SYSTÈMES ET RÉSEAUX INFORMATIQUES

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

EXAMEN DU 19 JUIN 2004

portant sur l'enseignement des SYSTÈMES INFORMATIQUES

Date : samedi 19 juin 2004
Heure : 14h30 à 17h30
Lieu : au CNAM2, rue Conté, amphi 2 et salle 30.-1.02

TOUS DOCUMENTS PAPIERS AUTORISES
(les ordinateurs portables ne sont pas autorisés)
****

Énoncé de 4 pages

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

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 : 5 points pour la première question, 4 points pour chacune des autres questions.
1. Ordonnancement des processus
2. Coopération de processus concurrents attachés au suivi d'activité d'une société avec 5 usines
2.1. Synchronisation des processus proxys
2.2. Dépôt des rapports
2.3. Allocateur de pages aux serveurs
2.4. Droit d'accès à la base d'images

1 Ordonnancement des processus

Un système monoprocesseur comprend 5 processus A, B, C, D, E, qui vont être activés à leur date de déclenchement (à sa date de déclenchement, un processus est mis alors dans la file des processus prêts) pour demander l'unité centrale pour une durée d'exécution. À chaque processus est associé une date d'échéance, avant laquelle il doit se terminer, sinon il y a une faute temporelle. Les temps de commutation de contexte sont considérés comme négligeables.

Processus	Durée d'exécution	date de déclenchement	Date d'échéance	Date de fin
A	12	1	16	
B	3	0	17	
C	2	2	16	
D	1	3	17	
E	1	4	15	

Question 1. Pour chaque processus on vous demande de calculer la date de fin d'exécution selon la nature de l'ordonnancement du processeur et de dire s'il y a faute temporelle ou non.

1.1 Selon la politique de l'ancienneté (FIFO)

1.2 Selon la politique du tourniquet avec un quantum de 2. Quand un processus a une date de déclenchement qui coïncide avec un quantum (cas de C et E), on suppose qu'il est déclenché juste avant le quantum et qu'il est donc déjà dans la file des processus prêts au moment du quantum.

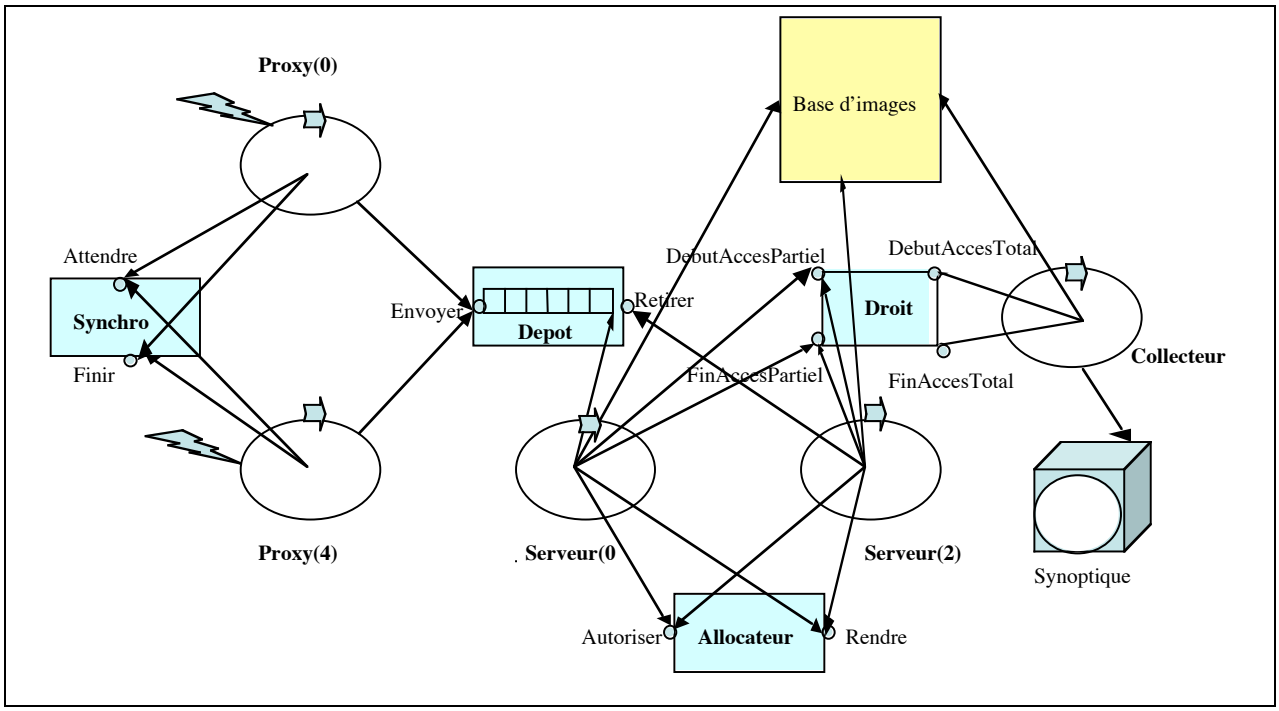
1.3 Selon un ordonnancement par échéance EDF (« earliest deadline »). Si deux processus ont la même échéance, on prend le plus anciennement déclenché en premier (A avant C). Ne pas oublier que les dates de déclenchement ne sont pas toutes les mêmes.

