

-----  
Thema: Backup-Spielereien mit "tar" unter Linux  
bash-script "mirror"  
-----

Hallo liebe Linuxer,

schon eine ganze Weile wollte ich das nächste BIC schreiben, aber erst jetzt werde ich damit fertig! Es ist doch eine etwas größere Sache, wie ich gedacht habe, und die Zeit reicht hinten und vorne nicht! Naja, mal sehen wie es bei Euch ankommt! Dieses BIC soll über den Mail-Lister an Euch alle in der BRELUG gehen, das ist doch etwas kostenschonender wie das Drucken auf Papier. Wenn ich da nur an die Teilnehmerzahl der letzten zwei Linuxtreffen denke ....! Es war wirklich schon eine kleine Sensation, ich hoffe es geht auch weiter so!

Heute will ich mich einmal mit einem Thema befassen, was mich schon seitdem ich mit Unixen/Linuxen spiele, beschäftigt, nämlich: Wie kann ich am einfachsten und geschicktesten Backups jedweder Art erstellen, und von diesen Backups auch am einfachsten wieder Daten oder ein ganzes System zurückbekommen? Wie schaffe ich das mit möglichst wenig Aufwand, mit wenigen (Kommandozeilen-)Befehlen und womöglich nur mit einer Bootdiskette? - Ich weiß, auch zu diesem Thema ist schon jede Menge geschrieben worden, aber vielleicht ist es für den Einen oder Anderen ja doch ganz interessant, dazu meine Spielereien und Erfahrungen zu hören. Außerdem erkennt man, wie einfach Backups unter Linux zu realisieren sind, Backups von M\$-WIN95/98/NT zum Beispiel, die theoretisch nur mit einer Linux-Bootdiskette und dem Backupmedium wieder aufzuspielen sind, und die dann ein bootfähiges WIN-XX wieder auferstehen lassen!

Unter Linux gibt es etliche Tools, mit denen ein Backup erstellt werden kann. Die Tools, mit denen ich gespielt habe, sind tar, afio, dd und das Paket tob. Afio ist inzwischen bei den meisten Distributionen mit dabei, er ist dem cpio in der Befehlssyntax sehr ähnlich. Auch cpio ist für Backups geeignet, mit diesem Tool habe ich mich aber nicht so eingehend beschäftigt. Von den diversen Backup-Komplett-Paketen habe ich nur ein wenig mit tob und taper gespielt, den taper habe ich dann aber wieder ganz schnell weggeworfen. Eigentlich sind mir aber doch die einfachen Kommandozeilen-Tools am liebsten! Über meine Erfahrungen mit dem tar möchte ich in diesem BIC berichten. Wenn Interesse besteht, kann ich auch gerne ein zweites (oder gar drittes) BIC über meine Erfahrungen mit den anderen Tools schreiben.

Vorab noch eine Bemerkung: Alle meine Spielereien und Anwendungen mit dem tar finden auf der Kommandozeile statt. Ich finde, ein wenig Erfahrung mit der Kommandozeile kann keinem von uns schaden, sind doch die allermeisten kniffligen Probleme nur auf der Kommandozeile lösbar! Also keine Angst und ran an die Kommandozeile!

In meinem System benutze ich schon seit einigen Jahren mit vollster Zufriedenheit einen (inzwischen schon veralteten) SCSI-Streamer mit den schönen großen, stabilen 250Mb Bandcassetten. Auch in der heutigen Zeit der multi Gb-Streamer oder CD-Rom Brenner leistet mir dieses Ding zusammen mit den oben erwähnten Linux Tools gute Dienste. Backups kann man aber auf jedweder anderen Art von Backup-Medium erstellen: Streamer, CD-ROMs, Festplatten, Disketten, Dateien usw usw.

Also fangen wir am Besten mit ein paar Beispielen an! Vorher noch dieses: Alle Pfadnamen und Devicenamen in den folgenden Beispielen sind aus meinem System entnommen, sie können bei Euch durchaus anders aussehen und lauten! Alle Beispiele, Spielereien und Experimente werden als Benutzer "root" durchgeführt, damit bei Backups und Restores alle Berechtigungen stimmen!

