

LINUX Einführung und Benutzung

Einführung

Linux ist ein Unix-ähnliches Betriebssystem, das von einer großen Entwicklergemeinschaft ständig weiterentwickelt wird. Es steht unter der GNU General Public License und ist damit frei verwendbar. Linux wird bereits von vielen privaten Anwendern, Schulen, Hochschulen, Behörden und Unternehmen benutzt.

Linux zeichnet sich als stabiles Multiuser-Betriebssystem aus und wird bevorzugt im Serverbereich eingesetzt. Die umfangreiche Palette von freier Anwendungssoftware macht Linux auch zunehmend attraktiv für den Desktopbereich.

Für Schule und Ausbildung ist Linux eine Plattform mit besonderen Vorteilen. Linux als UNIX-orientiertes Betriebssystem bietet eine gute Basis zur Schulung von Betriebssystemen, der Administration und Netzwerkverwaltung. Durch die General Public License (GPL) stehen alle Programme auch im Quelltext (Open Source) zur Verfügung und können nicht nur kostenlos verwendet, sondern auch erforscht und weiterentwickelt werden.

Der Pinguin **TUX** (Torvalds Unix) ist das Maskottchen und Symbol für Linux, für die Open Source Community und steht für Stabilität.

„Da Pinguine nicht fliegen können, können sie auch nicht abstürzen?“.



UNIX Geschichte

- 1965 Unix hat seine Wurzeln im Projekt MULTICS
- 1970 Beginn der Unix-Entwicklung von AT&T (Ken Thompson u. Dennis Ritchie)
- 1980 UNIX System V (AT&T), BSD (Berkeley Software Distribution)
und viele weitere UNIX-Derivate (Solaris von SUN, AIX von IBM, HP-UX von HP, Xenix von Microsoft, Sinix von Siemens, ...)
- 1990 zwei UNIX Standards
UNIX System V Release 4 (AT&T)
OSF Standard (Open Software Foundation = IBM, DEC, HP, Siemens)

Unix Merkmale

Portabilität - großteils in der Hochsprache C, nur geringer Teil in Assembler
Stabilität - ausgereiftes Multiuser-Multitasking-Betriebssystem
Sicherheit - Dateisystem, Prozessverwaltung, Userverwaltung, Rechtesystem
vielfältige Netzwerkfähigkeiten
umfangreiche Dienstprogramme und Entwicklungswerkzeuge
Standard an Hochschulen für Ausbildung und Entwicklung

Unix am PC

die ursprünglichen PC's waren für Unix leistungsmäßig unzureichend,
erst mit der CPU 386 begannen die ersten Portierungen.

- 1984 XENIX von Microsoft
- 1987 MINIX von Prof. Tanenbaum für Studienzwecke
- 1991 Linux (Linus Torvald)
- 1992 SOLARIS (Sun)
- 1993 386 BSD, FreeBSD, OpenBSD (Berkeley Software Distribution)

Linux Geschichte

- 1991 Linus Thorvald entwickelt als Student den ersten Linux Betriebssystemkern
- 1993 die Linux Entwickler Gemeinschaft ist bereits auf ca. 100 angewachsen,
der Kernel wird an die GNU-GPL angepasst
- 1994 Linux Version 1.0, Implementierung von X-Window-System als GUI
- 1996 Linux Version 2.0
- 1998 Desktop System KDE, ca. 10.000 Linux Programmierer, ca. 10 Mio. Anwender

Linux Eigenschaften

unter Linux versteht man in erster Linie den Betriebssystem-Kern, zum kompletten System gehören jedoch auch die umfangreichen Betriebssystem-Dienstprogramme und Anwendungssoftware.

Leistungsmerkmale des Linux Kerns (Kernels) -

POSIX-konform – definiert Bedingungen für portierbare Betriebssysteme
Multitasking - mehrere Prozesse (Programme) können gleichzeitig ausgeführt werden
Multiuser - mehrere Benutzer können gleichzeitig arbeiten
Paging - erlaubt die Auslagerung von Speicherinformationen auf die Festplatte
Shared Libraries – Programm-Bibliotheken, die bei Bedarf in den Speicher geladen werden
Shared Memory - gemeinsam genutzte Speicherbereiche für Daten von Programmen
Interprocess Communication (IPC) – Kommunikation von Prozessen untereinander
Symetric Multi Processing (SMP) - mehrere Prozesse laufen auf mehreren Prozessoren
Protected Mode - Speicherschutzmechanismen für Prozesse
Dateisysteme – unterstützt eine Vielzahl von Dateisystemen (ext2,ext3,reiserfs,vfat, ...)
PC-Hardware Unterstützung – aktuelle Linux Treiber mit wenigen Ausnahmen

Sammlung von UNIX-Werkzeugen (Utilities) -

Linux enthält eine umfangreiche Sammlung an UNIX-Werkzeugen, die unter der GNU-GPL stehen. Diese GNU-Werkzeuge sind bereits zum systemübergreifenden Standard geworden.

Distributoren bieten komplette Sammlungen an, ergänzt mit eigenen Installationstools, Handbüchern und Support.

Bekannte Distributionen - Suse, RedHat, Debian, Mandrake, Knoppix (Live-CD)

Grafische Benutzeroberfläche (GUI)

X Window System ist die graphische Standardschnittstelle auf Unix-Rechnern.

KDE und GNOME sind graphische Desktop-Umgebungen unter X.

Shell

Die Shell ist die Schnittstelle zur Kommandoeingabe im Textmodus.

Konsolen – Kommandos werden über den Shell Interpreter ausgeführt.

Die Shell ist auch Kommandointerpreter zur Ausführung von Shell-Skripts.

Standard Tools

Editoren (vi, Emacs.), Text-Formatierungssysteme (LaTeX), GNU-C-Compiler, Skriptsprachen wie Perl, Tcl/Tk (X-Anwendungen), Python

Netzwerk-Tools, TCP/IP, Samba zur Anbindung an Windows Netze, NFS (Net File System)

Boot-Manager (LILO, Grub), MS-DOS-Emulator

KDE-Desktop Programme - Browser (konqueror), Mailer (kmail) , Editoren (kwrite, kate) ,

Officepaket (koffice), CD-Brenner (k3b), ...

freie Anwendungs-Software

OpenOffice, gimp, mozilla, apache, php, mysql, ...

LINUX Dokumentation

Es stehen viele Büchern und Dokumente frei zur Verfügung -

Handbücher des Linux Documentation Project (LDP)

HOWTO - Dokumente zu vielen Themen

Frequently asked Questions (FAQ)

Distributionen enthalten neben diesen Dokumenten meist auch eigene Handbücher.

siehe Webportal „Linux in Österreich“ – **www.linux.at**

System Start

Nach dem Boot-Vorgang landet man beim Anmeldedialog des X-Display Managers (xdm), in früheren Versionen noch direkt in der Textkonsole beim Login – Prompt (getty).

Die Anmeldung erfolgt über Usernamen und Passwort, wobei Groß-Kleinbuchstaben unterschieden werden!

Über die Tastenkombinationen Strg-Alt-F1 bis Strg-Alt-F6 kann auf weitere Konsolenfenster im Textmodus umgeschaltet und auch parallel eingeloggt werden.

Mit der Tastenkombination Alt+F7 kommt man zurück in die X-Darstellung.

Nach erfolgreichem Einloggen befindet man sich im eigenen Home - Directory und hat dort seine eigenen und vor anderen geschützten Arbeitsverzeichnisse und auch die eigenen individuellen Konfigurationsdateien zu Anwendungen.

Der schreibende Zugriff auf systemweite Dateien ist einem Benutzer nicht möglich, sondern kann nur vom Systemadministrator "root" erfolgen.

