Prozesse und Dateien

Frank Trenkamp

17. Mai 1999

Dateien

Dateisystem

- Speichermedien werden durch Formatierung in Blöcke fester Größe aufgeteilt
- Zusammenfassung von Blöcken bildet das Dateisystem
- Dateien werden Gruppen von Blöcken zugeordnet, jeder Block gehört entweder zu genau einer Datei oder ist als frei markiert
- die Menge aller Dateiblöcke und freien Blöcke bildet Gesamtmenge der Blöcke eines Dateisystems
- die Namen von Dateien sind *Inodes* zugeordnet, unterschiedliche Namen können auf eine einzige Inode verweisen
- Vektor von Inodes (*llist*)bildet *lineares Dateisystem*, das vor direktem Zugriff geschützt ist

Datei

- dynamischer Vektor von Bytes mit Operationen Lesen, Schreiben, Positionieren
- keine Zuordnung von *Dateitypen* und *Kommandogruppen*, Programme können beliebige (auch sinnlose) Eingabe verarbeiten, Interpretation eines spezifischen Formats ist programmabhängig
- was geschieht bei Mehrfachzugriff auf eine Datei?

```
caladan:~$ zcat /usr/man/man1/tty.1.gz
.TH TTY 1 "GNU Shell Utilities" "FSF" \" -*- nroff -*-
.SH NAME
tty \- print the file name of the terminal connected to standard input
.SH SYNOPSIS
.B tty
[\-s] [\-\-silent] [\-\-quiet] [\-\-help] [\-\-version]

caladan:~$ man tty
Reformatting tty(1), please wait...

TTY(1)                                                      TTY(1)

NAME
       tty  -  print  the  file name of the terminal connected to
       standard input

SYNOPSIS
       tty [-s] [--silent] [--quiet] [--help] [--version]
```

- Namen sind *case sensitive*, d. h. *File* != *file*
- Zeilentrenner ist *newline*; kein Sonderzeichen zur Markierung von Dateienden

Kataloge

- fassen Dateien und Unterkataloge zusammen
- sind selbst Dateien im System
- als Pfadtrenner dient `"/"`
- besondere Bezeichner sind `"."` und `".."` als Verweis auf das aktuelle Verzeichnis und das Oberverzeichnis
- oberster Katalog der Hierarchie ist die Wurzel `"/"`
- Arbeitskatalog (*working directory*) als Attribut von Prozessen
- Heimatkatalog (*home directory*, `$HOME`)
- absolute Pfade beginnen bei der Wurzel, relative Pfade beim Arbeitskatalog

```
caladan:~$ ls /home/letho/Stuff/
Family.jpg                 mp3_complete_index
MP3_jausa_MP3s_Vol.35.txt  simpsons_list
Pics                       simpsons_list.sorted
RC5                        sound_oss.txt
SO_KEY                     vmware_license
ct-register                xpm

caladan:~/Uni/Shellprog/Dokumente$ echo $HOME
/home/letho

caladan:~/Uni/Shellprog/Dokumente$ ls .
auto             konzept.ind      konzept.tex.bak  referat.dvi
konzept          konzept.log      konzept.tex~     referat.log
konzept.aux      konzept.pdf      pics             referat.tex
konzept.dvi      konzept.ps       referat          referat.tex~
konzept.ilg      konzept.tex      referat.aux

caladan:~/Uni/Shellprog/Dokumente$ ls ../../../../..
amnt        boot        home        lost+found  root        usr
backup      dev         initrd      mnt         sbin        var
bin         etc         lib         proc        tmp         vmlinuz
```

