

Cours des Systèmes d'Exploitation LINUX

ENSAM Casablanca
2020-2021

Chapitre 4

Gestion des processus

Définitions : le processus

Un processus est un programme en cours d'exécution. Les attributs d'un processus appartiennent à ce que l'on appelle son environnement (ou contexte), parmi eux : le code, les données temporaires, les données permanentes, les fichiers associés, les variables.

L'environnement d'un processus contient aussi les entités que le système lui attribue : les descripteurs, la mémoire allouée, la pile d'exécution du noyau.

Donc, nous pouvons dire :

Programme = Suite d'instructions

Objet statique

Processus = Programme en exécution + Contexte

Objet dynamique

Définitions : multi-tâche

Le système Unix est **multi-tâche** et multi-utilisateur. Le noyau gère l'ensemble des *processus* grâce à un programme appelé l'ordonnanceur (**Eng** - *scheduler*). Autrement dit, l'ordonnanceur fournit une exploitation dans laquelle plusieurs processus sont simultanément en cours d'exécution. Cette simultanéité n'est qu'apparente. Le noyau Linux va affecter à l'unité centrale le processus en cours dont la priorité est la plus importante. Lorsque ce processus est en attente ou si un processus de priorité plus importante apparaît, le noyau libère l'unité centrale du processus en cours et charge le nouveau processus de haute priorité. De surcroît, afin qu'un processus de haute priorité ne monopolise pas l'unité centrale, le noyau Linux va modifier dans le temps les priorités des processus en attente et en cours afin de permettre à l'ensemble des processus d'être exécuté.

Définitions : catégories

Un processus peut être soit :

- Un processus **d'avant plan** (**Eng** : *foreground* (fg)) : qui interagit avec l'utilisateur (ex : un éditeur de texte).
- Un processus **d'arrière plan** ou tâche de fond (**Eng** : *background* (bg)) : il n'a pas besoin d'interaction avec utilisateur pour fonctionner (processus d'un serveur). Les processus de ce type se font appeler des démons (**Eng** : *daemons*).

NB : Nous pouvons faire passer un processus d'un plan à un autre

Définitions : identificateur

Chaque processus possède un numéro qui l'identifie auprès du système. Ce numéro est appelé son **pid** (**Eng :** *process identifier*).

On associe aussi au processus l'identificateur du processus parent **ppid** (**Eng :** *parent process identifier*).

Définitions : les états

Un processus peut se trouver dans quatre états :

- ❑ **actif** (en cours de fonctionnement) (Run): le processus utilise le CPU (le CPU se charge de mettre en pause ou de le réactiver)
- ❑ **prêt** : le processus attend que l'ordonnanceur lui fasse signe.
- ❑ **endormi** (mis en pause) (Sleep): le processus attend un évènement particulier (il ne consomme pas de cpu dans cet état) ;
 - ❖ endormi sur le disque (Disk Sleep).
- ❑ **suspendu** (sTopped): le processus a été interrompu par un signal.
- ❑ **Instable (Zombie)** : le processus est suspendu théoriquement, mais continu à utiliser des ressources mémoire. C'est un état instable causé par des erreurs inattendus (processus planté)

