COSC 264 Assignment

Phillip Kim – 63083227

Client.py

```
import socket
import sys
import os.path

if len(sys.argv) != 4:
    print("parameter error")
    sys.exit(0)

HOST = str(sys.argv[1])
PORT = int(sys.argv[2])
FILENAME = str(sys.argv[3])

def readdata(sock):
    initdata = bytearray(sock.recv(1024))
    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)
        data = bytearray(sock.recv(1024))
        while len(data) != 0:
            bytecount += len(data)
            content += data[8:]
            data = bytearray(sock.recv(1024))
        print("Received data")
        if len(content) != contentlen:
            print("file length does not match the length.")
            return -1
        else:
            print(bytecount, "bytes received.")
            return content

def client(host, port, filename):
    try:
        info = socket.getaddrinfo(host, port, proto=socket.IPPROTO_TCP)
        ipv4 = info[-1]
        ipport = ipv4[-1]
        ip, num = ipport
    except:
        print("invalid 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:
            csock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            csock.settimeout(2)
        except:
```

```
            print("Socket creation failed.")
            sys.exit(0)
        try:
            csock.connect((host, port))
            print("connected")
        except:
            csock.close()
            print("socket error")
            sys.exit(0)
        name = bytearray(filename.encode("utf-8"))
        nameleng = len(name)
        if nameleng < 1024:
            filerequest = bytearray(((0x497E << (24)) + (1 << (16)) +
(nameleng)).to_bytes(5, "big"))
            filerequest += name
            csock.sendall(filerequest)
            content = readdata(csock)
            if content != -1:
                decoded = content.decode('utf-8')
                file = open(filename, "w")
                file.write(decoded)
                file.close()
    sys.exit(0)


client(HOST, PORT, FILENAME)
```

Server.py

```
import socket
import sys
import os.path

HOST = '127.0.0.1'
PORT = int(sys.argv[1])

def readfilerequest(clisocket):
    data = bytearray(clisocket.recv(1024))
    if (data[0] << 8) | data[1] != 0x497E:
        print("Magic number error.")
        return -1, -1
    elif data[2] != 1:
        print("Packet type error.")
        return -1, -1
    elif (data[3] << 8) | data[4] > 1024:
        print("Name size error.")
        return -1, -1
    n = (data[3] << 8) | data[4]
    name = data[5:]
    while len(name) != n:
        data = bytearray(clisocket.recv(1024))
        name += data[5:]
    return name, n

def openfile(filename):
    file = open(filename, 'r')
    content = file.read()
    file.close()
    return content

def fileresponse(validity, sock, content=''):
    if validity is True:
        content = bytearray(content, "utf-8")
        fileresponseheader = bytearray(((0x497E << 48) + (2 << 40) + (1 << 32) +
len(content)).to_bytes(8, "big"))
        first = True
        bytecount = 0
        while len(content) >= 88 or first is True:
            pkt = fileresponseheader + content[:88]
            sock.sendall(pkt)
            content = content[88:]
            first = False
            bytecount += len(pkt)
        if len(content) != 0:
            pkt2 = fileresponseheader + content
            sock.sendall(pkt2)
            bytecount += len(pkt2)
        print(bytecount, "bytes sent.\n")
        return bytecount
    else:
        sock.sendall(bytearray(((0x497E << 48) + (2 << 40) + (0 << 32) +
len(content)).to_bytes(8, "big")))
        print("The file does not exist or cannot be opened.")

def server(port):
    if port <= 1024 or port >= 64000:
        print("Invalid port number.")
        sys.exit(0)
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    try:
        sock.bind((HOST, port))
```

```
    except:
        print("Binding failed.")
        sys.exit(0)
    try:
        sock.listen()
    except:
        print("Listening failed.")
        sys.exit(0)
    loop = 0
    while loop == 0:
        clisocket, cliip = sock.accept()
        print("Connected to", cliip[0])

        try:
            dataarray, n = readfilerequest(clisocket)
            if dataarray != -1:
                filename = dataarray.decode("utf-8")
                print("File name", filename)
                if len(dataarray) != n:
                    print("Data does not match the actual name size.")
                    clisocket.close()
                if os.path.exists(filename):
                    print("File found.")
                    content = openfile(filename)
                    bytecount = fileresponse(True, clisocket, content)
                    clisocket.close()
                else:
                    print("file does not exist.")
                    fileresponse(False, clisocket)
                    clisocket.close()
            else:
                clisocket.close()
        except:
            fileresponse(False, clisocket)
            clisocket.close()


server(PORT)
```

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: Phillip Kim

Student ID: 63083227

Signature: 

Date: 18/08/19