TP 9 Serveur WEB multi-processus en C

1 Problématique:

Le but de ce TP est de programmer un mini-serveur WEB. L'objet de ce TP n'est pas d'apprendre le protocole HTTP mais de réaliser un serveur capable de traiter plusieurs connections en parallèle grâce à des processus fils.

HTTP
On rappelle que le protocole HTTP 1.1 est décrit dans la RFC 2616.

2 Aide :

Outils
Les outils suivants seront utiles pour tester votre serveur:

- **netcat** et **telnet** permettent d'interagir avec le serveur;
- **netstat** permet d'afficher la liste des connections ouvertes sur le système; l'option **-t** permet de se limiter aux connections TCP et **-l** permet d'obtenir les sockets d'écoute;
- un navigateur WEB tel que **firefox** pour se connecter au serveur, on demandera simplement une URL de la forme **http://localhost:8888/index.html.**

3 Travail demandé :

1) Écrivez un serveur C qui écoute sur le port TCP **8888** (**socket**, **bind**, **listen**).
2) À chaque nouvelle connection d'un client (**accept**), le serveur crée un nouveau processus fils (**fork**) pour la traiter tandis que le processus père continue, en parallèle, à attendre d'autres connections.
   Une requête de la forme **GET /*chemin* HTTP/1.1** sera interprétée comme une demande de téléchargement du fichier se trouvant à la position *chemin* par rapport au répertoire où est lancé le serveur. Elle générera une réponse de la forme:

   ```
   HTTP/1.1 200 OK
   Content-length: nn

   contenu du fichier
   ```

   qui précise la taille *nn* et le contenu du fichier. Vous pouvez également adjoindre des en-têtes additionnelles telles que Date, Last-Modified ou Content-Type.

   les requêtes de la forme GET **/** HTTP/1.1 ou GET www/ HTTP/1.1 seront interprétées comme des demandes de téléchargement du fichier **index.html** se trouvant à la position / ou www/ par rapport au répertoire où est lancé le serveur.

   Si le fichier n'existe pas ou est inaccessible, le serveur donnera une réponse de la forme:

   ```
   HTTP/1.1 404 Not found
   Content-type: text/html
   ```

```
<html><head><title>Pas trouvé !</title><meta charset="UTF-8"></head<body><p>Désolé, le fichier demandé n'a pas été trouvé : <tt>/fichier.html</tt> </body></html>
```

Après avoir répondu à la requête , le processus fils ferme la connexion et quitte.

4 Journal d'accès :

Modifiez votre serveur pour qu'il garde en mémoire vive un journal (*log*) de chaque requête traitée: URL demandée, IP du client, date, statu d'erreur. Quand un client demande l'URL spéciale **/log**, le serveur envoie ce journal sous forme de page HTML.

Le journal étant partagé entre toutes les fils, tout accès doit être protégé par un verrou d'exclusion mutuelle.

La page **/log** étant dynamique, c'est une bonne idée d'interdire sa mise en cache par le client ou un éventuel proxy grâce à l'en-tête **Cache-control: no-cache.**