# Programming Assignment 1

**Instructions:**

- **Due by 11:59pm, Monday, Feb. 19, 2023**
- Late penalty: 10% penalty for each day late.
- This is an individual assignment. Although you may work in groups to brainstorm on possible
  solutions, your code implementation, report, and evaluation must be your own work.
- Upload your assignment on the Blackboard with the following name:
- Section_LastName_FirstName_PA1.zip
- Please do NOT email your assignment to the instructor or TA!
- Use any popular language, if in doubt ask the TA ASAP.

## Overview

This assignment is a client-server mode file downloading system. It involves the creation of a client and server program. The server hosts a list of files for the client to download. The client will get the file list from the server and then request for downloading one or more files of them from the server. Scale the implementation of client to evaluate the performance of the server.

## Detailed Description of Assignment

The server hosts a list of files in its directory, these files will be requested and downloaded by a client. The server listens for the client on a specific port.
The server performs at least the following tasks:

- Listens for connections from a port which can be specified either by:
  - Configuration file
  - Passed Arguments
- Hosts a list of files in its directory:
  - Users could add, modify, or delete the files in its directory when the server is running.

- When receiving a get_files_list request from a client, it must send the list of all files it hosts to the client
- When receiving a download request for downloading a file from a client, it will send the file content to the client if the file exists. Otherwise, it will send an error to the client.
- The server should have an automatic update mechanism. If registered files are modified or deleted, the effect should be reflected to the server in time.

For

example, if a file is deleted on the disk, the server should be notified in time and remove the corresponding item from the file list.

- Server output: The server maintains a log of the clients that connected to it, the files it sends to which client, the amount of data sent and the time it took to transfer the file and possible errors encountered.

The Client performs at least the following tasks:

- It connects to a server using an IP and a port, specified either by
    - Configuration file
    - Passed Arguments
- Get the file list from the Server through the get files list request.
- Downloading specific files from the Server, there are two requirements:
    - The user can select one or multiple files to download.
    - If user select multiple files to download, it could choose to download them serially or parallelly through user's input.

- Check if the downloaded file is same as the original one in the server, such as MD5. If they are different, you need to take appropriate methods to deal with this problem.
- Check if there is some error or exception when downloading the file, you need to take appropriate to deal with error or exception.
    - Redownload n times, you could set the number n in the Configuration file or through user's input
    - Or any other method to deal with the problem.
- Client output: The client can either store to a log or just print to the console the files selected, the files downloaded, time taken to download each file and the total download time, and any possible errors it may have encountered like file integrity failures, file non-exist etc.

Additional Requirements

- The server should be able to accept multiple client requests simultaneously. You may achieve this using threads. You may also use other means to achieve this goal.
- Simple command line interfaces are fine.
- The folder being watched by the server and the folder the client downloads its files need to be different and can be specified either by:
    - Configuration file
    - Passed Arguments

## Evaluation

1. Start by connecting one client to the server. Ensure you can transfer one file properly and both client and server could support its respective requirements aforementioned.

2. Scale to 4 clients connected to the server and make sure they could download the specified files simultaneously.

3. Measuring how the transfer speed changes when varying the number of concurrent clients (N). All the clients are requested to download files from the server concurrently. Each client should create at least 10 threads downing a fixed number (set by students) of files simultaneously. and each client needs to repeat the experiments for a fixed number (set by students).
    a. N can be 2, 4, 8, 16
    b. Graph your results

4. Measuring how the transfer speed changes when varying the download file size.
    a. Different file sizes are 128, 512, 2k, 8k, 32k
    b. For each size of file in a), performing the following test
        • 4 clients concurrently request to download files from the server. Each client creates at least 10 threads to download a fixed number (set by students) of files simultaneously, and each client needs to repeat the experiments for a fixed number (set by students).
    c. Graph your results

## Submission Information

When you have finished implementing the complete assignment as described above, you should submit your solution to the blackboard.
Each program must work correctly and be well documented. You should hand in:

1. **Source Code** (25 points): You must hand in all your source code, including with in-Line documentation.

2. **Makefile/Ant/requirements**(5 points): You must use Makefile or Ant to automate your programming assignment compilation or a requirements file to download any dependencies. If using external libraries (mainly for compiled languages) put them in a folder marked external. Note: The external library cannot do the heavy lifting
of
the assignment tasks for you. If in doubt reach out to the TA.

3. **Deployment scripts** (10 points): You must provide a server deployment and the different number of client deployment scripts.

4. **Readme** (10 points): A detailed manual describing how the program works. The manual should be able to instruct users other than the developer to run the program step by step.
5. **Compiles Correctly** (10 points): Your code must be compiled in a Linux environment.
6. **Output files & Performance Evaluation results** (25 points): A copy of the output generated on running your programs for each of the evaluations. For (3) & (4) in evals, provide the output for each level scaled. Write your conclusion along with the graph.
7. **Design Doc** (15 points): You must write separate document to describe about how your program was designed, what tradeoffs you made, etc. Also describe possible improvements and extensions to your program (and sketch how they might be made).
8. Please structure your assignment root folder as follows:
    a. Code: for your source code and make files and deployment scripts. Your file structure in here is up to you but please make sure it is clean.
    b. Docs: for all written documents, report, readme, design doc, etc.
    c. Out: for all output files from your server and client, named indicating which specific evaluation.
    d. Misc: for other files not mentioned specifically.
9. Please put all of the above into **one** .zip file, and upload it to the blackboard. The name of .zip should follow this.
   **format:"Section_LastName_FirstName_PA1.zip"**

## Submission checklist:

- Source code
- Makefile or equivalent specified above
- Deployment scripts
- Readme
- Your Evaluation output
- The output files of your server and client programs as specified above
- A Design Document