

TD Processus et ordonnancement

A Processus Linux

Soit le programme `essai.c` suivant :

```
#include <stdio.h>
#include <unistd.h>

main()
{
    int pid;
    pid = fork();
    if (pid == 0)
    {for(;;)
        printf ("je suis le fils\n");
    }
    else
    {for(;;)
        printf("je suis le père\n");
    }
}
```

1) On compile ce programme pour générer un exécutable appelé `essai` dont on lance l'exécution. La commande `ps -l` permettant d'afficher les caractéristiques de l'ensemble des processus de l'utilisateur donne les éléments suivants :

```
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME MD
100 S 0 467 1 0 60 0 - 256 read_c tty5 00:00:00 mingetty
100 S 0 468 1 0 60 0 - 256 read_c tty6 00:00:00 mingetty
100 S 0 576 570 0 60 0 - 498 read_c pts/0 00:00:00 cat
100 S 0 580 578 0 70 0 - 576 wait4 pts/1 00:00:00 bash
100 S 0 581 579 0 60 0 - 77 wait4 pts/2 00:00:00 bash
000 S 0 592 581 2 61 0 - 253 down_f pts/2 00:00:01 essai
040 S 0 593 592 2 61 0 - 253 write_ pts/2 00:00:01 essai
100 R 0 599 580 0 73 0 - 652 - pts/1 00:00:00 ps
```

Les champs `S`, `PID`, `PPID` et `CMD` codent respectivement l'état du processus (`S` pour Stopped, `R` pour Running), la valeur du `PID` et du `PPID` pour le processus et le nom du programme exécuté. On tape la commande `kill -9 593` qui entraîne la terminaison du processus dont le `pid` 593 est spécifié en argument.

L'exécution de la commande `ps -l` donne à présent le résultat suivant.
Que pouvez-vous dire à ce sujet ?

```
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD
100 S 0 467 1 0 60 0 - 256 read_c tty5 00:00:00 mingetty
100 S 0 468 1 0 60 0 - 256 read_c tty6 00:00:00 mingetty
100 S 0 580 578 0 70 0 - 576 wait4 pts/1 00:00:00 bash
100 S 0 581 579 0 60 0 - 77 wait4 pts/2 00:00:00 bash
000 S 0 592 581 2 60 0 - 253 write_ pts/2 00:00:03 essai
```

```
444 Z 0 593 592 2 60 0 - 0 do_exi pts/2 00:00:03 essai <zombie>
100 R 0 601 580 0 73 0 - 651 - pts/1 00:00:00 ps
```

2) On lance de nouveau l'exécution du programme essai. La commande `ps -l` permettant d'afficher les caractéristiques de l'ensemble des processus de l'utilisateur donne les éléments suivants :

```
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD
100 S 0 467 1 0 60 0 - 256 read_c tty5 00:00:00 mingetty
100 S 0 468 1 0 60 0 - 256 read_c tty6 00:00:00 mingetty
100 S 0 576 570 0 60 0 - 498 read_c pts/0 00:00:00 cat
100 S 0 580 578 0 65 0 - 576 wait4 pts/1 00:00:00 bash
100 S 0 581 579 0 60 0 - 577 wait4 pts/2 00:00:00 bash
000 S 0 604 581 3 60 0 - 253 write_ pts/2 00:00:00 essai
040 S 0 605 604 2 60 0 - 253 down_f pts/2 00:00:00 essai
100 R 0 606 580 0 76 0 - 649 - pts/1 00:00:00 ps
```

On tape la commande `kill -9 604` qui entraîne la terminaison du processus dont le pid 604 est spécifié en argument. L'exécution de la commande `ps -l` donne à présent le résultat suivant.

Que pouvez-vous dire à ce sujet ?

```
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD
100 S 0 467 1 0 60 0 - 256 read_c tty5 00:00:00 mingetty
100 S 0 468 1 0 60 0 - 256 read_c tty6 00:00:00 mingetty
100 S 0 576 570 0 60 0 - 498 read_c pts/0 00:00:00 cat
100 S 0 580 578 0 65 0 - 576 wait4 pts/1 00:00:00 bash
100 S 0 581 579 0 60 0 - 577 wait4 pts/2 00:00:00 bash
100 S 0 581 579 0 60 0 - 577 read_c pts/2 00:00:00 bash
040 S 0 605 1 2 60 0 - 253 write_ pts/2 00:00:02 essai
100 R 0 608 580 0 73 0 - 649 - pts/1 00:00:00 ps
```

