COSC264

-Socket Programming Assignment-

Vikas Shenoy

53329413

```python
"""
Client program for communication via UDP sockets. Takes a port, host and request
type. Sends the server a request for either date or time, and then receives a
response packet. Prints the payload  containing the desired information.
Author: Vikas Shenoy vsh33
Date: 16/8/2018
"""
import socket
import select
from sys import argv



def isValidResponse(packet):
    """Checks the response packet, ensuring the packet has the correct size
    and header. Takes the response packet. Returns a boolean of whether the
    packet is valid or not."""
    valid = True
    magicNumber = packet[0] << 8 | packet[1]
    packetType = packet[2] << 8 | packet[3]
    languageCode = packet[4] << 8 | packet[5]
    year = packet[6] << 8 | packet[7]
    month = packet[8]
    day = packet[9]
    hour = packet[10]
    minute = packet[11]
    length = packet[12]

    # Error checking for response packet
    if len(packet) < 13:
        print("Response packet must be at least 13 bytes long")
        valid = False
    elif magicNumber != 0x497E:
        print("Response pkt must have magic no 0x497E")
        valid = False
    elif packetType != 0x0002:
        print("Response pkt must have pkt type 0x0002")
        valid = False
    elif languageCode < 1 or languageCode > 3:
        print("Language code must be 1,2, or 3")
        valid = False
    elif year > 2100:
        print("Year must be below 2100")
        valid = False
    elif month < 1 or month > 12:
        print("Month must be between 1 and 12")
        valid = False
    elif day < 1 or day > 31:
        print("Day must be between 1 and 31")
        valid = False
    elif hour < 0 or hour > 23:
        print("Hour must be between 0 and 23")
        valid = False
    elif minute < 0 or minute > 59:
        print("Minute must be between 0 and 59")
        valid = False
    elif (len(packet) - 13) != length:
        print("Length field does not equal payload length")
        valid = False

    return valid



def constructReqPkt(reqType):
    """Constructs a request packet to be sent to the server. Takes a string
    reqType which is filled in the packet header, indicating date or time.
    Returns the bytearray request packet."""
    reqPkt = bytearray(6)
    magicNumber = 0x497E
    packetType = 0x0001
    bufferSize = 1024
    timeLimit = 1
    if reqType == "date":
        requestType = 0x0001
    else:
        requestType = 0x0002

    reqPkt[0] = (magicNumber >> 8) & 0xff
    reqPkt[1] = magicNumber & 0xff
    reqPkt[2] = (packetType >> 8) & 0xff
    reqPkt[3] = packetType & 0xff
    reqPkt[4] = (requestType >> 8) & 0xff
    reqPkt[5] = requestType & 0xff
    return reqPkt



def printPacket(packet):
    """Takes the server's response packet and prints the contents, formatted."""
    magicNumber = packet[0] << 8 | packet[1]
    packetType = packet[2] << 8 | packet[3]
    languageCode = packet[4] << 8 | packet[5]
    year = packet[6] << 8 | packet[7]
    month = packet[8]
    day = packet[9]
    hour = packet[10]
    minute = packet[11]
    length = packet[12]
    text = packet[13:].decode()

    print("Printing response packet contents...")
    print("Magic number = {}".format(hex(magicNumber)))
    print("Packet type = {}".format(packetType))
    print("Language code = {}".format(languageCode))
    print("Year = {}".format(year))
    print("Month = {}".format(month))
    print("Day = {}".format(day))
    print("Hour = {}".format(hour))
    print("Minute = {}".format(minute))
    print("Length = {}".format(length))
    print("Text = {}".format(text))



def receivePacket(client, timeLimit, bufferSize):
    """Takes the socket, time limit for how long to wait for a response message
    and size of the socket buffer. Either prints an error message or prints
    the packet contents."""