2 Coopération de processus concurrents

Une société qui possède 5 usines veut installer à son siège un écran synoptique pour suivre l'activité de ses usines. Après avoir étudié le système d'information et produit une architecture logique, cette société fait appel à vous pour gérer la coopération entre les processus concurrents de cette architecture.

```
-- les processus coopérants

task type Un_Proxy ;
   Proxy : array(Id_Usine) of Un_Proxy ;

task type Un_Serveur ;
Serveur : array(Id_S) of Un_Serveur ;

task Collecteur :

-- les paquetages de coopération

package Synchro is
  procedure Attendre(X : in Id_Usine);
 procedure Finir(X : in Id_Usine);
end Synchro;
```

```
package Depot is
  procedure Envoyer(X : in Rapport);
  procedure Retirer(X : out Rapport);
end Depot;

package Allocateur is
  procedure Autoriser(I : in Id_S ; X : in Integer);
  procedure Rendre(I : in Id_S ; X : in Integer);
end Allocateur;

package Droit is
procedure Debut_Acces_Partiel(X : in Id_Usine);
 procedure Fin_Acces_Partiel(X : in Id_Usine);
  procedure Debut_Acces_Total;
  procedure Fin_Acces_Total;
end Droit;
```



type **Id_Usine** is mod 5 ; -- modulo type **Rapport** is record Usine : Id_Usine ; Date : time ; Info : Donnees ; end record ;	type **Id_S** is mod 3 ; type **Image** is record Usine : Id_Usine ; Date : time ; Figure ; Dessin ; end record ;

Un processus **Proxy(X)** est attaché à chaque usine X pour en recevoir ses rapports , puis pour les déposer dans le tampon de dépôt. Les 5 processus Proxy sont cycliques et synchronisés entre eux pour qu'ils déposent les rapports les uns après les autres dans un ordre fixe.

```
task body Un_Proxy is
   X : Id_Usine ; Y : Rapport ;
begin
   loop
      transfert_Du_Rapport_De_X(Y) ;  -- réception du rapport par un accès réseau
      Synchro.Attendre(X) ;        -- synchronisation entre les proxys pour fixer l'ordre des dépôts
      Depot.Envoyer(Y) ;               -- maintenant l'envoi du rapport Y est possible
      Synchro.Finir(X) ;               -- permet de passer la main au successeur de X
   end loop ;
end Un_Proxy ;
```

Chacun des trois processus **Serveur** est cyclique et commence chaque cycle en retirant le plus ancien rapport encore présent dans le dépôt. Soit X l'usine qui a envoyé ce rapport. Le serveur acquiert la mémoire supplémentaire nécessaire pour traiter les statistiques et préparer l'image Z de X. On suppose qu'il faut X pages supplémentaires pour l'usine de numéro X. Quand il a cette mémoire supplémentaire, le Serveur exécute les calculs et construit l'image Z de X, puis il dépose cette image dans la base d'images à un emplacement réservé pour X et fixe.

```
task body Un_serveur is
    I : Id_S ; X : Integer; Y : Rapport ; Z : Image ;
begin
   I := Numero_Unique_Dans_Id_S ;     -- nom du Serveur
   loop
      Depot.Retirer(Y) ;                  --  retrait du plus ancien rapport présent dans le dépôt
      X := Integer(Y.Usine) ;             -- récupère le numéro d'usine et le convertit en un entier
   -- si X > 0, demande le droit de prendre X pages de mémoire
   if X > 0 then Allocateur.Autoriser(I, X) ; end if;
      Calculs_Statistiques_Et_Preparation_Image(Z);         -- utilise un progiciel spécifique
      Droit.Debut_Acces_Partiel(X) ;     -- demande le droit d'accéder à l'image Z de X dans la base
      Mise_A_Jour_De_Image_De_X_Dans_La_Base ;      -- accès à Z, réservé pour X
      Droit.Fin_Acces_Partiel(X);                                -- sort de cet accès
   if X > 0 then Allocateur.Rendre(I, X) ; end if;  -- n'a plus besoin de la mémoire supplémentaire
   end loop ;
end Un_Serveur ;
```

Le processus **Collecteur** est un processus cyclique activé périodiquement pour aller lire toutes les images de la base d'images, préparer une image de synthèse, la stocker dans la base. Ensuite il visualise les images sur le synoptique.

```
task body Collecteur is
begin
    loop
      Attendre_Le_Reveil_Periodique ;            -- déclenchement  externe
      Droit.Debut_Acces_Total ;  -- demande le droit d'accéder à la base en exclusion mutuelle
      Travail_Dans_La_Base ;                    -- avec un progiciel spécialisé
      Visualiser_Les_Images_De_La_Base ;        -- avec un outil de visualisation
      Droit.Fin_Acces_Total ;                    -- libère l'accès à la base
    end loop ;
end Collecteur ;
```

