

Proseminar Linux für Umsteiger 2008

# Archive, Backups & Versionsverwaltung

von Patrick Simianer

Betreuung durch Julian Kunkel & Olga Mordvinova

2008-06-24

# 0. Gliederung

---

## 1. Archive

1. tar

## 2. Komprimierung

1.compress

2.zip

3.bzip2

4.Sonstige

5.Vergleich

6.GUIs

# 0. Gliederung

---

## 2. Backup

- 1.rsync
- 2.rsnapshot
- 3.Alternativen

## 3. Versionsverwaltung

- 1.RCS/SCCS
- 2.CVS
- 3.SVN
- 4.git
- 5.Vergleiche und Weitere Systeme

# 1.0 Archive

- (Elektronische) Archivierung
  - Unveränderbare, langzeitige Aufbewahrung elektronischer Information
- Gemeint hier: Zusammenfassung von Dateien und Ordern in einer Datei
- Anwendung:
  - Einfach Verbreitung von Paketen etc.
  - Bei Backups
  - „Ordnung schaffen“
  - Bibliotheken

Folie 1 von 20

Definition aus: [http://de.wikipedia.org/wiki/Elektronische\\_Archivierung](http://de.wikipedia.org/wiki/Elektronische_Archivierung)

Dies ist hier aber nicht in exakt diesem Sinne gemeint, insbesondere nicht „unveränderbar“. Eine solche strikte Definition ist eher in offiziellen Umgebungen von Bedeutung.

In diesem Abschnitt wird lediglich ein Archivierungs-Programm behandelt werden: tar, dem Standard für diesen Zweck in Linux-Umgebungen. Der wesentlich größere Themenkomplex Archivierung im allgemeinen wird hier nicht besprochen.

Es kann sich natürlich auch um mehrere Dateien handeln, wenn das Archiv aufgespalten („gesplittet“) wird. Das funktioniert nur mit manchen „Archivern“.

- Verbreitung bspw. von Software die aus mehreren Dateien besteht über das WWW, Mail etc.
- Bspw. nach Datum benannte Pakete
- Zusammenfassen von mehreren zusammengehörigen Dateien um die Ordner-Struktur zu entschlacken (bspw. bei nicht mehr akut gebrauchten Dateien)
- Objektdateien werden gebündelt.

## 1.1 tar

- Ein Archiver (**t**ape **a**rchiver)
- Funktionsweise klassisch
- Gebräuchlichste Implementierung: GNU tar
- Syntax (für Dummies):
  - tar <c|x> [v] f [j|z] <tar-Datei>  
[<Quelldateien>]
- Heute meist in Verbindung mit Komprimierern

Folie 2 von 20

Mehrere Dateien -> eine Archivdatei. Dateien werden hintereinander geschrieben und Namen, Längen (evtl. Reihenfolge) der Dateien werden gespeichert.

Ursprünglich wurden damit Daten auf Bandlaufwerken gesichert. Sowohl das Kommandozeilenprogramm, als auch die Dateiergung von „Tarballs“ ist .tar.

Daten werden klassisch unkomprimiert aneinandergehängt, dadurch wahlfreier Zugriff unmöglich – sequentieller Zugriff (es können aber trotzdem einzelne Dateien entnommen werden). Dadurch können Archive einfach vergrößert werden.

Auf diese wird sich in folgendem bezogen, da sie bei den meisten GNU/Linux-Distributionen Standard ist. Nachteile von (GNU) tar:

Es werden keine Metadaten wie Zugriffskontrolllisten (aber rwxr--r-- schon, was normalerweise reichen sollte, zudem werden die Datums-Angaben der Dateien beibehalten) gespeichert. Manchmal auch Probleme mit Hardlinks. Unterstützt keine Kompression (Die resultierende tar-Datei kann natürlich verschlüsselt werden, aber nicht das Archiv selbst wie bspw. Bei (Win)Zip.

Es gibt verschiedene andere Implementierungen: ustar, star, bsdtar etc.

c oder x: create oder eXtract, v: verbose (mehr Ausgabe), f: File, j oder z: anschließende Komprimierung mit bzip2 oder gzip (bzw. Archiv liegt in einem solchen Format vor), <tar-Datei>: Datei welche das zur erzeugende bzw. zu entpackende Archiv ist, Quelldateien: Dateien, die ins Archiv aufgenommen werden sollen (mit Leerzeichen getrennt) – nur bei Erzeugung eines Archivs.

tar komprimiert selbst nicht, aber es können einfach Komprimierer eingebunden werden (s. Syntax).

Problem in Verbindung mit Kompression:

Durch solide Kompression (Kompression *nach* der Erstellung des Archivs), sind Bitfehler innerhalb der Datei nicht rückgängig zu machen. Zudem muss auch beim Entpacken einer Datei das komplette Archiv entpackt werden. Bietet bessere Komprimierung dank mehr Redundanz.

## 1.2 Komprimierung

- Dient der Reduktion des Speicherbedarfs
- Unterscheidung zwischen:
  - Verlustfreier Komprimierung
  - Verlustbehafteter Komprimierung
- Prominentestes Beispiel: ZIP
- Unter Linux am weitesten verbreitet: `gzip`, `bzip2`
- Alles fing an mit der Informationstheorie

Folie 3 von 20

Oder auch der Reduktion von Datenaufkommen bei Netzwerkverkehr (vgl. Komprimierung bei `http` etc.).

Wir werden uns nur mit verlustfreier Komprimierung befassen. Video- und Audiokompression sind ein ganz eigenes Thema.

Einfaches Beispiel für verlustfreie Kompression: Huffman-Kodierung (-Bäume), mit denen wohl jeder mal in Berührung gekommen ist oder kommen wird. Hierbei werden prinzipiell in den Daten oft vorkommende Zeichen durch ein einzelnes (kürzeres) Zeichen ersetzt – je öfter das Zeichen vorkommt, desto kleiner das Ersatzzeichen.

Zu Punkt 3 und 4: Zip ist zudem gleichzeitig ein Archiver.

Es gibt zudem viele Verwirrungen um Zip, aber auf die wird hier nicht eingegangen, da der Fokus auf GNU/Linux beschränken wollen.

Habe keine gesicherten Daten – beruht auf eigenen Beobachtungen.

Zu Punkt 5: Geht auf Claude Shannon zurück, Schlagwörter sind Entropie, Informationsgehalt.

