Linux-Training
Teil 2: Arbeiten an der Shell

Johannes Franken
<jfranken@jfranken.de>

Inhalt

Eigenschaften der Shell

- Interne/externe kommandos / type / which / PATH
- Command line editing
- Tab, Cursor, , ^A, ^E, ^C, ^U, ^R, fc, $_
- history, !x
- Prozesse starten und stoppen, Strg-Z, fg, bg, &, jobs, %%
- Signale senden (kill -l, kill)
- PATH, which

Environment

- Variablen
- Vererbung (export)
- umask
- LD_LIBRARY_PATH
- Wildcards, {}, [], $()
- Anführungszeichen: ', "
- Prozessketten: ;, &&, ||
- Umleitungen > >> < <<, Backticks
- Kanäle (stdin, stdout, stderr), Umleitungen >>, 2>, 2>&1, &>

Grundlegende Tools

Insb. zum Aufbau von Pipes geeignet

cat, tac

grep

reguläre Ausdrücke

cut

sed, tr

wc

nl, fmt

tail, head

less, more

od

tee

split

xargs

sort, uniq

Tools zur Verwaltung von Prozessen

Prozesstabelle auslesen

ps

```
$ ps
  PID TTY          TIME CMD
  29667 pts/0    00:00:00 bash
   4438 pts/0    00:00:00 vim
   4750 pts/0    00:00:00 ps
```

*Listing: **ps**: Prozesse der aktuellen Konsole anzeigen*

```
$ ps -e
 PID TTY          TIME CMD
   1 ?        00:00:22 init
 501 ?        00:05:44 syslogd
 504 ?        00:00:00 klogd
 531 ?        00:00:09 amavisd-new
 534 ?        00:00:00 amavis-milter
 587 ?        00:01:01 cyrmaster
 591 ?        00:00:06 dhcpd3
 605 ?        00:00:00 notifyd
 618 ?        00:00:08 exim3
 625 ?        00:00:00 famd
 646 ?        00:00:00 inetd
 655 ?        01:16:16 tcplogd
 893 ?        00:01:04 jabberd
[...]
```

*Listing: **ps -e**: Alle Prozesse anzeigen*

```
$ ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0  2006 ?        00:00:22 init [2]
root       501     1  0  2006 ?        00:05:44 /sbin/syslogd -r
root       504     1  0  2006 ?        00:00:00 /sbin/klogd -c 1
amavis     531     1  0  2006 ?        00:00:09 amavisd (master)
amavis     534     1  0  2006 ?        00:00:00 /usr/sbin/amavis-milter -D
cyrus      587     1  0  2006 ?        00:01:01 /usr/sbin/cyrmaster -d
root       591     1  0  2006 ?        00:00:06 /usr/sbin/dhcpd3 -q eth1 eth2
cyrus      605   587  0  2006 ?        00:00:00 notifyd
mail       618     1  0  2006 ?        00:00:08 /usr/lib/exim/exim3 -bd -q30m
root       625     1  0  2006 ?        00:00:00 /usr/sbin/famd -T 0
root       646     1  0  2006 ?        00:00:00 /usr/sbin/inetd
root       655     1  0  2006 ?        01:16:16 /usr/sbin/tcplogd
jabber     893     1  0  2006 ?        00:01:04 /usr/sbin/jabberd
[...]
```

*Listing: **ps -f**: Weitere Spalten*

Im folgenden Beispiel werden die Prozesse angezeigt, die am meisten RAM verbrauchen.

```
$ ps -eo vsz,user,cmd|sort -nr|head
276516 wml      perl -S /usr/bin/html2ps -f html2psrc -l de schottland_inhalt.de.html.high
121356 mysql    /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --user=mysql
 38212 wml      gs -q -dNODISPLAY /tmp/filebz8KDe.ps -c quit
 35884 clamav   /usr/sbin/clamd
 34860 root     spamd child
 31244 bind     /usr/sbin/named -u bind
[...]
```

*Listing: **ps -o**: Spalten auswählen*

```
$ ps -fu jfranken
UID        PID  PPID  C STIME TTY          TIME CMD
jfranken  1336     1  0  2006 tty3     00:00:00 -bash
jfranken  5795     1  0  2006 tty2     00:00:00 -bash
jfranken  6379  5795  0  2006 tty2     00:00:00 screen
jfranken  6380  6379  0  2006 ?        00:02:47 SCREEN
jfranken  8204  8202  0  2006 ?        00:00:08 sshd: jfranken@pts/7
jfranken  8205  8204  0  2006 pts/7    00:00:00 -bash
jfranken  8591     1  0 Jan17 ?        00:00:01 autossh -M 23000 -vCL 1025:localhost:25 grotesk
jfranken  6466     1  0 Feb03 tty1     00:00:00 -bash
jfranken  1411  1336  0 Feb08 tty3     00:00:00 screen -x
jfranken 20320     1  0 Feb09 ?        00:00:00 ssh-agent
jfranken 20277  6466  0 Feb10 tty1     00:00:00 screen -x
jfranken 12982  6380  0 Mar02 pts/14   00:00:01 /bin/bash
jfranken 29667  6380  0 Mar04 pts/0    00:00:00 /bin/bash
jfranken 31430  8205  0 Mar08 pts/7    00:00:00 mutt
jfranken 28178  8591  0 04:09 ?        00:00:00 /usr/bin/ssh -L 23000:127.0.0.1:23000 -R 23000:127.0.0.1:2300
jfranken  2729  6380  0 20:43 pts/15   00:00:00 /bin/bash
jfranken  2740  2729  0 20:43 pts/15   00:00:00 /usr/bin/ssh wml@localhost nice make www.jfranken.de
jfranken  4438 29667  0 20:44 pts/0    00:00:01 vim linux2.wml
jfranken 10363 12982  0 20:58 pts/14   00:00:00 ps -fu jfranken
```