On vous demande de programmer les quatre paquetages de coopération entre les processus en utilisant des sémaphores pour le contrôle de concurrence. Vous le ferez conformément à la syntaxe Ada indiquée ci-dessous:

```
package body C is
< déclaration des sémaphores >
< déclaration des variables persistantes, et éventuellement leur initialisation >
procedure A (X : in X_Type) is … begin ……end A;  --annoncées dans la partie déclarative
procedure B(Y : out Y_Type) is ….begin ……end B; --        du paquetage
begin
       < initialisation éventuelle des variables persistantes >
       < initialisation OBLIGATOIRE des sémaphores >
end C ;
```

Question 2.1. Synchro. Le paquetage Synchro sert à coordonner les Proxys de telle façon que leurs phases de dépôt de rapport soient exécutées les unes après les autres selon l'ordre des numéros d'usine: 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, …
Avec Attendre(X) le processus Proxy(X) attend que le processus Proxy(X – 1) ait fini et quand Proxy(X) a fini, il le signale par Finir(X), ce qui autorise le Proxy(X + 1) à s'exécuter. Les additions et soustractions sont modulo 5 car c'est le type de X.

Programmer ce paquetage en gérant la concurrence avec des sémaphores. Ne pas oublier d'initialiser les sémaphores sinon le programme est faux et sera donc noté comme tel.

Question 2.2. Depot. Le paquetage Depot sert à contenir les rapports déposés par les Proxys et retirés par les Serveurs. Le tampon est géré à l'ancienneté et peut contenir jusqu'à 8 rapports.

On demande de programmer pour avoir le maximum de concurrence d'accès entre les processus utilisant ce paquetage. À un instant donné, quel est le nombre maximum de Proxys qui peuvent appeler ce paquetage? En déduire la programmation qui utilise le moins de sémaphores.

Programmer ce paquetage en gérant la concurrence avec des sémaphores. Ne pas oublier d'initialiser les sémaphores sinon le programme est faux et sera donc noté comme tel.

Question 2.3. Allocateur. Le paquetage Allocateur est utilisé par les Serveurs pour contrôler leurs demandes de pages. On peut programmer selon l'un des deux cas de figure la procédure Autoriser(I, X) du paquetage Allocateur pour traiter la demande de X pages:

a) soit en demandant l'un après l'autre le droit de les utiliser et cela jusqu'à ce que l'utilisation des X pages soit autorisé. Cela permet d'éviter la famine. Le Serveur est alors autorisé à aller prendre ces X pages et à les utiliser puisqu'elles ont toutes été réservées.
b) soit en traitant la demande globalement. Aucune page n'est réservée et le Serveur n'est autorisé à aller prendre ces X pages et à les utiliser que lorsque l'allocateur a pu les lui allouer toutes d'un coup. Cette solution peut entraîner la famine.

Dans chaque cas, de quel minimum de ressources faut-il disposer pour qu'il n'y ait jamais interblocage, sachant qu'il y a trois Serveurs concurrents pouvant demander 4 pages chacun?

Programmer le paquetage Allocateur dans l'un ou l'autre des deux cas de figure, à votre choix, en supposant qu'on dispose du minimum de ressources correspondant. La gestion de la concurrence sera programmée avec des sémaphores. On ne programmera pas la recherche des X pages allouées, mais seulement l'autorisation de faire cette recherche. Ne pas oublier d'initialiser les sémaphores sinon le programme est faux et sera donc noté comme tel. Dans le cas b), on pourra utiliser le schéma des sémaphores privés et le paquetage suivant de gestion des files :

```
L : FileDe(Id_S, Integer) ;  -- objet fourni avec des procédures d'accès
  L.Ajouter(I, X) ; -- on met la requête X du serveur I dans la file d'attente L
  L.EstVide  -- fonction booléenne qui vaut vrai quand la file L est vide
  L.Premier(Lui, K) ; -- on copie le premier élément de L dans K (requête) et Lui (serveur)
  L.OterPremier ; -- on supprime de L le premier élément de L
```

Question 2.4. Droit. Le paquetage Droit est utilisé par les Serveurs et par le Collecteur.

Chaque Serveur fait accès uniquement à une seule image de la base d'images, celle de l'usine X pour laquelle il travaille. La procédure Debut_Acces_Partiel(X) autorise les Serveurs qui travaillent pour des images différentes à utiliser la base d'images en parallèle. Comme plusieurs Serveurs peuvent servir la même usine X concurrement et demander l'accès à la même image de X, on doit, dans cette procédure, imposer au préalable qu'il n'y ait que l'un d'eux à la fois qui puisse utiliser la base d'images pour X, et donc bloquer les autres demandes pour X. Dans la procédure Fin_Acces_Partiel(X), il faut alors in fine penser à réveiller un autre serveur qui travaille pour X et qui aurait été bloqué dans Debut_Acces_Partiel(X). Et ce contrôle doit être fait pour chaque X.

Le Collecteur fait un accès à plusieurs images de la base d'images. Il doit donc pouvoir utiliser la base en exclusion mutuelle avec les Serveurs.

Programmer ce paquetage en gérant la concurrence avec des sémaphores. Ne pas oublier d'initialiser les sémaphores sinon le programme est faux et sera donc noté comme tel.