Trabalho Prático 1 - Servidor de Campo Minado

Entrega Individual

Data de Entrega: 09 de outubro de 2023, as 23:59

Introdução

O Campo Minado é um clássico jogo de um único jogador que foi originalmente desenvolvido em 1989 e se tornou amplamente popular com a inclusão no sistema operacional Windows. O principal objetivo do jogo é explorar um campo minado sem detonar nenhuma das minas ocultas.

Neste contexto, você está prestes a embarcar em um desafio empolgante: desenvolver uma versão de Campo Minado que permitirá a interação entre dois computadores remotos. Em outras palavras, você será responsável por criar tanto um servidor quanto um cliente para possibilitar que uma máquina jogue campo minado remotamente.

No servidor, residirá a tarefa de manter e atualizar o estado atual do jogo, enquanto o cliente terá a função de enviar as coordenadas do campo escolhido e receber as informações atualizadas do servidor sobre o estado do jogo.

Ao concluir este trabalho, você estará mais familiarizado com os princípios de comunicação em rede, o uso de sockets e terá ampliado suas habilidades de programação. Vamos então começar a criar um Campo Minado remoto através da implementação de um servidor e um cliente em rede.

Objetivo

O objetivo deste trabalho é criar uma versão do *Campo Minado* que permitirá a interação entre um cliente e um servidor usando sockets em linguagem C. O servidor será projetado para acomodar a conexão de um ***único*** cliente, proporcionando uma experiência de jogo remoto.

No início do jogo, o cliente envia um comando para iniciar a partida e recebe do servidor um tabuleiro do jogo com todas as posições ocultadas. O cliente pode então enviar coordenadas para revelar uma célula, marcar uma célula como uma possível bomba ou remover essa marcação. O servidor atualiza o tabuleiro de acordo com as ações do cliente e envia de volta o tabuleiro atualizado com todas as células reveladas ou marcadas até o momento, ou termina a partida de uma das formas descritas abaixo.

O jogo pode terminar de duas maneiras:

Vitória: O jogo é vencido quando todas as células que não contêm bombas são reveladas. Nesse caso, o servidor informa ao cliente que o jogo foi vencido, e o tabuleiro completo com todas as células reveladas é enviado. A partida é encerrada com sucesso.

Derrota: Se, a qualquer momento, o cliente revelar uma célula contendo uma bomba, o jogo será encerrado instantaneamente, e o servidor enviará o tabuleiro completo com todas as células reveladas, incluindo a localização das bombas. A partida é encerrada com uma derrota.

Protocolo

Tabuleiro

O tabuleiro do jogo será de 4x4 e conterá **três** bombas.

Mensagem a ser trafegada

O cliente e o servidor irão se comunicar por meio de uma mensagem que será trafegada usando sockets. A estrutura da mensagem deve ser a seguinte:

```
struct action {
  int type;
  int coordinates[2];
  int board[4][4];
};
```

Os tipos são:

Action	Type	Descrição
start	0	Enviada pelo cliente ao servidor no início do jogo
reveal	1	Enviada pelo cliente ao servidor para revelar célula
flag	2	Enviada pelo cliente ao servidor para adicionar flag em célula
state	3	Enviada pelo servidor ao cliente sempre que houver mudança no estado do board
remove_flag	4	Enviada pelo cliente ao servidor para remover flag em célula
reset	5	Enviada pelo cliente ao servidor para iniciar o jogo do início
win	6	Enviada pelo servidor ao cliente quando o jogo for ganho
exit	7	Enviada pelo cliente ao servidor quando deseja se desconectar
game_over	8	Enviada pelo cliente ao servidor quando o jogo for perdido

Observação: é muito importante que a mensagem siga exatamente essa estrutura para que haja compatibilidade entre os clientes e servidores de diferentes alunos.

Formato das células

Para trafegar os dados entre o cliente e o servidor faremos a representação do tabuleiro usando apenas inteiros. Porém, ao imprimir a resposta para o cliente os inteiros devem ser traduzidos para suas devidas representações em caracteres.