*Listing: **`ps -u`**: Filtern nach Username*

```
$ ps -fp 2740
UID        PID  PPID  C STIME TTY          TIME CMD
jfranken  2740  2729  0 20:43 pts/15   00:00:00 /usr/bin/ssh wml@localhost nice make www.jfranken.de
```

*Listing: **`ps -p`**: Filtern nach PID*

pstree

```
$ pstree
init-+-amavis-milter
     |-amavisd-new---2*[amavisd-new]
     |-apache2---8*[apache2]
     |-atd
     |-autossh---ssh
     |-3*[bash---screen]
     |-bash---screen---screen-+-bash---mutt
     |                        |-bash-+-less
     |                        |      `-pstree
     |                        |-bash---vim
     |                        `-bash---ssh
[...]
```

*Listing: **`pstree:`** Prozesbaum ausgeben*

```
$ pstree -p
init(1)-+-amavis-milter(534)
        |-amavisd-new(531)-+-amavisd-new(10519)
        |                  `-amavisd-new(10544)
        |-apache2(1201)-+-apache2(30055)
        |               |-apache2(4238)
        |               |-apache2(4239)
        |               |-apache2(4240)
        |               |-apache2(4241)
        |               |-apache2(4242)
        |               |-apache2(4243)
        |               `-apache2(4244)
        |-atd(1307)
        |-autossh(8591)---ssh(28178)
        |-bash(1336)---screen(1411)
        |-bash(5795)---screen(6379)---screen(6380)-+-bash(9273)---mutt(10814)
        |                                          |-bash(12982)---pstree(13931)
        |                                          |-bash(29667)---vim(4438)
        |                                          `-bash(2729)---ssh(2740)
[...]
```

*Listing: **`pstre -p`**: PID mit anzeigen*

```
$ pstree -p 6380
screen(6380)-+-bash(9273)---mutt(10814)
             |-bash(12982)---pstree(15892)
             |-bash(29667)---vim(4438)
             `-bash(2729)---ssh(2740)
```

*Listing: **`pstree -p`**: Filtern nach PID*

```
$ pstree -pu jfranken
bash(1336)---screen(1411)

bash(5795)---screen(6379)---screen(6380)-+-bash(9273)---mutt(10814)
                                         |-bash(12982)---pstree(14731)
                                         |-bash(29667)---vim(4438)
                                         `-bash(2729)---ssh(2740)
bash(6466)---screen(20277)

ssh-agent(20320)
[...]
```

*Listing: **`pstree -u`**: Filtern nach User*

top

Selbstaktualisierende Anzeige, sortiert z.B. nach Last oder Speicherverbrauch

```
top - 21:07:48 up 70 days,  4:39,  7 users,  load average: 1.16, 1.19, 1.09
Tasks: 155 total,   2 running, 153 sleeping,   0 stopped,   0 zombie
Cpu(s):  3.8% us, 15.4% sy, 78.8% ni,  0.0% id,  1.9% wa,  0.0% hi,  0.0% si
Mem:   1036688k total,   318848k used,   717840k free,    18144k buffers
Swap:  1959920k total,   304240k used,  1655680k free,    97156k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
16376 wml       35  10 24404  13m  12m R 61.6  1.3   0:03.97 wml
 6380 jfranken  15   0 20400 4844 4208 S  3.8  0.5   2:47.88 screen
16387 jfranken  15   0  2068 1116 1852 R  3.8  0.1   0:00.05 top
    1 root      16   0  1504  436 1352 S  0.0  0.0   0:22.22 init
    2 root      34  19     0    0    0 S  0.0  0.0   7:35.80 ksoftirqd/0
    3 root       5 -10     0    0    0 S  0.0  0.0   1:08.75 events/0
    4 root       6 -10     0    0    0 S  0.0  0.0   0:10.05 khelper
```

Listing: top

Tastenbelegung:

Taste	Belegung
k	kill / Signale senden
s.5	Anzeigeaktualisierung schneller (0.5 sec)
M	sortieren nach Speicher
T	sortieren nach CPU
r	renice
o	Sortier-Reihenfolge
f	Spalten hinzufügen
q	beenden
W	Einstellungen speichern in **`.toprc`**

pgrep

pgrep gibt die PIDs der Prozesse aus, auf welche die angegebenen Filterkriterien zutreffen.

```
$ pgrep mysql
1215
1252
```

Listing: pgrep: Filter nach Prozessname

```
$ pgrep -u jfranken
1336
5795
6379
6380
[...]
```

Listing: pgrep -u: Filter nach User

```
$ pgrep -t pts/14
12982
```

Listing: pgrep -t: Filter nach Konsole

Beispiel:

```
$ ps -fp ‘pgrep mysql‘
UID     PID  PPID  C STIME TTY  STAT TIME CMD
root   1215     1  0  2006 ?    S    0:00 /bin/sh /usr/bin/mysqld_safe
mysql  1252  1215  0  2006 ?    S    0:04 /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql -
```

Listing: pgrep und ps verbinden

Signale an Prozesse senden

kill

kill sendet ein Signal an einen Prozess.

Aufruf: kill *-Signal PID*

Als *Signal* geben Sie entweder eine Signalnummer oder ein Signalkürzel (ohne **SIG** am Anfang) aus folgender Liste an:

```
$ kill -l|sed -e s/SIG//g
 1) HUP          2) INT          3) QUIT         4) ILL
 5) TRAP         6) ABRT         7) BUS          8) FPE
 9) KILL        10) USR1        11) SEGV        12) USR2
13) PIPE        14) ALRM        15) TERM        17) CHLD
18) CONT        19) STOP        20) TSTP        21) TTIN
22) TTOU        23) URG         24) XCPU        25) XFSZ
26) VTALRM      27) PROF        28) WINCH       29) IO
30) PWR         31) SYS         33) RTMIN       34) RTMIN+1
[...]
```

Listing: kill -l: Anzeige möglicher Signale

Interressant sind insb. folgende Signale:

Nr.	Kürzel	Beschreibung
15	TERM	Terminate: Prozess sauber beenden (Default-Signal)
9	KILL	Kill: Prozess sofort abschießen (kann nicht abgefangen werden)
1	HUP	Hangup: Konfiguration neu einlesen
19	STOP	Stop: Prozess anhalten (in der Shell: Strg-Z, kann nicht abgefangen werden)
18	CONT	Continue: Angehaltenen Prozess weiterlaufen lassen

Beispiel: alle Prozesse eines Users abschießen:

```
$ kill -9 ‘pgrep -u jfranken‘
```

Listing: kill -9

pkill, killall

pkill kombiniert pgrep und kill:

```
$ pkill -9 -u jfranken
```

Listing: pkill

Auch mit **killall** kann man Signale an Prozesse senden, die dem übergebenen Filterkriterium entsprechen:

```
$ killall --help
usage: killall [ OPTIONS ] [ -- ] name ...
       killall -l, --list
       killall -V --version

  -e,--exact          require exact match for very long names
  -I,--ignore-case-   case insensitive process name match
  -g,--process-group  kill process group instead of process
  -i,--interactive    ask for confirmation before killing
  -l,--list           list all known signal names
  -q,--quiet          don’t print complaints
  -s,--signal         send signal instead of SIGTERM
  -v,--verbose        report if the signal was successfully sent
  -V,--version        display version information
  -w,--wait           wait for processes to die
```

Listing: killall

Beispiel:

```
$ killall -9 mysql
```

Listing: killall-Beispiel

Prozesse priorisieren

Unter Unix ist jedem Prozess ein "nice" (Nettigkeits)-Wert zugeordnet. Durch Manipulation dieser Wertes nehmen Sie Einfluss auf die gegenseitigen Gewichtungen zwischen den Prozessen: Wenn das System überlastet ist, werden die "netteren" Prozesse zuerst abgebremst.
Der "nice"-Wert liegt zwischen **-20** (höchste Priorität) und **+19** (nur dann ausführen, wenn nichts anderes zu tun ist/sehr nett). **0** ist normal.

Achtung:
Jeder User darf die nice-Werte seiner eigenen Prozesse erhöhen, aber nur **root** darf negative nice-Werte anfordern.

nice

Mit dem **nice**-Kommando starten Sie einen neuen Prozess mit dem "nice"-Wert Ihrer Wahl.

Beispiel 1:
Datei komprimieren (nice-Wert: 10 ist Default)

```
$ nice gzip /var/log/www/access.log
```

Listing: nice

Beispiel 2:
Rechenaufgabe beschleunigen (Primfaktorzerlegung der größten 64bit-Primzahl):

```
$ nice gzip /var/log/www/access.log
```

Listing: nice

renice

Das **renice**-Kommando ändert den "nice"-Wert eines laufenden Prozesses.