## 1.2.1 *compress*

- Nur noch historisch von Bedeutung
- Mit tar verwendbar

Wurde ersetzt von gzip (abwärtskompatibel).

Zu Punkt 2: Endung .Z bzw. .tar.Z

Kommandozeilenbefehl: compress bzw. uncompress.

## 1.2.2 *gzip*

- GNU zip
- Nicht zu verwechseln mit ZIP (*zip*, *unzip*)
- Basierend auf Deflate-Algorithmus, Kombination aus:
  - LZ77 (Lempel-Ziv) und Huffman-Kodierung
- Guter Kompromiss (Schnelligkeit/Dateigröße)
- Häufige Verwendung auch in anderer Software dank der *zlib*
- Kommandozeile: `gzip [-d] <Datei(en)>`

Folie 5 von 20

Völlig frei von patentierten Algorithmen und unter der GPL, damit für praktisch alle Betriebssysteme verfügbar.

Dies ist die freie Implementierung des ZIP-Dateiformates welches u.A. von WinZip verwendet wird: *Info-Zip*. Diese hat jedoch einige Nachteile: u.A. kann nicht mit Archiven umgegangen werden die Dateien enthalten, die größer als 4GB sind.

Deflate-Algorithmus wurde aufgrund von Patent-Problemen mit LZW (Lempel-Ziv-Welch) entwickelt (ist Public Domain). Deflate wird auch von ZIP verwendet, hat aber sonst nichts mit *gzip* gemein (insbesondere findet bei ZIP keine solide Komprimierung statt, sondern die Dateien werden einzeln komprimiert und dann zusammengefasst).

Die *zlib*-Bibliothek beinhaltet den Umgang mit dem *gzip*-Dateiformat, sowie den Deflate Algorithmus. Verwendung in PNG, TIFF, OpenDocument etc.

`-d` steht für decompress. Alternativ kann auch das Alias *gunzip* verwendet werden. Es gibt auch das Kommandozeilenwerkzeug *zcat* mit dem sich (insbesondere Textdateien) ansehen lassen (vgl. *cat*) – kann auch zur Datenrettung benutzt werden. Zu Beachten ist hierbei, dass *gzip* lediglich einzelne Dateien packen kann, d.h. es können nicht mehrere Dateien gleichzeitig zu einer komprimierten Datei verpackt werden (werden mehrere Dateien angegeben, so resultiert das in mehreren komprimierten Dateien).

Dazu muss man vorher *tar* benutzen (mit Schalter `z`, `tar cfvz Bla.tar.gz datei1 datei2 datei3`).

Dateiendung: `.tar.gz` oder kürzer `.tgz`.

## 1.2.3 *bzip2*

- Wieder entstanden aufgrund von Patentproblemen (*bzip*)
- Kombination aus
  - Blocksort (Burrows-Wheeler-Transformation)
  - Wieder Huffman-Kodierung
- Oft effizienter als *gzip*, aber langsamer
- Ebenfalls verwendbar mit `tar`
- Kommandozeile: `b[un]zip2 <Datei(en)>`

Folie 6 von 20

Problem war ein Algorithmus, der arithmetische Komprimierung vornahm. Wird selbst unter einer BSD-ähnlichen Lizenz vertrieben (BSD-Lizenz beinhaltet u.A. dass aus dem Werk wieder ein kommerzielles Produkt entstehen und vertrieben werden darf, ohne dass der Quelltext verfügbar sein muss – Gegensatz zur GPL).

Der erste Schritt führt zu besser komprimierbaren Daten (ist auch wieder rückgängig zu machen).

Kann durch Multi-Threading ausgeglichen werden, was die Ausführung auf Mehrkernprozessoren (heute Quasi-Standard) wesentlich beschleunigt. Wurde leider noch nicht in die Hauptversion übernommen (PBZIP2).

Bei 128 Prozessoren Verbesserung um das 120-fache.

Schalter: `j` (also zum Erzeugen: `tar cfvj Bla.tar.bz2 datei1 datei2 datei3`).  
Endung: `.tar.bz2` oder `.tbz`

Entspricht in der grundlegenden Bedienung weitestgehend *gzip*. Es gibt auch ein Werkzeug um beschädigte Archive zu retten: `bzip2recover` (s. `man bzip2`).

## 1.2.4 Sonstige

- RAR
  - unrar -x datei.rar
- 7-Zip
  - 7z <e|x> datei.7z
- Ace
  - unace x datei.ace

Folie 7 von 20

RAR und 7z und Ace kommen ursprünglich aus der Windows-Welt.

RAR wurde entwickelt von der Firma RARLAB und ist ein proprietäres Format (aktuellste Version: 3).

Es gibt sowohl ein proprietäres Kommandozeilenwerkzeug `unrar`, als auch ein OpenSource Pendant mit exakt dem gleichen Namen (Gna! Project, unrarlib, RAR V3, GPL). Sehr kompliziertes Zusammenspiel von Lizenzen etc.

Fazit: Es können Dateien entpackt werden (auch Passwort-geschützte (zumindest beim proprietären Entpacker)), aber vom ständigen Umgang mit dem Format ist abzuraten (Proprietär+hinterherhinkende Weiterentwicklung des freien Pendants). Zudem kann man keine Archive erzeugen.

7-Zip ist eigentlich ein grafisches Packprogramm für Windows, das nahezu alle Komprimierungsmethoden unterstützt, es beinhaltet aber auch ein Dateiformat: `.7z` (moderne Implementierung des LZ77 Algorithmus: LZMA mit Verschlüsselung etc.). Das Programm ist OpenSource, daher recht gute Linux-Unterstützung. Das Paket nennt sich `p7zip` (enthält `7z` und `7za`, wobei `7z` `tar`, `gzip`, `bzip2` unterstützt – `7za` tendenziell viel mehr durch Plugins).

