Einführung in Linux

Frank Schuh Manfred Stettner

Fachbereich 12, Elektrotechnik und Informatik

22. - 25. September 2008

Gliederung

Wesentliche Stadien und Versionen der UNIX/Linux-Entwicklung

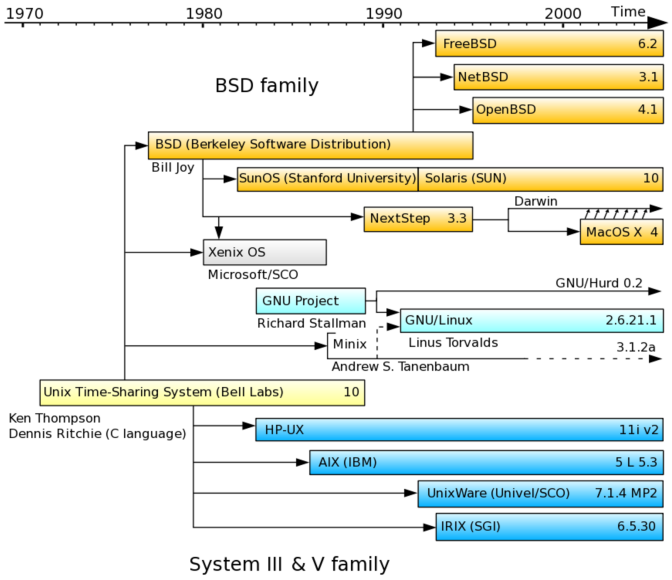
- 1969-1971: Auf einer PDP-7 wurde die erste Version von UNIX in Assembler geschrieben.
- 1973: UNIX wird in die Programmiersprache C umgeschrieben. UNIX konnte nun mit geringem Aufwand auf all jene Rechner portiert werden, die über einen C-Compiler verfügten.
- 1976-1982: UNIX spaltet sich in verschiedene Weiterentwicklungen
  - kommerziell: System V (AT&T), konservativ, solide, kommerzielle Produkte
  - frei: BSD (Berkely Software Distribution), fortschrittlicher und mutiger vorangetrieben durch die Universität von Kalifornien in Berkeley
- 1983: Richard Stallman gründet das GNU-Projekt mit dem Ziel, ein freies Betriebssystem zu erschaffen.
- 1987: UNIX auf Intel 386.


```
From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: What would you like to see most in minix?
Summary: small poll for my new operating system
Message-ID: <1991Aug25.205708.9541@klaava.Helsinki.FI>
Date: 25 Aug 91 20:57:08 GMT
Organization: University of Helsinki


Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and
professional like gnu) for 386(486) AT clones.  This has been brewing
since april, and is starting to get ready.  I'd like any feedback on
things people like/dislike in minix, as my OS resembles it somewhat
(same physical layout of the file-system (due to practical reasons)
among other things).

I've currently ported bash(1.08) and gcc(1.40), and things seem to work.
This implies that I'll get something practical within a few months, and
I'd like to know what features most people would want.  Any suggestions
are welcome, but I won't promise I'll implement them :-)

                Linus (torvalds@kruuna.helsinki.fi)

PS.  Yes - it's free of any minix code, and it has a multi-threaded fs.
It is NOT protable (uses 386 task switching etc), and it probably never
will support anything other than AT-harddisks, as that's all I have :-(.
```

- September 1991: Linux in der Version 0.01 von Linus Torvalds veröffentlicht.

- 1992: Der Linux-Kernel wird unter der GNU GPL vertrieben und es entstehen die ersten freien Linux-Distributionen.
- 1994: Linux in der Version 1.0 veröffentlicht. Der veröffentlichte Kernel ist erstmals netzwerkfähig.
- 2003: Ende des Jahres wird der Kernel 2.6 freigegeben.

Die Unix Timeline steht unter http://www.levenez.com/unix/



Freie Software

- Freie Software ist Software, die mit der Erlaubnis für jeden verbunden ist, sie zu benutzen, zu kopieren und zu verbreiten, entweder unverändert oder verändert, entweder gratis oder gegen ein Entgelt. Im Besonderen bedeutet das, daß der Quellcode verfügbar sein muss. "Wenn es kein Quelltext ist, ist es keine Software."

Open Source

- Quelloffene Software d.h. der Quelltext muss auch offen für Bearbeitung und Weiterverbreitung sein. Die GNU GPL (General Public License, generelle öffentliche Lizenz) ist ein spezifischer Satz an Vertriebsbedingungen, um ein Programm unter Copyleft zu stellen. Copyleft ist freie Software, bedeutet, daß jede Kopie dieser Software, sogar wenn sie verändert wurde, freie Software bleiben muss.

http://www.gnu.org/philosophy/categories.de.html

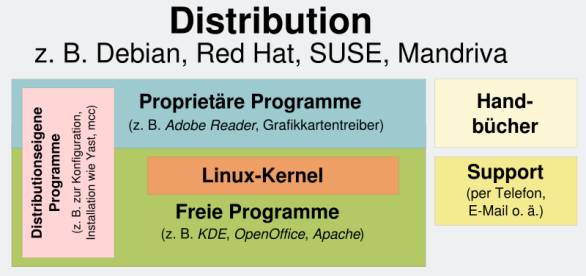
Software wird damit als frei bezeichnet, wenn die Lizenz, unter der sie steht, folgende vier definierte Rechte einräumt:

- Die Freiheit, das Programm zu jedem Zweck auszuführen
- Die Freiheit, das Programm zu studieren und zu verändern
- Die Freiheit, das Programm zu kopieren
- Die Freiheit, das Programm zu verbessern und zu verbreiten, um damit einen Nutzen für die Gemeinschaft zu erzeugen.

Einige bekannte Open Source Projekte neben dem Linuxkernel sind zum Beispiel:

- GCC - die GNU Compiler Kollektion. Mit dem Compiler läßt sich Code der u.a. in C, C++, Java geschrieben wurde in lauffähige Programme übersetzen.
- OpenOffice - ist eine Office Suite, die auf vielen Betriebssystemen und in zahlreichen Sprachen verfügbar ist.
- Mozilla Firefox/Thunderbird/Sunbird - Webbroser, Mailclient und Kalendertool (Mehrfachlizenzierung:MPL/GPL/LGPL)
- Audacity - ein freier Audioeditor und -rekorder. Digitalisieren von analogen Quellen, konvertieren in verschiedene Audioformate. Diese schneiden, mischen und kopieren gehören zu den Stärken von Audacity.
- Eine der weltweit größten Sammlungen an Open Source Softwareentwicklungen findet man unter http://sourceforge.net/

Da Linux im Prinzip nur aus dem Betriebssystemkern besteht, bedarf es weiterer Software um auch damit arbeiten zu können. Hierzu gibt es zahlreiche kommerzielle und freie Linux-Distributionen.
(http://futurist.se/gldt/)

