Partie 6 : Ordonnancement de processus

Exercice 1 :

Considérez un système d'exploitation qui ordonnance les processus selon l'algorithme du tourniquet. La file des processus prêts contient des pointeurs vers les entrées de la table des processus (les descripteurs des processus).

1) Citez deux problèmes qui peuvent survenir si la file contient des éléments identiques (deux ou plusieurs pointeurs égaux).

2) Supposez que le système d'exploitation est composé de deux unités de contrôle (deux processeurs CPU1 et CPU2) et d'une unité d'E/S. Chaque processeur exécute l'algorithme du tourniquet avec un quantum de trois unités de temps (qt = 3). Tous les processus prêts sont dans une même file d'attente. La commutation de contexte est supposée de durée nulle.

Considérez trois processus A, B et C décrits dans le tableau suivant :

Processus	Instant d'arrivée	Temps d'exécution
A	0	4 unités de CPU, 2 unités d'E/S, 2 unités de CPU
B	2	3 unités de CPU, 4 unités d'E/S, 2 unités de CPU
C	3.5	5 unités de CPU

La première ligne signifie que le processus A arrive dans le système à l'instant 0, son exécution nécessite dans l'ordre 4 unités de temps CPU, 2 unités de temps d'E/S et 2 unités de temps CPU. Au départ le processus A est élu par le processeur CPU1.

Si plusieurs événements surviennent en même temps, vous supposerez les priorités suivantes :

- Le CPU1 a la priorité d'accès à la file des processus prêts par rapport au CPU2.
- A la fin d'un quantum, le processus non terminé en cours est suspendu uniquement si la file des processus prêts n'est pas vide. Le traitement réalisé à la fin d'un quantum est plus prioritaire que celui d'une fin d'E/S qui, à son tour, est plus prioritaire que l'arrivée de nouveaux processus dans le système.

a) Donnez les diagrammes de Gantt montrant l'allocation des deux processeurs, de l'unité d'E/S et l'évolution des états des files d'attente (celle des processus prêts et celle des processus en attente de l'unité d'E/S).

b) Calculez le temps moyen de virement (temps moyen de séjour).

Exercice 2 :

1) Expliquez la différence entre les Ordonnanceurs d'exécution non préemptifs (ex : ceux des premières versions de MacOS) et préemptifs (ex. ceux d'Unix, de Windows2000…).

2) Supposez que plusieurs processus sont à l'état prêt et que le processus élu exécute le programme suivant :

```
int main ( )
{
        for ( int i=0; i<=100; i = min(100,i++));
        return 0;
}
```

a) Que se passe-t-il pour les processus qui sont à l'état prêt, si l'Ordonnanceur d'exécution est non préemptif?

b) Que se passe-t-il pour les processus qui sont à l'état prêt, si l'Ordonnanceur d'exécution est préemptif?

3) A l'instant t, deux processus utilisateur P1 et P2 existent dans un système monoprocesseur. Le processus P1 est composé de deux threads. Le processus P2 est composé de trois threads. Les temps nécessaires à leurs exécutions sont résumés dans le tableau suivant :

Proc.	Threads	Temps d'exécution
P1	T11 T12	1 unité de CPU 3 unités de CPU
P2	T21 T22 T23	3 unités de CPU 2 unités de CPU 1 unité de CPU

Supposez que le processeur est libre. Donnez les diagrammes de Gantt montrant l'allocation du processeur, pour chacun des cas suivants :

a) Les threads sont supportés par le noyau (threads noyau). Le noyau ordonnance l'exécution des threads selon l'algorithme du tourniquet (Round Robin) avec un quantum de 2 unités.

La file d'attente des threads prêts, à l'instant t, est : →T23 T12 T22 T21 T11 →

T11 est en tête de file.

b) Les threads sont implémentés entièrement au niveau utilisateur. Le noyau ordonnance l'exécution des processus selon l'algorithme du tourniquet avec un quantum de 2 unités. Les threads sont ordonnancés au niveau utilisateur selon l'algorithme du tourniquet avec un quantum de 1 unité.

