Examen 2023/2024 (Correction)

1. **Comment peut-on prévenir le blocage de thread ? (expliquer le mécanisme)1pt**

Dans certains SE comme UNIX il est possible de prédire le blocage de thread

Un appel système appelé *SELECT* permet de savoir si un *read* à venir donne un blocage

La procédure *Read est* remplacer par *Select,* si le Select s’exécute alors la procédure Read est invoqué

Si *Select* se bloque alors un autre thread est exécuter, quand il fini on essaye encore le thread bloqué

1. Qu’elles sont les trois règles à respecter pour le bon fonctionnement du producteur-consommateur ? **1pt**

le producteur ne doit pas produire si le tampon est plein;

le consommateur ne doit pas faire de retrait si le tampon est vide;

le producteur et le consommateur ne doivent jamais travailler dans une même case

1. Qu’est-ce qu’un Interblocage ? **1pt**

Chaque processus de l’ensemble attend un autre, mais aucun d’eux ne peux fonctionner (bloqué), aucun d’eux ne peut libérer une ressource et aucun ne peut être réveillé

1. Quelle est la règle à respecter pour utiliser le  *wait & signal* dans les moniteurs :

Le wait doit toujours être avant le signal**1pt**

**Exercice 1**

On considère 4 processus P1, P2, P3 et P4 ainsi que 5 types de ressources R1, R2, R3, R4 et R5. Les tableaux ci-dessous expriment les besoins en ressources des processus pour leur exécution complète ainsi que l’état du système E=(11,12,8,7,10)

Attribué Besoin Max

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **R1** | **R2** | **R3** | **R4** | **R5** |
| **P1** | **2** | **3** | **2** | **3** | **2** |
| **P2** | **2** | **3** | **2** | **0** | **2** |
| **P3** | **1** | **2** | **1** | **2** | **1** |
| **P4** | **2** | **3** | **1** | **1** | **2** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **R1** | **R2** | **R3** | **R4** | **R5** |
| **P1** | **3** | **4** | **3** | **5** | **3** |
| **P2** | **8** | **5** | **2** | **2** | **6** |
| **P3** | **4** | **5** | **3** | **4** | **5** |
| **P4** | **5** | **5** | **2** | **2** | **4** |

1. Donner le vecteur des ressources disponibles A, ainsi que la matrice de besoin.
2. Le système dans cet état est-il sûr ? (donnez tous les ordonnancements possibles)
3. Si P4 fait la demande (1,0,1,0,1) :
   1. Donner le nouvel état pour P4 (vecteurs demande, attribution) et disponible A
   2. Peut-on accorder cette demande à P4

**Solution Exercice =7,5 PTS**

1. Matrice de besoin et vecteur A 0,5 x2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **R1** | **R2** | **R3** | **R4** | **R5** |
| **P1** | **1** | **1** | **1** | **2** | **1** |
| **P2** | **6** | **2** | **0** | **2** | **4** |
| **P3** | **3** | **3** | **2** | **2** | **4** |
| **P4** | **3** | **2** | **1** | **1** | **2** |

A=(4 1 2 2 3)

1. Les ordonnancements : 0,25 POUR CHAQUE ETAPES même pour question 3

Exe de P1 donne A1= (6 4 4 5 5)

Ensuite on a les 3 processus qui peuvent passer

* + - * Exe de P2 donne A2 = (8 7 6 5 7 )

On a deux possibilités

P3 donne A3 = ( 9 9 7 7 8) ensuite P4 donne A4 = ( 11 12 8 8 10) = E

Ou bien

P4 done A3 = (10 10 7 6 9) ensuite P3 donne A4 = (11 12 8 8 10)= E

* + - * Exe de P3 donne A2 = (7 6 5 7 6 )

On a deux possibilités

P2 donne A3 = ( 9 9 7 7 8) ensuite P4 donne A4 = ( 11 12 8 8 10) = E

Ou bien

P4 done A3 = (9 9 6 8 8) ensuite P2 donne A4 = (11 12 8 8 10)= E

* + - * Exe de P4 donne A2 = (8 7 5 6 7 )

On a deux possibilités

P2 donne A3 = ( 10 10 7 6 9) ensuite P3 donne A4 = ( 11 12 8 8 10) = E

Ou bien

P3 done A3 = (9 9 6 8 8) ensuite P3 donne A4 = (11 12 8 8 10)= E

**Le système est dans un état sûr**

3) Si P4 fait la demande (1,0,1,0,1) : ici une exécution suffit, on ne compte qu’une seule

Le nouveau A=(3 1 1 2 2 ) Attribué de P4 (3 3 2 1 3) Besoin (2 2 0 1 1)

On fait passer P1 : A1 = (5 4 3 5 4)

On a deux cas

-Soit P3 : A2= (6 6 4 7 5) ensuite

P2 donne A3 = ( 8 9 6 7 7) ensuite P4 donne A4 = ( 11 12 8 8 10) = E

Ou bien

P4 done A3 = (9 9 6 8 8) ensuite P2 donne A4 = (11 12 8 8 10)= E

- Soit P4 donne A2 = (8 7 5 6 7 ) ensuite :

P2 donne A3 = ( 10 10 7 6 9) ensuite P3 donne A4 = ( 11 12 8 8 10) = E

Ou bien

P3 done A3 = (9 9 6 8 8) ensuite P3 donne A4 = (11 12 8 8 10)= E

**Oui on peut lui accorder sa demande**

**Problème : L’hôpital**

Dans un hôpital, la bonne gestion des interventions chirurgicales est très critique, des portes automatiques permettent l’accès entre la salle d’anesthésie et le bloc opératoire, ces portes ne permettent le passage que dans un seul sens à la fois pour éviter le croisement des brancards.

