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

João Victor Graciano Belfort de Andrade
Mateus Henrique Vieira Figueiredo
Vitor Ribeiro Lacerda

Florestal - MG
2021

Sumário

1. Introdução

Esta documentação se refere ao projeto feito no Trabalho Prático I, da disciplina ALGORITMOS E ESTRUTURA DE DADOS I, de código CCF 211. No trabalho, foi sugerido a implementação de um Tipo Abstrato de Dados (TAD) Lista Linear, para que um sistema operacional faça gerência de processos ativos. O projeto realizado descrito neste documento está disponível na página do GitHub: Lista-Linear-OS.

Algumas bibliotecas externas foram utilizadas para o programa, elas são “time.h”, “string.h”. “time.h” foi utilizada para pegar o horário da máquina e utilizá-lo na TAD “processo”, e a biblioteca “string.h” foi utilizada para: transformar números inteiros em strings, strings em inteiros e concatenar strings (…). O programa foi dividido em 2 pastas, sendo uma para armazenamento de testes aleatórios e outra para o algoritmo pedido no trabalho.

2. Organização

Na pasta principal do arquivo podem ser encontrados 4 arquivos, cada um destes com os seguintes nomes: Arquivos (Pasta de arquivos), src (Pasta de arquivos), README.md, “.gitignore.txt”. Os arquivos mencionados do tipo “pasta de arquivos” ainda contém outros, em “Arquivos”: “.idea” (pasta de arquivos), “cmake-build-debug” (pasta de arquivos), “teste1.txt”, “teste2.txt”, “teste3.txt”, teste4.txt, teste5.txt, teste6.txt, teste7.txt, “teste8.txt”, “teste9.txt”, “teste10.txt”, “teste11.txt”, “teste12.txt”. Já src tem-se: “.idea” (pasta de arquivos), “cmake-build-debug” (pasta de arquivos), “CMakeList.txt”, “main.c”, “lista.h”, “lista.c”, “processo.c”, “processo.h”.

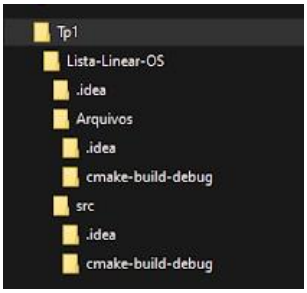

Figura 1- Repositório do projeto

Sendo assim, a pasta principal do projeto contém arquivos referentes ao repositório do github, e as duas pastas contendo os códigos e arquivos referentes ao projeto. Em “Arquivos” tem se os arquivos criados aleatoriamente pelo código em “main.c”, variando de "teste1.txt" até “teste 12.txt”. Já a pasta “Codes” contém os códigos tanto das TAD’s como do código principal do trabalho, além de pastas adicionadas pela ide CLion na compilação do código.

3. Desenvolvimento

3.1 “processo.c” e “processo.h”

Nestes arquivos foi descrita a “TAD TProcesso”, essa estrutura conta com um ID de processo "PID" (inteiro gerado aleatoriamente), uma prioridade (inteiro entre 1 e 5 gerado aleatoriamente) e um tempo de criação (retirado do computador do usuário, para tal funcionalidade a biblioteca "time.h" está sendo utilizada). Foram criadas as seguintes funções:

“inicializa_processo”: Cria o processo com números aleatórios definidos pela “rand()”.

“inicializa_tempo”: Retorna o tempo obtido do sistema operacional do usuário no formato padrão da biblioteca “time.h”.

“get_PID” e“set_PID”: Retorna e subscreve respectivamente o PID da estrutura TProcesso *processo.

“get_prioridade” e “set_prioridade”: Retorna e subscreve respectivamente a prioridade da estrutura TProcesso *processo.

“get_hora”: Retorna hora no horário formatado como uma string (“www MMM dd hh:mm:ss yyy”, sendo “w” semana, “M” mês, “d” dia, “h” horas, “m” minutos, “s” segundos e “y” anos) da função “asctime()”, oferecida pela biblioteca “time.h”.

