SYSTÈMES ET RÉSEAUX INFORMATIQUES

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

EXAMEN DE JUIN 2003
SOLUTIONS

SOLUTIONS POUR A. COOPÉRATION DE PROCESSUS

Premier cas : Faux, on lit un message qui n'existe pas

Deuxième cas : correct

Troisième cas : faux. Blocage d'attente dans la section critique. Interblocage car il faut qu'un processus puisse entrer en section critique pour débloquer celui qui s'y trouve déjà, et c'est impossible. Solution: mettre les operations P( ) avant la section critique

Quatrième cas : correct si un seul producteur. Faux si plusieurs producteurs. Par exemple Prod1 prend le numéro 1, Prod2 prend le numéro 2, dépose son message et fait V(Nplein) avant prod1. Le consommateur peut alors lire la case 1 qui ne contient pas encore de message .

Solution. Mettre T(Numero modulo N) := X; dans la section critique

SOLUTIONS POUR B. SERVEUR DE SIGNATURES D'IMAGES

a) durée d'une requête = 60 + 20 + 60 = 140 pour 20 ms d'UC

Taux à adopter = 7

b) M = 6*2 +1 = 13

c) package body Zone is

Disponible : array(1..M) of Boolean := (others => True); -- indique les zones non allouées

S_Z, Mutex : Semaphore;

procedure Allouer(I: out IdZone) is

begin

P(S_Z); I := 1;

P(Mutex); loop exit when Disponible(I); I := I + 1; end loop; Disponible(I) := False; V(Mutex);

end Allouer;

procedure Restituer(J : in IdZone) is

begin

P(Mutex); Disponible(J) := true; V(Mutex); V(S_Z);

end Restituer;

begin.E0(S_Z, M); E0(Mutex, 1); end Zone;

SOLUTIONS POUR C. SERVEUR AVEC TROIS DISQUES

a) package Disque is

```
    S : Semaphore;
    procedure Obtenir is begin P(S); end Obtenir;
    procedure Rendre is begin V(S); end rendre;
```

begin E0(S, 1); end Disque;

Interblocage si :

Serveur 12 a obtenu le Disque1 et attend Disque2,

Serveur 23 a obtenu le Disque2 et attend Disque3,

Serveur 31 a obtenu le Disque3 et attend Disque1,

b) pas d'interblocage car allocation globale, et restitution globale mais on n'a plus de concurrence.

Famine si la file d'attente du sémaphore Controle.S contient toujours plusieurs processus au moment du réveil par Controle.V(S). Un processus peu prioritaire peut ne jamais être choisi.

c) On évite l'interblocage entre les serveurs 12, 23, 31 (cf solution des philosophes assis), mais les serveurs 1, 2 et 3 peuvent se coaliser pour empêcher les serveurs 12, 23, 31 de dépasser Controle.Assurer.

Solution possible. Dans l'implémentation historique de sémaphores (celle de Dijkstra, celle du cours), si V(S) réveille un processus bloqué avant d'autoriser un nouvel appel de P(S) -- c'est l'ancêtre du modèle de l'oeuf pour l'objet protégé de ADA-- on peut éviter la famine si il n'y a pas plus d'un processus bloqué dans une file de sémaphore.

Il vient :

```
package body Controle  is
    S2, S3, S4, S5 : Semaphore;
    procedure Assurer is begin P(S5); P(S4); P(S3); P(S2); end Assurer;
    procedure Relacher is begin  V(S2); V(S3); V(S4); V(S5);end Relacher
begin
    E0(S2,2); E0(S23,3); E0(S4,4); E0(S5,5);
end Controle;
```

SOLUTIONS POUR D. ACCÈS DISQUES

a) On omet le délai rotationnel, le délai de transfert et le temps de lancement et de terminaison du processus. Mais pour comparer les solutions, on doit prendre ce qui peut varier d'une organisation à l'autre. Ici c'est seulement le délai rotationnel. On néglige donc 4ms devant 40 ms donc on a une marge d'erreur de l'ordre de 10%.

b) **Première politique**. On exécute les requêtes en séquence selon les priorités.

numéro de la requête	source, init 80	déplacements disque source	puit, init 80	déplacements disque puit
1	70	10	10	70
2	0	70	20	10
3	50	50	70	50
4	90	40	80	10
5	40	50	40	40
6	60	20	90	50
total		240		230

total : 470

Deuxième politique. On sert les requêtes en séquence en appliquant la politique de l'ascenseur au disque "source".

numéro de la requête	source, init 80	déplacements disque source	puit, init 80	déplacements disque puit
4	90	10	80	0
1	70		10	70
6	60		90	80
3	50		70	20
5	40		40	30
2	0	90	20	20
total		100		220

total : 320

Troisième politique. On sert les requêtes en séquence en appliquant la politique de l'ascenseur au disque "puit".

numéro de la requête	source, init 80	déplacements disque source	puit, init 80	déplacements disque puit
4	90	10	80	
6	60	30	90	10
3	50	10	70	
5	40	10	40	
2	0	40	20	
1	70	70	10	80
total		170		90

total : 260

Quatrième politique. On sert les requêtes en examinant la distance totale de déplacement pour une requête (déplacement sur le disque "source" + déplacement sur le disque "puit") et on élit la requête ayant la distance totale la plus courte. avec cette politique, on commence par servir la requête n° 4, puis la requête n° 6, etc... Réorganiser les requêtes pour cela et calculer :

numéro de la requête	source, init 80	déplacements disque source	puit, init 80	déplacements disque puit
4	90	10	80	0
6	60	30	90	10
3	50	10	70	20
5	40	10	40	30
2	0	40	20	20
1	70	70	10	10
total		170		90

total : 260