	Representação em caracteres	**Representação em inteiros**
Bomba	*	-1
Célula oculta	-	-2
Célula sem bomba na vizinhança	0	0
Célula com bomba na vizinhança	o próprio número	o próprio número
Célula com flag	>	-3

Comandos

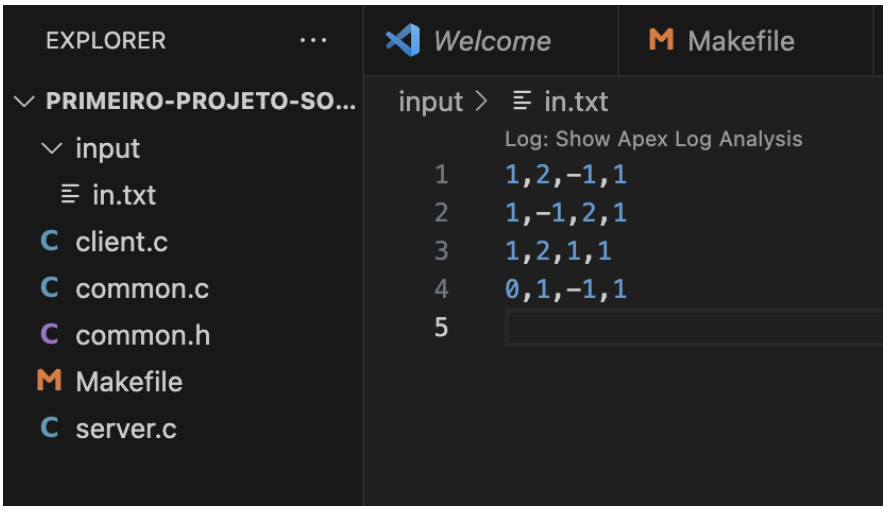
Início do jogo

No início do jogo o **servidor** lerá a resposta do tabuleiro de um arquivo. Esse arquivo deve conter todas as linhas e todos os caracteres separados por vírgula.

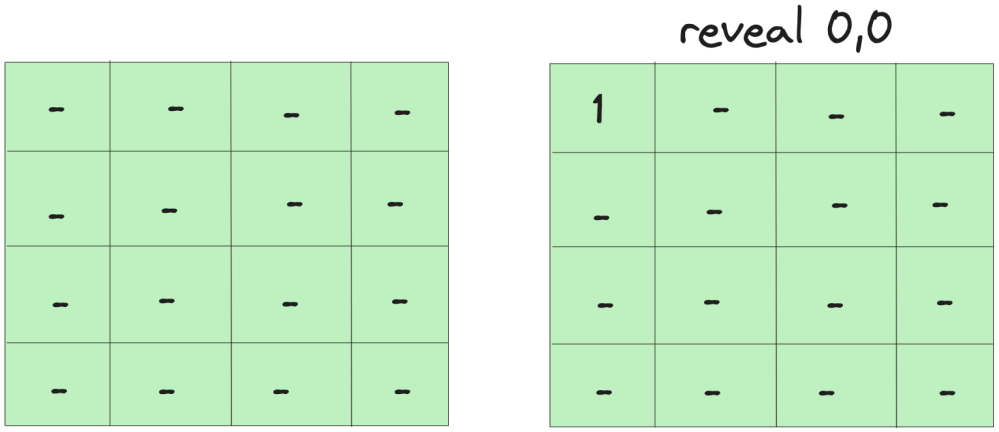
Ao inicializar o servidor o caminho do arquivo deve ser passado como parâmetro da seguinte maneira:

```
./server v6 51511 -i input/in.txt
```

Abaixo um exemplo da estrutura do projeto:



