Webserv

C'est à ce moment que vous comprenez enfin pourquoi une URL commence par HTTP

Résumé :

Ce projet concerne l'écriture de votre propre serveur HTTP.
Vous pourrez le tester avec un navigateur réel.

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

Connaître ses arcanes sera utile, même si vous ne travaillez pas sur un site Web.

Version : 20.3

Contenu

Chapitre I

Introduction

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

HTTP est la base de la communication de données pour le World Wide Web, où les documents hypertextes incluent des hyperliens vers d'autres ressources auxquelles l'utilisateur peut facilement accéder.
Par exemple, par un clic de souris ou en touchant l'écran dans un navigateur Web.

HTTP a été développé pour faciliter l'hypertexte et le 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 entre le client et le serveur s'effectue à l'aide du protocole de transfert hypertexte (HTTP).

Les pages livrées sont le plus souvent des documents HTML, pouvant 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'exploration Web, initie la communication en demandant une ressource spécifique à l'aide de HTTP et le serveur répond avec le contenu de cette ressource ou un message d'erreur s'il est incapable de le faire. La ressource est généralement un fichier réel sur le stockage secondaire du serveur, mais ce n'est pas nécessairement le cas et dépend de la façon dont le serveur Web est implémenté.

Bien que la fonction principale soit de servir du contenu, la mise en œuvre complète de HTTP inclut également des moyens de recevoir du contenu des clients. Cette fonctionnalité est utilisée pour soumettre des formulaires Web, y compris le téléchargement de fichiers.

Chapitre II

Règles générales

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

- Vous devez rendre un Makefile qui compilera vos fichiers source. Il ne faut pas reconnecter.

- Votre Makefile doit au moins contenir les règles : $(NAME), tous, nettoyer, fclean et re.

- Compilez votre code avec c++ et les drapeaux -Wall -Wextra -Werror

- Votre code doit respecter la norme C++ 98. Ensuite, il devrait encore compiler si vous ajoutez le drapeau -std=c++98.

- Essayez de toujours développer en utilisant le plus de fonctionnalités C++ possible (par exemple, choisissez <cstring> plutôt que <string.h>). Vous êtes autorisé à utiliser les fonctions C, mais préférez toujours leurs versions C++ si possible.

- Toute librairie externe et librairies 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	NAME, all, clean, fclean, re [Un fichier de configuration]
Arguments	
Fonctions externes.	Tout en C++ 98. execve, dup, dup2, pipe, strerror, gai_strerror, errno, dup, dup2, fork, htons, htonl, ntohs, ntohl, select, poll, epoll (epoll_create, epoll_ctl, epoll_wait), kqueue (kqueue, kevent), socket, accept, listen, send, recv, bind, connect, getaddrinfo, freeaddrinfo, setsockopt, getsockname, getprotobyname, fcntl n/a Un serveur HTTP en C++ 98
Libft autorisé	
Description	

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

Votre exécutable sera exécuté comme suit :

./webserv [fichier de configuration]

Même si poll() est mentionné dans le sujet et l'échelle d'évaluation, vous pouvez utiliser n'importe quel équivalent tel que select(), kqueue(), ou epoll().

Veuillez lire le RFC et faire quelques tests avec telnet et NGINX avant démarrage de ce projet.
Même si vous n'êtes pas obligé d'implémenter toute la RFC, sa lecture vous aidera à développer les fonctionnalités requises.

III.1 Exigences

- Votre programme doit prendre un fichier de configuration en argument, ou utiliser un chemin par défaut.
- Vous ne pouvez pas exécuter un autre serveur Web.
- Votre serveur ne doit jamais bloquer et le client peut être rebondi correctement si nécessaire.
- Il doit être non bloquant et n'utiliser qu'1 poll () (ou équivalent) pour toutes les opérations d'E/S entre le client et le serveur (écoute incluse).
- poll() (ou équivalent) doit vérifier la lecture et l'écriture en même temps.
- Vous ne devez jamais faire une opération de lecture ou d'écriture sans passer par poll() (ou équivalent).
- La vérification de la valeur de errno est strictement interdite après une opération de lecture ou d'écriture.
- Vous n'avez pas besoin d'utiliser poll() (ou équivalent) avant de lire votre fichier de configuration.

Parce que vous devez utiliser des descripteurs de fichiers non bloquants, il est possible d'utiliser les fonctions read/recv ou write/send sans poll() (ou équivalent), et votre serveur ne bloquerait pas.
Mais cela consommerait plus de ressources système.
Ainsi, si vous essayez de lire/recevoir ou d'écrire/envoyer n'importe quel descripteur de fichier sans utiliser poll() (ou équivalent), votre note sera 0.

