Glòria Algara Muñoz 1565402
Beatriz Tomás Piñana 1562644

PRACTICA 3 XARXES

Càlcul del port:
Grup: B → 2
Subgrup: 7

8000 + (100 * 2) + 7 = 8207

Executem *netstat -alt*:

```
gloria@gloria-VirtualBox:~$ netstat -alt
Conexiones activas de Internet (servidores y establecidos)
Proto  Recib Enviad Dirección local         Dirección remota       Estado
tcp         0      0 gloria-VirtualBo:domain 0.0.0.0:*              ESCUCHAR
tcp         0      0 localhost:domain        0.0.0.0:*              ESCUCHAR
tcp         0      0 localhost:ipp           0.0.0.0:*              ESCUCHAR
tcp         0      0 gloria-VirtualBox:60518 ec2-52-10-115-210:https ESTABLECIDO
tcp         0      0 gloria-VirtualBox:37086 liveboxfibr:netbios-ssn ESTABLECIDO
tcp         0      0 gloria-VirtualBox:34820 ec2-44-230-27-229:https TIME_WAIT
tcp         0      0 gloria-VirtualBox:37088 liveboxfibr:netbios-ssn ESTABLECIDO
tcp6        0      0 ip6-localhost:ipp       [::]:*                 ESCUCHAR
```

Segona execució *netstat -alt:*

```
gloria@gloria-VirtualBox:~$ netstat -alt
Conexiones activas de Internet (servidores y establecidos)
Proto  Recib Enviad Dirección local         Dirección remota       Estado
tcp         0      0 0.0.0.0:8207            0.0.0.0:*              ESCUCHAR
tcp         0      0 gloria-VirtualBo:domain 0.0.0.0:*              ESCUCHAR
tcp         0      0 localhost:domain        0.0.0.0:*              ESCUCHAR
tcp         0      0 localhost:ipp           0.0.0.0:*              ESCUCHAR
tcp         0      0 gloria-VirtualBox:60518 ec2-52-10-115-210:https ESTABLECIDO
tcp         5      0 gloria-VirtualBox:37086 liveboxfibr:netbios-ssn CLOSE_WAIT
tcp         5      0 gloria-VirtualBox:37088 liveboxfibr:netbios-ssn CLOSE_WAIT
tcp6        0      0 ip6-localhost:ipp       [::]:*                 ESCUCHAR
```

1. **Que fan els paràmetres especificats a la comanda netstat? Quines diferències podem apreciar entre la sortida prèvia a l'execució del servidor i l'actual?**
   -alt:
   - - a → Mostra totes les connexions TCP actives i els ports TCP i UDP en què l'equip està escoltant.
   - - l → Mostra només els ports d'escolta activa.
   - - t→ Mostra només connexions TCP.

   A la primera execució de la comanda netcat, obtenim alguns servidors diferents respecte a la segona execució, com per exemple el 0.0.0.0:8207 (sortida lligada al nostre port) que només l'obtenim a la segona o a l'inversa, el servidor gloria-VirtualBox:34820 que només apareix a la primera. Per el servidor que apareixen en ambdues execucions alguns canvien el seu estat de ESTABLECIDO a CLOSE_WAIT.

```
gloria@gloria-VirtualBox:~$ netstat -alt
Conexiones activas de Internet (servidores y establecidos)
Proto  Recib Enviad Dirección local         Dirección remota       Estado
tcp        0      0 gloria-VirtualBo:domain 0.0.0.0:*              ESCUCHAR
tcp        0      0 localhost:domain        0.0.0.0:*              ESCUCHAR
tcp        0      0 localhost:ipp           0.0.0.0:*              ESCUCHAR
tcp        0      0 gloria-VirtualBox:60518 ec2-52-10-115-210:https ESTABLECIDO
tcp        5      0 gloria-VirtualBox:37086 liveboxfibr:netbios-ssn CLOSE_WAIT
tcp        5      0 gloria-VirtualBox:37088 liveboxfibr:netbios-ssn CLOSE_WAIT
tcp6       0      0 ip6-localhost:ipp       [::]:*                 ESCUCHAR
```

2. **Per quin motiu ara no hi ha diferències entre les 2 execucions de netstat? Quins paràmetres hauríem d'especificar per veure les diferències?**

Els dos codis son molt similars, la diferència es troba en la implementació de SOCK_STREAM o de SOCK_DGRAM.

SOCK_STREAM és un protocol on s'estableix una connexió i les dues parts mantenen una conversa fins que una de les parts finalitza la connexió o per un error de xarxa. En canvi, SOCK_DGRAM és un protocol basat en el datagrama, on on s'envia un datagrama i s'obté una resposta i la connexió finalitza.

