**Trabalho 2**

Configuração de uma rede e desenvolvimento de uma aplicação de download

**Relatório Final**



Mestrado Integrado em Engenharia Informática e Computação

Redes de Computadores

Turma 4:

André Rodrigues Barros – up201303567

Leonardo Silva Ferreira – up201305980

Luís Miguel Azevedo Duarte – up201303585

Faculdade de Engenharia da Universidade do Porto Rua Roberto Frias, 4200-465 Porto, Portugal

19 Dezembro 2015

**Índice**

**Resumo……………………………………………………………..…...3**

**Introdução…………………………………………………………..…..4**

**Aplicação de download**

**Arquitetura……………………………………………………....5**

**Exemplo dos resultados da aplicação download……………...7**

**Configuração de uma rede**

**Experiência 1……………………………………………………..8**

**Experiência 2……………………………………………………..9**

**Experiência 3……………………………………………………..9**

**Experiência 4…………………………………………………….10**

**Experiência 5…………………………………………………….11**

**Experiência 6…………………………………………………….11**

**Conclusão…………………………………………………………...….12**

**Referências……………………………………………………………...12**

**Anexos…………………………………………………………………..13**

**Resumo**

Este relatório serve como complemento do segundo projeto da unidade curricular de Redes de Computadores, do MIEIC, na FEUP. O objectivo deste projecto consiste no desenvolvimento de uma aplicação de download de ficheiros e na configuração de uma rede de computadores, logo este relatório encontrar-se-á divido nessas duas respetivas metades, acompanhadas das suas respectivas subsecções.

A secção que refere a configuração de rede, foi abordada em seis experiências distintas propostas ao grupo. Tais experiências consistiam numa configuração de um IP de rede, de um router em Linux, de um router comercial e do DNS, e na implementação de duas VLAN’s(virtual Local Area Network)e do NAT (Network Address Translation), e de seguida, testar a aplicação de download desenvolvida anteriormente para verificação de um bom funcionamento nas ligações TCP(Transmission Control Protocol), sendo que todos estes conceitos e protocolos irão ser explicados com mais detalhe ao longo do relatório.

**Introdução**

Este segundo projeto divide-se em duas componentes:

- Desenvolvimento de uma aplicação de download.

- Configuração de uma rede.

Relativamente à a aplicação de download, o objetivo passava por ganhar um conhecimento mais profundo sobre o que é um cliente, um servidor e as suas especificidades em TCP/IP, a caracterização de protocolos em aplicações, como definir um URL e descrever o comportamento de um servidor FTP. De seguida, o grupo teria de implementar um cliente FTP e ligação TCP utilizando sockets.

O grande objetivo da configurar uma rede é permitir a execução de uma aplicação, neste caso a aplicação de download desenvolvida, num switch partilhado por duas *VLAN’s*, estando o *NAT* ativo numa destas. No entanto, para o correto funcionameto da aplicação a *VLAN* quer se encontra sem *NAT* terá de possuir uma ligação à Internet.

**Aplicação Download**

- Arquitetura

- Exemplo de resultados do download com a respetiva explicação

Uma das componentes a avaliar no segundo projeto de Redes de Computadores era o desenvolvimento de uma aplicação, em C, que permitisse fazer downloads de um servidor. Para o desenvolvimento desta aplicação, o grupo estudou vários links e documentos, entre os quais o RFC959 que explica detalhadamente o protocolo de transferência de dados, ou FTP.

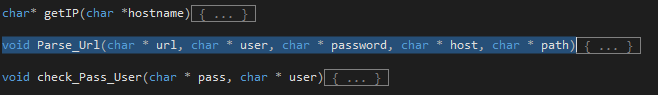
Iremos abordar detalhadamente todos os aspetos que envolvem esta aplicação assim como o método para a sua implementação.

**Arquitetura**

A aplicação download está dividida em 2 blocos: um dos blocos trata de extrair toda a informação necessária de um URL; outro contém todas as funções necessárias para o FTP assim como as responsarias pela ligação dos sockets aos servidores.

Estes dois blocos são conjugados no project.c permitindo ao utilizador descarregar um ficheiro de um servidor.

