Anton Krotenok (NETID: ak1847)

Kevin George (NETID: kmg328)

Brief Explanation of The Project

**The client wishes to send text data to the server for some kind of manipulation. After properly accepting the original file, the server completes a set of processes to generate text that is (separately) both the uppercase and reversed versions of the original. After the server reversed/uppercased the original file, it automatically *wrote* the manipulated text to output files “outr-proj.txt” and “outup-proj.txt”. The client was then notified in the terminal that the files had been manipulated and written to those files. This is a demonstration that data/information has successfully been transferred between the client and server, back-and-forth.**

1. Team details: Clearly state the names and netids of your team members (there are 2 of you).

**Anton Krotenok (NETID: ak1847)**

**Kevin George (NETID: kmg328)**

2. Collaboration: Who did you collaborate with on this project? What resources and references did you consult? Please also specify on what aspect of the project you collaborated or consulted.

**Both partners utilized past knowledge from classes including Computer Architecture, Software Methodology, and Principles of Information and Data Management. We did not collaborate with any outside sources.**

3. What did you observe after running step 2 above? Can you explain why you see what you see?

**After running step 2 above, we noticed the following: (1.) The print(“Done”) line [from the main “if” thread comparison] executes before the rest of the code from the server & client. (2.) Once the print(“Done”) statement is printed, the client socket is created before the server IP address is instantiated.**

**After running step 2 again, the following happened: (1.) The print statements only show that the server & client sockets have been created. (2.) Exception print statements are shown for both the server & client. The server statements specify that the “Address already in use” while the client statements specify “Connection refused”.**

**We see this in the results for the following reason: As a result of removing the “time.sleep…” statements, the iLab machine is attempting to operate two threads simultaneously. Given average computer specifications, it is not possible to have two threads working concurrently, or at the same time. The “time.sleep…” line allows enough time for the switching between threads to *seem* as if they are working together. In reality, the computer suspends the execution of a program for a specified number of seconds.**

4. You should name the output file from step 3 as out-proj.txt

**Output:**

**[S]: Server socket created**

**[S]: Server host name is ilab1.cs.rutgers.edu**

**[S]: Server IP address is 128.6.4.101**

**[S]: Got a connection request from a client at ('128.6.4.101', 58708)**

**[C]: Client socket created**

**[C]: Data received from server: Welcome to CS 352!**

**This request has been completed. The file can be found in the zipped folder.**

5. For each cases in step 4 please name the output file as outr-proj.txt (reversed string), and outup-proj.txt (upper case). Your output files must match exactly with the ones shown.

**outr-proj.txt output:**

**evaC A nI staB batS I naC ,avE**

**emit eht lla niw stoirtaP**

**madA m'I ,nedE nI madaM**

**trop dna sserdda PI sdeen tekcos a**

**outup-proj.txt output:**

**EVA, CAN I STAB BATS IN A CAVE**

**PATRIOTS WIN ALL THE TIME**

**MADAM IN EDEN, I'M ADAM**

**A SOCKET NEEDS IP ADDRESS AND PORT**

6. Is there any portion of your code that does not work as required in the description above? Please explain.

**No, all portions of our code work as required in the description above.**

7. Did you encounter any difficulties? If so, explain.

**We faced a few difficulties. The most notable difficulty was figuring out how to send messages from client to server and messages of an unknown size on the server side. When receiving the message in the server program we couldn’t assume the buffer size of the input message from the client. So, we had to implement a loop which repeatedly read a finite amount of letters from the input until there were none left. This led to another difficulty which was terminating the loop when recv() had nothing being sent to it. To fix this we added a try except statement that would catch the socket essentially timing out so we could continue with our code instead of letting it hang.**

8. What did you learn from working on this project? Add any interesting observations not otherwise covered in the questions above. Be specific and technical in your response. Contact the course staff on Piazza if you have any questions.

**Through practical experimentation, Anton and Kevin experimented with multithreading in Python to better understand its functionality and the importance of the “time.sleep()” function. Unfamiliar with socket programming, both partners learned the practicality of “sockets” and how different programs are able to communicate with one another via a network. It is also interesting to analyze the significance of system design and its relation to computer architecture. For example, both group members had to consider the potential ways in which the project could send, receive, and display output, this being both within the terminal and outside text files.**