****

**Relatório do 2º Trabalho Laboratorial**

***Computer Networks***

Uma imagem com texto, mapa

Descrição gerada automaticamente

Irene Amaral, up201506644

Rodrigo Caldas, up201708987

Tiago Ribeiro, up201708988

**Índice**

[Introdução 3](#_Toc27928072)

[Part 1: Development of download application 4](#_Toc27928073)

[Part 2: Configuration and Study of a Network 7](#_Toc27928074)

[ Part 2/Exp 1- Configure na IP Network 7](#_Toc27928075)

[ Part 2/Exp 2 - Implement two virtual LANs in a switch 8](#_Toc27928076)

[ Part 2/Exp 3 - Configure a Router in Linux 10](#_Toc27928077)

[ Part 2/Exp 4 - Configure a Commercial Router and Implement NAT 11](#_Toc27928078)

[ Part 2/Exp 5 – DNS 12](#_Toc27928079)

[ Parte 2/Exp 6 – TCP Connections 13](#_Toc27928080)

[Conclusão 15](#_Toc27928081)

[Anexos 16](#_Toc27928082)

Introdução

Este segundo trabalho laboratorial encontra-se dividido em duas partes. A primeira parte, relacionada com a implementação de uma aplicação de download FTP, é desenvolvida em ambiente de programação C.

A segunda parte do trabalho será a elaboração e configuração de uma rede de computadores.

A junção das duas partes terá como objetivo final o download de um ficheiro de forma a verificar a funcionalidade tanto da aplicação de download como da própria rede que foi configurada ao longo das aulas práticas.

Part 1: Development of download application

A parte 1 do trabalho tem como objetivo desenvolver uma aplicação que faça o download de qualquer tipo de ficheiros de um servidor FTP através de um link como argumento inserido pelo utilizador do tipo:

**ftp://[<user>:<password>@]<host>/<url-path>**

* **Biblioteca de funções: biblio.h**

**struct TCP** { (estrutura TCP)

int socket\_controlo; (descritor de ficheiro para o socket ​de controlo)

int socket\_dados; (descritor de ficheiro para o ​socket ​de dados)

char ipnovo[tamanho\_maximo]; (string para o novo ​ip)

int portanova; (um inteiro da nova porta)

};

**struct URL** { (responsável pelo armazenamento das informações retiradas do argumento especificado na linha de comandos)

char username[tamanho\_maximo];

char password[tamanho\_maximo];

char host[tamanho\_maximo];

char ip[tamanho\_maximo];

char path[tamanho\_maximo];

char ficheiro[tamanho\_maximo];

int porta; (é 21 o número da porta de controlo do protocolo ​FTP)

};

**int conexao**(char \* ip, int porta);   
(inicialização de um ​socket ​de controlo (socket\_controlo) e conexão ao server ftp)

**int login**(char \* username, char \* password);

**int ler\_url**(const char \*str);   
(recebe como argumento o input do utilizador e armazena a informação contida no argumento na estrutura acima referida)

**int modopassivo**();   
(é enviado o comando ​PASV que envia um pedido ao servidor ​FTP ​para transferir dados em modo passivo, dando a responsabilidade ao cliente pela abertura da ligação ​TCP ​para os dados)

**int nomedoficheiro**(const char \* path);   
(recupera o nome do arquivo por todo o path)

**int enviar\_RETR**(char \* path);   
(envia o comando RETR necessário para inicializar a transferência)

**int ip\_host**(const char \*host);(recupera ip através do nome do host e define o ip da estrutura)

**int download**(char \* ficheiro);  
(a receção dos dados (através do socket de dados) e a respetiva escrita dos mesmo em disco)

**int ler\_resposta**(int socket\_controlo, char \* buffer);(é lida a resposta dada pelo servidor através do ​socket ​de controlo (retorna 0 negativo ou 1 positivo)

**int msg\_para\_servidor**(int socket\_controlo, char \* cmd);   
(escreve msg para o servidor pelo socket de controlo)

**int desligar\_sockets**();   
(envio do comando ​QUIT​; irá fechar os ​sockets ​abertos anteriormente e libertar a memória alocada ao longo da execução)

* **Processador do url: url.c**

Relativamente ao URL inserido pelo utilizador no argv[1] é feito o seu processamento e validação. O nome do ficheiro é obtido a partir do *path contido no URL através da função “nomeficheiro”* e o endereço *ip* a partir da função “*ip\_host*”.

