Linux und Shell-Programmierung – Teil 1

Prof. Dr. Christian Baun

Fachhochschule Frankfurt am Main
Fachbereich Informatik und Ingenieurwissenschaften
christianbaun@fb2.fh-frankfurt.de

Heute

- Einführung in Linux
  - Eingabeaufforderung (Prompt)
  - Kommandos (Aufbau)
  - Linux/UNIX-Verzeichnisstruktur
  - Aktuelles Verzeichnis ausgeben (`pwd`) und wechseln (`cd`)
  - Verzeichnisse anlegen (`mkdir`) und löschen (`rmdir`)
  - Inhalte von Verzeichnissen ausgeben (`ls`)
  - Das Hilfesystem von Linux – die Manualseiten (`man`)
  - Verschiedene Dateiarten unter Linux
  - Leere Dateien anlegen (`touch`)
  - Dateien ausgeben und verknüpfen (`cat` und `tac`)
  - Inhalte von Dateien anzeigen (`more` und `less`)
  - Anfang (`head`) und Ende (`tail`) von Dateien anzeigen
  - Dateien kopieren (`cp`), verschieben/umbennen (`mv`) und löschen (`rm`)
  - Wildcards (`?`, `*`, `[]`, `!` und `^`)
  - Quoting
  - Dateirechte ändern (`chmod`)

Was ist Linux?

- Freies, plattformunabhängiges Mehrbenutzer-Betriebssystem
- Im September 1991 von Linus Torvalds ins Leben gerufen
- Eins der erfolgreichsten freien Softwareprojekte überhaupt
- Eins der am häufigsten portierten Systeme
- Fast komplett in C geschrieben
- Besteht aus:
  - Monolithischem Kernel
  - Viel freier Software
- Hervorragend geeignet für Forschung und Lehre (freie Software, flexibel einsetzbar, plattformunabhängig, kostenlos verfügbar, hohe Verbreitung, hohe Stabilität...)

Eingabeaufforderung (Prompt)

- Die Eingabeaufforderung ist die Markierung auf der Kommandozeile, an der Kommandozeilenbefehle eingegeben werden können
- Das Aussehen der Eingabeaufforderung ist systemabhängig
- Standard unter aktuellen Linux-Distributionen:
  - Für den Systemadministrator:

```
root@rechnernname:/verzeichnis#
```

  - Für normale Benutzer:

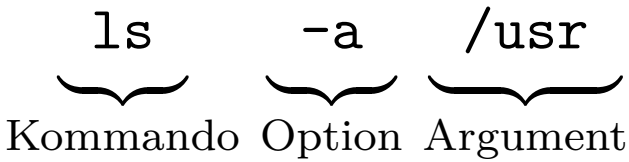
```
benutzername@rechnernname:/verzeichnis$
```

- Das Aussehen kann angepasst werden
  $\Longrightarrow$ Shellvariable `$PS1` (Primary Prompt String)

Kommandos (Aufbau)

```
Kommandoname [-Optionen] [Argumente...]
```

- Optionen werden in der Regel ein oder zwei Bindestriche vorangestellt
- Übersicht über die wichtigsten Optionen eines Kommandos mit den Optionen `--help`, `-help` oder `-h`



- Umfangreiche Dokumentationen zu (fast) jedem Kommando und seinen Optionen finden sich in den Manualseiten

```
$ man ls
```

Ein erstes Kommando – cal

- Mit dem Kommando cal wird der aktuelle Monat ausgegeben
- Optionen ändern das Verhalten von cal
- Mehr über die Optionen und Argumente des Kommandos erfahren:

```
$ cal -help
```

```
user@rechner:~$ cal
   September 2007
Mo Di Mi Do Fr Sa So
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
```

- Um alle Monate eines ganzen Jahres auszugeben, z.B. 2005:

```
$ cal 2005
```

- Um einen bestimmten Monat auszugeben, z.B. März 1979:

```
$ cal 3 1979
```

History

- Mit dem Kommando `history` hat man die Möglichkeit, eine Liste der zuletzt eingegeben Kommandos einzusehen

```
$ history
    1  ls
    2  nano Makefile
    3  make
    4  make gv
    5  cd ..
```

`history N`	Ausgabe der letzten `N` Einträge in der History
`history -c`	Den Inhalt der History löschen
`!!`	Den letzten Eintrag in der History ausführen
`!N`	Den `N`-ten Eintrag in der History ausführen
`!*`	Alle Parameter des letzten Eintrags in der History

Das Hilfesystem von Linux: Die Manualseiten (1)

- Alle Linux/UNIX-Systeme verfügen über ein ausgereiftes Hilfesystem: Die Manualseiten. Zu fast jedem Kommando gibt es in den Manualseiten eine ausführliche Beschreibung
- Aufruf von Manualseiten $\Longrightarrow$ `man`
- Die Manpage zu `<Stichwort>` aufrufen:

```
$ man <Stichwort>
```

- Alle Manualseiten zu `<Stichwort>` hintereinader aufrufen:

```
$ man -a <Stichwort>
```

- Den kompletten Pfad der Manualseite zu `<Stichwort>` anzeigen:

```
$ man -w <Stichwort>
```

Das Hilfesystem von Linux: Die Manualseiten (2)

- In den Manualseiten nach `<Stichwort>` suchen und die gefundenen Manpages auflisten:

```
$ man -f <Stichwort>
```

