SYSTÈMES INFORMATIQUES ET APPLICATIONS CONCURRENTES

COURS B4 : HTO (NFP137) et ICPJ (NFP137J)

EXAMEN DU 24 JUIN 2006

Date : samedi 24 juin 2006
Heure : 9h00 à 12h00
Lieu : au CNAM, rue Conté, salle 30.-1.02

TOUS DOCUMENTS PAPIERS AUTORISES

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

****

Énoncé de 5 pages.

L'examen comprend :
quatre questions sur la gestion des ressources
trois questions sur la synchronisation complétées par trois questions sur des ressources

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)

****

Barème indicatif

Question	R1	R2	R3	R4	S1	S2	S3	S4	S5	S6
Notation	2	1	2	1	3	4	1	2	2	2

A. Gestion des ressources

Un système comprend 12 processus qui se partagent un stock de 36 ressources banalisées qu'ils demandent dynamiquement. Chaque processus est un interpréteur de commandes et selon la commande reçue fait une des deux suites de requêtes suivantes :

A : commande de lecture :: demande 3 ressources ; utilisation ; demande 2 ressources supplémentaires ; utilisation ; demande 1 ressource supplémentaire ; utilisation ; restitution des 6 ressources.

B : commande de mise à jour :: demande 3 ressources ; utilisation ; demande 3 ressources supplémentaires ; utilisation ; demande 2 ressources supplémentaires ; utilisation ; demande 1 ressource supplémentaire ; utilisation ; restitution des 9 ressources.

QUESTION R1. Au démarrage du système, celui-ci reçoit les premières requêtes (demande de 3 ressources) de chacun des 12 processus. On ne sait pas quelles commandes (lecture ou mise à jour) vont être demandées. On suppose le pire pour l'allocateur. Celui-ci applique l'algorithme de prévention d'interblocage dit algorithme du banquier. Combien de processus peut-il servir sans crainte d'interblocage ? Expliquer votre réponse.

QUESTION R2. On veut augmenter le stock pour que l'on ait assez de ressources pour ne jamais avoir d'interblocage et donc pour ne pas avoir à utiliser l'algorithme du banquier. On ne sait toujours pas quelles commandes (lecture ou mise à jour) vont être demandées. On suppose encore le pire. Quel est le nombre minimal de ressources qui permette de garantir l'absence d'interblocage. Expliquer votre réponse.

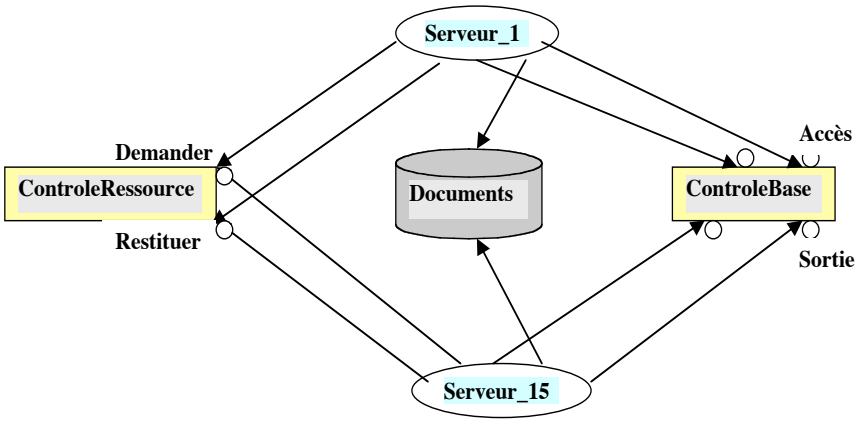
QUESTION R3. Au démarrage du système, celui-ci reçoit les premières requêtes (demande de 3 ressources) de chacun des 12 processus. On configure le système pour qu'il y ait toujours 6 commandes de lecture et 6 commandes d'écriture. L'allocateur applique l'algorithme du banquier. Combien de processus peut-il servir sans crainte d'interblocage ? Expliquer votre réponse.

