

# Unix und Unix Administration — eine Einführung

Klaus Knopper <unix@knopper.net>

V2.3 – 13. Februar 2001

## Zusammenfassung

Unix ist ein leistungsfähiges, stabiles und äußerst umfangreiches Betriebssystem. Im Rahmen dieses Kurses sollen die grundlegenden Merkmale von Unix erläutert und veranschaulicht sowie ein Einblick in die wichtigsten Konfigurations- und Administrationsaufgaben gegeben werden.

## Gliederung

- 1. Woche** Arbeiten unter Unix
  - Unix — Geschichte und Aufbau
  - Unix — Kommandos
  - Unix im Netzwerk
  - Das X-Window System – die netzwerkfähige, graphische Oberfläche
- 2. Woche** Unix-Administration
  - Netzwerk-Konfiguration
  - Konfigurationsdateien für ...
  - Routine-Tasks wie Benutzer einrichten, Hardware-Updates...
- 3. Woche** Vertiefung und Spezialitäten
  - Unix als Workgroup-Serverplattform
  - Automatisieren von Aufgaben und Arbeitserleichterungen
  - Einbindung ins Internet, Sicherheitsfragen, ...

## Unix & Co

Unix ist grundsätzlich nicht „einfacher“ oder „schwerer“ in der Anwendung und Administration als andere Betriebssysteme. Es gibt jedoch Unterschiede in der Funktions- und „Denkweise“ sowie in Aufbau und Komplexität.

## Features von Unix

- Mehrere Aufgaben gleichzeitig (Multitasking)
- Mehrbenutzerfähig (Multiuser)
- Auf vielen Hardware-Plattformen lauffähig (portabel)
- Effiziente Ausnutzung der Ressourcen (nicht proprietär)
- hierarchisches Filesystem
- Stabilität durch eigenen Speicherbereich für jedes Programm (Virtual Memory, Speicherschutz)

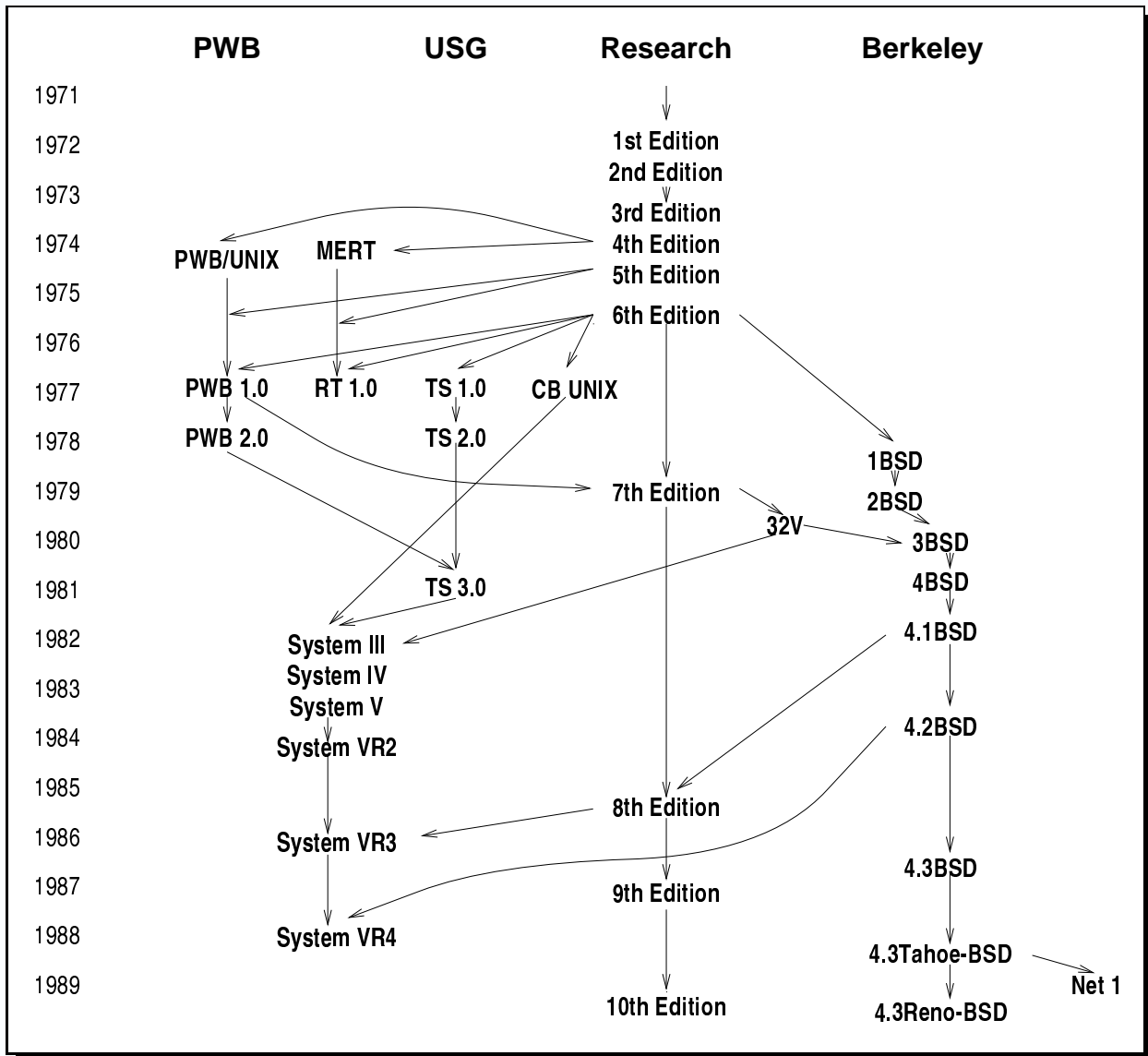
## Geschichtlicher Überblick

**um 1970:** In den Bell Labs wird von Ken Thompson ein Dokumentenverwaltungssystem (MULTICS) auf einer PDP-7 für kleine, modulare Aufgaben umgeschrieben, → Unix

**Ende der 70er:** Diverse innovative Universitäten benutzen und erweitern das System. An der University of California in Berkeley (UCB) entsteht die Berkeley System Distribution (BSD). Sie enthält u. a. Ansätze zur Netzwerkfähigkeit und zum virtuellen Speicher.

Zwischen den beiden Hauptderivaten System V Unix (aus dem AT&T-Zweig hervorgegangen) und der Berkeley-Version BSD zieht sich der Streit über die Vorherrschaft während der gesamten 80er und Anfang der 90er Jahre hin.

# Unix-Stammbaum



## Moderne Unices

1988 wird POSIX 1003.1 verabschiedet, ein Standard, der die Mindestanforderungen beider Lager vereint. Fast alle modernen Unices sind POSIX-compliant.

Es gibt eine ganze Reihe herstellereinspezifischer Varianten von Unix.

SunOS/Solaris	SUN
Aix	IBM
HPUX	Hewlett Packard
Sinix	Siemens/Nixdorf
Ultrix/DEC Unix(OSF/1)	Digital Equipment
Linux	freie Version für diverse Plattformen, z. B. Intel, Sparc
FreeBSD/NetBSD	diverse
...	

## Interoperabilität

Durch die weitgehende Standardisierung von Programmiersprachen (hauptsächlich C, C++) und Schnittstellen, sog. APIs (definiert durch POSIX) besteht weitgehende Sourcecodekompatibilität.

Den Standards entsprechender Code kann durch vergleichsweise wenig Aufwand von einem Unix-Derivat auf ein anderes portiert werden (erneute Übersetzung).



## Offene Systeme

Applikationen von verschiedenen Herstellern können auf unterschiedlichen Plattformen laufen und miteinander kommunizieren.

Beispiel: Datenbankserver von DEC, eine CAD-Anwendung auf einer Silicon Graphics, die auf diesen Server zugreift und Arbeitsplatzrechner mit Linux auf PC-Basis wären denkbar.

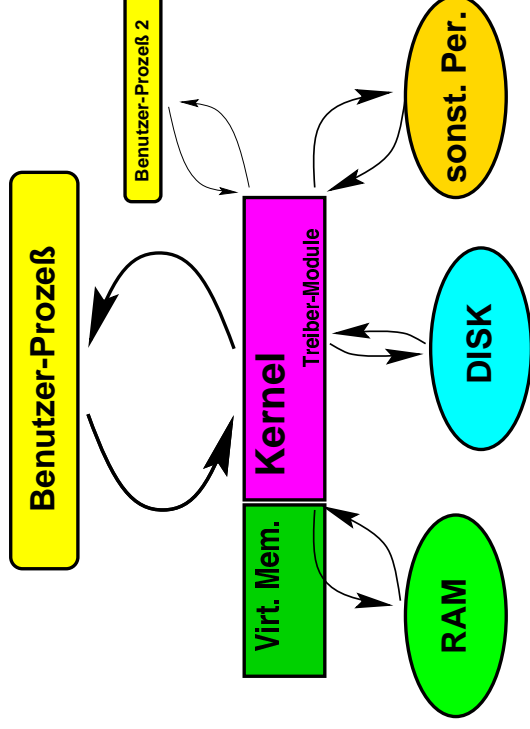
Dies ist ein großer Vorteil gegenüber proprietären, grundsätzlich auf einen Zweck spezialisierten Systemen.

## Bestandteile von Unix

- Der **Kernel** steuert die Vorgänge des Betriebes, insbesondere Zugriff auf die Hardware (Speichermedien, RAM).
- **Netzfähigkeit:** Das Internet in der heutigen Form geht auf die Netzfähigkeit von Unix zurück (erste Netzwerktechnologie, die sich für mehrere Hardware-Plattformen durchgesetzt hat).
- Graphische, netzwerkfähige Benutzeroberfläche **X-Window**.
- **Programme** und **Kommandos** für eine Vielzahl von Einsatzzwecken und angeschlossene Peripherie.

## Der Unix-Kernel

Der **Kern** eines Unix-Betriebssystems ist der Kernel. Er steuert den Ablauf der Programme im Multitasking-Betrieb (**Scheduler**) und ermöglicht ihnen den Zugriff auf die Hardware (Speicher, Dateisystem, Grafikkarte usw.) über Systemaufrufe. Dabei sorgt die **Speicherverwaltung** als Bestandteil des Kernels dafür, daß jeder Prozeß einzig und allein den ihm zugewiesenen Speicherbereich sieht und benutzen kann.



## Dateien

Viele Systemressourcen werden unter Unix auf **Dateien** abgebildet:

- gewöhnliche Dateien (Files):
  - Texte, Daten
  - Konfigurationsdateien
  - ausführbare Programme
- Verzeichnisse (Directories)
- Verweise (Links):
  - symbolische Links
  - sog. Hardlinks
- Devices etc.

Dateinamen können beliebig lang werden. Einige Dateiextensionen sind zwar üblich, aber selten verbindlich.

## Pfade

Man unterscheidet zwischen **relativen** und **absoluten** Pfaden:

- **Absolute Pfade** beginnen immer mit dem Wurzelverzeichnis / (Slash) und geben danach die einzelnen Unterverzeichnisse, getrennt durch weitere Slashes an: `/usr/bin/find`.
- Absolute Pfade gibt man für „unverrückbare“ Dateien an, z. B. `/bin/sh` in Scripten.
- **Relative Pfade** beziehen sich immer auf das **aktuelle Verzeichnis**. Um dies deutlich zu machen, schreibt man häufig auch `./daten/dateiname.txt`.
- Relative Pfade sind dann nützlich, wenn Daten verschoben werden sollen, z. B. von einem Homedirectory in ein anderes.

## Standard-Verzeichnisse (1)

(Pfade können je nach Unix-Derivat variieren)

<code>/</code>	Wurzelverzeichnis und oberste Verzeichnisebene.
<code>/bin</code>	Wichtige Programme, die immer zur Verfügung stehen müssen, auch während des Systemstarts.
<code>/usr/bin</code>	Global installierte (ausführbare) Programmdateien, die von allen Benutzern genutzt werden können.
<code>/sbin</code>	Programme, die beim Systemstart und für den Administrator gebraucht werden.
<code>/usr/sbin</code>	Programme, die nach dem Systemstart vorwiegend vom Systemadministrator gebraucht werden.