Dateihierarchie

- zwischen verschiedenen Unix-Derivaten bestehen große Unterschiede (Init-Skripte, Kernel-Files, Gerätedateien …)
- immer bilden jedoch die Kataloge die Knoten und die Dateien Blätter eines sogenannten *Dateibaums*

```
/                        Wurzel
   bin/                  wichtigere Kommandos (sh)
   boot                  Hilfsprogramm zum Systemstart
   dev/                  Gerte
      console            Systemkonsole
      fd0                Diskettenlaufwerk
      lp                 Zeilendrucker
      mem                Speicher insgesamt
      null               ewige Jagdgruende
      tty                aktuelles Terminal
      tty1a              Terminal an serieller Schnittstelle
      ttyp1              Terminal an Netzverbindung (Pseudo-Terminal)
   etc/                  Systemverwaltung
      group              Gruppen: Name/Nummer
      init               Programm fuer ersten Proze
      mount              Dateisystem verknuepfen
      passwd             Benutzer: Name/Nummer
   home/                 Heimatkataloge der Benutzer
      username/          ...der Benutzer username
   bin/                  ganz private Kommandos
   lib/                  Bibliotheken, Hilfsdateien
      libc.a             C Bibliothek
   lost+found/           zur Reparatur des Dateisystems
   tmp/                  temporaere Dateien
   unix                  System (fuer boot, ps)
   usr/                  Benutzer
      adm/               Systemverwaltung: Verbrauch
      bin/               restliche Kommandos
      include/           Definitionsdateien fuer C
      lib/               Bibliotheken, Textmakros, ...
      local/             lokale Entwicklungen
         bin/               lokale Kommandos
         lib/               lokale Bibliotheken
      man/               online Manual
      spool/             Bereich fuer Kommunikationsprogramme
         lp/             Druckauftraege
         mail/           elektronische Post
```

Zugriffsrechte

- die Vergabe von Rechten gewährleistet Sicherheit, Systemintegrität, Verteilung von Ressourcen etc.
- Rechte werden zusammengesetzt aus *read, write, executable*
- jeder Benutzer im System hat Benutzer-ID und Gruppen-ID
- jede Datei/Katalog im System definiert Rechte für Besitzer, Gruppe und den Rest der Welt
- *executable* auf Katalogen erlaubt Suche im Katalog

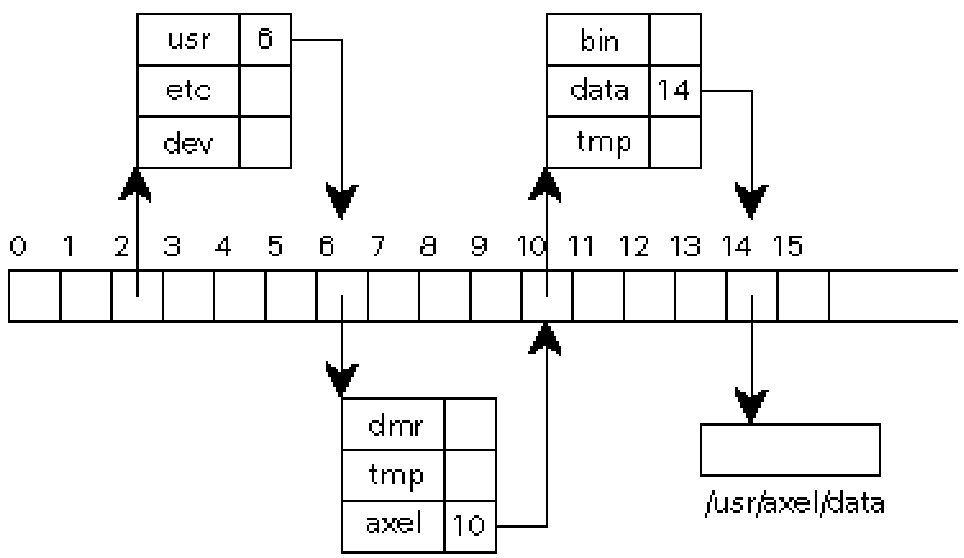
```
caladan:~/temp$ ls -ld katalog
d--x------   2 letho    letho        1024 May 14 20:30 katalog
caladan:~/temp$ ls katalog
ls: katalog: Permission denied
caladan:~/temp$ ls -l katalog/file
-rw-r--r--   1 letho    letho          10 May 14 20:31 katalog/file
caladan:~/temp$ chmod 400 katalog
caladan:~/temp$ ls -l katalog/file
ls: katalog/file: Permission denied
caladan:~/temp$ ls -l katalog
ls: katalog/file: Permission denied
total 0
```

- `SUID`-Bit und `SGID`-Bit erlauben während der Ausführung eines Programms den Wechsel zu anderen Benutzer- oder Gruppen-ID's und entsprechendem Zugriff auf Systemressourcen
- *reale* und *effektive* `UID / GID`

