

MC833 - Trabalho Final

Felipe Santana Dias - 215775

Para a implementação do jogo Jo-Ken-Po, foi implementado um servidor que se comunica com todos os clientes via protocolo TCP. Para suportar as múltiplas conexões que acontecem foi utilizada a função `select()`.

O servidor é responsável por interligar os jogadores em todas as partidas - sejam em uma partida contra o servidor, seja entre jogadores.

Assim que um jogador se conecta ao servidor, este atribui as informações do cliente a uma matriz que será utilizada durante a partida entre jogadores. Cada cliente recebe um ID que corresponde ao índice nessa matriz, que foi arbitrariamente limitada à 20 clientes.

Na matriz de jogadores não foram tratado os casos em que os clientes se desconectam do servidor e deixam de estar disponível para as partidas, a remoção do próprio cliente ao selecionar o oponente (o que pode gerar um jogo contra si mesmo) e o caso de algum jogador estar ocupado em outra partida.

A matriz possui a forma abaixo, sendo a primeira linha reservada para armazenar o tamanho da tabela e a última coluna é utilizada para armazenar a pontuação de cada jogador.

ID	IP	Porta	Pontuação
3	-	-	-
1	134.190.21.89	3000	0
2	59.33.20.129	3000	0
3	146.31.46.231	3000	0
...			

Tabela 1: Representação da matriz 'jogadores' implementada no arquivo `servidor.c`.

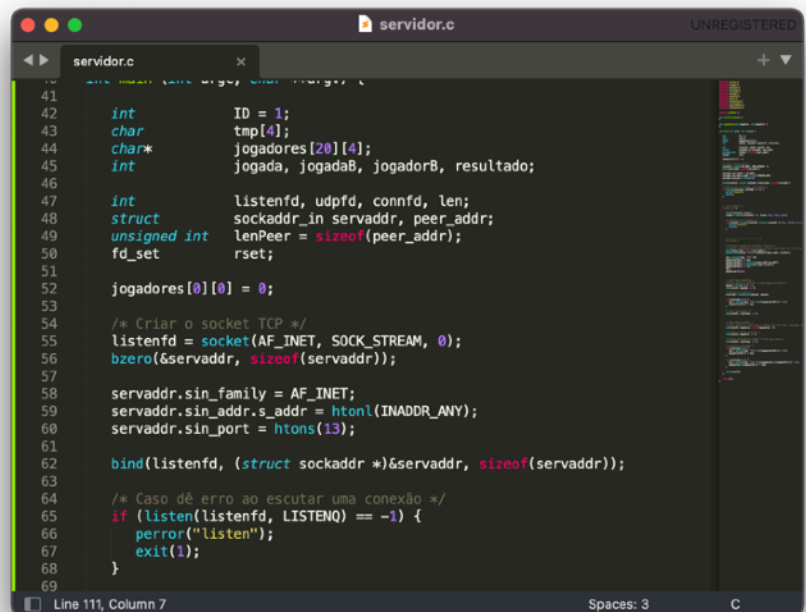


imagem 1: Inicialização do servidor TCP no arquivo `servidor.c`.

```
88
89 /* Permitir conexão com múltiplos clientes */
90 /* Receber informações do cliente e colocar numa lista: ID, IP, Porta, Pontuação = 0 */
91 bzero(&peer_addr, sizeof(peer_addr));
92 getpeername(connfd, (struct sockaddr*)&peer_addr, &lenPeer);
93
94 tmp = sprintf(tmp, "%d", ID);
95 jogadores[ID][0] = tmp;
96 jogadores[ID][1] = inet_ntoa(peer_addr.sin_addr);
97 jogadores[ID][2] = ntohs(peer_addr.sin_port);
98 jogadores[ID][3] = "0";
99 ID++;
100 jogadores[0][0]++;
101
```

imagem 2: Inserção de novo cliente conectado ao servidor na matriz 'jogadores'.

