Universidade de Coimbra

Faculdade de Ciências e Tecnologia

Departamento de Engenharia Informática

UNIDADE CURRICULAR

Introdução às Redes de Comunicação

2015-2016

TRABALHO 2 - Protocolos de transferência de ficheiros com cache

Docentes:
João P. Vilela
Tiago Cruz

Alunos:
Ricardo Cruz
Gilberto Rouxinol

ÍNDICE

1 Introdução 3

2 Cliente 4

3 Cache 6

4 Servidor 8

5 Conclusão 8

6. ANEXO 10

Protocolos de transferência de ficheiros com cache

1 Introdução

Pretende-se com este trabalho mostrar uma solução para implementar protocolos de transferência de ficheiros utilizando programação de *sockets* com TCP, em Python. Para tal foram utilizados os conceitos adquiridos nas aulas teóricas, teórico-práticas e laboratoriais sobre a camada de transporte TCP e os *sockets* - ver **Figura 1-1**. A solução consiste na implementação de um processo "servidor fim de linha" e um processo "servidor intermédio" designados de SERVIDOR e CACHE. Estes processos estão preparados para multi-cliente. A implementação do CLIENTE apresenta duas situações: (1) cliente público ou não autenticado; e (2) cliente privado ou autenticado. O cliente tem ao seu dispor diversos serviços, nomeadamente: (1) listar os ficheiros (**LIST**); (2) descarregar ficheiros (**DOWNLOAD**); e (3) carregar ficheiros (**UPLOAD**). O SERVIDOR contém uma pasta pública e várias pastas privadas, contendo diversos tipos de ficheiros. Inicialmente a CACHE contém zero ficheiros. À medida que os clientes públicos requerem um DOWNLOAD, a CACHE não os podendo satisfazer, requer um DOWNLOAD ao SERVIDOR, guarda o ficheiro e devolve uma cópia ao cliente público. Posteriormente havendo outro cliente público a requerer o mesmo ficheiro, automaticamente a CACHE devolve-o, sem necessitar de sobrecarregar o SERVIDOR.

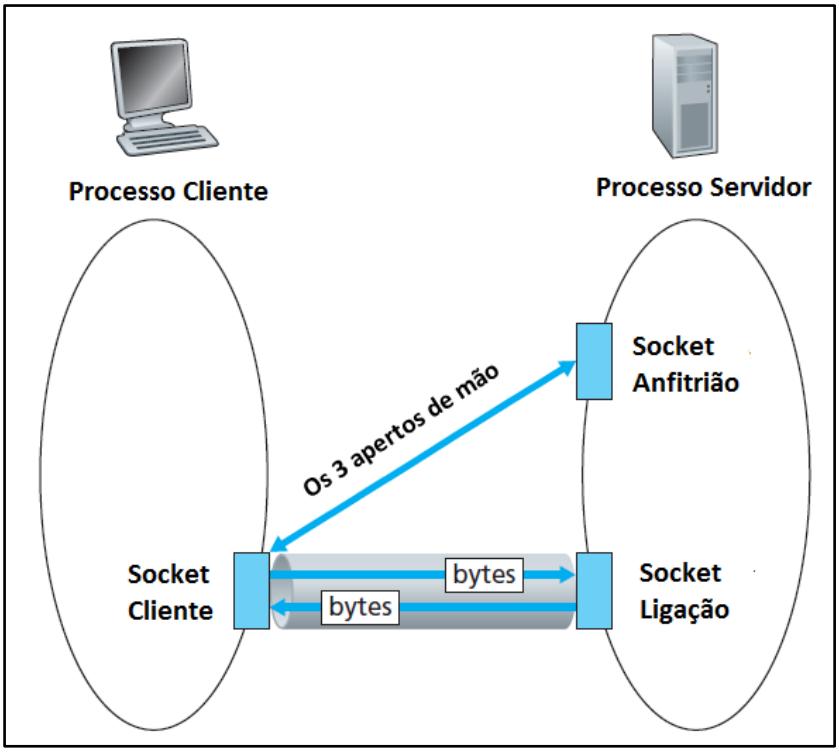

Figura 1-1: Processo TCP servidor com dois *sockets*

2 Cliente

A implementação do *socket* cliente consiste num algoritmo que, importando a livraria *socket* do Python, permite realizar a ligação TCP entre cliente e CACHE ou entre CLIENTE e SERVIDOR, tendo o cuidado de a cada "*send*" do *socket* cliente receber um "received" do socket CACHE ou do socket SERVIDOR para anular o diálogo.