    readList, writeList, exception = select.select([client], [], [], timeLimit)
    if len(readList) > 0:
        responsePacket, addr = client.recvfrom(bufferSize)
        packetValid = isValidResponse(responsePacket)
        if packetValid:
            print("Response packet received")
            printPacket(responsePacket)
        else:
            print("Packet not valid")
            return
    else:
        print('Response packet was not received within {} seconds'.format(timeLimit)
)
        return



def runClient(requestType, host, port):
    """Sends a request packet to a server for either date or time. Returns the
    response packet payload if received within the timelimit, otherwise returns
    an error message."""
    timeLimit = 1
    bufferSize = 1024
    # Error checking for parameters
    if requestType != "date" and requestType != "time":
        print("Request type must be for date or time.")
        return
    elif port < 1024 or port > 64000:
        print("Port num must be between 1024 and 64000")
        return
    try:
        addrInfo = socket.getaddrinfo(host, port)
        serverAddress = addrInfo[0][4]
    except:
        print('Invalid host')
        return
    # Construct a request packet to be sent to the server
    reqPkt = constructReqPkt(requestType)
    client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    client.sendto(reqPkt, serverAddress)
    print("Request packet sent")
    # Receive the response packet from the server and prints the contents
    receivePacket(client, timeLimit, bufferSize)



def main():
    name, reqType, IP, port = argv
    runClient(reqType, IP, int(port))
main()
```

```python
"""
Server program for communication via UDP sockets. Takes three port numbers to
bind sockets to and receive communication on. Waits for a request packet from
the client and then rsends a response packet back with appropriate information.
Author: Vikas Shenoy vsh33
Date: 16/8/2018
"""
import socket
import select
import sys
import datetime
from sys import argv



def isValid(packet):
    """Takes a bytearray request packet and checks if it's valid. Returns
    a boolean true if valid, false if not. Prints the packet contents."""

    valid = True
    magicNumber = packet[0] << 8 | packet[1]
    packetType = packet[2] << 8 | packet[3]
    requestType = packet[4] << 8 | packet[5]
    length = len(packet)

    if length != 6:
        print("Request packet must be 6 bytes long.")
        valid = False
    elif magicNumber != 0x497E:
        print("Magic number must be 0x497E")
        valid = False
    elif packetType != 0x0001:
        print("Packet type must equal 0x0001")
        valid = False
    elif requestType > 2 or requestType < 0:
        print("Request type must equal 0x0001 or 0x0002")
        valid = False

    # Print packet contents for debugging purposes
    print("Request packet contents: ")
    print("Magic number = {}".format(hex(magicNumber)))
    print("Packet type = {}".format(packetType))
    print("Request type = {}".format(requestType))
    return valid



def getMonthName(monthNum, languageCode):
    """Takes the current month number and language requested by the user.
    Returns the month's proper noun name."""
    monthNameList = [None,
                     [None, 'January', 'February', 'March', 'April', 'May',
                      'June', 'July', 'August', 'September', 'October',
                      'November', 'December'],
                     [None, 'Kohit¿tea', 'Hui-tanguru', 'Poutu-te-rangi',
                      'Paenga-whawha', 'Haratua', 'Pipiri', 'Hongongoi',
                      'Here-tuki-koka', 'Mahuru', 'Whiringa-a-nuku',
                      'Whiringa-a-rangi', 'Hakihea'],
                     [None, 'Januar', 'Februar', 'Marz', 'April', 'Mai', 'Juni',
                      'Juli', 'August', 'September', 'Oktober', 'November',
                      'Dezember']]
    return monthNameList[languageCode][monthNum]



def constructResponse(requestType, languageCode):
    """Takes a request type (date or time), and language code. Returns
    an appropriate response packet to send back to the client."""