## Standard-Verzeichnisse (2)

<b>/lib</b>	Laufzeit-Bibliotheken, die auch während des Systemstarts gebraucht werden.
<b>/usr/lib</b>	Systembibliotheken, die vorwiegend von Programmen aus <code>/usr/bin</code> benötigt werden.
<b>/dev</b>	Geräte-Dateien (Abbildung der Peripherie auf Dateien).
<b>/etc</b>	Systemweite Konfigurationsdateien.
<b>/home</b>	enthält die Benutzer-Unterverzeichnisse, die für den jeweiligen Benutzer schreibbar sind.

## Standard-Verzeichnisse (3)

<code>/tmp</code>	Verzeichnis für Temporärdateien, für alle Benutzer schreibbar.
<code>/var / *</code>	Log- und Spooldateien/Verzeichnisse.
<code>/usr / local / *</code>	enthält Programmpakete, die nicht zur Standard-Distribution gehören. Manchmal per NFS geteiltes, clusterweites Verzeichnis.
<code>/proc</code>	Linux-spezifisches, virtuelles Verzeichnis, das Informationen über das System und zur Laufzeit konfigurierbare Parameterdateien enthält.

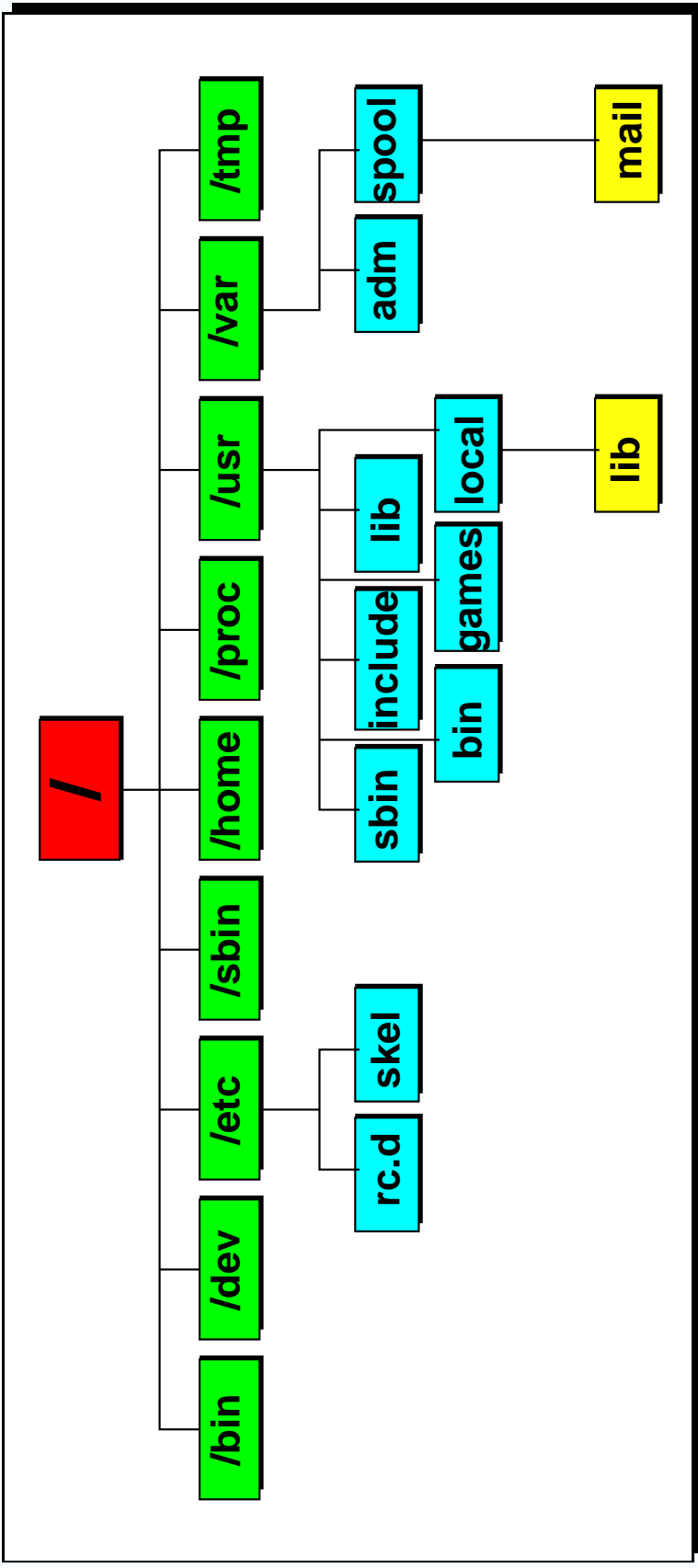
## Homedirectory

Jeder Benutzer (User) hat ein eigenes Verzeichnis, in dem er sich befindet, wenn er sich an System anmeldet (einloggt). Dieses Verzeichnis nennt man **Homedirectory**.

Die Homedirectories der User befinden sich meist im Dateisystem unter dem Pfad `/home/Username`.

In seinem eigenen Homedirectory kann jeder Benutzer seine Dateien nach eigenen Anforderungen selbst strukturieren und anlegen.

# Filesystemlayout



## Devices als Dateien

- Unter Unix sind fast alle Betriebsmittel auf Dateien abgebildet.
- Vorteil: Universeller Zugriff auf alle Gerätedateien möglich.
- Beispiel:

```
$ cat /dev/mouse  
#@$#^%$&^&UYUDFGHJK$^%&*&$%^RT&Y*U$%R^&*
```
- Devices sind im Verzeichnis `/dev` zusammengefasst.

## Die Shell

Der Befehlsinterpreter, Shell genannt, ist eins der zentralsten Programme im Betriebssystem, da jeder Benutzer ihn dazu gebraucht, um mit der Tastatur Befehle einzugeben.

Es gibt unter Unix mehrere Shells, die aber ähnlich arbeiten. Zum interaktiven Betrieb werden oft die (t) `csh`, `bash` oder `ksh` verwendet.

Die Shell ist damit das Gegenstück zu `COMMAND.COM` unter DOS, aber wesentlich leistungsfähiger als das Gegenstück, da beispielsweise Jokerzeichen (\*) in Dateinamen schon durch die Shell und nicht durch das Anwendungsprogramm aufgelöst werden.

Mit der Shell kann man auch komplexe Programme schreiben, sog. „Shellscripte“.

## Kommandos

Zur Benutzung in der Shell gibt es eine sehr große Anzahl von Kommandos für eine Vielzahl von Aufgaben. Folgende Gemeinsamkeiten gibt es bei allen Befehlen:

- Kommandoname
- Optionen
- Argumente

Zur Kombination von Programmen verwendet man **Ein- und Ausgabemlenkung** bzw. die sog. **Pipes**.

## Unix-Philosophie

Den Programmen, aus denen Unix besteht, liegt eine allgemeine „Philosophie“ zugrunde:

- Viele, kleine Programme für jeweils nur eine Aufgabe,
- Kurze, „sprechende“ Kommandonamen,
- leichte Kombinationsmöglichkeit dieser Teile,
- Forderung nach bequemerer Bedienbarkeit, z. B. durch das X-Window System und Windowmanager wie KDE und GNOME.

## Konfigurationsdateien

Um verschiedenen Benutzern unterschiedliche Konfigurationen zu erlauben, suchen viele Unix-Programme nach Konfigurationsdateien (auch Dotfiles oder Resources genannt) im Homedirectory.

Diese Dateien beginnen oft mit einem Punkt und werden daher mit dem normalen `ls`-Kommando nicht angezeigt.

Beispiele:

<code>.cshrc</code>	Konfiguration der Shell <code>cs</code> h und <code>tc</code> sh hier können z. B. Aliase definiert werden
<code>.xinitrc</code> oder <code>.xsession</code>	Datei die bei Start einer X-Window-Session ausgeführt wird: Start von X-Applikationen
<code>.plan</code>	Text, der bei Abfrage der <code>finger</code> -Information angezeigt wird.

## Editoren

Es gibt eine Reihe von Editoren unter Unix für verschiedene Anwendungszwecke.

Der **vi** ist ein sehr mächtiger Texteditor, der auf jeder Textkonsole funktioniert. Durch seine Unterscheidung von Einfüge- und Kommandomodus ist er besonders für Anfänger recht gewöhnungsbedürftig, jedoch auf jedem Unix-System mit Sicherheit vorhanden. Als Administrator sollten Sie mit der Bedienung des **vi** vertraut sein.

Den **Emacs** gibt es in einer Reihe von Implementationen, wobei der sog. **GNU-Emacs** am verbreitetsten ist. Dieser Editor erlaubt auch das bequeme Editieren sowohl mit Maussteuerung und Pulldown-Menus, als auch eine komplette Bedienung mit der Tastatur.

## Freie Software

Obwohl die meisten Unix-Varianten kommerziell vertrieben werden, gibt es im Unix-Umfeld eine Reihe von Tools, die frei weitergegeben werden dürfen und sollen.

**Freie Software** ist in vielen Standardbereichen (Textutilities, Compilern, Entwicklungstools, Administrationshilfen oder Editoren) oft qualitativ besser und zuverlässiger als Firmenprodukte.

Die sog. **GNU General Public License** der Free Software Foundation erlaubt beispielsweise die Verwendung ihrer Produkte auch für kommerzielle Zwecke, solange die Originalquellen weiterhin zur Verfügung gestellt werden.

Spezielle Anwendungssoftware ist schwieriger als freie Software zu bekommen.

## Unterschiede zu DOS

Einige augenfällige Unterschiede zu DOS:

- Groß-/Kleinschreibung wird beachtet.
- Foreslash „/“ statt Backslash „\“ als Trenner bei Pfaden.
- Es gibt keine „Laufwerksbuchstaben“.
- Umleitungen werden nicht über Hilfsdateien „simuliert“.
- Mehrere Programme können gleichzeitig laufen („&“).

## Unix-Kommandos und Pipes

In der Shell eingegebene Kommandos haben im Allgemeinen das folgende Format:

```
$ Programmname opt. Parameter Argumente Umleitung Ein-/Ausgabe
```

Die Umlenkung von Ein- und Ausgabe funktioniert, anders als bei DOS, wo Temporärdateien geschrieben werden, auf direktem Weg auch zwischen Programmen.

> Datei	Umlenkung der Ausgabe in Datei
< Datei	Umlenkung der Eingabe von Datei
Kommando	Umlenkung der Ausgabe in die Eingabe eines anderen Programmes

## cat — Dateien ausgeben

cat Datei

Der Befehl `cat` gibt den Inhalt einer Datei auf dem Bildschirm aus. Ist der Inhalt der Datei größer als eine Bildschirmseite, so kann man die Ausgabe mit dem Pipe-Symbol | in ein weiteres Programm umleiten.

```
$ cat briefe/susi.txt | more
```

Die Ausgabe wird in das Programm `more` umgeleitet und man kann durch die Ausgabe scrollen. Eine häufig verwendete Variante ist es, mehrere Dateien aneinanderzuhängen (concatenate):

```
$ cat archiv.001 archiv.002 archiv.003 > archiv.zip
```

Vergl. DOS: `copy /b daten.001+daten.002+daten.003 daten.zip`

## cd, pwd — Navigieren in Verzeichnissen

```
cd [Verzeichnis]
```

cd wechselt in das angegebene Verzeichnis. Wenn kein Verzeichnis angegeben wird wechselt cd in das Homedirectory.

pwd zeigt das aktuelle Verzeichnis an.

```
$ pwd  
/home/knopper  
$ cd briefe  
$ pwd  
/home/knopper/briefe  
$ cd ../lib/texte  
$ pwd  
/home/knopper/lib/texte  
$ cd  
$ pwd  
/home/knopper
```

## chgrp – Ändern der Gruppenzugehörigkeit

```
chgrp [Optionen] Gruppe Dateien...
```

chgrp ändert die Unix-Gruppe von Dateien und Verzeichnissen. Der Befehl kann vom Besitzer einer Datei ausgeführt werden, wenn er selbst Mitglied der angegebenen Unix-Gruppe ist (POSIX).

```
$ ls -l helloworld.c
-rw-r--r-- 1 knopper users    29 Aug  5 22:39 helloworld.c
$ groups
users developer
$ chgrp developer helloworld.c
$ ls -l helloworld.c
-rw-r--r-- 1 knopper developer 29 Aug  5 22:39 helloworld.c
```

Mit der Option `-R` kann rekursiv die Gruppenzugehörigkeit ganzer Verzeichnisbäume geändert werden.

