Question **3**

Correct

Note de 1,50 sur 1,50

Considérez la variante suivante du moniteur ProducteursConsommateurs vu en classe, où les deux variables de condition *nplein* et *nvide* sont remplacées par une seule variable de condition *wq* :

```
Moniteur ProducteursConsommateurs{
    const int N=100;
     int tampon[N];
     int compteur=0, ic=0, ip=0;
     boolc wq;
     void deposer (int objet) { // section critique pour le dépôt
              while(compteur==N) wait(wq);
              tampon[ip] = objet;
              ip = (ip+1)%N;
              compteur++;
              signal(wq);
     }
     void retirer (int& objet) { //section critique pour le retrait
                while(compteur==0) wait(wq);
                objet = tampon[ic];
                ic = (ic+1)%N;
                compteur--;
                signal(wq);
      }
}
```

Sélectionnez l'énoncé correct. Utilisez la page suivante pour justifier votre réponse.

- (●) Les produits sont consommés selon l'ordre des productions sans aucune perte ni double consommation d'un produit. ✔
- ( ) Aucunes de ces réponses.
- ( ) Les producteurs pourraient écraser des produits non encore consommés.
- ( ) Les consommateurs pourraient consommer d'un tampon vide.
- ( ) Les producteurs pourraient produire dans un tampon plein.
- ( ) Les consommateurs pourraient consommer plusieurs fois le même produit.

Votre réponse est correcte.

La réponse correcte est :
Les produits sont consommés selon l'ordre des productions sans aucune perte ni double consommation d'un produit.

Commentaire :

Question 4

Terminer

Non noté

Utilisez cette page pour justifier votre réponse à la question précédente.

En effet, ici, les consommateurs et producteurs seront en mesure d'effectuer leur fonctions. L'ajout de la variable même si elle est fixe permet de limiter le temps de la boucle et donc de pas pas bloque dans la boucle à l'infini. Ainsi la condition sera vérifier a chaque fois et cela permet de ne pas modifier le comportement des producteurs et consommateurs

Dans cette variante, les producteurs et les consommateurs qui ne peuvent pas produire ou consommer sont mis en attente dans la file d'attente de la même variable de condition wq. Par conséquent, lorsqu'un producteur (ou un consommateur) exécute la fonction signal, il pourrait aussi bien débloquer un producteur ou un consommateur.

Cependant, grâce, d'une part, à la lecture/modification en exclusion mutuelle des variables partagées et, d'autre part, aux boucles "while (compteur==N) wait(wq);" et "while (compteur==0) wait(wq);", un producteur (ou un consommateur) débloqué précocement ne pourra pas produire dans un tampon plein (ou consommer d'un tampon vide). Aucune double consommation ou perte de production n'est possible, car les opérations de production et de consommation sont exécutées en exclusion mutuelle.

Commentaire :
"le temps de la boucle et donc de pas pas bloque dans la boucle à l'infini" Comment ?

Question **5**

Terminer

Note de 0,50 sur 3,00

Considérez les deux threads concurrents T1 et T2 suivants d'un même processus :

```
/*0*/
T1 {                    T2 {
 while(1) {              while(1) {
  /*1*/                   /*4*/
  a1();                  b1();
  /*2*/                   /*5*/
  a2();                  b2();
  /*3*/                   /*6*/
  }                       }
}                        }
```