1) Ecrire avec les sémaphores les algorithmes du personnel médical tels que :

-Le personnel qui va de la salle d’anesthésie au bloc opératoire est nommé Pers\_Anes

-Le personnel qui sort du bloc opératoire est appelé Pers\_Bloc

-Plusieurs personnes peuvent aller dans un sens mais jamais dans les deux en même temps

-Il n’y a pas de priorité entre les deux sens

Le chef du service de chirurgie veut donner la priorité au personnel qui rentre au bloc opératoire.

Un système de priorité est donc mis en place, si un chirurgien veut rentrer au bloc il est prioritaire et le personnel qui sort du bloc doit attendre qu’il rentre.

2) Modifier l’algorithme précédant pour rajouter la priorité et laisser passer Pers\_Anes plus vite

3) Ecrire l’algorithme de la question 1 avec les moniteurs.

**Solution 8.5 POINTS**

1. **NbA :** Nombre de personnes Pers\_Bloc

**NbB :** Nombre de personnes Pers\_Anes

**S1 :** permet de réserver le couloir dans le sens vers le bloc opératoir

**S2 :** permet de réserver le couloir dans le sens vers la salle d’anesthésie

**Mut1 :** protège la variable NbA

**Mut2 :** protège la variable NbB

|  |  |
| --- | --- |
| **var**  Init(S1, 1); Init(S2, 1); Init(mut1, 1); Init(mut2, 1); 0.25\*4  NbA=0; NbB=0; 0.25\*2 | |
| **Processus Pers\_Salle**  **Début**  **{**  P(S1) 0.25 si V(s1) est bon  P(Mut1) 0.25 si V(mut1) est bon  NbA++ si (NbA==1) alors P(S2) fsi 0.25  V(Mut1)  V(S1)  <Entrer dans la cuisine >  P(Mut1) 0.25 si V(mut1) ok  NbA - - si (NbA==0) alors V(S2) fsi 0.25  V(Mut2)  **}** | **Processus Pers\_Cuisine**  **Début**  **{**  P(S2) 0.25 si V(s2) ok  P(Mut2) 0.25 si V(mut2) ok  NbB++ si (NbB==1) alors P(S1) fsi 0.25 V(Mut2)  V(S2)  <Entrer dans la salle>  P(Mut2) 0.25 si V(mut2) ok  NbB - - si (NbB==0) alors V(S1) fsi 0.25 V(Mut2)  **}** |

**Avec priorité : On ne compte que la modification**

|  |  |
| --- | --- |
| **var**  Init(S1, 1); Init(S2, 1); inti (S3,0) Init(mut1, 1); Init(mut2, 1); bool: Prio=False 0.25\*2  NbA=0; NbB=0; | |
| **Processus Pers\_Salle**  **Début**  **{**  P(S1) ;  If (prio ==True) : P(S3) ; 0.25  P(Mut1) ;  NbA++ si (NbA==1) alors P(S2) fsi ;  V(Mut1) ;  V(S1) ;  <Entrer dans la cuisine >  P(Mut1) ;  NbA - - si (NbA==0) alors V(S2) fsi ;  V(Mut2) ;  **}** | **Processus Pers\_Cuisine**  **Début**  **{**  Prio = True ; 0.25  P(S2) ;  P(Mut2) ;  NbB++ ; 0.25  V(Mut2) ;  V(S2) ;  <Entrer dans la salle>  P(Mut2) ;  NbB - - si (NbB==0) alors {V(S3) ; Prio =Fals} fsi ; 0.25  V(Mut2) ;  **}** |

**Solution avec moniteur**

|  |  |
| --- | --- |
| M = Moniteur  Var  Int nbA=nbB=attA=attB=0 0.25 x 4  Condition SV, SF ;  Procedure Sortie Anesthésie()  Begin  si (nbB>0) alors attA++ ; wait(SV) fsi 0,25  nbA++ ; 0,25  si (attA>0) alors attA-- ; signal(SV) ; fsi 0.25  End;  Procedure Entrée Bloc()  Begin  nbA-- ; 0,25  si (nbA==0) alors  si (attB>0) alors attB-- ; signal(SF) fsi 0,25  fsi  End; | Procedure Sortie Bloc()  Begin  si (nbA>0) alors attB++ ; wait(SF) fsi 0,25  nbB++ ; 0,25  si (attB>0) alors attB-- ; signal(SF) ; fsi 0,25  End;  Procedure Entrée Anesthésie()  Begin  nbB-- ; 0,25  si (nbB==0) alors  si (attA>0) alors attA-- ; signal(SV) fsi 0,25  fsi  End; |