## chmod – Ändern von Rechten

chmod [Optionen] Änderungen Dateien

chmod ändert die Zugriffsrechte von Dateien und Verzeichnissen.

Man kann **Rechte**

r = read lesen	w = write schreiben	x = execute ausführen
-------------------	------------------------	--------------------------

an bestimmte **Personenkreise** vergeben

u = user Besitzer	g = group Gruppe	o = others Andere
----------------------	---------------------	----------------------

Mit der Option `-R` werden die Änderungen auch für Unterverzeichnisse durchgeführt.

## Beispiele zu chmod

```
$ ls -l
total 11
-rw-r--r-- 1 knopper users 7185 Nov 20 23:17 auswertung.sh
-rw----- 1 knopper users 938 Nov 20 23:17 juli.dat
-rw----- 1 knopper users 469 Nov 20 23:17 juni.dat
-rw----- 1 knopper users 54 Nov 20 23:17 mai.dat
```

```
$ chmod u+x auswertung.sh
```

Das Script `auswertung.sh` wird zum Ausführen freigegeben.

```
$ chmod og+r *.dat
$ ls -l

total 11
-rwxr--r--  1 knopper  users      7185 Nov 20 23:17 auswer-
tung.sh
-rw-r--r--  1 knopper  users      938 Nov 20 23:17 juli.dat
-rw-r--r--  1 knopper  users     469 Nov 20 23:17 juni.dat
-rw-r--r--  1 knopper  users      54 Nov 20 23:17 mai.dat
```

Alle dürfen ab jetzt die „.dat“-Dateien lesen.

```
$ chmod -R o-rwx /home/knopper/tex
```

Entfernt jegliche Rechte für andere (außer der Gruppe) im TEX-Verzeichnis des Benutzers knopper und darunter.

## Spezielle Dateiattribute

Neben den Standard-Rechten Lesen, Schreiben und Ausführen existieren noch weitere Dateiattribute, die vom Besitzer einer Datei oder vom Systemadministrator gesetzt werden können.

```
$ chmod u+s /usr/bin/cdrecord
$ ls -l /usr/bin/cdrecord
-rwsr-xr-x 1 root root 13956 May 10 17:31 /bin/cdrecord
```

Durch das Setzen des s-Attributes ("s-Bit") für den Besitzer bzw. die Gruppe einer Datei wird beim Ausführen der Datei der Besitzer bzw. die Gruppe des neuen Prozesses auf den Besitzer bzw. die Gruppe der Datei gesetzt.

Eines der wichtigsten Programme, die diese Eigenschaft benutzen, ist das **su**-Kommando, das beim Start zunächst (wegen des s-Bit für den Dateibesitzer **root**) mit Systemadministrator-Rechten läuft. Anders wäre das Wechseln vom Benutzer- zum Systemadministratorstatus nicht möglich.

## chown – Setzen des Dateibesitzers

`chown` [Optionen] Login Dateien...

`chown` ändert das Besitzer-Attribut von Dateien und Verzeichnissen. Der `chown`-Befehl kann auf POSIX-konformen Unix-Systemen nur vom Systemadministrator ausgeführt werden.

Der ursprüngliche Besitzer der Datei verliert mit sofortiger Wirkung die Besitzer-Rechte an dieser Datei und kann nur noch aufgrund gesetzter Gruppen- oder globaler Rechte auf die Datei oder das Verzeichnis zugreifen.

```
$ chown -R demo /home/demo
```

Mit der Option `-R` kann rekursiv das Besitzerattribut ganzer Verzeichnisbäume geändert werden.

## compress/gzip — Komprimieren von Dateien

compress Datei, gzip Datei

compress komprimiert die angegebene Datei und hängt die Kennung .z an. Neuerdings wird vermehrt gzip zum selben Zweck gebraucht, weil es bessere Kompressionsraten liefert. Die Endung von gzip lautet .gz.

Entpacken kann man solche Dateien mit uncompress bzw. gunzip. Dies ist häufig nötig bei tar-Archiven:

```
$ ls  
XForms.tar.gz  
$ gunzip XForms.tar.gz  
$ ls  
XForms.tar
```

## cp — Kopieren von Dateien

cp [Optionen] Quelle Ziel

cp kopiert Dateien. Sollen mehrere Dateien auf einmal kopiert werden, so können Jokerzeichen (\* oder ?) angegeben werden, die von der Shell zu den entsprechenden Datei- oder Verzeichnisnamen ergänzt werden.

```
$ cp *.txt lib/texte
```

Es werden alle Dateien im Verzeichnis mit der Endung „.txt“ in das Verzeichnis lib/texte kopiert. Mit der Option -r bzw. -a können auch ganze Verzeichnisbäume kopiert werden.

```
$ cp -a lib/texte briefe
```

Das Verzeichnis lib/texte (und alle darin befindlichen Dateien) wird in das (neuangelegte) Verzeichnis briefe kopiert.

## du — Dateigrößen anzeigen

du [Optionen] [Verzeichnis]

du gibt Information über den Speicherbedarf von Dateien bzw. Verzeichnissen aus. Wenn im Verzeichnis-Parameter eine Dateipezifikation angegeben wird (etwa \* oder \*.txt), dann liefert du eine Liste mit der Größe der so angegebenen Dateien.

```
$ du tabellen
56  tabellen
$ du tabellen/*
15  tabellen/ausgaben.tab
23  tabellen/einnahmen.tab
17  tabellen/gehaelter.tab
```

## exit — Beendet eine Shell

exit

Mit `exit` beendet ein Anwender seine Arbeit in der aktuellen Kommandoshell auf der Konsole bzw. in einem X-Fenster. Ist dies die einzige oder die Haupt-Shell gewesen, von der aus alle weiteren Prozesse gestartet wurden, wird der Benutzer ausgeloggt.

**WICHTIG:** Das Beenden der Login-Shell bedeutet nicht, daß alle Tochterprozesse automatisch ebenfalls beendet werden. Wurde ihre Ein- und Ausgabe umgelenkt und wurden sie im Hintergrund („&“) gestartet, so laufen sie unbeeinflusst weiter.

## find — Suchen von Dateien

```
find Startverzeichnis -name Suchmuster -print
```

Das Kommando `find` lokalisiert Dateien in Verzeichnisbäumen. Dazu muß das Startverzeichnis angegeben werden, ab dem gesucht werden soll, z. B. „.“ für das aktuelle Verzeichnis. Das Suchmuster kann Jokerzeichen enthalten, sollte in diesem Fall jedoch in einfachen Anführungszeichen stehen:

```
$ find /usr -name sendmail -print  
/usr/lib/sendmail
```

Die Datei „`sendmail`“ wurde unterhalb von `/usr` gesucht.

**Vorsicht:** Sehr allgemeine Suchmuster und „hohe“ Startverzeichnisse (etwa „/“) können eine hohe Systemlast verursachen!

## finger — Informationen über Benutzer

```
finger [Loginname@Rechnername.Domain]
```

finger gibt Informationen über den angegebenen Benutzer aus (voller Name, wann die E-Mail gelesen wurde, Zeitpunkt des letzten Login). Wendet man `finger` auf einen lokalen User an, so kann man den Hostnamen weglassen.

Information, die in der eigenen `finger`-Information angezeigt werden sollen, legt man in der Datei `.plan` in seinem Homedirectory ab.

```
$ finger knopper@sushi
Login: knopper
Directory: /home/knopper
On since Sat Aug 9 17:20 (MET DST) on tty2 from ppp38
6 seconds idle
Mail last read Sat Aug 9 17:21 1997 (MET DST)
No Plan.
Name: Klaus Knopper
Shell: /bin/tcsh
```

## grep — Suchen in Dateien

```
grep Suchmuster [Dateien]
```

Mit `grep` können Textdateien einfach auf das Vorkommen bestimmter Wörter, den Suchmustern, hin untersucht werden. Das einfachste Suchmuster ist das Wort selbst:

```
$ grep knopper /etc/passwd  
knopper:x:26001:100:Klaus Knopper:/home/knopper:/bin/bash
```

Suche nach dem Wort „knopper“ in der Passwortdatei. Die gefundenen Zeilen werden ausgegeben.

## Erweiterte Suchmuster

Als Suchmuster können sog. **Regular Expressions** angegeben werden, mit denen auch sehr komplexe Suchen möglich sind:

```
$ grep '^ma[gr]\' /etc/passwd
marekki:x:26020:100:Marek Malcherek:/home/marek:/bin/tcsh
magnus:x:26001:100:Nils Magnus:/home/magnus:/bin/tcsh
```

Hier werden Zeilen gesucht, die mit `ma` beginnen (dazu das `^`) und danach mit `g` oder `r` weitergehen.

## Optionen von grep

Mit der Option `-v` kann man die Suchanfrage negieren, also sich diejenigen Zeilen ausgeben lassen, in denen der Suchbegriff **nicht** vorkommt:

```
$ grep -v n /etc/passwd  
lp:x:71:8:0000-lp(0000):/usr/spool/lp:  
uucp:x:5:5:0000-uucp(0000):/usr/lib/uucp:
```

Um unabhängig von Groß- oder Kleinschreibung zu suchen, kann die Option `-i` verwendet werden:

```
$ grep -i magnus /etc/passwd  
magnus:x:26001:100:Nils Magnus:/home/magnus:/bin/tcsh  
pfeffer:x:26078:100:Magnus Pfeffer:/home/pfeffer:/bin/tcsh
```

## kill — Prozesse beenden

```
kill [signal] Prozeßnummer
```

`kill` versendet Signale an einen laufenden Prozeß. Wenn `kill` ohne die `Signal-Option` ausgeführt wird, wird der angeführte Prozeß mit dem `Signal TERM` beendet. Die Signale geben an, **wie** der Prozeß gestoppt werden soll: Das `Signal HUP` (Hang Up) gibt dem Programm die Möglichkeit, sich „sauber“ zu beenden, oder dient bei einigen Systemprozessen (daemons) dazu, Konfigurationsdateien neu einzulesen. Bei hartnäckigeren Fällen hilft das `Signal KILL`, gegen das sich kein Programm wehren kann.

**\$ kill 1234** beendet den Prozeß 1234

**\$ kill -KILL 1233** beendet den Prozeß, falls der obere Versuch nicht wirkt.

## In — Anlegen von Links

```
ln [Optionen] Zielname [Lokalname]
```

Links kann man mit dem Kommando `ln` anlegen. Da man fast immer **symbolische Links** benötigt, ist die Option `-s` fast obligat. Zielname ist die Datei, auf die der Link **zeigen** soll. Lokalname ist der Name, unter dem **der Link selbst** ansprechbar ist. Fehlt er, wird er aus dem Dateiname des Zielnamens gebildet.

```
$ ls
abrechnung.dat  auswertung-september-1996-2.daten
$ ln -s auswertung-september-1996-2.daten ausw
$ ln -s /bin/sh
$ ls -l
-rw-r--r-- 1 knopper unixag 188 Nov 22 02:26 abrechnung.dat
lrwxrwxrwx 1 knopper unixag 8 Nov 22 02:31 ausw -> auswertung-september-1996-2.daten
-rw-r--r-- 1 knopper unixag 880 Nov 22 02:26 auswertung-september-1996-2.daten
lrwxrwxrwx 1 knopper unixag 8 Nov 22 02:31 sh -> /bin/sh
```

## lpr — Dateien drucken

```
lpr [Optionen] Dateiname
```

lpr druckt die angegebene Datei aus. Mit der Option `-PName` kann der Drucker ausgewählt werden.

```
$ lpr rechnung.txt  
$ ls -l | lpr -Plaserwriter
```

Druckt zunächst die Datei `rechnung.txt` auf dem Defaultdrucker, danach das Listing des aktuellen Verzeichnisses auf einem Laserdrucker.