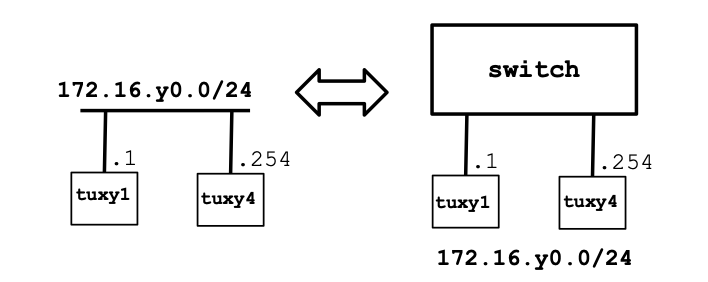
* **Aplicação de download – ftp\_app.c**

Inicialmente, é aberto o socket para a conexão entre o cliente e o servidor, onde são enviados os comandos USER e PASS para efetuar o login. De seguida entra em modo passivo chamando o comando PASV, o qual retorna a nova porta que serve para enviar e receber dados. Depois é enviado o comado RETR que pede o ficheiro e efetua o download do mesmo utilizando a função “*download*”. Por último, são fechados os sockets anteriormente abertos utilizando a função “*desligar\_sockets*”.

Part 2: Configuration and Study of a Network

* **Part 2/Exp 1- Configure an IP Network**

A experiência 1 tem como objetivo ligar o tuxy1 ao tuxy4 usando as configurações *default* do switch.

****

1. **What are the ARP packets and what are they used for?**

O ARP (*Address Resolution Protocol*) é um protocolo de comunicação que serve para descobrir o endereço da camada de ligação associado ao endereço IPv4. É utilizado para mapear o endereço de rede a um endereço físico (endereço MAC).

1. **What are the MAC and IP addresses of ARP packets and why?**

Ao fazer ping do tuxy1 para o tuxy4, o tuxy4 envia um pacote do tipo ARP a perguntar quem é o tuxy que lhe está a enviar algo – figura 1 e 2.

1. **What packets does the ping command generate?**

Primeiro são gerados pacotes ARP e de seguida são gerados pacotes ICMP (Internet Control Message Protocol).

1. **What are the MAC and IP addresses of the ping packets?**

Os pacotes ping têm os endereços MAC e IP do computador onde é enviado e os endereços MAC e IP do computador para onde será enviado (destino) – figura 3 (request) e figura 4 (reply).

1. **How to determine if a receiving Ethernet frame is ARP, IP, ICMP?**

Uma vez obtido o cabeçalho da trama, se tiver o valor 0x0800 é do tipo IP.  
Se o header IP tiver o valor 1, o tipo de protocolo é ICMP – figura 5.   
Se o cabeçalho tiver o valor 0x0806, é do tipo ARP – figura 6.

1. **How to determine the length of a receiving frame?**

Como é possível ver na figura 7, o Wireshark permite consular o tamanho da trama.

1. **What is the loopback interface and why is it important?**

A interface loopback é uma interface virtual da rede que permite ao computador receber respostas dele mesmo para testes da rede – figura 8.

* **Part 2/Exp 2 - Implement two virtual LANs in a switch**

Na experiência 2 foram criadas duas VLAN’s Y0 e Y1. O tuxy1 e tuxy4 fazem parte da VLANY0 enquanto o tuxy2 faz parte da VLANY1.

