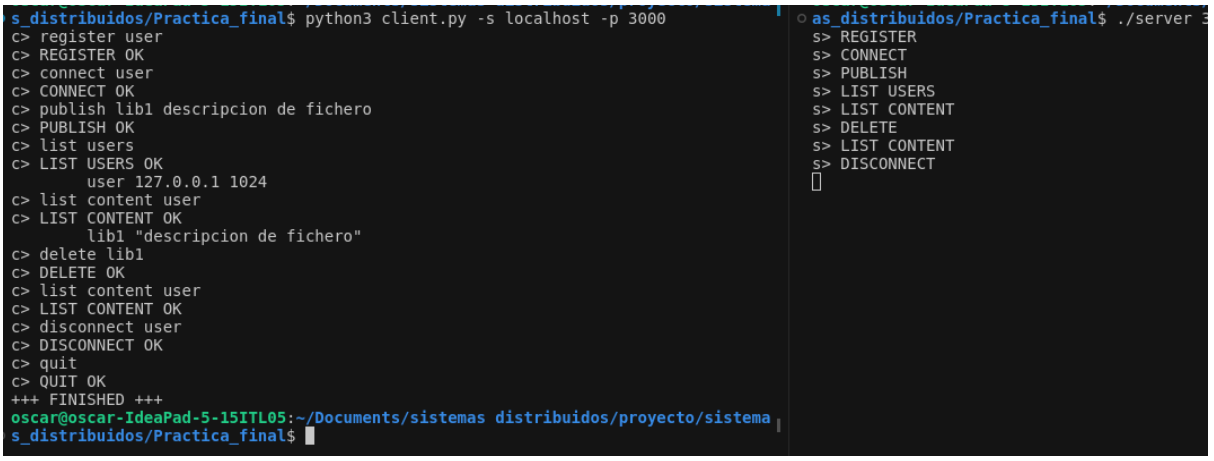
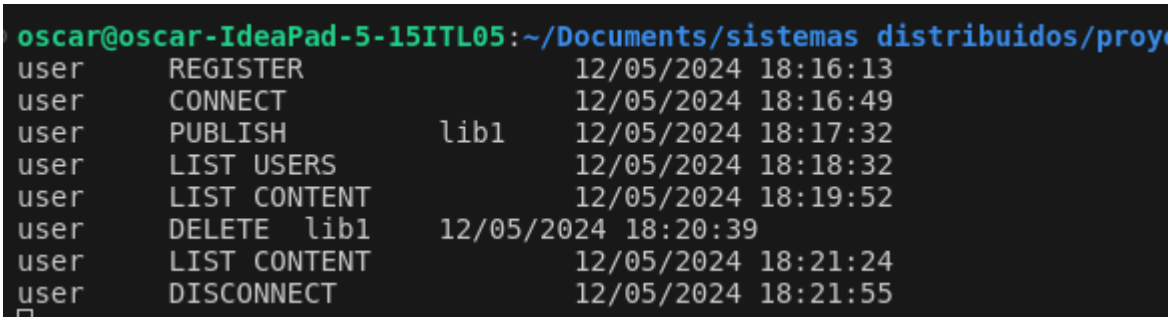
Pruebas

1. Uso esperado con RPC incluido, para el resto de pruebas no se mostrará la salida del servidor RPC, porque la implementación usada tarda mucho tiempo en ejecutarse, y se recomienda al profesor que para hacer las pruebas también comente las 8 líneas en el archivo server.c que son:
   `imprimir_operacion_usuario_1(&send_rpc, client);`
   En las líneas: `429, 463, 527, 574, 656, 721, 765, 817`

   Esto se debe a que puede tardar varios segundos la parte del servicio RPC.
   Para esta prueba se realizaron las operaciones exitosas de
   REGISTER user
   CONNECT user
   PUBLISHED lib1 descripcion de fichero
   LIST_USERS
   LIST_CONTENT user
   DELETE lib1
   LIST_CONTENT user
   DISCONNECT user

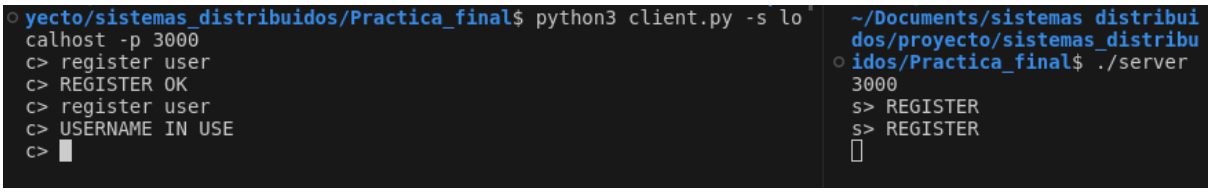


   terminal de servidor RPC:

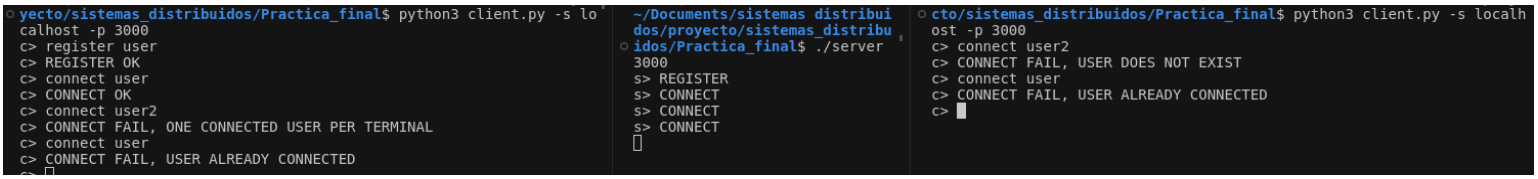


   A Partir de esta prueba se muestran solo la terminal del server y del client.py por lo comentando anteriormente

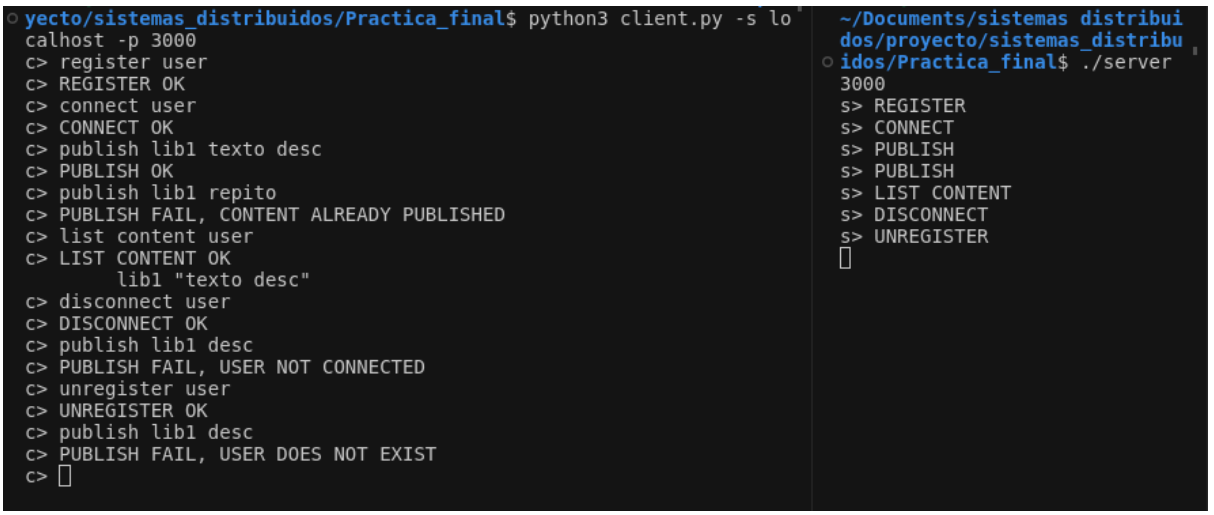
2. Vamos a probar la operación registrar.
   Las operaciones que se van a realizar son:
   REGISTER user
   REGISTER user



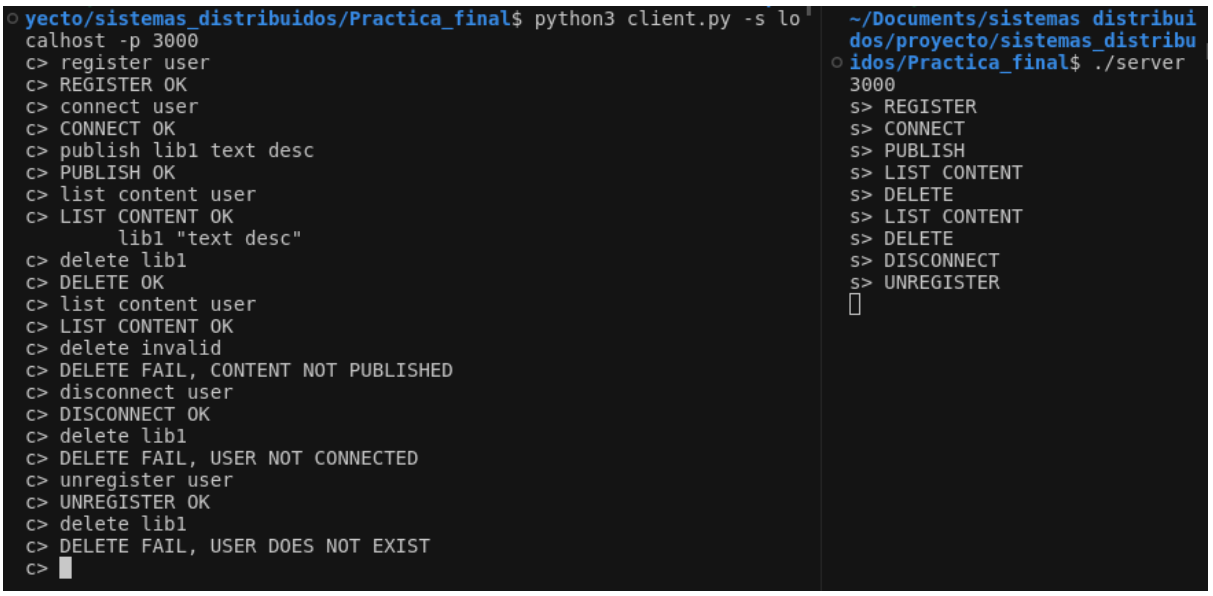
3. Probar operación connect, aquí es interesante destacar la aproximación que hemos hecho donde solo puede haber un usuario conectado por terminal.
   Las operaciones que se van a realizar:
   Tras haber hecho REGISTER user, y siguiendo el orden de ejecución.
   Terminal 1:
   CONNECT user
   CONNECT user2
   CONNECT user
   Terminal 2:
   CONNECT user2
   CONNECT user



