Rapport projet Un server HTTP

Lavigne Alexandre - Vallade Vincent

04/01/2017

Introduction

Ce rapport décrit notre raisonnement et les solutions apportées pour répondre aux exercices donnés par le projet de construction d'un mini-serveur HTTP pour l'UE Programation Répartie.

Ce serveur suit le protocole HTTP 1.1. par ailleurs, il n'accepte que la requette de type GET (dont le détail sera décrit par la suite)

Le projet est élaboré en langage C et la constante : XOPEN_SOURCE est définit avec la valeur : 700

0.1 Structure du Server

Le serveur HTTP prend en argument à son lancement :

— un port d'écoute

— le nombre de clients que l'on peut accepter à la fois

— un nombre d'octets qu'un client peut demander au maximum par minute

Pour cette première partie nous avons essayé de concevoir un serveur HTTP en anticipant les questions 2 et 3.

Nous avons conçu la partie principale du serveur pour que sa seule tâche soit de recevoir les requêtes des clients sur le socket d'écoute et de lancer un thread tout de suite après.

Le serveur établit donc toute l'initialisation puis lance une boucle infinie durant laquelle il écoute pour une connection de type TCP sur le port donné.

Le serveur redéfinit le *handler* pour le signal SIGINT afin de fermer le socket, détruire le sémaphore, et toutes autres ressources allouées, ce qui nous permet de fermer le serveur proprement via un CTRL+C. Pour faciliter le transfert d'informations entre le serveur principal et le thread qui traite avec un client, nous avons créer une structure contenant les informations suivantes :

— le socket de communication avec le client

— le descripteur de fichier de journalisation

— la structure sockaddr_in du client

— le semaphore pour gérer l'accès en concurrence du fichier de journalisation

La suite du document décrit le code du thread contenut dans le fichier "requete.c"

Le thread lit les caractères un par un sur le socket de communication jusqu'a trouver un caractère '\n'. Ensuite il extrait le premier mot pour vérifier qu'il corespond bien à 'GET'. Il extrait le deuxième mot pour vérifier le chemin que le client demande.

Sous unix pour demander une ressource à partir du dossier courant il suffit que le nom de cette ressource ne commence pas par '/', nous vérifions alors sa présence et le supprimons du chemin s'il existe. Nous vérifions si la ressource existe et si le server à les droits en lecture dessus.

Ensuite le thread vérifie le type mime du fichier grâce à une fonction permetant de parser le fichier '/etc/mime.types' et de trouver si un mime-type existe, si oui alors il l'envoie au serveur sinon le serveur envoie 'text/plain', de cette

manière nous protégeons le client contre toute exécution de code alors que le type du fichier est indéterminé.

La commande pour rechercher un type mime est la suivante :

```
grep -w %s /etc/mime.types |
awk '{if ($2 != \"\") if ($2 ~ /%s/) print $1}'
```

Cette ligne de commande BASH imprime les lignes du fichier /etc/mim.types qui contiennent le mot placé '%s' puis awk fils les lignes dont la 2ème colonne est vide, puis si la deuxième colonne match l'expression régulière /extention/ où extention est l'extention cherchée alors on imprime sur la sortie standard la première colonne, qui correspond au type mime recherché.

Une fois toutes les vérifications effectuées le thread envoie sa réponse au serveur, donc les headers contenant le code de retour, puis le type mime du fichier. Ensuite une fonction dédiée ouvre, lit et envoie le fichier au client. Une vérification des erreurs est éffectuée durant l'exécution et renvoie s'il le faut un code de retour 404 ou 403 ou 500 (voir questions suivantes)

0.2 Journalisation