples/rc.d/bacula-fd /etc/rc.d/  
# echo 'baculafd=yes' >> /etc/rc.conf
```

Listing 6-9: Bacula per rc.d starten

Sind alle notwendigen Prozesse gestartet, kann man mit der `bconsole` Verbindung zum Server aufnehmen. Allerdings sucht `bconsole` standardmäßig im aktuellen Verzeichnis nach der Konfigurationsdatei `bconsole.conf`. Eine Beispieldatei befindet sich in `/usr/pkg/share/examples/bacula/bconsole.conf`, so das man die Datei bspw. ins Homeverzeichnis kopieren und anpassen kann oder einfach ein passendes Alias in der Shell erzeugt.

Man kann hier Sicherungsläufe starten, stoppen oder auch überwachen. Stattdessen kann man Bacula auch in gewohnter Manier mit `vi(1)` in den `.conf`-Dateien anpassen. Standardmäßig hat Bacula bereits zwei FileSets definiert, nämlich *Full Set*, das das Quellverzeichnis von Bacula sichert, und *Catalog*, das den Index von Bacula sichert.

Neue Filesets werden in `/usr/pkg/etc/bacula/bacula-dir.conf` definiert. Dazu gibt es eine einfache Grammatik, die ausführlich im Handbuch [[Bacula User's Manual](#)] beschrieben ist. Listing 6-11 zeigt eine Beispielkonfiguration. Jedes FileSet kann beliebig mit JobDefs kombiniert werden. So kann man bspw. verschiedene Jobs definieren in dem in einem Job über einen längeren Zeitraum Komplet und Inkrementell sowie sonntags Komplet gesichert wird. Die sonntäglichen Bänder können dann archiviert oder ausgelagert werden.

### 6.2.3. Rücksicherung

Zur Rücksicherung verwendet Bacula ebenfalls wieder JobDefinitions, diese können aber manuell in der Konsole angepasst werden. In der Konsole gibt es den Befehl „`restore`“, dessen Ausgabe in 6-12 gezeigt wird.

Es bestehen die in Tabelle 6.5 automatisch konfigurierten Möglichkeiten zur Rücksicherung (gemäß Listing 6-12):

---

```
$ bconsole -c /usr/pkg/share/examples/bacula/bconsole.conf
Connecting to Director balmung:9101
1000 OK: balmung-dir Version: 1.38.2 (20 November 2005)
Enter a period to cancel a command.
*time
17-Jan-2006 21:55:38
*version
balmung-dir Version: 1.38.2 (20 November 2005)
*show filesets
FileSet: name=Full Set
  O M
  N
  I /usr/pkgsrc/sysutils/bacula/work/bacula-1.38.2
  N
  E /proc
  E /tmp
  E /.journal
  E /.fsck
  N
FileSet: name=Catalog
  O M
  N
  I /var/spool/bacula/bacula.sql
  N
```

---

Listing 6-10: Baculas bconsole starten

Tabelle 6.4.: Baculas bconsole-Befehle

Befehl	Wirkung
add	add media to a pool
autodisplay	autodisplay [on/off] – console messages
automount	automount [on/off] – after label
cancel	cancel job=nnn – cancel a job
create	create DB Pool from resource
delete	delete [pool=<pool-name>   media volume=<volume-name>]
estimate	performs FileSet estimate debug=1 give full listing
exit	exit = quit
help	print this command
label	label a tape
list	list [pools   jobs   jobtotals   media <pool>   files jobid=<nn>]; from catalog
llist	full or long list like list command
messages	messages
mount	mount <storage-name>
prune	prune expired records from catalog
purge	purge records from catalog
query	query catalog
quit	quit
relabel	relabel a tape
release	release <storage-name>
restore	restore files
run	run <job-name>
setdebug	sets debug level
show	show (resource records) [jobs   pools   ...   all]
sqlquery	use SQL to query catalog
status	status [storage   client]=<name>
time	print current time
unmount	unmount <storage-name>
update	update Volume or Pool
use	use catalog xxx
var	does variable expansion
version	print Director version
wait	wait until no jobs are running



```
FileSet {
  Name = "home"
  Include {
    Options {
      signature = MD5
      compression = GZIP
      verify = pins5
      oneofs = no
      sparse = yes
    }
    File = /home
    File = /etc
    File = /usr/pkg/etc
    File = /var/
  }
  Exclude {
    File = /home/stefan/temp
    File = /var/tmp
  }
}

JobDefs {
  Name = "HomeRun"
  Type = Backup
  Level = Incremental
  Client = balmung-fd
  FileSet = "home"
  Schedule = "WeeklyCycle"
  Storage = File
  Messages = Standard
  Pool = Default
  Priority = 10
}

Schedule {
  Name = "WeeklyCycle"
  Run = Full sun-sat at 21:00
  Run = Differential 2nd-5th sun at 21:00
  Run = Incremental mon-sat at 21:00
}
```

---

Listing 6-11: Beispiel einer Konfiguration für Bacula

---

\*restore

First you select one or more JobIds that contain files to be restored. You will be presented several methods of specifying the JobIds. Then you will be allowed to select which files from those JobIds are to be restored.

To select the JobIds, you have the following choices:

- 1: List last 20 Jobs run
- 2: List Jobs where a given File is saved
- 3: Enter list of comma separated JobIds to select
- 4: Enter SQL list command
- 5: Select the most recent backup for a client
- 6: Select backup for a client before a specified time
- 7: Enter a list of files to restore
- 8: Enter a list of files to restore before a specified time
- 9: Find the JobIds of the most recent backup for a client
- 10: Find the JobIds for a backup for a client before a specified time
- 11: Enter a list of directories to restore for found JobIds
- 12: Cancel

---

Listing 6-12: Daten mit Bacula rücksichern (1)

Tabelle 6.5.: Daten mit Bacula rücksichern (2)

- 
1. Die letzten 20 Jobs können restauriert werden
  2. Es werden alle Jobs angezeigt, die eine bestimmte Datei gesichert haben. Einer der Jobs kann restauriert werden.
  3. Eine kommaseparierte Liste mit JobIds wird übergeben.
  4. Man kann SQL-Anfragen stellen um die passende JobId zu finden. Diese ID kann dann mit Punkt 3 restauriert werden.
  5. Sichert den letzten Stand eines Clients zurück, in dem alle notwendigen Sicherungen identifiziert werden.
  6. Wie 5.), allerdings kann man ein Datum spezifizieren
  7. Eine Datei mit zurückzusichernden Dateinamen wird eingelesen
  8. Man kann Dateinamen und ein korrespondierendes Datum angeben
  9. Wie 6.), man kann danach aber mit den JobIds wie in Punkt 11.) weitermachen
  10. Wie 9.), die JobIds werden aber intern übergeben
  11. Man kann JobIDs und Pfadnamen übergeben. Es werden dann alle angegebenen Verzeichnisse (ohne Unterverzeichnisse) wiederhergestellt
  12. Bricht „restore“ ab.
-



**Teil IV.**

**PostgreSQL**



# 7. Sicherung eines PostgreSQL-Servers

Die Sicherung einer Datenbank ist einer der wichtigsten Punkte beim Betrieb eines Datenbankservers. PostgreSQL bietet verschiedene Möglichkeiten Anwendungsdaten zu sichern.

## 7.1. Den Datenbankcluster sichern

PostgreSQL legt alle Daten im sogenannten Cluster, also einem bestimmten Verzeichnis, ab. Es ist möglich dieses Verzeichnis wie normale Dateien zu sichern, nach der Wiederherstellung des Servers zurückzuspielen und PostgreSQL dann mit diesem Cluster zu starten. Problematisch bei der Sicherung ist die Konsistenz des Dateisystems, da im Cluster die Schreibaktivität entsprechend hoch ist. Abhilfe schaffen hier Dateisystem-Snapshots (siehe Kap. 5.3), die einen konsistenten Zustand des Dateisystems als Pseudogerät einbinden, das anschließend gesichert werden kann.

Ein gesicherter PostgreSQL-Cluster kann allerdings nur mit genau der selben PostgreSQL-Version gelesen werden, mit der er erzeugt wurde. Dies bedeutet das nach einem Totalausfall des Systems wieder die alte PostgreSQL-Version installiert werden muss.

Es empfiehlt sich den PostgreSQL-Cluster auf eigenen Partitionen, oder besser noch, eigenen Festplatten abzulegen. Dies verringert den Administrationsaufwand und kann den Datendurchsatz des Servers erhöhen.

Seit Version 8.0 unterstützt PostgreSQL sog. *Table-Spaces*, d.h. daß komplette Datenbanken, aber auch einzelne Tabellen, Indizes oder Schemas auf beliebige Verzeichnisse verteilt werden können. Dies ermöglicht den Einsatz mehrerer Festplatten, was Kapazität und Durchsatz erhöhen kann. Allerdings sollte beim Sichern des PostgreSQL-Clusters dann auch bedacht werden alle eingesetzten Table-Spaces mitzusichern.

Listing 7-1 zeigt wie man PostgreSQLs Cluster sichert und wieder zurückspielt.

## 7.2. Point-in-Time-Recovery

Der Einsatz von Point-in-Time-Recovery Mechanismen ermöglicht das Zurücksetzen der Datenbank in einen konsistenten Zustand, der an einem beliebigen Zeitpunkt vorgelegen hat.

---

```
# fssconfging -x -c fss0 /pgcluster0
# fssconfging -x -c fss1 /pgcluster1
# dump -0a -f /mnt/nfs/back/pcluster0.0 /dev/fss0
# dump -0a -f /mnt/nfs/back/pcluster1.0 /dev/fss1
# fssconfig -u fss0 && fssconfig -u fss1

