UNIVERSIDADE PRESBITERIANA MACKENZIE

SISTEMAS OPERACIONAIS

DIOGO OLIVEIRA UEMA

ARTHUR SILVA SANTANA

MURILO HENRIQUE SAKAMOTO

DANIEL MONTEIRO MALACARNE

SIMULADOR DE PAGINAÇÃO DE MEMÓRIA

SÃO PAULO

2025

DIOGO OLIVEIRA UEMA

ARTHUR SILVA SANTANA

MURILO HENRIQUE SAKAMOTO

DANIEL MONTEIRO MALACARNE

SIMULADOR DE PAGINAÇÃO DE MEMÓRIA

Trabalho apresentando a Universidade Presbiteriana Mackenzie

Orientador: Lucas Figueiredo

SÃO PAULO

2025

SUMÁRIO

1 INTRODUÇÃO

Este projeto implementa um simulador de sistemas de paginação em memória virtual, um componente essencial dos sistemas operacionais modernos. A paginação permite que programas utilizem mais memória do que fisicamente disponível através de técnicas de substituição de páginas.

Objetivos:

- Implementar algoritmos clássicos de substituição de páginas (FIFO e LRU)
- Desenvolver uma interface para simulação e análise comparativa
- Avaliar o desempenho dos algoritmos em diferentes cenários
- Demonstrar os princípios de tradução de endereços virtuais para físicos

2 DESENVOLVIMENTO

2.1 ESTRUTURA DE DADOS UTILIZADAS

A implementação foi feita em C, com foco em estruturas simples e eficientes. Usamos vetores e algumas structs para representar a memória física, as páginas dos processos e os acessos realizados. A struct Pagina, por exemplo, guarda informações básicas como número da página e tempo do último acesso, o que foi essencial para o algoritmo LRU.

```c
#ifndef ESTRUTURAS_H
#define ESTRUTURAS_H

typedef struct
{
    int presente;      // 1 se a página está na memória , 0 caso contrário
    int frame;         // Número do frame onde a página está alocada (-1 se não alocada)
    int modificada;    // 1 se a página foi modificada , 0 caso contrário
    int referenciada;  // 1 se a página foi referenciada recentemente , 0 caso contrário
    int tempo_carga;   // Instante em que a página foi carregada na memória
    int ultimo_acesso; // Instante do último acesso à página
} Pagina;

typedef struct
{
    int pid;               // Identificador do processo
    int tamanho;           // Tamanho do processo em bytes
    int num_paginas;       // Número de páginas do processo
    Pagina *tabela_paginas; // Tabela de páginas do processo
} Processo;

typedef struct
{
    int num_frames; // Número total de frames na memória física
    int *frames;    // Array de frames (cada elemento contém o pid e a página)
                    // Ex: frames[i] = (pid << 16) | num_pagina
    int *tempo_carga; // Tempo em que cada frame foi carregado (para FIFO)
} MemoriaFisica;

typedef struct
{
    int tempo_atual;           // Contador de tempo da simulação
    int tamanho_pagina;        // Tamanho da página em bytes
    int tamanho_memoria_fisica; // Tamanho da memória física em bytes
    int num_processos;         // Número de processos na simulação
    Processo *processos;       // Array de processos
    MemoriaFisica memoria;     // Memória física
    // Estatísticas
    int total_acessos; // Total de acessos à memória
    int page_faults;   // Total de page faults ocorridos
    // Algoritmo de substituição atual
    int algoritmo; // 0=FIFO , 1=LRU
} Simulador;

#endif
```

2.2 ALGORITMOS IMPLEMENTADOS

a) FIFO (First-In, First-Out):

- Substitui a página que está há mais tempo na memória
- Utiliza o campo tempo_carga para determinar a página mais antiga
- Implementado em traduzEnderecoFIFO()

b) LRU (Least Recently Used):

- Substitui a página não utilizada há mais tempo
- Utiliza o campo ultimo_acesso para rastrear acessos
- Implementado em traduzEnderecoLRU()

2.3 DECISÕES DO PROJETO

Durante o desenvolvimento do simulador, algumas decisões foram tomadas para tornar a ferramenta mais didática, flexível e próxima do funcionamento real de um sistema operacional. Essas decisões envolveram aspectos visuais, de estrutura de dados, interação com o usuário e suporte a múltiplos processos. Abaixo estão os principais pontos:

Representação Visual:

Para facilitar a análise dos resultados, foi criada a função imprimeEstadoMemoria(), que mostra o estado atual da memória física de forma visual. Cada quadro da memória é exibido no formato P<pid>-<página>, por exemplo, P1-3 representa a página 3 do processo 1. Essa visualização permite entender rapidamente quais páginas estão carregadas e a quais processos elas pertencem.

Gerenciamento de Memória:

As estruturas de dados do simulador foram implementadas com alocação dinâmica, permitindo ajustar o tamanho da memória e das tabelas de páginas de acordo com os parâmetros definidos pelo usuário. Sempre que um novo teste é iniciado com parâmetros diferentes (como tamanho de página ou quantidade de quadros), todas as estruturas são reinicializadas, garantindo a consistência dos dados e simulações independentes.

Interface de Usuário:

A interface é baseada em menus interativos no terminal, onde o usuário pode configurar os principais parâmetros da simulação, como o algoritmo a ser utilizado (FIFO ou LRU), o tamanho da página e da memória física. Além disso:

- Foi incluída uma sequência de teste pré-definida para facilitar demonstrações e testes rápidos.
- Ao final de cada simulação, o programa gera um relatório estatístico automático, exibindo informações como número total de acessos, page faults e taxa de acerto.

