CS 6378: Advanced Operating Systems
Programming Assignment 1

Instructor: Ravi Prakash

Assigned on: February 3, 2023
Due date: February 14, 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, running on C_3 has access to an empty text file, $F3$, in the beginning.
- P_3 operates as the server. P_1 and P_2, acting as clients, establish stream socket connection with P_3.
- Once the connections are established, the two processes do the following (these are the project requirements):
  - P_1 should send the original contents of F_1 to P_3 using three messages, each containing 100 bytes of F_1.
  - Similarly, P_2 should send the original contents of F_2 to P_3 using three messages, each containing 100 bytes of F_2.
  - P_3 should first append the information received from P_1, and then the information received from P_2 to the end of F_3.
  - You need to decide how to implement these file append steps in your project.
  - You also need to decide how the two client processes will communicate to the server that they have no more data to send.
  - When the clients are done sending their data to the server, the file F_3 will be of size 600 bytes and its contents will be the contents of F_1 followed by the contents of F_2.
  - Then, P_3 will send the contents of F_3 to both P_1 and P_2 along the socket connections. This will be done using multiple messages, each containing 200 bytes.

- When such information is communicated, both P_1 and P_2 will have identical files F_3 in their file systems.
- After this is done, the socket connections should be terminated and the corresponding processes should exit.

As you do not have access to three different file systems on three different machines, you will achieve the desired outcome as follows:

1. First create a subdirectory named $D1$, or some other name that isn't already taken, in your UTD home directory. Within that directory create three directories, one each for the server and two clients. In the subdirectory for the server create an empty file F_3, and in the subdirectories for the two clients create corresponding files (F_1 and F_2) of the desired size and contents.
2. 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.
3. Launch a socket server program that you will write as part of this project on C_3. Now, P_3 will be listening for incoming connection requests.
4. Next, launch the socket client program, also written by you as part of this project, on C_1 and C_2.
5. Have the clients establish a reliable socket connections (TCP) with the server running on C_3.
6. Once the connection is established, implement the project requirements stated above.

When your processes terminate at all three machines, in addition to the original F_1 and F_2 files, you should have three identical files F_3 in the corresponding subdirectories of the server and two clients.

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.
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