CORRIGÉ

EXAMEN DU 19 SEPTEMBRE 2000

partie portant sur l'enseignement des SYSTÈMES INFORMATIQUES
>>

INTERBLOCAGE

Un processus espion dont le travail consiste à mesurer l'activité du système est cyclique et utilise de 1 à 5 granules disques de la manière suivante :

```
task body espion is
begin
   loop
      demander 1 granule (attente si pas disponible) et travailler avec 1 granule ;
      demander 1 autre granule (attente si pas disponible) et travailler avec 2 granules ;
      demander 1 autre granule (attente si pas disponible) et travailler avec 3 granules ;
      demander 1 autre granule (attente si pas disponible) et travailler avec 4 granules ;
      demander 1 autre granule (attente si pas disponible) et travailler avec 5 granules ;
      envoyer le résultat sur le réseau et restituer les 5 granules;
   end loop;
end espion;
```

On réserve treize granules pour les mesures et on voudrait bien installer plusieurs espions.
On constate qu'avec deux espions, il y a toujours assez de granules car 5 + 5 < 13.
Avec cinq espions, il peut y avoir interblocage. Considérons plusieurs états possibles d'allocation représentés par la matrice d'allocation A(t) :
a) A(t1) = (3 3 2 2 1)
b) A(t2) = (3 3 3 2 1)

Question 1

a) *Montrer qu'avec 5 espions, A(t1) est un état fiable et que A(t2) n'est pas un état fiable.*
R(t1) =2 donc espion(1) peut se terminer et rendre 5 granules, puis chaque espion peut se terminer
R(t2) = 1 aucun espion peut se terminer

b) *Peut-on avoir interblocage si on installe 3 espions? Expliquer votre réponse (avec éventuellement un exemple de cas d'interblocage quand il existe).*
Non , car le pire est A(t) = (4 4 4 4) et alors R(t) =1 et n'importe que espion peut terminer, puis rendre ses granules pour les autres

c) *Peut-on avoir interblocage si on installe 4 espions? Expliquer votre réponse (avec éventuellement un exemple de cas d'interblocage quand il existe).*
S'il y a risque d'interblocage, expliquer comment faire pour l'éviter.
Oui par exemple A(t) = (3 3 3 3) R(t) = 1 mais aucun espion ne peut plus terminer
Solution l'algorithme du banquier simplifié : garder toujours de quoi satisfaire une eventuelle demande de l'espion qui a le plus de granules. on sait que son annonce max est 5 et cela pour tous.

REMPLACEMENT DE PAGE

On dispose d'un système de mémoire paginée à la demande et de deux algorithmes A et B. On observe l'exécution d'un programme auquel le système alloue 3 cases de mémoire centrale et qui accède successivement aux pages 1, 9, 3, 8, 9, 1, 3, 9, 8.

Avec l'algorithme A, on constate qu'il y a successivement en mémoire les pages suivantes :

1	1	1	3	3	1	1	1	3
	9	3	8	8	8	3	3	8
		9	9	9	9	9	9	9

Avec l'algorithme B, on constate qu'il y a successivement en mémoire les pages suivantes :

1	1	1	3	3	1	1	1	1
	9	3	8	8	3	3	8	8
		9	9	9	8	8	9	9

Question 2

a) *Lequel des deux algorithmes correspond à l'algorithme FIFO et lequel correspond à l'algorithme LRU? Justifier votre raisonnement*

Déterminer dans chaque cas le nombre total de défauts de page

A est LRU avec 3+4 = 7 défauts de page. En effet quand la page 1 est redemandée, on remplace la page la plus anciennement référencée parmi 3, 8, 9, c'est 3 qui est la victime

B est FIFO avec 3+3 = 6 défauts de page. En effet quand la page 1 est redemandée, on remplace la page la plus anciennement chargée parmi 3, 8, 9, c'est 9 qui est la victime car 9 a été chargée à t=2

b) *Quelles auraient été les pages en mémoire avec l'algorithme Optimal?*
et quel serait le nombre total de défauts de page?

