



Universidade de Brasília - UnB

Disciplina: Fundamento de Redes para Computadores

Engenharia de Software

Professor: Dr. Fernando William Cruz

Gustave A. Persijn – 190046091

Daniel Vinícius R. A. – 190026375

Lorrayne A. Cardozo - 190032863

## **Projeto de pesquisa - Criando ambientes virtuais de conversação com uso system call select().**

### **1. Objetivo**

A comunicação em rede é essencial nos dias de hoje, e compreender a arquitetura de aplicações que envolvem a troca de informações é de suma importância. Nesse contexto, o presente relatório aborda o desenvolvimento de uma aplicação de salas de bate-papo virtuais, baseada na arquitetura TCP/IP. O objetivo principal é permitir que os usuários criem salas de bate-papo, interajam entre si e realizem a gestão dessas salas. Para atingir esse objetivo, foram implementadas funcionalidades como criação de salas, listagem de participantes, ingresso e saída de clientes, além do diálogo entre eles. A solução proposta busca oferecer uma compreensão prática sobre a arquitetura de aplicações de rede, utilizando a linguagem C e a System call select() para o gerenciamento de diálogos.

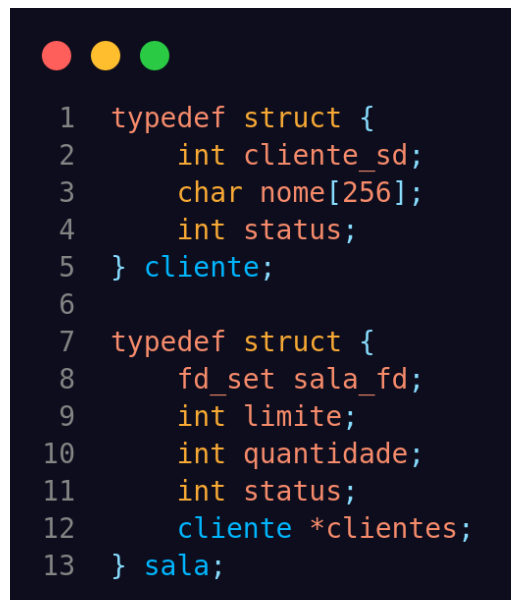
### **2. Metodologia utilizada**

O trabalho foi realizado em grupo, nos quais os membros se organizaram de forma colaborativa para desenvolver a aplicação de salas de bate-papo virtuais. Inicialmente, foram realizados encontros para discutir a arquitetura da aplicação e definir as funcionalidades que seriam implementadas. Durante essas reuniões, também foram divididas as tarefas entre os membros do grupo.. Após essa etapa, cada membro ficou responsável por desenvolver um módulo específico da aplicação, como o cliente ou o servidor. Durante o processo de

implementação, foram realizados encontros periódicos para revisar o código, solucionar dúvidas e garantir a integração adequada dos diferentes módulos. A troca de conhecimentos e a colaboração foram fundamentais para o progresso do projeto.

### 3. Descrição da Solução

O código fornecido implementa um servidor para a criação de salas de bate-papo utilizando a função `select`. A solução envolve a utilização de estruturas de dados para representar as *salas* e os *clientes*, bem como a manipulação dos conjuntos de *file descriptors* (`fd_set`) para controlar as conexões e as comunicações entre os clientes.

A screenshot of a code editor with a dark background and syntax highlighting. It shows two C struct definitions. The first struct, named 'cliente', has three members: 'cliente\_sd' (int), 'nome' (char array of size 256), and 'status' (int). The second struct, named 'sala', has five members: 'sala\_fd' (fd\_set), 'limite' (int), 'quantidade' (int), 'status' (int), and a pointer to the 'cliente' struct named 'clientes'. Line numbers 1 through 13 are visible on the left side of the code block.

```
1  typedef struct {
2      int cliente_sd;
3      char nome[256];
4      int status;
5  } cliente;
6
7  typedef struct {
8      fd_set sala_fd;
9      int limite;
10     int quantidade;
11     int status;
12     cliente *clientes;
13 } sala;
```

Figura 01 - Estruturas de dados para representar Salas e Clientes.

O código define estruturas `sala` e `cliente` para armazenar informações sobre as salas e os clientes, respectivamente. A função `send_message` é responsável por enviar mensagens para todos os clientes de uma sala, exceto para o cliente que enviou a mensagem. A função `exit_chat_room` é utilizada para remover um cliente de uma sala.

```

1 void send_message(int sd, int server_sd, int sala_id, int cliente_id) {
2     printf("Enviando mensagem do file descriptor %d para a sala %d\n", sd, sala_id);
3     for (int j = 0; j <= fdmax; j++)
4         if (FD_ISSET(j, &salas[sala_id].sala_fd))
5             if (j != sd && j != server_sd) {
6                 char mensagem[500] = "[";
7                 strcat(mensagem, salas[sala_id].clientes[cliente_id].nome);
8                 strcat(mensagem, "] => ");
9                 strcat(mensagem, buf);
10                send(j, mensagem, 500, 0);
11            }
12 }

```

Figura 02 - Função *send\_message*.

```

1 void exit_chat_room(int sd, int sala_id, int cliente_id, int remover) {
2     salas[sala_id].clientes[cliente_id].status = 0;
3     salas[sala_id].quantidade--;
4
5     if (remover == 1){
6         FD_CLR(sd, &master);
7     }
8
9     FD_CLR(sd, &salas[sala_id].sala_fd);
10    if (salas[sala_id].quantidade == 0) {
11        free(salas[sala_id].clientes);
12        salas[sala_id].status = 0;
13    }
14    printf("O file descriptor %d saiu da sala %d\n", sd, sala_id);
15 }

```

Figura 03 - Função *exit\_chat\_room*.