El SOCK_STREAM utilitza TCP i el SOCK_DGRAM l'utilitza UDP. Aquí veiem el perquè hi ha diferència, ja que al utilitzar el paràmetre *-t*, limitem a que només es mostrin les connexions de TCP. Per tant, tot i fer el canvi de codi amb SOCK_DGRAM, com seguim executant el netstat restringint les connexions a únicament TCP, no veiem cap canvi.

Per veure diferències, hauríem d'especificar el paràmetre *-u*, el qual mostra només connexions UDP.

```
gloria@gloria-VirtualBox:~/Escritorio$ python3 echoServer.py
Connected by ('127.0.0.1', 58490)
Connected by ('127.0.0.1', 58492)
Connected by ('127.0.0.1', 58494)
^CTraceback (most recent call last):
  File "echoServer.py", line 10, in <module>
    conn, addr = s.accept()
  File "/usr/lib/python3.6/socket.py", line 205, in accept
    fd, addr = self._accept()
KeyboardInterrupt
```

```
gloria@gloria-VirtualBox:~/Escritorio$ nc localhost 8207
Hola
Hola
12345
gloria@gloria-VirtualBox:~/Escritorio$ nc localhost 8207
Hola
Hola
12345
gloria@gloria-VirtualBox:~/Escritorio$ nc localhost 8207
12345
12345
Hola
```

```
gloria@gloria-VirtualBox:~/Escritorio$ sudo lxc-attach -n u1
[sudo] contraseña para gloria:
root@u1:/# nc 10.0.3.1 8207
Hola
Hola
12345
root@u1:/#
```

```
gloria@gloria-VirtualBox:~/Escritorio$ python3 echoServer.py
Connected by ('10.0.3.70', 53316)
```

3. **Torneu a repetir l'experiment de connexió amb netcat (tant local com remot), però executant el servidor amb els següents valors de la variable HOST (heu de repetir l'experiment 3 vegades, un per cada valor): Llegiu la documentació referent a la funció bind(), expliqueu quines diferències heu observat. Quin significat té assignar el valor '' com a HOST?**

   La funció bind() associa el socket amb l'adreça IP de la interfície de xarxa i un port de capa de transport.

   Les diferències que sorgeixen al fer les execucions és que quan el HOST és “0.0.0.0” obteniem respota en els dos casos, si és el “localhost” només en el terminal de la màquina virtual i si fem “10.0.3.1” només en el terminal del contenidor u1.

   Si assignem el valor ‘’, voldrem dir que s'utilitzarà el comportament predeterminat del Sistema Operatiu, al no especificar a quin HOST volem que es conecti, admetrà a qualsevol, seria el mateix que possar el “0.0.0.0”.

4. **Consulteu la documentació i expliqueu quin és el propòsit del paràmetre que rep la funció listen().**

listen() posa el port en escolta i habilita la recepció de connexions entrants. Posteriorment, el *accept* accepta noves connexions i les estableix per poder inicialitzar l'intercanvi d'informació.

El paràmetre del *listen(),* mostra la longitud màxima a la qual pot arribar la cua de connexions pendents per a sockfd (file descriptor). Si arriba una sol·licitud de connexió quan la cua està plena, el client pot rebre un error amb una indicació ECONNREFUSED. O, si el protocol que s'utilitza admet la retransmissió, es pot ignorar la sol·licitud de manera que es torni a enviar aquesta sol·licitud posteriorment i si que arribi correctament.

Ex:
Si fem un *listen(5)* i 6 connexions soliciten connectar-se, abans de que haguem fet un *accept(),* una d'aquestes 6 connexions es perd..

5. **Implementeu la vostra versió del client mitjançant python i la seva API de sockets (anomeneu l'arxiu echoClient.py), podeu fer servir l'esquema de crides de la Figura 1 com a guia.**

```
import socket
import sys

client= socket.socket(socket.AF_INET, socket.SOCK_STREAM) #definir un socket
client.connect(('0.0.0.0', 8207))   #obrir una connexió amb el servidor
line = sys.stdin.readline()     #llegir la informació que introdueixi l'usuari
per consola
bytes = str.encode(line)
client.send(bytes)      #enviar-li al servidor
client_recv = client.recv(4096)    #recepció
client.close()     #tancar connexió
print (client_recv)    #mostrem la resposta del servidor
```

Hem comprovat que el nostre codi funciona escrivint en el terminal on executem el programa la paraula "prova" i hem vist que es mostrava en el terminal del netcat.

```
gloria@gloria-VirtualBox:~/Escritorio$ python3 echoClient.py
prova
```

```
gloria@gloria-VirtualBox:~/Escritorio$ nc -lt 8207
prova
```