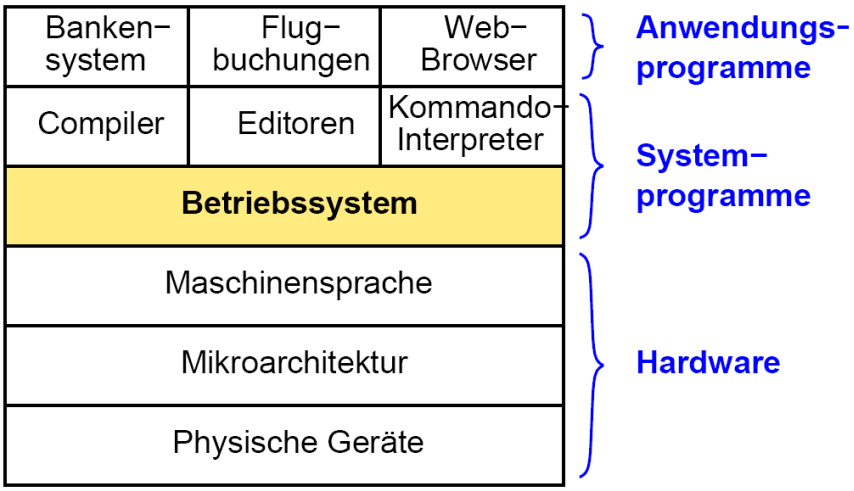


- Red Hat, Novel SUSE, Mandriva, Xandros... (kommerzielle Anbieter, teilw. kostenlos)
- Debian, Fedora, Ubuntu, Knoppix, openSUSE... (freie populäre Distributionen)

Welche Distribution passt zu mir?

http://www.zegeniestudios.net/ldc/

Einordnung in ein Rechnersystem



- Ein Betriebssystem ist Mittler zwischen Anwendung und Hardware

Was leistet ein Betriebssystem?

- **Erweiterung der Hardware**
- **Abkürzung der Hardware**
- **Verwaltung der Ressourcen**

Abkürzung: BS = Betriebssystem

Erweiterung der Hardware

- Hardware muß billig, schnell und zuverlässig sein
  - Beschränkung auf das absolut notwendige
  - Folge: schwierige, aufwändige Programmierung
- BS stellt Anwendungsprogrammen komplexe Funktionen bereit
  - Beispiel: Schreiben auf eine Festplatte
    - BS findet automatisch freie Sektoren und legt Verwaltungsinformationen an.
    - die interne Struktur der Platte (Anzahl der Köpfe, Zylinder, Sektoren, etc.) ist für die Anwendung nicht mehr wichtig
  - Folge: erhebliche Vereinfachung der Programmierung

Abstraktion der Hardware

- Problem: Rechner sind trotz ähnlicher Architektur im Detail unterschiedlich aufgebaut:
  - Einteilung des Adressraums (Speicher, E/A-Controller)
  - zahlreiche E/A-Controller und E/A-Geräte z.B. Southbridges, Festplatten
- Lösung: BS realisiert einheitliche Sicht für Anwendungen
  - Beispiel: Dateien abstrahieren externen Speicher
    d.h. für Anwendungen ist kein Unterschied zwischen Festplatte, Diskette, CD-ROM, USB-Stick, Netzwerklaufwerk, ...
    Unix/Linux: selbst Drucker werden wie Dateien behandelt
  - Konkret: Anwendungen nutzen Systemaufrufe (Syscalls)
- BS realisiert eine virtuelle Maschine für den Benutzer und seine Anwendung

Verwaltung der Ressourcen

- Ressource (Betriebsmittel) ist alles, was eine Anwendung zur Ausführung braucht
  - Prozessor, Speicher, Geräte, (Festplatte, Netzwerk ...)
- Früher: auf einem Rechner lief zu jedem Zeitpunkt nur eine Anwendung eines Benutzers
- Heute: Rechner im Mehrprozeß- und Mehrbenutzerbetrieb
  - mehrere Anwendungen verschiedener Nutzer werden „gleichzeitig“ ausgeführt (Multiuser, Multitasking)
- Notwendig:
  - Fairness: „gerechte“ Verteilung der Ressourcen
  - Sicherheit: Schutz der Anwendungen und Benutzer voreinander

Verwaltung der Ressourcen

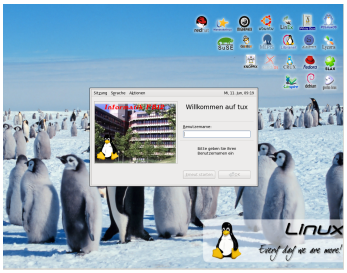
- Beispiel: Dateien
  - jeder Datei werden Rechte zugeordnet
    - Rechte legen fest, wer auf die Datei zugreifen darf
  - BS stellt die Einhaltung dieser Rechte sicher
    - unbefugter Zugriff wird verweigert
- Beispiel: Drucker
  - während Susi druckt, will auch Frank drucken
    - aber nicht auf dasselbe Papier
  - BS regelt den Zugriff auf den Drucker
    - der Auftrag von Frank wird zurückgestellt, bis der von Susi beendet ist

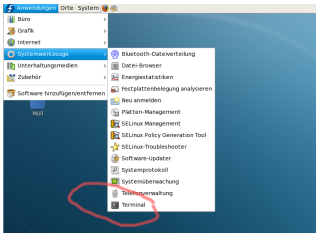
Eigenschaften von Linux

- interaktives Mehrnutzer- und Mehrprozess-System
  - Standart: Login mit Nutzerkennung und Passwort
- Benutzerschnittstellen:
  - Klassisch: Shell (textorientierter Kommandointerpreter)
  - Heute: moderne grafische Oberfläche (GUI) und Shell
- gut portabel durch die Programmiersprachen C, C++ und POSIX-Konformität
- Quellcode frei verfügbar (Kernel, Treiber, Systemprogramme etc.)
- Systemkonfiguration durch Text-Dateien - keine Registry
- Prozess-Konzept: Eltern- und Kind-Prozesse
  - Prozessbaum mit Prozess 1 (init) - Vater aller Prozesse
- Dateisystem hat Baumstruktur - die Wurzel ist /
  - Teilbäume können im Betrieb ein- oder ausgehängt werden
  - spezielle Dateien für Verzeichnisse, Geräte, Links, Prozesse usw.

Anmelden am System

- Der Systemverwalter/admin *'root'* und der *'normale User'*
- Grafischer Anmeldebildschirm:
  Mit der Eingabe des Benutzernamens (login) und dem zugehörigen Passwort kann man sich am Rechner anmelden.
- Terminal starten:
  `Anwendungen->Systemwerkzeuge->Terminal`

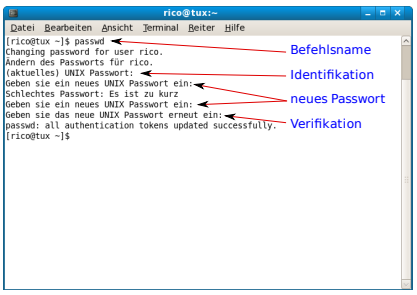




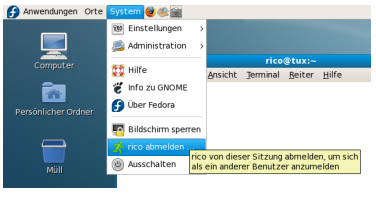
Ein Terminal ist **die** Kommunikationsschnittstelle mit dem Computer.

Passwort ändern und abmelden

Mit dem Befehl *passwd* fordert das System den Benutzer auf, sein Passwort zu ändern. Hierbei ist es wichtig ein 'gutes' Passwort zu wählen.



