USTHB
Faculté d'Informatique
Département des Systèmes Informatiques
Epreuve Finale – Durée : 1h30
(Systèmes d'exploitation- M1: IV) Le 15/01/2023 - Année 22/23

Exercice n°1 : (6 pts= 2 + 2 + 2)

Répondre aux questions suivantes :

A- En général, on considère 4 conditions pour vérifier si un protocole est retenu ou non comme solution au problème d'exclusion mutuelle. Citer la 5ieme condition et dire pourquoi on ne la considère pas dans ce contexte.

B- Pourquoi les sémaphores sont qualifiés d'outil de synchronisation de bas niveau?

C- Lister les trois types d'attente des processus dans les moniteurs classiques.

Exercice n°2 : (7 pts=4+3)

Le concept d'événements peut être défini de plusieurs manières. Cette fois-ci, on se propose de le formuler comme suit en définissant le concept de *compteur d'événements* au lieu d'*événements*.

Un **compteur d'événement** est défini comme une variable particulière permettant de compter le nombre d'occurrences d'un événement *E* donné. Il est utilisé pour la synchronisation entre processus. Notons ici que le nom de l'événement est aussi le compteur d'événement. Ce compteur est manipulé par deux primitives comme suit :

- ***await (E, v)*** : fait attendre le processus jusqu'à ce que *E* (ie. le compteur) atteigne ou dépasse la valeur *v*.
- ***advance (E, 1)*** : incrémente *E* de *1* et débloque tous les processus en attente pour lesquels *E* atteint la valeur indiquée dans la primitive *await()*.

A/ Utiliser cet outil pour réaliser un rendez-vous entre *10* processus de telle sorte que chaque processus arrivant à ce point de rendez-vous continue son exécution seulement si tous les autres processus sont arrivés à ce point.

B/ Adapter la solution dans le cas où chaque processus parmi les *10* continue son exécution après le point de rendez-vous si au moins *5* autres arrivent à ce même point.

Exercice n°3: (7pts= 2+5)

On s'intéresse à la gestion, à l'aide de moniteurs classiques, d'une classe de ressources à *M* exemplaires partagées par *N* processus numérotés de *1* à *N*. Chaque processus peut demander à la fois *k* exemplaires de ressources et peut continuer son exécution même si on satisfait partiellement sa demande. Par exemple, si P_i demande *10* exemplaires et le système ne peut lui donner que *4*, P_i se contente (ie. se suffit) de ces ressources allouées et continue son exécution (sa requête est alors satisfaite). La politique de satisfaction de demandes de ressources doit être équitable (ie. FIFO).

1/ Décrire les structures principales à utiliser, expliciter les procédures nécessaires du moniteur avec leurs paramètres et donner la forme générale d'un processus.

2/ Réaliser la solution de gestion de cette classe de ressource.

Bon Courage

Correction de l'Epreuve Finale

Exercice n°1 : (6 pts= 2 + 2 + 2)

Répondre aux questions suivantes :

A- La 5ieme condition consiste à obliger le processus de sortir de sa section critique au bout d'un temps fini. On ne la considère pas étant donné qu'elle dépend du programmeur et non du protocole d'exclusion mutuelle.

B- Les sémaphores sont qualifiés d'outil de synchronisation de bas niveau car leur expression est proche du langage machine que de celui de l'utilisateur, ce qui mène facilement à des erreurs durant leurs utilisations.

C- Les trois types d'attente des processus dans les moniteurs classiques : blocage à l'accès au moniteur quand un autre processus est à l'intérieur du moniteur, blocage sur une condition, blocage lors du réveil d'un processus.

Exercice n°2 : (7 pts=4+3)

A/ Application

Var E : compteur d'événement :=0;

Processus *Pi;(i=1..10)*
Debut
-
advance(E, 1) ;
await(E, 10) ;
-
Fin ;

B/ *Var E : compteur d'événement :=0;*

Processus *Pi;(i=1..10)*
Debut
-
advance(E, 1) ;
await(E, 6) ;
-
Fin ;

Exercice n°3: (7pts= 2+5)

1/ Structures de données

Var c : condition; / condition fifo de blocage des processus en attente de ressources*
nbr: entier :=0; /* nombre d'exemplaires libres de la ressource*/ ***0,75 pts***

Deux entrées du moniteur :

- *entree procedure demander (k: entier, var nbacquit: entier) :* Permet de demander *k* ressources, *nbacquit* est le nombre, de ressources allouées effectivement, retourné au processus demandeur.
- *entree procédure liberer (r : entier):* Permet de libérer *r* ressources. ***0,75 pts***

Forme d'un processus

Processus Pi ;
var k, nbacquit, r : entier ;
-
M. demander (k, nbacquit);
<Utiliser ressources>
M. liberer (r) ;
-
Fin. **0,5 pts**

2/ Implémentation

M : moniteur ();
Const M=….. ;
Var c : condition ; nbr: entier; **0,25 pts**

entree procédure demander (k: entier, var nbacquit: entier) ;
Debut
Si (nbr=0)
Alors c.wait()
Fsi;
Si (nbr<=k)
Alors nbacquit :=nbr; nbr :=0
Sinon nbacquit :=k; nbr :=nbr-k
Fsi
Fin; **2,25 pts**

Entrée procedure libérer (r : entier) ;
Debut
nbr :=nbr+r;
Tant que non c.empty() et (nbr<>0) Faire c.signal() Fait
Fin ; **2,25 pts**

Initialisation
Debut
nbr :=M ; **0,25 pts**

Fin ;