CHAPITRE 4 :

LA GESTION DE PROCESSUS SOUS LINUX

Objectifs spécifiques

- Connaître les différentes commandes Shell de manipulation de processus
- Comprendre les principe de la communication interprocessus

Eléments de contenu

I. Introduction

II. Mode de lancement de processus

III. Enchainement séquentiel de processus

IV. Commandes Shell de manipulation de processus

V. Les redirections

VI. Communication interprocessus : les tubes

VII. Regroupement de processus

Volume Horaire :

Cours : 4 heures 30 mn

TD : 1 heure 30 mn

4.1 Introduction

Lorsqu'on observe un écran, il apparaît clairement que l'ordinateur mène plusieurs activités en parallèle : une horloge, par exemple, affiche l'heure pendant qu'on utilise un traitement de texte. Mieux encore on apprécie de pouvoir imprimer un document tout en continuant à écrire un autre texte. Ces activités sont gérées, en apparence, simultanément alors que la machine ne dispose que d'un processeur unique. Ces processus échangent des informations: ainsi le traitement de texte déclenche régulièrement une copie sur disque qui sauvegarde le texte écrit.

4.2 Rappel

4.2.1 Définition

Un processus est une entité dynamique correspondant à l'exécution d'une suite d'instructions : un programme qui s'exécute, ainsi que ses données, sa pile, son compteur ordinal, son pointeur de pile, ses ressources et les autres contenus de registres nécessaires à son exécution.

4.2.2 Contexte d'exécution d'un processus

Le noyau d'un système d'exploitation maintient une table pour gérer l'ensemble des processus, chaque processus est identifié par son PID, entier de 0 à 2^{15} retourné par le noyau. Le contexte d'un processus est l'ensemble des données qui permettent de reprendre l'exécution d'un processus qui a été interrompu. Le contexte d'un processus est l'ensemble de :

- son état
- son mot d'état: en particulier
  - La valeur des registres actifs
  - Le compteur ordinal
- les valeurs des variables globales statiques ou dynamiques
- son entrée dans la table des processus
- La pile
- les zones de code et de données.

Quand il y a changement de processus courant, il y a réalisation d'une **commutation de mot d'état** et d'un **changement de contexte.** Le noyau s'exécute alors dans le nouveau contexte.

