Cosc264 Assignment1

Mohadesa Sharifi
ID: 89853938

Server.py

```
import socket
import sys
import os.path
from datetime import datetime
PORT = int(sys.argv[1])
SERVER = socket.gethostbyname(socket.gethostname())
ADDR = (SERVER, PORT)
FORMAT = 'utf-8'

def readFileRequest(conn):
    # read a FileRequest record from the connection.
    # stores them in a byte array
    # checks the validity of the FileRequest record
    # If the FileRequest record is correct and contains a filename, the server tries to open the file for reading.
    # If FileRequest is not valid server discconects and print error message.
    data = bytearray(conn.recv(1024))
    if len(data) < 5:
        print("packet is too small")
        return -1
    if (data[0] << 8) | data[1] != 0x497E:
        print("Magic number error.")
        return -1
    elif data[2] != 1:
        print("type error.")
        return -1
    elif (data[3] << 8) | data[4] > 1024:
        print("file name length error.")
        return -1
    n = (data[3] << 8) | data[4]
    filename = data[5:]
    while len(filename) != n:
        data = bytearray(conn.recv(1024))
        filename += data[5:]
    return filename.decode(FORMAT), n

def openFile(filename):
    # If the FileRequest record is correct and contains a filename, the server tries to open the file for reading.
    # If the file does not exist or cannot be opened, the server sends an FileResponse message
    file = open(filename, 'r')
    data = file.read()
    file.close()
    return data

def sendFileResponse(conn, data):
    # if file validation fails server send message back to the requesting client,
    # closes accept(), prints an informational message
    # if valid the server send message back to the client (actual contents of the file)
```

```
    # Once the file transfer has been completed, the server closes the file,
    # it closes accept(),prints message which includes the actual number of bytes transferred

    data = bytearray(data, FORMAT)
    condition = True
    bytecount = 0
    maxSize = 4096 - 64

    while len(data) >= maxSize:
        fileresponseheader = bytearray(((0x497E << 48) + (2 << 40) + (1 << 32) + len(data)).to_bytes(8, "big"))
        pkt = fileresponseheader + data[:maxSize]
        conn.sendall(pkt)
        data = data[maxSize:]
        bytecount += len(pkt)

    if len(data) != 0:
        fileresponseheader = bytearray(((0x497E << 48) + (2 << 40) + (1 << 32) + len(data)).to_bytes(8, "big"))
        pkt2 = fileresponseheader + data
        conn.sendall(pkt2)
        bytecount += len(pkt2)
    return bytecount


def server(port_number):
    if port_number < 1024 or port_number > 64000:
        print("invalid port number")
        sys.exit(0)
    #socket.AF_INET: All socket options starts with AF. Family socket ipv4.
    # .SOCK_STREAM: how to tranfer data over socket. it uses streaming
    mySocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    try:
        mySocket.bind(ADDR)
        print("Socket binded to the address")
        return mySocket
    except:
        print("Binding failed")
        sys.exit(0)

def start():
    mySocket = server(PORT)
    try:
        mySocket.listen()
        print("Listening on port: {}...".format(PORT))
    except:
        print("Listening failed")
        mySocket.close()
        sys.exit(0)

    while True:
```

```
        now = datetime.now()
        current_time = now.strftime("%H:%M:%S")
        conn, addr = mySocket.accept()
        print("At current time {} Socket connected to IP_address: {}, port: {}".format(current_time, addr[0], PORT))
        try:
            dataarray, n = readFileRequest(conn)
            if dataarray != -1:
                filename = dataarray
                print("The requested file name is: {}".format(filename))
                if len(dataarray) != n:
                    print("Data does not match the filename size.")
                    conn.close()
                if os.path.exists(filename):
                    print("File exists. Please read the content of data before transferring")
                    data = openFile(filename)
                    print("{}".format(data))
                    bytecount = sendFileResponse(conn, data)
                    print("The number of bytes which has been sent: ", bytecount)
                    conn.close()
                else:
                    print("file does not exist.\n")
                    conn.close()
            else:
                conn.close()
        except:
            print("File name oes not exist or it cannot be openned\n")
            conn.close()

#server(PORT)
start()
```

