Nouvelle Calédonie – 2023 – sujet2 - Correction

Exercice 3 (4 points)

1.

a. Montrer que la valeur de la variable *compteur* après l'intervalle de temps 3 est égale à 99.

Valeur initiale : `compteur = 100`

- **Intervalle 1**
  P1 :
  `compteur1 ← compteur`
  → `compteur1 = 100`

- **Intervalle 2**
  P2 :
  `compteur2 ← compteur` → 100
  `compteur2 ← compteur2 - 1` → 99
  `compteur ← compteur2` → 99

- **Intervalle 3**
  P1 reprend :
  `compteur1 ← compteur1 - 1` → 99
  `compteur ← compteur1` → 99

La valeur finale de `compteur` est donc **99**.

👉 **Commentaire :** *Une décrémentation est perdue car P1 travaille sur une copie locale obsolète de la variable partagée. Il s'agit d'un problème de concurrence (condition de compétition).*

---

b. Modifier l'ordonnancement des processus pour qu'à la fin la valeur de la variable *compteur* soit égale à 98.

Proposition d'un nouvel ordonnancement

On fait exécuter complètement P1 avant P2.

- P1 :
  - `compteur1 ← compteur` → 100
  - `compteur1 ← compteur1 - 1` → 99
  - `compteur ← compteur1` → 99

- Puis P2 :
  - `compteur2 ← compteur` → 99
  - `compteur2 ← compteur2 - 1` → 98
  - `compteur ← compteur2` → 98

Valeur finale : **98**

👉 **Commentaire :** *L'exécution séquentielle garantit que chaque processus lit une valeur mise à jour.*

2.

a. Justifier que la solution avec mutex permet de comptabiliser correctement le nombre de places restantes.

Avec le mutex :

```
mutex.verrouillage()
compteur_local ← compteur
compteur_local ← compteur_local - 1
compteur ← compteur_local
mutex.deverrouillage()
```

Le verrouillage empêche qu'un autre processus accède à la variable pendant la modification.

👉 **Commentaire :** *Le mutex assure l'exclusion mutuelle : la section critique est protégée, ce qui supprime toute interférence entre processus.*

---

b. Proposer un algorithme identique pour les processus P1 et P2 produisant le même résultat.

Algorithme commun :

```
mutex.verrouillage()
temp ← compteur
temp ← temp – 1
compteur ← temp
mutex.deverrouillage()
```

Les deux processus exécutent exactement le même code.

👉 **Commentaire :** *Un algorithme unique simplifie la conception et garantit un comportement cohérent.*

---

3. Compléter le diagramme d'ordonnancement avec priorité (quantum 15 ms, priorité fixe).

Données :

Processus	Priorité	Arrivée	Durée
P1	5	30 ms	30 ms
P2	5	15 ms	30 ms
P3	8	15 ms	30 ms
P4	20	15 ms	30 ms

Rappel :

- Plus le nombre est petit, plus la priorité est élevée.
- Quantum = 15 ms, non interrompable.

Ordonnancement :

- 15–30 ms : P2
- 30–45 ms : P2 (termine)
- 45–60 ms : P1
- 60–75 ms : P1 (termine)
- 75–90 ms : P3
- 90–105 ms : P3 (termine)
- 105–120 ms : P4

- 120–135 ms : P4 (termine)

Ordre :
P2 → P2 → P1 → P1 → P3 → P3 → P4 → P4

👉 **Commentaire :** *L'ordonnancement à priorité fixe sélectionne toujours le processus disponible de priorité la plus élevée. Le quantum insécable empêche toute préemption avant la fin des 15 ms.*
