



SV3: Linux Server

Prozessverwaltung

Agenda

Prozessverwaltung

- Prozessattribute
- Kommando kill
- Jobverwaltung
- Cronjob
- Prozessprioritäten

Vordergrund- und Hintergrundprozesse

Ein gestartetes Programm wird als Prozess bezeichnet, wobei ein gestartetes Programm auch aus einer Folge von Einzelprozessen (Unterprogrammen) bestehen kann.

Was bedeutet Vordergrund?

Ihre Shell nimmt das Kommando an, überprüft es, ersetzt evtl. Metazeichen und startet es. Solange dieses Programm nicht beendet ist, können Sie am Terminal kein weiteres Kommando sichtbar aufrufen.

Zu Beginn haben Sie gelernt, dass Linux/Unix ein Multi -User und -Tasking-System ist. Dies bedeutet, dass jeder Benutzer gleichzeitig mehrere Programme starten kann.

Wie soll das aber möglich sein, wenn jeweils ein Kommando nach dem anderen ausgeführt wird?

- Es gibt einmal die Pipe, die zur gleichen Zeit mehrere Programme startet – zum anderen können Programme im Hintergrund ablaufen.
- Der Unterschied zwischen Hintergrundprozessen und Vordergrundprozessen besteht darin, dass die Shell nicht auf die Beendigung des gestarteten Programms wartet, sondern sich sofort mit dem Bereitzeichen meldet.

Wie starten Sie ein Programm, das im Hintergrund ablaufen soll?

Als letzte Eingabe der Kommandozeile wird ein **&** Zeichen angefügt.

Kommando &

Das **&** - Zeichen schiebt einen Prozess in den Hintergrund

Hintergrundprozesse werden weder schneller noch langsamer abgearbeitet als Vordergrundprozesse.

Wann ist es sinnvoll, Programme im Hintergrund zu verarbeiten?

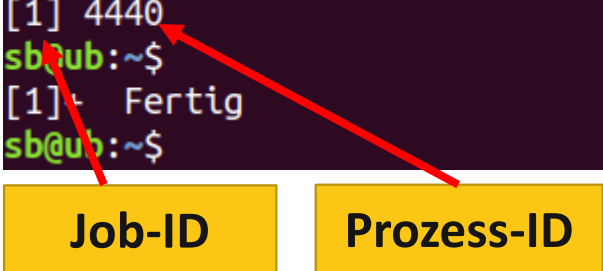
- Starten Sie von einem Terminal ein Programm, das für die grafische Oberfläche bestimmt ist, könnten Sie, solange dieses Programm nicht beendet wird, an Ihrem Terminal nicht mehr weiterarbeiten.

Prozessverwaltung

Angenommen, Sie haben eine große Adressdatei »Adressen«, in der Sie Namen und Adresse chronologisch eingetragen haben.

Da das Sortieren einer großen Datei zeitaufwendig ist, rufen wir das Kommando wie in der Abbildung als Hintergrundprozess auf:

```
sb@ub:~$ sort adressen > adressen_alphabetisch &  
[1] 4440  
sb@ub:~$  
[1]+  Fertig                  sort adressen > adressen_alphabetisch  
sb@ub:~$
```



Bevor sich die Shell mit »wieder bereit« meldet, teilt sie die Nummer des Hintergrundprogramms mit.

Wie brechen Sie einen Hintergrundprozess ab?

Um einen Hintergrundprozess abzubrechen, wird ein eigenes Kommando aufgerufen, das kill-Kommando. Für dieses Kommando benötigen Sie die Ihnen vom System beim Start des Programms mitgeteilte Prozessnummer

kill [-9] Prozessnummer(n)

Signal um den Prozess abzubrechen
töten/abschießen

Prozessverwaltung

Mit der Tastenkombination <Strg+c> können Sie einen Prozess, den Sie im Vordergrund gestartet hatten, abbrechen. Linux schickt hierbei an den Prozess ein Signal.