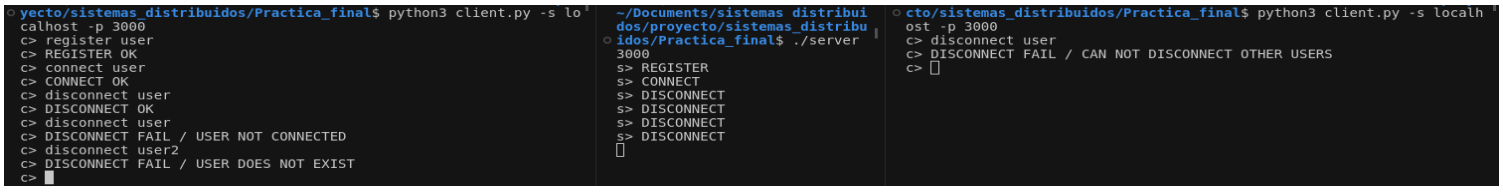
   Destacar que las 2 operaciones finales de la terminal 1 no se enviarán al servidor, es client.py el que directamente está gestionando esos casos.
4. Probar operación PUBLISH.
   Tras haber hecho REGISTER user y CONNECT user
   Las operaciones que se van a realizar:
   PUBLISH lib1 text desc
   PUBLISH lib1 repito
   LIST_CONTENT user
   DISCONNECT user
   PUBLISH user
   UNREGISTER user
   PUBLISH user



5. Probar operación DELETE.
   Tras haber hecho REGISTER user, CONNECT user, PUBLISH lib1 text desc y LIST_CONTENT user.
   Las operaciones que se van a realizar:
   DELETE lib1
   LIST_CONTENT user
   DELETE invalid
   DISCONNECT user
   DELETE lib1
   UNREGISTER user
   DELETE lib1



6. Probar operación DISCONNECT.
   Tras haber hecho REGISTER user, CONNECT user.
   Las operaciones que se van a realizar:
   Terminal 2:
   DISCONNECT user
   Terminal 1:
   DISCONNECT user
   DISCONNECT user
   DISCONNECT user2



7. Probar operación LIST_USERS.
   Tras haber hecho REGISTER user, CONNECT user.
   Las operaciones que se van a realizar:
   LIST_USERS
   DISCONNECT user

LIST_USERS
UNREGISTER user
LIST_USERS

```
yecto/sistemas_distribuidos/Practica_final$ python3 client.py -s lo
calhost -p 3000
c> register user
c> REGISTER OK
c> connect user
c> CONNECT OK
c> list users
c> LIST USERS OK
        user 127.0.0.1 1024
c> disconnect user
c> DISCONNECT OK
c> list users
c> LIST USERS FAIL, USER NOT CONNECTED
c> unregister user
c> UNREGISTER OK
c> list users
c> LIST USERS FAIL, USER DOES NOT EXIST
```

```
~/Documents/sistemas distribui
dos/proyecto/sistemas_distribu
idos/Practica_final$ ./server
3000
s> REGISTER
s> CONNECT
s> LIST USERS
s> DISCONNECT
s> UNREGISTER
```

