SYSTÈMES ET RÉSEAUX INFORMATIQUES

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

EXAMEN DU 22 JUIN 2002

portant sur l'enseignement des SYSTÈMES INFORMATIQUES

Date : samedi 22 juin 2002
Heure : 09h à 12h

Lieu : CNAM Amphi 1 et Amphi 2

TOUS DOCUMENTS AUTORISES

(les ordinateurs portables ne sont pas autorisés)

****

Texte de 4 pages plus 6 pages d'annexes

L'examen comprend :
des questions sur la gestion des ressources
des questions sur la synchronisation

Chaque question peut-être traitée indépendamment.

Attention, vous n'avez pas de programme Ada à écrire, vous n'avez qu'à initialiser, placer et utiliser des sémaphores et des variables globales.

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

****

Barème indicatif :

A.GESTION DES RESSOURCES : 5 points

A1 : (3 points)

A2 : (2 points)

B. LA FÊTE DES CAMÉNÉONS : 15 points

B1a (4 points) ; B1b (1 point) ;

B2a (3 points) ; B2b (2 points) ; B2c (2 points) ;

B3 (3 points pour une solution) (2 points en bonus si deuxième solution)

A. GESTION DES RESSOURCES

QUESTIONS A1 SUR LE COURS

Indiquer la bonne réponse en justifiant votre choix en 5 lignes maximum. Un choix non justifié sera considéré comme nul (0 point).

QUESTION A1a. Ce qui distingue un système multiprocesseur d'un système multiprogrammé, c'est que dans un système multiprocesseur :

a- la mémoire principale est partagée par plusieurs processus,

b- les travaux sont gérés par plusieurs files de travaux,

c- le temps des processeurs est découpé en quantums et partagé entre les processus,

d- plusieurs processeurs peuvent être actifs simultanément.

QUESTION A1b. On dit qu'il y a coalition entre processus quand :

a- les processus sont gérés par priorités,

b- un processus est bloqué en attente de requêtes et aucun processus ne le réveille en lui envoyant une requête,

c- la demande de ressources d'un processus est constamment retardée par suite des demandes des autres processus,

d- plusieurs processus sont programmés pour accéder à tour de rôle à des données critiques.

QUESTION A1c. Un système est en interblocage quand :

a- certains processus ne sont jamais activés,

b- les cases de mémoire attribuées précédemment aux processus prêts sont reprises par l'algorithme de remplacement pour charger des pages du processus élu,

c- des processus bloqués s'attendent circulairement,

d- les interruptions sont masquées et des processus sont en attente active.

****

QUESTION A2 : CHOIX DE MACHINES POUR DU TEMPS RÉEL

Jean possède une machine T avec un processeur ancien et un ordonnancement du processeur selon l'algorithme EDF (Earliest Deadline First) qui s'appuie sur l'échéance et la préemption. Il s'en sert pour gérer une application temps réel qui comprend deux processus A et B. Le processus A (processus de commande d'un moteur) est activé périodiquement et chaque activation demande 270 secondes de calcul et a une échéance de 320 secondes (autrement dit si le processus n'est pas terminé 320 secondes après son activation périodique, on a une faute temporelle); le processus B est déclenché par un manche à balai ("joystick"). Il demande 15 secondes de calcul et son échéance est de 21 secondes (au delà de 21 secondes, c'est la catastrophe).

On lui propose de changer sa machine pour une machine L avec un processeur dix fois plus rapide qui est associé à un ordonnancement à l'ancienneté sans préemption (A s'y exécute en 27 secondes et B en 1,5 seconde). Bien entendu les échéances ne changent pas.

Comparer le fonctionnement des deux machines dans le cas suivant :

- A l'instant 0, le processus A est déclenché,
- A l'instant 1 le processus B est déclenché.

On suppose nul le surcoût système pour la commutation de processus.

Au vu du résultat, Jean doit-il changer de machine ?

B. SYNCHRONISATION DE PROCESSUS

LA FÊTE DES CAMÉNÉONS

Le caménéon est une variété de caméléon qui peut prendre l'une des couleurs : bleu, jaune, rouge. Quand deux caménéons se rencontrent au mail (dans le midi de la France, le mail est une promenade bordée d'arbres, où l'on jouait jadis au jeu de maillet - ou jeu de mail) une mutation peut se produire. S'ils sont de couleurs différentes, ils changent de couleur et prennent la troisième couleur. S'ils sont de même couleur, ils gardent cette couleur.

