

**UnB - Universidade de Brasília**  
**Fundamento de Redes de Computadores**  
**Professor: Fernando William Cruz**

**Projeto Final da Disciplina - Criando Ambientes Virtuais de Conversação com  
Uso da System Call select()**

**Participantes**

Natanael Filho - 19/0058650

Sávio Cunha - 18/0130889

Vitor Diniz - 18/0132385

Brasília, DF  
25/09/2022

## 1. Introdução

É impossível imaginar a internet sem o uso de um mensageiro, visto que hoje, por exemplo as redes sociais, fazem uso de chats de bate papo, código autenticador enviado por SMS, ou entre outros. Sobretudo, um dos fatores que tornou isso possível ressaltamos o protocolo TCP/IP, pois com ele é possível permitir a criação de comunicação entre usuários a partir de uma aplicação servidor-cliente.

Neste trabalho, foi elaborado em C um servidor capaz de estabelecer e efetuar a gerência da comunicação entre os usuários. Além disso, usuários podem utilizar o comando telnet para criar sala, enviar mensagens, listar participantes da conversa, trocar e sair da sala.

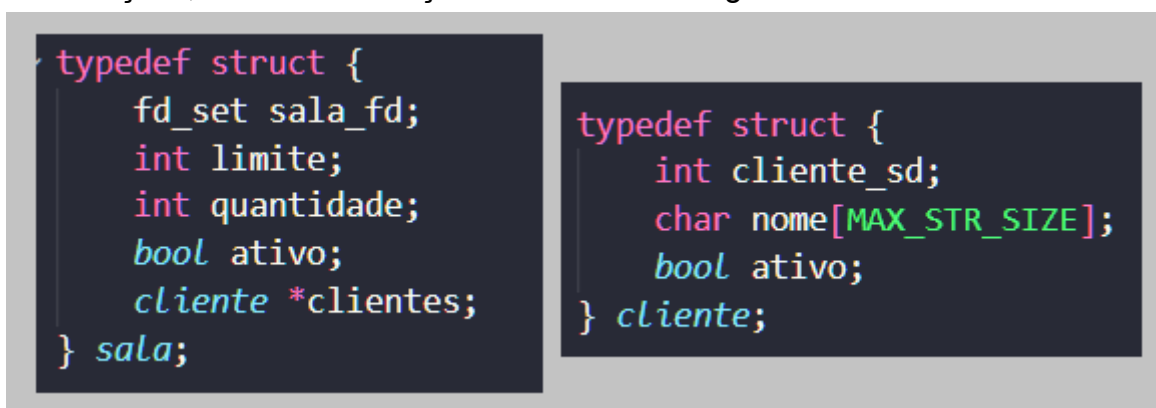
## 2. Metodologia Utilizada

Para a elaboração deste projeto de pesquisa, a equipe utilizou dos materiais disponibilizados pelo professor, além de conteúdos adicionais da internet, desta forma conseguimos mais informações sobre o uso da syscall select() e de outras funções necessárias para a elaboração do projeto.

Sobre a organização da equipe para a elaboração do trabalho, na semana anterior a entrega do projeto, separamos cada funcionalidade a partir de sua prioridade e dependências de outras funções e a partir disso aplicamos o pair programming durante quatro dias (terça, quarta, quinta e sábado), onde o cada dia foram utilizados cerca de duas horas. Por fim, no dia antecedente a entrega do projeto a equipe se juntou durante a tarde para o desenvolvimento das documentações necessárias, sendo eles slides de apresentação e este relatório em questão.

## 3. Descrição da Solução

A solução proposta consiste na declaração de duas structs, sala e cliente, e seis 6 funções, excluindo a função main desta contagem.



```
typedef struct {  
    fd_set sala_fd;  
    int limite;  
    int quantidade;  
    bool ativo;  
    cliente *clientes;  
} sala;  
  
typedef struct {  
    int cliente_sd;  
    char nome[MAX_STR_SIZE];  
    bool ativo;  
} cliente;
```

Figura 1: structs sala e cliente do projeto.

Fonte: Autor.

Mormente, na Figura 1, apresentamos as duas structs do projeto, onde a primeira, struct sala, contém as informações de uma sala, sendo elas:

- **sala\_fd**: fd\_set que será usado como identificador da sala.

- **limite:** inteiro que representa limite de usuários para aquela sala.
- **quantidade:** inteiro representando quantidade de clientes ativos na sala.
- **ativo:** booleano que representa se a sala está ativa para uso dos clientes.
- **clientes:** ponteiro responsável por armazenar usuários ativos na sala de bate papo.

Além disso, também na Figura 1, temos a struct cliente responsável por caracterizar as informações que o cliente necessitará declarar:

- **client\_sd:** inteiro representativo o descriptor.
- **nome:** nome escolhido pelo cliente
- **ativo:** booleano que representa se usuário está ou não ativo.

Falando um pouco sobre as funções do projeto, seguiremos o fluxo da função main, onde a primeira função chamada é a função de preparação do servidor. A partir da Figura 2, vemos que sua função é de limpar as salas e definir valores padrões de inicialização para cada.

```
void prepara_servidor() {
    // Faz a Limpeza dos sets master
    FD_ZERO(&master);
    FD_ZERO(&read_fds);
    // Inicializacao do servidor, zerando valores de todas as salas
    for (int i = 0; i < MAX_SALAS; i++) {
        FD_ZERO(&salas[i].sala_fd);
        salas[i].limite = 0;
        salas[i].quantidade = 0;
        salas[i].ativo = false;
    }
}
```

Figura 2: função prepara\_servidor() do projeto.

