1.103.5-6 Prozesse erzeugen, überwachen, killen und Prozessprioritäten ändern

www.lpi.org

Shell Jobs

Auf dem Weg
Certified Linu

Innerhalb einer Shell können mehrere *jobs* (Programme) parallel abgearbeitet werden. Einer läuft dabei im Vordergrund, alle anderen im Hintergrund.

Tastenkombinationen

STRG C	Beendet den Prozess der im Vordergrund läuft.
STRG Z	Hält den im Vordergrund laufenden Prozess an (Suspend Job)

Befehle (Shell builtins)

bg [jobnr]	Schiebt den im Vordergrund laufenden bzw. angehaltenen Prozess in den Hintergrund
fg [jobnr]	Schiebt den im Hintergrund laufenden Prozess in den Vordergrund
jobs [option] [jobnr]	zeigt alle *jobs* der Shell an (opt.: -l => PID anzeigen)
'Kommando' &	das &-Zeichen am Ende eines Kommandos schiebt den startenden Prozess sofort in den Hintergrund

Shell Jobs 2

Beispiel

```
[root@abc ~]# firefox
CTRL-Z
[1]+  Stopped
[root@abc ~]#kcalc
CTRL-Z
[2]+  Stopped
[root@abc ~]#jobs
[1]-  Stopped      firefox
[2]+  Stopped      kcalc
[root@abc ~]#kedit &
[root@abc ~]#jobs
[1]-  Stopped      firefox
[2]+  Stopped      kcalc
[3]   Running      kedit
[root@abc ~]#
```

Beispiel (Fortsetzung)

```
[root@abc ~]#bg 1
[root@abc ~]#jobs
[1]   Running      firefox
[2]+  Stopped      kcalc
[3]-  Running      kedit
[root@abc ~]#fg
kcalc
CTRL-C
[root@abc ~]#
```

aktueller job

nachfolgender job

Aufgaben

- Kontrollieren Sie im Folgenden immer wieder Ihre laufenden Jobs mit *jobs*
- Starten Sie auf der Konsole *firefox*
- Halten Sie den Prozess *firefox* an
- Starten Sie auf der selben Konsole den Prozess *thunderbird* und schieben ihn sofort in den Hintergrund
- Starten Sie weitere jobs (z.B. *Open Office*)
- Schieben Sie die Jobs in den Hintergrund und holen Sie immer einen anderen in den Vordergrund
- Beenden Sie die Shell so lange noch gestoppte Prozesse vorhanden sind. Was passiert?
- Beenden Sie die Shell erneut. Was passiert nun mit Ihren Jobs?