Uma imagem com captura de ecrã

Descrição gerada automaticamente

1. **How to configure vlany0?**

# criação da VLANY0  
configure terminal  
vlan 50  
end

# configuração das portas da VLANY0 (tuxy1)  
configure terminal  
interface fastethernet 0/1  
switchport mode access  
switchport access vlan 50  
end

# configuração das portas da VLANY0 (tuxy4)  
configure terminal  
interface fastethernet 0/4  
switchport mode access  
switchport access vlan 50  
end

1. **How many broadcast domains are there? How can you conclude it from the logs?**

Existem dois broadcast domains, um por cada VLAN criada. O primeiro contém o tuxy1 e 4 o segundo contém apenas o tuxy2.

* **Part 2/Exp 3 - Configure a Router in Linux**

Na experiência 3, o tuxy4 terá a configuração de um router entre as duas VLAN’S Y0 e Y1.

Uma imagem com captura de ecrã

Descrição gerada automaticamente

1. **What routes are there in the tuxes ? What are their meaning?**

Rotas do tuxy4 – figura9;  
Rotas do tuxy1 – figura10;  
Rotas do tuxy2 – figura11;

1. **What information does an entry of the forwarding table contain?**

Contém informação do destino (para onde vai ser enviado), do gateway (para onde a mensagem é redirecionada), da genmask (máscara de rede para a rede de destino), da Iface (interface para onde o pacote é enviado) e da Metric, Ref e Flags.

1. **What ARP messages, and associated MAC addresses, are observed and why?**

Uma mensagem ARP é enviada por um tux ao outro quando o primeiro não reconhece o endereço MAC do outro. Essa mensagem vai ter o endereço MAC do tux que envia e também o endereço MAC desconhecido do segundo tux, isto é, 00:00:00:00:00:00 pois não sabe qual o seu MAC.

1. **What ICMP packets are observed and why?**

São observados pacotes ICMP de request e reply no caso de os tux já se conseguirem encontrar. Se ainda não existirem rotas são observados pacotes ICMP HOST UNREACHABLE.

1. **What are the IP and MAC addresses associated to ICMP packets and why?**

Os endereços IP e MAC associados a pacotes ICMP são os endereços IP e MAC dos tux de origem e destino.

* **Part 2/Exp 4 - Configure a Commercial Router and Implement NAT**

Na quarta experiência, o router foi inicialmente configurado sem NAT e posteriormente com NAT. Esta última configuração já permitia o acesso à Internet por parte dos computadores.

Uma imagem com mapa, texto

Descrição gerada automaticamente

1. **How to configure a static route in a comercial router?**

configure terminal  
ip route <ip target> <mask> <ip gateway>  
exit

1. **What are the paths followed by the packets in the experiments carried out and why?**

Se existir na routing table é encaminhado para o destino indicado na rota correspondente.

