

Compte rendu FTP4

Plan

- I. Introduction
- II. Principales réalisations
- III. Tests
- IV. Difficultés rencontrés
- V. Exécution

I. Introduction

Le but du tp est de construire un serveur de fichier inspiré par le protocole FTP. Le travail demandé permet de mieux apprendre les notions vues en cours jusqu'à maintenant à savoir les processus, les signaux, communications entre un client et serveur de façon concurrente... Ceci se résume en quelques sortes la connaissance des interfaces d'accès aux socket, les fonctions d'accès, les structures de données pour les sockets etc, ainsi cela nous a permis d'envoyer un fichier a partir du serveur FTP au client dans son dossier courant , ensuite de charger un fichier à distance par blocs, de gerer les pannes cotés client en cas de craches pendant un transfert, nous n'avons pas pu implémenter la question 4 par manque de temps et soucis de comprehension, ensuite on a implementer le question 5 mais lors de l'exécution on avait des erreurs du coup le programme ne fonctionnait pas , on a pu implementer les différentes commandes de la question 6 et 7 .

ii_ Principales réalisations

question 1 : realisation de la commande « get »

dans un premier temps l orsqu'on exécute l'application serveur <./serveur> , elle se met en attente sur une socket, ensuite l'application client contact le Serveur sur la Socket en utilisant la fonction Clientfd , lorsqu'on exécute <./client localhost> ainsi la connexion etablie , le client fait une demande de fichier on serveurFTP grace a la commande < get Nomfichier> dans l'implementation, on

teste d'abord si la commande saisie est de type « get » grace la fonction CTRCMP, ensuite on crée et ouvre avec la fonction OPEN le fichier , on envoie la commande qui se trouve dans le buffer au serveurFTP grace à la primitive Rio_writen et le serveurFTP lit la commande “ get nomfichier>> avec la primitive Rio_readlineb, ainsi on extrait la commande « get » dans une variable et aussi le nom de fichier dans une autre variable grace a la fonction strtok, on compare si c'est une commande get grace à CTRCMP, ensuite on fait la lecture du dit fichier avec la fonction Open , on extrait le taille réel du fichier grâce au descripteur de fichier aa partir de la fonction implémenté size_line1(fd), le serveur envoie la taille du fichier au client avec la fonction Rio_writen(connfd, recup, strlen(recup)); le client recupère la taille du fichier avec la fonction Rio_readlineb(rio, size, MAX), ensuite vérifie si la taille existant est égale a la taille de fichier récupérer ainsi affiche un message de fichier déjà existant sinon le CLIENT recoit le fichier avec la fonction `int x=Rio_readn(clientfd, mem, MAX))` ; et l'écrit dans le descripteur de fichier avec `write(fd , mem, x)`; le SERVEUR lit le descripteur de fichier avec la fonction `n=Rio_readn(fd, mem, MAX)`; et envoie les données grace à `Rio_writen(connfd, mem, n)` aux CLIENT .

Question 2 : Découpage du fichier

nous avons pu réaliser cette étape , du coup nous avons effectuer les même procédé comme la question 1 et au moment de l'envoi des fichiers du serveur au client on a fait un découpage paquet par paquet , nous avons d'abord defini la taille de chaque paquet d'écriture et de lecture coté Client et Serveur “ `#define MAX 8192` ” , ensuite on fait un calcule pour savoir le nombre de lot de paquet qui pourra être envoyer au client (`nb_pq = (size_fichier-size_rcp)/MAX;`), on fait une boucle For qui envoie les paquets defini lot par lot , et si y'a des paquets restante non envoyer a la fin de la boucle For , on fait un `Rio_readn()`, `Rio_writen()`. Et dans le code Client nous avons fait pareil mais avec la fonction `Rio_readn(mem)`, et `write(fd)`.

Question 3 : Gestion simple des pannes coté client

nous avons utilisés une fonction `LSEEK()` qui permet de placer le curseur avant à la dernière caractère -1 , de la chaine lu ,`lseek(fd, size_rcp-1, SEEK_SET)`; du coup ça nous permet de retélécharger le fichier interrompu sans le retélécharger dès le debut .

Question 4 : Serveurs FTP avec équilibrage de charge

nous n'avons pas pu implémenter la question 4 par manque de temps et soucis de compréhension.

