Les processus

Cours n°3

Systèmes d’exploitation – ESIEE Paris - 3R-IN3

Document originel : Clément BOIN
Adaptation : Xavier HILAIRE

Plan

- Introduction sur les processus
- Gestion des processus sous Linux
  - principales commandes
  - les signaux
  - les modes d'exécution
  - attendre un processus enfant

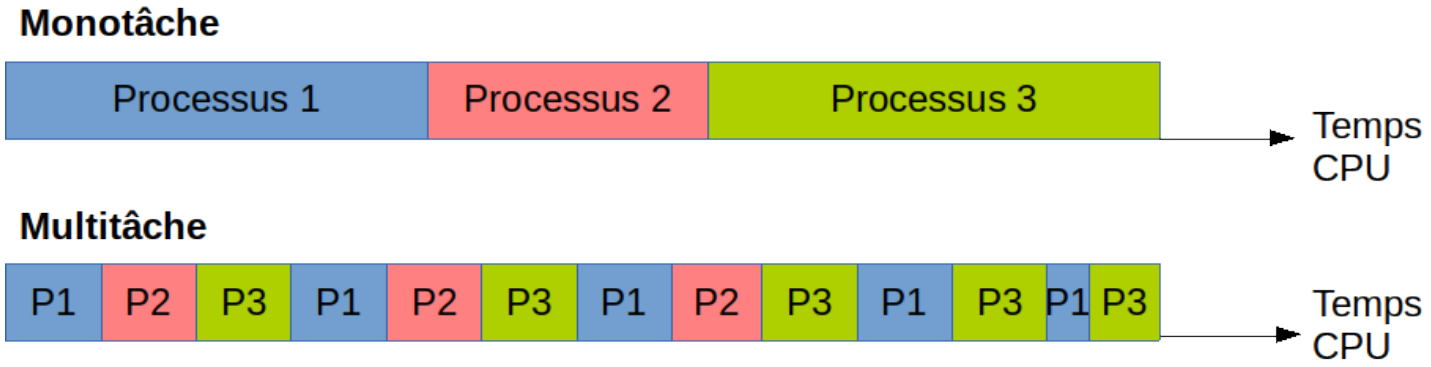
Introduction

Les processus : définitions

- Un processus est une unité de programme en exécution :
  - Un espace mémoire
  - Des ressources utilisées (fichiers, périphériques…)
- Plusieurs processus peuvent être exécutés en même temps.

Monotâche/Multitâche

- Monotâche :
  - un seul processus à la fois
  - Pas de problème de gestion de la mémoire
  - Pas de conflit de ressources
- Multitâche
  - Mise en commun, partage
  - Attention à la gestion, aux conflits et à la sécurité



Gestion de processus (sous Linux)

Arborescence des processus (sous Linux)

- Au démarrage du système un processus nommé systemd est lancé (init sur certains linux)
  - responsable du lancement des autres processus
  - hiérarchie de processus de racine systemd

- Chaque processus est identifié par :
  - un numéro unique (PID : process identifier),
  - l'identité du propriétaire
  - le numéro du processus père (PPID) qui l'a créé : un processus n'a qu'un seul parent
  - systemd a en principe un PID de 1, mais ce nombre peut changer (un systemd par utilisateur, en particulier)

Commande pstree

Hiérarchie des processus

```
clement@boko:~$ pstree
systemd─┬─ModemManager───2*[{ModemManager}]
        ├─NetworkManager─┬─2*[dhclient]
        │                └─2*[{NetworkManager}]
        ├─accounts-daemon───2*[{accounts-daemon}]
        ├─acpid
        ├─apache2───5*[apache2]
        ├─avahi-daemon───avahi-daemon
        ├─boltd───2*[{boltd}]
        ├─colord───2*[{colord}]
        ├─cron
        ├─cups-browsed───2*[{cups-browsed}]
        ├─cupsd───dbus
        ├─dbus-daemon
        ├─dropbox───86*[{dropbox}]
        ├─firefox─┬─Web Content───48*[{Web Content}]
        │         ├─Web Content───37*[{Web Content}]
        │         ├─Web Content───24*[{Web Content}]
        │         ├─WebExtensions───32*[{WebExtensions}]
        │         └─80*[{firefox}]
        ├─fwupd───4*[{fwupd}]
        ├─gdm3─┬─gdm-session-wor─┬─gdm-wayland-ses─┬─gnome-session-b─┬─gnome-sh+
        │      │                 │                 │                 ├─gsd-a11y+
        │      │                 │                 │                 ├─gsd-clip+
```

