Universidade Federal de Minas Gerais - UFMG
Trabalho Prático 1 - Redes de Computadores
Professor: Luiz Filipe Menezes Vieira
Aluno: João Lucas Lage Gonçalves - 2020054552

1. Introudução e Objetivo

Neste trabalho, foi desenvolvida uma versão do jogo Campo Minado que permite a interação entre um cliente e um servidor, utilizando sockets na linguagem de programação C. O objetivo deste projeto foi criar um ambiente no qual uma máquina pudesse jogar Campo Minado remotamente, permitindo que os usuários interajam com o jogo de diferentes locais.

Funcionalidades do Cliente

O cliente é responsável por enviar comandos para o servidor. Esses comandos incluem a revelação de células no tabuleiro, marcação de possíveis bombas e interação direta com as células do jogo.

- Revelação de células: o cliente pode revelar uma célula no tabuleiro, expondo-a ao jogador.
- Marcação de células: o cliente pode marcar uma célula como uma possível bomba, para que ela seja destacada no tabuleiro.

Funcionalidades do Servidor

O servidor é o componente central do sistema, responsável por manter e atualizar o estado do jogo. Ele recebe os comandos do cliente, processa as ações solicitadas e atualiza dinamicamente o tabuleiro de jogo. O servidor também gerencia a lógica do Campo Minado, garantindo que as regras do jogo sejam seguidas e que as interações dos clientes sejam refletidas adequadamente no estado do jogo.

- Recepção de comandos: o servidor recebe os comandos do cliente através de sockets.
- Processamento de ações: o servidor processa as ações solicitadas pelo cliente, atualizando o estado do jogo de acordo.
- Atualização do tabuleiro: o servidor atualiza o tabuleiro de jogo dinamicamente, refletindo as ações dos clientes.
- Gerenciamento da lógica do jogo: o servidor gerencia a lógica do Campo Minado, garantindo que as regras do jogo sejam seguidas.

Arquitetura TCP e Compatibilidade IPv4/IPv6:

Para garantir uma comunicação eficaz entre o cliente e o servidor, ambos os componentes foram implementados usando o protocolo TCP (Transmission Control Protocol). Isso assegura uma transmissão de dados confiável e ordenada entre os dispositivos. Além disso, a implementação foi projetada para ser compatível tanto com o IPv4 quanto com o IPv6, proporcionando flexibilidade na escolha do endereço IP para a conexão. Isso significa que o sistema pode ser configurado para operar em redes que utilizam ambas as versões do protocolo IP.

2. Desenvolvimento e Soluções Implementadas

O desenvolvimento do projeto foi organizado em diferentes arquivos, cada um com responsabilidades específicas para garantir uma estrutura modular e organizada:

- server.c: para gerenciar a conexão do servidor, estabelecer a comunicação com os clientes e implementar as funções lógicas do Campo Minado no lado do servidor;
- client.c: responsável pela criação da conexão do cliente com o servidor e pelas funções lógicas relacionadas às ações do cliente;
- common.c: contém funções de conexão que são compartilhadas tanto pelo cliente quanto pelo servidor;
- game.c: engloba funções gerais relacionadas à lógica do jogo de Campo Minado. Estas funções não estão diretamente ligadas à comunicação cliente-servidor, mas são cruciais para o funcionamento do jogo.

Arquivo server.c

O arquivo server.c começa lendo um tabuleiro inicial de um arquivo e aceita conexões de clientes. As funções atualiza_tabuleiro() e verifica_vitoria() são cruciais: a primeira atualiza o tabuleiro conforme os comandos dos clientes (revelando células, marcando bombas) e a segunda verifica se o jogo foi ganho. No loop principal do main(), o servidor recebe comandos do cliente usando recv() e processa as ações do jogo. Após cada ação, envia respostas aos clientes através do send(), indicando o estado atualizado do tabuleiro e se o jogo foi ganho ou perdido, mantendo a comunicação eficaz entre servidor e clientes durante o jogo.

