Sockets Assignment 5 Report

Instruction:

Sockets 5: Extend your chatbot server so that it can handle multiple simultaneous conversations at once by spawning a new thread for every client. (est 3 hours)

Explanation:

As this lab's instruction says, what we need to accomplish for this lab is to make multithreading for a chatbot.

On the client-side, there was no significant change regards the program itself. I only added the print statement on line 5 to show which socket the client is communicating with the server.

On the server-side, the multithreaded functionality was added. Since a new thread has to be created, I introduced a method called “start\_new\_thread,” which receives two arguments. The first argument will receive the function's name, and the second argument is the parameter for the function for the first parameter. The function ‘conv,’ which is passed as the first parameter of the start\_new\_thread function, contains the original chatbot server-side code, which handles the response and receives answers from the client-side. Every new connection (new thread) will run the while loop on line 55, and the counter (t\_num) will increment by one. The print statement will show which socket is handling the communication with the client-side. The function start\_new\_thread will returns immediately, and the child thread will start by calling the first argument with the second argument as the parameters. When the function returns, the thread is terminated.

Code:

Client:

1. #!/usr/bin/env python3
2. from socket import \*
3. s = socket(AF\_INET, SOCK\_STREAM)
4. s.connect(("127.0.0.1", 7069))
5. print("Connected to the server.\n\thandling communication with", s.getsockname())
6. while 1:
7. data = s.recv(10000)
8. print(data.decode())
9. message = input("Client:\t")
10. s.send(message.encode())
11. if message == "e":
12. print("Exiting...")
13. s.close()
14. break
15. s.close()

Server:

1. #!/usr/bin/env python3
2. from socket import \*
3. import \_thread
4. def conv(c, a):
5. counter = 0
6. while True:
7. if counter == 0:
8. c.send("System:\tHello, welcome to chatbot program. \nSystem:\tAre you importing messages from file? \nSystem:\tEnter only either \"yes\" or \"no\".".encode())
9. counter += 1
10. elif counter == 1:
11. if data == "yes":
12. c.send("System:\tImport file but currently not available. \nSystem:\tHello, are you male or female?".encode())
13. counter += 1
14. elif data == "no":
15. c.send("System:\tHello, are you male or female?".encode())
16. counter += 1
17. else:
18. c.send("System:\tPlease enter in the correct format ... \n".encode())
19. elif counter == 2:
20. if data == "female":
21. c.send("System:\tHow excellent! \nSystem:\tAre you a CS major?".encode())
22. elif data == "male":
23. c.send("System:\tMe too. \nSystem:\tAre you CS major?".encode())
24. else:
25. c.send("System:\tGreat! \nSystem:\tAnyways, are you CS major?".encode())
26. counter += 1
27. elif counter == 3:
28. if data == "no":
29. c.send("System:\tToo bad. \nSystem:\tAnyway, what's an animal you like, and two you don't?".encode())
30. elif data == "yes":
31. c.send("System:\tExcellent, I am too. \nSystem:\tWhat's an animal you don't like, and two you don't?".encode())
32. else:
33. c.send("System:\tCool! \nSystem:\tBy the way, what's an animal you like, and two you don't?".encode())
34. counter += 1
35. elif counter == 4:
36. data1 = data.split(',')
37. msg = "System:\t%s awesome, but i hate %s too. \nSystem:\tBye for now. \n\*\*\*Enter \'e\' to exit program." % (data1[0].strip(), data1[-1].strip())
38. c.send(msg.encode())
39. counter += 1
40. else:
41. c.send(''.encode())
42. data = c.recv(1000).decode().strip()
43. if data == "e":
44. print("Exiting...")
45. c.send("e".encode())
46. c.close()
47. break
48. c.close()
49. s = socket(AF\_INET, SOCK\_STREAM)
50. s.bind(("127.0.0.1", 7069))
51. s.listen(5)
52. t\_num=0
53. while 1:
54. c,a = s.accept()
55. \_thread.start\_new\_thread(conv, (c,a))
56. t\_num += 1
57. print("I am new thread %d\n handling communication with %s" % (t\_num, (a,)))

Output:

Server -> Text

Description automatically generated

Client1 -> Text

Description automatically generated

Client2 -> Text

Description automatically generated

Client3 -> Text

Description automatically generated