Systèmes Concurrents

1h30, documents autorisés

6 janvier 2023

Les réponses doivent être justifiées.

1 Programmation concurrente (4 points)

1. On considère un système avec quatre variables partagées et trois activités concurrentes. L'accès à chaque variable partagée x, y, z est protégé par un verrou d'accès exclusif $lockx$, $locky$, $lockz$. Le code des trois activités est le suivant :

$lockx.lock();$	$locky.lock();$	$lockz.lock();$
$lockz.lock();$	$lockx.lock();$	$locky.lock();$
$x \leftarrow x + z;$	$y \leftarrow x * y;$	$z \leftarrow z - y;$
$lockx.unlock();$	$lockx.unlock();$	$locky.unlock();$
$lockz.unlock();$	$locky.unlock();$	$lockz.unlock();$

   (a) Quel problème ce code présente-t-il ?

   (b) Proposer *deux* solutions pour corriger le problème.

2. Quel est l'intérêt d'utiliser des pools de threads, plutôt que de créer à la demande des activités ?

3. On cherche le maximum dans un tableau d'entiers non trié. Pour paralléliser la recherche est-il préférable d'utiliser un pool de threads ou un schéma fork/join ?

2 Moniteur (8 points)

Rappel du problème du barbier. Il s'agit de modéliser l'activité coordonnée d'un barbier et de ses clients. Le barbier dispose d'une pièce avec un unique siège. Le siège ne peut accueillir qu'un client à la fois : lorsqu'un client se présente et trouve le siège libre, il peut s'installer pour se faire raser ; sinon il attend que la place se libère. Un client installé sur le siège ne le quitte qu'une fois rasé. Un client suivant peut alors s'installer pour se faire raser.

Ajout au problème. Le barbier a du succès et embauche un apprenti pour l'assister. Tous les trois clients rasés, l'apprenti passe le balai. Pendant que l'apprenti passe le balai, un client ne peut pas entrer et s'asseoir.

Simplifications. On considère le problème sans salle d'attente ni abandon des clients.

Code des activités.

```
ClientRaseur() {
  boucle
    s'asseoir dans le siège
    ...
    se lever et partir
    ...
  finboucle
}
```

```
Barbier() {
  boucle
      débuter rasage
      ...
      terminer rasage
      ...
  finboucle
```

```
 Apprenti() {
   boucle
       débuter balayage
       ...
       terminer balayage
       ...
   finboucle
```

Questions Réaliser le moniteur en suivant la démarche présentée en cours et TD :

1. Écrire (en français) les conditions d'acceptation des six opérations.
2. Définir les variables d'état permettant de représenter ces conditions.
3. Établir un invariant du moniteur liant ces variables d'état.
4. Définir les variables conditions nécessaires.
5. Programmer les opérations du moniteur. Préciser (sous forme de commentaire dans le code) les pré/post conditions des signaler/attendre.

Extension L'apprenti en formation acquiert de l'autonomie et vient aider du barbier quand il est débordé. Un second siège est installé et quand il y a plus de 5 clients en attente, l'apprenti rase un client.

Questions Indiquer les modifications à effectuer sur la version précédente, en particulier :

6. le code de l'apprenti ;
7. les éventuelles nouvelles variables d'état ;
8. les éventuelles nouvelles variables conditions ;

Il n'est pas demandé de développer intégralement le moniteur jusqu'au code.

3 Processus communicants / tâche Ada (4 points)

Rappel du problème du barbier. La boutique du barbier comporte un salon avec un unique siège, et une salle d'attente avec N chaises. Lorsqu'un client se présente à la boutique, il entre dans la salle d'attente s'il y a des chaises libres, sinon il attend dehors. Quand le siège est libre, un client s'y assoit pour se faire raser. Quand le barbier a fini de raser le client, ce dernier quitte la boutique et le client suivant peut quitter la salle d'attente pour entrer dans le salon et se faire raser.