Um ciclo *while* é desenhado para alocar os dois tipos de clientes enunciados anteriormente na Introdução. Para o cliente público e para o cliente privado são criados separadamente dois *sockets* através do método *socket()* designados de *clientSocket* com dois parâmetros cada. O primeiro, AF_INET, indica que a rede subjacente utiliza a convenção IPv4 e o segundo parâmetro, SOCK_STREAM, indica que se trata de um socket TCP. De notar que não é especificado o número da porta do socket do cliente deixando-se a cargo do sistema operativo essa tarefa.

Tratando-se de uma ligação TCP (orientada à conexão), é necessário efetuar numa primeira fase a ligação para posteriormente efetuar a transferência de dados, quer num quer noutro sentido. Assim, é utilizado o método *connect()*, para estabelecer a ligação entre o CLIENTE e o SERVIDOR, com dois parâmetros que indicam o endereço do servidor: para a CACHE tem-se (‘*localhost*’, 20000) e para o SERVIDOR tem-se (‘*localhost*’, 30000).

Associado a cada um dos *sockets* são desenhados os serviços disponíveis, designadamente, **LIST**, **UPLOAD** e **DOWNLOAD** - ver **Figura 2-1**, **Figura 2-2** e **Figura 2-3**.

```
def listar(s):
    s.send("LIST".encode('utf-8'))
    print(s.recv(1024).decode('utf-8'))
```

Figura 2-1: Serviço LIST

```
def upload(s,nf):
    fn = "C:\\Users\\Palikir\\Dropbox\\rcruz-rouxinol\\CLIENTE\\" + nf
    s.send("UPLOAD".encode('utf-8'))
    s.send(nf.encode('utf-8'))

    try:
        f1 = open(fn,'rb')
    except NameError as txt_error:
        print("\nError: Not file exist\n\n" + txt_error)

    f1.seek(0,2)
    sizeFile = f1.tell()
    f1.seek(0)
    print("File size: " , sizeFile , " Bytes")

    for x in f1:
        s.send(x)
    print("Successfully upload")
    f1.close()
```

Figura 2-2: Serviço UPLOAD

```
def download(s,nf):
    fn = "C:\\Users\\Palikir\\Dropbox\\rcruz-rouxinol\\CLIENTE\\" + nf
    s.send("DOWNLOAD".encode('utf-8'))
    s.send(nf.encode('utf-8'))

    sizeFile = s.recv(1024).decode('utf-8')
    print("File size: " , sizeFile , " Bytes")

    f2 = open(fn,'wb')
    while True:
        x = s.recv(1024)
        if not x:
            print("Successfully download")
            f2.close()
            break
        f2.write(x)
    pass
```

Figura 2-3: Serviço DOWNLOAD

Por fim, não havendo interesse em continuar com a ligação o cliente seleciona a opção **QUIT** e o método *close()* do socket cliente é invocado, desligando assim a ligação entre o CLIENTE e o SERVIDOR. Este método TCP envia uma mensagem do cliente ao servidor a proclamar encerramento da ligação.

3 Cache

A implementação do *socketCACHE* consiste num algoritmo que, importando a livraria *socket* do Python, permite realizar a ligação TCP entre o CLIENTE e a CACHE, tendo o cuidado de a cada "*send*" do *soketCACHE* receber um "*received*" do *socketcliente* para anular o diálogo.

Uma definição *clientthread()* com um argumento *connectionSocket*, é desenhada para alocar dentro de um ciclo *while* os três serviços disponíveis. No entanto numa primeira fase é criado o socket CACHE para permitir a ligação. O *socket* CACHE criado é designado de *cacheSocket* e, tal como referido anteriormente para o *socket* CLIENTE, é criado através do método *socket()*, com dois parâmetros iguais aos que dizem respeito ao *socket* CLIENTE – o que diz respeito ao endereço da porta da CACHE obviamente.

