Aplicações Cliente/Servidor UDP

1

Servidor UDP Elementar/inicial

```
/*============================== SERVIDOR BASICO UDP ==========================
  ESTE SERVIDOR UDP DESTINA-SE A MOSTRAR OS CONTEUDOS DOS DATAGRAMAS RECEBIDOS, BEM COMO AS RESPECTIVAS ORIGENS.
  O PORTO DE ESCUTA ENCONTRA-SE DEFINIDO PELA CONSTANTE SERV_UDP_PORT.
  ASSUME-SE QUE AS MENSAGENS RECEBIDA SAO CADEIAS DE CARACTERES (OU SEJA, "STRINGS").
================================================================================*/

#include <unistd.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/param.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>

#define SERV_UDP_PORT 6000

#define BUFFERSIZE 4096
#define IP_SIZE 20

void Abort(char *msg);

/*_____________________________ MAIN _____________________________________
*/

int main( int argc , char *argv[] )
{
        int sockfd , nbytes;
        unsigned int length_addr, source_port;
        char source_ip[IP_SIZE];

        struct sockaddr_in serv_addr , cli_addr;
```

Estrutura que representa as coordenadas de uma aplicação na Internet: endereço IP do computador onde se encontra a correr + porto que a identifica de forma unívoca no computador hospedeiro

2

Servidor UDP Elementar/inicial

```
char buffer[BUFFERSIZE];

/*======================== PASSA PARA BACKGROUND ============================*/

switch( fork() ){
        case -1: Abort("Impossibilidade de passar para backgroud");
        case  0: break; /* O filho continua em background */
        default: exit(EXIT_SUCCESS); /* O pai termina */
}

/*============ CRIA O SOCKET PARA RECEPCAO DE DATAGRAMAS UDP ============*/

if((sockfd = socket( PF_INET , SOCK_DGRAM , 0))<0)
        Abort("Impossibilidade de abrir socket");

/*=============== ASSOCIA O SOCKET AO  ENDERECO DE ESCUTA ===============*/

/*DEFINE QUE PRETENDE RECEBER DATAGRAMAS VINDOS DE QUALQUER INTERFACE DE
  REDE, NO PORTO PRETENDIDO*/

bzero( (char*)&serv_addr , sizeof(serv_addr) );
serv_addr.sin_family = AF_INET; /*ADDRESS FAMILY = INTERNET*/
serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);  /*HOST TO NETWORK LONG*/
serv_addr.sin_port = htons(SERV_UDP_PORT);  /*HOST TO NETWORK SHORT*/


/*ASSOCIA O SOCKET AO PORTO PRETENDIDO*/

if(bind( sockfd , (struct sockaddr *)&serv_addr , sizeof(serv_addr)) <0)
        Abort("Impossibilidade de registar-se para escuta");
```


3

Servidor UDP Elementar/

Estrutura passada por referência para poder ser preenchida com as coordenadas do remetente do datagrama.

```
/*=============== PASSA A ATENDER CLIENTES INTERACTIVAMENTE ============*/

   while(1){
        fprintf(stderr,"<SER1>Esperando datagram...\n");

        length_addr = sizeof(cli_addr);
        nbytes = recvfrom(sockfd , buffer , sizeof(buffer) , 0 , (struct sockaddr *)&cli_addr ,
                                                                  &length_addr);

        if(nbytes<0)
                Abort("Erro na recepcao de datagrams");

        buffer[nbytes]='\0'; /*TERMINA A CADEIA DE CARACTERES RECEBIDOS COM '\0'*/

        source_port = ntohs( cli_addr.sin_port); /*NETWORK TO HOST SHORT*/
        strcpy( source_ip , (char *)inet_ntoa( cli_addr.sin_addr ) ); /*NETWORK TO ASCII*/

        printf( "\n<SER1>Mensagem recebida {%s} de {IP: %s; porto: %d}\n" , buffer, source_ip ,
                                                                        source_port );
   }
}

/*_____________________________________________ ABORT ____________________________________________
  MOSTRA A MENSAGEM DE ERRO ASSOCIADA AO ULTIMO ERRO NO SO E TERMINA COM "EXIT STATUS" A 1 (EXIT_FAILURE)
_____________________________________________________________________________________________*/

void Abort(char *msg)
{
        fprintf(stderr,"\a<SER1>Erro fatal: <%s>\n" , msg);
        perror("");
        exit(EXIT_FAILURE);

}
```

Buffer para onde é copiado o conteúdo do datagrama recebido.

4

Cliente UDP Elementar/inicial

```
/*============================ CLIENTE BASICO UDP ==============================
ESTE CLIENTE DESTINA-SE A ENVIAR MENSAGENS PASSADAS NA LINHA DE COMANDO, SOB
A FORMA DE UM ARGUMENTO, PARA UM SERVIDOR ESPECIFICO CUJA LOCACAO E' DADA
PELAS SEGUINTES CONSTANTES: SERV_HOST_ADDR (ENDERECO IP) E SERV_UDP_PORT (PORTO)

O PROTOCOLO USADO E' O UDP.
==============================================================================*/

#include <unistd.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/param.h>
#include <netinet/in.h>
#include <string.h>
#include <stdlib.h>


#define SERV_HOST_ADDR "127.0.0.1"
#define SERV_UDP_PORT  6000

#define BUFFERSIZE     4096

void Abort(char *msg);

/*_____________________________________ MAIN _____________________________________
*/

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

        int sockfd,msg_len;
        struct sockaddr_in serv_addr;
        char buffer[BUFFERSIZE];
```

Endereço IP especial (loopback / realimentação) que identifica o computador local.