Das Abmelden erfolgt üblicherweise über das Menü „Abmelden“ des Desktop Managers, oder aber von der Konsole aus über das Kommando „logout“ und „exit“ .

System shutdown

Das Niederfahren des Systems sollte immer korrekt ausgeführt werden, da eine abrupte Unterbrechung zu Datenverlusten von geöffneten Dateien führen kann.

Das Ausschalten kann über den Display Manager oder über die Konsolenkommandos - .

reboot , shutdown (jedoch nur als root)

Tastenkombination **STRG-ALT-DEL** (bei Freigabe auch als user)

erfolgen.

virtuelle Konsolen

Das Multiuser-System ermöglicht das gleichzeitige Arbeiten mehrerer User auf einem Computer über virtuelle Konsolen.

Die Umschaltung auf andere Konsolen erfolgt mit der Tastenkombination

Strg-Alt-F1 bis Strg-Alt-F12

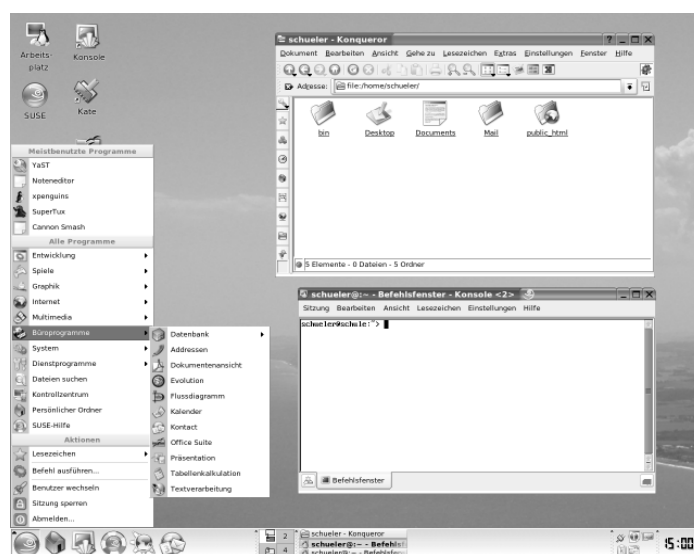
Arbeiten unter KDE

KDE (KDE Desktop Enviroment) ist ein komplettes Desktop System unter X.

Es enthält einen Window-Manager und eine Vielzahl an Standardprogrammen.

KDE als GUI ist einfach und intuitiv zu bedienen, einige Besonderheiten sind zu beachten

- Doppelklicks sind verpönt
- die rechte Maustaste öffnet Kontextmenüs
- der Zugriff auf Wechseldatenträger kann erst nach dem Einhängen erfolgen!
vor der Entnahme von Wechseldatenträger ist das Aushängen notwendig!
(in neuen Versionen erfolgt bereits standardmäßig automatisches Ein- u. Aushängen)



KDE - Desktop


Arbeiten unter der Shell

Die Kommandozeilen Ebene wird auf UNIX-Systemen Shell (Schale, auch Muschel) genannt. Im Laufe der Zeit wurden unterschiedliche Shells entwickelt. Die ursprüngliche UNIX-Shell war die Bourne-Shell, der aktuelle Standard unter Linux ist die Bourne-Again-Shell (Bash).

Die Shell ist als Schicht zwischen Betriebssystem und Benutzer zu verstehen, sie interpretiert die eingegebenen Kommandos und führt sie aus.

Die Shell kann unter X über ein X-Terminal oder im Textmodus über die Linux-Konsole ausgeführt werden. Die X-Terminals wie `xterm` (einfach) oder `konsole` (KDE-Tool) können über das KDE-Menü geöffnet werden.

Die Umschaltung von X auf Textmodus kann mit den Tasten <STRG>+<ALT>+<F1> erfolgen.



```
schueler@:~ - Befehlsfenster - Konsole <2>
Sitzung Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe

schueler@schule:~> ls -l
insgesamt 1
drwxr-xr-x  2 schueler users  48 2004-12-02 14:47 bin
drwxr-xr-x  3 schueler users 592 2004-12-02 14:47 Desktop
drwxr-xr-x  3 schueler users 328 2004-12-02 14:47 Documents
drwxr-xr-x  7 schueler users 520 2004-12-02 14:55 Mail
drwxr-xr-x  2 schueler users  80 2004-12-02 14:47 public_html
schueler@schule:~> ls /
bin  data1  dev  home  lib64  mnt  proc  sbin  sys  usr  windows
boot  data2  etc  lib   media  opt  root  srv  tmp  var
schueler@schule:~>
```

X-Terminal (konsole)

Datei- und Verzeichnisnamen

Dateinamen können eine Länge bis 255 Zeichen haben, es erfolgt die Unterscheidung zwischen Groß- und Kleinschreibung, der Punkt (".") hat keine Sonderwirkung, Dateierweiterungen haben keine Bedeutung.

Verzeichnisse werden mit "/" getrennt (nicht mit "\" wie unter DOS/Windows)

Das Wurzelverzeichnis (Root-Directory) ist mit "/" gegeben.

Das eigene Heimatverzeichnis wird mit "~", das aktuelle mit ".", das übergeordnete mit ".." abgekürzt.

Als Ersatzzeichen (Jokerzeichen) können verwendet werden

- ? ersetzt ein Zeichen ,
- * ersetzt beliebig viele Zeichen
- [abc] ersetzt genau eines der Zeichen
- [a-d] ein Zeichen aus dem angegebenen Bereich

Die **automatische Erweiterung** (Expansion) von Verzeichnisnamen, Dateinamen und Kommandos kann mit der [Tab]-Taste ausgeführt werden.

Laufwerke werden nicht über Laufwerksbuchstaben, sondern über Verzeichnisse angesprochen. (/floppy , /cdrom , /media/windows,)

Kommando Aufruf

kommandoname [-Optionen] [Argumente]

Optionen zu den Kommandos werden durch **Bindestrich (-)** eingeleitet.

Bsp.: `ls /` Ausgabe des Wurzelverzeichnisses
`ls -l /home` Ausgabe im langen Format (Option long)
`cp /* /home/schueler` kopiert alle Dateien vom aktuellen Verzeichnis auf das Verzeichnis /home/schueler/copy

Hilfe zu den Kommandos

Kurzhilfe zu Kommandos wird über die Angabe der **Option --help** ausgegeben.

Ausführliche Hilfe kann über den Aufruf des Manuals mit **man [befehl]** angefordert werden oder unter X auch über **xman**.

Unter KDE können die Man-Pages auch im Konqueror angezeigt werden mit der URL="man:/Kommando" oder "#Kommando"

die ersten Schritte mit Kommandos

den Inhalt vom aktuellen Verzeichnis über das Kommando ls (list) anzeigen

```
schueler@schule:~> ls
Desktop Documents Mail
```

nochmals im langen Format ausgeben (mit der Option -l)

```
schueler@schule:~> ls -l
insgesamt 1
drwx----- 3 schueler users 592 2004-11-28 17:07 Desktop
drwxr-xr-x  4 schueler users 232 2004-12-02 15:48 Documents
drwx----- 8 schueler users 696 2004-11-28 19:16 Mail
```

oder über das Kommando dir ausgeben

```
schueler@schule:~> dir
insgesamt 1
drwx----- 3 schueler users 592 2004-11-28 17:07 Desktop
drwxr-xr-x  4 schueler users 232 2004-12-02 15:48 Documents
drwx----- 8 schueler users 696 2004-11-28 19:16 Mail
```

die Datei „test.txt“ vom Verzeichnis „Documents“ auf das aktuelle Verzeichnis kopieren

```
schueler@schule:~> cp Documents/test.txt ./
```

das Verzeichnis „testdir“ erstellen

```
schueler@schule:~> mkdir testdir
schueler@schule:~> ls
Desktop Documents Mail testdir
```

die Datei „test.txt“ löschen

```
schueler@schule:~> rm test.txt
```

die Datei „test.txt“ vom Verzeichnis „Documents“ auf das aktuelle Verzeichnis verschieben

```
schueler@schule:~> mv Documents/test.txt ./
```

