ASSIGNMENT 1 COSC264

Socket Programming

Name: Christopher Kevin Hamdajani
STUDENT ID : 15995114

```
"""
CLIENT SIDE CODE
Christopher Kevin Hamdajani
15995114
"""
import socket
import sys
import os.path

def check_header(magic_no, type_no, status_code):
    if magic_no != '0x497e':
        print("[HEADER ERROR] Magic number is not 0x497E")
        return False
    elif type_no != 2:
        print("[HEADER ERROR] Type is wrong")
        return False
    elif status_code != 0:
        if status_code != 1:
            print("[HEADER ERROR] Unknown status code")
            return False
    return True

def create_fixed_header(name_file):
    len_msg = len(name_file.encode('utf-8'))
    magic_no = 18814
    type_no = 1
    header_temp = 0xFFFFFFFFFF
    header_temp = (header_temp & 0x0000FFFFFF) | (magic_no << 24)
    header_temp = (header_temp & 0xFFFF00FFFF) | (type_no << 16)
    header_temp = (header_temp & 0xFFFFFF0000) | (len_msg)
    header_hex = hex(header_temp)[2:]
    fixed_header_barray = bytearray.fromhex(header_hex)
    return fixed_header_barray

def main():
    state= True
    if (len(sys.argv) != 4):
        print("[PARAMETER ERROR] Only need 3 paramaeters which are IP address, port number, and name of the file")
        state = False
    else:
        if (int(sys.argv[2]) < 1024 or int(sys.argv[2]) > 64000):
            print("[PARAMETER ERROR] The port number that has been entered was not between 1024 and 64000(including)")
            state = False
        elif os.path.isfile(sys.argv[3]):
            state = False
            print('[FILE ERROR] File already exist')
        try:
            addrInfo = socket.getaddrinfo(sys.argv[1], sys.argv[2])
        except:
            print("[PARAMETER ERROR] Failed to identify ip address!")
            state = False
        else:
            ipaddress = addrInfo[0][4][0]
    if state:
        try:
            client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            client.settimeout(1.0)
        except:
            print("[SOCKET ERROR] Failed to create socket")
            sys.exit()
        else:
            #server try to listen
            try:
                address = (ipaddress, int(sys.argv[2]))
                client.connect(address)
```

```
        except:
            client.close()
            print("[SOCKET ERROR] Socket failed to connect")
            sys.exit()
    fixed_header_barray = create_fixed_header(sys.argv[3])
    client.send(fixed_header_barray)
    client.send(sys.argv[3].encode('utf-8'))
    #begin recv proces
    try:
        fixed_header = client.recv(8)
    except socket.timeout:
        print("[TIMEOUT ERROR] Exceed 1 second time limit when receiving data")
        client.close()
        sys.exit()
    else:
        if(len(fixed_header) != 0):
            hex_fheader = fixed_header.hex()
            temp_list=[hex_fheader[i:i+2] for i in range(0, len(hex_fheader), 2)]
            magic_no = "0x" + temp_list[0] + temp_list[1]
            type_no = int(temp_list[2])
            status_code = int(temp_list[3])
            data_length = int(temp_list[4]+temp_list[5]+temp_list[6]+temp_list[7], 16)
            header_state = check_header(magic_no, type_no, status_code)
            if header_state:
                if (status_code == 0):
                    print("[FILE ERROR] The requested file is not available in the server")
                    client.close()
                    sys.exit()
                else:
                    try:
                        f= open(sys.argv[3], 'wb')
                    except OSerror:
                        print("[FILE ERROR] Unable to open the indicated file")
                        client.close()
                        sys.exit()
                    else:
                        total_bytes = 0
                        total_msg = bytearray()
                        while True:
                            try:
                                msg = client.recv(4096)
                            except socket.timeout:
                                print("[FILE ERROR] The requested file is not available in the server")
                                client.close()
                                sys.exit()
                            else:
                                if len(msg) == 0:
                                    if(total_bytes != data_length):
                                        print("[FILE ERROR] The received file might be corrupted")
                                        client.close()
                                        sys.exit()
                                    else:
                                        f.write(total_msg)
                                        f.close()
                                        print("[SUCCES] The indicated file( {} of bytes) has been downloaded".format(total_bytes))
                                        client.close()
                                        sys.exit()
                                    break
                                else:
                                    total_bytes += len(msg)
                                    total_msg.extend(msg)
        else:
            print("[ERROR] no data is received")
main()
```

```
"""
SERVER SIDE CODE
Christopher Kevin Hamdajani
15995114
"""
import socket
import sys
import datetime
import os.path
SERVER = socket.gethostbyname(socket.gethostname())
def check_port(port_num):
    #check if port_num is in range between 1024 and 64000(including)
    if port_num < 1024 or port_num > 64000:
        return False
    return True

def create_file_response(status_code, filename = None):
    magic_no = 18814
    type_no = 2
    data_length = None
    if status_code == 0:
        data_length = 0
        header_temp = 0xFFFFFFFFFFFFFFFF
        header_temp = (header_temp & 0x0000FFFFFFFFFFFF) | (magic_no << 48)
        header_temp = (header_temp & 0xFFFF00FFFFFFFFFF) | (type_no << 40)
        header_temp = (header_temp & 0xFFFFFF00FFFFFFFF) | (status_code << 32)
        header_temp = (header_temp & 0xFFFFFFFF00000000) | (data_length)
        header_hex = hex(header_temp)[2:]
        fixed_header_barray = bytearray.fromhex(header_hex)
        return fixed_header_barray
    else:
        filedata = bytearray()
        try:
            with open(filename, "rb") as f:
                byte = f.read()
            filedata.extend(byte)
        except IOError:
            print('Error While Opening the file!')
