Série de TD sur la Communication Inter -Processus
AVEC Corrigés

Questions :

1. Qu'est ce qu'une section critique ?
2. Qu'est ce que l'exclusion mutuelle ?
3. Quelles sont les propriétés (conditions) que doit vérifier un protocole de gestion d'accès en SC?
4. Quel est le rôle d'un sémaphore?
5. Ecrire les primitives P et V.
6. Définir : Sémaphore binaire, sémaphore compteur et sémaphore bloquant.
7. Peut-on avoir un vecteur de sémaphore? Donner un exemple.

Exercice 1: Soient les codes des deux processus ci dessous:

P1();	P2();
{.... Répéter Tant que (Tour = 2) Faire sleep(délai); FinTQ <**section critique**> Tour :=2 Jusqu'à faux; ... }	{.... Répéter Tant que (Tour = 1) Faire sleep(délai); FinTQ <**section critique**> Tour :=1 Jusqu'à faux ... }

1. Discuter la vérification des propriétés attendues d'un protocole de manipulation de SC.

Corrigé : Propriétés attendues d'une solution

1. Exclusion mutuelle : A tout instant, un processus au plus exécute des instructions de sa section critique
2. Condition de progression =Absence de blocage (permanent)
   Si plusieurs processus attendent pour entrer en SC, et si aucun processus n'est déjà en SC, alors un des processus qui attend doit pouvoir entrer en SC au bout d'un temps fini
3. Condition de progression (cad. blocage que temporaire)
   Un processus qui se trouve hors de sa SC et hors du protocole contrôlant l'accès à la SC ne doit pas empêcher un autre processus d'entrer dans sa SC : un processus ne doit pas ralentir un autre
4. Equité (Absence de famine)
   Un processus qui est bloqué à l'entrée de la section critique n'attendra pas indéfiniment son tour
   Pour un processus qui veut entrer en SC, il existe une borne supérieure au nombre de fois où d'autres entités exécuteront leur SC avant lui (La valeur de la borne permet de mesurer à quel point une solution est équitable)

Solutions avec attente active

Une entité désirant entrer en SC attend de façon active qu'une condition soit vérifiée (ici, aucune autre entité en SC)

Tant que (condition indique SC non libre) Faire rien ou sleep(court délai) Fintantque
<section de code critique>
Modifier condition pour refléter SC libre

Problème : Consommation inutile de temps CPU

Répéter Tant que (Tour = 2) Faire sleep(délai); FinTQ <section critique> Tour :=2 Jusqu'à faux	Répéter Tant que (Tour = 1) Faire sleep(délai); FinTQ <section critique> Tour :=1 Jusqu'à faux

Analyse algorithme

Exclusion mutuelle : Ok
Absence de blocage : Ok
Condition de progression : NON
_ Lorsque Pi quitte sa SC, Tour:=j
_ Pour entrer à nouveau en SC, Pi doit attendre que Pj exécute sa SC (stricte alternance des 2)
Absence de famine : Ok
Code identique : NON

Exercice 2: Soit la solution ci-dessous

Tour : variable commune, initialement =1 Drapeau1, Drapeau2, initialement = faux (quand tour =i, Pi peut entrer en SC)	
P1 … Drapeau1 = vrai ; Tour:=2 Tant que (Drapeau2 ET Tour=2) Faire sleep(d); Fintantque <section critique> Drapeau1 := faux …	P2 … Drapeau2 = vrai ; Tour:=1 Tant que (Drapeau1 ET Tour=1) Faire sleep(d); Fintantque <section critique> Drapeau2 := faux …

1. Discuter la vérification des propriétés attendues d'un protocole de manipulation de SC.

SOLUTION :

Propriétés :

1 – E.M. : Par l'absurde : On suppose P1 et P2 en SC critiques les deux en même temps

P1 en SC, donc	P2 en SC, donc
(Drapeau2 ET Tour=2) =Faux ⇔ Drapeau2 = Faux OU Tour = 1 Or P1 en SC donc Drapeau1 = V ➔ Drapeau1 = Faux Impossible Il reste donc **Tour = 1** **Or, tour ne peut etre egale a 1 et a 2 en meme temps, donc P1 et P2 ne peuvent être en SC en même temps**	(Drapeau1 ET Tour=1) = Faux ⇔ Drapeau1 = Faux OU Tour=2 Or P2 en SC donc Drapeau2 = V ➔ Drapeau2 = Faux Impossible Et, **Tour = 2**

La condition 'Aucun processus s'exécutant à l'extérieur de sa S.C ne doit bloquer d'autres processus' peut être vue dans les deux situations ci-dessous.