Para o jogo acontecer, foram criadas interfaces através da função `printf()` que podem ser vistas nas funções `escolherJogador()`, `escolherJogada()` e `printResultado()`. Por questão de simplicidade na hora de compilar, algumas funções são duplicadas tanto no cliente quanto no servidor.

```
15
16 int escolherJogador(char (*jogadores)[3], int row) {
17
18     int i, j, oponente;
19
20     for(i = 0; i < row; i++) {
21
22         printf("%d ", i);
23
24         for(j = 0; j < 3; j++) {
25             printf("%s ", jogadores[i][j]);
26         }
27         printf("\n");
28     }
29
30     printf("Qual o número do seu oponente?");
31     scanf("%d", &opponente);
32
33     return oponente;
34 }
35
36
```

imagem 3: Funções que desempenham papel de interface para o jogo. De cima para baixo:

`escolherJogador`: Mostra todos os dados contidos na matriz 'jogadores'

`escolherJogada`: Recebe movimento que o jogador deseja realizar.

`printResultado`: Recebe o resultado e transmite para o jogador.

```
15
16 int escolherJogada() {
17     int jogada;
18
19     printf("Qual será sua jogada?\n1 - Pedra\n2 - Tesoura\n3 - Papel");
20     scanf("%d", &jogada);
21
22     return jogada;
23 }
24
```

```
61 void printResultado(int resultado) {
62
63     if (resultado == 1) {
64         printf("Parabéns! Você derrotou o seu oponente.\nNos vemos no próximo jogo :)\n");
65     }
66
67     if (resultado == 2) {
68         printf("Vixi! Seu oponente venceu essa partida.\nBoa sorte no próximo jogo :(\n");
69     }
70
71     if (resultado == 0) {
72         printf("Belo embate! Hora que acabou em empate.\nMais sorte na próxima vez :)\n");
73     }
74
75     else {
76         printf("Alguma coisa deu errado com o seu resultado\nO final desse jogo será pra sempre um mistério\n");
77     }
78 }
79
```

Definir o vencedor



```
46 int quemGanhou(int jogador1, int jogador2) {
47
48     // Caso dê empate
49     if (jogador1 - jogador2 == 0) {
50         return 0;
51     }
52
53     // Caso jogador 1 vença
54     if (jogador1 - jogador2 == -1 || jogador1 == jogador2 == 2) {
55         return 1;
56     }
57
58     // Caso jogador 2 vença
59     if (jogador1 - jogador2 == 1 || jogador1 == jogador2 == -2) {
60         return 2;
61     }
62
63     else {
64         return 0;
65     }
66 }
67
```

imagem 4: Função 'quemGanhou' utilizada para definir o vencedor da partida.

Para definir o vencedor, é realizada a comparação entre os números correspondente à jogada selecionada pelos participantes. As seguintes opção são disponibilizadas em ordem decrescente de força:

- 1 - Pedra
- 2 - Tesoura
- 3 - Papel

É então realizada a subtração entre os valores para encontrar o vencedor - caso a operação apresente valor igual a zero, é um empate; caso apresente valor -1 ou 2, o jogador 1 é o vencedor; caso o valor seja 1 ou -2, o jogador 2 é o vencedor, como pode ser visto na tabela abaixo. A atribuição do ponto ao vencedor pela função quemGanhou() segue a seguinte regra: retorna 1 no caso em que o jogador 1 venceu; retorna 2 caso o jogador 2 ou o servidor tenha vencido; retorna 0 caso ocorra um empate.

Jogador 1	Jogador 2	Resultado	Jogador 1	Jogador 2	Resultado
1	2	-1	1	3	-2
2	3	-1	2	1	1
3	1	2	3	2	1

Tabela 2: Demonstração da lógica utilizada para definir os vencedores de cada partida.

Jogo contra o servidor

Ao ser questionado, caso o jogador escolha o modo de jogo número 1, se inicia uma partida contra o servidor.

Inicialmente o cliente escolhe sua jogada e a envia ao servidor, que gera uma jogada aleatória e compara os resultados. Esse resultado é então enviado de volta ao cliente e, caso ele vença a partida, sua pontuação é atualizada pelo servidor.