Supposez que le processeur est libre et que les processus sont passés à l'état prêt dans l'ordre suivant :

P1 puis P2

Dans P1 : T11 puis T12

Dans P2 : T21 puis T22 puis T23

Dans tous les cas, le temps de commutation est supposé nul.

c) Calculez, pour chaque cas, le temps de virement (temps de séjour), relatif à l'instant t, de chaque processus. Comparez puis commentez les résultats.

Exercice 3 :

Une firme bien connue concurrente de Apple décide de lancer un nouvel appareil qui permet d'écouter de la musique de format mp3 et d'en « stocker » plus de 80 Go, soit 40Go de plus que le fameux « iPod ». Cependant, avant de faire l'envoi sur le marché, elle décide de vous engager pour vérifier si leurs choix technologiques ont été judicieux pour ce type d'application.

Ce fameux appareil possède 64 Mo de mémoire principale et 80 Go de mémoire secondaire. Il peut en tout temps n'exécuter que trois processus (en plus de l'OS) sur son processeur embarqué. Le processus le plus important parmi ces trois est l'Afficheur chargé en mémoire dès le départ (désigné par A). De plus, un processus Transfert (nommé T) permet de transférer les données d'un ordinateur à sa mémoire secondaire à partir d'un port externe de type « USB ». Finalement,

le dernier processus Son (appelé S) permet d'envoyer de la musique à partir des données de la mémoire secondaire vers un port externe d'écouteur. Il est important de savoir que l'écoute de la musique est la principale fonction de cet engin. L'affichage et le son doivent toujours être parfaits, tandis que le transfert de données n'est que secondaire.

Dans cette gigantesque firme, deux groupes d'ingénieurs ont donné leur avis sur le choix de la politique d'ordonnancement. Cependant, ces deux groupes ont proposé des algorithmes différents. Un diagramme de Gantt représente chacun de ces algorithmes. Chaque case du diagramme correspond à une unité de temps.

GROUPE A:

A	A	A	T	T	T	A	A	A	T	T	T	S	S	S	A	A	A	S	S	S	A	A	A	S	A	A	A

GROUPE B:

A	A	A	T	T	T	A	A	A	S	S	S	A	A	A	S	S	S	A	A	A	S	A	A	A	T	T	T

Sachant que :

- Le processus Afficheur (A) fait une entrée/sortie après chaque 3 unités de temps (de calcul), les autres n'en font aucune.
- Chaque entrée/sortie dure une unité de temps.
- Les processus terminent durant la simulation.

Processus	**Temps d'arrivée**
A	0
T	2
S	8

1- Identifiez les politiques d'ordonnancement choisies par les deux groupes.

2- Calculez les temps de séjour des processus, ainsi que les temps de séjour moyens pour les deux politiques.

3- Justifiez votre choix en tenant compte des caractéristiques du système.

Exercice 4 :

L'ordonnancement EDF (Earliest Deadline First) est un algorithme d'ordonnancement temps réel de processus. A chaque fois qu'un processus demande du temps CPU, il doit préciser une échéance (une date limite >0). Le processus doit avoir obtenu son temps CPU avant d'atteindre la date limite. L'ordonnanceur vise à satisfaire les demandes avant leurs échéances. Pour se faire, il gère une liste des processus prêts, classés par ordre croissant des échéances. L'algorithme exécute le premier processus de la liste qui correspond à celui dont l'échéance est la plus proche. Lorsqu'un processus devient prêt ou arrive dans le système, le système vérifie si son échéance est antérieure (strictement inférieure) à celle du processus en cours d'exécution. Si tel est le cas, le nouveau processus préempte le processus en cours.

a) Considérez trois processus A,B et C suivants :

• A demande le CPU toutes les 30ms et requiert à chaque fois 10 ms de temps CPU.

• B demande du temps CPU toutes les 40ms et requiert à chaque fois 15 ms.

