SYSTEMES ET RESEAUX INFORMATIQUES

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

EXAMEN DU 17 JUIN 2000

partie portant sur l'enseignement des SYSTÈMES INFORMATIQUES

Date : mardi 17 juin 2000
Durée : 3 heures
Heure : 14 H à 17 H
Lieu : Salles 30 -1 02 et 30 -1 04

TOUS DOCUMENTS AUTORISES

L'examen comprend deux problèmes indépendants :

LE JEU SIBER (14 points)
Une **annexe** est fournie pour les réponses aux questions 4, 5, 6; ne pas oublier de reporter sur chaque page de l'annexe, le numéro de votre copie (ne mettez ni votre nom, ni votre numéro de carte CNAM).

GESTION D'UN CACHE (6 points)

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

PREMIER PROBLÈME : **LE JEU SIBER** (14 points)

SIBER est un jeu réparti qui peut être joué à 100 personnes. Chaque joueur fait autant de sessions qu'il veut. Une session est une suite de 4 coups exécutés par un joueur et qu'il envoie à son rythme. A chaque coup, le joueur envoie un objet au maître du jeu et attend le feu vert pour le coup suivant. À la fin des 4 coups (c'est à dire à la fin d'une session), le joueur reçoit un score.

Le maître du jeu utilise un tableau T de 200 cases, pour stocker les objets envoyés à chaque coup par les joueurs en session. Pendant une session, il y a dans T de 0 à 4 objets par joueur. En fin de session, le maître calcule le score, stocke les 4 objets sur disque et libère les 4 cases utilisées dans le tableau T.

Le maître comporte 100 processus Serveur(i) concurrents, un par joueur potentiel. Ces processus dialoguent avec les joueurs, attendent les objets, les déposent dans le tableau T, les récupèrent et les envoient sur disque.

Le programme des processus Serveurs est donné en page 5 de l'énoncé.

1. ALLOCATION DYNAMIQUE DES CASES DU TABLEAU T

Les cases du tableau T sont allouées une à une aux serveurs, qui ont besoin de quatre cases pour terminer une session.

Question 1 (1pt)

Montrer qu'il peut y avoir interblocage entre les processus serveurs qui se partagent dynamiquement les 200 cases du tableau T.

****

Pour éviter l'interblocage, on met en place un algorithme de prévention dynamique, dit du banquier, qui se simplifie ici parce qu'il n'y a qu'une classe de 200 ressources banalisées et parce que toutes les annonces sont égales.

Question 2 (2pts)

Rappeler le principe de l'algorithme simplifié. Montrer une situation où cet algorithme ne laisse qu'un seul des 100 serveurs s'exécuter et en fait attendre 99.

****

On met alors en place une utilisation en deux niveaux : seuls 60 serveurs ont le droit d'être en sessions concurrentes et de demander des cases ; si 60 joueurs sont déjà en session, les autres attendent et quand un des 60 sort de session, il peut être remplacé par un joueur en attente.

Question 3(2pts)

Montrer qu'il n'y a pas de danger d'interblocage et que parmi les 60 serveurs en session, 20 ne sont jamais bloqués.

****

2. PROGRAMMATION DU CONTRÔLE DE SESSION

On veut limiter à 60 le nombre de serveurs concurrents en cours de session. Ce contrôle est géré par un paquetage `session`

```
package session is
  procedure entree;
  procedure sortie;
end session;
```

Avant d'entrer en session, chaque serveur exécute `session.entree` qui le met en attente si déjà 60 serveurs sont en cours de session; à la fin de chaque session, il exécute `session.sortie` qui autorise un autre serveur à entrer en session.

```
package body session is
-- déclaration de(s) sémaphore(s)
  procedure entree is
  begin -- à compléter; end entree;
  procedure sortie is
  begin -- à compléter; end sortie;
begin -- initialisation de(s) sémaphore(s);
end session;
```

Question 4(1pt)

Complétez les procédures `entree` *et* `sortie` *pour garantir le contrôle de session.* Utilisez pour votre réponse l'annexe .page 1.

3. PROGRAMMATION DE L'ALLOCATION DYNAMIQUE DES CASES

Cette allocation est gérée par un paquetage appelé `Casier`

```
package Casier is
  procedure Inserer(UnObjet:in Objet;S:in IdServeur;K:in Coup);
  procedure Recuperer(UnObjet:out Objet;S:in IdServeur;K:in Coup);
end Casier;
```

Un serveur I exécute `Casier.Inserer(O,I,X)` pour déposer l'objet O du Xième coup et Casier.`Recuperer(O,I,X)` pour retirer l' objet O du Xième coup.

