SIN351 – Sistemas Operacionais (Período Especial Remoto – PER - 2020-4)

Trabalho Prático II: Implementando um Gerenciador de Memória
Data de Entrega: 10/12/2020 até 23:59:59 (BRT)
Entrega: Submeter via *Microsoft Teams* na Tarefa correspondente.
Projeto deve ser implementado na Linguagem C, exclusivamente pelo aluno (dupla ou trio).

1. Objetivos

- Introduzir os conceitos de Gerenciamento de Memória;
- Familiarizar o estudante com as técnicas de subistituição de Páginas dos Sistemas Operacionais Modernos;
- Aprimorar as capacidades de programação em Linguagem C.

2. Aspectos Gerais

Neste projeto o aluno deverá implementar em linguagem C algum algoritmo de subistituição de páginas (Conteúdo de Memória Virtual). Deverá ser comparado o desempenho do Algoritmo implementado com o *Random* (Aleatório) disponibilizado pelo professor. Deverá ser produzido um relatório técnico explicando as principais funcionalidades do código e do algoritmo de subistituição de páginas escolhido. O código deverá ser comentado para fins de entendimento (correção do professor). Ao final da execução do programa, deverá ser exibido a quantidade de falta de páginas para cada algoritmo: o desenvolvido pelo aluno e o *Random*. Será fornecido materiais suplementares pelo professor.

3. Recursos Suplementares

O professor disponibilizará (após a aula de apresentação do projeto 10/11/2020) um programa implementado em linguagem C contendo a implementação básica do gerenciamento de memória, incluindo: estrutura de dados, estruturas de repetição, seleção, função principal e outros. Além disso, o professor disponibilizará um arquivo anomaly.dat (anomalias) que representa a estrutura da memória, isto é, quantidade de páginas virtuais e molduras de páginas. Por fim, cada linha representa um tipo de referência à memória (*write*) e um determinado endereço. Abaixo está a estrutura do arquivo de anomalias:

```
10 3
3 w
2 w
1 w
0 w
3 w
2 w
4 w
3 w
2 w
1 w
0 w
4 w
```

Figura 1 - Estrutura da Memória e Referências.

Como se vê na Figura 1, a linha 1 refere-se à estrutura da memória, 10 representa a quantidade de páginas virtuais e 3 refere-se à quantidade de molduras de página (páginas físicas). As demais linhas (2-13) referem-se aos acessos *write* (escrita) à página correspondente.

4. Algoritmos possíveis de implementação

O código base fornecido pelo professor permite que o aluno escolha e implemente dentro dessa estrutura os seguintes algoritmos de subistituição de páginas:

- *FIFO*;
- *FIFO + Bit R* (Segunda Chance);
- *NRU*;
- *Aging* (Envelhecimento);
- *Random* (já implementado)

No entanto, é possível utilizar a estrutura de código fornecida pelo professor para implementar outras políticas de subistituição de páginas, como *Least Recently Used (LRU)*.

5. Como será avaliado

A submissão consiste em dois itens dentro de um arquivo Compactado (*.zip .tar .rar*):

1. Código fonte (<file>.c) – não enviar executável – **70%**;
    a. Indentação, Nomes de Variáveis, Modularidade, Comentários (úteis);
    b. O arquivo deverá conter o código fonte da política (algoritmo) de substituição de páginas escolhida e implementada pelo aluno;
    c. Os testes avaliativos serão executados em Ubuntu 16 *LTS*.
    d. O Código fonte deverá compilar, executar e produzir saídas corretas
2. Um README contendo a documentação sobre o algoritmo de subistituição de páginas escolhido e o seu código.
    a. README: Nome e Matrícula, exemplos de como utilizar, *bugs* conhecidos ou problemas **10%**.
    b. Produzir documento (PDF) que descreva o mecanismo de gerenciamento de memória implementado e uma tabela comparativa considerando 10 execuções distintas abordando a quantidade de falta de páginas do algoritmo de substituição de Páginas *Random* (Aleatório) e o escolhido pelo aluno. Alternativamente, caso o aluno deseje, ele poderá incluir esse PDF no Git – **15%.**
    c. Versionamento do projeto em alguma plataforma especializada (Git, *Gitlab* ou outras) - **5%**.

6. Como o professor irá compilar e executar o código enviado pelos alunos

- `$ gcc -Wall vmm.c -o vmm` (Compilação do Gerenciador de Memória).

- `$ ./vmm random 10 < anomaly.dat` (Execução do Gerenciador de Memória passando como parâmetro o arquivo dos acessos)

7. Dicas

- Vocês precisam compreender inicialmente a estrutura do código base (*base-code* disponibilizado pelo professor).
- Após entender o código, vocês deverão implementar em formato de nova função a política de subistituição de páginas.
- Seja o testador do seu código – não seja gentil com as entradas. Usualmente os usuários não são.
- Utilize programas de versionamento para o código (*Git*, *Gitlab*, *Gerrit*, *Azure DevOps*, *Bitbucket* e outros).