Fonte: Autor.

Após a elaboração desta limpeza é feito a inicialização do servidor diretamente na função main, nela fazemos a inicialização dos métodos de select para a gerência e criação de fluxos e grupos na aplicação, conforme mostrado na figura 3.

```

// Faz a Limpeza dos sets master e das salas e inicializa o servidor
prepara_servidor();

// Configuracao de socket
int sd = socket(AF_INET, SOCK_STREAM, 0);
setsockopt(sd, SOL_SOCKET, SO_REUSEADDR, &yes, sizeof(int));

myaddr.sin_family = AF_INET;
myaddr.sin_addr.s_addr = inet_addr(argv[1]);
myaddr.sin_port = htons(atoi(argv[2]));
memset(&(myaddr.sin_zero), 0, 8);

// Bind e listen nesse socket descriptor
bind(sd, (struct sockaddr *)&myaddr, sizeof(myaddr));
listen(sd, 10);

// Adiciona os file descriptors no set master
FD_SET(sd, &master);
FD_SET(0, &master);

```

Figura 3: processo de inicialização do servidor.

Fonte: Autor.

Em seguida, com o servidor iniciado e o cliente ter estabelecido conexão a partir do telnet, são utilizadas as funções `recv` para capturar os dados inseridos pelo cliente sendo eles, nome, número da sala que irá entrar, caso insira -1 entrará no fluxo de ativação de sala requisitando também uma capacidade de pessoas que a sala irá suportar.

```

// Conexao, adiciona o socket descriptor no cesto
newfd = accept(sd, (struct sockaddr *)&remoteaddr, &addrlen);
FD_SET(newfd, &master);

int limite, tam_nome;
char nome[MAX_STR_SIZE];
// Recebe nome do usuario
tam_nome = recv(newfd, nome, MAX_STR_SIZE, 0);
tam_nome -= 2;
// Recebe numero da sala
recv(newfd, buf, MAX_STR_SIZE, 0);
sala = atoi(buf);

// Se a sala -1, cria uma nova
if (sala == NOVA_SALA) {
    // informa limite da sala
    recv(newfd, buf, MAX_STR_SIZE, 0);
    limite = atoi(buf);
    sala = cria_sala(limite);
}

// De qualquer forma insere ele na sala nova ou existente
entrar_na_sala(newfd, sala, nome, tam_nome);

```

Figura 4: fluxo de conexão de novo cliente no servidor.

Fonte: Autor.

Portanto, com clientes conectados e já inseridos em alguma sala, será possível via terminal a utilização do chat para bate papo. Os requisitos para estabelecimento de comunicação serão que os usuários deverão estar na conexão na mesma sala do servidor, ao entrar no servidor a capacidade máxima de clientes em uma mesma sala não deverão estourar e para que as mensagens enviadas no servidor serão apresentadas no terminal onde cliente executou o comando telnet, o mesmo serve para as mensagens que o cliente queira enviar para outros clientes.

#### **4. Conclusão**

A solução desenvolvida torna possível a criação de salas de bate-papo em ambiente virtual com o uso da `system call select()`, utilizando sockets. Também é possível listar os clientes conectados na sala, e promover a troca de clientes em cada sala.

##### **4.1. Relatos da equipe**

Natanael Fernandes: no geral o trabalho foi bem tranquilo e produtivo, com ressalvas na elaboração de inicialização do servidor, visto que tivemos alguns problemas iniciais na elaboração da conexão com usuários via telnet, acredito que pelo menos eu demorei um pouco para entender como o processo funciona. Acho que no geral como melhoria para disciplinas futuras, vejo a aplicação de aulas prática facilitaria bastante o desenvolvimento da disciplina, visto que quando ficamos muito tempo apenas na teoria, quando tentamos aplicar acaba ficando complexo e confuso.

Sávio Cunha: A princípio o trabalho parecia muito confuso, porém esse trabalho proporcionou um conhecimento extremamente relevante em relação aos conteúdos da matéria. Poder realizar os laboratórios de forma prática, ajudou bastante a entender os conteúdos e tenho certeza que essa metodologia de trabalhos práticos será lucrativa em todos os semestres.

Vitor Diniz: O trabalho proporcionou um aprendizado bastante interessante acerca do funcionamento de servidores para gerenciamento de troca de informações. Além disso, foi possível aprender a utilizar o comando `select()`, que permite que um sistema acompanhe vários descritores de arquivo. O principal ponto que eu gostei do trabalho, é pela a oportunidade de sair um pouco da teoria e colocar mais em prática os conceitos que aprendemos em sala de aula, e seus possíveis problemas.

#### **5. Referências Bibliográficas**

- [1] CRUZ, F. W. System Call Select(). Disponível em: <[https://aprender3.unb.br/pluginfile.php/2263476/mod\\_resource/content/1/selectserver.c](https://aprender3.unb.br/pluginfile.php/2263476/mod_resource/content/1/selectserver.c)>. Acesso em: 26 sep. 2022.
- [2] PRACTICALS, C. S. E. Learn Select system call in CCSE Practicals, , 19 Aug. 2018. Disponível em: <<https://www.youtube.com/watch?v=CfEShMmsUus>>
- [3] Man7. select(2) — Linux manual page. Disponível em: <<https://man7.org/linux/man-pages/man2/select.2.html>>. Acesso em: 25 sep. 2022.