3) Le programme `essai.c` est modifié :

```
#include <stdio.h>
main()
{
  int pid;
  pid = fork();
  if (pid == 0)
    {for(;;)
     printf ("je suis le fils\n");
    }
  else
    {
     printf("je suis le père\n");
     wait();
    }
}
```

On recompile ce programme pour générer un nouvel exécutable appelé `essai` dont on lance l'exécution. La commande `ps -l` permettant d'afficher les caractéristiques de l'ensemble des processus de l'utilisateur donne les éléments suivants :

```
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD
100 S 0 467 1 0 60 0 - 256 read_c tty5 00:00:00 mingetty
100 S 0 468 1 0 60 0 - 256 read_c tty6 00:00:00 mingetty
100 S 0 576 570 0 60 0 - 498 read_c pts/0 00:00:00 cat
100 S 0 580 578 0 70 0 - 577 wait4 pts/1 00:00:00 bash
100 S 0 581 579 0 60 0 - 577 wait4 pts/2 00:00:00 bash
000 S 0 627 581 0 60 0 - 253 wait4 pts/2 00:00:00 essai
040 S 0 628 627 3 61 0 - 253 write_ pts/2 00:00:00 essai
100 R 0 629 580 0 72 0 - 649 - pts/1 00:00:00 ps
```

On tape la commande `kill -9 628` qui entraîne la terminaison du processus dont le `pid 628` est spécifié en argument. L'exécution de la commande `ps -l` donne à présent le résultat suivant.

Que pouvez-vous dire à ce sujet ?

```
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD
100 S 0 467 1 0 60 0 - 256 read_c tty5 00:00:00 mingetty
100 S 0 468 1 0 60 0 - 256 read_c tty6 00:00:00 mingetty
100 S 0 576 570 0 60 0 - 498 read_c pts/0 00:00:00 cat
100 S 0 580 578 0 70 0 - 577 wait4 pts/1 00:00:00 bash
100 S 0 581 579 0 60 0 - 577 wait4 pts/2 00:00:00 bash
100 R 0 31 580 0 73 0 - 648 - pts/1 00:00:00 ps
```

B Ordonnancement

B.1 Exercice 1

Pour chacune des questions suivantes, vous choisissez la ou les bonnes réponses en justifiant votre choix, soit par un calcul, soit par une explication.

On considère quatre processus P_1 , P_2 , P_3 et P_4 , soumis dans cet ordre, dont les caractéristiques sont les suivantes :

	Temps d'exécution	Priorité (plus petite valeur = plus grande priorité)
P1	8 unités	4
P2	8 unités	3
P3	10 unités	1
P4	4 unités	2

1) Avec un ordonnancement FIFO, les temps de réponse des quatre processus P_1 , P_2 , P_3 et P_4 sont respectivement :

- 8, 16, 26, 30
- 8, 18, 28, 32
- 4, 14, 22, 30

2) Avec un ordonnancement par priorité, les temps de réponse des quatre processus P_1 , P_2 , P_3 et P_4 sont respectivement :

- 8, 16, 26, 30
- 10, 14, 22, 30
- 8, 16, 20, 30

3) Avec un ordonnancement par quantum de temps, $Q = 2$, ordre initial FIFO, les temps de réponse des quatre processus P_1 , P_2 , P_3 et P_4 sont respectivement :

- 16, 24, 26, 30
- 24, 26, 30, 16
- 24, 30, 18, 12

B.2 Exercice 2

On considère un système composé de 3 processus Linux P_1 , P_2 et P_3 appartenant à la classe d'ordonnancement `SCHED_RR`. L'ordonnancement est effectué selon une méthode du tourniquet associée à une priorité. Ainsi les processus P_1 et P_2 sont de même priorité `PRIOR1` tandis que le processus P_3 est de priorité `PRIOR2`, avec $PRIOR2 < PRIOR1$. Le quantum de temps Q est égal à 10 ms.

Par ailleurs chaque processus peut effectuer des entrées-sorties avec un disque.

Le disque est une ressource non requérable et l'ordre de service des requêtes disque est du type premier arrivé, premier servi (FIFO). Les profils des trois processus sont les suivants :

Processus P1	Processus P2	Processus P3
20 ms calcul 5 ms entrées-sorties 20 ms calcul 10 ms entrées-sorties 10 ms calcul	30 ms calcul 15 ms entrées-sorties 10 ms calcul 5 ms entrées-sorties 10 ms calcul	30 ms calcul 5 ms entrées-sorties 5 ms calcul

Représentez sur un chronogramme d'exécution où vous ferez apparaître les états prêt, élu (`task_running`) et bloqué (`task_interruptible`), l'exécution des trois processus et les changements d'états intervenant. Pour chacun des trois processus, donnez son temps de réponse.