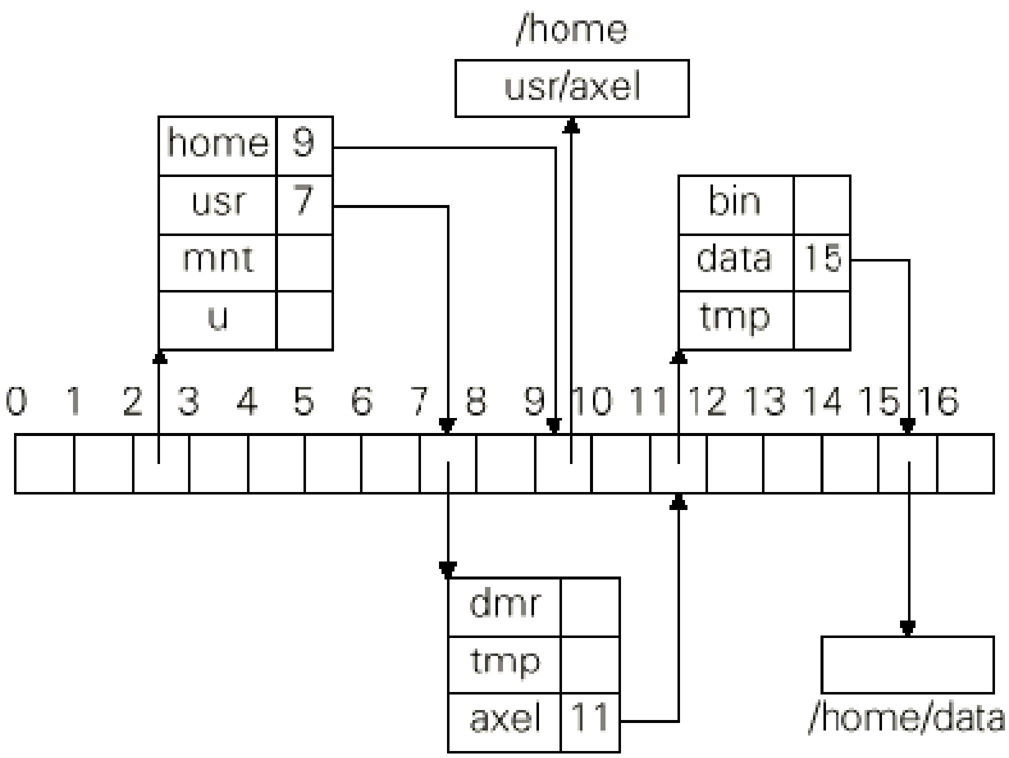
```
caladan:~$ id
uid=1000(letho) gid=1000(letho) groups=1000(letho),25(floppy),29(audio),100(users)
caladan:~$ ls -l /bin
-rwsr-xr-x   1 root     root        14768 Mar 25 04:55 ping
-rwsr-xr-x   1 root     root        13328 Mar 25 04:55 ping6
-rwxr-xr-x   1 root     root        12468 Oct 30  1998 ps
-rwxr-xr-x   1 root     root         4996 Nov 19 12:55 pwd
lrwxrwxrwx   1 root     root            4 Apr 10 22:06 rbash -> bash
-rwxr-xr-x   1 root     root        10124 Jul  3  1998 rm
-rwxr-xr-x   1 root     root         4916 Jul  3  1998 rmdir
-rwxr-xr-x   1 root     root         7864 Aug 16  1998 run-parts
drwx------   2 root     root         1024 Apr 10 22:02 secure-su
-rwxr-sr-x   1 root     shadow      10928 Mar 29  1998 vlock
```

Inodes

- Inodes sind die eigentlichen Verweise auf Dateien, Namen sind lediglich Erleichterungen für Benutzer
- die systeminterne Repräsentation besteht aus positiver Integerzahl
- Inodes als Beschreibungen von Dateien: Besitzer, Rechte, Verweise auf Blöcke, Name ...
- Kataloge stellen Tabellen dar, in denen Namen mit Inodes assoziiert werden; dadurch ist weitere Addressierung von Unterkatalogen oder Dateien möglich



- *harte Links* sind unterschiedliche Namen für die gleiche Inode
- *symbolische Links* sind indirekte Zeiger auf einen Pfad, der vom Dateisystem-Treiber interpretiert wird



- wird der letzte Link auf eine Datei gelöscht, ist die Datei endgültig verschwunden
- Inodes kennen die Anzahl der Verweise auf sich selbst
- Kataloge haben minimal zwei Links
- harte Links über Dateisystemgrenzen gibt es nicht, die Eindeutigkeit von Inodes in verschiedenen Dateisystemen ist nicht garantiert
- `stat`, `fstat`, `lstat` liefern die Informationen aus Inodes

```
caladan:~/temp$ ls -ldi .
  37698 drwxr-xr-x   2 letho    letho        1024 May 14 00:35 .

caladan:~/temp$ touch file
caladan:~/temp$ ln file h_link
caladan:~/temp$ ln -s file s_link
caladan:~/temp$ ls -li *
  37795 -rw-r--r--   2 letho    letho           0 May 14 01:37 file
  37795 -rw-r--r--   2 letho    letho           0 May 14 01:37 h_link
  37796 lrwxrwxrwx   1 letho    letho           4 May 14 01:37 s_link -> file
```