A seguir através da invocação do método *bind()* com dois parâmetros (o endereço da CACHE – IP e porto) é estabelecida a ligação anfitriã. Finalmente invocando o método *listen()* com um argumento (o número máximo de clientes que podem ficar em fila) o *socket* CACHE fica em *modo escuta*, i.e., à espera que um cliente "bata à porta". Quando um cliente x "bate à porta" é invocado o *método accept()* do socket, neste caso do *socket* designado de *cacheSocket*, que cria um novo socket na cache designado de *connectionSocket* dedicado ao cliente x - ver **Figura 3-1**. O CLIENTE e a CACHE completam os 3 apertos de mão (**Figura 1-1**), criando-se a ligação entre o *clientSocket* e a *connectionSocket*. Com o estabelecimento da ligação o LIST, UPLOAD e DOWNLOAD podem ser executados bastando para isso organizar a informação em Bytes e enviar de um lado para o outro. A ligação TCP tem a vantagem de garantir a transferência de todos os Bytes e ainda garante que os Bytes chegam por ordem correta. No final havendo um QUIT do lado do cliente é invocado do lado da CACHE o método *close()* do *connectionSocket* do cliente que desligou a ligação. O *socketcacheSocket* continua em modo de escuta.

```
while 1:
    connectionSocket, addr = cacheSocket.accept()
    _thread.start_new_thread(clientthread,(connectionSocket,))

connectionSocket.close()
```

Figura 3-1: Ciclo *while* para manter o *connectionSocket* ativo

No interior do ciclo *while* incorporado na definição *clientthread()* são desenhados os serviços disponíveis. O serviço **DOWNLOAD,** um pouco mais complexo, incorpora um método adicional para verificação da existência ou não de um determinado ficheiro, solicitado pelo cliente, e age em conformidade: (1) se existe devolve o ficheiro; (2) se não existe entra em comunicação com o SERVIDOR, serve-se e devolve ao cliente.

O serviço **LIST** é solicitado por ambos os tipos de clientes, porém o cliente público solicita à CACHE a listagem dos ficheiros disponíveis na pasta pública do SERVIDOR. Uma ligação adicional CACHE - SERVIDOR é estabelecida para tal.

4 Servidor

A implementação do *socketSERVIDOR* consiste num algoritmo que, importando a livraria *socket* do Python, permite efectuar a ligação TCP entre o CLIENTE e o SERVIDOR ou entre a CACHE e o SERVIDOR, tendo o cuidado de a cada "*send*" do *soketSERVIDOR* receber um "*received*" do *socket* CLIENTE ou do socket CACHE para anular o diálogo.

A metodologia seguida para implementar o socket SERVIDOR é similar à seguida para implementar o socket CACHE pelo que, facilmente se pode fazer o paralelismo entre ambos. A grande diferença reside no facto do algoritmo SERVIDOR incorporar o método validação das credenciais do cliente, nomeadamente, o nome de utilizador e a *password* e de o método **DOWNLOAD** ser mais simples.

5 Conclusão

Em anexo apresenta-se os três algoritmos implementados utilizando a linguagem de programação Python. Os métodos específicos de *Networking - Socket Programming*, podem ser vistos na **Figura 5.1**.

MÉTODOS SERVER SOCKET
s.bind()
s.listen()
s.accept()
MÉTODOS CLIENT SOCKET
s.connect()
MÉTODOS GERAIS SOCKET
s.recv()
s.send()
s.close()

Figura 5-1: Métodos específicos para *sockets* em Python

A validação do programa foi realizada, tendo-se verificado que as exigências estabelecidas no enunciado do Trabalho 2 foram todas satisfeitas. Apresenta-se de seguida a **Figura 5-2** com uma captura de ecrã, mostrando 2 CLIENTES a utilizarem o serviço.

