NOUVELLE CALÉDONIE – 2023 – sujet2

Exercice 3 (4 points)

1) Pour la réservation de billets à un concert, deux guichets utilisent une même variable compteur comptabilisant le nombre de places restantes. Celle-ci est stockée dans une mémoire partagée par les deux guichets. À chaque fois qu'un billet est délivré, la variable compteur diminue de 1.

Deux processus P1 et P2 associés à chacun des guichets accèdent en concurrence à la variable compteur.

Le déroulement de l'exécution des processus s'organise comme ci-dessous (du haut vers le bas). Des variables compteur1 et compteur2 sont spécifiques respectivement aux processus P1 et P2.

	Processus P1	**Processus P2**
Intervalle de temps 1	compteur1 ← compteur	
Intervalle de temps 2		compteur2 ← compteur compteur2 ← compteur2 – 1 compteur ← compteur2
Intervalle de temps 3	compteur1 ← compteur1 – 1 compteur ← compteur1	

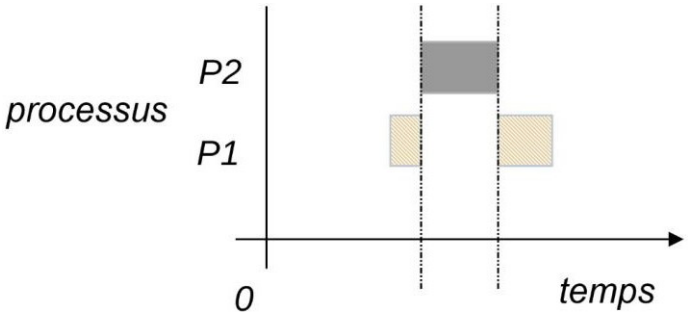

Figure 1

La valeur initiale de la variable compteur est 100 (en base 10).

a. **Montrer** que la valeur de la variable compteur après l'intervalle de temps 3 est égale à 99.
b. **Modifier** l'ordonnancement des processus pour, qu'à la fin, la valeur de la variable compteur soit égale à 98.

2) Une autre solution est envisagée.

La mémoire partagée est alors gérée par un jeton (appelé aussi sémaphore ou drapeau) de type exclusion mutuelle qui autorise ou non son accès. Ce jeton sera nommé mutex.

La classe Jeton contient les attributs et les méthodes suivants. Le jeton mutex est donc créé comme une instance de la classe Jeton.

Nom de la classe	Jeton
Attributs	id // identifiant état // jeton : libre ou occupé ...

Méthodes	verrouillage() // prise du jeton dès qu'il est libre
	déverrouillage() // libération du jeton
	...

L'algorithme des processus est donné ci-dessous (du haut vers le bas).

Processus P1	**Processus P2**

`mutex.verrouillage()`
`compteur1 ← compteur`
`compteur1 ← compteur1 − 1`
`compteur ← compteur1`
`mutex.deverrouillage ()`
	`mutex.verrouillage()`
	`compteur2 ← compteur`
	`compteur2 ← compteur2 − 1`
	`compteur ← compteur2`
	`mutex.deverrouillage()`

a. **Justifier** que cette nouvelle solution permet à la variable compteur de comptabiliser correctement le nombre de places restantes.
b. **Proposer** un algorithme identique pour les processus P1 et P2 produisant le même résultat.

3) Pour l'organisation du concert, l'administrateur crée des guichets avec des priorités d'attribution. Quatre processus P1, P2, P3, P4 sont associés aux quatre guichets et exécutent chacun l'algorithme de la question 2 b. L'ordonnancement choisi est un ordonnancement avec priorité, sans un nouveau calcul de la priorité.

L'allocation du processeur par tranche (quantum) de temps est de 15 ms. Le quantum est insécable (c'est-à-dire qui ne peut pas être découpé).

Le détail des processus est donné ci-dessous avec des priorités allant de 0, définissant une priorité haute à 20, définissant une priorité basse.

processus	**priorité**	**Date arrivée sur le processeur**	**Durée service**
P1	5	30 ms	30 ms
P2	5	15 ms	30 ms
P3	8	15 ms	30 ms
P4	20	15 ms	30 ms

Compléter le diagramme **sur le document réponse**, à l'image de la figure 1 de la question 1.

DOCUMENT RÉPONSE À RENDRE OBLIGATOIREMENT ET DANS SON INTÉGRALITÉ AVEC LA COPIE

EXERCICE 3. QUESTION 3)

Compléter le diagramme ci-dessous.

↑ indique la date d'arrivée des processus P1 à P4.