Kurzhilfe zum Kommando cp anzeigen

```
schueler@schule:~> cp --help
Aufruf: cp [OPTION]... QUELLE ZIEL
oder:   cp [OPTION]... QUELLE... VERZEICHNIS
oder:   cp [OPTION]... --target-directory=VERZEICHNIS QUELLE...
...
```

im Manual zum Kommando cp nachschlagen

```
schueler@schule:~> man cp
```

in das Verzeichnis „Documents“ wechseln

```
schueler@schule:~> cd Documents
schueler@schule:~/Documents>
```

in das übergeordnete Verzeichnis wechseln

```
schueler@schule:~/Documents> cd ..
schueler@schule:~>
```

Diskettenlaufwerk einhängen (mounten), verwenden und aushängen

```
schueler@schule:~> mount /media/floppy/
schueler@schule:~> cp test.txt /media/floppy/
schueler@schule:~> dir /media/floppy/
insgesamt 265
-rwxr-xr-x 1 schueler users      0 2004-12-02 17:53 test.txt
schueler@schule:~> umount /media/floppy/
```

auf Disketten im DOS-Format über mtools zugreifen

```
schueler@schule:~> mdir a:
Volume in drive A has no label
Directory for A:/
test      txt              0 2004-12-02 17:53 test.txt
```

Kommando Übersicht

Arbeiten mit Verzeichnissen

<code>pwd</code>	: print working directory = gibt das aktuelle Verzeichnis aus
<code>ls [Optionen][Pfad]</code>	: list = listet den Inhalt von Verzeichnissen auf
<code>ls -l</code>	Ausgabe im langen Format mit Dateityp, Rechten, Größe,...
<code>ls -R</code>	rekursive Ausgabe zeigt Inhalt der Unterverzeichnisse
<code>ls -a</code>	Ausgabe der unsichtbaren (versteckten) Dateien
<code>ls -i</code>	Ausgabe der Inode-Nummern von Dateien
<code>ls -lt</code>	sortiert nach Datum im langen Format
<code>cd Pfad</code>	: change directory
<code>cd Documents</code>	wechselt in das Verzeichnis „Documents“
<code>cd /</code>	wechselt ins Wurzelverzeichnis (Root-Directory)
<code>cd ..</code>	in das übergeordnete Verzeichnis wechseln
<code>cd ~</code>	in das eigene Home-Directory wechseln
<code>mkdir [Optionen] Dir</code>	: make directory = Verzeichnis erzeugen
<code>mkdir testdir</code>	erzeugt im aktuellen Verzeichnis das Verzeichnis „testdir“
<code>rmdir [Optionen] Dir</code>	: remove directory = Verzeichnis löschen
<code>rmdir testdir</code>	löscht das Verzeichnis texte vom aktuellen Verzeichnis
<code>tree [Optionen][Pfad]</code>	: tree = Verzeichnisbaum darstellen
<code>tree -d ~/</code>	nur Verzeichnisse ausgeben

Arbeiten mit Dateien

<code>cp [Optionen] Quelle Ziel</code>	: copy = kopiert Dateien
<code>cp Documents/* ./</code>	kopiert alle Dateien aus „Documents“ in das aktuelle Verzeichnis
<code>cp -u ...</code>	update
<code>cp -r ...</code>	kopiert rekursiv, alle Unterverzeichnisse werden mitkopiert
<code>cp -i ...</code>	(interaktiv) mit Bestätigung vor Überschreiben
<code>mv [Optionen] Quelle Ziel</code>	: move = verschieben oder umbenennen von Dateien
<code>mv * ~/prog</code>	verschiebt alle Dateien ins eigene Verzeichnis ~/prog
<code>mv test test.txt</code>	Datei test auf test.txt umbenennen
<code>rm [Optionen] Dateien</code>	: remove = löscht Dateien
<code>rm ~/test/*</code>	löscht alle Dateien vom Verzeichnis ~/prog
<code>rm -r ~/prog</code>	löscht rekursiv auch alle Unterverzeichnisse von ~/prog
<code>rm -ri ~/prog</code>	interaktiv, rekursives löschen
<code>touch Datei</code>	: legt eine neue Datei mit leerem Inhalt an
<code>file Datei</code>	: stellt den Dateityp fest (ist an der Endung nicht zu erkennen!)
<code>find [Pfad][Optionen][Suchmuster]</code>	: sucht nach Dateien
<code>find -name "test"</code>	nach allen Dateien, die mit Namen „test“ beginnen, suchen
<code>find -user schueler -name "test.*"</code>	nach Dateien des Besitzers „schueler“ suchen

Arbeiten mit Textdateien

cat Datei	: gibt den Inhalt von Textdateien aus
more [Optionen] Datei	: Pager für Textdateien zur seitenweisen Darstellung
less [Optionen] Datei	: leistungsfähiger Pager für Textdateien - "less is more" der Abbruch erfolgt über die Taste 'Q' die Hilfestellung über help das Suchen über /Suchmuster
head [Optionen] Datei	: gibt den Anfang einer Textdatei aus
tail [Optionen] Datei	: gibt das Ende einer Textdatei aus
grep [Optionen] Suchstring Datei	: durchsucht Dateien nach Ausdrücken
grep "ein" test.txt	durchsucht die Datei „test.txt“ nach String „ein“
grep -i "Ein" test.txt	ignoriert Groß-/Kleinschreibung
grep -n "ein" test.txt	gibt die Zeilennummern der Fundstellen aus

Komprimieren und Archivieren von Dateien

gzip [Optionen] Datei	: komprimiert/dekomprimiert Dateien
gzip test.txt	komprimiert die Datei test.txt ? test.txt.gz
gzip -d test.txt.gz	dekomprimiert die Datei test.txt.gz ? test.txt
tar [Optionen] Archivname [Dateien]	: tool archiv = Archiv Programm
tar -cf backup.tar ./testdir/*	packt mehrere Dateien zu einer Datei zusammen
tar -tf backup.tar	erstellt das Archiv „backup.tar“ vom Verzeichnis testdir
tar -xf backup.tar	testet das Archiv (listet die Dateien vom Archiv auf)
	entpackt (extrahiert) das Archiv
	auch mit Option -z (für zip = komprimieren)
tar -czf backup.tgz ./testdir/*	erstellt das komprimierte Archiv „backup.tgz“
tar -tzf backup.tgz	testet das komprimierte Archiv
tar -xzf backup.tgz	entpackt (extrahiert) das komprimierte Archiv

Prozessverwaltung

top	: zeigt dynamisch Prozessinformationen an Abbruch mit Taste <STRG>+<c> oder <q>
ps [Optionen]	: prozess state = zeigt die Prozessliste
ps	Prozessliste mit Prozessnummern (pid)
ps -l	langes Format mit Zusatzinformationen
ps -x	alle eigenen Prozesse, auch Hintergrundprozesse
ps -A	alle Prozesse, auch Systemprozesse
ps -u	mit ausführlichen Zusatzinformationen
ps -U root	alle Prozesse eines Users
kill pid	: beendet den Prozess mit der Prozessnummer pid
kill 1234	beendet den Prozess mit PID=1234
kill -s SIGKILL 1234	beendet den Prozess mit PID=1234 mit Signal SIGKILL
killall processname	: beendet Prozesse über den Prozessnamen
killall kwrite	beendet den Prozess „kwrite“