4.2.3 Cycle de vie d'un processus

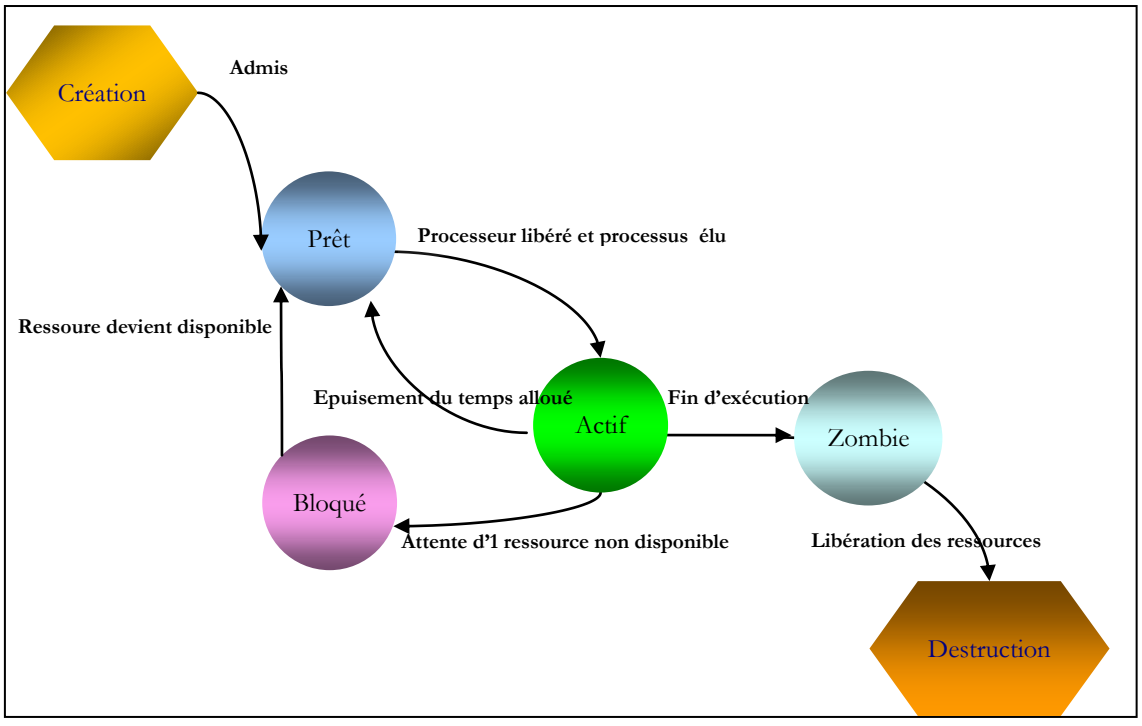

Figure 1 : Cycle de vie d'un processus

Un processus possède un cycle de vie allant de sa naissance à sa mort passant par divers phases d'activité et d'attente. La figure suivante illustre le cycle de vie d'un processus dans le cadre d'un système mono processeur avec stratégie de partage de temps entre les processus.

4.3 Modes de lancement de processus

4.3.1 Procesus en avant plan (foreground)

Lorsqu'on lance une commande, le Shell doit attendre, jusqu'à la fin de l'exécution de la commande avant l'exécution d'une deuxième commande, pour retrouver le prompt de nouveau. On appelle ce processus, un processus en avant plan.

4.3.2 Procesus en arriére plan (background)

Si on ajoute **&** derrière la commande, ceci permet de lancer un processus en arrière plan.
Le Shell n'attend plus la fin de son exécution. On dit qu'il lance la commande en arrière plan. Lorsqu'un utilisateur lance un processus en arrière plan, le Shell affiche entre crochets le numéro de tâche (job) et le PID du processus. Le premier processus lancé aura un numéro de tâche = 1.

```
cat &
[1]   32124
```

4.4 Enchaînements séquentiels de processus

Les commandes peuvent être enchaînées séquentiellement en utilisant un point virgule. Citons par exemple la ligne de commande suivante :

```
echo « bjr » ; cd /etc ; pwd
hello
/etc
```

4.5 Commandes shell de manipulation de processus

4.5.1 La commande ps

La commande ps (process) liste les processus de l'utilisateur. L'option -e affiche tous les processus en cours d'exécution sur un ordinateur et les options –l et –f affichent des informations détaillées. Ainsi, la commande produit une sortie semblable à :

```
ps -ef
UID        PID    PPID  C    STIME    TTY  TIME      CMD
root       1      0     0    12:44:04   -    0:00      /etc/init
```

```
Rsi21     5678     1234    11     12:48:36     pts/0  0:00     ps -ef
Rsi21     8901     5643    0      12:46:31     pts/0  0:00     bash
```

- **UID** nom de l'utilisateur qui a lancé le processus.
- **PID** correspond au numéro du processus.
- **PPID** correspond au numéro du processus parent.
- **C** au facteur de priorité : plus la valeur est grande, plus le processus est prioritaire
- **STIME** correspond à l'heure de lancement du processus
- **TTY** correspond au nom du terminal
- **TIME** correspond à la durée de traitement du processus
- **COMMAND** correspond au nom du processus.

Pour l'exemple donné, à partir d'un Shell nous avons lancé la commande ps -ef, le premier processus a pour PID 9892, le deuxième 10398. Nous notons que le PPID du process **ps -ef** est 10398 le PID du processus bash qui correspond au Shell, par conséquent le Shell est le processus parent, de la commande qu'on vient de taper.