*Primeiro Bloco:*

 É composto pelos ficheiros URL.c e URL.h e permite á aplicação aceder á informação passada como parâmetro aquando na execução do download na linha de comandos.

**Figura 1**: Funções responsáveis pela extração da informação do URL assim como do IP do host.

*Funções:*

**getIP** -> permite á aplicação aceder ao ip do servidor. Este processo dá-se graças à função gethostbyname que retorna uma estrutura do tipo hostent, que é usada na função inet\_ntoa através de um cast para a estrutura do tipo in\_ddr e é devolvida um char\* no formato de pontos e números relativos ao IP. Recebe como parâmetro o host;

**Parse\_Url** -> Faz o parse de toda a informação exposta pelo link passado na linha de comandos. Guarda a informação do user, password, host e do path do ficheiro que pretende fazer download. Nesta função, também é considerado o facto de o link poder não ter paswords ou não ter user e password (neste caso o utilizador entraria em modo anonymous). Caso falte argumentos como o host ou path do file no link, então é lançada uma mensagem de erro na consola. Recebe como parâmetros o user, a password, o host, o path assim como o url;

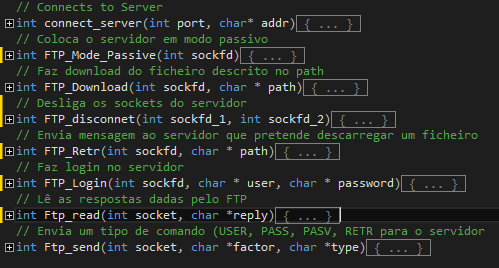
**Check\_Pass\_User** -> apenas faz print na consola para informar o utilizador se entrou em modo anonimo ou se não colocou a password no link passado na linha de comandos. Recebe como parâmetros o user e password do utilizador

*Segundo Bloco:*

Bloco que tem uma grande preponderância na aplicação do download e que é representado pelo clientFTP.c.

Após o parse do Url, é necessário ligar o cliente servidor FTP através de um socket TCP ao servidor FTP. Esta ligação é realizada pela função **connect\_server.**

De seguida, apresentaremos todas as funções necessárias assim como todos os passos a proceder para descarregar um ficheiro descrito no path inidicado pelo utilizador.



**Figura 2**: Funções que permitem aceder ao servidor e obter o ficheiro pretendido

*Funções:*

**connect\_server** -> responsável por estabelecer a ligação via sockets a um servidor. Recebe dois parâmetros: o endereço a que tem que se ligar assim como a porta (na primeira ligação que se estabelece com o servidor deve ser utilizada, por defeito, a porta 21 por se tratar de FTP). Se a ligação for bem-sucedida, é retornado o descritor da ligação que será necessária para os restantes processos;

**FTP\_Login** –> envia-se os comandos user e password ao servidor com a respetiva informação, através do Ftp\_Send, e recebe-se a resposta por parte do servidor, pela função Ftp\_read. Se após a transmissão do dados para autenticação, recebermos o código 530 (código que indica que indica que os dados de login estão errados) então o programa é terminado. Os parâmetros correspondem ao socket, ao user e á password;

**FTP\_Mode\_Passive** -> envia-se o comando PASV ao servidor. Uma vez feito este envio, recebemos uma resposta com o novo IP ao qual a aplicação download se deve associar para proceder á transferência do ficheiro assim como a nova porta a ligar. De seguida, repetimos o processo descrito na função connect\_server mas desta vez com o novo IP, com a nova porta associada a um novo socket (é calculada com base na resposta dada pelo servido após o envio do comando PASV). Recebe como parâmetro o socket;

**FTP\_Retr** -> envia-se o comando RETR ao servidor, através do primeiro primeiro socket com o path do ficheiro a descarrega. Toma como parâmetros o path do ficheiro e o socket;

**FTP\_Download** -> Cria-se um ficheiro com o mesmo nome que o ficheiro pretendido. De seguida, em cada iteração da leitura, é escrito para o novo ficheiro a informação contida no ficheiro original. Toma como parâmetros o path do ficheiro e o socket;

**FTP\_Disconnect** –> Após receber o o ficheiro corretamente (o utilizador recebe o código 226 que significa que recebeu o ficheiro corretamente), é enviado ao utlizador o comando QUIT que permite desconectar os sockets do servidor e fecha-se os sockets. Recebe os sockets envolvidos nas operações efetuadas pela aplicação.