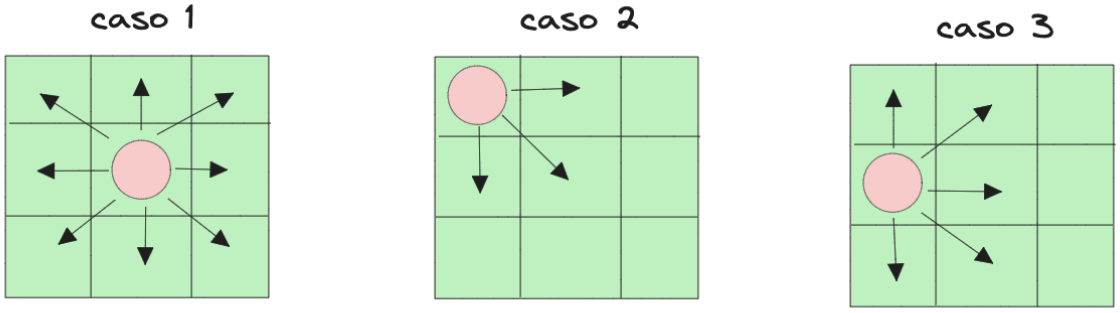
Revelar células
Para revelar uma célula você deve mandar as coordenadas **(linha e coluna, nessa ordem)** separadas por vírgula, e a célula a ser aberta será exatamente a célula das coordenadas enviadas.



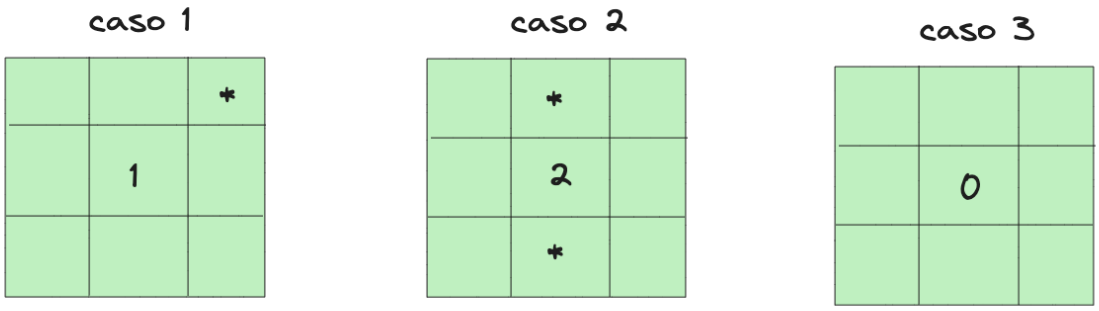
No exemplo, ao escolher revelar a célula de linha 0 e coluna 0 o tabuleiro mostrou o valor que estava naquela célula.

Bombas na vizinhança
Ao revelar uma célula, caso não haja uma bomba, um número irá aparecer, e esse número indica o número de bombas na vizinhança.

A vizinhança é representada da seguinte maneira:



A contagem das bombas funciona da seguinte maneira:

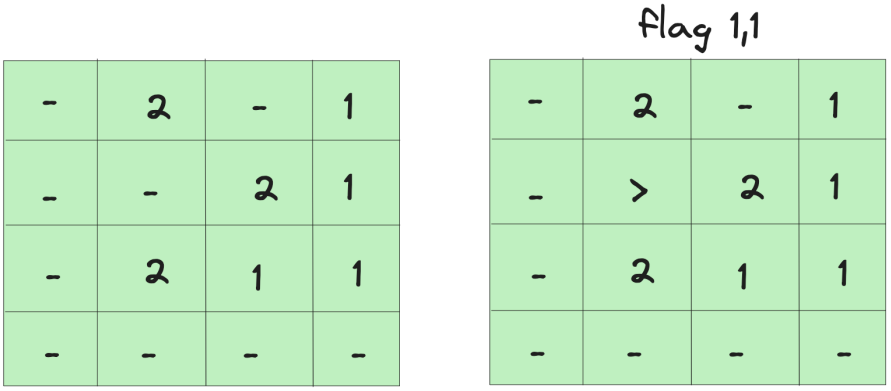


Para facilitar o entendimento, as bombas dos tabuleiros de exemplo das imagens acima estão todas reveladas, e a célula a ser revelada pelo jogador é a célula central, 1x1.

Caso 1: No primeiro caso, ao revelar a célula 1x1 o número exibido é 1, indicando que há apenas uma bomba na vizinhança.
Caso 2: No segundo caso, ao revelar a célula 1x1 o número exibido é 2, indicando que existem duas bombas na vizinhança.
Caso 3: No terceiro caso, ao revelar a célula 1x1 o número exibido é 0, indicando que não existe nenhuma bomba na vizinhança.