Beispiel: Fragment der entspr. Header-Datei im Source-Tree des Linux-Kernels 2.2.7

```
caladan:~# cat /usr/src/linux-2.2.7/include/asm-i386/stat.h
struct stat {
        unsigned short st_dev;
        unsigned short __pad1;
        unsigned long st_ino;
        unsigned short st_mode;
        unsigned short st_nlink;
        unsigned short st_uid;
        unsigned short st_gid;
        unsigned short st_rdev;
        unsigned short __pad2;
        unsigned long  st_size;
        unsigned long  st_blksize;
        unsigned long  st_blocks;
        unsigned long  st_atime;
        unsigned long  __unused1;
        unsigned long  st_mtime;
        unsigned long  __unused2;
        unsigned long  st_ctime;
        unsigned long  __unused3;
        unsigned long  __unused4;
        unsigned long  __unused5;
};
```

`man fstat` erzählt allerdings etwas mehr ...

```
 struct stat
              {
                  dev_t         st_dev;      /* device */
                  ino_t         st_ino;      /* inode */
                  mode_t        st_mode;     /* protection */
                  nlink_t       st_nlink;    /* number of hard links */
                  uid_t         st_uid;      /* user ID of owner */
                  gid_t         st_gid;      /* group ID of owner */
                  dev_t         st_rdev;     /* device type (if inode device) */
                  off_t         st_size;     /* total size, in bytes */
                  unsigned long st_blksize;  /* blocksize for filesystem I/O */
                  unsigned long st_blocks;   /* number of blocks allocated */
                  time_t        st_atime;    /* time of last access */
                  time_t        st_mtime;    /* time of last modification */
                  time_t        st_ctime;    /* time of last change */
              };
```

Gerätedateien

- Dateien unter `/dev` sind die Verbindung zu Peripheriegeräten wie Platten, Bandgeräten, Speicher, Terminals, seriellen und parallelen Schnittstellen ...

- Zugriffe auf diese Dateien werden vom Systemkern in Hardwarezugriffe umgewandelt

```
caladan:~/temp$ echo Huhu! > /dev/ttyp1
Huhu!
```

- *zeichenorientierte* Geräte (*character devices*): pro Datenübertragung werden eine variable Anzahl Bytes übertragen

- *blockorientierte* Geräte (*block devices*): pro Datenübertragung werden eine fixe Anzahl Bytes übertragen

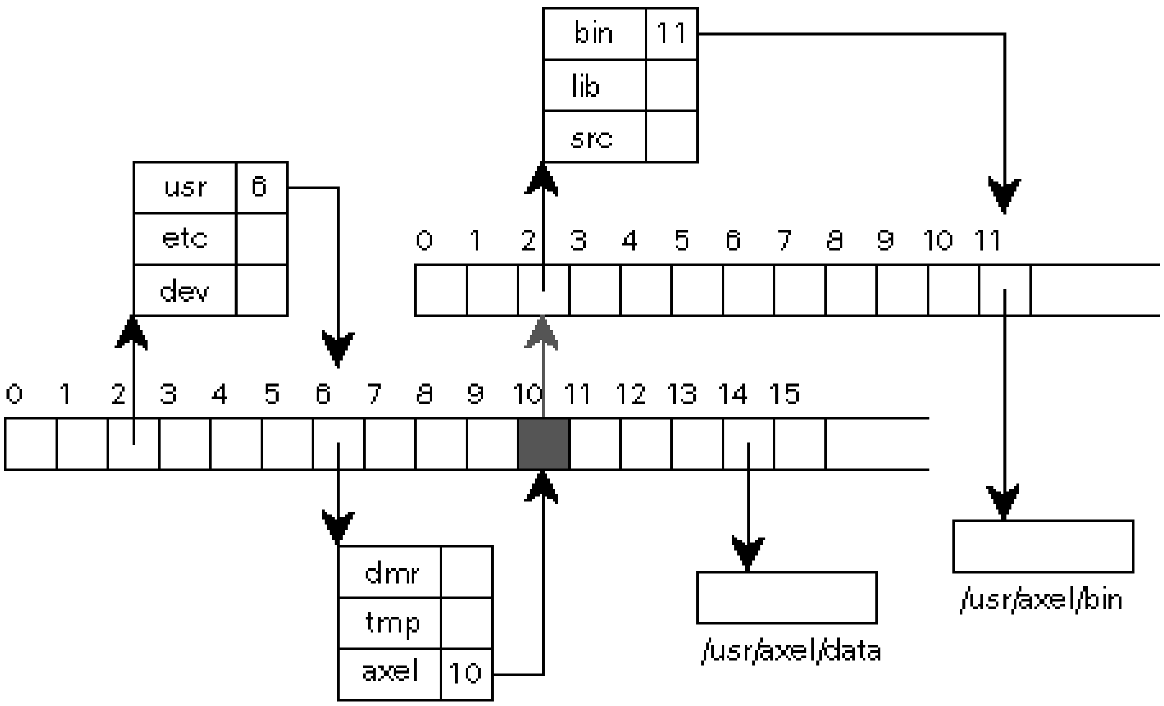
- Unterscheidung erfolgt durch Markierung in der Inode

