

Einführung in die  
**Linux-Systemverwaltung**

Unterlagen für einen Kurs von 20-40 Stunden

Version vom 2005-10-14

(C) 2005 Christian Mair

chris@1006.org  
<http://www.1006.org>

# Programm

- Begriffe: Unix / Linux, Freie Software / Open Source Software
- Linux Distributionen; Installation von Linux (CentOS)
- erste Schritte mit dem Gnome Desktop und der Unix Shell
- grundlegende Shell Befehle
- Gerätedateien, Laufwerke, Partitionen, Dateisysteme, der Unix Dateibaum
- Benutzer- und Gruppenverwaltung
- Zugriffsrechte im Dateisystem
- Standardeditoren im Textmodus (vi) und an der graphischen Benutzeroberfläche (gedit)
- TCP/IP Netzwerkkonfiguration
- Dienste verwalten; Beispiele: SSH, Webserver, SMB Server (Samba)
- RPM Paketverwaltung und Updates mit yum
- Archive mit tar
- Literaturliste
- Urheberrechtshinweis

## Unix / Linux

- 🕒 1971. Bell Labs (Teil von AT&T) präsentiert **Unix** (Dennis Ritchie und Ken Thompson)
  - läuft auf unterschiedlicher Hardware (Programmiersprache C)
  - “multitasking”- und “multiuser”-fähig
- 🕒 1978. University of California at Berkeley
  - die Universität erhält eine Lizenz um eigene Unix Version zu verteilen: die Berkeley Software Distribution (**BSD**); Mac OS X ist ein moderner Enkel dieses BSD!
  - Verschiedene Hersteller bieten daraufhin Unix Varianten an:  
SunOS/Solaris, HP-UX, AIX, IRIX, Digital OSF, SCO, Xenix, NeXTSTEP
- 🕒 1984. Projekt **GNU** und die Geburtstunde der Freien Software
  - Richard Stallmann beginnt mit der Entwicklung einer freien Unix Variante (GNU)
- 🕒 1991. **Linux**
  - Linus Torvalds schreibt aus Spaß einen Unix-kompatiblen Betriebssystem-Kern (Linux), der sich in GNU einfügt: GNU + Linux ergibt GNU/Linux, meist einfach Linux genannt
  - seit etwa 2000 setzt sich Linux zunehmend durch und tritt das Erbe von Unix an

# Freie Software / Open Source Software

**Freie Software** ist eine ältere (1984) und **Open Source Software** eine jüngere (1997) Definition von Software die sich in 2 Punkten wesentlich von proprietärer Software unterscheidet:

- ☞ die **Lizenz** räumt dem Lizenznehmer wesentlich mehr Rechte ein
  - die Software darf beliebig eingesetzt werden
  - der Source Code darf eingesehen und verändert werden
  - der Source Code und die Änderungen dürfen weitergegeben werden
- ☞ das **Entwicklungsmodell** ist grundsätzlich anders
  - meist mehrere beteiligte Parteien; stark dezentrale Entwicklung
  - häufige Releases; schnelle Entwicklung; unabhängige Qualitätskontrolle
  - starke Konkurrenz; parallele Ansätze; der beste Ansatz setzt sich durch
  - Code Recycling; auf bestehende Projekte aufsetzen, anstatt das Rad neu zu erfinden
  - auch kleine und mittlere Unternehmen können komplexe Projekte angehen

## Linux Distributionen

- ☞ Ein Linux-System besteht aus tausenden einzelnen Software Paketen
  - Wer wählt die geeigneten Pakete aus und testet sie?
  - Wer zertifiziert Hardware und proprietäre Software?
  - Wer sorgt für Updates, die genau auf das installierte System abgestimmt sind?
  - Wer liefert Support?
- ☞ **Distributoren!**
  - Red Hat (RHEL, Fedora Core)
  - Novell (SLES, SUSE Linux)
  - Debian (Debian GNU/Linux)
  - Canonical (Ubuntu Linux)
  - Mandriva (Mandriva Linux)
  - Klaus Knopper (Knoppix)