## ls — Verzeichnisse ansehen

ls [Optionen] [Pfade | Dateimuster]

ls zeigt eine Liste aller Dateien und Verzeichnisse an und entspricht im wesentlichen dem DOS-Kommando DIR.

Wenn ls ohne Optionen ausgeführt wird, liefert das Kommando eine mehrspaltige, nach Dateinamen sortierte Tabelle, in der alle Dateien, Links und Verzeichnisse im aktuellen Verzeichnis angezeigt werden.

Wird ein Dateimuster angegeben, so werden alle Dateien gezeigt, die auf das Muster passen:

```
$ ls *.dat  
rechnungen.dat mahnungen.dat erledigt.dat
```

## Optionen von ls

Der Befehl `ls` hat eine Reihe von nützlichen Optionen:

- `ls -l` zeigt Dateien und deren Zugriffsrechte in ausführlicher Form an.
- `ls -a` zeigt auch „versteckte“ Dateien an, die mit einem Punkt („.“) beginnen.
- `ls -d` zeigt nur den Namen eines Verzeichnisses an, nicht aber seinen Inhalt.
- `ls -t` sortiert die Dateien nach Datum und Uhrzeit der letzten Änderung.
- `ls -R` erfasst dann auch Dateien in Unterverzeichnissen.

## Ausgabe von ls

Die Ausgabe des Kommandos `ls -l` liefert eine Menge an Informationen über die Dateien im aktuellen Verzeichnis:

```
$ ls -l
total 659
drwx----- 8 knopper  unixag  512 Nov  8 00:17 Mail
drwxr-xr-x  2 knopper  unixag  512 Oct 29 15:26 News
drwxr-xr-x  2 knopper  unixag  512 Jan 17 1996 TODO
-rw-r--r--  1 knopper  unixag 81920 Nov 18 20:42 sicherung.tar
drwxr-xr-x  2 knopper  unixag  512 Oct 31 00:08 agenten
-r--r--r--  1 knopper  unixag   39 Nov 12 20:19 arbeit
-rw-rw-rw-  1 knopper  unixag   37 Nov 13 13:41 kommentare
```

## man — Onlinehilfe

man [Gruppe] [Optionen] Suchwort

man zeigt Online-Informationen zum angegebenen Kommando bzw. zur angegebenen Datei an. Durch die Angabe einer Gruppe kann die Suche eingeschränkt werden. Wichtige Gruppen sind 1 (Benutzerkommandos) und 5 (Konfigurationsdateien).

```
$ man pwd
```

```
PWD(1)
```

```
PWD(1)
```

```
NAME
```

```
pwd - print name of current/working directory
```

```
SYNOPSIS
```

```
pwd
```

```
pwd {--help,--version}
```

## mkdir — Verzeichnisse erzeugen

`mkdir` [Optionen] Verzeichnis

`mkdir` erstellt ein neues Verzeichnis.

```
$ mkdir tabellen
```

Erzeugt ein neues Verzeichnis mit dem Namen „tabellen“.

```
$ mkdir -p tabellen/berichte/maerz
```

erstellt auch Zwischenverzeichnisse. Wenn die Verzeichnisse `tabellen/berichte` noch nicht existieren, werden auch diese erstellt.

## more — Seitenweise Anzeige

more [Datei]

more zeigt den Inhalt einer Textdatei seitenweise an. Nach jeder Seite wird die Anzeige unterbrochen und `more` wartet auf eine Tastatureingabe:

LEERTASTE	eine Seite nach unten
RETURN	eine Zeile nach unten
b	eine Seite nach oben
q	Beenden

`more` eignet sich auch zum Anzeigen langer Listen, die durch andere Kommandos erzeugt werden:

```
$ ps aux | more
```

Die (evtl. lange) Liste der Prozesse wird seitenweise angezeigt.

## mt — Steuern des Bandlaufwerkes

```
mt [-f device] Kommando [Zahl]
```

mt sendet Befehle zum Vor- und Zurückspulen, Löschen oder Abfragen von Statusflags an das Bandlaufwerk („magnetic tape“).

```
$ mt -f /dev/tape status
```

Es werden die Statusflags des Tapedevices /dev/tape ausgegeben.

## mt Kommandos

Die Namen der Kommandos differieren zwischen den verschiedenen Unix-Derivaten. Unter Linux und BSD werden folgende Varianten häufig benutzt:

<b>Kommandoname</b>	<b>Wirkung</b>
<code>rewind</code>	Band wird an den Anfang zurückgespult
<code>erase</code>	Band wird initialisiert (Daten werden gelöscht)
<code>fsf [Zahl]</code>	Band wird Zahl Archive vorgespult
<code>bsf [Zahl]</code>	Band wird Zahl Archive zurückgespult
<code>eod</code>	Band wird bis zum Ende des letzten Archives vorgespult

## mv — Verschieben und Umbenennen

```
mv Quelle Ziel
```

mv benennt eine Datei um bzw. verschiebt eine oder mehrere Dateien in ein Verzeichnis. Quelle und Ziel können sowohl Verzeichnisse als auch Dateien sein.

```
$ mv susi.txt moni.txt
```

Die Datei „susi.txt“ wird zu „moni.txt“ umbenannt.

```
$ mkdir entsorgung
```

```
$ mv *.txt entsorgung
```

Alle Dateien mit der Endung „.txt“ werden in das Verzeichnis „entsorgung“ gelegt.

## passwd — Änderung des Passwortes

```
passwd
```

passwd ermöglicht die Veränderung des aktuellen Passworts. Dazu müssen zuerst das alte und dann zweimal hintereinander das neue Passwort eingegeben werden.

Normalerweise sind Passwörter 8 signifikante Zeichen lang und unterschieden Groß- und Kleinschreibung. Passwörter sollten nicht leicht zu erraten sein oder in Wörterbüchern vorkommen.

```
$ passwd  
Enter login password: *****  
New password: *****  
Re-enter new password: *****
```

## ps — Prozessliste anzeigen

```
ps [Optionen]
```

`ps` zeigt die Liste der laufenden Prozesse (=Programme) an. Das Kommando ist insbesondere im Zusammenspiel mit `kill` sehr praktisch, um hängenden Programme „gewaltsam“ zu beenden. `ps` ist eines der Programme, die sich in der BSD und System V-Variante unterscheiden:

‣ `ps -ef` (SYS V) oder `ps aux` (BSD) geben jeweils eine ausführliche Prozessliste aus.

```
USER      PID  %CPU  %MEM  SIZE  RSS  TTY  STAT  START  TIME  COMMAND
root      1    0.0  0.0   796   128  ?    S     Jun 12  0:58  init
root      2    0.0  0.0     0     0  ?    SW    Jun 12  3:45  (kflushd)
root      3    0.0  0.0     0     0  ?    SW    Jun 12  0:00  (kpiod)
root      4    0.0  0.0     0     0  ?    SW    Jun 12  6:53  (kswapd)
root     296  0.2  0.0   820   188  ?    S     Jun 12 101:09  syslogd
knopper  4832  0.1  0.8  3044  2148  pf  S     23:09  0:30  gv unixkurs.ps
knopper  4936  0.0  0.3  1436   996  q3  S     23:15  0:01  vi unixkurs.tex
```

## quota — Anzeige der Plattenbelegung

```
quota [Optionen]
```

Auf Mehrbenutzerechnern wird den Benutzern oft nur ein bestimmter Plattenplatzverbrauch zugestanden. Dieser kann mit dem Befehl `quota` überprüft werden.

```
$ quota -v
Disk quotas for knopper (uid 26001):
Filesystem  usage  quota  limit  files  quota  limit  timeleft  timeleft
/home       46451  50000  2997944  3068  50000  90000  ---      ---
```

Gibt eine Tabelle mit den Maximal- und Ist-Werten der Quota aus.

```
$ quota
```

Zeigt nur beim Überschreiten der Soll-Werte etwas an.

## rlogin — Login auf anderem Rechner

```
rlogin Hostname -l [Loginname]
```

rlogin ermöglicht es, sich auf einen anderen Rechner einzuloggen und dort eine Kommando-Shell zu öffnen.

```
$ rlogin Toaster.knopper.net -l knopper
```

Login unter dem Benutzernamen knopper auf dem Rechner Toaster.knopper.net.

## rm — Löschen von Dateien

rm [Optionen] Datei

rm löscht die angegebenen Dateien. Verzeichnisse werden — sofern nicht die Option `-r` (=rekursiv) gesetzt ist — nicht entfernt.

```
$ rm -rf verluste.dat
```

löscht ohne Rückfrage (auch Verzeichnisse). Vorsicht!

```
$ rm -i briefe/*
```

Fragt für jede Datei im Verzeichnis „briefe“ nach, ob sie gelöscht werden sollen.

## **rmdir** — Leeres Verzeichnis entfernen

`rmdir` [Optionen] Verzeichnis

`rmdir` löscht das angegebene Verzeichnis. `rmdir` kann nur ausgeführt werden, wenn das Verzeichnis leer ist. Eventuell vorhandene Dateien kann man mit `rm -r` entfernen.

```
$ rmdir -p lib/briefe
```

Löscht auch die Unterverzeichnisse (wenn sie leer sind).

## talk

```
talk Loginname [@Rechnername.Domain]
```

talk öffnet eine interaktive Text-Verbindung zu einem anderen Benutzer, evtl. auf einem anderen Rechner. Der angerufene Benutzer bekommt eine Aufforderung auf dem Bildschirm angezeigt, ebenfalls das talk-Kommando aufzurufen.

```
$ talk knopper@miraculix
```

Aufforderung an den Benutzer knopper, sich zu melden. Ihm wird folgende Bildschirmausgabe angezeigt:

```
$ Message from Talk_Daemon@miraculix at 1:44 ...  
talk: connection requested by knopper@idefix.  
talk: respond with: talk knopper@idefix
```

## tar — Tape Archivierer

tar [Optionen] Kommando [Dateien]

tar speichert Dateibäume in einem Archiv (Datei oder Bandlaufwerk) oder liest Dateien und Verzeichnisse aus einem Archiv zurück.

```
$ tar tvf /dev/tape
```

Gibt das Inhaltsverzeichnis des auf dem Bandlaufwerk /dev/tape gespeicherten Archivs aus.

```
$ tar cvf /dev/tape /home
```

Die Dateien unterhalb dem Verzeichnis /home werden auf Band gesichert.

## Optionen von tar

- f Archiv      Benutzt Archiv als Archivname
- v              Aktuell bearbeitete Dateinamen und Status-  
informationen ausgeben
- t              Inhaltsverzeichnis des Archives ausgeben
- x Datei...    Datei... aus dem Archiv restaurieren
- c Datei...    Datei... in neuem Archiv speichern
- d Datei...    Datei... aus dem Archiv löschen

Die angegebenen Dateien können auch Verzeichnisse sein. In diesem Fall werden auch alle Dateien und Unterverzeichnisse innerhalb dieses Verzeichnisses mitgesichert/restauriert/gelöscht.

## whatis — Information über Programm

```
whatis [Kommandoname]
```

whatis gibt eine kurze Beschreibung des angegebenen Kommandos. Der Befehl funktioniert nur, wenn eine man-Seite existiert.

```
$ whatis man      -- find and display reference manual pages
man (1)           -- macros to format Reference Manual pages
man (5)
```

## vi — Seitenorientierter Texteditor

### vi Datei

`vi` ist ein unscheinbarer, aber dennoch sehr mächtiger Fullscreen-Editor, der (im Gegensatz zum sehr beliebten Emacs) zum Standard-Equipment jedes Unix-Systems gehört. Er ist eines der Hauptwerkzeuge des Systemadministrators zur Bearbeitung von Konfigurationsdateien.

Der `vi` kennt zwei Modi: Der **Kommando-Modus**, in dem die Bearbeitung des Textes mit Hilfe von Kommandos und Makros geschieht, und den **Direkteingabe-Modus**, in dem die Tastatureingaben direkt in den Text übernommen werden.

