Project 1

Faizan Raza (mr5985)
Khadija Khalid (kk4597)

Project Description

The objective of this project is to develop a simplified FTP server and client that adheres to the basic functionalities outlined in the FTP protocol. The FTP server is designed to handle multiple simultaneous connections and support various commands such as USER, PASS, LIST, STOR, RETR, CWD, PWD, !CWD, and !PWD. This project ensures that the server can manage concurrent connections using the fork() and select() functions, facilitating efficient handling of multiple client requests.

Testing the Functionality and Success

To ensure the robustness and functionality of the FTP server, we devised comprehensive test cases that cover all the implemented commands. Each command was tested for both standard operation and edge cases. For instance, the USER and PASS commands were tested to validate proper authentication, while commands like LIST, STOR, and RETR were verified for correct file operations. Special attention was given to scenarios involving multiple clients performing operations simultaneously. Our testing process involved running the server and client in various configurations, issuing commands in succession, and verifying the expected outcomes. This thorough approach confirmed that the server behaves correctly under normal and edge case conditions, ensuring reliable file transfer operations.

Handling Multiple Clients

A significant aspect of our testing focused on verifying the server's capability to handle multiple clients concurrently. We tested multiple clients connected to the server simultaneously, each performing different operations such as file uploads, downloads, and directory changes. Successive command execution was tested by having clients repeatedly issue commands like LIST, STOR, and RETR, ensuring the server could handle continuous operations without failure. This rigorous testing demonstrated that the server efficiently manages multiple clients and successive commands, maintaining its functionality and stability in a multi-client environment.

Test Cases

USER Command

1. **Valid USER Command**
    - Input: USER bob
    - Expected Output: 331 Username OK, need password.

```
Socket successfully created..
Connected to the server..
ftp> USER bob
331 Username OK, need password.
```

2. **Invalid USER Command**
    - Input: USER invaliduser
    - Expected Output: 530 User not found.

```
Connected to the server..
ftp> USER bsb
530 User not found.
```

PASS Command

1. **Valid PASS Command after valid USER**
    - Input: PASS donuts (following USER bob)
    - Expected Output: 230 User logged in, proceed.

```
Socket successfully created..
Connected to the server..
ftp> USER bob
331 Username OK, need password.
ftp> PASS donuts
230 User logged in, proceed.
```

2. **Invalid PASS Command after valid USER**
    - Input: PASS wrongpassword (following USER bob)
    - Expected Output: 530 Wrong password.

```
Socket successfully created..
Connected to the server..
ftp> USER bob
331 Username OK, need password.
ftp> PASS wrongpassword
530 Wrong password.
```

3. **PASS Command without USER**
    - Input: PASS donuts
    - Expected Output: 503 Bad sequence of commands.

```
Socket successfully created..
Connected to the server..
ftp> PASS donuts
503 Bad sequence of commands.
```

Multiple Clients USER and PASS

```
Socket successfully created..
Connected to the server..
ftp> USER bob
331 Username OK, need password.
ftp> PASS donuts
230 User logged in, proceed.
ftp>
```

```
Connected to the server..
ftp> USER john
331 Username OK, need password.
ftp> PASS pizza
230 User logged in, proceed.
ftp>
```

```
From client: USER bob

From client: PASS donuts

From client: USER john

From client: PASS pizza
```

LIST Command

1. **LIST Command with Authenticated User**
    - Input: LIST (after successful USER and PASS)
    - Expected Output: List of files in the current directory

```
Connected to the server..
ftp> USER john
331 Username OK, need password.
ftp> PASS pizza
230 User logged in, proceed.
ftp> LIST
200 PORT command successful.
150 File status okay; about to open data connection.
.
..
ron
.DS_Store
bob
harry
ftp_server
gryffindor.png
226 Transfer completed.
ftp>
```

2. **LIST Command without Authentication**
    - Input: LIST
    - Expected Output: Please login with USER and PASS first.

```
ftp> LIST
Please login with USER and PASS first.
ftp>
```

3. **LIST Command with Multiple Clients**
    - Scenario: Multiple clients send LIST simultaneously after successful authentication
    - Expected Output: Each client receives the list of files in the current directory

```
ftp> USER bob
331 Username OK, need password.
ftp> PASS donuts
230 User logged in, proceed.
ftp> LIST
200 PORT command successful.
150 File status okay; about to open data connection.
.
..
ron
.DS_Store
bob
harry
ftp_server
gryffindor.png
226 Transfer completed.
ftp>
```

```
ftp> USER john
331 Username OK, need password.
ftp> PASS pizza
230 User logged in, proceed.
ftp> LIST
200 PORT command successful.
150 File status okay; about to open data connection.
.
..
ron
.DS_Store
bob
harry
ftp_server
gryffindor.png
226 Transfer completed.
ftp>
```

```
From client: PORT 127,0,0,1,31,145

Client IP: 127.0.0.1, Client Port: 8081
From client: LIST

From client: PORT 127,0,0,1,31,145

Client IP: 127.0.0.1, Client Port: 8081
From client: LIST
```