Manche Programme ignorieren ein »einfaches« kill-Signal. Schickt man ihnen jedoch kill -9

```
sb@ub:~$ cat /dev/random > /dev/null &  
[1] 4482  
sb@ub:~$  
sb@ub:~$ kill 4482  
sb@ub:~$  
[1]+  Beendet                cat /dev/random > /dev/null  
sb@ub:~$
```

So können sie den »Abschuss« nicht abwenden und werden sicher gekillt. Benutzer können nur ihre eigenen Prozesse »killen«. Nur der Superuser darf auch fremde Prozesse abbrechen.

Hinweis:

- Sollte die grafische Arbeitsfläche blockiert sein, könnten Sie unter Linux meist noch auf die virtuellen Terminals mit **<Strg+Alt+F1>** bis **<Strg+Alt+F6>** gehen, sich dort anmelden und weiterarbeiten bzw. den Fehler versuchen zu beheben oder den vermeintlichen Prozess killen.
- Dazu aber später mehr.

Welche Programme werden zurzeit vom Rechner bearbeitet?

Zwar wird nach Beenden des Hintergrundprozesses eine »**Fertigmeldung**« auf den Bildschirm ausgegeben, von dem der Hintergrundprozess gestartet wurde, doch zwischendrin wissen Sie nie genau, ob der gestartete Hintergrundprozess noch aktiv ist.

- Das Kommando **ps** (process status) gibt Ihnen über Ihre eigenen und durch die Option -e über alle zurzeit laufenden Prozesse Auskunft.

ps – Kommando, um den Status der Programme abzufragen

ps [-eF]

F = Full Format

e = Alle Prozesse (nicht nur die eigenen)

process status

```
sb@ub:~$ ps -eF
UID      PID  PPID  C   SZ   RSS  PSR  STIME TTY      TIME CMD
root      1    0    0 40051 8972   0 08:25 ?        00:00:01 /sbin/init splash
root      2    0    0    0    0    1 08:25 ?        00:00:00 [kthreadd]
root      3    2    0    0    0    0 08:25 ?        00:00:00 [rcu_gp]
root      4    2    0    0    0    0 08:25 ?        00:00:00 [rcu_par_gp]
root      6    2    0    0    0    0 08:25 ?        00:00:00 [kworker/0:0H-kb]
root      9    2    0    0    0    0 08:25 ?        00:00:00 [mm_percpu_wq]
root     10    2    0    0    0    0 08:25 ?        00:00:00 [ksoftirqd/0]
root     11    2    0    0    0    0 08:25 ?        00:00:00 [rcu_sched]
root     12    2    0    0    0    0 08:25 ?        00:00:00 [migration/0]
root     13    2    0    0    0    0 08:25 ?        00:00:00 [idle_inject/0]
root     14    2    0    0    0    0 08:25 ?        00:00:00 [cpuhp/0]
root     15    2    0    0    0    1 08:25 ?        00:00:00 [cpuhp/1]
root     16    2    0    0    0    1 08:25 ?        00:00:00 [idle_inject/1]
root     17    2    0    0    0    1 08:25 ?        00:00:00 [migration/1]
root     18    2    0    0    0    1 08:25 ?        00:00:00 [ksoftirqd/1]
root     20    2    0    0    0    1 08:25 ?        00:00:00 [kworker/1:0H-kb]
```

