Einführung GNU/Linux® Shell Kommandos

René Pfeiffer
GNU/Linux Manages!

pfeiffer@luchs.at
+43 676 5626390, +43 720 349387
+43 1 5046801

Versionsgeschichte

Version 0.3 13./14. August 2008
Komplette Überarbeitung, Umstrukturierung der Kapitel.
Version 0.2 24. April 2008
Anpassungen an der Struktur, Netzwerkteil überarbeitet.
Version 0.1 12. April 2008
Erster Entwurf und Grundstruktur des Dokuments mit Überblick und den wichtigsten Kommandos.

Dieses Dokument faßt die gängigsten Shell Kommandos zusammen, die man bei alltäglichen Prozeduren der Systemadministration verwenden kann. Es dient als Nachschlagewerk, um die Kommandos zusammen mit gebräuchlichen Optionen darzustellen.

1. Überblick

Diese kurze Beschreibung von wichtigen Kommandos mit den am häufigsten eingesetzten Optionen soll als Ergänzung zum Vortrag „GNU/Linux Grundlagen" dienen. Es gibt zu allen Kommandos detailierte Informationen in den Man Pages (Aufruf von **man befehl** zeigt diese an). In den Kapiteln dieses Dokuments werden Beispiele vorgezeigt, die immer das Kommando mit allen Optionen sowie die Ausgabe im Terminal veranschaulichen.

2. Benutzer- und Gruppenverwaltung

Die Benutzer- und Gruppenverwaltung eines eigenständigen GNU/Linux Systems wird durch

mindestens zwei, meist jedoch drei Dateien bestimmt.

- `/etc/passwd` - enthält alle Benutzerkonten mit deren Identifikationsnummern (UIDs)
- `/etc/shadow` - enthält alle Paßworte der Benutzerkonten in verschlüsselter Form sowie Konteninformationen (Gültigkeitsdauer, etc.); man nennt die Einträge auch *shadow passwords*
- `/etc/group` - enthält alle Gruppen mit deren Identifikationsnummern (GIDs)

Manche Systeme haben keine sogenannten *shadow passwords* und speichern die verschlüsselten Paßworte auch in der Datei `/etc/passwd`. `/etc/passwd` ist allerdings für alle Benutzer lesbar, daher sollte man die Paßworte immer in `/etc/shadow` halten.

2.1. Anlegen und Modifizieren von Gruppen

Gruppen lassen sich mit den Kommandos **groupadd** anlegen und mittels **groupmod** modifizieren. **groupdel** löscht eine Gruppe.

```
root@knightsbridge:~# groupadd reporter
root@knightsbridge:~# groupmod -n neuername alternname
```

Die Änderungen werden in `/etc/group` gespeichert.

2.2. Anlegen und Modifizieren von Benutzern

Benutzerkonten lassen sich mit den Kommandos **useradd** anlegen und mittels **usermod** verändern. Jeder Benutzer ist Mitglied einer primären Gruppe und optional Mitglied weiterer sekundären Gruppen. Neue Dateien und Verzeichnisse werden immer mit der primären Gruppe als Eigentümer angelegt.

```
root@knightsbridge:~# useradd -d /home/joe -g reporter -G wheel joe
root@knightsbridge:~# passwd joe
Changing password for joe
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
root@knightsbridge:~#
```

useradd legt zuerst ein neues Konto namens *joe* mit Home Verzeichnis `/home/joe` an. *joe* ist Mitglied der primären Gruppe *reporter* und Mitglied der sekundären Gruppe *wheel*. Nach Anlegen des Kontos wird anschließend das Paßwort durch den Befehl **passwd** gesetzt.

Der Befehl **userdel** löscht ein Benutzerkonto.

3. Umgang mit Textdateien

UNIX® Systeme sind traditionell textbasiert. Das bedeutet, daß alle Konfigurationen in Textform vorliegen und mit normalen Texteditoren verändert und eingesehen werden können. Man benötigt zum Konfigurieren keine speziellen Werkzeuge. Das Inspizieren von Textdateien in Form von Logs oder Konfigurationsdateien ist daher eine wichtige Tätigkeit.

3.1. Anzeigen von Text

Oft sind Ausgaben länger als die Anzahl der Zeilen in einem Terminal. Um Texte bildschirmweise anzuzeigen, verwendet man die sogenannten *pager*.

```
lynx@nightfall:~$ less /var/log/syslog
lynx@nightfall:~$ ls -l /usr/lib | more
```

Sowohl das Kommando **less** als auch **more** stellen die Ausgabe bildschirmseitenweise dar, erlauben das Blättern und das Suchen nach bestimmten Textpassagen. **more** ist älter als **less** und auf so ziemlich allen Systemen verfügbar. Alternativ kann man das Kommando **view** verwenden. Es ruft den Editor **vim** im Nur-Lese-Modus auf. Man kann dann die Textdatei sogar mit farblicher Syntaxbetonung lesen.

Möchte man Textdateien ohne *pager* anzeigen, so läßt sich dafür das Kommando **cat** mißbrauchen (**cat** verkettet Dateien und fügt sie zu einer Datei zusammen).

```
lynx@nightfall:~$ cat /proc/meminfo
MemTotal:      2060824 kB
MemFree:         75832 kB
Buffers:         36836 kB
Cached:        1428328 kB
⋮
lynx@nightfall:~$
```

3.2. Anfang und Ende von Textdateien anzeigen

Die Kommandos **head** und **tail** zeigen jeweils den Anfang oder das Ende einer Textdatei an. Mit der Option `-n` läßt sich die Anzahl der Zeilen festlegen. Voreingestellt sind 10 Zeilen. **tail** kombiniert mit der Option `-f` „verfolgt" eine Textdatei und zeigt in Echtzeit Zeilen an, die an diese Datei angehängt werden.

```
nightfall:~# tail -f /var/log/kern.log
Aug 13 20:25:03 nightfall kernel: Bluetooth: L2CAP socket layer initialized
Aug 13 20:25:03 nightfall kernel: Bluetooth: RFCOMM socket layer initialized
Aug 13 20:25:03 nightfall kernel: Bluetooth: RFCOMM TTY layer initialized
Aug 13 20:25:03 nightfall kernel: Bluetooth: RFCOMM ver 1.8
Aug 13 20:25:04 nightfall kernel: Bluetooth: BNEP (Ethernet Emulation) ver 1.2
Aug 13 20:25:04 nightfall kernel: Bluetooth: BNEP filters: protocol multicast
Aug 13 20:25:05 nightfall kernel: ACPI: PCI Interrupt 0000:01:00.0[A] -> GSI 16 (level, low) -> IRQ 16
Aug 13 20:25:05 nightfall kernel: PCI: Setting latency timer of device 0000:01:00.0 to 64
```

Durch die Tastenkombination *Strg-c* wird das Kommando abgebrochen.

3.3. Anzeigen von Binärdaten als Text

Die Textanzeiger zeigen normalerweise keine Binärdateien an. Es gibt allerdings einen Textfilter namens **strings**, der Dateien nach Texten durchsucht und nur diese anzeigt.

```
lynx@nightfall:~$ strings /bin/ls | head
/lib64/ld-linux-x86-64.so.2
librt.so.1
clock_gettime
_Jv_RegisterClasses
__gmon_start__
libacl.so.1
acl_entries
acl_get_file
⋮
lynx@nightfall:~$
```

Hier wird die Binärdatei des Kommandos **/bin/ls** nach Texten gefiltert und ausgegeben.

3.4. Zählen von Zeilen und Wörtern

Es gibt ein Kommando namens **wc**, welches die Analyse von Texten ermöglicht. Mittels **wc** kann man die Anzahl der Zeilen, die Worte oder die Anzahl der Buchstaben in einer Textdatei ermitteln.

```
lynx@nightfall:/tmp$ wc -l text.txt
14 text.txt
lynx@nightfall:/tmp$ wc -m text.txt
981 text.txt
lynx@nightfall:/tmp$ wc -w text.txt
187 text.txt
lynx@nightfall:/tmp$
```

Mit der Option `-l` zählt **wc** die Anzahl der Zeilen, mit `-m` die Anzahl der Buchstaben und mit `-w` die Anzahl der Wörter.

3.5. Suchen in Textdateien

Das Kommando **grep** durchsucht Textdateien nach bestimmten Inhalten. Der folgende Aufruf durchsucht die Datei `/var/log/mail.info` nach dem Text *NOQUEUE*:

```
lynx@nightfall:~$ grep NOQUEUE /var/log/mail.info
```

Soll Groß-/Kleinschreibung nicht beachtet werden, so muß man die Option `-i` angeben.

4. Systemresourcen

Unter Systemresourcen versteht man alles, was der Linuxkern verwaltet - Hauptspeicher, Prozessor, Geräte, Auslagerungsbereich und andere Teile des Systems.

