Serveur http évolué

# Introduction

## 1). Objectifs

L’objectif de ce projet était de mettre en place un serveur http codé en C++. Un serveur http ou serveur web est un logiciel qui une fois héberger permet le transfert de fichier de http, le http étant un protocole de communication client-serveur.

Lors du premier semestre, dans le cadre du cours de réseaux, nous avions déjà été amené à étudier des serveur web simple permettant le traitement de requêtes simples. Cependant pour ce projet il s’agira de concevoir et d’améliorer ce type de serveur afin de lui inclure de nouvelles fonctionnalités

## 2). Cahier des charges

Ce serveur http devra pouvoir permettre à tout client de lire une page HTML stocké sur le serveur.

Il devra aussi comporter les fonctionnalités suivantes :

-Traitements des requêtes du clients (GET ou POST).

-Traitements des erreurs.

-Génération d’une page affichant les statistiques du serveur.

-Ouverture d’une fenêtre (QWidget) affichant ces mêmes statistiques.

-Gestion du cache en mémoire RAM.

-Accès administrateur à une page Web permettant l’activation/désactivation du serveur ainsi que la gestion du cache.

La conception de ce serveur web et de ses fonctionnalités seront réalisés en langage C++ et en utilisant les classes proposées par la bibliothèque objets QT.

II/. Fonctionnement du serveur

1). Structures de base (Traitements des requêtes)

Lors de la récupération du code source d’initiale du serveur les fonctionnalités de base du serveur étaient déjà implémentées. En effet lorsque nous avons commencé le projet le serveur était déjà capable de traiter une requête GET d’un client afin de renvoyer une page HTML. Cependant pour que cette fonctionnalité soit réellement fonctionnelle nous avons dû rajouter l’envoi de l’entête HTML « HTTP/1.1 200 ».

Le traitement des requêtes est réalisé par la classe MySocketClient qui gère toutes les interactions avec les différents clients internet :

-On récupère le lien de communication avec le client ;

-On récupère la requête et sa taille avec la méthode readLine() ;

-On traite ensuite la chaine de caractère reçu afin de ne récupérer la commande de la requête, le nom du fichier ou du dossier demandé. On peut ensuite déterminer le chemin du fichier à partir du dossier « public\_html » ;

-Si le fichier existe on envoi la page HTML au client qui peut donc y accéder depuis son navigateur.

Ce code de base ne permet que de traiter les demandes GET de pages HTML par exemple il est impossible de traiter la demande d’un dossier ou téléchargement.

2). Gestion des dossiers

Grâce aux types QFile et QDir proposés par la bibliothèque Qt il est possible déterminer si le client demande un fichier ou un dossier.

Pour traiter la requête du client dans le cas d’un dossier nous avons créés un fonction directory() :

Tout d’abord la classe MySocketClient ouvre le dossier puis créée une QFileInfoList contenant le nom et le chemin des fichiers contenus dans le dossier. La fonction directory() prend ensuite en argument cette liste de fichier puis génère une page HTML. Cette page HTML affiche des liens cliquables renvoyons les pages HTML contenu dans le dossier. Pour générer la page chaque ligne du code HTML est écrite dans le fichier à partir de la fonction directory(), pour cela on a utilisé la classe QTextStream proposé par QT.

3). Page HTML de statistiques

Une des fonctionnalités rajoutées au serveur est l’utilisation de statistiques d’utilisation. Le client et les administrateurs doivent pouvoir avoir accès aux différentes statistiques d’utilisation du serveur :

-Le nombre de requêtes reçues par le serveur ;

-Le nombre de requêtes traitées ;

-Le nombre de clients connectés ;

-Le nombre d’erreurs détectées ;

-Le nombre d’octets reçus ;

-Le nombre d’octets envoyés ;

Toutes ces statistiques sont gérées par la classe server\_stat qui contient en attribut toutes les valeurs de statistiques. Tout ces attributs sont déclarés en static afin qu’ils soient mis à jour pour tous les clients. C’est la méthode updateStat() qui permet de mettre à jour et d’incrémenter ces attributs, cette fonction prend en argument un type énumération qui permet de choisir quel attribut de statistique incrémenter (NEWCLIENT, NEWREQUEST…). Il suffit d’appeler cette fonction au moment où une des statistiques varie pour la mettre à jour, par exemple au moment MySocketClient lance une connexion client (méthode run()) on fait appelle à la fonction updateStat() avec l’argument NEWCLIENT pour incrémenter le nombre de clients connectés.

Une fois les statistiques mises à jour, comme pour la gestion des dossier, la méthode statHTML() de la classe server\_stat va générer un fichier HTML qui va afficher les différentes statistiques du serveur.

Pour cette classe de statistiques un « log » a aussi été implémenté afin de garder une trace des différentes requêtes reçues par le serveur. Ce « log » est affiché sur la page HTML à l’aide d’une fenêtre déroulante afin d’éviter d’encombrer la page.

//Expliquer fonction addtyperequest

4). Gestion de la page administrateurs

Sur le serveur se trouve une page HTML « configuration » qui n’est accessible qu’aux administrateurs du serveurs possédant le bon identifiant ainsi que le bon mot de passe. Cette page de configuration doit permettre à un administrateur de pouvoir activer ou désactiver le serveur et de vider le cache. Lorsqu’un client tente de se connecter à la page de configuration il est automatiquement redirigé vers une page d’identification où il devra rentrer un identifiant et un mot de passe. Lorsque le client clique sur le bouton de validation un formulaire HTML contenant l’identifiant et le mot de passe est envoyé au serveur par une requête POST. La vérification des identifiants est géré par la classe admin qui validera ou non la connexion du client.