Client.py

```
import socket
import sys
import os.path

SERVER = socket.gethostbyname(socket.gethostname())
FORMAT = 'utf-8'
if len(sys.argv) != 4:
    print("The number of parameters in command line is not correct")
    sys.exit(0)
HOST = str(sys.argv[1])
PORT = int(sys.argv[2])
FILENAME = str(sys.argv[3])

def client(host, port, filename):
    # host, port, file are to be read from the input
    # host is an IP address or hostname of computer, getaddrinfo()
    # if IP address is not valid, print error message and exit
    # port number is (1024 and 64000)/ error message and exit
    # filename: check it exist and can be opened locally
    # if filename fails print error, exit and donot over_write the file with the same name
    # If there are more than 3 args print error and exit
    # create socket/ if fails: print error, exit
    # call connect/ if fails: close socket, error, and exit
    try:
        info = socket.getaddrinfo(host, port, proto=socket.IPPROTO_TCP)
        ipv4 = info[-1]
        ipport = ipv4[-1]
        ip, num = ipport
    except:
        print("invalid ipv4 address.")
        sys.exit(0)
    if port <= 1024 or port >= 64000:
        print("Invalid port number.")
        sys.exit(0)
    elif os.path.exists(filename):
        print("File already exists.")
        sys.exit(0)
    else:
        try:
            clientSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            clientSocket.settimeout(2)
        except:
            print("Socket creation failed.")
            sys.exit(0)
        try:
            clientSocket.connect((ip, port))
            print("connected")
```

```
            except:
                clientSocket.close()
                print("cannot connect to the given ipv4 address")
                sys.exit(0)
            try:
                sendFileRequest(clientSocket, filename)
                print("File request sent successfully")
            except:
                print("Sending a file request failed")
                sys.exit(0)
            try:
                storeFile(clientSocket, filename)
                print("File read and transferred successfully")
            except:
                print("file trasfer failed")
                sys.exit(0)
        clientSocket.close()
        sys.exit(0)

    def sendFileRequest(clientSocket, filename):
        # encode filename and store it in a bytearray
        # add header to the file request
        # send file request to the server
        file_name = bytearray(filename.encode(FORMAT))
        filename_len = len(file_name)
        if filename_len < 1024:
            filerequest = bytearray(((0x497E << (24)) + (1 << (16)) + (filename_len)).to_bytes(5, "big"))
            filerequest += file_name
            clientSocket.sendall(filerequest)
        else:
            print("filename length is not valid")
            return -1

    def storeFile(clientSocket, filename):
        # open file for writing/if fails: error, close, exit
        # at the end of file transfer print message, colse file, socket, exit
        content = readFileResponse(clientSocket)

        if len(content) > 0:
            decoded = content.decode('utf-8')
            file = open(filename, "w")
            file.write(decoded)
            file.close()
        else:
            print("file has no content")
            return -1

    def readFileResponse(clientSocket):
         # client will read a FileResponse record from the server
```

```
# stores them in a byte array, and checks the validity
# if not valid or takes more than 1 sec, error, close, exit
# if valid: if no content/data, error, close, exit
# process file data: tranfer to byte array, count bytes, no delay
try:
    initdata = bytearray(clientSocket.recv(4096))
    clientSocket.settimeout(1)
except:
    print("data recieved with a delay of more than 1 second")
    sys.exit(0)
bytecount = 0
if (initdata[0] << 8) | initdata[1] != 0x497E:
    print("Magic number error.")
    sys.exit(0)
elif initdata[2] != 2:
    print("Packet type error.")
    sys.exit(0)
elif initdata[3] != 1:
    print("file not found.")
    sys.exit(0)
else:
    contentlen = ((initdata[4] << 24) | (initdata[5] << 16) | (initdata[6] << 8) | initdata[7])
    content = initdata[8:]
    bytecount += len(initdata)
    try:
        data = bytearray(clientSocket.recv(4096))
    except:
        print("data recieved with a delay of more than 1 second")
        clientSocket.close()
        sys.exit(0)

    while len(data) != 0:
        bytecount += len(data)
        content += data[8:]
        try:
            data = bytearray(clientSocket.recv(4096))
        except:
            print("data recieved with a delay of more than 1 second")
            clientSocket.close()
            sys.exit(0)

    if len(content) != contentlen:
        print("file length does not match the length.")
        clientSocket.close()
        sys.exit(0)
    else:
        print(bytecount, "bytes received.")
        return content
```

```
client(HOST, PORT, FILENAME)
```

Plagiarism Declaration

This form needs to accompany your COSC 264 assignment submission.

I understand that plagiarism means taking someone else's work (text, program code, ideas, concepts) and presenting them as my own, without proper attribution. Taking someone else's work can include verbatim copying of text, figures/images, or program code, or it can refer to the extensive use of someone else's original ideas, algorithms or concepts.

I hereby declare that:

- My assignment is my own original work. I have not reproduced or modified code, figures/images, or writings of others without proper attribution. I have not used original ideas and concepts of others and presented them as my own.
- I have not allowed others to copy or modify my own code, figures/images, or writings. I have not allowed others to use original ideas and concepts of mine and present them as their own.
- I accept that plagiarism can lead to consequences, which can include partial or total loss of marks, no grade being awarded and other serious consequences, including notification of the University Proctor.

Name: ……

Student ID: ……

Signature: ……

Date: ……