# Installation von CentOS 4

- ☞ “Enterprise” Linux Distributionen, wie **Red Hat Enterprise Linux** (RHEL) empfehlen sich für den Produktionseinsatz:
  - offizieller Support
  - längere Lebensdauer, Updates sind 5-7 Jahre garantiert
  - größere Releasezyklen (typischerweise 1.5 Jahre)
  - Zertifizierung durch Hardware Hersteller (wie etwa HP oder Dell) und Hersteller proprietärer Software (wie etwa Oracle oder DB2)
- ☞ Die Installationsmedien von RHEL dürfen nicht frei kopiert werden, daher verwenden wir **CentOS**; das CentOS Projekt übersetzt einfach den (freien) RHEL Sourcecode unabhängig von Red Hat und stellt CDs zur Verfügung - ideal zum Ausprobieren und für alle, die nicht auf offiziellen Support angewiesen sind.
  - Red Hat Enterprise Linux:  
<http://www.redhat.com/software/rhel/>
  - Projekt CentOS:  
<http://www.centos.org/>

## GUI und Unix Shell

- ☞ Im GUI-Betrieb (graphical user interface) benutzt Linux das **X Window System** als graphisches Subsystem und **Gnome** (oder **KDE**) als graphische Benutzeroberflächen.
- ☞ Linux arbeitet mit mehreren “Consolen”: Umschaltung erfolgt mit **CTRL-ALT-Fx**. Normalerweise ist **F1 - F6** den Textconsolen zugewiesen, **F7** dem X Window System.
- ☞ Unter Unix GUIs ist es traditionell üblich Text mit der linken Maustaste zu kopieren und mit der mittleren Maustaste einzufügen. Moderne Anwendungen benutzen auch das bekannte **CTRL-C** und **CTRL-V**, mit einer Ausnahme: im Gnome-Terminal, den wir oft benutzen um mit der Shell zu arbeiten, lauten die Tastenkombinationen **SHIFT-CTRL-C** und **SHIFT-CTRL-V**. Dies, weil **CTRL-C** in der Shell bereits eine andere Bedeutung hat, nämlich ein laufendes Programm abzubrechen.
- ☞ RHEL bietet unter Gnome zahlreiche Werkzeuge um die grundlegende Systemkonfiguration per Maus zu erledigen, trotzdem ist die Unix Shell nach wie vor das mächtigste Werkzeug zur Systemverwaltung. Eine Shell erreicht man entweder an der Textconsole, als Anwendung “Gnome-Terminal” in der GUI oder indem man sich aus der Ferne über den SSH (secure shell) Dienst einloggt (`ssh username@hostname`).
- ☞ Dank der Netzwerktransparenz des X Window Systems ist es möglich graphische Anwendung auf dem entfernten Rechner zu starten während die graphische Darstellung auf dem lokalen Rechner erfolgt.

# Shell Grundlagen I

## ☞ Befehle zu **Systeminformationen**:

```
free          # freier Hauptspeicher (RAM)
df            # freier Plattenplatz
w            # Liste der eingeloggten Benutzer
uptime       # Uptime und Systemlast
```

## ☞ Goldene Regel Nr. 1 für angehende Systemverwalter: niemals als root arbeiten, wenn es nicht unbedingt nötig ist! Immer mit einem **nicht privilegiertes Benutzerkonto** arbeiten.

```
useradd name  # Benutzer name zufügen...
passwd name   # ...und Password setzen
su -l name    # neue Shell als Benutzer name öffnen
              # -l (Buchstabe L) ist wichtig um die Umgebungsvariablen
              # des neuen Benutzers zu übernehmen!
exit         # neue Shell wieder verlassen
```

# Shell Grundlagen 2

## ☞ Grundlegende Befehle zur **Dateiverwaltung**:

```
ls           # Verzeichnis Liste anzeigen
ls -l       # idem, ausführlich
ls -al      # idem, auch "versteckte" Dateien
cd verz    # ins Verzeichnis verz wechseln
cd /home/chris/Desktop # absolute Pfadangabe
cd Desktop  # relative Pfadangabe
pwd         # in welchem Verzeichnis befinde ich mich gerade?
cat files  # files ausgeben (anzeigen)
cp file1 file2 # file1 nach file2 kopieren (-r für rekursiv)
mv file1 file2 # file1 nach file2 umbenennen od. verschieben
rm files    # files löschen (-r für rekursiv, Vorsicht!)
touch files # leere files anlegen oder Zeitstempel updaten
grep str file # str in file suchen (-i case insensitive, -v invertiert)
wc files    # Zeilen, Worte und Buchstabe in files zählen
```

