CORRIGÉ DE L'EXAMEN DU 20 SEPTEMBRE 2001

portant sur l'enseignement des SYSTÈMES INFORMATIQUES

QUESTION 1 : ORDRE POUR LES ANNONCES DE DÉMARRAGE

```
package body Annonce is
   -- déclaration de sémaphores et de variables d'état
MainAAutres, AutresAMain : Semaphore;

   procedure Attendre is
   begin
      -- bloquer le processus appelant tant que Main n'a pas exécuté LancerTous
      P(MainAAutres);
   end Attendre;

   procedure Signaler is
   begin
      -- informer le processus Main que l'appelant a fini
      V(AutresAMain );
   end Signaler;

   procedure LancerTous is
   begin
      -- envoyer aux 5 processus le signal qui leur permet d'imprimer leur annonce
      for I in 1..5 loop V(MainAAutres); end loop;
   end LancerTous ;
   procedure AttendreLaFinDeTous is
   begin
      -- bloquer le processus Main tant que tous les 5 autres processus n'ont pas fait signe de vie
      for I in 1..5 loop P(AutresAMain ); end loop;
   end AttendreLaFinDeTous ;

begin
   -- initialiser chaque sémaphore
   E0(AutresAMain, 0); E0(MainAAutres, 0);
end Annonce;
```

QUESTIONS 2 : ALLOCATION DES RESSOURCES SANS INTERBLOCAGE

Question 2.1. Il faut que l'allocateur puisse ne pas être bloqué dans le cas où la réunion des demandes est la plus contraignante, c'est à dire où tous les 5 processus Acteur ont acquis chacun 4 ressources et où ils demandent leurs 3 dernières ressources. Il faut pouvoir en servir un au moins, car ensuite, celui qui aura été servi rendra toutes ses ressources et tous pourront terminer l'un après l'autre au pire. Soit 5*4 +3 = 23 ressources au moins
Question 2.2. Même raisonnement avec un autre contexte
soit 4*7 + 1 = 29 ressources au moins

QUESTIONS 3 : ACCÈS AU CANAL PARTAGÉ

Question 3.1.

```
package body Canal is

   Libre : Boolean := True ; -- le canal est disponible quand Libre = True
   Attend : array(2..5) of Boolean:= False;  -- le processus I est bloqué quand Attend(I) = True
   Attente : Boolean := False; -- quand Attente = True, il y a au moins un processus en attente
   Suivant : Integer := 5;  -- nom d'un processus, index de parcours du tableau Attend
   -- déclaration des sémaphores utilisés
```

```
Mutex: Semaphore; Sempriv : array(2..5) of Semaphore;
   procedure Ouvrir(I : in Integer) is
      -- le processus de nom I demande l'accès exclusif au canal
      OK : Boolean; -- accès possible ou non
   begin
      P(Mutex);
      OK := Libre;
      if OK then
         Libre := False;  -- I prend le droit d'utiliser le canal
         -- décider qu'il faut autoriser I à continuer son exécution
         V(Sempriv(I));
      else
         Attente := True; Attend(I) := True;
         -- décider qu'il faut bloquer I en attente de ressource
      end if;
      V(Mutex);
      P(Sempriv(I));  -- la décision est appliquée à ce niveau, hors section critique
   end Ouvrir;

   procedure Fermer(I : in Integer) is
      J : Integer;  -- nom du processus à réveiller
      function ChoixCandidat return Integer is -- choisit un processus en attente
      begin
         loop
            if Suivant = 5 then Suivant := 2 else Suivant := Suivant + 1; end if;
            exit when Attend(Suivant); -- sortie de la boucle  quand Attend(Suivant) = True
         end loop;
         return Suivant;
      end ChoixCandidat ;
   begin
      P(Mutex);
      Libre := True; -- libération du canal par le processus I
      if Attente then
         J := ChoixCandidat; Attend(J) := False; Libre := False;
         -- réveiller le processus J
         V(Sempriv(J));
         -- voir s'il y a au moins un autre processus en attente et le noter dans Attente
         for K in 2..5 loop Attente := Attend(K); exit when Attente; end loop;
      end if;
      V(Mutex);
   end Fermer;

begin            -- initialisation des sémaphores utilisés
   E0(Mutex, 1); for K in 2..5 loop E0(Sempriv(K), 0); end loop;
end Canal ;
```

Question 3.2.

Il suffit, dans ChoixCandidat de commencer la recherche du suivant toujours à partir de 2. Alors Suivant n'est plus une variable globale ; ChoixCandidat devient :

```
      function ChoixCandidat return Integer is -- choisit un processus en attente
         Suivant : Integer;
      begin
         for Suivant in 2..5 loop
            exit when Attend(Suivant);
         end loop;
         return Suivant;
      end ChoixCandidat ;
```

Le reste demeure inchangé.