1. **How to configure NAT in a comercial router?**

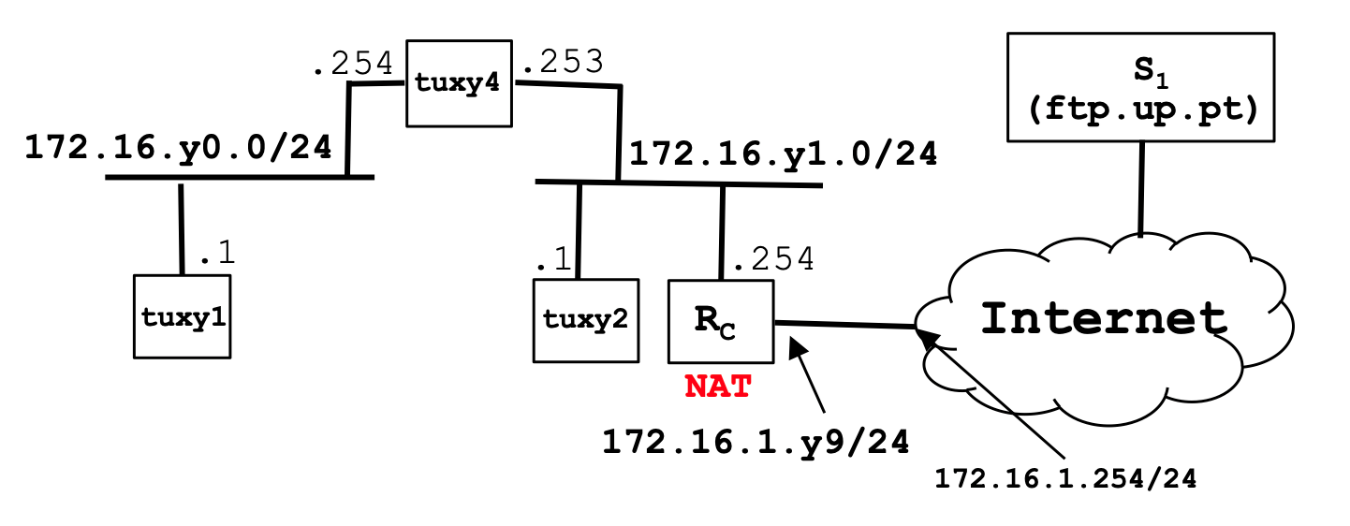
Para tal foi necessário configurar a interface do processo NAT com os comandos disponíveis no guião do projeto.

1. **What does NAT do?**

O NAT (Network Address Translation) permite que os computadores de uma rede interna tenham acesso ao exterior, em que apenas um endereço IP único é exigido para representar um grupo de computadores fora da sua própria rede.

* **Part 2/Exp 5 – DNS**

Na experiência 5, foi configurado o DNS nos tux 1, 2 e 4. O DNS é responsável por traduzir a informação associada aos nomes dos domínios.



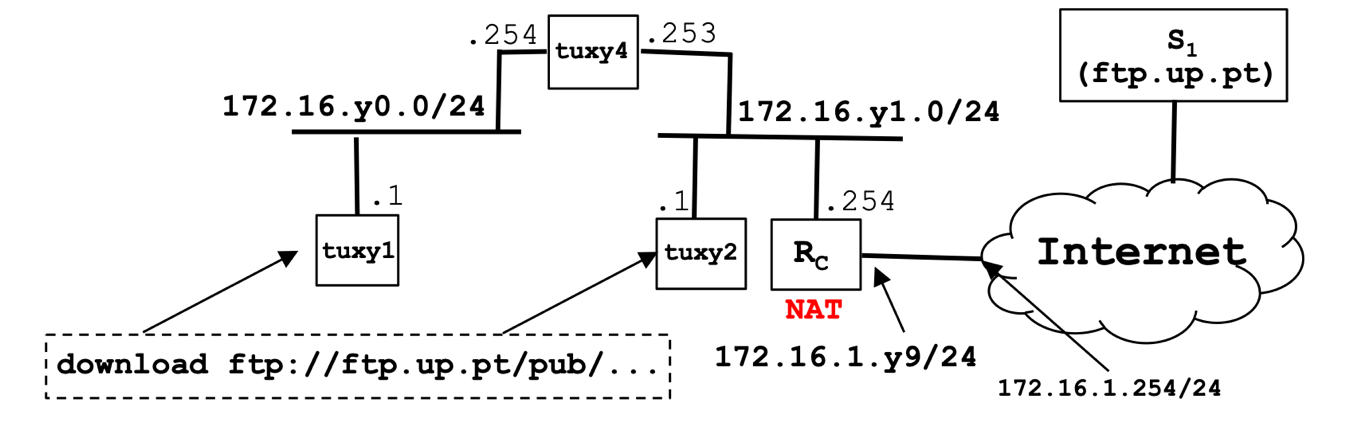
1. **How to configure the DNS service at an host?**

echo "search netlab.fe.up.pt" > /etc/resolv.conf (servidor DNS)  
echo "nameserver 172.16.2.1" >> /etc/resolv.conf (endereço IP)

Feito isto para cada tux já é possível a ligação à Internet.

