Introduction

Tout document autorisé.
L'examen comporte **deux parties** à rédiger sur des copies séparées.
Les réponses doivent être **claires et concises** : cela sera pris en compte lors de l'évaluation des réponses.
La durée de l'examen est 120 minutes.

Partie I (12 pts) —— Copie I

Exercice 1 (3 pts)

1. Supposons que deux processus longs, P1 et P2, s'exécutent dans un système. Aucun programme n'effectue d'appels système susceptibles de provoquer son blocage, et il n'y a pas d'autres processus dans le système. P1 a deux threads et P2 a un thread.
   a) Quel pourcentage de temps CPU recevra P1 si les threads sont des threads systèmes? Expliquez (**0,5 pt**).
   b) Quel pourcentage de temps CPU sera obtenu par P1 si les threads sont des threads utilisateurs ? Expliquez (**0,5 pt**).

Soient les deux threads systèmes suivants appartenant au processus P1.

Code du Thread1	Code du Thread2

void
Thread1Fonction() {
char buf;
for(i = 0;i < 5;i++) {
LireEntree(&buf);
EcrireSortie(&buf);
EcrireSortie("X"); }
Exit(0);
}
	void
	Thread2Fonction() {
	char buf;
	for(i = 0;i < 5;i++) {
	LireEntree(&buf);
	EcrireSortie(&buf);
	EcrireSortie("Y"); }
	Exit(0);
	}

Thread1 exécute la fonction Thread1fonction et Thread2 exécute la fonction Thread2fonction. Dans ces fonctions, *LireEntree*, *EcrireSorie*, et *Exit* sont des appels systèmes. L'appel *LireEntree* lit un caractère du périphérique d'entrée dans le buffer spécifié. C'est un appel bloquant, l'appelant se bloquera pendant une courte période pendant qu'un caractère est obtenu. L'appel *EcrireSortie* écrit un caractère du buffer spécifié dans le périphérique de sortie. C'est un appel non bloquant. L'appel système *Exit* met fin au processus appelant.

Supposons que la séquence de caractères suivante est tapée sur la console d'entrée: a b c d e f g h i j

2. Donnez la séquence de caractères qui sera générée sur la console de sortie par ce processus (**2 pts**).

Exercice 2 (4 pts)

On considère un système avec une mémoire logique segmentée paginée où la taille d'une page est de 4 ko (1 mot mémoire = 1 octet) et une mémoire physique de 64 ko. L'espace d'adressage d'un processus P est composé de trois segments S1, S2 et S3 de taille, respectivement 16 ko, 8 ko et 4 ko. À un moment donné, pour le processus P, les pages 2 et 3 du segment S1, la page 2 du segment S2 et la page 1 du segment S3 sont chargées en mémoire physique, respectivement dans les cases 2, 0, 9, 12.
Pour une donnée située dans l'espace d'adressage du processus P à l'adresse décimale 8212, indiquez en justifiant à chaque fois :

a) Le segment (**0.5 pt**)
b) Le numéro de page dans le segment (**0.5 pt**)
c) Le déplacement dans la page (**0.5 pt**)
d) Le numéro de case (**0.5 pt**)
e) Le déplacement dans la case (**0.5 pt**)
f) L'adresse physique (en décimal et en binaire) (**1.5 pt**)

Exercice 3 (5 pts)

Soient 3 processus qui exécutent les programmes suivants (où les Ai sont des actions):

P1	P2	P3
répéter A1 ; jusqu'à faux	répéter A2 ; jusqu'à faux	répéter A3 ; jusqu'à faux

Dans chacun des cas suivants, réécrire le code de P1, P2 et P3 en utilisant des sémaphores pour synchroniser ces 3 processus de telle manière que

a) Les actions Ai ne soient jamais simultanées (**0.5 pt**)
b) Les actions Ai ne soient jamais simultanées et se déroulent toujours dans l'ordre A1A2A3A1A2A3... (**1 pt**)
c) Les actions Ai ne soient jamais simultanées et se déroulent toujours dans l'ordre A1(A2 ou A3)A1(A2 ou A3)... (**1,5 pts**)
d) Les actions Ai se déroulent toujours séquentiellement mais dans un ordre quelconque, par exemple : (A2A1A3)(A2A3A1)... (**2 pts**)