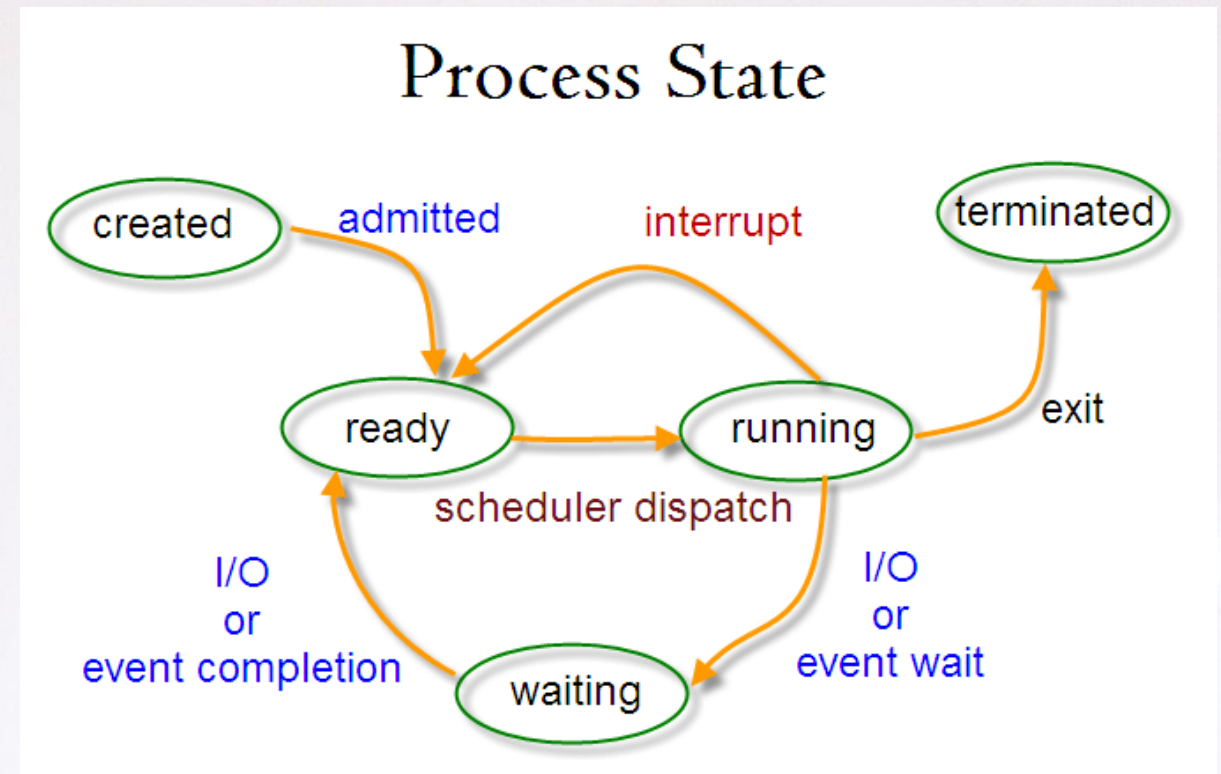
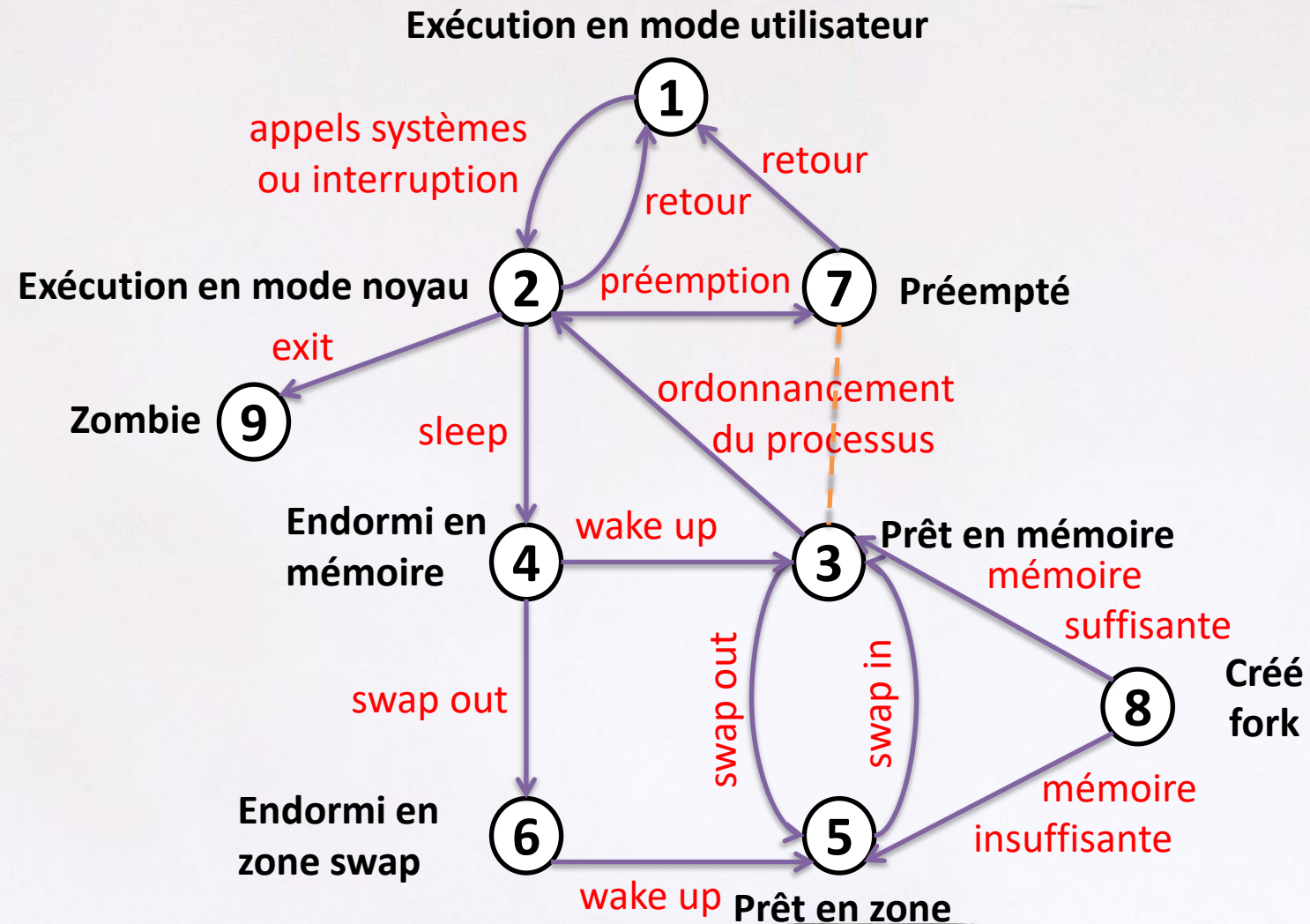