QUESTION R4. On veut augmenter le stock pour que l'on ait assez de ressources pour ne jamais avoir d'interblocage et donc pour ne pas avoir à utiliser l'algorithme du banquier. On configure le système pour qu'il y ait toujours 6 commandes de lecture et 6 commandes d'écriture. Quel est le nombre minimal de ressources qui permette de garantir l'absence d'interblocage. Expliquer votre réponse.

B. Synchronisation des processus

1. Présentation et schéma des processus

Un gestionnaire d'accès à une base de données documentaire permet des opérations de consultation ou de mise à jour. Son système comprend 15 processus serveurs qui ont des interactions par deux objets partagés, l'un pour l'allocation de ressources banalisées (ce sont, par exemple, des tampons de mémoire servant pour l'accès au disque contenant la base de données), l'autre pour le contrôle d'accès à la base. On va comparer plusieurs versions de système qui vont différer selon l'ordre d'utilisation de ces deux objets partagés.



1.2. Le contrôle d'accès à la base documentaire

L'accès à cette base documentaire est contrôlé par l'objet partagé ControleBase. Quand l'accès est fait pour des consultations concurrentes, on donne comme contrainte additionnelle qu'au maximum 10 processus puissent être simultanément en consultation. Le contrôle se fait en utilisant les appels : AccesConsultation, puis SortieConsultation. Quand il se fait pour des mises à jour cohérentes, on utilise les appels : AccesMiseAJour, puis SortieMiseAJour

```
package ControleBase is
    procedure AccesConsultation ;
    procedure SortieConsultation ;
    procedure AccesMiseAJour ;
    procedure SortieMiseAJour ;
end ControleBase;
package body ControleBase is
    -- déclaration du sémaphore de régulation pour garantir qu'il n'y a pas plus de 10 consultations simultanément
    -- déclaration des sémaphores de contrôle des accès
    -- déclaration et initialisation des variables rémanentes utilisées pour ce contrôle
    procedure AccesConsultation is
    begin
        -- régulation et blocage si plus de 10 accès
        -- contrôle de l'accès en consultation et blocage s'il y a en cours un accès en mise à jour
    end AccesConsultation ;
    procedure SortieConsultation  is
    begin
        -- sortie de l'utilisation en consultation et gestion des conséquences de cette sortie
        -- régulation des 10 accès simultanés et réveil éventuel d'un processus en attente
    end SortieConsultation ;
    procedure AccesMiseAJour  is
    begin
        -- contrôle de l'accès en mise à jour, respectant la cohérence des consultations et des mises à jour
    end AccesMiseAJour;
    procedure SortieMiseAJour
    begin
        -- sortie de l'utilisation en mise à jour et gestion des conséquences de cette sortie
    end SortieMiseAJour ;
begin
    -- initialisation des sémaphores et des variables de contrôle
end ControleBase;
```

QUESTION S1. Programmer le paquetage ControleBase en utilisant des sémaphores et les données de contrôle nécessaires. Ne pas oublier d'initialiser les sémaphores et les données de contrôle. Ne pas les initialiser laisse le contrôle imprécis et c'est compté comme une erreur.

1.3. Le contrôle d'accès aux ressources

On dispose d'un stock de ressources banalisées (ce sont des tampons de mémoire) dans lequel puisent les serveurs pour pouvoir traiter les demandes des utilisateurs et assurer le transfert avec les données de la base documentaire. Pour savoir s'il peut prendre X ressources, un serveur appelle l'objet partagé ControleRessource en utilisant l'appel *Demander(X)*. Puis quand il a rendu Y ressources, il appelle *Restituer(Y)* pour gérer les conséquences de ce retour.

Les processus demandeurs sont traités l'un après l'autre. Si un processus ne peut être traité immédiatement et doit attendre que des ressources soient restituées, aucun autre processus ne peut être traité avant lui, même s'il y a assez de ressources pour sa demande.

Chaque processus demandeur commence par s'assurer qu'il sera bien seul à être traité. S'il est traité immédiatement, il autorise un autre processus à demander des ressources. Sinon, il se met en attente d'arrivée de ressources rendues par *Restituer*. Dans *Restituer*, s'il y a un processus en attente et pouvant être servi, ce processus est réveillé, et en plus on autorise le service d'un autre processus (et d'un seul) qui a appelé *Demander*.