- Vous pouvez utiliser chaque macro et définir comme FD_SET, FD_CLR, FD_ISSET, FD_ZERO (comprendre quoi et comment ils le font est très utile).
- Une requête à votre serveur ne devrait jamais rester bloquée indéfiniment.
- Votre serveur doit être compatible avec le navigateur Web de votre choix.
- Nous considérerons que NGINX est conforme à HTTP 1.1 et peut être utilisé pour comparer les en-têtes et les 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 servir un site Web entièrement statique.
- Les clients doivent pouvoir télécharger des fichiers.
- Vous avez besoin d'au moins les méthodes GET, POST et DELETE.
- Stress teste votre serveur. Il doit rester disponible à tout prix.
- Votre serveur doit pouvoir écouter plusieurs ports (voir Fichier de configuration).

III.2 Pour MacOS uniquement

Étant donné que MacOS n'implémente pas write() de la même manière que les autres systèmes d'exploitation Unix, vous êtes autorisé à utiliser fcntl().

Vous devez utiliser les descripteurs de fichiers en mode non bloquant afin d'obtenir un comportement similaire à celui des autres systèmes d'exploitation Unix.

Cependant, vous êtes autorisé à utiliser fcntl() uniquement comme suit : fcntl(fd, F_SETFL, O_NONBLOCK);

Tout autre drapeau est interdit.

III.3 Fichier de configuration

Vous pouvez vous inspirer de la partie "serveur" du fichier de configuration NGINX.

Dans le fichier de configuration, vous devriez pouvoir :

- Choisissez le port et l'hôte de chaque 'serveur'.
- Configurez ou non les noms_serveurs.
- Le premier serveur pour un hôte:port sera le serveur par défaut pour cet hôte:port (c'est-à-dire qu'il répondra à toutes les requêtes qui n'appartiennent pas à un autre serveur).
- Configurer les pages d'erreur par défaut.
- Limitez la taille du corps du client.
- Configurer les routes avec une ou plusieurs des règles/configuration suivantes (les routes ne utiliser regexp):
  - Définissez une liste des méthodes HTTP acceptées pour la route.
  - Définissez une redirection HTTP.
  - Définissez un répertoire ou un fichier à partir duquel le fichier doit être recherché (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 pour répondre si la demande est un répertoire.
- Exécutez CGI en fonction d'une certaine extension de fichier (par exemple .php).
- Faites-le fonctionner avec les méthodes POST et GET.
- Rendez la route capable d'accepter les fichiers téléchargés et configurez où ils doivent être sauvé.

Vous demandez-vous ce qu'est un CGI est?

Comme vous n'appelez pas directement le CGI, utilisez le chemin complet comme PATH_INFO.

N'oubliez pas que, pour une requête fragmentée, votre serveur doit la décomposer, le CGI attendra EOF comme fin du corps. Même chose pour la sortie du CGI. Si aucun content_length n'est retourné du CGI, EOF marquera la fin des données renvoyées.

Votre programme doit appeler le CGI avec le fichier demandé comme premier argument.

Le CGI doit être exécuté dans le répertoire correct pour l'accès au fichier de chemin relatif. Votre serveur doit fonctionner avec un seul CGI (php-CGI, Python, etc.).

Vous devez fournir des fichiers de configuration et des fichiers de base par défaut pour tester et démontrer que chaque fonctionnalité fonctionne pendant l'évaluation.

Si vous avez une question sur un comportement, vous devez comparer le comportement de votre programme avec celui de NGINX.
Par exemple, vérifiez comment fonctionne nom_serveur.
Nous avons partagé avec vous un petit testeur. Il n'est pas obligatoire de le réussir si tout fonctionne bien avec votre navigateur et vos tests, mais cela peut vous aider à chasser quelques bogues.

L'important, c'est la résilience. Votre serveur ne devrait jamais mourir.

Ne testez pas avec un seul programme. Écrivez vos tests avec un langage plus pratique tel que Python ou Golang, etc. Même en C ou C++ si vous le souhaitez.

Chapitre IV

Partie bonus

Voici les fonctionnalités supplémentaires que vous pouvez ajouter :

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

La partie bonus ne sera évaluée que si la partie obligatoire est PARFAITE.
Parfait signifie que la partie obligatoire a été entièrement réalisée et fonctionne sans dysfonctionnement. Si vous n'avez pas satisfait à TOUTES les exigences obligatoires, votre partie bonus ne sera pas du tout évaluée.

Chapitre V

Soumission et évaluation par les pairs

Rendez votre devoir dans votre référentiel Git comme d'habitude. Seul le travail à l'intérieur de votre référentiel sera évalué lors de la soutenance. N'hésitez pas à revérifier les noms de vos fichiers pour vous assurer qu'ils sont corrects.