Université Bordeaux I
Session de janvier 2000

U.F.R. Mathématiques et Informatique
Alain Griffault

Maîtrise d'Informatique
Systèmes d'exploitation
(Durée : 3 heures)

Documents interdits

1 Cours : 2 points

Question 1.1 *Expliquez le problème de la liaison d'adresses (adresse logique ↔ adresse physique). Que signifient les termes* code absolu, code translatable, code relogable *?*

2 Algorithme du banquier : 5 points

Dans l'algorithme du banquier, chacun des processus exécute entre autres choses le code suivant :

```
Processus Pi
debut
  envoyer(banquier, annonceDesBesoinsMaximaux); // toujours autorisé
  Pour chaque vecteur V de ressources désirées
     envoyer(banquier,requete(V));             // soumis à un accord
     recevoir(banquier,accord);
  Pour chaque vecteur V' de ressources libérées
     envoyer(banquier,liberer(V'));            // toujours autorisé
fin
```

et le code du processus `banquier` ressemble à :

```
Processus banquier
debut
  tant que vrai faire debut
    recevoir(Pi, annonceDesBesoinsMaximaux);
      ou
    recevoir(Pi, liberer(V'));
      ou
    recevoir(Pi, requete(V));
    Suivant le résultat de l'algorithme du banquier:
      envoyer(Pi, accord);
        ou
      stocker la requete(V) de Pi
  fin
fin
```

Soit un système dont les ressources sont gérés par un processus banquier. Supposons maintenant que les événements suivants soient possibles :

Evt 1. un processus peut être créé dynamiquement par un processus P_i.

Evt 2. un processus peut être détruit dynamiquement par un processus P_i.

Evt 3. un processus P_i peut modifier son annonce maximale en cours d'exécution.

Question 2.1 *Dites pour chacun de ces événements, s'ils doivent être soumis à l'autorisation du banquier. Justifiez brièvement.*

Question 2.2 *Pensez vous qu'il soit nécessaire de modifier l'algorithme du banquier pour traiter ces nouveaux événements. Justifiez et/ou donnez les caractéristiques des modifications.*

3 Sémaphores : 5 points

Une ligne de chemin de fer reliant deux villes A et B comporte une section à voie unique

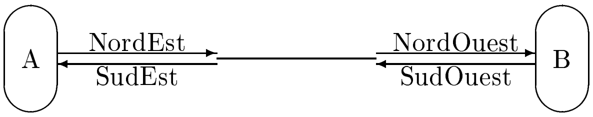

Figure 1: Une ligne de chemin de fer

On représente les trains par des processus :

```
Processus trainsA-B
debut
  NordEst
  ``Protocole d'entree''
  VoieUnique
  ``Protocole de sortie''
  NordOuest
fin
```

```
Processus trainsB-A
debut
  SudOuest
  ``Protocole d'entree''
  VoieUnique
  ``Protocole de sortie''
  SudEst
fin
```

Deux contraintes doivent toujours être satisfaites :

C 1. Pas de collision (c-à-d un `trainsA-B` et un `trainsB-A` en même temps sur la voie unique).

C 2. La voie unique ne reste pas indéfiniment inutilisée.

et si possible la contrainte suivante serait un plus :

C 3. Aucun train n'attend indéfiniment longtemps avant d'entrer sur la voie unique.

Definition 3.1 *Un sémaphore est une structure de données*

```
type semaphore = (n : entier; l : liste de processus);
```

manipulée uniquement par les actions atomiques `wait` *et* `signal`.

```
procedure wait(var S : semaphore)
debut
S.n := S.n - 1;
si (S.n < 0) alors debut
        ajouter(moi,S.l);
        block;
        fin
fin
```

```
procedure signal(var S : semaphore)
debut
S.n := S.n + 1;
si (S.n <= 0) alors debut
        enlever(P,S.l);
        wakeup(P);
        fin
fin
```

En s'inspirant du problème des lecteurs-rédacteur, donnez pour chacune des questions suivantes, les sémaphores nécessaires (ainsi que leurs valeurs initiales), les variables partagées utilisées (ainsi que leurs valeurs initiales), les codes des processus `trainsA-B` et `trainsB-A`, et les contraintes satisfaites par votre solution.

Question 3.1 *On suppose que le nombre de trains présents au même instant sur la voie unique est limité à un.*

Question 3.2 *On suppose que le nombre de trains présents au même instant sur la voie unique n'est pas limité.*

Question 3.3 *On suppose que le nombre de trains présents au même instant sur la voie unique est limité à une constante N.*

4 Transactions et verrouillage deux phases : 8 points

Une transaction est une suite d'actions élémentaires (*lire* et *ecrire*). On s'intéresse aux exécutions possibles d'un ensemble de transactions $\{T_i\}$ qui accèdent à un ensemble d'objets communs.

un exemple Soient $T_1 = \{ecrire(a,1); ecrire(b,1)\}$ et $T_2 = \{ecrire(a,2); ecrire(b,2)\}$ deux transactions. Nous avons un ensemble de six exécutions possibles :

E1 : $\{ecrire(a,1); ecrire(b,1); ecrire(a,2); ecrire(b,2)\}$

E2 : $\{ecrire(a,1); ecrire(a,2); ecrire(b,1); ecrire(b,2)\}$

E3 : $\{ecrire(a,1); ecrire(a,2); ecrire(b,2); ecrire(b,1)\}$

E4 : $\{ecrire(a,2); ecrire(a,1); ecrire(b,1); ecrire(b,2)\}$

E5 : $\{ecrire(a,2); ecrire(a,1); ecrire(b,2); ecrire(b,1)\}$

E6 : $\{ecrire(a,2); ecrire(b,2); ecrire(a,1); ecrire(b,1)\}$

Les exécutions E1 et E6 sont dites séquentielles, les autres parallèles (ou entrelacées).

