esprit Se former autrement HONORIS UNITED UNIVERSITIES	**Examen** **Semestre 2** **Session Principale**
Module : Système d'exploitation avancé 1 **Enseignant(s)** : UP Système **Classe(s) :** 4 année	**Documents non autorisés** **Nombre de pages** : 3 **Date** : 26/05/2023 **Heure** 16h00 **Durée** : 01h30

Questions de cours : (3 pts)

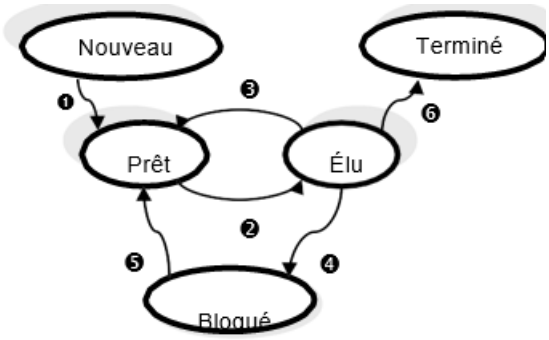
1. Qu'est-ce qu'un PCB ? **0.25pts**

PCB est une structure de données qui contient des informations sur le processus qui lui est lié.
Attributs d'un PCB:

- PID et PPID,
- État,
- Priorité,
- Compteur ordinal,
- Fichiers ouverts,
- Pointeurs: seg. code, seg. données, seg. Pile,
- Temps d'exécution.

2. Quels sont les différents états d'un processus ? Quelles sont les transitions entre ces états ?**0.5pts**

Prêt (Ready) : le processus attend son tour pour s'exécuter
Élu (Running) : les instructions sont en cours d'exécution.
Bloqué (Sleep) : le processus bloqué en attente d'événement : signal, E/S,



(1) Création du processus

(2) Allocation du processeur

(3) Fin du temps alloué sur le processeur, l'exécution du processus n'est pas terminée

(4) Opération E/S

(5) Fin opération E/S

(6) Exécution terminée

3. Comment juger l'efficacité d'un algorithme d'ordonnancement ?**0.5pts**

Voici quelques critères couramment utilisés pour évaluer l'efficacité d'un algorithme d'ordonnancement des processus :

Temps d'exécution : Il s'agit du temps total nécessaire pour exécuter tous les processus depuis leur arrivée jusqu'à leur achèvement. Un bon algorithme d'ordonnancement devrait minimiser le temps d'exécution global.
Temps d'attente : Il mesure la durée pendant laquelle les processus attendent d'être exécutés. Réduire les temps d'attente permet d'améliorer l'efficacité de l'algorithme.
Temps de réponse : Il s'agit du temps écoulé entre la soumission d'une requête ou d'un processus et le début de son exécution. Un algorithme d'ordonnancement efficace devrait réduire le temps de réponse pour améliorer la réactivité du système.

Rendement du processeur = (Temps d'activité du processeur / Temps total) * 100

Un rendement de processeur élevé indique une utilisation efficace du processeur,

4. Qu'appelle-t-on un processus qui s'est terminé mais son père n'a pas encore lu son code de retour.**0.25pts**
   Processus zombie
5. Que signifie préempter un processus ?
   A. Suspendre son exécution au profit d'un autre processus. **0.5pts**
   B. Arrêter définitivement son exécution au profit d'un autre processus.
   C. Geler le processus pour un temps indéterminé (fini).
   D. Transférer le processus en zone de swap
6. Qu'est-ce qu'un processus au sens d'un O.S ?**0.5pts**
   A. Une opération d'Entrée/Sortie.
   B. Un utilisateur connecté au système et utilisant des ressources.
   C. L'instance d'un programme en cours d'exécution.
   D. Un fichier statique stocké sur une mémoire de masse.
7. Quel est le rôle d'un ordonnanceur au sein d'un O.S. ? **0.5pts**
   A. Ordonnancer l'utilisation de la mémoire virtuelle.
   B. Ordonnancer les opérations d'E/S.
   C. Ordonnancer les interruptions provoquées par les opérations d'E/S.
   D. Ordonnancer les processus à exécuter selon un ou des critères.