Utilisez un moniteur et des variables de conditions pour forcer (1) l'exécution de la fonction a1 avant celle de b2 et (2) l'exécution de b2 avant celle de a2. **Attention : Aucune autre contrainte de précédence ne doit être forcée.**
Pour répondre à cette question, vous devez compléter le moniteur suivant ainsi que le code précédent :

```
Moniteur OrdrePartiel { /*7*/}

Moniteur OrdrePartiel {
semaphore mutex;
bool temp;
T1(){
   while(temp) {
      P(mutex);
      a(1);
      a(2);
      V(mutex)
      temp = !temp;
    }
 }
T2() {
   while(temp) {
      P(mutex);
      b(1);
      b(1);
      b2(mutex);
      temp = !temp;
    }
 }
 }
```

```
// Il y a plusieurs solutions à cette question. La solution suivante s'inspire de celles basées sur les sémaphores.
Moniteur OrdrePartiel {
bool go=false; // go = false si la condition de précédence est non satisfaite
int nbw=0; boolc wq; // pour attendre si la condition de précédence est non satisfaite
waitCond() { if (go==false) {nbw++; wait(wq); } else go=false; } // attendre la condition de précédence si nécessaire
signalCond() { if(nbw>0) {nbw-; signal(wq); } else go=true;}. //signaler la condition de précédence
}

/*0*/ Moniteur OrdrePartiel O1, O2;
T1 {                                    T2 {
while(1) {                              while(1) {
/*1*/                                   /*4*/
a1();                                   b1();
/*2*/ O2.signalCond(); O1.waitCond();   /*5*/O2.waitCond();
a2();                                   b2();
/*3*/                                   /*6*/ O1.signalCond();
}                                       }
}                                       }
```

Commentaire :
Si vous utilisez les sémaphores, vous n'avez plus besoin des moniteurs.

T1 et T2 dans le moniteur ?

Question **6**

Non répondue

Non noté

Utilisez cette page, si nécessaire, pour compléter votre réponse à la question précédente.

Question **7**

Terminer

Note de 0,00 sur 1,00

Dans un système, 3 processus partagent 40 ressources de même type R. Chaque processus a besoin au maximum de 14 ressources de type R.

Les 3 processus peuvent-ils se retrouver en interblocage ? Justifiez votre réponse. Les réponses non ou mal justifiées ne seront pas considérées.

Non, les processus peuvent se retrouver en interblocage.

On sait que trois processus partagent 40 ressources du même type R et que chaque processus a besoin au maximum de 14 ressources de type R.

On supposera alors que ces 3 processus ont besoin de 14 ressources de R (pour faire un test exhaustif). On supposera également que chaque processus n'a aucune ressource R qui lui est alloué. Ainsi le nombre de ressource R requise pour l'exécution de ces 3 processus est de 3*14 = 42 or 42 > 40 ainsi il est possible que le nombre de ressource soit insuffisant ainsi ces processus peuvent se retrouver en interblocage.

Les 3 processus ne peuvent pas se retrouver en situation d'interblocage, car même dans le cas où il manque une seule ressource à chacun des processus (13 *3 = 39 ressources allouées), il reste 1 ressource disponible (40-39=1). L'allocation de cette ressource libre à un processus va lui permettre d'accomplir son exécution. À la fin de l'exécution de ce processus, le système va se retrouver avec 14 ressources libres. Ce qui est amplement suffisant pour satisfaire les besoins en ressources des deux autres processus.

Commentaire :

Question **8**

Correct

Note de 2,00 sur 2,00

Dans un système, 4 processus (P1, P2, P3 et P4) partagent 3 types de ressources (R1,R2 et R3) en quantités respectives (9, 3, 6). Supposez l'état courant du système :

Matrice Alloc des ressources allouées :

Alloc :

	R1	R2	R3
P1	1	0	0
P2	6	1	2
P3	2	1	1
P4	0	0	2

Matrice Req des ressources requises mais non encore acquises :

Req :

	R1	R2	R3
P1	2	3	2
P2	0	0	1
P3	1	0	3
P4	4	2	0

1- Est-ce que cet état est sûr ? Justifiez votre réponse en déroulant pas-à-pas l'algorithme du banquier.

Oui ✔

2- Supposez que le système utilise l'algorithme du banquier pour éviter les interblocages. Le système reçoit de la part du processus P1, une demande d'une ressource de type R2. Le système va-t-il accepter cette demande ? Justifiez votre réponse.

Oui ✔

Utilisez la page suivante pour justifier vos réponses. Les réponses non ou mal justifiées ne seront pas considérées.