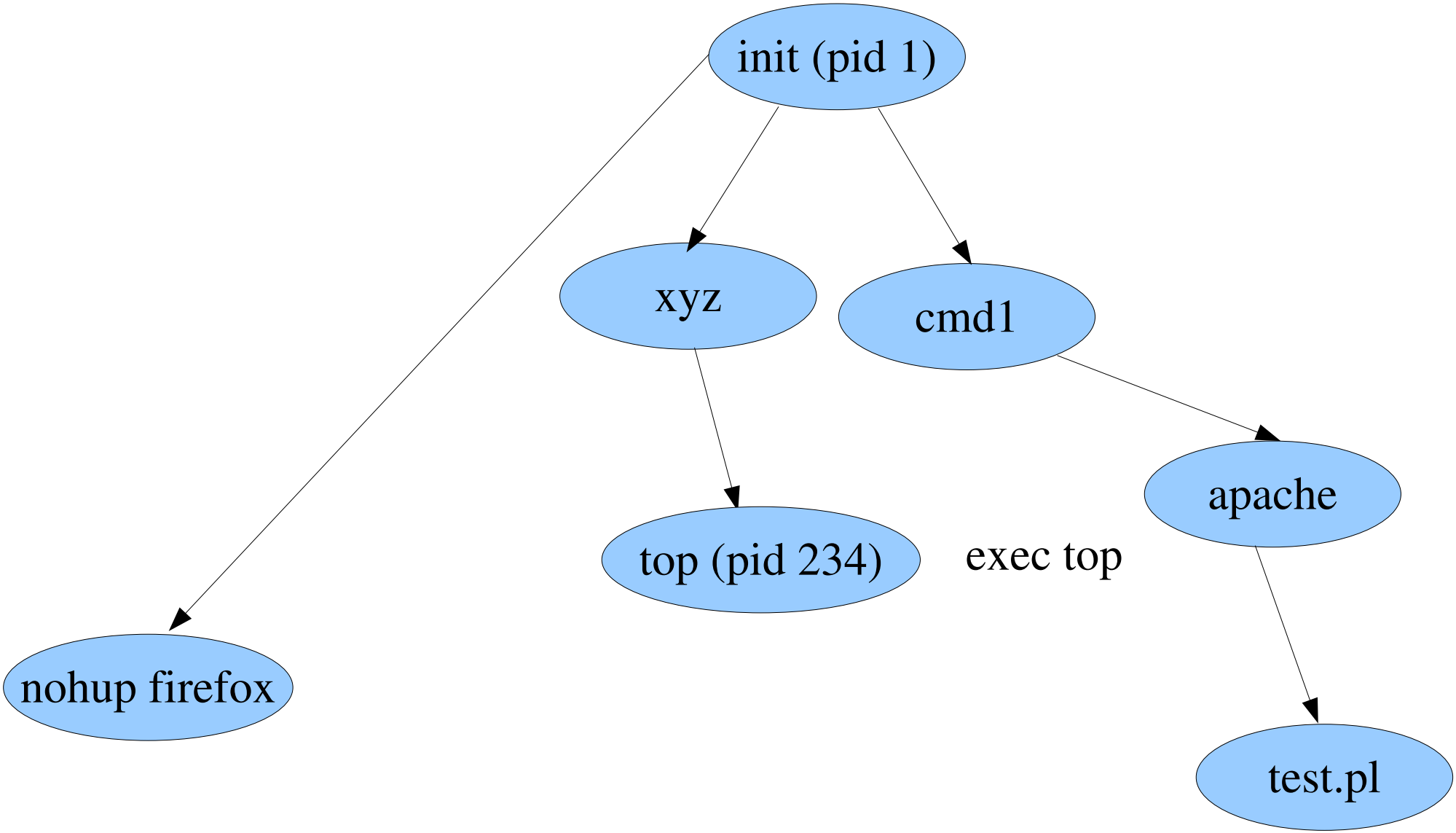
Wird die **Shell beendet** werden auch alle **Jobs** dieser Shell **beendet**, da diese **Kindprozesse** der Shell waren.

I.d.R. werden alle Kindprozesse beendet, wenn der Elternprozess beendet wird. Soll dies nicht der Fall sein, so kann man das Shell-Kommando nohup (no hangup) verwenden.

Beispiel:
nohup firefox

Erster Prozess
Der erste Prozess, der auf einem Linux-Sytem gestartet wird ist der Prozess **init**. Deswegen erhält er immer die Process Id (**PID**) mit der Nummer **1**. Dieser startet alle weiteren Prozesse des Systems.

Prozesshierarchie

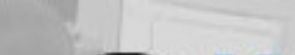


Begriffe

Prozess	Ausführung eines Programms auf einem Prozessor
Task	Synonym für einen Prozess
Multitask	mehrere (quasi)parallel ablaufende Prozesse
Multiprocessor	mehrere Prozessoren
Process Id (PID)	eigene eindeutige Nummer für jeden (laufenden) Prozess
User Id (UID), Group Id (GID)	unter welchen Benutzer und Gruppe läuft der Prozess
Parent Process Id (PPID)	Welcher Prozess hat den aktuellen Prozess gestartet (-> Prozess-Hierarchie)
Environment	Umgebung bzw. Umgebungsvariablen
Current Working Directory	Verzeichnis, in dem der Prozess abgearbeitet wird

Diese Infos finden sich im Verzeichnis /proc/{PID} z.B. in /proc/1 für den ersten Prozess in einem Linux-System => init.

ps – Process Status



Syntax:
ps [options]

