TP4 : La gestion des processus

Un **processus** (process) est un programme en **cours d'exécution** dans un ordinateur et occupe un espace en mémoire des ressources CPU (processeur).

Les processus sont référencés par un **identifiant unique**, le **PID**. Ce nombre peut être utilisé pour changer la priorité d'un processus ou pour l'arrêter.

Un processus correspond à n'importe quel exécutable exécuté. Si le processus 2 a été lancé par le processus 1, on l'appelle un **processus fils**. Le processus qui l'a lancé est appelé **processus parent**.

Un processus représente à la fois un programme en cours d'exécution et tout son environnement d'exécution (mémoire, état, identification, propriétaire, père...).

Voici une liste des données d'identification d'un processus :

- Un numéro de processus unique PID (Process ID) : chaque processus Unix est numéroté afin de pouvoir être différencié des autres. Le premier processus lancé par le système est le **processus n° 1** et il s'agit d'un processus appelé généralement **systemd** ou init. On utilise le PID quand on travaille avec un processus. Lancer 10 fois le même programme (même nom) produit 10 PID différents.
- Un numéro de processus parent PPID (Parent Process ID) : chaque processus peut lui-même lancer d'autres processus, des processus enfants (Child process). Chaque enfant reçoit parmi les informations le PID du processus père qui l'a lancé. Tous les processus ont un PPID sauf le processus 0 qui est un pseudo-processus représentant le démarrage du système (crée le 1 init).
- Un numéro d'utilisateur et un numéro de groupe : correspond à l'UID et au GID de l'utilisateur qui a lancé le processus. C'est nécessaire pour que le système sache si le processus a le droit d'accéder à certaines ressources ou non. Les processus enfants héritent de ces informations. Dans certains cas (que nous verrons plus tard) on peut modifier cet état.
- Durée de traitement et priorité : la durée de traitement correspond au temps d'exécution écoulé depuis le dernier réveil du processus. Dans un environnement multitâche, le temps d'exécution est partagé entre les divers processus, et tous ne possèdent pas la même priorité. Les processus de plus haute priorité sont traités en premier. Lorsqu'un processus est inactif, sa priorité augmente afin d'avoir une chance d'être exécuté. Lorsqu'il est actif, sa priorité baisse afin de laisser sa place à un autre. C'est l'ordonnanceur de tâches du système qui gère les priorités et les temps d'exécution.

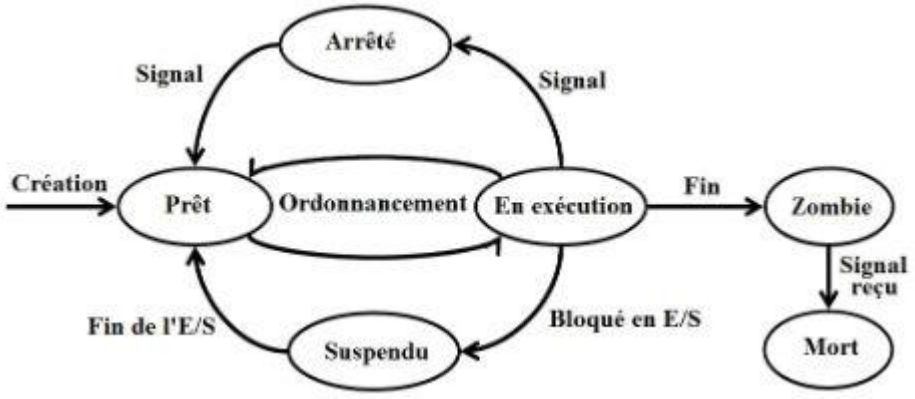
- Répertoire de travail actif : à son lancement, le répertoire courant (celui depuis lequel le processus a été lancé) est transmis au processus. C'est ce répertoire qui servira de base pour les chemins relatifs.
- Fichiers ouverts : table des descripteurs des fichiers ouverts. Par défaut au début seuls trois sont présents : 0, 1 et 2 (les canaux standards). À chaque ouverture de fichier ou de nouveau canal, la table se remplit. À la fermeture du processus, les descripteurs sont fermés (en principe).

On trouve d'autres informations comme la taille de la mémoire allouée, la date de lancement du processus, le terminal d'attachement, l'état du processus, les UID effectif et réel ainsi que les GID effectif et réel.

2. États d'un processus

Durant sa vie (temps entre le lancement et la sortie) un processus peut passer par divers états (ou process state) :

- Prêt à l'exécution
- Exécution en mode utilisateur (user mode)
- Exécution en mode noyau (kernel mode)
- Arrêté (en attente E/S)
- Suspendu
- Fin de processus (zombie)
- Mort.