Dateisystem

mount	: eingehängte Dateisysteme anzeigen
mount [mountpoint]	: Dateisystem einhängen (mounten) als user nur jene aus der Dateisystemtabelle „/etc/fstab“
umount [mountpoint]	: Dateisystem aushängen (unmounten) vor der Entnahme des Wechseldatenträgers!
fdformat [device]	: Disketten Formatierung im DOS(FAT)-Format
fdformat /dev/fd0	Diskette im 1.Laufwerk formatieren
fdformat /dev/fd0h1440	3-1/4-Zoll-HD-Diskette im 1.Laufwerk formatieren
mtools	: Zugriff auf Disketten im DOS(FAT)-Format
mdir a:	Laufwerk A: auflisten
mcopy test.txt a:\test.txt	auf Laufwerk A: kopieren
mdel a:*.*	Dateien vom Laufwerk A: löschen
mmd a:\testdir	Verzeichnis auf Laufwerk A: erstellen

Systeminformationen

who	: Liste aller eingeloggten User
whoami	: eigener aktueller Username
last	: Liste der zuletzt eingeloggten User
last -n 10	die letzten 10 Userlogins zeigen
finger [Optionen] [user]	: Information zu User abfragen
passwd [username]	: Passwort ändern
su [username]	: set user = auf anderen Benutzer wechseln
date	: Datum und Uhrzeit abfragen
du [Optionen][Pfad]	: disk usage = belegter Speicherplatz von Dateien
du -h	Anzeige im "human readable format"
df [Optionen][Pfad]	: disk free = Belegung der eingehängten Dateisysteme
free	: Belegung des Hauptspeichers (RAM u. Swap)
uname [Optionen]	: Betriebssystem-Informationen anzeigen
uname -a	: alle Informationen

Sonstiges

lpr Datei	: line print = Datei drucken
lpr test.txt	Datei test.txt ausdrucken
lpq	: printer queue = Druckaufträge auflisten
lprm nr	: Druckauftrag mit Nummer nr entfernen
at [Optionen] time	: zeitgesteuerte Programmausführung
at now + 10 Minutes	Job in 10 Minuten ausführen
at> cp test.txt test2.txt	Kommando Eingabe
at> <STRG>+<d>	Eingabe beenden

Bourne Again Shell (bash)

Die Bash ist die Standard-Shell unter Linux.

Die ursprüngliche Shell war die Bourne-Shell (sh), ein einfacher Kommando-Interpreter entwickelt von Steven R. Bourne. C-Programmierer entwarfen eine C-orientierte Shell, die C-Shell (csh), die zum Standard von Berkley-Unix wurde. Eine Weiterentwicklung der Bourne-Shell war die Korn-Shell (ksh), die auch Eigenschaften der C-Shell übernahm.

Die Free Software Foundation entwickelte dann die leistungsfähige Bourne Again Shell.

Mit dem Kommando `echo $SHELL` kann die Login-Shell ermittelt werden :

```
schueler@schule:~> echo $SHELL
/bin/bash
```

mit dem Kommando `csh` kann z.B. in die C-Shell gewechselt werden

```
schueler@schule:~> csh
/home/schueler>
```

mit dem Kommando `exit` wird die Shell beendet

```
/home/schueler> exit
```

die verfügbaren Shells sind in der Datei `/etc/shells` angeführt :

```
less /etc/shells
```

die Startup-Dateien (Start-Konfigurations-Dateien) der Bash sind :

```
~/.bashrc und ~/.profile
```

Bash Fähigkeiten

- Kommandozeilenspeicher (History)

- Kommando u. Dateinamen Expansion

- Aliase zu Kommandos

- Prozesse starten u. stoppen

- Kommando Verknüpfungen

- Pipes und Umleitungen

- Shellvariablen

- Shell Script Programmierung

Tastenkürzel

- Pfeiltasten ←→ Cursorposition

- Pos1, Ende Beginn/Ende der Zeile

- STRG+c Kommando Abbruch

- ALT+d Wort löschen

- STRG+k bis Zeilenende löschen

- ALT+t beide letzten Worte tauschen

- STRG+l Bildschirm löschen

History

jede Instanz einer Shell hat einen eigenen Kommandozeilenbuffer, in welchem die eingegebenen Befehle gespeichert werden. Nach Beenden der Shell wird die History in eine Datei (`~/.bash_history`) gespeichert und steht so der nächsten Shell wieder zur Verfügung.

Mit den Pfeiltasten ↑↓ kann im Kommandozeilenspeicher geblättert werden.

Zusätzlich gibt es Tastenkürzel, wie

- Strg-R rückwärts suchen (jedes eingegebene Zeichen führt zu passender Befehlszeile)

- Strg-S vorwärts suchen

- Esc- < zum ersten Befehl

- Esc- > zum letzten Befehl

mit dem Kommando `history` wird die aktuelle History angezeigt:

```
schueler@schule:~> history
```

mit dem Kommando `history -c` kann die History gelöscht werden:

```
schueler@schule:~> history -c
```

Kommando u. Dateinamen Expansion

Die **automatische Erweiterung** (Expansion) von Verzeichnisnamen, Dateinamen und Kommandos kann über die [Tab]-Taste ausgeführt werden.

Die Tab-Taste einmal gedrückt, ergänzt die Eingabe, soweit dies eindeutig möglich ist. Die Tab-Taste zweimal gedrückt, listet alle möglichen Ergänzungen auf.

z.B. die Expansion für das Kommando *whoami* :

es wird der Teil „wh“ eingegeben und danach die Tab-Taste gedrückt

```
schueler@schule:~> wh [TAB]
```

es erfolgt ein Signalton, da mehrere Kommandos zur Zeichenfolge passen und daher wird nochmals die Tab-Taste betätigt

```
schueler@schule:~> wh [TAB][TAB]
```

daraufhin wird die Liste aller passenden Kommandos ausgegeben

```
whatis whereis which while who whoami
```

zur Auswahl genügt jetzt die ergänzende Eingabe der Zeichen 'oa' und [TAB]

```
schueler@schule:~> whoa [TAB]
```

```
schueler@schule:~> whoami
```

und es wird automatisch auf den vollständigen Kommandonamen erweitert.

Die Expansion funktioniert auch für Pfadangaben und Dateinamen:

```
schueler@schule:~> less ~/test [TAB][TAB]
```

```
testdir/ test.txt
```

```
schueler@schule:~> less ~/test. [TAB]
```

```
schueler@schule:~> less ~/test.txt
```

Aliase zu Kommandos

zu häufig verwendeten Kommandozeilen können Abkürzungen (Aliase) definiert werden.

mit dem Kommando *alias* wird die aktuelle Alias-Liste angezeigt:

```
schueler@schule:~> alias
```

```
alias dir='ls -l'
```

```
...
```

die Definition erfolgt mit *alias kurzname='Kommandostring'* :

```
schueler@schule:~> alias md=mkdir
```

mit *unalias kurzname* wird ein Alias gelöscht:

```
schueler@schule:~> unalias md
```

Prozesse

Ein Programm, das sich in Ausführung befindet, ist ein Prozess.

Jeder Prozess hat eine eindeutige Prozessnummer (PID), einen Besitzer und weitere Prozesskennndaten (Prozessumgebung). Prozesse können gestartet, angehalten, reaktiviert, beendet und ihre Ausgaben unterbunden werden (Prozesssteuerung, Jobkontrolle).

Prozesse werden entweder als Vordergrund- oder als Hintergrund-Prozess gestartet.

Prozesse werden von der Shell grundsätzlich im Vordergrund gestartet, indem ein Kommando eingegeben wird. Die Shell wartet dann bis der Prozess beendet ist und kehrt erst danach zur Eingabeaufforderung zurück.