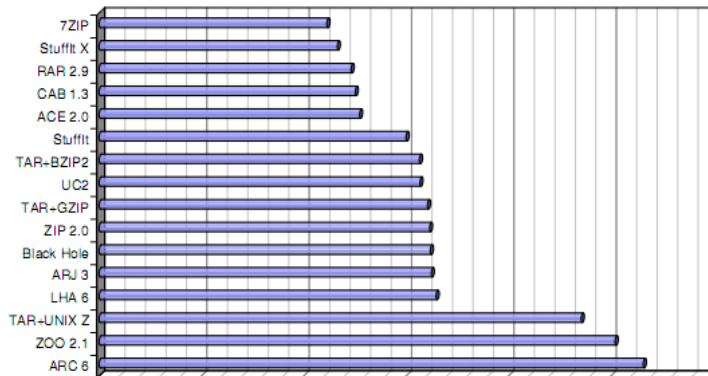
Kommandozeile: bei `e` wird alles in das aktuelle Verzeichnis entpackt, bei `x` wird die Ordnerstruktur beibehalten.

WinAce. Es gibt die Möglichkeit Archive (Version 1 des ACE-Formats bzw. in einer nicht-freien Version auch alle ACE-Formate) zu entpacken mittels `unace`. Von einer häufigen Verwendung dieses Formats ist auch abzuraten.

## 1.2.5 Vergleich

Compressor	Size	Ratio	Compression	Decompression
gzip	89 MB	54 %	0m 13s	0m 05s
bzip2	81 MB	49 %	1m 30s	0m 20s
7-zip	61 MB	37 %	1m 48s	0m 11s

Overall relative compression ratio  
(the shortest wins!)



Folie 8 von 20

Die Zeiten wurden mit den jeweiligen Standard-Einstellungen der Programm gemessen.

Es wird deutlich, dass die proprietären Formate - insbesondere beim Komprimierungsgrad - gegenüber den OpenSource-Formaten einen Vorsprung besitzen (wobei 7-Zip jedoch der allgemeine Gewinner ist und zugleich OpenSource).

In Sachen Schnelligkeit (gzip vs. bzip2 vs. 7z) liegt gzip weit vorne, bringt jedoch auch die schlechtesten Kompressions-Ergebnisse (knapp hinter bzip2). 7z ist schneller als bzip2, aber langsamer als gzip – es bringt jedoch auch das bei weitem beste Ergebnis.

Somit ist 7z eigentlich das überlegene Format – aber noch nicht weit verbreitet. Sollen Daten längerfristig archiviert werden bietet sich bzip2 an, gzip ist aufgrund seiner Schnelligkeit beim (Ent)Packen eine gute Wahl, wenn auf das Archiv oft zugegriffen wird.

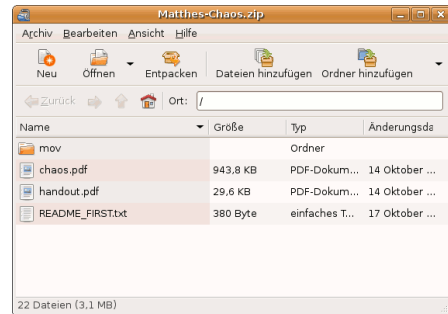
Alles in allem ist bei Archivierung und Komprimierung zu sagen, dass es hier (natürlich rein subjektiv) sehr schön ist, wenn die Quellen des Programms/Algorithmus offen liegen und man nicht auf Gedeih und Verderb einer einzelnen Firma ausgeliefert ist, da dies insbesondere in diesem Bereich Risiken birgt (wer garantiert mir, dass ich in 10 Jahre noch die Archive lesen kann?).

Bildquellen:

- [http://blogs.reucon.com/srt/2008/02/18/compression\\_gzip\\_vs\\_bzip2\\_vs\\_7\\_zip.html](http://blogs.reucon.com/srt/2008/02/18/compression_gzip_vs_bzip2_vs_7_zip.html)
- <http://flashlight.slad.cz/files/compression.pdf>

## 1.3 Archive - GUIs

- Auch Archivmanager genannt
- GNOME
  - File-Roller (`file-roller`)
- KDE
  - Ark (`ark`, `ark-kde4`)
- Xfce
  - Xarchiver (`xarchiver`)
- Alternative: Kommandozeile



Folie 9 von 20

Bildquelle:

Archivmanager bieten die Möglichkeit bestehende Archive zu öffnen und Dateien darauf zu extrahieren, neue zu erzeugen oder in bestehende Archive Dateien anzuhängen.

Alle drei bieten eine Dateiansicht, über der sich eine Toolbar befindet um Operationen vorzunehmen. Alle Ähneln sich sehr stark und allesamt sehen sie aus wie WinZip.

Es gibt noch wie bei Linux üblich Alternativen, diese sind aber bei den drei bekanntesten Window-Managern die vorinstallierten.

Bildquelle: [http://wiki.ubuntuusers.de/\\_/50031336f306481ecd32efef0fcd59c0.png](http://wiki.ubuntuusers.de/_/50031336f306481ecd32efef0fcd59c0.png)

## 2.0 Backup

- Backup (Dateisicherung), Sync (Synchronisierung), Mirroring (Spiegelung)
- Entgegengesetzt zu Backup: Restore
- Sicherungskonzepte:
  - Sicherung aller Daten
  - Differenzielle Datensicherung
  - Inkrementelle Datensicherung

Folie 10 von 20

Da Festplatten sowie optische Speichermedien wie CDs und DVDs irgendwann auf jeden Fall das „Zeitliche segnen“ ist es unabdingbar, Sicherungskopien seiner wichtigen Daten anzulegen. Dies erfolgt zumeist auf anderen Datenträgern (externe Festplatten, DVDs). Sicherungen auf ein und derselben Festplatte kann man nahezu als nutzlos bezeichnen, außer man will sich lediglich gegen versehentliche Veränderungen schützen.

Verwandt sind Sync und Mirroring.

Bei Sync sollen zwei Ordner synchron gehalten werden (bspw. Ordner auf Notebook und auf stationärem PC) und beim Mirroring sollen Daten schlicht mehrfach identisch vorhanden sein (bspw. Zur Lastverteilung bei Download-Archiven).

Sicherung aller Daten: Alle Daten werden unabhängig von vorherigen Sicherungen gesichert. Nachteil: lange Dauer der Sicherung.

Differenzielle Datensicherung: Alle Daten seit der letzten Vollsicherung werden gesichert. Vorteil: es müssen beim Restore lediglich max. 2 Sicherungen zurück gespielt werden. Nachteil: diese Sicherungen können groß werden.

Inkrementelle Sicherung (Zuwachssicherung): Nur die Daten, die seit der letzten Sicherung geändert wurden, werden gesichert. Vorteil: schnelle Sicherung (nach einer Vollsicherung). Nachteil: es müssen beim Restore evtl. viele Sicherungen zurück gespielt werden.