Beispiel 1 (PID angeben):

```
$ renice +10 -p 6573
6573: old priority 0, new priority 10
```

Listing: nice

Beispiel 2 (alle Prozesse eines Users beschleunigen)

```
$ renice 20 -u jfranken
1000: old priority 0, new priority -20
```

Listing: nice

Hauptspeichernutzung

free

Einmalige Abfrage:

```
$ free
             total       used       free     shared    buffers     cached
Mem:        127024      34060      92964          0       2860      17784
-/+ buffers/cache:      13416     113608
Swap:       369452          0     369452
```

Listing: free

Interpretation: Es sind noch 111 MB RAM frei, nämlich 92964kB ("free") + 17784kB ("cached"). Auf der Auslagerungspartition (langsam!) stehen weitere 369 MB zur Verfügung.

vmstat

Fortlaufende Anzeige:

```
$ vmstat 1 100
procs -----------memory---------- ---swap-- -----io---- -system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa
 0  0      0  92984   2876  17812    0    0    31    10  259   25  0  3 94  2
 0  0      0  92984   2876  17812    0    0     0     0  252    6  0  0 100  0
 0  0      0  92956   2892  17812    0    0     8    36  258   16  0  0 94  6
...
```

Listing: vmstat

Parameter: [Intervall] [Anzahl]

Eingeloggte User

id

Wer bin ich?

```
$ id
uid=1001(jfranken) gid=1001(jfranken) Gruppen=100(users),1001(jfranken)
```

Listing: id

w, finger

Wer ist eingeloggt?

```
$ w
 22:19:23 up 23 min,  4 users,  load average: 0,11, 0,06, 0,02
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
root     tty1     -                21:59   15:30m  0.43s  0.21s -bash
root     tty2     -                22:00   17:35m  0.27s  0.21s pppoe
root     pts/0    gate.jfranken.de 21:58   20:54m  0.03s  0.03s -bash
root     pts/1    192.168.230.1    22:04    0.00s  0.55s  0.02s w
$ finger
Login     Name       Tty      Idle  Login Time   Office     Office Phone
root      root      *tty1       15  Apr 22 21:59
root      root      *tty2       17  Apr 22 22:00
root      root      *pts/0      20  Apr 22 21:58 (gate.jfranken.de)
root      root      *pts/1          Apr 22 22:04 (192.168.230.1)
```

Listing: w und finger

last

Wer *war* zuletzt eingeloggt:

```
$ last
root     pts/1        192.168.230.1    Sun Apr 22 22:04   still logged in
root     tty2                          Sun Apr 22 22:00   still logged in
root     tty1                          Sun Apr 22 21:59   still logged in
root     pts/0        gate.jfranken.de Sun Apr 22 21:58   still logged in
root     pts/0        gate.jfranken.de Sun Apr 22 21:57 - 21:57  (00:00)
reboot   system boot  2.6.18-4-686     Sun Apr 22 21:56 - 22:21  (00:24)
root     pts/0        gate.jfranken.de Sun Apr 22 21:39 - down   (00:16)
```

Listing: last

Arbeiten mit Dateien und Verzeichnissen

Das "aktuelle Verzeichnis" gibt den Startpunkt für relative Pfadanagaben vor. Sie können das "aktuelle Verzeichnis" mit dem **cd**-Befehl einstellen und mit dem **pwd**-Befehl abfragen.

Beispiel:
Wenn das aktuelle Verzeichnis auf **/etc/init.d** eingestellt ist, bewirkt der Aufruf von **cat networking** die Ausgabe von **/etc/init.d/networking**.

Das aktuelle Verzeichnis kann mit **pwd** angezeigt und mit **cd** gewechselt werden.

Aktuelles Verzeichnis

pwd

Beispiele:

```
$ pwd
/home/jfranken
$ echo $PWD
/home/jfranken
```

*Listing: **pwd**: Anzeige des aktuellen Verzeichnisses*

cd

```
$ cd ..
$ cd /
$ cd /etc/init.d
```

*Listing: **cd**: Das aktuelle Verzeichnis wecshseln*

```
$ cd
$ cd ~
$ cd $HOME
$ cd ~$USERNAME
```

*Listing: **cd**: ins Homeverzeichnis wechseln*

```
$ cd -
$ cd $OLDPWD
```

*Listing: **cd**: ins vorherige Verzeichnis wechseln*

Dateien auflisten und suchen

ls

Der **ls**-Befehl gibt den Inhalt eines Verzeichnisses aus. Sie können dem **ls**-Befehl verschiedene Parameter und eine Liste zu betrachtender Filesystem-Elemente übergeben. Wenn Sie *keine* Liste übergeben, gibt **ls** alle Filesystem-Elemente des aktuellen Verzeichnisses aus, die nicht mit einem Punkt beginnen.