Jetzt nehmen wir einmal folgendes Szenario an:

Als bewußter Linux-Benutzer und -Experte haben wir, wie es der "filesystem-standard" vorschreibt, schön brav unsere sämtlichen (wenigstens die meisten) persönlichen Erweiterungen und Veränderungen am Linux-System im Verzeichnisbaum /usr/local vorgenommen. Im Verzeichnis /usr/local/bin sind ein Haufen binaries (Programme) zusammengekommen, notwendige Bibliotheken sind unter /usr/local/lib abgelegt, Manual-Seiten dazu unter /usr/local/man gespeichert, und vielleicht haben wir aus dem Internet noch einige Pakete gezogen, die wir uns später einmal genau ansehen möchten, sie sind derweil in einem Verzeichnis /usr/local/incoming oder auch /usr/local/src geparkt. So, nun wäre es an der Zeit, diesen ganzen Krempel einmal zu sichern, denn wenn jetzt die Platte einen Spagat macht (oder wir unser Linux aus irgendeinem Grunde abschießen), dann ist wie bei einem M\$-user auch hier die ganze Arbeit und Anstrengung zum Teufel! Ein S.u.S.E oder Red Hat Linux ist ja schnell wieder von der CD installiert, aber eben der Krempel unter /usr/local sollte halt irgendwie gesichert werden.

Also auf gehts, ein Band ins Streamer-Laufwerk eingelegt und rein ins kalte Wasser bzw. auf die Kommandozeile:

```
tar -cvf /dev/tape /usr/local
```

wird auf der Kommandozeile unter einem xterm oder einer ASCII-Konsole eingegeben. Das Format des tar-Befehles ist wie folgt:

```
tar -Aktion Device Zu_sichernde_Dateien/Verzeichnisse
```

Eine komplette Liste aller Aktionen, Optionen und Parameter gibt die Manual-Page von tar oder die Eingabe von tar --help. Der oben eingegebene Befehl sichert das ganze Verzeichnis /usr/local mit sämtlichen Unterverzeichnissen auf meinen Streamer, der unter dem "Device" /dev/tape (/dev/nst0 verlinkt mit /dev/tape) ansprechbar ist. - Die einzugebenden Kommandos werde ich immer auf einer eigenen Zeile mit einem Tab eingerückt darstellen. - Nach diesem Kommando sollte der ganze Inhalt des /usr/local Verzeichnisses auf dem Streamerband stehen. Wie kann man das kontrollieren? Nichts einfacher als das! Erst einmal das Band zurückspulen (mt -rewind erledigt das bei mir), und dann

```
tar -t /dev/tape
tar -tv /dev/tape (ausführlichere Anzeige)
```

zeigt den ganzen Inhalt des Bandes auf dem Bildschirm an. Dabei wird einem auffallen, daß der tar beim Speichern die absoluten Pfadnamen "relativiert" hat, er hat nämlich die "leading slashes" entfernt, so ist z.B. aus /usr/local/bin ein usr/local/bin auf dem Band geworden. Das hat einen großen Vorteil, beim Restore kann man die gesicherten Daten an jeden beliebigen Ort in seinem Dateisystem spielen. Der tar legt beim Zurückspielen exact die Verzeichnisstruktur an, die im Backup abgespeichert ist. Zum Beispiel kann man das gerade beschriebene Backup also auch nach /home/jumbo/usr/local/bin spielen, man muß sich dazu nur im /home/jumbo Verzeichnis befinden.

Will man aber den Inhalt des Bandes genau dorthin restaurieren, wo er hergekommen ist, so muß man in unserem Fall (als user root) "zu Fuß" ins Hauptverzeichnis / wechseln mit

```
cd /
```

und mit

```
tar -xvf /dev/tape
```

wird dann der Inhalt vom Band wieder zurückgespielt.

Wenn man nun keinen Streamer hat, sondern auf Diskette speichern will und muß? Kein Problem,

```
tar -cvf /dev/fd0h1440 /usr/local/bin
oder
tar -cvf /dev/fd0 /usr/local/bin
```

macht ein Backup vom Inhalt von /usr/local/bin auf dem Diskettenlaufwerk 0, also dem "ersten" Laufwerk analog zu A:\ unter DOS/WIN), bei der ersten Kommandozeile explizit auf eine 1440'er Diskette. Bei Disketten wird der Platz für Backups natürlich sehr schnell eng, deshalb kann man mit dem tar das Backup auch auf mehrere Disketten verteilen:

```
tar -cvMf /dev/fd0 /usr/local/bin
```

verteilt die Daten von /usr/local/bin auf mehrere Disketten im Diskettenlaufwerk 0. Beim Angucken oder Zurückspielen muß man dann natürlich auch die Option "groß M" einschalten:

```
tar -tvMf /dev/fd0
```

zeigt den Inhalt des Archives verteilt auf mehrere Disketten im Laufwerk 0 an, und

```
tar -xvMf /dev/fd0
```

spielt die Daten verteilt auf mehrere Disketten aus dem Diskettenlaufwerk 0 wieder an den Platz, wo man sich momentan im System befindet (hoffentlich da wo die Daten hingehören!).