# restore -r -f /mnt/nfs/back/pcluster0.0
# restore -r -f /mnt/nfs/back/pcluster1.0
# chown postgres.postgres /pgcluster0 /pgcluster1
# su -m postgres -c 'pg_ctl -D /pgcluster0 -l /pgcluster0/logfile start&'
# su -m postgres -c 'pg_ctl -D /pgcluster1 -l /pgcluster1/logfile start&'
```

---

Listing 7-1: PostgreSQL-Cluster mit dump sichern

PostgreSQL ab Version 8.0 verwendet die sogenannten „*Write Ahead Logs*“ (WAL) um jede Änderung am Datenbestand mitzuschneiden. Muss das System zurückgesetzt/-sichert werden, können die Änderungen seit dem letzten Checkpoint abgespielt werden, um den letzten konsistenten Zustand zu erreichen. Befindet sich der Cluster in einem inkonsistenten Zustand, wird durch einspielen der WALs wieder ein konsistenter Zustand erreicht.

Die WALs werden ähnlich dem Journal eines Journaling-Dateisystems fortlaufend weitergeschrieben. So werden alle Transaktionen<sup>1</sup> protokolliert und können bei Bedarf neu eingespielt werden. Innerhalb einer bestimmten Zeit werden die Transaktionen dann auf den eigentlichen Datenbestand übernommen. Dies erhöht dabei sogar den Datendurchsatz, da es einfacher ist Daten einfach an eine Datei anzuhängen, als ständig im Datenbestand der Datenbank hin- und herzuspringen.

Wenn man den Datenbankcluster und alle WAL-Logs danach sichert, kann man einen beliebigen Zustand der Datenbank im Zeitfenster der Logs wiederherstellen.

Da alle Transaktionen im Log sequentiell fortgeschrieben werden, würden die Protokolle unaufhörlich wachsen und irgendwann das Dateisystem überlaufen lassen. Daher werden die Logs ganz einfach rotiert, das heißt man definiert<sup>2</sup> einfach wieviele Dateien es geben soll, und wenn diese Zahl erreicht ist wird wieder in die erste Datei geschrieben. Um die Logdateien zu sichern, muss man daher die Logs vor der Rotation auf einen anderen Datenträger kopieren. Dies geschieht, in dem man in der postgresql.conf die Option `archive_command` wie in Listing 7-2 beschrieben setzt, so das die Logdateien vor dem Überschreiben in ein anderes Verzeichnis kopiert werden. Dabei sollte darauf geachtet werden, daß der Befehl auch ausgeführt wird, da ihn PostgreSQL sonst bis zum Gelingen

---

<sup>1</sup>PostgreSQL behandelt *jede* DML-Aktion als ACID-konforme Transaktion in Sinne der Datenbankentheorie

<sup>2</sup>Standardmäßig existieren in `pg_xlog/` drei WAL Dateien mit je 16MB Größe. Diese Größe kann beim kompilieren des Servers angegeben werden, die Anzahl in `postgresql.conf`



wiederholt. So kann es unter Umständen vorkommen, daß das Dateisystem mit WAL-Dateien vollläuft, da die Rotation der Logs nicht durchgeführt werden kann.

---

```
archive_command = 'cp %p /usr/backup/postgres-wal/%f'
```

---

Listing 7-2: postgresql.conf-Konfig zur Sicherung der Logs

Um eine Sicherung des Datenbankclusters durchzuführen, muss man die Datenbank vor- und nachbereiten. Dazwischen kann der Datenbankcluster gesichert werden. Dies geschieht mit den SQL-Befehlen in Listing 7-3, dort wird der Datenbankcluster zur Sicherung vorbereitet und gesichert, anschließend wird der Backup-Checkpoint entsperrt und es können die neuen WALs rotiert werden. Hat man einen derart gesicherten Datenbankcluster, reicht es aus die WALs die danach erzeugt werden zu sichern.

---

```
$ echo "select pg_start_backup('Label');" | psql -U postgres template1
# fssconfig -x -c fss0 /pgcluster /
# dump -0 -f /usr/backuppgcluster.0 /dev/fss0
# fssconfig -u fss0
$ echo "select pg_stop_backup();" | psql -U postgres template1
```

---

Listing 7-3: PostgreSQL-Datenbankcluster und WALs sichern

Um ein derartig gesicherten Datenbestand zurückzuspielen, sind folgende Schritte notwendig:

1. PostgreSQL neu installieren oder anhalten
2. Sicherung des Datenbankclusters zurückspielen
3. recovery.conf erstellen und
4. `restore_command` und `recovery_target_time` setzen
5. PostgreSQL starten

Punkt 2 wird erledigt, indem man den Dump aus Listing 7-3 mit `restore -r -f /usr/backuppgcluster.0` zurückspielt und ggf. die Dateirechte anpasst.

Punkt 3 und 4 setzen die entsprechende Befehle, die der Postmaster ausführen soll. `restore_command` ist der Befehl, um die WALs zurückzukopieren, also etwas derart: `restore_command=cp /usr/backup/postgres-wal/%f %p`, `recovery_target_time` erwartet einen Zeitstempel, bis zu dem rückgesichert werden soll.

### 7.3. pg\_dump, pg\_dumpall und pg\_restore

PostgreSQL verfügt auch über ein Werkzeug um logische Backups zu erzeugen. Dabei werden SQL-Befehle zum Erstellen der Datenbank(en) in eine einfache ASCII-Textdatei oder ein Archiv geschrieben. Die entstandene ASCII-Datei kann danach mit Sicherungsmechanismen für Dateisysteme gesichert werden.

Um eine Datenbank zu sichern, benutzt man `pg_dump(1)`, um alle Datenbanken in eine Datei zu sichern existiert `pg_dumpall(1)`.

---

```
$ /usr/pkg/bin/vacuumdb -Upgoperator -f -z meineDB

$ /usr/pkg/bin/pg_dump -Fc -Upgoperator -meineDB >
    /home/pgsql/meineDB_`date +%y%m%d`

$ /usr/pkg/bin/pg_dumpall > pg_`date +%y%m%d`
```

---

Listing 7-4: PostgreSQL-Datenbanken sichern

Der erste Befehl in Listing 7-4 bereinigt die Datenbank „meineDB“ als PostgreSQL-Benutzer „pgoperator“. Der dritte Befehl schreibt die Datenbank mit LOBs komprimiert in eine Sicherungsdatei.

Der letzte Befehl verwendet `pg_dumpall(1)` um alle Datenbanken in eine Datei zu sichern.

---

```
$ createdb meineDB
$ pg_restore -d meineDB -f meineDB_051206
$ psql -f pg_051206 template1
```

---

Listing 7-5: Rückspielen von PostgreSQL-Sicherungen

Zum Rückspielen einer mit `pg_dump(1)` erzeugten Sicherung verwendet man `pg_restore(1)`. Dazu muss die rückzusichernde Datenbank vorher allerdings schon mit `createdb(1)` erstellt worden sein. Anders verhält es sich mit `pg_dumpall(1)`, denn dessen Sicherung enthält alle Befehle um die gesicherten Datenbanken neu anzulegen, so das man sich mit einer beliebigen Datenbank verbinden kann.

### 7.4. Replikation

Replikationssysteme synchronisieren den Datenbestand von mehreren Servern. Dies kann auf verschiedene Arten geschehen, beispielsweise synchron/asynchron oder voll/teilweise.

Somit ist es möglich ein Ersatzsystem bereitzuhalten, das per Replikation auf dem aktuellen Stand des Originals gehalten wird und bei dessen Ausfall dessen Funktion übernehmen kann.

Asynchrone Replikation verteilt die Daten erst nach einem erfolgreichen `„BEGIN ... COMMIT“`-Block auf die angeschlossenen Replikanten. Dies schränkt das System aber etwas ein:

- nur ein einziger Master sinnvoll, da sonst Konsistenzprobleme drohen (OIDs, Primärschlüssel)
- keine Lastverteilung vorgebar

Synchrone Replikation hingegen verteilt die Daten sofort bei Beginn der Transaktion, so dass diese auf allen Rechnern ausgeführt wird. Hierbei ist allerdings die Konsistenz der Rechner untereinander ein Problem, denn es muss auf allen Maschinen ein erfolgreicher `COMMIT` garantiert werden. Die Slaves informieren nach einem erfolgten Schreibzugriff den Master von der Transaktion, so dass weitere Schreibtransaktionen ausgeführt werden können. Dieses Verfahren wird Zwei-Phasen-Commit genannt, da der Master jedesmal auf die Slaves warten muss, bevor eine Transaktion endgültig für den gesamten Rechnerverbund als `committed` markiert wird. So wird zwar eine sofortige Replikation des Datenbestandes erreicht, allerdings auf Kosten des Durchsatzes, da diese Transaktion eben auf jedem Rechner erfolgen muss und der Erfolg an den Master zurückgemeldet werden muss.

### 7.4.1. Synchroner Replikation mit Pgpool

Pgpool fungiert als sogenannter *„Connection Pool“*, d.h. er klinkt sich zwischen den eigentlichen PostgreSQL-Server und die Anwendungen. Dies funktioniert im Prinzip wie bei einem transparenten Proxy. Pgpool cached Verbindungen zum Datenbankserver, um so Lasten durch Verbindungsauf- und -abbau zu reduzieren. Weiterhin kann sich Pgpool mit zwei PostgreSQL-Servern verbinden und so im Falle eines Ausfalls auf den anderen, noch funktionierenden, Server umschalten.

Zwischen den beiden PostgreSQL-Servern kann Pgpool außerdem als synchroner Replikationsserver fungieren, d.h. alle Datenbankanfragen werden an beide Postmaster geschickt. Dies ermöglicht eine einfache Replikation des Datenbestandes auf zwei verschiedene Rechner, die lediglich durch das Netzwerk miteinander verbunden sein müssen. Diese Methode schließt allerdings einige Operationen aus die bspw. Abhängig vom Server (Abfrage der OID, eines Zeitstempels oder ähnliches) oder eben nicht deterministisch (z.B. Zufallsgenerator) sind.

Pgpool lässt sich aus `pkgsrc/databases/pgpool` installieren. Zur Konfiguration genügt es `/usr/pkg/share/examples/pgpool.conf.sample` nach

/usr/pkg/etc/pgpool.conf zu kopieren und anzupassen. Die Konfigurationsdatei ist wohldokumentiert und recht einfach zu verstehen. Wichtig sind folgende Optionen:

- **listen\_addresses** und **port** zu benutzende Netzwerkadresse und Port
- **backend\_host\_name** und **backend\_port** Netzadresse und Port des PostgreSQL-Servers
- **secondary\_backend\_host\_name** und **secondary\_backend\_port** Netzadresse und Port des zweiten PostgreSQL-Servers (Slave)
- **replication\_mode** Einsatz als Replikationssystem
- **replication\_strict** Wenn aktiviert, werden Deadlocks vermieden, was auf Kosten des Durchsatzes geht.
- **replication\_timeout** Wenn `replication_strict` deaktiviert ist könnten Deadlocks auftreten. Dieser Wert gibt den Timeout in  $\mu s$  an, nachdem die blockierte Transaktion abgebrochen wird.
- **replication\_stop\_on\_mismatch** Der Replikationsmodus soll bei inkonsistenten Daten zwischen Master und Slave abgebrochen werden.

Hat man Pgpool wie in Listing (7-6) konfiguriert, kann man sich an localhost:9999 mit dem Datenbankterminal verbinden. Alle DML-Operationen werden dann auf beiden PostgreSQL-Servern ausgeführt.

Mit `pgpool switch` kann man die Server umschalten, bspw. mit `pgpool -s master switch` die Verbindung zum Master beenden und auf den Secondary Server umschalten.

---

