PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

INF 1316
Relatório de Atividade - Trabalho 2
Gerência Simulador de Memória Virtual

2210872 Felipe Antelo Machado de Oliveira
2112171 Pedro Henrique de Oliveira Valentim Gomes

Objetivo:

Para o trabalho, os alunos devem desenvolver um simulador de **memória virtual** em C. O simulador deve ser capaz de interpretar uma sequência real de acessos à memória e simular o comportamento do sistema operacional na paginação e substituição de páginas, aplicando diferentes algoritmos de substituição. A proposta envolve aplicar diferentes algoritmos de substituição (LRU, Segunda Chance e Clock). O simulador deve, então, calcular o número de falhas de página (page faults) e o número de páginas sujas que precisam ser gravadas em disco durante a execução. No fim de tudo, então, cabe a nós alunos, através desse relatório, analisar as estatísticas e fazer comentários pertinentes do trabalho.

Resumindo, a entrada do programa deve ser um comando no terminal no formato “./sim-virtual algoritmo arquivo.log pagesize memory” e, após realizar o algoritmo seguindo todos os argumentos passados, o código deve retornar uma saída com diferentes estatísticas geradas.

Estrutura do Programa :

A estrutura principal é composta pelos seguintes elementos:

- Struct Frame: Representa cada quadro de memória física, contendo os campos necessários para simular a paginação: page_number, valid, referenced, modified e last_access_time.
- Argumentos de linha de comando: Permitem selecionar o algoritmo de substituição (LRU, 2nd ou clock), o arquivo de entrada (.log), o tamanho da página em KB e o tamanho da memória física em MB.
- Função calc_shift: Calcula quantos bits devem ser descartados do endereço lógico para se obter o número da página, com base no tamanho da página informado.
- Função find_page: Verifica se a página solicitada já se encontra carregada na memória.
- Funções de substituição: Três funções distintas (select_frame_LRU,

select_frame_SecondChance e select_frame_Clock) implementam os algoritmos de substituição exigidos.

- Função simulate: Realiza a leitura linha por linha do arquivo de entrada, simula os acessos à memória e aplica a lógica da substituição quando necessário.
- Função main: Realiza o controle geral da execução, incluindo validação de argumentos, alocação dinâmica dos quadros e impressão das estatísticas ao final.

Abaixo, farei uma breve explicação de cada um dos algoritmos utilizados.

LRU (Least Recently Used): O algoritmo LRU substitui a página que está há mais tempo sem ser utilizada. A ideia central é que páginas que foram utilizadas recentemente têm maior probabilidade de serem reutilizadas em breve, enquanto páginas menos acessadas no passado são menos úteis. Para aplicar esse conceito, é necessário manter o controle do tempo do último acesso de cada página, o que pode ser feito com um contador global ou uma pilha de acessos. Em cada acesso à memória, o simulador atualiza o instante de último uso da página. Quando ocorre uma falta de página e é necessário substituir um quadro ocupado, o LRU escolhe a página com o menor tempo de acesso registrado. Embora eficaz em termos de desempenho, o LRU tem maior custo de implementação, principalmente em sistemas reais, onde o rastreamento contínuo do histórico de acessos pode ser inviável em hardware.

Segunda Chance (2nd): A Segunda Chance é uma otimização do algoritmo FIFO (First-In, First-Out) que busca evitar a remoção de páginas que tenham sido utilizadas recentemente. Ela utiliza um bit adicional chamado bit de referência (R), que é ativado (R = 1) sempre que uma página é acessada. Ao escolher uma página para substituição, o algoritmo percorre os quadros em ordem circular. Se encontrar uma página com R = 1, o algoritmo entende que essa página foi recentemente utilizada e concede uma "segunda chance": zera o bit R e passa para o próximo quadro. Caso encontre uma página com R = 0, ela é escolhida para substituição. Essa abordagem balanceia simplicidade e desempenho, permitindo preservar páginas ativas sem exigir controle temporal detalhado.