Question 5 : Plusieurs demandes de fichiers par connexion

nous avons implémenter mais ça ne fonctionnait pas lors de l'exécution, on s'est basé sur les boucles infinis, on a fais une boucle infinis dans le coté client, et une autre boucle infini dans le coté du serveur, avec des petits changements dans le code, mais ca na pas marché on a compris que cette étape se base sur l'utilisation des boucles infinis pour pouvoir faire plusieurs demandes des fichiers mais on a pas a réussi à la faire marcher

Question 6 : Commandes `ls`, `pwd` et `cd`

nous avons pu implémenter les 3 commandes ,

LS : grace a la foncion `STRCPM` , nous avons acces au code de LS , la fonction `Rio_writen(clientfd, cmd, MAXLINE)`; envoie la commande au Serveur et dans le Serveur la fonction `opendir(".");` permettra d'ouvrir un tube pour lister le contenu du répertoire courant ensuite on fait une boucle `While` pour Extraire les noms des fihiers existant dans le repertoire courant , `sprintf(recup1, " %s", lecture->d_name);` , ensuite on fait un `Rio_writen(connfd, list_rep, strlen(list_rep));` pour envoyer le Inom des fichier au Client et le Client fait une lecture avec la fonction `Rio_readlineb(&rio, tampon, MAXLINE))` et les affiche.

PWD : l'implémentation coté client est pareil et au niveau ServeurFTP quand il recoit la commande ,dans le code défini nous utilisons la

fonction `getcwd()` qui copie le chemin d'accès absolu du répertoire de travail courant et le stock dans une variable, grâce a la fonction `Rio_writen()` le résultat est envoyé au client .

La commande `CD` à été expliquer dans la partie Tests.

Question 7: Nous avons pu implémenter la Commandes `mkdir` et `put`, que j'ai pu expliquer dans la partie Tests

III_Tests

question 1 et 2 : pour valider la question 1 et 2 nous avons essayé de télécharger tous types de fichier soit `.txt` `.pdf` `.jpg` `.c` et aussi des fichiers de type `.mp3`, et des vidéos de type `.mp4` avec un telechargement d'un seul coup ou par paquets .

Get avec un fichier texte

```
zm@zm-Lenovo-G50-30:~/Bureau/ReseauFTP/Client$ ./Client localhost
client connected to server OS
ftp>get toto.txt
téléchargement du fichier toto.txt de taille 0 / 134
Debut de telechargement d'un fichier de taille: 134
transfers en cours de 1 paquets ...(55 / 134)
transfers en cours de 2 paquets ...(110 / 134)
transfers en cours du reste des 24 Octets. (134 / 134)
tranfer successfully complete
134 bytes received in 2001067 seconds (0.000000 kbytes/s)
ftp>
```

Get avec une image `img02.png` par paquet

```
zm@zm-Lenovo-G50-30:~/Bureau/ReseauFTP/Client$ ./Client localhost
client connected to server OS
ftp>get img02.png
téléchargement du fichier img02.png de taille 0 / 596
Debut de telechargement d'un fichier de taille: 596
transfers en cours de 1 paquets ...(55 / 596)
transfers en cours de 2 paquets ...(110 / 596)
transfers en cours de 3 paquets ...(165 / 596)
transfers en cours de 4 paquets ...(220 / 596)
transfers en cours de 5 paquets ...(275 / 596)
transfers en cours de 6 paquets ...(330 / 596)
transfers en cours de 7 paquets ...(385 / 596)
transfers en cours de 8 paquets ...(440 / 596)
transfers en cours de 9 paquets ...(495 / 596)
transfers en cours de 10 paquets ...(550 / 596)
transfers en cours du reste des 46 Octets. (596 / 596)
tranfer successfully complete
596 bytes received in 10023074 seconds
ftp>
```

Get avec une image cr2.pdf par paquet

```
zm@zm-Lenovo-G50-30:~/Bureau/ReseauFTP/Client$ ./Client localhost
client connected to server OS
ftp>get cr2.pdf
téléchargement du fichier cr2.pdf de taille 0 / 6487
Debut de telechargement d'un fichier de taille: 6487
transfers en cours de 1 paquets ...(830 / 6487)
transfers en cours de 2 paquets ...(1660 / 6487)
transfers en cours de 3 paquets ...(2490 / 6487)
transfers en cours de 4 paquets ...(3320 / 6487)
transfers en cours de 5 paquets ...(4150 / 6487)
transfers en cours de 6 paquets ...(4980 / 6487)
transfers en cours de 7 paquets ...(5810 / 6487)
transfers en cours du reste des 677 Octets. (6487 / 6487)
tranfer successfully complete
6487 bytes received in 7002662 seconds
ftp>
```

- Get avec fichier video et audio, du coup nous avons modifié la taille du paquets definie par #define MAX 1830 , pour que le temps téléchargement puisse etre rapide.