Zeigt den momentanen Prozeßstatus aller aktuellen Prozesse an

Häufige Optionen (Schalter)

-a all, alle Prozesse aller User werden angezeigt, sofern sie in einem Terminal laufen
-f forest, die Prozesshierarchie wird als „Wald“ angezeigt
-l long, ausführlichere Anzeige (Priorität, PPID, ...)
-u user, Anzeige des Prozessbesitzers
-w wide, weite Ausgabe (Verschiebung des Zeilenumbruchs)
-x anzeige von Prozessen, welche ohne Terminal laufen (Dämonen)
-C cmd zeigt nur Prozesse des Kommandos cmd (Alternative ist Filterung des Outputs mit 'grep' => `ps aux | grep bash`
-U usr zeigt nur Prozesse des Users usr

Beispiel: `ps aux` (BSD-Stil), **`ps -aux`** (Unix-Variante) bringt Warnung da eigentlich andere Bedeutung der Optionen

Beispiel:

```
root@abc ~]# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.1  0.3   1744   592 ?        S    17:09   0:02 init [3]
...
root      1418  0.0  0.2   1572   508 ?        Ss   17:10   0:00 klogd -x
rpc       1434  0.0  0.3   1700   604 ?        Ss   17:10   0:00 portmap
rpcuser   1451  0.0  0.4   1844   868 ?        Ss   17:10   0:00 rpc.statd
root      2556  0.0  0.7   4624  1436 pts/1    Ss+  17:27   0:00 /bin/bash
...
root      2686  0.7  0.7   4620  1412 pts/2    Ss   17:37   0:00 -bash
root      2714  0.0  0.4   4728   920 pts/2    R+   17:37   0:00 ps aux
...
[root@abc ~]#
```