Abstração de Processos:

O simulador oferece suporte a múltiplos processos simultâneos, cada um com sua própria tabela de páginas. Isso simula o ambiente real de um sistema operacional multitarefa. Para isso, foi implementado um mecanismo de tradução de endereços por processo, ou seja, cada acesso à memória leva em consideração o processo que está solicitando o acesso, garantindo isolamento e controle correto das páginas.

2.4 LIMITAÇÕES DE IMPLEMENTAÇÃO

Apesar de cumprir os objetivos propostos, o simulador apresenta algumas limitações importantes que restringem sua fidelidade em relação a um sistema real de gerenciamento de memória. Essas limitações foram identificadas ao longo do desenvolvimento e testes:

Escopo de Processos:

Atualmente, o simulador suporta apenas um único processo por vez, o que limita a simulação de um ambiente multitarefa completo. A criação dinâmica de processos também não foi implementada, ou seja, o número e os dados dos processos são fixos e definidos no início da simulação.

Algoritmos:

A simulação contempla apenas dois algoritmos de substituição de páginas: FIFO e LRU. Embora esses dois algoritmos sejam amplamente utilizados para fins educacionais, a ausência de outros algoritmos como Clock, NRU (Not Recently Used) ou LFU (Least Frequently Used) reduz a abrangência da análise comparativa.

Memória Virtual:

O simulador trabalha apenas com a memória física e não possui implementação de área de swap ou disco, que seria usada em um sistema real para armazenar páginas que foram removidas da RAM. Além disso, não há tratamento de páginas sujas (dirty pages), ou seja, o simulador não distingue entre páginas modificadas e não modificadas durante a substituição.

Interface:

A interface de entrada ainda é simples e baseada em uma sequência fixa de acessos. Não há um mecanismo para geração automática de padrões de acesso, nem para carregar acessos aleatórios ou com variação temporal, o que limita a flexibilidade dos testes e a simulação de diferentes cargas de trabalho.

3 ANÁLISE COMPARATIVA DOS ALGORITMOS

3.1 COMPARAÇÃO ENTRE OS ALGORITMOS IMPLEMENTADOS

- Configuração:
    - Tamanho de página: 4KB
    - Memória física: 12KB (3 frames)
    - Processo com 8 páginas virtuais
- Sequência de acesso:
  [0, 4096, 8192, 0, 12288, 16384, 0, 4096, 8192, 12288, 16384]

Métrica	FIFO	LRU
Total de Acessos	11	11
Page Faults	10	9
Taxa de Page Faults	90,91%	81,82%

3.2 ANÁLISE DOS RESULTADOS OBTIDOS

Os resultados mostram que o algoritmo LRU obteve melhor desempenho que o FIFO, apresentando uma taxa de page faults menor (81,82%) em comparação com os 90,91% do FIFO. Isso se justifica pelo fato de que o LRU considera o histórico de uso das páginas, sendo mais eficiente ao manter páginas que foram acessadas recentemente.

Por outro lado, o FIFO não leva em consideração o uso recente das páginas, apenas a ordem de chegada. Isso faz com que ele, em certos cenários, substitua páginas que ainda estão sendo utilizadas, como ocorreu nesta simulação.

Embora a diferença de performance não tenha sido tão grande nesta configuração simples, em cenários com maior carga de acessos e mais processos, a tendência é que o LRU se destaque ainda mais pela sua estratégia mais inteligente de substituição.

4 CONSIDERAÇÕES FINAIS

Esse projeto foi uma ótima chance de colocar em prática os conceitos de paginação em memória virtual que a gente viu em sala. Implementar os algoritmos FIFO e LRU ajudou a entender melhor como cada um funciona e quais são as vantagens e desvantagens deles na hora de substituir páginas.

A simulação mostrou bem a diferença entre os dois algoritmos. O LRU teve uma taxa de page faults menor que o FIFO (81,82% contra 90,91%), o que já dá uma noção de como a escolha do algoritmo pode afetar o desempenho. A gente viu que o FIFO é mais simples de implementar, mas pode acabar tirando páginas que ainda são usadas. Já o LRU leva em conta o uso recente das páginas, então costuma funcionar melhor, mas é mais complexo de gerenciar.

Durante o desenvolvimento, também deu pra aprender bastante sobre programação em C, modularização, alocação dinâmica e como testar diferentes cenários. A visualização do estado da memória durante a simulação também foi super útil pra enxergar como os algoritmos estavam se comportando em tempo real.

REFERENCIAS

1. SILBERSCHATZ, A.; GALVIN, P. B.; GAGNE, G. **Fundamentos de Sistemas Operacionais**. 10ª ed. Rio de Janeiro: LTC, 2018.
2. TANENBAUM, A. S. **Sistemas Operacionais Modernos**. 4ª ed. São Paulo: Pearson, 2016.
3. ARPACI-DUSSEAU, R.; ARPACI-DUSSEAU, A. **Operating Systems: Three Easy Pieces**. Arpaci-Dusseau Books, 2018.
4. Prof. Marcel Rios - Informática. *Sistemas Operacionais - Memória Virtual - Aula Completa*. YouTube, 15 fev. 2022. Disponível em: https://www.youtube.com/watch?v=zI8e3Gu7APg&t=1338s.
5. Dicionário de Informática. *Memória Virtual e Paginação - Aula 1*. YouTube, 17 ago. 2021. Disponível em: https://www.youtube.com/watch?v=z11N1DhgYO0.
6. UNIVESP. *Memória Virtual e Gerenciamento de Memória*. YouTube, 3 mar. 2023. Disponível em: https://www.youtube.com/watch?v=kJBpG3v2P5o.