Université Bordeaux I
Session de janvier 1998

U.F.R. Mathématiques et Informatique
Alain Griffault

Maîtrise d'Informatique
Systèmes d'exploitation
(Durée : 2 heures)
CORRIGE

Documents interdits

Avertissement

L'exercice 1 est court, mais demande de l'attention.
L'exercice 2 est assez long, mais facile.
L'exercice 3 est assez long. La difficulté consiste à interpréter dans un autre contexte, vos connaissances en SE.
L'ensemble est long, la notation en tiendra compte. (Le barème est indicatif de l'importance respective des exercices).

1 Synchronisation avec des moniteurs : 5 points

Définition : Le type moniteur est une structure de synchronisation de haut niveau, ayant la même puissance de synchronisation que les sémaphores. Un moniteur comprend des variables, des conditions, des procédures et un code d'initialisation. Seules les procédures peuvent être manipulées par les processus, et le mécanisme du moniteur garantit l'exécution de chaque procédure en exclusion mutuelle.

Syntaxe d'un moniteur

type *nom du moniteur* = monitor
    déclarations de variables
    déclarations de conditions
procedure entry *P1(...)*
    begin ... end ;
⋮
procedure entry *Pn(...)*
    begin ... end ;
begin
code d'initialisation
end.

La synchronisation des processus est obtenu à l'aide des procédures *wait* et *signal* appliquées sur les variables de type condition. L'action X.signal provoque le réveil de tous les processus qui ont auparavant effectué X.wait.

Une mauvaise solution au problème de l'exclusion mutuelle

```
type SectionCritique = monitor
        token : booleen;
        sc : condition
procedure entry EntreeSC
        begin
        if token then token := false
        else sc.wait;
        end;
procedure entry SortieSC
        begin
        token := true;
        sc.signal;
        end;
begin
token := true
end.

shared var SC = SectionCritique
```

Code des processus

```
begin
while true do begin
        code de la section non critique
        SC.EntreeSC;
        code de la section critique
        SC.SortieSC
        end
end.
```

Question 1.1 *Une erreur classique en synchronisation de processus s'est glissée dans la solution proposée. Donnez brièvement :*

1. *L'erreur.*
2. *La condition au problème de l'exclusion mutuelle qui n'est pas satisfaite.*
3. *Une correction.*

Réponse 1.1(2 points) Dans la mauvaise solution proposée :

1. L'erreur classique est due au fait suivant :

Dans un algorithme séquentiel		
if (B) alors B ← non B	**est équivalent à**	while (B) do B ← non B

Dans un algorithme parallèle (ou de synchronisation)		
if (B) alors B ← non B	**n'est pas équivalent à**	while (B) do B ← non B

2. Il est possible que plusieurs processus soient simultanément en section critique. Par exemple :
   (a) un premier processus exécute et termine SC.EntreeSC,
   (b) un second exécute SC.EntreeSC : il s'endort,
   (c) un troisième exécute SC.EntreeSC : il s'endort,
   (d) le premier exécute et termine SC.SortieSC, il réveille ainsi le second et le troisième,
   (e) le second termine SC.EntreeSC et entre en section critique,
   (f) le troisième termine SC.EntreeSC et entre en section critique.
3. Une solution consiste à remplacer les lignes suivantes de la procédure EntreeSC

```
if token then token := false
else sc.wait ;
```

   par les lignes :

```
while not token do sc.wait ;
token ← false ;
```

Question 1.2 *Donnez le code des deux procédures* dormir *et* top *du moniteur Reveil suivant, afin qu'un processus puisse se retarder d'un nombre déterminé d'unité de temps. Vous pouvez supposer qu'une horloge matérielle, appelle à intervalle régulier la procédure* top.

Remarque : Des solutions, correctes dans un certain contexte, pourront être considérées comme bonnes si le contexte restrictif d'utilisation est mentionné.