• C demande du temps CPU toutes les 50ms et a besoin à chaque fois 5 ms de temps CPU.

Supposez qu'au départ, les processus sont prêts et que l'échéance de chaque demande est la date de la prochaine demande.

Donnez le diagramme de Gantt pour les 100 premières ms.

b) Peut-on avoir des cas de non respect d'échéances ? Justifiez votre réponse.

Exercice 5 :

1.On considère une implémentation de l'extension temps-réel de POSIX sur un système d'exploitation avec 30 niveaux de priorité (de 0 à 29, 29 étant la plus forte priorité). Par conséquent, le système offre trois politiques d'ordonnancement: SCHED_FIFO (pour les processus FIFO temps-réel), SCHED_RR (pour les processus tourniquet temps-réel) et SCHED_OTHER (pour les processus temps partagé). Le quantum d'exécution pour la politique SCHED_RR est d'une unité de temps. L'ordonnancement est préemptif.

Soit le jeu de processus présenté dans la table 1 suivante :

Processus	Politique	Pattern d'arrivée	Période	Instant d'arrivée	Temps d'exécution	Priorité
Fifo1	SCHED_FIFO	Périodique	8	0	1	28
Fifo2	SCHED_FIFO	Périodique	7	0	2	26
Fifo3	SCHED_FIFO	Apériodique	---	5	4	20
Fifo4	SCHED_FIFO	Apériodique	---	4	3	20
rr1	SCHED_RR	Apériodique	---	3	3	10
rr2	SCHED_RR	Apériodique	---	4	3	10
Other	SCHED_OTHER	Apériodique	---	0	1	0

Un processus périodique est un processus qui est activé régulièrement, aux temps 0, T, 2T, 3T, ….., où T est la période du processus.

Donnez le diagramme de Gantt pour l'ordonnancement des processus de la table 1 entre les instants 0 et 26.

2. Considérons un système d'exploitation doté d'un ordonnanceur préemptif, à priorité. Dans le système, il existe 20 niveaux de priorité, de 0 à 19, 0 étant la plus faible priorité. Supposons que le processeur exécute le jeu de processus présenté dans la table 2 suivante :

Processus	Instant d'arrivée	Séquence d'exécution	Priorité
A	2	EEQE	10
B	5	EE	7
C	0	EQQQQE	3

Les processus A et C partagent la ressource Q, qu'ils accèdent en exclusion mutuelle.

Dans la séquence d'exécution d'un processus, Q signifie que pour une unité de temps le processus s'exécute et accède à la ressource Q, tandis que E signifie que pour une unité de temps le processus s'exécute sans accéder à la ressource Q.

a) Donnez le diagramme de Gantt pour l'ordonnancement des processus de la table 2 entre les instants 0 et 11. Sur le diagramme, indiquez les accès des processus à la ressource Q.

b) Y a-t-il une inversion de priorité sur le diagramme? Si oui, **précisez** l'instant de début, ainsi que la durée de l'inversion de priorité.

Exercice 6 :

RMS est un algorithme d'ordonnancement temps-réel des processus périodiques. Un processus périodique (P_i) est caractérisé par sa période d'activation (T_i), son temps d'exécution (C_i) et sa date d'arrivée dans le système (S_i). Chaque processus possède une contrainte temporelle : il doit s'exécuter complètement avant que sa période arrive de nouveau. Le principe de l'algorithme RMS consiste à associer à chaque processus une priorité fixe, inversement proportionnelle à sa période. Lorsqu'il est appelé, l'ordonnanceur du système choisit le processus de plus forte priorité et le lance en exécution (ordonnacement à priorités).

Soit trois processus A, B et C définis par les paramètres suivants :

$S_A = S_B = S_C = 0$

$T_A = 29$ ms, $C_A = 7$

$T_B = 5$ ms, $C_B = 1$

$T_C = 10$ ms, $C_C = 2$

a) **Indiquez les priorités** des processus A, B et C selon RMS.

b) **Donnez le diagramme de Gantt** pour les 30 premières ms d'ordonnancement RMS, si l'ordonnanceur est préemptif.

c) **Donnez le diagramme de Gantt** pour les 30 premières ms d'ordonnancement RMS, si l'ordonnanceur est non préemptif.

d) **Est-ce que tous les processus respectent leurs contraintes temporelles** sur les 30 premières ms dans le cas b)? Et dans le cas c)? Sinon, justifiez.

