Faculté des Sciences & Techniques — Université de Limoges

Licence 3ème année

Programmation Concurrente

TP n°1

Appels systèmes

Cycle de vie d'un processus

Lancement en « tache de fond » d'un processus, on ajoute le caractère esperluette, « & » à la fin :

```
xterm
$ mon_processus &
```

Exemple :

```
xterm
more /etc/passwd &
```

Il est possible de suspendre un processus déjà lancé, en utilisant le raccourci clavier « `CTRL-Z` ».
Un processus en attente de l'entrée standard (le clavier sous la ligne de commande) va être suspendu.
Pour obtenir la liste des processus lancés depuis la ligne de commande et pour l'utilisateur courant :

```
xterm
$ jobs
[1]+  Stopped                 more /etc/passwd
```

ou la commande `ps` :

```
xterm
$ ps
PID   TTY    TIME   CMD
1547  ttys000  0:00.04  -bash
1620  ttys000  0:00.01  more /etc/passwd
```

La commande `ps -alx` permet d'obtenir la liste de tous les processus exécutés sur la machine.
Il est possible de récupérer un processus suspendu en utilisant la commande `fg`, pour « *foreground* » ou bien de le mettre si possible en tache de fond avec `bg` pour « *background* ».

La commande `kill` permet d'envoyer un « signal » à un processus. Ce signal correspond à une interruption que le processus va traiter immédiatement lors de sa réception. Parmi les différents signaux, il y a le signal 15, terminaison propre, et 9, terminaison directe.
Pour utiliser cette commande, on indique le numéro du signal à transmettre ainsi que le pid, « processus identifier » du processus cible.

```
xterm
$ kill -9 1547
```

Il également possible de « tuer » un processus en mode interactif (le processus est en attente de l'entrée standard du clavier) à l'aide du raccourci « `CTRL-C` ».

Rappels de compilation C

Pour compiler un source « `mon_source.c` » :

```
xterm
$ gcc -o mon_executable mon_source.c
```

Pour compiler en mode « debug », permettant de joindre la « table des symboles » :

```
xterm
$ gcc -g -o mon_executable mon_source.c
```

Pour avoir le détail des fonctions appelées dans votre programme :

```
xterm
$ nm mon_executable
```

Pour avoir des informations concernant la taille des segments :

```
xterm
$ size mon_executable
```

Création de processus avec `fork`

1 – Faire tourner le programme C suivant :

```
1 void main(void)
2 {
3    system("ps");
4 }
```

Expliquer les résultats obtenus.

2 – Tapez le code source suivant :

```
1  #include <unistd.h>
2  #include <stdio.h>
3  void main(void)
4  {
5     int f=fork();
6     switch(f)
7     {
8        case -1 :    perror("impossible de fourcher");
9              exit(1);
10       case 0 :    printf("Je suis le processus %d de pere %d\n",getpid(),
   getppid());
11             break;
12       default :    printf("Je suis le processus pere %d de fils %d\n",get
   pid(),f);
13    }
14 }
```

a. Faire exécuter ce programme tel quel, puis le modifier de façon à ce que le processus père appelle la commande « `ps -fl > pspere` », et le processus fils, la commande « `ps -fl > psfils` ». Commenter les résultats obtenus.

b. Modifier le programme de façon que le processus père attende la terminaison du processus fils avant d'afficher son identité (on utilisera la fonction `wait()`).

3 – Vous écrirez et testerez les exercices 3, 4, 5 et 6 de la fiche de TD n°1.