- 8 Zeichen, Sonderzeichen
- Kein Zusammenhang zum Benutzer, nicht aus Wörterbuch
- Passwort sorgfältig geheimhalten
- Passwort regelmäßig ändern



Vom Rechner Abmelden:

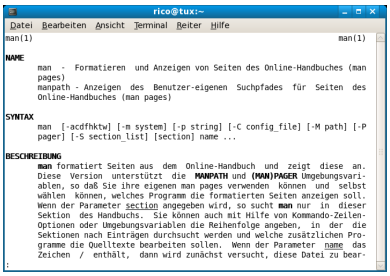
`Systemwerkzeuge-> <USER> abmelden`

Die *manpage*

Die Programme und Befehle unter Linux sind zumeist extrem vielfältig und mächtig. Man muß öfter schon mal nachschauen, welche Bedeutung die ein oder andere Option hat oder welche Anwendung nun gerade für seine Zwecke die richtige ist.

Die Handbuchseiten oder *manpage*:

`man CMD`	Beschreibung des Kommandos `CMD`
`man -f CMD`	Einzeilige `man`-Bescheibung
`man -k TEXT`	Einzeilige `man`-Bescheibung mit `TEXT` darin ausgeben
`CMD --help`	Kurzbeschreibung von `CMD`



Ein grafisches Hilfesystem ist im Menue unter `System->Hilfe` zu finden.

Die Linux Shell

- Die Shell dient als grundlegende interaktive Schnittstelle zwischen dem Benutzer und dem Betriebssystem.
- Shells sind textbasierende Programme, die in einem Terminal ablaufen.
- Aus der Shell können Kommandos abgesetzt und Programme aufgerufen werden.
- Shells sind vollständige Programmiersprachen, für große Programme allerdings nicht sehr effizient
- Es gibt eine Vielzahl von Shells.

Die gebräuchlichsten Shellvarianten

- **(Ba)sh** - Bourne-Again-Shell
- **(T)csh** - Erweiterte Form der C-Shell (**csh**)
- **Ksh** - Korn-Shell

Arbeiten mit der Bash

Einige nützliche Funktionen der *bash*

- Autocompletion; Automatisches vervollständigen von Dateinamen mit der `TAB`-Taste
- Editieren der Kommandozeile
  History; Rückgriff auf früher eingegebene Befehle
  Cursor-Tasten auf, ab, suchen mit `STRG-r`
- Umleiten von Ein- und Ausgabe; `> < |`
- Platzhalterzeichen; um Dateinamen abzukürzen; `* ? []`
- Jobverwaltung; bg, fg, jobs...
- Programm/Kommandoaufrufe per Script; `#!/bin/bash`

Beispiel

```
[rico@tux work]$ pdf<TAB><TAB>
pdf2dsc      pdfcrop      pdflatex     pdftex       pdfxtex
...

[rico@tux work]$ <STRG r>
(reverse-i-search)`ca': cat adressen

[rico@tux work]$ ls
adressen  bar.04  bar.08  bar.12  bar.16  bar.20
bar.01    bar.05  bar.09  bar.13  bar.17  link-zu-adressen
bar.02    bar.06  bar.10  bar.14  bar.18
bar.03    bar.07  bar.11  bar.15  bar.19
[rico@tux work]$ rm bar.0*
[rico@tux work]$ ls
adressen  bar.11  bar.13  bar.15  bar.17  bar.19  link-zu-adressen
bar.10    bar.12  bar.14  bar.16  bar.18  bar.20

[rico@tux work]$ grep Herr adressen > Herren
[rico@tux work]$ head Herren
1:Albert:Herr:Andrzejewski:34117:Kassel:Hessen:Ackerstr.:2
2:Alex:Herr:Arndt:34134:Kassel:Hessen:Ahornstr.:90
3:Andre:Herr:Barske:34266:Niestetal:Hessen:Antoniusstr.:69
7:Antonio:Herr:Berg:34474:Diemelstadt:Hessen:Bauhofstr.:38
...
[rico@tux work]$ cat adressen|grep Herr > Herren
```

Eingabe von Kommandos

Oft müssen den Kommandos weitere Informationen übergeben werden.

Kommandosyntax

Kommandoname [-Optionen] [Argumente]

- Optionen bestehen in der Regel aus Buchstaben und ggf. weiteren Parametern durch vorangestelltes Minuszeichen.
- Argumente sind z.B Dateinamen.
- Die eckigen Klammern zeigen an, dass Optionen und Argumente optional sind.

Beispiel

```
ls -l foo
cp -a foo bak
```


Dateioperationen

ls - **li**s**t** - zeigt den Verzeichnisinhalt an
ls [OPTION]... [DATEI]...

- Ohne Optionen stellt `ls` lediglich die Dateinamen in einer Liste dar.
- `ls -l` (long); gibt eine ausführliche Liste aus.
- `ls -a` zeigt auch mit Punkt beginnende Dateien (versteckte Dateien) an.

Beispiel

```
[rico@tux ~]$ ls -al
insgesamt 184
drwx------ 28 rico user 4096 20. Jun 07:48 .
drwxr-xr-x  3 root root 4096 12. Jun 13:43 ..
-rw-r--r--  1 rico user 6699 20. Jun 07:48 adressen
-rw-------  1 rico user  134 19. Jun 14:49 .bash_history
-rw-r--r--  1 rico user   33 15. Jan 15:30 .bash_logout
...
```

cp - **co**p**y** - kopiert Dateien und Verzeichnisse
cp [OPTION]... QUELLE ZIEL

- `cp` legt Kopien von ein oder mehreren Dateien an. Falls das Ziel ein Verzeichnis ist, können mehrere Dateien in dieses Directory unter Beibehaltung ihres Namens kopiert werden.
- `cp -p` Die neue Datei erhält die Rechte und den Zeitstempel der alten Datei.
- `cp -r` Falls die Quelle ein Verzeichnis ist, werden rekursiv alle Dateien und Unterverzeichnisse kopiert.

Beispiel

```
[rico@tux ~]$ cp -p adressen adressen2
[rico@tux ~]$ ls -l adressen adressen2
-rw-r--r-- 1 rico user 6699 20. Jun 07:48 adressen
-rw-r--r-- 1 rico user 6699 20. Jun 07:48 adressen2
```


mv - **m**o**v**e - eine Datei verschieben oder umbenennen
mv [OPTION]... QUELLE ZIEL

- Innerhalb eines Filesystems werden keine Daten bewegt, sondern nur Dateien und Verzeichnisse umbenannt.
- `mv -i` Eine existierende Datei wird, durch eine Abfrage, vor dem überschreiben geschützt.

Beispiel

```
[rico@tux work]$ ls -l
insgesamt 8
-rw-r--r-- 1 rico user 6699 20. Jun 07:48 adressen
[rico@tux work]$ mv adressen adressen1
[rico@tux work]$ ls -l
insgesamt 8
-rw-r--r-- 1 rico user 6699 20. Jun 07:48 adressen1
```

rm - **rem**ove - löscht Dateien oder Verzeichnisse
rm [OPTION]... DATEI...

- `rm` ist unter UNIX der übliche Befehl zum Löschen von Dateien und Verzeichnissen.
- `rm -i` löscht nach Rückfragen (interaktiv).
- `rm -r` löscht ein vollständiges Verzeichnis rekursiv.

