IFT6891 : Projet intégrateur
Automne 2001

Professeur:
Victor Ostromoukhov : ostrom@iro.Umontreal.CA
Démonstrateur:
Stéphane Gaudreault : gaudrs@IRO.Umontreal.CA

Date de remise: 21 Décembre 2001 avant 9h00

Ce projet se veut une synthèse de vos connaissances aquises dans les cours de structures de données et de systèmes d'exploitation. On vous demande d'écrire un petit serveur HTTP (en C) qui peut répondre à des requêtes venant d'un fureteur, comme Netscape, pour divers types de fichiers. En plus de cela, vous devrez écrire un petit fureteur (en java) qui permettra l'affichage de documents HTML.

Connaissances requises

1. utilisation des SOCKETS en C
2. protocole HTTP
3. communication entre processus parallèles
4. Pipes et redirection de STDOUT
5. Programmation orientée-objet en Java

1 Serveur

Le serveur accepte des commandes GET selon le protocole HTTP. En général, le serveur retourne au client le contenu du fichier indiqué dans le GET, mais le traitement exact depend de l'extention du fichier demandé. Vous devez pouvoir traiter les types de fichiers suivants (et retourner le "type" approprié dans l'entête de la réponse) :

```
.html ou .htm  ( retourner text/html).
.txt, .c, .java      ( ... text/plain)
.gif                 ( ... image/gif).
.jpg ou jpeg         ( ... image/jpeg).
.pdf                 ( ... application/pdf).
```

Si le fichier demandé est un répertoire, votre serveur doit se comporter comme un serveur WEB et afficher le fichier "index.html" (ou "index.htm") du répertoire. S'il n'existe pas de fichier pourtant un de ces deux noms, il faut alors retourner une liste des fichiers contenuent dans le répertoire (il faut retourner une page html avec une liste de liens vers les fichiers dans le repertoire). Note : vous pourrez utiliser la fonction **stat()** pour vérifier si un fichier est un répertoire.

Si le client demande un fichier d'un autre type que ceux énumérés ci-dessus, votre serveur doit refuser la requête en envoyant un message d'erreur approprié dans un fichier html.

Dans l'environnement du DIRO, chaque usager à la possibilité d'avoir un petit site web. Les fichiers de ce site web se trouvent dans le répertoire $HOME/HTML/. Notre serveur va donc utiliser ce répertoire pour y chercher ces fichiers. Le PATH donne dans le GET est donc consideré comme étant relatif au répertoire WWW du serveur du DIRO, c'est à dire que votre serveur trouve les fichiers en concatenant le Path du GET avec $HOME/HTML/Projet.

NOTE IMPORTANTE La fonction suivante devra être utilisée afin d'éviter qu'un client quelconque puisse lister tous vos fichiers privés. L'idée est de s'assurer que les appelés se trouvent bel et bien dans le répertoire HTML.

```
int fichierDangereux(char *strFile)
{
   int i, nLength;

   nLength = strlen(strFile) - 1;

   if (*strFile == '/') return 1;

   for(i = 0; i < nLength; i++)
      if (strFile[i] == '.' && strFile[i+1] == '.')
         return 1;
   return 0;
}
```

Cette fonction retournera TRUE si le path soumis contiend des "..". Dans ce cas, c'est qu'un client essait d'accéder à un fichier interdit. Vous devez alors retourner un message d'erreur dans un fichier HTML.

1.1 Detail du comportement

1. Pour chaque requête, le serveur fait démarrer un processus en parallèle.
2. À la fin d'un processus enfant, le parent doit faire un **wait** afin d'éviter d'avoir des **zombies**.

3. À chaque fois que votre serveur reçoit une requete ou envoit une réponse, vous devez afficher une trace à l'écran.

1.2 Démarrage

Au demarrage, le serveur devra imprimer l'adresse de son socket (host et port) au terminal. **IL NE FAUT PAS** fixer un numéro pour la socket dans votre programme, faites plutôt la création avec port = 0, se qui allouera un port libre.

1.3 Terminaison

Pour terminer le serveur, celui qui l'a lance entre la commande STOP au clavier (en majuscule ou en minuscule). Faites attention, cela n'est pas une requete HTTP. Ceci signifie que le serveur doit surveiller (avec SELECT) à la fois et STDIN et sa socket.[1]

À la fin de son exécution, le processus parent :

1. attend la fin du traitement des requêtes en cours,
2. ferme son socket,
3. ferme tous les fichiers ouverts (s'il y en a)
4. termine.

Un CTRL-C devra être traité comme un stop (clean shutdown). Il faudra donc intercepter le signal SIGKILL et y réagir comme décrit ci-dessus.