Exercice 7 :

L'ordonnancement EDF (Earliest Deadline First) est un algorithme d'ordonnancement temps réel de processus. A chaque fois qu'un processus demande du temps CPU, il doit préciser une échéance (une date limite >0). Le processus doit avoir obtenu son temps CPU avant d'atteindre la date limite. L'ordonnanceur vise à satisfaire les demandes avant leurs échéances. Pour se faire, il gère une liste des processus prêts, classés par ordre croissant des échéances. L'algorithme exécute le premier processus de la liste qui correspond à celui dont l'échéance est la plus proche.

Si deux processus ont la même échéance, l'ordonnanceur sélectionne le plus prioritaire. Lorsqu'un processus devient prêt ou arrive dans le système, le système vérifie si son échéance est antérieure ou égale à celle du processus en cours d'exécution. Le processus en cours est préempté pour l'une des deux raisons suivantes :

- L'échéance du processus en cours est plus loin que celle d'un processus prêt.
- L'échéance du processus en cours est la même que celle d'un processus prêt plus prioritaire.

Considérez trois processus A, B et C suivants :

- A demande le CPU toutes les 20ms et requiert à chaque fois 4 ms de temps CPU.
- B demande du temps CPU toutes les 30ms et requiert à chaque fois 10 ms.
- C demande du temps CPU toutes les 40ms et a besoin à chaque fois 20 ms de temps CPU.
- A est supposé plus prioritaire que B qui est, à son tour, plus prioritaire que C.

Supposez qu'au départ, les processus sont prêts et que l'échéance de chaque demande est la date de la prochaine demande. Par exemple, pour A, les dates d'échéances sont successivement 20, 2*20, 3*20, ….

Donnez le diagramme de Gantt pour les 90 premières ms.

Exercice 8 :

Considérons n processus P_1, P_2, …, P_n, arrivés en même temps et insérés dans cette ordre dans la file des processus prêts. Ces processus ne font pas d'E/S et leurs temps d'exécution sont respectivement c_1, … et c_n. Le temps de commutation est supposé nul.

1) Quel est le temps d'attente moyen des n processus dans chacun des cas suivants :

- D'un ordonnanceur circulaire avec un quantum qt.
- D'un ordonnanceur sans préemption fonctionnant selon la discipline premier arrivé, premier servi.

Dans quel cas, obtient-on un meilleur temps d'attente moyen ?

2) Supposons que le nombre de processus est 5 et que leurs temps d'exécution sont égaux à : 2*qt + r avec r<qt.

- Montrez comment les processus vont utiliser le processeur dans le cas d'un ordonnanceur circulaire avec un quantum qt. Calculer le temps moyen de séjour des processus.
- Quel serait le temps moyen de séjour des 5 processus dans le cas d'un ordonnanceur sans préemption fonctionnant selon la discipline premier arrivé, premier servi.

Dans quel cas, obtient-on un meilleur temps de séjour moyen ?

Exercice 9 :

On considère 4 processus, A, B, C, D. On suppose que l'exécution des processus nécessite :

- Pour A : 7 unités de temps CPU, 3 unités de temps d'E/S et 5 unités de temps CPU.
- Pour B : 6 unités de temps CPU, 4 unités de temps d'E/S, 4 unités de temps CPU.
- Pour C : 5 unités de temps CPU.
- Pour D : 1 unité de temps CPU, 4 unités de temps d'E/S et 2 unités de temps CPU.

On suppose que :

- A se présente en premier, à l'instant 0,
- B se présente à l'instant 1,
- C se présente à l'instant 9,
- D se présente à l'instant 12.

