Your assignment is to build a server capable of receiving text messages from clients. The server should print these text messages on the standard output, but should not print any other messages such as debug information. From the server perspective, a message corresponds to the data received from a particular client during a communication session with that client. The server should be listening for text messages to a port known to the clients. The server should be able to queue connections from multiple clients i.e., it will handle the client connections sequentially, but should be able to accept connections from multiple clients. After servicing the current client to completion, it should proceed to the next. If multiple clients simultaneously try to send text messages to the server, the server should print them one at a time (in any order). Note that you don't have to implement an event -driven or multi-threaded server. Serving one client at a time is enough. We also want you to write a client. The client should read from a file, transmit the message and exit. You can assume that the client is run as "client server -IP-address port-number"where server -IP-address is the IP address of the server, and port -number is the TCP port the server listens on. The server is runas "server port -number". If the server cannot bind on a port, print a message to standard error.
Make sure you handle the following correctly:
1. Buffer management: Assume that the file contents can be arbitrarily large (assume a typical size of 20KB), but the buffers you use to read/write to the file or socket must be small and of fixed size (e.g., 4096 bytes).
2. Handling return values: By default, sockets are blocking, and for this assignment we will use only blocking sockets. "Blocking" means that when we issue a socket call that cannot be done immediately (including not being able to read or write), our process is waits until it can perform the action. This includes the case when
(a) a socket's internal buffer is full and therefore, no data can be written, or
(b) a socket's buffer is empty, and no data is available to be read.
However, if there is some data available to be read or some can be written, the call will return the number of bytes read or written respectively. NOTE: This returned value can be less than specified in the length argument to the call or indicate an error.

Server:

```
import socket
import thread

def on_new_client(clientsocket,addr):
    while True:
        print addr, ' >> '
        while True:
            msg = clientsocket.recv(1024)
            print msg
            if msg[-3:] == 'bye':
                break
            msg = raw_input('SERVER >> ')
            clientsocket.send(msg)
    clientsocket.close()

s = socket.socket()
host = socket.gethostname()
port = 50002

print 'Server started!'
print 'Waiting for clients...'
```

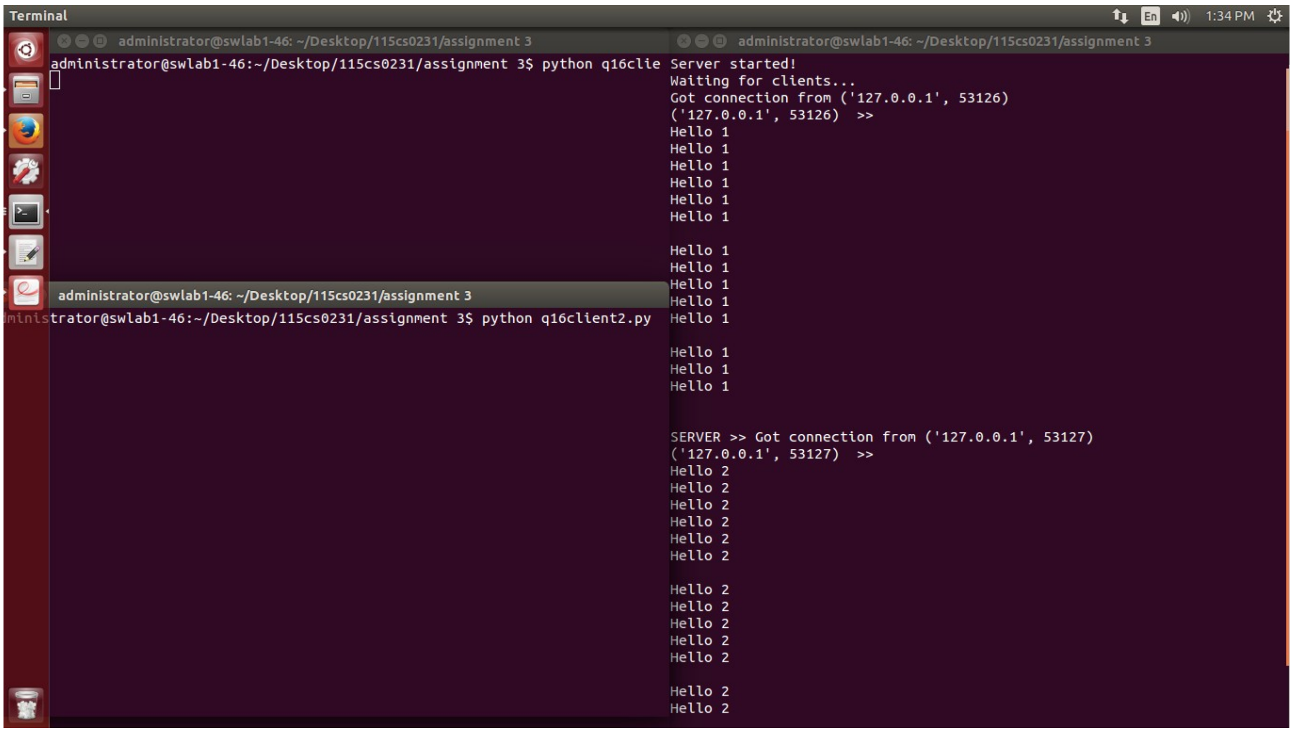
```
s.bind((host, port))
s.listen(5)

while True:
   c, addr = s.accept()
   print 'Got connection from', addr
   thread.start_new_thread(on_new_client,(c,addr))
s.close()
```

Client1:

```
import socket
s = socket.socket()
host = socket.gethostname()
port = 50002

f=open('16_1.txt','r')
msg=f.read()
#print msg
s.connect((host, port))
s.send(msg.encode())

print('Server: ',s.recv(1024).decode())
s.close()
```

Client 2:

```
import socket
s = socket.socket()
host = socket.gethostname()
port = 50002

f=open('16_2.txt','r')
msg=f.read()
#print msg
s.connect((host, port))
s.send(msg.encode())

print('Server: ',s.recv(1024).decode())
s.close()
```



Terminal

1:34 PM

administrator@swlab1-46: ~/Desktop/115cs0231/assignment 3

```
administrator@swlab1-46:~/Desktop/115cs0231/assignment 3$ python q16clie
```

administrator@swlab1-46: ~/Desktop/115cs0231/assignment 3

```
ministrator@swlab1-46:~/Desktop/115cs0231/assignment 3$ python q16client2.py
```

administrator@swlab1-46: ~/Desktop/115cs0231/assignment 3

```
Server started!
Waiting for clients...
Got connection from ('127.0.0.1', 53126)
('127.0.0.1', 53126)  >>
Hello 1
Hello 1
Hello 1
Hello 1
Hello 1
Hello 1

Hello 1
Hello 1
Hello 1
Hello 1
Hello 1

Hello 1
Hello 1
Hello 1


SERVER >> Got connection from ('127.0.0.1', 53127)
('127.0.0.1', 53127)  >>
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2

Hello 2
Hello 2
Hello 2
Hello 2
Hello 2

Hello 2
Hello 2
```