Beispiele:

```
$ cd /
$ ls
bin    dev   initrd      lost+found  opt   sbin     sys  var
boot   etc   initrd.img  media       proc  selinux  tmp  vmlinuz
cdrom  home  lib         mnt         root  srv      usr
$ ls /var/log
acpid       debug       exim4      lpr.log    news                wtmp
aptitude    dmesg       faillog    mail.err   ppp-connect-errors
auth.log    dmesg.0     fsck       mail.info  pycentral.log
boot        dmesg.1.gz  installer  mail.log   syslog
btmp        dmesg.2.gz  kern.log   mail.warn  user.log
daemon.log  dpkg.log    lastlog    messages   uucp.log
```

Listing: ls

```
$ ls -l /var/log
insgesamt 480
drwxr-s--- 2 Debian-exim adm    4096 2007-04-09 23:39 exim4
drwxr-xr-x 2 root        root   4096 2007-04-09 23:24 fsck
-rw-r--r-- 1 root        root  57789 2007-04-22 22:02 kern.log
-rw-rw-r-- 1 root        utmp 292584 2007-04-22 22:42 lastlog
-rw-r--r-- 1 root        root      0 2007-04-09 23:39 lpr.log
-rw-r--r-- 1 root        root  52173 2007-04-22 22:56 messages
-rw-r--r-- 1 root        root    225 2007-04-22 22:03 ppp-connect-errors
-rw-r----- 1 root        adm   65741 2007-04-22 22:59 syslog
-rw-rw-r-- 1 root        utmp  31104 2007-04-22 22:42 wtmp
[...]
```

Listing: ls -l: Details

```
$ ls -l /root
insgesamt 0
$ ls -la /root
insgesamt 36
drwxr-xr-x  4 root root 4096 2007-04-22 23:07 .
drwxr-xr-x 22 root root 4096 2007-04-09 23:34 ..
drwx------  2 root root 4096 2007-04-09 23:37 .aptitude
-rw-------  1 root root 1657 2007-04-22 21:57 .bash_history
-rw-r--r--  1 root root  412 2004-12-15 23:53 .bashrc
-rw-------  1 root root   86 2007-04-22 21:49 .lesshst
-rw-r--r--  1 root root  110 2004-11-10 17:10 .profile
drwx------  2 root root 4096 2007-04-22 21:58 .ssh
-rw-------  1 root root  970 2007-04-22 21:48 .viminfo
```

Listing: ls -a: Ausgabe versteckter Elemente

Mit dem Parameter **-d** aufgerufen, gibt **ls** bei Verzeichnissen den *Eintrag* statt des Inhalts aus:

```
$ ls -ld /root
drwxr-xr-x 4 root root 4096 2007-04-22 23:07 /root
```

Listing: ls -d: Verzeichniseintrag

Die Ausgabe von **ls** ist normalerweise alphabetisch aufsteigend nach dem Namen sortiert. Mit dem Parameter **-r** können Sie die Reihenfolge umkehren:

```
$ ls -r /var/log
wtmp                news        lpr.log    exim4       debug       acpid
uucp.log            messages    lastlog    dpkg.log    daemon.log
user.log            mail.warn   kern.log   dmesg.2.gz  btmp
syslog              mail.log    installer  dmesg.1.gz  boot
pycentral.log       mail.info   fsck       dmesg.0     auth.log
ppp-connect-errors  mail.err    faillog    dmesg       aptitude
```

Listing: ls -r: absteigend sortiert

Mit dem Parameter **-t** aufgerufen, gibt **ls** die Einträge in der Reihenfolge ihrer letzten Änderung aus:

```
$ ls -ltr /var/log
insgesamt 480
drwxr-xr-x 2 root        root   4096 2007-04-09 23:24 fsck
-rw-r--r-- 1 root        root      0 2007-04-09 23:39 lpr.log
drwxr-s--- 2 Debian-exim adm    4096 2007-04-09 23:39 exim4
-rw-r--r-- 1 root        root  57789 2007-04-22 22:02 kern.log
-rw-r--r-- 1 root        root    225 2007-04-22 22:03 ppp-connect-errors
-rw-rw-r-- 1 root        utmp  31488 2007-04-22 23:16 wtmp
-rw-rw-r-- 1 root        utmp 292584 2007-04-22 23:16 lastlog
-rw-r--r-- 1 root        root  52208 2007-04-22 23:16 messages
-rw-r----- 1 root        adm   66077 2007-04-22 23:17 syslog
[...]
```

Listing: ls -t: nach Datum sortieren

Sie können auch nach der dem Zeitpunkt des letzten Lesezugriffs sortieren:

```
$ ls  -lr --sort=atime /var/log
insgesamt 480
drwxr-s--- 2 Debian-exim adm    4096 2007-04-22 23:31 exim4
drwxr-xr-x 2 root        root   4096 2007-04-22 23:31 fsck
drwxr-xr-x 3 root        root   4096 2007-04-22 23:31 installer
-rw-r--r-- 1 root        root  57789 2007-04-09 23:39 kern.log
-rw-rw-r-- 1 root        utmp 292584 2007-04-22 23:16 lastlog
-rw-r--r-- 1 root        root  52243 2007-04-22 22:02 messages
-rw-r--r-- 1 root        root    225 2007-04-22 21:55 ppp-connect-errors
-rw-r----- 1 root        adm   66572 2007-04-09 23:25 syslog
-rw-rw-r-- 1 root        utmp  31488 2007-04-22 22:21 wtmp
[...]
```

