**CNT 4007 Computer Network Fundamentals**

**Project 2 Individual Project**

**Implementation of FTP client and server with multiple threads**

1. **Description**

In this project, you will implement a simple version of ftp client/server software. It consists of two programs: client and server. First, server is started on a computer. It listens on a certain TCP port (5106) and **is capable of supporting multiple clients with different threads**. Then, **two clients** are executed on the same computer where the server runs; the server’s port number are supplied in the command line, for example, “client 5106”. The user can issue a command at each of the two clients: “get <filename>”, which is to retrieve a file from the server, or “upload < filename>”, which is to upload a file to the server. Because the file could be arbitrarily large, you are required to split the file into chunks of 1K bytes and use a loop to send the chunks, each time one chunk.

Note 1: If port 5106 is used by another program, for example, you have an earlier run of server which you forget to kill, it will prevent the server’s socket from being created. You will need to kill that program before running your server again.

Note 2: In your code, after each TCP send, e.g., writeObject(msg), you may want to follow up with a flush(), which ensures that msg are sent immediately.

Note 3: The sender needs to inform the receiver how many chunks will be sent, so that the receiver knows where the end is.

We will test your program using a script with the following steps:

1. run “server” in a window for server
2. run “client 5106” in a window for client 1
3. run “client 5106” again in a window for client 2
4. In client 1 window, “get downloadTestFile1.pptx” from the server and write the file to the local disk as newdownloadTestFile1.pptx
5. In client 2 window, “get downloadTestFile2.pptx” from the server and write the file to the local disk as newdownloadTestFile2.pptx
6. In client 1 window, “upload uploadTestFile1.pptx” to the server, which will write the file to the local disk as newuploadTestFile1.pptx
7. In client 2 window, “upload uploadTestFile2.pptx” to the server, which will write the file to the local disk as newuploadTestFile2.pptx

To pass the script, please follow exactly the above steps, with the commands in quotes. Try not to include extra or different input/output, so that you have a better chance to pass the script.

The reason to change the name of a file with a prefix “new” is to allow you to place all files, including the client, the server, and the test files (downloadTestfile1.pptx, downloadTestfile2.pptx, uploadTestfile1.pptx, and uploadTestFile2.pptx), in the same directory; after download/upload, the newly created files will not overwrite the original test files. Note that the script will add the test files to the directory and you do not need to include them in your submission.

To pass the script, Java and Python are highly preferred because they are platform-independent.

The test files can be found on Canvas. They are the same as chap1.pptx, chap2.pptx, chap3.pptx, and chap5.pptx but under different names, downloadTestFile1.pptx, uploadTestFile1.pptx, downloadTestFile2.pptx, and downloadTestFile2.pptx, respectively. These files are large. You are required to use a loop to break them into smaller chunks and send each chunk at a time.

1. **Summary of Details**

**Implementation:**

1. **server:**
   * Run on port 5106 (or another available port if 5106 is occupied)
   * Serve multiple clients concurrently using threads
   * Support file download/upload
2. **client:**
   * Connect to the server using the specified port number, e.g., 5106
   * Download using the get command
   * Upload using the upload command

**Compilation:**

**For Java:**

javac server.java client.java

**For Python:** (No compilation needed)

**For C++:**

g++ server.cpp -o server  
g++ client.cpp -o client

**Execution for Self-Testing:**

1. **Start the server:**

* **For Java:**
  + - java server
* **For Python:**
  + - python server.py
* **For C++:**
  + - ./server

1. **Start two client instances in different windows:**

* **For Java:**
  + - java client 5106   
      java client 5106
* **For Python:**
  + - python client.py 5106   
      python client.py 5106
* **For C++:**
  + - ./client 5106   
      ./client 5106

1. **File Transfer Testing:**
   * In **Client 1** window:
   * get downloadTestFile1.pptx
     + The file should be received and stored locally as newdownloadTestFile1.pptx.
   * In **Client 2** window:
   * get downloadTestFile2.pptx
     + The file should be received and stored locally as newdownloadTestFile2.pptx.
   * In **Client 1** window:
   * upload uploadTestFile1.pptx
     + The file should be sent to the server and stored as newuploadTestFile1.pptx.
   * In **Client 2** window:
   * upload uploadTestFile2.pptx
     + The file should be sent to the server and stored as newuploadTestFile2.pptx.

**Files:**

* All files for upload/download should be placed in the same directory as the source code
* The test files (downloadTestFile1.pptx, uploadTestFile1.pptx, downloadTestFile2.pptx, uploadTestFile2.pptx) will be added to the directory by the script, so you do not need to include them in your submission.
* Upon submission, all project files must be packaged into a .zip, .rar, or .tar.gz archive.
* The archive should either contain no directories or a single top-level directory.

By following these guidelines precisely, your project will be compatible with the script.

1. **Programming Environment**

Programming language: Java, C, C++, Python

Operating System: Windows, Mac OS or Linux

Programming Tool: Eclipse, IntelliJ, Jcreator, Kawa, Netbeans, … whatever you like.

To use Eclipse, please go through the following list:

1. Download JDK from: <https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

2. Download Eclipse from: <http://www.eclipse.org/downloads/>

3. Here is a link for eclipse tutorial: <http://eclipsetutorial.sourceforge.net/totalbeginner.html>

4. Here is a tutorial for socket programming in Java: <http://java.sun.com/docs/books/tutorial/networking/sockets/>

1. **Code Submission**

If you use Java, you will need to submit the following files: server.java, client.java, server.class, and client.class in a zipped directory, e.g., project2.rar or project2.zip. Please make sure to include server.class and client.class in the submission.

If you use C/C++ or Python please put all source files and executables in a zipped directory.

If you are an online student, you must include README.txt to explain how a TA will run your program, particularly if you use C/C++.

Submit the project through Canvas:

1. Go to <https://lss.at.ufl.edu/>
2. Click “Login to e-Learning”
3. Login with your gator link username/password
4. Go in CNT 4007
5. Click “Assignments” and submit your project

This is an **individual** project. Students must submit their code via Canvas by the deadline. We will run an automatic tool to catch submissions with identical or similar code. There will be no late submissions allowed.