FEU Institute of Technology

(INTEGRATIVE PROGRAMMING AND TECHNOLOGIES)

EXERCISE

# 9

(Network Programming)

| Name of Student | Name of Professor |
|---|---|
| Justine Guillermo | |
| Data Performed | Date Submitted |
| | |

**SERVER**

```
# server.py
import socket
import time

# create a socket object
serversocket = socket.socket(
                socket.AF_INET, socket.SOCK_STREAM)

# get local machine name
host = socket.gethostname()

port = 9999

# bind to the port
serversocket.bind((host, port))

# queue up to 5 requests
serversocket.listen(5)

while True:
    # establish a connection
    clientsocket,addr = serversocket.accept()

    print("Got a connection from %s" % str(addr))
    currentTime = time.ctime(time.time()) + "\r\n"
    clientsocket.send(currentTime.encode('ascii'))
    clientsocket.close()
```

**CLIENT**

```
# client.py
import socket

# create a socket object
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# get local machine name
host = socket.gethostname()

port = 9999

# connection to hostname on the port.
s.connect((host, port))

# Receive no more than 1024 bytes
tm = s.recv(1024)

s.close()

print("The time got from the server is %s" % tm.decode('ascii'))
```

**OUTPUT**

```
C:\Python27>python.exe server.py &
Got a connection from ('192.168.0.11', 59519)
```

```
...
====================== RESTART: C:\Python27\client.py ======================
The time got from the server is Mon Apr 03 19:59:40 2017
```

**SERVER**

```
# echo_server.py
import socket

host = ''         # Symbolic name meaning all available interfaces
port = 12345      # Arbitrary non-privileged port
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((host, port))
s.listen(1)
conn, addr = s.accept()
print('Connected by', addr)
while True:
    data = conn.recv(1024)
    if not data: break
    conn.sendall(data)
conn.close()
```

**CLIENT**

```
# echo_client.py
import socket

host = socket.gethostname()
port = 12345                    # The same port as used by the server
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))
s.sendall(b'Hello, world')
data = s.recv(1024)
s.close()
print('Received', repr(data))
```

**OUTPUT**

```
C:\Python27>python.exe echo_server.py &
('Connected by', ('192.168.0.11', 59633))
```

```
========== RESTART: C:/Python27/echo_client.py ==========
('Received', "'Hello, world'")
```

CHAT APPLICATION

```
import sys
import socket
import select

HOST = ''
SOCKET_LIST = []
RECV_BUFFER = 4096
PORT = 9009

def chat_server():

    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    server_socket.bind((HOST, PORT))
    server_socket.listen(10)

    # add server socket object to the list of readable connections
    SOCKET_LIST.append(server_socket)

    print "Chat server started on port " + str(PORT)
```

```
while 1:

    # get the list sockets which are ready to be read through select
    # 4th arg, time_out  = 0 : poll and never block
    ready_to_read,ready_to_write,in_error = select.select(SOCKET_LIST,[],[],0)

    for sock in ready_to_read:
        # a new connection request recieved
        if sock == server_socket:
            sockfd, addr = server_socket.accept()
            SOCKET_LIST.append(sockfd)
            print "Client (%s, %s) connected" % addr

            broadcast(server_socket, sockfd, "[%s:%s] entered our chatting room\n" % addr)

        # a message from a client, not a new connection
        else:
            # process data recieved from client,
            try:
                # receiving data from the socket.
                data = sock.recv(RECV_BUFFER)
                if data:
                    # there is something in the socket
                    broadcast(server_socket, sock, "\r" + '[' + str(sock.getpeername()) + '] ' + data)
                else:
                    # remove the socket that's broken
                    if sock in SOCKET_LIST:
                        SOCKET_LIST.remove(sock)

                    # at this stage, no data means probably the connection has been broken
                    broadcast(server_socket, sock, "Client (%s, %s) is offline\n" % addr)

            # exception
            except:
                broadcast(server_socket, sock, "Client (%s, %s) is offline\n" % addr)
                continue

server_socket.close()
```

```
# broadcast chat messages to all connected clients
def broadcast (server_socket, sock, message):
    for socket in SOCKET_LIST:
        # send the message only to peer
        if socket != server_socket and socket != sock :
            try :
                socket.send(message)
            except :
                # broken socket connection
                socket.close()
                # broken socket, remove it
                if socket in SOCKET_LIST:
                    SOCKET_LIST.remove(socket)

if __name__ == "__main__":

    sys.exit(chat_server())
```