Le moniteur

```
type Reveil = monitor
        déclarations de variables
        déclarations de conditions
procedure entry dormir(...)
        begin ... end ;
⋮
procedure entry top(...)
        begin ... end ;
begin
code d'initialisation
end.
```

```
shared var TicTac = Reveil ;
```

Code des processus

```
var n : integer
begin
while true do begin
        code
        TicTac.dormir(n) ;
        end
end.
```

Code de l'horloge matérielle

```
begin
while true do begin
        code
        TicTac.top ;
        end
end.
```

Réponse 1.2(3 points)
Le moniteur

```
type Reveil = monitor
        NbReveil : integer ;
        delai[] : integer ;
        bip[] : condition
procedure entry dormir(n : integer)
        begin
        delai[NbReveil] ← n ;
        bip[NbReveil].wait
        end ;
procedure entry top
        begin
        for (i = 0, i < NbReveil, i++)
            delai[i] ← delai[i] - 1 ;
        if (delai[i]==0) then bip[i].signal ;
        end ;
begin
NbReveil ← 0
end.
```

Cette réponse est correcte, hormis le fait que la variable NbReveil ne fait que croître, ce qui peut finir par poser un problème. Pour y remédier, il suffit de remplacer les tableaux par des listes, et de décrémenter NbReveil lorsqu'un délai devient nul.

Deux autres réponses ont été acceptées :

1. un seul délai et une seule variable condition. Le contexte d'utilisation est donc restreint à un seul processus.
2. le délai est le paramètre de la procédure dormir, il est décrementé par le processus qui est réveillé à chaque top de l'horloge, et qui vérifie alors s'il vaut zéro. Le temps d'attente du processus est bien celui spécifié, mais il est constitué d'une suite de réveils et d'endormissements.

---

2 Algorithmes de remplacement de pages : 5 points

L'algorithme de la seconde chance est à la base un algorithme de remplacement FIFO, auquel on ajoute un mécanisme pour éventuellement donner un deuxième chance à la page la plus ancienne. Une variante de l'algorithme de la seconde chance utilise les deux bits de référence et de modification. Le bit de modification (1 pour modifiée, 0 sinon) est associé à chaque demande de page et le bit de référence (1 pour récemment utilisée, 0 sinon) est manipulé de la façon suivante :

- Mis à 1 au chargement de la page.
- Mis à 1 à chaque nouvel accès de la page.
- Périodiquement mis à 0 par le système.
- *Suivant l'implémentation choisi, éventuellement mis à 0 par l'algorithme de la seconde chance.*

Les quatre configurations possibles sont assimilables à des niveaux de priorité (P0, P1, P2 et P3).

Question 2.1 *Si P0 correspond au couple (0,0), donnez les deux seules associations réalistes (pour un algorithme de remplacement de pages) entre les niveaux (P0, P1, P2 et P3) et les quatre couples (bit_référence, bit_modification).*

Réponse 2.1(0,5 points) Les associations possibles sont :

Priorité	première association	deuxième association
	(bit_référence, bit_modification)	(bit_référence, bit_modification)
P0	(0,0)	(0,0)
P1	(0,1)	(1,0)
P2	(1,0)	(0,1)
P3	(1,1)	(1,1)

Les autres associations ne sont pas réalistes, car elles cherchent à évincer une page récemment utilisée et modifiée (1,1), au profit d'une qui soit ne serait pas modifiée (1,0), soit n'aurait pas été référencée depuis longtemps(0,1).

