**TD Synchronisation de processus**

I La piscine 1

II Lecteur / rédacteur 2

A. Utilisation d’un unique sémaphore en exclusion mutuelle (Mut) 2

B. Utilisation de plusieurs sémaphores 2

III Le diner des philosophes 3

A. Première solution: 4

B. Deuxième solution : 4

C. Troisième solution 4

D. Quatrième solution 4

## I La piscine

Dans cet exercice nous allons vous demander de modéliser le fonctionnement d'une piscine. Les clients de la piscine sont des threads, nous vous demandons donc d'écrire le code de ces threads correspondant au comportement décrit dans la suite de l'énoncé.

Un baigneur doit :

* Arriver à la piscine ;
* Prendre un panier ;
* Trouver cabine et entrer dedans afin de se déshabiller ;
* Libérer la cabine ;
* Aller nager
* Prendre une cabine et s’habiller ;
* Rendre son panier ;
* Quitter la piscine ;

Ressources critiques : certaines phases de ces activités mettent en jeu des ressources critiques, pour lesquelles les threads rentrent en compétitions :

* la piscine dispose d'un certain nombre NbPaniers de paniers dans lesquels les baigneurs laissent leurs habits pendant le temps de la baignade ;
* chaque baigneur se déshabille et se rhabille dans une cabine individuelle. La piscine dispose d'un nombre NbCabines de cabines.

Les phases de déshabillage et de rhabillage ne peuvent commencer que si certaines conditions sont réalisées :

* chaque baigneur doit disposer d'un panier et d'une cabine pour commencer à se déshabiller ;
* chaque baigneur ne doit disposer que d'une cabine pour commencer à se rhabiller.

Ces deux conditions sont donc éventuellement bloquantes.

Les fins des phases de déshabillage et de rhabillage ne sont pas bloquantes mais correspondent à la libération de ressources critiques :

* chaque baigneur rend sa cabine quand il a fini de se déshabiller ;
* chaque baigneur rend sa cabine et son panier quand il a fini de se rhabiller.

Le code du baigneur peut être le suivant :

//arrivée

sleep (XXX) ;

// déshabillage

DebutDeshabiller();

sleep(XXX);

FinDeshabiller();

//baignade

sleep (XXX) ;

// rhabillage

DebutRhabiller() ;

sleep(XXX) ;

FinRhabiller() ;

//départ

sleep (XXX) ;

Ecrire le corps des fonctions : DebutDeshabiller (), FinDeshabiller (), DebutRhabiller () et FinRhabiller () selon les spécifications précédentes.

## II Lecteur / rédacteur

### A. Utilisation d’un unique sémaphore en exclusion mutuelle (Mut)

Coté lecteur :

T Lire ()

 Mut.P();

 T info = RetirerInfo ();

 Mut.V ();

 return info;

Coté rédacteur :

void Ecrire (T info)

 Mut.P();

 PoserInfo (info);

 Mut.V();

Montrer que l’on peut obtenir un interblocage de processus.

### B. Utilisation de plusieurs sémaphores

La solution ci-dessous est inspirée de celle vue en cours, malheureusement elle n’est pas fonctionnelle. Dites pourquoi et corrigez la.

Indication : on donnera toujours priorité aux lecteurs par rapport aux rédacteurs.

Init : MutRedPresent = 0 ; MutRedEcr = 0 ; MutLect = 0;

Coté lecteur :

MutRedPresent.P ();

MutLect.P();

 T info = RetirerInfo ();

MutLect.V ();

return info ;

Coté rédacteur :

MutRedEcr.P();

 PoserInfo (info) ;

MutRedEcr.V();

MutRedPresent.V ();

## III Le diner des philosophes

Problème :

La situation est la suivante :

* cinq philosophes (initialement mais il peut y en avoir beaucoup plus) se trouvent autour d'une table ;
* chacun des philosophes a devant lui un plat de spaghetti ;
* à gauche de chaque plat de spaghetti se trouve une fourchette.

Un philosophe n'a que trois états possibles :

* penser pendant un temps indéterminé ;
* être affamé (pendant un temps déterminé et fini sinon il y a famine) ;
* manger pendant un temps déterminé et fini.

Des contraintes extérieures s'imposent à cette situation :

* quand un philosophe a faim, il va se mettre dans l'état « affamé » et attendre que les fourchettes soient libres ;
* pour manger, un philosophe a besoin de deux fourchettes : celle qui se trouve à gauche de sa propre assiette, et celle qui se trouve à droite (c'est-à-dire les deux fourchettes qui entourent sa propre assiette) ;
* si un philosophe n'arrive pas à s'emparer d'une fourchette, il reste affamé pendant un temps déterminé, en attendant de renouveler sa tentative.

Le problème consiste à trouver un ordonnancement des philosophes tel qu'ils puissent tous manger, chacun à leur tour. Cet ordre est imposé par la solution que l'on considère comme celle de Dijkstra avec sémaphores ou Courtois avec des compteurs.

Le corps du processus philosophe peut être le suivant :

Philo(i)

{

 while(true)

 {

 pense( );

 prendre\_fourchette( i );

 manger ( );

 poser\_fourchette( i );

 }

}

Conséquence, il faut écrire les fonctions prendre\_fouchette () et poser\_fourchette ().

### A. Première solution

On prend un sémaphore par fourchette (un mutex fait aussi l’affaire), chaque sémaphore est initialisé à 1.

sem fourch[ N ] = {1,........1}

prendre\_fourchette (int i)

{

 fourch[i].P() ;

 fourch[(i+1)% N].P() ;

}

poser\_fourchette (int i)

{

 fourch[i].V() ;

 fourch[(i+1)% N].V() ;

}

Monter que cette solution peut provoquer un interblocage de processus.

### B. Deuxième solution

On modifie la solution 1 de sorte que les philosophes ne puissent pas prendre tous en même temps la fourchette de droite.

On introduit pour cela un sémaphore dont le compteur est égal à N-1

Implémenter cette solution.

### C. Troisième solution

C'est la même chose que la première solution sauf qu'un philosophe est gaucher et ceci afin d'éviter l'interblocage.

Implémenter cette solution.

### D. Quatrième solution

On se centre ici sur les philosophes. Un sémaphore est attribué à chaque philosophe.

Etat = { PENSE , MANGE , A\_FAIM}

Un philosophe qui veut prendre les fourchettes (donc manger) déclare qu'il a faim. Si l'un de ses deux voisins est en train de manger, il ne peut donc pas prendre les deux fourchettes pour l'instant et donc se met en attente. Si les deux philosophes à coté ne mangent pas alors il peut prendre les deux fourchettes et déclarer qu'il mange. Quand le philosophe a fini de manger, il déclare donc qu'il pense et regarde si ses deux voisins qui forcément ne pouvaient pas manger, en ont dès lors l'opportunité. Si c'est le cas il les réveille.

Implémenter cette solution.