Im Kommandomodus sind komplexe Operationen (interaktives suchen und ersetzen, automatisches Formatieren von Textstellen, Record und Replay, Speichern, Laden, Anfügen, ...) mit wenigen Eingaben möglich, während im Direkteingabemodus der getippte Text unverändert von der Tastatur übernommen wird.

Direkt nach dem Start befindet sich der `vi` normalerweise im Kommandomodus!

## vi — Kommandomodus

< **Escape** >

**i** Rückkehr vom Insert- in den Kommandomodus  
Wechsel in den Insert-Modus (Direkteingabe)

**o** Open: Neue Zeile anfügen  Insert

**dd** Delete: Aktuelle Zeile löschen

**p** Paste: [Gelöschten] Text einfügen

**x** Zeichen, auf dem der Cursor steht, löschen

**:r Datei** Read: Datei ab Cursor einfügen

**:w** Write: Datei speichern

**:q** Quit: vi beenden

**:wq** Write and Quit: Speichern und Beenden

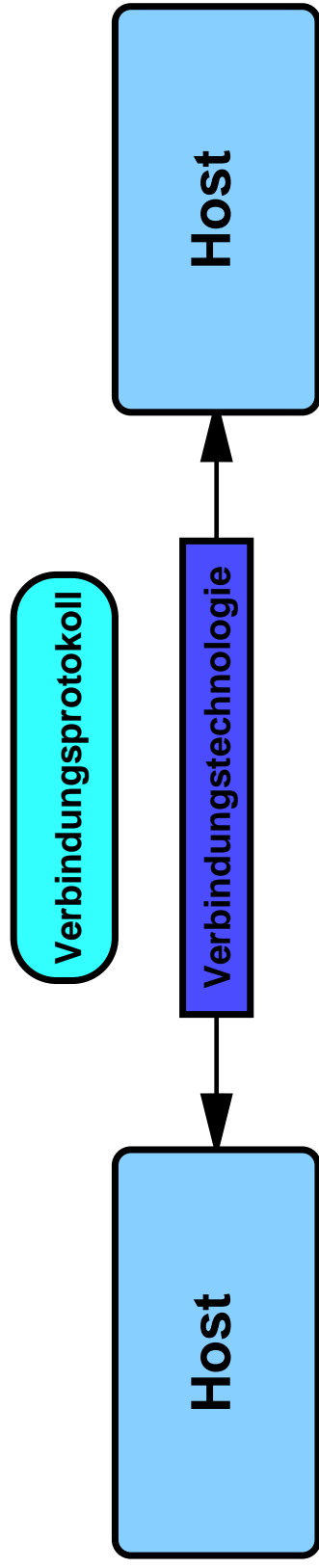
**:q!** Beenden ohne Speichern

**:%s/alt/neu/gc** Interaktiv `alt` durch `neu` ersetzen

Fast alle Kommandos lassen sich gruppieren oder mit einer vorangestellten Zahl mehrfach ausführen.

## Netzwerke

Das einfache Grundkonzept der Realisierung von Netzwerken hat wesentlich zum Erfolg von Unix beigetragen und letztlich auch Entwicklungen wie das Internet erst möglich gemacht.



## Hosts und IP-Nummern

Jede direkt und aktiv an ein Netz angeschlossene Komponente, also

- Rechner
- Router
- Netzdrucker

müssen eindeutig identifiziert werden können. Dazu bekommen sie Ihre **IP-Nummer**, beispielsweise 192.168.100.1.

## IP-Netze

IP-Nummern werden von einer zentralen Instanz, dem sog. NIC (Network Information Center) in „Netzen“ verteilt, die sich durch ihre „Anfangsnummern“ unterscheiden:

- A-Class-Netze: prinzipiell  $> 16$  Mio. Adressen; heute nicht mehr verteilt; Beispiel eines Hosts: 40.3.2.1
- B-Class-Netze: prinzipiell  $> 65.000$  Adressen; für große Firmen oder Universitäten; Beispiel: 131.246.89.8
- C-Class-Netze: bis zu 254 verwertbare Adressen; Beispiel: 192.168.91.42

Die IP-Nummern werden langsam knapp, daher wird gegenwärtig IPv6, die „nächste Generation“ entwickelt.

## IP-Namen

Um IP-Nummern besser für den Menschen besser darstellbar zu gestalten, wurden sog. **IP-Namen** eingeführt, z. B. `sunsite.unc.edu` oder `www.bundesregierung.de`.

Die Abbildung von Namen auf Nummern erfolgt mittels **DNS** (Domain-Name-Service) durch Verwendung von **Nameservern**. Die Namen sind mehr oder weniger hierarchisch strukturiert. Einige Toplevel-Domains sind:

*.com	„Commercial“, hauptsächlich amerikanische Firmen
*.de	Deutschland
*.fr	Frankreich
*.net	Netzwerkanbieter
*.org	„Non-Profit-Organisations“
*.uk	Großbritannien

## Integration von Verbindungsarten

Die unterschiedliche räumliche Entfernung spielt bei den folgenden Netzen keine Rolle:

- Lokale Netze (LANs): Eine Firma, eine Abteilung, ein Institut
- Weitverkehrsnetze (WANs): landesweite Netze, Datex-P, das Internet
- Metropolitan Area Networks (MANs): auf Stadt oder Region beschränkt

Gleichbehandlung der Verbindungsarten.

## Verbindungstechnologien

Netzverbindungen können „transparent“ über eine Reihe von „Medien“ übertragen werden:

- Ethernet: Koax-Kabel, verdrehte Telefondrähte
- Telefon/ISDN, serielles Kabel: PPP-Protokoll
- Glasfaser: FDDI-Ringe
- ATM (Asynchronous Transfer Mode)

## Protokolle

Es reicht nicht, „etwas“ zu übertragen, es muss vereinbart werden, **was wie** verschickt wird. Dazu definiert man Protokolle.

Basisprotokoll ist TCP/IP, nach dem der „Netzwerkstack“ von Unix benannt ist:

Erlaubt reine unstrukturierte, aber sichere Ende-zu-Ende Datenverbindung

Anwendungsprotokolle definieren verschiedene Dienste:

Benutzerdienste	Basisdienste	Hilfsdienste
E-Mail	NFS	NTP
News	NIS	identd
FTP	Finger	
World-Wide-Web		

# Netzwerkanwendungen

Offline-Dienste	
<b>E-Mail</b> Elektronische Post Lesen/Schreiben	<b>News</b> Elektronische Magazine Artikel Lesen/Schreiben
Online-Dienste	
<b>FTP</b> Datei-Transfer	<b>Talk</b> Kommunikation von Rechner zu Rechner
<b>TELNET</b> Arbeitssitzung auf fernem Rechner	<b>IRC</b> Kommunikation über Server oder Rechner/Rechner
<b>Info-Systeme</b>	
<b>TELNET</b> Textbasiert	<b>GOPHER</b> Texte/Menüs
	<b>WWW</b> Hypertext

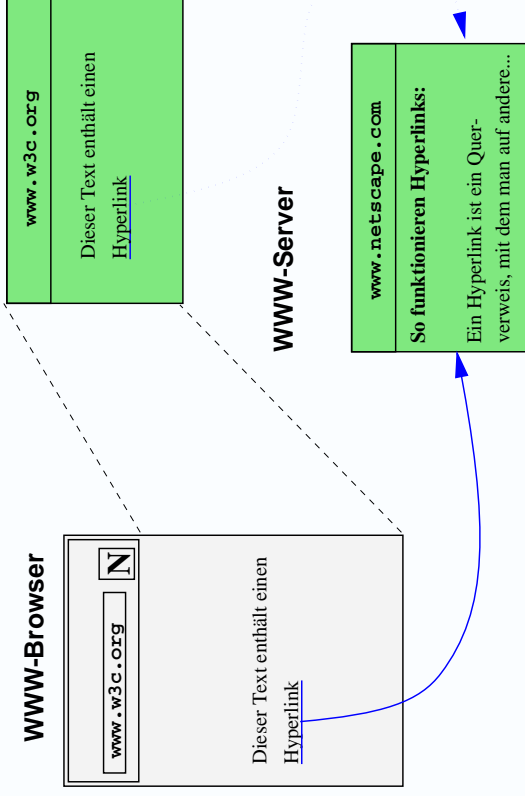
... und viele andere mehr.

## Nicht nur, aber auch Internet: World-Wide-Web

Das populäre World-Wide-Web ist ein verteiltes Hypermedia-Informationssystem:

- **Informationssystem:** Server, der in irgendeiner Form strukturierte Informationen zu einem oder mehreren Themenbereichen in Dokumentform anbietet.
- **verteilt:** Server stehen dezentral, weltweit verteilt. Gegenwärtig gibt es einige 10.000 Server mit diversen Mio. Seiten.
- **Hyperlink:** Die Dokumente sind durch ausgewiesene Stellen direkt im Dokument miteinander verkettet (typischerweise durch unterstrichenen Text).
- **Multimedia:** Dokumente können Texte, Grafiken, Bilder, Geräusche oder neuerdings sogar Programme sein.

## Aufbau des World-Wide-Web




Auf einem Server werden unter bestimmten Dokumentennamen, den sog. **URLs** (Uniform Resource Locator) Dokumente angeboten.

Ein Anwender weist seinen Browser (z. B. Netscape oder lynx) durch Angabe eines Start-URLs an, ein Dokument von dem angegebenen Server zu holen und anzuzeigen.

Durch Anklicken von Querverweisen („Hyperlinks“) werden neue Dokumente ausgewählt und angezeigt (Neudeutsch: „Surfen“).

## Probleme des WWW

Beim „Surfen“ stellt man schnell die ungeheure Größe und relative Unordnung des World-Wide-Web fest:

- „Lost in Cyberspace“: Ohne geeignete Selektionsmechanismen wird man von der Informationsfülle regelrecht „erschlagen“.
- Fehlende Struktur: Es gibt keine explizite Anordnung der Informationen, jeder Webserver ist anders aufgebaut.
- Unklare Gültigkeit: Oft ist nicht ganz einsichtig, wie korrekt die angebotenen Informationen überhaupt sind. Es wimmelt von halbfertigen Seiten. „This page is under construction...“ 

Fazit: Durch die ungeheure Expansion werden wir alle immer mehr vom World-Wide-Web mitbekommen; es fragt sich nur in welcher Form.

## Datenschutz und -sicherheit

**Datenschutz** ist das Ziel, vor der missbräuchlichen oder widerrechtlichen Nutzung von Informationen zu schützen; **Datensicherheit** ist das dazu verwendete Mittel. Aspekte der Datensicherheit

enthalten:

- Authentizität (ist jemand derjenige, für den er sich ausgibt?),
- Vertraulichkeit (bekommt nur derjenige die Informationen mit, für den sie bestimmt waren?),
- Autorisierung (ist gewährleistet, dass nur Berechtigte einen Dienst nutzen können?),

## Datensicherheit

Methoden, um Datensicherheit zu gewährleisten:

<b>Schutzart</b>	<b>Zu schützender Bereich</b>
Passwörter	Rechnerzugang, Login
Zugriffsrechte	Dateizugang (Lesen, Schreiben)
Kryptographie, PGP	E-Mail-Versand, allg. Daten
Verschlüsselte Übertragung	Ersatz rsh/rlogin  ssh/slogin, Secure Socket Layer (SSL) im WWW
Einmalpaßwörter	Zugang über unsichere Kanäle

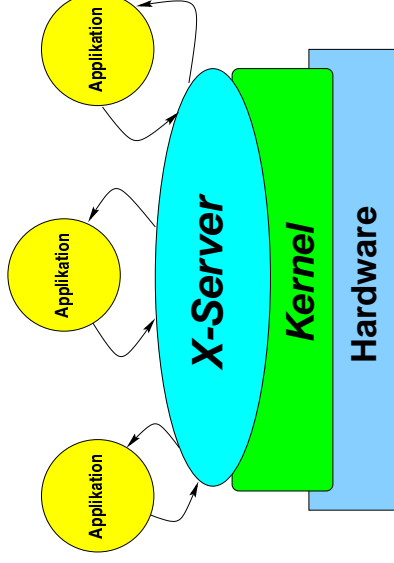
## Datenschutz, rechtliche Fragen

Für den Umgang mit EDV-Anlagen, insbesondere bei vernetzten, gelten eine Reihe von Gesetzen und Vorschriften:

- Urheberrechtsgesetz (Kopieren von Software),
- Strafgesetzbuch (Verbreitung strafrechtlich relevanter Inhalte, Ausspähen von Daten),
- Presserecht (Veröffentlichung von WWW-Homepages und sonstigen Informationen),
- Grundgesetz (Briefgeheimnis, Zensurverbot),
- Fernmeldeanlagengesetz (Rechner im Internet fallen darunter).

