UNIVERSIDADE FEDERAL DE PELOTAS
CENTRO DE DESENVOLVIMENTO TECNOLÓGICO

Relatório

Trabalho Programação de Computadores

Aluno(a): Lucio Vagner Carvalho Souza

Aluno(a): Carlos Henrique Leite Bianchin

Turma: M5

Título do Programa: Sistema de gerenciamento de tarefas

1. Objetivo do Programa

O programa tem como objetivo facilitar o gerenciamento de tarefas com um programa via terminal, sejam diárias ou tarefas necessárias de serem feitas no futuro. O desafio foi implementar uma lista encadeada que permita ao usuário criar, listar, editar e excluir tarefas, além de salvar os dados e carregar as informações em um arquivo binário.

2. Descrição Geral do Funcionamento

O programa carrega tarefas salvas em um arquivo “save.bin”, permite ao usuário gerenciar as tarefas por meio de um menu interativo utilizando lista encadeada dinâmica e, podendo criar uma nova tarefa, apagar antigas, marcar como concluído e ao finalizar, salva os dados novamente em arquivo binário.

3. Linguagem e Estruturas Utilizadas

Liste as principais estruturas de programação utilizadas no código:

- [X] printf / scanf
- [X] estruturas de seleção if / else
- [X] estruturas de seleção switch-case
- [X] estrutura de repetição while

- [ ] estrutura de repetição do-while
- [ ] estrutura de repetição for
- [ ] estruturas de repetição aninhadas;
- [X] contadores
- [X] acumuladores
- [X] *flags*
- [X] operadores relacionais
- [X] operadores lógicos
- [X] funções - por valor
- [X] funções - por referência
- [X] vetores
- [ ] matrizes
- [X] structs;
- [X] alocação dinâmica;
- [X] arquivos;
- [X] listas encadeadas;
- [X] Outros (detalhar): Uso de cores, vetores constantes, ponteiros, cabeçalhos, arquivos binários, portabilidade, icone para .exe, horário do sistema;

4. Explicação do Código (principais partes)

Escolha 2 trechos importantes do código e explique com suas palavras o que cada um faz.

```
void nova_tarefa(No** lista, int* contador_id){
      //usando a biblioteca time.h para atribuir a data da criação das tarefas
    time_t t = time(NULL);
    struct tm *tm = localtime(&t);
    limpar_tela();

    printf(CYAN "======== Criando nova tarefa ========" RESET "\n");

    No* novo_no = (No*) malloc(sizeof(No));
    if(novo_no == NULL){
        printf("Erro ao alocar a memória.\n");
        return;
    }
    novo_no->info.id = (*contador_id);
```

```
    (*contador_id)++;

    getchar();
    printf("Digite o Titulo: ");
        fgets(novo_no->info.titulo, sizeof(novo_no->info.titulo), stdin);



      printf("Digite a prioridade: \n(1) Urgente.\n(2) Importante.\n(3)
Intermediário.\n(4) Não importante.\n");
        if(scanf("%d", &novo_no->info.prioridade) == 0){
            erro_opcao();
            free(novo_no);
            return;
        }else {


                              while (novo_no->info.prioridade < 1 ||
novo_no->info.prioridade > 4){
            erro_opcao();
               printf(RED "Essa prioridade não existe."RESET "\nPor favor
digite uma das 4 prioridades.\n");
            getchar();
            free(novo_no);
            return;
        }
        }

    //valores padroes
    novo_no->info.concluido = 0;
    novo_no->info.prazo.dia = tm->tm_mday;
    novo_no->info.prazo.mes = tm->tm_mon + 1;
    novo_no->info.prazo.ano = tm->tm_year + 1900;

    //o "prox" aponta para onde a lista original apontava
    novo_no->prox = *lista;
    //a lista aponta para o novo no que criamos.
    *lista = novo_no;

     printf(GREEN "\n>> Tarefa criada com sucesso! (ID: %d)" RESET "\n",
novo_no->info.id);
    esperar(2000);
}
```

Explicação:

Esta função é responsável pela alocação dinâmica de um novo nó na memória usando malloc e pelo preenchimento dos dados da tarefa. A inserção na lista encadeada ocorre no início: o ponteiro prox do novo nó aponta para a cabeça atual da lista, e, em seguida, o ponteiro principal da lista é atualizado para o novo nó.

Trecho 2:

```
void salvar(No *lista){
    FILE *fp = fopen("save.bin", "wb");
    No *salvar = lista;


    if(fp == NULL){
        printf(RED "ERRO! IMPOSSÍVEL SALVAR." RESET "\n");
        esperar(1000);
        return;
     }

    while (salvar != NULL){
        fwrite(&salvar->info, sizeof(salvar->info), 1, fp);
        salvar = salvar->prox;
    }

    fclose(fp);
    while(lista != NULL){
        salvar = lista;
        lista = lista->prox;
        free(salvar);
    }
}
```

Explicação:

Essa função serve para salvar cada estrutura Tarefa em um arquivo binário atráves da função fwrite enquanto a lista auxiliar estiver com algum conteúdo, sempre sobrescrevendo o arquivo original por conta do “wb” e no final liberando a lista até ficar vazia.

5. Dificuldades Encontradas

As principais dificuldades foram quanto à lógica das funções com lista encadeada, principalmente na relação com os ponteiros e como se comporta na memória fora a criação da função de remover.

6. Conclusão

Nesse trabalho aprendemos a utilizar o serviço GitHub para cada um programar de seu computador, como fazer portabilidade, cabeçalhos, cores, além de aperfeiçoar nossa lógica com listas encadeadas e arquivos.