Marcar células
O usuário pode optar por marcar uma célula quando ele tiver a certeza de que ela possui uma bomba. Nesse caso, ele enviará o comando **flag** seguido da célula a ser revelada.

No exemplo abaixo fica claro que na célula 1,1 existe uma bomba, portanto o usuário poderá adicionar uma flag à essa célula.



Nesse caso a marcação "**>**" será colocada na célula que possui bomba.

Remover uma marcação

O usuário também poderá remover uma marcação quando achar necessário, usando o comando **remove_flag**.

Sair do jogo
O cliente poderá sair do jogo enviando o comando **exit.** Nesse caso, a conexão do cliente com o servidor se encerrará e o cliente terminará a execução.

Reset
O cliente poderá resetar o jogo, e nesse caso o jogo começará do início.

Erro
O usuário pode vir a digitar um comando inválido, ou fazer alguma das ações acima em uma célula cuja ação já foi feita, portanto esses erros devem ser tratados do lado do cliente.

Abaixo uma tabela com o mapeamento de **TODOS OS POSSÍVEIS** erros. **Os demais casos não mapeados na tabela não necessitam de uma mensagem de erro.**

revelar célula fora do range do tabuleiro	error: invalid cell
comando inexistente	error: command not found
revela uma célula já revelada	error: cell already revealed
flag em uma célula já marcada	error: cell already has a flag
flag em uma célula revelada	error: cannot insert flag in revealed cell

Exemplos de jogo

Jogo com derrota

Terminal do cliente	Terminal do Servidor
	./server v4 51511 -i input/jogo.txt 1 2 * 1 1 * 2 1 1 2 1 1 0 1 * 1
> ./client 127.0.0.1 51511	client connected
start - - - - - - - - - - - - - - - -	
reveal 0,0 1 - - - - - - - - - - - - - - -	
reveal 1,1 GAME OVER! 1 2 * 1 1 * 2 1 1 2 1 1 0 1 * 1	

Jogo com comando inválido

Terminal do cliente	Terminal do Servidor
	./server v4 51511 -i input/jogo.txt 1 2 * 1 1 * 2 1 1 2 1 1 0 1 * 1
> ./client 127.0.0.1 51511	client connected
start - - - - - - - - - - - - - - - -	
reveal 0,0 1 - - - - - - - - - - - - - - -	
reveal 0,0 error: cell already revealed	
reveal 1,0 1 2 - - - - - - - - - - - - - -	

Repare que nesse caso o usuário digita um comando errado, o erro aparece, porém ele consegue continuar a jogar.

Exemplo com flag e remove_flag

Terminal do cliente	Terminal do Servidor

	`./server v4 51511 -i input/jogo.txt`
	`1 2 * 1`
	`1 * 2 1`
	`1 2 1 1`
	`0 1 * 1`

`> ./client 127.0.0.1 51511`	`client connected`

`start`
`- - - -`
`- - - -`
`- - - -`
`- - - -`	

`reveal 0,0`
`1 - - -`
`- - - -`
`- - - -`
`- - - -`	

`reveal 0,1`
`1 2 - -`
`- - - -`
`- - - -`
`- - - -`	

`reveal 1,0`
`1 2 - -`
`1 - - -`
`- - - -`
`- - - -`	

`flag 1,1`
`1 2 - -`
`1 > - -`
`- - - -`
`- - - -`	

`remove_flag 1,1`
`1 2 - -`
`1 - - -`
`- - - -`
`- - - -`	

Exemplo com cliente conectando e desconectando

Terminal do cliente	Terminal do Servidor
	./server v4 51511 -i input/jogo.txt 1 2 * 1 1 * 2 1 1 2 1 1 0 1 * 1
> ./client 127.0.0.1 51511	client connected
start - - - - - - - - - - - - - - - -	
reveal 0,0 1 - - - - - - - - - - - - - - -	
exit	client disconnected
reveal 1,0 1 2 - - 1 - - - - - - - - - - -	
flag 1,1 1 2 - - 1 > - - - - - - - - - -	
remove_flag 1,1 1 2 - - 1 - - - - - - - - - - -	