L'utilisation de l'option u permet d'afficher les processus lancés par un utilisateur particulier.

```
ps  -u  imene
PID      TTY   TIME     CMD
3678     pts/0  0:00     cat
4971     pts/0  0:00     bash
```

4.5.2 La commande kill

Un processus peut recevoir plusieurs signaux engendrés par la commande kill, ou bien par l'utilisateur en frappant sur des touches du clavier ou causés par des erreurs matérielles ou logicielles. L'utilisateur peut utiliser des touches pour tuer un processus en avant plan en cliquant sur **CTRL^C** ou le suspendre en cliquant **CTRL^Z.**

La commande kill permet d'envoyer un signal à un processus, et ce signal est identifié par un numéro. Contrairement à ce que son nom semble indiquer, le rôle de cette commande n'est pas forcément de détruire ou de terminer un processus, mais d'envoyer des signaux aux processus.

```
kill [-l] -Num_signal PID1 [PID2...]
```

Le signal est l'un des moyens de communication entre les processus. Lorsqu'on envoie un signal à un processus, celui-doit doit l'intercepter et réagir en fonction de celui-ci. Certains signaux peuvent être ignorés, d'autres non.

```
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
```

Figure 2 : Résultat de la commande kill –l

Les signaux sont numérotés et nommés, la liste des signaux peut être visualisée en appliquant l'option –l à la commande kill.

Pour tuer un processus, on doit connaître son PID, et écrire la commande suivante :

kill -9 <PID>

Exemple :

```
ps
PID    TTY   TIME   CMD
1234   pts/2  00:00   bash
3456   pts/2  00:00   sleep
14733  pts/2  00:00   ls
2134   pts/2  00:00   ps
kill -9 14733
```

La commande ci-dessus permet de tuer le processus ayant le PID=10789 qui est le processus ls.

Ci-dessous les numéros, les noms et la description de quelques signaux que peuvent recevoir un processus :

Numéro	Nom	Description
9	KILL	Un signal qui va tuer à coup sûr le processus qui le reçoit.
15	TERM	(Terminate) envoyé à un processus pour le terminer de façon élégante si possible.
19	STOP	Ce signal permet de suspendre un processus.

4.5.3 Commande jobs

La commande jobs permet d'afficher la liste des tâches du Shell courant (suspendus ou s'exécutant en tâche de fond) avec leurs numéros de tâches, les PIDs et les états des processus.

```
jobs -l
[1]  12509  Running     sleep
[2]  3271   Suspended   cat
```

4.5.4 La commande fg

Relancer l'exécution d'un processus en arrière plan en un processus en avant plan.

```
fg %n
```

n désigne le numéro de tâche.

4.5.5 La commande bg

Relancer l'exécution d'un processus suspendu en processus en arrière plan.

```
bg %n
```

n désigne le numéro de tâche.

4.5.6 La commande top

La commande **top** vous permet d'afficher des informations en continu sur l'activité du système. Elle permet surtout de suivre les ressources que les processus utilisent (quantité de RAM, pourcentage de CPU, la durée de ce processus depuis son démarrage). Pour quitter l'utilitaire top, il suffit d'appuyer sur la touche **"q"**.

4.5.7 La commande time

La commande **time** mesure les durées d'exécution d'une commande, idéale pour connaître les temps de traitement, et retourne trois valeurs :

- real : durée totale d'exécution de la commande
- user : durée du temps CPU nécessaire pour exécuter le programme
- system : durée du temps CPU nécessaire pour exécuter les commandes liées à l'OS (appels système au sein d'un programme).

Si le résultat est inférieur à 10, la machine dispose de bonnes performances, au-delà de 20 la charge de la machine est trop lourde (performances basses).

4.6 Les redirections

Chaque processus dialogue avec l'extérieur par le biais de trois fichiers particuliers qui sont ouverts en permanence :

- **0 l'entrée standard** (standard input) où sont lues les données dont le processus a besoin (par défaut c'est le clavier).
- **1 la sortie standard** (standard output) sur laquelle écrit le résultat à l'utilisateur (par défaut c'est l'écran).
- **2 la sortie d'erreur** (standard error output) sur laquelle sont écrits les éventuels messages d'erreur (par défaut c'est l'écran).