Prinzipielle Empfehlungen:

Mehrere Sicherungen auf verschiedenen Datenträgern anfertigen (Externe Platten die möglichst auch an verschiedenen Orten stehen)

Oft sichern.

## 2.1 *rsync*

- Netzwerkprotokoll/Programm zur Synchronisation von Daten (auch über Netz)
- Möglichst geringe Datenübertragung
- Anwendung:
  - `rsync [OPTIONEN] <Quelle> <Ziel>`
  - Option: `-a` (`rsync -a <Quelle> <Ziel>`)
  - `rsync -avze ssh /home/me user@foo.de:/backups`
  - `rsync --delete -avzbe ssh user@foo.de:/etc /home/me/etcbak --backup-dir=~ /old`

Folie 11 von 20

Entwickelt im Rahmen des Samba-Projekts (ein Projekt welches – ursprünglich - Windows-Freigaben für Linux bereitstellen will/wollte).

Es sollen zwischen einem Client und einem Server Dateien übertragen werden und zwar mit möglichst geringem übertragenem Datenvolumen. Wir wollen hier auf das Kommandozeilenprogramm eingehen.

Zu beachten ist hier, dass *rsync* „nur“ unidirektionale Synchronisation bietet, was bedeutet, dass nur neue Dateien aus dem Quellverzeichnis übertragen werden, umgekehrt jedoch nicht. Legt man also im Zielverzeichnis eine neue Datei, so wird diese bei der Synchronisation nicht beachtet werden. *Unison* bietet dies und ist somit besser in Anwendungsbereichen wie z.B. der Synchronisation von Daten zwischen einem Laptop und einem stationären PC geeignet (*Unison* verwendet das *rsync*-Protokoll).

Zur Bestimmung, was im Zielverzeichnis geändert werden muss, werden die Größe der Datei, die Erstellungszeit als auch das Ergebnis eines speziellen Algorithmus betrachtet, welcher sicherstellt, dass nur die wirklich veränderten Daten (innerhalb der Dateien) übertragen werden. So wird das oben erwähnte möglichst geringe Datenübertragungsvolumen erzielt.

Prinzipieller Aufbau des Aufrufs von *rsync* sieht folgendermaßen aus:

```
rsync [Optionen=avP... oder auch Optionen wie -exclude] <Quelle> <Ziel>
```

Quelle und Ziel können hierbei auch Remote-Adressen sein.

-a fasst verschiedene nützliche Optionen zusammen: u.A. sorgt sie dafür dass Unterverzeichnisse mitkopiert werden, Metadaten (wie Rechte, Zeiten und Gruppenrechte) beibehalten werden, wie auch symbolische Links.

Noch zwei fortgeschrittenere Beispiele:

```
rsync -avze ssh /home/me user@foo.de:/backups
```

Hier wird von einem lokalen Verzeichns (/home/me) auf ein entferntes Verzeichnis /backups kopiert. Man greift hier mittels SSH auf den entfernten Rechner zu. Hierzu muss man die Option -e, sowie anschließend den Parameter ssh angeben.

Die Option sorgt für „verbose“ Output (es werden alle Operationen ausgegeben), -z schaltet die Komprimierung ein (empfiehlt sich über langsame Netze).

Das geht natürlich auch umgekehrt:

```
rsync --delete -avzbe ssh user@foo.de:/etc /home/me/etcbak --backup-dir=~ /old
```

In diesem Fall ist das Quellverzeichnis auf einem anderen Rechner und das Zielverzeichnis lokal. Hierzu muss auch wieder die Option -e übergeben werden und der Parameter ssh. Nach dem `user@foo.de` folgt wieder ein Doppelpunkt mit nachfolgendem Verzeichnis. Die Optionen `--delete`, `-b` und `-backup-dir` sorgen dafür, dass Dateien die im Quellverzeichnis gelöscht wurden im Zielverzeichnis auch gelöscht werden, diese aber im Ordner `~/old` gesichert werden.

Zu erwähnen wäre noch, dass die Übertragung auf entfernte Rechner nicht nur mittels SSH möglich ist, sondern auch mittels des *rsync*-Protokolls, wofür aber der *rsync*-Daemon laufen muss. SSH wird die einfachere und üblichere Methode sein.

Die Aufrufe von *rsync* lassen sich hervorragend mit *Cron* automatisieren.

## 2.2 *rsnapshot*

- Backup-System das auf rsync aufsetzt
- Einfach installierbar und konfigurierbar
- Optionen in der Konfigurationsdatei:
  - `snapshot_root /mnt/externehd/backups`
  - `interval <daily|weekly|monthly> <X>`
  - `backup /home`
  - `0 2 * * * root rsnapshot daily`
- Restore: Kopieren aus jeweiligem Backup-Verzeichnis

Folie 12 von 20

Rsnapshot ist ein Programm um Snapshots („Schnappschüsse“) von lokalen und remote Dateisystemen anzufertigen, für gewöhnlich zu Backup-Zwecken. Es verwendet im Hintergrund rsync und ist komplett in Perl geschrieben. Es können sowohl Vollbackups, als auch inkrementelle Backups vorgenommen werden. Bei der inkrementellen Datensicherung kommen Hardlinks zum Einsatz um bei jedem Schnappschuss jeweils ein vollständiges Backup „vorzutauschen“ (nicht veränderte Dateien werden hart auf bereits bestehende Versionen der Datei verlinkt, damit sie nicht erneut kopiert werden müssen). Es steht unter der GPL.

Rsnapshot kann bspw. mittels `apt-get` installiert werden (Debian, Ubuntu...) und ist mittels einer Konfigurationsdatei (standardmäßig unter `/etc/rsnapshot.conf`) leicht zu konfigurieren. Es wird ein `snapshot_root` angegeben, welches ein lokales Verzeichnis (oder ein gemountetes im Netzwerk), aber auch ein Verzeichnis auf einem entfernten Rechner sein kann, worauf mittels SSH zugegriffen werden kann – hier ist ein wenig mehr Konfigurationsaufwand vorhanden, was hier aber den Rahmen sprengen würde (v.A. die Automatisierung). Zudem kann man in der Konfigurationsdatei verschiedene Arten von Backups anlegen (`daily`, `weekly`, `monthly` – Namen können frei gewählt werden), wobei man noch angeben muss, wie viele aufgehoben werden sollen. Auch stellt man hier die zu sichernden Orte ein.