1. **What packets are exchanged by DNS and what information is transported**

Inicialmente verificamos que foram enviados pacotes DNS para resolver o endereço IP ao qual tentávamos acessar, neste caso, [www.google.pt](http://www.google.pt), e de seguida foram enviados e recebidos pacotes ICMP (protocolo integrante do Protocolo IP) de *request* e *reply.*

* **Parte 2/Exp 6 – TCP Connections** 

1. **How many TCP connections are opened by your ftp application?**

Foram abertas duas conexões TCP, uma para mandar os comandos FTP ao servidor, tais como USER, PASV e RETR e para receber as respostas, e outra para receber os dados enviados pelo servidor e enviar as respostas do cliente.

1. **In what connection is transported the FTP control information?**

O controlo de informação é transportado na conexão TCP responsável pela troca de comandos tais como USER, PASV e RETR.

1. **What are the phases of a TCP connection?**

A conexão TCP é composta por três fases: estabelecimento de conexão, transferência de dados e termino da conexão.

1. **How does the ARQ TCP mechanism work? What are the relevant TCP fields? What relevant information can be observed in the logs?**

O mecanismo ARQ TCP utiliza o método da janela deslizante responsável pelo controlo de erros na transmissão de dados.

Para tal, os acknowledgment numbers indicam que a trama foi recebida com sucesso, o window size indica o tamanho da janela dos pacotes que o emissor pode enviar e por fim, o sequence number, que é o número do pacote a ser enviado.

1. **How does the TCP congestion control mechanism work?What are the relevant fields. How did the throughput of the data connection evolve along the time? Is it according the TCP congestion control mechanism?**

Este mecanismo baseia-se no número máximo de bytes que é possível transmitir e não ultrapassando assim este limite definido para que não exista obstrução da linha de transmissão.

1. **Is the through put of a TCP data connection disturbed by the appearance of a second TCP connection? How?**

Teoricamente, a existência de uma segunda conexão TCP terá como consequência a diminuição da taxa de transmissão, isto porque a taxa total de transmissão é partilhada pelas duas ligações em simultâneo.

Conclusão

Ao longo de todo o trabalho verificou-se uma aprendizagem gradual tanto ao nível teórico como prático ao nível de redes de computadores.

Tendo sido este o primeiro contacto que tivemos ao longo do curso com redes físicas de computadores, tanto na montagem como na elaboração da mesma ao nível da programação de comandos, consideramos uma mais valia para a aprendizagem e compreensão das mesmas.

Por fim, tanto a aplicação de download desenvolvida como a configuração da rede estão concordantes e cumprem juntas o objetivo pedido para este segundo trabalho que é o download final de um ficheiro.

Anexos

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 1

Uma imagem com captura de ecrã

Descrição gerada automaticamenteFigura 2

Uma imagem com captura de ecrã

Descrição gerada automaticamenteFigura 3

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 4

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Uma imagem com captura de ecrã