Man kann ein Backup mit dem tar auch in eine Datei spielen:

```
tar -cvf trallalla /usr/local
```

erstellt ein Backup von allen Daten unter /usr/local in die Datei mit dem Dateinamen trallalla. Ansehen oder restaurieren geht dann genauso:

```
tar -tvf trallalla
tar -xvf trallalla
```

zeigt die Daten an oder spielt sie wieder auf. Wählt man beim Erstellen des Backups als Dateinamen dabei einen Namen der auf ".tar" endet, z.B. trallalla.tar, dann kann man sich den Inhalt dieser Datei (dieses tarfiles wie man auch sagt) mit diversen File-Managern, so auch mit dem midnight-commander, durch "Anklicken" oder "Anticken" ganz einfach ansehen!

Eine weitere Option des tar erlaubt auch das Erstellen von komprimierten Backups:

```
tar -cvzf /dev/tape /usr/local
```

erstellt ein komprimiertes Backup von /usr/local auf dem Streamerband. Ansehen oder Restaurieren erfolgt dann analog mit

```
tar -tvzf /dev/tape
tar -xvzf /dev/tape
```

Einzigster Nachteil beim Arbeiten mit komprimierten Backups unter tar, das Verteilen auf mehrere Disketten funktioniert dann nicht mehr!

Manchmal möchte man aus einem Backup nur einzelne Dateien wieder "herausziehen", weil vielleicht genau diese Dateien aus irgendwelchen Gründen zerstört oder verloren gegangen sind. Auch das ist kein Problem. Nehmen wir einmal an, die Datei /usr/local/bin/superprogram sei kaputt. Wie schon weiter oben beschrieben haben wir ein Backup von /usr/local auf ein Streamerband gemacht. Also Band in den Streamer einlegen, als user root nach / wechseln, und

```
tar -xvf /dev/tape usr/local/bin/superprog
```

eingeben, und der tar wird uns die gewünschte Datei aus dem Backup "herausfischen" und nach /usr/local/bin abspielen. Zur Erinnerung: Da der tar beim Erstellen des Backups die Pfadnamen relativiert hat, müssen wir beim Eingeben des gewünschten Dateinamens auf der Kommandozeile den "leading slash" weglassen, deshalb  
usr/local/bin/superprog! - Auch ganze Dateigruppen lassen sich so wieder herausfischen:

```
tar -xvf /dev/tape usr/local/bin
```

restauriert das ganze /usr/local/bin Verzeichnis aus dem Backup, welches ja das gesamte /usr/local Verzeichnis beinhaltet. Ebenso kann man mit

```
tar -xvf /dev/tape usr/local/bin/g*
```

alle Dateien restaurieren, die unter usr/local/bin liegen und mit dem Buchstaben "g" anfangen.

Sehr oft tauchen aus dem Internet, von CD's oder aus anderen Quellen sogenannte ".tgz" Dateien auf, meistens irgendwelche Pakete, Programme, Utilities oder sonstiges Dinge, die man in sein System installieren möchte. Diese Dinge sind tar-Backups in einer Datei, wie schon oben erwähnt, die dazu noch mit dem gzip Programm komprimiert worden sind (wie es auch der tar -cvzf Befehl erzeugt). So liegen z.B. bei der Slakware-Distribution alle Quell-Dateien in diesem Format vor. Mit unseren eben erlernten Fähigkeiten, den tar zu bedienen, ist es kein Problem mehr, mit diesen Dateien zu spielen:

```
tar -tvzf dateiname.tgz
```

zeigt den Inhalt dieser Datei an, und

```
tar -xvzf dateiname.tgz
```

installiert den Inhalt unter dem momentanen Verzeichnis. Ich sollte vielleicht noch erwähnen, das die Option v (verbose) beim tar ausführliche Informationen ausgibt, läßt man sie weg, ist der tar bei seiner Ausgabe von Informationen sehr viel schweigsamer.

Soweit sogut! Eigentlich habe ich bis jetzt nur Backup-Spielereien veranstaltet, wo ist denn nun der Vorteil von tar und Konsorten zu sehen? Ja, für mich sind das schon viele Vorteile! Am besten ich schildere einfach weiter, was ich noch alles mit dem tar mache.