Vale ressaltar de que para exibir o resultado correto a função `quemGanhou()` possui como parâmetros a jogada do cliente e do servidor respectivamente.



```
104 /* JOGO CONTRA SERVIDOR */
105 /* Envia opções do servidor e recebe jogada escolhida */
106 jogada = (rand() % 3) - 1;
107 recv(connfd, jogadaB, 4, 0);
108
109 /* Analisa quem ganhou */
110 resultado = quemGanhou(jogadaB, jogada);
111
112 if (resultado == 1) {
113     tmp = sprintf(tmp, "%d", (int{jogadores[ID][3]} + 1));
114     jogadores[ID][3] = tmp;
115 }
116
117 /* Envia quem ganhou */
118 send(connfd, resultado, 4, 0);
119
```



```
200 /* JOGO CONTRA SERVIDOR */
201 if (command == 1) {
202
203     /* Recebe opções do servidor e envia jogada escolhida */
204     jogada = escolherJogada();
205     send(sockfd, jogada, 4, 0);
206
207     /* Recebe resultado do jogo */
208     recv(sockfd, resultado, 4, 0);
209     printResultado(resultado);
210
211     /* volta começo loop */
212     continue;
213 }
214
```

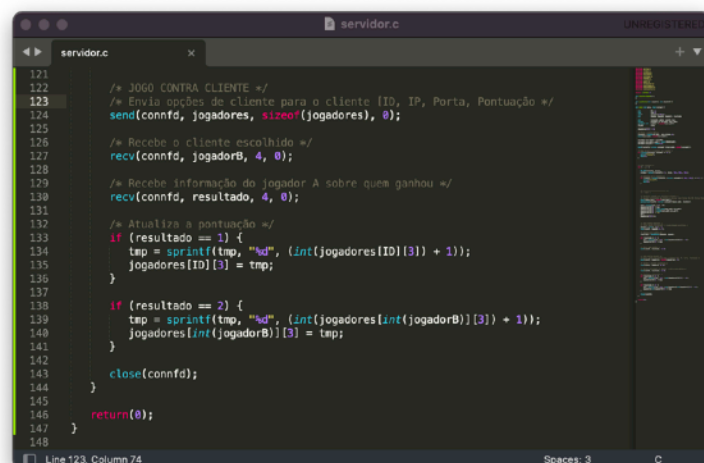
imagem 5: Arquivos servidor.c e cliente.c que realizam o jogo entre cliente e servidor através da conexão TCP.

Jogo contra o cliente

Quando o jogador escolhe a opção de jogo número 2, será iniciado uma partida contra outro jogador que também está conectado ao servidor. A conexão entre os dois jogadores será estabelecida através de uma comunicação UDP, portanto o arquivo cliente.c deve atuar tanto como receptor quanto remetente.

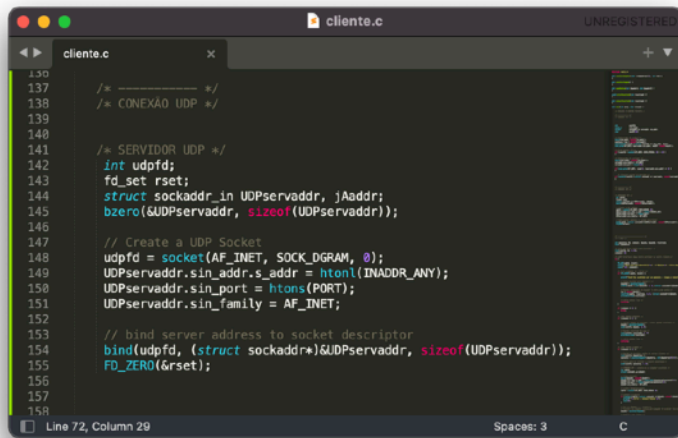
Quando o jogador A escolhe esse tipo de partida, ele sinaliza ao servidor esse interesse, que envia a lista de jogadores disponíveis. Como já dito anteriormente, não foram tratados os casos nessa lista em que o próprio jogador está disponível, jogadores que já se desconectaram do servidor e/ou estão ocupados em outras partidas.