Vorsicht: Dateien sind nicht wieder herzustellen

Beispiel

```
[rico@tux work]$ ls -l adressen adressen1
-rw-r--r-- 1 rico user 6699 20. Jun 07:48 adressen
-rw-r--r-- 1 rico user 6699 20. Jun 07:48 adressen1
[rico@tux work]$ rm -i adressen1
rm: reguläre Datei "adressen1" entfernen? y
[rico@tux work]$ ls -l
insgesamt 8
-rw-r--r-- 1 rico user 6699 20. Jun 07:48 adressen
```


ln - **l**i**n**k - setzt Verknüpfungen zwischen Dateien
ln [OPTION]... ZIEL [VERKNÜPFUNGSNAME]

- `ln` ist ein Kommando zum erstellen von Dateiverknüpfungen. Ohne Optionen werden sogenannte Hardlinks erzeugt d.h. eine Verknüpfung mit dem Inode der Originaldatei.
- `ln -s` ist die häufigste Verwendung unter Linux. Hier werden symbolische- oder Softlinks zur Originaldatei erstellt.

Beispiel

```
[rico@tux work]$ ln adressen adressen1
[rico@tux work]$ ln adressen adressen2
[rico@tux work]$ ln -s adressen adressen3
[rico@tux work]$ ls -li
insgesamt 24
65927 -rw-r--r-- 3 rico user 6699 20. Jun 07:48 adressen
65927 -rw-r--r-- 3 rico user 6699 20. Jun 07:48 adressen1
65927 -rw-r--r-- 3 rico user 6699 20. Jun 07:48 adressen2
65930 lrwxrwxrwx 1 rico user    8 20. Jun 13:59 adressen3 -> adressen
```

Verzeichnisse

mkdir - **m**a**k**e **dir**ectory - erstelle Verzeichnis
mkdir [OPTION] VERZEICHNIS...

- `mkdir` legt ein neues Verzeichnis an, wenn es noch nicht existiert.

rmdir - **r**e**m**ove **dir**ectory - löscht leere Verzeichnisse
rmdir [OPTION]... VERZEICHNIS...

- `rmdir` löscht Verzeichnisse, wenn diese leer sind.

Beispiel

```
[rico@tux work]$ mkdir foo
[rico@tux work]$ ls -l
insgesamt 12
-rw-r--r-- 1 rico user 6699 20. Jun 07:48 adressen
drwxr-xr-x 2 rico user 4096 20. Jun 14:58 foo
[rico@tux work]$ rmdir foo
[rico@tux work]$ ls -l
insgesamt 8
-rw-r--r-- 1 rico user 6699 20. Jun 07:48 adressen
```


pwd - **p**rint **w**orking **d**irectory - Den Namen des aktuellen Arbeitsverzeichnisses ausgeben

cd - **c**hange **d**irectory - wechselt das aktuelle Verzeichnis
cd [OPTION] [VERZEICHNIS]

- `cd` oder `cd ~` ohne Argument, wechselt ins Heimatverzeichnis
- `cd -` wechselt ins vorherige Verzeichnis

Beispiel

```
[rico@tux work]$ cd foo
[rico@tux foo]$ pwd
/home/rico/work/foo
[rico@tux foo]$ cd
[rico@tux ~]$ pwd
/home/rico
```

Ende Teil 1

Vielen Dank für Ihre Aufmerksamkeit!

Für Fragen stehen wir gerne zur Verfügung

Dateien verändern und betrachten

cat - con**cat**enate - Dateien aneinanderhängen ausgeben
cat [OPTION] [DATEI]...

- `cat` ist der einfachste Befehl zum Betrachten von einer oder mehreren Dateien. Ursprünglich zum verketten von Dateien gedacht.
- `cat -n` stellt jeder Zeile eine Zeilennummer voran

Beispiel

```
[rico@tux work]$ cat adressen
1:Albert:Herr:Andrzejewski:34117:Kassel:Hessen:Ackerstr.:2
2:Alex:Herr:Arndt:34134:Kassel:Hessen:Ahornstr.:90
...
99:Teo:Herr:Zelmanowski:45472:Muelheim:Nordrhein-Westfalen:Wiesenstr.:46
100:Waldemar:Herr:Znamenak:45529:Hattingen:Nordrhein-Westfalen:Ziegelei\
str.:82
```

less - zeigt Dateien Seitenweise an (Pager)

- Mit `less` kann man sehr gut Inhalte von Textdateien seitenweise anzeigen.
- Außerdem bietet dieser Befehl eine beeindruckend große Zahl an nützlichen Features. (s. Manpage)

Taste	Bedeutung
h, H	Zeigt den Hilfstext an
Leertaste, f	Blättert eine Seite vor
b	Blättert ein Seite zurück
/	schaltet den Suchmodus an (vor)
?	Suchmodus rückwärts
<	Springt an den Dateianfang
>	Springt an das Dateiende
Cursor auf/ab	Zeile auf/ab

head - Den ersten Teil einer Datei ausgeben
head [OPTION]... [DATEI]...

tail - Den letzten Teil einer Datei ausgeben
tail [OPTION]... [DATEI]...

- Es werden die ersten oder letzten 10 Zeilen einer Datei ausgegeben.
- `head|tail -n N` gibt jeweils die N ersten oder letzten Zeilen einer Datei aus.

Beispiel

```
[rico@tux work]$ head -n 3 adressen
1:Albert:Herr:Andrzejewski:34117:Kassel:Hessen:Ackerstr.:2
2:Alex:Herr:Arndt:34134:Kassel:Hessen:Ahornstr.:90
3:Andre:Herr:Barske:34266:Niestetal:Hessen:Antoniusstr.:69
```

Dateien archivieren

tar - **t**ape **ar**chiver - verwaltet Dateiarchive
tar [OPTION]... [DATEI]...

- `tar` ist das unter Unix/Linux mit am meisten verwendete Tool zum packen und archivieren von Dateien und Verzeichnissen.
- `tar cvf a.tar dat1 dat2` erzeugt ein Archiv.
- `tar xvf a.tar` entpackt das Archiv.
- `tar tvf` listet den Archivinhalt.
- `tar zvcf...` komprimiert zusätzlich das Archiv.

Beispiel

```
[rico@tux ~]$ tar cf archiv.tar work/
[rico@tux ~]$ ls -l archiv.tar
-rw-r--r-- 1 rico user 10240 24. Jun 08:53 archiv.tar
[rico@tux ~]$ tar tvf archiv.tar
drwxr-xr-x rico/user         0 2008-06-23 14:24 work/
-rw-r--r-- rico/user      6699 2008-06-23 14:24 work/adressen
```


gzip, gunzip - **G**nu **ZI**V **P**acker - komprimiert/dekomprimiert Dateien
gzip [-acdfhlLnNrtvV19] [-S suffix] [name ...]
gunzip [-acfhlLnNrtvV] [-S suffix] [name ...]

- `gzip` komprimiert (Lempel-Ziv-Kodierung) die angegebene Datei. Es entsteht eine neue Datei mit der Endung `.gz` .
- Mit `gunzip` kann diese Datei wieder dekomprimiert werden.
- `-r` (de)komprimiert rekursiv alle Dateien in dem angegebenen Verzeichnis.
- In Verwendung mit `tar(1)` entstehen gepackte Archive (.tgz; .tar.gz).