Eigentümer

PID

CPU-Last in %

Speicherverwendung
- in %
- virtuell (ges.)
- im RAM

Konsole

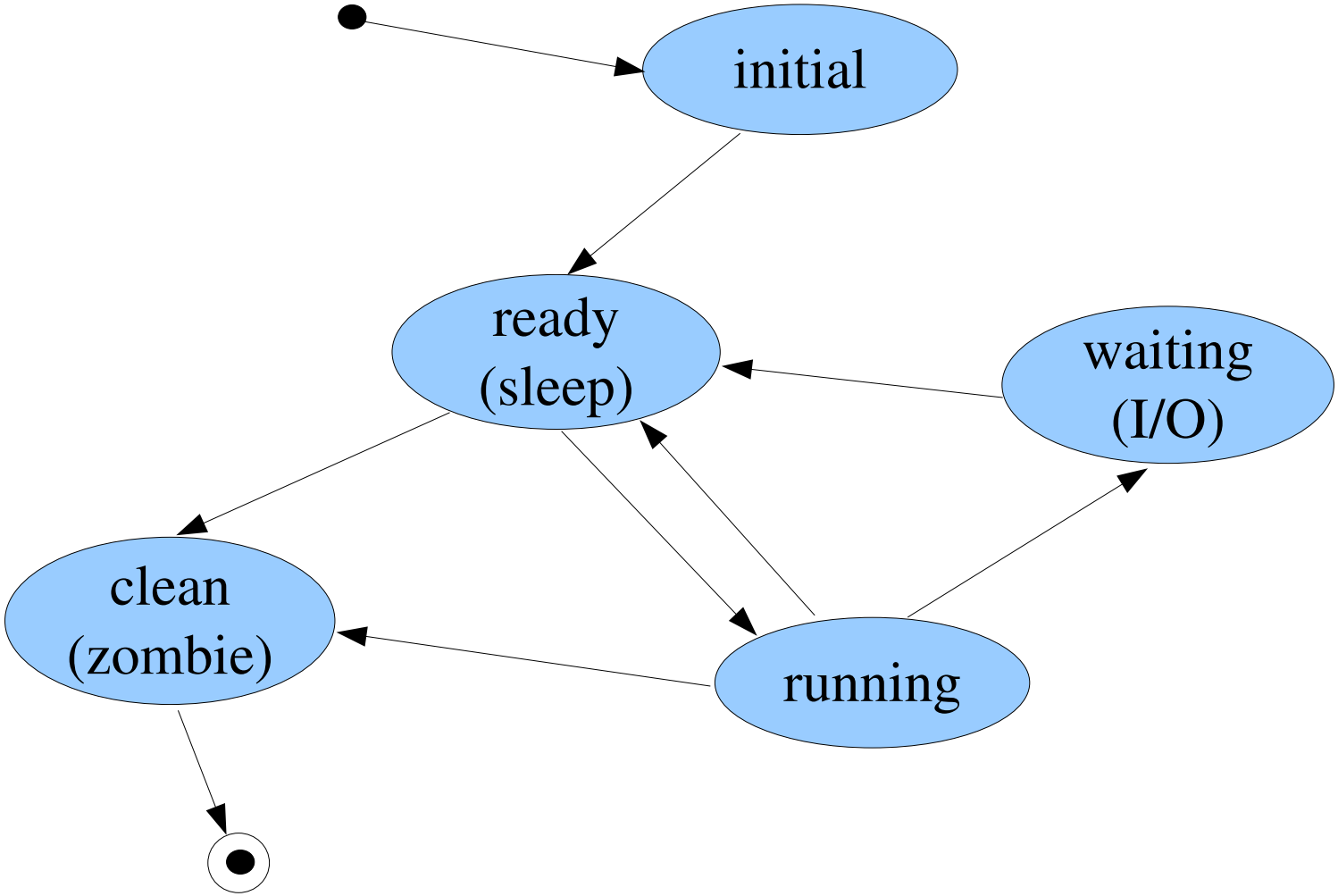
Zustand
S -> sleep
R -> Run

Startzeit

verbrauchte CPU-Zeit

Kommando

Prozesszustände



Stark vereinfacht

Prozesszustände

inital	der Prozess wird initialisiert, z.B. Reservierung von Speicherplatz, laden ins RAM, ...
ready	der Prozess hat gerade nichts zu tun. Er wartet auf ein Ereignis, z.B. eine Tastatureingabe
running	der Prozess wird bearbeitet, d.h. der Scheduler berücksichtigt seine Rechenzeitanforderungen
waiting	der Prozess wartet auf die Beendigung einer I/O-Sequenz, z.B. laden einer Datei, nachladen einer Speicherseite aus der Auslagerungspartition, ...
clean	die Prozessumgebung wird bereinigt, z.B. freigeben von RAM, ...

Syntax:
pstree [options] [pid|user]

Zeigt den Prozeßbaum aller Prozesse an

Häufige Optionen (Schalter)

-a all, zeigt auch die verwendeten Schalter der Kommandos an
-G verwendet VT100 Zeichen für die Baumdarstellung
-p zeigt auch die PID an

```
init─┬─acpid
     ├─auditd─┬─audispd
     │        └─{auditd}
     ├─avahi-daemon
     ├─avahi-dnsconfd
     ├─console-kit-dae───61*[{console-kit-dae}]
     ├─cron
     ├─cupsd
     ├─2*[dbus-daemon]
     ├─dbus-launch
     ├─dcopserver
     ├─dhcpcd
     ├─gconfd-2
     ├─gpg-agent
     ├─hald───hald-runner─┬─hald-addon-acpi
     │                    ├─hald-addon-inpu
     │                    └─hald-addon-stor
     ├─kded
```

Syntax:
top [options]