## Shell Grundlagen 3

### 🔊 Dateien **suchen**:

```
find wo was # im Verz. wo nach Kriterium was suchen
find /etc -newer file # Dateien in /etc neuer als file finden
find /home -name "*pdf" # Dateien in /home mit Endung pdf finden,
# die "*" sind wichtig!
```

### 🔊 Dateien **schnell suchen**

```
locate str # Dateien, die str enthalten finden
updatedb # Index jetzt erneuern (als root)
```

### 🔊 symbolische **Verknüpfungen**

```
ln -s ziel quelle # Verknüpfung von ziel nach quelle
```

### 🔊 **Dateityp** erkennen

```
file file # Dateityp von file erkennen
```

## Shell Grundlagen 4

### 🔊 Unix **Prozess Verwaltung**

```
ps aux # alle Prozesse anzeigen
top # alle Prozesse interaktiv anzeigen (mit q aussteigen)
kill pid # Prozess mit PID pid abbrechen
kill -9 pid # idem, aber ohne Reaktionsmöglichkeit (d.h. crash!)
```

### 🔊 **Verzeichnisse**

```
. # aktuelles Verzeichnis
.. # übergeordnetes Verzeichnis
~ # Heimatverzeichnis
mkdir verz # Verzeichnis verz erstellen
rmdir verz # Verzeichnis verz löschen
```

### 🔊 **Dateien betrachten**

```
tail file # Ende von file anzeigen (-f für live update)
less file # Datei file interaktiv betrachten (q um auszusteigen)
```

# Shell Grundlagen 5

## Shell Ein- und Ausgabe **Umleitung**

```
prog > file          # Ausgabe von prog in file umleiten
prog < file          # file in die Eingabe von prog leiten
prog1 | prog2       # Ausgabe von prog1 in die Eingabe von prog2 leiten
prog 2> file        # Fehlerausgabe von prog in file umleiten
```

## Spezielle Zeichen in der Shell

- das Leerzeichen trennt Befehle und Argumente voneinander. Um etwa ein Verzeichnis mit einem Leerzeichen im Namen anzulegen muss ich so vorgehen:

```
mkdir "My Documents"
```

- der Stern ist eines der Zeichen, die zum "file globbing" dienen - die Shell ersetzt ihn mit einer Liste von "passenden" Dateinamen. Z.B.:

```
echo /home/*
```

## Hilfe ;-)

```
man befehl          # zeigt die Handbuchseite zu befehl an
```

# Gerätedateien und Laufwerke

- Unter Unix werden Geräte ("devices") wie normale Dateien angesprochen. Diese sogenannten **Gerätedateien** ("device files") befinden sich im Verzeichnis `/dev`.

- Die **IDE** Laufwerke werden z.B. über folgende Gerätedateien angesprochen:

```
/dev/hda          # erster Master
/dev/hdb          # erster Slave
/dev/hdc          # zweiter Master
/dev/hdd          # zweiter Slave
```

- Die Partitionen auf den Laufwerken sind nummeriert, `/dev/hda1` ist also die erste Partition auf dem ersten IDE Master, usw...
- SCSI** Platten (und auch **USB Speichermedien!**) heißen `/dev/sda`, `/dev/sdb`, usw... Disketten heißen `/dev/fd0`, `/dev/fd0`, usw...

- Der Inhalt von `/dev` wird dynamisch erzeugt. D.h. die Gerätedateien werden entsprechend der vorhandenen Geräte eingeblendet. Dies erfolgt z.B. bei USB Speichermedien auch im laufenden Betrieb (Versuch!).

- Tipp: mit dem Befehl `dmesg` lassen sich die letzten Logmeldungen des Kernels anzeigen. Das hilft beim Herausfinden ob ein neu angeschlossenes Gerät erkannt wurde.

# Partitionen und Dateisysteme

☞ Laufwerke können mit dem Befehl `fdisk` **partitioniert** werden, einem einfachen interaktiven Befehlszeilen-Werkzeug. `fdisk -l` zeigt jederzeit die aktuelle Partitionstabelle an.

! Achtung: nach einer Neupartitionierung des Laufwerks auf dem sich die Systempartition (/) befindet ist ein Neustart erforderlich (Befehl `reboot`). !