**Exemplo dos resultados da aplicação download**

Testou-se a aplicação donwload com vários ficheiros de diferentes tipos e tamanhos e tudo correu como esperado.

Executou-se na consola -> ./proj <ftp://ftp:pass@ftp.up.pt/pub/robots.txt>

Output esperado na consola após execução da aplicação.

Info Aplication Download

User: ftp

Password: pass

Host: [ftp.up.pt](ftp://ftp.up.pt)

Path to file: pub/robots.txt

Start Connection…

IP Address: 193.136.37.8

Login…

Reply: 220 Bem-vindo à Universidade do Porto

Element sent: USER ftp

Reply: 331 Please specify the password.

Element sent: PASS pass

Reply: 230 Login successful.

Enter Pasv mode…

Element sent: PASV pasv

Reply: 227 Entering Passive Mode (193,136,37,8,180,67)

New IP Address: 193.136.37.8

Starts a new Connection…

Element sent: RETR pub/robots.txt

Reply: 10 Opening BINARY mode data connection for pb/robots.txt (23 bytes).

Filename described by path: robots.txt

New File received successfuly!!!

Reply: 226 File send OK.

Element sent: QUIT quit

Reply: 221 Goodbye

Disconnected from server

**Configuração de rede**

Segue-se agora a descrição das experiências realizadas que possibilitaram a configuração de uma rede. Os comandos utilizados como exemplo referem-se à estação de trabalho 2.

**Experiência 1**

Esta experiência consiste na configuração de um IP de Rede, ou seja, pretende-se que computadores diferentes sejam capazes de comunicar entre si através da configuração dos seus IP’s.

Inicialmente precede-se à configuração dos computadores através dos comandos:

* Ifconfig eth0 up
* Ifconfig eth0 172.16.20.1/24 (tux1) ou 172.16.20.254/24 (tux4)
* Route add default gw 172.16.20.254 (tux1)

Deste modo, as máquinas tux1 e tux4 são capazes de comunicar entre si através do switch sendo a gateway default o IP do tux4.

De seguida registou-se os IP’s e os endereços MAC dos dois computadores:

|  |  |  |
| --- | --- | --- |
|  | IP | MAC |
| tux1 | 172.16.20.1 | 00:0f:fe:8c:af:9d |
| tux4 | 172.16.20.254 | 00:22:64:a6:a4:f1 |

Posteriormente apagou-se as tabelas de ARP no tux1:

* arp –a
* arp –d 172.16.20.254

Finalmente, utilizou-se o comando ping de modo a analisar a ligação através do programa WireShark no qual foi possível verificar os requests e replys bem como os packets enviados e recebidos. (Registo log em anexo exp1.pcapng).

Da experiência realizada foi possível perceber como se configura uma rede IP, bem como verificar a importância de packets ARP, sendo que estes constituem um protocolo para encontrar um endereço da camada de ligações a partir do endereço da camada de rede, isto é, transforma endereços de IP em endereços MAC.

Na verdade, os endereços IP e MAC diferem sendo o primeiro o endereço do computador e o segundo o endereço físico do router.

Foi ainda possível perceber que é através do protocolo e do header do IP que se distingue uma trama de Ethernet de ARP, de uma IP e de uma ICMP.

Por fim, foi perceptível a importância de uma interface loopback pois esta constitui uma rede virtual que o computador utiliza para comunicar consigo mesmo. É utilizada principalmente para diagnostico e procura de erros e para conectar-se a servidores que se encontram a correr na máquina local.

**Experiência 2**

Esta segunda experiência tem como objectivo a implementação de duas VLAN’s no switch.