Zeigt die Prozesse an, wobei die Anzeige regelmäßig erneuert wird

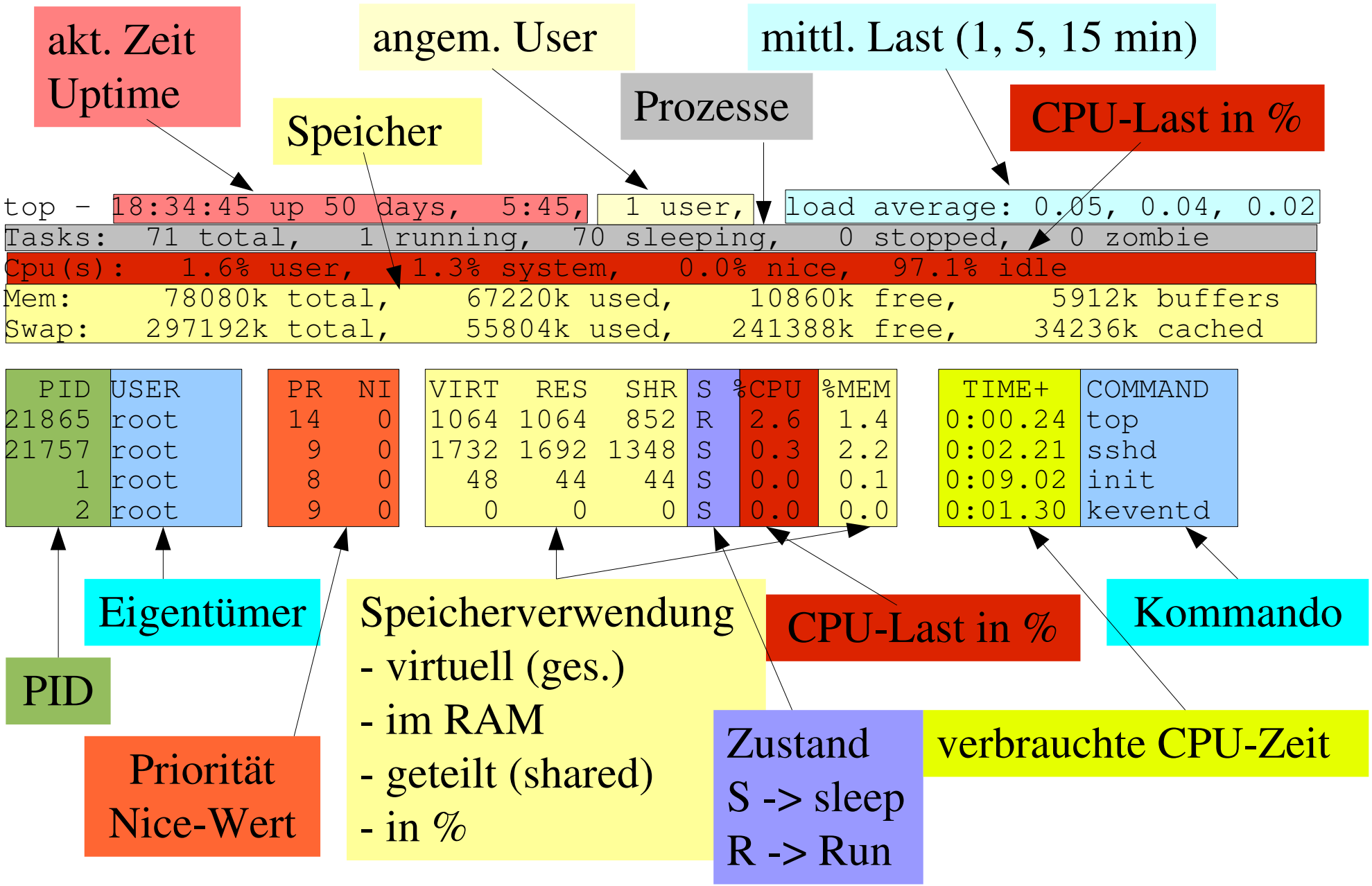
Häufige Optionen (Schalter auf der Kommandozeile)

-d	delay, Verzögerungszeit zwischen 2 Anzeigen (def.: 5 sec)
-i	idle, zeigt keine Idle-Prozesse an, d.h. nur die „running“
-s	secure, nur sichere Befehle werden in der top-Konsole angenommen

