

Betriebssysteme Praxis

SS 2011

Hans-Georg Eßer
Dipl.-Math., Dipl.-Inform.

Foliensatz D (07.05.2011)
Topic 103: GNU and Unix commands (2)
103.5 / 103.6



Topic 103: GNU and Unix Commands

103.5 Create, monitor, and kill processes

Description: Candidates should be able to perform basic process management.

Key Knowledge Areas:

- Run jobs in the foreground and background.
- Signal a program to continue running after logout.
- Monitor active processes.
- Select and sort processes for display.
- Send signals to processes.

The following is a partial list of the used files, terms and utilities:
&, bg, fg, jobs, kill, nohup, ps, top, free, uptime, killall

Quelle: http://www.lpi.org/eng/certification/the_lpic_program/lpic_1/exam_101_detailed_objectives

Topic 103: GNU and Unix Commands

103.6 Modify process execution properties

Description: Candidates should be able to manage process execution priorities.

Key Knowledge Areas:

- Know the default priority of a job that is created.
- Run a program with higher or lower priority than the default.
- Change the priority of a running process.

The following is a partial list of the used files, terms and utilities:
nice, ps, renice, top

Quelle: http://www.lpi.org/eng/certification/the_lpic_program/lpic_1/exam_101_detailed_objectives

103.5: Vorder-/Hintergrund (1)

- In der Shell gestartete Anwendungen laufen standardmäßig im **Vordergrund** – d.h.,
 - die Shell ist blockiert, solange das Programm läuft,
 - und es nutzt das aktuelle Terminal (-Fenster) für Ein- und Ausgabe
- Alternativ kann ein Programm im **Hintergrund** laufen:
 - die Shell kann dann sofort weiter genutzt werden (weitere Kommando eingeben),
 - keine Eingabe möglich, aber Ausgabe (auch ins aktuelle Terminal; besser umleiten)

103.5: Vorder-/Hintergrund (2)

- Typische Vordergrund-Programme
 - Kommandos, die eine Anfrage sofort beantworten
 - Text-Editoren
 - Compiler
- Typische Hintergrund-Programme
 - manuell gestartete Server (Dienste)
 - unter X Window: grafische Anwendungen (die kein Terminal brauchen, sondern ein eigenes Fenster öffnen)

103.5: Vorder-/Hintergrund (3)

- Programm im Vordergrund starten:
einfach den Namen eingeben
Bsp.: `ls -l`
- Programm im Hintergrund starten:
kaufmännisches Und (&, ampersand) anhängen
Bsp.: `/usr/sbin/apache2 &`
- Wechsel von Vordergrund in Hintergrund:
 - Programm mit [Strg-Z] unterbrechen
 - Programm mit `bg` in den Hintergrund schicken
- Wechsel von Hinter- in Vordergrund: `fg`

103.5: Job-Verwaltung (1)

- Programme, die aus einer laufenden Shell heraus gestartet wurden, heißen **Jobs** dieser Shell
- Anzeige mit: `jobs`

```
[esser@macbookpro:~]$ jobs
[esser@macbookpro:~]$ nedit &
[1] 77787
[esser@macbookpro:~]$ vi /tmp/test.txt
^Z
[2]+  Stopped                  vi /tmp/test.txt
[esser@macbookpro:~]$ find / > /tmp/ergebnisse.txt &
[3] 77792
[esser@macbookpro:~]$ jobs
[1]  Running                   nedit &
[2]+  Stopped                  vi /tmp/test.txt
[3]-  Running                   find / > /tmp/ergebnisse.txt &
[esser@macbookpro:~]$
```

103.5: Job-Verwaltung (2)

Zustand des Jobs:

- running: aktiv / bereit
- stopped: mit ^Z oder kill -STOP angehalten
- terminated: beendet, wird nur 1x angezeigt

```
[esser@macbookpro:~]$ jobs
[1]   Running                  nedit &
[2]+  Stopped                  vi /tmp/test.txt
[3]-  Running                  find / > ...&
```

Kommandos

1,2,3, ...: Job-
Nummer

- + : „current job“ = letzter Job, der
 - im Vordergrund gestartet und dann unterbrochen
 - oder im Hintergrund gestartet wurde
 viele Kommandos (fg, bg, ...) ohne Parameter
beziehen sich auf den current job
- : „previous job“ = vorletzter Job mit obiger Eigenschaft

103.5: Job-Verwaltung (3)

