Webserv

C'est à ce moment-là que vous comprenez enfin pourquoi les URL commencent par HTTP

Résumé :

Ce projet consiste à écrire votre propre serveur HTTP.
Vous pourrez le tester avec un vrai navigateur.

HTTP est l'un des protocoles les plus utilisés sur Internet.

Comprendre ses subtilités sera utile, même si vous ne travaillez pas sur un site Web.

Version : 22.0

Contenu

Chapitre I

Introduction

Le protocole de transfert hypertexte (HTTP) est un protocole d'application pour les systèmes d'information hypermédia distribués et collaboratifs.

HTTP est le fondement de la communication de données sur le Web, où les documents hypertextes incluent des hyperliens vers d'autres ressources auxquelles l'utilisateur peut facilement accéder, par exemple en cliquant sur un bouton de la souris ou en appuyant sur l'écran d'un navigateur web.

HTTP a été développé pour prendre en charge la fonctionnalité hypertexte et la croissance du World Wide Web.

La fonction principale d'un serveur Web est de stocker, de traiter et de fournir des pages Web aux clients.
La communication client-serveur s'effectue via le protocole de transfert hypertexte (HTTP).

Les pages livrées sont le plus souvent des documents HTML, qui peuvent inclure des images, des feuilles de style et des scripts en plus du contenu textuel.

Plusieurs serveurs Web peuvent être utilisés pour un site Web à fort trafic.

Un agent utilisateur, généralement un navigateur web ou un robot d'indexation, initie la communication en demandant une ressource spécifique via HTTP. Le serveur répond avec le contenu de cette ressource ou un message d'erreur s'il ne peut pas le faire. La ressource est généralement un fichier réel sur le stockage secondaire du serveur, mais ce n'est pas toujours le cas et dépend de la manière dont le serveur web est implémenté.

Bien que sa fonction première soit de diffuser du contenu, HTTP permet également aux clients d'envoyer des données. Cette fonctionnalité est utilisée pour soumettre des formulaires web, y compris le téléchargement de fichiers.

Chapitre II

Règles générales

- Votre programme ne doit en aucun cas planter (même s'il manque de mémoire).
  ory) ou se terminer de manière inattendue.
  Si cela se produit, votre projet sera considéré comme non fonctionnel et votre note sera de 0.

- Vous devez soumettre un Makefile qui compile vos fichiers sources. Il ne doit pas exécuter
  reconnexion inutile.
- Votre Makefile doit au moins contenir les règles :
  $(NAME), all, clean, fclean et re.
- Compilez votre code avec c++ et les indicateurs -Wall -Wextra -Werror
- Votre code doit être conforme à la norme C++ 98 et doit toujours être compilé lorsque
  ajout du drapeau -std=c++98.
- Assurez-vous d'exploiter autant de fonctionnalités C++ que possible (par exemple, privilégiez <cstring>
  à <string.h>). Vous pouvez utiliser des fonctions C, mais privilégiez toujours leurs versions C++ si
  possible.
- Toute bibliothèque externe et les bibliothèques Boost sont interdites.

Chapitre III

Partie obligatoire

Nom du programme	serveur Web
Remettre les fichiers	Makefile, *.{h, hpp}, *.cpp, *.tpp, *.ipp, fichiers de configuration
Makefile	NOM, tous, nettoyer, fclean, re
Arguments	[Un fichier de configuration]
Fonctions externes.	Toutes les fonctionnalités doivent être implémentées en C++ 98. execve, pipe, strerror, gai_strerror, errno, dup, dup2, fork, socketpair, htons, htonl, ntohs, ntohl, sélectionner, interroger, epoll (epoll_create, epoll_ctl, epoll_wait), kqueue (kqueue, kevent), socket, accepter, écouter, envoyer, recevoir, chdir, lier, connecter, getaddrinfo, freeaddrinfo, setsockopt, getsockname, getprotobyname, fcntl, fermer, lire, écrire, waitpid, tuer, signaler, accéder, stat, ouvrir, opendir, readdir et fermé.
Libft autorisé	n / A
Description	Un serveur HTTP en C++ 98

Vous devez écrire un serveur HTTP en C++ 98.