4.1. Systeminformation, Laufzeit und Last

Der Befehl **uname** zeigt Informationen über das System an. Mit der Option `-a` bekommt man das Wichtigste kompakt angezeigt.

```
lynx@nightfall:~$ uname -a
Linux nightfall 2.6.26 #1 SMP Tue Jul 15 00:13:50 CEST 2008 x86_64 GNU/Linux
lynx@nightfall:~$
```

Die Informationen bedeutet aufgeschlüsselt:

- *Linux* - der Kern ist ein Linuxkern
- *nightfall* - der Name des Systems
- *2.6.26* - die Version des Linuxkerns
- *SMP* - Symmetrisches Multiprozessorsystem (SMP)
- *Tue Jul 15 00:13:50 CEST 2008* - aktuelles Datum mit Zeitzone
- *x86_64* - Systemarchitektur (AMD64 in diesem Fall)
- *GNU/Linux* - Betriebssystemtyp

Mit Hilfe von **uptime** läßt sich herausfinden, wie lange das System schon läuft und wie die Systemlast ausschaut.

```
lynx@nightfall:~$ uptime
 23:40:48 up 1 day,  3:16,  3 users,  load average: 0.08, 0.12, 0.05
lynx@nightfall:~$
```

Das System wurde vor einem Tag, 3 Stunden und 16 Minuten gestartet. Es sind 3 Benutzer eingeloggt. Die Systemlast wird durch drei Zahlen angegeben. Es wird angegeben wieviele Prozesse im Mittel ausgeführt werden oder auf Ein-/Ausgabe warten. Es sind drei Angaben, weil die Werte über drei verschiedene Zeitintervalle gemittelt werden (1, 5 und 15 Minuten). Die Last wird nicht an die Anzahl der Prozessoren angepaßt. Eine Last von 1 bedeutet auf einen System mit einer CPU 100% Auslastung, aber auf einem System mit 4 CPUs lediglich 25%. Werte nahe bei 0 bedeutet, daß das System nichts tut.

Die eingeloggten Benutzer kann man durch **w** anzeigen.

```
root@knightsbridge:~# w
 00:32:50 up 11:45,  3 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
lynx     pts/0    client030.foomat 00:32    0.00s  0.09s  0.00s sshd: lynx [priv]
```

```
root@knightsbridge:~#
```

Die zuletzt eingeloggten Benutzer und Reboots lassen sich mittels **last** anzeigen.

```
root@knightsbridge:~# last -f /var/log/wtmp.1
lynx     pts/0        client030.foomat Fri Aug  1 00:23 - 00:42  (00:19)
lynx     pts/0        client030.foomat Thu Jul 31 14:08 - 22:45  (08:37)
lynx     pts/1        192.168.15.242   Wed Jul 30 23:48 - 23:48  (00:00)
lynx     pts/0        192.168.15.242   Wed Jul 30 23:37 - 23:49  (00:12)
⋮
reboot   system boot  2.6.26           Mon Jul 14 18:33 - 00:34 (31+06:01)
lynx     pts/1        192.168.15.242   Mon Jul 14 17:41 - 18:32  (00:50)
lynx     pts/0        192.168.15.242   Mon Jul 14 13:55 - down   (04:38)
lynx     pts/0        192.168.15.242   Sun Jul 13 21:09 - 23:39  (02:30)
⋮
lynx     pts/1        192.168.15.242   Sat Jul  5 14:19 - 14:24  (00:05)
lynx     pts/0        192.168.15.242   Sat Jul  5 12:53 - 16:11  (03:17)
⋮
wtmp.1 begins Fri Jul  4 20:02:15 2008
root@knightsbridge:~#
```

last liest die Informationen immer aus der `/var/log/wtmp`. Im abgebildeten Beispiel liest das Kommando mittels der Option `-f` die Datei `/var/log/wtmp.1`, welche aus dem vergangenen Monat stammt. Neustarts sind mit *system boot* und Abstürze des Linuxkerns mit *crash* markiert.

4.2. Speicher und Systemstatistiken anzeigen

Das Kommando **procinfo** kann Speichernutzung, Interrupts, Betriebszeiten, Anzahl der Taskwechsel (Context Switches) und Last kombiniert anzeigen.

```
lynx@nightfall:~$ procinfo
Linux 2.6.26 (lynx@nightfall) (gcc [can't parse]) #???  2CPU [nightfall.luchs.at]❶

Memory:      Total        Used        Free      Shared     Buffers      ❷
Mem:       2060824     1079552      981272           0       16136❸
Swap:       995988           0      995988                   ❹

Bootup: Wed Aug 13 20:24:11 2008    Load average: 0.14 0.13 0.10 1/144 6147❺

user  :       0:13:10.86   3.2%  page in :   484250  disk 1:       52r       4w❻
nice  :       0:00:40.96   0.1%  page out:   283959  disk 2:     2609r    2053w
system:       0:03:02.93   0.7%  page act:    74519  disk 3:    13804r   17109w
IOwait:       0:00:55.82   0.2%  page dea:        0
hw irq:       0:00:15.80   0.0%  page flt:  6410068
sw irq:       0:00:17.04   0.0%  swap in :        0
idle  :       6:49:48.54  99.5%  swap out:        0
uptime:       3:25:56.64         context : 26811847

irq  0:        37 timer                 irq 18:         2 ehci_hcd:usb3, uhci_ ❼
irq  1:     34762 i8042                 irq 19:   1230553 uhci_hcd:usb5
irq  6:         3 floppy [2]            irq 22:    101500 aic7xxx, HDA Intel
irq  8:         0 rtc0                  irq 23:    101982 uhci_hcd:usb4, ehci_
irq  9:         0 acpi                  irq314:    123390 0   PCI-MSI-edge
irq 16:   1065881 ahci, uhci_hcd:usb1,  irq315:     45439 0   PCI-MSI-edge
irq 17:        69 ide0, ide1, uhci_hcd
```

```
lynx@nightfall:~$
```

❶ Version des aktiven Linuxkerns, Anzahl der CPUs, Hostname.

❷❸❹Speichernutzung des physischen Speichers und des Auslagerungsspeichers.

❺ Zeitpunkt des Bootens, Systemlast.

❻ Betriebszeiten, Disk E/A, Paging, Auslagerungsstatistik.

❼ Interrupts.

procinfo liefert kompakt die wichtigsten Systeminformationen. Alle Daten lassen sich auch einzeln abfragen. Beispielsweise findet man die Nutzung des Auslagerungsspeichers auch in `/proc/swaps`.

```
lynx@nightfall:~$ cat /proc/swaps
Filename                                Type            Size    Used    Priority
/dev/sdb5                               partition       995988  0       1
lynx@nightfall:~$
```

4.3. Prozeßliste anzeigen

Die gerade laufenden Prozesse kann man sich nach CPU-Last geordnet mit dem Kommando **top** anzeigen lassen. Von Haus aus aktualisiert **top** diese Liste alle 5 Sekunden. Durch Eingabe von *d* gefolgt von einer Zahl läßt sich diese Periode verändern.

Das Kommando **ps** zeigt ebenfalls eine Liste aller Prozesse an. Häufig läßt man sich durch den Aufruf von **ps `auxf`** anzeigen. Die Optionen bedeuten einzeln folgendes.

- `a` zeigt alle Prozesse an.
- `u` zeigt den Besitzer der Prozesse an (*user*).
- `x` zeigt auch Prozesse ohne Terminal an.
- `f` zeigt die Verwandtschaftsverhältnisse der Prozesse an (*fork*).

Die Ausgabe von **ps** kann relativ breit werden. Wenn man möchte, daß **ps** weniger Zeichen bei der Ausgabe abschneidet, dann kann man die Option `w` mehrfach verwenden. **ps** zeigt dann die Prozesse mit der vollen Kommandozeile ihres Aufrufs.

4.4. Signale an Prozesse senden

Man kann an jeden Prozeß Signale senden, die bestimmte Aktionen auslösen. Es ist dazu nur Kenntnis der Prozeß-Identifikationsnummer (PID) notwendig, welche man aus der Anzeige von **top** oder **ps** ableiten kann. Die Signale selbst werden mit Hilfe des Kommandos **kill** an den Prozeß geschickt. **kill** kennt die folgenden Signale.

```
lynx@nightfall:~$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
```

```
 5) SIGTRAP      6) SIGABRT      7) SIGBUS       8) SIGFPE
 9) SIGKILL     10) SIGUSR1     11) SIGSEGV     12) SIGUSR2
13) SIGPIPE     14) SIGALRM     15) SIGTERM     16) SIGSTKFLT
17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU
25) SIGXFSZ     26) SIGVTALRM   27) SIGPROF     28) SIGWINCH
29) SIGIO       30) SIGPWR      31) SIGSYS      34) SIGRTMIN
35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3  38) SIGRTMIN+4
39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12
47) SIGRTMIN+13 48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14
51) SIGRTMAX-13 52) SIGRTMAX-12 53) SIGRTMAX-11 54) SIGRTMAX-10
55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7  58) SIGRTMAX-6
59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
lynx@nightfall:~$
```

Von dieser Liste sind nur vier Signale für den allgemeinen Gebrauch wichtig.