Prozessverwaltung

Wollen wir einen eben gestarteten Hintergrundprozess sofort wieder abbrechen, haben aber die Nummer des Programms nicht beachtet, erhalten wir durch das Kommando `ps` die wesentlichen Programminformationen.

```
sb@ub:~$ ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.4 160204  8972 ?        Ss   08:25   0:01 /sbin/init splash
root         2  0.0  0.0      0     0 ?        S    08:25   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        I<   08:25   0:00 [rcu_gp]
root         4  0.0  0.0      0     0 ?        I<   08:25   0:00 [rcu_par_gp]
root         6  0.0  0.0      0     0 ?        I<   08:25   0:00 [kworker/0:0H-kb]
root         9  0.0  0.0      0     0 ?        I<   08:25   0:00 [mm_percpu_wq]
root        10  0.0  0.0      0     0 ?        S    08:25   0:00 [ksoftirqd/0]
root        11  0.0  0.0      0     0 ?        I    08:25   0:00 [rcu_sched]
root        12  0.0  0.0      0     0 ?        S    08:25   0:00 [migration/0]
root        13  0.0  0.0      0     0 ?        S    08:25   0:00 [idle_inject/0]
root        14  0.0  0.0      0     0 ?        S    08:25   0:00 [cpuhp/0]
root        15  0.0  0.0      0     0 ?        S    08:25   0:00 [cpuhp/1]
root        16  0.0  0.0      0     0 ?        S    08:25   0:00 [idle_inject/1]
root        17  0.0  0.0      0     0 ?        S    08:25   0:00 [migration/1]
root        18  0.0  0.0      0     0 ?        S    08:25   0:00 [ksoftirqd/1]
root        20  0.0  0.0      0     0 ?        I<   08:25   0:00 [kworker/1:0H-kb]
root        21  0.0  0.0      0     0 ?        S    08:25   0:00 [kdevtmpfs]
root        22  0.0  0.0      0     0 ?        I<   08:25   0:00 [netns]
root        23  0.0  0.0      0     0 ?        S    08:25   0:00 [rcu_tasks_kthre]
root        24  0.0  0.0      0     0 ?        S    08:25   0:00 [kauditd]
```

Anwendungsbeispiele:

- Um z.B. Kollegen weiterzuhelfen, denen durch einen falschen Aufruf eines Kommandos das Terminal blockiert ist, kann der Systemverwalter anhand der durch `ps -ef` erhaltenen Informationen feststellen, welches Programm an welchem Terminal gestartet wurde, und den fehlerhaften Prozess killen.
- Programme hängen sich auf und reagieren nicht mehr.
- Programme beenden sich nicht obwohl Sie zum Beispiel nicht mehr im Vordergrund arbeiten.

Prozessverwaltung

Arbeiten Sie unter einer grafischen Oberfläche, d.h., Sie arbeiten immer mit mehreren »Terminals«, dann empfiehlt sich folgendes Kommando, um alle Ihre laufenden Prozesse zu sehen:

```
sb@ub:~$ ps -eF | grep tux
root      4557   4546   0 18064   3836    1 10:25 pts/2    00:00:00 su tux
tux       4558     1   0 19198   7808    1 10:25 ?          00:00:00 /lib/systemd/systemd --user
tux       4559   4558   0 49070   2680    1 10:25 ?          00:00:00 (sd-pam)
tux       4570   4557   0  7439   4924    1 10:25 pts/2    00:00:00 bash
tux       4599   4570   0 12862   4116    0 10:25 pts/2    00:00:00 top
sb        4602   2822   0  5383   1108    1 10:25 pts/1    00:00:00 grep --color=auto tux
sb@ub:~$
```

Einige Linux-Systeme haben speziell bei dem Kommando **ps** andere Optionen, Auskunft darüber gibt das Kommando **man ps**.