  $\Longrightarrow$ Identisches Ergebnis mit dem Kommando: `whatis`

- In allen Kapiteln der Manualseiten nach `<Stichwort>` suchen und die gefundenen Manualseiten auflisten:

```
$ man -k <Stichwort>
```

  $\Longrightarrow$ Identisches Ergebnis mit dem Kommando: `apropos`

- Die Manualseite zu `<Stichwort>` aus `<Kapitel>` aufrufen. z.B. `man 1 man` ruft die Manualseite zum Befehl `man` aus dem Kapitel 1 auf:

```
man <Kapitel> <Stichwort>
```

Kapitel der Manualseiten

1 Benutzerkommandos
2 Systemaufrufe. Funktionen, die vom Kernel bereitgestellt werden
3 C-Bibliotheksfunktionen. Funktionen innerhalb von Systembibliotheken
4 Gerätedateien (Devices). Üblicherweise im Verzeichnis `/dev` zu finden
5 Dateiformate, Protokolle und Konventionen. z.B. `/etc/passwd`
6 Spiele
7 Makropakete und Konventionen
8 Kommandos für die Systemadministration
9 Kernelroutinen (kein Standard!)

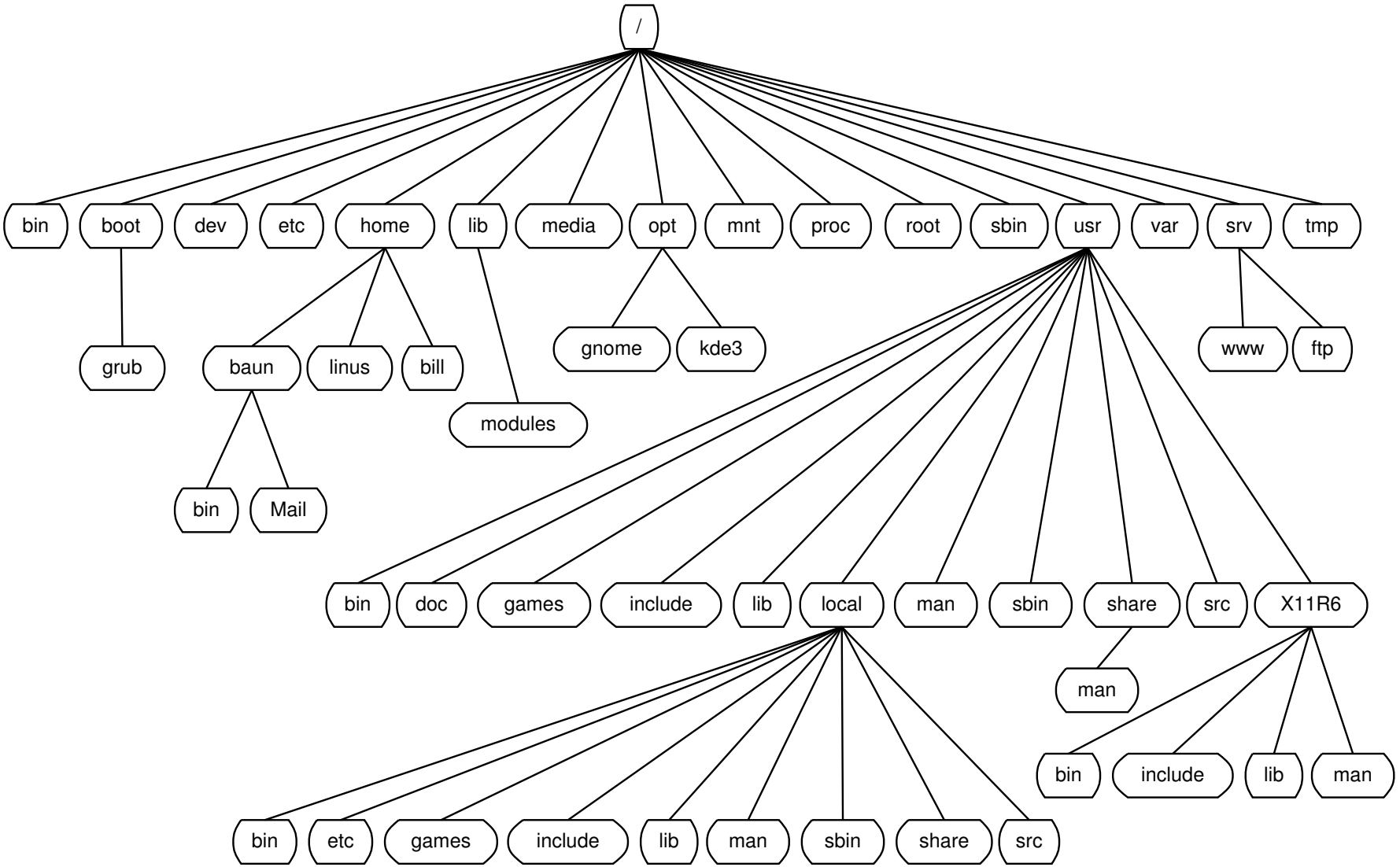
Aufbau der Manualseiten

- Manualseiten besitzen üblicherweise folgende Abschnitte. Es gibt aber keinen festen Standard für Inhalt und Reihenfolge der Abschnitte, an den sich alle halten:
  - **NAME**: Kommandoname bzw. Dateiname und eine Kurzbeschreibung
  - **SYNOPSIS** / **SYNTAX**: Aufrufsyntax. Schema der Argumente, Optionen und Parameter
  - **DESCRIPTION**: Beschreibung
  - **FILES**: Dateien, die benötigt, erzeugt oder verändert werden
  - **SEE ALSO**: Querverweise zu anderen Dokumenten und verwandte Kommandos
  - **EXAMPLE**: Beispiele zur Verwendung
  - **AUTHOR**: Angaben zum Autor
  - **BUGS**: Bekanntes Fehlverhalten und Einschränkungen

