Lab2 Assignment (Max 40 points)

(Due: Monday by 2 pm, April 8, 2019)

1. Goal

We will learn the basic concurrent File Transfer Services that consists of a client and server. In addition to, we will learn how to use separate processes to allow a server to handle many clients concurrently

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 should run on separate machines**.

2.1 Server

- A server should be able to spawn a process (or create thread) to handle each incoming connection from clients and let a client download or upload requested files.
- A server MUST support **at least 3 clients concurrently**.
- A server should support the following request from clients:

1. Show files at the server's current directory: '**catalog**'
   - Synopsis: *catalog*
     - Display the all file names under current directory
   - Execution result should be displayed to the request client's screen.
   - Hint: it is similar to 'ls' Unix/Linux service

2. Download files: '**download**'
   - Synopsis: *download source-filename dest-filename*
     - When a server gets "download" request with "source-filename", it checks first whether it has the request file name under its current directory or not. If the server has not the request name of file which was sent with "download" request from a client, it sends an error message back to the client without performing "download" service. Otherwise, it opens the request file and performs "download" operation.

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

- When a server gets “upload” request, it checks first whether it has already the same name of the requested file (dest-filename) under its current directory or not. 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 be overwritten”. Otherwise, it receives the file from the client and creates “dest-filename” at the current directory.

4. Display current directory of the server: ‘**spwd**’
    - Synopsis: *spwd [no-argument]*
    - If the server receives ‘spwd’ request from a client, the server perform ‘pwd. The execution result should be displayed to the request client’s screen.

5. 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 informed and displayed to the request client.

2.2 Client

- A client makes a connection to the server when it tries to download or upload files.
- A client requests following services to the server:
    - **catalog, ls, download, upload, pwd, spwd, 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. Show files at the client’s current directory: ‘**ls**’
    - Synopsis: *ls [options] … [File]*..

- Display the all file names under current directory of client.
- Refer 'Unix/Linux "ls" manual for detail options.

3. 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".

4. 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.

5. Display of current directory: "**pwd**"
   - Synopsis: *pwd [no-argument]*
   - Display current directory information

6. 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 'catalog','ls','spwd' and 'pwd' services. 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', and 'pwd'.

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.
- Please use the following convention when you create zipped file
  - First 3 letters of your last name + last 4 digits of your student ID
  - e.g.: If a student name is "Bill Clinton" and his ID is 999-34-5678, then his zipped file name is "cli5678.zip".
- If the zipped file is ready, **upload it to the class Canvas byMonday 2 pm, April 8, 2019.**

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: 10 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 currently
  - 0: Otherwise
- Completeness of the server: 12 points
  - Each service (catalog, bye, spwd) is worth 2 points
  - "download" and "upload" service is 3 points each
- Completeness of the client: 12 points
  - Each service (bye, ls, pwd) is worth 2 points
  - "download" and "upload" service is 3 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.