

# Process Management

Introduction to processes

Process priorities

Background jobs

**DE HOGESCHOOL  
MET HET NETWERK**

Hogeschool PXL – Dep. PXL-IT – Elfde-Liniestraat 26 – B-3500 Hasselt  
[www.pxl.be](http://www.pxl.be) - [www.pxl.be/facebook](https://www.pxl.be/facebook)



# Terminology

- **process**  
Programma of commando dat een bepaalde job uitvoert.
- **PID**  
Ieder proces heeft een process id.  
Dit is een uniek nummer (tussen 0 en 65535).
- **PPID**  
Ieder proces heeft een parent process, met een parent PID.  
Een child process wordt gestart door zijn parent process.

# Terminology

- **systemd** Het systemd proces, heeft als PID 1, wordt gestart door de kernel zelf en heeft geen parent process.  
→ wordt soms nog aangeduid met init
- **kill** Als een proces stopt, sterft het proces.  
Als je een proces wil stoppen, moet je het “killen”.
- **daemon** Een proces dat start bij het opstarten van je systeem en vervolgens continu blijft draaien.

# Terminology

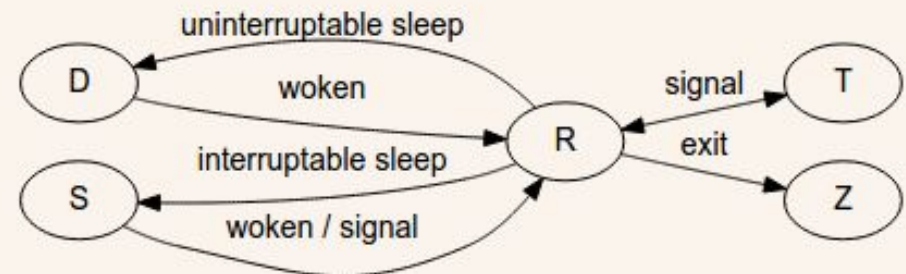
- **zombie**

Een “killed” process dat nog steeds zichtbaar is.

## PROCESS STATE CODES

R running or runnable (on run queue)  
D uninterruptible sleep (usually IO)  
S interruptible sleep (waiting for an event to complete)  
Z defunct/zombie, terminated but not reaped by its parent  
T stopped, either by a job control signal or because it is being traced

A process starts its life in an R "running" state and finishes after its parent reaps it from the Z "zombie" state.



# Basic Process Management

- **\$\$** Shell parameter: bevat het **huidig process ID**
- **\$PPID** Shell variabele: bevat het **parent ID**

```
student@ubuntutop01:~$ echo $$ $PPID  
2529 2514
```

# Basic Process Management

- **pidof** Vind alle process id's op naam.

```
student@ubuntudesktop01:~$ pidof bash
2529
student@ubuntudesktop01:~$ xclock &
[1] 2801
student@ubuntudesktop01:~$ xclock &
[2] 2802
student@ubuntudesktop01:~$ xclock &
[3] 2803
student@ubuntudes
```



# Basic Process Management

- **parent and child**

leder process (buiten systemd) heeft een parent process.

```
student@ubuntudesktop01:~$ echo $$ $PPID
2529 2514
student@ubuntudesktop01:~$ bash
student@ubuntudesktop01:~$ echo $$ $PPID
3001 2529
student@ubuntudesktop01:~$ exit
exit
student@ubuntudesktop01:~$ echo $$ $PPID
2529 2514
```

# Basic Process Management

## **fork vs exec**

Een programma wil een nieuw proces starten dat gelijktijdig uitgevoerd wordt met het proces van het programma zelf.

→ een **fork**: een process maakt een kopie van zichzelf.

Dit is een nieuw proces dat een exacte kopie is van het proces dat de opdracht gaf een fork uit te voeren, het enige verschil is de PID.

Het child process wordt meteen gestart en begint met de eerstvolgende instructie na de fork. Ook het parent process gaat verder met de uitvoering en met de eerstvolgende instructie na de fork. Er lopen nu dus twee vrijwel identieke processen die alleen een andere PID hebben.

**exec** vervangt het programma dat het huidige proces uitvoert. Er wordt dus geen nieuw proces gestart, maar het huidige proces start een ander programma.