Question 2.2 *Proposez un schéma de l'algorithme de la seconde chance qui utilise les notations (P0, P1, P2 et P3).*

Réponse 2.2(1,5 points) Un schéma de l'algorithme de la seconde chance :

```
fonction PageAEvincer retourne page
debut
P ← tete(FIFO) ;
pageCourante ← tete(FIFO) ;
tant que non dernièrePage(FIFO) faire
        si priorite(pageCourante) < priorite(P) alors
                P ← pageCourante;
        pageCourante ← pageSuivante;
retourner P
fin

procedure RemplacementPage(requete)
debut
si requete = top alors
        debut
        pageCourante ← tete(FIFO) ;
        tant que (non dernièrePage(FIFO)) faire
                debut
                bit_reference(pageCourante)← 0 ;
                pageCourante ← pageSuivante;
                fin
        fin
sinon
        debut
        pageCourante ← tete(FIFO) ;
        tant que ((non dernièrePage(FIFO)) et (pageCourante ≠ page(requete))) faire
                pageCourante ← pageSuivante;
        si (pageCourante ≠ page(requete)) alors
                debut
                pageCourante ← PageAEvincer;
                si bit_modification(pageCourante)== 1 alors
                        SauvegarderSurDisque(pageCourante) ;
                EnleverFIFO(pageCourante) ;
                AjouterQueueFIFO(page(requete)) ;
                pageCourante ← QueueFIFO ;
                fin
        si requete = ecriture alors
                bit_modification(pageCourante)← 1 ;
        bit_reference(pageCourante)← 1 ;
        fin
fin
```

Soit la suite de référence suivante, avec une indication supplémentaire indiquant si la page est modifiée. (L pour Lecture, E pour Ecriture), et des "top" indiquant les mises à 0 des bits de références par le SE.

(1,L), (2,E), top, (3,L), (4,L), (3,E), (1,L), top, (3,L), (2,E), top, (5,L), (4,L), (3,E)

Question 2.3 *Donnez pour chacune des deux associations possibles la suite des configurations en supposant que la mémoire est constituée de 3 cadres initialement vides.*

Réponse 2.3(2 points) Dans le tableau suivant, la lettre D (resp. W) indique les défauts de pages, (resp. les écritures forcées avant un défaut de page).

première association (0,0)(0,1)(1,0)(1,1)				deuxième association (0,0)(1,0)(0,1)(1,1)		
Cadre 1	Cadre 2	Cadre 3	référence	Cadre 1	Cadre 2	Cadre 3
D 1 (1,0)			(1,L)	D 1 (1,0)		
1 (1,0)	D 2 (1,1)		(2,E)	1 (1,0)	D 2 (1,1)	
1 (0,0)	2 (0,1)		top	1 (0,0)	2 (0,1)	
1 (0,0)	2 (0,1)	D 3 (1,0)	(3,L)	1 (0,0)	2 (0,1)	D 3 (1,0)
D 4 (1,0)	2 (0,1)	3 (1,0)	(4,L)	D 4 (1,0)	2 (0,1)	3 (1,0)
4 (1,0)	W 2 (0,1)	3 (1,1)	(3,E)	4 (1,0)	2 (0,1)	3 (1,1)
4 (1,0)	D 1 (1,0)	3 (1,1)	(1,L)	D 1 (1,0)	2 (0,1)	3 (1,1)
4 (0,0)	1 (0,0)	3 (0,1)	top	1 (0,0)	2 (0,1)	3 (0,1)
4 (0,0)	1 (0,0)	3 (1,1)	(3,L)	1 (0,0)	2 (0,1)	3 (1,1)
D 2 (1,1)	1 (0,0)	3 (1,1)	(2,E)	1 (0,0)	2 (1,1)	3 (1,1)
2 (0,1)	1 (0,0)	3 (0,1)	top	1 (0,0)	2 (0,1)	3 (0,1)
2 (0,1)	D 5 (1,0)	W 3 (0,1)	(5,L)	D 5 (1,0)	2 (0,1)	3 (0,1)
W 2 (0,1)	5 (1,0)	D 4 (1,0)	(4,L)	D 4 (1,0)	2 (0,1)	3 (0,1)
D 3 (1,1)	5 (1,0)	4 (1,0)	(3,E)	4 (1,0)	2 (0,1)	3 (1,1)
Nombre de défauts de pages						
9				7		
Nombre d'écritures forcées						
3				0		

Question 2.4 *A partir de cet exemple (en comparant le nombre de défauts de page, et le nombre d'écritures forcées sur disque lors d'un remplacement de page), pouvez vous conclure sur les qualités respectives des deux associations.*

Réponse 2.4(1 points) La première association est moins efficace sur cet exemple pour les deux critères considérés. Plus généralement, la première association engendrera toujours un nombre d'écritures forcées supérieur ou égal à la seconde association. Pour le nombre de défauts de pages, il n'y a pas de règle générale. Pour ces raisons, la seconde semble un meilleur choix.