- Jobs gezielt ansprechen: %n (mit n = Job-Nummer)

```
[esser@macbookpro:~]$ jobs
[1]    Running                  nedit /tmp/1 &
[2]    Running                  nedit /tmp/2 &
[3]    Running                  nedit /tmp/3 &
[4]-  Running                  nedit /tmp/4 &
[5]+  Running                  nedit /tmp/5 &
[esser@macbookpro:~]$ kill %3
[esser@macbookpro:~]$ jobs
[1]    Running                  nedit /tmp/1 &
[2]    Running                  nedit /tmp/2 &
[3]    Terminated             nedit /tmp/3
[4]-  Running                  nedit /tmp/4 &
[5]+  Running                  nedit /tmp/5 &
[esser@macbookpro:~]$ jobs
[1]    Running                  nedit /tmp/1 &
[2]    Running                  nedit /tmp/2 &
[4]-  Running                  nedit /tmp/4 &
[5]+  Running                  nedit /tmp/5 &
```

103.5: Job-Verwaltung (4)

Kommandos zur Job-Verwaltung

- `bg %n`: in den Hintergrund bringen
- `fg %n`: in den Vordergrund bringen
- `kill %n`: beenden
- `kill -SIGNALNAME %n`: Signal schicken,
siehe nächste Folie
- `disown %n`: Verbindung mit der Shell lösen;
`disown -a`: für alle Jobs
- `wait %n`: Warten, bis Job beendet ist

103.5: Job-Verwaltung (5)

Signale (mit Signalnummer)

- `TERM`, 15: terminieren, beenden (mit „Aufräumen“);
Standardsignal
- `KILL`, 9: sofort abbrechen (ohne Aufräumen)
- `STOP`, 19: unterbrechen (entspricht `^Z`)
- `CONT`, 18: continue, fortsetzen; hebt `STOP` auf
- `HUP`, 1: hang-up, bei vielen Server-Programmen:
Konfiguration neu einlesen (traditionell:
Verbindung zum Terminal unterbrochen)
- Liste aller Signale: `kill -l`

103.5: Jobs vs. Prozesse

- Die Bezeichnung **Job** bezieht sich immer auf die aktuelle Shell-Sitzung
- Jobs, die Sie in verschiedenen Shells starten, haben nichts miteinander zu tun
- Allgemeinerer Begriff: **Prozess**
- Tool für die Prozessanzeige: `ps`
- Die (Gesamt-) Prozessliste (`ps auxw`) enthält alle Prozesse auf dem Linux-System

103.5: Prozesse (1)

- `ps` (ohne Optionen) zeigt alle Prozesse an, die zur aktuellen Shell-Sitzung gehören – das sind dieselben wie in der Ausgabe von `jobs`:

```
[esser@quadamd:~]$ jobs  
[1]+  Angehalten    vi /tmp/test4
```

```
[esser@quadamd:~]$ ps  
  PID TTY          TIME CMD  
27967 pts/0        00:00:00 bash  
28160 pts/0        00:00:00 vi  
28168 pts/0        00:00:00 ps
```

- über Optionen (ohne „–“) lässt sich die Ausgabe von `ps` anpassen, z. B. `ps auxw`:
 - `a`: alle Prozesse (die ein Terminal haben)
 - `u`: „user oriented format“
 - `x`: auch Prozesse ohne Terminal
 - `w`: „wide“: Befehlszeilen nicht abschneiden