Beispiel

```
[rico@tux work]$ ls -l adressen
-rw-r--r-- 1 rico user 6699 23. Jun 14:24 adressen
[rico@tux work]$ gzip adressen
[rico@tux work]$ ls -l adressen.gz
-rw-r--r-- 1 rico user 2677 23. Jun 14:24 adressen.gz
```

Dateien suchen

find - sucht in einer Verzeichnishierarchie nach Dateien
find [Verzeichnis...] [Suchkriterium]

- find ist nützlich um nach verschiedenen Kriterien in dem angegebenen Pfad nach Dateien zu suchen.
- `find . -name DATEI` sucht im aktuellen und darunter liegenden Verzeichnissen nach der DATEI und gibt diese mir dem Pfad aus.
- Mehrere Suchkriterien müssen in Klammern angegeben werden. Shell-Metazeichen zur Dateinamenserweiterung (*;?..) werden mit Anführungsstrichen *geschützt*.

Beispiel

```
[rico@tux ~]$ find /usr \( -name "*.gz" -a -size 30k \) -ls
find: /usr/lib/audit: Keine Berechtigung
603685   32 -rw-r--r--   1 root   root   29766 Jun  9 10:44 \
 /usr/share/man/man1/perlfaq4.1.gz
603094   32 -rw-r--r--   1 root   root   30679 Apr 17 00:29 \
 /usr/share/man/man1/gpg.1.gz
...
```


which - gibt den Progammnamen mit vollem Pfad aus

- Das Kommando zeigt den vollständigen Zugriffspfad zu einem Progamm. Das Kommando ist vor allem hilfreich, wenn man sich nicht sicher ist, ob ein Kommando das Richtige ist.
- `which` sucht anhand der PATH-Variable.

locate - sucht Dateien

- Das Kommando dient zum schnellen Auffinden von Dateien.
- `locate` sucht nicht in Verzeichnissen sondern effizient in einer DB

Beispiel

```
[rico@tux ~]$ locate adress
/home/rico/work/adressen
/usr/share/system-config-network/netconfpkg/gui/editadress.glade
...
```

Textbearbeitung

grep - **G**lobal search for a **r**egular **e**xpression and **p**rint out matched lines - Zeichenmuster in Dateien suchen

```
grep [-CVbchilnsvwx] [-Anzahl] [-AB Anzahl] [[-e] Ausdruck \
     | -f Datei]
```

- `grep` gehört sicher mit den verwandten Befehlen `egrep/fgrep` zu den beliebtesten Kommandos.
- `grep` schreibt alle Zeilen die das Muster enthalten auf die Standardausgabe.
- `grep -i` unterscheidet nicht zwischen Groß- und Kleinschreibung.

Beispiel

```
[rico@tux work]$ grep  Nobody /etc/passwd
nobody:x:99:99:Nobody:/:/sbin/nologin
[rico@tux work]$ grep  -in Nobody /etc/passwd
16:nobody:x:99:99:Nobody:/:/sbin/nologin
31:nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
```


cut - Teile jeder Zeile einer Datei entfernen
cut [OPTION]... [DATEI]...

- `cut` wird verwendet, um bestimmte Bytes, Felder oder Zeichen aus den Zeilen einer Datei zu extrahieren.
- `cut -d: -f 2 DATEI` trennt anstelle von TAB die Zeile nach : auf und gibt das zweite Feld aus.

paste - Dateien zeilenweise zusammenfügen
paste [OPTION]... [DATEI]...

- `paste` ist das Gegenstück zu `cut` und erlaubt das zeilenweise Zusammenfügen von Dateien.

Beispiel

```
[rico@tux work]$ cut -d: -f 3,4,6 adressen
Herr:Andrzejewski:Kassel
...
[rico@tux work]$ cut -d: -f 3,4 adressen > name
[rico@tux work]$ paste -d$ name adressen
Herr:Andrzejewski$1:Albert:Herr:Andrzejewski:34117:Kas...
```

sort - Zeilen von Textdateien sortieren
sort [OPTION]... [DATEI]...

- `sort` verkettet alle Eingabedateien und gibt das Ergebnis sortiert auf die Standardausgabe aus.
- `sort -n` erzwingt eine nummerische Sortierung, `-t Z` einen angegebenen Feldtrenner Z ein (statt Leezeichen).
- `sort -r` kehrt die Sortierung um.

Beispiel

```
[rico@tux work]$ sort -n -t: -k 9 adressen
1:Albert:Herr:Andrzejewski:34117:Kassel:Hessen:Ackerstr.:2
21:Haribert:Herr:Drey:35799:Merenberg:Hessen:Eickhoffstr.:2
51:Albert:Herr:Martikke:64859:Eppertshausen:Hessen:Im Struethchen:2
71:Haribert:Herr:Rink:33739:Bielefeld:Nordrhein-Westfalen:Sangstr.:2
14:Eva:Frau:Brueggemann:35390:Giessen:Hessen:Brunnenstr.:3
64:Eva:Frau:Plate:32469:Bueckeburg:Nordrhein-Westfalen:Mittelstr.:3
45:Roger:Herr:Lass:64319:Pfungstadt:Hessen:Hochwaldstr.:4
...
```


uniq - Doppelte Zeilen aus sortierter Datei entfernen
uniq [OPTION]... [EINGABE [AUSGABE]]

- `uniq` entfernt doppelte aufeinanderfolgende Zeilen aus einer sortierten Datei.
- `uniq -c` gibt die Anzahl der vorkommenden Zeilen mit aus.
- `uniq` wird fast immer in Verbindung mit `sort` verwendet.

Beispiel

```
[rico@tux work]$ head -n 1 adressen
1:Albert:Herr:Andrzejewski:34117:Kassel:Hessen:Ackerstr.:2
[rico@tux work]$ cut -d ":" -f 9 adressen|sort|uniq -c
      2 10
      6 11
      2 12
      2 14
      4 2
...
```

diff - finde Unterschiede zwischen zwei Dateien
diff [option]... DATEI1 DATEI2

- `diff` vergleicht zwei Textdateien und gibt die Zeilen aus, in denen sich DATEI1 von DATEI2 unterscheidet. DATEI1 wird mit < markiert und DATEI2 mit > .

Beispiel

```
[rico@tux work]$ diff adressen adressen1
3c3
< 3:Andre:Herr:Barske:34266:Niestetal:Hessen:Antoniusstr.:69
---
> 3:Andre:Herr:Barske:34266:Niestetal:Hessen:Antoniusstr.:67
```


Rechte von Benutzern und Gruppen

Linux ist ein Multiuser Betriebssystem d.h. mehrere Benutzer können gleichzeitig an einem System arbeiten.

- Jede Datei und jedes Verzeichnis gehört nur einem Benutzer und einer Gruppe.
- Benutzer die auf bestimmte Systemressourcen zugreifen wollen, bilden eine gemeinsame Gruppe.
- Damit nicht jeder Benutzer bestimmte Dateien lesen oder die Festplatte formatieren kann, gibt es die sogenannten Zugriffsrechte.
- Aufteilung der Benutzerklassen einer Datei erfolgt in:
  Eigentümer, Gruppe und Sonstige
- Die Zugriffsrechte werden einfach mit `ls -al` angezeigt.

Grundlegende Rechte

- Einer Datei/Verzeichnis werden Typ und Benutzerklassen zugewiesen werden.
- Hierzu teilt sich die Rechte-Zeichenkette in 1+3+3+3 Blöcke.