---

3 Interblocages : 10 points

Soit un ensemble de processus et un ensemble de ressources. Il y a un interblocage dans ce système si et seulement si toutes les conditions suivantes sont satisfaites.

Exclusion mutuelle les ressources ne sont pas partageables.

Occupation et attente un processus qui possède déjà une ressource n'est pas obligé de la libérer avant d'en prendre une autre.

Pas de réquisition une ressource ne peut être retirée à un processus pour être allouée à un autre.

Attente circulaire il existe un cycle dans le graphe d'allocation de ressources.

Imaginez la situation de la circulation présentée dans la figure 1. Pour simplifier, vous supposerez que les voies sont toutes à sens unique, et que les véhicules vont tout droit.

Question 3.1 *Après avoir précisé les ressources et les processus, montrez que les quatre conditions nécessaires pour l'interblocage sont réellement vérifiées dans cette situation.*

Réponse 3.1(3 points) Les processus sont les véhicules de ce système, et les ressources sont les portions de route. Il y a ici plusieurs choix possibles :

- un carrefour est une ressource unique.
- un tronçon de route entre deux carrefours peut être vu comme :
  1. un ensemble de N ressources distinctes (N est le nombre de voitures possibles sur ce tronçon).
  2. une ressource en N exemplaires indistinguables.

La figure 2 utilise les notations suivantes pour schématiser la situation.

- l'ensemble de processus $\{P_1, P_2, \ldots, P_{15}\}$

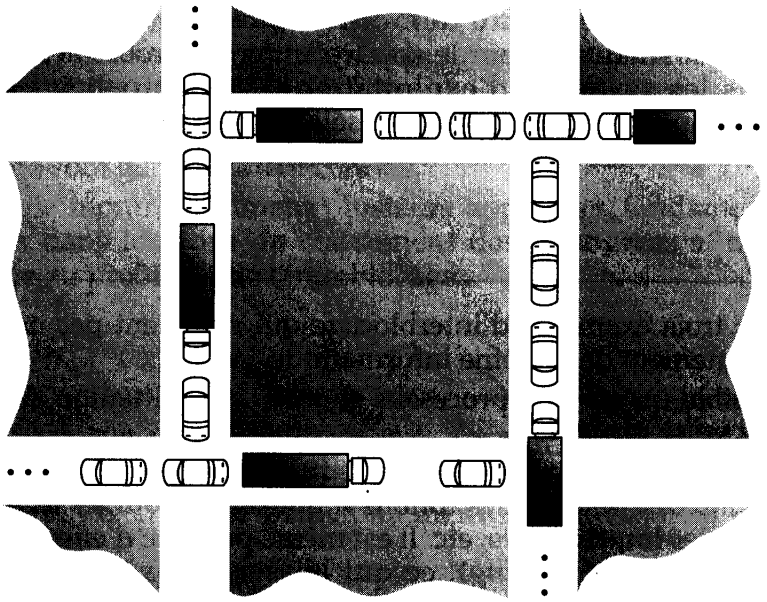

FIG. 1: un problème de circulation.

– l'ensemble de ressources $\{N_1, N_2, N_3, NE, E_1, E_2, E_3, SE, S_1, S_2, S_3, SO, O_1, O_2, O_3, NO\}$
– T pour *Tient* et D pour *Désirée par*.