- *SIGHUP* - kurzes „Auflegen” des Terminals, die meisten Prozesse lesen ihre Konfiguration neu.
- *SIGUSR1* - benutzerdefiniertes Signal, bestimmte Prozesse lesen ihre Konfiguration neu.
- *SIGTERM* - Signal zum Terminieren, Prozeß beendet sich selbst.
- *SIGKILL* - Signal zum sofortigen Terminieren, Prozeß wird vom Linuxkern beendet.

Ein Signal läßt sich unter Angabe der Signalnummer oder des Signalnamens ohne *SIG* als Option an einen Prozeß schicken. Einzig die Prozeß-ID muß noch angegeben werden.

```
nightfall:~# kill -15 5723
nightfall:~# kill -TERM 5724
nightfall:~# kill -9 12893
nightfall:~# kill -KILL 30278
```

Nur der Administrator *root* kann Signale an beliebige Prozesse senden.

Mehrere kill Kommandos

Es gibt möglicherweise mehr als ein **kill** Kommando. Beispielsweise hat die **bash** Shell ein eingebautes **kill** Kommando, welches immer zuerst ausgeführt wird. Es gibt noch das /**bin**/**kill** Kommando, dessen Optionen sich leicht von den eingebauten Kommandos unterscheiden können.

4.5. Priorität von Prozessen ändern

Der Linuxkern steuert durch das *Scheduling* wann welcher Prozeß wieviel CPU-Zeit bekommt. Dabei steuert die Priorität wie schnell ein Prozeß an die CPU kommen kann. Die Priorität wird durch eine Zahl zwischen -20 (Prozeß wird stark favorisiert) und 19 (Prozeß wird am wenigsten favorisiert) angegeben. Standardwert ist meist 0. Kernelprozesse haben negative Werte bis -5. Man kann den Prioritätswert vor dem Start eines Kommandos erhöhen (sprich der Prozeß bekommt weniger oft CPU-Zeit.

```
lynx@nightfall:~$ nice gpg --recv-key 1A5727B9
```

Durch das vorangestellte **nice** Kommando erhöht sich der Prioritätswert. Es gibt auch hier, wie bei **kill**, eine in die Shell eingebaute und eine „normale" Version. Die Priorität bereits aktiver Prozesse kann man durch den Befehl **renice** beeinflussen. Man benötigt dazu die Prozeß-ID, die man beispielsweise aus einem Aufruf von **top** oder **ps** nehmen kann.

```
nightfall:~# renice -h
usage: renice priority [ [ -p ] pids ] [ [ -g ] pgrps ] [ [ -u ] users ]
nightfall:~# renice 1 -p 10852
10852: old priority 0, new priority 1
nightfall:~#
```

Alle Benutzer können den Prioritätswert erhöhen. Nur der Administrator *root* kann den Prioritätswert eines Prozesses verringern.

5. Dateisysteme

Der Linuxkern kann eine zahlreiche Auswahl von Dateisystemen ansprechen. Die gängigsten Dateisysteme sind Ext2/Ext3, XFS, JFS und FAT16/FAT32 (auf Wechseldatenträgern). Für Flashspeicher gibt es das Journalling Flash File System (JFFSv2).

5.1. Formatieren von Blockgeräten

Das Formatieren von Blockgeräten geschieht durch das Kommando **mkfs**. Dieses Kommando wird allgemein aufgerufen. Es gibt dann aber die eigentliche Formatierung an das für das gewählte Dateisystem zuständig Kommando ab. Es gibt beispielsweise **mkfs.ext2**, **mkfs.ext3** oder **mkfs.vfat**. Diese Befehle kann man auch direkt aufrufen. Beispielsweise läßt sich so eine Partition formatieren, die vom Linuxkern als Gerät */dev/loop0* erkannt wurde.

```
root@knightsbridge:~# mkfs.vfat -v -n USBSTICK /dev/loop0
mkfs.vfat 2.11 (12 Mar 2005)
Loop device does not match a floppy size, using default hd params
/dev/loop0 has 64 heads and 32 sectors per track,
logical sector size is 512,
using 0xf8 media descriptor, with 102400 sectors;
file system has 2 16-bit FATs and 4 sectors per cluster.
FAT size is 100 sectors, and provides 25541 clusters.
```

```
Root directory contains 512 slots.
Volume ID is 48a41818, volume label USBSTICK    .
root@knightsbridge:~#
```

Der Schalter `-v` weist **mkfs.vfat** an Details anzuzeigen. `-n` gibt dem Dateisystem einen Volume Namen. Das Kürzel VFAT steht für *Virtual FAT*, einer Erweiterung des FAT Formats benannt nach dem Windows 95 VxD Treiber. Linux® faßt mit dem *vfat* Treiber alle FAT Formate (FAT32, FAT16) zusammen und kann alle verarbeiten.

Ein Ext3 Dateisystem läßt sich mit dem **mkfs.ext3** Befehl formatieren.

```
root@knightsbridge:~# mkfs.ext3 -v -j -L DiskName -O dir_index /dev/loop0
mke2fs 1.41.0 (10-Jul-2008)
fs_types for mke2fs.conf resolution: 'ext3', 'small'
Filesystem label=DiskName
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
12824 inodes, 51200 blocks
2560 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=52428800
7 block groups
8192 blocks per group, 8192 fragments per group
1832 inodes per group
Superblock backups stored on blocks:
        8193, 24577, 40961

Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 26 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
root@knightsbridge:~#
```

Man kann nun den Datenträger mit diesem Dateisystem verwenden.

5.2. Prüfen von Dateisystemen

Bestimmte Dateisysteme müssen periodisch überprüft werden. Ähnlich **mkfs** gibt es auch hier ein allgemeines und darüber hinaus spezielle Kommandos für jedes Dateisystem. Die Benennung erfolgt nach demselben Muster (**fsck.ext2**, **fsck.ext3**, . . .) mit der Ausnahme XFS, dort gibt es die Kommandos **xfs_check** und **xfs_repair**. Die **fsck** Befehle werden vom System automatisch bei Bedarf aufgerufen (primär beim Neustart). In Ausnahmefällen oder bei Wechseldatenträgern kann es jedoch vorkommen, daß man die Überprüfung von Dateisystemen selbst durchführen muß. Im Falle von Ext2/Ext3 schaut das so aus.

```
root@knightsbridge:~# fsck.ext3 -v -f /dev/loop0
e2fsck 1.41.0 (10-Jul-2008)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
```

```
Pass 4: Checking reference counts
Pass 5: Checking group summary information

      11 inodes used (0.09%)
       1 non-contiguous inode (9.1%)
         # of inodes with ind/dind/tind blocks: 0/0/0
    6550 blocks used (12.79%)
       0 bad blocks
       0 large files

       0 regular files
       2 directories
       0 character device files
       0 block device files
       0 fifos
       0 links
       0 symbolic links (0 fast symbolic links)
       0 sockets
--------
       2 files
root@knightsbridge:~#
```

Der Schalter `-v` sorgt für eine ausführliche Ausgabe. `-f` erzwingt eine Prüfung auch wenn diese nicht notwendig ist.

5.3. Einhängen und Aushängen von Dateisystemen

Dateisysteme werden normalerweise automatisch nach dem Start des Systems eingehängt. Jedes Dateisystem hat ein Gerät, auf dem es gespeichert ist, und einen Einhängepunkt (*mount point*). Die Datei `/etc/fstab` zeigt Dateisysteme und Einhängepunkte an, die das System immer verwendet. Wechseldatenträger können durch einen sogenannten *auto mounter* automatisch eingehängt werden. Hinter den Kulissen geschieht dies immer durch den Befehl **mount**. Unser eben formatiertes Dateisystem läßt sich so einhängen.

```
root@knightsbridge:~# mount -t ext3 /dev/loop0 /mnt/t
root@knightsbridge:~# df -h | grep mnt
/dev/loop0             49M  4.9M   42M  11% /mnt/t
root@knightsbridge:~#
```

Die Option `-t` gibt den Typ des Dateisystems an. Es folgt das Gerät und schließlich das Verzeichnis, an dem das Dateisystem eingehängt werden soll. Das Verzeichnis muß bereits existieren. Alle Dateisysteme lassen sich mit spezifischen Optionen einhängen, die man bei Ausführung des **mount** Befehls mit der Option `-o` angeben kann. Das wird häufig bei Netzwerkdateisystemen verwendet. Beispielsweise kann man mit diesem Kommando die Freigabe *daten* auf dem CIFS Dateiserver *w3kstorage* einhängen.

```
root@knightsbridge:~# mount -t cifs -o ip=2.3.4.5,user=login //w3kstorage/daten /net/daten
```

`-o ip=2.3.4.5,user=login` setzt die IP Adresse des Servers und den Benutzernamen zum Login (ohne Angabe von `user=` wird in diesem Fall *root* verwendet). Die Man Page von **mount.cifs** erklärt alle Optionen. Ein weiteres Beispiel ist das Einhängen einer Network File System (NFS) Freigabe.

```
root@knightsbridge:~# mount -t nfs -o tcp 10.2.3.4:/tmp /net/tmp
```

Für NFS ist die Angabe von `-t nfs` nicht zwingend notwendig.