Kommandos, die längere Zeit zur Ausführung benötigen oder dauernd laufen, können daher als Hintergrund-Prozess (Job) gestartet werden. Dabei gibt die Shell sofort nach dem Aufruf des Prozesses den Prompt wieder aus und steht somit für weitere Eingaben zur Verfügung.

Daemon-Prozesse sind unsichtbare Hintergrund-Prozesse, wie Serverprozesse.

Zombies sind Prozesse, die ihre Arbeit erledigt haben und damit bereits gestorben sind, aber noch auf andere Prozesse warten müssen, die ihre Daten brauchen.

Über die Inter-Process-Communication (IPC) können Prozesse auch Signale an andere Prozesse senden und welche empfangen.

Starten von Prozessen

mit *Kommando* wird ein Vordergrund-Prozess gestartet:

```
schueler@schule:~> kwrite
```

mit den Tasten <STRG>+<c> kann die Ausführung jederzeit abgebrochen werden

mit *Kommando &* (auch *Kommando&*) wird ein Hintergrund-Prozess gestartet:

```
schueler@schule:~> kwrite &
```

Prozessliste anzeigen

mit dem Kommando *jobs* können alle in der Shell laufenden Jobs angezeigt werden:

```
schueler@schule:~> jobs
[1]+  Running    kwrite &
```

mit dem Kommando *ps* wird die Prozessliste mit Prozessnummern (PID) angezeigt:

```
schueler@schule:~> ps
  PID TTY          TIME CMD
 7645 pts/1        00:00:00 bash
 8640 pts/1        00:00:00 kwrite
```

die Prozess-Kenndaten können mit *ps -f* und *ps -l* angezeigt werden:

```
schueler@schule:~> ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
schueler   7645    7620  0 12:00 pts/1        00:00:00 /bin/bash
```

mit *ps -U username* können alle Prozesse eines Users angezeigt werden:

```
schueler@schule:~> ps -U schueler
  PID TTY          TIME CMD
 7499 ?            00:00:00 kde
 7534 ?            00:00:00 gpg-agent
```

...

mit *ps -A* können alle Prozesse angezeigt werden:

```
schueler@schule:~> ps -A
  PID TTY          TIME CMD
    1 ?            00:00:01 init
    2 ?            00:00:00 ksoftirqd/0
```

...

die Bedeutung der einzelnen Felder:

UID	User-ID	PID	Process-ID
PPID	Parent-Process-ID	C	Priorität
STIME	Startzeit	TTY	Terminal
TIME	verbrauchte CPU-Zeit	CMD	Start-Kommando

mit *pstree* kann die Prozesshierarchie angezeigt werden:

```
schueler@schule:~> pstree
      ...
    |--kdeinit--bash--T--kwrite
      ...
```

Stoppen von Prozessen

mit dem Kommando *kill PID* können Prozesse beendet werden :

```
schueler@schule:~> kill 8640
```

es können auch Signale an einen Prozess gesendet werden,

die Liste aller möglichen Signale erhält man mit *kill -l*

```
schueler@schule:~> kill -l
```

Signale mit Signalnummern:

- 1) SIGHUP terminiert auch alle Folgeprozesse (Kindprozesse)
- 2) SIGINT unterbricht Prozess, wie <STRG>+<c>
- 9) SIGKILL Abbruch erzwingen, Notbremse
- 15) SIGTERM default-Signal für kill
- ...

mit dem Kommando *kill -SIGHUP PID* werden Prozess und Folgeprozesse beendet :

```
schueler@schule:~> kill -SIGHUP 8640
```

Jobmanagement - Foreground/Background

mit den Kommandos fg (foreground) und bg (background) kann ein laufender Prozess in den Vordergrund oder Hintergrund gewechselt werden.

in den Vordergrund wechseln:

```
schueler@schule:~> fg kwrite
```

in den Hintergrund wechseln, zuerst muss mit <STRG>+<z> der Prozess angehalten werden :

```
[STRG]+[z]  
schueler@schule:~> bg kwrite
```

Kommandoverkettungen

Mehrere Kommandos können verkettet ausgeführt werden, wobei die Verknüpfung über den Rückgabewert der Kommandos erfolgt.

Eine fehlerfreie Abarbeitung eines Kommandos oder Programmes ergibt den Rückgabewert=0, jeder andere Wert weist auf einen Fehler hin.

einfache Verkettungen

Programme gleichzeitig im Vordergrund starten

Programm1 ; Programm2 ; Programm3

Programme gleichzeitig im Hintergrund starten:

Programm1 & Programm2 & Programm3 &

abhängige Verknüpfungen

Ausführung von Programm2 nur dann, wenn Programm1 fehlerfrei abschließt

Programm1 && Programm2

Ausführung von Programm2 nur dann, wenn das Programm1 mit Fehler abschließt

Programm1 || Programm2

Beispiele:

```
schueler@schule:~> date ; who  
schueler@schule:~> dir test.* && cat test.*
```

Ein- und Ausgabeumleitungen

Die Ein- u. Ausgaben eines Programmes können in Dateien umgeleitet werden.

Jedes Programm hat drei Standard-Kanäle vordefiniert.

Standard-Eingabe (stdin mit Dateideskriptor = 0)

Standard-Ausgabe (stdout = 1)

Standard-Fehlerausgabe (stderr = 2)

Standardmäßig ist stdin der Tastatur, stdout und stderr dem Monitor zugeordnet,

können jedoch über > , < , << , >> , >& , 2> in Dateien umgeleitet werden.

Programm > Datei : Umleitung der Ausgaben in eine Datei

Programm >> Datei : Umleitung in Datei, an eine bestehende Datei wird angehängt

Programm < Datei : Umleitung der Eingabe aus einer Datei

Programm 2> Datei : Umleitung der Fehlerausgaben in eine Datei

Beispiele:

Inhaltsverzeichnis auf die Datei „liste“ ausgeben

```
schueler@schule:~> ls > liste
```

oder an die Datei liste anhängen

```
schueler@schule:~> ls >> liste
```

den Inhalt einer Datei ausgeben

```
schueler@schule:~> cat < liste
```

mit wc ("word count") die Anzahl der Wörter einer Datei ermitteln

```
schueler@schule:~> wc < liste
```

Pipes

Es ist möglich, die Standard-Ausgabe eines Programms gleich direkt mit der Standard-Eingabe eines anderen Programms über Pipes (|) zu verbinden.

Auf diese Weise lassen sich beliebig viele Kommandos zusammenfügen und komplizierte Aufgabenstellungen durch eine einzige Kommandozeile bewältigen.

Programm1 | Programm2

Die Standard-Ausgabe von Programm1 wird mit der Standard-Eingabe von Programm2 verbunden.

Beispiele:

Inhaltsverzeichnis ausgeben und mit less scrollen

```
schueler@schule:~> ls | less
```

Inhaltsverzeichnis ausgeben und mit wc ("word count") die Anzahl der Worte zählen

```
schueler@schule:~> ls | wc -w
```

Komplexe Pipekonstruktionen erfordern oft, dass einzelne Zwischenschritte in eine Datei gespeichert werden. Dazu gibt es das Programm *tee*, das die Eingabe auf die Ausgabe weitergibt und zusätzlich den Datenstrom in eine Datei speichert.

Es handelt sich also um eine Art T-Stück in einer Pipe - daher stammt auch der Name.

Programm1 | tee Datei1 | Programm2 > Datei2

Das Programm1 schickt seine Ausgabe an das Programm *tee*, die sie in die Datei1 schreibt, aber gleichzeitig auch an das Programm2 weitergibt. Dieses Programm speichert sein Ergebnis in der Datei2.