```
listen_addresses = 'localhost'
port = 9999

socket_dir = '/tmp'

backend_host_name = '192.168.2.1'
backend_port = 5432

backend_socket_dir = '/tmp'

secondary_backend_host_name = '192.168.2.2'
secondary_backend_port = 5432

replication_mode = true
replication_strict = true
replication_timeout = 5000

replication_stop_on_mismatch = false

reset_query_list = 'ABORT; RESET ALL; SET SESSION AUTHORIZATION DEFAULT'

print_timestamp = true
```

---

Listing 7-6: Pgpool-Konfiguration für die Replikation



**Teil V.**

**Sonstige Programme**





## 8. Sonstige Programme

### 8.1. Magnetbänder ansteuern mit mt(1)

mt(1) ist kein eigentliches Sicherungsprogramm, sondern das „*Magnetic Tape Manipulation Programme*“. Dieses Programm dient dazu Bandlaufwerke zu bedienen indem diese bspw. zurückgespult und offline geschaltet oder 3 Dateien vorgespult werden.

---

```
# mt rewoffl
# mt -f /dev/nrst0 fsf 3
```

---

Listing 8-1: Bänder mit mt manipulieren

Der erste Befehl spult das Band zurück und schaltet es offline, der Zweite spult drei Dateien auf dem Nonrewinding Tapedevice /dev/nrst0 vor.

### 8.2. Datenströme puffern

Bandlaufwerke arbeiten mit einer festen Laufgeschwindigkeit des Bandes. Kommt der schreibende Prozess nicht mit dem Anliefern der Daten nach, wird das Band vor- und zurückgespult. Dies ist schlecht für den Durchsatz und belastet das Band und das Laufwerk mechanisch. Um derartiges Verhalten zu verhindern, gibt es Programme, die den Datenstrom zwischenpuffern. Lediglich dump(8) verfügt über eingebaute Pufferungsmechanismen.

Team und Buffer sind aus pkgsrc/misc installierbar. Sie werden einfach in eine Pipe zwischengeschaltet (siehe Listing 8-3) und leiten die Ausgabe entsprechend auf das Bandlaufwerk um.

Problematisch ist hierbei die Fehlerbehandlung in Skripten, da ja nun mehrere Programme in der Pipe verkettet werden. Außerdem funktioniert die EOM-Erkennung der Sicherungsprogramme nicht mehr. Lediglich Buffer ist in der Lage das Bandende selbständig zu erkennen.

Befehl	Wirkung
asf	spule $n$ Dateien vom Anfang des bandes vor
eof, weof	schreibe $n$ EOF-Marken an jetziger Bandposition
fsf	spule $n$ Dateien vor
fsr	spule $n$ Blöcke vor
bsf	spule $n$ Dateien zurück
bsr	spule $n$ Blöcke zurück
rewind	zurückspulen
offline, rewoffl	Zurückspulen und Streamer offline schalten (Band auswerfen)
status	Status des Streamers angeben
retension	Band spannen (geräteabhängig)
erase	Band löschen (geräteabhängig)
eew	end-of-media Frühwarnung mit 0 deaktivieren, 1 aktiviert
eom	Bis zum Ende der Aufzeichnung (EOF-Marke) vorspulen
blocksize, setblk	Setzt die Blockgröße auf $n$ , 0 ist variable Blockgröße
density, setdensity	Setzt die Banddichte
rdspos	logische Bandposition anzeigen (geräteabhängig)
rdhpos	physikalische Bandposition anzeigen (geräteabhängig)
setspos	logische Bandposition setzen (geräteabhängig)
sethpos	physikalische Bandposition setzen (geräteabhängig)
compress	1 aktiviert Komprimierung, 0 deaktiviert

Listing 8-2: mt(1)-Optionen

---

```
# tar c f - /home | buffer -o /dev/nrst0
```

---

Listing 8-3: Datenströme puffern

### 8.3. Dateien mit `split(1)` zerlegen

Ist eine Datei zu groß um auf ein Medium, bspw. eine CD oder ein Band, zu passen, kann man sie mit `split(1)` zerlegen. Split erwartet lediglich die Größe der zu erzeugenden Dateien (in Byte, Kilobyte oder Megabyte), den Namen der zu zerlegenden Datei und optional einen Präfix für die erzeugten Dateien. Standardmäßig generiert `split` Dateinamen für die Ausgabe selbst, und zwar nach dem Muster `x[a-z][a-z]`. Gibt man einen Präfix an, wird dieser der Art `präfix[a-z][a-z]` gebraucht. Gesplittete Dateien lassen sich einfach mit `cat(1)` wieder zusammensetzen.

---

```
$ split -b 3m netbsd kernel
$ ls -lh
total 17M
-rw-r--r--  1 stefan  stefan  3.0M Feb 18 20:34 kernelaa
-rw-r--r--  1 stefan  stefan  3.0M Feb 18 20:34 kernelab
-rw-r--r--  1 stefan  stefan  2.5M Feb 18 20:34 kernelac
-rwxr-xr-x  1 stefan  stefan  8.5M Feb 18 20:33 netbsd

$ for i in kernela*
> do
> cat $i >> neu ;
> done
$ md5 netbsd neu
MD5 (netbsd) = 06c31725e23e432e4e040c1e37ac0e16
MD5 (neu) = 06c31725e23e432e4e040c1e37ac0e16
$
```

---

Listing 8-4: Dateien zerlegen und zusammensetzen

### 8.4. Dateien mit `find(1)` finden

Find kann dazu benutzt werden Dateien zu finden. Es kann mit verschiedenen Optionen gefüttert werden, bspw. Regulären Ausdrücken zum Dateinamen oder Zeitpunkte, an denen Dateien geändert wurden.

Listing 8-5 zeigt einige Beispiele für `find(1)`. Der erste Befehl gibt die Pfade aller `*.tex` und `*.pdf` in `/home` aus. Dabei kann anstatt des Sterns auch ein beliebig komplexer Regulärer Ausdruck verwendet werden.

Zeile 2 und 3 verknüpfen `find(1)` mit `grep(1)` – allerdings mit einem kleinen Unterschied. Zeile 2 sucht nach „test“ im *Dateinamen* – findet also nur Dateien

```
1$ find /home/ -name '*.tex' -or -name '*.pdf'
2$ find /home/ -name '*.tex' -printx | grep test
3$ find /home/ -name '*.tex' -printx | xargs grep test
4$ find /home -mtime 7
5$ find /home/ -user www
```

---

Listing 8-5: Dateien mit find(1) finden

die irgendwie „test“ heißen. Zeile 3 hingegen übergibt den Pfadnamen der gefundenen Dateien an grep(1), so daß grep(1) den *Dateiinhalte* untersucht.

Die 4. Zeile sucht nach Dateien, die vor 7 Tagen geändert wurden. Zeile 5 schließlich, sucht nach Dateien die dem Benutzer „www“ gehören.

## 8.5. Das kryptographische Dateisystem CFS

CFS ist das *Cryptographic Filesystem* von AT&T. Die Dateien liegen hierbei verschlüsselt auf der Festplatte und werden über einen NFS ähnlichen Mechanismus entschlüsselt zur Verfügung gestellt. Für ein Backup heisst dies, das die Dateien entweder entschlüsselt gesichert werden und das Archiv danach entsprechend zu schützen ist, oder aber das die verschlüsselten Dateien gesichert werden. Dabei ist es aber nachteilig das die verschlüsselten Dateien auch einen verschlüsselten Dateinamen haben und so nicht sehr einfach zu identifizieren sind. Daher sollte man hierbei auf find(1) oder entsprechende Datumsmechanismen der Programme vertrauen und inkrementell nach Zeit sichern.

Ich setze cfs in meinem Homeverzeichnis seit mehreren Jahren ein und konnte bisher keine Probleme mit dump/restore und verschiedenen Leveln feststellen, da sich cfs-verschlüsselte Verzeichnisse wie ganz normale Dateien verhalten.

## 8.6. Das kryptographische Pseudogerät CGD

cgd(4) ist der Cryptographic Devicedriver, der den Einsatz verschlüsselter Partitionen ermöglicht. Wird cgd(4) verwendet um eine Partition zu schützen, ist die Backupstrategie anzupassen d.h. die Partition muss entschlüsselt eingebunden und die erzeugten Archive verschlüsselt werden.

Als Beispiel wird in Listing 8-6 auf einem Rechner die Partition /dev/wd0e via cgd(4) verschlüsselt und als /dev/cgd0d nach /home gemountet.

Die Sicherung erfolgt im eingemounteten Zustand, die Partition ist also erst eingebunden worden. Danach kann wie gewohnt mit dump gesichert werden. Damit die Sicherung nicht ungeschützt geschrieben wird, wird sie per Pipe durch mcrypt verschlüsselt.