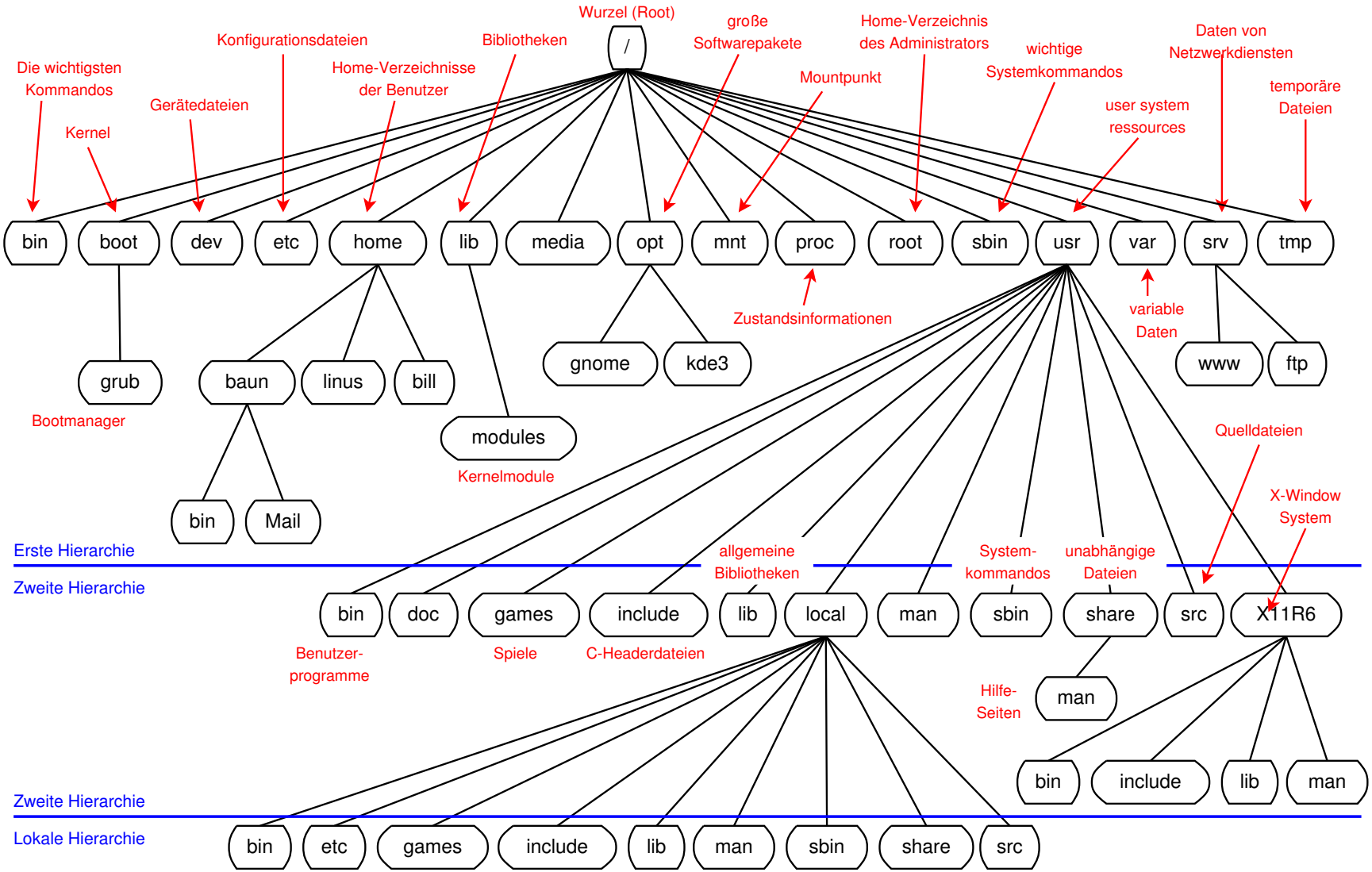
Mit Verzeichnissen arbeiten

- Durch Verzeichnisse werden Dateien hierarchisch strukturiert
- Jedes Verzeichnis kann beliebig viele Dateien und (Unter-)Verzeichnisse enthalten
- Die Struktur der Verzeichnisse ähnelt der eines Baumes
  $\Longrightarrow$ Wurzelverzeichnis /

Linux/UNIX-Verzeichnisstruktur



Die Linux/UNIX-Verzeichnisstruktur (kommentiert)



Aktuelles Verzeichnis ausgeben – `pwd`

- Das Kommando `pwd` ist eine Abkürzung für **P**resent **W**orking **D**irectory oder **P**rint **W**orking **D**irectory
- Gibt den kompletten Pfad des aktuellen Verzeichnisses aus

```
user@rechner:~$ pwd
/home/user
```

Verzeichnisse anlegen und löschen – `mkdir`, `rmdir`

```
mkdir [Verzeichnis] ...
```