Shellvariablen

Jede Shell kann auch Variablen (Shellvariable, Umgebungsvariable) benutzen.

Die Summe aller definierten Variablen nennt man die Umgebung der Shell (environment).

Es können lokale und auch globale Variable definiert werden.

Variable sind automatisch lokal, wenn sie nicht mit *export* exportiert werden.

Globale (exportierte) Variable werden an Folgeprozesse (Subshells) als Kopie weitergeleitet, lokale sind nur in der eigenen Shell gültig.

Shellvariable definieren:

Variable = Wert

globale Variable:

export Variable = Wert

Inhalt einer Variablen:

\$Variable

Variable löschen:

unset Variable

Umgebungsvariable anzeigen:

printenv

Beispiele:

```
x=abc          # der lokalen Shell Variablen x wird "abc" zugewiesen
echo $x        # Wert der Variablen x anzeigen
export x2=0    # globale Variable x2 definieren (exportieren)
export         # alle globalen Variablen anzeigen
printenv      # Umgebungsvariable anzeigen
unset x2      # löscht die Variable x2
```

Systemumgebungsvariable werden beim Start über Konfigurationsdateien als globale

Variable erstellt und enthalten systemweite oder auch benutzerbezogene Informationen, wie

HOME	Heimatverzeichnis
PATH	Liste mit Suchpfaden
HOSTNAME	Rechnername
PATH	Suchpfade

die Systemvariable PATH (definiert Suchpfade) ausgeben :

```
echo $PATH
/home/schueler/bin:/usr/local/bin:/usr/bin:/usr/X11R6/bin:/bin:
/usr/games:/opt/gnome/bin:/opt/kde3/bin
```

Benutzer und Zugriffsrechte

Ein Multiuser-System benötigt eine Benutzerverwaltung mit Zugangs- und Rechtesystem. Benutzer können sich nur über den Benutzernamen mit zugehörigem Passwort anmelden. Schreibrechte eines Benutzers sind im allgemeinen auf die Dateien des eigenen Heimat-Verzeichnisses (home-directory) beschränkt. Ein User gehört auch mindestens einer Gruppe an und kann damit auch kollektive Gruppenrechte erhalten. Nur der Administrator (root) verfügt über unbegrenzte Rechte und hat unlimitierten Zugriff auf alle Dateien.

Jede Datei (und jedes Verzeichnis) hat einen Besitzer und ist auch einer Gruppe zugeordnet. Für den Besitzer, die Gruppe und für alle restlichen Benutzer können unterschiedliche Zugriffsrechte auf eine Datei zugewiesen werden. Diese Verwaltungsinformationen werden im Inode (Informationsknoten) einer Datei gespeichert.

UID und GID

Jeder Benutzer wird intern über eine eindeutige Benutzernummer (**UID** = user identification), jede Gruppe über eine Gruppennummer (**GID** = group identification) verarbeitet.

Benutzer und Gruppen werden über die Dateien /etc/passwd und /etc/groups verwaltet.

Die Abfrage der eigenen UID kann mit dem Kommando **id** erfolgen.

Über das Kommando **id** die eigene UID und GID anzeigen:

```
schueler@schule:~> id
uid=1000(schueler) gid=100(users)
```

die Konfigurationsdatei /etc/passwd mit den Benutzerinformationen ausgeben:

```
schueler@schule:~> less /etc/passwd
root:x:0:0:root:/root:/bin/bash
...
schueler:x:1000:100::/home/schueler:/bin/bash
```

Benutzerkategorien

user	(u)	Besitzer (Eigentümer)
group	(g)	Gruppe
others	(o)	alle anderen (restlichen Benutzer)

Zugriffsrechte	für	Dateien	Verzeichnisse
readable	(r)	lesbar	lesbar
writable	(w)	schreibbar	Dateien anlegen, löschen
executable	(x)	ausführbar	Unterordner betreten

Anzeigen der Dateikennndaten mit **ls -l**

-	rwxr-x---	1	schueler	users	36	Jan 14,14:25	test.txt
Dateityp	Zugriffsrechte	Referenz	Besitzer	Gruppe	Größe	Datum	Dateiname

Zugriffsrechte - je drei Zeichen beschreiben die Rechte für user, group und others

rw	x	user	- Rechte
r	-	x	group - Rechte
-	-	-	others - Rechte

Dateitypen	-	reguläre (normale) Datei
	d	Verzeichnis (directory)
	l	symbolischer Link

Beispiel: Zugriffsrechte ermitteln

```
schueler@schule:~> ls -l
-rw-r--r-- 1 schueler users 36 2004-12-02 20:53 test.txt
```

die Datei test.txt gehört in den Besitz des Benutzers schueler und der Gruppe users, der Besitzer hat die Rechte rw (lesen,schreiben), die Gruppe und alle anderen r (lesen).

Ändern von Zugriffsrechten mit chmod

chmod Zugriffsrechte Datei1 [Datei2 ...]

Die Zugriffsrechte (Modus) können entweder numerisch oder symbolisch angegeben werden. Die **symbolische Modusangabe** hat folgendes Schema: ugoa += rwxstugo

Beispiele:

chmod u+w test.txt	der Datei test.txt für den Besitzer das w-Recht hinzufügen
chmod g=rx test.txt	für die Gruppe das r- und x-Recht setzen
chmod o-rwx test.txt	allen anderen alle Rechte entziehen
chmod o= test.txt	allen anderen kein Recht setzen (=alle Rechte nehmen)
chmod ug=rw test.txt	für Besitzer und Gruppe rw setzen
chmod a+x test.txt	für alle (a=ugo!) das x-Recht hinzufügen
chmod u=rw,g= test.txt	User auf rw setzen, Gruppe alles entziehen
chmod g=o test.txt	der Gruppe die gleichen Rechte des Besitzers geben

wird die Option -R (rekursiv) angegeben, so werden die Rechte eines ganzen Verzeichnisbaumes inklusive der enthaltenen Dateien und Unterverzeichnisse geändert.

Beispiel:

chmod -R g=r testdir den ganzen Verzeichnisbaum von testdir ändern

Die **numerische Angabe** für Zugriffsrechte erfolgt über Zahlenwerte nach folgender Bitmaske:

r	w	x
4	2	1

Beispiele:

chmod 750 test.txt	→ user= rwx , group= r-x, others= ---
chmod 644 test.txt	→ user= rw- , group= r-- , others= r--

Maske für den Zugriffsmodus mit umask festlegen

Das Kommando umask zeigt oder verändert die Dateierzeugungsmaske, nach der die Zugriffsrechte für neu erstellte Dateien bestimmt werden.

Die Erzeugungsmaske wirkt als Negativmaske : $7 - \text{Wert} = \text{Maske}$

Beispiele:

umask 022	→ user=7-0=7=rwx, group=7-2=5=r-x, others=7-2=5=r-x
umask 137	→ user=7-1=6=rw-, group=7-3=4=r-- , others=7-7=0=---

das x-Recht wird jedoch bei Dateien nur dann gesetzt, wenn bei einer Kopie die Quelldatei auch ausführbar ist.

Das Kommando umask kann mit Angabe der Option -S auch symbolisch verwendet werden.

Beispiel:

umask -S u=rwx,g=rx,o=rx

Eigentümer und Gruppenzugehörigkeit ändern

Mit dem Kommando **chown** kann der Administrator root die Besitzer von Dateien ändern.

Mit dem Kommando **chgrp** kann die Gruppenzugehörigkeit geändert werden,

als root uneingeschränkt, als Normaluser jedoch nur auf jene Gruppen, denen man angehört.

Spezielle Rechte

Neben den Zugriffsrechten für User, Group und Others gibt es noch spezielle Rechte, die über die führende vierte Ziffer oder die Symbole s,t dargestellt werden.

