Licence 2 — Info4B
Examen du Vendredi 9 Juin 2023
Durée 2h • Documents autorisés : une feuille A4 recto-verso
Le barème est donné à titre indicatif

Si un exercice vous conduit à faire des hypothèses, indiquez-les clairement sur votre copie.
Rédigez et justifiez précisément les réponses aux questions.

Exercice 1 - 6 pts

1. Expliquer comment la méthode `sleep()` du langage Java agit sur l'ordonnancement. Montrer sa différence avec les primitives `wait()` et `notify()`.
2. Définir la notion de *socket* et montrer comment elle permet à différents processus situés sur des machines distinctes de dialoguer entre eux. Décrire les différents types de *socket* et leur utilité.
3. Quel est l'intérêt d'utiliser des tables de hachage pour un noyau de système d'exploitation ?

Exercice 2 - 6 pts

On considère un tourniquet à 3 files d'attentes qui ont des quantum respectifs de 2, 3 et 5 unités de temps (ut). Le processeur possède un seul cœur. Les files sont gérées en mode plus court d'abord (en fonction du temps d'exécution restant). Les processus d'une file n ont accès au processeur si les files 1 à $n-1$ sont vides. Il y a réquisition lorsqu'un processus arrive dans la file 1 vide et que le processus en cours d'exécution provient de la file 3 et a une durée restante d'exécution supérieure au quantum de temps restant dans le processeur.

Nom	p0	p1	p2	p3	p4	p5
Top d'arrivée	1	2	5	15	22	28
Durée (ut)	10	11	1	4	3	4

1. À chaque fois qu'un processus est remis dans une file d'attente, il descend d'un niveau. Effectuer la simulation avec les données du tableau ci-dessus.
2. Calculer pour cette simulation le taux de retard pour chaque processus et le taux de retard moyen. Avec les données calculées et la simulation, que pouvez-vous observer ?

Exercice 3 - 8 pts

On se place dans une application producteurs/consommateurs, dans laquelle une file sert à stocker les messages des producteurs pour les mettre à disposition des consommateurs.

1. Écrire une classe `MessageGenerateur` qui est un thread générant des messages sous la forme d'objets sérialisables et les dépose dans une file d'attente de capacité maximum fixée. La durée d'attente entre deux messages est aléatoire (intervalle de 5 à 10 secondes).
2. Écrire une classe `FileMessages` qui réceptionne les messages et les met à disposition des consommateurs. Les threads `MessageGenerateur` doivent être mis en attente s'ils essaient d'envoyer un message alors que la file a atteint sa capacité maximum.
3. Donner un extrait de la méthode `main` pour instancier une file, et 4 threads `MessageGenerateur` puis les démarrer.
4. On souhaite mettre à jour l'application. Les threads et la file sont désormais sur des machines séparées. Faire un schéma du système, en faisant apparaître les différentes machines, les instances de classes qu'elles hébergent et les interactions ainsi que les modes de communication.
5. Modifier le code de la classe `MessageGenerateur` pour permettre l'envoi des messages par le réseau. Écrire également les modifications à apporter pour permettre à la classe `FileMessages` de réceptionner ces messages.

Numéro d'anonymat :

Question numéro :

1		13		25	
2		14		26	
3		15		27	
4		16		28	
5		17		29	
6		18		30	
7		19		31	
8		20		32	
9		21		33	
10		22		34	
11		23		35	
12		24			