Descrição gerada automaticamenteFigura 5

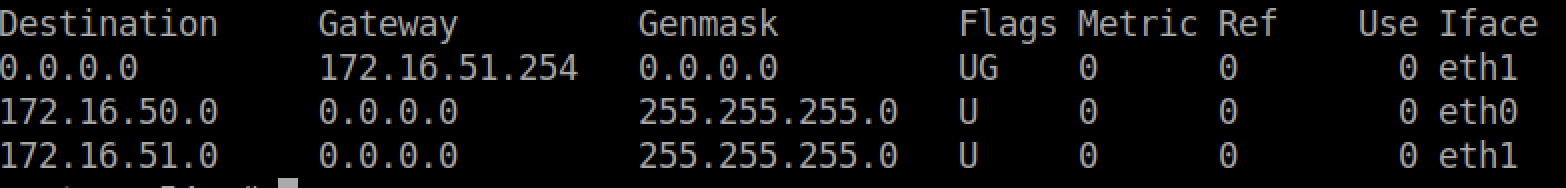
Figura 6

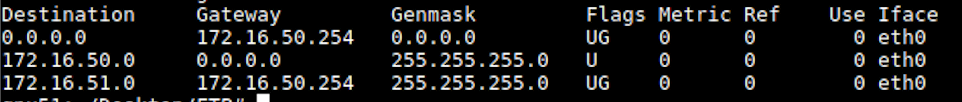
Uma imagem com captura de ecrã

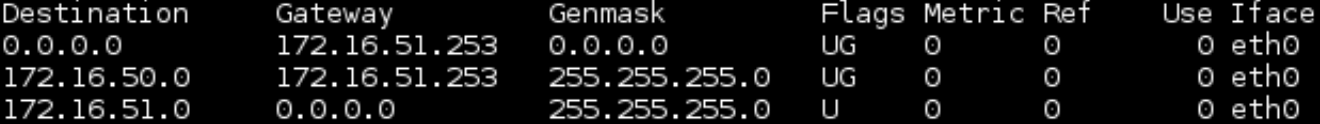
Descrição gerada automaticamenteFigura 7

Uma imagem com captura de ecrã

Descrição gerada automaticamenteFigura 8

Figura 9

Figura 10

Figura 11

Uma imagem com texto

Descrição gerada automaticamenteFigura 12 – Transferência apenas no tuxy1

Uma imagem com captura de ecrã

Descrição gerada automaticamente Figura 13 – Transferência no tuxy1 em simultâneo com o tuxy2

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 14 – Transferência no tuxy2 em simultâneo com o tuxy1

**Nota:** As figuras 12, 13 e 14 não correspondem ao que era previsto na teoria pois com a existência de uma segunda conexão TCP seria de esperar a diminuição da taxa de transmissão, isto porque a taxa total de transmissão é partilhada pelas duas ligações em simultâneo, mas a taxa de transmissão mantém-se bastante perto dos 1500 packets/s portanto não é coerente com o que estávamos à espera.

* **biblio.h**

#include <sys/types.h>

#include <sys/stat.h>

#include <sys/time.h>

#include <sys/socket.h>

#include <signal.h>

#include <unistd.h>

#include <errno.h>

#include <netdb.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <time.h>

#include <fcntl.h>

#include <termios.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define tamanho\_maximo 1024

struct TCP {

int socket\_controlo;

int socket\_dados;

char ipnovo[tamanho\_maximo];

int portanova;

};

struct URL

{

char username[tamanho\_maximo];

char password[tamanho\_maximo];

char host[tamanho\_maximo];

char ip[tamanho\_maximo];

char path[tamanho\_maximo];

char ficheiro[tamanho\_maximo];

int porta;

};

int conexao(char \* ip, int porta);

int login(char \* username, char \* password);

int ler\_url(const char \*str);

int modopassivo();

int nomedoficheiro(const char \* path);

int enviar\_RETR(char \* path);

int ip\_host(const char \*host);

int download(char \* ficheiro);

int ler\_resposta(int socket\_controlo, char \* buffer);

int msg\_para\_servidor(int socket\_controlo, char \* msg);

int desligar\_sockets();