On met en place un monde virtuel de caménéons qui de temps en temps entrent dans le mail.

- La vie des caménéons est décrite dans la procédure Main.
- La gestion de leur couleur est faite dans le paquetage Carnation

Le paquetage Carnation ne gère pas la cohérence en cas d'accès concurrents.

- Le paquetage Nom permet de leur attribuer un nom unique (même s'il y a appel concurrent).

Tout cela figure dans l'ANNEXE GÉNÉRALE.

On va étudier plusieurs organisations du paquetage Mail qui contient la procédure Rencontre(X) qu'utilise tout caménéon X pour entrer dans le mail et y faire la rencontre d'un autre caménéon. La procédure Rencontre est réentrante et plusieurs caménéons peuvent l'appeler et l'exécuter en concurrence. La procédure Rencontre met en relation deux par deux les caménéons qui l'appellent.

****

PREMIÈRE ORGANISATION DU MAIL
(processus magicien, sorcier et sémaphores)

La procédure Rencontre(X) comprend deux actions : le dépôt d'une requête avec le nom X puis l'attente d'un signal avant de sortir du mail. Un processus "Magicien" prend ces requêtes deux à deux, effectue la mutation éventuelle et envoie les signaux qui permettent aux caménéons mutés de sortir du mail.

QUESTION B1a. Les caménéons et le magicien communiquent par un tampon T (qui est géré à l'ancienneté par deux index Queue et Tete, et qui contient des noms de caménéons) et par un tableau Sem de sémaphores qui servent à attendre et envoyer le signal de sortie.
Compléter la programmation de "package body Mail" donné en ANNEXE MAIL UN. Ajouter les sémaphores nécessaires. (On ne fera pas une programmation modulaire, ce qui importe ici c'est de bien synchroniser les processus pour qu'ils respectent la cohérence des données et des index du tampon et pour que la mutation soit faite avant le réveil des caménéons.). Programmer de façon que le magicien et un caménéon puissent avoir accès concurrement au tampon T.

QUESTION B1b. On suppose qu'on ajoute un processus "Sorcier" qui déclenche des mutations "spontanées" selon le programme donné aussi dans l'annexe MAIL UN. Quelle contrainte nouvelle est introduite par cet ajout ? Programmer cette contrainte dans les processus.

REMARQUE. En fait le Sorcier joue avec les caménéons : en forçant quelques mutations, il influence les mutations naturelles en espérant amener tous les caménéons à avoir la même couleur. Voir le cours de R.O. (MOCA B) pour savoir si le sorcier a raison de jouer ainsi. ☙

****

DEUXIÈME ORGANISATION DU MAIL
(sémaphores privés)

Il n'y a pas de processus Magicien ni processus Sorcier. C'est aux caménéons de s'attendre et de gérer leur mutation. Le premier caménéon, C1, qui appelle "Rencontre(C1)" doit attendre qu'un second caménéon, C2, appelle à son tour Rencontre(C2)". Ce deuxième caménéon doit récupérer le nom du premier, faire la mutation, puis réveiller le premier pour qu'il puisse alors seulement sortir du Mail.

QUESTION B2a. Programmer avec un tableau Sem de sémaphores privés en complétant le "package body Mail" donné en ANNEXE MAIL DEUX.

QUESTION B2b. Variante pour réduire la section critique dans la procédure Rencontre.

On appelle variable persistante, une variable commune aux différents utilisateurs d'un paquetage, et variable locale une variable qui est propre à un processus utilisateur.

On constate, dans la question B2a, que le second processus qui appelle Rencontre(X) utilise, lorsqu'il appelle Mutation(A, B), des variables A et B qui sont persistantes (ou encore partagées par tous les utilisateurs du paquetage Mail). Cela entraîne donc une contrainte de cohérence lors de l'appel de Mutation(A, B).

Dans la question B2b, on utilise deux variables A, B qui ne sont plus persistantes mais qui sont des variables locales à chaque processus. Le second processus copie la variable persistante SauveA et

son paramètre X dans ses variables locales A et B. La contrainte de cohérence ne porte plus que sur SauveA. Il n'y a plus de contrainte de cohérence lors de l'appel de Mutation(A, B).