Damit können die Bits für SUID (Substitute UserID mit Wert=4), SGID (Substitute GroupID mit Wert=2) und das Sticky Bit (mit Wert=1) gesetzt werden.

SUID-Bit setzen und anzeigen:

```
schueler@schule:~> chmod 4755 test.txt
-rwsr-xr-x 1 schueler users      36 2004-12-02 20:53 test.txt
```

Das **SUID-Recht** gilt nur für ausführbare Dateien und erscheint beim Eigentümerrecht mit einem "s" statt dem "x". Jeder User, der dieses Programm ausführt, tut dies unter der effektiven UID des Besitzers. So kann z.B. das Programm /usr/bin/passwd auch als Normaluser zur Passwortänderung verwendet werden und mit root-Rechten auf die Passwortdateien zugreifen.

Zugriffsrechte mit SUID-Bit für passwd anzeigen :

```
schueler@schule:~> ls -l /usr/bin/passwd
-rwsr-xr-x 1 root shadow 90189 2004-10-02 07:00 /usr/bin/passwd
```

Das **SGID-Recht** gilt für ausführbare Dateien und auch für Verzeichnisse.

Bei ausführbaren Dateien erfüllt es die gleiche Aufgabe wie das SUID-Recht, jedoch für die Gruppenzugehörigkeit. Bei einem Verzeichnis werden alle neu angelegten Dateien und Unterverzeichnisse der Gruppe zugewiesen, der das Verzeichnis gehört.

Das **Sticky-Bit** kann auf Verzeichnisse gesetzt werden und verhindert das gegenseitige Löschen jener User mit Schreibrecht auf dieses Verzeichnis.

Sticky-Bit setzen und anzeigen :

```
schueler@schule:~> chmod 1640 testdir
schueler@schule:~> ls -ld
drwxrwxr-T 2 schueler users      48 2004-12-07 10:28 testdir
```

Access Control Lists

über Access Control Lists (ACLs) können unter Linux erweiterte Zugriffsrechte vergeben und weiteren Benutzern und Gruppen damit individuelle Zugriffsrechte erteilt werden.

ACL-Einträge zeigen sich bei ls -l über ein zusätzliches '+' :

```
schueler@schule:~> ls -l
-rw-r--r--+ 1 schueler users      36 2004-12-02 20:53 test.txt
```

Details können über das Kommando **getfacl** angezeigt werden:

```
schueler@schule:~> getfacl test.txt
# file: test.txt
# owner: schueler
# group: users
user::rw-
user:lehrer:rw-
group::r--
mask::rw-
other::---
```

Die Datei test.txt des Besitzers schueler hat über die erweiterte ACL zusätzlich für den Benutzer lehrer auch rw-Rechte eingetragen.

ACL-Einträge werden über das Kommando **setfacl** ausgeführt, über die Option -m (modify) kann eine bestehende ACL modifiziert werden.

```
schueler@schule:~> setfacl -m user:lehrer:rw,group:projekt:rw test.txt
```

Über die Option -d können weiters Default-ACL-Einträge für Verzeichnisse erfolgen, die beim Erstellen von Dateien im Verzeichnis als Access-ACL übernommen werden.

Dateisystem

Dateisysteme - Linux unterstützt mehrere Dateisysteme, die wichtigsten sind :

ext2	Standard von Linux
ext3	Weiterentwicklung mit Journaling File System
reiserfs	Journaling File System
nfs	Network File System
ramdisk	RAM-Speicher als Dateisystem
iso9660	CD-ROM Format
msdos	DOS
vfat	Windows95 (16 bit u. 32 bit)
ntfs	WindowsNT nur lesend
smb	Samba für Windows Freigaben

mit `ls /proc/filesystems` können die aktuell unterstützten Dateisysteme angezeigt werden.

Standard Verzeichnisbaum

Der Filesystem Hierarchie Standard (FHS) definiert die grundsätzliche Verzeichnisstruktur.

/	Wurzel-Verzeichnis (root directory)
/home	Heimatverzeichnisse der Benutzer (home directories)
/root	Heimatverzeichnis des Administrators (root)
/boot	Boot-Dateien (Bootloader, Kernelimage)
/etc	Systemkonfigurationsdateien
/bin	wichtige Kommandos (Binaries)
/sbin	Administrator-Kommandos
/lib	Systembibliotheken (Libraries)
/dev	Geräte-Dateien (device files)
/proc	virtuelle Informationsdateien (process files)
/usr	Anwendungsprogramme, Dokumentationsdateien
/mnt	temporärer Mountpoint
/tmp	temporäre Dateien
/var	permanent ändernde Dateien
/lib	Bibliotheken (Shared Libraries)
/opt	optionale Software (kde, gnome, mozilla, ...)

Wurzelverzeichnis ausgeben über das Kommando **ls /** :

```
schueler@schule:~> ls /
bin  dev  home  lib   mnt  proc  sbin  sys  usr
boot etc  lib   media opt  root  srv  tmp  var
```

Dateitypen

-	reguläre Dateien (regular file)
d	Verzeichnisse (directory)
l	symbolische Links (symbolic link)
b	blockorientierte Geräte (block device)
c	zeichenorientierte Geräte (char device)
p	feststehende Programmverbindungen (named pipe)
s	Netzwerk-Kommunikationsendpunkte (sockets)

Dateityp über **ls -l** anzeigen:

```
schueler@schule:~> ls -l /usr/
drwxr-xr-x 121 root root 3224 2004-12-20 12:11 share
lrwxrwxrwx  1 root root    5 2004-11-13 17:52 X11 -> X11R6
...
```

Dateityp über Kommando **file** anzeigen:

```
schueler@schule:~> file ./*
./sverweis.txt: symbolic link to `test.txt'
./test.txt:     ASCII text, with CRLF line terminators
```

Inodes

Inodes (Informationsknoten) speichern die Verwaltungsinformationen einer Datei, wie Besitzer, Gruppe, Zugriffsrechte, Zugriffszeiten, Größe und auch die Adressen der Datenblöcke. Jedes Verzeichnis (auch das Wurzelverzeichnis) ist nichts anderes als eine Datei, deren Inhalt die Dateinamen der enthaltenen Dateien samt ihren Inode-Nummern enthält.

Die Inode-Nummer einer Datei kann mit **ls -li** angezeigt werden.

```
schueler@schule:~> ls -li test.txt
184391 test.txt
schueler@schule:~> ls -li test.txt
184391 -rwxr-x-- 1 schueler users 36 2004-12-02 20:53 test.txt
```

Links

Links sind Verweise auf Dateien. Über Links kann über verschiedene Dateinamen auf dieselbe Datei zugegriffen werden. Es können damit Mehrfachnamen für eine Datei verwendet werden. Bei **festen Links** (hard links) verweist ein weiterer Dateiname zu einem vorhandenen Inode. Im Inode wird die Anzahl von festen Links auch über einen Zähler mitgeführt. Links können aufgrund des gemeinsamen Inodes nur gleiche Besitzer, Gruppe und Zugriffsrechte haben. Bei **symbolischen Links** (symbolic links) wird statt der Inode-Nummer der gesamte absolute Pfad angegeben. Symbolische Links können damit auf Dateien anderer Verzeichnisse oder auch auf Verzeichnisse selbst verweisen.