Les ressources disponibles :

A = (9, 3, 6) - (1+6+2, 1+1, 2+1+2)= (0, 1, 1)

L'agorithme du banquier

Req(P2) <= A ? Oui ==> A= A+Alloc(P2) = (6, 2, 3)

Req(P4) <= A ? Oui ==>. A= A+Alloc(P4) = (6, 2, 5)

ReqP3) <= A ? Oui ==> A = A+ Alloc(P3) = (8, 3, 6)

Req(P1) <= A ? Oui ==> A = A+ Alloc(P1) = (9, 3, 6)

Oui, l'état est sûr car l'ordre d'exécution P2, P4, P3, P1 permet aux processus de compléter leurs exécutions dans le cas pire cas (tous les processus demandent en même temps les ressources nécessaires à l'accomplissement de leurs exécutions).

2). P1 demande R2. Pour accepter cette demande, il faudrait qu'elle mène vers un état sûr :

A' = (0, 0, 1)

Alloc'

	R1	R2	R3
P1	1	1	0
P2	6	1	2
P3	2	1	1
P4	0	0	2

Req'

	R1	R2	R3
P1	2	2	2
P2	0	0	1
P3	1	0	3
P4	4	2	0

Req'(P1) <= A' ? Non
Req'(P2) <= A' ? Oui ==> A'= A'+Alloc'(P2) = (6, 1, 3)
Req'(P3) <= A' ? Oui ==> A'= A'+Alloc'(P3) = (8, 2, 4)
Req'(P4) <= A' ? Oui ==> A'= A'+Alloc'(P4) = (8, 2, 6)
Req'(P1) <= A' ? Oui ==>. A'= A'+Alloc'(P1) = (9, 3, 6)'

Cet état est sûr. La demande va être acceptée par le système.

Question **9**

Terminer

Non noté

Utilisez cette page pour justifier vos réponses.

1) On souhaite savoir si l'état est sur.

On sait que le nombre de ressource disponible est de A(9,3,6)

Les ressources requises par P1 <= A, ainsi P1 s'exécute, et les ressources disponible sont égale à A(10,3,6).

Les ressources requises par P2<= A, ainsi P2 s'exécute, et les ressources disponible sont égale à A(16, 4, 8).

Les ressources requises par P3 <= A, ainsi P3 s'exécute, et les ressources disponible sont égale à A(18, 5, 9).

Les ressources requises par P4<= A, ainsi P4 s'exécute, et les ressources disponible sont égale à A(18, 5, 11).

Ainsi tous les processus peuvent être exécuté et donc l'état est sur.

2) On sait que le système accepte la demande si l'allocation de R2 à P1 mène à un état sur. On verifie par l'algorithme du banquier

Or, ici cela est la cas. Ainsi si le système donne une ressource, A(9,2,6).

Les ressources requises par P1 >= A, ainsi P1 ne s'exécute pas

Les ressources requises par P2<= A, ainsi P2 s'exécute, et les ressources disponible sont égale à A(15, 4, 8).

Les ressources requises par P1 <= A, ainsi P1 s'exécute, et les ressources disponible sont égale à A(16,4,8).

Les ressources requises par P3 <= A, ainsi P3 s'exécute, et les ressources disponible sont égale à A(18,5,9).

Les ressources requises par P4<= A, ainsi P4 s'exécute, et les ressources disponible sont égale à A(18, 5, 11).

Ainsi même si le système accepte la demande, l'état reste sur.

Question **10**

Non répondue

Non noté

Utilisez cette page, si nécessaire, pour compléter la justification de vos réponses.

Question **11**

Partiellement correct

Note de 3,00 sur 7,00

Soit un système de gestion de la mémoire, à pagination pure, possédant les caractéristiques suivantes :

- une mémoire physique de 1 MiO,
- une adresse virtuelle codée sur 24 bits,
- des pages de 1 KiO chacune, et
- des entrées de tables des pages (TDPs) de 8 octets chacune.

La notation 2^x utilisée dans les choix de réponses signifie 2 puissance x.

1 - Quelle est la taille maximale en nombre de pages de l'espace virtuel ?

2^10 ✗

2 - Quel est le nombre de cadres dans l'espace physique ?