```
# cgdconfig cgd0 /dev/wd0e && mount /dev/cgd0d /home
# dump -0au -f - /dev/cgd0d | mrcrypt -flush > home0.nc
```

---

Listing 8-6: eine CGD-Partition verschlüsselt dumpen

## 8.7. symmetrische Verschlüsselung mit mrcrypt

mrcrypt ist als Ersatz und Nachfolger für das alte bdes(1) gedacht und unterstützt eine Vielzahl an symmetrischen Kryptoalgorithmen. Es ist recht schnell und bietet sich als symmetrisches Verschlüsselungsprogramm für die erstellten Backups an. Es ist in `pkgsrc/security/mrcrypt` enthalten und lässt sich problemlos installieren.

---

```
$ ls -l
total 20
-rw-r--r--  1 stefan  stefan  9143 Jan 16 11:55 1.txt
$ mrcrypt 1.txt
Enter the passphrase (maximum of 512 characters)
Please use a combination of upper and lower case
                                letters and numbers.
Enter passphrase:

Enter passphrase:

File 1.txt was encrypted.
$ ls -l
total 40
-rw-r--r--  1 stefan  stefan  9143 Jan 16 11:55 1.txt
-rw-----  1 stefan  stefan  9245 Jan 16 11:55 1.txt.nc
```

---

Listing 8-7: symmetrische Verschlüsselung mit mrcrypt

Listing 8-7 fragt nach einem Passwort und erzeugt damit die Datei `1.txt.nc`, die standardmäßig mit Rijndael (AES) verschlüsselt wird. `mrcrypt` unterstützt mehrere Algorithmen und Chiffre Modi, diese lassen sich in der manpage nachlesen oder mit `mrcrypt -list` anzeigen.

## 8.8. Versionsverwaltung mit CVS

CVS (Concurrent Version System) ist eigentlich kein Backupprogramm, sondern

---

```
$ mdecrypt 1.txt.nc
Enter passphrase:

File 1.txt.nc was decrypted.
$ ls -l
total 40
-rw----- 1 stefan stefan 9143 Jan 16 11:55 1.txt
-rw----- 1 stefan stefan 9253 Jan 16 11:55 1.txt.nc
$
```

---

Listing 8-8: Datei mit mdecrypt entschlüsseln

ein System aus verschiedenen Programmen, das entwickelt wurde um Quellcodeprojekte für mehrere Entwickler zu verwalten. Da CVS aber auch eine Versionierung der Dateien vornimmt, kann man es hervorragend verwenden um die verschiedenen Entwicklungsstadien einer Datei vorzuhalten. Prinzipiell kann man in CVS alle Arten von Quellcode (also plaintext) vorhalten, von Perl über Java zu HTML.

Entwickelt man nun ein größeres Projekt, bspw. dieses Dokument, kann man ein neues CVS-Modul anlegen und den Quellcode dort einfügen. Verändert man die Quellen, kann man sie in das CVS Repository einfügen und bei Bedarf wieder auschecken. Man kann also bspw. den Quellcode vom 19.04.2004 auschecken und mit dem vom 01.01.2005 vergleichen.

CVS ist in NetBSD standardmäßig enthalten und kann sofort benutzt werden.

## 8.9. Dateisystemintegrität prüfen

mtree(8) ist ein Programm um die Eigenschaften einer Dateisystemhierarchie in eine einfache Textdatei zu schreiben. Es wird hauptsächlich eingesetzt um bei der Installation von NetBSD-Sets deren Korrektheit zu überprüfen, da bspw. ein defektes `/bin/sh` fatale Folgen hätte.

Mtree kann natürlich auch genutzt werden um die Integrität eines Backups zu testen, in dem man einen Fingerabdruck der zu sichernden Daten erstellt und mit der Sicherung vergleicht.

---

```
# mtree -L -c -K sha1,rmd160,gname,uname,mode \
-p / -X /etc/mtree.excl > /root/mtree.orig
```

---

Listing 8-9: Fingerabdruck eines Systems mit mtree erstellen

Listing 8-9 erzeugt eine Liste mit den entsprechenden Dateieigenschaften, wobei die Pfade in `/etc/mtree.excl` ausgeschlossen werden. Die zu bearbeitenden Dateieigenschaften werden über die Schlüsselwörter `sha1`, `rmd160`, `gname`, `uname`, `mode` definiert, hier sind dies die beiden Prüfsummenverfahren SHA1 und RipeMD 160 sowie der Name der Gruppe und des Besitzers sowie der Dateizugriffsrechte im Oktalmodus. Weitere Schlüsselwörter kann man der `manpage` entnehmen, sind aber meiner Meinung nach nicht notwendig.

Ein Teil der erzeugten Liste ist exemplarisch in Listing 8-10 abgedruckt.

Mit Listing 8-11 können Dateien verglichen werden, dazu gibt `mtree` einen Bericht der Art 8-12 aus. Die Datei `new1` ist neu, existiert also im alten Fingerabdruck nicht, die Datei `removed.txt` existiert zwar im Fingerabdruck, aber nicht im System. `3.txt` wurde verändert, sie beinhaltet nun 6489 Byte statt 6460. Dadurch veränderten sich auch die Zugriffszeit und die Prüfsummen.

---

```
cat      size=10550 time=1117454217.0 \  
         rmd160=1665d6dc5bee4c14fd4f74a3c8c9197e9006d761 \  
         sha1=428bff79c7a1c7c925a584ba5a983411a42befa7  
chio    size=13730 time=1117454217.0 \  
         rmd160=dbff324c1b19d37e9281805c6deeb8b6479742a6 \  
         sha1=37c78289531ec99489bc8b1aaeb60fc1b8a5fb8f  
chmod   size=8594 time=1117454217.0 \  
         rmd160=436ff626a61fecf9569abb89aebb8dac060096da \  
         sha1=e390a28d3726aedd472838f8f55e67287970272b  
[...]
```

---

Listing 8-10: Beispiel eines `mtree`-Reports

---

```
mtree -L -p / -X /etc/mtree.excl -f /root/mtree.orig
```

---

Listing 8-11: Dateisystem mit Fingerabdruck vergleichen

Alles in allem ist `mtree` also ein vorzügliches Mittel um die Korrektheit eines Dateisystems zu prüfen und zu vergleichen.

## 8.10. Prüfsummen einzelner Dateien erstellen

Um einzelne Dateien zu vergleichen reicht ein einfaches Prüfsummenverfahren wie `sha1(1)` aus.

---

```
3.txt: size (6460, 6489)
      modification time (Wed Feb  9 18:48:17 2005,\
                          Wed Feb  9 18:54:52 2005)
      rmd160 (0xf30de94ae188d9114a00118e80bbedf8e2c6c6ed, \
              0x6a5b3781799b09e2b8d8037e89fe03957ebca247)
      sha1 (0x4c499470dcaed01c2681eb0676ddac8cf3024d5a, \
            0xc04159927a4e9ca5f521161d2cc8daa019fca19a)
extra: new1
missing: removed.txt
```

---

Listing 8-12: Ergebnisse eines Dateisystemvergleichs

---