## X-Server - Schema

Der X-Server stellt den „Grafikartentreiber“ zur Verfügung und kümmert sich als Server-Applikation um das Zeichnen von Grafikelementen, die von Clients angefordert werden (lokal oder übers Netz).



Während sich der Kernel um die direkte Ansteuerung der Hardware kümmert, kennt der X-Server die Grafikchip-spezifischen Eigenheiten verschiedener Karten.

## X-Server (1)

Der X-Server kann ein oder mehrere virtuelle „Screens“, angeben durch Displaynummern, in unterschiedlicher Farbtiefe und Auflösung verwalten. Beispiel:

```
$ X :0 -bpp 8
```

```
$ X :1 -bpp 24 -s 5
```

## X-Server (2)

Unter Linux kann der X-Server (XFree86-Implementation) mit folgenden Tastenkombination gesteuert werden (die Zeichen – und + befinden sich auf dem Nummernblock):

Control Alt F1	Zurück auf Textconsole 1
Control Alt F2	Zurück auf Textconsole 2
	...
Control Alt +	Auflösung eine Stufe höher
Control Alt -	Auflösung eine Stufe tiefer
Control Alt Backspace	X-Server terminieren

## X-Server (3)

Der X-Server ist ein (netzwerkfähiger) Dämon, der die Schnittstelle zwischen dem Grafik- und Soundsubsystem des Rechners und dem Anwenderprogramm darstellt.

Das Anwenderprogramm ist hier der Client, der vom Server Dienste wie das Zeichnen von Fenstern, Textdarstellung usw. anfordert, die vom Server abgearbeitet werden. Einen Zugriffsschutz bietet die Authentifizierung mittels sog. *Magic Cookies* (s. Folie 91), die vom Anwenderprogramm wie ein Passwort bei jeder Anfrage an den Server mit geschickt werden, damit dieser ihre Anforderungen bearbeitet.

## X-Server (4)

Im Netzwerk wird der X-Server, der auf einem Display läuft, unter der Adresse

Rechnername:Displaynummer

angesprochen. Es können also grafikorientierte Programme, die auf mehreren anderen Rechnern laufen, das eigene Display benutzen (wenn der "Besitzer" des Displays dies zulässt).

Beispiel:

```
$ xterm -display pizza.unix-ag.uni-kl.de:0
```

## X-Sessions und X-Displaymanager (xdm)

xdm verwaltet grafikorientiert den Zugang zu einem Unix-System (ähnlich `getty/login` im Textmodus).

Es gibt 2 Varianten der Zugriffs auf den xdm:

1. Login-Fenster (`xlogin`) holen mit **X -query Rechnername**
2. Auswahlfenster (`chooser`) holen mit **X -indirect Rechnername**

xdm ist auch selbst in der Lage, einen oder mehrere X-Server auf dem lokalen System zu starten (Konfigurationsdatei `Xservers`).

Nach erfolgreicher Anmeldung eines Benutzers über xdm wird dessen X-Window Startscript `.xsession` ausgeführt.

## Konfigurationsdateien des xdm

`xdm-config`

Graphische Einstellungen, Verweise auf weitere Konfigurationsdateien

`Xservers`

X-Displays und X-Server Programme, die von xdm verwaltet werden

`Xaccess`

Zugriffskontrolle und Hostlisten für den Chooser

`Xsetup`

globale Startup-Datei, wird ausgeführt, bevor das Login-Fenster erscheint

`Xsession`

globale Startup-Datei, wird nach dem Einloggen ausgeführt

`$HOME/.xsession`

Benutzer-Startupdatei, wird nach dem Einloggen ausgeführt

`Xrreset`

globale Startup-Datei, wird nach Beenden der Session ausgeführt

`xdm-errors`

Fehler-Logdatei des xdm

`$HOME/.xsession-errors`

Benutzer-Fehlerlogdatei

# Authentifizierung



Damit ein X11-Programm Zugriff auf den X-Server bekommt, muss es diesem eine Art Passwort beim Verbindungsaufbau schicken. Dieses Passwort wird beim Einloggen unter X-Windows automatisch erzeugt, und in die Datei `$HOME/.Xauthority` geschrieben. Es kann mittels des Programmes `xauth` zwischen verschiedenen accounts und Rechnern ausgetauscht werden. Natürlich muss der jeweilige Benutzer Zugriff auf die `.Xauthority`-Datei des jeweiligen „Besitzers“ des Displays haben, um dessen Cookie kopieren zu können.

```
$ xauth list
Blackbox:0 MIT-MAGIC-COOKIE-1 53704f169dc76acd515247578d9a0f55
131.246.98.59:0 MIT-MAGIC-COOKIE-1
ea264cb4c32064236e5fcc121e8c9423

$ xauth add xterminal:0 MIT-MAGIC-COOKIE-1
83f31aed984fff202c64350851cbfd04
```


## Windowmanager

Im Gegensatz zu anderen Betriebssystemen mit graphischer Benutzeroberfläche bietet der X-Server keinerlei standardisierte Bedienelemente für Grafikobjekte („Fenster“). Diese Aufgabe wird stattdessen von einem frei wählbaren und vom Benutzer konfigurierbaren *Windowmanager* übernommen. Dieser versieht automatisch alle neu erscheinenden Fenster mit einem Rahmen und Bedienelementen („Handles“) zum Bewegen, Iconifizieren und Verändern der Größe. Einige Windowmanager unterstützen mehrere „virtuelle“ Desktops, zwischen denen per Mausclick oder Tastatur gewechselt werden kann.

Der Windowmanager ist eine normale Applikation wie jeder andere X-Window Client. Prinzipiell ist sogar ein Arbeiten ganz ohne Windowmanager möglich.

## Beispiel für Bedienelemente bei Fenstern

Verschiedene Schaltflächen dienen zum Vergrößern und Verkleinern sowie zum Iconifizieren oder Schließen der Fenster.



```
calzone:/home/magnus/Mail
magnus@calzone:~/lib/tex/uni~vortrag/RRHT$ ls -l
total 5
drwxr-xr-x  2 magnus  unixag  512 Nov 20 18:40 farben
drwxr-xr-x  2 magnus  unixag  512 Nov 20 18:41 graph-package
drwxr-xr-x  2 magnus  unixag  512 Nov 22 02:31 x
drwxr-xr-x  2 magnus  unixag  512 Nov 20 20:06 y
drwxr-xr-x  2 magnus  unixag  512 Nov 20 18:43 z
magnus@calzone:~/lib/tex/uni~vortrag/RRHT$ cd Mail/
magnus@calzone:~/Mail$
```

# Vergleich der Benutzerführung verschiedener Betriebssysteme

- **DOS:** Ein textbasierter **Kommandozeileninterpreter** (Shell)  
☞ /bin/sh
- **Windows: Mehrere Fenster**, „statisches Multitasking“  
☞ Shellscript, das diverse X-Window Applikationen aufruft (.xinitrc).

## Benutzeroberflächen

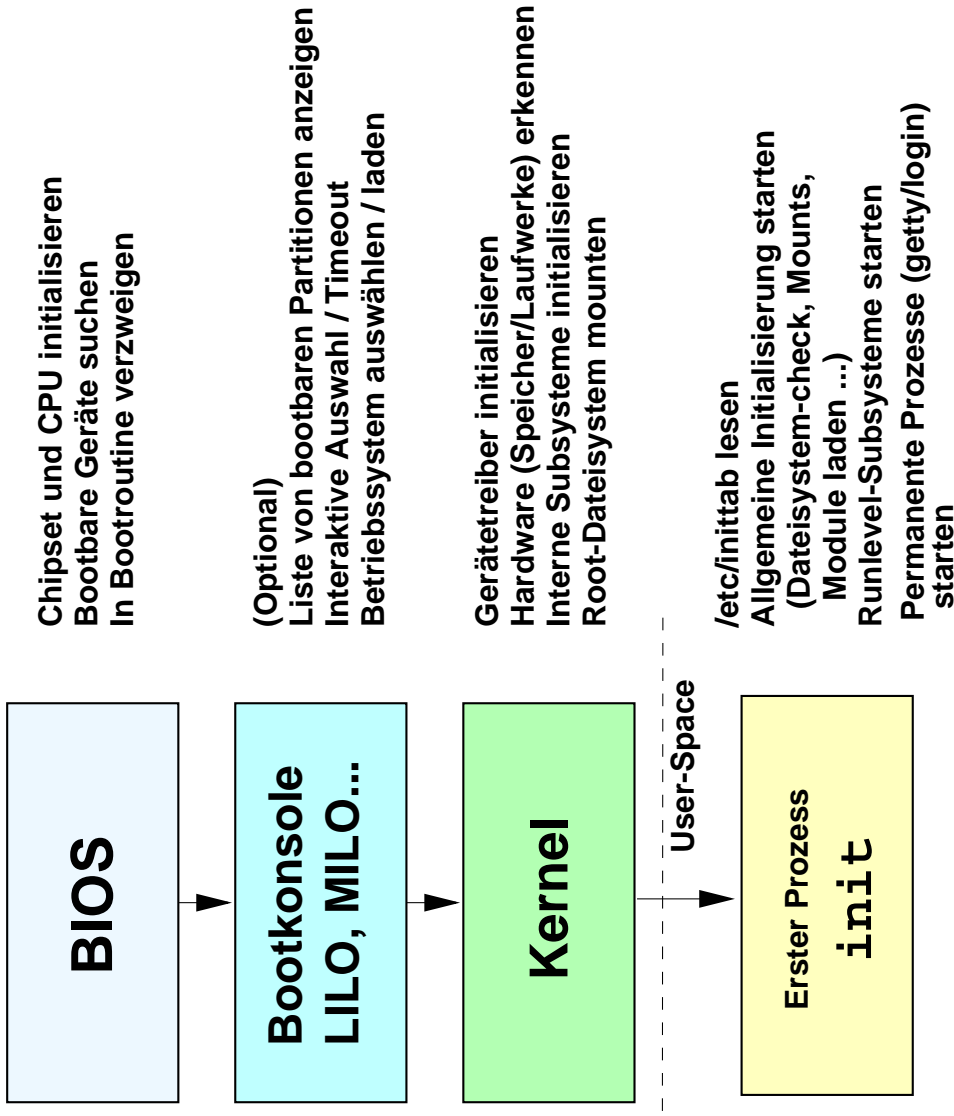
Name	System	
olwm	OpenWindows	Sun, Solaris
olwvm	wie olwm,	aber mit virtuellen Desktops
twm	X11	sehr einfacher Windowmanager, wenig Einstellmöglichkeiten
twm	X11	wie twm, aber mit virtuellen Screens
mwm	X11/Motif	Aix, Solaris, Irix...
fwmm2	X11	Linux, Windowmanager mit virtuellen Screens, sehr viele Einstellungsmöglichkeiten
fwmm95	X11	wie fwmm, mit standardmäßigen Windows95-Layout
KDE	X11	Linux, „Plug&Play“-Windowmanager

## Zustandsmanagement mit `init`

Das Programm `init` hat eine zentrale Bedeutung für den Betrieb eines Unix-Systems. Hauptsächlich kommt unter Linux das an System V angelehnte Paket zu Einsatz. `init` ist das erste Programm, welches nach dem Booten des Kernels die Aufgabe des Zustandsmanagements übernimmt, wobei der Zustand als sog. „Runlevel“ modelliert wird. `init` bestimmt also, welche Programme und Dämonen in welchem Systemzustand aktiv sind.