Diagramme du cycle de vie des transitions d'un processus



L'objectif

Nous allons voir dans ce chapitre (en pratique et avec des commandes) les principales commandes CLI qui permettent d'agir sur l'état d'un processus (mettre en pause, reprendre, arrêter, tuer...), à travers un envoi de signal vers le processus en question.

Afficher l'arbre de processus - pstree

L'ensemble des processus d'un système Unix constitue un arbre de processus ; il existe une descendance dans laquelle chaque processus possède un seul processus père (ou parent) qui lui a donné naissance, à l'exception du tout premier processus créé (**init**)* qui n'a pas de prédécesseur.

La commande qui permet d'afficher cet arbre de dépendance est : **ps tree**

```
├─xag-document-po───┐┌{xag-document-po}┐
└─xdg-permission───┐┌2*[{xdg-permission-}]┐
systemd-journal
systemd-logind
systemd-resolve
systemd-timesyn───{systemd-timesyn}
systemd-udev
thermald───{thermald}
udisksd───4*[{udisksd}]
unattended-upgr───{unattended-upgr}
upowerd───2*[{upowerd}]
whoopsie───2*[{whoopsie}]
wpa_supplicant
info@ubuntu:~$
```

From Wikipedia :

The developers of systemd aimed to replace the Linux init system inherited from UNIX System V and Berkeley Software Distribution (BSD) operating systems. Like init, systemd is a daemon that manages other daemons. All daemons, including systemd, are [background processes](#). Systemd is the first daemon to start (during booting) and the last daemon to terminate (during [shutdown](#)). The change is supposed to allow more processing to be done in [parallel](#) during system booting, and to reduce the [computational overhead](#) of the [shell](#).

Afficher l'arbre de processus - pstree

Si nous voulons affiché les processus associé à un certain utilisateur : pstree [nom d'utilisateur]

```
info@ubuntu:~$ pstree info
gdm-x-session--Xorg--{Xorg}
               |
               |--gnome-session-b--ssh-agent
               |                   |
               |                   |--2*[{gnome-session-b}]
               |
               |--2*[{gdm-x-session}]

gnome-keyring-d--3*[{gnome-keyring-d}]

ibus-daemon--ibus-dconf--3*[{ibus-dconf}]
             |
             |--ibus-engine-sim--2*[{ibus-engine-sim}]
             |
             |--ibus-extension--3*[{ibus-extension-}]
             |
             |--ibus-ui-gtk3--3*[{ibus-ui-gtk3}]
             |
             |--2*[{ibus-daemon}]

ibus-x11--2*[{ibus-x11}]

systemd--(sd-pam)
```


