Projeto 1 – MC833 – 1s 2021

Sistema Cliente-Servidor TCP

Relatório

Nome: Victor Rigatto

RA: 178068

Introdução

Este relatório tem por objetivo descrever o processo de implementação de um sistema cliente-servidor por protocolo TCP, desenvolvido em linguagem C, com funções de registro e armazenamento de informações de usuários, que serão melhor detalhadas nos próximos tópicos.

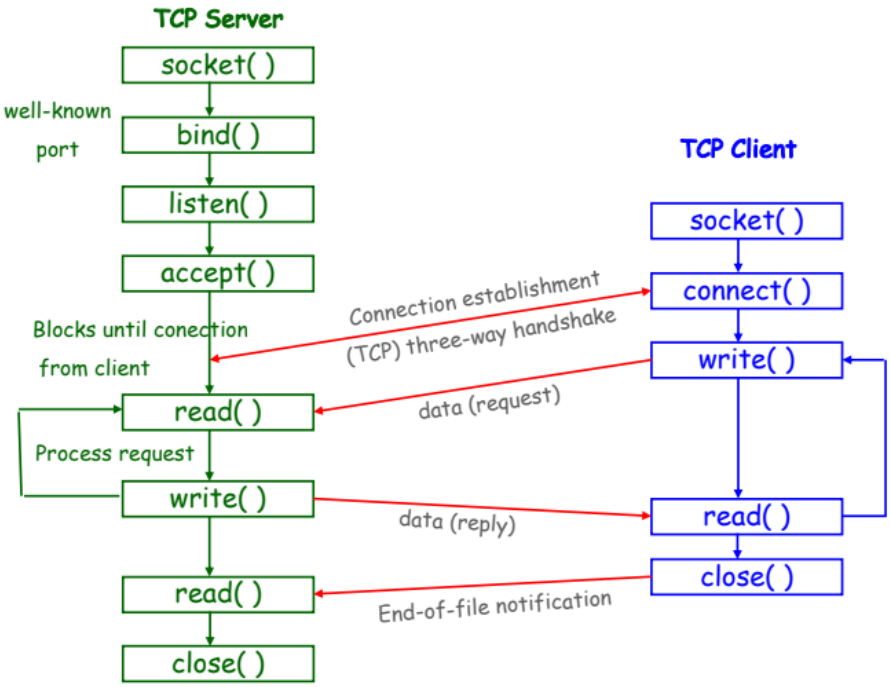
O desenvolvimento se deu através do entendimento do funcionamento da comunicação TCP e a implementação de socket nos aplicativos cliente e servidor, bem como as funções de leitura e escrita através dos sockets, e as respectivas funções de registro, exibição e manipulação de dados.

Como fonte de informações para a realização desse projeto, pode-se mencionar principalmente os slides disponibilizados e informações adicionais obtidas na internet.

Descrição geral

O sistema foi desenvolvido em ambiente Linux, utilizando-se a ferramenta WSL2 do Windows 10 em conjunto com o Visual Studio Code. O sistema consiste de 2 arquivos, denominados client.c e server.c, aplicativos cliente e servidor respectivamente.

O aplicativo cliente foi estruturado para inicialmente tentar estabelecer uma conexão TCP na porta 22000 com o servidor rodando localmente. Caso essa tentativa falhe, o programa informa um erro de conexão e encerra sua execução. A conexão com o servidor se da pela implementação padrão de socket TCP, exemplificada nos slides da disciplina, utilizando a biblioteca padrão socket.h. De maneira análoga, o aplicativo servidor foi estruturado para inicialmente criar um socket de escuta na porta 22000, para todos os endereços que quiserem se conectar. Um esquema da conexão TCP e sua implementação em C encontram-se abaixo.