Beispiele hierfür sind der **administrative Zustand**, in dem ausschließlich der Systemadministrator arbeitet, oder der **Mehrbenutzer-Zustand**, der als der „Normalzustand“ eines Unix-Systems gilt.

# Booten von Linux



## init-Varianten

- **BSD-Style**  
wenige Skripte für alle Aufgaben
- **System V-Style**  
Aufteilung in Subdirectories und einzelne Skripte  
(`/etc/rc.d/*`)

 **Linux-eigene Änderungen**

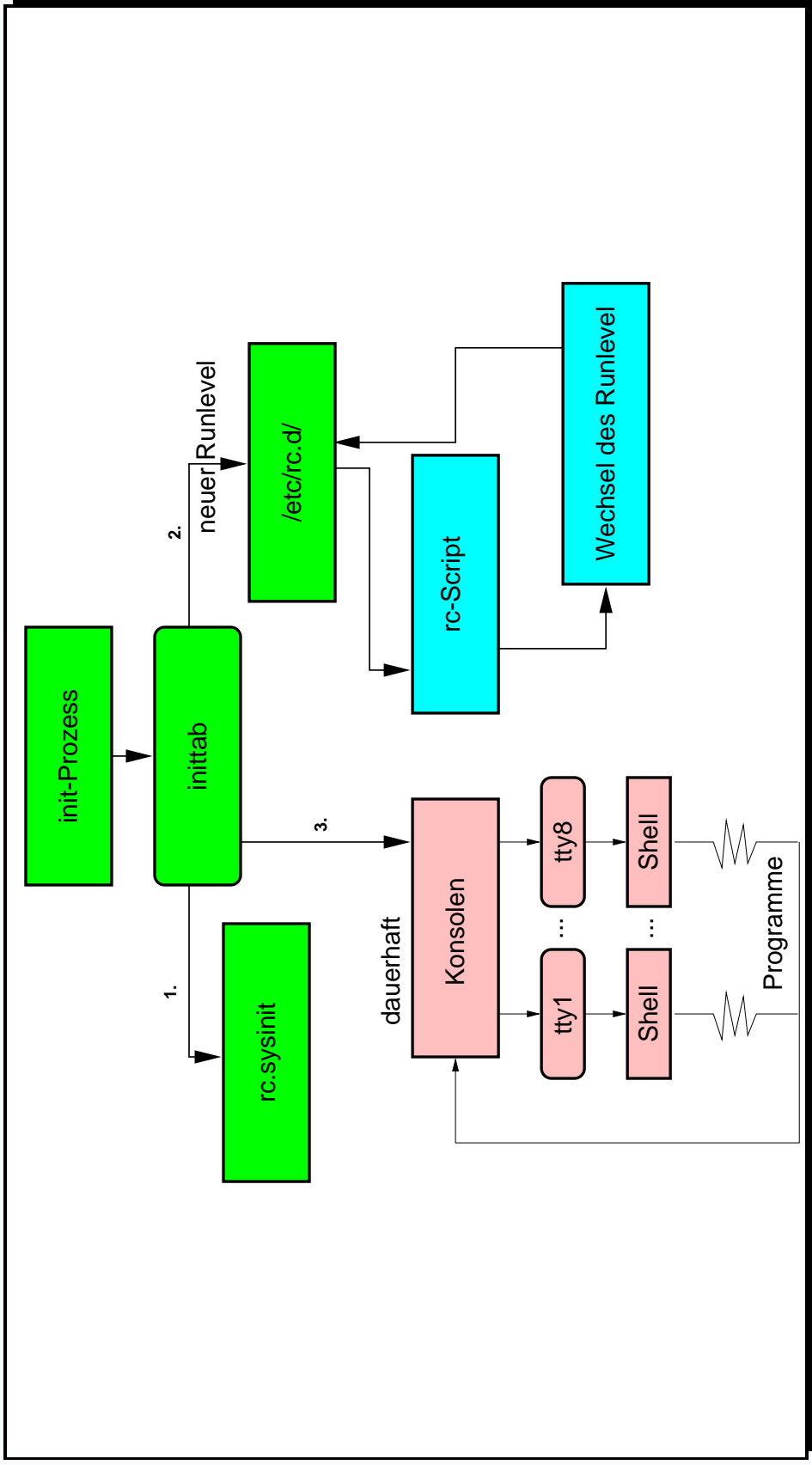
## Aufgaben von init

- **System-Initialisierungen** (`rc.sysinit`),
- **dauerhaft zu kontrollierende Prozesse** (Logins, USV-Kontrolle),
- **Subsysteme in bestimmten Systemzuständen** (Netz-Server, Host-Server, Remote-Filesystem, ...)

**Initialisierungen** und **dauerhafte Einstellungen** ändern sich selten.

**Subsysteme** werden häufiger im laufenden Betrieb umkonfiguriert  
☞ **Runlevel.**

# Ablauf von init



## Was sind Runlevel?

- Modellieren den „**Gesamtzustand**“ des Systems
- **Gruppieren** die zur Verfügung stehenden Services und Subsysteme
- **Aktivierung** von dynamischen Systemkomponenten, z. B. Netzwerkdämonen oder NFS
- Bezeichnung durch eine Ziffer (oder einige Buchstaben)

## Beispiele für Runlevel

- 1** Eingeschränktes System nur für den Systemverwalter zu Administrationszwecken (Single-User-Mode)
- 2** Komplettes System, fertig für Benutzerbetrieb, ohne Netzwerk Filesystem
- 3** Komplettes System, fertig für Benutzerbetrieb, mit Netzwerk Filesystem
- 5** Komplettes System, fertig für Benutzerbetrieb, mit Netzwerk Filesystem und X-Window Login

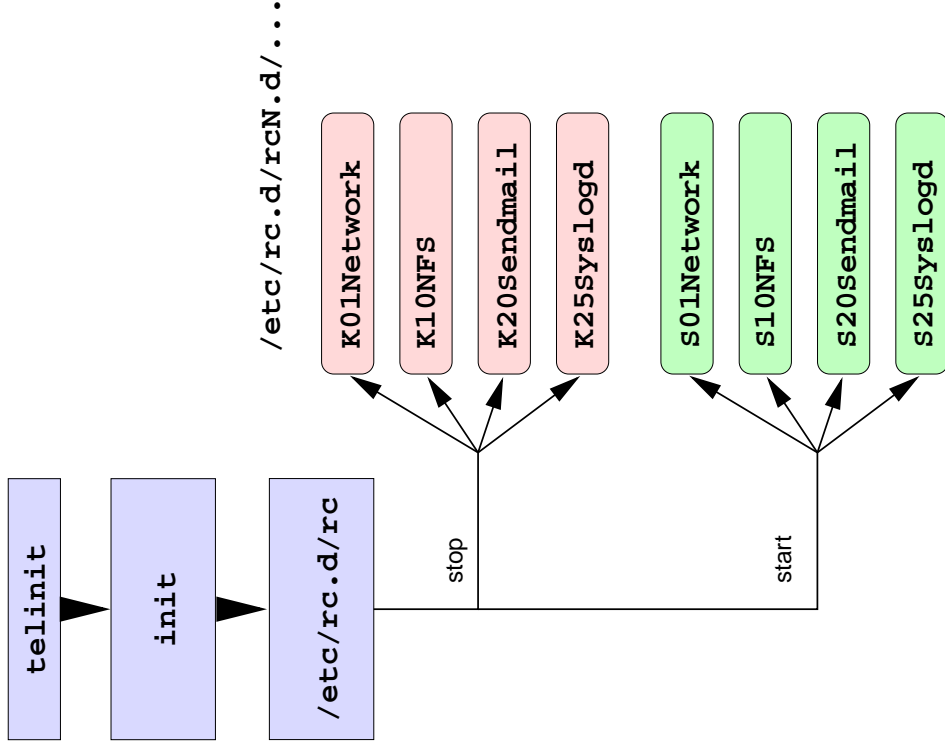
### **Spezielle Runlevel:**

- 0** Halt/Powerdown (nur in diesem Zustand kann der Rechner ohne möglichen Datenverlust ausgeschaltet werden)
- 6** Reboot

## Auslösen eines Runlevels

- Auslösen des **Default-Runlevels** (aus `/etc/inittab`)
- Übergabe als **Parameter beim Booten** durch die Boot-Konsole (LILO, MILO, SILO, ...)  
`lilo boot: linux 1`
- Expliziter **Aufruf** durch Systemverwalter bei laufendem Betrieb:  
`# telinit 4`
- **Shutdown** durch Befehl oder Hotkeys (Ctrl-Alt-Del)
- Sicheres Herunterfahren durch **powerd** wegen Stromausfall etc.

# Wechsel des Runlevels



- **telinit** schickt dem init-Prozess ein Signal
- init liest **inittab** neu aus, startet **rc**
- im neuen Runlevel werden **nicht benötigte Subsysteme abgeschaltet**
- im neuen Runlevel werden **neue Subsysteme aktiviert**

## Vorgänge beim Wechsel

- `rc` wird aufgerufen (Parameter: neuer (und alter) Runlevel).
- Im neuen Runlevel-Verzeichnis werden **Stoppskripte** aufgerufen.
- Im neuen Runlevel-Verzeichnis werden **Startskripte** aufgerufen.
- Skripte liegen in `/etc/rc.d/rcN.d/...`
- Die Skripte sind symbolische Links nach `/etc/rc.d/init.d/...`, wo **alle** Skripte für **alle** Runlevel gesammelt werden.
- Jedes Skript startet bzw. stoppt genau *ein* Subsystem, je nachdem, ob es mit Parameter `start` oder `stop` aufgerufen wird.

## /etc/printcap (1)

### Die Konfigurationsdatei für den Drucker-Daemon lpd

```
# sd=Spool Directory, mx#=Max Size, lp=Printer Device
# sh=Suppress Headers, rm=Remote Machine, rp=Remote Printer
# if=Input Filter, sf=Suppress Formfeed

lp|Standard Textdrucker:\
    :sd=/var/spool/lpd/lp:\
    :mx#0:\
    :lp=/dev/lp:\
    :sh:

ps|Deskjet mit Postscript-Filter:\
    :sd=/var/spool/lpd/ps:\
    :lp=/dev/lp:\
    :if=/var/spool/lpd/ps/ps2deskjet:\
    :sh:sf:
```

## /etc/printcap (2)

### Die Konfigurationsdatei für den Drucker-Daemon lpd

```
ps-pizza | Remote Postscript-Laserdrucker an Pizza:\
      :sd=/var/spool/lpd/ps-pizza:\
      :mx#0:\
      :rm=pizza:\
      :rp=ps:\

fax | Faxmodem (lpr -Jphone_number files...):\
      :sd=/var/spool/lpd/fax:\
      :lp=/dev/null:\
      :if=/usr/bin/faxlpr:\
      :sh:\

# EOF /etc/printcap. See printcap(5) for more information.
```

`/etc/hosts.lpd`

## Die Exportdatei für den Drucker-Daemon lpd

```
### This is /etc/hosts.lpd
### IP-Addresses of hosts who have print access on thi

pizza.unix-ag.uni-kl.de
lasagne.unix-ag.uni-kl.de
calzone.unix-ag.uni-kl.de
aix5.rhrk.uni-kl.de
```

## mknod

# Device-Dateien neu anlegen

Erzeugen einer neuen Gerätedatei, mit der Disketten auf 82 Tracks á 21 Sektoren formatiert werden können:

```
$ mknod /dev/fd0H1722 b 2 60  
$ chmod 666 /dev/fd0H1722 $ ls -l /dev/fd0H1722  
brw-rw-rw- 1 root root 2, 60 Apr 16 01:43 /dev/fd0H1722
```

## format

# Datenträgerstruktur (auf Disketten) anlegen und überprüfen (formatieren)

Formatieren ist in der Regel nur einmalig bei Disketten oder unformatierten Medien notwendig und dient der Erkennung (und Ersetzung) von physikalisch fehlerhaften Bereichen auf der Medienoberfläche. Im Falle von Festplatten kann dies auch durch das BIOS erledigt werden. Moderne Platten müssen vor der Einbindung ins System nicht vom Benutzer formatiert werden.

```
$ fdformat /dev/fd0H1440
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... 1 ... 80 done
Verifying ... 1 ... 80 done
```