Mit der Zeit fällt in einem heftig benutztem Linux-System nicht nur Krepel unter /usr/local an, sondern das System wird ja auch immer mehr an seine persönlichen Belange und Geschmäcker angepaßt. So kriegt der Rechner einen "hostname", es werden user und groups angelegt, evtl. über Netzwerk verbundene Rechner sind dem eigenen Rechner bekannt; Drucker, Ethernetkarten, Modems und sonstige Hardware sind installiert und angepaßt, usw usw. Die meisten dieser lokalen Einstellungen sind im Verzeichnis /etc in diversen Dateien festgehalten, manche sind vielleicht mit Yast oder anderen Sysadmin-Tools erstellt worden, andere vielleicht auch per Hand durch Editieren der notwendigen Dateien. So mache ich nun regelmäßig mit tar ein Backup von meinem /etc Verzeichnis, sogar noch auf Disketten! Desweiteren versuche ich, eine Liste aller wichtigen Dateien, die ich zur Anpassung meines Systems editiert und verändert habe, zu führen und immer up-to-date zu halten. Diese Datei liegt bei mir unter dem Heimatverzeichnis von root, sie heißt local.list. Ihr Inhalt sieht beispielsweise so aus:

```
/etc/X11/  
/etc/mail/genericstable  
/etc/mail/mailertable  
/etc/TextConfig  
/etc/XF86Config  
/etc/aliases
```

```
/etc/exports
/etc/fstab
/etc/group
/etc/hostname
/etc/hosts
/etc/hosts.equiv
/etc/hosts.lpd
/etc/host.conf
/etc/inetd.conf
/etc/inittab
/etc/ld.so.conf
/etc/lilo.conf
/etc/minicom.users
/etc/services
/sbin/conf.d
/sbin/init.d
/root
.
.
.
```

usw usw. Die Liste ist noch lange nicht vollständig! Alle die Dateien, die sich hinter dieser Liste von Dateinamen verbergen, sind von mir einmal verändert worden und wichtig für die persönlichen Einstellungen meines Linux-Systems, sie liegen alle wild verteilt in meinem Linux-System herum. Nun will ich alle diese Dateien auf einen Schlag mit einem Backup sichern. Auch das geht ganz einfach mit tar:

```
tar -cvMf /dev/fd0 `cat local.list`
```

Bei diesem Kommando zeigt sich eines der vielen genialen Konzepte eines Unix/Linux: Dort wo im Kommando für den tar die zu sichernden Datei-en/Verzeichnisse stehen müßte, steht jetzt in "umgekehrten" Hochkommas ein weiterer Kommandozeilen-Befehl, nämlich

```
cat local.list
```

Dieses Kommando in Hochkommas wird zuerst ausgeführt und das Ergebnis dieses Kommandos, die Ausgabe des Inhalts der Datei local.list, also eine Liste von Datei-/Pfadnamen, wird dem tar als Argument-Liste übergeben. Ergebnis: Der tar legt ein Backup über mehrere Disketten verteilt von genau den Dateien an, deren Pfadnamen vorher in der Datei local.list festgehalten worden sind. Wenn das nicht genial ist!!

So kann man sich beliebig viele Listen anlegen, und durch oben gezeigten Trick den tar dazu bringen, gemäß dieser Listen ein Backup von gewünschten Datei-Mengen zu erzeugen.

-----

Jetzt kommen wir zum Backup eines M\$-WIN Systems. Dazu muß ich noch schnell die Verzeichnisstruktur erläutern, die ich in mein System eingebaut habe, damit ich meine Struktur als Beispiel benutzen kann.

Im Hauptverzeichnis (vom Linux) habe ich ein Verzeichnis /dos angelegt. Darunter habe ich die Verzeichnisse /dos/c: erzeugt, danach /dos/d:, auch /dos/e:, und so weiter bis ich alle Laufwerke, die unter DOS/WIN existieren, auch hier als (leere) Verzeichnisse abgebildet habe.

Mein WIN95 ist auf Laufwerk c:\ installiert, unter meinem Linux als /dev/sdal bekannt. Ich verwende in meinem Linux im moment den Kernel 2.0.36, der auch das Dateisystem "vfat" unterstützt. Nun montiere ich die WIN95 Partition an das dafür vorgesehene (leere) Verzeichnis:

```
mount -t vfat /dev/sdal /dos/c:
```

und schwupp hängt mein c:\ Laufwerk am Verzeichnis /dos/c: dran! Man kann mit cd oder mit dem midnight-commander einmal hineinwechseln und sieht, es ist alles da, sogar die langen Dateinamen werden richtig

dargestellt. Ich muß noch dazusagen: Ich habe KEINE FAT32 Partition unter Windows im Einsatz, ich weiß auch nicht, wie weit die neuen Kernels jetzt die FAT32 unterstützen, oder auch das NTFS von WIN-NT. Mit FAT16 geht aber die beschriebene Prozedur problemlos, auch das spätere Restaurieren, deshalb lege ich IMMER Wert darauf, unter WIN NUR FAT16 Partitionen zu benutzen und die geringen Nachteile dafür in Kauf zu nehmen, ganz egal wie viel Propaganda für die FAT32 Unterstützung gemacht wird!

