**Documentation**

\_\_author\_\_ = 'Dhanashri Ostwal'

from sys import argv

import sys

import socket

import os

import thread

CRLF = "\r\n"

# returing content\_type according to requested file type

def content\_type(fileName):

if(os.path.splitext(fileName)[1] == ".html" or os.path.splitext(fileName)[1] == ".htm"):

return "text/html"

else:

return "application/octet-stream"

#creating logs

def write\_log(text):

with open('log.txt','a') as f:

f.write(str(text + '\n'))

f.closed

def handle\_connection(connectionSocket,addr):

try:

#printing details about the client

print ("\nserving client: ", addr)

ltext = "\nserving client: " + str(addr)

write\_log(ltext)

print ("client details :")

ltext = "client details :"

write\_log(ltext)

print ("hostname :",socket.gethostname())

ltext = "hostname :" + socket.gethostname()

write\_log(ltext)

myfile = connectionSocket.recv(4096)

print ("myfile: ", myfile)

line = str(myfile )

print ("line: ", line)

#getting http request header lines

print ("\nrequest header :\n")

ltext = "\nrequest header :\n"

write\_log(ltext)

if(line != ""):

httpMethod = line.split()[0]

fileName = ""+line.split()[1].replace('http://','')

fileName = fileName.replace('/','')

print ("fileName:", fileName)

ltext = "fileName:" + str(fileName)

write\_log(ltext)

print ("httpMethod: ", httpMethod)

ltext = "httpMethod: " + str(httpMethod)

write\_log(ltext)

fileServerName = line.split()[1].partition("/")[2]

fileServerName = fileServerName.replace('/','')

print ("fileServerName:", fileServerName)

ltext = "fileServerName:" + str(fileServerName)

write\_log(ltext)

filetouse = "/" + fileServerName

if(httpMethod == "GET" or httpMethod == "get"):

if os.path.isfile(fileName):

#sending requested file

file = open(filetouse[1:], "r")

data = file.readlines()

print ("File Present in Cache\n")

ltext = "File Present in Cache\n"

write\_log(ltext)

#Proxy Server Will Send A Response Message

#Proxy Server Will Send Data

for i in range(0, len(data)):

print (data[i])

connectionSocket.send(data[i])

print ("Reading file from cache\n")

else:

#File not found in cache, so creating on proxy server

ltext = "File not present in the cache"

write\_log(ltext)

print ("Creating socket on proxy server")

ltext = "Creating socket on proxy server"

write\_log(ltext)

c = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

hostn = fileServerName.replace("www.", "", 1)

print ("Host Name: ", hostn)

ltext = "Host Name: " + str(hostn)

write\_log(ltext)

try:

c.connect(( hostn, 80 ))

print ("Socket connected to port 80 of host")

ltext = "Socket connected to port 80 of host"

write\_log(ltext)

fileobj = c.makefile('r', 0)

fileobj.write("GET " + "http://" + fileServerName + " HTTP/1.0\n\n")

buff = fileobj.readlines()

tmpFile = open(fileName, "wb")

for i in range(0, len(buff)):

tmpFile.write(buff[i])

connectionSocket.send(buff[i])

tmpFile.close()

except:

print ("Illegal request")

ltext = "Illegal request"

write\_log(ltext)

else:

# handling bad request

print ("Bad request")

ltext = "Bad request"

write\_log(ltext)

statusLine = "HTTP/1.1 400 Bad Request" + CRLF;

contentTypeLine = "Content-type: " + content\_type(fileName) + CRLF;

entityBody = "<HTML>" + "<HEAD><TITLE>Bad Request</TITLE></HEAD>" + "<BODY>Not Found</BODY></HTML>";

connectionSocket.send(statusLine)

connectionSocket.send(contentTypeLine)

connectionSocket.send(CRLF)

connectionSocket.send(entityBody)

#freeing resources

connectionSocket.close()

print ("connection closed for client :", addr)

ltext = "connection closed for client :" + str(addr)

write\_log(ltext)

except socket.gaierror, value:

print (value[1])

ltext = value[1]

write\_log(ltext)

except socket.error, value:

print (value[1])

ltext = value[1]

write\_log(ltext)

except ValueError as e:

print (e)

ltext = e

write\_log(ltext)

except IndexError as e:

print (e)

ltext = e

write\_log(ltext)

def main():

try:

try:

#Entering Valid port number

port = int(argv[1])

if(port <=1024 or port >=65536):

print ("Invalid port numnber: please provide port number between 1024 and 65536")

ltext = "Invalid port numnber: please provide port number between 1024 and 65536"

write\_log(ltext)

sys.exit()

else:

serverPort= port

except IndexError:

serverPort = 9090

#creating serverSocket

serverSocket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

#binding Server socket to specified port

serverSocket.bind(('', serverPort))

#listening on Server socket for clients

serverSocket.listen(5)

print ("Server is ready to receive request on port : ", serverPort)

#writing logs

ltext = "Server is ready to receive request on port : "+ str(serverPort)

write\_log(ltext)

while(True):

#accepting new connection

connectionSocket, addr = serverSocket.accept()

#handling new client in new thread

thread.start\_new\_thread(handle\_connection,(connectionSocket,addr))

#socket closed

serverSocket.close()

except socket.gaierror, value:

print (value[1])

ltext = value[1]

write\_log(ltext)

except socket.error, value:

print (value[1])

ltext = value[1]

write\_log(ltext)

if \_\_name\_\_ == "\_\_main\_\_":

main()