Examiner les processus - top

La seconde commande simple qui permet de lister les processus lancés en mémoire : **top**

```
top - 01:11:10 up 18 min, 1 user, load average: 0,16, 0,29, 0,39
Tâches: 285 total, 1 en cours, 284 en veille, 0 arrêté, 0 zombie
%Cpu(s): 0,3 ut, 0,8 sy, 0,0 ni, 98,8 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
MiB Mem : 4514,5 total, 1885,3 libr, 1212,4 util, 1416,8 tamp/cache
MiB Éch: 1162,4 total, 1162,4 libr, 0,0 util. 3021,6 dispo Mem
```

PID	UTIL.	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TEMPS+	COM.
2067	info	20	0	3725884	235404	99524	S	1,0	5,1	0:33.12	gnome-shell
794	root	20	0	134456	9776	8940	S	0,3	0,2	0:00.64	thermald
917	kernoops	20	0	11240	444	0	S	0,3	0,0	0:00.09	kerneloops
2980	info	20	0	973304	53940	40944	S	0,3	1,2	0:07.67	gnome-terminal-
3470	info	20	0	20740	4300	3508	R	0,3	0,1	0:00.48	top
1	root	20	0	103420	12972	8392	S	0,0	0,3	0:07.16	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.04	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-kblockd
8	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	mm_percpu_wq
9	root	20	0	0	0	0	S	0,0	0,0	0:00.24	ksoftirqd/0
10	root	20	0	0	0	0	I	0,0	0,0	0:01.39	rcu_sched
11	root	rt	0	0	0	0	S	0,0	0,0	0:00.01	migration/0
12	root	-51	0	0	0	0	S	0,0	0,0	0:00.00	idle_inject/0

Examiner les processus - top

La première ligne nous donne :

- ❑ l'heure (top)
- ❑ le temps depuis lequel la machine est démarrée (up), et là on vient de lancer la machine.
- ❑ le nombre d'utilisateur connecté à la machine (users) et une information intéressante.
- ❑ la charge moyenne en nombre de processus au cours de la minute passée, des 5 dernières minutes et des 15 dernières minutes. Il est préférable que ces chiffres restent inférieurs au nombre de processeurs de la machine.

La seconde ligne nous donne :

- ❑ le nombre de processus qui tournent
- ❑ leur état suivant qu'ils tournent effectivement (running), qu'ils sont en sommeil (sleeping) car ils n'ont rien à faire, arrêtés (stopped) ou en cours d'arrêt (zombie)

Examiner les processus - top

La troisième ligne indique le temps CPU utilisé en fonction du type d'activité :

- ☐ us: temps CPU au profit de l'utilisateur
- ☐ sy : temps CPU au profit du noyau
- ☐ ni :
- ☐ id :
- ☐ wa :
- ☐ hi : temps cpu passé à réaliser des interruptions matérielles (quelqu'un a appuyé sur une touche du clavier)
- ☐ si :
- ☐ st : Steal time utilisé pour la virtualisation

Examiner les processus - top

La quatrième et la cinquième lignes traitent de la mémoire (physique et swap). La commande nous indique les quantités :

- ❑ totales
- ❑ utilisées
- ❑ disponibles
- ❑ les caches et les buffers sont des espaces de mémoire utilisés par le système pour accélérer le fonctionnement (on garde des choses en mémoire pour ne pas à avoir à les lire sur le disque dur). Ces espaces peuvent être libérés en cas de besoin et donc peuvent être considérés comme de la mémoire libre.

Examiner les processus - top

Le reste de l'écran nous liste les processus en cours et nous fournit certaines informations. Par défaut cette liste est triée par rapport au %CPU. Elle s'actualise automatiquement et contient les champs suivants:

- ❑ **PID** : l'identificateur du processus
- ❑ **USER /UTIL**: l'utilisateur qui l'a lancé
- ❑ **PR** : sa priorité en valeur attribuée par le système
- ❑ **NI** : son degré de gentillesse ou d'amicalité entre [-20 ,19]
- ❑ **COMMAND**: son nom
- ❑ **S** : son statut (S, R, DS...)
- ❑ **%CPU** : le pourcentage d'occupation du CPU
- ❑ **%MEM** : le pourcentage d'occupation de la mémoire
- ❑ ...