# Basic Process Management

- fork and exec

fork →

exec →

```
student@ubuntudesktop01:~$ echo $$  
2529  
student@ubuntudesktop01:~$ sh  
$ echo $$ $PPID  
3009 2529  
$ exec bash  
student@ubuntudesktop01:~$ echo $$ $PPID  
3009 2529  
student@ubuntudesktop01:~$ exit  
exit  
student@ubuntudesktop01:~$ echo $$  
2529
```

# Basic Process Management

- Werken met ps

man ps

```
PS(1)                                User Commands                                PS(1)

NAME
ps - report a snapshot of the current processes.
```

```
student@ubuntudesktop01:~$ ps ax | head -4
  PID TTY          STAT       TIME COMMAND
    1 ?           Ss        0:02 /sbin/init splash
    2 ?           S          0:00 [kthreadd]
    4 ?           I<        0:00 [kworker/0:0H]
```

```
student@ubuntudesktop01:~$ ps --pid 1 -o pid,cmd,comm
  PID CMD                                COMMAND
    1 /sbin/init splash                  systemd
```

# Basic Process Management

- Werken met ps

```
student@ubuntudesktop01:~$ echo $$ $PPID
2529 2514
student@ubuntudesktop01:~$ bash
student@ubuntudesktop01:~$ echo $$ $PPID
3097 2529
student@ubuntudesktop01:~$ bash
student@ubuntudesktop01:~$ echo $$ $PPID
3105 3097
student@ubuntudesktop01:~$ ps f
```

PID	TTY	STAT	TIME	COMMAND
2529	pts/0	Ss	0:00	bash
3097	pts/0	S	0:00	\_ bash
3105	pts/0	S	0:00	\_ bash
3113	pts/0	R+	0:00	\_ ps f

```
student@ubuntudesktop01:~$ exit
exit
student@ubuntudesktop01:~$ ps f
```

PID	TTY	STAT	TIME	COMMAND
2529	pts/0	Ss	0:00	bash
3097	pts/0	S	0:00	\_ bash
3114	pts/0	R+	0:00	\_ ps f

```
student@ubuntudesktop01:~$ exit
exit
student@ubuntudesktop01:~$ ps f
```

PID	TTY	STAT	TIME	COMMAND
2529	pts/0	Ss	0:00	bash
3115	pts/0	R+	0:00	\_ ps f

# Basic Process Management

## pgrep

processen op naam zoeken

```
student@ubuntu01:~$ sleep 1000 &  
[1] 3117  
student@ubuntu01:~$ pgrep sleep  
3117  
student@ubuntu01:~$ ps -C sleep  
  PID TTY          TIME CMD  
 3117 pts/0    00:00:00 sleep
```

het commando van een  
proces ook tonen

```
student@ubuntu01:~$ sleep 2000 &  
[2] 3122  
student@ubuntu01:~$ pgrep -l sleep  
3117 sleep  
3122 sleep
```



# Basic Process Management

## top

Ordent processen naargelang gebruik van CPU of andere properties.

```
top - 15:50:29 up 22 min,  1 user,  load average: 0,01, 0,05, 0,13
Tasks: 265 total,   1 running, 201 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0,0 us,  0,7 sy,  0,0 ni, 99,3 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
KiB Mem : 2017284 total,  127056 free, 1250416 used,  639812 buff/cache
KiB Swap:  969960 total,  964316 free,   5644 used.  577196 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1263	root	20	0	193384	11252	8832	S	0,3	0,6	0:01.77	vmtoolsd
1958	student	20	0	441620	68204	35904	S	0,3	3,4	0:05.73	Xorg
2091	student	20	0	2971884	171844	75728	S	0,3	8,5	0:19.90	gnome-shell
<b>3126</b>	<b>student</b>	<b>20</b>	<b>0</b>	<b>51316</b>	<b>4360</b>	<b>3636</b>	<b>R</b>	<b>0,3</b>	<b>0,2</b>	<b>0:00.06</b>	<b>top</b>
1	root	20	0	225552	8124	5956	S	0,0	0,4	0:02.96	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kthreadd
4	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/0:+
6	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	mm_percpu_+

h → help      q → quit

# Signalling Processes

**kill**

Process stoppen.