2 - **Absence de blocage :** Aucun processus s'exécutant à l'extérieur de sa S.C ne doit bloquer d'autres processus

Par l'absurde, on suppose, P1 et P2 bloqués sur le demande d'accès.

P1	P2
… Drapeau1 = vrai ; Tour:=2 Condition TQ : (Drapeau2 ET Tour=2) = VRAI ⇔ Drapeau2 = Vrai ET Tour=2 => Drapeau2 = Vrai →OK Et Tour=2	… Drapeau2 = vrai ; Tour:=1 Condition TQ (Drapeau1 ET Tour=1) = VRAI ⇔ Drapeau1 = Vrai ET Tour=1 ==> Drapeau1 = Vrai → OK Et Tour=1 //Impossible en même temps

3- **Condition de Progression** : Aucun processus s'exécutant à l'extérieur de sa S.C ne doit bloquer d'autres processus

Par l'absurde, on suppose que P1 n'est pas en SC mais P2 est bloqué et n'arrive pas à rentrer en SC

P1	P2
… Drapeau1 = vrai ; Tour:=2 Tant que (Drapeau2 ET Tour=2) Faire sleep(d); Fintantque <section critique> Drapeau1 := faux … …… …….. //P1 hors SC	… Drapeau2 = vrai ; Tour:=1 // on suppose P2 bloqué, donc (Drapeau1 ET Tour=1) = VRAI ⇔ Drapeau 1= VRAI ET Tour ==1 VRAI Or, P1 hors SC, donc soit il n'a pas encore exécute dutout sa SC et alors Drapeau1 initialisé à FAUX soit, il l'a exécute et dans ce cas, il a terminé l'execution en mettant Drapeau1=Faux, dans tus les cas Drapeau1 – Faux. PUISQUE la 1ere condition Drapeau1==Faux, ilest impossible que (Drapeau1 ET Tour=1) = VRAI

4 – Famine : Aucun processus ne doit attendre indéfiniment pour entrer en S.C.

a. Supposons que P2 est en attente de SC, et que P1 est en SC. Lorsque P1 quitte sa SC, est ce qu'il peut l'exécuter encore une fois alors que P2 est toujours Bloqué

P1	P2
… Drapeau1 = vrai (*1*) ; Tour:=2 Tant que (Drapeau2 ET Tour=2) Faire sleep(d); Fintantque <section critique> Drapeau1 := faux ----------→ … …… …….. //P1 hors SC	… Drapeau2 = vrai ; Tour:=1 (*2*) // on suppose P2 bloqué, donc (Drapeau1 ET Tour=1) = VRAI ⇔ Drapeau 1= VRAI (*1*) ET Tour ==1 VRAI (*2*) A ce moment là, Drapeau 1= FAUX ALORS, (Drapeau1 ET Tour=1 ==FAUX ⇨ P2 entre en SC, il ne retse pas bloqué

b. Supposons que P2 est en dehors de SC, et que P1 est en SC. Lorsque P1 quitte sa SC, est ce qu'il peut l'exécuter encore une fois alors que P2 est toujours Bloqué (Cas Progression vu pus haut)

Exercice 3:

On veut enregistrer les commandes des clients dans un fichier .COM et éditer les bons de commande. D'autre part, un processus facturation lit périodiquement les commandes et les édites sur imprimante. Donner les codes des processus Facturation et Enregistrement. Discuter les différents scénarios d'exécution.

