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**

**3 INSTRUÇÕES**

**3.1 COMPILAÇÃO E EXECUÇÃO**

**3.2 INSTRUÇÕES DO JOGO**