question 3 :

nous avons pu interrompu un fichier en cours de téléchargement avec « ctrl+c » et le relancer a son etat d'arret sans quil ne recommence a zéro, on peut vérifier aussi en ajoutant un sleep.

```
zm@zm-Lenovo-G50-30:~/Bureau/ReseauFTP/Client$ ./Client localhost
client connected to server OS
ftp>get img02.png
téléchargement du fichier img02.png de taille 0 / 596
Debut de telechargement d'un fichier de taille: 596
transfers en cours de 1 paquets ...(95 / 596)
transfers en cours de 2 paquets ...(190 / 596)
transfers en cours de 3 paquets ...(285 / 596)
transfers en cours de 4 paquets ...(380 / 596)
^C
zm@zm-Lenovo-G50-30:~/Bureau/ReseauFTP/Client$ ./Client localhost
client connected to server OS
ftp>get img02.png
téléchargement du fichier img02.png de taille 380 / 596
Debut de telechargement d'un fichier de taille: 596
transfers en cours de 1 paquets ...(475 / 596)
transfers en cours de 2 paquets ...(570 / 596)
transfers en cours du reste des 26 Octets. (596 / 596)
tranfer successfully complete
596 bytes received in 3002217 seconds
ftp>
```

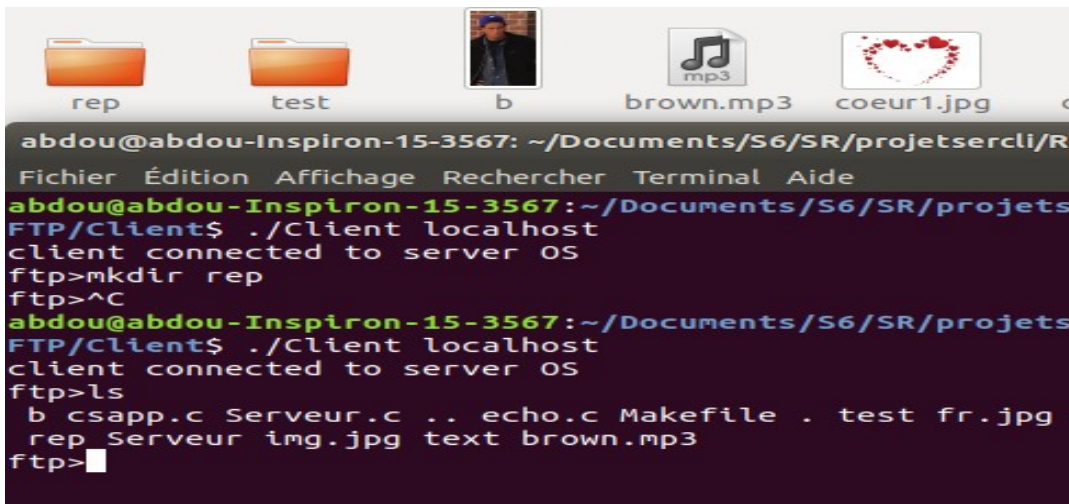
commande PWD : affiche le chemin courant du serveurFTP, c'est la même implémentation que celui de la commande LS coté Client et au niveau ServeurFTP lorsqu'il reçoit la commande PWD , grâce a la fonction GETCWD sauvegarde le chemin courant dans la variable tampon, et le transfert au client grâce à Rio_writen().

```
zm@zm-Lenovo-G50-30:~/Bureau/ReseauFTP/Client$ ./Client localhost
client connected to server 05
ftp>pwd
/home/zm/Bureau/ReseauFTP/Serveur
ftp>
```

commande LS : renvoie le contenu du repertoire courant du ServeurFTP, c'est la même implémentation que la commande pwd coté client et au niveau Serveur la fonction readidir() permet de recupérer les noms des fichiers dans le répertoire et le transfert au client.

```
zm@zm-Lenovo-G50-30:~/Bureau/ReseauFTP/Client$ ./Client localhost
client connected to server 05
ftp>ls
test Serveur csapp.h d1 cr1.pdf Serveur.c .. cr2.pdf csapp.c echo
.c toto.txt fr.jpg . img02.png Makefile dossierz
ftp>
```

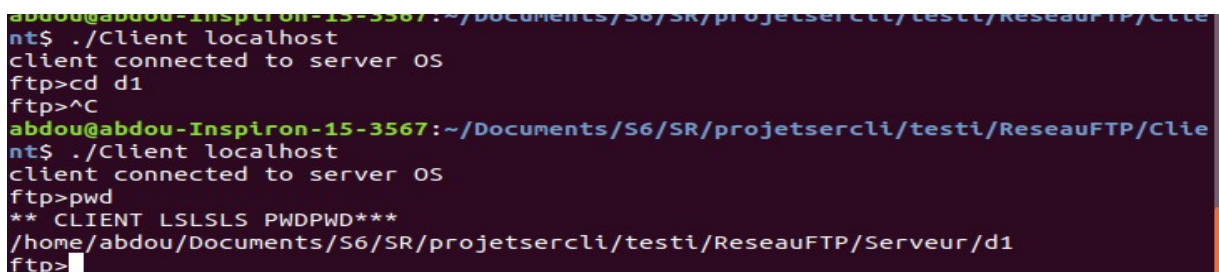
Commande MKDIR permet de créer un dossier sur le serveur FTP, le client envoie la commande au serveurFTP et le serveurFTP exécute la fonction mkdir(nomDossier) qui crée le dossier.