Exercice 4:

Soient deux processus P1 et P2,
P1 doit exécuter deux tâches <T11> et <T12>. P2 doit aussi exécuter deux tâches <T21> et <T22>. P2 ne peut exécuter sa deuxième tâche avant que P1 n'ai terminé sa première tâche. Donner le pseudo code de chacun des processus. Discuter la correction de votre solution.

Exercice 5:

Nous considérons une Base de données où plusieurs processus peuvent lire en même temps mais un seul peut écrire à la fois.

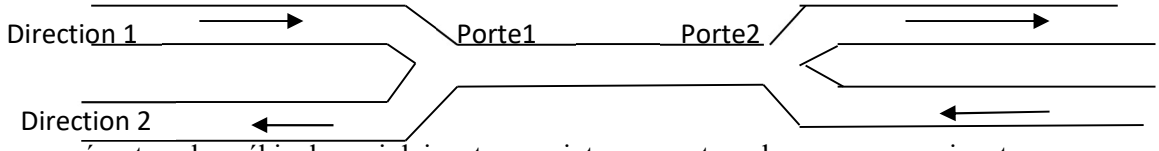
1. Discuter les différents cas possibles.
2. Proposer un code pour les processus lecteurs et le code d'un processus écrivain.
3. Discuter les propriétés que doit vérifier l'accès à la SC.

Exercice 6:

1. Définir le problème des philosophes.
2. Proposer une ou des solutions à ce problème sans utilisation de sémaphores ? Critiquer ces solutions.
3. Proposer une solution avec utilisation de sémaphores ? Critiquer la solution.

Exercice 7:

Nous considérons un pont de circulation à une seule voie sur lequel, il n'est pas possible d'autoriser le passage de deux véhicules circulant dans des directions différentes.



Nous représentons les véhicules qui doivent empreinter ce pont par les processus suivants:

Processus Direction1	Processus Direction2
...	.....
AccèsPont.Porte1()	AccèsPont.Porte2()
<circuler sur le pont>	<circuler sur le pont>
SortiePont.porte2	SortiePont.porte1()
....	.....

1/ Nous supposons que le pont peut comporter un seul véhicule qui le traversent à la fois.
Ecrire les procedure AccèsPont et SortiePont() en utilisant des semaphores pour la synchronisation

2/ Nous supposons que le pont peut comporter un nombre infini de véhicules qui le traversent dans un même sens à un moment donné.
Ecrire les procedure AccèsPont et SortiePont() en utilisant des semaphores pour la synchronisation.

3/ Nous supposons maintenant, que le pont ne peut comporter qu'un nombre N de véhicules à la fois.
Donner, dans ce cas, les procédure AccèsPont() et SortiePont() en utilisant les semaphores.

4/ Examiner, dans les cas précédents, les risques de privation.

Exercice 8:

On veut offrir des procédures de mise à jour d'une variable avec la condition que la variable doit être manipulée en EM. Proposez un moniteur pour ces procédures.

Exercice 9:

On veut traiter le problème de Rendez-vous de N processus en utilisant les moniteurs.

Exercice 10:

Reprendre le problème des Producteurs Consommateurs avec les moniteurs. Nous supposons N tampons.

Exercice 11:

Donner une solution à base de moniteur pour le problème des Lecteurs /Rédacteurs. Traiter les cas, où la priorité est donnée aux lecteurs. On peut étudier les cas de priorité aux rédacteurs ou l'équité.

Exercice 12:

Un abri pour oiseau comporte une porte. L'abri peut contenir au maximum trois oiseaux. La porte comporte un bouton à l'extérieur et un bouton à l'intérieur. Les oiseaux sont dressés pour appuyer sur les boutons s'ils veulent entrer ou sortir.

1. Donner les codes simulant cette situation en utilisant les sémaphores.
2. Précisez le rôle de chaque sémaphore.
3. Quelles sont les conditions que doit respecter cette synchronisation ?
4. Est-ce que votre proposition respecte ces conditions ? Justifier.
5. Proposez une solution à base de moniteurs.

L'abri a été élargi pour accueillir 100 oiseaux. Pour cela l'abri a été muni de N portes (N étant une constante).

6. Proposez les codes simulant cette nouvelle situation.
7. Proposez une solution à base de moniteurs.