Course COMP-8567
Project: Distributed File System (using socket programming)
Summer 2024
Due Date: Aug/13/202, 11PM EDT
100 Marks
Plagiarism Detection Tool: MOSS

- 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 **Aug 14th ,15th and 16th** and will be announced suitably ahead of time.

Introduction

In this project, you are required to implement to a **distributed file system** through socket programming.

The distributed file system has three servers:

- **Smain**
- **Spdf**
- **Stext**

and can support multiple client connections.

Section A – Servers : Smain, Spdf and Stext

Clients are allowed to upload/store three file types (.c,.pdf and .txt) onto **Smain**, however **Smain** only stores .c files locally and **transfers** all .**pdf files** to the **Spdf sever** and **.txt files** to the **Stext sever** (in the background). Clients are not aware of this operation and assume all files are stored at **Smain**.

All clients communicate with **Smain only** and are **not aware** of the presence of **Spdf** and **Stext**.

- Upon receiving a connection request from a client, **Smain** forks a child process that services the client request exclusively in a function called prcclient() and (**Smain**) returns to listening to requests from other clients.
    - The prcclient() function enters an infinite loop waiting for the client to send a command
    - Upon the receipt of a command from the client, prclient() performs the action required to process the command as per the requirements listed **in section B** and returns the result to the client
- **Spdf** and **Stext** act as servers to **Smain** and service its requests based on the commands entered in **client24s (Section B)**

Note:

- The servers **Smain, Spdf, Stext** and client24s process/es must run on different machines/terminals and must communicate using sockets only.
- Files in **Smain** must be saved under ~/smain //lowercase 's'
- Files in **Stext** must be saved under ~/stext //lowercase 's'
- Files in **Spdf** must be saved under ~/spdf //lowercase 's'

Section B (client24s)

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 **Smain**.

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

Client Commands : (5 commands)

ufile *filename destination_path*

Transfers (uploads) *filename* from the PWD of the client to smain

- filename : valid filename (.c /.pdf/ .txt) in client's PWD
- *destination_path*: A path **in Smain** //must belong to ~/smain of the main server
    - if destination path is not already present in the main server, it must be newly created

- Only .c files are stored in the main server (but the user is not aware of it)
- **.txt files are transferred from Smain to Stext** and are stored in the corresponding folders in the Stext server (replace Smain with Stext)
- **.pdf files are transferred from Smail to Spdf** and are stored in the corresponding folders in the Spdf server (replace Smain with Spdf)

Examples:

- **client24s$ ufile sample.c ~smain/folder1/folder2** //should store **sample.c** in the specified folder on the **Smain** server
- **client24s$ ufile sample.txt ~smain/folder1/folder2** // **Smain** transfers sample.txt to the **Stext** server and the **Stext** server in turn stores **sample.txt** in **~Stext/folder1/folder2** //User assumes sample.txt is stored in **Smain**, but all text files must actually be stored in the **Stext** server in the corresponding path (replace ~smain with ~stext)
- **client24s$ ufile sample.pdf ~smain/folder1/folder2** // **Smain** transfers sample.pdf to the **Spdf** server and the **Spdf** server in turn stores **sample.pdf** in **~spdf/folder1/folder2** //User assumes sample.pdf is stored in **Smain,** but all pdf files must actually be stored in the **Spdf** server in the corresponding path (replace ~smain with ~spdf)
- **Note: Clients can directly communicate with Smain only and are not aware of the presence of Spdf and Stext servers**

dfile filename

Transfers (downloads) *filename* from Smain to the PWD of the client

- filename : valid path of a file in **Smain** (.c /.pdf/ .txt files only)
  - If the request is for a .c file, **Smain** processes the request (locally) and sends the corresponding file to the client
  - If the request is for a .txt file, **Smain** obtains the file from **Stext** and then sends the corresponding file to the client
  - If the request is for a .pdf file, **Smain** obtains the file from **Spdf** and then sends the corresponding file to the client

Examples:

- **client24s$ dfile ~smain/folder1/folder2/sample.c** // **Smain** processes the request (locally) and sends sample.c to the client
- **client24s$ dfile ~smain/folder1/folder2/sample.pdf** // **Smain** obtains sample.pdf from the corresponding directory in **Spdf** and then sends sample.pdf to the client
- **client24s$ dfile ~smain/folder1/folder2/sample.txt** // **Smain** obtains sample.txt from the corresponding directory in **Stext** and then sends sample.txt to the client

rmfile filename

Removes (deletes) *filename* from Smain to the PWD of the client

- filename : valid path of a file in **Smain** (.c /.pdf/ .txt files only)
  - If the request is for a .c file, **Smain** processes the request (locally) and deletes the corresponding file
  - If the request is for a .txt file, **Smain** sends a request to **Stext** to delete the text file in the corresponding directory.
  - If the request is for a .txt file, **Smain** sends a request to **Stext** to delete the text file in the corresponding directory.

Example:

client24s$ rmfile ~smain/folder1/folder2/sample.pdf // **Smain** requests **Spdf** to delete sample.pdf in the corresponding directory

dtar filetype

Creates a tar file of the specified file type and transfers (downloads) the tar file from Smain to the PWD of the client

- **Filetype: .c/.txt/.pdf**
  - If the filetype is .c , **Smain** creates a **tar file (cfiles.tar) of all .c files present in the directory subtree** rooted at ~/smain and sends the tar file to the client

- If the filetype is .pdf , **Smain** requests and obtains pdf.tar **of all .pdf files present in the directory subtree** rooted at ~/spdf from the **Spdf** server and sends pdf.tar to the client
- If the filetype is .txt , **Smain** requests and obtains text.tar **of all .txt files present in the directory subtree** rooted at ~/stext from the **Stext** server and sends pdf.tar to the client

display *pathname*

Transfers (downloads) *filename* from Smain to the PWD of the client

- pathname : valid path of a directory in **Smain** that belongs to ~/smain
  - **Smain** obtains the list of all .pdf and .txt files (if any) from the corresponding directories in **Spdf** and **Stxt.**
  - **Smain** then combines the list obtained in the previous step with the list of .c files present locally in *pathname* and transfers the consolidated list of .c,.pdf and .txt files (in that order) to the client
    - //Please Note: only the names of files are transferred to the client and not the actual files

Submission Instructions:

- Comments must be included to explain the working of the program
- The program must reasonably handle error conditions based on the requirements

Plagiarism Detection Tool: MOSS

You are required to **submit 4 files**.

1. Smain.c
2. Spdf.c
3. Stext.c
4. client24s.c