# fdisk (1)

## Partitionieren von Medien

```
$ fdisk /dev/hda
Command (m for help): m

Command action
a toggle a bootable flag
b edit bsd disklabel
c toggle the dos compatibility flag
d delete a partition
l list known partition types
m print this menu
n add a new partition
p print the partition table
q quit without saving changes
t change a partition's system id
u change display/entry units
v verify the partition table
w write table to disk and exit
x extra functionality (experts only)
```

## fdisk (2)

### Partitionieren von Medien

```
$ fdisk /dev/hda
```

```
Command (m for help): p
```

```
Disk /dev/hda: 64 heads, 63 sectors, 827 cylinders
```

```
Units = cylinders of 4032 * 512 bytes
```

Device	Boot	Begin	Start	End	Blocks	Id	System
/dev/hda1	1	1	1	21	42304+	6	DOS 16-bit >=32M
/dev/hda2	22	22	22	42	42336	83	Linux native
/dev/hda3	43	43	43	53	22176	82	Linux swap
/dev/hda4	54	54	54	827	1560384	5	Extended
/dev/hda5	54	54	54	354	606784+	83	Linux native
/dev/hda6	355	355	355	385	62464+	83	Linux native
/dev/hda7	386	386	386	416	62464+	83	Linux native
/dev/hda8	417	417	417	827	828544+	83	Linux native

```
Command (m for help):
```

## mkfs

### Anlegen von Dateisystemen

```
$ mkfs -t ext2 /dev/fd0
mke2fs 1.04, 16-May-96 for EXT2 FS 0.5b, 95/08/09
360 inodes, 1440 blocks
72 blocks (5.00%First data block=1
Block size=1024 (log=0)
Fragment size=1024 (log=0)
1 block group
8192 blocks per group, 8192 fragments per group
360 inodes per group
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
```

## fscck

# Dateisysteme überprüfen und reparieren

```
$ fscck -t ext2 /dev/fd0
Parallelizing fscck version 1.04 (16-May-96)
e2fscck 1.04, 16-May-96 for EXT2 FS 0.5b, 95/08/09
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
/dev/fd0: 91/360 files (1.1
```

## rc.sysinit:

```
$ fscck -R -A -V -a
-R  exclude /          -A  all filesystems in /etc/fstab
-V  be verbose         -a  automatically repair
```

## /etc/fstab

### Die Dateisystem-Konfigurationsdatei

#	<device>	<mountpoint>	<type>	<options>	<dump>	<fsck>
	/dev/hda2	/	ext2	defaults	1	1
	/dev/hda3	/usr	ext2	defaults	1	2
	/dev/hda5	/home	ext2	defaults	1	3
	pizza:/www	/mnt/pizza/www	nfs	rw,hard,intr,bg	0	0
	/dev/hda3	none	swap	sw		
	none	/proc	proc	defaults		
	/dev/fd0	/floppy	msdos	noauto	0	0
	/dev/hdc4	/mnt/cdrom	iso9660	noauto,ro,user,nosuid,nodev		

## mount

### Einbinden von Dateisystemen

```
$ mount -t ext2 /dev/hda2 /usr
$ mount -t iso9660 -o ro,nosuid /dev/hdc /mnt/cdrom
$ mount -t msdos /dev/fd0 /floppy
$ mount /floppy
```

```
$ umount /floppy
```

# Einbinden von neuen Datenträgern

## Schritte

1. `$ fdisk /dev/hdb`
2. `$ mkfs -t ext2 /dev/hdb1`
3. `$ fsck -t ext2 /dev/hdb1`
4. `$ mount -t ext2 /dev/hdb1 /tmp/big`

## /etc/exports

### NFS: Exportieren von Dateisystemen

```
/mnt/cdrom    pizza(ro,nosuid)    sushi(ro,nosuid)
/usr          lasagne(ro)         sushi(ro)
/tmp         lasagne(rw,nosuid,nodev)
```

```
$ exportfs -a
$ showmount -e sushi
Export list for sushi:
/usr          bratwurst.unix-ag.uni-kl.de
/mnt/cdrom    (everyone)
/export/home  pizza.unix-ag.uni-kl.de, calzone.unix-ag.uni-kl.de
/export/projekte pizza.unix-ag.uni-kl.de, calzone.unix-ag.uni-kl.de
$ mount sushi:/mnt/cdrom /mnt/sushi-cd
```

## Netzwerk-Hardware

Unix hat schon von seiner Grundstruktur her die Netzwerkfähigkeit fast aller Programme und Dienste im Kernel „eingebaut“. Um ein tatsächliches Netzwerk zu realisieren, müssen zunächst die Kernel-Module für angeschlossene Peripherie wie Netzwerkkarte oder Modem aktiviert werden (geschieht meist beim ersten Zugriff automatisch durch den `kernel`). Die meisten Module unter Linux unterstützen `autoprobe`, jedoch kann der Administrator auch die Hardware-Eigenschaften der Komponenten explizit angeben, wenn diese bekannt sind.

```
### /etc/conf.modules
alias eth0 wd
options wd io=0x360 irq=5
```

# ifconfig

## Netzwerkinterface konfigurieren

Beim Laden der Kernel-Module werden Meldungen ausgegeben.

```
Nov 7 00:14:58 kernel: loading device 'eth0' ...
Nov 7 00:14:58 kernel: wd.c:v1.10 9/23/94 Donald Becker (becker@cesdis.gsfc.nasa.gov)
Nov 7 00:14:58 kernel: eth0: WD80x3 at 0x300, 00 00 C0 68 FB 29 WD8003, IRQ 5,
shared memory at 0xca000-0xcbfff.
```

```
$ ifconfig eth0 131.246.89.4 netmask 255.255.255.0 broadcast
131.246.89.255
```

```
eth0 Link encap:10Mbps Ethernet HWaddr 00:00:C0:68:FB:29
inet addr:131.246.89.4 Bcast:131.246.89.255 Mask:255.255.255.
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0
TX packets:0 errors:0 dropped:0 overruns:0
Interrupt:5 Base address:0x310 Memory:ca000-cc000
```

```
route
```

## Netzwerk-Routen setzen

```
$ route add default 131.246.89.254 131.246.89.254 eth0
```

```
Kernel routing table
```

Destination	Gateway	Genmask	Flags	MSS	Window	Use	Interface
131.246.89.0	*	255.255.255.0	U	1500	0	1428	eth0
127.0.0.0	*	255.0.0.0	U	3584	0	9	lo
default	131.246.89.254	*	UG	1500	0	11200	eth0

## network

### INIT-Scripte zur Netzwerkkonfiguration

```
$ /etc/rc.d/init.d/network start  
$ /etc/rc.d/init.d/network stop
```

Das `network`-initscript wird in den Multiuser-Runleveln automatisch gestartet, im Singleuser-Modus und bei Reboot/Halt werden die Netzwerkinterfaces entsprechend heruntergefahren.

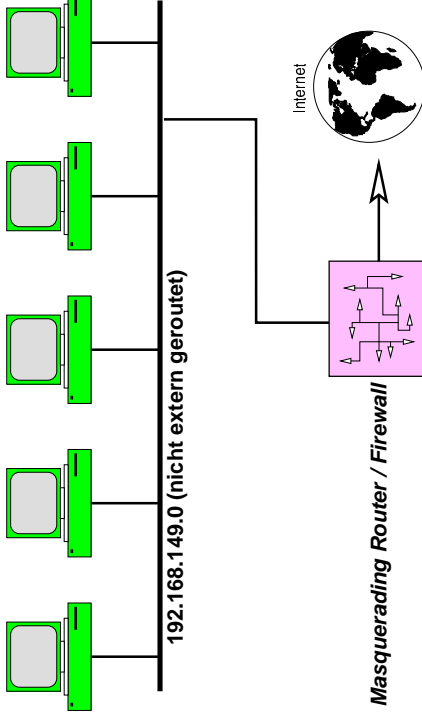
## `/etc/inetd.conf` Konfiguration der Netzdienste

```
# inetd.conf This file describes the services that will be available
# through the INETD TCP/IP super server. To re-configure
# the running INETD process, edit this file, then send the
# INETD process a SIGHUP signal.
#
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
#
finger stream tcp nowait root /usr/sbin/in.fingerd fingerd
ftp stream tcp nowait root /usr/sbin/tcpd in.ftpd -l -a
telnet stream tcp nowait root /usr/sbin/tcpd in.telnetd
shell stream tcp nowait root /usr/sbin/tcpd in.rshd
talk dgram udp wait root /usr/sbin/tcpd in.talkd
```

Mit Hilfe der Konfigurationsdateien für den TCP-Wrapper `tcpd`,  
`/etc/hosts.allow` und `/etc/hosts.deny`, kann der Zugriff  
auf Netzdienste IP- und sogar User-spezifisch kontrolliert werden.

👉 `man 5 hosts_access`

# IP Forward und Masquerading Kernel 2.0.x

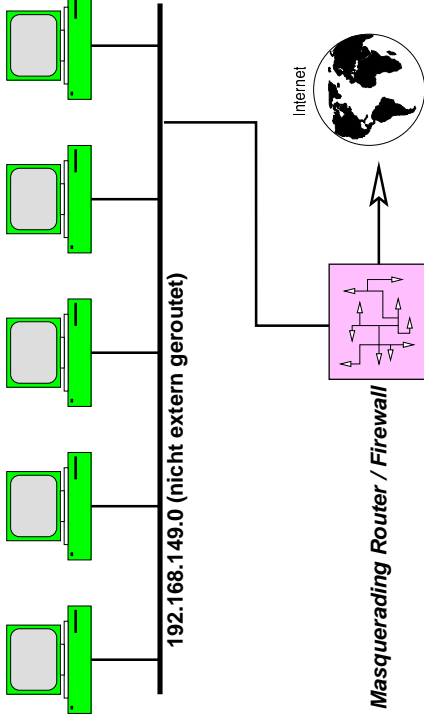


```
# Enable IP forwarding feature
echo "1" > /proc/sys/net/ipv4/ip_forward

# Set up forwarding/Masquerading rules
/sbin/ipfwadm -F -p deny
/sbin/ipfwadm -F -m -a accept -S 192.168.149.0/255.255.0 -D 0/0

# Increase timeout (auto-disconnect) for TCP sessions to 1h
/sbin/ipfwadm -M -s 3600 0 0
```

# IP Forward und Masquerading Kernel 2.2.x



```
# Enable IP forwarding feature
echo "1" > /proc/sys/net/ipv4/ip_forward

# Set up forwarding/Masquerading rules
/sbin/ipchains -P forward DENY
/sbin/ipchains -A forward -s 192.168.149.0/24 -d 0/0 -j MASQ

# Increase timeout (auto-disconnect) for TCP sessions to 1h
/sbin/ipchains -S 3600 0 0
```

## Virtuelle Konsolen

Unter Linux gibt es in der Standard-Konfiguration im Textmodus mehrere „virtuelle Terminals“, an denen sich der Benutzer an der Hauptkonsole des Rechners anmelden kann.

In der Regel ist die Belegung wie folgt:

Alt-F1	Console 1, System-Statusmeldungen und Login
Alt-F2	Console 2, Login
Alt-F3	Console 3, Login
Alt-F4	Console 4, Login
Alt-F5	Console 5, Login
Alt-F6	Console 6, Login
Alt-F7	kein Login, evtl. Zurückschalten zu X-Windows

# Literatur zur Unix-Systemadministration



**Unix Systemadministrati-  
on**  
Eleen Frisch,  
O'Reilly & Associates



**Linux in a Nutshell**  
Daniel Gilly,  
O'Reilly & Associates



**Linux Anwenderhandbuch**  
Sebastian Hetze et al.,  
Lunetix  
<http://www.lunetix.de/>

# Übungsaufgabe

# Übungsaufgabe

# Übungsaufgabe

# init 0