Inicialmente, procede-se à configuração do tux2 como exemplificado na experiência anterior.

|  |  |  |
| --- | --- | --- |
|  | IP | MAC |
| tux2 | 172.16.21.1 | 00:21:5ª:61:2b:72 |

De seguida procede-se à criação das VLAN’s 20 e 21:

* conf t
* vlan 20
* end
* conf t
* vlan 21
* end

Procede-se posteriormente à configuração das portas às respectivas VLAN’s (20 – tux1 e tux4, 21 – tux2):

* conf t
* interface fastethernet 0/Número da porta
* switchport mode access
* switchport access vlan 20/21
* end

Por fim procede-se à captura dos packets através do comando ping nos diferentes tux’s verificando-se que o tux1e tux4 são capazes de comunicar mas não conseguem comunicar com o tux2 que se encontra numa VLAN diferente (consultar os logs exp2\_log\_tux1.pcapng, exp2\_log2\_tux1.pcapng e exp2\_log3\_tux2.pcapng).

**Experiência 3**

Esta experiência teve como objectivo configurar a máquina tux4 como router entre as duas sub-redes criadas na experiência anterior. Para tal, foi necessário ligar a interface ethernet 1 do tux4 e configurá-la com um IP dentro da mesma gama que o tux2, posteriormente adicionando esta interface à sub-rede da máquina 2.

Após estes passos, foi adicionada uma rota ao tux1 com o comando:

* route add -net 172.16.y1.0/24 gw 172.16.y0.254

o primeiro endereço identifica a gama de endereços para a qual se quer adicionar a rota, sendo que o segundo é o IP para o qual se deve reencaminhar o pacote (tux4).

De seguida, repete-se o mesmo passo para a máquina 2, mas desta vez usando endereços distintos:

* route add -net 172.16.20.0/24 gw 172.16.21.253

sendo o segundo IP novamente o da máquina 4, mas na sub-rede comum ao tux2 e tux4.

Percorridos estes passos, foi possível efectuar *ping* da máquina 1 para a máquina 2, começando com um pedido para o IP do tux2, que é reencaminhado para o tux4, pois este encontra-se conectado a ambas as sub-redes, sendo, assim, possível aceder às duas máquinas (ao tux1 através da interface eth0 e ao tux2 através da eth1) e enviar os pacotes da primeira para a segunda máquina. O processo de resposta é exatamente o reverso, sendo o pacote reencaminhado da máquina 2, pela máquina 4 até à máquina 1. (consultar logs log\_exp3\_tux1.pcapng, log\_exp3\_tux2.pcapng, log\_exp3\_tux4.pcapng, log2\_exp3\_tux1.pcapng, log2\_exp3\_tux2.pcapng e log2\_exp3\_tux4.pcapng).

**Experiência 4**

Nesta experiência era pretendido configurar um router comercial com **NAT**.

A implementação do **NAT** foi necessária para possibilitar a comunicação entre as máquinas conectadas às redes criadas com redes externas. Tal necessidade provém do facto de ao estarem numa rede privada, os IPs dos computadores não serem reconhecidos fora dessa mesma. Para conseguir esse reconhecimento, a solução encontrada foi fazer um mapeamento baseado no IP interno e na porta local do computador. Com estes dois dados o **NAT** gera um número de 16 bits usando uma **tabela hash**, este número é então escrito no campo da porta de origem. Na resposta, o processo é revertido, e o router fica a saber para qual computador da rede interna deve enviar a resposta.

Para configurar o router, foi necessário configurar a interface interna no processo **NAT**. Para tal usou-se o comando

* interface fastethernet 0/0

seguido pela especificação do IP para essa interface com o comando

* ip address [ip] [mask]

(ip é 172.16.21.524 e a mask é 255.255.255.0, neste caso). De seguida, foi configurada a interface ligada ao router da sala, a interface 1, com os mesmos comandos de cima, mas com as seguintes diferenças:

* interface fastethernet 0/1
* ip address 172.16.1.29 255.255.255.0

Para evitar que as configurações se perdessem no acontecimento do router ser desligado foi inserido em ambos os casos o comando

* no shutdown

Posteriormente, para garantir a gama de endereços, introduziram-se os comandos:

* ip nat pool ovrld 172.16.1.29 172.16.1.29 prefix 24
* ip nat inside source list 1 pool ovrl overload

Num passo seguinte foi criada uma lista de acessos e permissões de pacotes, com o comando

* acesslist 1 permit [ip] [máximo]

(neste caso o IP utilizado foi 172.16.20.0 e 172.16.21.0 e máximo foi 0.0.0.255). Por fim, de modo definir as rotas internas e externas, aplicou-se os comandos:

* ip route 0.0.0.0 0.0.0.0 172.16.1.254
* ip route 172.16.20.0 255.255.255.0 172.16.21.253

para que, quando o IP de destino for 172.16.20.0, redireccionar os pacotes para o IP 172.16.21.253. A fim de teste foi executado um ping, apartir da máquina 1, para o router da sala verificando que os pacotes enviados passavam pela máquina 2, sendo reencaminhados para o router no IP 172.16.21.254.

