Université Bordeaux1
2009

Rapport Projet réseaux et programmation système

Antoine Lucas
Alexandre Mothe

Chargé de TD Programmation sytèeme: M. WACRENIER
Chargé de TD Réseaux : M. Thibault

Table des matières

Chapitre 1

LINA (LINA Is Not Apache)

1.1 Compilation et Exécution

Le projet se compile via la commande 'make', ce qui créé l'exécutable 'lina'.

Pour obtenir des informations sur le serveur, l'exécuter avec l'option 'help'

1.2 Fonctionnement

En mode Thread, un select() est utilisé pour savoir sur quel socket (on général un soket ipv4 et un soket ipv6) nous parvient une connexion, puis nous lançons un thread en mode détaché qui s'occupera de la lecture, du traitement et de l'envoi de la réponse. Suivant le protocole utilisé dans la requête (en 0.9 et 1.0 = fermeture par défaut) et la valeur du paramètre 'Connect' (si celui ci existe), nous gardons la connexion ouverte. Le Thread ne se finira donc qu'une fois que le client ou le serveur aura décidé de fermer la connexion.

En mode Select, tout les sockets (celles en attente de connexions, comme celles qui traitent les requêtes) son écouté par un select(). Ensuite le comportement est très similaire à celui du mode thread (dans le fichier 'requete.c', fonction requete() pour le mode Select et fonction requeteThread() pour le mode Thread), la seul grosse différence réside dans la fonction de lecture de la requête (dans 'requete.c' fonction readRequete()). En mode Select celle-ci retourne le code HTTP 100 (Continue) dés qu'elle a fini de lire tout ce qu'il se trouvait sur le soket, sauf dans le cas ou elle trouve la suite de caractères de fin de requête 'CRLFCRLF' où elle lance le traitement de la requête, alors qu'en mode Thread, la fonction de lecture boucle (et bloque) tant qu'elle n'a pas trouvé cette suite de caractères. Le mode Select n'est donc 'parallèle' que sur la réception des requêtes, l'envoi d'un gros fichier monopolise totalement le serveur.

Dans les deux cas le champs 'timeout' de la fonction select() est utiliser pour aller déconnecter les clients trop longtemps inactif (quand cette option du serveur est activé).

Chapitre 2

Travail réalisé

Nous avons implémenté toutes les fonctionnalités du sujet (mise a par la gestion des gros fichiers) :

- Interdiction de remonter au delà de la racine.
- Gestion des réponses d'erreurs.
- Parallélisme du traitement des clients[1].
- Génération d'un fichier d'index (en l'absence des fichier 'index.htm' et 'index.html') listant le contenu du dossier.
- Exécution des scripts CGI, et de touts autres fichiers exécutables du dossier 'cgi-bin' (utilisation du paramètre 'x' de 'other').
- Création d'un log à chaque requête.
- Utilisation de préprocesseur de CPP.
- Gestion de la compression 'gzip' et 'bzip2' quand celles-ci est demandée.
- Gestion des commandes GET, HEAD[2], POST.
- Utilisation des variables (GET et POST), et copie ce celles-ci dans la variable d'environnement 'QUERY_STRING'.
- Authentification pour les dossiers muni du fichier '.htpasswrd'.

Nous avons aussi ajouté quelque options personnelles :

- Gestion de l'IPv6.
- Gestion des droits de lecture des fichiers (utilisation du paramètre 'r' de 'other').
- Gestion du keep-alive du protocole HTTP 1.1
- Fichier de configuration (et des paramètres)

Nous avons aussi légèrement modifié le traitement de l'authentification, celle-ci n'est plus lancée uniquement dans le dossier 'auth', mais dans tout les dossiers possédant un fichier '.htpasswrd'. Dans la globalité nous nous somme repartis le travail de la façon suivante :

Antoine s'est chargé du serveur dans le sens large du terme, la gestion des thread + select, les module de configurations, ainsi que des fonctions d'envoi de la réponse. Il s'est aussi occupé du module 'buffer'.

Alexandre s'est occupé de décrypter la requête, puis de reconstruire la réponse. Cela comprend la gestion de l'authentification, des erreurs (existence, droit de lecture ou d'exécution...).

Nous avons essayer de rendre le serveur le plus souple possible (utilisation de buffer dynamique...) et de traiter un maximum d'erreurs possible. Un de nos objectif a été d'exploiter au maximum les connaissances acquises au cour du semestre en programmation Système (pipe, fork, 'signaux', stat...) ainsi qu'en Réseaux (socket, IPv6...).

Pour éviter les fuites mémoires, nous avons tout de suite décider d'utiliser le module 'memoire' créé en semestre 3 et modifié pour l'occasion. Celui-ci nous liste chaque allocation, ré-allocation ou dés-allocation fait par le processus. En cas de fuite mémoire, celle-ci nous apparaissait donc d'elle même presque instantanément. Pour plus de sécurité nous avons complété ce module avec l'utilisation de 'valgrind'.

[1] Partiel pour le mode Select.
[2] Partiel pour la méthode HEAD.

Chapitre 3

Problèmes rencontrés et Améliorations possibles

La principale cause de problèmes dans notre groupe a été le manque d'organisation, nous nous sommes lancé dans la programmation aprés avoir très succinctement réparti les taches. Nous avons donc abouti à des problèmes de coordination au moment ou les fonctions de l'un voulaient utiliser les fonctions de l'autre. Mais ces problèmes on finalement tous été résolus.

Le protocole HTTP 0.9 n'est pas très bien géré, l'entête est parfois absente de la réponse, ceci est dut à notre peu de considération pour ce protocole.

Ajouter un module php a été en projet, mais l'est resté... dommage.

Un autre projet était de crée un fichier log d'erreurs.

Nous gérons quelques mimes (pdf, svg, jpg, html...), une idée était d'utiliser le fichier 'mimes.types' d'Apache pour automatiquement résoudre la valeur du champ 'Content-Types'.

Des buffer statiques subsistent ici et là alors qu'une utilisation d'un buffer dynamique pourrait se justifier, cela viens du fait que nous avons introduit le module 'buffer' en milieu de projet pour gagné en souplesse, et nous n'avons pas eu le temps de corriger le code partout.

La gestion du https a aussi trotté dans nos têtes, mais encore une fois elle est resté a l'état d'un vague projet.

la gestion des code de redirection HTTP.

Chapitre 4

Conclusion

Nous avons tout de suite eu une très bonne impression sur ce projet, celle-ci n'a fait que se renforcer au cour du temps. Mais le temps nous a manqué pour implémenter toutes nos idées (arrg !).

Chapitre 5

Sources

- Source de base du projet : TP5 et 6 de reseaux
- Squelette du raport : Frédérique Noret