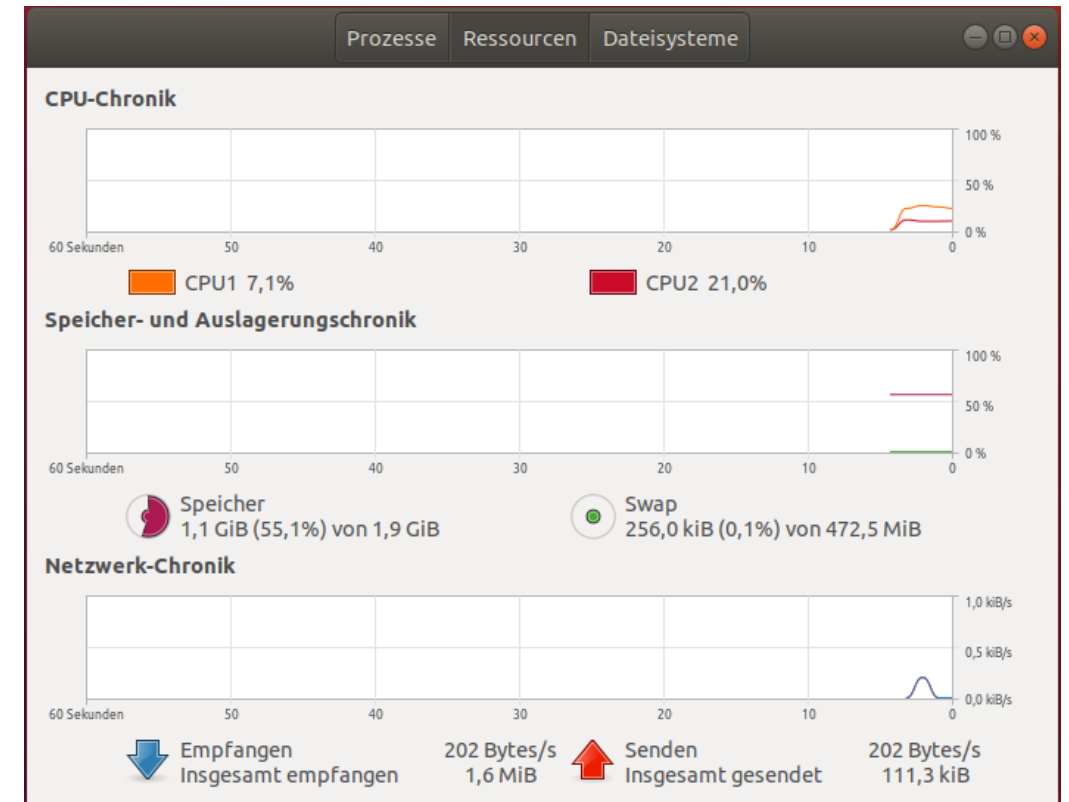
Prozessverwaltung

- Einen Eindruck, wie die Hierarchie der Prozesse aufgebaut ist, erhalten Sie mit dem Kommando **ps tree**.
- In der grafischen Oberfläche KDE gibt es ein übersichtliches Programm, das Sie mit ksysguard & von Ihrem Terminal aufrufen können oder über **Start → System → Monitor**

```
sb@ub:~$ ps tree
systemd--ModemManager--2*[{ModemManager}]
      |NetworkManager--dhclient
      |                  |2*[{NetworkManager}]
accounts-daemon--2*[{accounts-daemon}]
acpid
agetty
avahi-daemon--avahi-daemon
boltd--2*[{boltd}]
colord--2*[{colord}]
cron
cups-browsed--2*[{cups-browsed}]
cupsd
dbus-daemon
fail2ban-server--2*[{fail2ban-server}]
fwupd--4*[{fwupd}]
gdm3--gdm-session-work--gdm-x-session
```