Esquema de comunicação TCP entre cliente e servidor

```
struct in_addr {
    uint32_t s_addr; // that's a 32-bit int (4 bytes)
};

struct sockaddr_in {
    short int          sin_family;  // Address family, AF_INET
    unsigned short int sin_port;    // Port number
    struct in_addr     sin_addr;    // Internet address
    unsigned char      sin_zero[8]; // Same size as struct sockaddr
};
```

Estrutura padrão com informações do servidor para conexão através do socket

Abaixo encontra-se uma foto da implementação em C do processo de criação de socket e conexão com o servidor do aplicativo cliente. Os comentários descrevem as funções utilizadas.

```c
//Estrutura padrão com dados para a conexão
struct sockaddr_in servaddr;

//Limpa estrutura
bzero(&servaddr, sizeof(servaddr));

//Socket de domínio de internet IPv4, TCP stream e protocolo padrão (0)
//Cria socket para acesso à camada de transporte, retorna um descritor de socket
sockfd=socket(AF_INET, SOCK_STREAM, 0);

//Especifica o domínio de internet IPv4 e conexão na porta 22000
servaddr.sin_family=AF_INET;
servaddr.sin_port=htons(22000);
//Especifica o endereço IPv4 em formato binário na estrutura
inet_pton(AF_INET, "127.0.0.1", &(servaddr.sin_addr));

//Tenta conexão com o servidor especificado na estrutura utilizando o socket criado
if (connect(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr)) < 0){
    printf("Nao pode conectar.\n");
    return 0;
}
```

Implementação de socket do cliente em C

Após o sucesso da conexão com o servidor, o aplicativo cliente exibe um menu com as funções disponíveis. Para cada opção, a função com um número identificador é transmitida para o servidor e, em seguida, as informações pertinentes. Esse processo se repete até que o cliente escolha encerrar o programa.

```c
case 3:
    func = htonl(func);
    //Envia função para o servidor
    write(sockfd, &func, sizeof(func));
    printf("Insira um email: ");
    fgets(sendline, 100, stdin);
    //Envia string para o servidor
    write(sockfd, sendline, strlen(sendline)+1);
    //Função para receber strings uma a uma sincronizadamente
    receiveFromServer();
    break;
```

Envio da função que o cliente deseja para o servidor

Mensagens podem ser trocadas entre o cliente e servidor utilizando-se das funções write (envia dados pelo socket) e read (recebe dados pelo socket). Optou-se por realizar o envio e recebimento de strings de dados separadamente e de maneira sincronizada, uma a uma, para evitar a necessidade de serialização para envio e recebimento de estruturas inteiras de uma vez.

```
read(sockfd, recvline, 100);
printf("Email: %s",recvline);
read(sockfd, recvline, 100);
printf("Nome: %s",recvline);
read(sockfd, recvline, 100);
printf("Sobrenome: %s",recvline);
read(sockfd, recvline, 100);
printf("Cidade: %s",recvline);
read(sockfd, recvline, 100);
printf("Curso: %s",recvline);
read(sockfd, recvline, 100);
printf("Habilidades: %s",recvline);
read(sockfd, recvline, 100);
printf("Experiencias: %s",recvline);
```

Implementação do recebimento de um registro completo pelo cliente

A implementação do servidor é semelhante ao cliente. A primeira etapa é a criação de um socket para escuta de novas conexões de qualquer endereço de cliente. Após o recebimento de um pedido de conexão e o seu estabelecimento, o servidor aguarda pelas funções escolhidas pelo cliente. O recebimento de uma função inicia os próximos processos dentro do servidor.

```
//Estrutura padrão com dados para a conexão
struct sockaddr_in servaddr;

//Limpa estrutura
bzero(&servaddr, sizeof(servaddr));

//Socket de domínio de internet IPv4, TCP stream e protocolo padrão (0)
//Cria socket para acesso à camada de transporte, retorna um descritor de socket
listen_fd = socket(AF_INET, SOCK_STREAM, 0);

//Especifica o domínio de internet IPv4 e recebe qualquer endereço na porta 22000
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htons(INADDR_ANY);
servaddr.sin_port = htons(22000);

//Aloca um "nome" (endereço local) para o socket
//listen_fd é o descritor do socket criado
//Recebe endereço do cliente
bind(listen_fd, (struct sockaddr *) &servaddr, sizeof(servaddr));

//Anuncia a disponibilidade de aceitar conexões, informa o tamanho da fila
listen(listen_fd, 10);

//Aceita conexão e retorna um socket criado pelo SO para o cliente que conectou
comm_fd = accept(listen_fd, (struct sockaddr*) NULL, NULL);
```

