UPMC
SORBONNE UNIVERSITÉS

_____ : Systèmes Répartis Client/Serveur

Examen du 15 Mars 2011

durée 2h - **Tous les documents papiers sont autorisés**
Téléphones portables, baladeurs et autres appareils électroniques doivent être éteints
Attention l'énoncé est imprimé recto-verso sur 2 feuilles
Le barème est donné à titre indicatif.

Julien Sopena

Exercice 1 : Construction d'un serveur de fichier avec RPC.

On souhaite construire un serveur de fichier en RPC. Le serveur de fichier doit offrir deux fonctions :

1. une fonction de lecture qui lit le contenu d'un fichier (tableau d'octets) symbolisé par une chaîne de caractères. Cette fonction renvoie 0 en cas d'échec.
2. une fonction d'écriture qui écrit le contenu (tableau d'octets) d'un fichier symbolisé par une chaîne de caractères. Cette fonction renvoie 0 en cas d'échec.

Question 1 – 2points

Est-ce que votre serveur est obligé de posséder un état associé aux clients ? Quels sont les avantages et inconvénients à avoir un serveur avec ou sans état ?

Question 2 – 2 points

Donnez, en langage RPC, l'interface du serveur.

Question 3 – 2 points

Donnez, en C, le code de la fonction d'écriture. On vous demande de gérer les erreurs et d'utiliser les fonctions de lecture/écriture du système d'exploitation. On considère que tous les fichiers font 42 octets.

Question 4 – 2 points

On souhaite maintenant améliorer notre serveur. Créez une nouvelle version de l'interface avec une méthode supplémentaire. Cette fonction de statistique doit rendre la taille (entier) et l'identifiant du propriétaire du fichier (entier) dont le nom est passé en paramètre.

Question 5 – 2 points

On souhaite faire une nouvelle amélioration à notre serveur. L'implémentation actuelle de la fonction d'écriture est actuellement bloquante tant que le serveur n'a pas terminé son écriture. Proposez une technique (sans en donnez l'implémentation) pour rendre cette écriture non bloquante.

Exercice 2 : Questions de cours Java

Question 1 – 1 point

Combien y a t-il de niveaux d'accès en java ? Listez-les dans une hiérarchie du plus fermé au plus ouvert (sans en donner de définition).

Question 2 – 1 point

On considère le code suivant ou le type `xxx` est soit un `boolean` et `Boolean`. Préciser dans les deux cas si le code compile et/ou s'exécute sans erreur.

```
public void uneSynchro (xxx b) {
  synchronized (b) {
    b = true ;
    b.notify() ;
  }
}
```

Question 3 – 1 point

En java où sont alloués les objets ? Quel problème d'efficacité cela pose-t-il en comparaison du C++ ?

Question 4 – 1 point

Pour quelle(s) raison(s) peut-on vouloir appeler la méthode `run` d'une classe implémentant l'interface `Runnable` ?

Exercice 3 : Programmation Publish/Subscribe

Dans cet exercice on veut réaliser un service de Publish/Subscribe (Publication/Abonnement) qui permet à des "subscribers" de s'enregistrer auprès de "publishers" dont ils souhaitent recevoir les publications.

S'il est largement répandu aujourd'hui, ce paradigme pose néanmoins un problème de surcharge au niveau du "publisher". On se propose donc de diffuser, de façon compressée, les publications dans un arbre de "subscribers" dont le fonctionnement suit le principe suivant :

1. Connexion sur une socket TCP d'un publisher (ou d'un autre subscriber) dont le port est donné en paramètre au lancement du programme.
2. Ouverture d'une socket de connexion TCP et affichage de son port sur le terminal.
3. Il se met alors à attendre :
   - d'une part des nouvelles connexions de "subscriber" sur sa socket de connexion.
   - d'autre part des nouvelles diffusions émises par le "publisher" (ou par le "subscriber" sur lequel il s'est lui-même connecté). La valeur reçue est alors décompressée et affichée sur le terminal puis rediffusée (avec compression) à l'ensemble des "subscribers" qui se sont connectés à lui.

Question 1 – 7 points

Donnez, dans son intégralité, le code d'une classe implémentant un "subscriber" qui suit le comportement défini ci-dessus.

Pour simplifier on considère que :

- tous les processus sont en local. Vous pouvez donc utiliser "localhost" comme adresse.
- il n'y a qu'un publisher et que son code est déjà connu.
- les publications sont des entiers *int*.
- vous n'avez pas à gérer les exceptions.
- vous n'avez pas à gérer les déconnexions des subscribers.
- vous pouvez utiliser la classe `List<Socket>` qui a trois méthodes :
  - *add(Socket s)* qui ajoute *s* dans la liste.
  - *size()* qui retourne le nombre d'éléments présents dans la liste.
  - *get(int id)* qui retourne l'élément d'index *id* (numérotation à partir de zéro).

Dans cet exercice vous ferez particulièrement attention :

- aux sockets réseaux.
- à la création et à la gestion des threads.
- à l'utilisation des flux et de la compression.
- à la synchronisation d'accès aux ressources critiques.

Lors de la correction de cet exercice il sera tenu compte de la simplicité et de la lisibilité de la réponse.