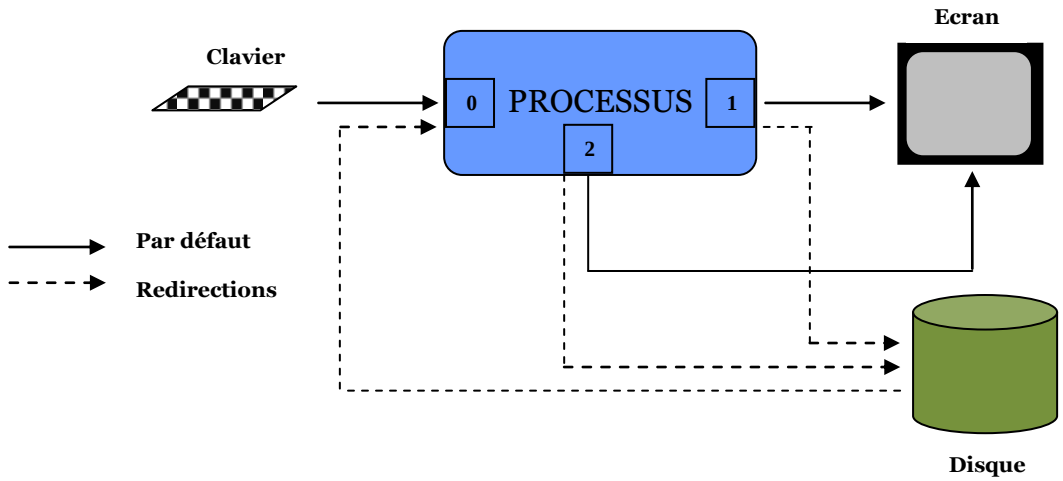

Figure 3 : Redirections

Par défaut, ces fichiers sont liés au terminal, et représentent le clavier (input) et l'écran (output).

Au moment de la création d'un processus, on peut rediriger ses entrées-sorties standards : cela signifie que l'on connecte les fichiers standards vers d'autres organes, comme les fichiers, les tubes, un périphérique...

La redirection des sorties peut être réalisée par effacement et création du fichier ou par ajout à la fin du fichier si ce dernier existe sinon un nouveau fichier sera crée. Pour la redirection de l'entrée standard il est évident que le fichier doit exister.

- Le caractère **<** suivie d'un nom de fichier indique la redirection de l'entrée standard à partir de ce fichier.
- Le caractère **>** suivie d'un nom de fichier indique la redirection de la sortie standard vers ce fichier. Ce dernier sera écrasé s'il existe déjà et si la variable booléenne noclobber n'est pas initialisée.

- Le caractère **2>** suivie d'un nom de fichier indique la redirection de la sortie d'erreur vers ce fichier.
- Si on double le caractère **>>**, l'information ou la redirection sera ajoutée au fichier de sortie.

Exemple

```
ls
f1   work   docs   fich2
ls > resultat
cat resultat
f1
work
docs
fich2
resultat
```

Pour rediriger la sortie standard à la fin du fichier sans écraser son contenu, on remplace > par >>

```
 echo « Bonjour » >> resultat
cat resultat
f1
work
docs
fich2
resultat
Bonjour
```

La sortie d'erreur peut être redirigée en utilisant *2>fichier*, et si on ne veut pas écraser le fichier mais ajouter à la fin du fichier, on utilise (*2>>fichier*).

```
 ls
f1   work   docs   fich2
ls  f1  toto
f1
toto:Aucun fichier de ce type
ls  f1 toto 2> erreurs
f1
cat erreurs
```

```
toto:Aucun fichier de ce type
```

La redirection de l'entrée standard se fait par <fichier. Dans ce cas, la commande lit ses données à partir du fichier.

```
cat toto
titi
tata
bebe
coco
sort < toto
bebe
coco
tata
titi
```

4.7 La communication interprocessus : les tubes (pipes)

Le principe des tubes est assez simple : la sortie standard d'un processus est redirigée vers l'entrée d'un tube dont la sortie est dirigée vers l'entrée standard d'un autre processus. Ainsi les deux processus peuvent échanger des données sans passer par un fichier intermédiaire de l'arborescence.

