Conception des systèmes concurrents

2IR, novembre 2014
1h45, documents autorisés

Les exercices sont (techniquement) indépendants. Barème sous réserve.

Axel, Baptiste et Charline décident de s'inscrire à un club d'aviron, afin de pratiquer l'aviron par équipes. Malheureusement, il ne reste plus qu'un bateau disponible, disposant de trois places, mais de seulement quatre rames. Or il est impératif qu'un rameur dispose de deux rames (sinon l'embarcation tourne en rond).

Axel propose donc d'établir un protocole qui devrait permettre à chacun de tirer le meilleur parti du matériel qui est mis à leur disposition par le club :

Tout d'abord, les rames seront disposées au centre de l'embarcation, à portée de chacun des rameurs potentiels. Les rameurs suivront alors le protocole suivant :

```
répéter
   prendreUneRame(); //bloquant si plus de rame disponible
   prendreUneRame(); //bloquant si plus de rame disponible
   répéter 50 fois
      attaque();
      poussée();
      retour();
   fin répéter;
   poserUneRame();
   poserUneRame();
   récupérerUnPeu();
jusqu´ à arrivé();
```

1 Interblocage (2 points)

1. (1 point) Expliquez pourquoi ce protocole garantit l'absence d'interblocage.

De retour de leur entraînement, Axel, Baptiste et Charline ont une bonne surprise : Paul le Poulpe aimerait se joindre à leur équipe, ce qui devrait améliorer nettement les performances (car Paul dispose de 8 bras et du soutien du club, désireux de lutter contre toutes les exclusions). Cependant, il faut tout d'abord généraliser le protocole précédent, afin gérer une équipe de R rameurs, chaque rameur utilisant un nombre spécifique de rames.

Ce protocole généralisé est :

```
répéter
   répéter monNombreDeBras() fois
      prendreUneRame(); //bloquant si plus de rame disponible
   fin répéter;
   répéter 50 fois
      attaque();
      poussée();
      retour();
  fin répéter;
  répéter monNombreDeBras() fois
      poserUneRame();
  fin répéter;
  récupérerUnPeu();
jusqu´ à arrivé();
```

2. (1 point) Pour une équipe donnée,
   — R désigne le nombre de rameurs
   — B désigne le nombre total de bras de l'ensemble des rameurs de l'équipe.
   Par exemple, dans l'équipe constituée par Axel, Baptiste, Charline et Paul le Poulpe, on a $R = 4$ et $B = 14$.
   Exprimez le nombre minimum de rames nécessaires pour que le protocole généralisé soit exempt d'interblocage, en fonction de R et de B

2 Threads Java et sémaphores (6 points)

On se propose de réaliser le code de ce protocole en Java, en utilisant les threads et le sémaphores de cette plateforme.

Pour cela, on définira

— une classe *Bateau*, fournissant
   — un constructeur avec un paramètre : le nombre de rames.
   — les méthodes *prendreUneRame()*, *attaque()*,*poussée()*, *retour()*, *poserUneRame()*, *récupérerUnPeu()*, *arrivé()*.

— une classe *Rameur*, fournissant
   — un constructeur avec trois paramètres : le nom (une chaîne), le nombre de bras, le bateau dans lequel le rameur a pris place.
   — une méthode *monNombreDeBras*(), qui renvoie le nombre de bras du rameur.
   — une méthode réalisant le code du protocole proprement dit.

— une classe principale *Course*, qui crée un bateau et l'ensemble de rameurs qui prendront place à son bord, puis lance les activités correspondant aux rameurs, et enfin attend la fin de la dernière activité lancée pour afficher un message "Terminé". Pour

simplifier, on suppose qu'il n'y a pas de contrainte sur le nombre de places disponibles dans un bateau.

3. (3 points) Donner le code de la classe principale *Course* et celui de la classe *Rameur*, pour l'exemple (uniquement) de l'équipe constituée par Axel, Baptiste, Charline et Paul le Poulpe. La classe *Bateau* sera instanciée et référencée, mais son code sera défini dan la question suivante : il suffit de la représenter ici par une classe vide.
4. (3 points) Donner le code de la classe *Bateau*, en utilisant les sémaphores pour coordonner l'accès aux rames. On se limitera à la définition des attributs de la classe, de son constructeur, et des deux opérations *prendreUneRame*() et *poserUneRame*(). Les autres opérations (*attaque*(), ...) resteront vides.