2^10 ✓

10

3 - Quel est le nombre de bits nécessaires pour stocker le déplacement dans un cadre ?

14 ✗

4 - Quel est, en hexadécimal, le numéro de page référencé par l'adresse virtuelle 0x014BA3 ?

0x14 ✗

5 - Quelle est la taille en nombre de cadres de la TDP, dans le cas d'une TDP à un niveau ?

2^8 ✗

6 - Dans le cas d'une TDP à un niveau, quel est le nombre minimal de cadres requis (en mémoire physique) pour y stocker les informations nécessaires à la localisation des pages virtuelles référencées par un processus en cours d'exécution ? L'adresse de la table des pages est supposée chargée dans un registre, lorsque le processus est en cours d'exécution.

2^6 ✗

7 - Quelle est la taille en nombre de cadres de la TDP, dans le cas d'une TDP à 2 niveaux et des tables de pages de même taille peu importe le niveau ?

2^8 ✗

8 - Dans le cas d'une TDP à 2 niveaux de la question précédente, quel est le nombre minimal de cadres requis (en mémoire physique) pour y stocker les informations nécessaires à la localisation des pages virtuelles référencées par un processus en cours d'exécution ? L'adresse de la table des pages du 1er niveau est supposée chargée dans un registre lorsque le processus est en cours d'exécution.

2 ✓

9 - Toujours dans le cas d'une TDP à 2 niveaux de la question précédente, quel est, en hexadécimal, le numéro de l'entrée, dans la table des pages du 1er niveau, référencée par l'adresse virtuelle 0x014BA3 ?

0x0 ✔

1- La taille maximale en nombre de pages de l'espace virtuel est 2^24 / 2^10 pages = 2^14 pages.

2- Le nombre de cadres dans l'espace physique est 2^20 / 2^10 cadres = 2^10 cadres.

3- Le nombre de bits nécessaires pour stocker le déplacement dans un cadre est 10 bits (car 1KiO= 2^10 octets).

4- Le numéro de page, en hexadécimal, référencé par l'adresse virtuelle 0x014BA3 est la valeur en hexadécimal des 14 premiers bits de poids le plus fort du code binaire de 0x014BA3 : 0x52.

5- La taille en nombre de cadres de la TDP à un niveau est : (2^14 entrées * 2^3 octets/entrée)/ 2^10 octets/cadre) = 2^7 cadres

6- Le nombre minimal de cadres requis est : 2^7 (espace nécessaire pour y stocker la TDP).

7- La taille en nombre de cadres de la TDP, dans le cas d'une TDP à 2 niveaux est : (1 + 2^7) cadres, car les 14 bits premiers bits de poids le plus fort d'une adresse virtuelle donnent le numéro de page. Comme toutes les tables du premier et second niveaux sont de même taille, chaque table a donc 2^7 entrées de 2^3 octets chacune (= 2^10 octets = 1 cadre). La taille de la table du premier niveau est donc 1 cadre et chaque entrée de cette table pointe vers une table du second niveau. Cette dernière a aussi une taille de 1 cadre. Nous avons au total (1 + 2^7) tables de 1 cadre chacune.

8 - Le nombre minimal de cadres requis est 2 (le premier est réservé à la table du 1er niveau alors que le second est réservé à une table de second niveau).

9 - Le numéro de l'entrée, en hexadécimal, dans la table des pages du 1er niveau, référencée par l'adresse virtuelle 0x014BA3 est la valeur, en hexadécimal, des 7 premiers bits de poids le plus fort du code binaire de 0x014BA3 : 0x0.

Commentaire :

Question **12**

Terminer

Note de 1,00 sur 1,00

Certains systèmes d'exploitation monoprocesseur offrent la possibilité à un thread en cours d'exécution de désactiver/activer l'ordonnanceur. Lorsqu'il est désactivé par un thread, l'ordonnanceur restera dans cet état jusqu'à ce que le thread l'active, se termine, cède le processeur ou se bloque.

Pensez-vous que la désactivation/activation de l'ordonnanceur serait une bonne solution au problème d'exclusion mutuelle dans un système monoprocesseur ? Justifiez votre réponse.