Montrez comment les 4 processus vont utiliser le processeur dans chacun des cas suivants:

1) Chaque processus a son propre périphérique d'E/S et l'ordonnanceur fonctionne selon Premier Arrivée Premier Servi PAPS (sans préemption).

2) Chaque processus a son propre périphérique d'E/S et l'ordonnanceur utilise l'algorithme du tourniquet, avec un quantum de 5. Le temps de commutation est égal à 0. Donnez, dans ce cas, les temps de séjour des processus A, B, C et D.

3) Les trois processus utilisent le même périphérique d'E/S dont la file d'attente est gérée premier arrivée premier servi. L'ordonnanceur du processeur utilise l'algorithme du tourniquet, avec un quantum de 5.

Le temps de commutation est supposé égal à 0.

Exercice 10 :

1. Citez quatre événements qui provoquent l'interruption de l'exécution d'un processus en cours, dans le système Unix.
2. Quel est le rôle de l'ordonnanceur ?
3. Décrire brièvement l'ordonnanceur du système Unix. Favorise-t-il les processus interactifs ?
4. Expliquez une raison pour laquelle certains systèmes (tel Unix) assignent des priorités internes plutôt qu'externes aux processus.
5. Identifiez l'algorithme d'ordonnancement qui est le plus utilisé, et spécifiez la raison.
6. Identifiez l'algorithme d'ordonnancement sans réquisition qui fournitla meilleure performance. Pourquoi ?
7. Expliquez la différence fondamentale entre un algorithme d'ordonnancement à réquisition et sans réquisition. Lequel fournit la meilleure performance ?
8. Quel sera l'effet sur la performance du système d'un temps de quantum trop long et trop petit.
9. Expliquez la différence fondamentale existant entre un ordonnanceur de bas-niveau et un ordonnanceur de haut-niveau. Existe-il des systèmes d'exploitation qui utilisent un seul ordonnanceur ? Expliquez aussi.
10. Expliquez quel mécanisme permet à un processus d'effectuer un partage volontaire de l'UCT. Quels sont les avantages et désavantages du partage volontaire ?
11. Expliquez comment le système détermine le moment d'effectuer un changement de contexte entre deux processus. Durant un changement de contexte, quels sont les processus (utilisateur ou système) qui sont chargés et déchargés ?
12. Considérez la charge de processus montré sur la table suivante. Pour chacun des algorithmes d'ordonnancement spécifiés ci-bas, dessinez un diagramme montrant l'état d'occupation de l'UCT, des périphériques, et des files d'attente. Aussi, calculez les temps de virement et d'attente. Supposez un temps de changement de contexte instantané (infiniment petit). Finalement, comparez les performances des divers algorithmes pour cette charge de processus.

a) L'algorithme du premier-arrivé-premier-servi (PAPS).

b)

Processus	**Temps d'arrivée**	***Temps de service***	**Durée des E/S**	**Périodes des E/S**
P0	0	9	2	5,9
P1	2	6	0	-
P2	5	5	1	4

TAB. - Charge de processus.

b) L'algorithme du tourniquet avec un quantum de 5 unités de temps.

c) L'algorithme du tourniquet avec un quantum de 5 unités de temps et des niveaux de priorité externe de 3, 1, 2 respectivement pour P0, P1, P2.

d) L'algorithme du tourniquet avec un quantum de 5 unités de temps et deux UCTs fonctionnant en parallèle.

e) L'algorithme du tourniquet avec un quantum de 5 unités de temps et des niveaux de priorité interne initiaux de 20 respectivement pour P0, P1, P2. Au moment d'un changement de contexte, la priorité du processus débarquant diminue de 1 unité. Aussi, un processus qui est bloqué dans une file d'attente des E/S perd 2 unités de priorité. Si le processus n'attend pas pour effectuer son E/S alors il ne perd que 1 unité de priorité. Lorsqu'un processus dans la file d'attente de l'UCT a plus haute priorité qu'un processus en exécution, l'UCT est alors préempté avant la fin du quantum.