University of Canterbury

COSC 264 ASSIGNMENT

with TCP sockets

Name: LUU Khanh Linh

Student id: 68697438

```python
"""
Author: LUU Khanh Linh
Class: COSC264
Lecturer: Andreas Willig
Start_date: 5 Aug 2019
End_date: 18 Aug 2019
Assignment: Write a client and a server application which allows a client to download a file of its choosing from the server.
            They will communicate through TCP sockets, exchanging both control and actual file data.
"""

import socket
import sys
from datetime import datetime
import os
import errno

# Sets up FileResponse record included first 8 bytes (fixed header) and n bytes of File Data
def FileResponse(filename):
    """Returns a bytearray of the FileResponse record"""
    # making a bytearray
    byte_array = bytearray()

    # magicNo
    magicNo = 0x497E
    magicNo_1 = magicNo >> 8
    magicNo_2 = magicNo & 0xFF
    byte_array.append(magicNo_1)
    byte_array.append(magicNo_2)

    # Type
    Type = (2).to_bytes(1, byteorder='big')
    byte_array += Type

    # StatusCode
    try:
        f = open(filename, 'rb')
```

```
            fileData = f.read()
            byte_array += (1).to_bytes(1, byteorder='big')
    except:
            byte_array += (0).to_bytes(1, byteorder='big')

    # DataLength
    if byte_array[3] == 0:
        byte_array += (0).to_bytes(4, byteorder='big')
    if byte_array[3] == 1:
        byte_array += len(fileData).to_bytes(4, byteorder='big')

    # FileData
    if byte_array[3] == 1:
        byte_array += fileData

    return byte_array

def checkPort(portNum):
    """Checks the port number which it should be between 1024 and 64000 otherwise prints an error message and exit"""
    if portNum >= 1024 and portNum <= 64000:
        pass
    else:
        print("Invalid Port Number")
        sys.exit()

def create_socket(portNum):
    """Returns a socket binded to the port number"""
    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)    # create a socket
        s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        s.bind(('', portNum))   # bind
    except socket.error as msg:
        print(msg)
        sys.exit()
    # listen
```

```python
    try:
        s.listen(5)
    except socket.error as msg:
        print(msg)
        s.close()
        sys.exit()

    return s

def accept(s):
    """Returns a new comming connection by accept()"""
    getDatetime = datetime.now().strftime("%H:%M:%S")
    conn, addr = s.accept()
    print("Connection at {} from {} has been created".format(getDatetime, addr))
    return conn, addr

def getFileNameLen(record):
    """Returns the number of bytes of filenameLen in FileRequest record"""
    filenameLen = (record[3] << 8) + record[4]
    return filenameLen

def checkFileRequest(record):
    """Check magic number, type and filename length in FileRequest record"""
    magicNo = (record[0] << 8) + record[1]
    Type = record[2]
    filenameLen = getFileNameLen(record)
    if magicNo == 0x497E or Type == 1 or (filenameLen >= 0 and filenameLen <= 1024):
        pass
    else:
        print("The received FileRequest is errorneous")

def receiveFileName(conn, header):
    """Returns filename in bytes and checks if server receives as many bytes as the filenameLen"""
    filenameLen = getFileNameLen(header)
    # attempts to read exactly n bytes from the FRes
    filename_b = conn.recv(filenameLen)
    # if server reads not equal n bytes then concludes processing failed and perform error processing
    if filenameLen == len(filename_b):
        pass
    else:
        print("Error occurred, the server reads the number of bytes in filename which is not equal to the filenameLen in FileRequest record")
        print("Processing failed")
    return filename_b

def serverProcessing(s):
    """The process of a new connection with a socket, checking FileRequest and opening then sending a file to the client (described by comments)"""
    while True:
        # wait for connection
        print("Waiting for connection")
        # create a new connection
        conn, addr = accept(s)
        conn.settimeout(1)
        try:
            header = conn.recv(5)    # receive fixed header

        except socket.timeout as msg:    # for timeout
            print("The received FileRequest is erroneous")
            print(msg)    # appropriate error
            conn.close()    # close socket obtained from accept()
            continue    # back to while loop

        checkFileRequest(header)    # check FileRequest

        conn.settimeout(1)
        try:
            filename_b = receiveFileName(conn, header)    # get filename by bytes
        except socket.timeout as msg:
            print(msg)
```

```python
            conn.close()
            continue

        actual_filename = filename_b.decode('utf-8')    # get actual filename ie/text.txt
        try:
            f = open(actual_filename, 'rb')
            conn.settimeout(1)
            try:
                fileResponse = FileResponse(actual_filename)
                conn.sendall(fileResponse)    # send fileResponse record include actual file
            except socket.timeout as msg:
                print(msg)
                conn.close()
                continue

            f.close()    # close file
            conn.close()
            # print message includes the actual number of bytes transferred
            print("Transfer succeeded\nThe actual number of bytes transferred of {} is {}".format(actual_filename, os.stat(actual_filename).st_size))
            continue

        except IOError as msg:
            if msg.errno == errno.EACCES:
                print("Unable to read the file")
            elif msg.errno == errno.ENOENT:
                print("The file does not exist")
            conn.close()
            continue

def main():
    # attempts a port number
    portNum = int(sys.argv[1])
    # check port number
    checkPort(portNum)
    # create a socket and bind it to the port number given on command line and listen
    s = create_socket(portNum)
    # a connection to communicate with client then open, read file then send the file the client wishes to retrieve
    serverProcessing(s)

main()
```

```python
import socket
import sys
import os

PATH = './client/'

# Sets up FileRequest record included first 5 bytes (fixed header) and n bytes of Filename
def FileRequest(filename):
    """Returns a bytearray of the FileRequest record"""
    byte_array = bytearray()
    # read file as byte(s) (filename)