3.2 “lista.c” e “lista.h”

Nestes arquivos foi descrita a “TAD TLista” *, uma estrutura abstrata para uma lista de processos ao qual simula processo de uma máquina. A estrutura lista é composta pela estrutura célula onde que abriga um processo (tipo de dado que simula um processo), além de possuir dois "Cursor" que são inteiros usados para interligar Células em que são dispostas em um array na lista.

Além da estrutura lista possuir um array de "Celula", a mesma também possui um tamanho (tamanho da lista que será usada pelo usuário), dois "Cursor" que são inteiros usados para apontar para a primeira e última célula na lista, um inteiro que demonstra a célula que está atualmente disponível para receber um processo e um inteiro que representa o número de células sendo ocupadas no array. Foram criadas as seguintes funções:

“inicializa_lista”: Essa função é usada para inicializar uma lista. Primeiramente ocorre alocações na memória da estrutura "TLista" e do array contido nessa estrutura. Após as alocações são inicializadas "Celulas" com cursores predefinidos aos quais apontam para uma próxima célula e uma célula anterior, esses cursores serão posições no array. Então essas "Celulas" são colocadas no array. Há também a inicialização de valores iniciais para as variáveis da estrutura "TLista". Retorna o ponteiro da lista criada.

“destroi_lista”: Procura por processos não nulos na lista, afim de libera-los, então libera o array alocado

“insere_na_lista”: Função usada para adicionar um elemento na lista. Primeiramente tentamos achar uma célula disponível (posição no vetor) para adicionar um processo que é gerado pelo método "inicializa_processo()". Como a lista deve se manter ordenada, os elementos já são colocados nas posições corretas, sendo as mesmas definidas pelo PID do processo de cada célula.

Em outras palavras, os cursores das células serão usados para ordenar as mesmas na lista de forma que o PID esteja em uma ordem crescente.

“remove_da_lista”: Essa função é usada para remover um elemento na primeira posição da lista. Ao remover um elemento a célula que o elemento anteriormente ocupava será definida como nula e essa célula passará a ser uma "célula disponível" a qual irá apontar para uma próxima "célula disponível".

“imprime_conteudo”: Função utilizada para obter o número atual de células sendo ocupadas na lista.

“get_numCelOcupados”: Função utilizada para percorrer a lista com intuito de obter a primeira "célula disponível", identificando uma célula como disponível quando o seu processo é nulo. Caso não exista célula disponível "-1" será retornado.

3.3 “teste.c” e “teste.h”

Neste arquivo foram criadas as funções abstratas relacionadas à criação de arquivos de teste. Foram criadas as seguintes funções:

“gera_arquivo”: Este método é utilizado para gerar um arquivo customizado pelo usuário. Ele irá receber informações para incluir dentro de um arquivo com nome "testeNN" onde "NN" é um número especificado pelo usuário. Como explicado anteriormente o número tem que seguir o formato "NN", ou seja, só números de 2 algarismos são válidos.

“mostra_opcoes”: Esse método é usado para descrever as opções que o usuário pode escolher, por meio do console.

“ler_arquivo”: O método seguinte foi criado com o intuito de ler a data de um arquivo em que o nome será passado pelo usuário. Esse método lerá um arquivo seguindo um formato escrito. O mesmo irá criar uma lista onde irá realizar operações sobre. Tem como retorno o ponteiro da lista onde foram realizadas as operações especificadas no arquivo.

3.4 “main.c”

Neste arquivo foi implementado o código principal do programa que utiliza dos demais executáveis e headers, para criar uma interface interativa com o usuário, mostrando as opções do que se pode fazer com o código e gerando mensagens de erro caso valores inválidos sejam inseridos pelo usuário. Foram criadas as seguintes funções:

“mostra_opcoes”: Esse método é usado para descrever as opções que o usuário pode escolher, por meio do console.
“limpar_spdin”: Função usada para limpar o "stdin" para evitar problemas com "scanf()" dentro de loops.

“inicializa_relogio”: Função usada para iniciar o clock.

