GABRIEL MATTHEUS BEZERRA ALVES DE CARVALHO – 9779429

JUN -

VICTOR SOUZA CEZARIO – 9790919

**1º TRABALHO PRÁTICO**

**SSC250 – COMPUTAÇÃO GRÁFICA**

**SÃO CARLOS – SP**

**ABRIL 2018**

1 INTRODUÇÃO

Este relatório descreve a implementação e funcionamento do primeiro trabalho prático, no qual deveria ser desenvolvida uma aplicação que fizesse uso de sockets para a comunicação entre hosts. Foi escolhido pelo grupo o jogo “Batalha Naval”, um jogo em turnos que, da maneira como foi implementado, pode ser jogado pelo próprio terminal por dois jogadores em computadores diferentes conectados à mesma rede.

**2 IMPLEMENTAÇÃO**

A comunicação da aplicação foi desenvolvida utilizando-se sockets e o modelo cliente-servidor. O servidor é preparado primeiro e aguarda a solicitação dos dois clientes (jogadores), e, após a conexão ser confirmada, o jogo se inicia.

**2.1 SERVIDOR**

A implementação do servidor está contida nos arquivos main\_serv.c e server\_client.c, e as partes mais significativas são descritas a seguir:

* int socket(AF\_INET, SOCK\_STREAM, 0): chamada de sistema que cria um socket e retorna seu descritor. Recebe os parâmetros AF\_INET, que descreve o tipo de endereços com os quais o socket se comunicará (neste caso do tipo IPv4), SOCK\_STREAM, que descreve o tipo do socket e pode ser SOCK\_STREAM ou SOCK\_DGRAM, nos quais os caracteres são lidos de forma contínua e sequencial ou em blocos, respectivamente. O último corresponde ao protocolo, e ao fornecermos 0 o sistema operacional ficará responsável por escolher o que melhor se encaixe, que no caso do tipo SOCK\_STREAM é o TCP;
* (\*serv\_addr).sin\_family = AF\_INET: atribuindo o tipo de endereços do servidor, que deve ser igual ao do socket;
* (\*serv\_addr).sin\_addr.s\_addr = INADDR\_ANY: atribuindo o endereço IP do host, no caso do servidor ele deve ser o endereço dá máquina na qual o servidor está rodando, representado pela constante INADDR\_ANY;
* (\*serv\_addr).sin\_port = htons(port): atribuindo o número da porta que será utilizada. Para isso precisamos antes converter o valor para um ‘network byte order’ que é o que a função htons() faz;
* bind((\*sockfd), (struct sockaddr \*) serv\_addr, sizeof(\*serv\_addr)): chamada de sistema que enlaça o socket referenciada pela variável sockfd ao endereço contido na struct serv\_addr. O terceiro parâmetro é o tamanho da struct serv\_addr;
* listen(sockfd, 5): chamada de sistema que faz com que o processo escute por conexões na socket apontada pelo descritor sockfd. O segundo parâmetro se refere ao número de conexões que podem estar aguardando enquanto o processo está lidando com outra;
* accept(sockfd, (struct sockaddr \*) cli\_addr, len): chamada de sistema que bloqueia o processo até que um cliente conecte-se ao servidor. Ela retorna um novo descritor que deve ser usado para a comunicação com o cliente. O segundo parâmetro é o endereço para a variável que armazenará as informações do cliente, e o terceiro é o tamanho dessa variável;
* write((\*cli\_sockfd), msg, strlen(msg)): método que envia uma mensagem usando o socket passado como primeiro parâmetro. A mensagem em si é enviada como segundo parâmetro e o terceiro é o tamanho da mensagem;

**2.2 CLIENTE**

A implementação do cliente está contida nos arquivos main\_client.c e server\_client.c. As partes mais significativas que não foram discutidas na seção do servidor estão descritas a seguir:

* gethostbyname(hostname): método que retorna informações sobre o servidor recebendo como parâmetro seu nome;
* connect((\*sockfd),(struct sockaddr \*) serv\_addr, sizeof((\*serv\_addr))): chamada de sistema que tenta estabelecer uma conexão com um servidor. Recebe de parâmetros o socket a ser utilizado, uma struct com informações sobre o endereço do servidor e o tamanho da struct;

**3 INSTRUÇÕES**

**3.1 COMPILAÇÃO E EXECUÇÃO**

**3.2 INSTRUÇÕES DO JOGO**

O jogo Batalha Naval dispõe de dois campos, um para cada jogador, contendo navios distribuídos em cem posições, listadas em uma matriz, com linhas que variam de 1 a 10, e colunas que variam de A a J. A intersecção de uma linha e uma coluna corresponde a uma posição, e o objetivo do jogo é afundar todos os navios do jogador adversário. Assim, a cada turno um jogador insere uma posição do campo adversário e, caso esta posição corresponda a um navio, tal parte da frota adversária é afundada. Nesse sentido, vence o jogador que conseguir afundar todas as partes de todos os navios adversários. Cada jogador possui um porta-avião, dois navios-tanque, três contratorpedeiros e quatro submarinos, sendo que cada um ocupa, respectivamente, cinco, quatro, três e duas posições.

Após conectado, cada cliente deve indicar, em um primeiro momento, as posições de cada navio disponível em sua frota. Serão requisitadas, então, as posições de cada navio e a sua orientação, devendo ser inserido no formato “[número da linha] [letra da coluna] [orientação] [Enter]”. A linha e a coluna indicam o início do navio, e a orientação indica em qual sentido as outras posições devem ser inseridas. Podendo ser à direita [D], esquerda [E], cima [C] ou baixo [B]. Por exemplo, caso queira inserir um submarino que ocupe as posições 1D, 1E, deve-se inserir “1 D D” ou, mais ainda, “1 E E”.

Após cada jogador dispor toda a sua frota em seu campo, a batalha começa. Alternando os turnos, um jogador que estiver em sua vez de atacar recebe uma mensagem requisitando a inserção da posição de ataque. Ela deve ser dada no formato “[número da linha] [letra da coluna] [Enter]”, e deve corresponder a uma posição ainda não atacada. O campo adversário é visto da seguinte maneira:

* “O”: corresponde a uma posição ainda não atacada e que pode ser escolhida pelo jogador.
* “X”: corresponde a uma posição atacada e que pertencia a um navio adversário.
* “Y”: corresponde a uma posição atacada e que pertence ao mar.
* “N”: visível apenas ao observar o próprio campo, corresponde a um pedaço de um navio.

Os jogadores seguem alternando as jogadas até que um dos jogadores tenha afundado todos os navios adversários, tornando-se então o vencedor.