- *major*-Nummern bezeichnen Geräteklassen, *minor*-Nummern bezeichnen ein bestimmtes Gerät in der Klasse

```
caladan:~/temp$ ls -lsi /dev/hda[1-4]
 125003    0 brw-rw----   1 root     disk       3,   1 Feb 23 03:41 /dev/hda1
 125054    0 brw-rw----   1 root     disk       3,   2 Feb 23 03:41 /dev/hda2
 125055    0 brw-rw----   1 root     disk       3,   3 Feb 23 03:41 /dev/hda3
 125056    0 brw-rw----   1 root     disk       3,   4 Feb 23 03:41 /dev/hda4
caladan:~/temp$ ls -lsi /dev/ttyS[0-3]
 125892    0 crw-rw----   1 root     dialout    4,  64 Feb 23 03:42 /dev/ttyS0
 125893    0 crw-rw----   1 root     dialout    4,  65 Feb 23 03:42 /dev/ttyS1
 125894    0 crw-rw----   1 root     dialout    4,  66 Feb 23 03:42 /dev/ttyS2
 125895    0 crw-rw----   1 root     dialout    4,  67 Feb 23 03:42 /dev/ttyS3
```

Einhängen von Dateisystemen

- mount hängt die Wurzel eines Dateisystems in den Knoten eines übergeordneten ein, wenn z. B. die Dateihierarchie auf verschiedene Platten oder Partitionen verteilt wurde
- die Inode des entspr. Knotens wird temporär mit der Inode der einzuhängenden Wurzel überlagert



Prozesse

Eigenschaften

- Prozesse sind Programme in Ausführung
- sie bestehen aus einem *Image* mit folgenden Bestandteilen:
  - Code
  - dynamisch erworbene Speicherflächen
  - Zustandsvariablen: rechnend, bereit, blockiert
  - Programmzähler
  - Signaltabellen
  - Dateiverbindungen als *Filedeskriptoren*
  - `open`, `creat` liefern Filedesk., `read`, `write` werden sie übergeben
  - `lseek` positioniert, `close` schließt Dateien
  - u. a. `UID/GID` als Attribute
  - Shell-Variablen bilden das *Environment*