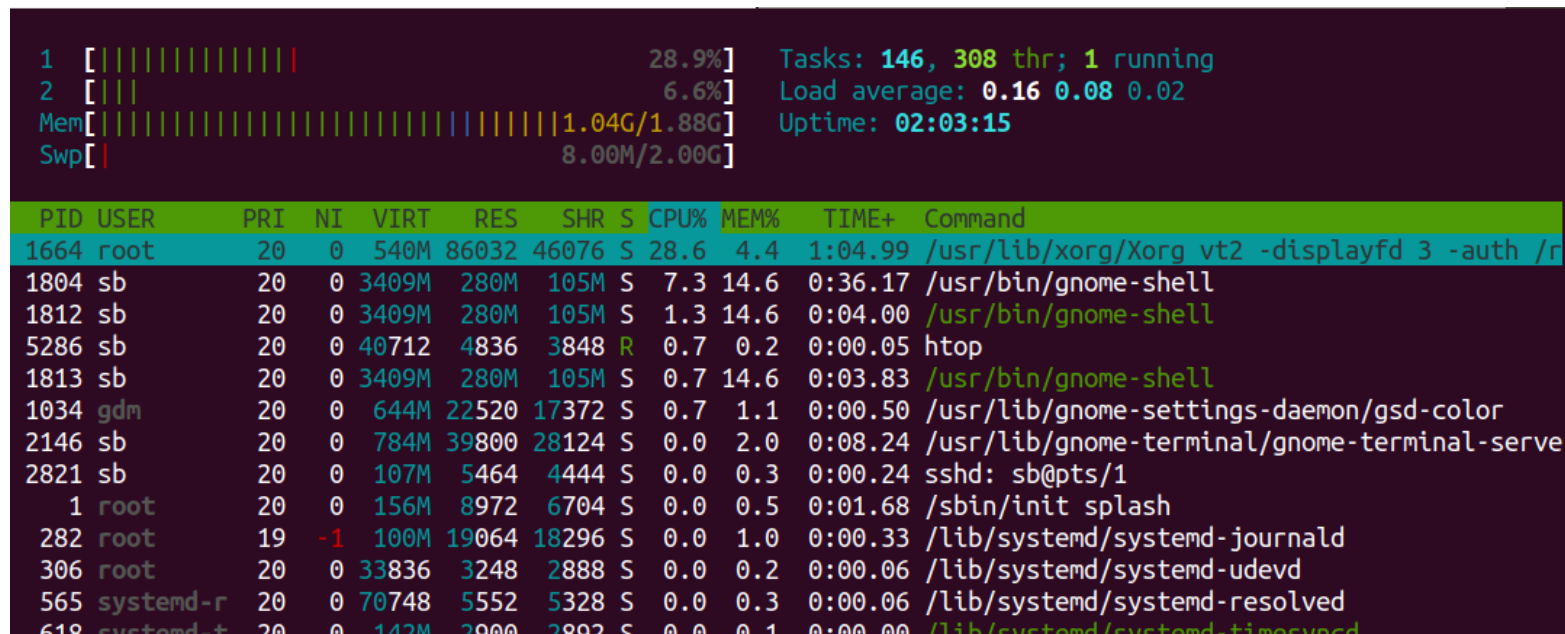

Prozessverwaltung

- Grafische Systemüberwachung bei Ubuntu 18.04 LTS
- Hier können Prozesse, Ressourcen und die Auslastung des Dateisystems überwacht werden.



Prozessverwaltung

Eine weitere Möglichkeit, ist das Programm htop, dies können Sie unter Ubuntu mit dem Befehl: `sudo apt install htop` installieren.



The screenshot displays the htop interface. At the top, system statistics are shown: CPU usage at 28.9%, memory usage at 1.04G/1.88G, and swap usage at 8.00M/2.00G. The load average is 0.16, 0.08, 0.02, and the system uptime is 02:03:15. Below these statistics, a table lists the running processes with columns for PID, USER, PRI, NI, VIRT, RES, SHR, S, CPU%, MEM%, TIME+, and Command.

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
1664	root	20	0	540M	86032	46076	S	28.6	4.4	1:04.99	/usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /r
1804	sb	20	0	3409M	280M	105M	S	7.3	14.6	0:36.17	/usr/bin/gnome-shell
1812	sb	20	0	3409M	280M	105M	S	1.3	14.6	0:04.00	/usr/bin/gnome-shell
5286	sb	20	0	40712	4836	3848	R	0.7	0.2	0:00.05	htop
1813	sb	20	0	3409M	280M	105M	S	0.7	14.6	0:03.83	/usr/bin/gnome-shell
1034	gdm	20	0	644M	22520	17372	S	0.7	1.1	0:00.50	/usr/lib/gnome-settings-daemon/gsd-color
2146	sb	20	0	784M	39800	28124	S	0.0	2.0	0:08.24	/usr/lib/gnome-terminal/gnome-terminal-serve
2821	sb	20	0	107M	5464	4444	S	0.0	0.3	0:00.24	sshd: sb@pts/1
1	root	20	0	156M	8972	6704	S	0.0	0.5	0:01.68	/sbin/init splash
282	root	19	-1	100M	19064	18296	S	0.0	1.0	0:00.33	/lib/systemd/systemd-journald
306	root	20	0	33836	3248	2888	S	0.0	0.2	0:00.06	/lib/systemd/systemd-udev
565	systemd-r	20	0	70748	5552	5328	S	0.0	0.3	0:00.06	/lib/systemd/systemd-resolved
618	systemd-t	20	0	142M	2900	2892	S	0.0	0.1	0:00.00	/lib/systemd/systemd-timesyncd