Jogo com exit

Terminal do cliente	Terminal do Servidor
	./server v4 51511 -i input/jogo.txt 1 2 * 1 1 * 2 1 1 2 1 1 0 1 * 1
> ./client 127.0.0.1 51511	client connected
start - - - - - - - - - - - - - - - -	
exit	client disconnected

Jogo com reset

Terminal do cliente	Terminal do Servidor
	./server v4 51511 -i input/jogo.txt 1 2 * 1 1 * 2 1 1 2 1 1 0 1 * 1
> ./client 127.0.0.1 51511	client connected
start - - - - - - - - - - - - - - - -	
reveal 0,0 1 - - - - - - - - - - - - - - -	
reset - - - - - - - - - - - - - - - -	starting new game

Jogo com vitória

Terminal do cliente	Terminal do Servidor
	./server v4 51511 -i input/jogo.txt 1 2 * 1 1 * 2 1 1 2 1 1 0 1 * 1
> ./client 127.0.0.1 51511	client connected
start - - - - - - - - - - - - - - - -	
reveal 0,0 1 - - - - - - - - - - - - - - -	
exit	client disconnected
> ./client 127.0.0.1 51511	client connected
start - - - - - - - - - - - - - - - -	

Tabela de comandos

Comando	Cliente	Servidor
conexão do cliente	não imprime nada no cliente	imprime client connected no servidor.
start	imprime o tabuleiro com todas as posições ocultadas	Inicializa o estado do jogo. Não imprime nada do lado do servidor
reveal x,y	imprime o tabuleiro com a célula revelada	Altera o estado da célula para revelado, e envia novo estado para o cliente ***Não imprime nada na tela***
flag x,y	imprime o tabuleiro com a célula com flag	Altera o estado da célula, e envia novo estado para o cliente ***Não imprime nada na tela***
remove_flag x,y	imprime o tabuleiro com a célula sem flag	Altera o estado da célula, e envia novo estado para o cliente ***Não imprime nada na tela***
reset	imprime o tabuleiro com todas as posições ocultadas	Reseta o estado do jogo Imprime: "starting new game"
exit	desconecta do servidor	Reseta o estado do jogo Imprime: client disconnected

Detalhes de implementação

- Note que o servidor irá guardar tanto a matriz contendo a resposta do tabuleiro quanto a matriz contendo o estado atual do jogo. Sempre que uma ação é feita a matriz de estado é atualizada
- Tanto o cliente quanto o servidor devem imprimir o tabuleiro com a representação em caracteres.
- **As posições na matriz devem ser imprimidas com duas tabs de separação: "\t\t"**

Desenvolvimento

O projeto requer a implementação de dois componentes essenciais: um servidor e um cliente, ambos baseados no protocolo **TCP**. É crucial garantir que ambos os componentes sejam compatíveis **tanto com o IPv4 quanto com o IPv6**, proporcionando flexibilidade na escolha do endereço IP.

O cliente tem a responsabilidade de receber mensagens diretamente do teclado e exibi-las na tela. O servidor será responsável por fazer todo o processamento do jogo.

É importante observar que, para este trabalho, **não é necessário que o servidor suporte múltiplos clientes simultaneamente**. Ele será projetado para lidar com conexões de um cliente por vez.

Para configurar o servidor corretamente, este deve ser capaz de receber, seguindo rigorosamente essa **ordem, o tipo de endereço desejado (v4 para IPv4 ou v6 para IPv6) e um número de porta espe**cificado na linha de comando. Recomenda-se a utilização da porta 51511 para manter a padronização do projeto.

Da mesma forma, o cliente precisa receber, também rigorosamente nessa ordem, o endereço IP do servidor e o número de porta para estabelecer a conexão com sucesso. Certifique-se de que ambos os componentes sejam configurados de acordo com essas diretrizes para uma integração eficaz e a comunicação adequada entre servidor e cliente.