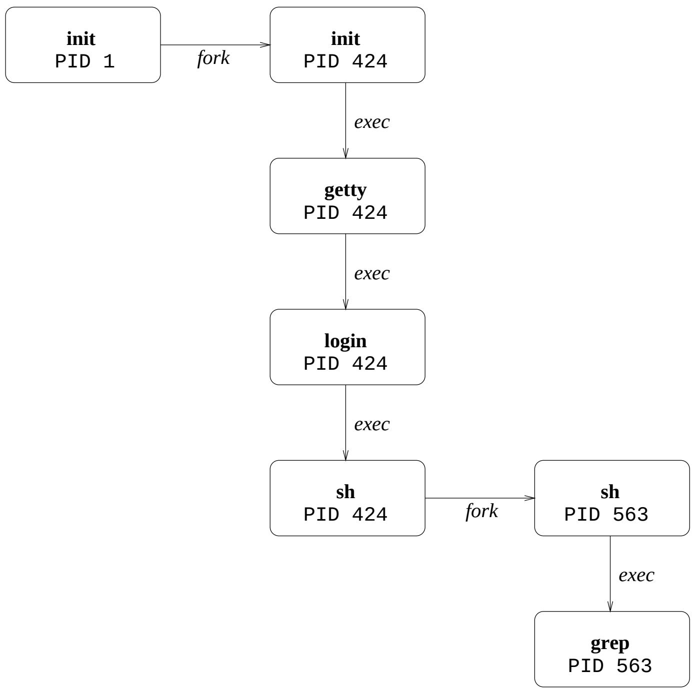
```
WINDOWID=46137346
HZ=100
HOSTNAME=caladan
ignoreeof=0
QTDIR=/usr/local/qt
MANPATH=/usr/local/qt/man:/usr/man:/usr/local/man/:/usr/X11R6/man
PS1=\h:\w\$
PS2=>
KDEDIR=/usr/local/kde
USER=letho
COLORTERM=rxvt
DISPLAY=:0.0
LOGNAME=letho
SHLVL=3
HUSHLOGIN=FALSE
MINICOM=-c on
SHELL=/bin/bash
HOSTTYPE=i486
OSTYPE=linux-gnu
TERM=xterm
HOME=/home/letho
```

- die *Prozeßtabelle* verwaltet pro Prozeß Zustand, Programmzähler, Speicheradressen, geöffnete Dateien, `PID`, `PPID` ...

- Lösung für den Mehrfachzugriff auf Dateien: jeder Prozess in der Prozesstabelle unterhält Tabelle mit den Deskriptoren der von ihm geöffneten Dateien mit dem jeweiligen Offset-Wert, Löcher können entstehen
- alle Prozesse besitzen Dateiverbindungen zu *stdin, stdout, stderr* mit Filedeskriptoren $\{0,1,2\}$, die in der Shell umgelenkt werden können
- bei Interaktion mit Dateien nimmt der Kern Prüfung der Rechte von Prozess und Datei vor

Generierung

- allererster Prozeß nach dem Systemstart ist `init`
- `login` setzt initial `UID/GID`
- neue Prozesse werden durch *fork-and-exec* erzeugt
- daher sind alle Prozesse Nachfolger von `init`
- *fork* erzeugt vollständige Kopie des aufrufenden Prozesses
- *exec* redefiniert das Image
- erhalten bleiben
  - Arbeitskatalog
  - Umgebungsvariablen
  - alle benutzten Filedeskriptoren
  - zu ignorierende Signale
- nicht erhalten bleiben
  - andere Shell-Variablen
  - nicht zu ignorierende Signale
- aufrufende Prozesse werden synchron weiterbearbeitet bzw. erwarten mit `wait` das Ende ihrer Kinder


init
PID 1
fork
init
PID 424
exec
getty
PID 424
exec
login
PID 424
exec
sh
PID 424
fork
sh
PID 563
exec
grep
PID 563

- Prozesshierarchie mit `pstree`

```
init-+-apmd
     |-bash---startx---xinit-+-X
     |                       `-kwm-+-emacs-+-xdvi.bin---gs
     |                             |       `-xdvi.bin
     |                             |-kaudioserver---maudio
     |                             |-kbgndwm
     |                             |-kfm-+-rxvt---bash---slattach
     |                             |     `-rxvt---bash
     |                             |-kpager
     |                             |-kpanel
     |                             |-krootwm
     |                             |-kwmsound
     |                             |-rxvt---su
     |                             |-rxvt---bash
     |                             `-rxvt---bash---pstree
     |-cardmgr
     |-getty
     |-gpm
     |-inetd
     |-kflushd
     |-klogd
     |-kpiod
     |-kswapd
     |-logger
     |-portmap
     |-rpc.mountd
     |-rpc.nfsd
     |-sshd
     |-syslogd
     |-update
     `-vmnet-bridge
```

Signale

- werden von anderen Prozessen oder am Terminal erzeugt
- `kill` verschickt Signale
- Interprozeßkommunikation
- Programme können Signalbehandlung beeinflussen
- `signal()` installiert einen Signal-Handler
- Prozesse können `SIGKILL` und `SIGSTOP` nicht abfangen
- `SIGHUP` geht an alle Prozesse bei Beendigung der Terminalsesion, `nohup` schützt Hintergrundprozesse
- die Shell kann Signale mit `trap` abfangen

```
caladan:~$ trap "echo SIGINT empfangen ... ; date" SIGINT
caladan:~$ SIGINT empfangen ...
Mon May 17 04:05:26 /etc/localtime 1999
caladan:~$ bash
caladan:~$
caladan:~$ exit
caladan:~$ SIGINT empfangen ...
Mon May 17 04:05:42 /etc/localtime 1999
```