Exercice 1 : (8 pts)

1. Écrire un programme en langage C qui utilise la fonction **dup2** pour rediriger la sortie standard vers un fichier "**file.txt**" en utilisant le descripteur de fichier **file_desc**. Les messages sont écrits dans le fichier en utilisant la fonction **printf**. **3.5pts**

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

```
#include<fcntl.h>

int main()

{
   system("touch file.txt");

   int file_desc = open("file.txt",O_WRONLY | O_APPEND);

   // here the newfd is the file descriptor of stdout (i.e. 1)
   dup2(file_desc, 1) ;

   // All the printf statements will be written in the file
   // "file.txt"
   printf("I will be printed in the file file.txt\n");

return 0;
}
```

2. Écrire un programme **tube_anonyme_multi** qui consiste à créer deux processus écrivains qui écrivent des chiffres dans le tube : le premier écrivain écrit les chiffres de 1 à 5 et le deuxième écrivain écrit les chiffres de 6 à 10. Le processus lecteur lit les chiffres du tube et les ajoute à la somme totale. Finalement, la somme totale est affichée à l'écran. **4.5pts**

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

int childProcess(int start, int end, int writePipe) {
   int sum = 0;
```

```
    for (int i = start; i <= end; i++) {
        write(writePipe, &i, sizeof(int));
        sum += i;
    }
    close(writePipe);
    return(sum);
}

int main() {

    int sum1 = 0;
    int sum2 = 0;
    int pipefd[2];
    pipe(pipefd);

    pid_t pid1, pid2;
    pid1 = fork();

    if (pid1 == 0) {
        // Premier processus écrivain
        close(pipefd[0]); // Ferme l'extrémité de lecture du tube
        sum1=childProcess(1, 5, pipefd[1]);
    } else {
        pid2 = fork();

        if (pid2 == 0) {
            // Deuxième processus écrivain
            close(pipefd[0]); // Ferme l'extrémité de lecture du tube
            sum2=childProcess(6, 10, pipefd[1]);
        } else {
            // Processus lecteur
            close(pipefd[1]); // Ferme l'extrémité d'écriture du tube

            int sum = 0;
            int num;
          char bytesRead;

            while ((bytesRead = read(pipefd[0], &num, sizeof(int))) > 0) {
                sum += num;
            }
```

```
        close(pipefd[0]); // Ferme l'extrémité de lecture du tube

        wait(NULL); // Attend la terminaison du premier écrivain
        wait(NULL); // Attend la terminaison du deuxième écrivain
        printf("Somme totale : %d\n", sum + sum1 + sum2);
    }
  }