Após escolhido o oponente, o servidor recebe o ID do jogador B e, após a partida, recebe o resultado. Com essas informações, ele atualiza a pontuação dos jogadores.



```
121 /* JOGO CONTRA CLIENTE */
122 /* Envia opções de cliente para o cliente (ID, IP, Porta, Pontuação) */
123 send(connfd, jogadores, sizeof(jogadores), 0);
124
125 /* Recebe o cliente escolhido */
126 recv(connfd, jogadorB, 4, 0);
127
128 /* Recebe informação do jogador A sobre quem ganhou */
129 recv(connfd, resultado, 4, 0);
130
131 /* Atualiza a pontuação */
132 if (resultado == 1) {
133     tmp = sprintf(tmp, "%d", (int{jogadores[ID][3]} + 1));
134     jogadores[ID][3] = tmp;
135 }
136
137 if (resultado == 2) {
138     tmp = sprintf(tmp, "%d", (int{jogadores[int{jogadorB}][3]} + 1));
139     jogadores[int{jogadorB}][3] = tmp;
140 }
141
142 close(connfd);
143
144 return(0);
145
146 }
147
148
```

imagem 6: Atuação do servidor.c no modo de jogo entre jogadores.



```
130
137 /* ----- */
138 /* CONEXÃO UDP */
139
140
141 /* SERVIDOR UDP */
142 int udpfd;
143 fd_set rset;
144 struct sockaddr_in UDPServaddr, jAaddr;
145 bzero(&UDPServaddr, sizeof(UDPServaddr));
146
147 // Create a UDP Socket
148 udpfd = socket(AF_INET, SOCK_DGRAM, 0);
149 UDPServaddr.sin_addr.s_addr = htonl(INADDR_ANY);
150 UDPServaddr.sin_port = htons(PORT);
151 UDPServaddr.sin_family = AF_INET;
152
153 // bind server address to socket descriptor
154 bind(udpfd, (struct sockaddr*)&UDPServaddr, sizeof(UDPServaddr));
155 FD_ZERO(&rset);
156
157
158
```

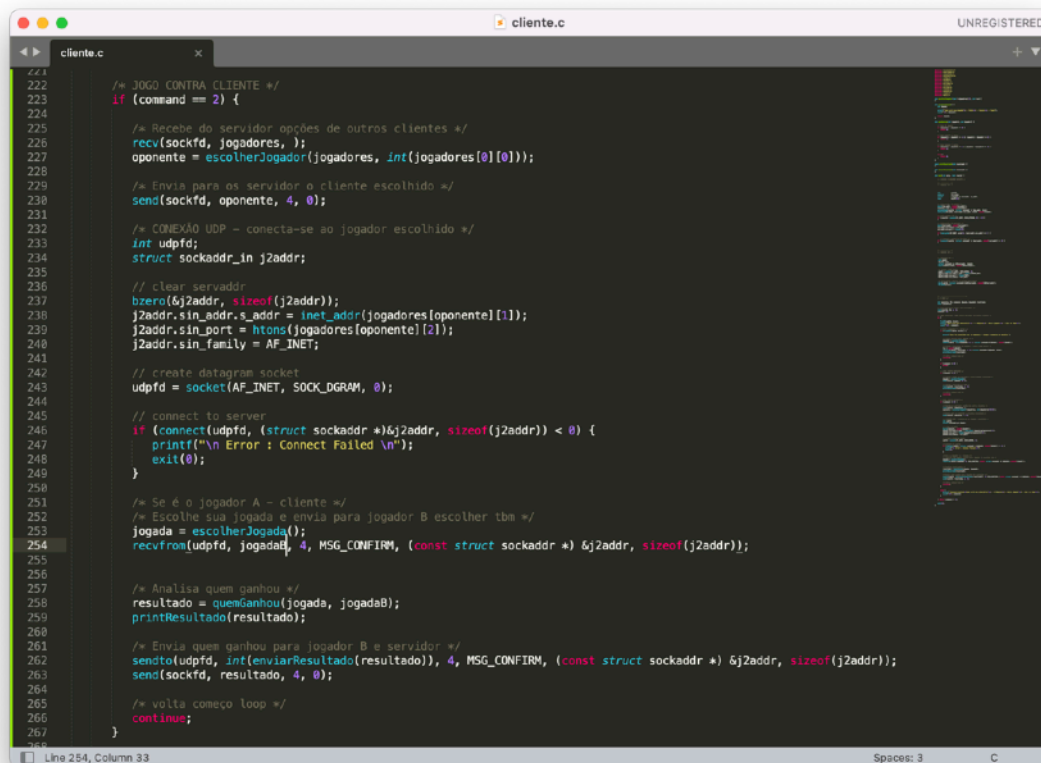
imagem 7: Cliente.c estabelecendo servidor UDP.



```
177
178 /* Se A o jogador B - server */
179 if (FD_ISSET(udpfd, &rset)) {
180
181     printf("Você foi escolhido por um oponente - chegou o momento da batalha.");
182
183     /* Envia escolha para jogador A */
184     jogadaB = escolherJogada();
185     sendto(udpfd, (int*)jogadaB, 4, 0, (struct sockaddr*)&jAaddr, sizeof(jAaddr));
186
187     /* Recebe informação do jogador A sobre quem ganhou */
188     len = sizeof(jAaddr);
189     recvfrom(udpfd, resultado, 4, 0, (struct sockaddr*)&jAaddr, &len);
190     printResultado(resultado);
191
192     /* volta começo loop */
193     continue;
194 }
195
```