So, jetzt ist das Backup ganz einfach. Mit

```
tar -cvf /dev/tape /dos/c:
```

spiele ich den ganzen Kram vom Verzeichnis /dos/c:, also der Inhalt der montierten c:\ Partition, auf mein Band. So lege ich z.B. immer ein Backup von c:\ direkt nach einer ausführlichen und sorgfältigen Neuinstallation von WIN95 auf diesem Wege an, habe also immer ein "jungfräuliches" System auf Band in Reserve.

Geht mir nun mein WIN95 (was ja kaum vorkommen soll :-)) hinüber, so ist eine Neuinstallation nicht wie üblich eine langwierige und Frage-Antwort-intensive Prozedur mit Dutzenden von "Der Rechner muß jetzt neu gestartet werden" Vorgängen, sondern ein schnelles und einfaches Vergnügen. Folgendermaßen läuft es ab:

- Flachschielen der alten, versauten c:\ Partition, entweder durch Booten von DOS/WIN95 von Diskette und Formatieren von c:\, oder durch Eingabe von

```
mkdosfs -F 16 /dev/sda1 Größe_der_Partition
```

von Linux aus, eventuell auch noch durch vorheriges "Nullen" mit dem dd, wie das geht beschreibe ich in einer späteren BIC
- Starten von Linux
- Montieren der neu erstellten (leeren) c:\ Partition durch

```
mount -t vfat /dev/sda1 /dos/c:
```

  - Wechsel ins Hauptverzeichnis mit

```
cd /
```
- Beginn des Aufspielens mit

```
tar -xvf /dev/tape
```
- Nun ist bei mir ca. eine halbe Stunde Kaffeetrinken oder Fernsehgucken angesagt, der tar mach alles alleine, ohne jedwede dämliche Rückfrage!
- Wenn der tar fertig ist, abmontieren der c:\ Partition mit

```
umount /dos/c:
```

 und Rückspulen/Entnahme des Bandes
- Fertig!

Device und Verzeichnisse sind bei Euch durchaus anders! Außerdem muß ich noch darauf hinweisen, daß der Kernel des laufenden Linux natürlich eine Unterstützung des vfat-filesystems aufweisen muß.

Die so wiederaufgespielte c:\ Partition ist in den meisten Fällen bootfähig. Falls es beim Booten von WIN95 trotzdem nicht klappen sollte, so hilft ein Booten der WIN95 Bootdiskette (die man aus der Systemsteuerung von WIN95 heraus erstellen kann) und dann die Eingabe des Befehls

```
sys c:
```

um sie wieder bootfähig zu machen.

Die gleiche Prozedur kann man auch durchspielen, wenn man nur eine Boot/Root-Diskette von Linux hat, bzw. eine Boot/Rescue Diskette oder ein Boot/Root Diskettenpäarchen oder Ähnliches. Der Einfachheit halber nehmen wir an, daß die c:\ Partition schon flachgeschlagen und unter DOS/WIN95 neu formatiert worden ist. Auch hier muß der Kernel auf der Boot-Diskette das vfat Filesystem unterstützen. Ist das Linux gebootet, so hat man ja ein laufendes Mini-Linux im RAM. Ist ein Backup wie oben beschrieben auf Band oder sonstwohin gespeichert worden, so muß man erst einmal eine entsprechende Verzeichnisstruktur in seinem RAM-Linux anlegen. Das geht ganz schnell:

```

mkdir /mnt/dos (das Verzeichnis /mnt sollte im RAM-Linux schon
                vorhanden sein)
mkdir /mnt/dos/c:
mount -t vfat /dev/sdal /mnt/dos/c:
cd /mnt (damit beim Wiederaufspielen die Daten richtig nach
        dos/c: fließen)
tar -xvf /dev/nst0