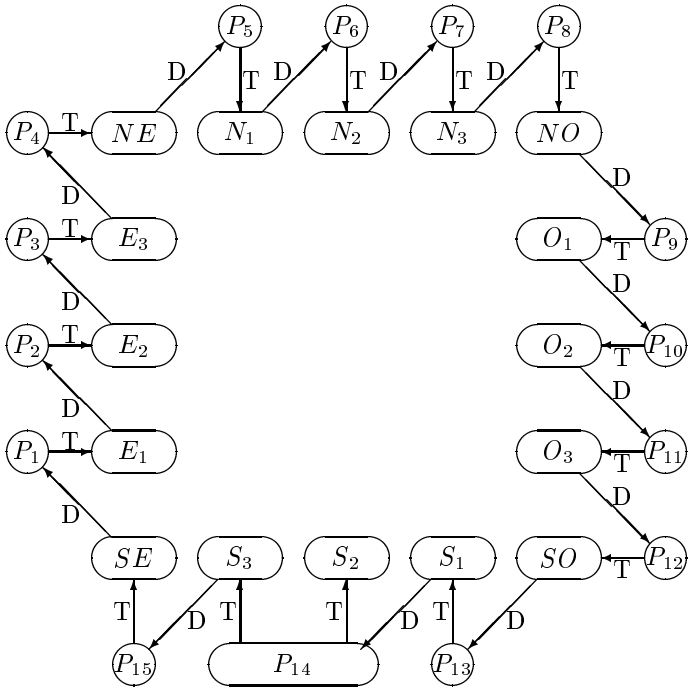

Figure 2: Le graphe d'allocation de ressources.

Les quatres conditions de l'interblocage sont effectivement satisfaites.

Exclusion mutuelle Deux véhicules ne peuvent pas occuper le même emplacement.

Occupation et attente En ce déplaçant, un véhicule occupe un nouvel emplacement avant de libérer l'ancien (les véhicules ne sont pas des wagons d'un train).

Pas de réquisition Un emplacement ne peut pas être réquisitionné (que faire du véhicule qui l'occupe ?).

Attente circulaire La figure 2 contient un cycle dans lequel alternent les arcs T et D.

Le problème de l'interblocage peut être résolu de plusieurs façons :

- Par **prévention** en veillant à ce qu'une des quatre conditions soit toujours fausse.
- Par **évitement** en veillant à ce que le système soit toujours dans un état sain, (définition vue en cours et TD).
- Par **correction** après détection d'un interblocage.

Question 3.2 *Pour chacune des trois méthodes, donnez si possible une règle correspondante pour notre problème de circulation.*

Réponse 3.2(2 points) Pour notre problème de circulation, les différentes méthodes s'interpretent ainsi :

- La **prévention** est **impossible** pour notre problème de circulation.

  Exclusion mutuelle Il est impossible de rendre les ressources partageables. Deux véhicules ne peuvent pas occuper le même emplacement. Rendre obligatoire l'utilisation de véhicules plus petit, ou bien élargir la chaussée ne résout évidemment pas le problème.

  Occupation et attente Les véhicules n'étant pas des wagons d'un train, il faut que le véhicule qui occupe la place que l'on désire la libère, afin qu'on puisse l'occupée pour libérer à son tour celle que l'on occupait. Il est donc impossible d'empêcher cette condition.

  Pas de réquisition Un emplacement ne peut pas être réquisitionné (que faire du véhicule qui l'occupe ?).Il est donc également impossible d'empêcher cette condition.

  Attente circulaire Il est impossible de trouver un ordre total sur les quatre ressources $\{NE, SE, SO, NO\}$ qui soit respecté par les parcours de tous les véhicules.

- L'**évitement** consiste à ne prendre une ressource que si l'on est certain que cela ne risque pas de provoquer ultérieurement un interblocage. Pour notre exemple, la seule situation malsaine est, si l'on applique l'algorithme du banquier, celle de l'interblocage déssiné. Dans toutes les autres situations, il est toujours possible de faire avancer les voitures sans provoquer un interblocage.

