Trabalho Prático 1

Comunicação Cliente - Servidor TCP

Guilherme Mendes de Oliveira
Departamento de Ciência da Computação

Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG – Brasil

guilhermemendes@ufmg.br

1. Introdução

Este trabalho tem como objetivo a implementação de um sistema de comunicação cliente -servidor utilizando protocolo TCP. O sistema será desenvolvido tendo como base cadastro, remoção e listagem de locais de vacinação, comandos da categoria SAÚDE, a serem utilizados pelas autoridades de saúde. Já o comando que informa o local de vacinação mais próximo, comando da categoria CIDADÃO, será utilizado para consultas do cliente.

Para tais comandos há restrições na quantidade de locais a serem cadastrados, no valor máximo de coordenada a ser cadastrada ($0 \leq X \leq 9999$) e ($0 \leq Y \leq 9999$) e limitação de *bytes* a ser enviados por mensagem, no máximo 500.

Para cada comando enviado pelo cliente, o servidor por sua vez deve retornar mensagens de confirmação de êxito dos comandos recebidos e caso haja qualquer comando que por sua vez não possa ser executado mensagens de retorno devem ser enviadas também como por exemplo quando tentarmos remover algum ponto que não exista na base de cadastro.

2. Implementação

2.1 Instruções de compilação e execução

O programa foi desenvolvido na linguagem C e compilado pelo GCC da Minimalist GNU for Windows (MinGW) no ambiente Linux distribuição Ubuntu versão 18.04.02.

2.2 Estrutura de Dados

Para a implementação foi utilizado um tipo abstrato de dados.

```
typedef struct ponto{
    int X;
    int Y;
} Ponto;
```

Figura 1 – Implementação do TAD Ponto.

O TAD representa os pontos de coordenadas recebidos do cliente, para sua instância foi utilizado um *array* de Ponto com o tamanho máximo permitido (50 pontos). A ordem dos pontos é sempre a ordem de entrada.

2.3 Código

Para a implementação do Cliente e do Servidor (client.c e server.c) foi realizada com base nos arquivos disponibilizados pelo professor Ítalo na sua aula de Introdução à Programação em Redes. Portanto o código final segue boa parte da estrutura criada nas aulas, as decisões de implementação serão discutidas nos tópicos seguintes;

2.3.1 Cliente

O Cliente é responsável por iniciar a comunicação, conectando na instância do servidor recebendo como parâmetros o endereço do localhost 127.0.0.1 e a porta de conexão definida no servidor 51511. Após ser inicializado através da função fgets() recebe na entrada os comandos definidos nos requisitos(add.rm,list,query e kill) juntamente com as coordenadas para aqueles comandos que necessários. Quando o comando “kill” é enviado ao servidor ele retorna ao cliente uma resposta para desconectar e o cliente fecha sua conexão após a resposta do servidor.

2.3.2 Servidor

O Servidor é instanciado definindo a versão do protocolo IPv4 ou IPv6 e a porta de conexão, após isto, recebe a mensagem do cliente através de um buffer e a partir disso identifica o comando enviado e caso necessário recebe as coordenadas enviadas junto a ele. Para cada comando há um condicional que trata o caso (add, rm, query, list e kill)

add: Verifica se há espaço para cadastro, caso não haja retorna a mensagem “limit exceeded”, se houver espaço verifica se o ponto já existe, se existir retorna uma mensagem de aviso “X Y already exists” e caso não exista, instancia um Ponto e aloca no array de Pontos após isso retorna “X Y added”

rm: Verifica se o ponto existe, caso exista remove e retorna uma mensagem “X Y removed”, caso não exista retorna “X Y does not exist”.

list: Caso não haja pontos cadastrados retorna a mensagem “none”, caso exista retorna as coordenadas separadas por espaço de todos os pontos cadastrados.

query: Caso não haja pontos cadastrados retorna a mensagem “none”, caso exista retorna o ponto cadastrado mais próximo às coordenadas fornecidas, para esse cálculo é utilizada a função distância, implementação de distância euclidiana, para todos os pontos e retorna o ponto que tem a menor distância dentre os cadastrados. Caso haja empate o ponto cadastrado primeiro dentre os empatados será retornado.

kill: Comando que encerra a conexão, este comando envia uma flag "disconnect" que faz o cliente fechar o *socket* de conexão com o servidor.

3. Desafios, dificuldades e imprevistos

Durante o desenvolvimento devido a uma abordagem errada na tratativa do buffer recebido pelo servidor foi necessário começar do zero uma vez que tentar adaptar e consertar se tornou uma tarefa complexa, fazendo com que eu perdesse além dos dias desenvolvendo a primeira versão abandonada mais um dia para planejar uma outra abordagem que tornasse a resolução do problema possível.

Além disso, durante a execução dos testes automáticos disponibilizados encontrei mais um problema no tratamento do buffer sendo necessário marcar uma monitoria para melhor entendimento do problema e de como seriam as mensagens enviadas para o servidor.

Mesmo com a ajuda do monitor não consegui resolver o envio do caracter nulo ‘\0’ não permitido na especificação do projeto, fazendo com que , mesmo que a nível de visualização o retorno do servidor esteja certo, no comparativo de *string* do *script* de teste haja diferença no *output.*

4.Conclusão

Embora o grau de corretude esteja baixo, de modo geral o entendimento e desenvolvimento de todos os princípios básicos da comunicação Cliente-Servidor seguindo o protocolo TCP foram alcançados. Todos os problemas que existem na execução monitorado pelos testes apontam que os problemas, que não são do tipo “diff output”, estão em algumas exceções não identificadas na implementação como múltiplos comandos em uma única linha,

5.Referências

<http://linguagemc.com.br/a-biblioteca-string-h> .Acesso em: 22 de Junho de 2021

Banahan, M;Brady,D;Doran,M. The C Book. Capítulo 5: Arrays and Pointer.

Banahan, M;Brady,D;Doran,M. The C Book. Capítulo 9:Libraries.