```
student@ubuntudesktop01:~$ sleep 1000 &
[1] 3134
student@ubuntudesktop01:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
student      2529    2514  0   15:29 pts/0        00:00:00 bash
student      3134    2529  0   15:53 pts/0        00:00:00 sleep 1000
student      3135    2529  0   15:53 pts/0        00:00:00 ps -f
student@ubuntudesktop01:~$ kill 3134
[1]+  Terminated                  sleep 1000
student@ubuntudesktop01:~$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
student      2529    2514  0   15:29 pts/0        00:00:00 bash
student      3137    2529  0   15:53 pts/0        00:00:00 ps -f
```

kill 3134      =      kill -15 3134      =      kill -s (SIG)TERM 3134



# Signalling Processes

## list signals

Draaiende processen kunnen signals ontvangen van andere processen of van users.

```
kill -l          (letter l)
```

```
student@ubuntudesktop01:~$ kill -l
```

1) SIGHUP	2) SIGINT	3) SIGQUIT	4) SIGILL	5) SIGTRAP
6) SIGABRT	7) SIGBUS	8) SIGFPE	9) SIGKILL	10) SIGUSR1
11) SIGSEGV	12) SIGUSR2	13) SIGPIPE	14) SIGALRM	15) SIGTERM
16) SIGSTKFLT	17) SIGCHLD	18) SIGCONT	19) SIGSTOP	20) SIGTSTP
21) SIGTTIN	22) SIGTTOU	23) SIGURG	24) SIGXCPU	25) SIGXFSZ
26) SIGVTALRM	27) SIGPROF	28) SIGWINCH	29) SIGIO	30) SIGPWR
31) SIGSYS	34) SIGRTMIN	35) SIGRTMIN+1	36) SIGRTMIN+2	37) SIGRTMIN+3
38) SIGRTMIN+4	39) SIGRTMIN+5	40) SIGRTMIN+6	41) SIGRTMIN+7	42) SIGRTMIN+8
43) SIGRTMIN+9	44) SIGRTMIN+10	45) SIGRTMIN+11	46) SIGRTMIN+12	47) SIGRTMIN+13
48) SIGRTMIN+14	49) SIGRTMIN+15	50) SIGRTMAX-14	51) SIGRTMAX-13	52) SIGRTMAX-12
53) SIGRTMAX-11	54) SIGRTMAX-10	55) SIGRTMAX-9	56) SIGRTMAX-8	57) SIGRTMAX-7
58) SIGRTMAX-6	59) SIGRTMAX-5	60) SIGRTMAX-4	61) SIGRTMAX-3	62) SIGRTMAX-2
63) SIGRTMAX-1	64) SIGRTMAX			

# Signalling Processes

```
kill -1      (cijfer 1)
```

```
SIGHUP
```

Process laten weten dat het de configuratiefile moet herlezen.

- Afhankelijk van het proces kan dit wel of niet.  
Sommige processen moeten gestopt en gestart worden.  
Zie documentatie van het programma/daemon!



# Signalling Processes

**kill -15**

SIGTERM

standard kill

Wordt uitgevoerd als er geen signal wordt meegegeven.  
Er wordt vriendelijk gevraagd aan het proces of het zich wil afsluiten.

**kill -9**

SIGKILL

sure kill

Wordt niet naar het proces gestuurd, maar naar de kernel.  
De kernel zal het proces stoppen!

# Signalling Processes

**kill -19**

SIGSTOP

Met SIGSTOP wordt een proces gepauzeerd (suspended).

Zo'n proces gebruikt geen cpu cycles, maar blijft in het geheugen.

**kill -18**

SIGCONT

Een gepauzeerd proces kan terug gereanimeerd worden met SIGCONT.

# Signalling Processes

```
kill -2
```

```
SIGINT
```

Met `SIGINT` wordt een proces beëindigd/afgebroken (interrupt).

We doen dit via **Ctrl-C**, waarbij het actieve (foreground) proces wordt afgebroken.

zie ook:

```
man 7 signal
```

# Signalling Processes

## pgrep

pgrep sleep  
zou ook gaan, want  
pgrep werkt met pattern  
matching

```
student@ubuntu01:~$ pgrep sleep
3148
3149
student@ubuntu01:~$ pkill sleep
[1]-  Terminated      sleep 1000
[2]+  Terminated      sleep 2000
```

## killall

stuurt SIGTERM naar  
alle processen met  
de opgegeven naam  
(geen pattern matching)