Die eigentliche Ausführung kann manuell erfolgen (mit `rsnapshot daily` zum Beispiel), oder aber auch mittels *Cron* angestoßen werden – hier bekommen die willkürlich gewählten Namen nun auch eine Bedeutung (`weekly` wird beispielsweise jeden Sonntag um 9 ausgeführt).

Um Dateien aus dem Backup wiederherzustellen genügt es dank der Hardlinks einfach die Dateien aus dem entsprechendem Backup-Verzeichnis zurück zu kopieren. Das neueste Backup befindet sich jeweils immer unter dem `snapshot_root` Verzeichnis unter dem Backup-Namen mit der Endung `.0` (Beispiel: `daily.0`).

## 2.3 Alternativen

- rsync mit eigenen Skripten
- Unison
- Komplette eigene Skripte (mit dd, cp, diff etc.)
- Bacula
- BackupPC
- Sbackup

Folie 13 von 20

Es gibt unter Linux sehr viele Arten, sein komplettes System oder seine persönlichen Daten zu sichern. Ob man nun vorgefertigte Pakete benutzt oder Kommandozeilenwerkzeuge oder gar GUIs ist wohl nur eine Frage des Geschmacks. Hier sollen noch ein paar Alternativen erwähnt werden:

- Man kann natürlich seine eigene Backup-Lösung mittels rsync entwickeln – rsnapshot ging auch aus einem solchen Vorhaben hervor.
- Unison baut (wie schon erwähnt) auf rsync auf, und kann auch zu Backup-Zwecken benutzt werden. Es bietet eine nette GUI, was für viele ansprechend sein dürfte.
- Im Prinzip reicht für ein Vollbackup eines Linux-System ein einzelner Aufruf von dd, welches ein komplettes Abbild von Festplatten erstellen kann. Man kann aber auch mit Shell-Skripten in Kombination mit Standard-Tools seine komplett eigene Lösung finden.
- Bacula ist für größere Netzwerke (also mehrere Rechner und ein Backup-Server) geeignet und bietet einige nette Features wie ein Webinterface, ist aber für Einzelbenutzer ein wenig überdimensioniert, da es einen Server benötigt und mehrere Daemons involviert sind.
- In die gleiche Sparte wie Bacula fällt BackupPC, welches ebenfalls ein Webinterface besitzt. Zudem bietet es die Möglichkeit durch „data deduplication“, dass wenn auf beliebigen PCs die gleichen Dateien liegen, diese nur ein einziges mal gesichert werden, was Speicherplatz spart (je mehr, desto mehr Rechner auf dem Server gesichert werden).
- Sbackup ist ein bei Ubuntu (mit GNOME) standardmäßig installiertes grafisch-orientiertes Backup System, dessen Augenmerk auf einfacher Bedienbarkeit liegt (daher der Name **S**imple Backup).

## 3.0 Versionsverwaltung

- System zur Versionierung und gemeinsamen Zugriff auf Daten
- Sicherstellung von Konsistenz der Daten und Nachvollziehbarkeit
- Haupt-Anwendungsgebiete:
  - Softwareentwicklung
  - Dokumentation

Folie 14 von 20

Versionsverwaltungssysteme bieten, wie der Begriff schon sagt, Versionsverwaltung und üblicherweise auch Funktionen zum gemeinsamen Zugriff auf Daten.

Unter Versionen ist sind hier zeitlich getrennte und für gewöhnlich auch inhaltlich verschiedene *Revisionen* von Dateien zu verstehen.

Typischerweise werden hier mindestens die laufenden Änderungen, sowie ein Zeitstempel und ein Benutzername (pro *Revision*) gespeichert.

Gemeinsamer Zugriff erfolgt meist nach dem Client/Server (*Repository* auf der Serverseite wo die Verwaltung stattfindet, *Arbeitskopie* (*working copy*; die aus dem *Repository*, entsteht indem es „ausgecheckt“ wird - *Checkout*) beim Client bei dem Änderungen vorgenommen werden die dann „eingchecked“, „committed“ werden) Konzept (auf dem Client kann natürlich auch der Server laufen, wenn das System beispielsweise lediglich von einer Person an einem Rechner genutzt wird).

Sinn und Zwecke solcher Systeme ist, dass jeder Benutzer jeweils mit den aktuellsten Datenbestand arbeiten kann, aber auch Änderungen nachvollzogen und gegebenenfalls auch wieder rückgängig gemacht werden können.

Typischer Funktionsumfang:

- Protokollierung von Änderungen (meist durch Delta-Kodierung: Um Speicherplatz zu sparen, werden nur die Änderungen zwischen 2 Versionen werden gespeichert.)
- Wiederherstellung von *Revisionen*
- Archivierung von „Meilensteinen“ (Releases oder Ähnliches)
- Gleichzeitige Entwicklung von Entwicklungszweigen (*Branches*, stable vs. Devel)
- Umgang mit Konflikten (*Lock*= Änderungen an Datei verhindern, *Merge* = Unterschiedliche Versionen zusammenführen – nicht nur bei Konflikt, sondern auch auch beim *Commit*)

Anwendung finden solche Systeme meist in der Softwareentwicklung, da man es hier mit einer Vielzahl von Änderungen zu tun hat und diese Systeme zudem den Entwicklungsablauf nachvollziehbar halten, was eine große Erleichterung darstellt.

Aber auch bei Dokumentations-Projekten (bspw. auch in der Softwareentwicklung oder auch bei technischer Dokumentation).

Prinzipiell lassen sich diese Systeme natürlich für jegliche Projekte einsetzen, jedoch ist ihre „natürliche Umgebung“ die Softwareentwicklung.

Wichtige Begriffe:

**Repository, Arbeitskopie (Checkout), Commitment (Checkin), Branches, Lock, Merge**

## 3.1 RCS/SCCS

- **Revision Control System, Source Code Control System**
- Bieten den typischen Funktionsumfang von VCS
- CVS ging aus RCS hervor
- Syntax
  - Checkin: `ci -l <Datei>`
  - Checkout: `co -l <Datei>`
  - Diff: `rcsdiff <Datei>`

Folie 15 von 20

RCS wurde in den 1980er Jahren von Walter F. Tichy an der Purdue Universität entwickelt. Ist heutzutage ein Teil des GNU Projekts. SCCS wird meist bei kommerziellen UNIX Distributionen mitgeliefert. Beide sind nur noch von historischer Bedeutung.