CLIENT CODE

```
import sys, socket, select

def chat_client():
    if(len(sys.argv) < 3) :
        print 'Usage : python chat_client.py hostname port'
        sys.exit()

    host = sys.argv[1]
    port = int(sys.argv[2])

    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.settimeout(2)

    # connect to remote host
    try :
        s.connect((host, port))
    except :
        print 'Unable to connect'
        sys.exit()

    print 'Connected to remote host. You can start sending messages'
    sys.stdout.write('[Me] '); sys.stdout.flush()
```

```
while 1:
    socket_list = [sys.stdin, s]

    # Get the list sockets which are readable
    read_sockets, write_sockets, error_sockets = select.select(socket_list , [], [])

    for sock in read_sockets:
        if sock == s:
            # incoming message from remote server, s
            data = sock.recv(4096)
            if not data :
                print '\nDisconnected from chat server'
                sys.exit()
            else :
                #print data
                sys.stdout.write(data)
                sys.stdout.write('[Me] '); sys.stdout.flush()

        else :
            # user entered a message
            msg = sys.stdin.readline()
            s.send(msg)
            sys.stdout.write('[Me] '); sys.stdout.flush()
```

SERVER TERMINAL

```
// server terminal
$ python chat_server.py
Chat server started on port 9009
Client (127.0.0.1, 48952) connected
Client (127.0.0.1, 48953) connected
Client (127.0.0.1, 48954) connected


// client 1 terminal
$ python chat_client.py localhost 9009
Connected to remote host. You can start sending messages
[Me] [127.0.0.1:48953] entered our chatting room
[Me] [127.0.0.1:48954] entered our chatting room
[Me] client 1
[('127.0.0.1', 48953)] client 2
[('127.0.0.1', 48954)] client 3
[Me] Client (127.0.0.1, 48954) is offline
[Me]
```

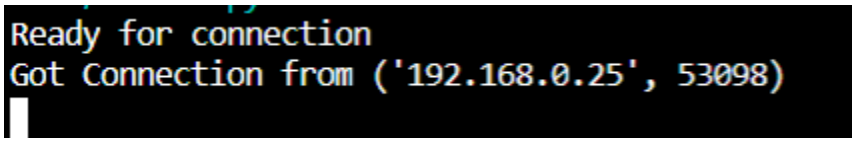
```
// client 2 terminal
$ python chat_client.py localhost 9009
Connected to remote host. You can start sending messages
[Me] [127.0.0.1:48953] entered our chatting room
[Me] [127.0.0.1:48954] entered our chatting room
[Me] client 1
[('127.0.0.1', 48953)] client 2
[('127.0.0.1', 48954)] client 3
[Me] Client (127.0.0.1, 48954) is offline
[Me]

// client 3 terminal
$ python chat_client.py localhost 9009
Connected to remote host. You can start sending messages
[('127.0.0.1', 48952)] client 1
[('127.0.0.1', 48953)] client 2
[Me] client 3
[Me] ^CTraceback (most recent call last):
  File "chat_client.py", line 52, in
    sys.exit(chat_client())
  File "chat_client.py", line 30, in chat_client
    read_sockets, write_sockets, error_sockets = select.select(socket_list
KeyboardInterrupt
```

Note that the client #3 did go off the line at the end by typing ^C

## Connection

### Server:

```python
import socket
print("Ready for connection")
s = socket.socket()
host = socket.gethostname()
port = 1248
s.bind((host, port))
s.listen(5)
while True:
    c, addr = s.accept()
    print(f'Got Connection from {addr}')
    v = b'You are connected!'
    c.send(v)
    c.close()
```




### Client:

```python
import socket
try:
    s = socket.socket()
    host = socket.gethostname()
    port = 1248
    s.connect((host,port))
    print(s.recv(1024))
    s.close()
except ConnectionRefusedError:
    print("Server not available!")
```