STOR Command

1. **STOR Command to Upload a File (with repeated uploads)**
    - Input: STOR filename (after successful USER and PASS)
    - Expected Output: 150 File status okay; about to open data connection., followed by 226 Transfer completed.

```
Socket successfully created..
Connected to the server..
ftp> USER bob
331 Username OK, need password.
ftp> PASS donuts
230 User logged in, proceed.
ftp> STOR magic_items.csv
200 PORT command successful.
150 Opening binary mode data connection for file reception.
226 Transfer complete.
ftp> STOR potion.txt
200 PORT command successful.
150 Opening binary mode data connection for file reception.
226 Transfer complete.
ftp>
```

2. **STOR Command without Authentication**
    - Input: STOR filename
    - Expected Output: Please login with USER and PASS first.

```
ftp> STOR sd.txt
Please login with USER and PASS first.
ftp>
```

3. **STOR Command with Invalid Filename**
    - Input: STOR invalid/filename (after successful USER and PASS)
    - Expected Output: 550 No such file or directory.

```
ftp> STOR non_existent.txt
200 PORT command successful.
150 Opening binary mode data connection for file reception.
550 No Such File or Directory.
ftp>
```

4. **STOR Command with Multiple Clients**
    - Scenario: Multiple clients upload different files simultaneously after successful authentication
    - Expected Output: Each file is uploaded successfully without interference

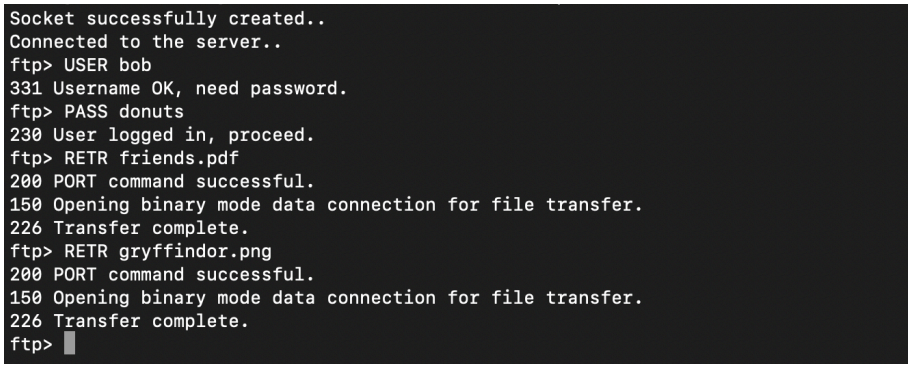
```
Socket successfully created..
Connected to the server..
ftp> USER bob
331 Username OK, need password.
ftp> PASS donuts
230 User logged in, proceed.
ftp> STOR magic_items.csv
200 PORT command successful.
150 Opening binary mode data connection for file reception.
226 Transfer complete.
ftp> STOR potion.txt
200 PORT command successful.
150 Opening binary mode data connection for file reception.
226 Transfer complete.
ftp>
```

```
Socket successfully created..
Connected to the server..
ftp> USER john
331 Username OK, need password.
ftp> PASS pizza
230 User logged in, proceed.
ftp> STOR potion.txt
200 PORT command successful.
150 Opening binary mode data connection for file reception.
226 Transfer complete.
ftp> STOR magic_items.csv
200 PORT command successful.
150 Opening binary mode data connection for file reception.
226 Transfer complete.
ftp>
```

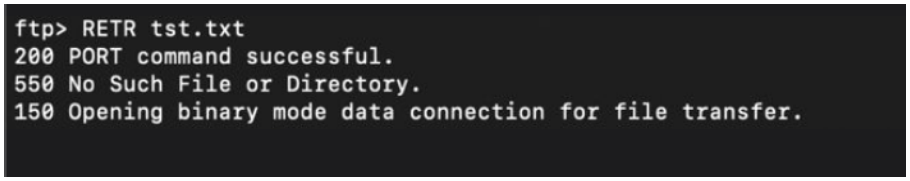
```
From client: PORT 127,0,0,1,31,145

Client IP: 127.0.0.1, Client Port: 8081
From client: STOR magic_items.csv

From client: PORT 127,0,0,1,31,146

Client IP: 127.0.0.1, Client Port: 8082
From client: STOR potion.txt

From client: PORT 127,0,0,1,31,145

Client IP: 127.0.0.1, Client Port: 8081
From client: STOR potion.txt

From client: PORT 127,0,0,1,31,145

Client IP: 127.0.0.1, Client Port: 8081
From client: STOR magic_items.csv
```

