FCTUC FACULDADE DE CIÊNCIAS
E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

IRC
Introdução Redes e Comunicação

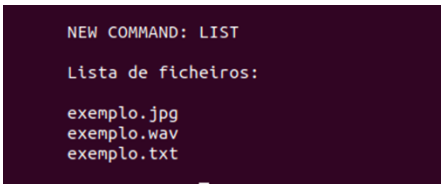
TRABALHO PRÁTICO

Curso de Engenharia Informática
2019/2020

Trabalho realizado no âmbito da cadeira de IRC por:
Diogo Ferreira Sobreira, 2018294861
Rafael Lopes Belo Baptista, 2018277007

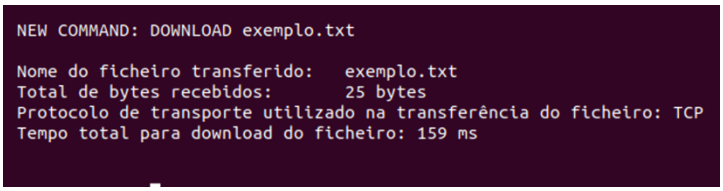
Manual de Utilizador

Quando se inicia o programa, é pedido ao utilizador que introduza um novo comando. Este tem como opção 3 comandos (“LIST”, “DOWNLOAD” e “QUIT”). Para a opção “LIST”, o cliente irá receber a lista do nome dos ficheiros do diretório “server_files”.



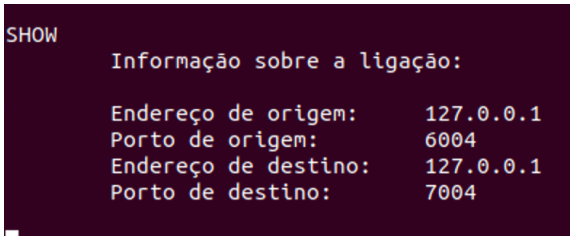
(1) Lista de ficheiros

A opção “DOWNLOAD” é acompanhada de outros dois comandos (“TCP/UDP” e “ENC/NOR”), isto é, se o download é feito por TCP ou UDP, com ou sem encriptação de dados.
Devido à falta de tempo, não desenvolvemos código para a transmissão de ficheiros por UDP. Após a escolha do ficheiro, o utilizador introduz o comando “DOWNLOAD <TC/UDP> <ENC/NOR> <NOME>” que transfere o referido ficheiro para o diretório do cliente, criado no início do programa.



(2) Exemplo de download

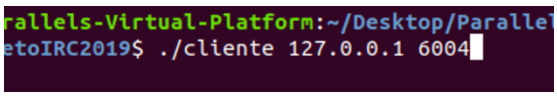
Enquanto o server e o cliente trocam informação entre si, a proxy trata de fornecer a informação sobre o que está a ser enviado para cada lado (server ou cliente), bem como executar 3 funções conforme os 3 comandos dados (“LOSSES”, “SHOW” e “SAVE”). Devido à impossibilidade de executar o código para UDP, foi ocultado a opção “LOSSES”, devido a apenas se referir à perda de pacotes por UDP. O comando “SHOW” informa o utilizador sobre a ligação em curso entre o cliente e o servidor (endereço e porto de origem, endereço e porto de destino, protocolo utilizado). Já o comando “SAVE” ativa ou desativa a gravação em ficheiro dos dados (ficheiros) transferidos do servidor para o cliente.



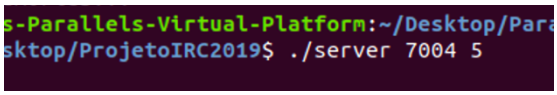
(3) Informação sobre a ligação

Manual de Programador

Os clientes são iniciados com ./cliente 127.0.0.1 <proxy port> (relembrar que o endereço "127.0.0.1" é o enderenço da máquina, então o endereço do server e da proxy é igual.



O server é iniciado com ./server <server port> <máximo numero de clientes>.



O proxy é iniciado com ./ircproxi <client port> <server port>.