    # Constructs the response packet header
    packet = bytearray(13)
    dateTime = datetime.datetime.now()
    magicNumber = 0x497E
    packetType = 0x0002
    year = dateTime.year
    monthNum = dateTime.month
    day = dateTime.day
    hour = dateTime.hour
    minute = dateTime.minute

    packet[0] = (magicNumber >> 8) & 0xff
    packet[1] = magicNumber & 0xff
    packet[2] = (packetType >> 8) & 0xff
    packet[3] = packetType & 0xff
    packet[4] = (languageCode >> 8) & 0xff
    packet[5] = languageCode & 0xff
    packet[6] = (year >> 8) & 0xff
    packet[7] = year & 0xff
    packet[8] = monthNum
    packet[9] = day
    packet[10] = hour
    packet[11] = minute
    packet[12] = 0

    message = ''
    monthName = getMonthName(monthNum, languageCode)
    # date requested
    if requestType == 0x0001:
        if languageCode == 0x0001:
            message = "Today's date is {} {}, {}".format(monthName, day, year)
        elif languageCode == 0x0002:
            message = "Ko te ra o tenei ra ko {} {}, {}".format(monthName
                                                                , day, year)
        else:
            message = "Heute ist der {}. {} {}".format(day, monthName, year)
    # time requested
    else:
        if languageCode == 0x0001:
            message = "The current time is {0}:{1:02d}".format(hour, minute)
        elif languageCode == 0x0002:
            message = "Ko te wa o tenei wa {0}:{1:02d}".format(hour, minute)
        else:
            message = "Die Uhrzeit ist {0}:{1:02d}".format(hour, minute)

    # Add the message and return the packet
    messageBytes = message.encode('utf-8')
    packet[12] = len(messageBytes)
    for byte in messageBytes:
        packet.append(byte)
    return packet



def serverLoop(s1, s2, s3, p1, p2, p3):
    """Takes three sockets and three port numbers. Waits for a request packet
    from the client. If valid, returns a response packet with the correct
    info."""
    bufferSize = 1024
    while True:
        print("Server waiting for request packets...")
        readList, writeList, error = select.select([s1, s2, s3], [], [], None)
        if len(readList) > 0:
            clientSkt = readList[0]
            requestPkt, addr = clientSkt.recvfrom(bufferSize)
            packetValid = isValid(requestPkt)
            clientPort = clientSkt.getsockname()[1]

            # Construct and send the response packet
            if packetValid:
                requestType = requestPkt[4] << 8 | requestPkt[5]
                languageCode = 0x0001
                if clientPort == p1:
                    languageCode = 0x0001
                elif clientPort == p2:
                    languageCode = 0x0002
                elif clientPort == p3:
                    languageCode = 0x0003
                responsePkt = constructResponse(requestType, languageCode)
                clientSkt.sendto(responsePkt, addr)
                print("Reponse packet sent")
            else:
                print("Packet is invalid and has been discarded")



def runServer(port1, port2, port3):
    """Takes three port numbers which are checked for correctness. Creates
    sockets which are bound to these ports, then starts the server loop."""
    # Port numbers checked for correcteness
    if (port1 == port2 or port1 == port3 or port2 == port3):
        print("Port numbers must be unique.")
        return
    elif  (port1 < 1024 or port2 < 1024 or port3 < 1024):
        print("Port numbers must be between 1024 and 64000")
        return
    elif  (port1 > 64000 or port2 > 64000 or port3 > 64000):
        print("Port number must be between 1024 and 64000.")
        return

    # Create sockets and bind to the given ports
    skt1 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    skt2 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    skt3 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    try:
        skt1.bind(('', port1))
        skt2.bind(('', port2))
        skt3.bind(('', port3))
    except:
        print("Failed to bind sockets to given port numbers")
        return
    serverLoop(skt1, skt2, skt3, port1, port2, port3)



def main():
    name, port1, port2, port3 = argv
    runServer(int(port1), int(port2), int(port3))
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:	Vikas Shenoy
Student ID:	~~[illegible]~~ 53329413
Signature:	[illegible]
Date:	21/8/2018