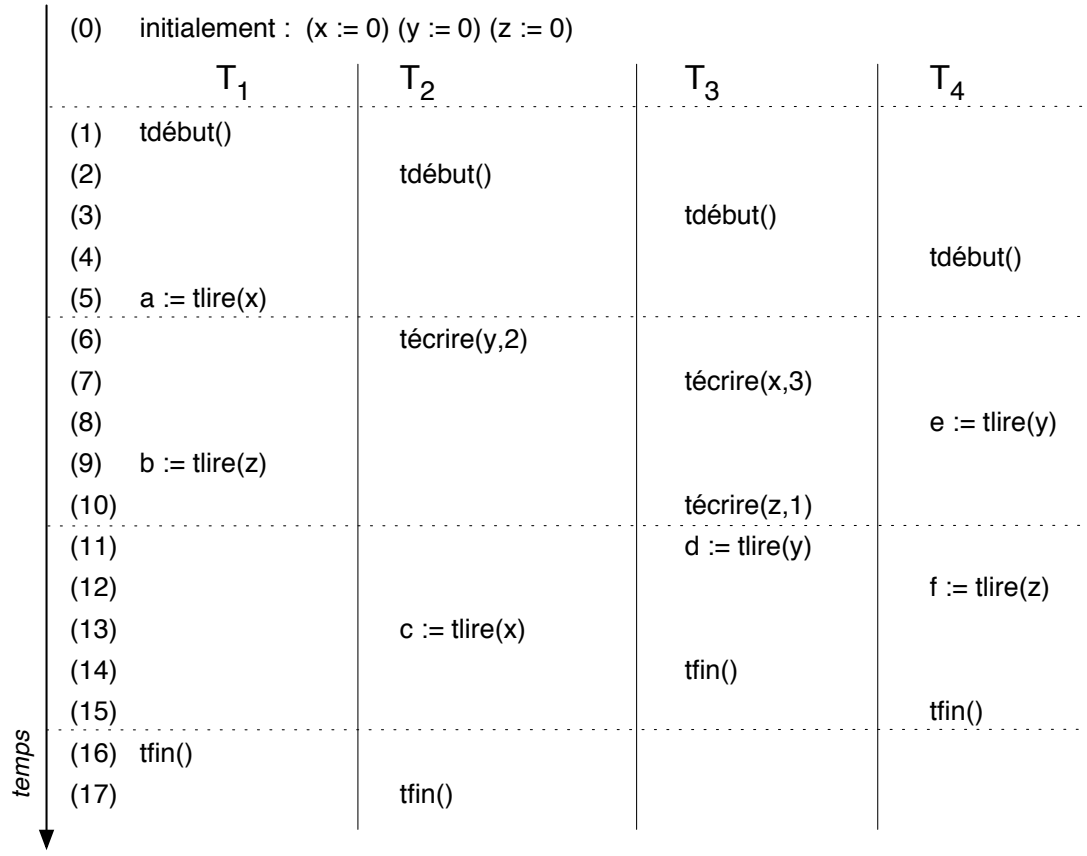
L'objectif est de proposer une solution à ce problème avec des canaux de communication ou des tâches Ada (au choix). Les activités sont d'une part les clients, d'autre part le barbier et la boutique (serveurs).

Questions

1. Déterminer l'interface du barbier et de la boutique (canaux de communication ou point d'entrée de tâche).
2. Donner le code d'un client.
3. Donner le code du barbier.
4. Donner le code de la boutique.

4 Transactions (4 points)

Le chronogramme suivant représente l'exécution de quatre transactions T_1, T_2, T_3, T_4 opérant sur les variables partagées x, y et z. Les variables a, b, c, d, e, f sont locales à respectivement T_1, T_2, T_3, T_4. Dans ce chronogramme, le temps réel s'écoule de haut en bas, l'opération tdébut() lance une transaction, l'opération tfin() demande la validation de la transaction qui l'appelle.



1. Construire le graphe de dépendance correspondant à cette exécution :
   (a) dans le cas où l'on utilise une stratégie de propagation directe des écritures ;
   (b) dans le cas où l'on utilise une stratégie de propagation différée des écritures.

   Préciser pour chacun des arcs le numéro de l'opération entraînant la création de l'arc, et indiquer, en justifiant la réponse, si cette exécution est sérialisable.
2. On suppose que l'exécution des transactions est contrôlée selon un protocole de contrôle de concurrence. L'ordre d'exécution local à chaque transaction reste le même (tant que la transaction n'est pas abandonnée) mais l'ordre relatif des opérations de transactions distinctes peut être différent, dans la mesure où lorsqu'une transaction est bloquée, ses opérations sont retardées.
   (a) Indiquer le résultat final de cette exécution avec un protocole de contrôle de concurrence par certification et une propagation différée des écritures : dernière valeur des variables x, y et z, transactions validées/abandonnées/bloquées. Justifier la réponse en précisant la cause des blocages ou des abandons. Donner l'ordre série équivalent correspondant à cette exécution.
   (b) Même question dans le cas d'un contrôle de concurrence continu par estampilles, en supposant que les estampilles sont attribuées selon l'ordre de lancement des transactions.
   (c) Même question dans le cas d'un contrôle de concurrence continu par verrouillage à deux phases strict.