Das Aushängen von Dateisystemen geschieht durch **umount**.

```
root@knightsbridge:~# umount /mnt/t
root@knightsbridge:~# umount /net/daten
root@knightsbridge:~# umount /net/tmp
```

umount funktioniert nur, wenn das Dateisystem von keinem Programm mehr verwendet wird.

5.4. Blockweises Kopieren von Datenträgern

Mit Hilfe des Kommandos **dd** (Abkürzung für *data definition*) kann man blockweise Daten kopieren und konvertieren. Die wichtigsten Optionen sind die folgenden.

- `if` - *input file*, gibt Quelle an
- `of` - *output file*, gibt Ziel an
- `bs` - *block size*, gibt die Größe der Blöcke an
- `count` - gibt die Anzahl der Blöcke an

Minimal notwendig sind die Parameter `if` und `of`. Blockgröße und Anzahl der Blöcke ist optional. **dd** beendet den Kopiervorgang, wenn entweder die Quelle komplett gelesen, das Ziel komplett beschrieben oder die Anzahl der Blöcke erreicht wurde. Das folgende Beispiel kopiert ein CD-ROM im Laufwerk `/dev/hdb` blockweise in eine Datei `/srv/iso/bartpe.iso` auf der Festplatte.

```
root@knightsbridge:~# dd if=/dev/hdb of=/srv/iso/bartpe.iso bs=2k
112113+0 records in
112113+0 records out
229607424 bytes (230 MB) copied, 58.224 s, 3.9 MB/s
root@knightsbridge:~#
```

Alternativ kann man mit **dd** auch Datenträger löschen. Das folgende Kommando überschreibt die eben formatierte Partition mit Nullen und anschließend mit Zufallszahlen.

```
root@knightsbridge:~# dd if=/dev/zero of=/dev/loop0
dd: writing to '/dev/loop0': No space left on device
102401+0 records in
102400+0 records out
52428800 bytes (52 MB) copied, 0.939332 s, 55.8 MB/s
root@knightsbridge:~# dd if=/dev/urandom of=/dev/loop0
dd: writing to '/dev/loop0': No space left on device
102401+0 records in
102400+0 records out
52428800 bytes (52 MB) copied, 10.8956 s, 4.8 MB/s
root@knightsbridge:~#
```

Das spezielle Gerät `/dev/zero` produziert beim Auslesen immer Nullbytes. `/dev/urandom` produziert beim Lesen Zufallszahlen.

dd überschreibt ungefragt

dd wird auch gerne *destroy disk* oder *delete data* genannt. Es überschreibt das Ziel ohne zu fragen. Man kann sich damit sehr leicht komplette Datenträger zerstören, wenn man Quelle und Ziel verwechselt oder andere Fehler macht. Man sollte sich also sehr genau überlegen wann man **dd** einsetzt.

6. Netzwerk

Die Netzwerkeinstellungen lassen sich in `/etc/network/interfaces` (Debian, Ubuntu) oder `/etc/sysconfig/network-scripts/ifcfg-eth0` (für das Netzwerkgerät *eth0)*bzw. `/etc/sysconfig/network` (Red Hat, CentOS, Fedora) einstellen. Die Datei `/etc/resolv.conf` enthält die DNS Server, die für DNS Abfragen benutzt werden. Die Textdatei `/etc/hosts` speichert statische Zuordnungen zwischen Hostnamen und IP-Adressen.

6.1. Netzwerkgeräte anschauen und konfigurieren

GNU/Linux® Systeme haben zwei Befehle, die das Konfigurieren von Netzwerkkarten ermöglichen. Ein Kommando ist **ifconfig**.

```
nightfall:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:17:31:91:13:29
          inet addr:62.116.64.107  Bcast:62.116.64.111  Mask:255.255.255.240
          inet6 addr: fe80::217:31ff:fe91:1329/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:274425 errors:0 dropped:0 overruns:0 frame:0
          TX packets:172877 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:313957912 (299.4 MiB)  TX bytes:26310915 (25.0 MiB)
          Interrupt:58

nightfall:~#
```

Hier wurde der Status der Netzwerkkarte *eth0* angezeigt. Man sieht die IP Adressen, die Ethernetadresse, die Paketstatistik und den Status *UP*, sprich aktiv. Diese Information schaut mit dem **ip** Kommando so aus.

```
nightfall:~# ip address show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:17:31:91:13:29 brd ff:ff:ff:ff:ff:ff
```

```
    inet 62.116.64.107/28 brd 62.116.64.111 scope global eth0
    inet6 fe80::217:31ff:fe91:1329/64 scope link
       valid_lft forever preferred_lft forever
nightfall:~#
```

Um ein Netzwerkgerät mit einer neuen IP Adresse zu versehen, kann man folgende Befehle verwenden.

```
nightfall:~# ifconfig eth0 down
nightfall:~# ifconfig eth0 192.168.73.14 netmask 255.255.255.0 up
nightfall:~# ifconfig eth0
```

Dasselbe mit dem **ip** funktioniert so.

```
nightfall:~# ip link set dev eth0 down
nightfall:~# ip address add 192.168.73.14/24 dev eth0
nightfall:~# ip link set dev eth0 up
```

Um die IP Adresse zu ändern, ist es normalerweise nicht notwendig das Netzwerkgerät zu deaktivieren und anschließend wieder zu aktivieren.

6.2. Netzwerkrouten anschauen und verändern

Zur Verwaltung der Routingtabelle gibt es die Kommandos **route** und **ip**. Die Routingtabelle zeigt man so an:

```
nightfall:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.16.84.1     172.16.84.5     255.255.255.255 UGH   0      0        0 eth1
172.16.84.5     0.0.0.0         255.255.255.255 UH    0      0        0 eth1
78.229.84.96    0.0.0.0         255.255.255.240 U     0      0        0 eth0
10.0.0.0        172.16.84.5     255.255.0.0     UG    0      0        0 eth1
169.254.0.0     0.0.0.0         255.255.0.0     U     0      0        0 *
0.0.0.0         78.229.84.97    0.0.0.0         UG    0      0        0 eth0
```

Das **ip** Kommando kann ebenso Routen anzeigen.

```
nightfall:~# ip route show
172.16.84.1 via 172.16.84.5 dev eth1
172.16.84.5 dev eth1  proto kernel  scope link  src 172.16.84.6
78.229.84.96/28 dev eth0  proto kernel  scope link  src 62.116.64.107
10.0.0.0/16 via 172.16.84.5 dev eth1
blackhole 169.254.0.0/16
default via 78.229.84.97 dev eth0
nightfall:~#
```

ip benutzt von Haus aus keine DNS Auflösung. Bei **route** muß man das explizit mit der Option `-n` erzwingen.

Eine Netzwerkroute zum Netzwerk *10.23.42.0/24* über das Gateway *172.16.84.5* läßt sich mit beiden Kommandos wie folgt hinzufügen.

```
nightfall:~# ip route add 10.23.42.0/24 via 172.16.84.5
nightfall:~# route add -net 10.23.42.0 netmask 255.255.255.0 gw 172.16.84.5
```

6.3. DHCP verwenden

Weder **ifconfig** noch **ip** unterstützen eine dynamische Konfiguration der Netzwerkgeräte. Um DHCP zu verwenden bedarf es zusätzlicher Programme. Eine Möglichkeit die die Verwendung von **dhclient** (Standard auf Debian und Ubuntu Systemen).

```
root@grml ~ # dhclient eth0
Internet Systems Consortium DHCP Client V3.0.6
Copyright 2004-2007 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/

Listening on LPF/eth0/52:54:00:12:34:56
Sending on   LPF/eth0/52:54:00:12:34:56
Sending on   Socket/fallback
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 8
DHCPOFFER from 172.16.84.1
DHCPREQUEST on eth0 to 255.255.255.255 port 67
DHCPACK from 172.16.84.1
bound to 172.16.84.11 -- renewal in 111443 seconds.
root@grml ~ #
```

Weitere Optionen von **dhclient** lassen sich in der Konfigurationsdatei `/etc/dhcp3/dhclient.conf` einstellen.