RETR Command

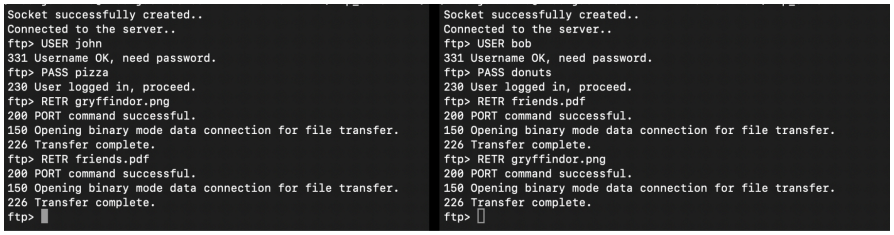
1. **RETR Command to Download a File**
    - Input: RETR filename (after successful USER and PASS)
    - Expected Output: 150 File status okay; about to open data connection., followed by 226 Transfer completed.



2. **RETR Command without Authentication**
    - Input: RETR filename
    - Expected Output: 530 Not logged in.
3. **RETR Command with Non-existent File**
    - Input: RETR nonexistentfile (after successful USER and PASS)
    - Expected Output: 550 No such file or directory.



4. **RETR Command with Multiple Clients**
    - Scenario: Multiple clients download different files simultaneously after successful authentication
    - Expected Output: Each file is downloaded successfully without interference



```
New connection, socket fd is 5, ip is : 127.0.0.1, port : 53024
Adding to list of sockets as 1
From client: USER bob

From client: PASS donuts

From client: PORT 127,0,0,1,31,145

Client IP: 127.0.0.1, Client Port: 8081
From client: RETR friends.pdf

From client: PORT 127,0,0,1,31,145

Client IP: 127.0.0.1, Client Port: 8081
From client: RETR gryffindor.png

New connection, socket fd is 5, ip is : 127.0.0.1, port : 53038
Adding to list of sockets as 1
From client: USER john

From client: PASS pizza

From client: PORT 127,0,0,1,31,145

Client IP: 127.0.0.1, Client Port: 8081
From client: RETR gryffindor.png

From client: PORT 127,0,0,1,31,145

Client IP: 127.0.0.1, Client Port: 8081
From client: RETR friends.pdf
```

CWD Command

1. **CWD Command to Change Directory**
    - Input: CWD foldername (after successful USER and PASS)
    - Expected Output: 200 directory changed to pathname/foldername.
    -
```
Socket successfully created..
Connected to the server..
ftp> USER bob
331 Username OK, need password.
ftp> PASS donuts
230 User logged in, proceed.
ftp> PWD
257 "/Users/abraiz/Downloads/project1_final/server" is the cur
rent directory.
ftp> CWD bob
250 Directory successfully changed.
ftp> PWD
257 "/Users/abraiz/Downloads/project1_final/server/bob" is the
 current directory.
ftp>
```
2. **CWD Command without Authentication**
    - Input: CWD foldername
    - Expected Output: 530 Not logged in.

```
Socket successfully created..
Connected to the server..
ftp> CWD bob
530 Please login with USER and PASS.
ftp>
```

3. **CWD Command with Invalid Directory**
    - Input: CWD nonexistdir (after successful USER and PASS)
    - Expected Output: 550 No such file or directory.

```
Socket successfully created..
Connected to the server..
ftp> USER bob
331 Username OK, need password.
ftp> PASS donuts
230 User logged in, proceed.
ftp> PWD
257 "/Users/abraiz/Downloads/project1_final/server" is the
current directory.
ftp> CWD non-exist
550 Failed to change directory.
ftp>
```