Häufige Optionen in der top-Konsole (interaktiv während top läuft)

h	help, zeigt die Hilfe an
k	kill, senden eines Signals an einen Prozess
r	renice, Priorität eines Prozesses ändern
q	quit, beendet die top-Konsole

top


akt. Zeit
Uptime
angem. User
Prozesse
mittl. Last (1, 5, 15 min)
Speicher
CPU-Last in %
top - 18:34:45 up 50 days, 5:45, 1 user, load average: 0.05, 0.04, 0.02
Tasks: 71 total, 1 running, 70 sleeping, 0 stopped, 0 zombie
Cpu(s): 1.6% user, 1.3% system, 0.0% nice, 97.1% idle
Mem: 78080k total, 67220k used, 10860k free, 5912k buffers
Swap: 297192k total, 55804k used, 241388k free, 34236k cached
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
21865 root 14 0 1064 1064 852 R 2.6 1.4 0:00.24 top
21757 root 9 0 1732 1692 1348 S 0.3 2.2 0:02.21 sshd
1 root 8 0 48 44 44 S 0.0 0.1 0:09.02 init
2 root 9 0 0 0 0 S 0.0 0.0 0:01.30 keventd
Eigentümer
Kommando
PID
Speicherverwendung
- virtuell (ges.)
- im RAM
- geteilt (shared)
- in %
CPU-Last in %
Priorität
Nice-Wert
Zustand
S -> sleep
R -> Run
verbrauchte CPU-Zeit

Benötigt man nur die erste Zeile von top (uptime, anz. User, Load) genügt der Befehl:

uptime

```
root@ki-laptop:~# uptime
 11:58:43 up  1:21,  3 users,  load average: 0.12, 0.11, 0.14
```

Benötigt man nur die Speicherauslastung, genügt der Befehl:

free

```
root@ki-laptop:~# free -m -o
             total       used       free     shared    buffers     cached
Mem:          2017       1582        435          0        113        843
Swap:         3851          0       3851
```

Berechnung Load:

$$load\,(t) = load\,(t-1)\,e^{-5/60R} + n(t)\,(1 - e^{-5/60R})$$

n(t) Anzahl der aktiven Prozesse

R Zeit über die gemittelt wird (1 min, 5 min, 15 min) in min

Signale

Einem Prozess können Signale übermittelt werden. Damit kann man den Prozess anweisen spezielle Tätigkeiten zu erledigen. Es gibt ca. 30 verschieden Signale, einige haben eine feste Bedeutung, andere können vom Programmierer frei verwendet werden.

Bez.	**Nr**	**Beschreibung**
HUP	1	Hang Up, oft für das neue Einlesen der Konfiguration verwendet
INT	2	Interrupt, Beendet Prozess => CTRL C
KILL	9	Beendet den Prozess unfreundlich, d.h. der Prozess darf auch keine Aufräumarbeiten ausführen und wird sofort beendet
TERM	15	Terminate, fragt den Prozess (freundlich), ob er sich beendet
TSTP	18	Anhalten des Prozesses => CTRL Z

Syntax:
kill [-s signal | -signal] pid
kill -l [signal]

Sendet ein Signal an einen Prozess. Als Default wir das Signal *TERM* (15) verwendet. Der Schalter -l zeigt alle dem Befehl bekannten Signale an.

Beispiele

kill 1234	Sendet das Signal *TERM* (15) an den Prozess 1234
kill -9 4321	Sendet das Signal *KILL* (9) an den Prozess 4321
kill -kill 4321 1253	Sendet das Signal *KILL* (9) an die Prozesse 4321 und 1253
kill -HUP `cat /var/run/inetd.pid`	Sendet das Signal *HUP* (1) an den Prozess inetd

Achtung: Klemmt ein Prozess während einer I/O-Aktion, läßt er sich häufig auch nicht mit kill -9 beenden (State D)!

Syntax:
killall [options] name

Sendet ein Signal an alle Prozesse mit dem Namen *name*. Als Default wird das Signal TERM (15) verwendet.

Häufige Optionen (Schalter)

-i interactiv, vor dem Senden des Signals wird nachgefragt
-v verbose, liefert eine detailliertere Rückmeldung
-w wait, wartet bis alle Prozesse beendet sind. Läßt sich ein Prozess nicht beenden, wartet der Befehl für immer
-*signal* Signal *signal* wird an den Prozess geschickt (*signal* => Nummer oder Bezeichnung)

Prozess-Prioritäten

Wie kann man erreichen, dass manche Prozesse vor anderen Prozesse bei der Bearbeitungszeit bevorzugt werden?

