

Memorando

De: nome do estudante

Nº de Matrícula:

Disciplina:

Assunto:

Data: ...

1. Introdução

Este relatório apresenta o desenvolvimento de um sistema de gestão de tarefas utilizando a linguagem C. O sistema permite criar, listar e completar tarefas, organizadas por prioridade e data de criação. Foram utilizados conceitos de estruturas de dados (listas ligadas) e manipulação de strings. A estrutura de dados principal é baseada em listas ligadas simples, garantindo eficiência na inserção e remoção de tarefas. O projeto foi implementado com o objetivo de praticar habilidades em manipulação de ponteiros e gestão de memória.

2. Experiências Realizadas

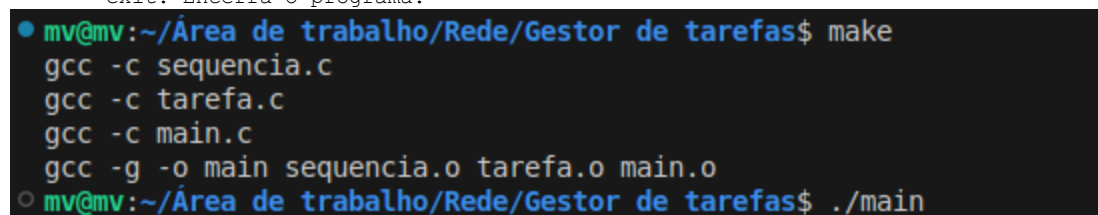
A implementação do projeto seguiu uma abordagem modular, com arquivos separados para as funcionalidades principais: gestão de sequências de caracteres (`sequencia.c` e `sequencia.h`) e gestão de tarefas (`tarefa.c` e `tarefa.h`). O arquivo principal (`main.c`) integra os módulos e realiza o controle da interface de comandos.

Para compilar e executar o programa, utilizou-se um Makefile com os seguintes alvos:

- **main:** Gera o executável principal.
- **sequencia.o:** Compila o módulo de sequências.
- **tarefa.o:** Compila o módulo de tarefas.
- **main.o:** Compila o arquivo principal.
- **clean:** Remove arquivos objeto e o executável.

Comandos Principais

- `new <prioridade> <id>:` Cria uma nova tarefa com prioridade e ID especificados.
- `list <prioridade>:` Lista as tarefas da prioridade especificada.
- `complete <id>:` Marca a tarefa com o ID fornecido como concluída.
- `exit:` Encerra o programa.



```
mv@mv:~/Área de trabalho/Rede/Gestor de tarefas$ make
gcc -c sequencia.c
gcc -c tarefa.c
gcc -c main.c
gcc -g -o main sequencia.o tarefa.o main.o
mv@mv:~/Área de trabalho/Rede/Gestor de tarefas$ ./main
```

Figura 1 - Compilando o programa usando o make

```

mv@mv:~/Área de trabalho/Rede/Gestor de tarefas$ make
gcc -c sequencia.c
gcc -c tarefa.c
gcc -c main.c
gcc -g -o main sequencia.o tarefa.o main.o
mv@mv:~/Área de trabalho/Rede/Gestor de tarefas$ ./main
$:new 5 fazerComprar
$:new 7 dormirTarde
Prioridade invalida. Use valores entre 0 e 5.
$:list 5
Prioridade 5:
ID: fazerComprar | Data: Sun Mar 16 23:29:56 2025