☞ Eine neue Partition muss **formatiert** werden. Dabei stehen unter Linux mehrere Dateisysteme zur Verfügung:

`ext2` # klassisches Linux Dateisystem

`ext3` # jüngere Version von ext2, mit "Journaling"

und viele andere, u.a. auch `vfat`, das zu Windows' FAT32 kompatibel ist.

Offiziell nicht unterstützt wird hingegen NTFS.

- Eine Formatierung mit `ext3` erfolgt durch:

```
mkfs.ext3 devicefile
```

```
z.B.: mkfs.ext3 /dev/hda3
```

## Unix Dateibaum I

☞ Formatierte Partitionen können in den Unix Dateibaum **eingebunden** werden, dessen Wurzel bei / liegt. Ein Beispiel:

```
Partition: eingebunden unter ("mount point"):
```

```
/dev/sda1 /
```

```
/dev/sda2 /home
```

☞ NB: es gibt **nur eine einzelne Hierarchie** - alle zugänglichen Dateisysteme sind unter / eingebunden - egal ob es lokale Dateisysteme sind oder Fileserver im Netzwerk.

☞ Dies erlaubt es u.a. folgendes Problem zu lösen: der Benutzer Tux möchte mehr Platz als auf `/home/tux` verfügbar ist, also kriegt Tux eine neue Partition. Seine Dateien werden auf die neue Partition kopiert und diese unter `/home/tux` eingebunden. Änderungen im Pfad sind somit nicht nötig.

☞ Auf einem Server werden die Verzeichnisse `/home`, `/usr`, `/var` und `/tmp`, in denen nicht privilegiert Benutzer Schreibzugriff haben häufig auf andere Partitionen als die Systempartition (/) angelegt, so dass ein Benutzer diese nicht versehentlich oder auch absichtlich füllen kann.

## Unix Dateibaum 2

☛ Das **Einbinden** von Dateisystemen erfolgt mit:

```
mount devicefile mountpoint
```

```
z.B.: mount /dev/hda1 /home2
```

- Das Verzeichnis unter dem eingebunden wird ("mount point") muss schon existieren und sollte leer sein, da dort schon vorhandene Dateien sonst "verdeckt" werden.

☛ `mount` ohne Parameter zeigt alle zur Zeit eingebunden Dateisysteme an.

- Die Option `-t` erlaubt es einen Dateisystemtyp explizit anzugeben (was normalerweise nicht nötig ist):

```
mount -t vfat /dev/fd0 /mnt/floppy
```

☛ Die Datei `/etc/fstab` enthält eine Liste von Dateisystemen mit vordefinierten "mount points", die beim Systemstart eingebunden werden, sowie zahlreiche Optionen.

## Unix Dateibaum 3

☛ Ein typischer Unix Dateibaum sieht folgendermaßen aus:

```
/bin, /sbin # grundlegende Systembefehle
/boot       # zum Systemstart nötig ("Boot")
/dev        # Gerätedateien
/etc        # systemweite Konfigurationsdateien
/home       # Dateien der Benutzer
/lib, /usr  # Systembibliotheken, Anwendungen
/mnt        # Einhängpunkte
/var        # veränderliche Dateien, z.B. Log Dateien in /var/log
/proc, /sys # virtuelle Dateien, die das laufende System abbilden
/tmp        # zeitweilige Dateien
```

# Benutzer und Gruppen

- ☞ Mindestens ein Benutzeraccount ist auf jedem Unix System vorhanden: **root**, das Systemverwalteraccount. root wird auch "Superuser" genannt. Neben den GUI Werkzeugen der jeweiligen Linux Distribution können u. a. folgende Befehle für die Verwaltung lokaler Benutzer verwendet werden:

```
useradd user; userdel user    # Benutzer hinzufügen; löschen
passwd user                   # Passwort setzen
```

- ☞ Jeder Benutzer ist einer oder mehreren Gruppen zugeordnet, wovon eine die primäre Gruppe ist. Folgende Befehle verwenden wir zum Arbeiten mit Gruppen:

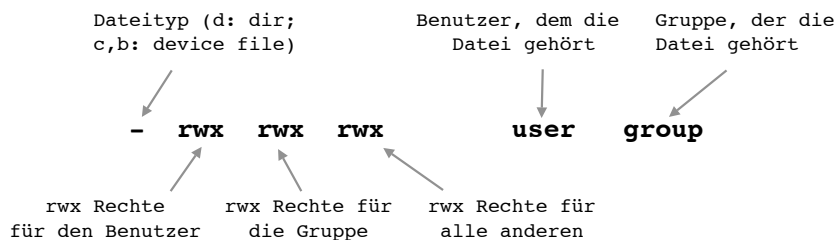
```
groupadd group; groupdel group # Gruppe hinzufügen; löschen
usermod -g group user         # primäre Gruppe setzen
usermod -G group1, group2, ... user # weitere Gruppen setzen
id user                       # Infos über Benutzer
```

- ☞ Jede Datei gehört genau einem Benutzer und einer Gruppe! root kann den Benutzer und die Gruppe mit folgendem Befehl ändern:

```
chown user:group file        # Dateieigentümer setzen
                              # (-R für rekursiv)
```

## Zugriffsrechte I

**Zugriffsrechte** werden pro Datei wie folgt definiert:



rwx steht dabei für die 3 Rechte:

```
r (read):      Leserecht
w (write):     Schreibrecht
x (execute):   Ausführrecht bei Dateien, bzw. Zugriffsrecht bei Verzeichnissen
```

# Zugriffsrechte 2

Mit **chmod** können die Dateirechte verändert werden:

```
chmod {wer}{+-}{was} file
```

{wer} steht für **u** (user), **g** (group) oder **o** (others)

{+-} steht für **+** (Recht geben) oder **-** (nehmen)

{was} steht für **r**, **w** oder **x**

Mit der Option **-R** erfolgt die Rechteänderung rekursiv auf alle Unterverzeichnisse.

Beispiele:

```
chmod ugo+rx /home/tux # jeder darf die Dateien in /home/tux auflisten
```

```
chmod g+rxw /home/chris/Desktop  
# die Gruppe erhält Zugriff auf Chris' Desktop
```

# Texteditoren/vi

- ☞ Texteditoren sind unter Unix für Systemverwalter sehr wichtig, da viele Verwaltungsaufgaben über die Shell und mit Konfigurations- und Skriptdateien erledigt werden. Für Linux sind sowohl sehr bequeme **Texteditoren** mit GUI (etwa **gedit** unter Gnome und **kwrite** unter KDE) als auch sehr mächtige Text Editoren (z.B. **emacs**) verfügbar.
- ☞ Der Standardeditor in der Unix Befehlszeile ist aber nach wie vor **vi**, einer der ältesten Editoren überhaupt. Da dies der einzige Editor ist, der garantiert immer verfügbar ist, sollen hier die wichtigsten **vi** Befehle aufgelistet werden. Es gibt verschiedene erweiterte, aber zum **Ur-vi** kompatible Versionen - unter Linux z.B. **vim**.
- ☞ Es gibt 2 Modi unter **vi**: der Editor startet im *Befehlsmodus*; mit **i** wechselt man in den *Eingabemodus*; mit **ESC** zurück in den Befehlsmodus. Nur im Eingabemodus darf man drauf los tippen! Im Befehlsmodus wird jede Eingabe als Befehl interpretiert, Beispiele:

<b>i</b>	# > Eingabemodus	<b>x</b>	# Zeichen löschen
<b>A</b>	# > Eingabemodus Zeilenende	<b>dd</b>	# Zeile löschen
<b>:w</b>	# abspeichern	<b>d\$</b>	# bis zum Ende der Zeile löschen
<b>:q</b>	# aussteigen	<b>yy</b>	# Zeile kopieren
<b>:wq</b>	# abspeichern und aussteigen	<b>P</b>	# Gelöschtes od. Kopiertes einfügen
<b>:q!</b>	# Änderungen verwerfen	<b>/str</b>	# suchen nach str
<b>:r!</b>	# Befehlsausgabe einlesen	<b>:%s/str1/str2/g</b>	# ersetzen

# TCP/IP Netzwerke I

Die meisten Linux Distributionen bieten Werkzeuge zur **Konfiguration der Netzwerkparameter** in der GUI oder im Textmodus.