Definition 4.1 *Soient A_i et A_j deux actions d'une même exécution. A_i dépend de A_j si et seulement si :*

- *A_i est après A_j,*
- *A_i et A_j accèdent au même objet,*
- *A_i ou A_j* **au moins** *est une écriture.*

Cette relation définit entre les actions d'une exécution un graphe de dépendance étiqueté et orienté. Chaque arc du graphe sera noté $A_i^m \xrightarrow{o} A_j^n$ ou o est l'objet sur lequel porte le conflit et A_k^p représente la $p$ème actions de la transaction T_k

Question 4.1 *Soit une exécution d'un ensemble de transaction $\{T_i\}$. Montrez que son graphe de dépendance est sans cycle.*

Question 4.2 *Montrez que si deux exécutions de l'ensemble $\{T_i\}$ produisent le même graphe de dépendance, alors elles ont le même effet (c-à-d les valeurs des objets à la fin des deux exécutions sont identiques).*

Definition 4.2 *Une exécution d'un ensemble de transactions $\{T_i\}$ est cohérente si et seulement si il existe une exécution séquentielle (c-à-d un ordonnancement des transactions) qui produit le même effet sur les objets.*

Le verrouillage à deux phases est une méthode pour interdire certaines exécutions entrelacées afin que les seules exécutions possibles d'un ensemble de transactions $\{T_i\}$ soient cohérentes.
Une transaction **avec verrouillage** T_i est maintenant une suite A_i^m d'actions élémentaires (*lire*, *ecrire*, *verrouiller*, *deverrouiller*). Intuitivement, on n'accède qu'aux objets verrouillés dans un mode compatible. Le mode *ecriture* étant exclusif et le mode *lecture* partagé. Plus formellement :

- $A_i^m = verrouiller(o, lecture)$ est acceptée si et seulement si o n'est pas verrouillé en *ecriture* par une autre transaction. La transaction est mise en attente sinon.
- $A_i^m = verrouiller(o, ecriture)$ est acceptée si et seulement si o n'est pas verrouillé ni en *lecture* ni en *ecriture* par une autre transaction. La transaction est mise en attente sinon.
- $A_i^m = deverrouiller(o)$ est toujours acceptée.
- $A_i^m = lire(o) \Rightarrow \begin{cases} \exists n < m \ / \ A_i^n = verrouiller(o, mode), mode \in \{lecture, ecriture\} \\ \forall p \ / \ n < p < m, A_i^p \neq deverrouiller(o) \end{cases}$
- $A_i^m = ecrire(o) \Rightarrow \begin{cases} \exists n < m \ / \ A_i^n = verrouiller(o, ecriture) \\ \forall p \ / \ n < p < m, A_i^p \neq deverrouiller(o) \end{cases}$

Une transaction avec verrouillage est dite **à deux phases** si de plus elle respecte les règles suivantes :

- $A_i^m = verrouiller(o, mode) \Rightarrow \exists p \ / \ p > m, A_i^p = deverrouiller(o)$
- $A_i^m = deverrouiller(o) \Rightarrow \forall o', \forall p \ / \ p > m, A_i^p \neq verrouiller(o', mode)$

Intuitivement, tous les objets doivent être déverrouillés, et dès lors que l'on a commencé à déverrouiller un objet, on ne peut plus en verrouiller aucun. **La seconde phase d'une transaction débute donc avec le premier déverrouillage.**
Dans toute la suite de l'exercice, les transactions T_i sont supposées avec verrouillage à deux phases.

Question 4.3 *Soit $A_i^{d_i}$ l'action qui débute la seconde phase de T_i.*
Montrez que le graphe de dépendance d'une exécution d'un ensemble de transactions $\{T_i\}$ est identique à celui correspondant à l'exécution séquentielle des $\{T_i\}$ suivant l'ordre chronologique des actions $\{A_i^{d_i}\}$.

Question 4.4 *Toutes les exécutions d'un ensemble de transactions $\{T_i\}$ sont-elles cohérentes ? Justifiez.*

Question 4.5 *Le verrouillage à deux phases possède un gros défaut. Lequel ?*

Definition 4.3 *Le schéma* `Wait-Die` *pour éviter l'interblocage s'appuie sur un technique sans réquisition. Lorsqu'un processus P_i désire une ressource détenue par P_j, soit il attend s'il est plus vieux, soit il meurt s'il est plus jeune.*

Definition 4.4 *Le schéma* `Wound-Wait` *pour éviter l'interblocage s'appuie sur un technique avec réquisition. Lorsqu'un processus P_i désire une ressource détenue par P_j, soit il attend s'il est plus jeune, soit il blesse P_j en lui prenant ses ressources s'il est plus vieux.*

Question 4.6 *Dans le cas de transactions vues comme des processus, quel est le problème posé par la mort et/ou la blessure d'une transaction. Proposez une solution.*

Question 4.7 *Listez les besoins et les problèmes rencontrés pour la mise en œuvre de ces deux schémas. Lequel des deux schémas précédents vous semble le mieux adapté dans le cas du verrouillage à deux phases ? Justifiez.*