Avec optimal, la victime est la page qui sera référencée le plus loin dans le futur, ou qui ne sera plus jamais référencée

1	1	1	1	1	1	3	3	3
	9	3	8	8	8	8	8	8
		9	9	9	9	9	9	9
victime			3			1		

nombre de défauts de page 3 + 2 = 5

CONCURRENCE ET COHÉRENCE

On veut mettre au point le programme suivant (on a mis côte à côte les deux tâches T1 et T2 pour gagner de la place sur ce texte; dans un programme, elles sont écrites l'une après l'autre).

```
procedure Essai is
                                Total : Integer := 0;
                                Sema : semaphore;          -- E1
   task T1;                                                 task T2;
   task body T1 is                                          task body T2 is
      X, TotalT1 : Integer;                                    Y, TotalT2 : Integer;
   begin                                                    begin
      for I in 1..4 loop                                    for J in 1..4 loop
         P(Sema);                  -- I1                        P(Sema);                  -- J1
         X := Total;               -- I2                        Y := Total;               -- J2
         exit when X >= 4 ;        -- I3                        exit when Y >= 4 ;        -- J3
         X := X + 1;               -- I4                        Y := Y + 1;               -- J4
         Total := X;               -- I5                        Total := Y;               -- J5
         V(Sema);                  -- I6                        V(sema);                  -- J6
      end loop;                                                 end loop;
      imprimer(X);                       -- I7                imprimer(Y);                       --J7
      TotalT1 := Total; imprimer(TotalT1);--I8                 TotalT2 := Total ; imprimer(TotalT2); --J8
   end T1 ;                                                 end T2 ;
begin
   E0(Sema,2);                        -- E2
end Essai;
```

--**Nota** : la notation exit (instructions I3 et J3) fait sortir de la boucle et aller derrière end loop

Après un premier essai où on a fait tourner T1 seul, on imprime X = Total1 = 4 sans problème.

Quelle surprise de voir qu'en exécutant T1 et T2 en concurrence, on imprime des valeurs différentes d'une exécution à une autre.
Pour expliquer les différences, supposer que chaque instruction commentée I1, I2, ..., J7, J8 compte pour une unité, et que l'allocateur de l'unité centrale alloue celle-ci par nombre d'unités à chaque tâche.
1ère exécution : 3 unités pour T1, 3 unités pour T2, 3 unités pour T1, 3 unités pour T2, etc
(on a la séquence I1 I2 I3 J1 J2 J3 I4 I5 I6 J4 J5 J6 I1 I2 I3 J1 J2 J3 I4 I5 I6 J4 J5 J6....)
2ème exécution : 3 unités pour T1, 18 unités pour T2, 3 unités pour T1, 3 unités pour T2, 20 unités pour T1, 5 unités pour T2,
(on a la séquence I1 I2 I3 J1 J2 J3 J4 J5 J6 J1 J2 J3 J4 J5 J6 J1 J2 J3 J4 J5 J6 I4 I5 I6 J1 J2 J3 I1 I2 I3 I4 I5 I6 I1 I2 I3 I4 I5 I6 I1 I2 I3 I4 I5 I6 I7 I8 J4 J5 J6 J7 J8)
3ème exécution : on exécute T1 en entier, puis T2

Question 3
a) *Donner les valeurs imprimées : X, TotalT1, Y, TotalT2, lors de chacune des trois exécutions.*
1ère exécution : X = 4, TotalT1 = 4, Y = 4, TotalT2 = 4
2ème exécution : X= 4, TotalT1 = 4, Y = 2, TotalT2 = 2
3ème exécution : X = 4, TotalT1 = 4, Y = 4, TotalT2 = 4 >>>>>>>>>>
b) *Pour obtenir toujours le résultat de la 3ème exécution, résultat correct, il y a une instruction à corriger. Laquelle et comment?*
Instruction E2 Mettre E0(Sema, 1) et on a une exclusion mutuelle

---

c) On va fonctionner à l'alternat, c'est à dire une itération de T1 (I1 à I6) suivie d'une itération de T2 (J1 à J6) etc., et pour cela :
E1 est remplacée par : Sema1, Sema2 : semaphore;
E2 est remplacée par : E0(Sema1, 1); E0(Sema2, 0);
I1, J1, I6 et J6 sont modifiées

Programmer les instructions I1, J1, I6 et J6 pour obtenir l'exécution à l'alternat. Qu'obtient-on comme valeurs imprimées. Cela peut-il remplacer l'exclusion mutuelle dans tous les cas? Expliquer votre réponse.
I1: P(Sema1); I6 : V(Sema2)
J1 : P(Sema2) J6 : V(Sema1)
Cela ne peut remplacer l'exclusion mutuelle que si les deux processus peuvent fonctionner à l'alternat, donc faire le même nombre de cycles, ce qui est le cas ici. Ce n'est pas généralisable à tous les coups