Je ne pense pas que la désactivation/activation de l'ordonnanceur est une bonne solution au problème d'exclusion mutuelle. En effet, si un thread décide de désactiver l'ordonnancement, et qu'il est dans l'attente d'une ressource, ce thread ne pourra pas s'exécuter et donc cela accentue encore plus le problème d'exclusion mutuelle, car il ne pourra obtenir cette ressource, l'ordonnanceur ne pourra pas enclanché l'execution d'un autre processus/thread et il y aura interblocage

Du côté des utilisateurs, cette solution permet d'assurer l'exclusion mutuelle en désactivant l'ordonnanceur avant d'entrer en section critique et en l'activant ensuite à la sortie de la section critique, sous réserve d'éviter les appels système bloquants, la terminaison et les attentes actives dans les sections critiques.

Cependant, cette solution est dangereuse pour le système car un utilisateur pourrait lui faire perdre le contrôle de la gestion de l'allocation du processeur. Ce qui n'est pas acceptable. Elle pourrait être toutefois utilisée avec précaution par le système lui-même pour assurer l'exclusion mutuelle. Il faudrait notamment s'assurer qu'il n'ait pas de condition de concurrence avec les routines des interruptions car la désactivation de l'ordonnanceur ne désactive pas les interruptions. Il faudrait aussi éviter les attentes actives, la terminaison et les appels bloquants dans les sections critiques.

En conclusion, ce n'est pas une bonne solution car un utilisateur pourrait, en monopolisant le processeur, nuire aux autres utilisateurs et au système lui-même.

Commentaire :

Question **13**

Partiellement correct

Note de 1,00 sur 2,00

Supposez un système monoprocesseur et les 5 processus (P1, P2, P3, P4 et P5) décrits dans le tableau suivant :

Processus	Date d'arrivée	Priorité	Temps d'exécution
P1	0	3	5 (2) 3
P2	1	3	6
P3	2	2	2 (3) 2
P4	4	1	2
P5	5	2	2

Ce système dispose d'un seul périphérique d'E/S qui gère les requêtes d'E/S selon la discipline FIFO. Le temps de commutation de contexte est nul. La priorité 1 est la plus basse. Le temps d'exécution **X (Y) Z** d'un processus *Pi* signifie que l'exécution de *Pi* nécessite, dans l'ordre, *X* unités de temps CPU, *Y* unités de temps en E/S et *Z* unités de temps CPU.

Donnez le diagramme de Gantt montrant l'ordre d'exécution des 5 processus, dans le cas d'un ordonnancement préemptif à files multiples et priorités fixes. L'ordonnancement des processus de même priorité est circulaire avec un quantum égal à 3. Pour répondre à cette question, complétez le tableau suivant :

0	P1	✔
1	P1	✔
2	P1	✔
3	P2	✔
4	P2	✔
5	P2	✔
6	P1	✔
7	P1	✔
8	P1	✖
9	P2	✔
10	P2	✔
11	P2	✖
12	P1	✔
13	P1	✔
14	P1	✖
15	P1	✖
16	P3	✖
17	P3	✖
18	P3	✖
19	P5	✖
20	P5	✖

21	P3	✖
22	P3	✖
23	P4	✖

Votre réponse est partiellement correcte.

Vous en avez sélectionné correctement 12.
La réponse correcte est :
0 → P1,

1 → P1,

2 → P1,

3 → P2,

4 → P2,

5 → P2,

6 → P1,

7 → P1,

8 → P2,

9 → P2,

10 → P2,

11 → P1,

12 → P1,

13 → P1,

14 → P3,

15 → P3,

16 → P5,

17 → P5,

18 → P4,

19 → P3,

20 → P3,

21 → P4,

22 → rien,

23 → rien

Question **14**

Terminer

Note de 1,00 sur 1,00

Un système temps réel gère 3 tâches indépendantes périodiques (T1, T2 et T3) avec respectivement des périodes de (5, 10 et 20 ms). Supposez que les tâches (T1, T2 et T3) aient besoin, au maximum, respectivement de (2, 3 et x ms) de temps processeur pour réaliser leurs traitements périodiques.