103.5: Prozesse (2)

```
[esser@quadamd:~]$ ps auw
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1122	0.0	0.0	1872	580	tty4	Ss+	Apr17	0:00	/sbin/getty -8 38400 tty4
root	1127	0.0	0.0	1872	580	tty5	Ss+	Apr17	0:00	/sbin/getty -8 38400 tty5
root	1150	0.0	0.0	1872	576	tty2	Ss+	Apr17	0:00	/sbin/getty -8 38400 tty2
root	1151	0.0	0.0	1872	576	tty3	Ss+	Apr17	0:00	/sbin/getty -8 38400 tty3
root	1153	0.0	0.0	1872	580	tty6	Ss+	Apr17	0:00	/sbin/getty -8 38400 tty6
root	1776	0.0	0.0	5248	2156	tty1	Ss	Apr17	0:00	/bin/login --
esser	1941	0.0	0.1	9272	4816	tty1	S	Apr17	0:01	-bash
esser	2956	0.0	0.0	1912	540	tty1	S+	Apr17	0:00	/bin/sh /usr/bin/startx
esser	2973	0.0	0.0	3128	708	tty1	S+	Apr17	0:00	xinit /home/esser/.xinitrc -- /etc/X11/x
root	2974	0.0	1.8	85616	75712	tty8	Ss+	Apr17	14:56	/usr/bin/X -nolisten tcp :0 -auth /tmp/s
esser	2977	0.0	0.0	1912	564	tty1	S	Apr17	0:00	/bin/sh /home/esser/.xinitrc
esser	3037	0.0	0.0	3456	560	tty1	S	Apr17	0:00	dbus-launch --sh-syntax --exit-with-sess
esser	3045	0.0	0.0	1700	64	tty1	S	Apr17	0:00	/usr/lib/kde4/libexec/start_kdeinit +kcm
esser	3174	0.0	0.0	1832	244	tty1	S	Apr17	0:00	kwrapper4 kmsserver
esser	3454	0.0	0.0	6552	2040	pts/2	Ss	Apr17	0:00	/bin/bash
root	3775	0.0	0.0	8560	2112	pts/2	S	Apr17	0:00	sudo su
root	3776	0.0	0.0	8316	1764	pts/2	S	Apr17	0:00	su
root	3784	0.0	0.0	6656	2172	pts/2	S+	Apr17	0:00	bash
esser	10674	1.4	7.0	443588	289940	pts/6	Sl	Apr30	131:17	/usr/lib/opera/opera
esser	10694	0.0	0.1	21764	7552	pts/6	S	Apr30	0:08	/usr/lib/opera//operapluginwrapper 101 1
esser	10695	0.0	0.0	3040	548	pts/6	S	Apr30	0:02	/usr/lib/opera//operaplugincleaner 10674
esser	10699	1.3	0.0	0	0	pts/6	Z	Apr30	119:12	[gtk-gnash] <defunct>
esser	12198	0.0	0.0	6552	1828	pts/4	Ss	Apr17	0:00	/bin/bash
root	12583	0.0	0.0	8560	2116	pts/4	S	Apr17	0:00	sudo su
root	13077	0.0	0.0	8316	1768	pts/4	S	Apr17	0:00	su
root	13097	0.0	0.0	6612	2052	pts/4	S+	Apr17	0:00	bash
esser	13653	0.0	0.0	6768	2140	pts/3	Ss	Apr17	0:00	/bin/bash
esser	18905	0.0	0.1	9128	4712	pts/6	Ss+	Apr22	0:02	/bin/bash
esser	27587	0.0	0.0	5256	2144	pts/3	S+	19:40	0:00	ssh backup
esser	28613	0.0	0.1	11744	7264	pts/5	Ss+	22:45	0:00	-bash
esser	29091	0.0	0.1	12156	7704	pts/0	Ss	22:58	0:00	-bash
esser	29307	0.0	0.0	5856	1172	pts/0	R+	23:42	0:00	ps auw

- Spalten in der Ausgabe von `ps auw`:
 - USER: Benutzer, dem der Prozess gehört
 - PID: Prozess-ID
 - %CPU: CPU-Nutzung in Prozent (Verhältnis Rechenzeit / Lebenszeit)
 - %MEM: RSS / RAM-Größe in Prozent
 - VSZ: Größe des virtuellen Speichers (in KByte)
 - RSS: Resident Set Size, aktuell genutzter Speicher (KByte)
 - TTY: Terminal
 - STAT: Prozess-Status
 - START: Startzeit des Prozesses (ggf. Datum)
 - TIME: Lebenszeit
 - COMMAND: Kommando (Aufruf)

103.5: Prozesse (4)

- Signale an beliebige Prozesse schicken
 - wie vorher: Kommando `kill`
 - aber: nicht `kill %n` (n =Job-ID), sondern `kill p` (p = PID)
 - auch hier Angabe eines Signals möglich
- `killall Name`: alle Prozesse beenden, deren ausführbares Programm *Name* heißt
- mit `killall` auch (wie bei `kill`) andere Signale an alle Prozesse mit passendem Namen schicken

103.5: Prozesse (5): pstree

- Darstellung der Prozessliste auch in Baumansicht möglich: `ps tree`
- Jeder Prozess hat einen Vaterprozess
- identische Teilbäume nur 1x
- Option `-p`: Prozess-IDs anzeigen