  L'évitement est donc **possible** pour notre problème de circulation. Il suffit pour cela de ne pas s'engager dans un carrefour si l'on est pas certain de pouvoir en sortir (ie, il faut que la ressource suivante dans la direction soit également libre).

- La **correction** après détection d'un interblocage est réalisée soit par libération de ressources (une grue qui soulève la voiture), soit par destruction de processus (OUPS ! !). La correction est donc à priori **impossible**, en pratique, elle est pourtant **possible** et réalisée en *tassant* légérement quelques véhicules. Cela correspond à créer temporairement des ressources supplémentaires.

Dans le code de la route français, il est dit qu'un conducteur ne peut s'engager dans un carrefour, que s'il est certain de ne pas y rester bloqué.

Question 3.3 *Expliquez brièvement sur quelle méthode est basé le code de la route français ?*

Réponse 3.3(1 points) Le code de la route français, qui spécifie qu'un conducteur ne peut s'engager dans un carrefour, que s'il est certain de ne pas y rester, cherche à maintenir le système dans une situation, à partir de laquelle, il existe une possibilité d'évolution. Il est donc basé sur l'évitement.

Question 3.4 *Le respect du code de la route transforme le problème de l'interblocage en un autre : lequel ?*
Donnez un exemple de situation dans laquelle ce nouveau problème est présent.
Dites pourquoi le code de la route (malgré ce problème) est sans doute une solution acceptable.

Réponse 3.4(2 points) La condition, qui spécifie qu'un conducteur ne peut s'engager dans un carrefour, que s'il est certain de ne pas y rester, est une contrainte purement locale, légérement différente du calcul des états sains de l'algorithme du banquier.

Cette contrainte possède malheureusement deux défauts :

1. Elle peut empêcher des voitures d'avancer alors que la situation résultante est saine.
2. Elle peut conduire à une situation malsaine.

Ces deux défauts du code de la route transforment le problème de l'interblocage en un problème de famine générale.

Dans certaine situation, afin de respecter le code de la route, personne ne doit avancer, mais si un seul ne respecte pas cette règle, tous les véhicules peuvent ensuite avancer. C'est un problème d'élection, il faut choisir celui qui doit enfreindre le code de la route.

Pour avoir cette situation, il suffit de compléter la situation précédente de la façon suivante :

- une route de SE par le Sud vers SO, avec tous les emplacements occupés par des voitures.
- une route de NO par le Nord puis par l'Est vers SE, avec tous les emplacements occupés par des voitures.
- une route de SO par l'Ouest vers NO, avec tous les emplacements occupés par des voitures.
- une route de NE par L'Est puis le Nord vers NE, avec tous les emplacements occupés par des voitures.
- Les quatre carrefours libres de véhicules.

La situation précédente (la famine générale) est fortement improbable. Le respect de la code de la route qui évite l'interblocage avec des contraintes purement locales est donc une solution acceptable.

Question 3.5 *Proposez une solution au problème de l'interblocage pour la circulation routière. (Cette solution peut-être nouvelle, ou bien basée sur vos connaissances en systèmes d'exploitation).*

Réponse 3.5(2 points) Le problème est du à une non connaissance de la situation globale. Pour construire le graphe d'allocation global des ressources, et activer un algorithme d'élection lorsque le problème de famine générale survient, il suffit que chaque voiture émette vers un satellite sa position et sa direction. La construction du graphe d'allocation est alors possible, et le choix d'un véhicule qui ne respecte pas le code de la route aussi.

Les agents de la circulation reliés par un réseau de talkie-walkie peuvent également remplacer le satellite. Ils peuvent en effet reconstituer à partir de leur connaissance locale, une approximation du graphe global d'allocation des ressources ; et ainsi, prendre la bonne décision.

La construction d'ouvrages d'arts (ponts, tunnels) est également une solution. Cette idée correspond à un ajout de ressources, afin de pouvoir construire l'ordre total sur les ressources, et prévenir la présence de cycle dans le graphe.