```
student@ubuntu01:~$ sleep 1000 &
[1] 3152
student@ubuntu01:~$ sleep 1500 &
[2] 3153
student@ubuntu01:~$ jobs
[1]-  Running           sleep 1000 &
[2]+  Running           sleep 1500 &
student@ubuntu01:~$ killall sleep
[1]-  Terminated      sleep 1000
[2]+  Terminated      sleep 1500
```

# Signalling Processes

## Signalling vanuit top

Start firefox en enkele andere applicaties  
Zoek naar de processen van firefox

```
student@ubuntudesktop01:~$ pgrep -l firefox  
3200 firefox
```

firefox

of ps -e | grep

Start top, en druk op "k" voor kill

```
PID to signal/kill [default pid = 3483] 3200
```

```
Send pid 3200 signal [15/sigterm] 15
```

Zoek opnieuw naar de processen van firefox

```
student@ubuntudesktop01:~$ pgrep -l firefox  
student@ubuntudesktop01:~$
```

firefox

of ps -e | grep

# Priority and Nice Values

Ieder process heeft een priority en een nice waarde.

Hogere nice waarde = hogere priority waarde

Hogere priority waarde = minder CPU tijd

Je kan dit beïnvloeden met `nice` en `renice`

De verhouding tussen nice, priority en de overeenkomstige CPU-tijd is niet éénduidig en hangt af van dynamische factoren zoals het aantal processen en moeilijke wiskundige formules. Er is dus geen vaste relatie.

# Priority and Nice Values

We bekijken met “top” vier processen die elk evenveel CPU innemen en samen 100% van de CPU innemen

top zonder argumenten uitgevoerd

```
top - 16:24:38 up 5:33, 3 users, load average: 1,83, 0,69, 0,34
Tasks: 318 total, 3 running, 313 sleeping, 2 stopped, 0 zombie
%Cpu(s): 38,2 us, 61,8 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
KiB Mem: 1010460 total, 887128 used, 123332 free, 6976 buffers
KiB Swap: 1046524 total, 60692 used, 985832 free. 170888 cached Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3841	student	20	0	11668	616	520	S	23,1	0,1	0:38.57	proj33
3850	student	20	0	11668	616	520	R	23,1	0,1	0:38.57	proj33
3853	student	20	0	11668	616	520	R	23,1	0,1	0:17.18	proj42
3855	student	20	0	11668	616	520	S	22,8	0,1	0:17.17	proj42

CPU idle time is 0.0

Onze processen gebruiken de volledige CPU-kracht



# Priority and Nice Values

## top -p

Monitoring van specifieke processen

```
top -p 3841,3850,3853,3855
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3841	student	20	0	11668	616	520	R	22,6	0,1	12:27.83	proj33
3850	student	20	0	11668	616	520	S	22,3	0,1	12:27.82	proj33
3853	student	20	0	11668	616	520	R	22,3	0,1	12:06.43	proj42
3855	student	20	0	11668	616	520	R	22,3	0,1	12:06.44	proj42

4 processen – zelfde prioriteit – moeten vechten voor processortijd



# Priority and Nice Values

## renice

Met renice kan je de nice waarde wijzigen van draaiende processen.

Hier +8 voor de proj33-processen

```
student@UbuntuDesktop:~/procs$ renice +8 3841
3841 (process ID) old priority 0, new priority 8
student@UbuntuDesktop:~/procs$ renice +8 3850
3850 (process ID) old priority 0, new priority 8
```

Normale users kunnen een nice waarde toevoegen van 0 tot 19 aan hun eigen processen.

Enkel root kan negatieve nice waarden toekennen tot -20.  
(*Voorzichtig zijn met negatieve nice waarden !!*)

Verlagen van een nice waarde dient ook steeds te gebeuren als root!

# Priority and Nice Values

## impact of nice values

```
top -p 3841,3850,3853,3855
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3853	student	20	0	11668	616	520	R	42,3	0,1	16:17.74	proj42
3855	student	20	0	11668	616	520	S	42,3	0,1	16:17.75	proj42
3841	student	28	8	11668	616	520	R	7,0	0,1	13:44.30	proj33
3850	student	28	8	11668	616	520	S	7,0	0,1	13:43.94	proj33