        data_length = len(filedata)
        header_temp = 0xFFFFFFFFFFFFFFFF
        header_temp = (header_temp & 0x0000FFFFFFFFFFFF) | (magic_no << 48)
        header_temp = (header_temp & 0xFFFF00FFFFFFFFFF) | (type_no << 40)
        header_temp = (header_temp & 0xFFFFFF00FFFFFFFF) | (status_code << 32)
        header_temp = (header_temp & 0xFFFFFFFF00000000) | (data_length)
        header_hex = hex(header_temp)[2:]
        fixed_header_barray = bytearray.fromhex(header_hex)
        full_barray = fixed_header_barray+(filedata)
        print("[FILE SENT] Actual number of bytes transfered :", len(full_barray))
        return full_barray

def check_header(magic_no, type_no, filenamelen):
    if magic_no != '0x497e':
        print("[HEADER ERROR] Magic number is not 0x497E")
        return False
    elif type_no != 1:
        print("[HEADER ERROR] type is wrong")
        return False
    elif filenamelen < 1 or filenamelen > 1024:
        print("[HEADER ERROR] The length of filename is not between between 1 and 1024(including)")
        return False
    return True

def logging(addr):
    ip, port = addr
    print("[INCOMING CONNECTION]")
```

```
    print("Current time : ", datetime.datetime.now())
    print("IP address : ", ip)
    print("Port number : ", port)

def main():
    port_num = int(sys.argv[1])
    connected = True
    if check_port(port_num):
        #create socket
        new_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        address = (SERVER, port_num)
        #server try to bind
        try:
            new_socket.bind(address)
        except:
            conected = False
            print("[BINDING ERROR] Failed to bind!")
            sys.exit()
        else:
            #server try to listen
            try:
                new_socket.listen()
            except:
                conected = False
                print("[LISTEN ERROR] Failed to listen")
                new_socket.close()
                sys.exit()
        if connected:
            print("[SERVER LISTENING] server is running on ", SERVER, " on port ", port_num)
        while connected:
            conn, addr = new_socket.accept()
            logging(addr)
            conn.settimeout(1.0)
            state_timeout = True
            try:
                file_request = conn.recv(2048)
            except socket.timeout:
                state_timeout = False
                print("[TIMEOUT ERROR] Exceed 1 second time limit when receiving data")
            fixed_header = file_request[:5]
            hex_fheader = fixed_header.hex()
            temp_list=[hex_fheader[i:i+2] for i in range(0, len(hex_fheader), 2)]
            magic_no = "0x" + temp_list[0] + temp_list[1]
            type_no = int(temp_list[2])
            filenamelen = int('0x'+ temp_list[3] + temp_list[4], 16)
            header_state = check_header(magic_no, type_no, filenamelen)
            if header_state == False or state_timeout == False:
                conn.close()
            else:
                filename = file_request[5:].decode('utf-8')
                if os.path.isfile(filename):
                    try:
                        infile = open(filename)
                        infile.close()
                    except OSerror:
                        print("[FILE ERROR] Cannot open the file")
                        response_barray = create_file_response(0)
                        conn.send(response_barray)
                        conn.close()
                    else:
                        response_barray = create_file_response(1, filename)
                        conn.send(response_barray)
                        conn.close()
                else:
                    response_barray = create_file_response(0)
```

```
                conn.send(response_barray)
                conn.close()
                print("[FILE NOT FOUND] The file requested is not available!")
    else:
        print("[PORT NUMBER ERROR] The port number that has been entered was not between 1024 and 64000(including)")
        sys.exit()


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: CHRISTOPHER KEVIN HAMDAJANI

Student ID: 15995114

Signature: 

Date: 31/08/2021