TP 04

1 Gestion de multiples clients avec un serveur TCP

1.1 Version thread

- Le code :

```
#!/usr/bin/python3.6
import os
import socket
import sys
import threading

def handle (sclient) :
    while True :
        data = sclient.recv(1500)
        if data == b"" : break
        sclient.send(data)

    print ("Requet from : ", a[0])

    sclient.close()

try:
    host = ""
    port = 7777

    s = socket.socket(socket.AF_INET6, socket.SOCK_STREAM, 0)
    try :
        s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        s.bind((host, port))
        s.listen(1)

        try :
            while True :
                # Récupère le socket de la connexion du client et sont ac
                sclient, a = s.accept()
                t = threading.Thread(None, handle, None, (sclient,))
                t.start()

        except Exception as e :
            print("Error with accept : ", e)
            s.close()

    except Exception as e :
        print("Error on bind the socket : ", e)

except KeyboardInterrupt:
    print('Interrupted')
    s.close()
    try:
        sys.exit(0)
    except SystemExit:
        os._exit(0)
```

- Pendant l'exécution, j'effectue la commande `netstat -tuap | grep 7777` :

```
tcp    0  0 localhost:47282  localhost:7777   ESTABLISHED 14275/nc
tcp    0  0 localhost:47292  localhost:7777   ESTABLISHED 14372/nc
tcp    0  0 localhost:47286  localhost:7777   ESTABLISHED 14323/nc
tcp6   0  0 [::]:7777        [::]:*           LISTEN      14207/python3.6
tcp6   0  0 localhost:7777   localhost:47282  ESTABLISHED 14207/python3.6
tcp6   0  0 localhost:7777   localhost:47286  ESTABLISHED 14207/python3.6
tcp6   0  0 localhost:7777   localhost:47292  ESTABLISHED 14207/python3.6
```

1.2 Version select

- Le code :

```
#!/usr/bin/python3.6
import os
import select
import socket
import sys

try:
    host = ""
    port = 7777

    s = socket.socket(socket.AF_INET6, socket.SOCK_STREAM, 0)
    try :
        s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        s.bind((host, port))
        s.listen(1)

        l = []

        try :
            while True :
                (rl, wl, xl) = select.select(l+[s], [], [])
                for soc in rl :
                    if soc == s :
                        # Récupère le socket de la connexion du client et
                        sclient, a = s.accept()
                        l.append(sclient)
                        print(a[0], "is connect!")
                    else:
                        data = soc.recv(1500)
                        if data == b"" :
                            soc.close()
                            l.remove(soc)
                        else:
                            sclient.sendall(data)

        except Exception as e :
            print("Error with accept : ", e)
            s.close()

    except Exception as e :
        print("Error on bind the socket : ", e)

except KeyboardInterrupt:
    print('Interrupted')
    s.close()
    try:
        sys.exit(0)
    except SystemExit:
        os._exit(0)
```

- Pendant l'exécution, j'effectue la commande `netstat -tuap | grep 7777` :

```
tcp    0  0 localhost:47646  localhost:7777   ESTABLISHED 17983/nc
tcp    0  0 localhost:47644  localhost:7777   ESTABLISHED 17962/nc
tcp    0  0 localhost:47648  localhost:7777   ESTABLISHED 17993/nc
tcp6   0  0 [::]:7777        [::]:*           LISTEN      17946/python3.6
tcp6   0  0 localhost:7777   localhost:47646  ESTABLISHED 17946/python3.6
tcp6   0  0 localhost:7777   localhost:47648  ESTABLISHED 17946/python3.6
tcp6   0  0 localhost:7777   localhost:47644  ESTABLISHED 17946/python3.6
```

2 Serveur de chat

2.1 Une première version simple

```
#!/usr/bin/python3.6
import os
import select
import socket
import sys

try:
    host = ""
    port = 7777

    s = socket.socket(socket.AF_INET6, socket.SOCK_STREAM, 0)
    try :

        s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        s.bind((host, port))
        s.listen(1)

        try :
            l = []
            while True :
                (rl, wl, xl) = select.select(l+[s], [], [])
                for soc in rl :
                    if soc == s :
                        # Récupère le socket de la connexion du client et
```

```
                          sclient, a = s.accept()
                          l.append(sclient)
                          print(a[0], "is connect!")
                      else :
                          data = soc.recv(1500)
                          if data == b"" :
                              soc.close()
                              l.remove(soc)
                          else :
                              # Envois le message a tous le monde sauf au s
                              for sc in l :
                                  if soc != sc and s != sc:
                                      sc.sendall(data)

          except Exception as e :
              print("Error with accept : ", e)
              s.close()

      except Exception as e :
          print("Error on bind the socket : ", e)

  except KeyboardInterrupt:
      print('Interrupted')
      s.close()
      try:
          sys.exit(0)
      except SystemExit:
          os._exit(0)
```

2.2 A propos d'une version thread

- Si on utilise la version thread, il peut se passer que 2 messages soient traités en même temps sur 2 threads différents. IL peut y avoir aussi des éléments qui ne sont pas à jour dans la liste des clients connectés.

