Internet Architecture Board (IAB)
Request for Comments: 7990
Category: Informational
ISSN: 2070-1721

Tyler Henn
RFC Editor
October 2018

RFC

Abstract

The objective of these programs was to establish a server that another computer could connect to. This server was given parameters on how to respond and responded accordingly. This code is written in python. The client side of the program will input the server name of where the server is running as well as the port number to connect. The user will be prompted to input a message and how they would like their response from the server to be formatted. From there the message is sent to the server and the server organizes and figures out what the user request is and responds.

Terminology

Transport Layer: This layer provides the protocols for logical communication between application. This sets up the experience of direct connectivity. We use in the project to create the server and allow the user the ability to get direct connectivity.

Port Numbers: A port number is a 16-bit integer that is put in the header appended to a message unit. We use port numbers to allow a specific process to be able to connect to a server a send messages to that server.

Socket: Used to allow communication between two programs running over a network. In this project used to allow the client to communicate with the server.

Connect: This function is used to establish connection with the server from the client.

Bind: This function is used to bind the server to the socket the client is running on.

How to operate

To begin you will need to run the server program which is written in python. When you run the server, the server will first create a TCP socket using the socket () function. After it has successfully created the socket the server will print a message saying, “Socket created successfully.” Next the server will bind to the port 9999 by default. The server will bind to this port using the bind()function. It will print a message saying, “The socket binded to 9999.” Finally, the server will begin listening using the listen(5)function and will listen for clients and print out a message “The server is ready to receive.” If you have gotten all above messages the server is now ready to receive communications.

Next, you will want to run the client program through command line. This code is also in python. You will want to begin by running the program and on the same line you will want to input the server name (ex. Localhost or an ip address) and the server port number that the server is running on. In this case you will want to input 9999. Example command line input, python .\folder\EnhancedTCPclient.py localhost 9999. After pressing enter the client will run. It will begin by connecting to the socket of the server using the connect() function. On the server side, the server will accept the connection using the accept() function and will output a message saying “Got connection from” (destination of client). From there the client will prompt you to enter a message to send to the server. After you have decided on the message you would like to send you will be prompted to choose 1 of 3 formats. You can either choose upper, lower, or reverse. After selecting the format you would like the client will encode the string into bytes using the encode() function and it will send the messages to the server using the send()function. The server will then receive the two messages and will decode the messages, using the decode() function, back to a string. From there the server will determine what format you have choosen and will either change the sentence to all uppercase using the upper() function, change the sentence to all lowercase using the lower() function, or will reverse sentences of more then 2 words using the “”.join(message.split()[::-1]) function. After determining the desired format, the server will reencode the message you have sent and send it back to you. The client program running will receive the message and decode the message and output the respond from the server as well as your desired format.