CS 6378: Advanced Operating Systems
Programming Assignment 1

Instructor: Ravi Prakash

Assigned on: June 13, 2023
Due date: June 27, 2023

This is an individual project. Sharing of code among students or using fragments of code written by others is strictly prohibited, and will be dealt with as per the university's rules governing academic misconduct. You are expected to demonstrate the operation of your project to the instructor or the TA.

Requirements

1. Source code must be in the C/C++/Java programming language.
2. The program must run on UTD lab machines (`dc01, dc02, ..., dc45`).
3. You will need to know thread and socket programming and its APIs for the language you choose. It can be assumed that each process (server/client) is running on its own machine (`dcXY`), where $01 \leq XY \leq 45$, and two processes communicate through a reliable socket connection between them. Please get familiar with basic UNIX commands to run your program on *dcXY* and UNIX/Linux system calls for directory and file operations.

Project Description

In this project, you will use socket connections to emulate communication between three network-connected computers, say C_1, C_2 and C_3, and implement the following:

- Process P_1, running on C_1, has access to a text file, F_1, of size 300 bytes. Process P_2, running on C_2, has access to a text file, F_2, also of size 300 bytes. The contents of these two text files are different. Process P_3 is running on C_3.
- P_3 operates as the client, while P_1 and P_2, act as servers. P_3 establishes a stream socket connection with P_1, and another stream socket connection with P_2.
- Once the connections are established, P_3 does the following (these are the project requirements):
  - P_3 shows a message in its terminal window asking the user to input a file name, and waits for the user to type in a file name.
  - Once P_3 has obtained a file name, it sends a message to P_1 asking if P_1 has a file of that name.
  - If P_1 has a file by that name, it replies with a YES message. Otherwise, it replies with a NO message.
  - If P_3 receives a YES message from P_1:
    - P_3 sends a request message to P_1 to send the contents of the named file.
    - On receiving this request, P_1 should send the contents of the file to P_3 using three messages, each containing 100 bytes of F_1.
    - P_3 should append the information received from P_1 into a newly created file with the name provided by the user.

- If P_3 receives a NO message from P_1, then it sends a message to message to P_2 asking if P_2 has the file of that name.
- The subsequent interaction between P_3 and P_2 is similar to what has been described for P_3 and P_1 earlier.
- If neither P_1, nor P_2 has the file, then P_3 should output "File transfer failure" on its terminal. Otherwise, it should output "File successfully fetched from $< server\_name >$" on its terminal.
- After this is done, the socket connections should be terminated and the corresponding processes should exit.

To execute your code, consider the following steps:

1. Establish a VPN connection with UTD, and then log into three of the *dcXY* machines mentioned above. One of them will act as C_1, second as C_2 and the third as as C_3. Ideally, you would have three separate terminal windows, one in which you are logged into C_1, another in which you are logged into C_2 and a third in which you are logged into C_3. You should have access to your UTD home directory on these machines.
2. Launch socket server program that you will write as part of this project on C_1 and C_2, executing on these machines as server processes P_1 and P_2, respectively, listening for incoming connection requests.
3. Next, launch the socket client program, also written by you as part of this project, on C_3, running as process P_3.
4. Have the client establish reliable socket connections (TCP) with the servers.
5. Once the connection is established, implement the project requirements stated above.

Your code should run for the following scenarios:

1. When the user inputs file name F_1, that file is fetched by P_3 from P_1.
2. When the user inputs file name F_2, the search at P_1 is unsuccessful, and then it is successfully fetched by P_3 from P_2.
3. When the user inputs some other file name, the file is not found at P_1 or P_2, and a "File transfer failure" message is shown to the user.

Submission Information

The submission should be through eLearning in the form of an archive consisting of:

1. File(s) containing the source code.
2. The makefile used for compilation purposes, or other instructions to compile the code for the relevant development environment.
3. The directories and files you used to test the execution of your code.

Please do "make clean" before submitting the contents of your directory. This will remove the executable and object code that is not needed for submission.

Your source code must have the following, otherwise you will lose points:

1. Proper comments indicating what is being done
2. Error checking for all function and system calls