Lab 1 Assignment (Max 40 points)

(Due: Wednesday by 11:59 pm, October 7, 2020)

1. Goal

We will learn the basic concurrent File Transfer Services that consists of a client and server. We will use the TCP protocol for this lab.

2. Project Description: SFTS (Simple File Transfer Services)

In this programming assignment, you are to build a simple file copy service that consists of a client and server. **The server and client program MUST run on separate machines**.

2.1 Server

- A server does NOT have to handle multiple clients concurrently, but it must handle multiple clients consecutively.
- The server will be started on a machine in a directory. That directory will be considered the HOME directory for the server.
- A server should support the following requests from clients:

  1. Show files at the server's current directory: '**catalog**'
     - Synopsis: *catalog*
       - Display the all file names under current directory
     - A list of files should be sent from the server to the client and displayed on the client's screen.
     - Hint: it is similar to 'ls' Unix/Linux service

  2. Download files: '**download**'
     - Synopsis: *download source-filename*
       - When a server gets "download" request with "source-filename", it checks first whether it has the requested file name under its current directory or not. If the server does not have the file, it sends an error message back to the client. Otherwise, it opens the requested file and sends it ("download") to the client.

  3. Upload files: '**upload**'
     - Synopsis: *upload source-filename*

    - When a server gets an “upload” request, it checks first whether it has already the same name of the requested file (dest-filename) in its current directory. If the server has already the same name of file which was sent with “upload” request from a client, it performs “upload” service and sends the following warning message to the client for display: “‘dest-filename’ file has been overwritten”. Otherwise, it receives the file from the client and creates “dest-filename” at the current directory.

4. Terminating the server: “**bye**”
    - Synopsis: *bye [no-argument]*
    - When a server receives “bye” request from a client, it terminates itself with display message “File copy server is down!” and gracefully disconnect the connection with the client.

- A server should be able to handle exceptions (errors) for each command.
- Any error message occurred at the server should be informative and sent to and displayed to the client.

2.2 Client

- A client makes a connection to the server when it tries to download or upload files. You can specific the IP address for the server.
- A client requests following services to the server:
    - **catalog, ls, download, upload, bye.**
    - Synopsis of each services refer to the description of the server

- A client also supports following requests from the user:

1. Show files at the server’s current directory: ‘**catalog**’
    - Synopsis: *catalog*
        - Get list the all file names under current directory from the server and display the received files.

2. Download files: ‘**download**’
    - Synopsis: *download source-filename dest-filename*
        - When a client gets “download” request, it sends its request to the server with two arguments (source-filename and dest-filename). If the client receives an error message from the server, it displays an error message to the user without performing “download” service. Otherwise, it receives the file from the server and copies to the dest-name. If the dest-name file is

already exists, the client should display the following message after the finish the 'download' request: "'dest-filename" has been overwritten".

3. Upload files: '**upload**'
   - Synopsis: *upload source-filename dest-filename*
     - When a client gets "upload" request, it checks first whether it has the requested file(source-filename) under its current directory or not. If the client does not have the request file under current directory, it displays "error" message back to the user without performing "upload" service. Otherwise, it sends its request to the connected server and performs "upload" operation. If the client receives the message about the overwriting, it displays the received the message.

4. Terminating the server: "**bye**"
   - Synopsis: *bye*
     - A client sends "bye" request to the connected server and terminates itself with display message "Internet copy client is down!".

- A client should be able to handle exceptions (errors) for each command.

Hint: you don't have to write entire code for the 'catalog' service. Your program (server or client) needs to just use (or call) a Unix/Linux service that provides the same result as your program is expected to produce such as 'ls'.

2.3 Programming environment

- All programs have to be written C or C++ and run on UNIX like platform.
- All connections between a server and clients should be TCP/IP socket.

2.4 Required Skills

Everyone is expected to know following skills and knowledge in order to complete this programming assignment.

- TCP/IP Socket programming
- Understanding UNIX like Operating System
- Creating/invoking processes in UNIX like environment
- Makefile
- C or C++

3. Deliverables

The deliverables for this assignment include the following files:

- Written C or C++ Code for Server and Client program
- Makefile
- Readme: A short description of your programs includes: the names of created executable images which will be created after run your makefile, how to run your programs.

4. Submission

Please do the followings when you submit your programming assignment.

- Create a zip file that contains your written source code, makefile and readme. DO NOT INCLUDE EXECUTABLES AND COMPILED OBJECT FILES.
- **Upload it to the class Canvas by deadline.**

5. Grading

The maximum point for the assignment is 40. This programming assignment will be graded by following criteria.

- Completeness: 40 points
  - Connection :
    - 10 points – if your server and client get connected over TCP network: 15 points
    - If a server and client do not work over TCP, but they work within a machine: 5

- Concurrent Server:
  - 6: if your server handle at least 3 clients consecutively
  - 0: Otherwise
- Completeness of the server: 12 points
  - Catalog, bye are worth 2 points each
  - “download” and “upload” service is 4 points each
- Completeness of the client: 12 points
  - Each service (bye, catalog) is worth 2 points
  - “download” and “upload” service are 4 points each

6. IMPORTANT

- Your program will be tested on csegrid.ucdenver.pvt machine (linux based) with another linux based machine. Please make sure your program runs without problem on both platform and test your program before submit it. If for some reason you can’t get it to work on those systems, but it works on YOUR systems, we will figure out a way to grade it. BUT it must work on 2 machines.