2 Client

Vous devez aussi implanter un client (un mini fureteur internet) qui permettra de visualiser des pages html (par exemple celles qui sont envoyées par votre serveur HTTP). Ce client doit être une application Java (pas d'applet) avec une interface graphique. La fenêtre pourrait être de dimension fixe (vous fixez une taille d'avance) ou encore permettre à l'usager de modifier la taille de la fenêtre (voir la section "améliorations possibles").
Pour faire votre interface graphique, nous vous recommendons fortement d'utiliser les classes **swing** de l'API de Java. En particulier, vous devez faire le rendu de vos fichiers HTML soit dans un objet JTextPane, soit dans un objet de type java.awt.canvas (voir la section "améliorations possibles"). Vous pouvez faire un choix entre ces deux mode d'affichage uniquement.
Pour simplifier les choses, vous devez reconnaître uniquement un sous-ensemble strict du langage HTML (autrement dit, les documents qui suivent cette syntaxe pourraient être visualisés sans problèmes avec Netscape).

[1] Ne pas confondre avec la requête STOP du dernier devoir.

1. Le document commence par <HTML> et se termine par </HTML>. Si ce n'est pas le cas, on l'affiche en entier comme s'il s'agissait d'un simple fichier texte.
2. Ensuite, il doit y avoir la sequence de balise suivantes

   ```
   <head>
   <title>Un titre</title>
   </head>
   ```

   Le titre "Un titre" est le titre de la page et il devra etre affiche dans la barre de titre (en haut de la fenêtre) de votre navigateur.
3. Le texte du document est contenu entre <body> et </body>. Ces balise doivent absolument se trouver après le </head>.
4. Dans le corps du document (i.e entre <body> et </body>), on peut trouver les balises suivantes qui servent à formater le texte :
   (a) <P> et </P> servent a encadrer un paragraphe. Il faudra donc afficher le texte compris entre ces balises en laissant une ligne blanche au debut. L'attribut "align", qui est optionnel sert à indiquer l'alignement du texte. Les valeurs qui sont permises sont "center" et "left". Par exemple, si on a

   ```
   <P align=''center''>texte1</P>
   ```

   Cela signifie que le texte "texte1" doit être centré dans l'écran. Notez que l'attribut se trouve dans la balise de départ et pas dans la balise de fin. Par defaut, si aucun attribut n'est spécifié, l'alignement se fait a gauche.
   (b) <B> et </B> servent à encadrer un texte en caractères gras.
   (c) <I> et </I> servent à encadrer un texte en caractères italiques.
   (d) <H1> et </H1> servent à encadrer le titre du texte. Celui-ci sera écrit en gros caractères, suivi d'un retour de ligne.
   (e) <H2> et </H2> même chose que <H1>, sauf que le texte est un peu moins gros.
   (f) <A href="URL"> et sert à encadrer un lien vers le document dont l'URL est donné en paramètre. Si on clique sur ce lien avec la sourie, cela devrait nous permettre d'afficher ce document à la place de la page actuelle. Le document qui est appelé par le lien doit remplacer celui qui était affiché au départ dans votre fureteur.
   Si le lien fait référence à un document PDF, il faut l'ouvrir dans le logiciel Acrobat Reader (la commande, sous Linux, est acroread). De même, les fichiers de types texte (.txt, .c et .java) devront être ouvert avec Emacs.
   (g) <IMG width="nombre_entier" height="nombre_entier" src="URL"> (sans balise de fin) sert à insérer une image, dont l'URL est donnée

en paramêtre, dans votre document. Ces images devront être en format gif ou jpeg. Les attributs *width* et *height* servent à indiquer la largeur et la hauteur de l'image[2]

5. De façon générale, les retours de chariot (fin de lignes) et les tabulations qui se trouvent dans le code HTML d'un document devront être affichés comme un simple espace.
6. Les balises peuvent être imbriquées. Par exemple <B><I>Un texte</B></I>, donne un texte en italique et en gras. L'ordre n'a pas d'importance. Le résultat aurait été le même si on avait fait <I><B>Un texte</I></B> ou encore <I><B>Un texte</B></I>.
7. Si votre fureteur rencontre un balise inconnue (elle ne fait pas partie de la liste ci-dessus), il faut l'ignorer. **Votre fureteur ne doit pas reconnaître d'autres balises que celles énumérées plus haut.**

Dans votre interface graphique, il doit y avoir au moins les éléments suivants :

1. Une zone de saisie pour tapper l'URL du document
2. Un bouton "recharger" qui permet de recharger le document (il faut faire une nouvelle requête au serveur).

Si la page web est trop longue, il faut un "scrollbar" pour pouvoir la visionner au complet. Ce "scrollbar" doit être affiché seulement si la page est trop longue pour etre contenu entièrement dans la fenêtre du fureteur.

3 Compilation et documentation

Vous devez remettre une Makefile pour le client et un autre pour le serveur. Vous devez faire de même pour la man page. Les nom des fichiers des man page seront serveur.1.gz et fureteur.1.gz (n'oubliez pas de compresser votre fichier avec gzip).

Les règles de vos Makefile doivent être : all, debug et clean. Les programmes C doivent être compilés avec les options **-Wall -ansi (et -g s'il y a lieu)**. Pour Java, on utilise l'option -g dans la règle "debug". Vous devez utiliser les classes de l'API standard de Java. Vous pouvez utiliser d'autre option en plus de celles que nous vennons d'énumérer.