Links erstellen mit Kommando **ln**

ln [Optionen] Ziel Verweisname

Hard Link mit **ln** erstellen und die Inode-Nummer mit **ls -li** anzeigen :

```
schueler@schule:~> ln test.txt verweis.txt
schueler@schule:~> ls -li
174488 -rw-r----- 2 schueler users 36 2004-12-02 19:15 test.txt
174488 -rw-r----- 2 schueler users 36 2004-12-02 19:15 verweis.txt
```

die Ausgabe zeigt die gleiche Inode-Nummer für beide Dateien und den Zählereintrag=2

Symbolic Link mit **ln -s** erstellen und anzeigen :

```
schueler@schule:~> ln -s test.txt sv.txt
schueler@schule:~> ls -l
lrwxrwxrwx 1 schueler users 8 2005-01-23 15:14 sv.txt -> test.txt
-rw-r----- 1 schueler users 36 2005-01-04 19:15 test.txt
```

die Ausgabe zeigt den Dateityp l und den Dateieintrag mit einem Pfeil auf die Zieldatei an.

Verweis auf den Linux-Kernel in /boot anzeigen :

```
schueler@schule:~> ls -l /boot/
lrwxrwxrwx 1 root root 24 2004-11-13 17:51 vmlinuz -> vmlinuz-2.6.8
-rw-r--r-- 1 root root 1608070 2004-10-06 15:00 vmlinuz-2.6.8
```

der Standard-Dateiname des Kernels „vmlinuz“ verweist auf den aktuell verwendeten Kernel.

Mounten von Dateisystemen

Bevor ein Dateisystem verwendet werden kann, muss es in den Verzeichnisbaum eingehängt werden. Das Einhängen erfolgt über das Kommando mount, das Abhängen über umount.

Einhängen und Abhängen sind standardmäßig nur vom Administrator (root) ausführbar, außer den vordefinierten Mountpoints der Filesystemtable der Datei „/etc/fstab“.

aktuelle Mountpoints anzeigen über mount (auch als Normaluser)

```
schueler@schule:~> mount
/dev/hda2 on / type ext3
```

...

vordefinierten Mountpoint „/media/floppy“ als User ausführen

```
schueler@schule:~> mount /media/floppy/
```

die Dateisystem-Tabelle von „/etc/fstab“ anzeigen

```
/dev/hda2 / ext3 defaults 1 1
/dev/fd0 /media/floppy auto noauto,user, sync 0 0
```

...

Konsolen Editoren

vi der Standard UNIX-Editor zur Administration, ist jedoch wenig benutzerfreundlich
(unter Linux als vim = vi improved)

Aufruf : vi [dateiname]

Nach dem Start befindet sich der Editor automatisch im Kommandomodus,
über die [Ins]-Taste oder die Taste [i] gelangt man in den Eingabemodus,
der Wechsel von Eingabe- in den Kommandomodus erfolgt durch die Esc-Taste.

einige **Kommandos**

Öffnen, Speichern und Beenden im Kommandomodus

: e datei	Datei öffnen
: r datei	Datei öffnen und an Cursorposition einfügen
: w [datei]	Datei speichern
: w >> datei	Inhalt an eine Datei anfügen
: w q	Datei speichern und vi verlassen
: q !	vi verlassen ohne zu speichern

Wechsel in den Kommandomodus

Esc auf Kommandomodus

Wechsel in den Eingabemodus

a	umschalten auf Texteingabemodus für Text anhängen
i	umschalten für Einfügen
o	umschalten für neue Zeile

Textbearbeitung im Kommandomodus

dd	löscht eine Zeile
D	löscht bis Zeilenende
yy	aktuelle Zeile in den Puffer kopieren
p	fügt Puffer nach aktueller Zeile ein
u	UNDO letzte Aktion
/muster	Suche nach 'muster'

Hilfe im Kommandomodus

: help	Hilfe aufrufen
: help command	Hilfe zu Kommando
: quit	Hilfe beenden

emacs der erste Standard Linux-Editor

umfangreiche Möglichkeiten mit Syntax-Highlighting für verschiedene Formate,
auch unter X (xemacs) mit Menübedienung verwendbar.

Aufruf : emacs [dateiname]

einige **Kommandos**

Öffnen, Speichern und Beenden

Strg+X,STRG+F	Datei öffnen
Strg+X,STRG+S	Datei speichern
Strg+X,STRG+W	Datei speichern als
Strg+X,STRG+C	Emacs verlassen

Textbearbeitung

Strg+K	löscht ab Cursor bis Zeilenende
Strg+Y	fügt den gelöschten Text an Cursorposition ein
Strg+S	Suche
Strg+R	Suche rückwärts
Strg+G	Suche abbrechen

Online Hilfe

Strg+H	Hilfe
--------	-------

X - Window-System

X-Window-System (**X11**) ist die Basis für grafische Benutzeroberflächen und beruht auf einem Client-Server-System. XFree86 ist die freie Version von X11 für Linux auf 80x86 Systemen.

Der X-Server steuert die Zugriffe auf die Grafikhardware, Tastatur und die Maus.

Der X-Client ist das Anwendungsprogramm, das über das X-Protokoll mit dem X-Server kommuniziert und so über den X-Server alle Ein- und Ausgaben abwickelt.

Der X-Server ist somit der geräteabhängige, der X-Client der geräteunabhängige Teil.

Das X-Protokoll ist außerdem unter TCP/IP netzwerktauglich und somit können grafische Anwendungen auch auf einem entfernten Rechner laufen, die Ein- und Ausgaben aber auf dem X-Server des lokalen Rechners ausgeführt werden.

Window-Manager übernehmen die Verwaltung der Fenster und das Aussehen des Desktops.

Es stehen mehrere Window Manager zur Verfügung, wie der Open Look Window Manager (olwm), der Free Virtual Window Manager (fvwm) oder der Motif Window Manager (mwm).

Auch die KDE-Desktop-Umgebung verwendet einen eigenen Window-Manager (kwm) und ist zusätzlich ein vollständiges Desktop-System. Die Gnome-Desktop-Umgebung kann dagegen mit vielen Window-Managern zusammenarbeiten. Diese Desktop-Umgebungen bieten eine einheitliche Darstellung der Programme, eine Programmier-Schnittstelle, einen Dateimanager, Drag&Drop und viele Applikationen.

Start des X-Servers

Der Start des X-Servers erfolgt üblicherweise vom Display-Manager (xdm), kann aber auch vom Textmodus aus über das Kommando **startx** erfolgen. Beim Abmelden unter KDE wird der X-Server angehalten, kann aber im Notfall auch über die Tasten **Strg+Alt+Backspace** abgebrochen werden.

X- Programme

es gibt eine Vielzahl von X-Programmen, wie xterm (Terminal Fenster), xman (Online Hilfe), ... die üblicherweise im Verzeichnis „/usr/X11/bin/“ untergebracht sind.

KDE-Programme

KDE hat als Desktop-System viele eigene X-Anwendungen, wie konqueror (Dateimanager u. Browser), kwrite (Text-Editor), konsole (Konsolenfenster), kdevelop (Entwicklungsumgebung), kcalc (Rechner), korganizer (Terminplaner) bis zu koffice (Office-Paket), die sich im Verzeichnis „/opt/kde3/bin/“ befinden.

X11 im Netz (xhost)

Über das X-Protokoll können X-Anwendungen auch auf einem entfernten Rechner (remote host) laufen und die Ausgaben auf den lokalen Rechner leiten.

Dazu muß die Zugriffskontrolle am X-Server zuerst freigeschalten werden und dann am X-Client (remote host) die Ausgabe auf den X-Server gelenkt werden.

X-Server freigeben :	<i>xhost +</i>	für alle Hosts
	<i>xhost hostname</i>	für bestimmten Host
sperrern :	<i>xhost -</i>	für alle Hosts
	<i>xhost -hostname</i>	für bestimmten Host

Remote Login auf X-Client:

slogin username@hostname
*Password: ******

X-Client Display Variable auf X-Server setzen :

export DISPLAY = hostname:servernummer.displaynummer