Zusammenfassung der Eigenschaften von Hintergrundprozessen:

- Jedes Programm kann in den Hintergrund geschickt werden.
- Das Kommando wird von der Shell ähnlich wie ein Vordergrundprozess aufbereitet
- Die Shell meldet beim Start von Hintergrundprozessen deren Nummer (PID – Process IDentification Number) und ist sofort bereit für weitere Anweisungen.
- Hintergrundprozesse können zuverlässig durch das Kommando `kill -9 PID` abgebrochen werden.
- In der Korn- und C-Shell gibt es speziell eine Jobcontrol, mit der Prozesse im Vorder- und Hintergrund gesteuert werden können.

Zusammenfassung der Befehle:

ps zeigt die aktuellen Prozesse

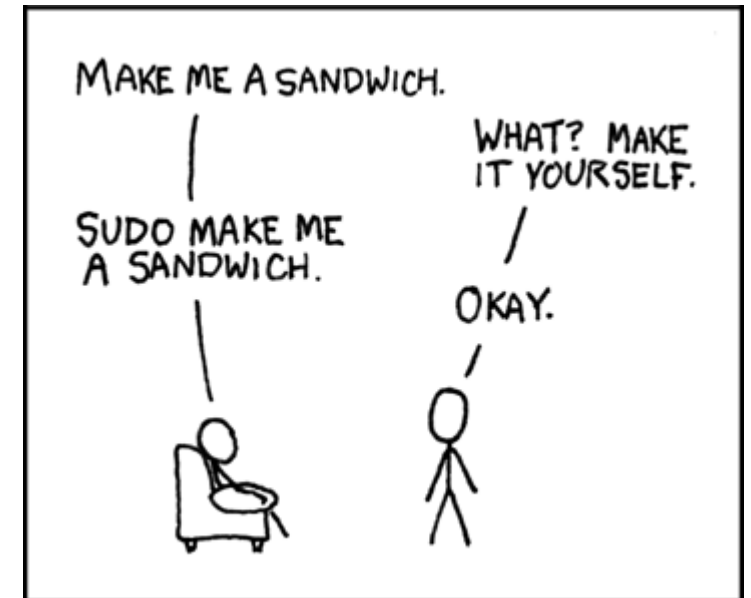
kill bricht vorzeitig einen Prozess ab

& startet ein Kommando als Hintergrundprozess

Prozesse unter einer anderen Identität ausführen (sudo)

sudo verfolgt einen ganz anderen Ansatz als su-Varianten.

- Das Programm ermöglicht nach entsprechender Konfiguration bestimmten Benutzern die Ausführung bestimmter Programme mit root-Rechten.
- Zur Sicherheit muss nochmals das eigene Passwort angegeben werden, also eben nicht das root-Passwort.



Prozesse unter einer anderen Identität ausführen (sudo)

sudo merkt sich das Passwort für 15 Minuten. Wenn Sie innerhalb dieser Zeit ein weiteres Kommando mit sudo ausführen, werden Sie nicht neuerlich nach dem Passwort gefragt.

- Die Merkzeit kann in `/etc/sudoers` mit dem Schlüsselwort `timestamp_timeout` verändert werden.
- Die Konfiguration von **sudo** erfolgt durch die Datei `/etc/sudoers`.

Wichtig:

Änderungen an der `/etc/sudoers` sollte ausschließlich mit dem Programm **visudo** bearbeitet werden!

- **visudo** führt vor der Speicherung der Datei eine Syntaxprüfung durch.
- **visudo** nutzt standardmäßig **vi** als Editor. möchte man dies ändern. muss man einen anderen Editor in die Umgebungsvariable exportieren.

```
export EDITOR=nano
```

sudo bei Ubuntu

Bei Ubuntu und einigen anderen Distributionen wird der Benutzer root standardmäßig ohne gültiges Passwort eingerichtet. Ein root-Login ist damit unmöglich! Auch **su** oder **ssh -l root** funktionieren nicht. Die einzige Möglichkeit zur Ausführung administrativer Kommandos bietet somit **sudo**.