3 Moniteurs de Hoare (7 points)

Le protocole mis en place entraîne des difficultés importantes au niveau de la gestion des rames : il apparaît qu'il y peut y avoir moins de rames que de bras disponibles. En outre, la distribution et la collecte quotidienne des rames entraînement un surcroît de travail pour les organisateurs. Le club a donc pris une mesure drastique : tous les bateaux seront équipés d'un même nombre fixe *NbRames* de rames (supérieur au nombre maximum de bras de tout membre du club, car il faut que tous puissent pratiquer).

5. (1 points) Cette nouvelle organisation entraîne des risques d'interblocage. Donnez deux **principes** d'adaptation du protocole généralisé qui, dans cette situation, permettraient de **prévenir** l'interblocage.

Le choix est fait, qui va dans le sens d'une meilleure ergonomie du protocole, de remplacer les méthodes *prendreUneRame*() et *poserUneRame*() par deux méthodes prendreRames(int r) et poserRames(int r). Ces méthodes éviteront aux rameurs d'avoir à exécuter une boucle pour obtenir ou restituer leurs rames. On suppose par ailleurs que chaque rameur demande **en bloc la totalité** des rames dont il a besoin.

On suppose que l'on dispose de **moniteurs de Hoare** (exécution automatique en exclusion mutuelle des opérations, priorité au signalé, files FIFO associées aux variables condition).

Définir un moniteur fournissant les opérations prendreRames(int r) et poserRames(int r), de manière à

— éviter l'interblocage,

— assurer qu'à tout moment **le plus possible de rameurs** puissent s'exercer

On réalisera le moniteur en suivant la démarche vue en TD, basée sur l'évaluation de conditions d'état pour contrôler la progression des processus.

6. (1 point) Ecrire (en français) les conditions d'acceptation des opérations prendreRames(int r) et poserRames(int r).
7. (1 point) Définir les variables d'état permettant de représenter ces conditions ; ainsi qu'un invariant du moniteur liant ces variables d'état

8. (3 points) Programmer les opérations prendreRames(int r) et poserRames(int r) du moniteur. Préciser (sous forme de commentaire dans le code) les pré/post conditions des signaler/attendre.
9. (1 point) Ce protocole d'accès présente-t-il un risque de famine ? Justifiez votre réponse.

4 Processus communiquants (3 points)

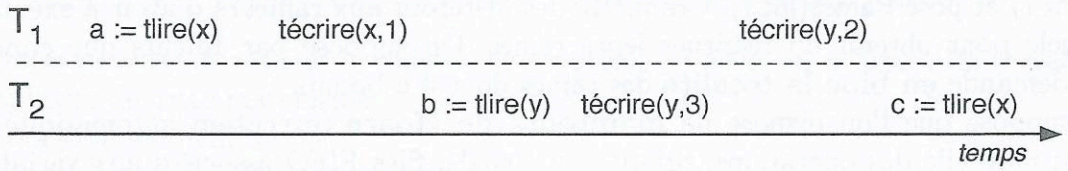
A l'initiative de Paul le Poulpe, et sous la pression d'une partie de l'équipe de développement, qui estime que les moniteurs de Hoare restent une notion théorique, qui manque de support pratique, les dirigeants du club demandent le développement d'une nouvelle version, en Ada.

Cette version doit éviter l'interblocage, mais aussi (et c'est sur ce point que Paul le Poulpe se trouve concerrné) garantir une forme d'**équité** dans l'accès aux rames.

10. (2 points) Construire une tâche serveur Ada répondant à ces nouvelles contraintes, en suivant une méthodologie au choix (conditions d'acceptation ou automate).
11. (1 point) Justifiez le fait que la solution proposée est vivace.

5 Transactions (2 points)

Le chronogramme suivant représente l'exécution suivante de deux transactions T_1 et T_2 opérant sur les variables partagées x et y. Les variables a, b, c sont locales à T_1 ou à T_2.Dans ce chronogramme, le temps réel s'écoule de gauche à droite, et les opérations sont placées par rapport à ce temps réel.



12. (2 points) Construire le graphe de dépendance correspondant à cette exécution. Précisez pour chacun des arcs l'opération entraînant la création de l'arc, et indiquer, en justifiant votre réponse, si cette exécution est sérialisable.