CS 6378: Advanced Operating Systems
Programming Assignment 1

Instructor: Ravi Prakash

Assigned on: January 29, 2022
Due date: February 8, 2022

This is an individual project and sharing of code 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`) 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 a socket connection to emulate communication between two network-connected computers, say C_1 and C_2, 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.
- Either one of P_1 or P_2 initiates a socket connection with the other. The choice is yours.
- Once the connection is established, the two processes do the following (these are the project requirements):
  - P_1 should send the original contents of F_1 to P_2 using four messages with the first message containing the first 75 bytes of F_1, the next containing the next 75 bytes of F_1 and so on.
  - Similarly, P_2 should send the original contents of F_2 to P_1 using three messages with the first message containing the first 100 bytes of F_1, the next containing the next 100 bytes of F_1 and so on.
  - P_1 should append the information received from P_2 to the end of F_1.
  - P_2 should insert the information received from P_1 in the beginning of F_2.
  - You need to decide how to implement these file append and/or insertion steps in your project.
  - You also need to implement how the two processes will communicate to each other that they have no more data to send. When such information is communicated, the socket connection should be terminated and the corresponding processes should exit.
- At the end of the steps described above, files F_1 and F_2 should be identical and of size 600 bytes.

As you do not have access to two different file systems on two 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 the two files F_1 and F_2 of the desired size and contents.

2. Establish a VPN connection with UTD, and then log into two of the *dcXY* machines mentioned above. One of them will act as C_1 and another as C_2. Ideally, you would have two separate terminal windows, one in which you are logged into C_1 and another in which you are logged into C_2. 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_1. Now, C_1 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_2.

5. Have the client running on C_2 establish a reliable socket connection (TCP) with the server running on C_1.

6. Once the connection is established, implement the project requirements stated above.

When your processes terminate at both the machines, you should have two identical files F_1 and F_2 in the subdirectory $D1$ of your UTD home directory.

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