Bryan Arnold

11/15/18

CSE 3300

Programming Assignment 3

Exercise 0

Here is an example scenario of output generated by the client and server interaction (**NOTE:** although \n isn't explicitly printed, it does still exist as the console just prints out a newline instead of showing \n). The server numbers in steps 5 and 7 are 1475160574 and 1475160575 respectively:

First message to the server: ex0 137.99.3.212-3300 137.99.81.245-4764 6389 B.Arnold

Server response to first message: CSE 3300 Server Thu Nov 15 19:49:08 2018

OK 6390 B.Arnold 1475160574

Second message to server: ex0 6391 1475160575

Second server response: CSE 3300 Server Thu Nov 15 19:49:08 2018 OK 1475160575

Process finished with exit code 0

Code

```
# Imports
import urllib.request, socket, random, re

############################
# Bryan Arnold             #
# 11/15/18                 #
# CSE 3300                 #
# Programming Assignment 3 #
############################

# Used this as a resource in order to find the local ip address of the machine
```

```
# https://stackoverflow.com/questions/2311510/getting-a-machines-external-ip-address-
with-python
# Generate random integer for usernum and initialize ip
ipAddress = urllib.request.urlopen('https://ident.me').read().decode('utf8')
port = random.randint(0, 9000)

# Get the server ip as well as the port
hostIP = socket.gethostbyname('tao.ite.uconn.edu')
hostPort = 3300

# Create new socket and initialize it
# Generate random usernum and create student name
newSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
usernum = random.randint(1000, 8000)
studentName = 'B.Arnold'

# Establish connection to the server with socket
newSocket.connect((hostIP, hostPort))

# Create string to send to the server and send it
stringToSend = ('ex0' + ' ' + str(hostIP) + '-' + str(hostPort) + ' ' + ipAddress + '-
' + str(port) + ' ' + str(usernum) + ' ' + studentName + '\n').encode()
newSocket.send(stringToSend)
print('First message to the server: ' + stringToSend.decode())

# Get the data in the response back from the server
recvBuf = 1500
data = newSocket.recv(recvBuf).decode()

# If the data contained 'OK' within it, the message sent was good, so read the
contents
# of the server response
if data.find("OK") != -1:

    # Print the contents of the server response and prepare the ACK message to send
back
    # to the server by parsing the servernum from the response data
    print('Server response to first message: ' + data)
    recvBuff = 1500
    servernum = int(data[data.rindex(' ') + 1:])
    newStringToSend = ('ex0' + ' ' + str(usernum + 2) + ' ' + str(servernum + 1) +
'\n').encode()
    print('Second message to server: ' + newStringToSend.decode())

    # Send the ack response back to the server
    # and await another server response
    newSocket.send(newStringToSend)
    newData = newSocket.recv(recvBuff).decode()

    # If the response of the ack message to the server contains
    # 'OK', the request was good so print the data
    if newData.find("OK") != -1:

        print('Second server response: ' + newData)

    # Bad request, print an error
    else:

        print('An error has occurred with sending the second message to the server: '
+ newData)

# Bad request, print an error
else:
```

```
    print('An error has occurred with sending the first message to the server: ' +
data)

# Close connection of the socket to the server
newSocket.close()
```

Exercise 1

Here is an example scenario of output generated by the client and server interaction (**NOTE:** although \n isn't explicitly printed, it does still exist as the console just prints out a newline instead of showing \n). The three server numbers received from the server were 816664702, 1589221326, and 816664703:

First message to the server: ex1 137.99.3.212-3300 137.99.81.245-61462 1742 B.Arnold

Server response to first message: CSE 3300 Server Thu Nov 15 19:55:32 2018

OK 1743 B.Arnold 816664702

Message from the server to local machine waiting: CSE 3300 server calling 1589221326

Second message to server: 816664703 1589221327

Server response to second message: CSE 3300 Server Thu Nov 15 19:55:32 2018 OK 816664703

Process finished with exit code 0

Code

```
# Imports
import urllib.request, socket, random, re

############################
# Bryan Arnold             #
# 11/15/18                 #
```

```
# CSE 3300                 #
# Programming Assignment 3 #
############################

# Used this as a resource in order to find the local ip address of the machine
# https://stackoverflow.com/questions/2311510/getting-a-machines-external-ip-address-
with-python
# Generate random integer for usernum and initialize ip
ipAddress = urllib.request.urlopen('https://ident.me').read().decode('utf8')

# Get the server ip as well as the port
hostIP = socket.gethostbyname('tao.ite.uconn.edu')
hostPort = 3300

# Create new socket and initialize it
# Generate random usernum and create student name
startSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
usernum = random.randint(1000, 8000)
studentName = 'B.Arnold'

# Create another socket to wait for the server to communicate
# with the local client and initialize it. Get the ip address
# and specific TCP identifier of the client.
psock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
psock.bind(('', 0))
port = psock.getsockname()[1]
psock.listen(10)

# Establish connection to the server with socket
startSocket.connect((hostIP, hostPort))

# Create string to send to the server and send it
stringToSend = ('ex1' + ' ' + str(hostIP) + '-' + str(hostPort) + ' ' + ipAddress + '-
' + str(port) + ' ' + str(usernum) + ' ' + studentName + '\n').encode()
startSocket.send(stringToSend)
print('First message to the server: ' + stringToSend.decode())

# Get the data in the response back from the server
# and hold onto the servernum the server sends back
recvBuf = 1500
data = startSocket.recv(recvBuf).decode()
servernum = int(data[data.rindex(' ') + 1:])

# If the data contained 'OK' within it, the message sent was good, so read the
contents
# of the server response and continue
if data.find("OK") != -1:

    # Print the first message response
    print('Server response to first message: ' + data)

    while True:

        # Create a new buffer to receive the incoming request to
        # connect to the locla client from the server. Initialize the
        # client credentials and socket. Get the connection request
        # from the server.
        recvBuff = 1500
        client, clientAddress = psock.accept()
        newData = client.recv(recvBuff).decode()
        print('Message from the server to local machine waiting: ' + newData)

        # If the server connection was established, continue
```

```
        if newData:

            # Create another buffer to accept server response,
            # parse the newservernum out of the message sent by the server
            # after establishing a connection, and create the next message to send
            # the server.
            recvBuffFinal = 1500
            newServerNum = int(newData[newData.rindex(' ') + 1:])
            finalStringToSend = (str(servernum + 1) + ' ' + str(newServerNum + 1) +
'\n').encode()
            print('Second message to server: ' + finalStringToSend.decode())

            # Send the last message to the server and get the response
            client.send(finalStringToSend)
            finalData = startSocket.recv(recvBuffFinal).decode()

            # If the response contains 'OK', the request was good, so continue
            # on a print out the response and close the client socket.
            if finalData.find("OK") != -1:

                print('Server response to second message: ' +
data[:(data.rindex('2018') + 4)] + ' ' + finalData)
                client.close()

            # Bad request from server, throw an error
            else:

                print('An error occurred with sending the last message to the server:
' + data[:(data.rindex('2018') + 4)] + ' ' + finalData)

        # Bad request, print an error
        else:

            print('An error occurred with getting a message from the server: ' +
newData)

        break

# Bad request, print an error
else:

    print('An error has occurred with sending the first message to the server: ' +
data)

# Close connection of the socket to the server
startSocket.close()
```