Votre exécutable doit être exécuté comme suit :

./webserv [fichier de configuration]

Même si poll() est mentionné dans le sujet et les critères de notation,
vous pouvez utiliser n'importe quelle fonction équivalente telle que select(), kqueue() ou
sondage électronique().

Veuillez lire la RFC et effectuer des tests avec telnet et NGINX avant de démarrer ce projet.
Bien que vous ne soyez pas obligé d'implémenter l'intégralité de la RFC, sa lecture vous aidera à développer les fonctionnalités requises.

III.1 Exigences

- Votre programme doit prendre un fichier de configuration comme argument ou utiliser un chemin par défaut.
- Vous ne pouvez pas exécuter un autre serveur Web.
- Votre serveur doit rester non bloquant à tout moment et gérer correctement les interruptions des clients. connexions si nécessaire.
- Il doit être non bloquant et utiliser un seul poll() (ou équivalent) pour toutes les opérations d'E/S entre le client et le serveur (écoute incluse).
- poll() (ou équivalent) doit surveiller simultanément la lecture et l'écriture.
- Vous ne devez jamais effectuer une opération de lecture ou d'écriture sans passer par poll() (ou équivalent).
- La vérification de la valeur de errno est strictement interdite après avoir effectué une lecture ou une écriture opération.
- Vous n'êtes pas obligé d'utiliser poll() (ou équivalent) avant de lire votre configuration déposer.

Étant donné que vous devez utiliser des descripteurs de fichiers non bloquants, il est possible d'utiliser des fonctions de lecture/réception ou d'écriture/envoi sans poll() (ou équivalent), et votre serveur ne serait pas bloquant.
Mais cela consommerait plus de ressources système.
Ainsi, si vous tentez de lire/recevoir ou d'écrire/envoyer sur un descripteur de fichier sans utiliser poll() (ou équivalent), votre note sera être 0.

- Vous pouvez utiliser toutes les macros et définir comme FD_SET, FD_CLR, FD_ISSET et FD_ZERO (comprendre ce qu'ils font et comment ils fonctionnent est très utile).
- Une requête adressée à votre serveur ne doit jamais rester bloquée indéfiniment.
- Votre serveur doit être compatible avec les navigateurs Web standards de votre choix.
- Nous considérerons que NGINX est conforme à HTTP 1.1 et peut être utilisé pour comparer en-têtes et comportements de réponse.
- Vos codes d'état de réponse HTTP doivent être exacts.
- Votre serveur doit avoir des pages d'erreur par défaut si aucune n'est fournie.
- Vous ne pouvez pas utiliser fork pour autre chose que CGI (comme PHP, ou Python, etc.).
- Vous devez être en mesure de proposer un site Web entièrement statique.
- Les clients doivent pouvoir télécharger des fichiers.
- Vous avez besoin au moins des méthodes GET, POST et DELETE.

• Testez le stress de votre serveur pour vous assurer qu'il reste disponible à tout moment.

• Votre serveur doit pouvoir écouter plusieurs ports (voir Fichier de configuration).

Nous avons délibérément choisi de ne proposer qu'un sous-ensemble de la RFC HTTP. Dans ce contexte, la fonctionnalité d'hôte virtuel est considérée comme hors de portée.

III.2 Pour MacOS uniquement

Étant donné que macOS gère write() différemment des autres systèmes d'exploitation basés sur Unix, vous êtes autorisé à utiliser fcntl().
Vous devez utiliser des descripteurs de fichiers en mode non bloquant pour y parvenir comportement similaire à celui des autres systèmes d'exploitation Unix.

Cependant, vous êtes autorisé à utiliser fcntl() uniquement avec les indicateurs suivants :

F_SETFL, O_NONBLOCK et FD_CLOEXEC.
Tout autre drapeau est interdit.