“finaliza_relogio”: Função usada para finalizar o clock e calcular o tempo total.

4. Resultados

Devido a interface interativa, foi criado um menu em que o usuário poderia escolher as opções previamente definidas no programa. O menu está sendo demonstrado abaixo:

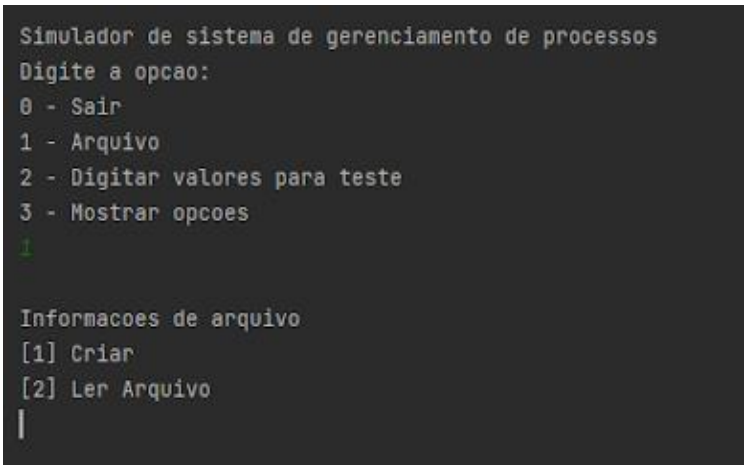

Figura 2 – Menu inicial do programa.

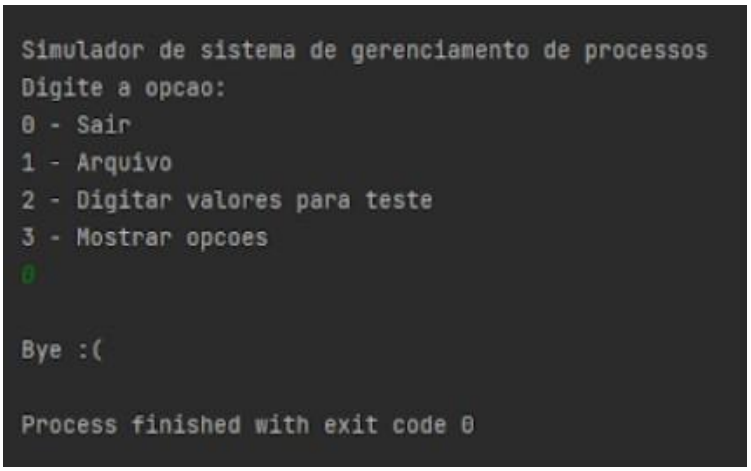

Figura 3 - Opção de Sair.