# Priority and Nice Values

## **nice**

nice werkt hetzelfde als renice, maar wordt gebruikt bij het starten van een proces. Hier zien we de standaardwaarde.

```
student@ubuntudesktop01:~$ xclock &  
[1] 3570
```

```
top -p 3570
```

```
top - 16:09:56 up 41 min, 1 user, load average: 0,03, 0,02, 0,05  
Tasks: 1 total, 0 running, 1 sleeping, 0 stopped, 0 zombie  
%Cpu(s): 0,3 us, 0,3 sy, 0,0 ni, 99,3 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st  
KiB Mem : 2017284 total, 392372 free, 1191784 used, 433128 buff/cache  
KiB Swap: 969960 total, 903644 free, 66316 used. 656788 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3570	student	20	0	78452	7936	6876	S	0,0	0,4	0:00.00	xclock

standaard waarden

# Priority and Nice Values

**nice**

```
student@ubuntudesktop01:~$ nice -n 8 xclock &  
[2] 3572
```

`top -p 3572`

```
top - 16:11:44 up 43 min, 1 user, load average: 0,00, 0,01, 0,04  
Tasks: 1 total, 0 running, 1 sleeping, 0 stopped, 0 zombie  
%Cpu(s): 0,7 us, 0,7 sy, 0,0 ni, 98,6 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st  
KiB Mem : 2017284 total, 389512 free, 1194524 used, 433248 buff/cache  
KiB Swap: 969960 total, 903644 free, 66316 used. 654024 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3572	student	28	8	78448	7920	6864	S	0,0	0,4	0:00.00	xclock

# Background Processes

## jobs

jobs toont de jobs (processen) die in de background draaien in je huidige shell.

```
student@ubuntudesktop01:~$ jobs  
student@ubuntudesktop01:~$
```

Standaard draaien er geen jobs in de background.

# Background Processes

## control-Z

Sommige processen kan je pauzeren (stoppen) naar de background met ctrl-Z

→ SIGSTOP

```
student@ubuntutdesktop01:~$ vi procdemo.txt  
[1]+  Stopped                  vi procdemo.txt
```



# Background Processes

## & ampersand

Processen die gestart worden met een & achteraan, worden naar de background gebracht waar ze blijven uitvoeren

Ook deze jobs kunnen getoond worden met het commando jobs.

```
student@ubuntudesktop01:~$ find / > allfiles.txt 2> /dev/null &  
[2] 3579  
student@ubuntudesktop01:~$ jobs  
[1]+  Stopped                  vi procdemo.txt  
[2]-  Running                  find / > allfiles.txt 2> /dev/null &
```

# Background Processes

**jobs -p** of **jobs -l**

Om de PID van de background processen te tonen.

```
student@ubuntutop01:~$ sleep 500 &
[1] 3585
student@ubuntutop01:~$ sleep 400 &
[2] 3586
student@ubuntutop01:~$ jobs -p
3585
3586
student@ubuntutop01:~$ jobs -l
[1]-  3585  Running                  sleep 500 &
[2]+  3586  Running                  sleep 400 &
```



# Background Processes

## fg

Om een background proces naar de foreground te brengen en opnieuw door te laten uitvoeren.

De parameter is het nummer van de background job.

```
student@ubuntudesktop01:~$ jobs
[1]    Running                  sleep 1000 &
[2]-   Running                  sleep 1000 &
[3]+   Running                  sleep 2000 &
student@ubuntudesktop01:~$ fg 3
sleep 2000
```

# Background Processes

## bg

Om een background proces dat gepauzeerd is terug te starten en op de background te houden.

```
student@ubuntudesktop01:~$ sleep 5000 &
[1] 3787
student@ubuntudesktop01:~$ sleep 3000
^Z
[2]+  Stopped                  sleep 3000
student@ubuntudesktop01:~$ jobs
[1]-  Running                  sleep 5000 &
[2]+  Stopped                  sleep 3000
student@ubuntudesktop01:~$ bg 2
[2]+ sleep 3000 &
student@ubuntudesktop01:~$ jobs
[1]-  Running                  sleep 5000 &
[2]+  Running                  sleep 3000 &
```

# Background Processes

## **SIGSTOP (19) en SIGCONT (18)**

Om een background proces dat runt op de achtergrond opnieuw te pauzeren.

```
student@ubuntudesktop01:~$ sleep 1000 &
[1] 3793
student@ubuntudesktop01:~$ jobs
[1]+  Running                  sleep 1000 &
student@ubuntudesktop01:~$ kill -19 3793
student@ubuntudesktop01:~$ jobs
[1]+  Stopped                  sleep 1000
student@ubuntudesktop01:~$ kill -18 3793
student@ubuntudesktop01:~$ jobs
[1]+  Running                  sleep 1000 &
```

OF: `bg 1`