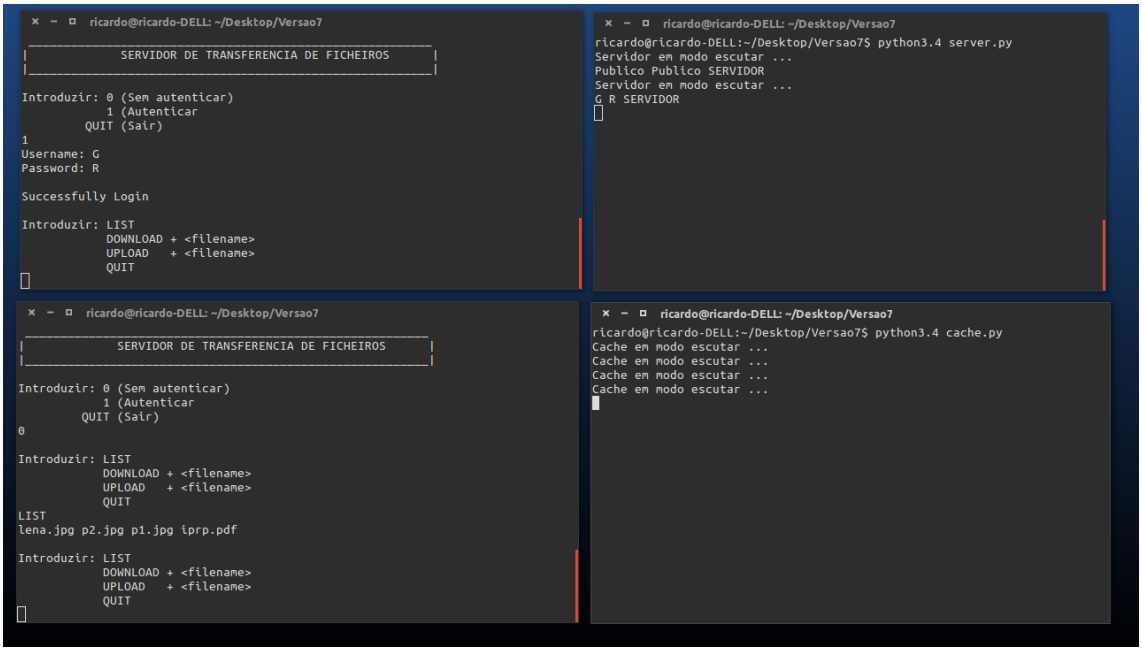

Figura 5-2: Aplicação a correr com multi-cliente

6. ANEXO

```
################################################################################
# Faculdade de Ciências e Tecnologia da Universidade de Coimbra                #
# Departamento de Engenharia Informática                                       #
# Unidade curricular: Introdução às Redes de Comunicação (2.°Ano / 1.° Sem.)   #
# Alunos: Ricardo Cruz e Gilberto Rouxinol                                     #
# Docentes: João P. Vilela e Tiago Cruz                                        #
#                                                                              #
# Projeto 2: Protocolo de Transferência de ficheiros com cache         2015/16 #
#                                                                              #
#                         Módulo "CLIENTE.PY"                                  #
#                                                                              #
################################################################################
import sys
import os
import getpass
from socket import *


def escolha():
   print("\nIntroduzir: LIST")
   print("            DOWNLOAD + <filename> ")
   print("            UPLOAD   + <filename> ")
   print("            QUIT")
   inCl = input("")
   return inCl


def cabecalho():
   print(" _____________________________________________________________ ")
   print("|             SERVIDOR DE TRANSFERENCIA DE FICHEIROS      |")
   print("|_____________________________________________________________|")
   print("\nIntroduzir: 0 (Sem autenticar)")
   print("            1 (Autenticar   ")
   print("         QUIT (Sair)")
```

```
    o = input("")
    return o


def listar(s):
    s.send("LIST".encode('utf-8'))
    print(s.recv(1024).decode('utf-8'))


def upload(s,nf):
    fn = "C:\\Users\\Palikir\\Dropbox\\rcruz-rouxinol\\CLIENTE\\" + nf
    s.send("UPLOAD".encode('utf-8'))
    s.send(nf.encode('utf-8'))


    try:
        f1 = open(fn,'rb')
    except NameError as txt_error:
        print("\nError: Not file exist\n\n" + txt_error)


    f1.seek(0,2)
    sizeFile = f1.tell()
    f1.seek(0)
    print("File size: " , sizeFile , " Bytes")


    for x in f1:
        s.send(x)
    print("Successfully upload")
    f1.close()


def download(s,nf):
    fn = "C:\\Users\\Palikir\\Dropbox\\rcruz-rouxinol\\CLIENTE\\" + nf
    s.send("DOWNLOAD".encode('utf-8'))
    s.send(nf.encode('utf-8'))


    sizeFile = s.recv(1024).decode('utf-8')
    print("File size: " + sizeFile + " Bytes")


    f2 = open(fn,'wb')
```

```
   while True:
      x = s.recv(1024)
      if not x:
         print("Successfully download")
         f2.close()
         break
      f2.write(x)
   pass


def valida_cliente(s):
   # Cliente fornece credenciais
   username = input("Username: ")
   password = input("Password: ") #password = getpass.getpass("Pass: ")
   cred = username + " " + password + " " + "SERVIDOR"
   s.send(cred.encode('utf-8'))
   return s.recv(1024).decode('utf-8')


cliente_em_atividade = 1


opcao = cabecalho()
if opcao == "QUIT":
   cliente_em_atividade = -1
   print ('\nExit ... \n')


while cliente_em_atividade > 0:
   if opcao == "0": #CLIENTE PUBLICO CLIENTE PUBLICO CLIENTE PUBLICO CLIENTE PUB
      # Cria Socket Cliente
      ender_Ch = ('localhost', 20000)
      clientSocket = socket(AF_INET, SOCK_STREAM)
      clientSocket.connect(ender_Ch)


      # Opcao do Cliente
      input_Cl = escolha()
      command = input_Cl.split()[0]
      if command == "LIST":
         listar(clientSocket)
```

```