6. **(2 Punts) Implementeu una aplicació anomenada quoteCollector.py, aquesta aplicació ha de contenir un llistat de strings (inicialment buit), establirà connexions successives als 3 servidors de QOTD, emmagatzemant cada cop la cita retornada a la**

seva llista, fins a recopilar un total de 31 cites, un cop finalitzada la tasca emmagatzemarà la llista en format JSON a disc amb el nom quotes.json. Addicionalment la vostra aplicació ha de complir els següents requeriments:

- Accediu als servidors de forma seqüencial utilitzant una estratègia round-robin.
- Feu una espera de 2 segons entre connexions.
- Les cites no s'han de repetir, abans de fer una inserció heu de comprovar si ja existeix, descartar la repetida i continuar iterant.

La major part d'estructures de dades de python es poden transformar a JSON (serialitzar) i a la inversa (desserialitzar) mitjançant el mòdul JSON, doneu un cop d'ull a la referència [5] per veure els detalls.

```
import json
import socket
import time

quotes = []
x = 0
y = 0
z = 0

def Crida_Servidor(servidor):
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect((servidor, 17))
    client_recv = client.recv(4096)
    try:
        if (client_recv not in quotes) and (len(client_recv) <= 512):
            time.sleep(2)
            quotes.append(client_recv)
            client.close()
            return 0
        else:
            time.sleep(2)
            return 0
    except socket.error:
        return 1

while len(quotes) < 31:
    if x == 0:
        x = Crida_Servidor('djxmmx.net')
    if y == 0:
        y = Crida_Servidor('cygnus-x.net')
        y = 1
    if z == 0:
        z = Crida_Servidor('alpha.mike-r.com')
        z = 1
```

```
    if (len(quotes) < 31):
        client1 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        client1.connect(('127.0.0.1', 8207))
        client1_recv = client1.recv(4096)
        if ((client1_recv not in quotes) and (len(client1_recv) <= 512)):
            time.sleep(2)
            quotes.append(client1_recv)
            client1.close()
        else:
            time.sleep(2)


with open("quotes.json", "w") as file:
    json.dump(quotes, file)
```

7. (2 punts) Implementeu un servidor QOTD que compleixi les especificacions de la RFC. Per implementar el servidor podeu agafar com a punt de partida el fitxer echoServer.py. Com a requisit addicional, el servidor ha de carregar les cites que heu guardat al fitxer quotes.json, les haureu d'emmagatzemar en alguna estructura de dades a la vostra elecció. Aquesta serà l'estructura que haureu de consultar cada cop que un client us faci una petició. Normalment el bind del socket hauria de fer-se amb el port 17, però al ser del rang de ports reservats és una operació que requereix privilegis de root (feu la prova), en el vostre cas continueu fent servir el port del vostre grup. Teniu llibertat per implementar de la manera que cregueu més convenient tot allò que no s'hagi especificat.

```
import socket
import random
import json

HOST = ''
PORT = 8207
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((HOST, PORT))
s.listen(1)

with open("quotes.json","r") as file:
    cites = json.load(file)

while 1:
    conn, addr = s.accept()
    print ('Connected by', addr)
    q = random.choice(cites)
    rand = q

    file.close()
    conn.send(rand)
    conn.close()
```

8. (3 punts) A continuació implementareu una versió millorada del protocol (customServer.py), ara la vostra aplicació ja no seguirà la RFC i per tant haurieu de cambiar el port per un fora del rang de ports reservats. Feu servir el port que correspongui al vostre grup. Fins el moment el servidor envia una cita i tanca la connexió com a resposta a les connexions entrants. Ara el que haurà de fer és esperar a que el client li envii una comanda amb instruccions. Les comandes que enviarà el client estaràn en format JSON i han de ser les següents:

{"op":"get", "mode":"random"}	**El servidor respondrà amb una cita aleatoria**
{"op":"get", "mode":"day"}	**El servidor respondrà amb la cita que correspongui al dia dels mes (1 al 31).**
{"op":"get", "mode":"index", "index":[1-31]}	**El servidor respondrà amb la cita a l'index especificat per param.**
{"op":"count"}	**El servidor respondrà amb un valor numeric que representa el número de cites que te a la seva estructura de dades.**
{"op":"add", "quote":"Text de la cita"}	**El servidor afegirà la cita especificada a "quote" a la seva estructura de dades.**
{"res": "OK"}	**Resposta del servidor per indicar que una operació sense output s'ha finalitzat correctament (en el nostre cas només aplicaria a l'operació add).**
{"res": "KO"}.	**Resposta del servidor en cas que alguna operació no es pugui realitzar.**

Haureu de llegir les dades que envia el client i parsejar el seu contingut, en cas que les dades que ha enviat no segueixin un format de missatge valid podeu tancar la connexió.