* **url.c**
* #include "biblio.h"
* struct URL url;
* int ip\_host(const char \* host) {
* struct hostent \*h;
* if ((h = gethostbyname(host)) == NULL)
* {
* printf("erro a obter ip host com a funçao gethostbyname()");
* return -1;
* }
* char \*ip = inet\_ntoa(\*((struct in\_addr \*)h->h\_addr));
* strcpy(url.ip, ip);
* return 0;
* }
* int nomedoficheiro(const char \* path) {
* int i;
* int j=0;
* char ficheiro[256];
* size\_t str\_max = strlen(path);
* for(i=0; i<str\_max; i++) {
* if(path[i] == '/') {
* memset(ficheiro, 0, 20);
* j=0;
* continue;
* }
* if(i==str\_max-1) {
* ficheiro[j] = path[i];
* strcpy(url.ficheiro, ficheiro);
* break;
* }
* ficheiro[j] = path[i];
* j++;
* }
* return 0;
* }
* int ler\_url(const char \*str) {
* size\_t str\_max = strlen(str);
* char str\_aux[256];
* int i = 6;
* int j = 0;
* strncpy(str\_aux, str, i);
* memset(str\_aux, 0, 20);
* for(i = 6; i<str\_max; i++) {
* if(str[i] == ':') {
* strcpy(url.username, str\_aux);
* memset(str\_aux, 0, 20);
* j = 0;
* continue;
* }
* if(str[i] == '@') {
* strcpy(url.password, str\_aux);
* memset(str\_aux, 0, 20);
* j = 0;
* continue;
* }
* if(str[i] == '/'){
* if(strcmp(url.host, "") == 0) {
* strcpy(url.host, str\_aux);
* memset(str\_aux, 0, 20);
* j = 0;
* continue;
* }
* }
* if(i == str\_max - 1) {
* str\_aux[j] = str[i];
* strcpy(url.path, str\_aux);
* break;
* }
* str\_aux[j] = str[i];
* j++;
* }
* nomedoficheiro(url.path);
* if (ip\_host(url.host) < 0) {
* printf("Não encontra ip do host %s.\n", url.host);
* return -1;
* }
* return 0;
* }
* **ftp\_app.c**

#include "biblio.h"

#define tamanho\_maximo 1024

struct TCP tcp;

struct URL url;

int login(char \* username, char \* password) {

char username\_cmd[tamanho\_maximo];

char password\_cmd[tamanho\_maximo];

char buf[tamanho\_maximo];

sprintf(username\_cmd, "USER %s\r\n", username);

printf("%s\n", username\_cmd);

if (msg\_para\_servidor(tcp.socket\_controlo, username\_cmd) < 0) {

printf("Erro na escrita do USER\n");

return -1;

}

if (ler\_resposta(tcp.socket\_controlo, buf)) {

printf("Erro na leitura do USER\n");

return -1;

}

sprintf(password\_cmd, "PASS %s\r\n", password);

printf("%s\n", password\_cmd);

if (msg\_para\_servidor(tcp.socket\_controlo, password\_cmd) < 0) {

printf("Erro na escrita do PASS \n");

return -1;

}

if (ler\_resposta(tcp.socket\_controlo, buf)) {

printf("Erro na leitura do PASS\n");

return -1;

}

return 0;

}

int modopassivo() {

char buffer[tamanho\_maximo];

int ip1, ip2, ip3, ip4;

int porta1, porta2;

if (msg\_para\_servidor(tcp.socket\_controlo, "PASV\r\n") < 0) {

printf("Erro no comando PASV\n");

return -1;

}

if (ler\_resposta(tcp.socket\_controlo, buffer)) {

printf("Erro no comando PASV\n");

return -1;

}

if ((sscanf(buffer, "227 Entering Passive Mode (%d,%d,%d,%d,%d,%d)", &ip1, &ip2, &ip3, &ip4, &porta1, &porta2)) < 0) {

printf("Erro no modo passivo\n");

return -1;

}

sprintf(tcp.ipnovo, "%d.%d.%d.%d", ip1, ip2, ip3, ip4);

tcp.portanova = porta1 \* 256 + porta2;

printf("Novo IP: %s\n", tcp.ipnovo);

printf("Nova Porta: %d\n", tcp.portanova);

tcp.socket\_dados = conexao(tcp.ipnovo, tcp.portanova);

if (tcp.socket\_dados < 0) {

printf("\nNão foi possível conectar ao servido\n");

return -1;