8. Probar operación LIST_CONTENT.
   Tras haber hecho REGISTER user, CONNECT user, PUBLISH lib1 text desc.
   Las operaciones que se van a realizar:
   LIST_CONTENT user
   LIST_CONTENT user2
   DISCONNECT user
   LIST_CONTENT user
   UNREGISTER user
   LIST_CONTENT user

```
yecto/sistemas_distribuidos/Practica_final$ python3 client.py -s lo
calhost -p 3000
c> register user
c> REGISTER OK
c> connect user
c> CONNECT OK
c> publish lib1 text desc
c> PUBLISH OK
c> list content user
c> LIST CONTENT OK
        lib1 "text desc"
c> list content user2
c> LIST CONTENT FAIL, REMOTE USER DOES NOT EXIST
c> disconnect user
c> DISCONNECT OK
c> list content user
c> LIST CONTENT FAIL, USER NOT CONNECTED
c> unregister user
c> UNREGISTER OK
c> list content user
c> LIST CONTENT FAIL, USER DOES NOT EXIST
c>
```

```
~/Documents/sistemas distribui
dos/proyecto/sistemas_distribu
idos/Practica_final$ ./server
3000
s> REGISTER
s> CONNECT
s> PUBLISH
s> LIST CONTENT
s> LIST CONTENT
s> DISCONNECT
s> UNREGISTER
```

9. Probar operación UNREGISTER.
   Tras haber hecho REGISTER user, CONNECT user.
   Las operaciones que se van a realizar:
   UNREGISTER user
   DISCONNECT user
   UNREGISTER user
   UNREGISTER user

```
yecto/sistemas_distribuidos/Practica_final$ python3 client.py -s lo
calhost -p 3000
c> register user
c> REGISTER OK
c> connect user
c> CONNECT OK
c> unregister user
c> CAN NOT UNREGISTER CONNECTED USERS
c> disconnect user
c> DISCONNECT OK
c> unregister user
c> UNREGISTER OK
c> unregister user
c> USER DOES NOT EXIST
c>
```

```
~/Documents/sistemas distribui
dos/proyecto/sistemas_distribu
idos/Practica_final$ ./server
3000
s> REGISTER
s> CONNECT
s> UNREGISTER
s> DISCONNECT
s> UNREGISTER
s> UNREGISTER
```

10. Probar operación GET_FILE.

    Tras haber hecho en Terminal 1 REGISTER user, CONNECT user. y en Terminal 2 REGISTER user2 CONNECT user2

    Las operaciones que se van a realizar:

    Terminal 1:

        GET_FILE user2 prueba.txt primero.txt
        GET_FILE user2 no-existe.txt tercero.txt

    Terminal 2:

        GET_FILE user prueba.txt segundo.txt
        DISCONNECT user2

    Terminal 1:

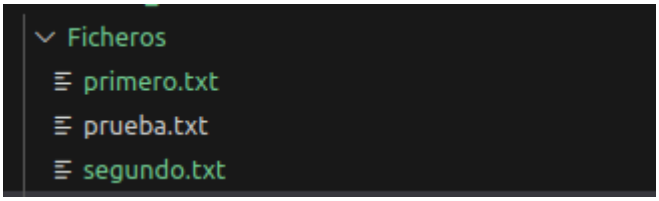
        GET_FILE user2 prueba.txt cuarto.txt
        DISCONNECT user
        GET_FILE user2 prueba.txt quinto.txt
        UNREGISTER user2
        GET_FILE user2 prueba.txt sexto.txt

```
yecto/sistemas_distribuidos/Practica_final$ python3 client.py -s lo
calhost -p 3000
c> register user
c> REGISTER OK
c> connect user
c> CONNECT OK
c> GET FILE user2 prueba.txt primero.txt
c> GET FILE OK
c> get file user2 no-existe.txt tercero.txt
GET FILE FAIL / FILE NOT EXIST
c> get file user2 prueba.txt cuarto.txt
c> GET FILE FAIL
c> disconnect user
c> DISCONNECT OK
c> get file user prueba.txt quinto.txt
c> GET FILE FAIL, USER NOT CONNECTED
c> unregister user
c> UNREGISTER OK
c> get file user2 prueba.txt sexto.txt
c> GET FILE FAIL, USER DOES NOT EXIST
c>
```

```
~/Documents/sistemas distribui
dos/proyecto/sistemas_distribu
idos/Practica_final$ ./server
3000
s> REGISTER
s> CONNECT
s> REGISTER
s> CONNECT
s> GET FILE
s> GET FILE
s> GET FILE
s> DISCONNECT
s> GET FILE
s> DISCONNECT
s> UNREGISTER
```

```
cto/sistemas_distribuidos/Practica_final$ python3 client.py -s localh
ost -p 3000
c> register user2
c> REGISTER OK
c> connect user2
c> CONNECT OK
c> GET FILE user prueba.txt segundo.txt
c> GET FILE OK
c> disconnect user2
c> DISCONNECT OK
c>
```

Aquí se puede ver los archivo generado, que son nuevo:



En este caso tanto primero.txt como segundo.txt estan porque ambas terminales están ejecutando el mismo archivo