**Experiência 5**

Com esta experiência pretendia-se conseguir aceder a redes externas, de modo a acceder à Internet, apartir da rede interna criada. Para o efeito foi necessária a configuração do DNS, alterando, em todos os hosts da rede interna, o ficheiro ***resolv.conf***, colocando ***“nameserver 172.16.1.1”***, sendo este o IP do servidor que deve ser acedido.

Para testar foi feito um ping a *www.google.com*, verificando que o DNS pergunta a informação contida num determinado nome de dominio, sendo a resposta o tempo de vida, tal como o tamanho dos pacotes de dados.

**Experiência 6**

Nesta última experiência, foi usada a aplicação de download desenvolvida na primeira parte deste projeto.

A fim de demonstrar que a rede estava corretamente configurada, usou-se se um servidor FTP e efectou-se o download de um ficheiro, sendo que este foi corretamente descarregado, provando a correta configuração.

O FTP utiliza o protocolo de entrega de tramas *Selective Repeat ARQ*. Com este protocolo o emissor continua a enviar as frames mesmo em caso de da perda de alguma delas. Contrariamente ao protocolo *Go-Back-N ARQ*, o receptor irá continuar a aceitar e a fazer o *acknowledgement* das restantes frames após o erro inicial. O recetor guarda o número de sequência da primeira trama que não recebeu e envia-a para o receptor aquando de cada *acknowlegment* seguinte. No final, quando o emissor já tiver enviado todas as tramas, reenvia a frame dada pelo número enviado pelo emissor, e resume de onde parou.

**Conclusão**

Em jeito de conclusão, após o término do segundo projecto da unidade curricular de RCOM, obtém-se uma análise positiva da execução do mesmo, pelo que considera-se que todos os elementos do grupo adquiriram os conceitos necessários para a configuração de uma rede. Note-se ainda que todos as metas propostas foram atingidas com qualidade, sendo que quer a aplicação de download quer as experiências foram conduzidas de forma correta e os respectivos resultados finais verificaram-se funcionais.

**Referências**

https://moodle.up.pt/pluginfile.php/30741/mod\_resource/content/5/lab2.pdf

<http://beej.us/guide/bgnet/output/print/bgnet_A4.pdf>

<http://cr.yp.to/ftp/retr.html>

**Anexos**

**Comandos**

- Comandos do reboot ao switch ->

+ configure terminal

+ no vlan 2-4094

+ exit

+ copy flash:tux2-clean startup-config

+ reload

- Comandos do reboot ao router ->

+ copy flash:tux2-clean start-config

+ reload

Experiencia 1:

Configurar tuxy1 e tuxy4:

/etc/init.d/networking restart

ifconfig eth0 up

ifconfig

ifconifg eth0 172.16.20.1/24 / ifconfig eht0 172.16.20.254/24

route add default gw 172.16.20.254

route-n

arp-a

arp -d 172.16.20.254

Experiencia 2:

Configurar o tuxy2.

+ /etc/init.d/networking restart

+ ifconfig eth0 up

+ ifconfig

+ ifconifg eth0 172.16.21.1/24

Criar vlany0 no tux1(manda) no switch

+ No gtkterm:

enable

configure terminal

vlan 20

end

+ Para as portas:

configure terminal

interface fastethernet 0/Numero da porta do tux1 e do tux4

switchport mode access

switchport access vlan x0

end

show running-config interface fastethernet 0/Numero da porta do tux1 e do tux4

show interfaces fasethernet 0/Numero da porta do tux1 e do tux4 switchport

Repetir o processo para a vlan1

Experiencia 3:

Ligar o cabo

Configurar o tux4 na eth1:

+ ifconfig eth1 up

+ ifconfig

+ ifconifg eth1 172.16.21.253/24

+ echo 1 > /proc/sys/net/ipv4/ip\_forward

+ echo 0 > /proc/sys/net/ipv4/icmp\_echo\_ignore\_broadcasts

Associou se o tux 4 á vlan1

+ configure terminal

+ Interface fastethernet 0/Numero da porta do tux4

+ Switchport mode access

+ switchport access vlan 21

+ end

+ route add –net 172.16.20.0/24 gw 172.16.21.253

+ apaga-se os ips dos 3 tux através do arp –d ip

Experiencia 4

Configurar router:

Ligar o cab da net ao geo1 e ligar o go0 a uma porta

Tira-se o cabo do switch e liga se ao router

+ conf t

+ interface gigabitethernet 0/0

+ ip address 172.16.21.254 255.255.255.0

+ no shutdown

+ exit

+ interface gigabitethernet 0/1

+ ip address 172.16.1.29 255.255.255.0

+ no shutdown

+ exit

+ ip route 0.0.0.0 0.0.0.0 172.16.1.254

+ ip route 172.16.20.0 255.255.255.0 172.16.21.253

+ end

Liga se o tux2 ao Rc:

+ route add default gw 172.16.21.254

Fazer o mesmo com o tux4

Ligar o router á vlan 21

+ configure terminal

+ interface fastethernet 0/Numero da porta do router

+ switchport mode access

+ switchport access vlan 21

+ end

Pings a funcionar para todos.

No tux2

+ echo 0 > /proc/sys/net/ipv4/conf/eth0/accept\_redirects

+ echo 0 > /proc/sys/net/ipv4/conf/all/accept\_redirects

+ route del –net 172.16.20.0 gw 172.16.21.253 netmask 255.255.255.0 dev eth0

+ traceroute 172.16.20.1

+ route add –net 172.16.20.0 gw 172.16.21.253 netmask 255.255.255.0 dev eth0

+ traceroute 172.16.20.1

+ echo 1 > /proc/sys/net/ipv4/conf/eth0/accept\_redirects

+ echo 1 > /proc/sys/net/ipv4/conf/all/accept\_redirects

Configurar o router com nat:

+ conf t

+ interface gigabitethernet 0/0

+ ip address 172.16.21.254 255.255.255.0

+ no shutdown

+ ip nat inside

+ exit

+ interface gigabitethernet 0/1

+ ip address 172.16.1.29 255.255.255.0

+ no shutdown

+ip nat outside

+ exit

+ ip route 0.0.0.0 0.0.0.0 172.16.1.254

+ ip route 172.16.20.0 255.255.255.0 172.16.21.253

+ end

+ ip nat pool ovrld 172.16.1.29 172.16.1.29 prefix 24

+ ip nat inside source list 1 pool ovrld overload

+ access-list 1 permit 172.16.20.0 0.0.0.7

+ access-list 1 permit 172.16.21.0 0.0.0.7

+ ip route 0.0.0.0 0.0.0.0 172.16.1.254

+ ip route 172.16.20.0 255.255.255.0 172.16.21.253

+ end

Experiencia 5

No ficheiro resolv.conf:

search lixa.netlab.fe.up.pt nameserver 172.16.1.1

**Código**

**Ficheiro: project.c**

#include "project.h"

int main(int argc, char\*\* argv)

{

if(argc != 2)

{

printf("Wrong number os arguments in function main\n");

return -1;

}

char host[STRING\_SIZE];

char user[STRING\_SIZE];

char password[STRING\_SIZE];

char path[STRING\_SIZE];

// Get all the info about URL

Parse\_Url(argv[1], user, password, host, path);

printf("\nInfo Aplication Download\n\n");

printf("User: %s\n", user);

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

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

printf("Path to File: %s\n", path);

// Get IP from host

char\* Server\_Address = getIP(host);

// Prepare Connection

printf("\nStart Connection...\n\n");

printf("IP Address: %s\n\n", Server\_Address);

/// Start Connection

int sockfd;

sockfd = connect\_server(FTP\_PORT, Server\_Address);

printf("Login...\n");

check\_Pass\_User(password, user);

char reply[STRING\_SIZE];

