TD1 : Gestion des processus

Exercice 1 :

1. Soient trois processus A, B, C soumis dans cet ordre de temps d'exécution respectif 4, 2, 9. Quelles propositions sont exactes ?
   a) Avec une politique *FIFO*, l'ordre d'exécution est C, B puis A.
   b) Avec une politique *Plus Court d'Abord*, l'ordre d'exécution est B, A, puis C.
   c) Avec une politique par priorité fixe, l'ordre d'exécution est A, B, puis C.
2. Le processus A de priorité 5 s'exécute. Le processus B de priorité 7 se réveille. Quelles sont les propositions justes ?
   a) B interrompt l'exécution de A car B est plus prioritaire et l'ordonnancement est préemptif
   b) A continue son exécution car il est plus prioritaire et l'ordonnancement est préemptif.
   c) A continue son exécution car l'ordonnancement est non préemptif.
   d) B interrompt l'exécution de A car B est plus prioritaire et l'ordonnancement est non préemptif.
3. Pour chacun des algorithmes suivants, indiquer s'il est préemptif et/ou non préemptif. Justifier.

Algorithmes	Préemptif	Non Préemptif
FCFS – Premier arrivé, premier servi		
SJF – Plus court d'abord		
SRTF – Temps restant le plus court d'abord		
RR – Round Robin		
PS – Ordonnancement par priorité		

Exercice 2 : ordonnancement non préemptif

Soit le tableau suivant où les processus arrivent tous à l'instant $t = 0$ dans l'ordre suivant : P1, P2, P3, P4 et P5. Le temps est considéré ici par unité d'utilisation (uu). On suppose que plus le nombre de priorité est petit, plus le processus correspondant est prioritaire.

Processus	Durée utilisation CPU (uu)	Priorité
P1	8	4
P2	6	1
P3	1	2
P4	9	2
P5	3	3

1. En supposant que l'ordonnancement est **non préemptif** ; alors pour chacun de ces algorithmes *FCFS, SJF, RR (quantum=1uu) et PS* :
   – Dessiner son diagramme de Gantt.
   – Déterminer le temps d'attente de chaque processus, puis le temps d'attente moyen.
   – Déterminer le temps de traitement (temps de rotation, temps de complétion, délai d'exécution) de chaque processus et le temps de traitement moyen.

   Temps de traitement = date de fin d'exécution du processus - date d'arrivée du processus
2. Comparer les performances des algorithmes

ousmane.sadio@ucad.edu.sn

uu					5					10					15					20					25							
P1																															
P2																															
P3																															
P4																															
P5																															

Exercice 3 : ordonnancement préemptif vs non préemptif

Soit le tableau suivant où les processus P1, P2, P3, P4 et P5 arrivent à des instants différents. Le temps est considéré ici par unité d'utilisation (uu). On suppose que plus le nombre de priorité est petit, plus le processus correspondant est prioritaire.

Processus	**Instant d'arrivé**	**Durée utilisation CPU (uu)**	**Priorité**
P1	0	8	4
P2	2	6	1
P3	2	1	2
P4	1	9	2
P5	3	3	3

1. Pour chacun de ces algorithmes *FCFS, PS non préemptive, PS, RR (quantum=1uu)*
   - Dessiner son diagramme de Gantt.
   - Déterminer le temps d'attente de chaque processus, puis le temps d'attente moyen.
   - Déterminer le temps de traitement (temps de rotation, temps de complétion, délai d'exécution) de chaque processus et le temps de traitement moyen.

     Temps de traitement = date de fin d'exécution du processus - date d'arrivée du processus
2. Comparer les performances des algorithmes.

uu					5					10					15					20					25							
P1																															
P2																															
P3																															
P4																															
P5																															

Exercice 4 : multithreading préemptif

Un système exécute cinq threads (T1, T2, T3, T4, T5) sur un processeur double-cœur avec les arrivées et durées suivantes :

Thread	**Instant d'arrivé (ms)**	**Durée utilisation CPU (ms)**	**Priorité**
P1	0	8	4
P2	1	6	2
P3	2	4	1
P4	3	3	2
P5	4	5	1

1. Proposer un ordonnancement basé sur la priorité.
2. Simuler l'effet du temps de commutation de contexte en supposant un coût de 1 ms.
3. Evaluer l'impact sur le temps total d'exécution.

ousmane.sadio@ucad.edu.sn

Cœur1

uu					5					10					15					20					25				
P1																												
P2																												
P3																												
P4																												
P5																												

Cœur2

uu					5					10					15					20					25				
P1																												
P2																												
P3																												
P4																												
P5																												

Exercice 5 :

On considère le programme suivant :

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

int main(void)
{
	fork(); printf("fork 1\n");
	fork(); printf("fork 2\n");
	fork(); printf("fork 3\n");
	wait();
	return(0);
}
```

1. Expliquez l'exécution de ce programme (*sans l'exécuter*). Précisez en particulier :
   - Le nombre total de processus engendrés par cette exécution.
   - Combien d'occurrences de chaque type de messages fork *i* sont affichées ?
   - Quel est l'ordre d'apparition des différents types de messages ?
2. Remplacez le premier appel à `fork()` par un appel à `execv()` qui exécute le programme lui-même. Que se passe-t-il ?

Exercice 6 :

Quels sont les traces générées par le code suivant. Expliquer et commenter.

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

int i;

void main()
{
	pid_t process_id;
	i=0;
```


ousmane.sadio@ucad.edu.sn

```
        process_id = fork();
        if (process_id == 0){
                i+=10;
                printf("fils %d\n",i);
                i+=20;
                printf("fils %d\n",i);
        }
        else
        {
                i+=1000;
                printf("pere %d\n",i);
                i+=2000;
                printf("pere %d\n",i);
                wait();
        }
}
```