Taper :

- ❑ **h** pour l'aide sur les options
- ❑ **L** pour chercher dans cette liste
- ❑ **f** pour modifier l'apparence de la liste
- ❑ **q** pour quitter.

Examiner les processus - ps

La commande **ps** (**Eng** - **p**rocess **s**tatus) permet d'examiner d'une autre façon la liste des processus « tournant » sur le système. Cette commande comprend un nombre très important d'options. Exécuter sans options, elle affiche des informations minimales sur les processus actifs en avant plan.

Options :

- A ou - e : liste tous les processus

```
info@ubuntu:~$ ps
  PID TTY          TIME CMD
 2992 pts/1        00:00:00 bash
 3489 pts/1        00:00:00 ps
info@ubuntu:~$
```


Examiner les processus

À partir du terminal, il est possible de lancer des programmes graphiques (exp: soffice, libreoffice --writer).

Pour l'illustration, on lance le programme **xeyes** à partir du terminal.

Dès son lancement, nous remarquons que nous avons plus la main sur la ligne de commande, par ce que le processus xeyes est passé en premier plan.

Pour ré-avoir la main sur le terminal, il faut faire le raccourcis **ctrl+z**. par conséquent, le processus xeyes a été **mis en pause**.

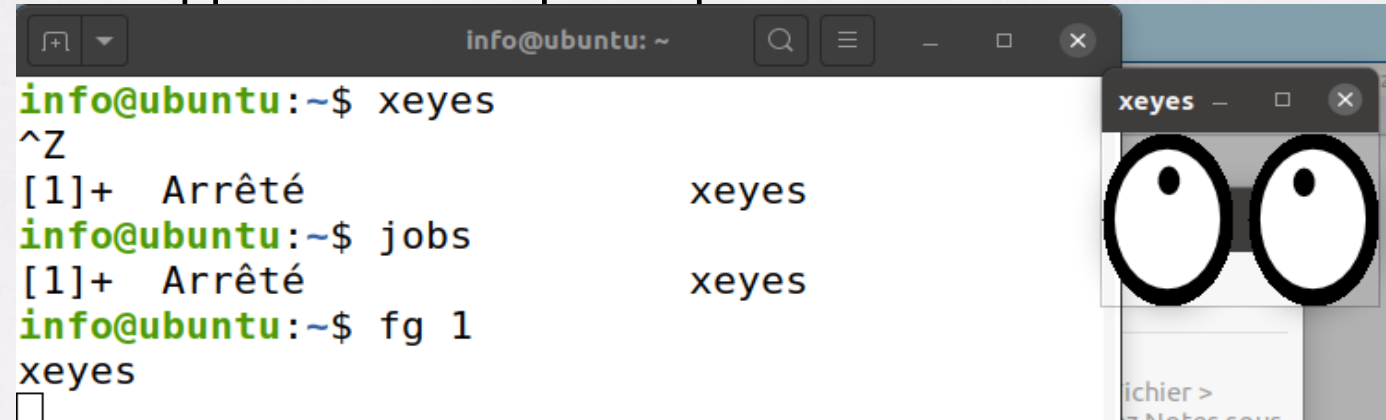
NB : remarquons le numéro qui s'ait associé ([1]) au processus xeyes une fois mis en pose. Ce n'est pas le pid mais plutôt le numéro de la tâche (job).



```
info@ubuntu: ~  
info@ubuntu:~$ xeyes  
[1]+  Arrêté  
info@ubuntu:~$ xeyes
```

Le job control - jobs - fg

La commande **jobs** permet d'examiner les processus qui ont été stoppés et mis en pause pendant la session du terminal :



```
info@ubuntu: ~  
info@ubuntu:~$ xeyes  
^Z  
[1]+  Arrêté          xeyes  
info@ubuntu:~$ jobs  
[1]+  Arrêté          xeyes  
info@ubuntu:~$ fg 1  
xeyes  
□
```

The screenshot shows a terminal window with the title 'info@ubuntu: ~'. The user runs 'xeyes', which is then suspended with '^Z'. The prompt changes to '[1]+ Arrêté' and the process name 'xeyes' is shown. Running 'jobs' lists the suspended process. Finally, 'fg 1' brings the process back to the foreground, and the prompt returns to 'info@ubuntu:~\$'. In the background, another window titled 'xeyes' is visible, showing a cartoon face with large eyes.