- bei Red Hat / CentOS gibt es den Befehl `system-config-network`, der je nachdem, ob man über eine graphische Session verfügt oder nicht, ein GUI-Programm oder ein interaktives Textprogramm aufruft

Unabhängig von der spezifischen Distribution werden die resultierenden Konfigurationsdateien in `/etc` abgelegt, d.h. diese können auch einfach von Hand editiert werden. Die relevanten Dateien und Verzeichnisse sind:

```
/etc/resolv.conf           # DNS (Server für Namensauflösung)
/etc/sysconfig/network-scripts/ # TCP/IP Einstellung pro Schnittstelle
(/etc/network/ unter Debian)
```

Nach einer Umstellung können alle Netzwerkschnittstellen mit folgendem Befehl neu gestartet werden:

```
/etc/init.d/network restart
(/etc/init.d/networking restart unter Debian)
```

# TCP/IP Netzwerke 2

Die Netzwerkparameter können auch **manuell** gesetzt werden mit den Befehlen `ifconfig` und `route`:

- beide befinden sich im Verzeichnis `/sbin`, das nur root in seinem Suchpfad hat - falls dieser nicht gesetzt wurde, also mit `/sbin/ifconfig` und `/sbin/route/` absolut benennen
- beachte: derart manuell getätigte Einstellungen überleben einen Neustart nicht!

Beispiele:

```
ifconfig                   # IP Adressen anzeigen
ifconfig eth0 192.168.1.2  # 1. Ethernet Schnittstelle (eth0)
                           # auf 192.168.1.2 setzen (LAN)

route -n                   # Route anzeigen
route add default gw 192.168.1.1 # Default Route auf 192.168.1.1
                           # setzen (Router im LAN)
```

Eine Netzwerkschnittstelle muss auf jedem TCP/IP kompatiblen System vorhanden sein, nämlich das "Loopback" mit der reservierten Adresse 127.0.0.1 ("localhost"). Diese virtuelle Schnittstelle heißt unter Linux `lo`.

# TCP/IP Netzwerke 3

☞ Nützliche Befehle zur **Netzwerkanalyse** (auch unter Windows bekannt, Microsoft hat TCP/IP von Unix übernommen):

```
ping          # "ping" Paket an entfernten Rechner schicken
ping -c 3     # wie unter Windows: stop nach 3 Paketen
telnet        # Verbindung zu entferntem Rechner aufbauen
traceroute    # Schritte zu entferntem Rechner ermitteln (Win: tracert)
nslookup      # Zuordnung IP Adresse / Internet Domain Name ermitteln
arp           # Zuordnung IP Adressen zu MAC Adressen ermitteln
```

☞ Mächtiger Befehl um ganze Netzwerke nach Rechnern ("hosts") und Diensten ("ports") zu scannen (Vorsicht!) - z.B. zur Firewall Analyse und Fehlersuche:

```
nmap          # Portscanner
nmapfe        # interaktive GUI Oberfläche zu nmap
```

☞ Befehl um TCP/IP Verkehr mitzuhören ("sniffen") zur Fehlersuche:

```
tcpdump       # Netzwerkverkehr mithören
```

## Dienste und run levels

☞ Linux kennt verschiedene Zustände des Systems ("run levels"). Die Definition der run levels und der Werkzeuge um damit umzugehen hängt im Detail von der Distribution ab. Typischerweise sind folgende run levels definiert:

- 0 shut down (Rechner ausschalten)
- 1 single user mode (Wartung, kein Netzwerk)
- 3 multi user mode (Server, kein X Window System, Netzwerk aktiv)
- 5 multi user GUI mode (Client, X Window System läuft, Netzwerk aktiv)
- 6 reboot (Rechner neustarten)

☞ Der manuelle Wechsel des Runlevels erfolgt mit `init runlevel`. Der Default run level (3 oder 5) wird in der Datei `/etc/inittab` festgehalten.

☞ Beim Wechsel des run levels startet bzw. stoppt das System entsprechende **Dienste**, auch **Services** oder **Dämonen** genannt. Welche das sein sollen kann mit dem Befehl `chkconfig` definiert werden:

```
chkconfig --list          # Übersicht Dienste anzeigen
chkconfig Dienst on|off   # Dienst autom. starten: ein|aus
```