Ftp\_read(sockfd, reply);

//Login

FTP\_Login(sockfd, user, password);

printf("\nEnter Pasv mode...\n\n");

//Pasv mode

int sockfd\_2 = FTP\_Mode\_Passive(sockfd);

//Retr mode

FTP\_Retr(sockfd, path);

//Download File

FTP\_Download(sockfd\_2, path);

//Disconnects from server

FTP\_disconnet(sockfd, sockfd\_2);

return 0;

}

**Ficheiro: project.h**

#include <stdio.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

#include <netdb.h>

#include <string.h>

#define FTP\_SIZE 1024

#define STRING\_SIZE 255

#define FTP\_PORT 21

#define FAIL\_LOGIN 530

int connect\_server(int port, char\* addr);

void check\_Pass\_User(char \* pass, char \* user);

int FTP\_Login(int sockfd, char \* user, char \* password);

int FTP\_disconnet(int sockfd\_1, int sockfd\_2);

int FTP\_Mode\_Passive(int sockfd);

int FTP\_Retr(int sockfd, char \* path);

int FTP\_Download(int sockfd, char \* path);

int Ftp\_read(int socket, char \* reply);

int Ftp\_send(int socket, char \*factor, char \*type);

char\* getIP(char \*hostname);

void Parse\_Url(char \* url, char \* user, char \* password, char \* host, char \* path);

**Ficheiro: URL.c**

#include "project.h"

char\* getIP(char \*hostname){

struct hostent \*h;

if((h = gethostbyname(hostname)) == NULL)

{

herror("gethostbyname");

exit(2);

}

char \*ip = inet\_ntoa(\*((struct in\_addr \*)h->h\_addr));

return ip;

}

void Parse\_Url(char \* url, char \* user, char \* password, char \* host, char \* path)

{

if(sscanf(url, "ftp://%[^:]:%[^@]@%[^/]/%s", user, password, host, path) == 4)

{

return;

}

if (sscanf(url, "ftp://%[^@]@%[^/]/%[^\n]", user, host, path) == 3)

{

strcpy(password, "pass\_temp");

return;

}

else if (sscanf(url, "ftp://%[^:@/]/%[^\n]", host, path) == 2)

{

strcpy(user, "anonymous");

strcpy(password, "pass\_temp");

return;

}

else

{

fprintf(stderr, "Invalid URL! Usage: ftp ftp://<user>:<password>@<host>/<url-path>\n");

exit(2);

}

}

void check\_Pass\_User(char \* pass, char \* user)

{

if(strcmp (pass, "pass\_temp") == 0)

printf("\nBy default your password is pass\_temp. Fill the password field next time.\n");

if(strcmp (user, "anonymous") == 0)

printf("\nYou have entered on anonymous mode.\n\n\n");

}

**Ficheiro: clientFTP.c**

#include "project.h"

// Connects to Server

int connect\_server(int port, char\* addr){

int sockfd;

struct sockaddr\_in server\_addr;

/\*server handler address\*/

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

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

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

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

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

perror("socket()");

exit(0);

}

/\*connect to the server\*/

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

perror("connect()");

exit(0);

}

return sockfd;

}

// Coloca o servidor em modo passivo

int FTP\_Mode\_Passive(int sockfd)

{

char reply[STRING\_SIZE];

int pasv\_1, pasv\_2, pasv\_3, pasv\_4, pasv\_5, pasv\_6;

//Sends pasv to server

if(Ftp\_send(sockfd, "pasv", "PASV") < 0)

{

printf("Error sending pasv to server\n");

return -1;

}

//Reads from the server

if(Ftp\_read(sockfd, reply) < 0)

{

printf("Error while receiving answer from pasv information \n");

return -1;

}

sscanf(reply, "%\*[^(](%d,%d,%d,%d,%d,%d)\n", &pasv\_1, &pasv\_2, &pasv\_3, &pasv\_4, &pasv\_5, &pasv\_6);

char ip[256];

int port = pasv\_5\*256 + pasv\_6;

sprintf(ip,"%d.%d.%d.%d", pasv\_1, pasv\_2, pasv\_3, pasv\_4);