Comme il n'y a qu'un seul processus appelant bloqué en attente de retour de ressources, on peut n'avoir qu'un seul sémaphore de blocage au lieu d'un tableau de sémaphores privés. Il vient alors.

```
package ControleRessource is
    procedure Demander(X : in Integer);
    procedure Restituer(Y : in Integer);
end ControleRessource;
```

```
package body ControleRessource is
    Stock : Integer := StockInitial; -- variable rémanente partagée, donnée de contrôle
    EnAttente : Boolean := False; -- variable rémanente partagée, donnée de contrôle
    S_Solo, S_Patience : Semaphore; -- pour garantir l'unicité du client servi, et pour le faire attendre si nécessaire
    -- déclaration des autres sémaphores nécessaires
    procedure Demander(X : in Integer) is
        OK : boolean := False; -- variable non partagée, créée à chaque appel de cette procédure, détruite au retour de l'appel
    begin
        Assurer avec S_Solo que l'on ne sert bien chaque client l'un après l'autre
        Assurer la cohérence des données rémanentes de contrôle et le non blocage en section critique
        Stock := Stock – X; -- le résultat est négatif s'il n'y a pas assez de ressources en Stock
        if Stock < 0 then
            EnAttente:= True; OK := False;
        else
            EnAttente:= False; OK := True;
        end if;
        if OK then
            Autoriser un autre client à examiner sa demande
        else
            Mettre le client demandeur en attente de retour de ressources
        end if;
    end Demander;
    procedure Restituer(Y : in Integer) is
    begin
        Assurer la cohérence des données rémanentes de contrôle
        Stock := Stock + Y;
        if EnAttente and Stock >= 0 then
            EnAttente := False;
            Réveiller le processus en attente de retour de ressource
            Autoriser un autre client à examiner sa demande dans Demander
        end if;
    end Restituer;
begin
    -- initialisation des sémaphores et des variables de contrôle
end ControleRessource;
```

QUESTION S2. Programmer le paquetage ControleRessource en utilisant des sémaphores et les données de contrôle nécessaires. Penser à assurer la cohérence des données rémanentes de contrôle et à éviter le blocage en section critique Ne pas oublier d'initialiser les sémaphores et les données de contrôle. Ne pas les initialiser laisse le contrôle imprécis et c'est compté comme une erreur.

QUESTION S3. Dans le schéma des sémaphores privés, un processus bloqué sur son sémaphore privé peut être réveillé, aussi bien dans la section critique où la décision de réveil est prise, qu'après la sortie de cette section critique ; cette dernière programmation permet plus de concurrence en libérant plus tôt l'exclusion mutuelle. Dans la question précédente, on n'a utilisé qu'un seul sémaphore de blocage au lieu d'un tableau de sémaphores privés parce qu'on n'a qu'un seul serveur à bloquer à ce niveau. Si dans *Restituer* on reporte le réveil de ce processus serveur après la sortie de section critique, cette solution pourrait être incorrecte. Expliquer pourquoi. Dans quelle condition de programmation, cette optimisation pourrait-elle être correcte ?

2. Architecture et paradigmes

On configure le système pour qu'il y ait 12 serveurs de consultation et 3 serveurs de mise à jour. Un serveur de consultation peut avoir besoin de 1 à 6 ressources pour un accès en consultation et un serveur de mise à jour peut avoir besoin de 1 à 8 ressources pour faire une mise à jour.
On va comparer des architectures utilisant ces deux objets partagés (« pattern architectural design »).

2.1. Les ressources acquises avant la base

Chaque serveur demande ses ressources (éventuellement une à une) avant de demander l'accès à la base documentaire. Cela donne.

```
task type ServeurConsultation; task type ServeurMiseAJour
task body ServeurConsultation is
begin
    loop
        AttendreConnexion(Consultation);
        PriseDeRessourceEtPreparationDeCetAcces;
        AccesAlaBaseDocumentaireEnLecture;
        UtilisationDeLaBaseEnConsultation; -- on n'autorise que 10 processus simultanément en consultation
        SortieDeLaBaseDocumentaireEnLecture;
        LibérationDesRessources;
    end loop;
end ServeurConsultation;
task body ServeurMiseAJour is
begin
    loop
        AttendreConnexion(MiseAJour);
        PriseDeRessourceEtPreparationDeCetAcces;
        AccesAlaBaseDocumentaireEnMiseAJour;
        UtilisationDeLaBaseEnMiseAJour;
        SortieDeLaBaseDocumentaireEnMiseAJour;
        LibérationDesRessources;
         end if;
    end loop;
end ServeurMiseAJour;
```

