

Diener Dornelas  
Franciane Gonçalves  
Thiago Toledo

## **DOCUMENTAÇÃO DO TRABALHO PRÁTICO DE AEDS II**

UFSJ  
2019

## INTRODUÇÃO

Este trabalho é um simulador de um servidor de e-mails, feito com o objetivo de figurar todos os processos de uma caixa de e-mail convencional, entre eles: Cadastrar usuário, enviar mensagens, consultar a caixa de entrada de cada login e excluir usuário.

O programa conta com duas estruturas, várias funções e a função principal(main). Além disso, foi organizado em três partes, sendo F\_tp.c , F\_tp.h e Tp\_main.c.

## IMPLEMENTAÇÃO

O tipo conta é uma lista simplesmente encadeada, que possui um ponteiro para caixa de entrada e um para os usuários (contas cadastradas).

O tipo caixa de entrada é uma lista bifurcada simplesmente encadeada, que possui um ponteiro para as caixas de entradas e um outro ponteiro para as mensagens de cada ID existente no servidor.

## FUNÇÕES

**Int verificarID (int login, Conta \*ContaMain)** : essa função recebe como parâmetro o ID e um ponteiro para a lista de contas já existentes( que pode ser vazia ou não). Essa função é usada para verificar a existência desse ID (login) que foi recebido, é utilizada no cadastro de um novo usuário e no envio de mensagens para qualquer um deles. A função verificarID retorna 0 caso o ID exista, 1 caso não exista e 2 caso seja maior que 100000.

**Void cadastraID (Conta \*conta, CaixaEntrada \*email)** : a função cadastra ID recebe como parâmetro um ponteiro para a lista de conta e um para a caixa de entrada. Nesse processo a função verificar ID é chamada para que caso esse ID já exista a função retorna um ERRO: CONTA JÁ CADASTRADA, e caso não haja nenhum id com esse mesmo valor de identificação a função executa o cadastro normalmente, aparecendo : CONTA CADASTRADA. Quando o ID que teve cadastro solicitado for o primeiro da lista de contas a célula cabeça aponta diretamente para ele e ele para NULL, o que sempre acontece com o último da lista. Quando o ID solicitado for maior que 100000 a função retorna uma mensagem : ERRO: CONTA INVÁLIDA.

**Void OrdenaMsg( CaixaEntrada \*dest, char email[ ], int prioridade, int id):** essa função tem como objetivo ordenar as mensagens na caixa de entrada de acordo com a prioridade de cada uma. A função OrdenaMsg recebe como parâmetro a caixa de entrada do destinatário, a mensagem, a prioridade e o ID do destinatário. A prioridade vai de 0 a 9, sendo 9 a maior prioridade que uma mensagem pode ter. A função ordena as mensagens cada vez que uma nova mensagem chega a caixa de entrada.

**Void InserirMsg(CaixaEntrada \*email, int id, Conta \*EmailMain, FILE \*Arquivo):** a função InserirMsg recebe como parâmetro um ponteiro para as

caixas de entrada, o ID do destinatário, a lista de contas e o arquivo com a mensagem e a prioridade. Essa função chama a função verificarID, caso o ID exista ele verifica a prioridade, se ela for maior que 9, retorna a mensagem: ERRO: PRIORIDADE INVALIDA, se for menor que 9 ela chama função OrdenaMsg para já colocar essa nova mensagem no seu devido lugar e retorna a mensagem: OK: MENSAGEM PARA ID ENTREGUE. Caso o ID não exista ou seja maior 100000 a função retorna: CONTA ID NÃO EXISTENTE.

**Void ConsultarID(CaixaEntrada \*UsuarioMain, int endereco, Conta \*ListaConta):** essa função recebe como parâmetro um ponteiro para a lista de caixa de entrada, o ID a ser consultado e a lista de contas. Essa função tem a finalidade de abrir sempre a mensagem de maior prioridade, que é a primeira da lista de mensagens na caixa de entrada. Se o ID solicitado não existir aparece a mensagem: CAIXA DE ENTRADA INEXISTENTE, e caso não haja nenhuma mensagem para esse usuário retorna a mensagem: CAIXA DE ENTRADA VAZIA.

## **FUNÇÃO MAIN**

Essa função cria um ponteiro para conta e caixa de entrada, esses ponteiros são alocados dinamicamente e somente quando for o primeiro dessas listas. A função recebe o arquivo dos comandos que executam o programa e através de um while lê a quantidade de linhas deste arquivo. Em um segundo while, que tem como parâmetro um contador que enquanto for menor que o número de linhas, o arquivo será lido. Dentro desse while são feitas comparações de string que definem e chamam a função desejada para aquele processo.

## **ORGANIZAÇÃO DO CÓDIGO**

O código foi dividido em arquivos: F\_tp.c , F\_tp.h implementam todas as funções e estruturas necessárias, enquanto tp\_main.c é a função principal do programa.

O código tem como uma constante o valor máximo de contas, que é 100000.

Foram criadas duas estruturas, sendo elas Conta e Caixa de entrada. A primeira é uma lista simplesmente encadeada e a outra é uma lista bifurcada simplesmente encadeada. A Caixa de Entrada tem um ponteiro para a próxima caixa de entrada e um outro ponteiro para o próximo e-mail. A Conta possui um ponteiro para a próxima conta e um outro para suas respectivas caixas de entrada.

O compilador utilizado na execução do programa foi o gcc na versão 5.4.0, no Ubuntu 16.04.11 LTS. Para executar o programa é preciso inserir as entradas no arquivo Comandos.txt e digitar MAIN na linha de comando. O programa também foi testado no Windows através do programava Dev C++ 5.11, com compilador na versão 4.9.2 64bits.

