SERVIÇO PÚBLICO FEDERAL · MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE VIÇOSA · UFV
CAMPUS FLORESTAL

# Trabalho 1 - AEDS 1

## Sistema de gerenciamento de processos utilizando lista de cursores

LUIZ CÉSAR GALVÃO LIMA [EF04216]
IURY MARTINS PEREIRA [EF04671]
JOÃO VITOR GONÇALVES VIEIRA [EF04212]

Florestal - MG
2021

# Sumário

## 1. Introdução

Este trabalho tem como foco criar um programa que gerencia uma lista de processos, simulando um sistema operacional. O método de organização de tal lista será por um vetor dinamicamente alocado e duplamente encadeadas por cursores, foi usado Tipo Abstrato de dados(TAD) que é um modelo matemático que encapsula modelos de dados e conjuntos de procedimentos que trabalham com exclusividades sobre os dados encapsulados, tornando se mais fácil a compreensão de algoritmos assim se tornando fundamental em projetos de software que usa a modelagem prévia de dados.

## 2. Organização

Inicialmente definimos onde seria o local para armazenar/manipular o trabalho, concordamos em usar o Github, pela facilidade de acesso, popularidade e registro de progresso. Posteriormente, após a monitoria de destrinchamento do trabalho, dividimos as tarefas pelo grupo e caso houvesse alguma dificuldade, um ajudaria o outro.

O repositório foi organizado para que todos os integrantes pudessem desenvolver seus papéis de forma clara e objetiva, o aluno João ficou responsável pelo arquivo main, organização da leitura e escrita dos arquivos e o menu interativo, o Iury ficou responsável pela lista de processos e o Luiz ficou responsável em pesquisar soluções para problemas encontrados e correções de erros no código.

Na Figura 1 é possível visualizar que_foram criadas duas pastas a primeira /Libs onde ficaram os arquivos cabeçalho do projeto e por último a pasta /Sources onde continha as implementações de funções que foram declaradas nos arquivo .h contidos dentro da pasta /Libs

.

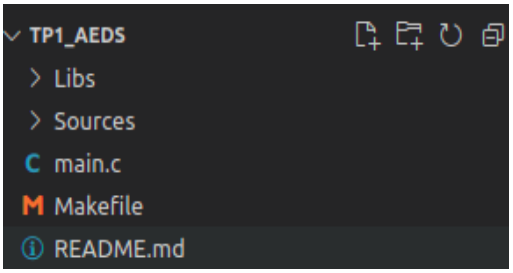


**Figura 1 - Repositório do projeto.**

Para executar o projeto, foi utilizado um arquivo Makefile com os comandos utilizados para compilar e executar os códigos. Por fim, foi criado um README para que auxiliasse os usuários de Linux e Mac a como executar o código usando os comandos declarados dentro do makefile.

## 3. Desenvolvimento

### 3.1 Cronologia

6 de Dezembro - Segunda

Local de armazenamento do trabalho: Github.

7 de Dezembro - Terça

Exemplo de trabalho usado como referência de criação de pastas e organização: https://github.com/Globson/TP-Aeroporto-AEDS

8 de Dezembro - Quarta

Monitoria de explicação sobre o trabalho

10 de Dezembro - Sexta

Início de desenvolvimento das funções de tempo.

Início de desenvolvimento das funções de arquivo.

11 de Dezembro - Sábado

Início de desenvolvimento do TAD lista de processos.

13 de Dezembro - Segunda

Correções nas funções de arquivo.

15 de Dezembro - Quarta

Correções nas funções de tempo.

16 de Dezembro - Quinta

Terminado as funções de ordenação.

Início do desenvolvimento da função de remoção de primeiro elemento

17 de Dezembro - Sexta

Menu interativo terminado;

Impressão sendo implementado.

18 de Dezembro - Sábado

Testes de desempenho.