VCS = Version Control System

RCS/SCCS wurde mittlerweile auch durch CVS verdrängt, jedoch ist es eine schöne, einfache Möglichkeit einzelne Dateien unter Versionskontrolle zu stellen.

RCS war anfangs nicht netzwerkfähig, daher wurde CVS als netzwerkfähiger Aufbau verwendet. CVS verwendet heute noch das RCS- Dateiformat für die History (Speicherung der Unterschiede). Ein großer Nachteil von SCSS und RCS ist, dass sie lediglich einzelne Dateien verwalten können, daher sind sie für größere Projekte ungeeigneter, bzw. heutzutage werden lieber Systeme benutzt, die dazu fähig sind.

Syntax:

- Checkt eine Datei ein und sichert diese Version somit, damit sie mittels `co` wiederhergestellt werden kann. Der Parameter bedeutet, dass die Version sofort wieder ausgecheckt und für den Aufrufer gelockt wird.
- Holt die neueste Version der Datei (außer man gibt explizit eine Revisionsnummer an mit `-l<Nummer>`).
- Zeigt Unterschiede zwischen zwei Versionen direkt auf der Konsole an. Man kann auch die beiden zu vergleichenden Revisionen zu benennen (mit `-r`).

## 3.2 CVS

- **C**oncurrent **V**ersions **S**ystem, Prototypisches VCS
- Heimisch auf UNIX, aber auch für Windows verfügbar (Client und Server)
- Hat (begründete) Nachteile
- Syntax
  - `cvs -d :pserver:user@cvs.srv.de:/pfad/zu/rep login`
  - `cvs import -m "init import" ProjektName bla start`
  - `cvs checkout ProjName , cvs update , cvs add/remove`
  - `cvs commit -m "Hallo!" hello.c`

Folie 16 von 20

Concurrent = gleichlaufen, gleichzeitig

CVS wurde in den 80er Jahren von Dick Grune entwickelt und steht unter der GPL. Ist daher nach wie vor sehr beliebt bei OpenSource-Projekten (wenn auch nach und nach modernere Systeme beliebter werden). Wird bspw. bei Sourceforge.net eingesetzt. Bietet GUIs für nahezu alle Plattformen, sowie Integration in IDEs wie Eclipse.

CVS bietet alles, was ein Versionskontrollsystem der vorherigen Beschreibung nach haben sollte:

- Client/Server Architektur (mit anonymen Checkout)
- Datei/Projekt-Revisionen
- Branches
- Locks (eher unüblich bei CVS)

Nachteile:

- Es können keine Dateien oder Ordner verschoben oder umbenannt werden (Grund: Refactoring)
- Keine symbolischen Links (Grund: Sicherheit)
- Kein Unicode/Non-ASCII Support für Dateinamen (Grund: verschiedene OSs)
- Keine atomic commits (Commit auf einmal, statt in atomaren Teilen, Grund: ?)
- Binärdateien stellen ein Problem dar, nur bestimmte Dateiendungen unterstützt

Einige wichtige Befehle für CVS (es fehlen einige, insbesondere `diff` um Unterschiede zwischen der Arbeitskopie und der Version im Repository anzuzeigen, aber das würde zu weit führen).

- `cvs -d :pserver:user@cvs.srv.de:/pfad/zu/rep login`: Man muss sich zunächst am Server einloggen um mit dem Repository arbeiten zu können. `pserver` ist die Login-Methode (es gibt noch `kserver` (Kerberos) und `gserver`, wird aber selten verwendet). `User` ist der Benutzername (bei anonymem Zugang oft `anonymous`), das nach dem `@` ist der Server und nach dem anschließend `:` kommt der Pfad zum Repository auf dem Server. `login` ist der Befehl, der abgesetzt wird – anschließend wird man nach dem (evtl. leeren) Passwort gefragt und ist wenn das erfolgreich war eingeloggt und kann mit dem Repository arbeiten. Gibt noch die Möglichkeit das abzukürzen mit der `CVSROOT` Umgebungsvariablen.
- `cvs import -m „Log-Message“ ProjektName HerstellerMarke VersionsMarke`: Ein neues Projekt wird angelegt, die `*Marken` sind praktisch egal, sie dienen CVS nur zur Verwaltung. `ProjektName` der Name, mit dem das Repository später ausgecheckt werden kann. Hier wird davon ausgegangen, dass man sich in dem Verzeichnis befindet (auf unterster Ebene) das man unter Versionskontrolle stellen will.
- `cvs checkout ProjName`: Hiermit holt man sich den aktuellen Stand ins momentan (hoffentlich) leere Verzeichnis. Mit `update` holt man sich nach einem initialen Checkout immer die neuesten Änderungen. Mit `add/remove` `Dateiname` fügt man Dateien zur Versionsverwaltung hinzu oder entfernt sie.
- `cvs commit -m „Message“ [Datei]`: Mit dem Commit-Befehl weist man CVS, an seine lokalen Änderungen ins Repository zu übernehmen. Das `-m` (Message) ist wichtig, da man hier eine Log-Message zum Commit abgibt, in der möglichst stehen sollte, was man denn gemacht hat. Sind Dateinamen angegeben werden nur diese „committed“, sonst die ganze Arbeitskopie.

## 3.3 Subversion (SVN)



- Modernes VCS (Konverter cvs2svn frei erhältlich)
- Unterschiede zu CVS:
  - Rangfolge: zeitlich, dann örtlich
  - Unterschiede werden lokal berechnet
  - Commits atomar, Binärdatei-Unterstützung
- Syntax:
  - `svn import, commit, co, update, add, rm, mv, help`
  - Server: `svn+ssh://foo.de/svn/ProjektName`

Folie 17 von 20

Subversion wird seit 2000 von CollabNet entwickelt und steht unter einer Lizenz im Apache Stil (freie Software). Beweggründe waren Mängel in dem in die Jahre gekommenen Quasi-Standard für VCS: CVS.

Es stehen auch für andere VCS Konvertierungsprogramme zur Verfügung (welche die hier nicht behandelt werden). Auch gibt es ausgereifte GUIs und Integration in IDEs (Subclipse etc.).

