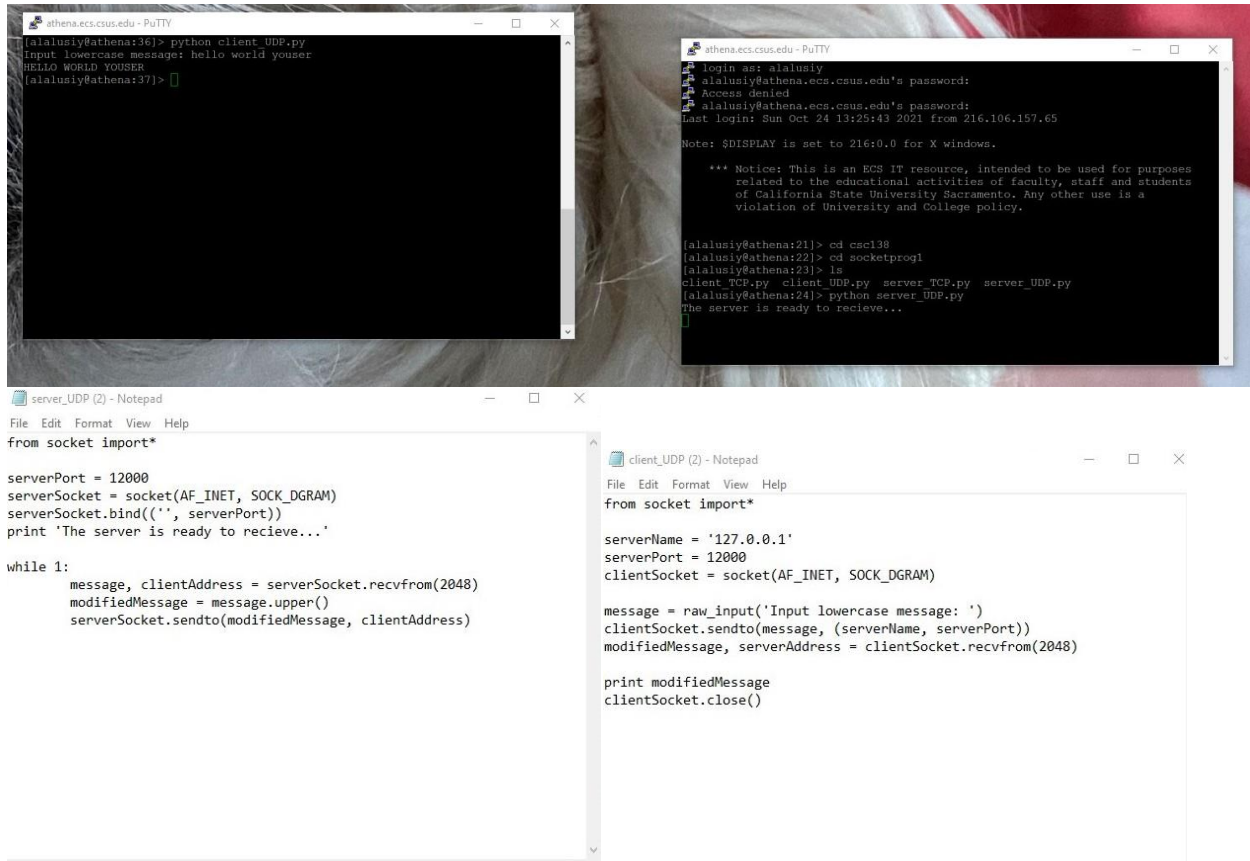


Youser Alalusi
10/24/2021

Socket Programming Assignment 1

The screenshots below are the python UDP client and server source code:



The image contains four screenshots of a computer screen. The top-left screenshot shows a terminal window titled 'athena.ecs.csus.edu - PuTTY' with the following text:

```
(alalusiy@athena:36)> python client_UDP.py
Input lowercase message: hello world youser
HELLO WORLD YOUSER
(alalusiy@athena:37)> 
```

 The top-right screenshot shows a terminal window titled 'athena.ecs.csus.edu - PuTTY' with the following text:

```
login as: alalusiy
alalusiy@athena.ecs.csus.edu's password:
Access denied
alalusiy@athena.ecs.csus.edu's password:
Last login: Sun Oct 24 13:25:43 2021 from 216.106.157.65

Note: $DISPLAY is set to 216:0.0 for X windows.

*** Notice: This is an ECS IT resource, intended to be used for purposes
related to the educational activities of faculty, staff and students
of California State University Sacramento. Any other use is a
violation of University and College policy.

(alalusiy@athena:21)> cd cscl38
(alalusiy@athena:22)> cd socketprogl
(alalusiy@athena:23)> ls
client_TCP.py client_UDP.py server_TCP.py server_UDP.py
(alalusiy@athena:24)> python server_UDP.py
The server is ready to receive...