- Das Kommando `mkdir` (Make Directory) erzeugt ein Verzeichnis mit dem als Argument übergebenen Verzeichnisnamen
- Es können mehrere Verzeichnisse mit einem Aufruf erzeugt werden

```
rmdir [Verzeichnis] ...
```

- Das Kommando `rmdir` (Remove Directory) löscht ein **leeres** Verzeichnis
- Nicht-leere Verzeichnisse löschen $\Longrightarrow$ `rm -r Verzeichnisname`

Einfache Beispiele zu mkdir und rmdir

```
user@rechner:~/SYS1$ mkdir testVerzeichnis
user@rechner:~/SYS1$ ls -a
.  ..  testVerzeichnis

user@rechner:~/SYS1$ rmdir testVerzeichnis
user@rechner:~/SYS1$ ls -a
.  ..

user@rechner:~/SYS1$ rmdir testVerzeichnis
rmdir: testVerzeichnis: Datei oder Verzeichnis nicht gefunden
```

Verzeichnisse wechseln – cd

`cd [Verzeichnis]`

- Wenn das Kommando `cd` (**C**hange **D**irectory) ohne Verzeichnis als Argument aufgerufen wird, wechselt `cd` in das Home-Verzeichnis des aktuellen Benutzers
- Die Tilde `~` steht für das Homeverzeichnis und kann in jedem Befehl als Synonym für das eigene Home-Verzeichnis eingesetzt werden

```
user@rechner:/tmp$ cd ~
user@rechner:~$ pwd
/home/user
```

- Jedes Verzeichnis hat die Einträge `.` und `..`
  $\Longrightarrow$ `.` verweist auf das aktuelle Verzeichnis
  $\Longrightarrow$ `..` verweist auf das übergeordnete Verzeichnis (Vaterverzeichnis)

Einige einfache Beispiele zu Verzeichniswechseln

```
user@rechner:~/SYS1$ cd .

user@rechner:~/SYS1$ cd ..

user@rechner:~$ pwd
/home/user

user@rechner:~/SYS1$ cd /

user@rechner:/$ pwd
/

user@rechner:/$ cd /usr/local/

user@rechner:/usr/local$ pwd
/usr/local
```

Verzeichnisinhalt ausgeben – `ls`

```
ls [Option] ... [Datei] ...
```

- Mit `ls` (List) kann der Inhalt von Verzeichnissen ausgegeben werden
- Das Kommando kennt sehr viele Optionen. Einige ausgewählte sind:

`-a`	*Alle* Einträge im Verzeichnis. Auch die Geheimen
`-l`	*Liste* mit Benutzer- und Gruppenrechten, Dateigrößen und Datum ausgeben
`-R`	Auch den Inhalt der Unterverzeichnisse (*rekursiv*) ausgeben
`-h`	Größenangaben in menschenlesbarem (*human readable*) Format. z.B. 23M
`-B`	Einträge, die mit ~ enden (*backups*) ignorieren
`-d`	Nur Verzeichnisse (*directories*) ausgeben. Keine Dateien
`-t`	Sortiert nach dem Zeitpunkt (*time*) der letzten Änderung
`-S`	Sortiert nach der Dateigröße (*size*)

Weitere Optionen von `ls`

`-s` Ausgabe der Dateigrößen in Blöcken.
`-m` Alle Einträge durch Kommata getrennt in einer Zeile ausgeben
`-r` Umgekehrte Reihenfolge (*reverse*) beim Sortieren
`-g` Wie `-l`, aber den Besitzer nicht auflisten
`-G` Wie `-l`, aber die *Gruppe* nicht auflisten

- Um die Lesbarkeit zu erhöhen, können unterschiedliche Arten von Dateien und Verzeichnisse mit verschiedenen Farben markiert werden

`--color=always` Verwendet immer Farben
`--color=none` Verwendet nie Farben

- Eine kurze Hilfe ausgeben:

`--help`

Einige einfache Beispiele zu Verzeichnissen

```
user@rechner:~$ pwd
/home/user
user@rechner:~$ mkdir SYS1
user@rechner:~$ cd SYS1/
user@rechner:~/SYS1$ pwd
/home/user/SYS1

user@rechner:~/SYS1$ ls -la
insgesamt 20
drwxr-xr-x   2 user user    48 2006-10-16 10:42 .
drwxr-xr-x 253 user user 20400 2006-10-16 10:42 ..

user@rechner:~/SYS1$ mkdir test
user@rechner:~/SYS1$ ls -a
.  ..  test
user@rechner:~/SYS1$ rmdir test
```

Unterschiedliche Dateiarten unter Linux/UNIX (1)

- **Reguläre Dateien** (normale Dateien) mit einer Bitfolge als Inhalt
- **Symbolische Links** sind Verweise auf eine bestehende Datei. Ein Symbolischer Link ist eine Datei, die nur den Dateinamen einer anderen Datei enthält
- **Verzeichnisse** sind Dateien, die eine Liste mit Dateinamen enthalten
- **Warteschlangen**, auch **Pipes** oder **FIFO-Dateien** genannt, sind feststehende Verbindungsleitungen zwischen verschiedenen Programmen bzw. Prozessen

Unterschiedliche Dateiarten unter Linux/UNIX (2)