Si on veut réactiver le processus stopper par , nous utilisons la commande **fg** (foreground) qui permet de reprendre et repasser l'exécution du processus en avant plan.

Syntaxe :

fg numéro_jobs

Le job control - jobs - bg

Il est possible de passer un ensemble de tâches en exécution sans perdre la main sur le terminal. Pour cela il faut les exécuter en arrière plan avec la commande bg (background) :

```
info@ubuntu:~$ soffice &
[3] 3754
info@ubuntu:~$ xeyes &
[4] 3766
info@ubuntu:~$ jobs
[1]-  Arrêté                  xeyes
[2]+  Arrêté                  soffice
[3]   En cours d'exécution    soffice &
[4]   En cours d'exécution    xeyes &
info@ubuntu:~$
```

Un symbole & s'ajoute indiquant que la tâche est en exécution en arrière plan

En effet, il est possible de lancer au départ les processus avec un & à la fin, et obtenir le même résultat :

Exp : xeyes &

Pid d'un processus - pidof

Le nom d'un processus n'est pas suffisant pour l'identifier, car il se peut qu'un programme soit lancé en plusieurs instance. Pour cela, La plupart des commandes de gestion de processus utilise le **pid** pour identifier un processus.

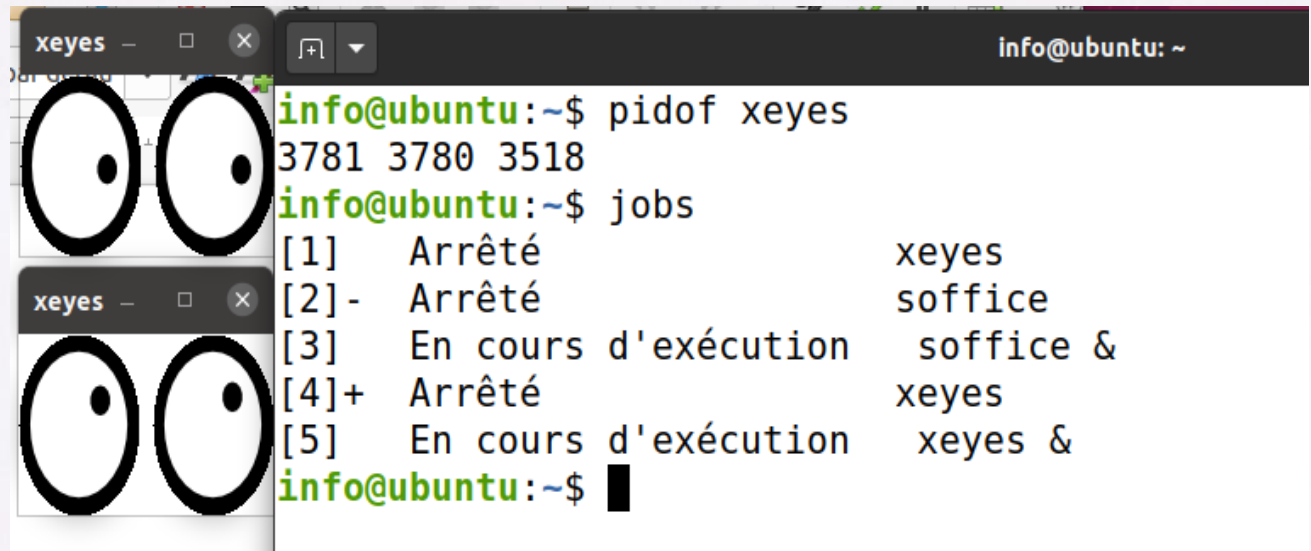
La commande qui permet de retrouver le **pid** d'un processus et **pidof**.