Listing: ls --sort=atime

Einige Versionen von **ls** (insb. Version 5.97) akzeptieren statt dessen den Parameter **-u**. Sie können die Ausgabe auch nach der Dateigröße sortieren:

```
$ ls  -lr --sort=size /var/log
insgesamt 480
-rw-r--r-- 1 root        root      0 2007-04-09 23:39 lpr.log
-rw-r--r-- 1 root        root    225 2007-04-22 22:03 ppp-connect-errors
drwxr-xr-x 2 root        root   4096 2007-04-09 23:24 fsck
drwxr-s--- 2 Debian-exim adm    4096 2007-04-09 23:39 exim4
-rw-rw-r-- 1 root        utmp  31488 2007-04-22 23:16 wtmp
-rw-r--r-- 1 root        root  52243 2007-04-22 23:36 messages
-rw-r--r-- 1 root        root  57789 2007-04-22 22:02 kern.log
-rw-r----- 1 root        adm   66342 2007-04-22 23:36 syslog
-rw-rw-r-- 1 root        utmp 292584 2007-04-22 23:16 lastlog
```

Listing: ls --sort=size

Die Größe von Verzeichniseinträgen ist abhängig von der Gesamtlänge der direkt darin enthaltenen Einträge. Mit dem Parameter **-R** aufgerufen, gibt **ls** zusötzlich die Eitnträge aller Unterverzeichnisse aus:

```
$ ls -lR /var/log
/var/log/:
insgesamt 480
drwxr-s--- 2 Debian-exim adm    4096 2007-04-09 23:39 exim4
drwxr-xr-x 2 root        root   4096 2007-04-09 23:24 fsck
-rw-r--r-- 1 root        root  57789 2007-04-22 22:02 kern.log
-rw-rw-r-- 1 root        utmp 292584 2007-04-22 22:42 lastlog
[...]

/var/log/exim4:
insgesamt 4
-rw-r----- 1 Debian-exim adm 1120 2007-04-22 23:26 mainlog

/var/log/fsck:
insgesamt 8
-rw-r----- 1 root adm 122 2007-04-22 21:56 checkfs
-rw-r----- 1 root adm 195 2007-04-22 21:56 checkroot
```

Listing: ls -R: Unterverzeichnisse mit ausgeben

find

Mit dem Programm **find** können Sie Dateien im Filesystem suchen.

Eine Minimalinstallation von Debian enthält bereits 25.000 Dateien in 3.000 Verzeichnissen. Auf einem ausgewachsenen Serversystem finden sich z.B. 600.000 Dateien in 20.000 Verzeichnissen. Beispiel:

```
$ find / -name "hosts"
/etc/hosts
```

Listing: find

Syntax:
find <Verzeichnis> [<Filteroptionen>] [<Ausgabeoptionen>] **Filteroptionen**

Option	sucht alle Filesystemelemente,...
-name "*Dateiname*"	deren Name mit dem genau angegebenen übereinstimmen
-name "*Vergleichsmuster*"	deren Name mit dem angegebenen Muster beschrieben werden kann (z.B. h*sts)
-mtime -*Tage*	die innerhalb der letzten Tage verändert wurden.
-mtime +*Tage*	deren letzte Änderung schon mindestens soviel Tage zurückliegt.
-mtime *Tage*	die genau *Tage* alt sind.
-type d	die ein Verzeichnis sind
-type f	die eine Datei sind
-type l	die ein symbolischer Link sind

Logikoptionen Sie können auch *mehrere* Filteroptionen angeben. Dann gibt **find** nur die Filesystemelemente aus, für die *alle* Optionen gleichzeitig zutreffen (Schnittmenge),

Beispiel:

```
$ find /etc -type l -name "*netw*"
/etc/rc6.d/S35networking
/etc/rcS.d/S40networking
/etc/rc0.d/S35networking
```

*Listing: **find**: mehrere Optionen*

Mit dem zusätzlichen Parameter **-or** gibt **find** all die Filesystemelemente aus, für die *mindestens eine* Filteroption zutrifft (Vereinigungsmenge). **Ausgabeoptionen** Mit der Option **-ls** erhalten Sie eine detailiertere Ausgabe:

```
$ find /etc -type l -name "*netw*" -ls
745191  0 lrwxrwxrwx   1 root root 20 Apr  9  2007 /etc/rc6.d/S35networking -& ../init.d/networking
745192  0 lrwxrwxrwx   1 root root 20 Apr  9  2007 /etc/rcS.d/S40networking -& ../init.d/networking
745190  0 lrwxrwxrwx   1 root root 20 Apr  9  2007 /etc/rc0.d/S35networking -& ../init.d/networking
```

*Listing: **find -ls***

Mit der Option **-exec *Kommando* \\;** bringen Sie **find** dazu, für jeden Treffer das angegebene Kommando aufzurufen. Mit {} können Sie die genaue Fundstelle als Parameter übergeben.

```
$ find / -name "*~" -exec rm -i {} \;
rm: reguläre Datei »/boot/grub/menu.lst~« entfernen? n
rm: reguläre Datei »/etc/apt/trusted.gpg~« entfernen? n
```

*Listing: **find -exec***

Achtung:
Vergessen Sie nicht das abschließende Backslash-Semikolon (**\;**)!

