SYSTÈMES ET RÉSEAUX INFORMATIQUES

COURS B4 : HTO (19339) et ICPJ (21937)
CYCLE PROBATOIRE INFORMATIQUE
(Conception et développement informatique)

EXAMEN DU 22 SEPTEMBRE 2005

portant sur l'enseignement des SYSTÈMES INFORMATIQUES

Date : jeudi 22 septembre 2005
Heure : 18h à 19h30 (suivi de l'examen de la partie réseaux de 19h30 à 21h)
Lieu : au CNAM, amphi 3, 2 rue Conté

TOUS DOCUMENTS PAPIERS AUTORISES

(Les portables téléphoniques, les ordinateurs et les calculatrices ne sont pas autorisés)

****

Énoncé de 5 pages et une annexe.

L'examen comprend :
quatre questions sur la gestion des ressources dont une question de cours
trois questions sur la synchronisation

Chaque question peut être traitée indépendamment.

Ne pas écrire au crayon ni à l'encre rouge
(Sous peine de nullité, selon le règlement des examens)

L'examen de septembre pour le cours Systèmes et réseaux informatiques comprend deux parties de 1h30 chacune, une en réseaux et une en systèmes. Chaque partie compte pour 50% de la note finale.

Barème *indicatif* pour la partie systèmes:

Question	QUESTION DE COURS	R1	R2	R3	S1	S2	S3
Notation	2	1	1	1	2	1	2

GESTION DES RESSOURCES : PRÉVENTION DE L'INTERBLOCAGE

QUESTION DE COURS SUR L'INTERBLOCAGE. Définir la notion d'état fiable et expliquer le principe de la prévention de l'interblocage et de l'algorithme du banquier comme application de l'état fiable. Préciser les conditions de l'utilisation de l'algorithme du banquier.

PROBLÈME DE GESTION DE RESSOURCES. Soit un système ayant un pool total de 150 cases de mémoire qui sont allouables dynamiquement. Pour chaque processus, on note son annonce (maximum du total de cases qu'il a le droit de cumuler par demandes successives), les cases qui lui sont allouées, la distance par rapport à l'annonce (soit, annonce - cases allouées).

R1 : ÉTAT1. Dans un premier état, ces cases ont été allouées à trois processus comme suit :

Processus	Annonce	Allocation	Distance
1	70	45	
2	60	40	
3	60	15	

QUESTION R1. Montrer que cet état est fiable, c'est à dire qu'en sérialisant leur exécution selon l'algorithme du banquier, l'allocateur pourrait servir tous les processus dans le cas extrême où ils demanderaient toutes les cases permises par leur annonce. Indiquer une séquence d'exécution qui permettrait d'éviter l'interblocage dans ce cas extrême.

R2 : ÉTAT2. Arrive un quatrième processus, avec une annonce de 60 et une première demande de 25 cases.

Processus	Annonce	Allocation	Distance
1	70	45	
2	60	40	
3	60	15	
4	60	25	

QUESTION R2. Le système peut-il accepter cette demande sans danger d'interblocage ? Si oui, indiquer une séquence d'exécution qui permettrait d'éviter l'interblocage dans le cas extrême où les processus demanderaient toutes les cases permises par leur annonce. Si non, indiquer l'allocation maximale que l'on pourrait faire à ce quatrième processus sans danger d'interblocage.

R3 : ÉTAT3. Le quatrième processus arrive cette fois avec une annonce de 60 et une première demande de 35 cases.

Processus	Annonce	Allocation	Distance
1	70	45	
2	60	40	
3	60	15	
4	60	35	

QUESTION R3. Le système peut-il accepter cette demande sans danger d'interblocage ? Si oui, indiquer une séquence d'exécution qui permettrait d'éviter l'interblocage dans le cas extrême où les processus demanderaient toutes les cases permises par leur annonce. Si non, indiquer l'allocation maximale que l'on pourrait faire à ce quatrième processus sans danger d'interblocage.

PROBLÈMES DE SYNCHRONISATION DES PROCESSUS

Un système d'acquisition et de présentation de mesures comporte 4 processus cycliques numérotés 1, 2, 3, 4 qui partagent une zone de données communes appelée « Blackboard » où ils stockent ou lisent les mesures.