- **Gerätedateien** ermöglichen den Zugriff auf Hardwaregeräte oder andere Systemkomponenten und befinden sich im Verzeichnis `/dev`
  - **Blockorientierte Geräte** sind Repräsentanten von Hardware-Geräten, die nicht einzelne Zeichen verarbeiten, sondern ganze Blocks
    $\Longrightarrow$ Festplatten, Disketten, usw.
  - **Zeichenorientierte Geräte** sind Repräsentanten von Hardware-Geräten, die nicht blockweise angesteuert werden sondern durch einzelne Bytes
    $\Longrightarrow$ Serielle oder Parallele Schnittstellen, Soundkarten, usw.
  - **Socketorientierte Geräte** machen im Prinzip Netzwerkverbindungen im Dateisystem sichtbar (und nutzbar)
    $\Longrightarrow$ Druckerwarteschlange, Syslogdeamon, usw.
  - **Virtuelle Gerätedateien** steuern kein reales Gerät an
    $\Longrightarrow$ `/dev/null`, `/dev/zero`, `/dev/random`, usw.

Eine leere Datei anlegen – touch

- Mit dem Kommando `touch` wird eine leere Datei angelegt

```
$ touch <Dateiname>
```

- Wird mit dem Kommando `touch` auf eine Datei zugegriffen, die bereits besteht, werden die Zugriffs- und Modifikationszeiten der Datei auf die aktuelle Zeit gesetzt
- Mit der Option `-t [[CC]YY]MMDDhhmm[.ss]` können Dateien auch beliebige Zugriffs- und Modifikationszeiten erhalten

Einfache Beispiele zu touch

```
user@rechner:~/SYS1$ touch test.txt

user@rechner:~/SYS1$ ls -la
insgesamt 20
drwxr-xr-x   2 user user    72 2006-10-17 08:42 .
drwxr-xr-x 253 user user 20400 2006-10-16 21:34 ..
-rw-r--r--   1 user user     0 2006-10-17 08:42 test.txt

user@rechner:~/SYS1$ touch -m -t 197905231215 test.txt

user@rechner:~/SYS1$ ls -la
insgesamt 20
drwxr-xr-x   2 user user    72 2006-10-17 08:42 .
drwxr-xr-x 253 user user 20400 2006-10-16 21:34 ..
-rw-r--r--   1 user user     0 1979-05-23 12:15 test.txt
```

Dateitypen ermitteln – `file`

`file` *[Option] ... [Datei] ...*

- Linux/UNIX verwaltet im Gegensatz zu anderen Betriebssystemen keine Dateitypen
- Dateiendungen haben unter Linux/UNIX ursprünglich keine Bedeutung und sollen dem Eigentümer nur als Merkhilfe dienen, was für Daten sich in der Datei befinden
- Mit `file` wird versucht den Inhalt von Dateien anhand ihres Inhalts zu ermitteln

```
$ file /bin/bash /etc/passwd bild.bmp bild2.jpg dokument.doc
/bin/bash:     ELF 32-bit LSB executable, Intel 80386 ...
/etc/passwd:   ASCII text
bild.bmp:      PC bitmap, Windows 3.x format, 785 x 427 x 24
bild2.jpg:     JPEG image data, EXIF standard
dokument.doc: Microsoft Office Document Microsoft Word Document
```

Hilfreiche Optionen von `file`

`-b` Den Dateinamen der untersuchten Datei nicht mit ausgeben

`-i` Anstelle der üblichen Ausgabe wird der MIME-Typ der Datei ausgegeben

```
$ file -i /bin/bash /etc/passwd bild.bmp bild2.jpg dokument.doc
/bin/bash:     application/x-executable
/etc/passwd:   text/plain charset=us-ascii
bild.bmp:      image/x-ms-bmp
bild2.jpg:     image/jpeg
dokument.doc: application/msword
```

`-L` Folgt symbolischen Links. Ermittelt den Typ der Zieldatei und nicht des Links

`-z` Versucht bei komprimierten Dateien den unkomprimierten Inhalt zu ermitteln

```
$ file linux-2.6.18.tar.bz2
linux-2.6.18.tar.bz2: bzip2 compressed data, block size = 900k
$ file -z linux-2.6.18.tar.bz2
linux-2.6.18.tar.bz2: POSIX tar archive (bzip2 compressed data, block size = 900k)
```

Dateien ausgeben und verknüpfen – cat

```
cat [Option] ... [Datei] ...
```

- Mit dem Kommando cat (*concatenate*) ist es möglich, den Inhalt von Dateien auszugeben und Dateien miteinander zu verknüpfen
- Wenn cat beim Aufruf ohne Option eine Datei übergeben wird, wird der Inhalt der Datei in der Standardausgabe ausgegeben
- Mit der Option -n (*number*) wird vor jeder von cat vor jeder Zeile die Zeilennummer ausgegeben

`cat Datei`	Gibt den Inhalt von `Datei` aus
`cat Datei1 Datei2 > Alles`	Fügt mehrere Dateien zur Datei `Alles` zusammen
`cat Datei1 >> Datei2`	Hängt `Datei1` an `Datei2` an
`cat > Datei`	Erzeugt eine neue Datei. Eingabe beenden: `Strg-C`
`cat > Datei << ENDE`	Erzeugt eine neue Datei. Eingabe beenden: `ENDE`

Dateien rückwärts ausgeben und verknüpfen – tac

```
tac [Option] ... [Datei] ...
```