Programmer avec des sémaphores privés en complétant le "package body Mail" donné en annexe MAIL DEUX BIS.

QUESTION B2c. Pour ne pas avoir à créer autant de sémaphores privés Sem(IdProc) que de caménéons, alors qu'il ne faut bloquer qu'un processus au cours de la rencontre entre deux caménéons, on essaie de n'utiliser qu'un seul sémaphore de blocage Sbloc dans le schéma des sémaphores privés. Le processus bloqué dans l'appel de Rencontre(X) est bloqué par Sbloc qui remplace le sémaphore privé du schéma donné en cours. Montrer que cette solution est fausse.
Utiliser le scénario suivant.
Supposons que l'on ait un appel concurrent de Rencontre(X) par trois caménéons C1, C2, C3.
Supposons que l'on exécute d'abord C1 et C2 et que, par le jeu de l'allocateur de processeur, chaque processus C1 et C2 ait été préempté après être sorti de section critique et avant qu'il ait pu se bloquer éventuellement sur le sémaphore Sbloc (qui remplace son sémaphore privé). Quelle est alors la valeur du compteur du sémaphore Sbloc, soit Sbloc.E ? Le processeur est alors alloué à C3 qui appelle Rencontre et s'exécute entièrement sans être préempté. C3, qui est alors reconnu par l'algorithme comme un premier processus d'un couple de caménéons mutants va-t-il être bloqué ?
****

TROISIÈME ORGANISATION DU MAIL

(tache serveur en Ada avec rendez-vous)

On utilise les tâches Ada comme serveur de synchronisation. On n'utilise plus de sémaphore. Cela peut se faire de plusieurs manières.

La première manière (première version) consiste à créer :

- une tâche serveur Requête appelée pour déposer et Retirer des noms de caménéons,
- un tableau Signal(IdProc) de tâches serveurs sur le modèle de UnSignal appelée pour Attendre et Envoyer un signal.

La deuxième manière (deuxième version) consiste à n'utiliser qu'une seule tâche serveur "Magicien". Pour cette tâche Magicien, on s'inspire du chapitre 10 du cours et de l'exemple de la tâche serveur de rendez-vous symétrique anonyme qui emboîte les "accept d'entry" Emettre et Recevoir.

Ici on emboîte les "entry" Rencontre1 et Rencontre2 appelées alternativement par un caménéon de numéro pair et par un caménéon de numéro impair.

QUESTION B3. Compléter la programmation du "package body Mail" donné en ANNEXE MAIL TROIS soit avec la VERSION UN soit avec la VERSION DEUX, selon votre choix.
****