- In Linux bekommt jeder Prozess einen „Nice“-Wert
- Je netter („nicer“) ein Prozess ist, desto weniger Rechenzeit bekommt er
- Minimaler Nice-Wert -20 (sehr unfreundlich, viel Rechenzeit)
- Maximaler Nice-Wert +19 (sehr freundlich, wenig Rechenzeit)

- Die Priorität in Linux, wird für jeden Prozess nach einer Zeitscheibe (slice) verringert
- Nach einer Epoche wird die Priorität wieder entsprechend dem Nice-Wert gesetzt
- => die Priorität ändert sich in Linux dynamisch

Syntax:
nice [-n wert] [cmd]
nice [-wert] [cmd]

Startet ein Kommando *cmd* mit dem Nice-Wert *wert*. Wird *wert* weggelassen wird der Nice-Wert *10* gewählt. Wird ein Kommando ohne nice gestartet, ist sein Nice-Wert *0*.

normaler User
Nice-Wert 0...+19

root
Nice-Wert -20...+19

Beispiele

nice -19 firefox	Nice-Wert: +19 (bei Last ist firefox fast nicht mehr zu bedienen!)
nice --10 postfix	Nice-Wert: -10 (root!)
nice -n -20 top	Nice-Wert: -20 (root!)

Syntax:
renice [+|-]wert [option] target

Ändert den Nice-Wert eines laufenden Prozesses.

Optionen
-u *target* ist ein (oder mehrere) User. Alle Prozesse dieses Users werden verändert
-p *target* ist ein (oder mehrere) pid (default)

normaler User
Nice-Wert 0...+19
Nice-Wert kann nur vergrößert werden!

root
Nice-Wert -20...+19

Beispiele
renice -10 1232 Prozess 1232 erhält den Nice-Wert -10
renice 5 -u maier Alle Prozesse des Users *maier* erhalten den Nice-Wert +5

Aufgaben

1. Geben Sie in der Konsole *exec top* ein. Was passiert, wenn Sie *q* drücken? Warum?
2. Wie viel Prozesse laufen auf Ihrem System?
3. Läuft auf Ihrem System *apache*, *samba* oder *postfix*? (Hinweis: verwenden Sie Pipes und *grep*)
4. Welche Prozesse laufen ohne Konsole?
5. Welche Prozesse laufen auf Konsole *tty1*?
6. Welche Prozesse laufen unter dem Besitzer *gast*?
7. Wie lange ist Ihr System up?
8. Wie viel RAM-Speicher hat Ihr System und wie viel Swap-Speicher?
9. Wie viel Prozent der Zeit läuft Ihre CPU im Augenblick leer?
10. Starten Sie *firefox* und verändern Sie danach den Nice-Wert auf 19. Starten danach load-start (hohe Last).
11. Beenden Sie *firefox* über das Signal 15.
12. Geben Sie alle Prozessen des Users *gast* den Nice-Wert -5.
13. Wie heißt das Signal mit der Nummer 14?
14. Starten Sie das Skript *load-start*. Beobachten Sie die Verteilung der 4 Prozesse auf die CPU-Zeit, wenn der Nice-Wert geändert wird.
15. Beenden Sie alle 4 Prozesse mit dem Befehl *killall -i*.

Lösungen

1. Die Konsole wird beendet, da die *bash* durch *top* ersetzt wurde.
2. ps aux|wc -l (Anmerkung: der Wert -1 wg. erster Zeile)
3. ps aux|grep smb (die Anderen entsprechend)
4. ps aux|grep \?
5. ps aux|grep "tty1"
6. ps aux|grep gast
7. Anzeige mit top oder uptime
8. Anzeige mit top
9. Anzeige mit top
10.
11. kill mit entsprechender PID (PID ermitteln mit top oder ps)
12. renice -5 -u gast
13. siehe man kill (-> ALMR)
14. Hinweis: renice mit top und Taste r
15. killall -i load-test