# SSH Server

- Die **SSH** ("secure shell") erlaubt es eine Shell auf einem entfernten Rechner zu benutzen und Dateien zu transferieren. Aus Sicherheitsgründen hat SSH die "klassischen" Werkzeuge `telnet` und `rlogin` und teilweise auch `ftp` weitestgehend abgelöst.
- Der SSH Dienst (`sshd`) gehört zur Grundinstallation jeder Linux Distribution. Bei den meisten Distributionen läuft der Dienst standardmäßig (ist aber eventuell von einer Firewall geschützt):

```
/etc/ssh/sshd_config          # Konfigurationsdatei
/etc/init.d/sshd start|stop|restart # D. stoppen|starten|neu starten
```
- Der Client kann auf vielfältige Weise benutzt werden:

```
ssh user@rechner          # verbinden (-X für GUI Anwendungen)
sftp user@rechner         # Dateien austauschen (wie ftp)
scp file1 user@rechner:file2 # einzelne Datei übertragen (wie cp)
```

Gnomes Nautilus kann mit der URL `sftp://user@rechner` Dateien per Drag&Drop übertragen.
- Unter Windows haben sich die Clients "Putty" (`ssh`) und "Filezilla" (`sftp`) bewährt.

# Webserver

- Der **Webserver** stellt auf einfache Weise Webseiten ins Netz, die dann über einen Webbrowser betrachtet werden können. Der Webserver Dienst (`httpd`) ist unter Linux der bekannte Server der Apache Software Foundation. Trotz seiner Mächtigkeit ist er sehr einfach zu benutzen:
  - der Dienst wird gestartet mit

```
/etc/init.d/httpd start
```
  - die Dateien, die die Webseite ausmachen werden in das Verzeichnis `/var/www/html/` kopiert und sind dann im Browser über die URL `http://rechner/` zu erreichen
  - die Datei `index.html` ist jeweils die Startdatei in einem Verzeichnis
  - unter CentOS wird standardmäßig eine Begrüßungsseite eingeblendet, falls das Hauptverzeichnis (`/var/www/html/`) keine Datei namens `index.html` enthält - um dies zu unterbinden kann man die Datei `/etc/httpd/conf.d/welcome.conf` entfernen oder umbenennen
- Der Apache Webserver ist erweiterbar z.B. mit der beliebten Skriptsprache **PHP** um dynamische Webseiten zu erstellen (also solche, die auf Abfrage erstellt werden). Auch PHP und viele andere Erweiterungen gehören zum Umfang einer typischen Linux Distribution.

# SMB Server (Samba)

- ☞ Samba ist eine Implementierung für Unix von Microsofts "Windows Networking". Dazu gehört unter anderem ein Server der "**Windows Dateifreigaben** (*shares*) und **Druckdienste**" über Microsofts SMB Protokoll im Netz zur Verfügung stellt. Samba ist also **kein** klassischer Unix Dienst, sondern ist besonders in gemischten Netzen (hauptsächlich Windows Clients mit Linux Servern) im Einsatz.
  - der Samba Dienst wird gestartet mit

```
/etc/init.d/smb start
```
  - die Konfigurationsdateien befinden sich im Verzeichnis `/etc/samba/` - dort ist bereits eine kommentierte Beispieldatei vorhanden, die für eigene Experimente dienen kann (`smb.conf`), z.B. für eine öffentliche Freigabe
  - nützliche Werkzeuge zum Umgang mit dem Samba-Server sind:

```
testparm          # überprüft Konfiguration
smbstatus         # zeigt Status an
```
- ☞ Die Samba Software macht es auch möglich Linux als *Client* zu benutzen. So kann eine Windows Dateifreigabe in den Unix Dateibaum eingebunden werden:

```
mount -t smbfs //server/share mount_point
```

# Paketverwaltung I

- ☞ Eine Linux Installation ist in **Pakete** unterteilt - viele Distributionen verwenden **RPM** (Red Hat Package Manager) um Pakete zu verwalten (Debian verwendet **DPKG**).
- ☞ Der gleichnamige Befehl (`rpm`) bietet folgende Möglichkeiten:

```
rpm --import Key      # Signaturschlüssel des Paketherstellers einlesen
rpm -qa               # alle installierten Pakete anzeigen
rpm -qi Paketname    # Informationen über Paketname anzeigen
rpm -ql Paketname    # mit Paketname installierte Dateien auflisten
rpm -qf file         # in welchem Paket ist file enthalten?
rpm -ivh Paket.rpm   # Paket.rpm installieren (vh für info & progress bar)
rpm -Uvh Paket.rpm   # Paket.rpm updaten oder installieren
rpm -Fvh Paket.rpm   # Paket.rpm auffrischen (=update nur falls installiert)
rpm -e Paketname     # Paketname löschen
```
- ☞ Anstelle von `Paket.rpm` kann auch jeweils eine URL in der Form `ftp://` oder `http://` angegeben werden, was den manuellen Download der RPM Datei überflüssig macht.