### 3.2 Explicação Lista de processos

A função organiza_vetor é chamada apenas uma vez, logo após criar o vetor da estrutura nomeada de Vetor_Celula, ela é responsável por colocar -1 no campo célula_anterior de todas as células (chamarei a partir de agora cada posição do vetor de célula), bem como a posição da próxima célula no campo célula_proxima, estes dois campos são do tipo inteiro.

A função preenche_vetor como seu nome indica é responsável por inserir de forma ordenada e crescente no vetor o processo. O processo é uma estrutura que contém um campo do tipo int que guarda o pid do qual é feita a ordenação, um que guarda prioridade que varia de 1 a 5 e um campo que guarda a hora que foi gerado o processo. O processo é gerado ao chamar a função implementa_processo que por sua vez chama outras funções que gera o pid e a prioridade além da função que guarda a hora que foi gerado o processo.

As informações contidas neste parágrafo serão essenciais para a explicação de como irá funcionar a inserção e remoção de elementos do vetor. Dentro do TAD lista_Processo a uma estrutura nomeada Lista_Processo que guarda algumas informações do vetor, sendo que essas informações são: a posição do menor pid no vetor no campo posição_menor_pid, a posição do maior pid no vetor no campo posição_maior_pid, a posição da primeira célula disponível no campo célula_disponivel, a quantidade de células ocupadas no campo quantidade_celulas_ocupadas e o total de células do vetor no campo total_celulas sendo todos esses campos do tipo int por serem números inteiros além de contar com um ponteiro do tipo Vetor_Celula denominado célula.

Dentro da função preenche_vetor são implementados alguns if, else e else if com o objetivo de ordenar o vetor através do pid gerado. Bem o 1º if é executado quando o vetor não tem nenhuma de suas células ocupadas, para que tanto a posição_menor_pid quanto a posição_maior_pid receba esta primeira posição, para que ao longo das inserções seja possível determinar realmente a posição tanto do menor quanto do maior pid. Dessa forma nas demais vezes que o programa rodar a cada novo pid gerado verifica-se se este é menor que o que está em posição_menor_pid através de um if, caso seja verdade a posição_menor_pid vai

receber a posição do novo pid gerado tornando esta a posição do menor pid. A mesma verificação é feita para a posição do maior pid, caso o novo pid gerado esteja entre o menor e o maior pid o programa executa um while que começa pelo menor pid que varre a lista até achar um pid maior do que aquele que foi gerado, ‘inserindo’ o pid gerado atrás desse pid maior, o while para após achar um pid maior do que o pid gerado. Esse programa de inserção está dentro de um for que é executado N vezes, N este que é passado pelo usuário ou lido no arquivo.

O programa que retira os menores pids nomeado de retirar_menor_pid possui um for que é executado N vezes, passado pelo usuário ou lido no arquivo. A cada execução do for, a célula_proxima da posição do menor pid se torna o menor pid, e a posição que o antigo menor pid ocupava se torna uma célula vazia, que pode ser usada para armazenar um novo processo. Após criação da lista de processos foi feita criação das funções ler e escreve_arquivo_tempo onde que a função ler seria responsável em obter os valores que são de importância para o usuário, e a função escreve ficou responsável por escrever no arquivo saida o numero de teste informado pelo usuario e tempo de execução para a leitura e escrita do programa.

## 4. Resultados

O programa consegue funcionar, porém o uso de cursores não é a maneira mais eficiente. Inserção ordenada, quando superior a 100000 processos, exige muito da ordenação, o que prolonga a inserção e adia o processo de impressão de arquivo de saída e de outras possíveis ações desejadas pelo usuário.

## 5. Conclusão

Concluímos que o método de ordenação por cursores, mesmo sendo funcional, não é um dos mais eficientes, tendo um gasto demasiado de tempo quando há mais de 100000 inserções, assim se tornando inviável para ser usado em empresas onde o tempo é de suma importância para a produtividade.

## 6. Referências

[1] Github. Disponível em <https://github.com/joaoVGvieira/TP1_AEDS> Último acesso em: 18 de Dezembro de 2021.