

Travail sur les transferts de fichiers

Réaliser un mini serveur et un mini client de type FTP possédant les caractéristiques suivantes :

- Lorsqu'un client se connectera, le serveur enverra la liste des fichiers downloadables situés dans un dossier précis du serveur (à déterminer au préalable). Les fichiers peuvent être de tous formats et de toutes tailles possibles. Le serveur attendra ensuite du client le nom du fichier à downloader et enverra le fichier au client. Il faudra veiller à optimiser le serveur afin que le transfert soit le plus rapide possible (utiliser une API adéquate pour minimiser les changements de contexte userspace/kernel (cf. cours : `sendfile()`).
- Le client recevra l'adresse du serveur sur la ligne de commande. Une fois la connexion réalisée, le client affichera la liste des fichiers du serveur et demandera d'entrer le nom d'un fichier à downloader (ou un numéro correspondant au fichier désiré). Le fichier reçu sera enregistré sur le disque du client.

1 Protocole

La connexion est engagée par le client en envoyant un byte d'identification au serveur. Ce byte permet de vérifier que le client parle bien le bon protocole. Il pourrait aussi permettre au serveur de sélectionner un autre protocole en envoyant un byte d'identification différent.

Si le byte d'identification n'est pas celui attendu, le serveur ferme la connexion mais reste à l'écoute de nouvelles connexions. Sinon, il répond avec la liste des fichiers téléchargeables préfixée de sa taille. Ces fichiers sont ceux auxquels le serveur a accès en lecture dans le sous-répertoire *files*.

Si le dossier *files* ne contient aucun fichier, aucun fichier accessible au serveur ou qu'il n'existe pas, le serveur envoie une liste vide (un préfixe à 0 qui n'est suivi de rien) puis ferme la connexion. Le client affiche un message d'erreur et s'arrête.

Après réception d'une liste valide, le client attend le choix de fichier de l'utilisateur : le nombre correspondant au fichier lors de l'affichage de la liste. Une nouvelle liste ne contenant que les informations sur ce fichier est envoyée au serveur.

Le serveur renvoie ce fichier préfixé de sa taille en utilisant *sendfile*.

Le client reçoit ce fichier et l'écrit par morceaux de taille fixe définie dans les constantes afin de ne pas stocker en mémoire vive la totalité du fichier si celui-ci est trop volumineux. Les fichiers sont téléchargés dans le sous-répertoire *download* relatif à l'endroit à partir duquel le client a été lancé. Si le fichier existe déjà dans le dossier *download*, il est écrasé.

2 Compilation

Pour la sérialisation des données, j'ai repris la librairie *libserial* précédemment développée. Je l'ai mise à jour à cause d'un bug découvert lors de mes tests et j'utilise maintenant la version 1.1. Pour la mise en place des connexions TCP, j'ai repris la librairie *libtcp* dont j'ai amélioré les codes d'erreurs : ce projet utilise donc la version 2.0 de la librairie.

Comme pour les projets précédents, j'ai compilé mon code en local en exécutant `gmake ex-files` à la racine de mes projets. J'ai ensuite copié le client et le serveur sur les machines virtuelles et je les ai rendus exécutables avec `chmod +x`. J'ai enfin ajouté les versions mises à jour de mes bibliothèques au dossier *lib* et j'ai recréé les liens symboliques avec `ldconfig -n ..`.

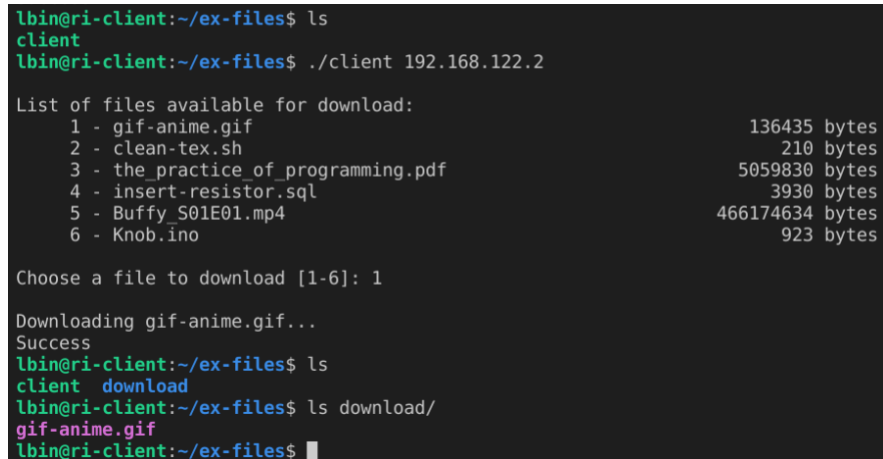
J'obtiens les structures de fichiers suivantes sur mes machines virtuelles :

```
- ex-files
  client      | server
  + download  | + files
+ ex-lib
+ ex-serial
- lib
  libserial.so.1 -> libserial.so.1.1
  libserial.so.1.1
  libtcp.so.1 -> libtcp.so.1.0
  libtcp.so.1.0
  libtcp.so.2 -> libtcp.so.2.0
  libtcp.so.2.0
```