4 Améliorations possibles

Il n'y a pas de "bonus" possibles dans ce devoir. Par contre, ceux qui veulent relever un défis supplémentaire pourraient :

1. Permettre à l'usager de redimentionner la fenêtre. Il faut alors repositionner les différents éléments graphiques à chaque fois.

[2] Notez que les valeurs des attributs sont toujours placés entre guillemet (""). Par exemple, pour faire un lien sur une image : <A href="index.html"><IMG src="image.gif" width="10" height="15"></A>

2. Lorsqu'un usager clique sur un lien avec le bouton de droite de la sourie, la page s'ouvre dans une nouvelle fenêtre du fureteur. Celle-ci doit être identique à la première. Pour faire cela, il faudra porter un attention particulière à l'organisation de votre code.
3. Plutôt que d'utiliser un JTextPane pour faire l'affichage du HTML, vous pouvez utiliser un Canvas (AWT) et y dessiner les caractères. Cela vous demandera beaucoup plus de temps.

Les éléments qui sont suggérés ici sont **facultatifs**. Ils ne peuvent **en aucun cas** faire augmenter la note d'une équipe. Par contre, il peuvent vous pénalisé s'ils introduisent des bugs dans votre programme.

5 Références

Spécification complète du HTTP : http ://www.w3.org/Protocols/HTTP/1.1/rfc2616.pdf
Code d'erreur en HTTP : http ://www.w3.org/Protocols/HTTP/HTRESP.html
Le langage HTML : http ://www.eisti.fr/doc/norm/html4/contents.html
Exemple de JTextPane : http ://java.sun.com/docs/books/tutorial/uiswing/components/simpletext.html
Tutoriel de Sun sur le Swing : http ://java.sun.com/docs/books/tutorial/uiswing/TOC.html

Correction

Fonctionnement :
-Votre travail devra fonctionner avec Linux sur les PC dans les laboratoires du DIRO. Votre responsabilité est d'effectuer les vérifications requises avant de remettre votre travail. Aucune excuse du type *"ça fonctionnait pourtant chez moi ..."* ne sera acceptée.
-Vos programmes ne doivent pas générer d'avertissements (*warning*) lors de la compilation.
-Si vous connaissez des défauts ou des bogues de votre programme, vous devez les indiquer dans votre man page.

Critères d'évaluation du code remis :
- Le programme doit avoir une entete qui précise : auteur, date (**sinon il y aura penalite**).
- Structure du programme : le programme doit être facile à comprendre et découpé en méthodes dont les significations sont claires et simples.
- Commentaires abondants qui expliquent la signification de chaque méthode et les parties importantes de ces méthodes, sans simplement répéter ce qui est évident grâce à la syntaxe. Vous **DEVEZ** commenter votre code. Si vous ne le faites pas, vous allez perdre des points.

  – Les grandes parties de votre programmes doivent être commentés. Par exemple : le code du processus parent, le code des enfants, etc.

- La définition de chaque fonction doit être précédée d'un commentaire décrivant se que fait la fonction, sa valeur de retour (sauf lorsque la fonction est de type void) et ses paramêtres (se à quoi ils servent, s'il y a des restriction sur leur valeur, etc).
- Les boucles complexes doivent être précédées d'un commentaire.
- Les macros (#define, #ifndef, #if, etc) doivent être précédés d'un commentaire, à moins que leur nom indique clairement leur fonction.

- Les noms des fonctions et des variables devraient indiquer leurs fonctions.
- Votre code doit être correctement indenté (Si vous utilisez emacs ou - Xemacs, cela peut être fait automatiquement !). Un travail mal documenté ou dont la présentation laisse à désirer recevra la moitié de la note, même s'il fonctionne correctement.
- Vous ne devez **PAS** changer les noms de fichiers et des fonctions qui sont demandé dans l'énoncé de ce devoir.

Modalités de remise :
- Aucun retard ne sera accepté. Vous devez remettre vos fichiers à l'aide du programme de remise électronique **le 21 decembre avant 9h00 (en avant-midi)**. Vous **devez** ensuite vous présenter au local 3181 à 10h00 pour faire une démonstration de l'exécution de votre programme.
La commande de remise est la suivante[3] :

```
remise 6891 projet serveur.c Makefile Fureteur.java ...etc...
```

- Ce travail peut être fait en équipe de 4 personnes (maximum).
- **Aucun plagiat ne sera accepté**. Toute solution copiée d'un livre ou d'ailleurs sans références recevra **la note 0**. Pensez à inclure une bibliographie dans votre man page s'il y a lieu.
- Un utilitaire de comparaison automatique sera employé pour vérifier si vos devoir se ressemblent trop. Si c'est le cas, chaque équipe recevrat **la note 0**.
- Nous n'acceptons **pas** les devoirs remis par courrier électronique ou sur disquette.
- Il est possible que des précisions ou des modifications de l'énoncé soient envoyées par courriel. Il est de votre responsabilité de tenir compte de ces informations dans votre travail, sinon vous pourriez être pénalisés.

Bon travail !

[3] La commande "tar" ne fait pas de remise électronque ;-)