Typ		
-	normale Datei	
d	Verzeichnis	
l	Soft-Link	
c	Zeichenorientiertes Gerät	
b	Blockorientiertes Gerät	
s	Socket	
p	Pipe	
Rechte	**Datei**	**Verzeichnis**
r	Inhalt lesen (**r**ead)	Verzeichnis listen
w	Inhalt schreiben (**w**rite)	Inhalt ändern (zB. Dateien hinzufügen)
x	Datei ausführen (e**x**ecute)	Ins Verzeichnis navigieren

Beispiel

```
-rw-r--r--  1 rico user 6699 23. Jun 14:24 adressen
lrwxrwxrwx  1 rico user    8 26. Aug 14:16 link-zu-adressen -> adressen
```


Dateirechte ändern

chmod - ändert Dateizugriffsrechte
chmod [OPTION]... MODUS[,MODUS]... DATEI...

- Es gibt einen symbolischen und einen absoluten Modus um Dateirechte zu ändern.
- `chmod g+w <DATEI>` Setzt die Schreibrechte für die Gruppe. (symbolisch)
- `chmod 600 <DATEI>` Der Eigentümer darf die Datei nur lesen und schreiben.(absolut)

Beispiel

```
[rico@tux work]$ ls -al adressen
-rw-r--r-- 1 rico user 6699 23. Jun 14:24 adressen
[rico@tux work]$ chmod 640 adressen
[rico@tux work]$ ls -al adressen
-rw-r----- 1 rico user 6699 23. Jun 14:24 adressen
[rico@tux work]$ chmod o+rw adressen
[rico@tux work]$ ls -al adressen
-rw-r--rw- 1 rico user 6699 23. Jun 14:24 adressen
```

chgrp - ändert Gruppenzugehörigkeit
chgrp [OPTION]... GRUPPE DATEI...

- Mit dem Kommando `chgrp` kann man die Gruppenzugehörigkeit von Dateien oder Ordnern ändern.
- Es kann jedoch jeweils nur der Besitzer einer Datei bzw. eines Ordners die Gruppenzugehörigkeit ändern und zwar nur für die Gruppen, in denen er Mitglied ist.
- ***root*** darf alle Rechte von allen Dateien/Verzeichnissen zuordnen.
  (siehe auch: `chown`)

Ende Teil 2

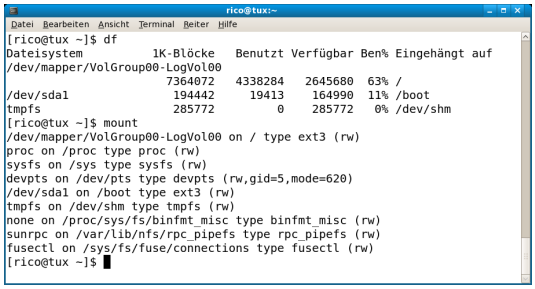
Vielen Dank für Ihre Aufmerksamkeit!

Für Fragen stehen wir gerne zur Verfügung

Dateisystem

- Daten werden nicht beliebig auf dem Datenträger verteilt, sondern organisiert abgelegt.
- Die Stuktur und Regeln wie Daten gespeichert wird vom Dateisystem übernommen.

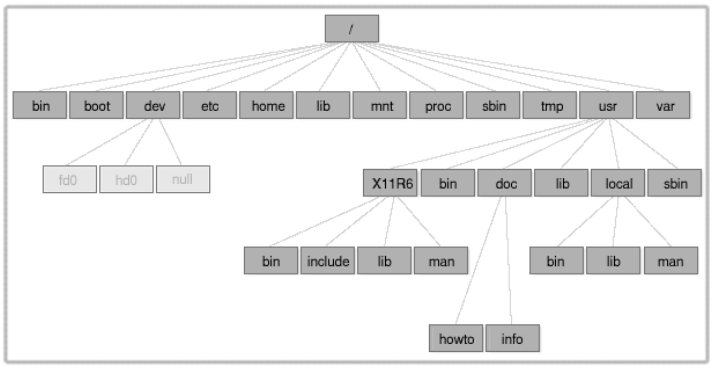
- Es geben eine Vielzahl von Dateisystemtypen. Die unter Linux am verbreitesten sind ext2, ext3, ...
- Ein Dateisystem wird vom Betriebssystem unter einem Pfad eingehangen (gemountet).



Dateien und Verzeichnisse

- Bei Linuxdateinamen wird zwischen Groß- und Kleinschreibung unterschieden.
- Dateinamen können bis zu 255 Zeichen lang werden. Damit sollte eine es möglich sein aussagekräftige Namen zu finden.
- Es ist nicht zwingend vorgeschrieben Dateiendungen (.c; .tar.gz) zu benutzen, aber es erleichtet die Übersicht ungemein.
- Verzeichnisse sind eine besondere Form von Dateien. Sie beinhalten die Inodeeinträge (Speicherplatz) und die dazugehörigen Dateinamen. Die Dateien selbst liegen also an einem x-beliebigen Ort im Dateisystem.

Von der Wurzel an...



Verzeichnishierarchie unter Linux (Quelle: www.linuxfibel.de)

Verzeichnis	Deren Inhalt
/	*root* Wurzelverzeichnis
/bin	Wichtige Programme für den Bootvorgang
/usr	**U**nix **S**ystem **R**esources `/usr/bin /usr/sbin ...`
/sbin	Wichtige Programme für den Admin
/var	dynamische Dateien, Systemlogs, Druckjobs...
/etc	Konfigurationsdateien
/lib	Bibliotheken für den Systemstart, Kernelmodule
/dev	Gerätedateien
/home	Benutzerverzeichnisse

Filesystem Hierarchy Standard

An diese Standardisierung halten sich die meisten Linux-Distributionen. (http://www.pathname.com/fhs)

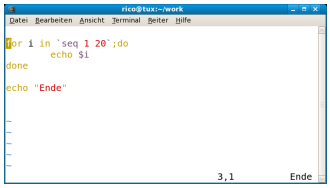
Beispiel

```
[rico@tux /]$  tree -a -d -L 2 /
/
|-- bin
|-- boot
|   |-- grub
|   `-- lost+found
|-- dev
|   |-- .udev
|   |-- VolGroup00
|   |-- bsg
|   |-- bus
...
```

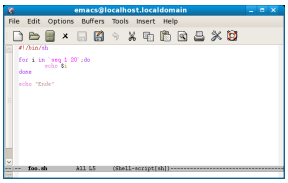
Auswahl des Editors

Editoren bieten dem Nutzer die Möglichkeit, Texte in Dateien zu schreiben und zu verändern.

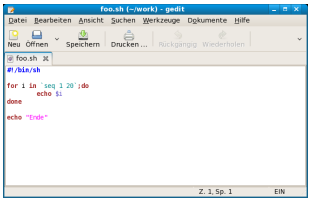
- **vi** - klassischer Unixeditor
  auf jedem Linuxsystem installiert
  rein textbasierend, gewöhnungsbedürftig



- **(x)emacs** - sehr umfangreich mit vielen Funktionen
  Text- und Grafikversion
  kann alles... außer Kaffee kochen ;-)



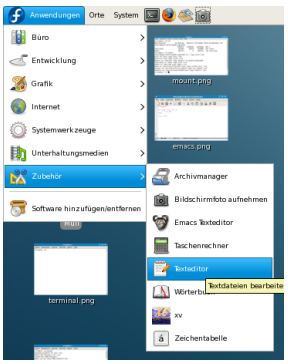
- **gedit** - rein grafischer Editor
  einfach und intuitiv bedienbar.



