Trabalho Prático 1

Redes de Computadores - Universidade Federal de Minas Gerais

Mateus Brandão Damasceno Góes - 2020054706

Introdução

Este trabalho tem como objetivo desenvolver uma versão interativa do jogo clássico Campo Minado, possibilitando a comunicação entre um cliente e um servidor por meio do uso de sockets. Para atingir esse propósito, foram criados dois executáveis: um para o cliente e outro para o servidor, estabelecendo uma estrutura para a troca de informações entre as partes.

No contexto desse projeto, o cliente envia comandos ao servidor, que, por sua vez, responde fornecendo atualizações do estado do jogo. As seções subsequentes detalharão a implementação desse processo e seu funcionamento em maior profundidade.

Uma referência crucial utilizada na criação da etapa de comunicação foi o livro *TCP/IP Sockets in C*, disponibilizado na página da disciplina. Esse livro ofereceu uma abordagem abrangente, orientando desde a criação do servidor até a conexão do cliente.

Soluções adotadas para os sockets

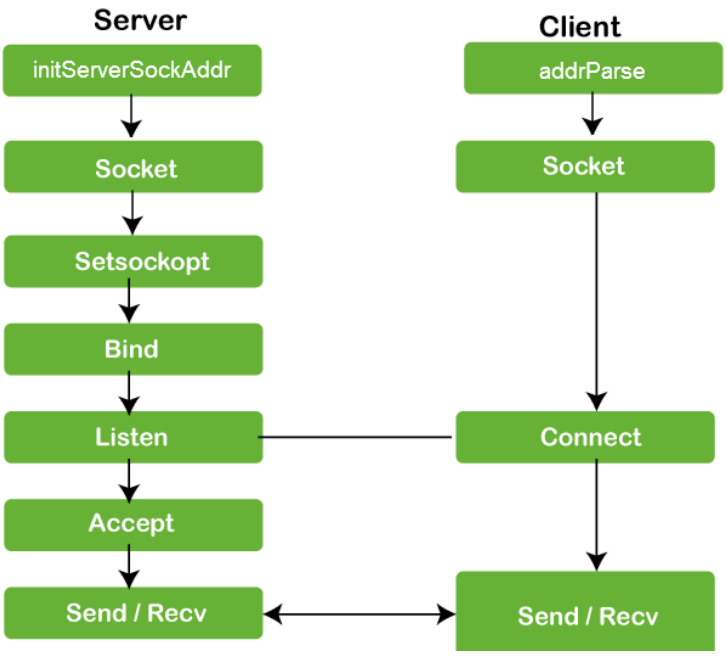
Para a criação dos programas de cliente e servidor foi usada a interface POSIX de sockets de redes, de maneira a criar a comunicação baseada no protocolo TCP para IPv4 e IPv6.

Além das funções especificadas pela biblioteca padrão como socket(), bind() e listen(). Foram criadas outras duas funções para ser feito o parsing dos comandos passados para o servidor e para o cliente para determinar a versão do protocolo IP usado e preencher a estrutura interna do socket (sockaddr_in).

A função ‘initServerSockaddr()’ realiza essa lógica para o servidor, convertendo os bytes do hospedeiro para bytes da rede com a função htons (de maneira a evitar problemas de endianness) e preenchendo a estrutura com família, endereço e porta de acordo com os valores passados.

Dessa forma, o servidor preenche o storage, cria o socket com as opções, assimila o endereço ao socket com bind() e passa a permitir conexões na porta com o listen(), caso haja conexão, aceita através da função accept(). O cliente faz o parsing do endereço passado com a função criada por mim (addrParse) e se conecta ao servidor com a função connect().

Abaixo segue uma imagem de um diagrama que foi adaptada do site *javatpoint.com* (adaptação para acrescentar as funções criadas) que demonstra a comunicação entre server e client através das suas chamadas de função:



Soluções adotadas para a implementação do jogo

Para a criação do jogo, foi necessário criar primeiro a estrutura que servirá para o envio e recebimento do servidor chamada de *action*. Nela contém o tipo da mensagem que será enviada ou recebida, as coordenadas (que são enviadas pelo cliente e tratadas pelo servidor) e o estado atual do board. Além disso foi criado uma estrutura chamada *gameSetup* que é armazenada pelo servidor que contém o estado inicial do jogo, ou seja o campo completo.