EXAMEN DU 22 JUIN 2002
ANNEXES QUESTION B. SYNCHRONISATION DE PROCESSUS
ANNEXE GÉNÉRALE

```
-- ********************                    *************************
package Carnation is
    type Couleur is (Bleu, Jaune, Rouge);
    NbCameneons : Constant Integer := 50;
    type IdProc is mod NbCameneons; -- type modulo, range 0..NbCameneons-1
    function GetCouleur(X : in IdProc) return Couleur;  -- c'est la couleur de X
    procedure Mutation(X, Y : in out IdProc) ; -- effectue l'éventuel changement de couleur
            -- cette procédure ne comprend aucune synchronisation en cas de concurrence
end Carnation;

-- ************************
with Carnation; use Carnation;

package Nom is
    procedure Unique(X : out IdProc);
end Nom;

-- *************************
with Carnation; use Carnation;

package Mail is
    procedure Rencontre(X : in out IdProc); -- X rencontre un autre caménéon
end Mail ;
-- *************************

with Nom; with Mail; with Carnation; use Carnation;
with Ada.Text_IO; use Ada.Text_IO;

procedure Main is                    -- ****    décrit le comportement des caménéons

    task type Animal;
    Seconds : constant Duration := 1.0;

    task body Animal is
        MaCouleur, Y : Couleur;
        Ego : IdProc;
    begin
        Nom.Unique(Ego); -- acquiert un numero unique
        MaCouleur := GetCouleur(Ego);
        for I in 1..50 loop
            --  ce delai simule une vie en dehors du mail
             delay Duration((Float(Ego)/Float(I))*Seconds);
            Mail.Rencontre(Ego);
            Y:= GetCouleur(Ego); -- pour voir si elle a changé
            if Y /= MaCouleur then
                Put_Line("Le caménéon" & IdProc'Image(Ego) &" : "&
                     Couleur'Image(MaCouleur)& "=>" & Couleur'Image(Y ));
                MaCouleur := Y;
            end if;
        end loop;
    end Animal;

    Cameneon : array(IdProc) of Animal;

begin null; end Main;
-- ******************    fin de l'ANNEXE GÉNÉRALE    ************************
```

```
--                    ANNEXE  MAIL  UN  (questions B1a et B1b)
--- ******      avec un processus magicien (+ sorcier) et des sémaphores  *******
-- ***********                                                            *******
package body Mail is
      -- *** tampon de requêtes     *********
       -- déclarations pour gerer le tampon
      << 1 : à faire  pour la question B1a >> : Semaphore;
      type Index is mod 5;
      T : array(Index) of IdProc;
      Tete , Queue : Index := 0;
      procedure Deposer(X : in IdProc) is
      begin
      << 2 : à faire  pour la question B1a >>
      end Deposer;
      procedure Retirer(Y : out IdProc) is
      begin
      << 3 :  à faire  pour la question B1a >>
      end Retirer;
      --  ****autres déclarations de sémaphores
      << 4 : à faire pour B1b >> : Semaphore; -- déclaration pour la question B1b
      Sem : array(IdProc) of Semaphore; -- déclaration pour attente et envoi de signaux
--  ******************** ******************
      procedure Rencontre(X: in out IdProc) is
      begin
            Deposer(X);  -- dépose sa demande
            -- X attend le signal pour sortir de la procedure
      << 5 : à faire  pour la question B1a >>
      end Rencontre;
--  ******************** ******************
      task Magicien; task body Magicien is
            A, B : IdProc;
      -- ne pas oublier d'assurer la coherence quand il y a le sorcier,
      << 6 : dans la question B1b >>
      begin
            loop
                  Retirer(A);  Retirer(B);     -- il faut les requêtes de deux caménéons
                  Mutation(A, B);              -- modifie éventuellement les couleurs de A et de B
      << 7 : à faire  pour la question B1a>> -- envoyer les signaux à A et B
            end loop;
      end Magicien ;
-- *** pour la question B1b*****************
      task Sorcier; task body Sorcier is
            I, J : IdProc := IdProc'First;
      -- ne pas oublier d'assurer la coherence quand il y a le sorcier,
       << 8 : dans la question B1b >>
      begin
            for K in 1..10*NbCameneons loop
                  Mutation(I, J);
                  I := I + 2; J := J + 3; -- additions modulo NbCameneons
            end loop;
      end Sorcier;
begin  --  *****    initialisations de tous les sémaphores   **
      << 9 et 10 : à faire  pour la question B1a et pour la question B1b >>
      < mettre aussi  les valeurs initiales de Sem pour B1a>
      for I in IdProc loop E0(Sem(I),    ); end loop;
end Mail;
-- **********************              fin de l'annexe UN        **********************
```

```
--                    ANNEXE  MAIL  DEUX   (question B2a)
--- ***************** procédures avec des sémaphores privés                ****
-- ******************                                          ********** **
```

```
package body Mail is
    --  assurer la coherence des variables persistantes
    << 1 :  à faire pour B2a >>: Semaphore;
    Sem : array(IdProc) of Semaphore;
    Premier : Boolean := True;          -- variable persistante (ou commune à tous utilisateurs)
    A, B : IdProc;      -- pour sauvegarder les noms des 2 caménéons : variables persistantes
--  ************************************
    procedure Rencontre(X : in out IdProc) is
    -- assurer la coherence des variables persistantes
    -- ne pas oublier le blocage du premier processus appelant,
    -- puis son reveil par l'appelant suivant
     -- le deuxième appelant réveille le premier processus (le seul a être bloqué)
    << 2 : à faire pour B2a : completer le code ci-dessous >>
    begin
        if Premier then
            A := X; Premier := False; -- pour le prochain appel
        else
            B := X; Premier := True; -- pour le prochain appel
            Mutation(A, B);          -- utilise les variables persistantes A et B
        end if;
    end Rencontre;

--  ******  initialisations de tous les sémaphores
begin
    << 3 :  à faire pour B2a >>
    < initialiser correctement Sem>for I in IdProc loop E0(Sem(I),    ); end loop;
end Mail;

--  **********************          fin de l'annexe DEUX          ********************
```

