

Introdução as Redes de Comunicação - Aula 2

Prof Leandro Dihl

Sumário

- Resolução do exercício n.º 2 da ficha de exercícios em casa
- Resolução do exercício n.º 3 da ficha de exercícios
- Resolução do exercício n.º 4 da ficha de exercícios
- Modificação do cliente de modo a que este verifique o conteúdo da confirmação

Exercício 2 - Reenvio pelo servidor

Altere a aplicação anterior de modo a que o servidor reenvie as mensagens recebidas aos respectivos clientes. Estes devem aguardar pelas respostas e apresentá-las na saída standard.

Exercício 2 - Reenvio pelo servidor


```
int main(int argc, char* argv[])
{
    SOCKET sockfd;
    int iResult, nbytes;
    struct sockaddr_in serv_addr;
    char buffer[BUFFERSIZE];
    WSADATA wsaData;

    /*===== INICIA OS WINSOCKS =====*/

    iResult = WSASStartup(MAKEWORD(2, 2), &wsaData);
    if (iResult != 0) {
        printf("WSASStartup failed: %d\n", iResult);
        getchar();
        exit(1);
    }

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

    if ((sockfd = socket(PF_INET, SOCK_DGRAM, 0)) == INVALID_SOCKET) {
        char* msg = NULL;
        char mm[40] = "Impossibilidade de abrir socket";
        strcpy_s(mm, msg);
        Abort(msg);
    }
```



```
SOCKET sockfd;
int iResult, nbytes;
int length_addr, source_port;
char source_ip[IP_SIZE];
struct sockaddr_in serv_addr, cli_addr;
char buffer[BUFFERSIZE], resposta[MAX_RESPOSTA];
WSADATA wsaData;
```

Exercício 2 - Reenvio pelo servidor

```
/*Define que pretende receber datagramas vindos de qualquer interface de rede, no porto pretendido*/  
memset((char*)&serv_addr, 0, 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*/
```

- Importante:
 - As funções **htonl** e **htons** convertem os dados das constantes para bytes.
 - Ao ligar um soquete de escuta, **INADDR_ANY** permite conexões de entrada em qualquer endereço IPv4 local que pertença diretamente à máquina em que o soquete de escuta está sendo executado, o que inclui endereços de loopback. No entanto, você não pode se vincular a um endereço IP externo que esteja fora da máquina, como o IP público de um roteador de rede.

Exercício 2 - Reenvio pelo servidor

htonl function

The htonl function converts a u_long from host to TCP/IP network byte order.

Syntax

C++

```
u_long htonl(  
    u_long hostlong  
);
```

Exercício 2 - Reenvio pelo servidor

htons function

The htons function converts a u_short from host to TCP/IP network byte order.

Syntax

C++

```
u_short htons(  
    u_short hostshort  
);
```

Exercício 2 - Reenvio pelo servidor

While: obter o endereço do cliente e enviar a mensagem ao cliente

```
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 == SOCKET_ERROR)
        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);
    sprintf(resposta, "%d", strlen(buffer));

    nbytes = sendto(sockfd, resposta, strlen(resposta), 0, (struct sockaddr*) & cli_addr, sizeof(cli_addr));
    if (nbytes == SOCKET_ERROR) {
        printf("\n<SER1>Error ao reenviar a mensagem ao cliente\n");
    }
}
```


Exercício 2 - Reenvio pelo servidor

Alteração no Cliente:

```
/*===== ENVIA MENSAGEM AO SERVIDOR =====*/
msg_len = strlen(argv[1]);

if (sendto(sockfd, argv[1], msg_len, 0, (struct sockaddr*) & serv_addr, sizeof(serv_addr)) == SOCKET_ERROR)
    Abort("SO nao conseguiu aceitar o datagram");
printf("<CLI1>Mensagem enviada ... a entrega nao e' confirmada.\n");

/*===== AGUARDA RESPOSTA =====*/

nbytes = recvfrom(sockfd, buffer, sizeof(buffer), 0, NULL, NULL);
if (nbytes == SOCKET_ERROR)
    Abort("Erro enquanto aguarda pela resposta");
buffer[nbytes] = '\0';
printf("<CLI1>Mensagem recebida: %s\n", buffer);

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

closesocket(sockfd);
```

Exercício 3 - Apresentar o porto

3. Altere o cliente do exercício anterior de modo a que este apresente, na saída standard, o porto local automaticamente atribuído ao seu socket UDP. Esta operação deve ser executada depois do envio da mensagem ao servidor.

Exercício 3 - Apresentar o porto

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

msg_len = strlen(argv[1]);

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

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

tam = sizeof(cli_addr);
if (getsockname(sockfd, (struct sockaddr*) & cli_addr, &tam) != SOCKET_ERROR) {
    printf("<CLI1>Porto local automatico: %d\n", ntohs(cli_addr.sin_port));
}

/*===== AGUARDA RESPOSTA =====*/

nbytes = recvfrom(sockfd, buffer, sizeof(buffer), 0, NULL, NULL);

if (nbytes == SOCKET_ERROR)
    Abort("Erro enquanto aguarda pela resposta");

buffer[nbytes] = '\0';
printf("<CLI1>Mensagem recebida: %s\n", buffer);

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

Exercício 3 - getsockname function

The getsockname function retrieves the local name for a socket.

Syntax

C++

 Copy

```
int getsockname(  
    SOCKET    s,  
    sockaddr *name,  
    int       *namelen  
);
```

Exercício 4 - Mostrar as localizações

4. Altere o servidor desenvolvido no exercício 2 de modo a que este mostre, além dos conteúdos, as localizações de origem das mensagens recebidas.