Travaux Dirigés Programmation Système: Feuille 9

Informatique 2ème année. ENSEIRB 2018/2019

—*Mathieu Faverge* - `mfaverge@enseirb.fr` —

RPC

Les appels à distance de fonctions (Remote Procedure Call) est un mécanisme qui permet d'appeler un service (une fonction) proposé par un serveur. Le nom de la fonction, ses paramètres sont envoyés au serveur qui execute la fonction puis retourne la valeur de retour.

▸**Exercice 1.** *Serveur/Client RPC*

1. *Implémenter les deux fonctions suivantes :*
   - `char *lirefichier(char *t)` *qui prend un nom de fichier en entrée et retourne le tableau d'octets t de ce fichier. Le tableau d'octets retourné sera alloué par malloc dans la fonction. La fonction retourne NULL en cas d'erreur.*
   - `int ecrirefichier(char *f, int n, char *t)` *qui prend un nom de fichier en paramètre f ainsi qu'un tableau de caractères t de taille n. Cette fonction retourne 0 s'il n'y a pas d'erreur, 1 sinon..*
2. *Proposer un protocole réseau pour pouvoir envoyer à un serveur le nom de l'une de ces fonctions, avec ses paramètres, et recevoir en retour la valeur de retour de la fonction.*
3. *Ecrire un programme serveur qui attend les connexions réseau et en fonction de ce qu'il reçoit, appelle l'une ou l'autre des fonctions et renvoie sa valeur de retour.*
4. *Ecrire un programme client pour pouvoir appeler ces fonctions sur le serveur. On definiera les fonctions* `char *lirefichier(char *t)` *et* `int ecrirefichier(char *f, int n, char *t)` *pour qu'elles fassent l'appel distant sur le serveur. Ces fonctions attendront la valeur de retour fournie par le serveur. Tout le code de communication devra être dans ces fonctions, le fait que ces fonctions soient appelées par le réseau ne doit pas apparaitre dans le reste du programme.*
5. *On souhaite modifier les fonctions du coté du client afin que les fonctions soient asynchrones : le programme ne devra pas rester bloqué en attendant que les fonctions s'exécutent. Pour cela, on modifie le type des fonctions en* `void lirefichier(char *t, void (*callback)(char *))` *et* `void ecrirefichier(char *f, int n, char *t, void (*callback)(int))` *: les deux fonctions doivent rendre la main immédiatement et quand la valeur de retour est prête, la fonction* `callback` *est appelée. Modifier le programme client pour implémenter cette fonctionnalité.*

Serveur distribué

Un serveur unique, centralisé a l'inconvénient de ne pas pouvoir gérer de très nombreuses connections car il risque d'être rapidement surchargé. Il est possible de réaliser de multiples serveurs, sur différentes machines, pour l'implémentation d'un service commun.

▸**Exercice 2.** *Table de hachage distribuée*

Le service étudié dans cet exercice est un service d'annuaire : pour un nom de personne, on souhaite obtenir son email. On réalisera cette correspondance par une table de hachage, calculant à partir de la clé (le nom de la personne) un numéro d'index dans un tableau de liste puis cherchant dans la liste le nom de la personne et son email.

Pour distribuer ce service entre plusieurs serveurs, le principe est que chaque serveur ne conserve qu'une partie de l'annuaire global :

- *chaque serveur ne gère les informations d'une partie de la table de hachage, c'est à dire, que certains couples de valeur nom,email.*

— *chaque serveur se connecte au lancement à un autre serveur, et le principe est qu'il interroge ce serveur lorsqu'il ne trouve pas dans son annuaire une entrée.*

1. *Implémenter une table de hachage permettant d'inserer un nom et une adresse email et de chercher l'adresse en donnant le nom. On prendra un tableau à* 1000 *entrées.*
2. *Ecrire un programme d'annuaire interactif :*
   — *quand l'utilisateur tape* `a nom email`*, le nom et l'email sont ajoutés à la table de symboles,*
   — *quand l'utilisateur tape* `r nom`*, l'email correspondant au nom est affiché (ou un message indiquant qu'il n'est pas trouvé).*
3. *Modifier ce programme pour qu'il propose le même service par le réseau, simultanément à l'interaction sur la console. On le fera sur le port* 3000*. Une fois lancé, on peut soit ajouter des entrées ou interroger l'annuaire à partir de la console, soit le faire à distance en envoyant les mêmes chaines de caractère (par telnet par exemple). Tester en utilisant telnet. Attention, il y a une section critique a mettre en exclusion mutuelle...*
4. *Modifier ce programme lui donner, de façon facultative, au lancement l'adresse d'une autre machine. Lorsqu'une entrée de la table n'est pas trouvée, il interroge alors le serveur d'annuaire distant. Tester ce fonctionnement.*
5. *Tester ce fonctionnement de serveur distribué dans la salle de TD, entre plusieurs machines. Est-ce que cela fonctionne et tout l'annuaire est disponible ? Dans la suite, on va faire en sorte que chaque serveur connaisse les autres.*
6. *Ajouter deux commandes que peut taper l'utilisateur ou qui peuvent être reçue par le réseau :*
   — `s adresse index` *qui indique que le serveur à l'adresse* `adresse` *a des entrées pour l'index* `index`*.*
   — `l` *qui donne la liste des serveurs actifs connus par le programme et pour chacun, la liste des index de la table pour lesquels il a des entrées.*
7. *Faire en sorte que toutes les 5 secondes, le programme envoie au serveur auquel il est connecté la commande* `l` *demandant cette liste et envoie sa propre liste avec des commandes* `s`*.*
8. *Tester ce fonctionnement dans la salle de TD.*