Université de Provence
MASTER INFORMATIQUE
Programmation Parallèle
et Distribuée
2007-2008

Programmation Parallèle et Distribuée

Examen – jeudi 15 mai 2008
3h00

Aucun document autorisé. Les réponses doivent être argumentées.
Les parties sont indépendantes. Le barème est indicatif.

A Programmation Parallèle

A.1 Qu'est-ce qu'une machine PRAM ? Préciser les différents mode d'accès possibles à la mémoire partagée. *(1 point)*

A.2 Etant donné un tableau d'entiers de taille t et un entier p, on cherche à déterminer si p appartient au tableau.

Proposer un algorithme CREW en pseudo-code pour une machine PRAM à n éléments. (on pourra ne proposer une solution que pour des valeurs particulières de n et t)

Rappeler la définition de la complexité et de l'efficacité. Quelles en sont les valeurs pour votre algorithme ? *(3,5 points)*

B Correction de Programme

B.3 Le programme `Mutex` donné en annexe est-il correct pour l'exclusion mutuelle ? (on considérera qu'on a adopté les mêmes conventions que dans le cours) *(1,5 points)*

B.4 Montrer, par la méthode des annotations de code, que le programme Java suivant est correct pour le calcul de la division euclidienne de x par y.

```
q = 0;
while ( x >= y ) {
   q=q+1;
   x=x-y;
}
```

On souhaite accélérer ce calcul en utilisant une station bi-processeur et des techniques de parallélisation. Supposons donc que l'on utilise deux instances du programme précédent (initialisation `q=0` mise à part). Le système est-il encore correct ? Quelle(s) méthode(s) générique(s) peut-on utiliser pour prouver de tels systèmes ? *(3 points)*

Proposez éventuellement une correction de ce code. Observera-t-on effectivement l'amélioration du temps de calcul attendue ?

C Synchronisation JAVA

C.5 Proposer une définition formelle du "problème des producteurs et consommateurs" (bien spécifier les "agents" et les structures de données en jeu). *(1,5 points)*

C.6 En vous appuyant sur le code fourni en annexe, proposer deux solutions `FileABQ` et `FileALS` au problème producteurs/consommateurs implémentant l'interface `FileDAttente`.

La première devra utiliser `ArrayBlockingQueue`, la seconde `ArrayList` et `Semaphore`. *(4 points)*

C.7 Connaissez d'autre(s) structure(s) de données utile(s) pour implémenter une solution à ce problème ? Les comparer brièvement (y compris avec celles de l'exercice précédent). *(1 points)*

D Election et RMI

D.8 Rappeler la spécification du problème de l'Election. *(0,5 point)*

D.9 Implémenter en RMI, l'algorithme de LeLann, Chang et Roberts. Rappeler préalablement dans quelle(s) condition(s) celui-ci fonctionne. *(4 points)*

On respectera les consignes suivantes pour représenter l'anneau de taille n :

- $n < 255$ (sans que cette valeur soit utilisable par l'algorithme)
- les machines ont pour adresses $192.168.20.1$ $192.168.20.254$
- la machine de n°i ($1 \leq i \leq 254$) a pour adresse $192.168.20.i$
- le voisin de gauche (resp. de droite) de la machine d'adresse `ip` est la machine d'adresse `ip-1` (resp. `ip+1`).
- on communiquera via deux objets RMI nommés `"gauche"` et `"droit"` qui représenteront les canals de communication en mode FIFO. Ces objets possédent deux méthodes `envoyer(Object msg)` et `Object recevoir()`.
- Seules les machines correspondantes peuvent utiliser les méthodes précédentes. Par exemple, une machine peut `envoyer` sur le `gauche` de la machine située à sa droite. Celle-ci est la seule à pouvoir faire `recevoir()`.
  Il ne vous est pas demandé d'implémenter syntaxiquement ces interdictions.

On ne gèrera pas les aspects déploiements et on supposera que les identités sont correctement fournies, lors du déploiement, en ligne de commande :

192.168.20.i $ `java LCR id`