## Paketverwaltung 2

☞ RPM überprüft automatisch **Abhängigkeiten** und warnt gegebenenfalls, ohne die Anweisung auszuführen, wenn Probleme erwartet werden. Allerdings ist es sehr mühsam, entsprechend alle RPMs selbst zu finden und bereitzustellen.

☞ **YUM** ist ein bequemes Werkzeug, das Pakete automatisch von einem Server holt, Abhängigkeiten auflöst und alle benötigten Pakete einspielt. Die Benutzung ist denkbar einfach:

```
yum install Paketname      # installiere Paketname
yum search Suchbegriff     # Infos über Pakete ähnlich Suchbegriff
yum update                 # update alle installierten Pakete
```

☞ Informationen über welche Paket-Repositories yum verwendet sind im Verzeichnis /etc/yum.repos.d abgelegt.

## Archive mit tar

☞ Das klassische Archivierungswerkzeug unter Linux ist **tar** (steht für “**t**ape **a**rchiver”). **tar** wird folgendermaßen verwendet:

```
tar cf bu.tar /home      # Create File: archiviert /home in die Datei bu.tar
tar xf bu.tar            # eXtract File: extrahiert den Inhalt von bu.tar in
                        # das aktuelle Verzeichnis
tar tf bu.tar            # lisT File: zeigt die Dateien in bu.tar an
```

☞ Die Optionen **zcf** (bzw. **zxf**, **ztf**) erzeugen (bzw. extrahieren, listen) eine **komprimierte** Datei (Endung meist **.tar.gz** oder **.tgz**). Die Komprimierung kann auch unabhängig von **tar** mit den Befehlen **gzip** (komprimieren) oder **gunzip** (dekomprimieren) erfolgen.

☞ Anstatt eine Datei (wie **bu.tar**) kann auch direkt ein Bandlaufwerk angegeben werden, das Linux erkannt hat (siehe **dmesg**). Das könnte zum Beispiel **/dev/st0** (das erste SCSI Bandlaufwerk) sein.

# Literaturliste

- Linux - Installation, Konfiguration, Anwendung  
von Michael Kofler, erschienen bei Addison Wesley
- Red Hat Enterprise Linux 4 Handbücher (englisch):  
<http://www.redhat.com/docs/manuals/enterprise/>
- CentOS Home Page:  
<http://www.centos.org>
- Grundlagen Computernetze, Skriptum von Prof. Jürgen Plate:  
<http://www.netzmafia.de/skripten/netze/index.html>
- PuTTY (bewährter SSH Client für Windows):  
<http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe>
- Samba Bücher zum Download als PDF:  
<http://www.samba.org/samba/docs/>
- Linux User Group Bozen:  
<http://www.lugbz.org/>
- Kurse mit mir als Referent ;-)  
[http://www.1006.org/training\\_current/](http://www.1006.org/training_current/)

## Urheberrechtshinweis

Das vorliegende Material unterliegt dem Urheberrecht.



Sie dürfen diese Unterlagen unter den Bestimmungen der Creative Commons Lizenz "Namensnennung 2.0 Italien" weitergeben.

Die detaillierte Lizenz ist unter der URL <http://creativecommons.org/licenses/by/2.0/it/legalcode> erhältlich. Es folgt eine Zusammenfassung:

- Sie dürfen:
  - \* den Inhalt vervielfältigen, verbreiten und öffentlich aufführen
  - \* Bearbeitungen anfertigen
  - \* den Inhalt kommerziell nutzen
- Zu den folgenden Bedingungen:
  - \* Namensnennung. Sie müssen den Namen des Autors/Rechtsinhabers nennen.
- Hinweise:
  - \* Im Falle einer Verbreitung müssen Sie anderen die Lizenzbedingungen, unter die dieser Inhalt fällt, mitteilen.
  - \* Jede dieser Bedingungen kann nach schriftlicher Einwilligung des Rechtsinhabers aufgehoben werden.
  - \* Die gesetzlichen Schranken des Urheberrechts bleiben hiervon unberührt.