5

Cliente UDP Elementar/inicial

```
/*======================== TESTA A SINTAXE ========================*/

 if(argc != 2){
        fprintf(stderr,"Sintaxe: %s frase_a_enviar\n",argv[0]);
        exit(EXIT_FAILURE);
 }

 /*=============== CRIA SOCKET PARA ENVIO/RECEPCAO DE DATAGRAMAS =============*/

 sockfd = socket( PF_INET , SOCK_DGRAM , 0 );
 if(sockfd < 0)
        Abort("Impossibilidade de criar socket");


 /*================== PREENCHE ENDERECO DO SERVIDOR ====================*/

 bzero( (char*)&serv_addr , sizeof(serv_addr) ); /*COLOCA A ZERO TODOS OS BYTES*/
 serv_addr.sin_family = AF_INET; /*ADDRESS FAMILY: INTERNET*/

 serv_addr.sin_addr.s_addr = inet_addr(SERV_HOST_ADDR); /*IP NO FORMATO "DOTTED DECIMAL" => 32 BITS*/
 serv_addr.sin_port = htons(SERV_UDP_PORT); /*HOST TO NETOWORK SHORT*/


 /*====================== ENVIA MENSAGEM AO SERVIDOR ==================*/

 msg_len = strlen(argv[1]);

 if(sendto( sockfd , argv[1] , msg_len , 0 , (struct sockaddr*)&serv_addr , sizeof(serv_addr) ) != msg_len)
        Abort("SO nao conseguiu aceitar o datagram");
```

Como não foi associado qualquer porto local ao socket, o sistema operativo atribui um de forma automática. Em alternativa, pode ser usada a rotina bind (ver servidor) e o porto zero (addr.sin_port = htons(0);).

6

Cliente UDP Elementar/inicial

```
        printf("<CLI1>Mensagem enviada ... a entrega nao e' confirmada.\n");

        /*========================= FECHA O SOCKET ============================*/

        close(sockfd);

        printf("\n");
        exit(EXIT_SUCCESS);
}


/*_____________________________ ABORT ____________________________________
  MOSTRA A MENSAGEM DE ERRO ASSOCIADA AO ULTIMO ERRO NO SISTEMA OPERATIVO
  E TERMINA A APLICACAO COM "EXIT STATUS" A 1 (CONSTANTE EXIT_FAILURE)
_____________________________________________________________________________*/

void Abort(char *msg)
{

        fprintf(stderr,"<CLI1>Erro fatal: <%s>\n",msg);
        perror("Erro do sistema");
        exit(EXIT_FAILURE);

}
```


Compilação e Execução

- Compilar

  gcc -o *servidor servidorUDP_v2.c*

  gcc -o *cliente clienteUDP.c*

- Obter ajuda

  man *inet_ntoa*

- Executar

  ./servidor

  ./cliente Hello!

  ./cliente "Hello servidor!"

  - Caso o servidor e os clientes não sejam lançados no mesmo computador, o valor da constante *SERV_HOST_ADDR* deve ser ajustado nos clientes e estes recompilados

Aulas práticas

- Encontram-se disponíveis no Moodle, desde o primeiro dia de aulas, todos os elementos de estudo necessários, incluindo:
  - O plano das aulas práticas;
  - A descrição dos comandos elementares Unix;
  - A forma de compilar programas em C em ambientes Unix;
  - Uma introdução aos sockets BSD;
  - Os exemplos abordados nas aulas práticas.
- Para estudar programação é necessário programar e fazer experiências
- Ao longo das aulas práticas, os códigos fonte do cliente e do servidor UDP, apresentados nos acetatos anteriores, vão evoluir de modo a incluir e explorar funcionalidades adicionais

9

Aulas práticas

- Estratégia a adoptar nas aulas práticas
  - O docente identifica a(s) funcionalidade(s) pretendida(s) e/ou o(s) problema(s) a resolver durante a aulas;
  - O docente descreve qualquer matéria nova que seja relevante no âmbito da aula;
  - Os alunos tentam atingir os objectivos propostos de uma forma autónoma;
  - No final da aula, o docente indica um trabalho que deverá ser realizado até à próxima aula.
- Evolução do Servidor UDP
  - Reenviar o datagrama recebido ao cliente
  - Reenviar, ao cliente, o tamanho da mensagem recebida
    - Em formato ascii (é reenviada uma *string*);
    - Em formato binário (é reenviado um inteiro).
  - Obter o porto de escuta pretendido através da linha de comando

10

Aulas práticas

- Evolução do cliente UDP
  - Aguardar pela recepção da resposta e visualizá-la
    - Versão 1: mensagem original devolvida;
    - Versão 2: tamanho da mensagem em formato ascii;
    - Versão 3: tamanho da mensagem em formato binário.
  - Determinar o porto automático atribuído pelo sistema operativo
  - Verificar se a origem do datagrama recebido coincide com o servidor
  - Verificar se a resposta é a esperada
  - Obter o endereço IP e o porto do servidor através da linha de comando
  - Aguardar pela resposta apenas durante um determinado tempo máximo (*timeout*)
    - Com *SIGALRM* / *alarm()* / *signal()*;
    - Com *setsockopt()* / *errno* == *EAGAIN*;
    - Com *select()*.