Es kann zu Problemen führen, wenn Dateinamen Zeilenumbrüche enthalten. In diesem Fall verwenden Sie bitte **find . -print0 | xargs -0 *Befehl***.

locate, updatedb

Die Suche mit **find** über größere Verzeichnisbäume kann einige Zeit in Anspruch nehmen.

Auf den meisten Linuxsystemen befindet sich ein Index (Liste) aller Dateinamen, welcher täglich aktualisiert wird.

```
$ locate 'sshd'
/etc/ssh/sshd_config
/usr/sbin/sshd
/usr/share/man/man5/sshd_config.5.gz
/usr/share/man/man8/sshd.8.gz
/var/run/sshd
/var/run/sshd.pid
```

*Listing: **locate***

Die Aktualisierung des Index erfolgt mit dem Programm **updatedb**. Dieses wird normalerweise automatisch jede Nacht aufgerufen, z.B. bei Debian über **/etc/cron.daily/find**.

df, du

Disk free:

```
$ df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/work-root
                       30G  4,3G   24G  16% /
tmpfs                 753M     0  753M   0% /lib/init/rw
udev                   10M  104K  9,9M   2% /dev
tmpfs                 753M     0  753M   0% /dev/shm
/dev/sda1             236M   26M  199M  12% /boot
/dev/mapper/export-export
                      1,0T  824G  201G  81% /export

$ df -Pm /var
Filesystem         1048576-blocks      Used Available Capacity Mounted on
/dev/mapper/work-root        29927      4402     24006      16% /
```

Listing: df

Disk Usage:

```
$ du -s /var/* | sort -n
4       /var/agentx
4       /var/local
4       /var/mail
4       /var/opt
4       /var/tmp
16      /var/lock
124     /var/www
664     /var/run
752     /var/spool
4992    /var/backups
9512    /var/log
35584   /var/cache
118100  /var/lib
```

Listing: du

Operationen auf Filesystem-Elementen

mkdir

Mit dem Kommando **mkdir** können Sie neue Verzeichnisse anlegen.

```
$ mkdir /tmp/dir1 /tmp/dir1/dir2
```

Listing: mkdir

Mit dem Parameter **-p** aufgerufen, legt **mkdir** zusätzlich weitere Verzeichnisse an, die "auf dem Weg " fehlen.

```
$ mkdir -p /tmp/dir1/dir2
```

Listing: selbes Ergebnis, mit -p

cp

Mit dem Kommando **cp** können Sie Filesystem-Elemente (z.B. Dateien und Verzeichnisse) an eine andere Stelle im Filesystem kopieren. Die Quelle bleibt dabei erhalten. Beispiele:

```
$ # Datei innerhalb eines Verzeichnisses kopieren:
$ cp /etc/passwd userliste

$ # Datei in ein anderes Verzeichnis kopieren:
$ cp /etc/passwd /tmp/
```

Listing: cp

Weitere wichtige Parameter:

Parameter	Beschreibung
-R	"Recursive": Ein Verzeichnis inkl. aller Unterverzeichnisse kopieren
-p	"Preserve Permissions": Besitzer/Gruppe/Zeitstempel mitkopieren
-d	"No-dereference": Symlinks als solche kopieren
-a	"archive": entspricht -dpR

Beispiel: Vollständige Kopie eines Verzeichnisses erzeugen:

```
$ cp -a /etc /tmp/etc.sicher
```

Listing: cp -a

mv

Mit dem Kommando **mv** können Sie Filesystem-Elemente (z.B. Dateien und Verzeichnisse) an eine andere Stelle im Filesystem verschieben. Die Quelle wird dabei entfernt. Beispiele:

```
$ # Datei umbenennen:
$ cd /etc
$ mv motd motd.old

$ # ist identisch zu:
$ mv /etc/motd{,.old}

$ # Eine Datei in ein anderes Verzeichnis verschieben:
$ mv /var/log/messages /tmp

$ # Mehrere Dateien und Verzeichnisse verschieben:
$ mv /var/log/messages* /etc.sicher /tmp
```

Listing: mv

rm, shred

Mit dem Kommando **rm** ("remove") können Sie Filesystem-Elemente (z.B. Dateien und Verzeichnisse) löschen.