Syntaxe :

pidof nom_processus

Exemple →:

On peut remarquer qu'il y a 3 instance de xeyes chacune a son propre pid.



```
info@ubuntu:~$ pidof xeyes
3781 3780 3518
info@ubuntu:~$ jobs
[1] Arrêté xeyes
[2]- Arrêté soffice
[3] En cours d'exécution soffice &
[4]+ Arrêté xeyes
[5] En cours d'exécution xeyes &
info@ubuntu:~$
```

La priorité d'un processus

Chaque processus actif sur le système s'exécute avec une priorité donnée, aussi appelée valeur amicale ou **Eng – nice**.

Les processus ayant la priorité maximale utilisent plus de ressources système que les autres. Les processus ayant une priorité inférieure s'exécuteront lorsque le système sera libéré par les tâches de priorité plus élevée. Les utilisateurs autres que le super-utilisateur root ne pourront que diminuer la priorité de leurs processus dans la fourchette 0 à 19. La priorité maximale pour un processus est -20, alors que la priorité minimale est 19. Si elle n'est pas définie, tous les processus s'exécutent avec une priorité de 0 par défaut (la priorité de planification de « base »). Le super-utilisateur pourra quant à lui définir de manière arbitraire la priorité de n'importe quel processus.

Changement de priorité - top

La première façon de changer la priorité d'un processus est d'utiliser la commande top (en mode **sudo**). En tapant **r** c'est possible de changer le **nice** d'un processus. Il suffit d'introduire le **pid** du processus, et puis d'introduire une valeur entre -20 et 19, ici nous avons donné la valeur 19 a top :

```
top - 01:29:38 up 37 min, 1 user, load average: 0,08, 0,10, 0,16
Tâches: 284 total, 1 en cours, 279 en veille, 4 arrêté, 0 zombie
%Cpu(s): 0,8 ut, 1,0 sy, 0,0 ni, 98,2 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
MiB Mem : 4514,5 total, 1939,0 libr, 1151,4 util, 1424,1 tamp/cache
MiB Éch: 1162,4 total, 1162,4 libr, 0,0 util. 3080,2 dispo Mem
PID dont la courtoisie va changer [pid par défaut = 2067]

```

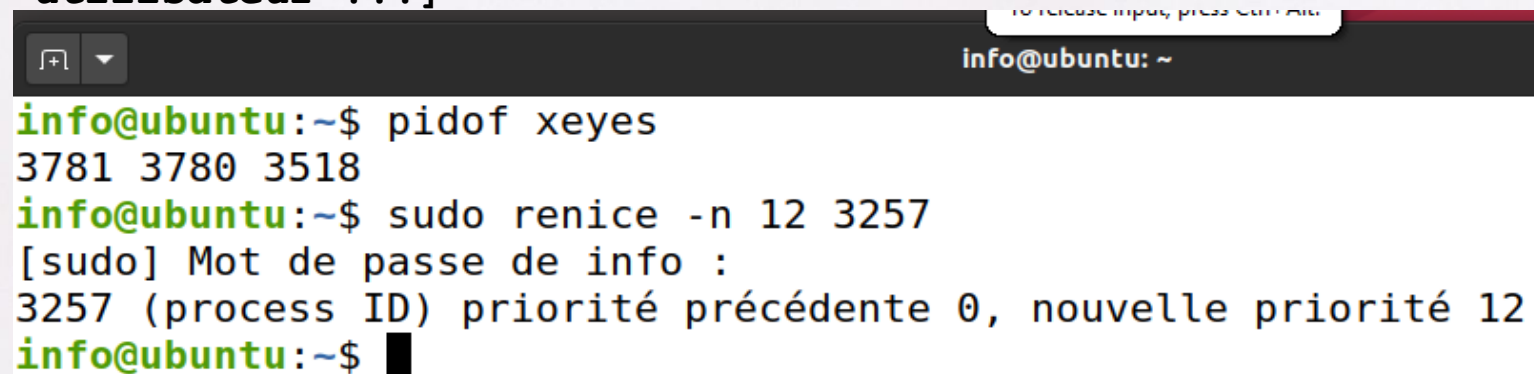
PID	UTIL.	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TEMPS+	COM.
2067	info	20	0	3991816	236240	99756	S	1,7	5,1	0:58.21	gnome-shell
1453	info	20	0	332084	106192	48280	S	0,7	2,3	0:28.19	Xorg
2980	info	20	0	973304	54152	41140	S	0,3	1,2	0:12.89	gnome-terminal-
3799	info	20	0	20744	4140	3348	R	0,3	0,1	0:00.09	top
1	root	20	0	103420	12972	8392	S	0,0	0,3	0:07.51	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.04	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

Changement de priorité - renice

Une autre manière de changer la priorité d'un processus est d'utiliser la commande **renice** (en mode **sudo** pour les valeurs <0) on utilisant l'option **-n** pour spécifier la valeur de la priorité.