  return 0;
}
```

Exercice 2 : (4 pts)

Soient les codes ci-dessous :

Code 1	Code 2

int main()
{
pid_t pid ;
int value = 9 ;
if (fork() == 0) {
value += 10;
return 0; }
else
{
wait (NULL) ;
printf (" value = %d " , value) ; }
return 0 ;
}
	int main(){
	pid_t pid;
	int status;
	pid = fork();
	if (pid < 0) {
	perror("fork");
	return 1; }
	else if (pid == 0) {
	printf("Je suis le processus fils (PID: %d)\n", getpid());
	execlp("python3", "python3", "script.py", NULL);
	perror("exec");
	return 1;
	}
	else {
	printf("Je suis le processus parent (PID: %d)\n", getpid());
	waitpid(pid, &status, 0);
	return 0; }}

1. Donner le résultat d'exécution du code 1 . **2pts**
   Le code affiche **value=9** ; pour le père value=9, pour le fils value =19
2. Quel est l'objectif du code 2 pour le processus père et fils **2pts**
   Le fils exécute un script python script.py et le père attend la terminaison de son fils

Exercice 3 : (5 pts)

On considère l'ensemble des processus suivants :

N° Processus	Date d'arrivée	Temps CPU	Priorité
1	7h00	10 mn	2
2	7h00	15 mn	3
3	7h03	8 mn	4
4	7h10	18 mn	5

$$\textbf{Priorité} = \frac{Temps\ d'attente + Temps\ CPU\ restant}{Temps\ CPU}$$

1. En utilisant un algorithme d'ordonnancement basé sur la **priorité préemptif**. Donnez le diagramme de Gantt en se basant sur les priorités données dans le tableau. (**Les priorités sont croissantes : 5 est le plus prioritaire) 2.5 pts**

2. On suppose que la priorité des processus est dynamique au cours du temps. Pour calculer la priorité d'un processus, on utilise la formule ci-dessus.

- Lors des calculs, on arrondit selon l'exemple suivant : 3.5 ou 3.6 ➔ 4, 3.1 ou 3.4➔3.
- Au démarrage, les priorités des processus sont égales à leurs priorités statiques (indiquées dans le tableau).
- Le temps d'attente d'un processus est son temps d'attente depuis sa dernière exécution.
- Si on a le choix entre deux processus de même priorité, on choisit celui qui attend depuis le plus longtemps.

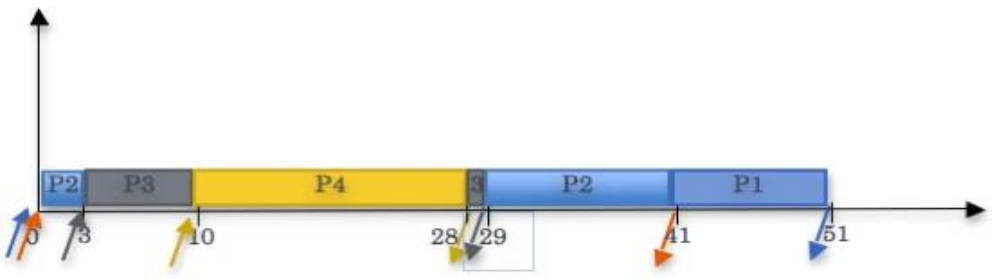
Donnez le diagramme de Gantt sachant que la priorité est recalculée toutes les 5 minutes. **2.5 pts**

Ordonnancement avec priorité dynamique

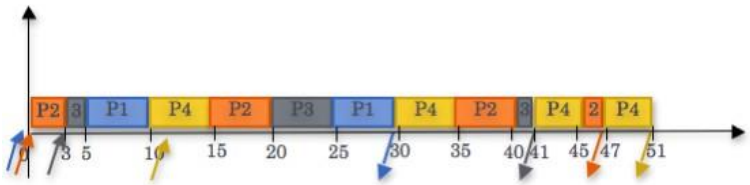
On considère l'ensemble des processus suivants :

Processus	Date d'arrivée	Temps CPU	Priorité
P1	7h00	10 mn	2
P2	7h00	15 mn	3
P3	7h03	8 mn	4
P4	7h10	18 mn	5

A-



B-



	0	3	5	10	15	20	25	30	35	40	41	45
P 1	2	2	5+10/10 =2	0+5/10= 1	5+5/10=1	10+5/10 =2	15+5/10= 2					
P 2	3	3	2+12/15 =1	7+12/15 =1	12+12/15 =2	0+7/15= 0	5+7/15=1	10+7/15= 1	15+7/15 =1	0+2/15 =0	0	5+2/15 =0
P 3		4	0+6/8=1	5+6/8=1	10+6/8=2	15+6/8= 3	0+1/8=0	5+1/8=1	10+1/8= 1	15+1/8 =2		
P 4				5	0+13/18= 1	5+13/18 =1	10+13/18 =1	15+13/18 =2	0+8/18= 0	5+8/18 =1	1	0+4/18 =0

Bonne chance