```
[esser@quadamd:~]$ pstree
init--NetworkManager--dhclient
    `--2*[{NetworkManager}]
        -acpid
        -akonadi_control--2*[{akonadi_contact}]
            |
            |   -3*[{akonadi_ical_re}]
            |   -akonadi_maildir
            |   -akonadi_maildis
            |   -akonadi_nepomuk
            |   -akonadi_vcard_r
            |   -akonadiserver--mysql--23*[{mysqld}]
            |       `--15*[{akonadiserver}]
            `--3*[{akonadi_contro}]
        -atd
        -avahi-daemon---avahi-daemon
        -console-kit-dae---64*[{console-kit-da}]
        -cron
        -cupsd
        [...]
        -knotify4---6*[{knotify4}]
        -konsole--2*[{bash---sudo---su---bash}]
            |
            |   -bash---ssh
            |   -bash---opera--operapluginlea
            |       |
            |       |   -operapluginwrap---gtk-gnash
            |       |   `--6*[{opera}]
            `--2*[{konsole}]
        -krunner---11*[{krunner}]
        -kuiserver
        -kwalletd
        -login---bash---startx---xinit--.xinitrc---kwrapper4
            |
            |   `--Xorg
        -upstart-socket-
        -upstart-udev-br
        -vpnagentd
        -wpa_supplicant
```

103.5: Hang-up, No Hang-up

- Wenn Sie sich in der Konsole abmelden (`exit`) oder unter X Window ein Terminalfenster schließen, erhalten alle in der Shell laufenden Jobs das `HUP`-Signal (Hang-up).
- Die Standardreaktion auf `HUP` ist: beenden
- Abmelden / Fenster schließen beendet also alle darin gestarteten Programme
- Auswege:
 - Programme mit `nohup` starten oder
 - Prozess mit `disown` von der Shell lösen

- `nohup` hat zwei Funktionen:
 - der gestartete Prozess ignoriert `HUP`-Signale
 - Ausgaben des Prozesses (auf die Standardausgabe) erscheinen nicht im Terminal, sondern werden in die Datei `nohup.out` geschrieben

```
[esser@macbookpro:~]$ nedit /tmp/1 &  
[1] 79142  
[esser@macbookpro:~]$ nohup nedit /tmp/2 &  
[2] 79144  
appending output to nohup.out
```

- Prozesse nach CPU-Auslastung sortiert anzeigen: `top`
- Anzeige wird regelmäßig aktualisiert

```
top - 00:07:30 up 19 days, 6:15, 7 users, load average: 0.00, 0.02, 0.05
Tasks: 194 total, 2 running, 191 sleeping, 0 stopped, 1 zombie
Cpu(s): 1.2%us, 0.7%sy, 0.0%ni, 98.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 4120180k total, 2353392k used, 1766788k free, 560756k buffers
Swap: 4191936k total, 0k used, 4191936k free, 566868k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3177	esser	20	0	302m	45m	20m	S	2	1.1	260:37.29	knotify4
10674	esser	20	0	433m	283m	27m	S	2	7.0	131:47.89	opera
3432	esser	20	0	893m	155m	33m	S	1	3.9	253:02.73	seamonkey-bin
1093	messageb	20	0	4524	2664	840	R	0	0.1	16:55.65	dbus-daemon
1576	root	20	0	12072	7272	3408	S	0	0.2	8:50.93	python
2076	root	20	0	5560	972	692	S	0	0.0	5:07.52	udisks-daemon
3238	esser	20	0	367m	64m	35m	S	0	1.6	1:13.69	krunner
29348	esser	20	0	2632	1180	852	R	0	0.0	0:00.03	top
1	root	20	0	3028	1892	1236	S	0	0.0	0:02.06	init
2	root	20	0	0	0	0	S	0	0.0	0:00.73	kthreadd
3	root	20	0	0	0	0	S	0	0.0	1:36.46	ksoftirqd/0
6	root	RT	0	0	0	0	S	0	0.0	0:00.00	migration/0
17	root	0	-20	0	0	0	S	0	0.0	0:00.00	cpuset
18	root	0	-20	0	0	0	S	0	0.0	0:00.00	khelper
19	root	0	-20	0	0	0	S	0	0.0	0:00.00	netns
21	root	20	0	0	0	0	S	0	0.0	0:02.28	sync_supers
22	root	20	0	0	0	0	S	0	0.0	0:00.05	bdi-default
23	root	0	-20	0	0	0	S	0	0.0	0:00.00	kintegrityd
24	root	0	-20	0	0	0	S	0	0.0	0:00.00	kblockd
25	root	0	-20	0	0	0	S	0	0.0	0:00.00	kacpid
26	root	0	-20	0	0	0	S	0	0.0	0:00.00	kacpi_notify
27	root	0	-20	0	0	0	S	0	0.0	0:00.00	kacpi_hotplug

