

2. Implement command line interface FTP using raw socket

What you have to do in this term project is to implement FTP protocol. All client interface is followed from the existing program “sftp”. For simplicity, your program will be tested in the Ubuntu system. Actually, your implementation is working exactly same as the “sftp”.

Experimental setup

1. In server side, we execute “python ftp_server.py” to construct server.
2. In client side, we execute “python ftp_client.py” to execute client. Then, the program will immediately require authentication information.

Requirements

Correctness

- A client has to authenticate their identity.
- A client can traverse the server directory structure.
- A client can traverse the client directory structure.
- A file needs to be uploaded from client to server
- A file needs to be downloaded from server to client.
- Multiple file could be uploaded from client to server.
- Multiple file could be downloaded from server to client.
- A server can handle more than 100 clients at the same time.

Performance

- The upload speed will be similar to that of existing implementation.
- The download speed will be similar to that of existing implementation.

Term project will be rigorously tested so that you must check many edge cases. If you think some edge cases could be handled in a variety of ways, provide how to handle that case. Just ignoring it without proper reasoning will be considered as not implementing that requirement.

Submission

You have to submit a zip file with “**ftp_server.py**”, “**ftp_client.py**”, **pdf report**, **ppt** and **demo video**.

```

sftp> help
Available commands:
bye                               Quit sftp
cd path                           Change remote directory to 'path'
cnggrp grp path                   Change group of file 'path' to 'grp'
chmod mode path                   Change permissions of file 'path' to 'mode'
chown own path                    Change owner of file 'path' to 'own'
df [-hi] [path]                  Display statistics for current directory or
                                  filesystem containing 'path'
exit                              Quit sftp
get [-afPpRr] remote [local]     Download file
reget [-fPpRr] remote [local]    Resume download file
reput [-fPpRr] [local] remote    Resume upload file
help                              Display this help text
lcd path                          Change local directory to 'path'
lls [ls-options] [path]          Display local directory listing
lmkdir path                       Create local directory
ln [-s] oldpath newpath          Link remote file (-s for symlink)
lpwd                              Print local working directory
ls [-lafhlNrSt] [path]           Display remote directory listing
lumask umask                      Set local umask to 'umask'
mkdir path                        Create remote directory
progress                          Toggle display of progress meter
put [-afPpRr] local [remote]     Upload file
pwd                               Display remote working directory
quit                              Quit sftp
rename oldpath newpath            Rename remote file
rm path                           Delete remote file
rmdir path                        Remove remote directory
symlink oldpath newpath           Symlink remote file
version                           Show SFTP version
!command                          Execute 'command' in local shell
!                                 Escape to local shell
?                                 Synonym for help

```

Fig. 2.a. Client Interface (the commands inside the red box are the requirement).

```

(base) sh0416@digpu:~$ sftp localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is S[REDACTED]
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
sh0416@localhost's password:
Connected to localhost.

```

Fig. 2.b. Authentication.