- Die Datei **/etc/sudoers** enthält außer den Kommentaren nur wenige Zeilen:

```
sb@ub:~$ sudo grep "^[^#]" /etc/sudoers
Defaults    env_reset
Defaults    mail_badpass
Defaults    secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin"
root    ALL=(ALL:ALL) ALL
%admin  ALL=(ALL) ALL
%sudo   ALL=(ALL:ALL) ALL
sb@ub:~$
```


Systemprozesse (Dämonen)

Als Dämonen (englisch daemons) werden Hintergrundprozesse zur Systemverwaltung bezeichnet. Diese Prozesse werden normalerweise während des Hochfahrens des Rechners im Rahmen des Init-V-Prozesses gestartet.

Prozess	Bedeutung
Smbd	Datei-Server für Windows/Samba-Freigaben
Squid	Web-Proxy
Sshd	Openssh-Server
Rsyslogd	Systemprotokoll-Dämon
Udevd	Geräteverwaltung
Vsftpd	FTP-Server
Xdm	X-Display Manager
xinetd	Verwaltung von anderen Netzwerkdämonen

Wenn Sie mit der Windows-Diktion vertraut sind, entsprechen Linux-Dämonen einfach Diensten.

Prozesse automatisch starten (`cron`)

Dieses Programm wird beim Rechnerstart durch den Init-Prozess automatisch gestartet. Es wird einmal pro Minute aktiv, analysiert alle `crontab`-Dateien und startet die dort angegebenen Programme.

- `cron` wird in erster Linie für Wartungsarbeiten verwendet.

Prozesse automatisch starten (Cron)

Die globale Konfiguration von Cron erfolgt durch die Datei `/etc/crontab`. Darüber hinaus dürfen Benutzer ihre eigenen Cron-Jobs in den benutzerspezifischen Dateien `/var/spool/cron/[tabs/]username` definieren.

- Die Datei `/etc/crontab` bzw. `/etc/crontab` die Dateien in `/etc/cron.d` enthalten zeilenweise Einträge für die auszuführenden Programme. Die Syntax sieht so aus:

```
* * * * * Befehl der ausgeführt werden soll
- - - - -
| | | | |
| | | | +----- Wochentag (0 - 7) (Sonntag ist 0 und 7; oder Namen, siehe unten)
| | | +----- Monat (1 - 12)
| | +----- Tag (1 - 31)
| +----- Stunde (0 - 23)
+----- Minute (0 - 59; oder Namen, siehe unten)
```

Prozesse automatisch starten (Cron)

Die Crontab-Syntax erfordert einen Zeilenumbruch nach der letzten Zeile. Achten Sie darauf, dass alle Cron-Konfigurationsdateien mit einem Zeilenumbruch enden müssen! Andernfalls wird die letzte Zeile ignoriert!

Kürzel	Code	Bedeutung
@reboot	-	nach jedem Reboot ausführen
@yearly	0 0 1 1 *	einmal im Jahr ausführen
@annually	0 0 1 1 *	wie @yearly
@monthly	0 0 1 * *	einmal pro Monat ausführen
@weekly	0 0 * * 0	einmal pro Woche ausführen
@daily	0 0 * * *	einmal pro Tag ausführen
@hourly	0 * * * *	einmal pro Stunde ausführen

Prozesse automatisch starten (Cron)

Wie der Name schon andeutet, wird für wiederkehrende Aufgaben eine Tabelle erstellt (chronologische Tabelle - crontab).

```
crontab [-elr] [Datei]
```

e = edit (editiert die vom Benutzer aufgerufene crontab-Tabelle, Oder erstellt diese neu)

l = list (zeigt alle angelegten Tabellen des Benutzers an)

r = remove (löscht die angegebene Datei)

Wird keine Datei angegeben, wird unter dem Benutzernamen eine Tabelle unter **/var/spool/cron/tabs** angelegt

Jobverwaltung

```
GNU nano 2.9.3 /tmp/crontab.5JsZQ8/crontab

# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
*/1 * * * * echo $(date) >> /home/sb/datum.txt
```

Verteilung der Rechenzeit(nice, renice, ionice)

Im alltäglichen Betrieb von Linux ist die Rechenkapazität meistmehr als ausreichend, um alle laufenden Prozesse ohne Verzögerungen auszuführen.