```
# dump -0 -f home.dump /home
# sha1 home.dump >> SHA1.dump
# bzip2 -9 home.dump
# sha1 home.dump.bz2 >> SHA1.dump
# mount /dev/sd0a /mnt/zip
# cp home.dump.bz2 SHA1.dump /mnt/zip
# sha1 /mnt/zip/home.dump.bz2
```

---

Listing 8-13: Dateien mit SHA1 vergleichen



**Teil VI.**

**Der Test**



## 9. Der Test

*„Nobody gonna take my head  
I got speed inside my brain“*  
Deep Purple – Highway Star

Elizabeth D. Zwicky hat in ihren Artikeln [Zwi91] und [Zwi03] zur Zuverlässigkeit von Datensicherungsprogrammen verschiedene Pakete auf Unix-Systemen getestet. Dazu erstellte sie ein Perlskript um ein Testverzeichnis mit verschiedenen Problemedateien zu generieren. Ich habe dieses Perlskript<sup>1</sup> verwendet um ein Testverzeichnis („Quelle“) zu erstellen und es mit verschiedenen Programmen zur Datensicherung getestet.

### 9.1. Problemfelder

Zwicky beschreibt die Tests und deren Hintergründe ausführlich in ihrem o.g. Artikel, ich gebe hier lediglich eine kurze Zusammenfassung wieder.

#### 9.1.1. Datumsprobleme

Eine Datei hat eine mtime vor 1970, eine weitere in der Zukunft.

#### 9.1.2. Dateigrößen

Eine leere Datei, ein leeres Verzeichnis und eine Datei mit 4098 MB wurde erstellt.

#### 9.1.3. Verschiedene Dateitypen

Es wurden drei verschiedene Dateitypen erzeugt: `block`, `character` und `named` (auch als FIFO bezeichnet).

---

<sup>1</sup><http://greatcircle.com/~zwicky/torture.tar> in der Version vom 04.11.2005

#### 9.1.4. Zeichensätze

Es wurden Verzeichnisse, Dateinamen, Hardlinks und Symlinks angelegt, in dem einfach die 7-Bit-ASCII-Tabelle und UTF8 von 128 – 255 durchiteriert wurde.

Beim Erzeugen der Quelle gab es hierbei sechs Fehler, da Dateien bzw. Pfade mit `.` oder `/` verständlicherweise als Namen nicht erzeugt werden können. Trotzdem können andere Betriebssysteme derartige Dateinamen mit NFS, AFS oder Sambe erzeugen.

#### 9.1.5. Löcher in Dateien

Löcher in Dateien werden erzeugt, wenn Blöcke referenziert werden, die Daten enthalten, die Referenz aber `NULL` ist.

Erzeugt werden sie im Allgemeinen von Coredumps oder bspw. in Datenbankpools oder explizit mit `seek(2)`, in dem man  $n$  Blöcke vorspult.

#### 9.1.6. lange Pfadnamen

Die Länge eines Dateinamens oder einzelnen Verzeichnisses (`MAXNAMELEN`) ist auf 255 Zeichen, und der gesamte Pfad (`MAXPATHLEN`) auf 1023 (`MAXPATHLEN-1`) Zeichen begrenzt.

Übersteigt ein Element des Pfadnamens `MAXNAMELEN` oder der gesamte Pfad `MAXPATHLEN` wird mit Fehler „63 ENAMETOOLONG File name too long“ abgebrochen. Näheres dazu findet sich in `errno(2)`.

Einige Sicherungsprogramme haben eingebaute maximale Pfadlängen, `pax` verweigerte sich bspw. ab 100 Zeichen.

Leider ist es möglich überlange absolute Pfadnamen zu erzeugen, da zwar der relative, nicht aber der absolute Pfad beim Erzeugen einer Datei überprüft wird. Das heißt, ein Benutzer kann eine Datei oder ein Verzeichnis mit 255 Zeichen langem Namen problemlos erzeugen und dies beliebig oft aneinanderreihen, wenn er relative Pfade verwendet.

Ein Beispiel zur Illustration, (`[1-7]NAME`, `[1-2]DATEI` sei je genau 250 Zeichen lang):

```
1# cd /
2# mkdir 1NAME && cd 1NAME
3# mkdir 2NAME && cd 2NAME
4# mkdir 3NAME && cd 3NAME
5# mkdir 4NAME && cd 4NAME
6# mkdir 5NAME && cd 5NAME
9# date > 1DATEI
```

```
10# cp 1DATEI /2DATEI
11# cd /
12# cp /1NAME/2NAME/3NAME/4NAME/5NAME/1DATEI /3DATEI
cp: 1NAME/2NAME/3NAME/4NAME/5NAME/1DATEI File name too long
```

Jeder Dateiname bzw. jede Pfadkomponente ist 250 Zeichen lang, was auch erlaubt ist. Wenn die Datei aber wie in Zeile 12 absolut referenziert wird, ergibt sich ein Pfadname von  $6 * 250 + 6 = 1506$  Zeichen, was vom Kernel nicht mehr verarbeitet wird. Ein mögliche Gegenmaßnahme wäre das begrenzen der maximalen Pfadlänge auf  $MAXPATHLEN - MAXNAMELEN - 1 = 1024 - 255 - 1 = 768$  Zeichen.

### 9.1.7. Zugriffsrechte

Einige Programme haben Probleme mit „komischen“ Zugriffsrechten, wenn bspw. ein Verzeichnis nicht gelesen oder beschrieben werden darf aber trotzdem schon Dateien beinhaltet. Die korrekte Vorgehensweise zur Sicherung ist hierbei einfach: Verzeichnis mit `rxwxrwxrwx` erstellen, Dateien ins Verzeichnis rücksichern, Zugriffsrechte setzen. `tar` ist hierbei bspw. grandios mit 100% Fehlerquote gescheitert.

### 9.1.8. Testgestaltung und Testdurchführung

NetBSD 3.0\_BETA vom 04.11.2005 mit GENERIC  
Pentium III 500MHz, 2\*128MB SDRAM 100MHz  
20GB IDE Systemplatte  
40GB IDE Festplatte, eine Partition mit FFSv1

---

