Relatório T.1

Gustavo Vicente Barroso Moser (16204938)
Patrick Machado da Silva (15205393)
Dezembro 2021

1. Objetivo

O objetivo do presente trabalho foi desenvolver um *jogo da forca* distribuído utilizando comunicação por sockets entre cliente e servidor utilizando a linguagem C e suas bibliotecas nativas. Desta forma, deveria ser criado um servidor e diversos clientes, podendo dessa forma, ter vários jogos acontecendo simultaneamente.

2. Detalhes de implementação e decisões do projeto

A implementação se deu de forma bem simples, seguindo as regras do jogo. O jogo foi todo desenvolvido para apresentar as informações no console, simplificando o desenvolvimento.

a. Servidor

A imagem abaixo apresenta a função main do servidor.

```c
int main() {
  int server_socket, client_socket, client_addr_size;
  struct sockaddr_in server_addr, client_addr;

  validate(server_socket = socket(AF_INET, SOCK_STREAM, 0), "[-]Falha ao criar socket");

  server_addr.sin_family = AF_INET;
  server_addr.sin_addr.s_addr = INADDR_ANY;
  server_addr.sin_port = htons(SERVER_PORT);

  validate(bind(server_socket, (struct sockaddr*) &server_addr, sizeof(server_addr)), "[-]Bind falhou");

  listen(server_socket, 5);
  printf("[+]Servidor disponível na porta %d\n", SERVER_PORT);

  pthread_t thread[5];

  for (int i = 0; i < NUM_THREADS; i++) {
    client_addr_size = sizeof(client_addr);
    client_socket = accept(server_socket, (struct sockaddr*) &client_addr, &client_addr_size);
    validate(client_socket, "[-]Falha ao aceitar conexão");
    printf("[+]Conexao em %d estabelecida!\n", i);

    int* cli_ptr = malloc(sizeof(int));
    *cli_ptr = client_socket;

    printf("[+]Executando thread %d\n", i);
    pthread_create(&thread[i], NULL, handle_connection, cli_ptr);
  }

  for (int i = 0; i < NUM_THREADS; i++) {
    pthread_join(thread[i], NULL);
  }

  return 0;
}
```

Imagem 1 - Main do servidor.

Observa-se que o servidor foi desenvolvido utilizando threads, assim, a tarefa a ser executada pelo servidor é delegada a uma thread, buscando disponibilizar acesso a diversos clientes, não bloqueando o servidor. O valor de threads disponíveis foi definido como 5 threads.

Também foram definidas globalmente duas listas de strings, contendo as palavras utilizadas no jogo e as partes do corpo no jogo da forca.

```
16  char *words[8] = {"computador\0", "paralela\0", "distribuida\0", "concorrencia\0",
17    "servidor\0", "cliente\0", "ciencia\0", "computacao\0" };
18  char *body_pieces[5] = {"cabeca\0", "bracos\0", "tronco\0", "pernas\0", "pes\0"};
```

Imagem 2 - Listas de strings com definições de palavras do jogo.

A palavra escolhida para ser utilizada no jogo é copiada para uma string e convertida em uma string de zeros, por exemplo: *parada = 000000*. Essa string será utilizada para apresentar ao jogador o seu progresso no jogo. Essa operação está contida dentro da lógica do jogo, onde, a cada letra encontrada, essa letra é substituída, então seguindo o exemplo apresentado anteriormente, caso a letra "*a*" seja encontrada, a string apresentada ficará como: *0a0a0a*.

```
validate(recv(client_socket, &read_ch, sizeof(char), 0), "[-]Erro em recv\n");

found = 0;
int used_letter = 0;
for (int i = 0; i < word_size; i++) {
  char pos = word[i];
  if (pos == read_ch) {
    if (underscored[i] != '0') {
      used_letter++;
    } else {
      found++;
      underscored[i] = pos;
      zeros_remaining--;
    }
  }
}

if (found <= 0 && used_letter == 0) {
  game_piece_count++;

  memset(game_pieces, 0, sizeof(game_pieces));
  for(int i = 0; i < game_piece_count; i++) {
    if (i == 0) {
      strcat(game_pieces, body_pieces[i]);
    } else {
      char separator[3] = ", ";
      strcat(game_pieces, separator);
      strcat(game_pieces, body_pieces[i]);
    }
  }

  if (game_piece_count == NUM_BODY_PIECES) break;
}
```

Imagem 3 - Lógica do jogo no servidor.

A figura acima apresenta a lógica desenvolvida para o jogo. Um caractere é lido de uma requisição do cliente e em seguida, é validado na palavra escolhida para confirmar existência (ou não). Em seguida, caso o caractere não seja encontrado na palavra de referência, há a contabilização dos erros, ou seja, das partes do corpo presentes na forca. Letras que já foram encontradas não são contabilizadas caso haja novas inserções da mesma.

O jogo acaba se todas as peças forem apresentadas, e o jogador perde, ou caso não restem zeros na palavra, e o jogador vence.

b. Cliente

O cliente é o responsável por apresentar as informações e o andamento do jogo ao usuário, bem como tratar sua entrada, o caractere, no jogo quando necessário. Também é responsável por apresentar o resultado do jogo, fechando a conexão em caso de vitória ou derrota do usuário.

3. Limitações da implementação atual

Apesar de ser um jogo, a disponibilização do mesmo para produção não é recomendada. Existem diversos pontos de melhoria, o primeiro é a respeito do tratamento do uso de memória do jogo, onde algumas variáveis poderiam ser melhor alocadas, ou sua utilidade melhor avaliada.

Existem também alguns problemas relacionados à boas práticas de clean code e organização do código, a manutenção do mesmo para problemas seria um tanto quanto complexa. Por fim, a implementação de mais regras, validações e até uma interface mais intuitiva e interativa contribuiria para um resultado melhor do desenvolvimento do jogo.