Syntaxe :

```
renice priorité [[-p] pid ...] [[-g] pgrp ...] [[-u]
utilisateur ...]
```

A terminal window screenshot from an Ubuntu system. The prompt is 'info@ubuntu: ~'. The user runs 'pidof xeyes' which returns '3781 3780 3518'. Then the user runs 'sudo renice -n 12 3257'. The terminal shows the password prompt '[sudo] Mot de passe de info :', followed by the output '3257 (process ID) priorité précédente 0, nouvelle priorité 12', and finally the prompt 'info@ubuntu:~\$' with a cursor.

```
info@ubuntu:~$ pidof xeyes
3781 3780 3518
info@ubuntu:~$ sudo renice -n 12 3257
[sudo] Mot de passe de info :
3257 (process ID) priorité précédente 0, nouvelle priorité 12
info@ubuntu:~$
```

Les options **-u** permet de changer la priorité de tous les processus d'un utilisateur (**-g** pour le groupe). Exemple:

```
sudo renice +20 -u smi
```

Tuer un processus - kill

La commande **kill** permet d'envoyer des signaux au processus un permettent de changer son état.

Si on lance **kill** sans option sur le pid d'un processus, cela provoque son arrêt normal, exactement comme si on ferme graphiquement l'application.

Syntaxe :

```
kill [-signal] pid
```

Le signal est une liste de (+/- 63) valeurs numériques (ou verbeuse) associé a un type particulier de signal.

En particulier:

```
kill -kill pid / kill -9 pid
```

permet de forcer l'arrêt d'un processus. Ce n'est pas un arrêt normal, donc son utilisation est déconseillé. Sauf pour les processus zombie.

Les signaux - kill

- | | |
|---|---|
| 1 SIGHUP Instruction (HANG UP) - Fin de session | 17 SIGCHLD ou SIGCLD Modification du statut d'un processus fils |
| 2 SIGINT Interruption | 18 SIGCONT Demande de reprise du processus |
| 3 SIGQUIT Instruction (QUIT) | 19 SIGSTOP Demande de suspension imblocuable |
| 4 SIGILL Instruction illégale | 20 SIGTSTP demande de suspension depuis le clavier |
| 5 SIGTRAP Trace trap | 21 SIGTTIN lecture terminal en arrière-plan |
| 6 SIGABRT (ANSI) Instruction (ABORT) | 22 SIGTTOU écriture terminal en arrière-plan |
| 6 SIGIOT (BSD) IOT Trap | 23 SIGURG évènement urgent sur socket |
| 7 SIGBUS Bus error | 24 SIGXCPU temps maximum CPU écoulé |
| 8 SIGFPE Floating-point exception - Exception arithmétique | 25 SIGXFSZ taille maximale de fichier atteinte |
| 9 SIGKILL Instruction (KILL) - termine le processus immédiatement | 26 SIGVTALRM alarme horloge virtuelle |
| 10 SIGUSR1 Signal utilisateur 1 | 27 SIGPROF Profiling alarm clock |
| 11 SIGSEGV Violation de mémoire | 28 SIGWINCH changement de taille de fenêtre |
| 12 SIGUSR2 Signal utilisateur 2 | 29 SIGPOLL (System V) occurrence d'un évènement attendu |
| 13 SIGPIPE Broken PIPE - Erreur PIPE sans lecteur | 29 SIGIO (BSD) I/O possible actuellement |
| 14 SIGALRM Alarme horloge | 30 SIGPWR Power failure restart |
| 15 SIGTERM Signal de terminaison | 31 SIGSYS Erreur d'appel système |
| 16 SIGSTKFLT Stack Fault | 31 SIGUNUSED ... |