QUESTION S4. On veut avoir assez de ressources pour ne jamais tomber en interblocage, quelles que soient les opérations faites par les serveurs. Quelle est pour cela la valeur minimale à donner à StockInitial, le nombre initial de ressources ? Expliquer votre réponse.

2.2. La base acquise avant les ressources

Chaque serveur demande l'accès à la base documentaire avant de demander ses ressources (éventuellement une à une). Cela donne.

```
task type ServeurConsultation; task type ServeurMiseAJour
task body ServeurConsultation is
begin
    loop
        AttendreConnexion(Consultation);
        AccesAlaBaseDocumentaireEnLecture;
        PriseDeRessourceEtPreparationDeCetAcces;
        UtilisationDeLaBaseEnConsultation; -- on n'autorise que 10 processus simultanément en consultation
        SortieDeLaBaseDocumentaireEnLecture;
        LibérationDesRessources;
    end loop;
end ServeurConsultation;
task body ServeurMiseAJour is
begin
    loop
        AttendreConnexion(MiseAJour);
        AccesAlaBaseDocumentaireEnMiseAJour;
        PriseDeRessourceEtPreparationDeCetAcces;
        UtilisationDeLaBaseEnMiseAJour;
        SortieDeLaBaseDocumentaireEnMiseAJour;
        LibérationDesRessources;
    end loop;
```

```
end ServeurMiseAJour;
```
QUESTION S5. On veut avoir assez de ressources pour ne jamais tomber en interblocage, quelles que soient les opérations faites par les serveurs. Quelle est pour cela la valeur minimale à donner à StockInitial, le nombre initial de ressources ? Expliquer votre réponse. Quelle valeur minimale faut-il donner à StockInitial pour qu'on ait à la fois l'absence d'interblocage et la concurrence maximale entre serveurs (avec la programmation indiquée ci-dessus) ?

2.3. Architecture mixte

On décide d'expérimenter une architecture mixte : quand un serveur fait une mise à jour, il demande d'abord les ressources selon son besoin ; quand il s'agit d'une consultation, le serveur demande d'abord l'accès à la base documentaire. Il vient .

```
task type ServeurConsultation; task type ServeurMiseAJour
task body ServeurConsultation is – avec accès à la base avant prise de ressources
begin
    loop
        AttendreConnexion(Consultation);
        AccesAlaBaseDocumentaireEnLecture;
        PriseDeRessourceEtPreparationDeCetAcces;
        UtilisationDeLaBaseEnConsultation; -- on n'autorise que 10 processus simultanément en consultation
        LibérationDesRessources;
        SortieDeLaBaseDocumentaireEnLecture;
    end loop;
end ServeurConsultation;
task body ServeurMiseAJour is – avec prise de ressources avant l'accès à la base
begin
    loop
        AttendreConnexion(MiseAJour);
        PriseDeRessourceEtPreparationDeCetAcces;
        AccesAlaBaseDocumentaireEnMiseAJour;
        UtilisationDeLaBaseEnMiseAJour;
        SortieDeLaBaseDocumentaireEnMiseAJour;
        LibérationDesRessources;
         end if;
    end loop;
end ServeurMiseAJour;
```

QUESTION S6. Analyser cette architecture sous divers points de vue système
-- règlage de StockInitial, comme dans les questions précédentes pour éviter l'interblocage dans l'allocation des ressources, et choix d'une valeur minimale.
-- taux de concurrence entre serveurs
-- danger d'interblocage à cause de l'utilisation des deux d'objets partagés dans un ordre différent ?
Dans tous les cas expliquer les réponses. Une réponse non expliquée n'est pas acceptée (c'est considéré comme une réponse faite au hasard).