| | |
|---|---|
| **Commencé le** | mercredi 28 février 2024, 17:06 |
| **État** | Terminé |
| **Terminé le** | mercredi 28 février 2024, 17:07 |
| **Temps mis** | 57 s |
| **Note** | **10,00** sur 10,00 (**100**%) |

**Question 1**

Correct

Note de 1,00 sur 1,00

Une condition de concurrence se produit lorsque plusieurs processus/threads modifient en même temps une donnée/un objet partagé. Elle peut ainsi causer des résultats incohérents qui varient d'une exécution à une autre.

Veuillez choisir une réponse.

- Vrai ✔
- Faux

La réponse correcte est « Vrai ».

**Question 2**

Correct

Note de 1,00 sur 1,00

Un sémaphore, au sens de E. W. Dijkstra (1965), est un compteur entier :

1 - qui peut être décrémenté, si sa valeur est strictement positive,

2 - qui bloque tout processus/thread qui essaye de le décrémenter alors que sa valeur n'est pas strictement positive,

3 - qui peut être incrémenté, si aucun processus/thread n'est bloqué par le sémaphore, et

4 - qui débloque un processus/thread bloqué par le sémaphore, lorsqu'un autre processus/thread tente d'incrémenter la valeur du sémaphore et il y a au moins un processus/thread bloqué par le sémaphore.

Veuillez choisir une réponse.

- Vrai ✔
- Faux

La réponse correcte est « Vrai ».

**Question 3**

Correct

Note de 1,00 sur 1,00

Complétez la phrase suivante :

Un processus/thread passe à l'état bloqué lorsqu'il

Veuillez choisir une réponse.

- a. crée un sémaphore.
- b. tente de décrémenter un sémaphore dont la valeur n'est pas strictement positive. ✔
- c. tente d'incrémenter la valeur d'un sémaphore.
- d. tente de décrémenter un sémaphore dont la valeur est strictement positive.

Votre réponse est correcte.

La réponse correcte est :
tente de décrémenter un sémaphore dont la valeur n'est pas strictement positive.

**Question 4**

Correct

Note de 1,00 sur 1,00

Un sémaphore fort peut être implémenté par une structure de données comportant la valeur courante du sémaphore, la file d'attente du sémaphore et un verrou. Ce dernier sert à assurer des accès en exclusion mutuelle aux attributs du sémaphore.

Veuillez choisir une réponse.

- Vrai ✔
- Faux

La réponse correcte est « Vrai ».

**Question 5**

Correct

Note de 1,00 sur 1,00

Par défaut, la file d'attente d'un sémaphore est gérée selon la discipline FIFO (First IN First Out). Est-ce que la gestion selon la discipline LIFO (Last In First Out) de cette file d'attente pourrait causer une famine ?

Veuillez choisir une réponse.

a. Non

b. Oui ✔

Votre réponse est correcte.

La réponse correcte est :
Oui

**Question 6**

Correct

Note de 1,00 sur 1,00

Le pseudo-code suivant montre comment utiliser l'instruction atomique Test and Set Lock (TSL) dans une attente active pour exécuter en exclusion mutuelle les sections critiques des processus P1 et P2 :

```
int lock = 0;

Processus P1
while (1)
{    ....
  while(TSL(lock)!=0);
   section_critique_P1() ;
   lock=0; ....
}

Processus P2
while (1)
{  .....
  while(TSL(lock)!=0);
   section_critique_P2() ;
   lock=0; ....
}

int TSL(int &lock)
{
   int r = lock;
   lock =  1;
   return r;
}
```

Votre coéquipier vous propose de remplacer "while(TSL(lock(!=0));" par "while(lock==1 || TSL(lock)!=0);". Est-ce que cette variante du pseudo-code permet d'exécuter toujours en exclusion mutuelle les sections critiques des processus P1 et P2 ?

Veuillez choisir une réponse.

a. Oui ✔

b. Non

Votre réponse est correcte.

La réponse correcte est :
Oui

**Question 7**

Correct

Note de 1,50 sur 1,50

Considérez les processus A et B suivants :

Sémaphore x=1;

| A | B |
|---|---|
| P(x); | P(x); |
| a; | b; |
| c; | d; |
| V(x); | V(x); |

Sélectionnez les ordres possibles d'exécution des actions atomiques a, b, c et d, sachant que ces deux processus s'exécutent en concurrence.

Veuillez choisir au moins une réponse.

- [x] a. a;c;b;d; ✔
- [x] b. b;d;a;c; ✔
- [ ] c. a;b;c;d;
- [ ] d. b;a;c;d;

Votre réponse est correcte.

Les réponses correctes sont :

a;c;b;d;,

b;d;a;c;

**Question 8**

Correct

Note de 1,50 sur 1,50

Considérez les processus A, B et C suivants :

Sémaphore x = 0;

| A | B | C |
|---|---|---|
| a; | P(x); | P(x); |
| V(x); | b1; | c1; |
| | b2; | c2; |
| | V(x); | V(x); |

Sélectionnez les différents ordres possibles d'exécution des actions atomiques a1, b1, b2, c1 et c2, sachant que ces trois processus s'exécutent en concurrence.

Veuillez choisir au moins une réponse.

- [x] a. a; b1; b2; c1; c2; ✔
- [ ] b. a; c1; b1; b2; c2;
- [x] c. a; c1; c2; b1; b2; ✔
- [ ] d. c1; c2; b1; b2; a;

Votre réponse est correcte.

Les réponses correctes sont :

a; b1; b2; c1; c2;,

a; c1; c2; b1; b2;

**Question 9**

Correct

Note de 1,00 sur 1,00

Considérez la solution au problème des producteurs et consommateurs vue en classe. La permutation (en rouge) :

```
int tampon [N];
int ip=0,ic=0;
Semaphore libre=N, occupe=0, mutex=1;
Producteur ()
{   while(1)
    {     P(libre);
          P(mutex);
          produire(tampon, ip);
          ip = Modulo (ip +1,N);
          V(mutex);
           V(occupe);
    }
}
```

```
Consommateur ()
 {
   while(1)
   {   P(mutex);
       P(occupe) ;
       consommer(tampon,ic);
        ic= Modulo (ic+1, N);
        V(mutex);
        V(libre);
    }
}
```

Veuillez choisir une réponse.

a. n'affecte pas le comportement des producteurs et des consommateurs.

b. peut mener vers un interblocage. ✔

c. va imposer une alternance entre les productions et les consommations.

d. va forcément mener vers un interblocage.

Votre réponse est correcte.

Cette permutation peut mener vers un unterblocage dans le cas où, par exemple, dès le départ, c'est un consommateur qui obtient le sémaphore mutex (P(mutex)) avant de se retrouver en attente du sémaphore occupe (P(occupe)). Les producteurs et les autres consommateurs finiront par se retrouver en attente du sémaphore mutex ou libre.

Notez cependant, qu'il est possible que plus exécutions de votre code ne mènent pas vers un interblocage.

La réponse correcte est :
peut mener vers un interblocage.