Implementação de socket do servidor em C

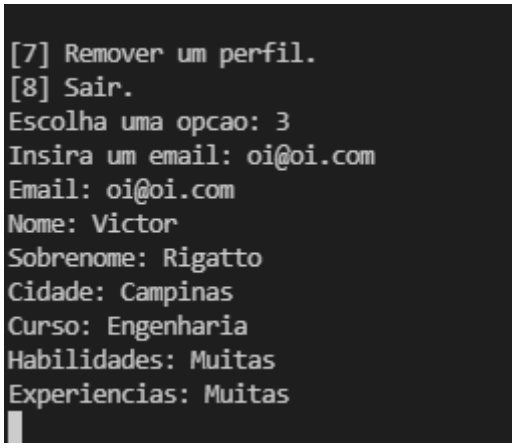
Assim como o cliente, o servidor também envia e recebe os dados de maneira sincronizada uma por uma, para evitar a necessidade de serialização para envio de estruturas inteiras.

```
//Função para enviar separadamente os dados requisitados pelo cliente
//Poderia utilizar serialização para enviar a estrutura inteira de uma vez
void sendToClient(struct data Data){
    write(comm_fd, Data.email, 100);
    write(comm_fd, Data.nome, 100);
    write(comm_fd, Data.sobrenome, 100);
    write(comm_fd, Data.cidade, 100);
    write(comm_fd, Data.curso, 100);
    write(comm_fd, Data.habilidades, 100);
    write(comm_fd, Data.experiencias, 100);
}
```

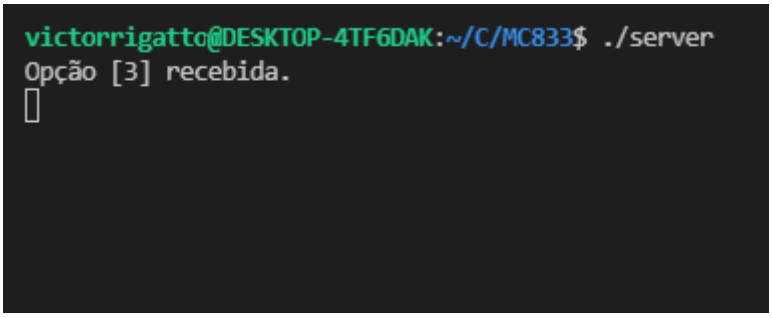
Implementação do envio de um registro completo pelo servidor

Casos de uso

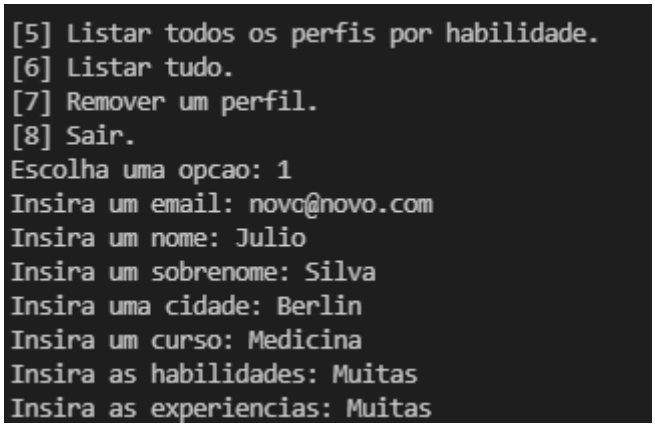
Abaixo encontram-se fotos de casos de uso do sistema cliente-servidor. Adotou-se a exibição do recebimento de funções no console do servidor, para fins de visualização.