QUESTION 4 : TRANSFERT DES DOCUMENTS SUR DISQUE

Le bras porte-têtes est initialement sur le cylindre 10.

Paquet	taille Nb	ordre de lecture des Nb requêtes	total des déplacements du bras quand tout le service est FIFO	réarrangement pour la politique mixte	total des déplacements du bras pour la politique mixte
S1	4	9, 8, 18, 3	1+1+10+15 = 27	9, 8, 3, 18	7+15=22
S2	5	7, 12, 2, 1, 4	4+5+10+1+3 = 23	12, 7, 4, 2, 1	17
S3	6	12, 4, 6, 5, 7, 8	8+8+2+1+2+1 = 22	4, 5, 6, 7, 8, 12	11
S4	1	14	6	14	2
S5	3	7, 9, 15	7+2+6= 15	7, 9, 15	7+2+6= 15
Total			a = 93		b = 67

Le gain obtenu, en pourcentage, avec la politique mixte est de (93 - 67)/93 = 28%
Le rapport coût/efficacité est intéressant car la politique de l'ascenseur n'est pas trop coûteuse en temps d'exécution et on a une amélioration importante.

QUESTION 5. CONTRÔLE PAR TÂCHE ADA SERVEUR

```
--Première solution. Remplacer le paquetage Annonce par
task body Annonce is
   NbReveil : Integer := 0; -- compte les réveils consommés par les acteurs
   NbSignaux : Integer := 0; -- compte les signaux envoyés vers Main
begin
   loop
      select
         when NbReveil >0 accept Attendre; NbReveil := NbReveil -1; -- en accepte 5
      or
         accept LancerTous; NbReveil := 5;  -- en réveille 5
      or
         accept Signaler; NbSignaux := NbSignaux + 1;  -- envoie un signal
      or
         when NbSignaux >= 5 accept AttendreLaFindeTous; -- a reçu 5 signaux
      or
         terminate;
      end select;
   end loop;
end Annonce ;

--Deuxième solution. Remplacer le paquetage Annonce par
task body Annonce is
begin
   loop
      select
         accept LancerTous; -- Main doit passer le premier à ce rendez-vous
         for K in 1..4 loop accept Attendre; end loop; -- rendez-vous des 4 acteurs
      or
         accept AttendreLaFindeTous do
               for K in 1..4 loop accept Signaler; end loop; -- signaux des 4 acteurs
            end AttendreLaFindeTous ; -- Main peut poursuivre
      or
         terminate;
      end select;
   end loop;
end Annonce ;
```

```
--remplacer le paquetage Canal par
task body Canal is
   Libre : Boolean := True ; -- le canal est disponible
   Attend : array(2..5) of Boolean:= False;  -- le processus est bloqué
   Attente : Boolean := False; -- gestion de l'attente ; il y a au moins un processus en attente
   Suivant : Integer := 5;  -- nom d'un processus, index de parcours du tableau Attend
   Choisi : Integer := 0; --indique le suivant à réveiller; si 0, assure l'équité
   function ChoixCandidat return Integer is -- choisit un processus en attente
   begin
      loop
         if Suivant = 5 then Suivant := 2 else Suivant := Suivant + 1; end if;
         exit when Attend(Suivant);
      end loop;
      return Suivant;
   end ChoixCandidat ;

begin
   loop
      select
         accept Ouvrir(I : in Integer) do
            if Libre then Libre := False;
            else
               Attend(I) := True;
               case I is
                     when 2 => requeue Ouvrir2(I : in Integer);
                     when 3 => requeue Ouvrir3(I : in Integer);
                     when 4 => requeue Ouvrir4(I : in Integer);
                     when 5 => requeue Ouvrir5(I : in Integer);
                  end case;
               -- en Ada, peut se faire aussi avec une famille d'entrées (non vues dans le cours)
            end if;
            end Ouvrir;
      or
         when Choisi = 2 => accept Ouvrir2(I : in Integer); Choisi  := 0;
                                    -- Choisi  ne doit servir qu'une fois, on le remet donc à 0
      or
         when Choisi = 3 => accept Ouvrir3(I : in Integer); Choisi  := 0;
      or
         when Choisi = 4 => accept Ouvrir4(I : in Integer); Choisi  := 0;
      or
         when Choisi  = 5 => accept Ouvrir5(I : in Integer); Choisi  := 0;
      or
         accept Fermer(I : in Integer);
            Libre := True;
            -- voir s'il y a au moins un processus en attente
            for K in 2..5 loop Attente := Attend(K); exit when Attente; end loop;
            if Attente then
               Choisi := ChoixCandidat; Libre := False; Attend(Choisi) := False;
               -- comme Libre est remis à False, cela assure l'équité, car
               -- le processus qui rend le canal ne peut pas le reprendre tout de suite
            end if;
      or
         terminate
      end select;
   end loop;
end Canal;
```