-- **ANNEXE MAIL DEUX BIS (question B2b)**
-- ********** **variante avec section critique courte** *********
--- ********** **sémaphores privés** **********
-- ********** ********** **

```
package body Mail is
    --  assurer la coherence des variables persistantes
    << 1 : à faire pour B2b >>: Semaphore;
    Sem : array(IdProc) of Semaphore;
    Premier : Boolean := True;              -- variable persistante, commune à tous utilisateurs
    SauveA: IdProc; -- une seule variable persistante, sauvegarde le nom du premier caménéon

--  **********************************
    procedure Rencontre(X : in out IdProc) is
        A, B: IdProc; -- variables locales qui ne servent qu'au 2eme processus
    -- assurer la coherence des variables persistantes
    -- ne pas oublier le blocage du premier processus appelant,
    -- puis son reveil par l'appelant suivant, une fois la mutation faite
     -- le deuxième appelant réveille le premier processus (le seul a être bloqué)
    << 2 : à faire pour B2b : completer le code ci-dessous >>
    begin
        if Premier then
            SauveA := X; Premier := False; -- pour le prochain appel
        else
            Premier := True; -- pour le prochain appel
            -- récupérer le nom du 1er processus dans la copie locale au 2eme processus
            A := SauveA;
            B := X;                     -- c'est le nom du 2eme processus
            Mutation(A, B);     -- il n'y a pas de problème de cohérence, A et B sont locales
        end if;
    end Rencontre;

--  **     initialisations de tous les sémaphores
begin
    << 3 : à faire pour B2b >>
    < initialiser correctement Sem > for I in IdProc loop E0(Sem(I),    ); end loop;
end Mail;
```

-- ********** fin de l'annexe DEUX BIS **********

```
--                    ANNEXE   MAIL TROIS  : VERSION UN  (question B3)
--**********       solution avec tache serveur en Ada avec rendez-vous *****
--   ** pour simplifier l'écriture des solutions, on n'utilise pas la clause terminate
-- ************                                                    **********

package body Mail is

-- ***************************
      task  Requete is
            entry Deposer(X : in IdProc);
            entry Retirer(Y : out IdProc);
      end Requete;

      package body Requete is
      << 1 : à compléter pour la question B3 >>
      end Requete;

-- ***************************
      task type UnSignal is
            entry Attendre;
            entry Envoyer;
      end UnSignal;

      task body UnSignal is
            SignalRecu : Boolean := False;
      begin
      << 2 : à compléter pour la question B3 >>
      end UnSignal;

      Signal : array(IdProc) of UnSignal; -- tableau de taches serveur Signal
-- ***************************
      procedure Rencontre(X : in out IdProc) is
      begin
            Requete.Deposer(X);  -- dépose sa demande
            Signal(X).Attendre;   -- attend le signal pour sortir de la procedure
      end Rencontre;
-- ***************************
      task Magicien;
      task body Magicien is
            A, B : IdProc;
      begin
            loop
                  Requete.Retirer(A);
                  Requete.Retirer(B);
                  Mutation(A, B);
                  Signal(A).Envoyer; Signal(B).Envoyer;
            end loop;
      end Magicien ;

begin null; end Mail;

-- ***************          fin de l'annexe TROIS version UN          ************
```

```
--                ANNEXE   MAIL   TROIS : VERSION DEUX
--                      (autre solution pour Question B3
--**********      solution avec tache serveur en Ada avec rendez-vous      *****
--   ** pour simplifier l'écriture des solutions, on n'utilise pas la clause terminate
-- **************                                                          ********

package body Mail is

-- ***************************
      task Magicien is
            entry Rencontre1(X : in out IdProc);
            entry Rencontre2(Y : in out IdProc);
      end Magicien ;

      task body Magicien is
      begin
            loop
      << 1 : à compléter pour la question B3 >>
            end loop;
      end Magicien ;
-- ***************************
      procedure Rencontre(X: in out IdProc) is
      begin
            if (X mod 2 = 0) then
                  Magicien.Rencontre1(X);
            else
                  Magicien.Rencontre2(X);
            end if;
      end Rencontre;

begin null; end Mail;

-- ***************      fin de l'annexe TROIS version DEUX      ************
```