```

Figura 2 - Executando o arquivo main gerado

```

18
19
20 int main() {
21     list_tarefa *lista_tarefas = inicializar_list_tarefa();
22     char *linha = NULL;
23     size_t tamanho = 0;
24
25     printf("$:");
26     while (getline(&linha, &tamanho, stdin) != -1) {
27         char *comando[MAX_ARG + 1];
28         quebrar_comando(linha, comando);
29
30         if (strcmp(comando[0], "new") == 0 && comando[1] && comando[2]) {
31             int prioridade = atoi(comando[1]);
32             if (prioridade >= 0 && prioridade <= 5) {
33                 inserir_tarefa_prioritaria(lista_tarefas, prioridade, comando[2]);
34             } else {
35                 printf("Prioridade invalida. Use valores entre 0 e 5.\n");
36             }
37         } else if (strcmp(comando[0], "list") == 0 && comando[1]) {
38             int prioridade = atoi(comando[1]);
39             mostrar_tarefa_normal(lista_tarefas, prioridade);
40         } else if (strcmp(comando[0], "complete") == 0 && comando[1]) {
41             completar_tarefa(lista_tarefas, comando[1]);
42         } else if (strcmp(comando[0], "exit") == 0) {
43             break;
44         } else {
45             printf("Comando invalido. Use: new <prioridade> <id>, list <prioridade>, complete <id>");
46         }
47
48         printf("$:");
49     }
50 }

```

Figura 3 - Arquivo main.c

```

1 #meu Make para executar o projecto
2 main: sequencia.o tarefa.o main.o
3     gcc -g -o main sequencia.o tarefa.o main.o
4 sequencia.o: sequencia.c sequencia.h
5     gcc -c sequencia.c
6 tarefa.o: tarefa.c tarefa.h
7     gcc -c tarefa.c
8 main.o: main.c sequencia.h tarefa.h
9     gcc -c main.c
10 clean:
11     rm -f *.o main

```

Figura 4 - Makefile

```
1 #include <stdlib.h>
2 #include <stdio.h>
3 #include "tarefa.h"
4 #include "sequencia.h"
5 #include <time.h>
6 list_tarefa *inicializar_list_tarefa(){
7     list_tarefa *lst_t=(list_tarefa*)malloc(sizeof(list_tarefa));
8     lst_t->item=NULL;
9     return lst_t;
10 }
11
12 void inserir_tarefa_prioritaria(list_tarefa *list_t, int prioridade, char *vet_id_tarefa) {
13
14     tarefa *item = (tarefa *)malloc(sizeof(tarefa));
15     item->prioridade = prioridade;
16     item->seq_id = inicializar_sequencia();
17     item->data=time(NULL);
18     priencher_sequencia(item->seq_id, vet_id_tarefa);
19     item->prox = NULL;
20     if (list_t->item == NULL || list_t->item->prioridade < prioridade) {
21         item->prox = list_t->item;
22         list_t->item = item;
23         return;
24     }
25     tarefa *iitem = list_t->item->prox;
26     while (iitem != NULL) {
27         if(equals_seq( item->seq_id,iitem->seq_id)){
28             printf("Tarefa Ja existe no gestor como um ID");
29             free(item);
30             return;
31         }
32         iitem=iitem->prox;
33     }
```

Figura 5 - Arquivo tarefa.c

```
10 void priencher_sequencia(sequencia *seq, char *vet_seq) {
12     if (seq->item == NULL) {
13         sequencia_iitem *ultimo = seq->item;
14         while (ultimo->prox != NULL) {
15             ultimo = ultimo->prox;
16         }
17         for (; i < strlen(vet_seq); i++) {
18             sequencia_iitem *item = (sequencia_iitem *)malloc(sizeof(sequencia_iitem));
19             item->c = vet_seq[i];
20             item->prox = NULL;
21             ultimo->prox = item;
22             ultimo = item;
23         }
24     }
25 }
26
27 void limpar_sequencia(sequencia *seq){
28     sequencia_iitem *item=seq->item;
29     while(item){
30         sequencia_iitem *actual= item;
31         item=item->prox;
32         free(actual);
33     }
34     free(seq);
35 }
36
37 int count_seq(sequencia *seq){
38     int el=0;
39     if(seq!=NULL){
40         sequencia_iitem *item=seq->item;
41         while(item!=NULL){
42             el++;
43         }
44     }
45 }
```