3 Tests

3.1 Cas nominal

Je teste le téléchargement d'un fichier simple sans avoir préalablement créé le dossier *download* :



```
lbin@ri-client:~/ex-files$ ls
client
lbin@ri-client:~/ex-files$ ./client 192.168.122.2

List of files available for download:
 1 - gif-anime.gif           136435 bytes
 2 - clean-tex.sh            210 bytes
 3 - the_practice_of_programming.pdf 5059830 bytes
 4 - insert-resistor.sql      3930 bytes
 5 - Buffy_S01E01.mp4        466174634 bytes
 6 - Knob.ino                923 bytes

Choose a file to download [1-6]: 1

Downloading gif-anime.gif...
Success
lbin@ri-client:~/ex-files$ ls
client download
lbin@ri-client:~/ex-files$ ls download/
gif-anime.gif
lbin@ri-client:~/ex-files$
```

Le dossier a bien été créé et le fichier téléchargé s'y trouve.

Côté serveur, tout s'est bien déroulé :

```
lbin@ri-server:~/ex-files$ ./server
[server] waiting for connections...
[server] connection received from 192.168.122.21...
[server:192.168.122.21] list of available files sent
[server:192.168.122.21] successfully sent gif-anime.gif
[server:192.168.122.21] closing
^C
lbin@ri-server:~/ex-files$
```

Pour vérifier que les deux fichiers sont identiques, je télécharge le fichier d'origine sur le client avec `scp` et j'exécute un `diff`. Le code de retour du `diff` est 0. Les deux fichiers sont bien identiques.

```
lbin@ri-client:~/ex-files$ scp 192.168.122.2:ex-files/files/gif-anime.gif .
lbin@192.168.122.2's password:
gif-anime.gif 100% 133KB 46.4MB/s 00:00
lbin@ri-client:~/ex-files$ diff download/gif-anime.gif gif-anime.gif
lbin@ri-client:~/ex-files$ echo $?
0
lbin@ri-client:~/ex-files$
```

3.2 Envoi d'un gros fichier

Parmi les différents fichiers testés, j'ai testé le téléchargement d'un fichier de 3.1 GB :

```
lbin@ri-server:~/ex-files$ ./server
[server] waiting for connections...
[server] connection received from 192.168.122.21...
[server:192.168.122.21] list of available files sent
[server:192.168.122.21] successfully sent 2003-Oldboy.mkv
[server:192.168.122.21] closing
```

```
lbin@ri-client:~/ex-files$ ./client 192.168.122.2
List of files available for download:
 1 - gif-anime.gif 136435 bytes
 2 - binacchi-cv-en.pdf 182394 bytes
 3 - 2003-Oldboy.mkv 3103372065 bytes
 4 - Barbara_KRUGER_Untitled_1989.jpg 782099 bytes
 5 - clean-tex.sh 210 bytes
 6 - binacchi-cv-en.tex 8117 bytes
 7 - the_practice_of_programming.pdf 5059830 bytes
 8 - insert-resistor.sql 3930 bytes
 9 - Buffy_S01E01.mp4 466174634 bytes
10 - Knob.ino 923 bytes

Choose a file to download [1-10]: 3
Downloading 2003-Oldboy.mkv...
Success
```

Le code de retour du diff est 0. Les deux fichiers sont bien identiques.

```

Downloading 2003-Oldboy.mkv...
Success
lbin@ri-client:~/ex-files$ scp 192.168.122.2:ex-files/files/2003-Oldboy.mkv .
lbin@192.168.122.2's password:
2003-Oldboy.mkv                                100% 2960MB 112.8MB/s   00:26
lbin@ri-client:~/ex-files$ diff download/2003-Oldboy.mkv 2003-Oldboy.mkv
lbin@ri-client:~/ex-files$ echo $?
0
lbin@ri-client:~/ex-files$

```

3.3 Envoi d'un mauvais byte d'identification

Si un mauvais byte d'identification est envoyé au serveur, ce dernier ferme la connexion mais reste à l'écoute de nouvelles connexions :

```

[lbin@nibbler ex-files]$ ./client 127.0.0.1
[lbin@nibbler ex-files]$ ./client 127.0.0.1

```

```

[server] waiting for connections...
[server] connection received from 127.0.0.1...
[server:127.0.0.1] closing - wrong identification byte
[server] connection received from 127.0.0.1...
[server:127.0.0.1] closing - wrong identification byte

```

3.4 Aucun fichier téléchargeable

Si le dossier contenant les fichiers téléchargeables n'existe pas sur le serveur, il envoie une liste vide au client puis ferme la connexion mais reste à l'écoute de nouvelles connexions. Le client affiche un message d'erreur avant de s'arrêter. Ce test donne les mêmes résultats si le dossier existe mais ne contient aucun fichier ou si il ne contient aucun fichier accessible en lecture pour le serveur.

```

lbin@ri-server:~/ex-files$ ./server
[server] waiting for connections...
[server] connection received from 192.168.122.21...
[server:192.168.122.21] closing: no file available for download