    filename_b = filename.encode('utf-8')
    # magicNo
    magicNo = 0x497E
    magicNo_1 = (magicNo >> 8)
    magicNo_2 = magicNo & 0xFF
    # Type
    Type = (1).to_bytes(1, byteorder='big')
    # FilenameLen
    filenameLen = len(filename).to_bytes(2, byteorder='big')
    byte_array.append(magicNo_1)
    byte_array.append(magicNo_2)
    byte_array += Type
    byte_array += filenameLen
    byte_array += filename_b

    return byte_array

def checkParameters(host, portNum, fileName):
    """Prints an error if there are more than 5 parameters on the command line"""
    #'', '' are standed for python3 and the name of the file ie/ client.py
    param_list = ['', '', host, portNum, fileName]
    if len(param_list) > 5:
        print("Error caused, there are more than 5 parameters needed")
        sys.exit()    # exit
```

```python

def checkFileResponse(data):
    """Checks magic number, type and statuscode in FileResponse record"""
    try:
        magicNo = (data[0] << 8) + data[1]
        Type = data[2]
        StatusCode = data[3]
        # check if one of these wrong
        if magicNo != 0x497E or Type != 2 or (StatusCode != 0 and StatusCode != 1):
            print("The received FileResponse is erroneous")
    except:
        print("Error occurred while sending FileResponse(fixed header) record from the server")

def checkHost(host, portNum):
    """Checks the validity of the host in form of IP adrress or hostname
       (if it is a hostname, changes to IP address).
       If fails, prints an error message and exit"""
    try:
        if socket.gethostbyname(host) == host:
            pass
        if socket.gethostname() == host:
            addrinfos = socket.getaddrinfo(host, portNum)
            for addr in addrinfos:
                (family, socktype, proto, cannonname, sockaddr) = addr
                host = sockaddr[0]
    except socket.error:
        print("The ip address/hostname does not exist or in a bad form")

    return host

def checkFileName(filename):
    """Checks if the filename wished to retrieve from the server exists locally in client"""
    try:
        if os.path.exists(PATH + filename) != True:
            pass
    except:
        print("Error occurs, the file exists or be opened locally while it should not")
        sys.exit()

def checkPort(portNum):
    """Checks the port number of to use on the server which should be between 1024 and 64000
       otherwise prints an error message and exit"""
    if portNum >= 1024 and portNum <= 64000:
        pass
    else:
        print("Invalid Port Number")
        sys.exit()

# Creates a socket, if does not succeed, prints an error message and exits
# Calls connect() to connect with the server
#if does not succeed, prints an error message, closes socket and exits
def create_socket():
    """Returns a socket connected with the server"""
    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    except socket.error as msg:
        print(msg)
        sys.exit()

    return s

def connect(s, host, portNum):
    """Returns a socket connected with the server"""
    try:
        s.connect((host, portNum))
        print("Connecting to port {}".format(s.getsockname()))
```

```python
    except socket.error as msg:
        print(msg)
        sys.exit()

    return s

def recv_header(s, byte):
    """Returns bytes of the fixed header to check the FileResponse"""
    s.settimeout(1)
    try:
        data = s.recv(byte)
        #check fixed header in FileResponse
        checkFileResponse(data)
    except socket.timeout as msg:
        print("The received FileResponse is erroneous")
        print(msg)
        s.close()    # close socket
        sys.exit()

    return data

def checkFileData(s, data):
    """Checks if there is no file data following by checking
    if the file exists on the server side (StatusCode)"""
    try:
        if data[3] == 1:
            pass
        else:
            print("The file does not exist on server side")
            s.close()
            sys.exit()
    except:
        print("Error occurred while sending FileResponse record from the server")
        s.close()
        sys.exit()

def writeFile(s, fileName, data):
    """Checks if the file can be open then writes file data into the file"""
    received = 0
    dataLength = (data[4] << 24) + (data[5] << 16) + (data[6] << 8) + data[7]
    try:
        with open(fileName, 'wb') as f:
            while received < dataLength:
                s.settimeout(1)
                try:
                    buffer = s.recv(4096)
                except s.timeout as msg:
                    print(msg)
                    s.close()
                    f.close()
                    sys.exit()
                try:
                    f.write(buffer)
                    received += len(buffer)
                except:
                    print("Error occurs, the file is unable to be written")
                    s.close()
                    f.close()
                    sys.exit()
    except:
        print("Unable to open the file for writing")
        s.close()
        sys.exit()
    # check there are exactly as many data bytes as indicated in the 'datalength'
    if received != dataLength:
        print("Error occurs, there are missing bytes occured during sending and receiving the filedata")
    # print actual file size received
    print("Download succeed\nThe actual number bytes received of {} is {}".format(fileName, received))
    s.close()
    sys.exit()
```

```python

def main():
    host = sys.argv[1]
    portNum = int(sys.argv[2])
    fileName = sys.argv[3]
    # check the num of parameters
    checkParameters(host, portNum, fileName)
    # check the port number
    checkPort(portNum)
    # check host name
    host_checked = checkHost(host, portNum)
    # check the fileName
    h = checkFileName(fileName)
    # create a new connection
    socket = create_socket()
    s = connect(socket, host_checked, portNum)
    # prepare for FileRequest and send to the server
    read_byte = FileRequest(fileName)
    s.send(read_byte)
    # receive 8 bytes of the fixed header in the FileResponse record
    data = recv_header(s, 8)
    # check if no file data
    checkFileData(s, data)
    # process the file data (write data to the file)
    writeFile(s, fileName, data)

main()
```

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:	LUU KHANH LINH
Student ID:	68697438
Signature:	[illegible]
Date:	18/08/2019