    elif command == "UPLOAD":
        fup = input_Cl.split()[1]
        upload(clientSocket,fup)
    elif command == "DOWNLOAD":
        fdow = input_Cl.split()[1]
        download(clientSocket,fdow)
    elif command == "QUIT":
        clientSocket.send("QUIT".encode('utf-8'))
        cliente_em_atividade = -1
        clientSocket.close()
    else:
        print ("Option not found")
        escolha()


elif opcao == '1': #CLIENTE PRIVADO CLIENTE PRIVADO CLIENTE PRIVADO CLIENTE P
    # Cria Socket Cliente
    ender_Sv = ('localhost', 30000)
    clientSocket = socket(AF_INET, SOCK_STREAM)
    clientSocket.connect(ender_Sv)


    if valida_cliente(clientSocket) == "True":
        print("\nSuccessfully Login")
        # Opcao do Cliente logado
        input_Cl = escolha()
        command = input_Cl.split()[0]
        if command == "LIST":
            listar(clientSocket)
        elif command == "UPLOAD":
            fup = input_Cl.split()[1]
            upload(clientSocket,fup)
        elif command == "DOWNLOAD":
            fdow = input_Cl.split()[1]
            download(clientSocket,fdow)
        elif command == "QUIT":
            clientSocket.send("QUIT".encode('utf-8'))
            cliente_em_atividade = -1
```

```
            clientSocket.close()
         else:
            print("Erro: Sem tarefa para ordenar")
      else:
         print("Invalid username or password, Goodbye.")
         cliente_em_atividade = -1
         clientSocket.close()
   else:
      print ("Opcao invalida")
      cabecalho()
print ('\nA sair ... \n')
```

```
###############################################################################
# Faculdade de Ciências e Tecnologia da Universidade de Coimbra               #
# Departamento de Engenharia Informática                                      #
# Unidade curricular: Introdução às Redes de Comunicação (2.°Ano / 1.° Sem.)  #
# Alunos: Ricardo Cruz e Gilberto Rouxinol                                    #
# Docentes: João P. Vilela e Tiago Cruz                                       #
#                                                                             #
# Projeto 2: Protocolo de Transferência de ficheiros com cache         2015/16 #
#                                                                             #
#                          Módulo "CACHE.PY"                                  #
#                                                                             #
###############################################################################
from socket import *
import _thread
import sys
import os


#BEGIN: Estas funcoes sao para colocar no modulo iarc.py
def valida_publico(s):
   # Credenciais automaticas
   username = "Publico"
   password = "Publico"
   cred = username + " " + password + " " + "SERVIDOR"
   s.send(cred.encode('utf-8'))
   return s.recv(1024).decode('utf-8')
```

```
def download(s,nf):
   fn = "C:\\Users\\Palikir\\Dropbox\\rcruz-rouxinol\\CACHE\\Publico\\" + nf
   s.send("DOWNLOAD".encode('utf-8'))
   s.send(nf.encode('utf-8'))

   sizeFile = s.recv(1024).decode('utf-8')
   print("File size: " + sizeFile + " Bytes")

   f2 = open(fn,'wb')
   while True:
      x = s.recv(1024)
      if not x:
         print("Successfully download")
         f2.close()
         break
      f2.write(x)
   pass

#END: Estas funcoes sao para colocar no modulo iarc.py

# Cria Socket Cache
meu_ender_Ch = ('', 20000)
cacheSocket = socket(AF_INET,SOCK_STREAM)
cacheSocket.bind(meu_ender_Ch)
cacheSocket.listen(10)

def clientthread(connectionSocket):
   print("Cache em modo escutar ...")
   while 1:
      #connectionSocket, addr = cacheSocket.accept()
      command = connectionSocket.recv(1024).decode('utf-8')

      #********************************************************************* LIST
      if command == "LIST":
      # Cria Socket na Cache modo Cliente
```

```
        ender_S = ('localhost',30000)
        clientSocket_c = socket(AF_INET, SOCK_STREAM)
        clientSocket_c.connect(ender_S)
    # Cache vai Servidor
        if valida_publico(clientSocket_c) == "True":
           clientSocket_c.send(command.encode('utf-8'))
           lista = clientSocket_c.recv(1024).decode('utf-8')
           # Cache vai cliente
           connectionSocket.send(lista.encode('utf-8'))
     #****************************************************************** UPLOAD
     elif command == "UPLOAD":
        f_Up = connectionSocket.recv(1024).decode('utf-8')
        n = "C:\\Users\\Palikir\\Dropbox\\rcruz-rouxinol\\CACHE\\Publico\\" + f_Up
        f2 = open(n,'wb')
        while True:
           x = connectionSocket.recv(1024)
           if not x:
              print("Upload concluido")
              f2.close()
              break
           f2.write(x)
     #****************************************************************** DOWLOAD
     elif command == "DOWNLOAD":
        f_Dn = connectionSocket.recv(1024).decode('utf-8')
        pf = "C:\\Users\\Palikir\\Dropbox\\rcruz-rouxinol\\CACHE\\Publico\\"
        file_name = pf + f_Dn

        # Verifica existencia de f_Dn em pf
        filenames = os.listdir(pf)
        lista = ' '.join(filenames).split(" ")
        if f_Dn in lista:
           f1 = open(file_name,'rb')
           f1.seek(0,2)
           sizeFile = f1.tell()

           f1.seek(0)
```

```
            connectionSocket.send(str(sizeFile).encode('utf-8'))
            for x in f1:
               connectionSocket.send(x)
            f1.close()
         else:
            # Vai servidor
            ender_S = ('localhost',30000)
            clientSocket_c = socket(AF_INET, SOCK_STREAM)
            clientSocket_c.connect(ender_S)
            valida_publico(clientSocket_c)
            download(clientSocket_c,f_Dn)

            # Envia Cliente
            f1 = open(file_name,'rb')
            f1.seek(0,2)
            sizeFile = f1.tell()
            f1.seek(0)
            connectionSocket.send(str(sizeFile).encode('utf-8'))
            for x in f1:
               connectionSocket.send(x)
            f1.close()
      #********************************************************************** QUIT
      elif command == "QUIT":
         pass
      #else:
         #print("Erro: Sem tarefa para executar")
         #pass
   #connectionSocket.close()
while 1:
   connectionSocket, addr = cacheSocket.accept()
   _thread.start_new_thread(clientthread,(connectionSocket,))

connectionSocket.close()
```

```
################################################################################
# Faculdade de Ciências e Tecnologia da Universidade de Coimbra                #
```

```
# Departamento de Engenharia Informática                                        #
# Unidade curricular: Introdução às Redes de Comunicação (2.°Ano / 1.° Sem.)    #
# Alunos: Ricardo Cruz e Gilberto Rouxinol                                      #
# Docentes: João P. Vilela e Tiago Cruz                                         #
#                                                                               #
# Projeto 2: Protocolo de Transferência de ficheiros com cache         2015/16 #
#                                                                               #
#                          Módulo "SERVIDOR.PY"                                 #
#                                                                               #
#################################################################################
