UNIVERSIDADE PRESBITERIANA MACKENZIE
Faculdade de Computação e Informática

Relatório – Simulador de Paginação de Memória

Disciplina: Sistemas Operacionais

Professor: Lucas Cerqueira Figueiredo

Integrantes:

Eric Itallo – RA: 10428565

Juan Nacif – RA: 10428509

Totti Gomes – RA: 10428490

Richard Barbosa – RA: 10420179

Victor Romano – RA: 10416696

SUMÁRIO

1. Introdução

Durante a disciplina de Sistemas Operacionais, aprendemos como funciona o gerenciamento de memória em sistemas reais. Um dos conceitos mais importantes vistos foi a paginação, que permite que os processos usem a memória de forma mais eficiente, dividindo-a em páginas e frames.

Pensando nisso, desenvolvemos este projeto com o objetivo de criar um simulador de paginação, capaz de mostrar na prática como o sistema operacional realiza a alocação de páginas na memória e como funciona a substituição quando a memória está cheia.

O projeto tem como foco os algoritmos de substituição de páginas FIFO (First-In First-Out) e LRU (Least Recently Used), que foram implementados e testados ao longo do desenvolvimento. Dessa forma, conseguimos visualizar como cada um deles se comporta durante uma sequência de acessos à memória.

2. Como o simulador foi construído

Estruturas do simulador

Para que o simulador refletisse bem o que acontece em um sistema operacional real, criamos algumas estruturas principais no nosso código:

- Página: Guarda informações de cada página, como se ela está presente na memória, qual frame ela ocupa, quando foi carregada e quando foi acessada pela última vez.

- Processo: Cada processo tem seu identificador (PID), seu tamanho em bytes, o número total de páginas e uma tabela de páginas que ele usa.

- Memória Física: Guarda os frames que estão disponíveis no sistema, indicando quais páginas de quais processos estão atualmente ocupando cada frame.

- Frame: Representa uma posição na memória física, dizendo qual processo e qual página estão naquele frame.

- Simulador: É a estrutura que gerencia tudo, incluindo os processos, a memória física, o tempo de simulação e as estatísticas, como quantidade de acessos e de page faults.

Algoritmos implementados

O simulador trabalha atualmente com dois algoritmos de substituição de páginas:

- FIFO (First-In First-Out): Substitui a página que está há mais tempo na memória.

- LRU (Least Recently Used): Substitui a página que ficou mais tempo sem ser usada.

Organização do código

Optamos por dividir o código em três partes principais: simulador.h, simulador.c e main.c. Essa divisão ajuda bastante na organização e na manutenção do código.

Funcionalidades

- Adicionar novos processos, informando o tamanho em bytes.

- Executar a simulação, onde os processos fazem acessos a seus endereços de memória.

- Visualizar o estado da memória física.

- Ver estatísticas da simulação.

Limitações

- Até o momento, só os algoritmos FIFO e LRU foram implementados.

- O simulador foca na gerência de memória e não simula outros aspectos como acesso ao disco ou tempo de execução dos processos.

3. Tecnologias, técnicas e ferramentas

Durante o desenvolvimento, utilizamos diversas ferramentas e práticas que fazem parte do dia a dia de quem trabalha com desenvolvimento de software.

Técnicas usadas:

- Programação em linguagem C.

- Manipulação de ponteiros e alocação dinâmica de memória.

- Estruturas de dados bem definidas com struct.

- Modularização do código em vários arquivos.

- Uso de um contador de tempo na simulação para gerenciar algoritmos como LRU e FIFO.

Ferramentas usadas:

- VSCode como ambiente de desenvolvimento.

- GCC como compilador C.

- Git e GitHub para controle de versões e trabalho colaborativo.

O uso do GitHub trouxe muitos benefícios para o trabalho em grupo. Conseguimos organizar as tarefas, cada um trabalhou em partes diferentes do projeto, e foi possível acompanhar todas as mudanças feitas no código.

4. Resultados e observações

O simulador funciona exatamente como planejado. Ao executar uma sequência de acessos à memória, podemos observar claramente quando ocorrem page faults e quando uma página precisa ser substituída.

Testando com acessos totalmente aleatórios, percebemos que tanto o FIFO quanto o LRU apresentaram taxas de page faults muito parecidas. Isso acontece porque, sem padrão de localidade temporal, a vantagem do LRU desaparece.

Exemplo de resultados em cenário aleatório:

Algoritmo	Page Faults	Taxa de Page Fault
FIFO	16	80
LRU	15	75

Como era esperado, o LRU apresentou apenas uma pequena redução de page faults em relação ao FIFO, mesmo sendo considerado um algoritmo mais sofisticado. Essa diferença reduzida se deve ao fato de que, em acessos aleatórios, não há localidade temporal para o LRU explorar.

5. Conclusão

Desenvolver esse simulador foi fundamental para consolidar nossos conhecimentos sobre gerenciamento de memória. Aprendemos na prática o funcionamento dos algoritmos FIFO

e LRU e vimos que, em cenários de acesso totalmente aleatórios, a vantagem do LRU sobre o FIFO é bem pequena.

Além disso, o uso do GitHub permitiu que trabalhássemos de forma organizada, colaborativa e com controle de versão completo, melhorando a qualidade do projeto como um todo.

6. Referências

- Especificação do Projeto 2 – Simulador de Paginação de Memória.