```
wd1 at atabus1 drive 0: <SAMSUNG SP0411N>
wd1: drive supports 16-sector PIO transfers, LBA48 addressing
wd1: 38204 MB, 77622 cyl, 16 head, 63 sec, 512 bytes/sect x 78242976 sectors
wd1: 32-bit data port
wd1: drive supports PIO mode 4, DMA mode 2, Ultra-DMA mode 5 (Ultra/100)
wd1(piixide0:1:0): using PIO mode 4, Ultra-DMA mode 2 (Ultra/33) (using DMA)
```

---

Listing 9-1: Testgestaltung für den Belastungstest

Der Rechner wurde acht Stunden mit [prime95](#) erfolgreich einem Belastungstest unterzogen. Hierbei werden große Primzahlen berechnet und verglichen, so dass bei einem Hardwaredefekt (insbesondere RAM/CPU) sehr schnell Fehler auftreten. Zusätzlich wurden insgesamt fünf je 10GB große Dateien mit OpenSSL verschlüsselt, entschlüsselt und verglichen um Hardwarefehler auszuschließen.

Auf dem Testsystem wurde mit `perl maketestdir.pl` das Quellverzeichnis erstellt. Das Quellverzeichnis wurde vermessen (Größe mit `du` und Anzahl der Dateien mit `ls -laR | wc -l`) und mit `mtree` wurde ein Fingerabdruck der Quellen erstellt, um die Korrektheit des Dateiinhalts zu prüfen.

Mit jedem Sicherungsprogramm wurde eine Sicherung durchgeführt und die gesicherte Datei wieder zurückgespielt. Dabei wurden mit `script` alle Ausgaben<sup>2</sup> zur Auswertung mitgeschnitten und mit `time` die Zeit genommen. Anschließend wurde die Quelle mit dem Ziel verglichen, zuerst wurde mit `mtree` die Korrektheit der Daten verifiziert und danach die Quelle wieder größen- und zahlenmäßig vermessen.

Jedes Programm wurde dreimal getestet, es konnten aber keine Unterschiede in der Fehlerquote festgestellt werden.

## 9.2. Testergebnisse

In den folgenden Tabellen sind die Ergebnisse des Vergleichs von Quelle und Ziel wiedergegeben. Zu jedem getesteten Programm wurde die Anzahl der Dateien und deren Größe je Testverzeichnis aufgelistet. Zusätzlich wird die Differenz zur Quelle und die Erfolgsquote in Prozent angegeben.

---

<sup>2</sup>Teilweise mehr als 80MB

Tabelle 9.1.: Auswertung von dump und tar

Verzeichnisse	Original		dump			tar				
	Größe	Anzahl	Größe	$\delta$	Anzahl	$\delta$	Größe	$\delta$	Anzahl	$\delta$
dates	6	5	6	0	5	0	0	-6	0	-5
filesize	4195364	11	4195364	0	11	0	0	-4195364	0	-11
filetypes	2	6	2	0	6	0	2	0	6	0
funnydirname	2036	3556	2036	0	3556	0	2036	0	3556	0
funnyfilenames	1022	512	1022	0	512	0	1022	0	512	0
funnyhard	1022	510	1022	0	510	0	1022	0	510	0
funnysym	14	1013	14	0	1013	0	14	0	1013	0
holes	196	7	196	0	7	0	0	-196	0	-7
longfilenames	1098	510	1098	0	510	0	1062	-36	496	-14
longhardlinks	2	3	2	0	3	0	0	-2	0	-3
longpathnames	280550	131071	280526	-24	131060	-11	0	-280550	0	-131071
longsymlinks	259936	131827	259936	0	131827	0	237954	-21982	120785	-11042
overmaxpathlen	22	33	8	-14	21	-12	0	-22	0	-33
permissions	24722	32819	24722	0	32819	0	0	-24722	0	-32819
plainfilenames	514	258	514	0	258	0	514	0	258	0
<b>Summe</b>	4766506	302141	4766468	-38	302118	-23	243626	-4522880	127136	-175005
<b>Quote</b>	100%	100%	99,99%		99,99%		5,11%		42,08%	

Legende:

Größe = Größe der Datei / des Verzeichnisbaumes in KB, gemessen mit `du -sk`;

Anzahl = Anzahl aller Dateien / Verzeichnisse, gemessen mit `ls -laR | wc -l`;

$\delta$  = Differenz zwischen Quelle und Sicherung

Tabelle 9.2.: Auswertung von pax und cpio

Verzeichnisse	Original		pax				cpio			
	Größe	Anzahl	Größe	$\delta$	Anzahl	$\delta$	Größe	$\delta$	Anzahl	$\delta$
dates	6	5	6	0	5	0	6	0	5	0
filesize	4195364	11	4195364	0	11	0	4	-4195360	10	-1
filetypes	2	6	2	0	6	0	2	0	6	0
funnydirname	2036	3556	2036	0	3556	0	2028	-8	3538	-18
funnyfilenames	1022	512	1022	0	512	0	1018	-4	508	-4
funnyhard	1022	510	1022	0	510	0	1022	0	510	0
funnysym	14	1013	14	0	1013	0	14	0	1013	0
holes	196	7	100	-96	6	-1	52	-144	5	-2
longfilenames	1098	510	404	-694	198	-312	1062	-36	496	-14
longhardlinks	2	3	2	0	3	0	2	0	3	0
longpathnames	280550	131071	53340	-227210	26535	-104536	271340	-9210	127487	-3584
longsymlinks	259936	131827	9560	-250376	7407	-124420	259936	0	131827	0
overmaxpathlen	22	33	2	-20	3	-30	8	-14	21	-12
permissions	24722	32819	24722	0	32819	0	24722	0	32819	0
plainfilenames	514	258	514	0	258	0	514	0	258	0
<b>Summe</b>	4766506	302141	4288110	-478396	72842	-229299	561730	-4204776	298502	-3639
<b>Quote</b>	100%	100%	89,96%		24,11%		11,78%		98,79%	

Legende:

Größe = Größe der Datei / des Verzeichnisbaumes in KB, gemessen mit `du -sk`;

Anzahl = Anzahl aller Dateien / Verzeichnisse, gemessen mit `ls -laR | wc -l`;

$\delta$  = Differenz zwischen Quelle und Sicherung



Tabelle 9.3.: Auswertung von Rsync und Rdiff-Backup

Verzeichnisse	Original		Rsync				Rdiff-Backup			
	Größe	Anzahl	Größe	$\delta$	Anzahl	$\delta$	Größe	$\delta$	Anzahl	$\delta$
dates	6	5	6	0	5	0	6	0	5	0
filesize	4195364	11	4195364	0	11	0	4195364	0	11	0
filetypes	2	6	2	0	6	0	2	0	6	0
funnydirnames	2036	3556	2036	0	3556	0	2036	0	3556	0
funnyfilenames	1022	512	1022	0	512	0	1022	0	512	0
funnyhard	1022	510	1022	0	510	0	1022	0	510	0
funnysym	14	1013	14	0	1013	0	14	0	1013	0
holes <sup>†</sup>	196	7	0	-196	0	-7	0	-196	0	-7
longfilenames	1098	510	1098	0	510	0	1098	0	510	0
longhardlinks	2	3	2	0	3	0	2	0	3	0
longpathnames	280550	131071	277980	-2570	131064	-7	277968	-2582	131059	-12
longsymlinks	259936	131827	259936	0	131827	0	259936	0	131827	0
overmaxpathlen	22	33	8	-14	21	-12	8	-14	21	-12
permissions	24722	32819	24722	0	32819	0	24722	0	32819	0
plainfilenames	514	258	514	0	258	0	514	0	258	0
<b>Summe<sup>†</sup></b>	4766506	302141	4763726	-2780	302115	-26	4763714	-2792	302110	-31
<b>Quote</b>	100%	100%	99,94%		99,99%		99,99%		99,89%	

Legende:

Größe = Größe der Datei / des Verzeichnisbaumes in KB, gemessen mit `du -sk`;

Anzahl = Anzahl aller Dateien / Verzeichnisse, gemessen mit `ls -laR | wc -l`;

$\delta$  = Differenz zwischen Quelle und Sicherung

<sup>†</sup> In den Standardeinstellungen konnten die Dateien mit Löchern nicht gesichert werden, da das Zielverzeichnis anschwell bis das Dateisystem voll war. Daher habe ich den Test ohne holes wiederholt.

Tabelle 9.4.: Auswertung von star und Bacula

Verzeichnisse	Original		star				Bacula			
	Größe	Anzahl	Größe	$\delta$	Anzahl	$\delta$	Größe	$\delta$	Anzahl	$\delta$
dates	6	5	6	0	5	0	6	0	5	0
filesize	4195364	11	4195364	0	11	0	4195364	0	11	0
filetypes	2	6	2	0	6	0	2	0	6	0
funnydirname	2036	3556	2036	0	3556	0	2036	0	3556	0
funnyfilenames	1022	512	1022	0	512	0	1022	0	512	0
funnyhard	1022	510	1022	0	510	0	1022	0	510	0
funnysym	14	1013	14	0	1013	0	14	0	1013	0
holes	196	7	10*	-186	7	0	X <sup>†</sup>	X	X	X
longfilenames	1098	510	1098	0	510	0	1098	0	510	0
longhardlinks	2	3	2	0	3	0	2	0	3	0
longpathnames	280550	131071	93124	-187426	46639	-84432	280478	-72	131039	-32
longsymlinks	259936	131827	259936	0	131827	0	259936	0	131827	0
overmaxpathlen	22	33	9	-13	4	-29	8	-14	21	-12
permissions	24722	32819	24722	0	32819	0	24722	0	32819	0
plainfilenames	514	258	514	0	258	0	514	0	258	0
<b>Summe</b>	4766506	302141	4578881	-187625	217680	-84461	4766224 <sup>†</sup>	-282	302090 <sup>†</sup>	-51
<b>Quote</b>	100%	100%	96,06%		72,05%		99,99% <sup>†</sup>		99,98% <sup>†</sup>	

Legende:

Größe = Größe der Datei / des Verzeichnisbaumes in KB, gemessen mit `du -sk`;

Anzahl = Anzahl aller Dateien / Verzeichnisse, gemessen mit `ls -laR | wc -l`;

$\delta$  = Differenz zwischen Quelle und Sicherung

\* Es wurden zwar alle Dateien wiederhergestellt, jedoch mit völlig falschen Größen. Daher unbrauchbar.

<sup>†</sup> In den Standardeinstellungen konnte Bacula die Dateien mit Löchern nicht sichern, da der Sicherungspool anschwell bis das Dateisystem voll war. Daher habe ich den Test ohne holes wiederholt.

## 9.3. Auswertung

### 9.3.1. dump

Dump ist der unangefochtene Gewinner des Vergleichs. Es konnte lediglich 23 Dateien (ca. 38KB) nicht sichern, welche allesamt zu lange Dateinamen hatten.

Das entspricht einer Sicherungsquote von über 99,99%. Problematisch war hier `longpathnames` und `overmaxpathlen`, also überlange Pfade. Dump verwendet absolute Pfadnamen von `/` aus, kann also betriebsystembedingt keinen Pfad über `MAXPATHLEN` ansprechen.

Die Datensicherung dauerte 13 Minuten, die Rücksicherung 32 Minuten, wobei hier als Option `-y` angegeben wurde, um trotz der Fehlermeldung zu den überlangen Pfadnamen die Rücksicherung fortzusetzen.

### 9.3.2. Pax

Pax wurde mit dem Standardarchivformat `ustar` verwendet und ist „lediglich“ an zu langen Dateinamen und dem großen Loch gescheitert. Es konnten zwar nur 24% der Dateien, aber 90% der Datenmenge gesichert werden. Pax ist zwar besser als Tar oder Cpio, aber in meinen Augen durch die Beschränkung der Pfadnamen nicht wirklich geeignet.

### 9.3.3. Tar

NetBSD verwendet Pax im Kompatibilitätsmodus anstatt ein „echtes“ Tar oder Cpio.

Mit dem Tar-Format konnten lediglich 42% der Dateien (mit nur 5% der Datenmenge) gesichert werden. Probleme bereiten vor allem die Löcher, lange Pfadnamen, das Datum und die Zugriffsrechte. Alles in allem ist Tar zur Datensicherung komplett ungeeignet.

### 9.3.4. Cpio

Cpio konnte zwar 98,8% der Dateien sichern, scheiterte aber an den löchrigen und großen Dateien, da Cpio diese von vornherein ausschloss<sup>3</sup>. Weitere Probleme hatte Cpio mit der Pfadlänge jenseits von 1025 und einigen Dateinamen (insbesondere ASCII Wert 10, Line Feed). Da insbesondere die großen Dateien nicht mitgesichert wurden, fällt die Quote der Datenmenge mit nur knapp 12% entsprechend verheerend aus.

---

<sup>3</sup>cpio: File is too large for bcpio format source/holes/bighole

### 9.3.5. Afio

Afio hat sich leider auch an den Löchern verschluckt und erzeugte eine über 30GB große Sicherungsdatei, bis das Dateisystem randlos voll war und Afio abgebrochen werden musste.

### 9.3.6. Star

Star ist mit der Headeroption EXUSTAR wesentlich besser als Tar zur Datensicherung geeignet. Es nur mit den überlangen Dateinamen und den Löchern Probleme. Die Dateien mit Löchern konnten zwar gesichert werden (was aber sehr lange dauerte), waren nach der Rücksicherung aber sehr klein.

### 9.3.7. Amanda

Amanda ist imt Dump wesentlich robuster als mit Gtar. Daher sollte, sofern es möglich ist, Dump verwendet werden.

### 9.3.8. Bacula

Bacula hatte im ersten Testlauf massive Probleme mit Löchern in Dateien. Das Spoolverzeichnis wurde gefüllt bis es volllief und Bacula musste abgebrochen werden. Im zweiten Testlauf habe ich auf löchrige Dateien verzichtet. Hiermit hatte Bacula kaum noch Probleme, lediglich ein paar Dateien mit überlangen Namen konnten nicht gesichert werden.

### 9.3.9. Rsync und Rdiff-Backup

Beide Programme hatten massive Probleme mit löchrigen Dateien. Rsync kann zwar mit der Sparse Option mit Löchern umgehen, ist dabei aber sehr langsam. Nichtsdestotrotz sind beide Programme empfehlenswert, vor allem auch da Rsync einen Prüfsummenvergleich zwischen Ziel und Quelle durchführen kann.

## 9.4. Fazit

Das Sicherungswerkzeug für den Einzelrechner und kleinere Netzwerke ist Dump. Es ist robust, zuverlässig, jahrelang verbessert worden und speziell für die Datensicherung ausgelegt.

Hat man keine Root-Rechte und kann Dump nicht einsetzen, bietet sich Star an. Es ist wesentlich mächtiger und stabiler als Tar/Cpio/Pax.

Die beiden Netzwerksysteme Amanda und Bacula hatten im Allgemeinen keine großen Probleme mit dem Test. Amanda ist mit Dump verständlicherweise robuster als mit Gtar, daher sollte auch hier, sofern es die Bandgröße erlaubt, Dump verwendet werden.

Abschließend lässt sich zum Test noch anmerken, daß die Problemfälle die hier betrachtet wurden, zwar existieren, im Allgemeinen in der freien Wildbahn aber eher selten auftreten. Trotzdem sollte man sich nie in Sicherheit wiegen und immer Verifikationsmethoden einsetzen.



**Teil VII.**

**Anhang**





## A. Einen einzelnen Rechner sichern

Die meisten Heimrechner verfügen über einen CD- oder DVD-Brenner, so das man am einfachsten Dumps erstellt, die sich anschließend brennen lassen. Das folgende Shellskript verwende ich seit Jahren auf meinem Produktivrechner.

---