```

 The bottom-left screenshot shows a Notepad window titled 'server_UDP (2) - Notepad' with the following code:

```
from socket import*

serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print 'The server is ready to receive...'

while 1:
    message, clientAddress = serverSocket.recvfrom(2048)
    modifiedMessage = message.upper()
    serverSocket.sendto(modifiedMessage, clientAddress)
```

 The bottom-right screenshot shows a Notepad window titled 'client_UDP (2) - Notepad' with the following code:

```
from socket import*

serverName = '127.0.0.1'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)

message = raw_input('Input lowercase message: ')
clientSocket.sendto(message, (serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)

print modifiedMessage
clientSocket.close()
```

UDP Client Analysis:

- For the client side I established a server name and server port number.
- Client socket is then created using AF_INET and SOCK_DGRAM.
- Using a variable we store raw input from keyboard.
- Send message with server name/port into socket.
- Read received reply message into string.
- Print out received message and close the socket

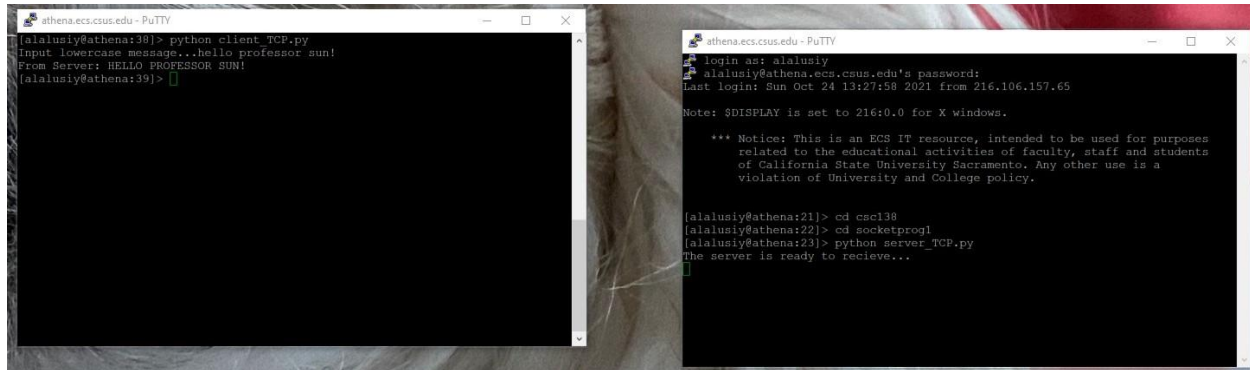
UDP Server Analysis:

- Establish server port that is the same as client.
- Create UDP socket via AF_INET and SOCK_DGRAM.
- Bind socket to local port that has been established.
- Print message that server is active and ready to receive.
- Create loop that will loop forever.
- Variable message stores whats read from UDP socket.
- Variable client address stores whats read from UDP socket.

Youser Alalusi
10/24/2021

- Modified message var changes it to all caps.
- Message is sent back via serverSocket.sendto command

The screenshots below are the python TCP client and server source code:



```
server_TCP - Notepad
File Edit Format View Help
from socket import*

serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
serverSocket.listen(1)
print 'The server is ready to recieve...'

while 1:
    connectionSocket, addr = serverSocket.accept()

    message = connectionSocket.recv(1024)
    messageCap = message.upper()
    connectionSocket.send(messageCap)
    connectionSocket.close()

client_TCP - Notepad
File Edit Format View Help
from socket import*

serverName = '127.0.0.1'
serverPort = 12000

clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
message = raw_input('Input lowercase message...')
clientSocket.send(message)

modifiedMessage = clientSocket.recv(1024)
print 'From Server:', modifiedMessage
clientSocket.close()
```

TCP Client Analysis:

- Established server name/port number.
- Created TCP socket for server via AF_INET and SOCK_STREAM.
- Connect socket using clientSocket.connect command.
- Message variable stores user raw input.
- Message sent via clientSocket.send command.
- Modified message var stores what is received from server.
- Print modified message.
- Close socket via clientSocket.close() command

TCP Server Analysis:

- Establish port number to connect to client.
- Create TCP welcoming socket via AF_INET and SOCK_STREAM.
- Bind the socket to the port via serverSocket.bind command.
- Server then listens via serverSocket.listen command.
- Print to screen that server is ready to receive.
- Create loop that will loop forever.

Youser Alalusi

10/24/2021

- Create socket to accept incoming request via `serverSocket.accept` command.
- `Message` var stores what is received from client.
- `MessageCap` var modifies message to all caps.
- Send modified message via `connectionSocket.send` command.
- Close connection socket via `connectionSocket.close` command.
- Does not close welcoming socket