Course COMP-8567
Project
Fall 2023
Due Date: Dec/13/2023
100 Marks

- The project work can be carried out alone or in teams of two students.
- Only students from the same section can form a team.
- In case of a team, each team member is expected to contribute evenly (in reasonable terms) towards the development of the project.
- Along with the file submission, the working of the project must be demonstrated during the scheduled slot (TBA) which will be followed by a **viva**.
  - In case of a team, the working of the project must be demonstrated individually by team members as per the stipulated schedule.
  - Demo slots can be scheduled anytime on Dec 14th ,15th and 16th and will be announced suitably ahead of time.

Introduction

In this client-server project, a client can request a file or a set of files from the server. The server searches for the file/s in its file directory rooted at its ~ and returns the file/files requested to the client (or an appropriate message otherwise). Multiple clients can connect to the server from different machines and can request file/s as per the commands listed in section 2

- **The server, the mirror and the client processes** must run on different machines and must communicate using sockets only.

Section 1 (Server)

- The **server** and an identical copy of the server called the **mirror** [see section 3] must both run before any of the client (s) run and both of them must wait for request/s from client/s
- Upon receiving a connection request from a client, the server forks a child process that services the client request exclusively in a function called pclientrequest() and (the server) returns to listening to requests from other clients.
  - The pclientrequest() function enters an infinite loop waiting for the client to send a command

    - Upon the receipt of a command from the client, pclientrequest() performs the action required to process the command as per the requirements listed in section 2 and returns the result to the client
- Upon the receipt of **quitc** from the client, pclientrequest() exits.
- **Note:** for each client request, the server must fork a separate process with the pclientrequest() function to service the request and then go back to listening to requests from other clients

Section 2 (Client)

The client process runs an infinite loop waiting for the user to enter one of the commands.

Note: The commands **are not** Linux commands and are defined(in this project) to denote the action to be performed by the server.

Once the command is entered, the client verifies the **syntax of the command** and if it is okay, sends the command to the server, else it prints an appropriate error message.

List of Client Commands:

- **getfn** *filename*
    - If the file *filename* is found in its file directory tree rooted at ~, the server must return **the filename, size(in bytes), date created and file permissions** to the client and the client prints the received information on its terminal.
        - Note: if the file with the same name exists in multiple folders in the directory tree rooted at ~, the server sends information pertaining to the first successful search/match of *filename*
        - Else the client prints "File not found"
    - **Ex: Client$ getfs sample.txt**

- **getfz** *size1 size2*
    - The server must return to the client **temp.tar.gz** that contains all the files in the directory tree rooted at its ~ whose file-size in bytes is >=*size1* and <=*size2*
        - *size1 < = size2 (size1>= 0 and size2>=0)*
    - If none of the files of the specified size are present, the server sends "No file found" to the client (which is then printed on the client terminal by the client)
    - **Ex: Client$ getfz 1240 12450**

- **getft** <extension list>   //up to 3 different file types
    - the server must return temp.tar.gz that contains all the files in its directory tree rooted at ~ belonging to the file type/s listed in the extension list, else the server sends the message "No file found" to the client (which is printed on the client terminal by the client)
    - The extension list **must have at least one file type** and can have up to 3 different file types
    - **Ex: Client$ getft c txt**
    - **Client$ getft jpg bmp pdf**

- **getfdb** *date*
    - The server must return to the client temp.tar.gz that contains all the files in the directory tree rooted at ~ whose date of creation is <=*date*
    - **Ex: Client$ getfdb 2023-01-01**

- **getfda** *date*
    - The server must return to the client temp.tar.gz that contains all the files in the directory tree rooted at ~ whose date of creation is >=*date*
    - **Ex: Client$ getfda 2023-03-31**

- **quitc** The command is transferred to the server and the client process is terminated

Note: All files returned from the server must be stored in a folder named **f23project** in the home directory of the client.

Note:

- It is the responsibility of the client process to **verify** the syntax of the command entered by the user (as per the rules in Section 3) before processing it.
    - Appropriate messages must be printed when the syntax of the command is incorrect.

Section 3 Alternating Between the Server and the Mirror

- The server and the mirror (the server's copy possibly with a few additions/changes) are to run on two different machines/terminals.
- The first 4 client connections are to be handled by the server.
- The next 4 client connections are to be handled by the mirror.
- The remaining client connections are to be handled by the server and the mirror in an alternating manner- (ex: connection 9 is to be handled by the server, connection 10 by the mirror, and so on)

Submission:

- **Turnitin similarity report will be enabled for all 4 sections, and you will be able to access the report after you submit the files.**

You are required to **submit 6 files** with **adequate and pertinent comments** briefly explaining/describing various parts of the programs.

1. server.c
2. server.txt
3. mirror.c
4. mirror.txt
5. client.c
6. client.txt