- Signale werden an einen Prozess oder an eine Vordergrund-Prozessgruppe geliefert
- die Shell muss sich selbst und Hintergrundprozesse schützen (`kill -0`) ...
- sterbende Prozesse schicken Signal an Vaterprozeß , der Statusinformationen abfragen kann und Prozeßtabelle aufräumen muß
- wenn Vaterprozeß vorher verstirbt, wird Kindprozeß Nachfolger von `init`

Ausschnitt aus signal.h

```
#define SIGHUP          1
#define SIGINT          2
#define SIGQUIT         3
#define SIGILL          4
#define SIGFPE          8
#define SIGKILL         9
#define SIGUSR1        10
#define SIGSEGV        11
#define SIGUSR2        12
#define SIGPIPE        13
#define SIGALRM        14
#define SIGTERM        15
#define SIGCHLD        17
```

```
USER       PID %CPU %MEM  SIZE   RSS TTY STAT START   TIME COMMAND
daemon     167  0.0  0.0  1192     0  ?  SW   21:17   0:00 (portmap)
letho      220  0.0  0.0  2232     0   1 SW   21:17   0:00 (bash)
letho      248  0.0  0.0  1968     0   1 SW   21:21   0:00 (startx)
letho      249  0.0  0.0  2172     0   1 SW   21:21   0:00 (xinit)
letho      253  0.0  1.9  6128  1224   1 S    21:22   0:05 (kwm)
letho      268  0.0  0.0  5748     0   1 SW   21:22   0:00 (kaudioserver)
letho      269  0.0  0.7  5528   468   1 S    21:22   0:00 kwmsound
letho      270  0.0  1.2  7272   784   1 S    21:22   0:01 kfm
letho      271  0.0  1.4  5628   892   1 S    21:22   0:00 krootwm
letho      272  0.0  1.9  6216  1260   1 S    21:22   0:06 kpanel
letho      273  0.0  0.7  5712   496   1 S    21:22   0:00 kbgndwm
letho      281  0.0  0.0  5776    48   1 S    21:22   0:00 maudio -media 0
letho      292  0.0  0.0  2008     0   1 SW   21:22   0:00 (rxvt)
letho      293  0.0  1.3  6288   884   1 S    21:22   0:01 kpager -restore kpagerrc.1
letho      295  0.0  0.0  2068     0   1 SW   21:22   0:01 (rxvt)
letho      296  0.0  0.6  2068   400   1 S    21:22   0:00 rxvt
letho      299  0.0  1.0  2228   692  p2 S    21:22   0:00 bash
letho      312  0.2  0.6  7520   420   1 S    21:24   0:23 emacs
letho     1033  0.0  0.0  2076     0   1 SW   22:17   0:00 (rxvt)
letho     2407  0.0  0.8  2128   508   1 S    23:44   0:00 rxvt
letho     2408  0.0  1.0  2224   656  p3 S    23:44   0:00 bash
letho     2446  0.0  1.0  1300   664  p3 R    23:56   0:00 ps auxw
root         1  0.0  0.0  1096    60  ?  S    21:17   0:03 init
root         2  0.0  0.0     0     0  ?  SW   21:17   0:00 (kflushd)
root         3  0.0  0.0     0     0  ?  SW   21:17   0:00 (kpiod)
root         4  0.0  0.0     0     0  ?  SW   21:17   0:00 (kswapd)
root        13  0.0  0.0  1060    36  ?  S    21:17   0:00 update
root       150  0.0  0.1  1428    72  ?  S    21:17   0:00 /sbin/syslogd
root       152  0.0  0.0  1136     0  ?  SW   21:17   0:00 (klogd)
root       162  0.0  0.0  1172     0  ?  SW   21:17   0:00 (cardmgr)
root       181  0.0  0.0  1036     0  ?  SW   21:17   0:00 (sshd)
root       189  0.0  0.0  1080     0  ?  SW   21:17   0:01 (apmd)
root       195  0.0  0.0  1104    52  ?  S    21:17   0:00 /usr/sbin/gpm -m /dev/psaux -t ps2
root       199  0.0  0.0  1384     0  ?  SW   21:17   0:00 (inetd)
root       204  0.0  0.1  1780    76  ?  S    21:17   0:00 /usr/sbin/rpc.nfsd
root       207  0.0  0.1  1760    68  ?  S    21:17   0:00 /usr/sbin/rpc.mountd
root       215  0.0  0.0  1060     0  ?  SW   21:17   0:00 (vmnet-bridge)
root       216  0.0  0.0  1080     0  ?  SW   21:17   0:00 (logger)
root       221  0.0  0.0  1224     0   2 SW   21:17   0:00 (getty)
root       250  1.5  2.8  8188  1804  ?  R    21:21   2:27 /usr/X11R6/bin/Xwrapper :0
root       297  0.0  0.0  1452     0  p0 SW   21:22   0:00 (su)
root       298  0.0  0.0  2228     0  p1 SW   21:22   0:00 (bash)
root      1204  0.0  0.0  1080    36  p4 S    22:37   0:00 /sbin/slattach -p cslip -s 57600 /dev/ttyS0
```