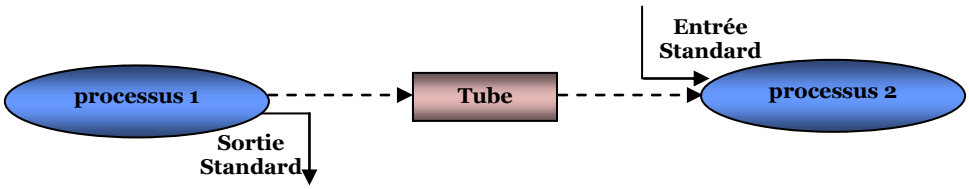

Figure 4 : Principe de fonctionnement d'un tube de communication interprocessus

Le lancement concurrent de processus communiquant deux par deux par l'intermédiaire des tubes sera réalisé par une commande de la forme :

commande1 | commande2 | ... | commandeN

Exemple 1

Sans l'utilisation d'un tube:

```
who > tmp ; wc -l < tmp
5
```

```
rm tmp
```

Avec l'utilisation d'un tube:

```
who | wc –l
5
```

Exemple 1

Pour vérifier que plusieurs processus existent simultanément, il suffit de lancer la séquence suivante :

```
ps –l | tee /dev/tty | wc –l
F      S      UID    PID   PPID   PRI ... CMD
1      S      102    241   234    158     –bash
1      R      102    294   241    179     ps
1      S      102    295   241    154     tee
1      S      102    296   241    154     wc
5
```

La commande **tee** permet d'envoyer sur la sortie standard et sur le fichier référencé en paramètre, ce qu'elle lit sur son entrée standard (dans ce cas, le fichier référencé est le fichier écran **/dev/tty** qui permet de visualiser sur l'écran l'entrée standard de la commande **tee**, qui est en fait la sortie standard de la commande **ps**). On remarque que les 3 processus concurrents existent tous simultanément.

Exemples :

```
ls | wc –l
```

Cette commande compte le nombre de lignes du résultat de ls qui est en fait le nombre de fichiers du répertoire courant.

```
who|sort
```

Cette commande permet de trier alphabétiquement la liste des utilisateurs connectés.

```
ls –l /home | more
```

Grâce à un pipe, la commande more peut recevoir sur son entrée standard ce qu'une autre commande a envoyé vers la sortie standard. On peut ainsi effectuer un affichage page par page du résultat envoyé par la commande **ls –l /home**.

```
ls -l /bin | grep '^d'
```

Cette commande permet d'envoyer la sortie standard de la commande ls –l /bin (la liste des fichiers et des répertoires du répertoire /bin avec un format détaillé). Ensuite, la commande grep '^d' recherche dans le résultat de ls –l /bin les lignes qui commencent par d, donc ce sont les lignes qui correspondent à des répertoires.

4.8 Regroupement des processus

Il existe plusieurs méthodes pour enchaîner des commandes sur une même ligne :

- Exécution sous condition d'erreur :

  cmd1 || cmd2 || cmd3 || … || cmdN

  Si cmd1 ne se termine pas correctement, alors cmd2 est exécuté, et ainsi de suite. Sinon, si cmd1 s'exécute correctement les autres commandes ne seront pas exécutées.

- Exécution sous condition de réussite :

  cmd1 && cmd2 && cmd3 && … && cmdN

  Si cmd1 se termine correctement, alors cmd2 est exécuté, et ainsi de suite. Sinon, si cmd1 est exécutée avec erreur les autres commandes ne seront pas éxécutées.

Exercice

a. Ecrire un script shell permettant de compter le nombre de processus

```
#!/bin/bash
ps | tail +2 |wc -l
```

b. Ecrire un script shell qui affiche la liste des processus triées par ordre alphabétique.

```
#!/bin/bash
ps | sort
```