Eine andere Möglichkeit ist die Verwendung von **pump** (wird verwendet auf Red Hat, CentOS oder Fedora Maschinen).

```
root@grml ~ # pump -i eth0
root@grml ~ # pump -s
Device eth0
        IP: 172.16.84.12
        Netmask: 255.255.255.0
        Broadcast: 172.16.84.255
        Network: 172.16.84.0
        Boot server 172.16.84.1
        Next server 172.16.84.1
        Gateway: 172.16.84.1
        Gateways: 172.16.84.1
        Hostname: dhcppc2
        Domain: luchs.at
        Nameservers: 192.168.15.220 192.168.15.2
```

```
        Renewal time: Thu Aug 14 23:12:33 2008
        Expiration time: Fri Aug 15 00:42:33 2008
root@grml ~ #
```

6.4. Ethernetaktivität prüfen

Beim Isolieren von Netzwerkproblemen arbeitet man sich meist durch die einzelnen Schichten des OSI Modells. Zum Prüfen, ob ein Ethernetgerät an einem Switch oder Hub angeschlossen ist, kann man die Kommandos **mii-diag** oder **mii-tool** verwenden.

```
nightfall:~# mii-tool eth0
eth0: negotiated 100baseTx-FD flow-control, link ok
nightfall:~# mii-diag eth0
Basic registers of MII PHY #32:  1000 796d 001c c912 0de1 c5e1 000d 2001.
 The autonegotiated capability is 01e0.
The autonegotiated media type is 100baseTx-FD.
 Basic mode control register 0x1000: Auto-negotiation enabled.
 You have link beat, and everything is working OK.
 Your link partner advertised c5e1: Flow-control 100baseTx-FD 100baseTx 10baseT-FD 10baseT, w/ 802.3X flow control.
   End of basic transceiver information.

nightfall:~#
```

In beiden Fällen sieht man den verwendeten Ethernet Modus, wobei **mii-diag** relativ viele Informationen ausgibt, die für eine Diagnose durch Netzwerkkundige sehr nützlich sind.

Ob die Verbindung zum Ethernet richtig funktioniert, zeigt ein Blick auf den Address Resolution Protocol (ARP) Cache.

```
agamemnon:~# arp -a -n
? (192.168.15.30) at 00:05:5d:6a:3d:dc [ether] on eth0
? (192.168.15.5) at 00:14:bf:60:23:9d [ether] on eth0
? (192.168.15.4) at 00:c0:f0:4c:81:82 [ether] on eth0
? (192.168.15.10) at 00:e0:81:5f:ea:5e [ether] on eth0
? (192.168.15.1) at 00:05:5d:6a:3d:dc [ether] on eth0
? (192.168.15.11) at 00:30:1b:45:ce:33 [ether] on eth0
? (192.168.15.220) at 00:c0:f0:4d:89:2e [ether] on eth0
? (192.168.15.2) at 00:40:f4:ee:32:6d [ether] on eth0
agamemnon:~# ip neighbor
fe80::2c0:f0ff:fe4d:892e dev eth0 lladdr 00:c0:f0:4d:89:2e router STALE
fe80::2c0:f0ff:fe4d:892e dev eth2 lladdr 00:c0:f0:4d:89:2e router STALE
192.168.15.30 dev eth0 lladdr 00:05:5d:6a:3d:dc REACHABLE
192.168.15.5 dev eth0 lladdr 00:14:bf:60:23:9d STALE
192.168.15.4 dev eth0 lladdr 00:c0:f0:4c:81:82 STALE
192.168.15.10 dev eth0 lladdr 00:e0:81:5f:ea:5e STALE
192.168.15.1 dev eth0 lladdr 00:05:5d:6a:3d:dc REACHABLE
192.168.15.11 dev eth0 lladdr 00:30:1b:45:ce:33 STALE
192.168.15.220 dev eth0 lladdr 00:c0:f0:4d:89:2e REACHABLE
192.168.15.2 dev eth0 lladdr 00:40:f4:ee:32:6d DELAY
agamemnon:~#
```

7. Umgang mit Dateien und Verzeichnissen

Man wird oft Dateien manipulieren müssen. Kopieren, Umbenennen, Ändern der Berechtigungen und Löschen sind die typischen Operationen. Darüber hinaus gibt es Werkzeuge zum Archivieren, Komprimieren und Dekomprimieren von Daten.

7.1. Wechsel des aktuellen Verzeichnisses

Mit Hilfe des Kommandos **cd** (*change directory*) kann man das aktuelle Verzeichnis wechseln. Der Shell Prompt zeigt immer das aktuelle Verzeichnis an.

```
lynx@agamemnon:~ $ cd /tmp/
lynx@agamemnon:/tmp $ cd /var/tmp/
lynx@agamemnon:/var/tmp $ cd /home/lynx
lynx@agamemnon:~ $
```

Die Tilde ~ markiert immer das Home Verzeichnis des Benutzers, dem die Shell gehört. Die Eingabe von **cd** alleine oder **cd ~** wechselt immer in das Home Verzeichnis des Benutzers.

Man kann auch zwischen zwei Verzeichnissen hin- und herwechseln. Dazu verwendet man den Bindestrich - als Argument.

```
lynx@agamemnon:~ $ cd /dev
lynx@agamemnon:/dev $ cd /usr/src/build/
lynx@agamemnon:/usr/src/build $ cd -
/dev
lynx@agamemnon:/dev $ cd -
/usr/src/build
lynx@agamemnon:/usr/src/build $ cd -
/dev
lynx@agamemnon:/dev $ cd -
/usr/src/build
lynx@agamemnon:/usr/src/build $
```

7.2. Anzeigen von Verzeichnissen

Der Befehl **ls** listet Inhalte von Verzeichnissen auf.

```
root@misspiggy:~# ls /bin/
ash         cp          egrep       gzip        ls          netmsg      pwd         sync        uname
busybox     date        false       ipcalc.sh   mkdir       netstat     rm          tar         vi
cat         dd          fgrep       kill        mknod       pidof       rmdir       touch       zcat
chgrp       df          firstboot   ln          mktemp      ping        sed         true
chmod       dmesg       grep        lock        mount       ping6       sh          uci
chown       echo        gunzip      login       mv          ps          sleep       umount
root@misspiggy:~#
```

Der Schalter `-l` wie *long* zeigt Details an.

```
root@misspiggy:~# ls -l
drwxr-xr-x    2 root     root           40 Jan  1 00:00 lock
drwxr-xr-x    2 root     root           80 Jan  1 00:00 log
-rw-r--r--    1 root     root           32 Jan  1 00:00 resolv.conf
drwxr-xr-x    2 root     root          100 Jan  1 00:00 run
drwxr-xr-x    3 root     root           60 Jan  1 00:00 spool
drwxr-xr-x    2 root     root           80 Jan  1 00:00 state
drwxr-xr-x    3 root     root           60 Jan  5 05:24 usr
root@misspiggy:~#
```

Der Schalter `-a` wie *all* zeigt auch die sogenannten Punkt-Dateien (englisch *dot files*) an, die normalerweise nicht angezeigt werden.

```
root@misspiggy:~# ls -al
drwxrwxrwt    8 root     root          200 May 29 01:28 .
drwxr-xr-x    1 root     root            0 Jan  1 00:03 ..
-rwx------    1 root     root            0 Jan  1 00:00 .failsafe
drwxr-xr-x    2 root     root           40 Jan  1 00:00 lock
drwxr-xr-x    2 root     root           80 Jan  1 00:00 log
-rw-r--r--    1 root     root           32 Jan  1 00:00 resolv.conf
drwxr-xr-x    2 root     root          100 Jan  1 00:00 run
drwxr-xr-x    3 root     root           60 Jan  1 00:00 spool
drwxr-xr-x    2 root     root           80 Jan  1 00:00 state
drwxr-xr-x    3 root     root           60 Jan  5 05:24 usr
root@misspiggy:~#
```

Der einfache Punkt . ist immer das aktuelle Verzeichnis. Der doppelte Punkt .. ist immer das übergeordnete Verzeichnis. Beide zählen auch zu den *dot files*.

7.3. Manipulieren von Berechtigungen

Alle Dateien haben Eigentümer und Berechtigungen. Diese kann man mittels der folgenden Kommandos verändern.

- **chown** - ändert Eigentümer (Benutzerkonto)
- **chgrp** - ändert Eigentümer (Gruppe)
- **chmod** - ändert Berechtigungen

```
root@knightsbridge:~# chown joe /tmp/text.log              ❶
root@knightsbridge:~# chgrp adm /var/log/messages          ❷
root@knightsbridge:~# chgrp -R reporter /home/joe          ❸
root@knightsbridge:~# chmod u+rw /tmp/text.log             ❹
root@knightsbridge:~# chmod g-w /tmp/text.log              ❺
root@knightsbridge:~# chmod o-rxw /tmp/text.log            ❻
root@knightsbridge:~# chmod ugo+rw /home/joe/status        ❼
root@knightsbridge:~# chmod a+x /usr/local/bin/dailywtl    ❽
```

❶ Überträgt die Datei `/tmp/text.log` dem Benutzer *joe*

❷ Überträgt die Datei `/var/log/messages` der Gruppe *adm*

❸ Überträgt den Inhalt des Verzeichnisses `/home/joe` rekursiv der Grupper *reporter*

❹ Setzt Lese- und Schreibberechtigung für den Eigentümer

❺ Entzieht Schreibberechtigung für die Gruppe

❻ Entzieht Lese-, Schreib- und Ausführberechtigung für alle außer Gruppe und Eigentümer

❼ Setzt Lese- und Schreibberechtigung für alle

❽ Setzt Ausführberechtigung für alle

Alle Befehle verstehen den Schalter `-R` für rekursive Operation.