Figura 6 - Arquivo sequencia.c

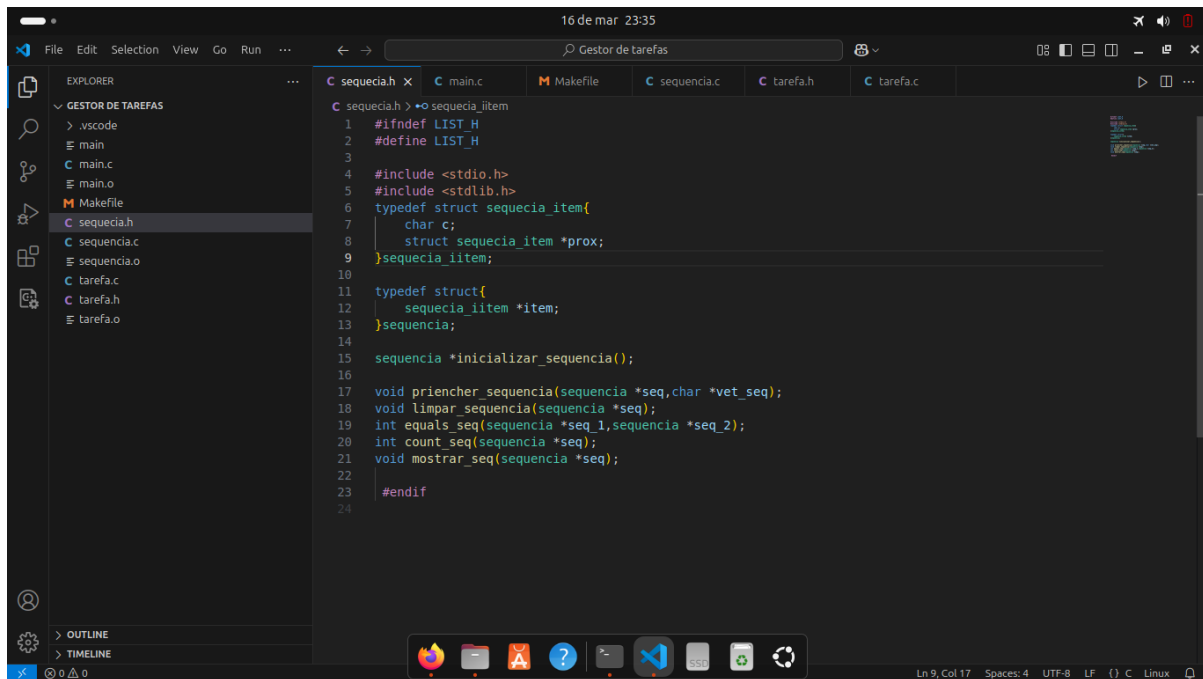


Figura 7 - Arquivo sequencia.h

3. Desafios

Um dos principais desafios foi a gestão de memória dinâmica, especialmente para manipular corretamente as listas ligadas e garantir que não houvesse vazamentos de memória. A função de completar tarefas exigiu um tratamento cuidadoso para a remoção eficiente e segura dos nós da lista. Também houve a necessidade de garantir que os IDs fossem únicos para evitar duplicidade de tarefas.

Outro grande desafio foi a utilização de uma sequência (lista ligada de caracteres) no lugar de um vetor de caracteres para armazenar os identificadores das tarefas. Isso permitiu que o número de caracteres em cada ID fosse ilimitado, garantindo maior flexibilidade ao gestor de tarefas. A implementação da sequência está detalhada nos arquivos sequencia.c e sequencia.h.

4. Referências Bibliográficas

- Documentação oficial da linguagem C.
- Artigos sobre listas ligadas e gestão de memória em C.
- Exemplos de Makefile para projetos em C.
- Arquivo **cpd-lab1** do professor João Costa

5. Repositório GitHub

Coloque aqui o link do repositório e envie o convite de colaborador para o utilizador GitHub **joaojdacosta**.