Arbeiten mit dem *gedit*

- Der GNOME-Editor - gedit - ist der Standard-Texteditor in der GNOME Desktopumgebung.
- Der Editor beherrscht unter anderem Syntaxhervorhebung für viele Programmiersprachen.
- ist über Plugins erweiterbar.

- Starten: Im Terminal mit ***gedit*** oder im Menü unter
  `Anwendungen -> Zubehör -> Texteditor`



Tastenkürzel

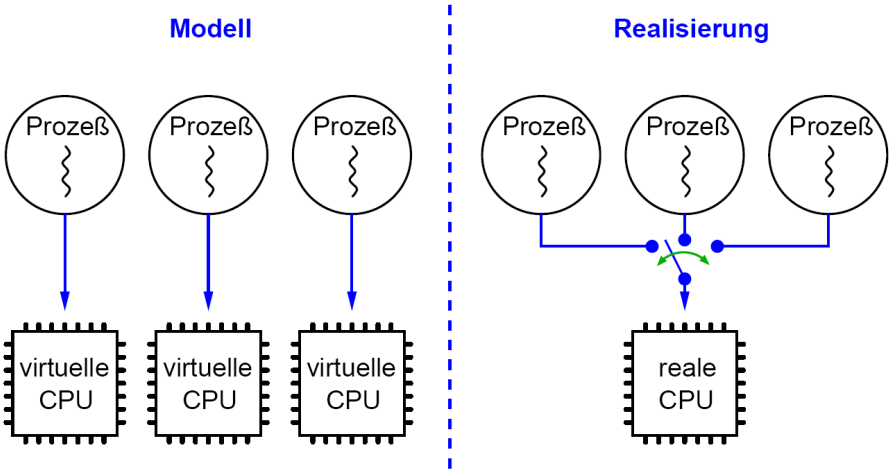
Beim gedit sind alle Tastenkürzel über das Menü abrufbar. In der Regel sind es Tastenkombinationen die in anderen Programmen auch benutzt werden.

Tastenkürzel	Bedeutung
Strg + N	Eine neue Datei erstellen
Strg + O	Eine vorhandene Datei öffnen
Strg + S	Die geöffnete Datei speichern
Strg + Z	Letzten Vorgang rückgängig machen.
Umschalt + Strg + Z	Den rückgängig gemachten Vorgang wiederholen.
Strg + X	Markierung ausschneiden und in die Zwischenablage verschieben.
Strg + C	Markierung in die Zwischenablage kopieren.
Strg + V	Den Inhalt der Zwischenablage einfügen.
Strg + A	Alles markieren.
Umschalt + Strg + S	Unter anderem Namen speichern
Umschalt + Strg + P	Die Druckvorschau anzeigen
Strg + P	Die geöffnete Datei drucken
F1	Hilfe
F7	Rechtschreibprüfung
F9	Seitenleiste (de-)aktivieren
Strg + W	Die Datei schließen
Strg + Q	Das Programm beenden

Was sind Prozesse?

Prozesse werden unter Linux benutzt, um Anwendungssoftware und Teile des Betriebssystems selbst „quasiparallel“ auführen zu können.

sequentielles Prozessmodell mit einem Ausführungsweg (Thread)



Was sind Prozesse?

1. Definition

Ein Prozess ist ein Programm in Ausführung

2. Definition

Ein Prozess ist eine sich ändernde Aktivitätseinheit, gekennzeichnet durch:

- eine Ausführungsumgebung mit
  - Adressbereich (Programmcode und Daten)
  - Zustandsinformationen benutzter Ressourcen, z.B. offene Dateien
- einem oder mehreren Ausführungswegen (Threads) mit
  - dem Befehlszähler
  - den Registern des Prozessors
  - dem Zeiger auf den Kellerspeicher (Stack)

Prozesserzeugung und Hierarchie

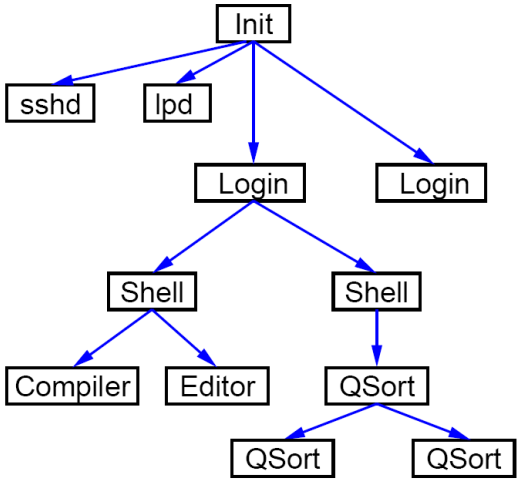
Gründe für die Prozesserzeugung:

- Rechnerstart
  - Hintergrundprozesse (Daemons) für das BS werden benötigt
- Benutzeranfragen
  - interaktive Anmeldung, Start eines Programms
- Erzeugung durch Systemaufruf eines bestehenden Prozesses

Durch Systemaufrufe werden neue Prozesse erzeugt. Dabei wird jedem Prozess eine eindeutige Identifikationsnummer - **PID** - zugewiesen. Es entsteht eine Prozesshierarchie.

Prozesserzeugung und Hierarchie

Beispiel: Prozeßhierarchie unter UNIX



Initialisierungsprozeß

Daemons

Für jeden Benutzer ein Login−Prozeß

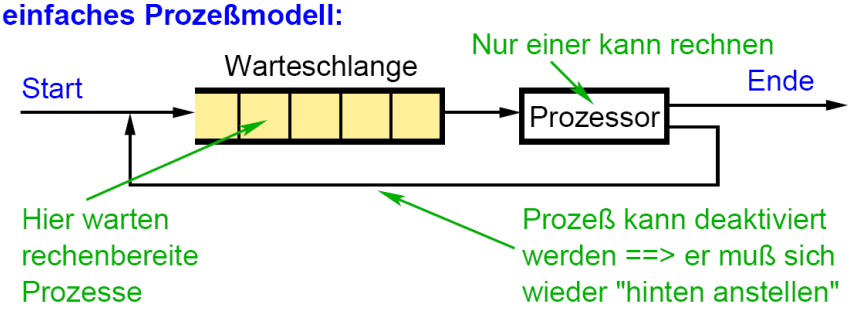
Mehrere Kommando−interpreter (Shells) pro Benutzer

Anwendungen

Prozesszustände und Kontrolle

Nach dem Start gelangen Prozesse in eine Warteschlange. Von dort aus erhalten sie nach bestimmten Regeln Rechenzeit zugeteilt. Diese Zuteilung geschieht durch einen besonderen Teil des BS, den Scheduler.

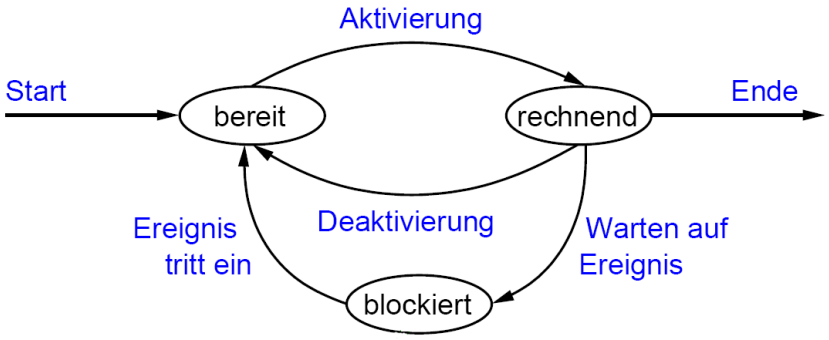
Ein einfaches Prozeßmodell:



Prozesszustände

Grundlage für das Scheduling sind u.a. bestimmte Prozesszustände:

- rechnend - der Prozess wird gerade ausgeführt - running
- rechenbereit - kurzzeitig gestoppt, aber ausführbereit - ready
- blockiert - blockiert bis ein externes Ereignis eintritt - blocked



Das Scheduling kann durch das Setzen von Prioritäten beeinflusst werden.