Sur une machine cliente (extrait)

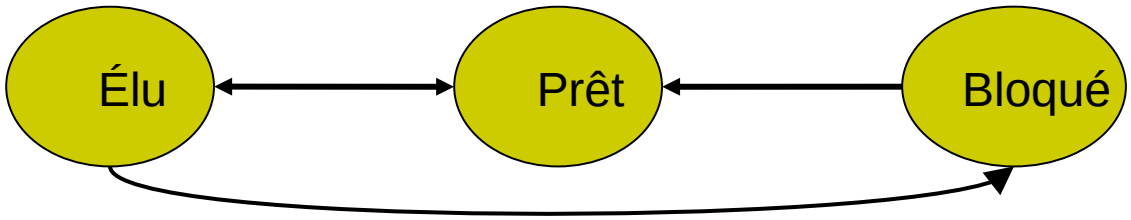
```
test@ubuntu:~$ pstree
systemd─┬─accounts-daemon─┬─{gdbus}
        │                 └─{gmain}
        ├─acpid
        ├─apache2───5*[apache2]
        ├─atd
        ├─cron
        ├─dbus-daemon
        ├─dhclient
        ├─2*[iscsid]
        ├─login───bash───pstree
        ├─lvmetad
        ├─lxcfs───2*[{lxcfs}]
        ├─mdadm
        ├─mysqld───26*[{mysqld}]
        ├─polkitd─┬─{gdbus}
        │         └─{gmain}
        ├─rsyslogd─┬─{in:imklog}
        │          ├─{in:imuxsock}
        │          └─{rs:main Q:Reg}
        ├─snapd───6*[{snapd}]
        ├─systemd───(sd-pam)
        ├─systemd-journal
        ├─systemd-logind
        ├─systemd-timesyn───{sd-resolve}
        └─systemd-udevd
test@ubuntu:~$ _
```

Sur un serveur
sans interface graphique
(tous les processus)

Etat d'un processus

- Chaque processus possède son propre environnement d'exécution.

- 3 états possibles :
  - élu : le processus est en cours d'exécution
  - prêt : tout sauf le processeur
  - bloqué : attend une ressource non encore disponible (saisie d'une valeur par exemple)



Quelques commandes système liées à la gestion des processus

- `top`
- `ps`
- `pstree`

- `free`: état de la mémoire vive

- `iostat –h` : état du processeur

- `nohup` : exécuter une commande résistante aux déconnexions

- `nice`: modifier la priorité d'une commande

La commande **top**

- Permet d'afficher des informations en continu sur l'activité du système.
  - ressources que les processus utilisent
    - quantité de RAM
    - pourcentage de CPU
    - la durée de ce processus depuis son démarrage

- On peut par exemple stopper un processus
  - taper **k**.
  - Saisir le PID du processus
  - Saisir le signal de fin de processus : 15 (SIGTERM)
  - 9 (SIGKILL) est plus brutal !

- Pour quitter top, appuyer sur la touche "**q**".