Das proc – Filesystem

- das `proc`-Dateisystem ist ein virtuelles Filesystem, das zur Laufzeit vom Kernel erzeugt wird
- Prozesse werden als Dateien bzw. in Katalogen repräsentiert, verschiedene Eigenschaften lassen sich abfragen
- Programme lassen sich portabel implementieren
- unter Linux dient das `proc`-Filesystem auch zur Systemkontrolle (*sysctl*), Parameter bezüglich einiger Subsysteme wie TCP/IP-Konfiguration sind manipulierbar

```
caladan:/proc/self$ ls -l
total 0
-r--r--r--   1 letho    letho           0 May 16 17:58 cmdline
lrwx------   1 letho    letho           0 May 16 17:58 cwd -> /proc/261
-r--------   1 letho    letho           0 May 16 17:58 environ
lrwx------   1 letho    letho           0 May 16 17:58 exe -> /bin/bash
dr-x------   2 letho    letho           0 May 16 17:58 fd
pr--r--r--   1 letho    letho           0 May 16 17:58 maps
-rw-------   1 letho    letho           0 May 16 17:58 mem
lrwx------   1 letho    letho           0 May 16 17:58 root -> /
-r--r--r--   1 letho    letho           0 May 16 17:58 stat
-r--r--r--   1 letho    letho           0 May 16 17:58 statm
-r--r--r--   1 letho    letho           0 May 16 17:58 status
caladan:/proc/self$ ls -l fd/
total 0
lrwx------   1 letho    letho          64 May 16 17:59 0 -> /dev/ttyp2
lrwx------   1 letho    letho          64 May 16 17:59 1 -> /dev/ttyp2
lrwx------   1 letho    letho          64 May 16 17:59 2 -> /dev/ttyp2
lrwx------   1 letho    letho          64 May 16 17:59 255 -> /dev/ttyp2
caladan:/proc/self$ od -c cmdline
0000000   b   a   s   h
0000004
caladan:/proc/self$ cat environ
HZ=100HOSTNAME=caladanlibignoreeof=0QTDIR=/usr/local/qtMANPATH=/usr/local/qt/man:/usr/man:/usr/local/man/:/us
r/X11R6/manKDEDIR=/usr/local/kdeUSER=lethoLS_COLORS=no=00:fi=00:di=01;34:ln=01;36:pi=40;33:so=01;35:bd=40;33;
01:cd=40;33;01:or=40;31;01:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arj=01;31:*.taz=01;31:*.lzh=01;31:*.zip=01;31:*
.z=01;31:*.Z=01;31:*.gz=01;31:*.deb=01;31:*.jpg=01;35:*.gif=01;35:*.bmp=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=0
1;35:*.xpm=01;35:*.tif=01;35:*.mpg=01;37:*.avi=01;37:*.gl=01;37:*.dl=01;37:MACHTYPE=i486-pc-linux-gnuMAIL=/va
/spool/mail/lethoLANG=CDISPLAY=:0.0LOGNAME=lethoSHLVL=3HUSHLOGIN=FALSEMINICOM=-conSHELL=/bin/bashHOSTTYPE=i48
6OSTYPE=linux-gnuTERM=xtermHOME=/home/lethoPATH=/usr/local/qt/bin:/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/
usr/games:/usr/X11R6/bin:/usr/games:.:/usr/local/kde/bin:~/bin:/usr/local/snns/bin:/usr/local/bin_=/usr/local
/bin/rxvtCOLORFGBG=0;15WINDOWID=46137346COLORTERM=rxvt
```