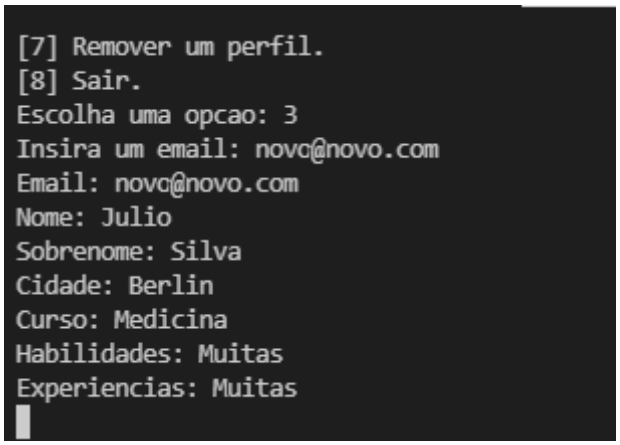
Terminal do cliente mostrando escolha da opção 3 do menu e exibição do resultado da busca pelo servidor



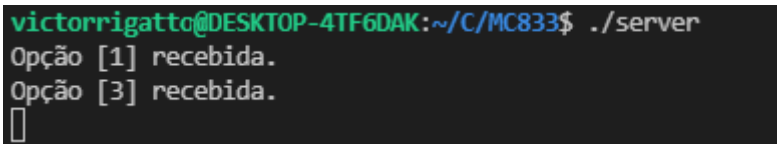
Terminal do servidor confirmando o recebimento da opção 3



Terminal do cliente mostrando a escolha da opção 1 do menu e inserção de dados para novo registro



Terminal do cliente mostrando escolha da opção 3 do menu e exibição do resultado da busca pelo servidor



Terminal do servidor confirmando o recebimento das opções 1 e 3

Armazenamento

Optou-se por utilizar a manipulação direta de arquivos binários em C, abrindo, lendo e escrevendo no arquivo definido como data.dat, no mesmo diretório do aplicativo servidor. O servidor recebe do cliente os dados no formato de strings de até 100 caracteres, uma de cada vez, e os insere em uma estrutura. Essa estrutura è então inteiramente escrita no arquivo, e posteriormente inteiramente lida em outras funções.

```c
//Estrutura para os dados do arquivo de registros
struct data{
    char email[20];
    char nome[20];
    char sobrenome[20];
    char cidade[20];
    char curso[20];
    char habilidades[40];
    char experiencias[40];
};
```

Estrutura no servidor para registro dos dados em arquivo binário

```c
//Função para escrever os dados no arquivo de registros
void writeFile(struct data Data){
    FILE *file;
    file = fopen("data.dat","a");
    if(file){
        fwrite (&Data, sizeof(struct data), 1, file);
    } else{
        printf("Nao pode abrir arquivo.\n");
    }
    fclose(file);
}
```

Função para escrita dos registros em arquivo binário

Conclusão

A realização desse projeto permitiu um entendimento prático importante da teoria que estudamos em MC832, especificamente da comunicação TCP entre cliente e servidor. A implementação teve desafios de nível técnico interessantes, como a manipulação dos dados para transmissão, que não è simples como simplesmente ler e escrever. Um passo importante nessa direção seria a possibilidade de transmitir estruturas inteiras, que pode ser feita através de serialização, para fins também de compatibilidade entre plataformas e arquiteturas distintas.

Outro ponto importante foi a elaboração do armazenamento. As operações com arquivos binários em C è um tanto quanto complexa, especialmente para funções de registro e alteração. Um sistema com banco de dados, como o SQL relacional, poderia ser mais interessante, porém acreditou-se não ser necessário.