```

Diese Latte von Befehlen sollte das Wiederaufspielen genauso wie schon weiter oben beschrieben bewerkstelligen. Es geht, ich habe es schon mehrfach ausprobiert! Das Device /dev/nst0 ist bei mir das SCSI-Streamerband, im normalen System ist es verlinkt mit dem Device /dev/tape, nur vom Diskettensystem aus muß ich halt das richtige Device ansprechen! Ist der tar fertig, sollte man noch folgende Befehle eingeben:

```

cd /
sync
umount /mnt/dos/c:

```

um unter dem RAM-Linux auch sicher alle Daten auf die Platte zu schreiben und dann die Platte sauber abzumontieren. Das ist alles! Wo gibt es so etwas unter M\$-WINDoof, ein WIN-Backup mit einer Boot-Diskette und ein paar Kommandozeilen-Befehlen wieder aufzuspielen, und dann ein bootfähiges WIN95/98/NT zu erhalten??!!

Noch ein denkbare Szenario für ein Backup eines M\$-WIN Systems. Es sei leider kein Streamer oder ein ähnliches Backup-Medium vorhanden, es sei aber eine hinreichend große Festplatte da mit viel freiem Platz unter Linux. Warum nicht ein Backup von M\$-WIN mit tar in eine Datei machen?? Nehmen wir an, unter /usr/local wäre entsprechend viel Platz vorhanden. Dann würde ich es folgendermaßen machen:

- Anlegen eines separaten Verzeichnisses für das Backup, z.B.
 

```
mkdir /usr/local/winbackup
```
- Montieren der WIN-Partition
 

```
mount -t vfat /dev/sdal /dos/c:
```
- Erstellen des Backups mit
 

```
tar -cvzf /usr/local/winbackup/winimage.tgz /dos/c:
```
- Abmontieren
 

```
umount /dos/c:
```

Das Ergebnis ist eine (durch die Eingabe der "z" Option) komprimierte Imagedatei der WIN-Partition, sie wird sicher eine ziemlich mächtige Größe haben!

Das Zurückspielen aus dieser Datei gestaltet sich nun ein wenig anders wie von einem Band! Zur Erinnerung: Das Zurückspielen beim Band sollte vom Hauptverzeichnis / aus beginnen, da im Backup ja die Datenstruktur immer mit dem Pfad dos/c: beginnt. Nun liegt das Backup jetzt aber tief in unserem Verzeichnisbaum unter /usr/local/winbackup, was tun, sprach Zeus?! Die Lösung ist genauso einfach:

- Wir schlage wie oben beschrieben die c:\ Partition flach, bereiten sie vor un montieren sie nach /dos/c:, wechseln ins Hauptverzeichnis mit
 

```
cd /
```

 und geben das tar-Kommando mit dem kompletten Pfadnamen der Backupdatei
 

```
tar -xvzf /usr/local/winbackup/winimage.tgz
```
- Anschließend montieren wir die c:\ Partition mit
 

```
umount /dos/c:
```

 wieder ab.
- Fertig!

Noch Fragen, Kienzle?? Ja, Hauser ...! Man kann noch unendlich viele andere Wege finden, um ein Backup seiner WIN-Partition zu erstellen, das ist ja das schöne an Linux!

Nun noch eine weitere Anwendung für den tar. In den letzten Wochen habe ich im Internet gegrast und eine Reihe von interessanten Paketen (Dateien) für den Flugsimulator gezogen. Ich weiß, bei den meisten beginnt jetzt das große Gähnen, wenn sie nur den Namen Flugsimulator hören, es soll ja hier auch nur als Beispiel dienen!! Also, ich habe ein paar Dateien, alles .zip Archive, und ich spüre am Telefon wie Willi ganz große Stielaugen kriegt, als er hört was ich da alles entdeckt und gezogen habe! Wie kriege ich den ganzen Krempel jetzt am elegantesten auf Willi's Rechner rüber? Auf Disketten? Einige dieser Dateien sind nur ein paar 50k groß, andere bringen 3Mb auf die Matte! Sicher: Auch unter M\$-WIN gibt es einige Tools (zu kaufen!), die den Krempel auf Disketten verteilen können, aber mit dem tar geht es auch und mit einem Handgriff. Außerdem ist der tar schon von Haus aus beim Betriebssystem mit dabei!