Implantation du paquetage `Casier` :

Le paquetage `Casier` utilise un tableau T de 200 cases pour conserver les objets.

procédure Inserer :
Met en attente le demandeur si les 200 cases sont occupées, puis recherche une case libre dans T et remplit la case.

procédure Recuperer :
Recherche une case occupée pour S pour le Kième coup, lit l'objet, libère la case puis ajoute une autorisation d'allocation d'une case

```
package body Casier is
type Case is record
   Nom : IdServeur;
   Obj : Objet;
   Co : Coup;
end record;
-- Une case C est libre si C.nom=0, occupée sinon
T : array(1..200) of Case;
-- Déclaration des sémaphores

   procedure Inserer(UnObjet:in Objet;S:in IdServeur;K:in Coup);
   C:=integer; -- numero de case;
   begin
   --  à compléter
   for i in 1..200 loop
     if T(i).nom=0 then C:=i; exit;
     endloop;
   T(C).nom:=S; T(C).Obj:=UnObjet;T(C).Co:=K;
   --  à compléter
   end Inserer;

   procedure Recuperer(UnObjet:out Objet;S:in IdServeur;K:in Coup) is
   C:=integer; -- numero de case;
   begin
   --  à compléter
   for i in 1..200 loop
     if T(i).nom:=S and T(i).Co:=K then C:=i; exit;
     end loop;
   UnObjet:=T(C).Obj;
   T(C).Nom:=0;
   --  à compléter
   end Recuperer;

begin for i in 1.. loop T(i).Nom:=0; end loop;
-- Initialisation des sémaphores
end Casier;
```

Question 5 (3pts).

Programmer avec des sémaphores le maintien de la cohérence dans le paquetage Casier. Utilisez pour votre réponse l'annexe page 2.

4. PROGRAMMATION DU STOCKAGE SUR DISQUE

Le maître comprend aussi un processus Disquaire qui stocke les objets sur disque et qui va les y rechercher.Le programme du processus Disquaire est donné en page 5 de l'énoncé.

Pour stocker les objets les Serveurs et le Disquaire communiquent par un tampon. Ils se communiquent des messages qui contiennent un objet et le nom du serveur . Pour faciliter la gestion des transferts par le Disquaire, chaque serveur doit placer ses 4 objets dans des cases consécutives du tampon.

Le contrôle d'accès au tampon et de la communication des messages sont gérés par le paquetage **Tampon.**

```
package Tampon is
   procedure DebutSequence;
   procedure FinSequence;
   procedure Deposer(M:in Message);
   procedure Retirer(M:out Message);
end Tampon;
```

La procédure DébutSequence est appelée par un Serveur pour réserver l'accès au tampon pendant le dépôt de quatre messages .
La procédure FinSequence est appelée par un Serveur pour libérer l'accès au tampon
La procédure Deposer est appelé par un Serveur pour déposer un message dans le tampon pour le disquaire.
La procédure Retirer est appelée par le Disquaire pour retirer un message du tampon.

Le paquetage `Tampon` gère un tableau `Tamp` de six messages.

```
package body Tampon is
Tamp : array (0..5) of Message;
Tete, Queue : Integer;
-- Déclarations des sémaphores
   procedure DebutSequence is
   -- à programmer
   procedure FinSequence is
   -- à programmer
   procedure Deposer(M:in Message) is
   -- à programmer
   procedure Retirer(M:out Message) is
   -- à programmer
begin
-- Initialisation des variables et des sémaphores
end Tampon;
```

Question 6 (3pts)

Programmer la coopération des processus par le paquetage Tampon et la gestion du tableau Tamp. Utilisez pour votre réponse l'annexe page 3.

****

Pour optimiser le rangement sur disque et le transfert, le disquaire doit envoyer les quatre messages d'un serveur sur des secteurs consécutifs d'une piste. On veut que le serveur n'alerte le disquaire qu'en fin de séquence de dépôt de ses quatre messages.

Question 7 (2pts)

Modifier en conséquence les procédures FinSequence et Deposer.

PROGRAMMATION DES PROCESSUS

```
Type Objet; Type IdServeur;Type Coup is Integer range 1..4;
Type Message is record
            Obj : Objet;
            Nom : IdServeur;
      end record;

package Session is
      procedure Entree;
      procedure Sortie;
end Session;

package Casier is
  procedure Inserer(UnObjet:in Objet;S:in IdServeur;X:in Coup);
      procedure Recuperer(UnObjet:out Objet;S:in IdServeur;X:in Coup);
end Casier ;

package Tampon is
      procedure DebutSequence;
      procedure FinSequence;
      procedure Deposer(M : in Message);
      procedure Retirer(M : out Message);
end Tampon;

task type UnServeur;
Serveur : array(1..100) of UnServeur;
task body UnServeur is
      UnObjet : Objet; M : Message;
      Ego : IdServeur := Nom du serveur;
-- Les procedures de dialogue du serveur et du joueur sont fournies :
-- AttendreUnMessageDuJoueur, DonnerLeFeuVertEtAttendreLeCoupSuivant(UnObjet),
-- CalculerEtEnvoyerLeScoreAuJoueur
begin loop
      AttendreUnMessageDuJoueur;
      Session.Entree; -- controle d'entree

      for X in 1..4 loop
            DonnerLeFeuVertEtAttendreLeCoupSuivant(UnObjet);
            Casier.Inserer(UnObjet, Ego, X); -- depot de l'objet recu
      end loop;

      CalculerEtEnvoyerLeScoreAuJoueur;
      for X in 1..4 loop
            Casier.Recuperer(UnObjet, Ego, X);
            M.Obj := UnObjet; M.Nom := Ego; -- prepare le message
            if X = 1 then Tampon.DebutSequence; end if;
            Tampon.Deposer(M);
            if X = 4 then Tampon.FinSequence; end if;
      end loop;
end loop;
end UnServeur;

task Disquaire;
task body Disquaire is
      M : Message;
begin loop
      Tampon.Retirer(M);
      EcrireSurDisque(M);
      end loop;
end Disquaire;
```

