COEN 366 - Winter 2022
Python Project
Python CIient - Server File Transfer
Aydin Azari Farhad - 40063330
Design Document

In short this program is a pair of two python scripts, client.py and server.py. The program utilizes the functionality of sockets to create a connection between a client and a server and simulate a file transfer protocol.

Client.py

The client.py program begins by importing necessary libraries, defining some parameters, and defining the necessary functions.

- **fileNameToBin(fileName)**
  This takes the name of a file and returns its name as Unicode binaries.
- **getFileSize(file)**
  This takes the name of a file and returns its size as a set of 8 binary digits.
- **fileNameLength(file)**
  This takes the name of a file and returns the length of the file name as a set of 8 binary digits.
- **put(fileName)**
  This function handles “put” commands. It passes the file name to the three functions mentioned above. Then it creates a byte object by concatenating the result of the above functions and sends that object to the socket. Next it begins reading the file in 4 Bytes portions and sends those to the socket. Once the number of bytes sent is equal to the size of the file to be sent, the function stops sending and waits for an ok response.
- **get(fileName)**
  This works in the opposite way to the put() function. It lets the server know which file it wants, and receives the binaries that server sends through the client and decodes them. Once it gets the name and size of the file, it creates the file and writes to it.
- **change(oldfileName, newFileName)**
  This takes the old name of the file and a new name, and lets the server know that the user wishes to rename the file corresponding to parameter 1 to parameter 2 and then waits for a confirmation from the server.
- **help()**
  This lets the server know that the user wants a list of available commands and prints the received message from the server
- **bye()**
  This closes the open socket and quits the program.
- **getCmd(splitCmd).**
  This checks the first portion of user inputs and calls the function corresponding to the operation. If the operation is wrong, it lets the user know.

In the main part of the program, the user is asked to enter an IP and port number as well as a command. The program splits the command into an array to distinguish between the operator

and its operands. The in an infinite loop (until bye is called), the program creates a socket connection, connects to it, and executes the corresponding function for the inputted operation.

Server.py

The server.py program begins by importing necessary libraries, defining some parameters, and taking user input for IP and port number. It then connects to that port and listens fo incoming messages from a client. Once it receives a message, it checks the first 3 bits of the message and uses a series of if/else statements to determine what operation needs to be done. If the 3 bits do not match any of the if statements, the finale else statement sends a message back to the client to inform it of the unknown operation.

To convert the incoming message, the function uses decode(). However, some strings require a process of conversion back to Unicode and then back to string, which the program takes care of. The functionalities work almost identically to the ones in client.py.