A função *init\_server* é chamada para inicializar as salas, e a função *create\_chat\_room* é responsável por criar uma nova sala com um limite de clientes definido. A função *insert\_chat\_room* é utilizada para adicionar um cliente a uma sala específica.

```

1 void init_server() {
2     for (int i = 0; i < MAX_SALAS; i++) {
3         FD_ZERO(&salas[i].sala_fd);
4         salas[i].limite = 0;
5         salas[i].quantidade = 0;
6         salas[i].status = 0;
7     }
8 }

```

Figura 04 - Função *init\_server*.

```

1  int create_chat_room(int limite) {
2      int sala;
3      for (sala = 0; sala < MAX_SALAS; sala++)
4          if (salas[sala].status == 0)
5              break;
6
7      salas[sala].status = 1;
8      salas[sala].limite = limite;
9      salas[sala].clientes = malloc(limite * sizeof(cliente));
10     for (int i = 0; i < limite; i++){
11         salas[sala].clientes[i].status = 0;
12     }
13     printf("A sala %d foi criada com sucesso\n", sala);
14     return sala;
15 }

```

Figura 05 - Função *create\_chat\_room*

```

1  void insert_chat_room(int sd, int sala_id, char nome[], int tam_nome) {
2      printf("File descriptor %d entrando na sala %d\n", sd, sala_id);
3      FD_SET(sd, &salas[sala_id].sala_fd);
4      salas[sala_id].quantidade++;
5      for (int i = 0; i < salas[sala_id].limite; i++) {
6          if (salas[sala_id].clientes[i].status == 0) {
7              salas[sala_id].clientes[i].cliente_sd = sd;
8              salas[sala_id].clientes[i].status = 1;
9              strncpy(salas[sala_id].clientes[i].nome, nome, tam_nome);
10             break;
11         }
12     }
13 }

```

Figura 06 - Função *insert\_chat\_room*.

A função *control* é responsável por processar comandos enviados pelos clientes, como sair da sala, listar os clientes conectados ou trocar de sala. Esses comandos são executados de acordo com a mensagem recebida do cliente.

```

1 void control(int sd, int sala_id, int cliente_id) {
2     buf[strlen(buf) - 2] = '\0';
3     printf("Comando \"%s\" acionado na sala %d pelo file descriptor %d\n", buf, sd, sala_id);
4     char resp_buf[256];
5
6     if (strncmp(buf+1, "sair", 4) == 0) {
7         printf("Desconectando file descriptor %d\n...", sd);
8         strcpy(resp_buf, "Desconectado!\n");
9         send(sd, resp_buf, strlen(resp_buf), 0);
10        close(sd);
11        exit_chat_room(sd, sala_id, cliente_id, 1);
12    }
13
14    if (strncmp(buf+1, "integrantes_conectados", 6) == 0) {
15        send(sd, "\nIntegrantes conectados na sala: ", 40, 0);
16        for (int i = 0; i < salas[sala_id].limite; i++) {
17            cliente c = salas[sala_id].clientes[i];
18            if (c.status == 1 && c.cliente_sd != sd) {
19                char nome[] = "\n";
20                strcat(nome, c.nome);
21                send(sd, nome, strlen(nome), 0);
22            }
23            else if (c.status == 1 && c.cliente_sd == sd) {
24                char nome[] = "\n[";
25                strcat(nome, c.nome);
26                strcat(nome, "]");
27                send(sd, nome, strlen(nome), 0);
28            }
29        }
30        send(sd, "\n\n", 2, 0);
31    }
32
33    if (strncmp(buf+1, "trocar_de_sala", 11) == 0) {
34        recv(sd, buf, 256, 0);
35        int nova_sala = atoi(buf);
36        char nome[256];
37        strcpy(nome, salas[sala_id].clientes[cliente_id].nome);
38        exit_chat_room(sd, sala_id, cliente_id, 0);
39        insert_chat_room(sd, nova_sala, nome, strlen(nome));
40    }
41 }

```

Figura 07 - Função *control*.

No main, o código realiza a configuração do socket, aceita as conexões dos clientes e gerencia as comunicações entre eles por meio do uso do select. Ele lê as mensagens recebidas dos clientes e as encaminha para a função apropriada, dependendo se é um comando ou uma mensagem para ser enviada para a sala. A solução utiliza a função select para monitorar múltiplos file descriptors e permite que os clientes interajam em salas de bate-papo, enviando mensagens, executando comandos e trocando de sala quando necessário.

#### 4. Conclusão

Em suma, a realização deste projeto de desenvolvimento da aplicação de salas de bate-papo virtuais trouxe valiosas contribuições para a compreensão da arquitetura de aplicações de rede e a utilização do select. Com isso, a colaboração em equipe fortaleceu a troca de conhecimentos para a realização e desenvolvimento do trabalho proposto, permitindo o aprendizado do conteúdo e melhor entendimento do mesmo.

## **5. Referências Bibliográficas**

TANENBAUM, Andrew S.; WETHERALL, David. Redes de Computadores. 5. ed. São Paulo: Pearson, 2012.