Beim Löschen einer *Datei* wird der Plattenplatz erst dann freigegeben, wenn alle Prozesse die Datei geschlossen haben und alle ggf. vorhandenen weiteren Hardlinks, die auf denselben Dateiinhalt verweisen, beseitigt sind.

Beispiel:

```
$ # Löschen einer Datei
$ rm ~/.bash_history

$ # Löschen mehrerer Dateien:
$ rm /var/log/messages.* /tmp/*

$ # Löschen ganzer Verzeichnisse
$ rm -Rf /usr/src/kernel-2.6.8
```

Listing: rm

Mit der Option **-f** ("force") vermeiden sie Rückfragen ("Sind sie sicher?"), die ggf. durch die vom Alias gesetzte Option **-i** ("interrogate") hervorgerufen werden.

rm löscht nur den Eintrag aus dem Filesystem. Der Dateiinhalt bleibt zunächst auf der Festplatte erhalten und wird erst später mit neuen Inhalten überschrieben. Um ganz sicher zu gehen, dass in der Zwischenzeit niemand den Dateiinhalt wieder herstellen kann, verwenden Sie **shred -u**, welches die Datei vor dem Löschen mehrmals mit Zufallszeichen überschreibt.

ln

Mit dem Kommando **ln -s** erstellen Sie einen "symbolischen Link" auf ein (hoffentlich) vorhandenes Filesystem-Element.

Syntax: **ln -s Quelle Ziel** legt einen Symlink an, der den Namen **Ziel** hat und auf das Filesystem-Element **Quelle** verweist.

Beispiel:

```
$ cd    # Wechselt ins Home-Verzeichnis
$ ln -s /var/log/messages abk
$ ls -l abk
lrwxrwxrwx 1 root root 17 2007-10-04 09:13 abk -> /var/log/messages
$ less abk
```

Listing: ln

Wenn das Ziel bereits existiert, müssen Sie es zuerst löschen oder verwenden Sie **-f**, um es mit einem neuen Symlink zu ersetzen.

```
$ ln -s /var/log/syslog abk
ln: Erzeugen der symbolischen Verknüpfung »Log« zu »/var/log/syslog«: Die Datei existiert bereits
$ ln -sf /var/log/syslog abk
```

Listing: ln -sf

Archive packen

tar

tar steht für *tape archiver*. Ursprünglich für Filesystem-Backups auf Magnetband vorgesehen, ermöglicht **tar** auch den Einsatz sog. "Archivdateien".

tar hat folgende Argumente: Kommando:

Arg.	Beschreibung
c	*create*: mehrere Dateien und Verzeichnisse zu einer "Archivdatei" zusammenfassen (taren) und diese anschließend komprimieren (zippen)
x	*extract*: die in der Archivdatei gespeicherten Dateien und Verzeichnisse in das Filesystem übernehmen.
t	*type*: den Inhalt einer Archivdatei anzeigen (Liste der Dateien)

Dateiname:

Arg.	Beschreibung
f *dateiname.tar*	*file*: Name der Architdatei
f -	auf Konsole statt Datei (zur Weiterverarbeitung in Pipes)

Optionen:

Arg.	Beschreibung
v	*verbose*: Liste der Dateien (bei c,x) oder zusätzlich Größe/Datum (bei t) ausgeben
z	*zip*: Architdatei mit gzip (de)komprimieren (.tar.gz = .tgz)
j	Architdatei mit bzip2 (de)komprimieren (.tar.bz = .tbz). stärker aber langsamer als gzip

Beispiele:

Aufruf	Beschreibung
`tar cf mein.tar Datei1 Datei2 Datei3`	einpacken
`tar cvf mein.tar mein/`	komplettes Unterverzeichnis einpacken (verbose)
`tar xvf mein.tar`	auspacken (verbose)
`tar xf - < mein.tar`	auspacken (über Stdin)
`tar tvf mein.tar`	Dateiliste

Alternative: **`cpio`**

Dateien komprimieren

- gzip Datei, gunzip Datei.gz, zcat Datei.gz, zless Datei.gz, zgrep, ...
- bzip2 Datei, bunzip2 Datei.bz, bzcat Datei.bz, ...
- div. weitere (PKZIP: zip, unzip)

Weitere Werkzeuge

dd

`dd` steht für *data duplicator*. Ursprünglich für das Kopieren von Magnetband-Abschnitten vorgesehen, kann **`dd`** heute heute auch zum Einlesen von CDs oder zum schnellen kopieren von Festplatten über das Netz eingesetzt werden. Aufruf: **`dd if=`**Eingabedatei **`of=`**Ausgabedatei **`bs=`**Blocksize

`lsof`, `fuser`

`lsof` steht für "list open files".

- **`lsof -u User`**: Offene Files eines Users
- **`lsof +D /tmp`**: Offene Files unterhalb /tmp
- **`lsof Datei`**: wer hat diese Datei geöffnet
- **`lsof +L1`**: Liste der gelöschten aber noch offenen Files
- **`fuser -km /mnt`**: killt alle Prozesse, die noch dateien auf mnt offen haben