7.4. Anlegen und Löschen von Verzeichnissen

Die Kommandos **mkdir** und **rmdir** erzeugen bzw. löschen ein Verzeichnis. **rmdir** löscht nur leere Verzeichnisse.

```
lynx@knightsbridge:~$ mkdir verzeichnis
lynx@knightsbridge:~$ rmdir verzeichnis
```

Möchte man ganze Pfade anlegen ohne jedes Verzeichnis auf dem Weg einzeln anlegen zu müssen, so empfiehlt sich die Option `-p`.

```
lynx@knightsbridge:~$ mkdir -p pfad/mit/mehreren/verzeichnissen
```

7.5. Kopieren von Dateien

Das **cp** kopiert Dateien. Man kann mehrere Dateien gleichzeitig kopieren. Die Dateien werden in das letzte Argument, das Ziel, kopiert.

```
lynx@knightsbridge:~$ cp -v a b c d e f /etc/passwd /tmp/
‘a’ -> ‘/tmp/a’
‘b’ -> ‘/tmp/b’
‘c’ -> ‘/tmp/c’
‘d’ -> ‘/tmp/d’
‘e’ -> ‘/tmp/e’
‘f’ -> ‘/tmp/f’
‘/etc/passwd’ -> ‘/tmp/passwd’
lynx@knightsbridge:~$
```

Die Option `-v` läßt **cp** den Kopiervorgang kommentieren. Wichtige Optionen sind noch die folgenden.

- `-a` - erhält alle Berechtigungen und Eigentümer (*archive*)
- `-f` - überschreibt das Ziel ohne zu fragen (*force*)
- `-i` - fragt vor Überschreiben um Erlaubnis (*interactive*)
- `-r` - kopiert rekursiv (*recursive*)
- `-u` - überschreibt das Ziel nur, wenn es älter als die Quelle ist (*update*)

Die Optionen lassen sich beliebig kombinieren.

7.6. Umbenennen und Verschieben von Dateien

Das Umbenennen und Verschieben von Dateien wird beides durch den Befehl **mv** (*move*) ausgeführt. Die für **cp** vorgestellten Optionen gelten auch analog für **mv**.

7.7. Löschen von Dateien

Der Befehl **rm** (*remove*) löscht Dateien und Verzeichnisse. Die Optionen `-f`, `-i` und `-r` des **cp** Befehles gelten analog auch für **rm**.

```
lynx@knightsbridge:~$ rm -f a b c d e f
lynx@knightsbridge:~$ rm -i /tmp/a
```

Vorsicht beim (rekursivem) Löschen

Nur sehr wenige Dateisysteme haben die Möglichkeit Löschoperationen rückgängig zu machen. Man muß daher beim Einsatz von **rm** sehr vorsichtig sein. Gelöschte Daten können nur unter großem Aufwand oder gar nicht wiederhergestellt werden, wenn man keine Backups hat. Insbesondere das Kommando **rm `-rf`** ist mit größter Vorsicht einzusetzen, weil es rekursiv alle Dateien oder Verzeichnisse löscht ohne Fragen zu stellen.

7.8. Verwaltung von Hard- und Softlinks

Die meisten UNIX® Systeme kennen sogenannte *Hard-* und *Softlinks*. *Softlinks* werden auch *symbolische Links* genannt. Dabei handelt es sich um Verweise auf existierende Dateien oder Verzeichnisse, die sich so wie das Objekt verhalten, auf das sie zeigen. Mit Hilfe des Kommandos **ln** lassen sich diese Links anlegen. Der folgende Befehl legt einen symbolischen Link im `/tmp` Verzeichnis auf die Datei `/etc/fstab` an.

```
lynx@agamemnon:~ $ cd /tmp
lynx@agamemnon:/tmp $ ln -s /etc/fstab
```

```
lynx@agamemnon:/tmp $ ls -l fstab
lrwxrwxrwx 1 lynx lynx 10 2008-08-14 12:35 fstab -> /etc/fstab
lynx@agamemnon:/tmp $
```

ls zeigt den symbolischen Link an. Man sieht auch bei den Berechtigungen den Buchstaben *l*, was für *link* steht. Greift man jetzt auf `/tmp/fstab` zu, so wird man immer auf */etc/fstab* „umgeleitet" und liest bzw. schreibt tatsächlich diese Datei. Man kann mit diesen Umleitungen Zugriffe auf Dateien oder Verzeichnisse transparent steuern. Das ist sehr nützlich wenn sich beispielsweise Verzeichnisstrukturen verändern, man aber Pfade beibehalten möchte. Symbolische Links dürfen von beliebigen Orten auf beliebige andere zeigen, d.h. es gibt keine Einschränkungen bezüglich Partitionen oder Dateisystemen. Das Kommando **rm** entfernt symbolische Links ohne die Datei oder das Verzeichnis anzutasten, auf die der Link zeigt.

Hardlinks funktionieren auf den ersten Blick wie symbolische Links. Sie lassen sich auch durch **ln** allerdings ohne Angabe der Option `-s` anlegen.

```
lynx@agamemnon:/tmp $ ln neue_datei hardlink
lynx@agamemnon:/tmp $ ls -l neue_datei hardlink
-rw-r--r-- 2 lynx lynx 61440 2008-08-14 12:50 hardlink
-rw-r--r-- 2 lynx lynx 61440 2008-08-14 12:50 neue_datei
lynx@agamemnon:/tmp $
```

Der Eintrag `hardlink` ist jetzt ein Hardlink auf die Datei `neue_datei`. Man kann jetzt allerdings nicht unterscheiden, ob der Hardlink nur ein Verweis oder eine wirkliche Datei ist. Hardlinks können nur auf demselben Dateisystem existieren (sprich Ziel und Link müssen im selben Dateisystem vorliegen). Man setzt Hardlinks gelegentlich ein, wenn man Dateien und Verzeichnisse im selben Dateisystem kopieren möchte. Gibt man dem **cp** Kommando die Option `-l`, so wird nicht der komplette Inhalt kopiert, sondern nur ein Hardlink mit Verweis auf das Original angelegt. Das geht wesentlich schneller und spart Platz auf dem Datenträger.

7.9. Dateiarchive erzeugen und auspacken

Ein sehr gebräuchlicher Weg, um Dateien und Verzeichnisse abzupacken, ist das Kommando **tar** (abgeleitet von *tape archive*). Man kann ein ganzes Verzeichnis rekursiv in eine *tar* Datei schreiben. Der folgende Befehl erzeugt eine Archivdatei `archiv.tar` im Home Verzeichnis des Benutzers und packt alle Dateien und Verzeichnisse im Unterverzeichnis `software/` im aktuellen Verzeichnis ein.

```
lynx@knightsbridge:~$ tar -c -f ~/archiv.tar software/
```

Die Option `-c` bedeutet *create*, `-f` bedeutet *file*. Das Auspacken geschieht analog.

```
lynx@knightsbridge:~$ cd /tmp
lynx@knightsbridge:/tmp$ tar -x -f ~/archiv.tar
```

Hier wurden die Dateien im Archiv in das Verzeichnis `/tmp/` ausgepackt. Die Option `-x` bedeutet *extract*. **tar** entpackt immer ins aktuelle Verzeichnis. Möchte man das nicht, so kann man mit der Option `-C` vor dem Auspacken in ein anderes Verzeichnis wechseln.

```
lynx@knightsbridge:~$ tar -x -C /tmp/ -f ~/archiv.tar
```

Wenn **tar** als Benutzer *root* ausgeführt wird, dann werden alle Informationen über die Eigentümer der Dateien mit eingepackt bzw. ausgepackt. Normale Benutzer sind immer selbst der Eigentümer aller Daten im Archiv. Berechtigungen werden immer mitgenommen.

Man kann mittels des Schalters `-t` wie *test* und `-v` wie *verbose* den Inhalt eines *tar* Archivs inspizieren.

```
lynx@knightsbridge:/tmp$ tar -t -v -f utf8.tar
drwxr-xr-x lynx/lynx         0 2008-07-08 20:22 utf8/
-rw-r--r-- lynx/lynx      8618 2008-07-08 20:22 utf8/core.h
-rw-r--r-- lynx/lynx      8636 2008-07-08 20:22 utf8/unchecked.h
-rw-r--r-- lynx/lynx     11778 2008-07-08 20:22 utf8/checked.h
-rw-r--r-- lynx/lynx      1555 2008-07-08 20:21 utf8.h
lynx@knightsbridge:/tmp$
```

Die Ausgabe ähnelt der Ausgabe des Befehls **ls**.