1. Visualiser les processus en cours d'exécution : ps

La commande **ps** (*process status*) permet d'avoir des informations sur les processus en cours. Lancée seule, elle n'affiche que les processus en cours lancés par l'utilisateur et depuis la console actuelle.

- Exécuter la commande **ps** et donner son résultat.

```
centrecallbd@Ch2Lab1:~$ ps
  PID TTY          TIME CMD
 1994 pts/0    00:00:00 bash
 2385 pts/0    00:00:00 ps
```

- Exécuter la commande **ps -f** et donner son résultat.

```
root@Ch2Lab1:/home/centrecallbd# ps -f
UID          PID  PPID  C STIME TTY          TIME CMD
root        2485  1994  0 08:53 pts/0    00:00:00 sudo su
root        2486  2485  0 08:53 pts/0    00:00:00 su
root        2487  2486  0 08:53 pts/0    00:00:00 bash
root        2493  2487  0 08:59 pts/0    00:00:00 ps -f
```

- Exécuter la commande **ps -ef** et donner son résultat.

```
root        2393     2  0 08:29 ?        00:00:00 [kworker/u8:0-events_
root        2394     2  0 08:29 ?        00:00:00 [kworker/0:0-cgroup_de
root        2485  1994  0 08:53 pts/0    00:00:00 sudo su
root        2486  2485  0 08:53 pts/0    00:00:00 su
root        2487  2486  0 08:53 pts/0    00:00:00 bash
root        2491     2  0 08:54 ?        00:00:00 [kworker/3:0-ata_sff]
root        2494     2  0 08:59 ?        00:00:00 [kworker/3:1-ata_sff]
root        2495  2487  0 08:59 pts/0    00:00:00 ps -ef
```

- Exécuter la commande **ps -u root** (ou un autre utilisateur) et donner son résultat.

```
root@Ch2Lab1:/home/centrecallbd# ps -u root
  PID TTY          TIME CMD
    1 ?        00:00:01 systemd
    2 ?        00:00:00 kthreadd
    3 ?        00:00:00 rcu_gp
    4 ?        00:00:00 rcu_par_gp
    6 ?        00:00:00 kworker/0:0H-kblockd
    8 ?        00:00:00 mm_percpu_wq
    9 ?        00:00:00 ksoftirqd/0
   10 ?        00:00:00 rcu_sched
   11 ?        00:00:00 rcu_bh
   12 ?        00:00:00 migration/0
   14 ?        00:00:00 cpuhp/0
   15 ?        00:00:00 cpuhp/1
   16 ?        00:00:00 migration/1
```

- Exécuter la commande **ps -aux** et donner son résultat.

```
root@Ch2Lab1:/home/centrecallbd# ps -aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1 169632 10208 ?        Ss   08:22   0:01 /sbin/init
root         2  0.0  0.0      0     0 ?        S    08:22   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        I<   08:22   0:00 [rcu_gp]
root         4  0.0  0.0      0     0 ?        I<   08:22   0:00 [rcu_par_gp]
root         6  0.0  0.0      0     0 ?        I<   08:22   0:00 [kworker/0:0H-k
root         8  0.0  0.0      0     0 ?        I<   08:22   0:00 [mm_percpu_wq]
root         9  0.0  0.0      0     0 ?        S    08:22   0:00 [ksoftirqd/0]
root        10  0.0  0.0      0     0 ?        I    08:22   0:00 [rcu_sched]
root        11  0.0  0.0      0     0 ?        I    08:22   0:00 [rcu_bh]
root        12  0.0  0.0      0     0 ?        S    08:22   0:00 [migration/0]
root        14  0.0  0.0      0     0 ?        S    08:22   0:00 [cpuhp/0]
```

Signification de quelques colonnes :

Colonne	Définition
UID	User ID, nom de l'utilisateur.
PID	Process ID, numéro du processus.
PPID	Parent Process ID, numéro du processus père.
C	Facteur de priorité, plus la valeur est grande plus la priorité est élevée.
STIME	Heure de lancement du processus.
TTY	Nom du terminal virtuel depuis lequel le processus a été lancé.
TIME	Durée de traitement du processus.
CMD	Commande exécutée.
VSZ	Taille de mémoire virtuelle que Linux a alloué à un processus en Ko. Quantité maximale de mémoire qu'un processus peut utiliser.
RSS	Taille de la mémoire qu'un processus a actuellement utilisée pour charger toutes ses pages.
S	État du processus S (sleeping) R (running) Z (zombie).
PRI	Priorité du processus.
…	

