CS 6378: Advanced Operating Systems
Programming Assignment 1

Instructor: Ravi Prakash

Assigned on: February 5, 2021
Due date: February 14, 2021

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 a single machine (`dcXY`). 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:

- C_2 has a local directory D_1 with two text files F_1 and F_2.
- C_1 has no such directory.
- C_2 communicates with C_1 to make a copy of D_1 and all its contents at C_1. As a consequence, now there are two replicas of directory D_1 on different machines.

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

1. First create a subdirectory named $D1$ in your UTD home directory.
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, have C_2 send a message to C_1 to create a directory $D1copy$.

7. On receiving the request from C_2, the server running on C_1 should create a subdirectory $D1copy$ in the home directory and send an acknowledgment to C_2.

8. Next C_2 should send a message to C_1 to create the first file named F_1.

9. C_1 must create a file named F_1 in subdirectory $D1copy$ and acknowledge the successful creation of the file.

10. C_2 should read and send contents of file F_1 to C_2 in successive messages of size 256 bytes (or remaining size of the file when you get near the end of the file).

11. C_1 should append the file contents received from C_2 into file F_1 in subdirectory $D1copy$.

12. When all contents of file F_1 have been successfully sent by C_2 and copied by C_1, C_2 should send a message indicating end of file, and C_1 should respond with a message indicating successful creation of the file in subdirectory $D1copy$.

13. Repeat the steps described above to also copy file F_2 from subdirectory $D1$ to $D1copy$.

14. Have C_2 send a message indicating *end of session* to C_1, close the socket connection and terminate.

15. Have C_1 also close the connection and terminate on receiving the *end of session* message from C_2.

When your processes terminate at both the machines, you should have two subdirectories in your UTD home directory, names $D1$ and $D1copy$ with identical contents.

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