// Get IP from host

printf("New IP Address: %s\n\n", ip);

char\* Server\_Address = getIP(ip);

printf("\nStarts a new Connection...\n\n");

printf("New IP Address: %s\n\n", Server\_Address);

// Connects with the new ip

int sockfd\_2;

sockfd\_2 = connect\_server(port, Server\_Address);

if(sockfd\_2 < 0)

{

printf("Error connecting to the new socket\n");

return -1;

}

return sockfd\_2;

}

// Faz download do ficheiro descrito no path

int FTP\_Download(int sockfd, char \* path)

{

char buf[STRING\_SIZE];

char file\_name[450];

int i, j;

i = 0, j = strlen(path) - 1;

while(j > 0)

{

if(path[j] == '/')

{

j++;

break;

}

j--;

}

while(j < strlen(path))

{

file\_name[i]=path[j];

i++;

j++;

}

file\_name[i]='\0';

printf("Filename described by path: %s\n",file\_name);

int w;

FILE \*file\_download = fopen(file\_name, "w");

while((w=read(sockfd, buf, STRING\_SIZE)) > 0)

{

fwrite(buf,sizeof(char),w,file\_download);

}

fclose(file\_download);

printf("New File received successfuly!!!\n\n");

return 0;

}

// Desconecta do Servidor e fecha os sockets

int FTP\_disconnet(int sockfd\_1, int sockfd\_2)

{

char reply[STRING\_SIZE];

if(Ftp\_read(sockfd\_1, reply) < 0)

{

printf("Error after receiveing file\n");

return -1;

}

if(Ftp\_send(sockfd\_1, "quit", "QUIT") < 0)

{

printf("Error in sending quit command\n");

return -1;

}

if(Ftp\_read(sockfd\_1, reply) < 0)

{

printf("Error on disconnecing from server\n");

return -1;

}

if(close(sockfd\_1) < 0)

{

printf("Failed to close ftp socket 1.\n");

return -1;

}

if(close(sockfd\_2) < 0)

{

printf("Failed to close ftp socket 2.\n");

return -1;

}

printf("Disconnected from server.\n");

return 0;

}

// Envia mensagem ao servidor que pretende descarregar um ficheiro

int FTP\_Retr(int sockfd, char \* path)

{

char reply[STRING\_SIZE];

//Sends retr to server

if(Ftp\_send(sockfd, path, "RETR") < 0)

{

printf("Error sending retr with file path to server\n");

return -1;

}

//Reads from the server

if(Ftp\_read(sockfd, reply) < 0)

{

printf("Error while receiving answer from retr information \n");

return -1;

}

return 0;

}

// Faz login no servidor

int FTP\_Login(int sockfd, char \* user, char \* password)

{

//Checks user

char reply[STRING\_SIZE];

if(Ftp\_send(sockfd, user, "USER") < 0)

{

printf("Error sending user information\n");

return -1;

}

if(Ftp\_read(sockfd, reply) < 0)

{

printf("Error while receiving answer from user information \n");

return -1;

}

//Checks password

if(Ftp\_send(sockfd, password, "PASS") < 0)

{

printf("Error sending password information\n");

return -1;

}

if(Ftp\_read(sockfd, reply) < 0)

{

printf("Error while receiving answer from password information \n");

return -1;

}

int code;

sscanf(reply, "%d", &code);

if(code == FAIL\_LOGIN)

{

printf("Failed to login. Username or password don't exist.\n");

exit(2);

}

return 0;

}

// Lê as respostas dadas pelo FTP

int Ftp\_read(int socket, char \*reply)

{

int read\_bytes;

memset(reply, 0, STRING\_SIZE);

read\_bytes = read(socket, reply, STRING\_SIZE);

printf("Reply: %s\n", reply);

return read\_bytes;

}

// Envia um tipo de comando (USER, PASS, PASV, RETR para o servidor

int Ftp\_send(int socket, char \*factor, char \*type)

{

char buff[50];

int write\_bytes;

memset(buff, 0, 50);

strcat(buff, type);

strcat(buff, " ");

strcat(buff, factor);

strcat(buff, "\r\n");

printf("Element sent: %s\n", buff);

write\_bytes = write(socket, buff, strlen(buff));

return write\_bytes;

}