A seguir, um exemplo de execução dos programas em dois terminais distintos:

IPv4:

```
no terminal 1: ./server v4 51511 -i input/jogo1.txt
no terminal 2: ./client 127.0.0.1 51511
```

IPv6:

```
no terminal 1: ./server v6 51511 -i input/jogo1.txt
no terminal 2: ./client ::1 51511
```

Materiais para Consulta:

● Capítulo 2 e 3 do livro sobre programação com sockets disponibilizado no Moodle.

- Playlist de programação com sockets do professor Ítalo Cunha

Avaliação

O trabalho deve ser realizado individualmente e **deve ser implementado com a linguagem de programação C (não sendo permitida a utilização de C++)** utilizando somente a biblioteca padrão (interface POSIX de sockets de redes). Deve ser possível executar seu programa no sistema operacional **Linux** e **não deve utilizar bibliotecas Windows, como o winsock**. Seu programa deve interoperar com qualquer outro programa implementando o mesmo protocolo **(você pode testar com as implementações dos seus colegas)**. Procure escrever seu código de maneira clara, com comentários pontuais e bem identados. Isto facilita a correção dos monitores e tem impacto positivo na avaliação.

Correção Semi-automática
Seu servidor será corrigido de forma semi-automática por uma bateria de testes. Cada teste verifica uma funcionalidade específica do servidor. O seu servidor será testado por um cliente implementado pelo professor com funcionalidades adicionais para realização dos testes. Os testes avaliam a aderência do seu servidor ao protocolo de comunicação inteiramente através dos dados trocados através da rede (a saída do seu servidor na tela, e.g., para depuração, não impacta os resultados dos testes).

Para a correção os seguintes testes serão realizados **(com IPv4 e IPv6)**:

- Iniciar o jogo **10%**
- Revelar célula adequadamente **10%**
- Terminar o jogo quando uma bomba for revelada **10%**
- Adicionar uma flag **5%**
- Remover uma flag **5%**
- Fechar conexão com o servidor corretamente **10%**
- Resetar o jogo **10%**
- Cliente envia exit para o servidor e fecha a execução: **10%**
- Jogo terminar com vitória quando todas as células que não sejam bombas forem reveladas **10%**
- Documentação **20%**

Informações importantes

Cada aluno deve entregar documentação em PDF de até 4 páginas (duas folhas), sem capa, utilizando fonte Arial tamanho 12, e figuras de tamanho adequado ao

tamanho da fonte. A documentação deve discutir desafios, dificuldades e imprevistos do projeto, bem como as soluções adotadas para os problemas.

Atenção: Será dado grande importância a parte da *discussão dos desafios encontrados*, pois as dificuldades ao longo do projeto sempre existem.

A documentação corresponde a 20% dos pontos do trabalho, mas só será considerada para as funcionalidades implementadas corretamente.

Será utilizado um sistema para detecção de código repetido, portanto não é admitido cola de trabalhos.

Será adotada média harmônica entre as notas da documentação e da execução, o que implica que a nota final será 0 se uma das partes não for apresentada.

Cada aluno deve entregar, além da documentação, o código fonte em C e um Makefile para compilação do programa.

Instruções para submissão e compatibilidade com o sistema de correção semi-automática:

- O Makefile deve compilar o cliente em um binário chamado "**client**" e o servidor em um binário chamado "**server**" dentro de uma pasta chamada "**bin**" na raiz do projeto.
- Seu programa deve ser compilado ao se executar apenas o comando "make", ou seja, sem a necessidade de parâmetros adicionais.
- A entrega deve ser feita no formato ZIP, com o nome seguindo o seguinte padrão: TP1_MATRICULA.zip

Desconto de Nota por Atraso

Os trabalhos poderão ser entregues até a meia-noite do dia especificado para a entrega. O horário de entrega deve respeitar o relógio do sistema Moodle, ou seja, a partir de 00:01 do dia seguinte à entrega no relógio do Moodle, os trabalhos já estarão sujeitos a penalidades.
A fórmula para desconto por atraso na entrega do trabalho prático é:

$$desconto = 2^{d} - 1$$

d: atraso em dias úteis.

Obs.: Note que após 3 dias, o trabalho não pode ser mais entregue.