```
#!/bin/sh
## backup.sh - a shellskript to prepare a dump-driven backup
## written by Stefan Schumacher <stefan [at] net-tex . de>. 0xb3fbae33
## Id: backup.sh,v 1.38 2006/02/14 11:32:26 stefan Exp

## determines the dump level by the day of week (Mon=0,Tue-Fri=1,Sat/Sun=2)
## cleans up PostgreSQL
## calculates the size of dump files to be generated (so they can fit on a DVD)
## creates a tarball with system config files in $SRC
## generates a mtree fingerprint of the data
## creates a snapshot
## dumps the snapshot, emails me if an error occurred
## gzip's the dumped files
## creates ISO-images and copies them to a dircetory exported with samba

## get the date to generate filenames
DATE='date +%y%m%d-%a'
DOW='date +%a'

## dir that should be dumped, target to dump to
SRC=/home/
TGT=/usr/home/backups/
PGUSER=pgsql

## Size of the media to generate
## 715000=>698MB, 2042880 => 1995MB, 4398000 => 4294MB
MEDSIZE=4398000 # for a DVD
```

---

Listing A-1: Shellskript für Dump (1/3)

```
## Level 0 at Monday, Level 1 Tu-Fr, 2 Else
if [ $DOW = "Mon" ]
    then LEVEL=0
elif [ $DOW = "Sun" ] || [ $DOW = "Sat" ]
    then LEVEL=2
else
    then LEVEL=1
fi
#####
## PostgreSQL Maintenance and Backup
## vacuum all databases to collect garbage and update analyzer
/usr/pkg/bin/vacuumdb -U$PGUSER -f -z -a
/usr/pkg/bin/pg_dumpall -U$PGUSER | bzip2 -9 > \
    /home/pgsql/pg_`date +%y%m%d-%a`.bz2
#####
## Calc size of source in KB, respect the NODUMP flag (-n)
MEDIA=`du -snk $SRC | awk '{print $1}'`

## compute size of the media, without modulo
MEDIA=$(( $MEDIA / $MEDSIZE ))
## add 2 to the number of media as security premium
MEDIA=$(( $MEDIA + 2 ))

## generate filenames for the media
FILENAME=$TGT$DATE-L$LEVEL-FO
i=0
while [ $i -lt $MEDIA ]
do
    i=`expr $i + 1`
    # generate next filename
    FILES=${TGT}${DATE}-L${LEVEL}-F$i
    # concat filenames to one string
    FILENAME=${FILENAME},${FILES}
done
## remove kommas from filenames for BZIP2
GZFILENAME=`echo $FILENAME | sed 's/,/ /g'`

#####
## tar up config files to be backed up within the dump
/bin/tar cfj $SOURCE/back.tbz2 /etc/ /root/ /usr/pkg/etc/ \
    /var/cron /var/log /var/backups
```

---

Listing A-2: Shellskript für Dump (2/3)

---

```
## Generate filesystem fingerprint with mtree
/usr/sbin/mtree -c -K mode,gname,uname,mode -X mtree-exclude.nodump \
    -p $SRC | gzip > logs/mtree.$DATE.gz

#####
## create a file system snapshot
/usr/sbin/fssconfig -x fss0 /home /

## dump /home, the messages are redirected to a file
/sbin/dump -$LEVEL -u -h0 -b 1 -B $MEDSIZE -L '8677.'$LEVEL'.'$DATE \
    -f $FILENAME /dev/fss0 2> $TGT/logs/dumplog.$DATE

## email the admin about dump problems
[ $? = 1 ] && cat logs/dumplog.$DATE | mail -s 'DUMP Fehlgeschlagen' stefan ;

/usr/sbin/fssconfig -u fss0
## create SHA1 checksum
for i in $FILENAME
do
    sha1 $i >> logs/shallist
done

## compress dumps and generate SHA1 cksum
/usr/bin/gzip -f $GZFILENAME

## create SHA1 checksum
for i in $GZFILENAME
do
    sha1 $i.gz >> logs/shallist
done

## create an ISO image to be burnt
## iterate all created GZ files
for i in $GZFILENAME
do
    /usr/pkg/bin/mkisofs -o $i.iso -rTJ $i.gz /usr/home/backups/logs
done