imagem 8: Cliente.c atuando como jogador B na conexão UDP.

Com isso a partida é encerrada e ambos os jogadores estão disponíveis para novos jogos. O jogador é desconectado do servidor caso escolha o número 0 nesse menu inicial.



```
221
222 /* JOGO CONTRA CLIENTE */
223 if (command == 2) {
224
225     /* Recebe do servidor opções de outros clientes */
226     recv(sockfd, jogadores, );
227     oponente = escolherJogador(jogadores, int(jogadores[0][0]));
228
229     /* Envia para o servidor o cliente escolhido */
230     send(sockfd, oponente, 4, 0);
231
232     /* CONEXÃO UDP - conecta-se ao jogador escolhido */
233     int udpfd;
234     struct sockaddr_in j2addr;
235
236     // Clear servaddr
237     bzero(&j2addr, sizeof(j2addr));
238     j2addr.sin_addr.s_addr = inet_addr(jogadores[opponente][1]);
239     j2addr.sin_port = htons(jogadores[opponente][2]);
240     j2addr.sin_family = AF_INET;
241
242     // create datagram socket
243     udpfd = socket(AF_INET, SOCK_DGRAM, 0);
244
245     // connect to server
246     if (connect(udpfd, (struct sockaddr*)&j2addr, sizeof(j2addr)) < 0) {
247         printf("\n Error : Connect Failed \n");
248         exit(0);
249     }
250
251     /* Se é o jogador A - cliente */
252     /* Escolhe sua jogada e envia para jogador B escolher tbm */
253     jogada = escolherJogada();
254     recvfrom(udpfd, jogadaB, 4, MSG_CONFIRM, (const struct sockaddr*)&j2addr, sizeof(j2addr));
255
256     /* Analisa quem ganhou */
257     resultado = quemGanhou(jogada, jogadaB);
258     printResultado(resultado);
259
260     /* Envia quem ganhou para jogador B e servidor */
261     sendto(udpfd, int(enviarResultado(resultado)), 4, MSG_CONFIRM, (const struct sockaddr*)&j2addr, sizeof(j2addr));
262     send(sockfd, resultado, 4, 0);
263
264     /* volta começo loop */
265     continue;
266 }
267
```

imagem 9: Cliente.c atuando como jogador A na conexão UDP.

O arquivo cliente por sua vez, precisa estabelecer e estar preparado para realizar a comunicação via UDP, exercendo o papel de servidor ou cliente, dependendo do caso.

Por esse motivo, logo no início é estabelecido um servidor UDP e, antes mesmo de analisar as escolhas do modo de jogo do jogador, se o servidor UDP escuta o recebimento de dados, automaticamente ele assume a posição de jogador B na partida.

Nesse cenário, o jogador B que atua como servidor escolhe sua jogada e a envia ao jogador A, aguardando o retorno do resultado.

O jogador A, que atua como cliente, após iniciar a comunicação, realiza sua jogada e recebe a escolha do jogador B. Esse resultado é então processado e enviado tanto para o jogador B quanto para o servidor do jogo.