- Sortierung in `top` anpassbar (Sortierspalte ändern mit `<` und `>`)
- Über der Prozessliste: Informationen zur Gesamtauslastung des Systems
- umschaltbar auf Anzeige/CPU bzw. /Kern: 1

```
top - 00:14:22 up 19 days, 6:22, 7 users, load average: 0.05, 0.03, 0.05
Tasks: 194 total, 2 running, 191 sleeping, 0 stopped, 1 zombie
Cpu0  :  0.7%us,  0.3%sy,  0.0%ni, 99.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu1  :  1.7%us,  1.0%sy,  0.0%ni, 97.3%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu2  :  0.0%us,  0.3%sy,  0.0%ni, 99.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu3  :  3.6%us,  0.7%sy,  0.0%ni, 95.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   4120180k total, 2353400k used, 1766780k free, 560948k buffers
Swap:  4191936k total, 0k used, 4191936k free, 566868k cached
```

- Weitere Systeminformationen:
 - `free` (freien Speicher anzeigen)

```
[esser@quadamd:~]$ free
              total        used         free       shared    buffers     cached
Mem:          4120180     2347264     1772916          0      561488     566896
-/+ buffers/cache:    1218880     2901300
Swap:          4191936           0      4191936
```

- `uptime` (wie lange läuft das System schon?)

```
[esser@quadamd:~]$ uptime
00:34:08 up 19 days,  6:42,  6 users,  load average: 0.06, 0.07, 0.05
```

103.6: Prozess-Priorität (1)

- Jeder Linux-Prozess hat eine **Priorität**. Diese bestimmt, welchen Anteil an Rechenzeit der Prozess erhält.
- Priorität ist ein Wert zwischen -20 und 19.
- Konvention: hohe Priorität = kleiner Wert (also: -20 = maximale Prior., 19 = minimale Prior.)
- unter Linux/Unix auch als **nice value** („Nettigkeit“) bezeichnet: 19 = extrem nett, -20 = gar nicht nett
- Bei Programmstart Priorität mit `nice` setzen

103.6: Prozess-Priorität (2)

- `nice` mit Priorität (Option) und auszuführendem Kommando (folgende Argumente) aufrufen, z. B.

```
[esser@quadamd:~]$ nice -5 program &
```

```
[esser@quadamd:~]$ ps -eo user,pid,ni,cmd
```

```
USER          PID  NI  CMD
```

```
...
```

```
root          28299  0  [kworker/2:0]
```

```
root          28300  0  [kworker/0:1]
```

```
esser         28301  5  program
```

```
esser         28303  0  ps -eo user,pid,ni,cmd
```

- negative Nice-Werte kann nur Administrator *root* setzen:

```
[esser@quadamd:~]$ nice --10 vi
```

```
nice: kann Priorität nicht setzen: Keine Berechtigung
```


103.6: Prozess-Priorität (3)

- Alternative Syntax für bessere Lesbarkeit:
`nice -n Wert` (statt `nice -Wert`)
- vor allem für negative Werte intuitiver:
`nice -n -10` (statt `nice --10`)

```
[esser@quadamd:~]$ su
Passwort:
root@quadamd:~# nice -n -10 program &
[1] 28373
root@quadamd:~# ps -eo user,pid,ni,cmd
USER          PID    NI  CMD
[...]
root         28311     0  su
root         28319     0  bash
root         28373   -10  program
root         28375     0  ps -eo user,pid,ni,cmd
```

103.6: Prozess-Priorität (4)

- Genauer: Nice-Wert in `nice`-Aufruf ist relativ zum „aktuellen Nice-Level“ (Standard: 0)
- angegebener Wert wird zum Nice-Wert addiert:

```
[esser@quadamd:~]$ nice
0
[esser@quadamd:~]$ nice -n 5 bash
[esser@quadamd:~]$ nice
5
[esser@quadamd:~]$ nice -n 10 bash
[esser@quadamd:~]$ nice
15
[esser@quadamd:~]$ _
```

103.6: Prozess-Priorität (5)

- Nice-Wert für laufendes Programm ändern: `renice`
- Wert <0 setzen darf nur *root*
- in alten Linux-Versionen galt auch: aktuellen Wert verringern darf nur *root*)

```
[esser@quadamd:~]$ program &
[5] 28937
[esser@quadamd:~]$ ps -eo user,pid,ni,cmd
USER      PID  NI  CMD
esser    28937  0  program
[esser@quadamd:~]$ renice 5 28937
28937: Alte Priorität: 0, neue Priorität: 5
[esser@quadamd:~]$ ps -eo user,pid,ni,cmd
USER      PID  NI  CMD
esser    28937  5  program
[esser@quadamd:~]$ renice 0 28937
28937: Alte Priorität: 5, neue Priorität: 0
[esser@quadamd:~]$ renice -10 28937
renice: 28937: setpriority: Keine Berechtigung
```