4. **CWD Command with Multiple Clients**
    - Scenario: Multiple clients change directories simultaneously after successful authentication
    - Expected Output: Each client successfully changes to their specified directory

```
Socket successfully created..
Connected to the server..
ftp> USER bob
331 Username OK, need password.
ftp> PASS donuts
230 User logged in, proceed.
ftp> PWD
257 "/Users/abraiz/Downloads/project1_final/server" is the cur
rent directory.
ftp> CWD bob
250 Directory successfully changed.
ftp> PWD
257 "/Users/abraiz/Downloads/project1_final/server/bob" is the
 current directory.
ftp>
```

```
ftp> USER kk
331 Username OK, need password.
ftp> PASS 123
230 User logged in, proceed.
ftp> PWD
257 "/Users/abraiz/Downloads/project1_final/server" is the
current directory.
ftp> CWD kk
250 Directory successfully changed.
ftp> PWD
257 "/Users/abraiz/Downloads/project1_final/server/kk" is t
he current directory.
ftp>
```

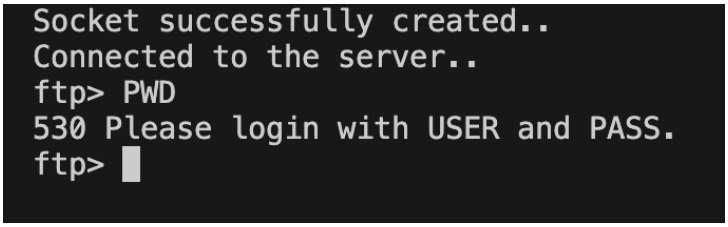
```
ftp_clien...
ftp_clien...
```

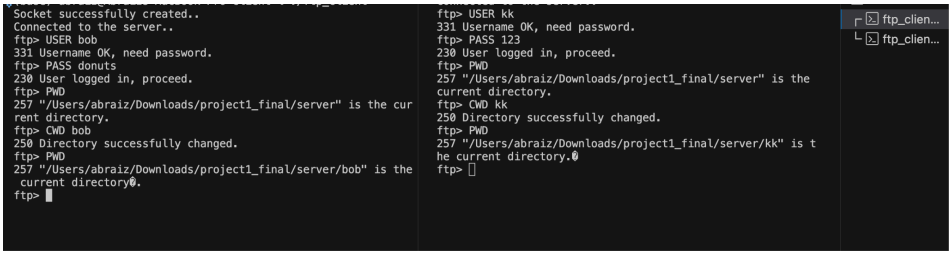
PWD Command

1. **PWD Command to Display Current Directory**
    - Input: PWD (after successful USER and PASS)
    - Expected Output: 257 pathname.

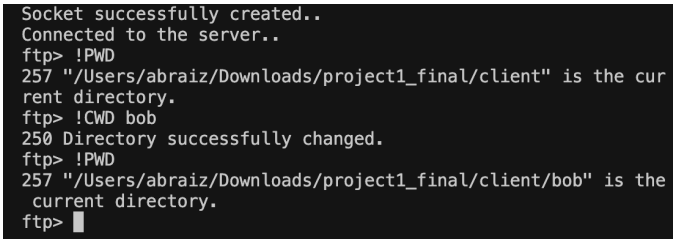
```
Socket successfully created..
Connected to the server..
ftp> USER bob
331 Username OK, need password.
ftp> PASS donuts
230 User logged in, proceed.
ftp> PWD
257 "/Users/abraiz/Downloads/project1_final/server" is the cur
rent directory.
ftp> CWD bob
250 Directory successfully changed.
ftp> PWD
257 "/Users/abraiz/Downloads/project1_final/server/bob" is the
 current directory.
ftp>
```

2. **PWD Command without Authentication**

- Input: PWD
- Expected Output: 530 Not logged in.
- 

3. **PWD Command with Multiple Clients**
   - Scenario: Multiple clients request the current directory simultaneously after successful authentication
   - Expected Output: Each client receives the current directory path
   - 

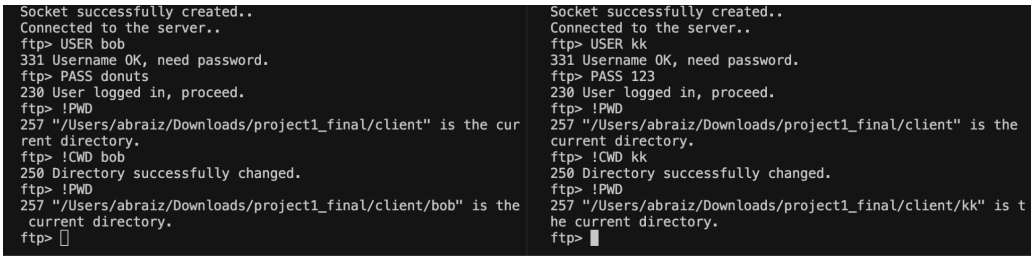
!CWD Command

1. **!CWD Command to Change Local Directory**
   - Input: !CWD localfoldername (after successful USER and PASS)
   - Expected Output: Local client directory changes to localfoldername and shows success message.
   - 