}

return 0;

}

int enviar\_RETR(char \* path) {

char aux[tamanho\_maximo];

char buffer[tamanho\_maximo];

sprintf(aux, "\nRETR %s\r\n", path);

printf(">%s",aux);

if(msg\_para\_servidor(tcp.socket\_controlo, aux) < 0) {

printf("Erro a escrever o comando RETR\n");

return -1;

}

if(ler\_resposta(tcp.socket\_controlo, buffer) != 0){

printf("Erro a ler o comando RETR\n");

return -1;

}

return 0;

}

int download(char \* ficheiro){

char buffer[tamanho\_maximo];

FILE\* novo\_ficheiro;

novo\_ficheiro = fopen(ficheiro, "w");

int aux,res;

while ((res = read(tcp.socket\_dados, buffer, tamanho\_maximo))) {

if (res < 0) {

printf("Erro na receção(leitura) dos dados através do socket de dados\n");

return -1;

}

aux = fwrite(buffer, res, 1, novo\_ficheiro);

if (aux< 0) {

printf("Erro na escrita dos dados no novo ficheiro.\n");

return -1;

}

}

if(ler\_resposta(tcp.socket\_controlo, buffer)) {

printf("Erro na leitura da resposta\n");

return -1;

}

fclose(novo\_ficheiro);

printf("Download terminado com sucesso.TOP\n");

return 0;

}

int ler\_resposta(int socket\_controlo, char \* buffer) {

FILE \* f = fdopen(socket\_controlo, "r");

do {

memset(buffer, 0, tamanho\_maximo);

fgets(buffer, tamanho\_maximo, f);

printf("%s", buffer);

} while (!('1' <= buffer[0] && buffer[0] <= '5') || buffer[3] != ' ');

return buffer[0] >= '4';

}

int msg\_para\_servidor(int socket\_controlo, char \* msg) {

int aux;

aux = write(socket\_controlo, msg, strlen(msg));

return aux;

}

int conexao(char \* ip, int porta) {

struct sockaddr\_in server\_addr;

int fd;

bzero((char\*)&server\_addr,sizeof(server\_addr));

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_addr.s\_addr = inet\_addr(ip);

server\_addr.sin\_port = htons(porta);

if ((fd = socket(AF\_INET,SOCK\_STREAM,0)) < 0) {

return -1;

}

if(connect(fd, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) < 0){

printf("Erro na conexão ao servidor");

return -1;

}

return fd;

}

int desligar\_sockets() {

close(tcp.socket\_controlo);

close(tcp.socket\_dados);

printf("\nSockets fechados com sucesso.TOP\n");

return 0;

}

///////////////////////////////////////////////////MAIN//////////////////////////////////////

int main(int argc, char \*argv[]) {

if (argc != 2)

{

printf("\nNúmero de argumentos inválidos.\n");

return -1;

}

if (ler\_url(argv[1]) < 0) {

printf("\nO URL inserido não é válido.\n");

return -1;

}

printf("Utilizador: %s\n", url.username);

printf("Password: %s\n", url.password);

printf("Path: %s\n", url.path);

printf("Host: %s\n", url.host);

printf("IP: %s\n\n", url.ip);

printf("Nome do Ficheiro: %s\n", url.ficheiro);

tcp.socket\_controlo = conexao(url.ip, 21);

if (tcp.socket\_controlo < 0) {

return -1;

}

char buffer[tamanho\_maximo];

if(ler\_resposta(tcp.socket\_controlo, buffer)) {

printf("Erro na leitura do socket de controlo.\n");

return -1;

}

if(login(url.username, url.password) < 0) {

return -1;

}

if(modopassivo() < 0) {

return -1;

}

if(enviar\_RETR(url.path) < 0){

return -1;

}

if(download(url.ficheiro) < 0) {

return -1;

}

if(desligar\_sockets() < 0) {

return -1;

}

return 0;

}