Per fer proves podeu utilitzar netcat com a client.

```
import socket
import random
import json
from datetime import date

HOST = ''
PORT = 8207
```

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((HOST, PORT))
s.listen(1)

with open("quotes.json","r") as file:
    cites = json.load(file)

while 1:
    conn, addr = s.accept()
    print ('Connected by', addr)
    data = conn.recv(1024)
    data_str = str(data)


    index = str('{"op":"get", "mode":"index", "index":[00]}\n')

    index2= str('{"op":"add", "quote":"Text de la cita"}\n')

    if not data: break
    elif (data == str('{"op":"get", "mode":"random"}\n')):
            q = random.choice(cites)

    elif (data == str('{"op":"get", "mode":"day"}\n')):
            data = str(date.today())
            dia = data[8]+data[9]
            intdia = int(dia)
            day = intdia-1
            q = cites[day]

    elif (data_str[0:37] == index[0:37]):

            if ((data_str[-5] not in "0123456789") and (data_str[-3] not in "0123456789")) or
((data_str[-6] not in "0123456789") and (data_str[-3] not in "0123456789")):
                    q = str('{"res": "KO"}\n')

            elif ((data_str[-3] not in "0123456789") and (data_str[-4] not in "0123456789")):
                    q = str('{"res": "KO"}\n')

            elif ((data_str[-3] not in "0123456789") and (data_str[-4] == ":")):
                    q = str('{"res": "KO"}\n')

            elif ((data_str[-3] in "0123456789") and (data_str[-4] in "0123456789") and
(data_str[-5] in "0123456789")):
                    q = str('{"res": "KO"}\n')

            elif ((data_str[-3] in "0123456789") and (data_str[-4] in "0123456789") and
(data_str[-5] not in "0123456789")):
                    num = data_str[-4]+data_str[-3]
                    num = int(num)
                    if num >= len(data_str) or  num < 0:
                            q = str('{"res": "KO"}\n')
                    else:
                            q = cites[num]
```

```
                        q= str(q)

        elif ((data_str[-3] in "0123456789") and (data_str[-4] == ":")):
                num = data_str[-3]
                num = int(num)
                if num >= len(data_str) or  num < 0:
                        q = str('{"res": "KO"}\n')
                else:
                        q = cites[num]
                        q= str(q)


elif (data == str('{"op":"count"}\n')):
        q = len(cites)

elif (data[0:21] == index2[0:21]):
        ind = data[22:-3]
        st = str(ind)
        if st in  cites:
                q = str('{"res": "KO"}\n')
        else:
                cites.append(st)
                q = str('{"res":"OK"}\n')
else:
        q = str('{"res": "KO"}\n')


file.close()
q = str(q)
q = str.encode(q)
conn.send(q)
conn.close()
```

9 (1.5 punts): Creeu un client que implementi el format de missatges anterior i permeti seleccionar quin es vol enviar vía paràmetres de consola.

Exemples d'execució:

python customClient.py -op get -mode random
python customClient.py -op add -quote "Never underestimate the bandwidth of a station wagon full of tapes hurtling down the highway."

```
import socket
import sys
import argparse

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
client.connect(('127.0.0.1', 8207))
arg = sys.argv
q = 0
print('arg',arg)

if  ("-op" and "get" and "mode" and "random") in arg:
    q = str('{"op":"get", "mode":"random"}\n')

elif  (("-op" and "add" and "-quote") in arg):
    cita = arg[-1]
    array = (str("{") + str('"op"') + str(":") + str('"add",') + ' ' + str('"quote"') + str(":") + '"' + cita
+ '"' + str("}\n"))
    q = str(array)

elif  (("-op" and "get" and "-mode" and "day") in arg):
    q = str('{"op":"get", "mode":"day"}\n')

elif  (("-op" and "get" and "-mode" and "index" and "-index") in arg):
    valor = arg[-1]

    array = {"op":"get", "mode":"index", "index":valor}
    q = str('array\n')

elif  (("-op" and "count") in arg):
    q = str('{"op":"count"}\n')


if q != 0:
    client.send(q)
client_recv = client.recv(4096)
client.close()
print (client_recv)
```

```
import json


quotes =
[“Q”,”W”,”E”,”R”,“T”,”Y”,”U”,”I”,”O”,”P”,”A”,”S”,”D”,”F”,”G”,”H”,”J”,”K”,”L”,”Ñ”,”Z”,”X”,”C”,”V”,”B
”,”N”,”M”,”1”,”2”,”3”,”4”]


with open(“quotes.json”,”w”) as file:
        json.dump(quotes,file)
```

```
file.close()
```