Prozesskontrolle

Über Signale kann der Benutzer von der Kommandozeile aus Prozesse steuern. Abhängig von der verwendeten Shell werden mehr als 60 verschiedene Signale unterstützt. Grundsätzlich gibt es zwei Gruppen von Signalen.

- Signale, die von einem Prozess nicht abgefangen werden können:
  - Signal 9, SIGKILL oder KILL
    Der Prozess wird zwingend durch das BS beendet
  - Signal 19, SIGSTOP oder STOP
    die Verarbeitung des Prozesses wird unterbrochen
  - Signal 18, SIGCONT oder CONT der gestoppte Prozess wird fortgesetzt

Prozesszustand und Steuerung

- Signale, die von einem Prozess abgefangen (ignoriert) werden können:
  - Signal 1, SIGHUP oder HUP
    Der Prozess soll sich selbst beenden und anschließend neu starten
  - Signal 15, SIGTERM oder TERM
    Der Prozess soll sich freiwillig beenden

Kommando pstree

pstree

pstree zeigt die Prozesse eines Systems in einer Baumstruktur

```
init-+-acpid
     |-auditd---{auditd}
     |-cron
     |-cupsd
     |-2*[dbus-daemon]
     |-dcopserver
     |-dhcpd
     |-events/0
     |-events/1
     |-gpg-agent
     |-hald---hald-runner-+-hald-addon-acpi
     |                    |-2*[hald-addon-keyb]
     |                    `-hald-addon-stor
     |-kded
     |-kdeinit-+-kio_file
     |         |-klauncher
     |         |-konqueror
     |         |-konsole---bash-+-less
     |         |                `-pstree
     |         |-kwin
     |         `-zen-updater---6*[{zen-updater}]
     |-kdesktop
     |-kdm-+-Xorg
     |     `-kdm---kde-+-kwrapper
     |                 `-ssh-agent
     |-khelper
```

Kommando ps

ps

ps zeigt eine Momentaufnahme der aktuellen Prozesse
Option: a = alle Prozesse, u = user, f = Vollformat

```
bsgate1:~ #<|> ps auf
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root     25454  0.0  0.1   3368  1888 pts/1    Ss   10:48   0:00 /bin/bash
root     25973  0.0  0.0   2476   800 pts/1    R+   11:13   0:00  \_ ps auf
root      3873  0.0  1.8  68912 19336 tty7     Ss+  Aug22   0:22 /usr/bin/Xorg -br -r
root      4071  0.0  0.0   2056   636 tty6     Ss+  Aug22   0:00 /sbin/mingetty tty6
root      4067  0.0  0.0   2060   636 tty5     Ss+  Aug22   0:00 /sbin/mingetty tty5
root      4064  0.0  0.0   2060   636 tty4     Ss+  Aug22   0:00 /sbin/mingetty tty4
root      4060  0.0  0.0   2060   640 tty3     Ss+  Aug22   0:00 /sbin/mingetty tty3
root      4056  0.0  0.0   2060   640 tty2     Ss+  Aug22   0:00 /sbin/mingetty tty2
root      4054  0.0  0.0   2060   652 tty1     Ss+  Aug22   0:00 /sbin/mingetty --noc
bsgate1:~ #<|>
```


Kommando top

top

top zeigt eine dynamische Echtzeitansicht eines Systems.
Es stellt eine Zusammenfassung von Systemdaten und eine Liste der vom Kernel verwalteten Prozesse zur Verfügung. Anzeige und Aktualisierungsrate sind konfigurierbar. Prozessen können Signale gesendet werden.

```
top - 11:00:27 up 14 days, 15 min,  3 users,  load average: 0.09, 0.08, 0.03
Tasks: 116 total,   1 running, 115 sleeping,   0 stopped,   0 zombie
Cpu(s):  3.5%us,  1.5%sy,  0.0%ni, 95.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   1027840k total,   858864k used,   168976k free,   144500k buffers
Swap:  1052216k total,      136k used,  1052080k free,   453308k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 3873 root      15   0  316m  18m 4784 S    8  1.9   0:19.62 Xorg
 4285 root      15   0 28396  13m  10m S    2  1.3   0:01.85 kwin
 4287 root      15   0 31100  16m  13m S    0  1.7   0:01.17 kdesktop
    1 root      15   0   740  288  240 S    0  0.0   0:01.08 init
    2 root      RT   0     0    0    0 S    0  0.0   0:00.00 migration/0
    3 root      34  19     0    0    0 S    0  0.0   0:00.00 ksoftirqd/0
    4 root      RT   0     0    0    0 S    0  0.0   0:00.00 migration/1
    5 root      34  19     0    0    0 S    0  0.0   0:00.00 ksoftirqd/1
    6 root      10  -5     0    0    0 S    0  0.0   0:00.03 events/0
    7 root      10  -5     0    0    0 S    0  0.0   0:00.00 events/1
    8 root      15  -5     0    0    0 S    0  0.0   0:00.00 khelper
```

Kommandos pgrep und pkill

pgrep, pkill

pgrep wird verwendet um Prozesse mit Namen zu suchen.
pkill sendet benamten Prozessen ein Signal.

```
bsgate1:~ #<|>
bsgate1:~ #<|> pgrep nfs -l
3666 nfsd4
3667 nfsd
3668 nfsd
3669 nfsd
3670 nfsd
bsgate1:~ #<|>
```


Kommando kill

kill

kill beendet Prozesse.
Standardsignal ist SIGTERM (15). Oft hilft nur SIGKILL (9)
Es können auch andere Signale gesendet werden.

```
bsgate1:~ #<|>
bsgate1:~ #<|> ps ax|grep acroread
26082 ?        R      1:03 /usr/lib/Adobe/Reader8/Reader
26201 pts/1    R+     0:00 grep acroread
bsgate1:~ #<|>
bsgate1:~ #<|> kill -9 26082
bsgate1:~ #<|>
bsgate1:~ #<|> ps ax|grep acroread
26203 pts/1    S+     0:00 grep acroread
bsgate1:~ #<|>
```

Quellen und weiterführende Literatur

- Linux in a Nutshell, ISBN 978-3-89721-426-2
- Vorlesung Betriebssysteme I, Prof. Wismüller
- Plötner, Wendzel: Linux - das distributionsunabhängige Handbuch
- Tanenbaum: Moderne Betriebssysteme 2. Aufl.
- `http://www.galileocomputing.de/openbook/linux/`
- `http://linwiki.org/`
- `http://linuxwiki.de`
- `http://de.wikipedia.org/wiki/Linux`

Ende

Vielen Dank für Ihre Aufmerksamkeit!

Für Fragen stehen wir gerne zur Verfügung

`schuh@informatik.uni-siegen.de`
`manfred.stettner@uni-siegen.de`