Also spiele ich alle Dateien für Willi an einen Platz, in irgendein Verzeichnis. Sie sind schon auf der Linux-Ebene vorhanden, da ich alle Internetzugriffe sowieso nur noch unter Linux veranstalte! Alle Dateien haben einen Dateinamen, der auf ".zip" endet. Also ist der entsprechende tar-Befehl ganz einfach:

```
tar -cvMf /dev/fd0 *.zip
```

und schon fließen alle Dateien verteilt auf Disketten im Diskettenlaufwerk 0! Natürlich muß man sich gerade in dem Verzeichnis befinden, in dem auch die .zip Dateien liegen. Wenn dort keine anderen Dateien vorhanden sind wie die beschriebenen .zip Dateien, dann könnte man den tar auch folgendermaßen anweisen:

```
tar -cvMf /dev/fd0 *
```

und bekäme das gleiche Ergebnis!

Nun lade ich mich beim Willi zum Kaffeetrinken ein und nehme meinen Stapel von 5 oder 6 oder 11 Disketten mit. Willi erstellt dann ein Verzeichnis, wo er den Flugsimulator-Krempel hinhaben will, und als user root wechselt er dann in dieses Verzeichnis. Die erste Diskette des Stapels wird eingelegt, und mit

```
tar -xvMf /dev/fd0
```

fließt der ganze Flugsimulator Kram zu ihm auf die Platte! Auf diese Art und Weise habe ich schon mehrmals solche Dinge zu Willi (und auch anderen) transportiert!.

-----

Zum Schluß noch einen Bonbon. - In meinem System versuche ich alles, was so an eigenen Daten anfällt, also Texte, Briefe, Rechenblätter unter SC oder Star-Calc (Star-Office), Grafiken, Bilder, Abrechnungen, Programmierertexte usw. usw. an einem möglichst zentralen Platz abzuspeichern. Dafür gibt es bei mir ein Unterverzeichnis namens "daten". In diesem Unterverzeichnis gibt es dann wieder Unterverzeichnisse, die heißen dann daten/briefe, daten/texte, daten/grafik, daten/calc usw. usw. Mit der Zeit fallen immer mehr Daten an, so sollte das "daten" Unterverzeichnis dringend ausgemistet werden, trotzdem möchte ich die Daten gerne aufheben, aber nicht alle hier an dieser Stelle. Zu diesem Zweck habe ich ein kleines Shell-Skript geschrieben, das ich am Ende in dieses BIC einfügen werde, wer daran Interesse hat, kann es sich ja ausschneiden (mit einem Editor wie vi zum Beispiel), abspeichern und selber rennen lassen, oder auch Änderungen vornehmen. Das Skript läuft unter dem Dateinamen "mirror" und funktioniert folgendermaßen:

Ich habe mir eine freie Plattenpartition mit 400Mb Platz unter Linux eingerichtet, die nur als Backup-Partition für die Daten da sein soll. Bei mir handelt es sich um die Partition /dev/sdc8. Diese Partition ist NICHT ständig am System montiert. Wenn ich nun

Daten-Backup betreiben möchte, dann montiere ich diese Partition mit

```
mount -t ext2 /dev/sdc8 /mnt
```

an das /mnt Verzeichnis. Anschließend gehe ich dorthin, wo mein daten-Unterverzeichnis liegt, unter Linux nach /home/juergen. Es existiert dort also ein Verzeichnis /home/juergen/daten, und darunter die ganzen Daten-Unterverzeichnisse wie beschrieben. Jetzt gebe ich folgendes Kommando (das mirror Skript liegt bei mir in einem Verzeichnis /root/bin, und dieses Verzeichnis ist für den Benutzer "root" auch gepfadet)

```
mirror daten /mnt
```

Dieser Befehl kopiert alle neueren Datendateien aus dem Verzeichnis daten (also /home/juergen/daten) in ein Verzeichnis mirror.d, welches unter /mnt liegt oder angelegt wird, wenn es nicht existiert. Dabei werden die Dateien gleich mit gzip komprimiert. Bei jedem Neuaufruf von mirror wird geprüft, ob die Dateien im daten Verzeichnisbaum neueren Datums sind als die "gzipten" Kopien im mirror.d Verzeichnis.

Unter meiner M\$-WIN Partition habe ich genau so ein Daten-Verzeichnis, nach der gleichen Struktur und Philosophie geordnet. Dort liegen Daten, die ich mit den wenigen Programmen, die ich noch unter Windows laufen habe, erzeuge. Auch diese Daten sichere ich mit dem "mirror" Programm, ich montiere dazu wie oben beschrieben die Backup-Partition sdc8 an das Verzeichnis /mnt, und die Windows-Partition wie weiter oben beschrieben an das /dos/d: Verzeichnis. Bei mir liegen die Win-Daten unter D:\daten! Anschließend wechsle ich mit

```
cd /dos/d:
```

dorthin und rufe von dort aus genauso das mirror-script auf mit

```
mirror daten /mnt
```

und auch meine Win-Daten fließen in das Verzeichnis /mnt/mirror.d!

