COSC264 Assignment 1

Socket Programming

Name: Richard Vong

Student ID: 67436350

```
    1   import socket
    2   import datetime
    3   import sys
    4   import os
    5   from os import path
    6
    7   def less_than_three():
    8       if len(sys.argv) < 4 or len(sys.argv) > 4:
    9           sys.exit("ERROR: LESS OR MORE THAN THREE ↵
        ↳       PARAMETERS")
   10       else:
   11           pass
   12
   13   def get_ip():
   14       address = str(sys.argv[1])
   15       try:
   16           ip_address = socket.gethostbyname(address)
   17       except socket.gaierror:
   18           sys.exit("ERROR: HOST NAME DOES NOT EXIST OR IP ↵
        ↳       IS NOT WELL-FORMATTED")
   19       print("IP IS VALID") #just to check
   20       return address
   21
   22   def get_port():
   23       port = int(sys.argv[2])
   24       if port < 1024 or port > 64000:
   25           sys.exit("ERROR: PORT NUMBER MUST BE BETWEEN ↵
        ↳       1024 AND 64000 (INCLUSIVE)")
   26       else:
   27           print("PORT IS VALID") #just to check
   28       return port
   29
   30   def name_of_file():
   31       filename = str(sys.argv[3])
   32       if path.exists(filename):
   33           sys.exit("ERROR: FILE ALREADY EXISTS LOCALLY")
   34       else:
   35           print("FILE DOES NOT EXIST") #just to check
   36           return filename
```

```
37
38  def create_socket():
39      try:
40          s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #AF_INT is adress for IPV4, #SOCK_STREAM is socket type for TCP
41      except socket.error:
42          sys.exit("ERROR: FAILED TO CREATE SOCKET")
43      print("SOCKET CREATION SUCCESS") #just to check
44      return s
45
46  def connect(s, HOST, PORT):
47      try:
48          s.connect((HOST, PORT))
49      except socket.error:
50          s.close()
51          sys.exit("ERROR: CONNECT FAILURE")
52      print("CONNECTION SUCCESS") #just to check
53
54  def file_request(filename):
55      file = bytearray()
56      MagicNo = (0x497E).to_bytes(2, byteorder='big') #equivalent to 0x497E
57      Type = (1).to_bytes(1, byteorder='big')
58      FilenameLen = (len(filename)).to_bytes(2, byteorder='big')
59      Encoded_filename = filename.encode('utf-8') #returns utf-8 encoded version of the string
60
61      return bytearray(MagicNo + Type + FilenameLen + Encoded_filename)
62
63  def recieve_data(s):
64      recieved_data = s.recv(4096)
65      return recieved_data
66
67  def read_fixed_header(s, filename):
68      s.settimeout(1)
69      try:
```

```
   70           data = s.recv(8)
   71       except socket.timeout:
   72           print("ERROR: CONNECTION TIMEOUT")
   73           s.close()
   74           sys.exit()
   75
   76       MagicNo = (int).from_bytes(data[0:2], "big")
   77       Type = data[2]
   78       StatusCode = data[3]
   79       DataLength = (int).from_bytes(data[4:], "big")
   80
   81       FixedHeader = MagicNo + Type + StatusCode + DataLength
   82
   83       if MagicNo == 0x497E and Type == 2 and (StatusCode ==↵
        ↳    1 or StatusCode == 0):
   84           print("CONDITIONS ARE CORRECT")
   85           pass
   86       else:
   87           print("ERROR: FILE REQUEST IS ERRONEOUS")
   88           s.close()
   89
   90       if StatusCode == 0:
   91           print("ERROR: FILE DOES NOT EXIST ON SERVER SIDE")
   92           s.close()
   93           sys.exit()
   94       else:
   95           try:
   96               f = open(filename, "wb+") # wb+  =  create ↵
        ↳          write bytes
   97           except IOError:
   98               print("ERROR: FILE CANNOT BE OPENED FOR ↵
        ↳          WRITING")
   99               s.close()
  100               sys.exit()
  101
  102           DataLength_recieved = 0 #initialise
  103           while True:
  104               try:
  105                   f_data = s.recv(4096) #buffer
```

```
106                 except IOError:
107                     print("ERROR: FIXED HEADER IS ERRONEOUS, CONNECTION TIMEOUT")
108                     s.close()
109                     f.close()
110                     sys.exit()
111                 except socket.error:
112                     print("ERROR: FILE DATA CANNOT BE RECIEVED FROM SERVER")
113                     s.close()
114                     f.close()
115                     sys.exit()
116                 byte_array = bytearray(f_data)
117                 try:
118                     f.write(f_data)
119                 except IOError:
120                     print("ERROR WRITING TO FILE")
121                     s.close()
122                     f.close()
123                     sys.exit()
124                 if not f_data:
125                     break
126                 DataLength_recieved += len(f_data)
127             if DataLength_recieved != DataLength:
128                 print("ERROR: DATA BYTES VALID")
129                 s.close()
130                 sys.exit()
131 
132             print("FILE RECIEVED")
133             print("THE NUMBER OF BYTES RECIEVED IS: {}".format(DataLength_recieved))
134             f.close()
135             sys.exit()
136 
137 
138     def main():
139         less_than_three()
140         address = get_ip()
141         port = get_port()
```

```
  142       filename = name_of_file()
  143       s = create_socket()
  144       connect(s, address, port)
  145       fileRequest = file_request(filename)
  146       s.sendall(fileRequest)
  147       read_fixed_header(s, filename)
  148       recieve_data(s)
  149
  150   main()
```