Quelle est la plus grande valeur de x pour laquelle il est possible d'ordonnancer ces trois tâches, selon un ordonnancement préemptif à priorités ? Justifiez votre réponse.

On souhaite déterminer la plus grande valeur de x pour qu'il soit possible d'ordonnancer ces 3 taches.

On souhaite respecter la condition suivants:

2/5 + 3/10 + x/20 <= 1

Ainsi on a x/20 <= 1 - 7/10

x/20 <= 3/10

x= 3/10 * 20 = 6

Ainsi, pour qu'il soit possible d'ordonnancer ces 3 tâches selon un ordonnancement préemptif à priorité, la valeur maximal de x est de 6.

La condition d'ordonnançabilité la plus permissive, dans le cas d'un ordonnancement préemptif de tâches périodiques indépendantes, est celle de EDF. De plus, pour les tâches à échéances sur requêtes (l'échéance de chaque tâche est égale à sa période), nous avons une condition nécessaire et suffisante d'ordonnançabilité : $\sum_{i=1,n} Ci/Pi \leq 1$, où n est le nombre de tâches, Ci et Pi sont respectivement le

temps d'exécution du traitement périodique et la période de la tâche Ti. Il suffit d'appliquer cette condition pour extraire la plus grande valeur de x:

2/5+3/10+x/20 <= 1 ==> 4*2 + 3*2 + x <=20 => x <= 6. La plus grande valeur de x est 6 ms.

Commentaire :

Question **15**

Partiellement correct

Note de 0,75 sur 1,50

Sous Windows, un processus crée un objet "héritable" de type sémaphore. Il crée ensuite un processus fils en lui faisant hériter le descripteur de l'objet sémaphore créé. Le descripteur de l'objet est transmis, au processus fils, via un paramètre de son programme.

1- Le père et le fils peuvent-ils utiliser ce sémaphore pour accéder en exclusion mutuelle à la sortie standard (et éviter ainsi un mélange de résultats) ?

Oui ✔

2 - Le père et le fils peuvent-ils utiliser ce sémaphore pour accéder en exclusion mutuelle à un tableau alloué dynamiquement par le père avant la création du processus fils ? L'adresse de ce tableau est aussi transmise, au processus fils, via un paramètre de son programme.

Non ✔

Utilisez la page suivante pour justifier vos réponses. Les réponses non ou mal justifiées ne seront pas considérées.

1- Oui, car le père et le fils partagent la sortie standard et le sémaphore (les sémaphores font parties des objets héritables par un processus fils).

2- Non, car le père et le fils ont chacun un espace d'adressage privé. L'allocation d'espace dynamique réalisée par le père n'existe pas dans l'espace virtuel du fils.

Commentaire :
Justification partiellement correcte

le pere et le fils peuvent utiliser semaphore pour acceder en exclusion mutuelle a la sortie standard car l'usage des semaphores est de synchroniser des processus et des threads, on synchronise donc notre pere et notre fils. Comment ?

2- un tableau alloué par le pere est une ressource qui fait reference a l'espace d'adressage privé du pere (et ce genre de données n'est pas heritable par le fils), le tableau modifié par le fils est donc bien une copie <== non

mais n'est pas le même tableau que celui du pere, les modifications ne se font pas sur le même tableau alors avoir un semaphore pour cela serait inutile vu qu'on a pas besoin d'avoir de l'exclusion mutuelle si on a pas de ressource partagée.

Question **16**

Terminer

Non noté

Utilisez cette page pour justifier vos réponses à la question précédente.

1) On sait que le pere et le fils partagent la sortie standard, et l'utilisation d'un sémaphore leur permettra de chacun y avoir acces sans risque de résultats non désirés.

2) L'adresse ne sera pas transmise au fils, il peut y avoir acces mais l'espace d'adressage reste privée au père.

◀ Hiver 2022 - Contrôle périodique - 9 mars

Aller à...