Partie II (8 pts) —— Copie II

Exercice 1 (4 pts) *Ordonnancement de processus*

1. Définissez le temps de traitement moyen. (**0.5 pt**)
2. Donnez une situation où la stratégie d'ordonnancement avec réquisition ne sera pas adaptée, justifiez. (**0.5 pt**)

Un ordonnanceur utilise 3 files d'attente, la file n° 3 étant hiérarchiquement la plus élevée. Les processus ont un numéro de priorité entre 1 et 3, et ils entrent directement dans la file d'attente correspondant à leur numéro. La file dans laquelle se trouvera un processus sera donc la même jusqu'à la fin de son exécution. Chaque file est gérée par un algorithme d'ordonnancement. Cet algorithme n'est activé que si les files des niveaux supérieurs sont toutes vides et que la file à laquelle il s'applique n'est pas elle-même vide. De plus, l'ordonnanceur ne pourra basculer d'une file de niveau inférieur à une file de niveau supérieur qu'à la fin du temps CPU alloué au processus en cours d'exécution. Les temps sont donnés en unité de temps.

On considère le système de tâches défini par le tableau suivant :

Processus	Durée	Arrivée	Priorité
P1	7	0	2
P2	4	0	3
P3	6	1	1
P4	1	1	2
P5	2	0.5	3
P6	4	2	1
P7	1	2	2
P8	4	8.5	3

3. Calculez le temps de traitement moyen sachant que les algorithmes d'ordonnancement SRT (Shortest Remaining Time) avec un quantum=1, Tourniquet : RR (Round Robin) avec un quantum=2 et FCFS (First Come Fisrt Served) s'appliquent dans les files 3, 2 et 1 respectivement. Vous illustrerez les calculs avec des dessins et diagrammes. (**3 pts**)

Exercice 2 (4 pts) *Inter-blocage*

Dans un système, on considère la description simplifiée de la gestion des processus suivante. La création d'un processus se fait en deux phases : Le processus est d'abord créé dans une mémoire swap dans la première phase, ensuite, dans la deuxième phase le processus est déplacé en mémoire centrale où il s'exécutera. Au cours de ces deux phases, le processus occupe successivement des places (espaces) de même taille en mémoire swap et en mémoire centrale. Lorsque le processus se trouve en mémoire centrale, l'espace occupé dans la mémoire swap est libéré. Il est possible d'avoir simultanément plusieurs processus en mémoire centrale. Un processus est transféré de la mémoire centrale vers la mémoire swap si : i) il est dans un état bloqué en mémoire centrale en attente d'un évènement d'un processus en mémoire swap, ii) sa durée d'exécution est 4s alors que au

moins un processus prêt en mémoire swap attend depuis au moins 3s. Lorsqu'un processus est transféré en mémoire swap, l'espace qu'il occupait en mémoire centrale est libéré. Enfin, lors d'un *fork()*, le processus père est recopié en mémoire swap pour la création de son fils. On supposera que la taille des mémoires centrale et de réserve est fixe.

a) Identifiez deux situations d'interblocage qui peuvent se produire. (**1 pt**)
b) Proposez des solutions pour prévenir ou éviter ces problèmes. (**1 pt**)

L'état courant d'allocation des ressources d'un système est donné par la figure ci-dessous :

	ALLOCATION					MAX					DISPONIBLE			
	R0	R1	R2	R3		R0	R1	R2	R3		R0	R1	R2	R3
P0	0	0	1	2	P0	0	0	1	2		1	5	2	0
P1	1	0	0	0	P1	1	7	5	0					
P2	1	3	5	4	P2	2	3	5	6					
P3	0	6	3	2	P3	0	6	5	2					
P4	0	0	1	4	P4	1	6	5	6					

1. Vous illustrerez toutes vos réponses avec l'exécution des algorithmes du Banquier
   a) Cet état est-il sain ? (**0,5 pt**)
   b) Si le processus P1 fait la demande (0, 5, 2, 0), le système lui attribuera-t-il la ressource ? (**1,5 pts**)