Chaque processus X (avec X = 1, 2, 3 ou 4) se déroule selon le cycle :

```
loop
        attendre l'arrivée de données externes ; -- attente aléatoire
        lire ces données externes ;
        demande d'accès au Blackboard ; -- en appelant Controle.Entree(X)
        écritures et lectures dans le Blackboard ;
        fin d'accès au Blackboard ; -- en appelant Controle.Sortie(X)
        préparation de la présentation des données ;
        présentation interactive des données ; -- durée aléatoire
end loop ;
```

Pour assurer la cohérence des données du Blackboard, on examine plusieurs méthodes différentes pour programmer le paquetage Controle qui contrôle l'accès au Blackboard :

```
package Controle is
        procedure Entree(X : in Integer) ;
        procedure Sortie(X : in Integer) ;
end Controle;
```

MÉTHODE AVEC CHACUN SON TOUR. On agit sur l'ordre d'exécution des processus de telle façon que les processus accèdent au Blackboard chacun à son tour dans un ordre fixe : d'abord le processus 1, puis le processus 2, puis le processus 3, puis le processus 4, puis on recommence avec le processus 1, etc. A l'origine tous les processus X qui appellent Controle.Entree(X) sont bloqués sauf le processus 1. Quand le processus 1 a fini son accès et qu'il le signale par Controle.Sortie(1), ceci a pour effet d'autoriser l'accès pour le processus 2. Quand le processus 2 a fini son accès et qu'il le signale par Controle.Sortie(2), ceci a pour effet d'autoriser l'accès pour le processus 3, etc…Chaque processus, en exécutant Controle.Sortie(X), envoie donc un signal de continuation au processus suivant.

QUESTION S1. Compléter la programmation du package body de Controle

```
package body Controle is
        S : array(1..4) of Semaphore ; -- S(1) sert à bloquer le processus 1, S(2) le processus 2, etc…
        procedure Entree(X : in Integer) is begin …  end Entree;
        procedure Sortie(X : in Integer) is begin …  end Sortie;
begin
        -- initialisation des sémaphores
end Controle;
```

MÉTHODE AVEC ORDRE QUELCONQUE. Dans cette méthode on ne se préoccupe plus du tout de l'ordre dans lequel les processus font accès au Blackboard, mais seulement de la cohérence du Blackboard en obligeant chaque processus à être seul pendant l'accès au Blackboard. Les procédures Entree(X) et Sortie(X) se chargent d'imposer cette unicité d'accès. Si il y a plusieurs demandes d'accès concurrentes, on accepte que l'ordre de l'accès soit déterminé par la file d'attente du sémaphore utilisé.

QUESTION S2. Compléter la programmation du package body de Controle

```
package body Controle is
        Acces : Semaphore ; -- Acces sert à garantir qu'un seul processus puisse faire l'accès au Blackboard
        procedure Entree(X : in Integer) is begin …  end Entree;
        procedure Sortie(X : in Integer) is begin …  end Sortie;
begin
        -- initialisation du sémaphore
end Controle;
```

MÉTHODE AVEC ORDRE QUELCONQUE APRÈS RENDEZ-VOUS. Cette méthode, comme la méthode avec ordre quelconque, impose seulement qu'il n'y ait qu'un seul processus à la fois accédant au Blackboard et en cas de demandes concurrentes se satisfait de l'ordre d'attente fixé par la file d'attente du sémaphore, mais elle impose une certaine équité entre les processus en empêchant un processus de faire un nouvel accès avant que les 3 autres processus aient pu faire leur accès. Pour cela on impose aux processus de commencer la procédure Entree par un rendez-vous entre tous avant d'autoriser un nouvel accès.

QUESTION S3. Compléter la programmation du package body de Contrôle

```
package body Controle is

        Sempriv : array(1..4) of Semaphore ; -- Sempriv(X) sert à bloquer le processus X
        Compteur : Integer := 0 ; -- compte les arrivées au rendez-vous
        Mutex_Compteur : Semaphore ; -- assure la cohérence  du compteur
        Acces : Semaphore ; -- Acces sert à garantir qu'un seul processus puisse faire l'accès au Blackboard

        procedure Entree (X : in Integer) is
        begin
                -- assurer la cohérence du comptage
                Compteur := Compteur + 1 ;
                if Compteur = 4 then
                        Compteur := 0 ;
                        -- autoriser tous les 4 processus
                end if ;
                -- fin de cohérence de comptage
                bloquer éventuellement le processus X appelant
                contrôle pour l'accès un seul processus à la fois ;
        end Entree;

        procedure Sortie(X : in Integer) is
        begin
                contrôle de sortie et réveil d'un processus bloqué
        end Sortie;

begin
        -- initialisation des sémaphores
end Controle;
```