- Mit dem Kommando tac ist es möglich, den Inhalt von Dateien rückwärts auszugeben und Dateien miteinander zu verknüpfen
- Wenn tac beim Aufruf ohne Option eine Datei übergeben wird, wird der Inhalt der Datei in der Standardausgabe rückwärts ausgegeben

```
$ cat testdatei.txt
Zeile 1
Zeile 2
Zeile 3
Zeile 4
Zeile 5
```

```
$ tac testdatei.txt
Zeile 5
Zeile 4
Zeile 3
Zeile 2
Zeile 1
```

Inhalt einer Datei anzeigen – more und less

```
more [Option] ... [Datei] ...
less [Option] ... [Datei] ...
```

- more und less sind **Pager**, mit denen man durch Dateien blättert
- less hat eine höhrere Funktionalität als more

SPACE	eine Bildschirmseite weiter blättern
RETURN	eine Zeile weiter blättern
b	eine Bildschirmseite zurück blättern
q	den Pager beenden
=	gibt die aktuelle Zeilennummer aus (nur more)
:n	nächste Datei in der Dateiliste öffnen (nur less)
.p	vorherige Datei in der Dateiliste öffnen (nur less)
:x	erste Datei in der Dateiliste öffnen (nur less)

Ende von Dateien anzeigen – `tail`

```
tail [Option] ... [Datei] ...
```

- Das Kommando `tail` gibt die **letzten** 10 Zeilen einer oder mehrerer Dateien auf der Standardausgabe aus. Einen anderen Wert als 10 Zeilen, kann man mit der Option `-n <Zeilen>` festlegen

```
$ tail -n 5 dateiname
```

- Das Kommando eignet sich gut zum Verfolgen von Log-Dateien
- Durch die Option `-f` (*follow*) werden neue Einträge ausgegeben. z.B.:

```
$ tail -f /var/log/messages
```

- Werden dem Kommando mehrere Dateien übergeben, wird zur besseren Orientierung vor der Ausgabe jeder Datei eine Kopfzeile ausgegeben. Diese Kopfzeile kann mit der Option `-q` (*quiet*) unterdrückt werden
- Mit der Option `-v` (*verbose*) wird auch bei nur einer Datei eine Kopfzeile ausgegeben

Anfang von Dateien anzeigen – head

```
head [Option] ... [Datei] ...
```

- Das Kommando `head` gibt die **ersten** 10 Zeilen einer oder mehrerer Dateien auf der Standardausgabe aus
- Einen anderen Wert als 10 Zeilen, kann man mit der Option `-n <Zeilen>` festlegen

```
$ head -n 5 dateiname
```

- Werden dem Kommando mehrere Dateien übergeben, wird zur besseren Orientierung vor der Ausgabe jeder Datei eine Kopfzeile ausgegeben. Diese Kopfzeile kann mit der Option `-q` (*quiet*) unterdrückt werden
- Mit der Option `-v` (*verbose*) wird auch bei nur einer Datei eine Kopfzeile ausgegeben

Dateien kopieren – cp

```
cp [Option] ... [Datei1] [Datei2]
cp [Option] ... [Datei] ... [Verzeichnis]
```

- Das Kommando cp (*copy*) kopiert eine oder mehrere Dateien und Verzeichnisse
- Kopiert `~/SYS1/Datei1.text` nach `/tmp/Datei2.txt`

```
$ cp ~/SYS1/Datei1.txt /tmp/Datei2.txt
```

`-f` Überschreibt existierende Zieldateien ohne Warnung (*force*)
`-b` Erzeugt Sicherheitskopien von Dateien, die sonst überschrieben würden (*backup*)
`-i` Verlangt vor überschreiben einer existierende Zieldateien eine Bestätigung.
`-r` Kopiert Dateien und Unterverzeichnisse samt Inhalt rekursiv
`-v` Gibt den Namen jeder Datei aus, die kopiert wird (*verbose*)

Dateien verschieben/umbennen – mv

```
mv [Option] ... [Quelle] ... [Ziel]
```

- Das Kommando mv (*move*) verschiebt eine oder mehrere Dateien und Verzeichnisse bzw. benennt sie um
- Verschiebt ~/SYS1/Datei1.text nach /tmp/Datei2.txt

```
$ mv ~/SYS1/Datei1.txt /tmp/Datei2.txt
```

-f	Überschreibt existierende Zieldateien ohne Warnung (*force*)
-b	Erzeugt Sicherheitskopien von Dateien, die sonst überschrieben würden (*backup*)
-i	Verlangt vor dem überschreiben existierender Zieldateien eine Bestätigung
-v	Gibt den Namen jeder Datei aus, die kopiert wird (*verbose*)
-u	Verschiebt Dateien nur, wenn Sie neuer sind als die Gleichnamigen (*update*)

Dateien löschen – rm

```
rm [Option] ... [Datei] ...
```

- Das Kommando `rm` (*remove*) löscht eine oder mehrere Dateien und Verzeichnisse
- löscht `Datei1` und `Datei2`

```
$ rm Datei1 datei2
```

- löscht das Verzeichnis `TestVerzeichnis` und seinen Inhalt rekursiv

```
$ rm -rf TestVerzeichnis
```

`-f` Löscht schreibgeschützte Dateien ohne ohne Warnung (*force*).
`-r` Löscht Dateien und Unterverzeichnisse samt Inhalt rekursiv
`-i` Verlangt vor jeder Löschung eine Bestätigung (*interactive*)
`-v` Gibt den Namen jeder Datei aus, die gelöscht wird (*verbose*)

