SYSTÈMES ET RÉSEAUX INFORMATIQUES

COURS B4 : HTO(19339) et ICPJ(21937)

EXAMEN DU 14 SEPTEMBRE 2004

portant sur l'enseignement des SYSTÈMES INFORMATIQUES

SOLUTIONS

SYNCHRONISATION DES PROCESSUS

Réponse à la question S1

S1 a) package body **Synchro** is

```
        Signal : array(Id_Cam) of Semaphore ;
        procedure Attendre(X : in Id_Cam) is begin P(Signal(X)); end Attendre ;
        procedure Signaler(X : in Id_Cam) is begin V(Signal(X)); end Signaler;
begin for I in Id_Cam loop E0(Signal(I), 0) ; end loop ;
end Synchro;
```

S1 b) package **Ressource** is

```
        Mutex : array(Id_Cam) of Semaphore ;
         procedure Demander(X : in Id_Cam) is begin P(Mutex(X)); end Demander;
         procedure Rendre(X : in Id_Cam) is begin V(Mutex(X)); end Rendre;
begin for I in Id_Cam loop E0(Mutex(I), 1) ; end loop ;
end Ressource;
```

S1 c) Serveur(2) a pris les caméras 6 et 1 et attend la caméra 2 occupée par Serveur(0)
Serveur(1) a pris les caméras 4 et 5 et attend la caméra 6 occupée par Serveur(2)
Serveur(0) a pris la caméra 2 et attend la caméra 4 occupée par Serveur(1)
On a une attente circulaire et interblocage

S1 d) Méthodes : classes ordonnées : demander les caméras selon leur ordre de numérotation. CEla donnerait :

a) *Le Serveur(0) demande les caméras 1, 2, 4*
b) *Le Serveur(1) demande les caméras 4, 5, 6*
c) *Le Serveur(2) demande les caméras 1, 2, 6*

Autre méthode : demandes globales. A programmer

Réponse à la question S2

S2 a)

```
package Depot is
        type Id_Depot is mod 6 ;
        Tete, Queue : mod 6 := 0 ;
        T : array (Id_Depot) of Image ;
        Mutex, Nplein, Nvide : Semaphore ;
        Notification : array(Id_Cam) of Semaphore ;
        procedure Envoyer(X : in Id_Cam ; Y : in Image) is
        begin
                P(Nvide) ;
                P(Mutex) ;
                T(Queue) := Y ;
                Queue := Queue + 1 ;
                V(Mutex) ;
                V(Nplein) ;
                V(Notification(X)) ;
        end Envoyer ;
```

```
        procedure Retirer(X : in Id_Cam ; Y : out Image) is
        begin
                P(Notification(X)) ;    -- à faire avant P(Nplein) sinon on bloque les autres serveurs
                P(Nplein) ;
                P(Mutex) ;              -- une seule exclusion mutuelle car retrait fait accès à tout
                Retrait(X, Y) ;         -- retire Y et laisse une case vide devant Queue
                Queue := Queue – 1 ;
                V(Mutex) ;
                V(Nvide) ;
begin
        E0(Mutex, 1) ; E0(Nplein, 0) ; E0(Nvide, 6) ;
        for I in Id_Cam loop E0(Notification(I), 0) ; end loop ;
end Depot;
```

b) En signalant l'envoi par X, on signale aussi l'envoi d'une nouvelle image. Il est inutile de le faire à l'ensemble des consommateurs. Nplein devient redondant car il représente le nombre total des notifications et celui-ci est égal à la somme des notifications partielles. On peut supprimer Nplein, P(Nplein) et V(nplein).

c) On remarquera que Tete n'est lu que par Extraire et jamais modifié. Il vaut toujours 0. On pourrait le supprimer et, dans Extraire, initialiser I à 0.

REMPLACEMENT DE PAGE

Réponse à la question R1

Question a) A avec LRU : 3 chargements + 4 remplacements

référence	2	3	5	2	1	5	2	4	5	3	2	5	2	3
état	2	2	2	3	5	2	1	5	2	4	5	3	3	5
après		3	3	5	2	1	5	2	4	5	3	2	5	2
chargement			5	2	1	5	2	4	5	3	2	5	2	3
remplacement					oui			oui		oui	oui			

Question a) B avec FIFO : 3 chargements + 6 remplacements

référence	2	3	5	2	1	5	2	4	5	3	2	5	2	3
état	2	2	2	2	3	3	5	1	2	4	5	5	5	5
après		3	3	3	5	5	1	2	4	5	3	3	3	3
chargement			5	5	1	1	2	4	5	3	2	2	2	2
remplacement					oui		oui	oui	oui	oui	oui			

Question b) Avec OPT , 3 chargements + 3 remplacements

référence	2	3	5	2	1	5	2	4	5	3	2	5	2	3
état	2	3	3	3	1	1	1	4	4	3	2	5	5	3
après		2	5	2	2	5	2	2	5	5	3	3	2	2
chargement			2	5	5	2	5	5	2	2	5	2	3	5
remplacement					oui			oui		oui				

Réponse à la question R2

a) : 1 ; b) : 2 ; c) : 3