7.10. Komprimieren von Daten

Es gibt zwei Kommandos, die Dateien verlustfrei komprimieren und dekomprimieren können. **gzip** ist schneller. **bzip2** komprimiert besser, ist aber langsamer. Beide Kommandos verstehen dieselben Optionen. Komprimieren und Dekomprimieren läßt sich so durchführen.

```
lynx@nightfall:~$ gzip -9 -v ctio.tar
ctio.tar:        76.7% -- replaced with ctio.tar.gz
lynx@nightfall:~$ gzip -d -v ctio.tar.gz
ctio.tar.gz:     76.7% -- replaced with ctio.tar
lynx@nightfall:~$
```

Die Schalter `-1` bis `-9` steuern den Grad der Kompression. Der Wert 1 bedeutet schnell und weniger kompakt, der Wert 9 langsamer und stärker komprimiert. Die Option `-d` wird zur Dekompression verwendet. Sowohl **gzip** als auch **bzip2** löschen nach der Operation die Originaldatei.

tar Archive sind oft komprimiert. Dies wird durch die Dateiendungen `.tgz`, `.tar.gz` oder `.tar.bz2` angezeigt. Viele Programme können solche Archive verarbeiten (beispielsweise **WinZIP**).

7.11. Suchen von Dateien

Um Dateien und Verzeichnisse zu suchen, gibt es mehrere Möglichkeiten. Die meisten GNU/Linux Distributionen verwalten eine Datenbank aller Dateien und Verzeichnisse auf dan Datenträgern. Diese Datenbank wird täglich in der Nacht aktualisiert. Darin kann man mit Hilfe des Befehls **locate** suchen.

```
lynx@knightsbridge:~$ locate traceroute
/etc/alternatives/traceroute6
/etc/alternatives/traceroute6.8.gz
```

```
/usr/bin/traceroute6
/usr/bin/traceroute6.iputils
/usr/share/man/man8/traceroute6.8.gz
/usr/share/man/man8/traceroute6.iputils.8.gz
/usr/src/build/nmap-4.65/traceroute.cc
/usr/src/build/nmap-4.65/traceroute.h
/usr/src/build/nmap-4.65/traceroute.o
/var/lib/dpkg/alternatives/traceroute6
lynx@knightsbridge:~$
```

Diese Suche geht relativ schnell, weil die Daten schon vorindiziert sind. Auf Systemen mit deaktivierter **locate** Datenbank funktioniert diese Suche nicht. In diesem Fall muß man das Kommando **find** verwenden. **find** ist sehr vielseitig. Es kann aber nur angegebene Verzeichnisse durchsuchen. Das Beispiel mit der Suche nach **traceroute** schaut dann so aus.

```
lynx@knightsbridge:~$ find /usr -name traceroute
/usr/share/doc/traceroute
/usr/bin/traceroute
/usr/sbin/traceroute
find: /usr/lost+found: Permission denied
lynx@knightsbridge:~$
```

Man muß ein Verzeichnis zum Durchsuchen angeben, in diesem Falle `/usr`. Danach gibt man mit der Option `-name` an was man sucht. Der Nachteil liegt in der fehlenden Datenbank. **find** muß für jede Suche den angegebenen Dateibaum komplett durchqueren, was sehr lange dauern kann. Weiterhin beachtet **locate** die Berechtigungen und zeigt nur Einträge an, die der aufrufende Benutzer auch sehen darf. Daher gibt es bei **find** die Fehlermeldung */usr/lost+found: Permission denied*, denn `/usr/lost+found` ist ein Systemverzeichnis. **find** wird dennoch oft in Shellskripten verwendet, um Listen von Dateien und Verzeichnissen zu erstellen.

7.12. Suchen von Kommandos im Pfad

Die Pfadvariable einer Shell gibt die Verzeichnisse an, in denen nach ausführbaren Kommandos gesucht wird. Wenn man wissen möchte ob ein Kommando im Pfad liegt, so kann man dies durch **which** erfahren.

```
lynx@nightfall:~$ which ping
/bin/ping
lynx@nightfall:~$ which arp
lynx@nightfall:~$ which telnet
/usr/bin/telnet
lynx@nightfall:~$
```

arp ist nicht im Pfad, die anderen beiden Kommandos schon.

8. Editoren

Es gibt unter den UNIX® Systemen zahlreiche Editoren verschiedener Verbreitung. Die folgenden Programme sind eine kleine Auswahl.

- GNU **nano** - sehr leicht zu bedienender Texteditor, mittlerweile recht weit verbreitet.
- **vi** bzw. **vim** - de fakto überall anzutreffender Editor; besitzt mehrere Modi (z.B. Einfügen, Kommandomodus) zwischen denen man umschalten muß.
- **mcedit** - sehr leicht zu bedienender Texteditor mit Menüführung; ist aufgrund seiner Größe sind auf allen Systemen anzutreffen.
- **joe** - Editor mit Tastaturabkürzungen, die WordStar bzw. Turbo C nachempfunden sind.
- **emacs** - sehr komplexer Editor; findet sich oft nur auf größeren Installationen.

8.1. vi Schummelzettel

Der **vi** Editor ist für Neulinge oder Ungeübte sehr gewöhnungsbedürftig. Es kann aber vorkommen, daß er der einzige Editor auf einem System ist. In diesem Fall merkt man sich am besten die folgenden Tastaturkommandos.

Tabelle 1. vi Tastaturkommandos

Kommando	Beschreibung
i	Aktiviert Einfügemodus
Escape	Verläßt Einfügemodus
:w	Speichert Textdatei
:w text.txt	Speichert Textdatei mit Dateiname `text.txt`
:w!	Erzwingt Speichern
:q	Verläßt **vi**
gg	Springt an den Anfang des Textes
G	Springt ans Ende des Textes

Ausführlichere Übersicht findet man beispielsweise auf der tuxfiles.org (http://www.tuxfiles.org/linuxhelp/vimcheat.html) oder der viemu.org (http://www.viemu.com/a_vi_vim_graphical_cheat_sheet_tutorial.html) Webseite. Letztere hat auch eine grafische Übersicht über die Tastenkombinationen zum Ausdrucken.

8.2. cat als Editor

Auf bestimmten Systemen ist gar kein Texteditor installiert. Man kann unter diesen Umständen trotzdem

Textdateien erstellen. Man bedient sich dazu des Kommandos **cat** und leitet *STDIN* in eine Datei um.

```
lynx@nightfall:/tmp$ cat > resolv.conf
search pentex.at
nameserver 10.16.23.220
nameserver 10.16.23.2
nameserver 127.0.0.1
                                                                    ❶
lynx@nightfall:/tmp$
```

❶ An dieser Stelle drückt man *Strg-d*, um das Ende der Eingabe zu markieren.

Anschließend kann man die Datei `resolv.conf` verwenden. Man muß den Inhalt nicht unbedingt eingeben. Man kann ihn auch per Markieren, Kopieren und Einfügen in das Terminalfenster aus einer anderen Applikation übernehmen.

9. Transferieren von Daten

Gelegentlich muß man Daten zwischen zwei Systemen transferieren. Es gibt mehrere Möglichkeiten dies zu tun.

- Markieren, Kopieren und direkt im Terminal Einfügen, wenn man mit einer serielle Schnittstelle verbunden ist.
- Verwendung von Modemübertragungsprotokollen auf der seriellen Schnittstellenverbindung (z.B. zmodem).
- File Transfer Protocol (FTP)
- Hypertext Transfer Protocol (HTTP, HTTPS)
- Secure Copy (SCP)
- RSYNC Protokoll
- Netzwerkprotokolle für File Sharing (NFS, SMB, CIFS)
- Secure Shell (SSH) Tunnel

9.1. FTP Transfers

Die meisten Systeme haben einen FTP Client installiert, der oft auch **ftp** heißt. Damit lassen sich Daten von und zu FTP Servern übertragen. Man muß dann die FTP Kommandos teilweise selbst eingeben. Wenn man das nicht machen möchte, so kann man alternativ **wget** oder **curl** verwenden.

```
lynx@agamemnon:/tmp $ wget ftp://ftp.univie.ac.at/systems/linux/debian/debian/README.mirrors.txt
```

```
lynx@agamemnon:/tmp $ curl -o README.mirrors.txt ftp://ftp.univie.ac.at/systems/linux/debian/debian/README.mirrors.txt
```

wget speichert die Datei automatisch unter ihrem Namen ab. Bei Verwendung von **curl** muß man die Daten mit Hilfe der Option `-o` in eine lokale Datei schreiben.