## **ANÁLISE DE COMPLEXIDADE**

**Int verificarID** : no pior caso, a função executa um laço no while que é  $n$  vezes  $O(1)$ , o que faz a função ser  $n \cdot O(1) = O(n)$ .

**Void CadastraID** : em seu pior caso, entrará no while para percorrer toda a lista de contas, que é  $O(n)$ .

**Void OrdenaMsg** : a função tem como complexidade  $O(n)$ , porque no pior dos casos ela terá que verificar todas as mensagens existentes na caixa de entrada até encontrar o lugar em que a mensagem recebida terá a sua ordenação.

**Void InserirMsg** : no pior caso dessa função, ela atenderá a primeira opção do switch, que terá um while que é  $O(n)$  e um else que é  $O(1)$ .  $O(n) = O(n)$ , o que faz com que a função seja  $O(n) + O(n) = O(n)$ .

**Void RemoverID**: em seu pior caso a função entra em um while, que possui dois if, que tem outro laço dentro dela, com a função de remover as mensagens da caixa de entrada desse usuário, deste modo será  $O(n) \cdot O(1) \cdot O(1) \cdot O(m) = O(nm)$ .

**Void ConsultarID**: nessa função seu pior caso também é um while com um if dentro dela, sendo  $O(n) + O(1) = O(n)$ .

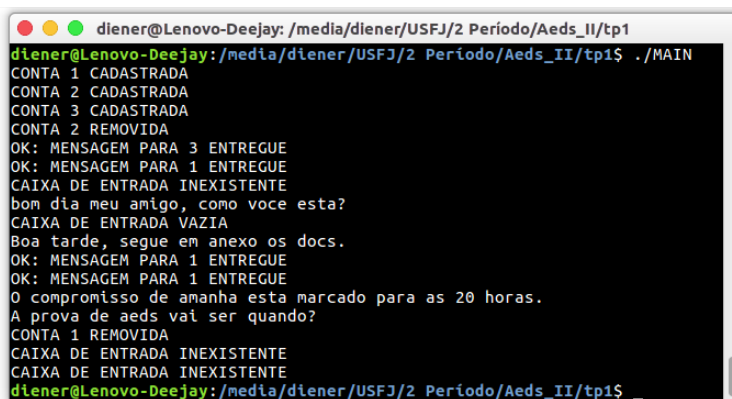
**Void Main** : a função main possui um while com um contador que faz com que esse laço seja executado  $x$  vezes, em seu pior caso, dentro desse while entrará na opção para remover id, que é  $O(nm)$ , então a função terá complexidade  $x \cdot O(nm)$ .

## TESTES

Vários testes foram realizados a fim de testar o funcionamento do programa com um arquivo contendo comandos e entradas. Os testes foram realizados em um Intel I3-8130U com 8Gb de memória.

A Imagem abaixo mostra os comandos que foram os argumentos de entrada e as saídas esperadas durante a execução do programa.

```
CADASTRA 1
CADASTRA 2
CADASTRA 3
REMOVE 2
ENTREGA 3 4 bom dia meu amigo, como voce esta? FIM
ENTREGA 1 7 Boa tarde, segue em anexo os docs. FIM
CONSULTA 6
CONSULTA 3
CONSULTA 3
CONSULTA 1
ENTREGA 1 7 O compromisso de amanha esta marcado para as 20 horas. FIM
ENTREGA 1 2 A prova de aeds vai ser quando? FIM
CONSULTA 1
CONSULTA 1
REMOVE 1
CONSULTA 1
CONSULTA 2
```



```
diener@Lenovo-Deejay: /media/diener/USFJ/2 Período/Aeds_II/tp1$ ./MAIN
CONTA 1 CADASTRADA
CONTA 2 CADASTRADA
CONTA 3 CADASTRADA
CONTA 2 REMOVIDA
OK: MENSAGEM PARA 3 ENTREGUE
OK: MENSAGEM PARA 1 ENTREGUE
CAIXA DE ENTRADA INEXISTENTE
bom dia meu amigo, como voce esta?
CAIXA DE ENTRADA VAZIA
Boa tarde, segue em anexo os docs.
OK: MENSAGEM PARA 1 ENTREGUE
OK: MENSAGEM PARA 1 ENTREGUE
O compromisso de amanha esta marcado para as 20 horas.
A prova de aeds vai ser quando?
CONTA 1 REMOVIDA
CAIXA DE ENTRADA INEXISTENTE
CAIXA DE ENTRADA INEXISTENTE
diener@Lenovo-Deejay: /media/diener/USFJ/2 Período/Aeds_II/tp1$
```

## CONCLUSÃO

Durante a programação do servidor de e-mail a maior dificuldade foi organizar a lista de mensagens dentro da caixa de entrada específica de cada

usuário. A solução para este problema foi fazer uma lista bifurcada simplesmente encadeada, com estas duas estruturas dentro de uma única. Além disso, existe um apontador para a próxima mensagem e outro para a próxima caixa de entrada.

Após o fim da implementação do programa ele atendia à todas as expectativas do simulador do servidor através do arquivo .txt com as instruções dos comandos.

## **REFERÊNCIAS BIBLIOGRÁFICAS**

[1] ZIVIANI, Nivio. **Projetos de Algoritmos: com implementações em Pascal e C**. 2ª ed. São Paulo: Cengage Learning, 2009. 552 p.

[2] [https://wiki.portugal-a-programar.pt/dev\\_geral:c:scanfparastrings](https://wiki.portugal-a-programar.pt/dev_geral:c:scanfparastrings)

## **ANEXOS**

Listagem dos programas:

- F\_tp.c
- F\_tp.h
- Tp\_main.c