Unterschiede zu CVS:

- Die Rangfolge ist bei CVS genau umgekehrt (bei beiden auf höchster Ebene ist das Projekt) und es wird immer auf Projektebene „committed“. Das hat zur Folge, dass bei SVN jede Revision eine eindeutige Nummer hat. Bei CVS ist dies so einfach nicht möglich, da jede Datei in einer anderen Revision sein kann (weil sie nicht verändert wurde) – hier muss man sich bei jeder Datei an das aktuellste Datum halten (also SVN: Rev 1312, bei CVS 2008-06-24 12:00 ...). Zudem ist auch das Umbenennen/Verschieben von Dateien bei SVN einfach.
  - Bei CVS muss auf dem Server festgestellt werden, was sich nun verändert hat. Bei SVN hingegen ist die letzte Version der Datei im `.svn` Verzeichnis vorhanden, daher können die Änderungen lokal berechnet werden und der Netzwerktraffic und die Last auf dem Server bleiben annähernd minimal.
  - Commits sind atomar bei SVN, was zur Folge hat, dass Änderungen komplett oder gar nicht übertragen werden, was Inkonsistenzen im Repository durch Verbindungsabbrüche und Abstürze verhindern soll.
- Binärdateien werden bei SVN erkannt und es werden ebenfalls nur die veränderten Teile übertragen, bei CVS würde immer die komplette Datei übertragen werden müssen.

Mehr:

Kopien unterstützt, Dateien und Ordner als gelöscht markierbar, einige weitere.

Syntax:

Einloggen ist bei SVN nicht mehr so umständlich nötig, man wird bei Aktionen gegebenenfalls nach dem Passwort gefragt.

Die Befehle sind denen von CVS sehr ähnlich, aber (zumindest meiner Meinung nach) etwas intuitiver.

Der „Server-String“ ist folgendermaßen aufgebaut:

`svn+ssh` – Protokoll: Hier SVN über SSH, anschließend eine Standard-URL, die man schöner weise (wenn der Server das mitmacht) auch mit einem Browser aufrufen kann (mittels WebDAV und wenn man statt `svn+ssh` `http(s)://` angibt)

Logo von der Subversion-Homepage.

## 3.4 GIT



- Entwickelt von Linus Torvalds
- Besonderheiten:
  - Ausgerichtet auf nicht-lineare Entwicklung
  - Effizient, auch für große Projekte
- Syntax:
  - `git init/add ./commit`
  - `git clone <URL>`
  - `git add/rm/mv <Datei>`
  - `git commit -a`

Folie 18 von 20

Git (benannt nach Linus Torvalds selbst. „I'm an egotistical bastard, and I name all my projects after myself. First Linux, now git.“) wurde ursprünglich von Linus Torvalds für die Entwicklung des Linux-Kernels entwickelt. Es sollte eigentlich kein vollständiges VCS werden, sondern nur ein „Backend“ für das dann verschiedene Frontends geschrieben werden. Jedoch hat git sich zu einem vollwertigen VCS (und mehr) entwickelt und wird heute neben der Entwicklung des Linux-Kernels auch bei anderen umfangreichen Projekten eingesetzt (u.A. X.org, OLPC). Git steht unter der GPL und ist auf vielen Plattformen verfügbar.

Git wurde mit besonderem Augenmerk auf starke nicht-lineare Entwicklung entwickelt. Es bietet gute Möglichkeiten einfach und schnell Branches auszugliedern und diese auch wieder zu mergen. Eine genauere Betrachtung würde hier zu weit führen.

Zudem skaliert git sehr gut, was es besonders für (sehr) große Projekte (wie bspw. den Linux-Kernel) attraktiv macht. Es ist nahezu komplett in C geschrieben.

Es soll hier noch erwähnt werden, dass Repositories über viele Protokolle veröffentlicht werden können (u.A. HTTP, SSH, FTP und ein git eigenes Protokoll), auch kann mit git direkt mit SVN-Repositories gearbeitet werden mittels git-svn.

Die Syntax von git ist für Subversion-Benutzer zumindest bei den grundlegenden Funktionen sehr ähnlich:

- Um ein neues Projekt zu erzeugen muss man das Repository initialisieren, Dateien hinzufügen (hier gehe ich davon aus, dass wir im betreffenden Verzeichnis sind) und dann committen.
- Statt einem Checkout wird hier ein „Clone“ vorgenommen, der Effekt ist aber der gleiche – man hat den aktuellen Stand des Projekts auf der Platte.
- Datei Funktionen sind identisch.
- Für normale Commits ist der Parameter -a erforderlich.

Logo von der GIT-Homepage.

## 3.5 Vergleiche und Weitere Systeme

- Eine Vielzahl von VCS Systemen
- CVS, Subversion und git sind „Weltmarktführer“
- Für Einsteiger: Subversion

Folie 19 von 20

Neben den hier vorgestellten Systemen gibt es noch eine große Anzahl weiterer, von denen viele frei verfügbar sind (Mercurial, Monotone, Darcs, Bazaar) andere auch nicht (BitKeeper (Lizenzänderungen bei BitKeeper waren der Grund für die Entwicklung von git), Perforce).

Die drei hier vorgestellten stellen jedoch die momentan am verbreitetsten Systeme dar, da sie einerseits frei verfügbar sind und zudem leistungsfähig und (mehr oder minder) einfach in der Bedienung.

Subversion stellt jedoch wohl den besten Kompromiss aus Features, Leistung und einfacher Bedienung dar. Daher soll dies hier meine Empfehlung für Einsteiger ins „Versionskontrollgeschäft“ sein.

RCS ist für einzelne Dateien gut geeignet, bei denen man beispielsweise eine Änderung evtl. schnell wieder rückgängig machen will.