9.2. HTTP/HTTPS Transfers

Für HTTP/HTTPS kann man ebenso die Befehle **wget** und **curl** verwenden. Es gibt zusätzlich eine Reihe von textbasierten Webbrowsern, die HTTP-Funktionalität bieten. Diese sind beispielsweise **lynx**, **elinks/links** oder **w3m**.

Im Falle von Verschlüsselung (HTTPS) und nicht erkannten Zertifikaten muß man eventuell die Überprüfung des Zertifikats deaktivieren. Bei **wget** läßt sich dies mit Hilfe der Option `--no-check-certificate` einstellen.

9.3. SCP Transfers

Wenn man für ein System Zugriff via Secure Shell (SSH) hat, dann sind oft auch Datentransfers via dem Befehl **scp** (Abkürzung für *secure copy*) möglich. **scp** hat weitgehend dieselben Optionen wie das bereits bekannte **cp**. Der folgende Befehl kopiert die Datei `utf8.tar` vom Host mit der IP Adresse 192.168.15.11 auf den lokalen Rechner.

```
lynx@agamemnon:/tmp $ scp 192.168.15.11:/tmp/utf8.tar ~/
lynx@192.168.15.11's password:
utf8.tar                    100%   40KB  40.0KB/s   00:00
lynx@agamemnon:/tmp $
```

Man addiert zu den Pfaden einfach die IP Adressen oder Namen der betroffenen Systeme. Per Default versucht sich **scp** mit demselben Benutzernamen wie auf der lokalen Maschine einzuloggen. Ist dies nicht erwünscht, so setzt man *benutzer@* vor den Host, sprich **scp `benutzer@192.168.15.11:/tmp/utf8.tar ~/`**. Die Option `-C` aktiviert Kompression.

Da man Pfade kennen muß, kann das Kopieren mittels **scp** teilweise viel Schreibarbeit erfordern. Man kann alternativ zu kopierende Dateien in *tar* Archive abpacken oder auf der lokalen Maschine grafische SCP Clients verwenden, wenn möglich. **gftp** ist ein grafischer SCP Client für GNU/Linux Systeme. Unter Microsoft® Windows® kann man beispielsweise **WinSCP** verwenden.

9.4. RSYNC Protokoll

Die bisher vorgestellten Werkzeuge übertragen Daten immer komplett. Änderungen in Datenbeständen werden nicht berücksichtigt. Das RSYNC Protokoll (http://rsync.samba.org/) überträgt nur Änderungen, womit die Synchronisierung von Datenbeständen in eine Richtung realisiert werden kann. Realisiert wird die Übertragung durch das Kommando **rsync**. **rsync** kann alleine oder mit einem RSYNC Server

eingesetzt werden (das Programm **rsync** kann sowohl die Client- als auch die Serverrolle erfüllen). Anwendungsbeispiele finden sich in der Man Page **man rsync** oder in Tutorials (http://de.wikipedia.org/wiki/Rsync). Für RSYNC gibt es ebenso grafische Clients unter verschiedenen Betriebssystemen.

9.5. Transfer über File Sharing

Freigaben, die über Netzwerkprotokolle wie das Network File System (NFS) oder das Microsoft® Netzwerk (SMB, CIFS) für den Datentransfer verwendet werden, lassen sich nach dem Einhängen wie lokale Verzeichnisse ansprechen. Man kann dann alle Befehle zur Manipulation von Dateien verwenden.

9.6. SSH Tunnel

Die Secure Shell erlaubt es gleichzeitig mit Logins Datentunnel aufzubauen, um bestimmte TCP Ports von der entfernten Maschine zur lokalen oder von der lokalen Maschine zur entfernten weiterzuleiten (sogenannte *Port Forwardings*). Beides geht mit jeweils einer Option des **ssh** Befehls. Das folgende Kommando führt einen Login am entfernten Server *192.168.15.242* durch und leitet den Port 25 auf der Maschine *192.168.15.242* auf den lokalen Port 2525 weiter.

```
lynx@nightfall:~$ ssh -L 2525:192.168.15.242:25 192.168.15.242
```

`-L` markiert immer den lokalen Port gefolgt von der Adresse der entfernten Maschine und dem entferten Port. Wenn man nun lokal **telnet 127.0.0.1 2525** ausführt, so spricht man in Wirklichkeit mit dem Dienst auf Port 25 der entfernten Maschine.

Dasselbe läßt sich umgekehrt auch einrichten. Dazu verwendet man die Option `-R`.

```
lynx@nightfall:~$ ssh -R 110:172.16.84.11:7110 172.16.84.11
```

Die SSH leitet nun den lokalen Port 110 zum Port 7110 der entfernten Maschine weiter. Solche Portweiterleitungen werden oft von der `-C` begleitet, um die Kompression der Daten einzuschalten.

10. Aufbau von Shellskripten

Jede UNIX® Shell erlaubt das Ausführen von Shellskripten. Die Shell selbst fungiert dabei als Interpreter und führt alle im Skript enthaltenen Kommandos aus.

10.1. Format der Shellskripte

Shellskripte lassen sich mit jedem beliebigen Editor erstellen. Sie bestehen ausschließlich aus Text. Die allererste Zeile muß jedoch der Pfad zum Interpreter des Skripts sein. Ein sehr einfaches Skript schaut damit so aus.

```
#!/bin/sh

echo "Das echo Kommando gibt Text aus."
cd /tmp
ls -l
```

Die erste Zeile beginnt mit **#!** gefolgt vom für dieses Skript zuständigen Interpreter **/bin/sh**. **/bin/sh** ist die Default Shell und entspricht auf den meisten GNU/Linux Systemen der Bash (**/bin/bash**). Alle weiteren Zeilen stellen dann das eigentliche Shellskript dar.

Wenn ein Shellskript unter einem anderen Betriebssystem erstellt wurde, dann muß man auf die Zeilenenden achten. Microsoft® Windows® und Apple Mac OS X verwenden für die Zeilenendemarkierungen andere Kodierungen. Der Fehler äußert sich meist mit der Fehlermeldung *bad interpreter: No such file or directory*. Zur Überprüfung des Formats kann man das Skript dann mit einem Textanzeiger anschauen. Wenn man an den Zeilenenden Zeichen findet, die als *^M* dargestellt werden, dann liegt das Problem an der Kodierung. Der Befehl **dos2unix** kann das beheben. Notfalls muß man die Datei mit anderen Werkzeugen konvertieren oder die *^M* Zeichen manuell herauslöschen.

10.2. Shellvariablen

Die Shell kennt Variablen, die man beliebig belegen und verwenden kann. Variablen werden mit einem *$* markiert, aber ohne dieses Zeichen definiert.

```
#!/bin/sh

VARIABLE="Hier steht der Inhalt."
echo $VARIABLE
```

Es gibt eine Reihe von vordefinierten Variablen in der Shellumgebung. Man kann sich diese durch das Kommando **env** ausgeben lassen. Alles, was **env** ausgibt, wurde bereits beim Start der Shell definiert.

Möchte man in Skripten Argumente verwenden, die beim Aufruf mitgegeben werden, so kann man auf die Argumente mittels *$1*, *$2*, etc. zugreifen.

```
#!/bin/sh

echo "Erstes Argument: $1"
echo "Zweites Argument: $2"
echo "Drittes Argument: $3"
```

10.3. Weiterleitungen von Ein- und Ausgabe

Mit Hilfe der Operatoren >, < und | werden Ein- und Ausgabe umgeleitet. Man kann diese Operatoren ausnutzen, um die Ausgabe eines Kommandos als Eingabe eines anderen zu verwenden. Beispielsweise läßt sich mit dieser Befehlskombination der Inhalt der `/etc/passwd` Datei alphabetisch sortiert in eine weitere Datei schreiben.

```
lynx@nightfall:~$ cat /etc/passwd | sort > /tmp/sorted.txt
lynx@nightfall:~$ head /tmp/sorted.txt
backup:x:34:34:backup:/var/backups:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
boinc:x:109:116:BOINC core client:/var/lib/boinc-client:/bin/false
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
games:x:5:60:games:/usr/games:/bin/sh
gdm:x:104:109:Gnome Display Manager:/var/lib/gdm:/bin/false
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
gsmsms:x:107:1001::/var/spool/sms:/bin/false
haldaemon:x:105:110:Hardware abstraction layer:/home/haldaemon:/bin/false
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
lynx@nightfall:~$
```

cat liest die Datei `/etc/passwd` und schreibt sie nach *STDOUT*. Durch die Pipe | wird die Ausgabe an **sort** weitergeleitet, was die Ausgabe von **cat** als Eingabe verwendet. Die Ausgabe von **sort** wird wiederum in eine Datei umgeleitet. Die ersten 10 Zeilen werden mittels **head** ausgegeben.

11. Lizenz dieses Dokuments