/home/cosc/student/rvo16/Documents/Cosc264/Assignment - Socket/Server/server.py

```
 1  import socket
 2  import datetime
 3  import sys
 4  import os
 5
 6  HOST = '0.0.0.0'
 7
 8  def get_port():
 9      PORT = int(sys.argv[1])
10      if PORT < 1024 or PORT > 64000:
11          sys.exit("ERROR: PORT NUMBER MUST BE BETWEEN 1024 AND 64000 (INCLUSIVE)")
12      else:
13          print('PORT IS VALID') #just to check
14          return PORT
15
16  def create_and_bind(PORT):
17      s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
18      try:
19          s.bind((HOST, PORT))
20      except Exception as e:
21          print("Error {}".format(e))
22          sys.exit("ERROR: SOCKET CREATION IS BAD")
23      print("CREATE AND BIND SUCCESS") #just to check
24      return s
25
26  def listen(s):
27      try:
28          s.listen()
29      except socket.error:
30          s.close()
31          sys.exit("ERROR: LISTENING FAILURE")
32      print("LISTENING...")  #just to check
33
34      while True:
35          connection_socket, address = s.accept()
36          now = datetime.datetime.now()
37          print(now.strftime("Time: %H:%M:%S")) #current time
38          ip_address, port_number = address
```

```
39          print('Connected by IP adress:{} and Port number:{}'.format(ip_address, port_number))
40
41          #=========READ FIXED HEADER==============
42          connection_socket.settimeout(1)
43          try:
44              data = connection_socket.recv(5)
45          except socket.timeout:
46              print("ERROR: CONNECTION TIMEOUT. RESTARTING LOOP")
47              connection_socket.close()
48              continue #goes back to the start of the loop
49
50          #==============VALIDATING DATA============
51
52          MagicNo = data[0] << 8 | data[1]
53          Type = data[2]
54          FilenameLen = data[3] << 8 | data[4]
55          if MagicNo == 0x497E and Type == 1 and FilenameLen > 1 and FilenameLen < 1024:
56              print("CONDITIONS ARE CORRECT")
57              pass
58          else:
59              print("ERROR: FILE REQUEST IS ERRONEOUS")
60              connection_socket.close()
61              continue
62
63          #========READING MORE BYTES FOR FILENAME============
64
65          connection_socket.settimeout(1)
66          try:
67              filename_data = connection_socket.recv(FilenameLen)
68          except socket.timeout:
69              print("ERROR: CONNECTION TIMEOUT. RESTARTING LOOP")
70              connection_socket.close()
71              continue #goes back to the start of the loop
```

```
72
73          #======OPEN FILE FOR READING===========
74          requested_filename = filename_data.decode('utf-8')
75
76          try:
77              f = open(requested_filename, 'rb') #rb means to 'read bytes'
78              print("FILE EXISTS AND CAN BE OPENED") #just to check
79
80              MagicNo_Response = (0x497E).to_bytes(2, byteorder='big')
81              Type_Response = (2).to_bytes(1, byteorder='big')
82              StatusCode = (1).to_bytes(1, byteorder='big')
83
84              cwd = os.getcwd()
85              DataLength = os.path.getsize(cwd + '/' + str(requested_filename))
86              DataLength = DataLength.to_bytes(4, byteorder='big')
87
88              header = bytearray(MagicNo_Response + Type_Response + StatusCode + DataLength)
89              connection_socket.send(header)
90              DataLength_sent = 0
91
92          except IOError:
93              MagicNo_Response = (0x497E).to_bytes(2, byteorder='big')
94              Type_Response = (2).to_bytes(1, byteorder='big')
95              StatusCode = (0).to_bytes(1, byteorder='big')
96              DataLength = (0).to_bytes(0, byteorder='big')
97              # StatusCode is 0 so FileData field contains no bytes.
98
99              header = bytearray(MagicNo_Response +
```

```
            Type_Response + StatusCode + DataLength)
100             connection_socket.send(header)
101
102             print("ERROR: FILE DOES NOT EXIST OR CANNOT BE OPENED")
103             connection_socket.close()
104             continue
105
106         while True:
107             f_data = f.read(4096)
108             connection_socket.send(f_data)
109             if len(f_data) == 0:
110                 break
111             DataLength_sent += len(f_data)
112         print("THE NUMBER OF BYTES TRANSFERED IS: {}".format(DataLength_sent))
113         connection_socket.close()
114         continue
115
116
117 def main():
118     port = get_port()
119     s = create_and_bind(port)
120     listen(s)
121
122
123 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: Richard Vang

Student ID: 67436350

Signature: 

Date: 18/08/19