---

DUO DE PRODUCTEURS POUR UN CONSOMMATEUR

On veut installer une variante du producteur consommateur qui permette à deux producteurs P1 et P2 (processus clients) de partager un même consommateur C (processus serveur), tout en ayant chacun son tampon de dépôt de message (chacun a un tampon de 5 cases). On ajoute en plus que les messages déposés par P1 sont prioritaires par rapport aux messages déposés par P2. Cela donne la gestion de données suivante :

```
procedure Duo is
        T1, T2 : array(0..4) of Message; -- T1 pour P1 et T2 pour P2
        Nombre: Integer := 0; -- pour P1 et C
        -- déclaration des sémaphores;          -- à faire

task P1 ;  -- producteur prioritaire        task P2 ;  -- deuxième producteur
task body P1 is                             task body P2 is
   X1: Message;                                X2 : Message;
```

```
    Queue1 : Integer;:= 0;                    Queue2: Integer := 0;
  begin                                       begin
    loop                                        loop
      preparer(X1);                               preparer(X2);
      T1(Queue1) := X1;                           T2(Queue2) := X2;
      Queue1 := (Queue1 + 1) mod 5;               Queue2 := (Queue2 + 1) mod 5;
      Nombre := Nombre+ 1;                      -- P2 n'incrémente jamais Nombre
    end loop;                                     end loop;
  end P1 ;                                    end P2 ;

  task  C ;       -- c'est le consommateur
  task  body C  is
    Prioritaire : Boolean;              -- permet de choisir entre T1 et T2
    Tete1, Tete2 : Integer := 0; Y : Message;
  begin
    loop
      Prioritaire := Nombre > 0;      -- vrai s'il y a au moins un message dans T1
      if Prioritaire then Nombre := Nombre - 1;  end if;        -- et C va le consommer
      if Prioritaire then
        Y := T1(Tete1); Tete1 := (Tete1 + 1) mod 5;
      else
        Y := T2(Tete2); Tete2 := (Tete2 + 1) mod 5;
      end if;
    end loop;
  end C;
begin
  -- Initialisation  obligatoire des sémaphores-  -- à faire
end Duo;
```

Question 4

Compléter le programme Duo en ajoutant le contrôle de concurrence et la synchronisation par sémaphores (ne mettre que ceux qui sont nécessaires).

```
procedure Duo is
                    T1, T2 : array(0..4) of Message; -- T1 pour P1 et T2 pour P2
                    Nombre: Integer := 0; -- pour P1 et C
                    Plein, Vide1, Vide2, Mutex : Semaphore; -- déclaration des sémaphores;

  task P1;       -- producteur prioritaire        task P2;  -- deuxième producteur
  task body P1 is                                   task body P2 is
    X1: Message;                                      X2 : Message;
    Queue1 : Integer;:= 0;                          Queue2: Integer := 0;
```

```
    begin                                                   begin
        loop                                                    loop
            P(Vide1);                                               P(Vide2);
            preparer(X1);                                           preparer(X2);
            T1(Queue1) := X1;                                       T2(Queue2) := X2;
            Queue1 := (Queue1 + 1) mod 5;                           Queue2 := (Queue2 + 1) mod 5;
            V(Plein);  -- unique semaphore de consommation          V(Plein); -- meme semaphore de consommation
            P(Mutex); -- si l'incrementation n'est pas atomique
            Nombre := Nombre+ 1;                                    -- P2 n'incrémente jamais Nombre
            V(Mutex);
        end loop;                                               end loop;
    end P1 ;                                                end P2 ;


    task  C ;       -- c'est le consommateur
    task  body C is
        Prioritaire : Boolean;                  -- permet de choisir entre T1 et T2
        Tete1, Tete2 : Integer := 0; Y : Message;
    begin
        loop
            P(Plein); -- attendre un message
            P(Mutex); -- exclusion mutuelle pour cohérence de la lecture et decrementation
            Prioritaire := Nombre > 0;      -- vrai s'il y a au moins un message dans T1
            if Prioritaire then Nombre := Nombre - 1;  end if;          -- et C va le consommer
            V(Mutex);
            if Prioritaire then
                Y := T1(Tete1); Tete1 := (Tete1 + 1) mod 5; V(Vide1);
            else
                Y := T2(Tete2); Tete2 := (Tete2 + 1) mod 5; V(Vide2);
            end if;
        end loop;
    end C;
begin
    -- Initialisation  obligatoire des sémaphores
    E0(Vide1, 5); E0(Vide2, 5); E0(Plein, 0); E0(Mutex, 1);
end Duo;
```