DEUXIÈME PROBLÈME : **GESTION D'UN CACHE** (6 points)

On veut utiliser un album d'images stocké sur disque. L'album comprend au plus 100 images.

Les opérations possibles sur l'album sont
- Ajouter_image ajoute une nouvelle image à l'album;
- Retirer_image retire une image de l'album;
- Visualiser_image lit une image dans l'album ;

On ne se préoccupera pas de l'implantation de l'album sur disque, mais seulement de la gestion des transferts disque.

Afin d'améliorer les performances, on décide d'utiliser un cache de 10 images en mémoire centrale.

Question 8 (2pts)

Rappeler le principe de fonctionnement d'un cache. Supposant que pour toute opération une image transite par le cache, expliquer pour chaque opération l'utilisation du cache.

****

Quand le cache est plein, il faut trouver une image à remplacer. L'algorithme de remplacement utilisé prend toujours comme victime l'image contenue dans la première place du cache.Une opération retirer_image provoque une lecture de l'image, puis la libération de sa place dans le cache.

Question 9 (1pt)

Quels sont à votre avis les avantages et inconvénients de cette politique de remplacement?

****

Pour chaque image dans le cache, on note si c'est une nouvelle image ou non. Si ce n'est pas une nouvelle image, il est inutile de l'écrire sur disque.

Question 10 (1pt)

Proposer une amélioration de l'algorithme de remplacement.

****

Pour chaque image dans le cache, on note la date de la dernière opération sur cette image.

Question 11 (1pt)

Proposer un autre algorithme de remplacement utilisant cette date. Quels en sont à votre avis les avantages et les inconvénients?

****

On décide de doubler la taille du cache.

Question 12 (1pt)

Quel algorithme de remplacement choisiriez-vous? Justifier votre réponse en considérant par exemple la fréquence des défauts de cache, la complexité des algorithmes.

ANNEXE page 1 *Pour l'anonymat* ***, Mettez ici votre numéro de copie*** *:*
Ne mettez ni nom, ni numéro de carte CNAM.

Question 4 (1pt)

```
package body session is
-- déclaration de(s) sémaphore(s)


procedure entree is
begin -- à compléter;


end entree;
procedure sortie is
begin -- à compléter;


end sortie;
begin -- initialisation de(s) sémaphore(s);


end session;
```

ANNEXE page 2 *Pour l'anonymat* ***, Mettez ici votre numéro de copie*** *:*
Ne mettez ni nom, ni numéro de carte CNAM.

Question 5 (3pts)

```
package body Casier is
type Case is record
Nom : IdServeur;
Obj : Objet;
Co : Coup;
end record;
-- Une case C est libre si C.nom=0, occupée sinon
T : array(1..200) of Case;
-- Déclaration des sémaphores


procedure Inserer(UnObjet:in Objet;S:in IdServeur;K:in Coup);
C:=integer;
begin
-- à compléter



for i in 1..200 loop
if T(i).nom=0 then C:=i; exit;
endloop;
T(C).nom:=S; T(C).Obj:=UnObjet;T(C).Co:=K;
--à compléter


end Inserer;

procedure Recuperer(UnObjet:out Objet;S:in IdServeur;K:in Coup)
is
C:=integer;
begin
-- à compléter


for i in 1..200 loop
if T(i).nom:=S and T(i).Co:=K then C:=i; exit;
end loop;
UnObjet:=T(C).Obj;
T(C).Nom:=0;
-- à compléter;




end Recuperer;
begin for i in 1..200 loop T(i).Nom:=0; end loop;
-- Initialisation des sémaphores


end Casier;
```

ANNEXE page 3 *Pour l'anonymat* ***, Mettez ici votre numéro de copie*** *:*
Ne mettez ni nom, ni numéro de carte CNAM.

Question 6 (3pts)

```
package body Tampon is
Tamp : array (0..5) of Message;
Tete, Queue : Integer;
-- Déclarations des sémaphores


procedure DebutSequence is
begin
-- à programmer



end DebutSequence;
procedure FinSequence is
-- à programmer



end FinSequence;

procedure Deposer(M:in Message) is
-- à programmer







end Deposer;
procedure Retirer(M:out Message) is
-- à programmer







end Retirer;
begin
-- Initialisation des variables et des sémaphores




end Tampon;
```