Einfache Beispiele zu rm (1)

```
user@rechner:~/SYS1$ ls

user@rechner:~/SYS1$ mkdir testVerzeichnis

user@rechner:~/SYS1$ touch testVerzeichnis/testDatei

user@rechner:~/SYS1$ rm -rf testVerzeichnis

user@rechner:~/SYS1$ ls -la
insgesamt 20
drwxr-xr-x   2 user user    48 2007-03-22 12:24 .
drwxr-xr-x 259 user user 20120 2007-03-22 00:23 ..
```

Einfache Beispiele zu rm (2)

```
user@rechner:~/SYS1$ mkdir testVerzeichnis

user@rechner:~/SYS1$ touch testVerzeichnis/testDatei

user@rechner:~/SYS1$ rm testVerzeichnis
rm: Entfernen von »testVerzeichnis« nicht möglich:
    Ist ein Verzeichnis

user@rechner:~/SYS1$ rmdir testVerzeichnis
rmdir: testVerzeichnis: Das Verzeichnis ist nicht leer

user@rechner:~/SYS1$ ls
testVerzeichnis
```

Wildcards (Auswahl)

`?`	Ersetzt ein beliebiges Zeichen
`*`	Ersetzt beliebig viele oder Null Zeichen
`abc*`	Beginnt mit `abc`, danach beliebig viele Zeichen
`.*`	Das erste Zeichen muss ein Punkt sein. Danach ist egal
`*abc*`	Es muss mindestens die Zeichenfolge `abc` enthalten sein
`abc*xyz`	Fängt mit `abc` an und hört mit `xyz` auf
`[xyz]`	Platzhalter für eines der drei Zeichen in den eckigen Klammern
`[abc]de`	Kann sein: `ade`, `bde` oder `cde`
`[a-z]`	Genau ein Zeichen aus dem Bereich der Kleinbuchstaben `a` bis `z`
`[a-z]x`	Kann sein: `ax`, `bx`, `cx`, `dx` ... `zx`
`[A-Za-z]xyz`	Kann sein: `Axyz` ... `Zxyz`, `axyz` ... `zxyz`
`[!x]`	Nicht `x` an dieser Stelle
`[^x]`	Nicht `x` an dieser Stelle
`[!abc]`	Nicht `a`, `b` oder `c` an dieser Stelle

Quoting

- Der Inhalt von einfachen Anführungszeichen wird auf der Shell wörtlich genommen und nicht interpretiert:

```
$ echo 'Der Inhalt der Variable HOME ist $HOME'
Der Inhalt der Variable HOME ist $HOME
```

- Variablen zwischen doppelten Anführungszeichen werden ausgewertet:

```
$ echo "Der Inhalt der Variable HOME ist $HOME"
Der Inhalt der Variable HOME ist /home/benutzername
```

- Bei Backquotes wird der Inhalt als Kommando betrachtet und durch die Standardausgabe des Kommandos interpretiert:

```
$ /usr/bin/whoami
benutzername
$ echo Der Benutzername ist `/usr/bin/whoami`
Der Benutzername ist benutzername
```

Dateirechte (1/2)

Dateityp	Besitzer			Gruppe			Andere		
-/d/l	r	w	x	r	w	x	r	w	x
	4	2	1	4	2	1	4	2	1

- Dateitypen:
  - `-` $\Longrightarrow$ Datei
  - `d` $\Longrightarrow$ Verzeichnis
  - `l` $\Longrightarrow$ symbolischer Link (Verweis)
  - `b` $\Longrightarrow$ Blockorientiertes Gerät (Device)
  - `c` $\Longrightarrow$ Zeichenorientiertes Gerät
  - `p` $\Longrightarrow$ FIFO-Datei (named pipe)
  - `s` $\Longrightarrow$ UNIX domain socket
- Rechtebits:
  - `r` $\Longrightarrow$ lesender Zugriff erlaubt
  - `w` $\Longrightarrow$ schreibender Zugriff erlaubt
  - `x` $\Longrightarrow$ Datei darf ausgeführt werden
- Kommando zum Ändern der Dateirechte $\Longrightarrow$ `chmod`

Dateirechte (2/2)

- Beispiele:

bel. Ausgangslage	$\Longrightarrow$	`chmod 000 <dateiname>`	$\Longrightarrow$	`----------`
bel. Ausgangslage	$\Longrightarrow$	`chmod 644 <dateiname>`	$\Longrightarrow$	`-rw-r--r--`
bel. Ausgangslage	$\Longrightarrow$	`chmod 666 <dateiname>`	$\Longrightarrow$	`-rw-rw-rw-`
bel. Ausgangslage	$\Longrightarrow$	`chmod 744 <dateiname>`	$\Longrightarrow$	`-rwxr--r--`
bel. Ausgangslage	$\Longrightarrow$	`chmod 755 <dateiname>`	$\Longrightarrow$	`-rwxr-xr-x`
bel. Ausgangslage	$\Longrightarrow$	`chmod 777 <dateiname>`	$\Longrightarrow$	`-rwxrwxrwx`
`----------`	$\Longrightarrow$	`chmod a+r <dateiname>`	$\Longrightarrow$	`-r--r--r--`
`----------`	$\Longrightarrow$	`chmod a+rwx <dateiname>`	$\Longrightarrow$	`-rwxrwxrwx`
`----------`	$\Longrightarrow$	`chmod u+rwx <dateiname>`	$\Longrightarrow$	`-rwx------`
`----------`	$\Longrightarrow$	`chmod g+rwx <dateiname>`	$\Longrightarrow$	`----rwx---`
`----------`	$\Longrightarrow$	`chmod o+rwx <dateiname>`	$\Longrightarrow$	`-------rwx`
`-rwxrwxrwx`	$\Longrightarrow$	`chmod a-rwx <dateiname>`	$\Longrightarrow$	`----------`