No lado do cliente, foi criada a função *computeInput()* que faz o parsing dos comandos enviados pelo cliente no *stdin*. Essa função determina o comando e as coordenadas e envia pro servidor o tipo da mensagem de acordo com o comando e as coordenadas como um *array* de inteiros. Essa função também é responsável por determinar e tratar quando há erros (como está estabelecido na página 6 da especificação do trabalho) e indicá-los para o cliente.

Se não há erro, o cliente envia a mensagem pro servidor que realiza suas ações na função *computeCommand()*, que recebe a estrutura recebida e os dados do campo inicial. Quando o comando é revelar alguma célula, por exemplo, o servidor checa se existe bomba no local, depois checa se aquele movimento resulta em uma vitória (usando funções auxiliares), se não, envia o campo atualizado e o tipo de mensagem como STATE (indicando que mudou o estado do campo).

Após o servidor enviar a mensagem pro cliente com o novo estado do campo, ou algum outro tipo de mensagem. O cliente chama a função *handleReceivedData()*, que de acordo com o tipo de mensagem recebida, pode imprimir o campo para o cliente, imprimir as mensagens de vitória ou derrota, ou encerrar a conexão.

Todas as opções de comando enviados pelo cliente e pelo servidor foram determinadas na especificação do trabalho. Foi estabelecido também que o cliente ao se conectar, começa indicando o comando *start* que carrega o campo oculto no campo de estado do cliente e o imprime. O cliente depois pode revelar as células

com o comando *reveal*, adicionar a bandeira com o comando *flag*, e removê-la com o comando *remove_flag*, resetar o campo com o comando *reset*, ou desconectar com o comando *exit*. O servidor envia para o cliente as mudanças de estado do campo, se o cliente ganhou ou perdeu.

Dessa maneira, a lógica do jogo ficou com funções e responsabilidades divididas entre o cliente e servidor. Onde o servidor é o responsável por guardar o campo inicial e realizar as ações e o cliente trata a resposta enviada pelo servidor e tem a responsabilidade de informar o usuário do programa das mudanças de estado, vitória e derrota.

Desafios e dificuldades do projeto

A dificuldade inicial que tive foi no estabelecimento da conexão entre o cliente e o servidor. Estava pesquisando com base na documentação da biblioteca de sockets o que estava me deixando mais confuso, porque a biblioteca tem casos de uso generalizados e várias vezes estava criando estruturas mais extensas e diferentes do escopo do trabalho.

Após esse período inicial, passei a usar o livro disponibilizado para entender a biblioteca de sockets, o que ajudou bastante. O livro explica muito bem as informações que devem ser passadas e tem vários exemplos de código. A maior dificuldade nessa etapa foi entender e filtrar as várias verificações de erro que o autor do livro implementa que não são necessárias na implementação desse trabalho, visto que, os erros que devem ser impressos foram todos descritos na especificação do projeto. O livro também foi de grande ajuda para conseguir fazer a implementação tanto do IPv4 quanto o IPv6 já que consegui entender os argumentos passados para preencher cada uma das estruturas.

Logo depois de estabelecer a conexão, o desafio que veio foi o de enviar a mensagem usando a struct *action*, como estabelecido. Estava conseguindo enviar apenas strings entre o cliente e o servidor e após pesquisar mais, consegui fazer o envio da estrutura.

Em seguida, a implementação da lógica do jogo se mostrou relativamente tranquila, uma vez que a lógica em si não era tão complexa e o campo já estava inicializado pelo servidor, o que evitou problemas significativos nessa fase.

Além disso, outra dificuldade foi o uso da linguagem C, que já faz tempo que tive contato, especialmente na parte de comparar strings do comando e tokenizá-las. A alocação de memória e cópia de áreas de variáveis também foram problemas complexos que foram facilitados usando bibliotecas padrão da linguagem como a <string.h> e <stdlib.h>.

Por fim, acredito que esse trabalho foi bem interessante e permitiu a aplicação dos conceitos vistos na disciplina de maneira prática. Apesar dos desafios, consegui implementar tudo o que foi estabelecido na especificação e consegui melhorar minhas habilidades com programas e ferramentas que usam comunicação cliente/servidor.