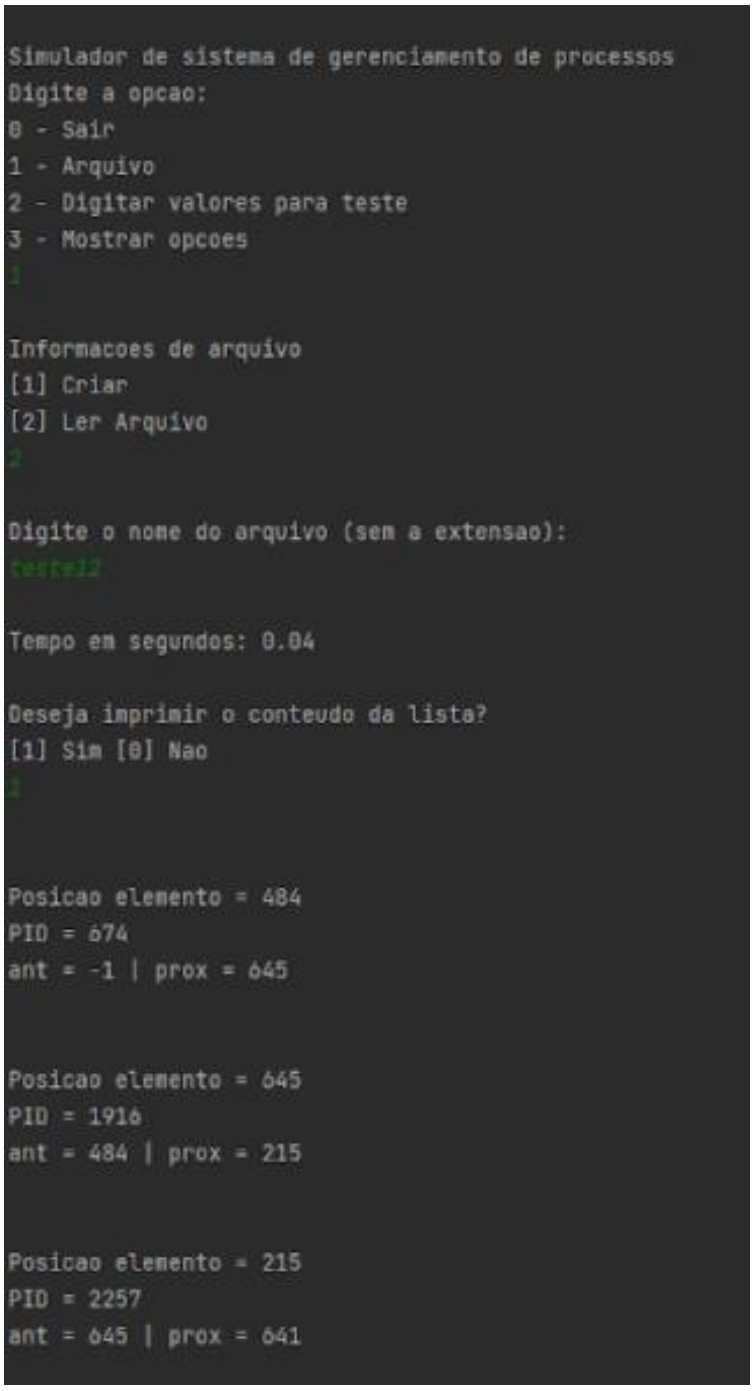

Figura 4 – Lendo Arquivo.

```
Simulador de sistema de gerenciamento de processos
Digite a opcao:
0 - Sair
1 - Arquivo
2 - Digitar valores para teste
3 - Mostrar opcoes
1

Informacoes de arquivo
[1] Criar
[2] Ler Arquivo
1

Digite o numero do arquivo que deseja criar:
12

Digite o tamanho da lista para o arquivo:
100000

Digite a quantidade de operacoes:
44

Arquivo com nome 'teste12.txt' gerado!
```

Figura 5 e 6 – Criando arquivo e o arquivo de Output

```
Simulador de sistema de gerenciamento de processos
Digite a opcao:
0 - Sair
1 - Arquivo
2 - Digitar valores para teste
3 - Mostrar opcoes
3

0 - Sair
1 - Arquivo
2 - Digitar valores para teste
3 - Mostrar opcoes


Simulador de sistema de gerenciamento de processos
Digite a opcao:
0 - Sair
1 - Arquivo
2 - Digitar valores para teste
3 - Mostrar opcoes
```

Figura 7 – Mostrando opções

5. Conclusão

De forma geral, foram implementados TADs e códigos, de forma organizada, comentada e de fácil entendimento, para o problema solicitado pelo trabalho, Foi decidida uma abordagem mais interativa com o usuário, que ao fim foi uma ótima opção abstraindo o código, e o deixando mais fácil de ser visualizado e compreendido pelo usuário. Os resultados saíram assim como o esperado, sendo que algumas implementações feitas, apesar de deixarem o código mais extenso, facilitam seu uso e entendimento, sendo assim o trabalho, de forma geral atende as exigências passadas pela Professor do curso de AEDES-2021/2

6. Referências

Para versionar o projeto foi utilizado o Github [1].

[1] Github. Disponível em: <https://github.com/BelfortJoao/Lista-Linear-OS> Último acesso em: 20 de dezembro de 2021.