SYSTÈMES ET RÉSEAUX INFORMATIQUES
CORRIGÉ DE L'EXAMEN DU 22 SEPTEMBRE 2005

portant sur l'enseignement des SYSTÈMES INFORMATIQUES

Réponses aux questions

GESTION DES RESSOURCES : PRÉVENTION DE L'INTERBLOCAGE

QUESTION DE COURS SUR L'INTERBLOCAGE

La prévention de l'interblocage vise à empêcher la venue d'un interblocage en imposant un ordre d'allocation des ressources (méthode des classes ordonnées ou méthode de l'estampillage des processus) ou en imposant des restrictions sur l'allocation (algorithme du banquier).

L'algorithme du banquier impose aux processus, en nombre connu, de faire une annonce contractuelle préalable, ce qui permet à l'allocateur de vérifier avant de servir une nouvelle requête si ce service laisse le système dans un état fiable, c'est à dire un état où il n'y a pas de risque d'interblocage si l'allocateur continue à appliquer l'algorithme du banquier. Dans un état fiable, l'allocateur conserve la possibilité de répondre à toutes les demandes futures autorisées par l'annonce contractuelle, en bloquant éventuellement les processus et les retardant de façon à les servir en séquence, ne réveillant un nouveau processus que lorsque le précédent a rendu toutes les ressources qui lui ont été allouées.

Conditions d'utilisation de l'algorithme du banquier :

- connaître le nombre des processus du système et l'annonce (contrat indiquant le maximum du total de cases qu'il a le droit de cumuler par demandes successives) de chacun d'eux
- allocation dynamique, allocations successives dans le respect de cette annonce
- pas d'ordre fixe d'allocation imposé à priori, mais un ordre éventuel résultant de l'algorithme
- refus de service possible si l'allocation peut conduire à un état non fiable
- restitution partielle ou totale des ressources, pas de contrainte de restitition, pas de durée max d'allocation, sinon de rendre les ressources au bout d'un temps fini,
- pas de rétention définitive des ressources
- pas de destruction ou panne de ressources. Une perte de ressource oblige à réexaminer si l'état courant est toujours un état fiable après cette perte de ressource. Si l'état n'est pas fiable, on n'a plus la prévention de l'interblocage et pour éviter l'interblocage si c'est possible, il faut un dialogue avec les processus pour réexaminer leur contrat. Si l'état reste fiable, tout va bien et on tolère cette panne.

PROBLÈME DE GESTION DE RESSOURCES.

RÉPONSE R1. Il reste 50 cases en stock. Toutes les distances (25, 20, 45) sont inférieures à ce stock de 50. On peut 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 : toutes les séquences sont fiables car toutes les annonces des processus sont accessibles avec ce stock. Il n'y a pas de danger d'interblocage si le contrat défini par l'annonce est respecté.

RÉPONSE R2. Il resterait 25 cases en stock. Les distances sont (25, 20, 45, 35). Toute séquence commençant par P1 ou P2 sont valables car après l'exécution de P1 (resp. P2) le stock deviendrait de 70 (resp. 65) ce qui suffirait pour servir les autres processus jusqu'à leur annonce. Cette demande peut être acceptée car le système restera fiable, sans interblocage.

RÉPONSE R3. Il resterait 15 cases en stock. Les distances sont (25, 20, 45, 25). Aucun processus ne pourrait être servi dans le cas extrême où l'un d'eux demanderait toutes les cases permises par son annonce. L'allocation maximale pour le quatrième processus est de 30 cases sans danger d'interblocage. Le résidu est alors de 20 cases, les distances de (25, 20, 45, 30). Toute séquence commençant par P1 est fiable.

PROBLÈMES DE SYNCHRONISATION DES PROCESSUS

MÉTHODE AVEC CHACUN SON TOUR. RÉPONSE S1

```
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 P(S(X)) ; end Entree;
        procedure Sortie(X : in Integer) is
        begin
                if X = 4 then V(S(1)); else V(S(X + 1)) ; end if ;
        end Sortie;
begin
        -- initialisation des sémaphores
        E0(S(1), 1) ; for I in 2..4 loop E0(S(I), 0) end loop ;
end Controle;
```

On n'a pas besoin d'un semaphore d'exclusion mutuelle, car celle-ci découle du service l'un après l'autre.

MÉTHODE AVEC ORDRE QUELCONQUE. RÉPONSE S2.

On installe un sémaphore d'exclusion mutuelle classique

```
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 P(Acces) ; end Entree;
        procedure Sortie(X : in Integer) is begin V(Acces) ; end Sortie;
begin
        -- initialisation du sémaphore
        E0(Acces, 1) ;
end Controle;
```

MÉTHODE AVEC ORDRE QUELCONQUE APRÈS RENDEZ-VOUS. RÉPONSE S3.

```
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 ait l'accès au Blackboard
        procedure Entree (X : in Integer) is
        begin
                P(Mutex_Compteur) ; -- assurer la cohérence du comptage
                Compteur := Compteur + 1 ;
                if Compteur = 4 then
                        Compteur := 0 ;
                        -- autoriser tous les 4 processus
                        for I in 1..4 loop V(Sempriv(I)) ; end loop ;
                end if ;
                V(Mutex_Compteur) ; -- fin de cohérence de comptage
                P(Sempriv(X)) ; -- bloquer éventuellement le processus X appelant
                P(Acces) ; -- contrôle pour l'accès un seul processus à la fois = exclusion mutuelle
        end Entree;
        procedure Sortie(X : in Integer) is
        begin V(Acces) ; end Sortie; -- contrôle de sortie et réveil d'un processus bloqué
begin
        -- initialisation des sémaphores
        E0(Mutex_Compteur, 1) ; E0(Acces, 1) ; for I in 1..4 loop E0(Sempriv(I), 0) ; end loop ;
end Controle;
```