The screenshot shows a terminal window with a dark background. At the top, there is a horizontal bar with several icons: two orange folders labeled 'rep' and 'test', a small portrait of a man labeled 'b', a music note icon labeled 'brown.mp3', and a heart icon labeled 'coeur1.jpg'. Below this bar, the terminal text shows the user 'abdou' at a prompt 'abdou@abdou-Inspiron-15-3567: ~/Documents/S6/SR/projetsercli/R'. The user enters './Client localhost', and the terminal shows 'client connected to server OS'. Then, the user enters 'ftp>mkdir rep', and the terminal shows 'ftp>^C'. Next, the user enters './Client localhost', and the terminal shows 'client connected to server OS'. Then, the user enters 'ftp>ls', and the terminal shows a list of files: 'b csapp.c Serveur.c .. echo.c Makefile . test fr.jpg rep Serveur img.jpg text brown.mp3'. Finally, the user enters 'ftp>' and the terminal shows a cursor.

Commande CD :

le client envoie la commande CD au serveurFTP et le serveurFTP exécute la fonction chdir(nomRepertoire) qui change le répertoire courant vers le repertoire qui à été donné a la fonction.



The screenshot shows a terminal window with a dark background. The terminal text shows the user 'abdou' at a prompt 'abdou@abdou-Inspiron-15-3567: ~/Documents/S6/SR/projetsercli/test/ReseauFTP/Client'. The user enters './Client localhost', and the terminal shows 'client connected to server OS'. Then, the user enters 'ftp>cd d1', and the terminal shows 'ftp>^C'. Next, the user enters './Client localhost', and the terminal shows 'client connected to server OS'. Then, the user enters 'ftp>pwd', and the terminal shows '** CLIENT LSLSL PWD***' and '/home/abdou/Documents/S6/SR/projetsercli/test/ReseauFTP/Serveur/d1'. Finally, the user enters 'ftp>' and the terminal shows a cursor.

Commande PUT : permet de charger un fichier du client vers le serveurFTP, l'implémentation est presque pareille a celle de la commande GET mais inversé car c'est le serveurFTP qui reçoit le fichier, du coup nous pu envoyer tous types de fichier texte image etc

```

zm@zm-Lenovo-G50-30:~/Bureau/projetSR/Client$ ./Client localhost
client connected to server OS
ftp>ls
  Serveur csapp.h d1 Serveur.c img.jpg .. b text csapp.c echo.c vid.mp4 . brown.m
p3 Makefile coeur1.jpg
zm@zm-Lenovo-G50-30:~/Bureau/projetSR/Client$ ./Client localhost
client connected to server OS
ftp>put img03.png
Envoi du fichier img03.png de taille 596.
Lecture à partir de 0 / 596...
  1 paquets de taille MAXLINE + un paquet de taille 596
Envoi : (596) 596 octets
Fin de lecture.
zm@zm-Lenovo-G50-30:~/Bureau/projetSR/Client$ ./Client localhost
client connected to server OS
ftp>ls
  img03.png Serveur csapp.h d1 Serveur.c img.jpg .. b text csapp.c echo.c vid.mp4
. brown.mp3 Makefile coeur1.jpg
zm@zm-Lenovo-G50-30:~/Bureau/projetSR/Client$

```

Difficultés rencontrés :

- la première difficulté qu'on a eu c'était le matériel, un de nous avait un PC très lent (après avoir ajouter linux a coté du windows), et l'autre avait des difficultés à installer linux sur son PC.
- implémenter la première étape a pris du temps, car on a changer ce qu'on a implémenté au début pour qu'il soit adapté avec étape 3.
- l'étape 4 on a pris beaucoup temps, on devait tout d'abord la comprendre, et on a essayer à la faire mais on a pas réussi.

Execution :

pour l'exécution, il suffit d'ouvrir deux terminaux, un dans le répertoire du client, et l'autre dans le repertoire du serveur, et lancer un make, apres dans le coté vous tapez : ./Serveur et un entrez,
pour le coté client : vous tapez : ./Client localhost.