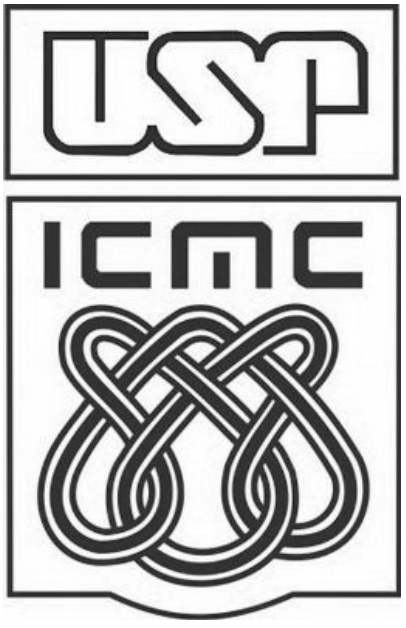
Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação

RELATÓRIO - REDES DE COMPUTADORES

Fernando Moura Leite Vendratameto 9875973
Guilherme Correa Fernandes 9278174
Rodrigo Geurgas Zavarizz 9791080
William Quelho Ferreira 9293421



São Carlos - SP
2018

Descrição do programa

O programa é a implementação de um jogo de batalha naval (), utilizando comunicação entre o servidor e os clientes (jogadores) através de sockets. Os dois jogadores podem se conectar e enviar comandos ao servidor, que por sua vez realiza a lógica do jogo.

Para mais informações a respeito das regras do jogo de batalha naval: https://pt.wikipedia.org/wiki/Batalha_naval_(jogo).

O programa foi implementado em linguagem C em ambiente Linux. Para utilização da conexão por sockets foi utilizada a biblioteca *sys/socket* e para construir a interface dos jogadores foi utilizada a biblioteca *ncurses.* Para utilizar essa biblioteca é necessário realizar sua instalação antes, pacote *libncurses-dev* no caso do sistema operacional *Debian* e derivados.

Instruções de Compilação e Execução

Para compilação e execução deve se utilizar o *Makefile* enviado junto com o trabalho, digitando os seguintes comandos:

Excluir os arquivos já compilados: *make clean*
Compilar: *make all*
Executar o servidor: *make run-server*
Executar o cliente: *make run-client*

Os comandos acima são apenas atalhos. Em especial, *make run-client* tenta se conectar a um servidor local na porta default. Para configuração mais pontual do servidor e do cliente, é necessário executar o arquivo *build/out* com as seguintes flags:

Executar o servidor: *./build/out -p <porta> -s*
Executar o cliente: *./build/out -p <porta> -c <ip_servidor>*

Os argumentos devem aparecer sem colchetes angulares.
Se nem *-s* nem *-c* forem especificados, assume-se *-s* e o programa executa o servidor.

Instruções de Uso

Para iniciar o jogo é necessário que o servidor e dois clientes conectados ao servidor estejam rodando devendo executar primeiro o servidor e em seguida os dois clientes para que o jogo possa começar.

Primeiro cada um dos jogadores movimentam um seletor pelo tabuleiro do jogo utilizando as setas do teclado e selecionam cada um dos quadrados que querem colocar uma das extremidades de um navio com a tecla *enter*, após selecionar esse quadrado os jogadores devem selecionar a direção em que o navio será colocado utilizando as setas. Após todos os navios serem posicionados por ambos os jogadores o jogo passa para a próxima fase.

Na segunda fase os jogadores se alternam em turnos para tentar atingir os navios adversários. Para selecionar o quadrado e disparar utilizam-se novamente as setas do teclado e a tecla *enter*. Quando um navio é atingido em todos os seus quadrados, ele é afundado. Ganha o jogador que afundar todos os navios de seu adversário primeiro.

Explicação da arquitetura

Para organização do código, foram criados os arquivos fonte e de cabeçalho *client* e *server*. As estruturas e suas funções abstraem as funcionalidades de socket a fim de facilitar o desenvolvimento do resto do código.

A estrutura *client* tenta se conectar a um ip e uma porta onde se assume haver um servidor aguardando conexões. A estrutura *server* reserva um socket e uma porta e, ao chamar da função *server_awaitClients* bloqueia a execução aguardando a conexão do número especificado de clientes.

Após tal conexão, o servidor pode aguardar uma mensagem de um cliente específico ou mandar uma mensagem a um cliente específico. O cliente se comunica diretamente apenas com o servidor, mas possui as mesmas operações.

Funcionamento de sockets

Sockets são extremidades de um protocolo de comunicação. Para estabelecer uma conexão no modelo de cliente-servidor, o servidor usa *bind* para reservar uma porta, e escuta por solicitações de clientes para comunicação através de *listen*.

Após a chamada de *listen*, clientes podem começar a requisitar conexão através de *connect* se souberem o IP do servidor. A chamada de *listen*, porém, não estabelece a conexão: apenas abre as portas às requisições. Para aceitar uma requisição de um cliente, o servidor usa a função *accept*, que retorna um descritor de arquivo que pode ser usado para escrita e leitura de/para o cliente.