2. **!CWD Command with Invalid Directory**
   - Input: !CWD nonexistdir (after successful USER and PASS)
   - Expected Output: Error message indicating the directory does not exist.

```
Socket successfully created..
Connected to the server..
ftp> USER bob
331 Username OK, need password.
ftp> PASS donuts
230 User logged in, proceed.
ftp> !CWD non-exist
550 Failed to change directory.
ftp>
```

3. **!CWD Command with Multiple Clients**
    - Scenario: Multiple clients change local directories simultaneously after successful authentication
    - Expected Output: Each client's local directory changes to their specified directory without interference

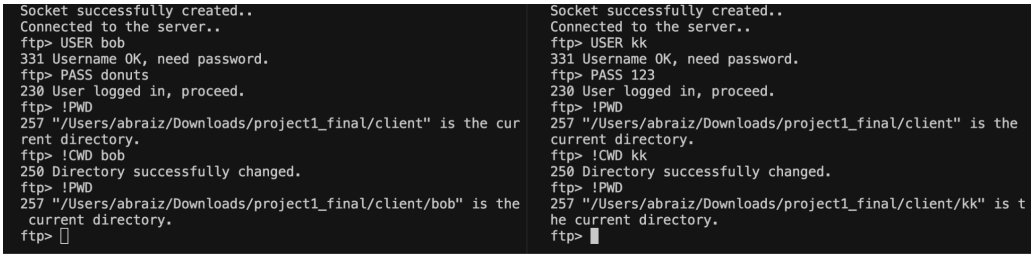


!PWD Command

1. **!PWD Command to Display Local Current Directory**
    - Input: !PWD (after successful USER and PASS)
    - Expected Output: Displays the local client directory path.

```
Socket successfully created..
Connected to the server..
ftp> !PWD
257 "/Users/abraiz/Downloads/project1_final/client" is the cur
rent directory.
ftp> !CWD bob
250 Directory successfully changed.
ftp> !PWD
257 "/Users/abraiz/Downloads/project1_final/client/bob" is the
 current directory.
ftp>
```

2. **!PWD Command without Authentication**
    - Input: !PWD
    - Expected Output: 530 Not logged in.
    -
3. **!PWD Command with Multiple Clients**
    - Scenario: Multiple clients request their local current directory simultaneously after successful authentication
    - Expected Output: Each client receives their local current directory path without interference

- 

QUIT Command

1. **QUIT Command After Successful Authentication**
   - Input: QUIT (after successful USER and PASS)
   - Expected Output: 221 Service closing control connection.

```
Socket successfully created..
Connected to the server..
ftp> USER john
331 Username OK, need password.
ftp> PASS pizza
230 User logged in, proceed.
ftp> RETR gryffindor.png
200 PORT command successful.
150 Opening binary mode data connection for file transfer.
226 Transfer complete.
ftp> RETR friends.pdf
200 PORT command successful.
150 Opening binary mode data connection for file transfer.
226 Transfer complete.
ftp> QUIT
221 Service closing control connection.
Client Exit...
```

2. **QUIT Command Without Authentication**
   - Input: QUIT
   - Expected Output: 221 Service closing control connection.

```
Socket successfully created..
Connected to the server..
ftp> QUIT
221 Service closing control connection.
Client Exit...
```

3. **QUIT Command with Multiple Clients**
   - Scenario: One client issues QUIT while other clients continue operations
   - Expected Output: The server closes the connection for the quitting client with 221 Service closing control connection. while other clients remain unaffected and continue their operations.

```
Socket successfully created..
Connected to the server..
ftp> USER john
331 Username OK, need password.
ftp> PASS pizza
230 User logged in, proceed.
ftp> RETR gryffindor.png
200 PORT command successful.
150 Opening binary mode data connection for file transfer.
226 Transfer complete.
ftp> RETR friends.pdf
200 PORT command successful.
150 Opening binary mode data connection for file transfer.
226 Transfer complete.
ftp> QUIT
221 Service closing control connection.
Client Exit...
```

```
Socket successfully created..
Connected to the server..
ftp> USER bob
331 Username OK, need password.
ftp> PASS donuts
230 User logged in, proceed.
ftp> RETR friends.pdf
200 PORT command successful.
150 Opening binary mode data connection for file transfer.
226 Transfer complete.
ftp> RETR gryffindor.png
200 PORT command successful.
150 Opening binary mode data connection for file transfer.
226 Transfer complete.
ftp> QUIT
221 Service closing control connection.
Client Exit...
```