## notify syslog
/usr/bin/logger 'backup.sh finished'
```

---

Listing A-3: Shellskript für Dump (3/3)



## B. Prüfliste zur Strategie

Ort, Datum: .....

- **Wer** ist verantwortlich?  
Name: .....  
Email, Raum, Telefon: .....  
.....  
Name: .....  
Email, Raum, Telefon: .....  
.....
- **Wie** wird gesichert?  
täglich / wöchentlich / monatlich  
Komplett / Inkrementell / Differentiell
- **Was** wird gesichert?  
.....  
.....  
.....
- **Welche** Medien und Geräte werden verwendet?  
.....  
.....
- **Wo** werden die Medien gelagert?  
im Hause: .....  
extern: .....
- **Wie** lange werden die Medien gelagert?  
Rotationszyklus im Hause: .....  
Rotationszyklus extern: .....
- **Wann** wird gesichert? .....
- **Wie/Wann** wird die Integrität der Sicherung geprüft?  
Prüfsummenverfahren: .....  
vor/nach .....  
Testrücksicherung (am/um/von): .....

- **Welche** Backupstrategie? (Wann welche Level?) taegl. Level:
  - Montag: .....
  - Dienstag: .....
  - Mittwoch: .....
  - Donnerstag: .....
  - Freitag: .....
  - Sonnabend: .....
  - Sonntag: .....
- **Inventur**
  - Welche Zyklen? .....
  - Wann fällig? .....
  - Wer prüft? .....
- **Mediendatenbank**
  - Onlineadresse: .....
  - Sicherungskopie der DB: .....
  - Offline-Version: .....
- **Wo** ist die Dokumentation zur Backupstrategie?
  - online: .....
  - offline: .....
  - .....

# Literaturverzeichnis

[Zwi91]

ZWICKY, Elizabeth D.  
*Torture-testing Backup and Archive Programs: Things You Ought to Know But Probably Would Rather Not*  
Proceedings of the Fifth LISA Conference, USENIX, 1991.  
<http://grox.net/doc/zwicky/testsuite/torture.pdf>

[Zwi03]

ZWICKY, Elizabeth D.  
*Further Torture: More Testing of Backup and Archive Programs*  
Proceedings of the 17th Large Installation Systems Administration Conference  
[http://www.usenix.org/events/lisa03/tech/full\\_papers/zwicky/zwicky.pdf](http://www.usenix.org/events/lisa03/tech/full_papers/zwicky/zwicky.pdf)

[Sch03]

SCHUMACHER, Stefan  
*Einführung in kryptographische Methoden*  
<http://www.cryptomancer.de/21c3/>

[File Flags]

WENNAMCHER, Alexandre  
*File Flags Proposal*  
<http://www.free-x.ch/pub/proposal.txt>

[Sch02]

SCHUMACHER, Stefan  
*Encrypted filesystem with cfs*  
[www.net-tex.de/unix/cfs.html](http://www.net-tex.de/unix/cfs.html)

[secure deletion]

Peter Gutmann  
*Secure Deletion of Data from Magnetic and Solid-State Memory*  
[http://www.cs.auckland.ac.nz/~pgut001/pubs/secure\\_del.html](http://www.cs.auckland.ac.nz/~pgut001/pubs/secure_del.html)

[cgd]

DOWDESWELL, Roland C.  
*The Cryptographic Disk Driver*  
<http://www.imrryr.org/~elric/cgd/cgd.pdf>

- [Pre99]  
PRESTON, W. Curtis  
*UNIX Backup and Recovery*  
<http://www.oreilly.com/catalog/unixbr/index.html>
- [The NetBSD Guide]  
The NetBSD Developers  
*The NetBSD Guide*  
<http://netbsd.org/guide/en/>
- [Fr03]  
FRISCH, Aileen  
*Unix System-Administration*  
<http://www.oreilly.de/catalog/esa3ger/chapter/>
- [Datenrettung FAQ]  
pro-datenrettung.net  
*Datenrettung FAQ*  
<http://www.pro-datenrettung.net/datenrettung-faq.html>
- [Die Top Fünf Open Source-Pakete für Systemadministratoren]  
FRISCH, Aileen  
*Die Top Fünf Open Source-Pakete für Systemadministratoren*  
[http://www.oreilly.de/artikel/amanda\\_1.html](http://www.oreilly.de/artikel/amanda_1.html)
- [AMANDA]  
WEICHINGER, Stefan G. and *AMANDA Core Team*  
*The Official AMANDA Documentation*  
<http://www.amanda.org/docs/AMANDA-HOWTO-Collection.pdf>
- [Amanda auf Platte]  
Wolfgang Barth  
*Netzwerkweite Datensicherung mit Amanda auf Festplatte statt auf Band*  
<http://linux.swobspace.net/misc/linuxtag/2004/index.html>
- [Bacula User's Manual]  
SIBBALD, Kern  
*Bacula User's Manual*  
<http://www.bacula.org/rel-bacula.pdf>
- [Bacula Developer's Manual]  
SIBBALD, Kern  
*Bacula Developer's Manual*  
<http://www.bacula.org/bacula.pdf>
- [Samba-HOWTO]  
Herausgeber: VERNOOIJ, Jelmer R.; TERPSTRA, John H.; CARTER  
Gerald; German Translation Team  
TERPSTRA, John H. und WEICHINGER, Stefan G.  
*Samba-HOWTO-Sammlung, Kap. „Backup-Techniken“*  
<http://gertranssmb3.berlios.de/output/Backup.html>



## [PostgreSQL]

The PostgreSQL Global Development Group

*PostgreSQL 8.1.3 Documentation*<http://www.postgresql.org/docs/>

## [UNIX-Guru]

WILLEMER, Arnold

*Wie werde ich UNIX-Guru?*<http://www.galileocomputing.de/katalog/buecher/titel/gp/titelID-431>

## [Tao]

WILIAMS, Ross

*The Tao of Backup*<http://www.taobackup.com/>

## [CVS]

VESPERMAN, Jennifer

*Essential CVS*<http://ximbiot.com/cvs/cvshome/docs/>

## [OSCVS]

FOGEL, Karl; BAR, Moshe

*Open Source Development with CVS*<http://cvsbook.red-bean.com/>



# Index

/etc/daily, [45](#)  
/etc/security, [45](#)  
/var/backups/etc, [45](#)

Ablage  
  systematische, [20](#)  
Ablaufsteuerung, [43](#)  
ACPI, [58](#)  
Administration  
  dokumentieren, [22](#)  
  vereinfachen, [22](#)  
Administratorenfehler, [18](#)  
Afiio  
  Testauswertung, [114](#)  
afio, [56](#)  
Amanda, [65](#)  
  Clients, [69](#)  
  Einsatz, [70](#)  
  Konfiguration, [66](#)  
  Programme, [69](#)  
  Rücksicherung, [71](#)  
  Server, [66](#)  
  Sicherungsstrategie, [65](#)  
  Sisklist, [67](#)  
  Spoolplatte, [67](#)  
  Statistik, [65](#)  
  Testauswertung, [114](#)  
Anacron, [41](#)  
Anwenderdaten, [21](#)  
Anwendungsdaten, [21](#)  
Archivierung, [17](#)  
Archivierungsdauer  
  maximale, [25](#)  
  minimale, [17](#)  
at, [43](#)  
Ausfallzeiten minimieren, [27](#)

Automatisierung  
  Programme, [41](#)  
  Zeitgesteuert, [41](#)  
Backup, [17](#)  
  differentielles, [29](#)  
  inkrementelles, [29](#)  
Backups  
  inkrementelle, [49](#)  
Bacula, [71](#)  
  Client, [73](#)  
  Konfiguration, [73](#)  
  Rücksicherung, [76](#)  
  Sicherungsart, [73](#)  
  Testauswertung, [114](#)  
  Verschlüsselung, [72](#)  
bacula, [76](#)  
Bacula  
  rc.d, [76](#)  
baculadir, [76](#)  
baculafd, [76](#)  
baculasd, [76](#)  
Basissystem  
  sichern, [45](#)  
batch, [43](#)  
Batchjob, [21](#)  
bconsole, [76](#)  
Bedrohungen, [18](#)  
Bedrohungsszenarien, [17](#), [18](#)  
Benutzer  
  Einweisung, [20](#), [21](#)  
Benutzerfehler, [18](#)  
Betriebssystem  
  sichern, [61](#)  
Binaries, [20](#)  
buffer, [95](#)

- Bzip2
  - Verifikation, 23
- cfs, 98
- cgd, 98
- Cpio
  - Testauswertung, 113
- cpio, 56
- cron, 41
- Cronjob, 21
- crontab, *siehe* cron
- Cryptographic Devicedriver, 98
- Cryptographic Filesystem, 98
- CVS, 99
- cwrsync, 59
  
- Datei
  - zerlegen, 97
  - zusammensetzen, 97
- Dateien
  - finden, 97
  - suchen, 97
- Dateisystem
  - Abbild, 48
  - verschlüsselndes, 32
- Dateisysteme
  - verifizieren, 23
- Dateisystemintegrität, 100
- Dateisystemkorruption, 19
- Daten
  - bewerten, 20
  - nicht wiederherstellbare, 19
  - Verifikation, 23
  - wichtige, 19
- Datenbank
  - nachbereiten, 87
  - vorbereiten, 87
- Datenrettung, 28
- Datenrettungsunternehmen, 28
- Dauer der Rücksicherung, 34
- dd, 60
- diff(1), 56
- Disaster Recovery, 17
- Dokumentation, 18, 22, 26
  
- Dump
  - Dateien ausschließen, 49
  - Fileflag respektieren, 50
- dump, 49
  - Details, 52
  - Header, 54
  - Inkonsistenzen, 55
  - Level, 29
  - Log
    - Inhalt, 52
    - pass, 54
    - Testauswertung, 113
- dump\_lfs, 49
  
- exustar, 55
  
- Fehlertypen, 18
- Feiertage, 25
- Festplatte
  - spiegeln, 61
- Festplatte
  - defekt, 19
- Fileflag, 49
- Filesystem Snapshots, 48
- find(1), 97
- fss, 48
  
- g4u, 61
- GAU, 23
- Geräte
  - schreiben auf, 60
- Ghost for Unix, 61
- GnuPG, 33
- grep, 97
- Gzip
  - Verifikation, 23
  
- Hanoi
  - Türme, 31
- Hardwarekomprimierung, 32
- Homeverzeichnis, 20
  
- Image, 61
- Inventar
  - Datenbank, 18

- Inventarisierungssystem, 36
  - Attribute, 36
- Inventur, 17, 18
- Komplettbackup, 29
- Kompressionsrate, 32
- Komprimierung, 32
- Konfigurationsdateien, 20
- Kopie
  - Eins-zu-Eins, 60
- Löschen
  - sicher, 33
- Laufgeschwindigkeit, 95
- Live-CD, 24
- Logdateien, 23
- Maßnahmen
  - soziale, 21
- mcrypt, 33, 99
- Medien
  - wiederherstellen, 28
- Medium, 33
  - Organisation, 35
  - Dauer der Rücksicherung, 34
  - Geschwindigkeit, 34
  - Integrität, 35
  - Kapazität, 34
  - Kosten, 35
  - Lagerung, 35
  - Spezifikation, 35
  - vergleich, 36
  - Verlässlichkeit, 33
- mklivecd, 24
- mt, 95
- mtree, 56, 100
- Naturkatastrophen, 19
- NODUMP, 49
- Notfallplan, 17
- OpenSSL, 33
- Pax
  - Testauswertung, 113
- pax, 57
- pg\_dump, 88
- pg\_dumpall, 88
- pg\_restore, 88
- Pgpool, 89
  - umschalten, 90
- Point-in-Time-Recovery, 85
- PostgreSQL
  - Backup-Checkpoint, 87
  - Rücksicherung, 88
  - Replikation, 88
  - Sicherung, 85
    - Cluster, 85
  - Table-Spaces, 85
  - WAL
    - zurückspielen, 87
- Prüfsumme, 23, 101
- Puffer, 95
- Quellcode, 20
- Rücksicherung
  - Situationen, 27
  - Test, 26
- Ragnarøk, 23
- RAIT, 66
- RCS, 45
- Rdiff-Backup
  - Testauswertung, 114
- rdiff-backup, 59
  - Versionierung, 60
- Replikation, 28
  - Asynchrone, 89
  - Synchrone, 89
- Replikationsmechanismen, 28
- Replikationsserver, 89
- restore, 49
- restore
  - x, 52
  - bestimmte Dateien, 52
  - Dateien finden, 52
  - Lauf fortsetzen, 52
- Rsync
  - Prüfsumme, 58

- Testauswertung, 114
- rsync, 57
  - MS Windows, 59
- schtasks, 59
- Sicherung
  - Organisation, 21
- Sicherung
  - physikalische, 36
- Sicherungskreise, 25
- Sicherungssoftware
  - Anforderungen, 18
- Sicherungsstrategie, 17
  - Test, 27
- Sicherungssystem
  - Einweisung, 26
- Sicherungssysteme
  - Index, 21
- Sicherungsvarianten, 29
- Snapshot
  - konfigurieren, 48
- Snapshots, 48
- Spiegelung, 17, 27
- split(1), 97
- SSH
  - Tunnel, 57
- Star
  - Testauswertung, 114
- star, 55
  - Header, 55
- Strategie
  - Beispiel, 30
  - Differentiell, 30
  - Inkrementell, 30
  - Komplett, 30
  - Prüfliste, 36
- Streamer
  - steuern, 95
- symmetrische Verschlüsselung, 99
- Synchronisation, 57
  - Archivierung, 59
  - Problem, 59
- System
  - Information, 47
- Systemdaten, 20
- Systeme
  - eingesetzten, 18
  - hochverfügbare, 27
- Systeme
  - homogene, 22
- Systemfehler, 19
- Systempfade, 20
- Systemverzeichnisse, 46
- Türme von Hanoi, 31
- Tar
  - Testauswertung, 113
- tar, 55
- team, 95
- Testauswertung
  - Fazit, 114
- Testen, 18
- Testläufe, 24, 26
- Testsysteme, 24
- Time to Data, *siehe* Dauer der
  - Rücksicherung
- Toleranzen, 21
- Transaktionen
  - Protokolle, 86
- Verschlüsselung, 32
  - symmetrische, 99
- Versionsverwaltung, 99
- Verzeichnisse
  - synchronisieren, 57
- Verzeichnisse
  - synchronisieren, 57
- Weltenbrand, 23
- Windows
  - MS, 59
- Write Ahead Log, 86
- xargs, 97
- Zerlegen/Zusammensetzen, 97