La commande **ps**

- Permet de connaître les processus actifs à un moment donné
  - Chaque processus est identifié par un nombre unique (PID).
  - TTY : à quel port de terminal est associé le processus.
  - STAT : état du processus
    - S comme "sleep" : endormi
    - R comme "run" : en cours d'exécution
  - TIME : depuis combien de temps le processus utilise les ressources du microprocesseur.
  - COMMAND précise la commande

```
test@ubuntu:~$ ps au
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root      1034  0.0  0.3  65832  3400 tty1     Ss   22:45   0:00 /bin/login --
test      1294  0.0  0.5  22916  5508 tty1     S    22:46   0:00 -bash
test      1427  0.0  0.3  37472  3412 tty1     R+   23:40   0:00 ps au
```

- `ps aux` (syntaxe BSD)
  - permet de connaître les utilisateurs associés à chaque processus (en incluant les applications du serveur graphique X)
- `ps axms`
  - permet d'avoir des informations sur les threads

Communication inter-processus

- Les processus peuvent communiquer entre eux par l'envoi de messages appelés signaux.
  - Asynchrone
  - Nombre limité
  - Réactions prédéfinies

- Arrêter un processus (interrompre)
  - signal SIGINT
  - signal numéro 2
  - généré depuis le clavier en appuyant simultanément sur les touches CTRL+C.

- Le signal de terminaison
  - SIGKILL
  - signal numéro 9
  - ne peut pas être inhibé

La commande **kill**

- Permet d'expédier un signal à un processus en cours.
  `kill [options] PID`

- Si vous n'arrivez pas à tuer un processus, vous devez utiliser la commande :
  `kill -9 PID`

- **`killall`** permet aussi de tuer un processus en indiquant son nom.

Commande kill -l

Liste des signaux

```
clement@boko:~$ trap -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
```

- 64 messages différents
- Interruption du flot de contrôle d’un processus
- Proviennent d'un événement matériel ou logiciel

Les modes d’exécution

Mode d'exécution séquentiel

On tape les commandes les unes après les autres

- Entrée entre chaque commande

```
$sleep 5
pwd
$pwd
/home/uti
```

Apparition au bout de 5 secondes

- On peut également séparer les commandes par des ;

```
$sleep 5;pwd;uname
/home/uti
Linux
```

- Pour arrêter une exécution on tape Ctrl-C

Exécution en arrière plan

- Opérateur &
- Permet de lancer une commande et récupérer la main immédiatement

```
cboin@clement-boin:~$ sleep 5&
[1] 1085
cboin@clement-boin:~$ ps
  PID TTY          TIME CMD
  986 tty1     00:00:00 bash
 1085 tty1     00:00:00 sleep
 1086 tty1     00:00:00 ps
cboin@clement-boin:~$ ps
  PID TTY          TIME CMD
  986 tty1     00:00:00 bash
 1085 tty1     00:00:00 sleep
 1087 tty1     00:00:00 ps
cboin@clement-boin:~$ ps
  PID TTY          TIME CMD
  986 tty1     00:00:00 bash
 1088 tty1     00:00:00 ps
[1]+  Done                    sleep 5
cboin@clement-boin:~$ ps_
```

La commande **wait**

- Commande interne du shell, qui permet d'attendre la fin de ses processus enfants

- Deux formes possibles :
  - wait PID : attend ce PID précisément. Le code de sortie (= valeur renvoyée par exit) du fils décédé est récupérable dans la variable spéciale '?'
  - wait : attend tous les fils. '?' vaut toujours 0.

La commande **wait**

```
[hilairex@pc5352a ~]$ gedit &
[3] 10789
[hilairex@pc5352a ~]$ echo $!
10789
[hilairex@pc5352a ~]$ wait 10789
[3]+  Fini                    gedit
[hilairex@pc5352a ~]$ echo $?
0
[hilairex@pc5352a ~]$ gedit & emacs &
[1] 11117
[2] 11118
[hilairex@pc5352a ~]$ wait
[1]-  Fini                    gedit
[2]+  Fini                    emacs
[hilairex@pc5352a ~]$  echo $?
0
[hilairex@pc5352a ~]$
```

Validation au terminal (entrée) **puis** fermeture de la fenêtre gedit

Fermeture de gedit et emacs (peu importe l'ordre)

La commande **wait**

- Remarque : le fils peut déjà être terminé lorsque l'appel à wait est fait, sans que cela soit un problème