Disciplina: Sistema Operacionais Embarcados **Código:** 120961 **Turma:** A
Professor: Diogo Caetano Garcia
Aluno/Matrícula: Fábio Barbosa Pinto – 11/0116356

Questionário: 09_Sockets_2

1. Crie dois programas em C para criar um cliente e um servidor. Faça com que o cliente envie os valores 1, 2, 3, 4, 5, 6, 7, 8, 9 e 10 para o servidor, com intervalos de 1 segundo entre cada envio. Depois de o cliente enviar o número 10, ele aguarda 1 segundo e termina a execução. O servidor escreve na tela cada valor recebido, e quando ele receber o valor 10, ele termina a execução.

Cliente:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <unistd.h>

int main (int argc, char* const argv[])
{
	int socket_id;
	struct sockaddr socket_struct;
	int numbers;


	socket_id = socket(PF_LOCAL, SOCK_STREAM,0);
	socket_struct.sa_family = AF_LOCAL;
	strcpy(socket_struct.sa_data,"/tmp/socket1");

	connect(socket_id,&socket_struct,sizeof(socket_struct));

	for(numbers=1;numbers<=10;numbers++){
		write(socket_id,&numbers,sizeof(int));
		printf("O cliente está enviando = %d\n",numbers);
		sleep(1);
	}

	close(socket_id);

	sleep(1);

	return 0;
}
```

Servidor:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <signal.h>

int socket_id;
void sigint_handler(int signum);
void print_numbers(int socket_id_client);
void end_server(void);

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

	struct sockaddr socket_struct;

	signal(SIGINT, sigint_handler);
	socket_id = socket(PF_LOCAL,SOCK_STREAM,0);

	socket_struct.sa_family = AF_LOCAL;
	strcpy(socket_struct.sa_data,"/tmp/socket1");
	bind(socket_id,&socket_struct,sizeof(socket_struct));
	listen(socket_id,2);

	while(1){

		struct sockaddr client;
		int socket_id_client;
		socklen_t cliente_len;

		socket_id_client = accept(socket_id,&client,&cliente_len);

		print_numbers(socket_id_client);

		close(socket_id_client);
	}

	return 0;
}

void sigint_handler(int signum){
	end_server();
```

```
}

void print_numbers(int socket_id_client){

        int numbers = 0;
        int state = 1;

        while(read(socket_id_client,&numbers,sizeof(int))){
                printf("O servidor recebeu = %d, state = %d\n",numbers,state);
        }

        end_server();
}

void end_server(void){
        unlink("/tmp/socket1");
        close(socket_id);
        exit(0);
}
```

2. Crie dois programas em C para criar um cliente e um servidor. Execute a seguinte conversa entre cliente e servidor:

CLIENTE: Pai, qual é a verdadeira essência da sabedoria?
SERVIDOR: Não façais nada violento, praticai somente aquilo que é justo e equilibrado.
CLIENTE: Mas até uma criança de três anos sabe disso!
SERVIDOR: Sim, mas é uma coisa difícil de ser praticada até mesmo por um velho como eu...

Neste exercício, o cliente deve escrever no terminal as mensagens enviadas e recebidas.

###Cliente

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <unistd.h>

int main (int argc, char* const argv[])
{
        char *socket_name;
        int socket_id;
        struct sockaddr name;
   int length;

        socket_name = argv[1];
```

```
	char mensagem_1[] = "Pai, qual é a verdadeira essência da sabedoria?";
char mensagem_2[] = "Mas até uma criança de três anos sabe disso!";

char* text;
	socket_id = socket(PF_LOCAL, SOCK_STREAM,0);
	name.sa_family = AF_LOCAL;
	strcpy(name.sa_data, socket_name);
connect(socket_id, &name, sizeof(name));

printf("\n%s\n", mensagem_1);
length = strlen(mensagem_1) + 1;
	write(socket_id, &length, sizeof(length));
	write(socket_id, mensagem_1, length);

	while(1)
	{
		read(socket_id, &length, sizeof (length));
		text = (char*) malloc (length);
		read(socket_id, text, length);

		if(!strcmp (text, "chave"))
		{
			read(socket_id, &length, sizeof (length));
			text = (char*) malloc (length);
			read(socket_id, text, length);
			fprintf(stderr, "\n\nCliente leu: %s.\n\n", text);
			break;
		}
	}