2.3 Extensions du serveur de chat

- MSG fait !
- Invalid command fait !
- Notif fait !
- NICK fait !
- Modifs fait !
- WHO fait !
- QUIT fait !
- KILL fait !

```
#!/usr/bin/python3.6
import os
import re
import select
import socket
import sys
```

```python
# get host of client
def getScHost(sc) :
    return sc.getpeername()[0] + ':' + str(sc.getpeername()[1])

# get selecte comment and execut it
def commands(scSender, data) :
    m = re.match(r"^KILL\s([^\s]+)\s(.+)$", data)
    if m :
        for sc, nick in nicks.items() :
            if nick == m[1] :
                sc.sendall(('[' + nicks[scSender] + '] ' + m[2] + '\n')
                .encode("utf-8"))
                disconnect(sc)
                break
    else :
        m = re.match(r"^NICK\s([^\s]+)$", data)
        if m :
            if logs : print('client "' + nicks[scSender] + '" => "' +
            m[1] + '"')
            nicks[scSender] = m[1]
        else :
            m = re.match(r"^WHO$", data)
            if m :
                online = ''
                for scClient in scList :
                    if scClient != scServer :
                        online += ' ' + nicks[scClient]
                scSender.sendall(('[' + nicks[scServer] + ']' + online +
                '\n').encode("utf-8"))
            else :
                m = re.match(r"^([A-Z]+)\s(.+)$", data)
                if m : # RegEx au lieu de split
                    if m[1] == 'MSG' or m[1] == 'QUIT':
                        if len(m[2]) > 2000 :
                            scSender.sendall('The message must be less
                            than 2000 characters !\n'.encode("utf-8"))
                        else :
                            sendall(scSender, '[' + nicks[scSender] + ']
                            ' + m[2])
                        if m[1] == 'QUIT' :
                            disconnect(scSender)
                    else :
                        scSender.sendall(('[' + nicks[scServer] + '] ' +
                        'Invalid command !\n').encode("utf-8"))
                else :
                    scSender.sendall(('[' + nicks[scServer] + '] ' + 'In
                    valid command !\n').encode("utf-8"))

# Semd all messate to everybody without sever and sender
def sendall(scSender, msg) :
    for scClient in scList :
        if scClient != scSender and scClient != scServer :
            scClient.sendall((msg + "\n").encode("utf-8"))

def disconnect(scClient) :
    sendall(scClient, '[' + nicks[scServer] + '] ' + nicks[scClient] +
    " is disconnected")
    if logs : print('client disconnected "' + nicks[scClient] + '"')
    del nicks[scClient]
    scList.remove(scClient)
    scClient.close()

try:
    logs = True

    host = ""
    port = 7777

    scServer = socket.socket(socket.AF_INET6, socket.SOCK_STREAM, 0)
    try :

        scServer.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        scServer.bind((host, port))
        scServer.listen(1)
        if logs :
            print("Chat server is running in verbose mode")
        else :
            print("Chat server is running")

        try :
            scList = []
            nicks = {scServer: 'server'}
            while True :
                (rl, wl, xl) = select.select(scList+[scServer], [], [])
                for scSelected in rl :
                    if scSelected == scServer :
                        # Get client socket and ip address
                        scNewClient, a = scServer.accept()

                        # Connect
                        scList.append(scNewClient)
                        nicks[scNewClient] = getScHost(scNewClient)
                        scNewClient.sendall(('[' + nicks[scServer] + ']
                        ' + 'Welcome to Chat Server !\n').encode("utf-8")
                        sendall(scNewClient, '[' + nicks[scServer] + ']
                        ' + nicks[scNewClient] + " is connected")
                        if logs : print('client connected "' + nicks[
                            scNewClient] + '"')
                    else :
                        data = scSelected.recv(2008).decode("utf-8") #8 +
                        if data == "" :
                            disconnect(scSelected)
                        else :
                            commands(scSelected, data)
        except Exception as e :
            print("Error with accept : ", e)
            scServer.close()

    except Exception as e :
        print("Error on bind the socket : ", e)

except KeyboardInterrupt:
    print('Interrupted')
    scServer.close()
    try:
        sys.exit(0)
    except SystemExit:
        os._exit(0)
```

2.4 Bonus : les channels