B.3 Exercice 3

On considère un système monoprocesseur de type Linux dans lequel les processus partagent un disque comme seule ressource autre que le processeur. Cette ressource n'est accessible qu'en accès exclusif et non requérable, c'est-à-dire qu'une commande disque lancée pour le compte d'un processus se termine normalement avant de pouvoir en lancer une autre. Un processus peut être en exécution, en attente d'entrées-sorties, en entrées-sorties ou en attente du processeur. Les demandes d'entrées-sorties sont gérées à l'ancienneté.

Dans ce système, on considère 4 processus P_1 , P_2 , P_3 , P_4 pour lesquels on sait que :

- P_1 et P_2 sont des processus appartenant à la classe `SCHED_FIFO`. Dans cette classe, le processeur est donné au processus de plus haute priorité. Ce processus peut être préempté par un processus de la même classe ayant une priorité supérieure ;
- P_3 et P_4 sont des processus appartenant à la classe `SCHED_RR`. Dans cette classe, le processeur est donné au processus de plus haute priorité pour un quantum de temps égal à 10 ms. La politique appliquée est celle du tourniquet.

Les processus de la classe `SCHED_FIFO` sont toujours plus prioritaires que les processus de la classe `SCHED_RR`. Les priorités des processus sont égales à 50 pour le processus P_1 , 49 pour le processus P_2 , 49 pour le processus P_3 et 49 pour le processus P_4 . La plus grande valeur correspond à la priorité la plus forte.

Les quatre processus ont le comportement suivant :

P_1 Calcul pendant 40 ms
Lecture disque pendant 50 ms
Calcul pendant 30 ms
Lecture disque pendant 40 ms
Calcul pendant 10 ms

P_2 Calcul pendant 30 ms
Lecture disque pendant 80 ms
Calcul pendant 70 ms
Lecture disque pendant 20 ms
Calcul pendant 10 ms

P_3 Calcul pendant 40 ms
Lecture disque pendant 40 ms
Calcul pendant 10 ms

P_4 Calcul pendant 100 ms

Établissez le chronogramme d'exécution des quatre processus en figurant les états prêt, élu, en attente d'entrées-sorties et en entrées-sorties.

B.4 Exercice 4

On considère quatre processus P_1 , P_2 , P_3 et P_4 dont les caractéristiques sont les suivantes :

	Temps d'exécution	Priorité (plus petite valeur = plus grande priorité)
P1	6 unités	3
P2	4 unités	2
P3	14 unités	4
P4	2 unités	1

- 1) Les quatre processus sont présents à l'instant $t = 0$ dans la file des processus prêts dans l'ordre donné par la liste (P_1 est la tête de liste). L'ordonnancement est en FIFO. Donnez l'ordre d'exécution des processus et le temps de réponse de chaque processus.
- 2) Les quatre processus sont présents à l'instant $t = 0$ dans la file des processus prêts dans l'ordre donné par les priorités. L'ordonnancement est par priorité. Donnez l'ordre d'exécution des processus et le temps de réponse de chaque processus.
- 3) Les quatre processus sont présents à l'instant $t = 0$ dans la file des processus prêts dans l'ordre donné par la liste (P_1 est la tête de liste). L'ordonnancement est en tourniquet avec un quantum Q égal à deux unités. Donnez l'ordre d'exécution des processus et le temps de réponse de chaque processus.
- 4) Les quatre processus parviennent à présent à des instants différents dans la file d'attente des processus prêts. Ainsi :
 - Date arrivée de $P_1 = 0$;
 - Date arrivée de $P_2 = 3$;
 - Date arrivée de $P_3 = 2$;
 - Date arrivée de $P_4 = 5$.

L'ordonnancement se fait selon un mode priorité préemptif. Représentez l'exécution des quatre processus et donnez leur temps de réponse.

C Primitives fork, exec, wait

Soient les programmes suivants (a,b). Pour chacun d'eux, expliquez combien de processus sont créés et quels sont les affichages réalisés.

`calcul` est un exécutable qui additionne les deux entiers transmis en paramètres et affiche le résultat.

(a)

```
Pour i de 1 à 2
{
    ret = fork();
    if (ret == 0) afficher (i) ;
    else afficher(ret);
}
```

(b)

```
Pour i de 1 à 2
{
    ret = fork();
    if (ret == 0) execl("/home/calcul", "calcul", "5", "7", NULL);
}
wait();
```

D Processus zombie et orphelin

Écrivez un programme qui engendre deux fils, dont l'un deviendra zombie, et l'autre deviendra orphelin.