Ein schöner Vergleich von 26 verschiedenen Systemen findet sich hier: <http://better-scm.berlios.de/comparison/comparison.html>

Ende

---

Text durch Klicken hinzufügen

## Interessante Links

---

- Komprimierung:
  - <http://flashlight.slad.cz/files/compression.pdf>
- Subversion:
  - <http://svnbook.red-bean.com/>
  - <http://subclipse.tigris.org/>
  - <http://coding-time.blogspot.com/2008/04/subversion-visually-explained-in-30sec.html>
- GIT:
  - <http://random-state.net/log/3410155710.html>
  - <http://smalltalk.gnu.org/blog/bonzinip/using-git-without-feeling-stupid-part-1>

- Komprimierung
  - Vergleich von Komprimierungs-Programmen
- Subversion
  - DIE Referenz für SVN
  - Plugin für Eclipse IDE
  - Nette Visualisierung des VCS-Konzepts
- GIT
  - Zwei schöne Artikel über GIT

# Quellen 1

---

- <http://de.wikipedia.org/wiki/Archiver>
- [http://de.wikipedia.org/wiki/Solide\\_Kompression](http://de.wikipedia.org/wiki/Solide_Kompression)
- <http://schmidt.devlib.org/file-formats/tar-archive-file-format.html>
- <http://de.wikipedia.org/wiki/Tar>
- <http://www.gnu.org/software/tar/manual/tar.html>
- <http://de.wikipedia.org/wiki/Datenkompression>
- <http://de.wikipedia.org/wiki/Compress>
- [http://de.wikipedia.org/wiki/ZIP\\_\(Dateiformat\)](http://de.wikipedia.org/wiki/ZIP_(Dateiformat))
- <http://de.wikipedia.org/wiki/Gzip>
- <http://de.wikipedia.org/wiki/Bzip2>
- <http://compression.ca/pbzip2/>
- <http://linux.die.net/man/1/bzip2>
- <http://de.wikipedia.org/wiki/Archiver>
- [http://de.wikipedia.org/wiki/Solide\\_Kompression](http://de.wikipedia.org/wiki/Solide_Kompression)
- <http://schmidt.devlib.org/file-formats/tar-archive-file-format.html>
- <http://de.wikipedia.org/wiki/Tar>

# Quellen 2

---

- <http://www.gnu.org/software/tar/manual/tar.html>
- <http://de.wikipedia.org/wiki/Datenkompression>
- <http://de.wikipedia.org/wiki/Compress>
- [http://de.wikipedia.org/wiki/ZIP\\_\(Dateiformat\)](http://de.wikipedia.org/wiki/ZIP_(Dateiformat))
- <http://de.wikipedia.org/wiki/Gzip>
- <http://de.wikipedia.org/wiki/Bzip2>
- <http://compression.ca/pbzip2/>
- <http://linux.die.net/man/1/bzip2>
- <http://www.linux.com/articles/59888>
- <http://7-zip.org/7z.html>
- <http://packages.ubuntu.com/de/hardy/utils/unace-nonfree>
- <http://flashlight.slad.cz/files/compression.pdf>
- [http://blogs.reucon.com/srt/2008/02/18/compression\\_gzip\\_vs\\_bzip2\\_vs\\_7\\_zip.html](http://blogs.reucon.com/srt/2008/02/18/compression_gzip_vs_bzip2_vs_7_zip.html)
- <http://wiki.ubuntuusers.de/Packprogramme>
- <http://de.wikipedia.org/wiki/Backup>
- <http://de.wikipedia.org/wiki/Rsync>

# Quellen 3

---

- <http://rsync.samba.org/>
- <http://en.wikipedia.org/wiki/Rsync>
- <http://linuxwiki.de/rsync>
- <http://wiki.ubuntuusers.de/rsync>
- <http://manual.sidux.com/de/sys-admin-rsync-de.htm>
- <http://de.wikipedia.org/wiki/Rsnapshot>
- <http://www.rsnapshot.org/>
- <http://wiki.ubuntuusers.de/rsnapshot>
- <http://www.tim-bormann.de/index.php?section=134>
- [http://www.linux-magazin.de/heft\\_abo/ausgaben/2007/02/sicherheitspack?special=Storage&category=13791](http://www.linux-magazin.de/heft_abo/ausgaben/2007/02/sicherheitspack?special=Storage&category=13791)
- <http://wiki.ubuntuusers.de/sbackup>
- <http://www.cis.upenn.edu/~bcpcierce/unison/index.html>
- <http://packages.debian.org/de/lenny/backupper>
- [http://de.wikipedia.org/wiki/Dd\\_\(Unix\)](http://de.wikipedia.org/wiki/Dd_(Unix))
- <http://www.bacula.org/de/>
- <http://en.wikipedia.org/wiki/Bacula>

# Quellen 4

---

- <http://de.wikipedia.org/wiki/Versionsverwaltung>
- <http://agave.garden.org/~aaronh/rcs/tichy1985rcs/rcs.html>
- [http://de.wikipedia.org/wiki/Revision\\_Control\\_System](http://de.wikipedia.org/wiki/Revision_Control_System)
- [http://de.wikipedia.org/wiki/Source\\_Code\\_Control\\_System](http://de.wikipedia.org/wiki/Source_Code_Control_System)
- <http://agave.garden.org/~aaronh/rcs/tichy1985rcs.html>
- [http://en.wikipedia.org/wiki/Concurrent\\_Versions\\_System](http://en.wikipedia.org/wiki/Concurrent_Versions_System)
- [http://de.wikipedia.org/wiki/Concurrent\\_Versions\\_System](http://de.wikipedia.org/wiki/Concurrent_Versions_System)
- [http://informatik.asn-graz.ac.at/selflinux/html/cvs\\_buch\\_kapitel\\_201.html](http://informatik.asn-graz.ac.at/selflinux/html/cvs_buch_kapitel_201.html)
- [http://kb.gnuher.de/zzz\\_old\\_articles/various/HOWTO%20-%20CVS%20Kurzanleitung.txt](http://kb.gnuher.de/zzz_old_articles/various/HOWTO%20-%20CVS%20Kurzanleitung.txt)
- [http://de.wikipedia.org/wiki/Subversion\\_\(Software\)](http://de.wikipedia.org/wiki/Subversion_(Software))
- <http://terror-im-wohngebiet.de/node/1>
- [http://www.linux-fuer-alle.de/doc\\_show.php?docid=230&catid=3](http://www.linux-fuer-alle.de/doc_show.php?docid=230&catid=3)
- <http://de.wikipedia.org/wiki/Git>
- [http://en.wikipedia.org/wiki/Git\\_\(software\)](http://en.wikipedia.org/wiki/Git_(software))
- <http://git.or.cz/course/svn.html>
- <http://better-scm.berlios.de/comparison/comparison.html>