Zum guten Schluß mache ich dann regelmäßig ein Backup von diesem /mnt/mirror.d Verzeichnis mit tar auf ein Streamerband, und somit habe ich zentral alle Daten gesichert!

Noch ein Tip: Rufe ich das Programm mirror folgendermaßen auf:

```
mirror -n daten /mnt
```

so läuft das Programmchen im "showmode", das heißt es legt gar kein "mirror"-Backup an, sondern zeigt nur was es anlegen und kopieren würde.

So, nun mache ich Schluß, ich hoffe diese Zusammenstellung meiner Backup-Spielereien mit dem tar ist jetzt nicht zu langatmig geworden und ist für den Einen oder Anderen doch interessant. Wenn es noch Fragen gibt, werde ich sie gerne beantworten (wenn ich kann!), und außerdem sehen wir uns ja alle regelmäßig bei unseren Brelug-Treffen, da kann man auch weiter über dieses Thema reden! Jetzt erst einmal viel Spaß beim Lesen, Spielen und Ausprobieren, und nachstehend wie versprochen das kleine "mirror" Programm. Beim Abtrennen und Abspeichern als "mirror" nicht vergessen, es mit

```
chmod +x mirror
```

"executable" zu machen! Vielleicht hat ja auch der Eine oder Andere (Woody ganz bestimmt) Kritik oder eigene Anregungen oder Erweiterungen zu dem mirror-scriptchen auf der Pflanze!

\*\*\*\*\* hier abtrennen \*\*\*\*\*

```
#!/bin/bash
#      mirror
#      script to create/update a mirror backup from a chosen directory
#      to a target directory (adding mirror.d to the pathname)
#      using the gzip utility.
#
#      programmed Dec 03-08, 1998 by jz (df6om@delug.de) (alias jumbo!)
#      modified Dec 08, 1998 by woody (woody@walbrecht.com)
#      updated Jan 11, 1999 by jz (df6om@delug.de) (alias jumbo!)

#      setup some message strings used throughout the program

PROGNAME=mirror
USE_STR="usage: $PROGNAME [-n] SourceDir TargetDir"

#      function usage(), called whenever program mirror was invoked
#      with incorrect syntax, arguments or options

usage()
{
    echo $1; echo $USE_STR; exit 1
}

#      the following case-block checks number and content of
#      arguments/options of invokation of mirror
#      only 2 or 3 arguments/options is correct number!

case $# in
    0) usage "$PROGNAME: missing options and/or arguments!" ;;
    1) usage "$PROGNAME: missing argument!";;
    2) if [ ! -d $1 ]
        then
            usage "$PROGNAME: directory $1 does not exist! "
        fi;;
    3) if [ $1 = -n ]
        then
            Showmode=TRUE
            shift
        else
            usage "mirror: incorrect option!"
        fi;;
    *) usage "$PROGNAME: too many arguments";;
esac

#      now we check if the target-directory contains a leading slash "/",
#      it is required to keep the script simpler!

if ! expr "$2" : "^/" > /dev/null
then
    usage "$PROGNAME: target must be an absolute pathname to a directory!"
fi
```

```

#      strip leading slashes from source and target

SOURCE=`echo $1 | sed 's,/$,,'`
TARGET=`echo $2 | sed 's,/$,,'`

#      concat target-directory with a trailing "/mirror.d"

TARGET=$TARGET/$PROGRAMME.d/$SOURCE

#      if target-directory does not exist, then just make it!

[ ! -d $TARGET ] && mkdir $TARGET

#      now change to source-dir

cd $SOURCE

#      and finally the most important part
#      find all files with type "f" (ordinary files) in source(dir)
#      form a list of pathnames of these files including embedded
#      directories
#      compare if files are newer than possible existing backups
#      in target or if they are missing in target completely
#      if so, then copy to target-dir creating any embedded directories
#      also gzip the copied files, and chmod -x in case the mirror
#      is made from a DOS/WIN partition, where all files have the
#      "execute-bit" set
#      ---- That's all folks! ----

find -type f -printf "%P\n" | while read File
do
    IFS=', '
    if [ $File -nt $TARGET/$File.gz -o ! -f $TARGET/$File.gz ]
    then
        echo updating $File to $TARGET/$File.gz ...
        if [ ! $Showmode ]
        then
            cp -P $File $TARGET
            gzip -f $TARGET/$File
            chmod -x $TARGET/$File.gz
        fi
    fi
done
exit 0

```