III.3 Fichier de configuration

Vous pouvez vous inspirer de la section « serveur » du fichier de configuration NGINX.

Dans le fichier de configuration, vous devriez pouvoir :

- Choisissez le port et l'hôte de chaque « serveur ».
- Configurer les noms de serveur ou non.
- Le premier serveur pour un hôte:port sera le serveur par défaut pour cet hôte:port (ce qui signifie qu'il répondra à toutes les requêtes qui n'appartiennent pas à un autre serveur).
- Configurer les pages d'erreur par défaut.
- Définissez la taille maximale autorisée pour les corps de requêtes client.
- Configurez des itinéraires avec une ou plusieurs des règles/configurations suivantes (les itinéraires n'utiliseront pas d'expression régulière) :
  - Définissez une liste de méthodes HTTP acceptées pour l'itinéraire.
  - Définir une redirection HTTP.
  - Définissez un répertoire ou un fichier où le fichier demandé doit être situé (par exemple, si l'url /kapouet est enracinée dans /tmp/www, l'url /kapouet/pouic/toto/pouet est /tmp/www/pouic/toto/pouet).
  - Activer ou désactiver la liste des répertoires.

- Définissez un fichier par défaut à servir lorsque la demande concerne un répertoire.
- Exécuter CGI en fonction d'une certaine extension de fichier (par exemple .php).
- Faites-le fonctionner avec les méthodes POST et GET.
- Autoriser l'itinéraire à accepter les fichiers téléchargés et configurer leur emplacement sauvé.

Vous vous demandez ce qu'est un CGI est-ce que ? Parce que vous n'appellerez pas le CGI directement, utilisez le chemin complet comme PATH_INFO.

N'oubliez pas que pour les requêtes fragmentées, votre serveur doit les déstructurer ; le CGI attendra la fin du corps du message. Il en va de même pour la sortie du CGI. Si aucune longueur de contenu n'est renvoyée par le CGI, la fin des données renvoyées sera indiquée. Votre programme doit appeler le CGI avec le fichier demandé comme premier appel. argument.

Le CGI doit être exécuté dans le répertoire correct pour l'accès au fichier de chemin relatif. Votre serveur doit prendre en charge au moins un CGI (php-CGI, Python, etc.) en avant).

Vous devez fournir des fichiers de configuration et des fichiers par défaut pour tester et démontrer que toutes les fonctionnalités fonctionnent pendant l'évaluation.

Si vous avez une question sur un comportement spécifique, vous devez comparer le comportement de votre programme avec celui de NGINX.
Par exemple, vérifiez comment fonctionne le nom_serveur.
Nous avons fourni un petit testeur. Son utilisation n'est pas obligatoire si tout fonctionne bien avec votre navigateur et vos tests, mais cela peut vous aider à trouver et à corriger des bugs.

La résilience est essentielle. Votre serveur doit rester opérationnel en permanence.

Ne testez pas avec un seul programme. Écrivez vos tests dans un langage plus adapté, comme Python ou Golang, entre autres, voire en C ou C++ si vous préférez.

Chapitre IV

Partie bonus

Voici quelques fonctionnalités supplémentaires que vous pouvez implémenter :

- Prise en charge des cookies et de la gestion des sessions (fournir des exemples simples).
- Gérer plusieurs CGI.

La partie bonus ne sera évaluée que si la partie obligatoire est complétée intégralement et sans problème. Si vous ne remplissez pas toutes les conditions obligatoires, votre partie bonus ne sera pas évaluée.

Chapitre V

Soumission et évaluation par les pairs

Soumettez votre devoir dans votre dépôt Git comme d'habitude. Seul le contenu de vos dépôts sera évalué lors de la soutenance. Vérifiez bien les noms de vos fichiers pour vous assurer qu'ils sont corrects.

16D85ACC441674FBA2DF65190663F42A3832CEA21E024516795E1223BBA77916734D1
26120A16827E1B16612137E59ECD492E46EAB67D109B142D49054A7C281404901890F
619D682524F5