Spezielle Zugriffsrechte – Das t-Bit

- Zu den speziellen Zugriffsrechten gehört das **t-Bit** (**Sticky Bit**)

```
user@rechner:~$ ls -ld /tmp/
drwxrwxrwt 18 root root 1080 2007-04-24 09:16 /tmp/
```

- Das t-Bit steht für **save program text on swap device** und kann auf Dateien und Verzeichnisse angewendet werden
- Das t-Bit bewirkt, dass alle Daten, die in dieses Verzeichnis geschrieben werden sollen, so lange wie möglich im Hauptspeicher oder in der Swap-Partition existieren sollen
  - Die Daten sollen nach Möglichkeit erst beim Herunterfahren des Systems tatsächlich auf die Festplatte zurückgeschrieben werden
- Besonders bei temporären Daten kann das zu Performancesteigerungen führen, da zeitaufwändige Schreiboperationen vermieden werden
- In Verzeichnissen mit dem t-Bit darf ein Benutzer immer nur seine eigenen Daten umbenennen oder löschen

Spezielle Zugriffsrechte – Das s-Bit

- Das **s-Bit** steht für **set user or group ID on execution**

```
user@rechner:/usr/bin$ ls -l passwd
-rwsr-xr-x 1 root root 31736 2008-07-26 19:24 passwd
```

- Das s-Bit bewirkt, dass ein Programm beim Aufruf mit den Rechten des Eigentümers gestartet wird
- Achtung! Das s-Bit unüberlegt einsetzen, kann extrem gefährlich sein
  - Wenn ein Programm dem Systemadministrator `root` gehört, hat das Programm bei seiner Ausführung hier auch immer die Rechte des Systemadministrators
- Ist das **s-Bit** nicht beim Besitzer, sondern bei der Gruppe gesetzt, steht es für **set group ID on execution**
  - Das s-Bit bei den Gruppenrechten bewirkt, dass ein Programm beim Aufruf mit den Rechten der Gruppe gestartet wird
  - Auch hier muss man vorsichtig sein, denn das s-Bit stellt auch bei den Gruppenrechten ein Sicherheitsrisiko dar

Spezielle Zugriffsrechte ändern (1/2)

- Beispiele:

bel. Ausgangslage	$\Longrightarrow$	`chmod 1000 <dateiname>`	$\Longrightarrow$	`----------T`
bel. Ausgangslage	$\Longrightarrow$	`chmod 1666 <dateiname>`	$\Longrightarrow$	`-rw-rw-rwT`
bel. Ausgangslage	$\Longrightarrow$	`chmod 1777 <dateiname>`	$\Longrightarrow$	`-rwxrwxrwt`
bel. Ausgangslage	$\Longrightarrow$	`chmod 2000 <dateiname>`	$\Longrightarrow$	`------S---`
bel. Ausgangslage	$\Longrightarrow$	`chmod 2666 <dateiname>`	$\Longrightarrow$	`-rw-rwSrw-`
bel. Ausgangslage	$\Longrightarrow$	`chmod 2777 <dateiname>`	$\Longrightarrow$	`-rwxrwsrwx`
bel. Ausgangslage	$\Longrightarrow$	`chmod 4000 <dateiname>`	$\Longrightarrow$	`---S------`
bel. Ausgangslage	$\Longrightarrow$	`chmod 4666 <dateiname>`	$\Longrightarrow$	`-rwSrw-rw-`
bel. Ausgangslage	$\Longrightarrow$	`chmod 4777 <dateiname>`	$\Longrightarrow$	`-rwsrwxrwx`

Spezielle Zugriffsrechte ändern (2/2)

- Beispiele:

`----------`	$\Longrightarrow$	`chmod u+s <dateiname>`	$\Longrightarrow$	`---S------`
`-rw-rw-rw-`	$\Longrightarrow$	`chmod u+s <dateiname>`	$\Longrightarrow$	`-rwSrw-rw-`
`-rwxrwsrwx`	$\Longrightarrow$	`chmod u+s <dateiname>`	$\Longrightarrow$	`-rwsrwxrwx`
`----------`	$\Longrightarrow$	`chmod g+s <dateiname>`	$\Longrightarrow$	`------S---`
`-rw-rw-rw-`	$\Longrightarrow$	`chmod g+s <dateiname>`	$\Longrightarrow$	`-rw-rwSrw-`
`-rwxrwxrwx`	$\Longrightarrow$	`chmod g+s <dateiname>`	$\Longrightarrow$	`-rwxrwsrwx`
`----------`	$\Longrightarrow$	`chmod o+t <dateiname>`	$\Longrightarrow$	`---------T`
`-rw-rw-rw-`	$\Longrightarrow$	`chmod o+t <dateiname>`	$\Longrightarrow$	`-rw-rw-rwT`
`-rwxrwxrwx`	$\Longrightarrow$	`chmod o+t <dateiname>`	$\Longrightarrow$	`-rwxrwxrwt`