from socket import *
import _thread
import sys
import os


raiz_Dir = "C:\\Users\\Palikir\\Dropbox\\rcruz-rouxinol\\"


def valida_UserPass(utilizador,password):
   name = raiz_Dir + "SERVIDOR\\registosClientes.txt"
   f = open(name, 'r',encoding = 'utf-8')
   registos = f.readlines()
   f.close()
   list_up = []
   for i in registos:
      aux = i[:].split(',')
      list_up.append(aux)
   list_u = []
   list_p = []
   for i in range(len(list_up)):
      aux = list_up[i][0][:-1]
      if aux == "BEGIN":
         for j in range(i+1,len(list_up)):
            aux_u = list_up[j][0]
            if aux_u != "END":
               list_u.append(aux_u)
               aux_p = list_up[j][1][1:-1]
```

```
                list_p.append(aux_p)
    dic = {}
    dic = dict(zip(list_u,list_p))
    for i in range(len(list_u)):
        if utilizador == list_u[i]:
            p = dic.get(utilizador)
            if password == p:
                a = "True"
                break
            else:
                a = "False"
        else:
            a = "False"
    return a


# Cria Socket Servidor
meu_ender_S = ('',30000)
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind(meu_ender_S)
serverSocket.listen(10)


def clientthread(connectionSocket):
    print('Servidor em modo escutar ...')
    while 1:
        cred = connectionSocket.recv(1024).decode('utf-8')
        u, p, d = cred.split()[0], cred.split()[1], cred.split()[2]
        print(cred)
        aut = valida_UserPass(u, p)
        connectionSocket.send(aut.encode('utf-8'))


        comando = connectionSocket.recv(1024).decode('utf-8')
        #****************************************************************** LIST
        if comando == 'LIST':
            filenames = os.listdir(raiz_Dir + d + "\\" + u)
            lista = ' '.join(filenames)
            connectionSocket.send(lista.encode('utf-8'))
```

```
       #************************************************************** UPLOAD
      elif comando == 'UPLOAD':
         f_Up = connectionSocket.recv(1024).decode('utf-8')
         n = "C:\\Users\\Palikir\\Dropbox\\rcruz-rouxinol\\SERVIDOR\\" + u + "\\" + f_Up
         f2 = open(n,'wb')
         while True:
            x = connectionSocket.recv(1024)
            if not x:
               print("Upload concluido")
               f2.close()
               break
            f2.write(x)
       #************************************************************ DOWNLOAD
      elif comando == "DOWNLOAD":
         f_Dn = connectionSocket.recv(1024).decode('utf-8')
         file_name  =  "C:\\Users\\Palikir\\Dropbox\\rcruz-rouxinol\\SERVIDOR\\"  +  u   +
"\\" + f_Dn
         try:
            f1 = open(file_name,'rb')
         except NameError as txt_error:
            print("\nErro: Ficheiro inexistente !\n\n" + txt_error)


         f1.seek(0,2)
         sizeFile = f1.tell()
         f1.seek(0)
         connectionSocket.send(str(sizeFile).encode('utf-8'))


         for x in f1:
            connectionSocket.send(x)
         print("Successfully dowload")
         f1.close()
      #************************************************************** QUIT
      elif comando == "QUIT":
         pass
      #else:
       #  print("Erro: Sem tarefa para executar")
```

```
while 1:

   connectionSocket, addr = serverSocket.accept()

   _thread.start_new_thread(clientthread,(connectionSocket,))


connectionSocket.close()
```