printf("\n%s\n", mensagem_2);
	length = strlen("chave") + 1;
	write(socket_id, &length, sizeof(length));
	write(socket_id, "chave", length);

length = strlen(mensagem_2) + 1;
	write(socket_id, &length, sizeof(length));
	write(socket_id, mensagem_2, length);

	while(1)
	{
		read(socket_id, &length, sizeof (length));
		text = (char*) malloc (length);
		read(socket_id, text, length);

		if(!strcmp (text, "chave"))
		{
```

```
                read(socket_id, &length, sizeof (length));
                text = (char*) malloc (length);
                read(socket_id, text, length);
                fprintf(stderr, "\n\nCliente leu: %s.\n\n", text);
                break;
            }
        }

        close(socket_id);
        fprintf(stderr, "Feito!\n");
        return 0;
}
```

###Server

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <signal.h>


char socket_name[50];
int socket_id;
void sigint_handler(int signum);
void print_client_message(int client_socket);
void end_server(void);

int main (int argc, char* const argv[])
{
        struct sockaddr socket_struct;
        strcpy(socket_name, argv[1]);
        signal(SIGINT, sigint_handler);
        socket_id = socket(PF_LOCAL, SOCK_STREAM, 0);
        socket_struct.sa_family = AF_LOCAL;
        strcpy(socket_struct.sa_data, socket_name);
        bind(socket_id, &socket_struct, sizeof(socket_struct));

        listen(socket_id, 10);

        while(1)
        {
            struct sockaddr cliente;
            int socket_id_cliente;

            socklen_t cliente_len;
```

```
                fprintf(stderr, "Aguardando a conexao de um cliente... ");

                socket_id_cliente = accept(socket_id, &cliente, &cliente_len);
                fprintf(stderr, "Feito!\n");

                fprintf(stderr, "Obtendo a informacao transmitida pelo cliente...");

                print_client_message(socket_id_cliente);
                fprintf(stderr, "Feito!\n");

                fprintf(stderr, "Fechando a conexao com o cliente... ");
                close(socket_id_cliente);
                fprintf(stderr, "Feito!\n");
        }
        return 0;
}

void sigint_handler(int signum)
{
        fprintf(stderr, "\nRecebido o sinal CTRL+C... vamos desligar o servidor!\n");
        end_server();
}


void print_client_message(int client_socket)
{
        char mensagem_1[] = "Não façais nada violento, praticai somente aquilo que é justo e
equilibrado.";
        char mensagem_2[] = "Sim, mas é uma coisa difícil de ser praticada até mesmo por
um velho como eu...";
        int length;
        char* text;

        read(client_socket, &length, sizeof (length));
        text = (char*) malloc (length);
        read(client_socket, text, length);
        fprintf(stderr, "\n\nServidor leu: %s.\n\n", text);

        length = strlen("c") + 1;
        write(client_socket, &length, sizeof(length));
        write(client_socket, "c", length);

        length = strlen(mensagem_1) + 1;
        write(client_socket, &length, sizeof(length));
        write(client_socket, mensagem_1, length);
```

```
        while(1)
        {
                read(client_socket, &length, sizeof (length));
                text = (char*) malloc (length);
                read(client_socket, text, length);

                if(!strcmp (text, "c"))
                {
                        read(client_socket, &length, sizeof (length));
                        text = (char*) malloc (length);
                        read(client_socket, text, length);
                        fprintf(stderr, "\n\nServidor leu: %s.\n\n", text);
                        break;
                }
        }

        length = strlen("c") + 1;
        write(client_socket, &length, sizeof(length));
        write(client_socket, "c", length);

        length = strlen(mensagem_2) + 1;
        write(client_socket, &length, sizeof(length));
        write(client_socket, mensagem_2, length);

    end_server();
}


void end_server(void)
{
        fprintf(stderr, "Apagando \"%s\" do sistema... ", socket_name);
        unlink(socket_name);
        fprintf(stderr, "Feito!\n");
        fprintf(stderr, "Fechando o socket local... ");
        close(socket_id);
        fprintf(stderr, "Feito!\n");
        exit(0);
}
```