- Wenn Linux aber gerade mit rechenaufwendigen Prozessen beschäftigt ist – z. B. während des Kompilierens eines umfangreichen Programms –, versucht es, die zur Verfügung stehende Rechenzeit gerecht an alle Prozesse zu verteilen.

Verteilung der Rechenzeit(nice, renice, ionice)

Dazu wird an **nice** die gewünschte Priorität übergeben, die von 19 (ganz niedrig) bis -20 (ganz hoch) reicht. Per Default werden Prozesse mit der Priorität 0 gestartet.

- Im folgenden Beispiel wird ein Backup-Programm mit niedrigerer Priorität gestartet, damit es keine anderen Prozesse beeinträchtigt. (Es ist ja egal, ob das Backup ein paar Sekunden länger dauert.)

```
$ nice -n 10 ./my-backup-script.sh
```


Verteilung der Rechenzeit(nice, renice, ionice)

Mit **renice** kann auch die Priorität von bereits laufenden Prozessen geändert werden. Als Parameter muss die Prozess-ID angegeben werden, die vorher mit top oder ps ermittelt wurde.

```
top - 11:09:35 up 2:44, 2 users, load average: 0,01, 0,00, 0,00
Tasks: 224 gesamt, 1 laufend, 184 schlafend, 0 gestoppt, 0 Zombie
%CPU(s): 5,2 be, 0,3 sy, 0,0 ni, 94,5 un, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
KiB Spch : 1968412 gesamt, 226904 frei, 1061216 belegt, 680292 Puff/Cache
KiB Swap: 2097148 gesamt, 2088956 frei, 8192 belegt, 725076 verfü Spch
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	ZEIT+	BEFEHL
1664	root	20	0	554072	86032	46076	S	9,3	4,4	1:25.49	Xorg
1804	sb	20	0	3490836	286920	108048	S	1,3	14,6	0:44.65	gnome-shell
2146	sb	20	0	803400	39888	28124	S	0,3	2,0	0:10.63	gnome-terminal-
3141	root	20	0	0	0	0	I	0,3	0,0	0:00.49	kworker/1:4-mm_
3191	root	20	0	304832	21740	10176	S	0,3	1,1	0:02.07	fail2ban-server
4599	tux	20	0	51448	4116	3340	S	0,3	0,2	0:05.10	top
5519	sb	20	0	51432	4104	3352	R	0,3	0,2	0:00.01	top
1	root	20	0	160204	8972	6704	S	0,0	0,5	0:01.72	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kthreadd
3	root	0	20	0	0	0	I	0,0	0,0	0:00.00	rcu_gp
4	root	0	20	0	0	0	I	0,0	0,0	0:00.00	rcu_par_gp
6	root	0	20	0	0	0	I	0,0	0,0	0:00.00	kworker/0:0H-kb
9	root	0	20	0	0	0	I	0,0	0,0	0:00.00	mm_percpu_wq

Verteilung der Rechenzeit(nice, renice, ionice)

Oft ist nicht die CPU, sondern der Datenträger der limitierende Faktor bei der Ausführung von Programmen. Wenn Sie vermeiden möchten, dass beispielsweise ein Backup-Script die gesamte I/O-Kapazität des Rechners für sich beansprucht und damit andere, vielleicht zeitkritischere Prozesse bremst, können Sie es mit **ionice** mit reduzierter I/O-Priorität ausführen.

- Das folgende Kommando liest ein Logical Volume aus, komprimiert seinen Inhalt und speichert ihn in einer Image-Datei:

```
$ ionice -c 3 cat /dev/vg1/snap | lzop -c > /backup/image.lzo
```

**VIELEN DANK
FÜR IHRE
AUFMERKSAMKEIT!**