Na função cliente é preciso criar 1 socket TCP, o qual conectar-se-á à proxy de maneira a estabelecer ligações cliente-proxy e um buffer que vai ler o que é introduzido. Também é criada uma diretória de downloads para aquele cliente (Exemplo: "downloads1").

```c
//CODIGO DAS FUNÇÕES

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

  system("clear");

  //O CLIENT SÓ PODE RECEBER 3 ARGUMENTOS
  if (argc != 3){
    printf("cliente <proxy address> <client port>\n");
    exit(0);
  }

  if (sodium_init() != 0) {
    return 1;
  }

  //CONFIGURAÇÃO DA LIGAÇÃO TCP
  strcpy(endServer, argv[1]);
  if ((hostPtr = gethostbyname(endServer)) == 0){
    perror("Nao consegui obter endereço");
  }
  bzero((void *) &addr, sizeof(addr));
  addr.sin_family = AF_INET;
  addr.sin_addr.s_addr = ((struct in_addr *)(hostPtr->h_addr))->s_addr;
  addr.sin_port = htons((short) atoi(argv[2]));

  //CRIAÇÃO DO SOCKET
  if((fd = socket(AF_INET,SOCK_STREAM,0)) == -1){
    perror("socket");
  }

  if( connect(fd,(struct sockaddr *)&addr,sizeof (addr)) < 0){
    perror("Connect");
  }

  write(fd, message, 1 + strlen(message));
  read(fd, buffer, sizeof(buffer));
  sscanf(buffer, "Hello client number %d", &n_cliente);
  bzero(buffer, BUF_SIZE);
```

```c
  struct stat st = {0};

  snprintf(diretoria, BUF_SIZE, "%s%d", "downloads", n_cliente);

  if (stat(diretoria, &st) == -1) {
    mkdir(diretoria , 0700);
  }
```

Na função server é criado um socket com o port desejado e fica à espera até receber um request do proxy para esse port. Para recebermos informações do client usamos o read() e usamos o write() para poder mandar mensagens do server para o client. Sempre que um Utilizador acede ao servidor é criado um novo processo para este, de maneira que vários utilizadores consigam comunicar com server sem haver perda de informação ou mistura desta.

```c
/* create a shared memory */
cria_mem();

signal(SIGINT, cleanup);

/* create listening TCP socket */
listenfd = socket(AF_INET, SOCK_STREAM, 0);

bzero(&servaddr, sizeof(servaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons((short) atoi(argv[1]));

// binding server addr structure to listenfd
bind(listenfd, (struct sockaddr*)&servaddr, sizeof(servaddr));
listen(listenfd, 5);

/* create UDP socket */
udpfd = socket(AF_INET, SOCK_DGRAM, 0);
// binding server addr structure to udp sockfd
bind(udpfd, (struct sockaddr*)&servaddr, sizeof(servaddr));

while (i < max_clientes) {
  //wait for new connection
  connfd = accept(listenfd, (struct sockaddr*)&cliaddr, &len);
  if (connfd > 0) {
    client->i++;
    if ((childpid = fork()) == 0) {
      tcp_processo(connfd);
    }
  }
}
printf("\n\t[Excedeu o número máximo de clientes!]\n");
printf("\n\t[Closing server...]\n");
exit(0);
```

```c
//CRIAÇÃO DO SOCKET
if((fd = socket(AF_INET,SOCK_STREAM,0)) == -1){
  perror("socket");
}

if( connect(fd,(struct sockaddr *)&addr,sizeof (addr)) < 0){
  perror("Connect");
}

write(fd, message, 1 + strlen(message));
read(fd, buffer, sizeof(buffer));
sscanf(buffer, "Hello client number %d", &n_cliente);
bzero(buffer, BUF_SIZE);

```

Na função proxy é criado um outro processo que espera por um comando (“SHOW” ou “SAVE”). Ambas as funções server e proxy têm uma função cleanup() que é ativada quando o C^ C é ativado.

```c
void controlo(int argc, char **argv){

  char comando[BUF_SIZE];


  while(1){

    scanf("%s", comando);

    if (strcmp(comando, "SHOW") == 0){
      printf("\tInformação sobre a ligação:\n\n");
      printf("\tEndereço de origem:\t127.0.0.1\n");
      printf("\tPorto de origem:\t%s\n", argv[1]);
      printf("\tEndereço de destino:\t127.0.0.1\n");
      printf("\tPorto de destino:\t%s\n\n", argv[2]);
    }
    else if(strcmp(comando, "SAVE") == 0){
      if (salvar->state == 0){
        salvar->state = 1;
      }
      else{
        salvar->state = 0;
      }
    }
    else{
      printf("\t[Wrong command]\n");
    }
  }
}
```

```c

void cleanup(int sig){

    shmdt(&salvar);
    shmctl(shm_save, IPC_RMID, NULL);
    printf("\nShutting down proxy...\n");
    exit(0);

}
```

```c
void cleanup(int sig){

    printf("\n\tCleaning memories...\n");
    //liberar memoria partilhada
    shmdt(&client);
    shmctl(shm_client, IPC_RMID, NULL);
    exit(0);
}

```

A encriptação nas transmissões dos ficheiros do server para a proxy é feita usando a biblioteca <sodium.h> e, quando digitado o argumento “ENC”, o ficheiro é encriptado no server e guardado no diretório “proxy_files” e como a proxy não tem a chave para a desencriptação, o ficheiro não poderá ser lido.

```c

if (strcmp(enc, "ENC") == 0){
  crypto_secretstream_xchacha20poly1305_keygen(key);
  if (encrypt("proxy_files/exemplo_encrypted.wav", "server_files/exemplo.wav", key) != 0) {
      printf("Nao deu!\n");
  }
  write(connfd, key, sizeof(key));
}
```