Arquivo client.c

O arquivo client.c se conecta ao servidor utilizando o endereço IP e a porta fornecidos na linha de comando. O código contém uma função verifica_erro() que valida os comandos do jogador, exibindo mensagens de erro se o jogador tentar realizar ações inválidas, como revelar células já reveladas ou inserir bandeiras em células já marcadas. O cliente envia os comandos ao servidor e recebe atualizações do estado do jogo. O programa imprime mensagens indicando se o jogador ganhou ou perdeu, bem como o tabuleiro atualizado após cada jogada. O loop principal do main() permite que o jogador insira comandos (como coordenadas de células para revelar ou marcar) a partir do teclado, envie-os para o servidor e receba as respostas correspondentes. O cliente continua em execução até que o jogador envie um comando de encerramento. Em caso de erros ou comandos inválidos, mensagens de erro são exibidas para orientar o jogador. Após cada jogada, o tabuleiro atualizado é impresso na tela com o estadao atual do jogo.

Arquivo game.c

O arquivo game.c contém funções essenciais para a lógica do jogo Campo Minado. A função inicia_tabuleiro() é usada para inicializar um tabuleiro, enquanto copia_tabuleiro() copia o conteúdo de um tabuleiro para outro. traduz_caracter() converte códigos numéricos em caracteres para exibição no tabuleiro do jogo. A função imprime_tabuleiro() imprime o tabuleiro atual na tela. A função traduz_acao() mapeia a ação do jogador para um código numérico de acordo com a tabela. le_mensagem() processa os comandos recebidos do cliente, traduzindo-os para a estrutura action. gera_resposta() é responsável por elaborar a mensagem que será transmitida ao cliente pelo servidor. Ela atualiza o tabuleiro da resposta com base no estado atual e nas ações do cliente, indicando se o jogo foi ganho, perdido, se deve ser reiniciado ou se houve um erro.

Além disso seguem algumas das soluções implementadas para o funcionamento do programa.
Protocolo TCP: o protocolo TCP foi escolhido para garantir uma comunicação eficaz entre o cliente e o servidor, assegurando uma transmissão de dados confiável e ordenada entre os dispositivos.
Compatibilidade IPv4/IPv6: a implementação foi projetada para ser compatível com o IPv4 e o IPv6, proporcionando flexibilidade na escolha do endereço IP para a conexão.
Comunicação estruturada: a comunicação entre o cliente e o servidor foi implementada de forma estruturada, utilizando a estrutura de dados **action** para representar os comandos e respostas.
Validação de comandos: o cliente possui uma função para validar os comandos do jogador, exibindo mensagens de erro se o jogador tentar realizar ações inválidas.

3. Desafios e Dificuldades

Durante o desenvolvimento do trabalho encontrei alguns desafios e dificuldades. A primeira dificuldade enfrentada foi a implementação da comunicação cliente-servidor usando sockets, algo completamente novo para mim. Os materiais de referência, foram de grande ajuda para superar esse desafio. Em segundo lugar ainda na lógica da novidade que foi a programação de uma comunicação servidor-cliente, entender claramente quem seria responsável por cada parte do sistema foi um desafio. Embora a proposta do trabalho fornecesse orientações claras, a aplicação prática desses conceitos inicialmente foi complexa. A chave para superar essa dificuldade foi revisitar a proposta várias vezes e analisar cada requisito com cuidado, anotando passo a passo o que cada parte faria ajudou no processo.
Por fim, a necessidade de implementar uma comunicação eficiente e complexa entre cliente e servidor, utilizando uma estrutura de dados (a struct action), foi outro desafio significativo. Enquanto os materiais de referência

forneceram uma compreensão básica, adaptar essa compreensão para uma comunicação estruturada foi um passo adicional. A solução para esse desafio envolveu a compreensão detalhada da estrutura de dados e a forma como ela seria transmitida entre as partes.