2. Visualiser l'arborescence des processus : pstree

Par défaut, la commande **pstree** n'est pas installée. Pour pouvoir l'utiliser, il faut installer le paquet **psmisc.**

- Exécuter la commande **pstree** et donner son résultat. Quel est le nom et le PID du processus parent de tous les processus ?

```
root@Ch2Lab1:/home/centrecallbd# pstree
systemd─┬─ModemManager───2*[{ModemManager}]
        ├─NetworkManager─┬─dhclient
        │                └─2*[{NetworkManager}]
        ├─2*[VBoxClient───VBoxClient───3*[{VBoxClient}]]
        ├─VBoxClient───VBoxClient───4*[{VBoxClient}]
        ├─VBoxClient───VBoxClient───2*[{VBoxClient}]
        ├─VBoxDRMClient───4*[{VBoxDRMClient}]
        ├─VBoxService───8*[{VBoxService}]
        ├─accounts-daemon───2*[{accounts-daemon}]
        ├─alsactl
        ├─apache2───6*[apache2]
        ├─avahi-daemon───avahi-daemon
        ├─colord───2*[{colord}]
        ├─cron
        ├─dbus-daemon
        ├─fwupd───4*[{fwupd}]
        ├─gdm3─┬─gdm-session-wor─┬─gdm-wayland-ses─┬─gnome-session-b─┬─evolutio+
        │      │                 │                 │                 └─gnome-sh+
```

3. Visualisation dynamique des processus en cours : top

- Exécuter la commande **top** et donner son résultat.

```
top - 09:12:18 up 49 min,  1 user,  load average: 0,00, 0,05, 0,02
Tasks: 173 total,   1 running, 172 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0,4 us,  0,1 sy,  0,0 ni, 99,5 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem :   8874,5 total,   7314,8 free,    874,6 used,    685,2 buff/cache
MiB Swap:   1022,0 total,   1022,0 free,      0,0 used.   7739,6 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 1113 centrec+  20   0 3784984 284748 105172 S   3,3   3,1   1:02.25 gnome-shell
 1857 centrec+  20   0  370156  39972  29532 S   0,3   0,4   0:01.39 gnome-terminal-
    1 root      20   0  169632  10208   7816 S   0,0   0,1   0:01.66 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-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   0:00.00 ksoftirqd/0
   10 root      20   0       0      0      0 I   0,0   0,0   0:00.35 rcu_sched
   11 root      20   0       0      0      0 I   0,0   0,0   0:00.00 rcu_bh
   12 root      rt   0       0      0      0 S   0,0   0,0   0:00.01 migration/0
   14 root      20   0       0      0      0 S   0,0   0,0   0:00.00 cpuhp/0
   15 root      20   0       0      0      0 S   0,0   0,0   0:00.00 cpuhp/1
   16 root      rt   0       0      0      0 S   0,0   0,0   0:00.65 migration/1
   17 root      20   0       0      0      0 S   0,0   0,0   0:00.00 ksoftirqd/1
   19 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 kworker/1:0H-kblockd
```

- Quelle est la différence entre **ps** et **top** ?

La commande top permet d'avoir en direct les performances des processus en cours, alors que ps permet d'avoir un instant les performances des processus en cours.

4. Arrêter, tuer un processus

Pour arrêter un processus lancé en console, on utilise la combinaison des touches **Ctrl+c**.

Pour tuer un processus, il faut utiliser la commande **kill.** Pour pouvoir l'utiliser, il faut récupérer le PID du processus que l'on souhaite arrêter.

- Consulter l'aide concernant la commande **kill**.
- Lancer le navigateur Firefox en tapant dans votre terminal **firefox**. Que remarquez-vous ?

Ça marche

- Rechercher le **PID** du processus Firefox en exécutant la commande : **ps -u nom-utilisteur | grep firefox**. Quel son PID ?
- Exécuter la commande **kill pid-firefox.** Que remarquez-vous ?

Ça ne marche pas car on na pas renseigné le processus.

- Vérifier le résultat en recherchant le processus lié à **firefox**. Que remarquez-vous ?

Remarque : si le processus refuse de mourir, il faut utiliser la commande **kill -9 pid-processus.**

Pour tuer plusieurs processus portant le même nom, utiliser la commande **kilall nom-processus.**

5. Arrêter le système :

Pour arrêter le système proprement, on utilise les commandes :

- **shutdown now**
- **poweroff**
- **halt** : arrêt du système sans couper l'alimentation.

Pour redémarrer le système :

- **reboot**