```

```

lbin@ri-client:~/ex-files$ ./client 192.168.122.2
List of files available for download:
No file available for download
lbin@ri-client:~/ex-files$

```

Puisque le serveur reste à l'écoute de nouvelles connexions, il est possible d'ajouter le dossier *files* avec des fichiers pour qu'une nouvelle connection d'un client fonctionne :

```

lbin@ri-server:~/ex-files$ ./server
[server] waiting for connections...
[server] connection received from 192.168.122.21...
[server:192.168.122.21] closing: no file available for download
[server] connection received from 192.168.122.21...
[server:192.168.122.21] list of available files sent

```

```
lbin@ri-client:~/ex-files$ ./client 192.168.122.2

List of files available for download:
 1 - gif-anime.gif                136435 bytes
 2 - binacchi-cv-en.pdf           182394 bytes
 3 - 2003-Oldboy.mkv             691470336 bytes
 4 - Barbara_KRUGER_Untitled_1989.jpg 782099 bytes
 5 - clean-tex.sh                 210 bytes
 6 - binacchi-cv-en.tex           8117 bytes
 7 - the_practice_of_programming.pdf 5059830 bytes
 8 - insert-resistor.sql          3930 bytes
 9 - Buffy_S01E01.mp4            466174634 bytes
10 - Knob.ino                     923 bytes

Choose a file to download [1-10]: █
```

3.5 Inputs invalides

Lorsque l'utilisateur choisi un fichier à télécharger, seul un nombre entier entre 1 et la taille de la liste est accepté, sinon un message d'erreur est affiché. Si l'utilisateur appuie juste sur **enter**, rien ne se passe. Si un nombre entier est suivi d'autres nombres entiers ou de caractères, seul le premier nombre entier est pris en compte.

```
lbin@ri-client:~/ex-files$ ./client 192.168.122.2

List of files available for download:
 1 - gif-anime.gif                136435 bytes
 2 - binacchi-cv-en.pdf           182394 bytes
 3 - 2003-Oldboy.mkv             3103372065 bytes
 4 - Barbara_KRUGER_Untitled_1989.jpg 782099 bytes
 5 - clean-tex.sh                 210 bytes
 6 - binacchi-cv-en.tex           8117 bytes
 7 - the_practice_of_programming.pdf 5059830 bytes
 8 - insert-resistor.sql          3930 bytes
 9 - Buffy_S01E01.mp4            466174634 bytes
10 - Knob.ino                     923 bytes

Choose a file to download [1-10]: -2
Please enter a valid id [1-10]: 0
Please enter a valid id [1-10]: 11
Please enter a valid id [1-10]: deux
Please enter a valid id [1-10]:

1 2 3 4 5 6

Downloading gif-anime.gif...
Success
lbin@ri-client:~/ex-files$ █
```

3.6 Tests avec valgrind

Enfin j'ai testé mon programme avec valgrind pour vérifier l'absence de fuites mémoire.

Ce test m'a permis de mettre en évidence un bug dans la librairie *libserial* : lors de la désérialisation des chaînes de caractère, un 0 aurait dû être ajouté en fin de chaîne.

```
List of files available for download:
==184947== Invalid read of size 1
==184947==    at 0x483DB94: strlen (vg_replace_strmem.c:459)
==184947==    by 0x48DF97D: __vfprintf_internal (in /usr/lib64/libc-2.31.so)
==184947==    by 0x48CA40E: printf (in /usr/lib64/libc-2.31.so)
==184947==    by 0x4018C6: print_list (file.c:130)
==184947==    by 0x4014F3: main (client.c:66)
==184947== Address 0x4a3f132 is 0 bytes after a block of size 18 alloc'd
==184947==    at 0x483A809: malloc (vg_replace_malloc.c:307)
==184947==    by 0x484F5D7: read_str (serial-util.c:119)
==184947==    by 0x401BA9: receive_list (file.c:226)
==184947==    by 0x4014A1: main (client.c:59)
```

J'ai donc mis à jour la librairie et sa version.

Le client libère bien chaque espace de mémoire alloué :

```
==188315== HEAP SUMMARY:
==188315==    in use at exit: 0 bytes in 0 blocks
==188315==   total heap usage: 27 allocs, 27 frees, 7,150 bytes allocated
==188315== All heap blocks were freed -- no leaks are possible
==188315== For lists of detected and suppressed errors, rerun with: -s
==188315== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Tout comme le serveur :

```
==188316== HEAP SUMMARY:
==188316==    in use at exit: 0 bytes in 0 blocks
==188316==   total heap usage: 48 allocs, 48 frees, 82,162 bytes allocated
==188316== All heap blocks were freed -- no leaks are possible
==188316== For lists of detected and suppressed errors, rerun with: -s
==188316== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```