Clock: O algoritmo Clock é uma implementação otimizada da Segunda Chance. Em vez de percorrer os quadros com base em uma fila lógica, o Clock organiza os quadros em uma estrutura circular (como os ponteiros de um relógio). Um ponteiro gira continuamente pelos quadros. Quando ocorre uma falta de página, o algoritmo verifica a página apontada: se R = 1, ele zera o bit e move o ponteiro para o próximo quadro; se R = 0, substitui aquela página imediatamente. O Clock tem a mesma lógica de Segunda Chance, mas melhora a eficiência com o uso de um ponteiro único, evitando percorrer todos os quadros a cada substituição. É amplamente utilizado em sistemas operacionais reais por seu bom equilíbrio entre desempenho e complexidade.

Ótimo: O algoritmo Ótimo, ou Optimal Replacement, define o limite teórico de

desempenho para algoritmos de substituição de páginas. Ele substitui a página que será utilizada mais tarde no futuro (ou nunca mais será utilizada). Como essa decisão exige conhecimento prévio de toda a sequência de acessos, o algoritmo não pode ser implementado em tempo real, sendo utilizado apenas em simulações e estudos comparativos. Em um simulador, é possível ler previamente toda a sequência de acessos e, no momento da substituição, escolher a página cuja próxima aparição esteja mais distante. O Ótimo garante o menor número possível de faltas de página, sendo útil como base de comparação para avaliar o quão perto da perfeição os algoritmos práticos conseguem chegar.

Solução do programa:

Durante a execução, o simulador lê endereços de memória no formato hexadecimal seguidos por uma operação (R para leitura ou W para escrita). Cada endereço é convertido no número da página por meio de um deslocamento de bits (>> page_shift), de acordo com o tamanho da página.

Se a página já estiver carregada na memória física, suas flags são atualizadas. Caso contrário, ocorre uma falta de página (page fault) e o simulador verifica se há um quadro livre. Se todos estiverem ocupados, entra em ação o algoritmo de substituição escolhido. Por exemplo, no LRU, o simulador seleciona a página que não é acessada há mais tempo, utilizando um contador global de tempo (time_counter). Já no Segunda Chance e no Clock, o simulador percorre os quadros com base em um ponteiro circular (clock_pointer) e dá uma nova chance a páginas cujo bit referenced esteja ativo, zerando-o. Caso a página removida esteja com a flag modified ativada, ela é contabilizada como uma página suja escrita.

Ao término da execução, o simulador exibe um relatório contendo os parâmetros utilizados, o número total de falhas de página e o número total de páginas sujas escritas. Naturalmente, o arquivo que é passado, o algoritmos escolhido, o tamanho de cada página em KB e o total de memória físca.

A seguir, irei apresentar alguns prints e conclusões tiradas em relação a testes. Por exemplo: (quebra de página para caber os prints, olhar a próxima)

Teste alternando os algoritmos:

```
~/workspace$ ./sim-virtual LRU compilador.log 8 1
./sim-virtual 2nd compilador.log 8 1
./sim-virtual clock compilador.log 8 1
./sim-virtual otimo compilador.log 8 1
Executando o simulador...
Arquivo de entrada: compilador.log
Tamanho da memoria fisica: 1 MB
Tamanho das paginas: 8 KB
Algoritmo de substituicao: LRU
Numero de Faltas de Paginas: 36707
Numero de Paginas Escritas: 5775
Executando o simulador...
Arquivo de entrada: compilador.log
Tamanho da memoria fisica: 1 MB
Tamanho das paginas: 8 KB
Algoritmo de substituicao: 2nd
Numero de Faltas de Paginas: 38842
Numero de Paginas Escritas: 6253
Executando o simulador...
Arquivo de entrada: compilador.log
Tamanho da memoria fisica: 1 MB
Tamanho das paginas: 8 KB
Algoritmo de substituicao: clock
Numero de Faltas de Paginas: 38842
Numero de Paginas Escritas: 6253
Executando o simulador...
Arquivo de entrada: compilador.log
Tamanho da memoria fisica: 1 MB
Tamanho das paginas: 8 KB
Algoritmo de substituicao: otimo
Numero de Faltas de Paginas: 21271
Numero de Paginas Escritas: 3712
```

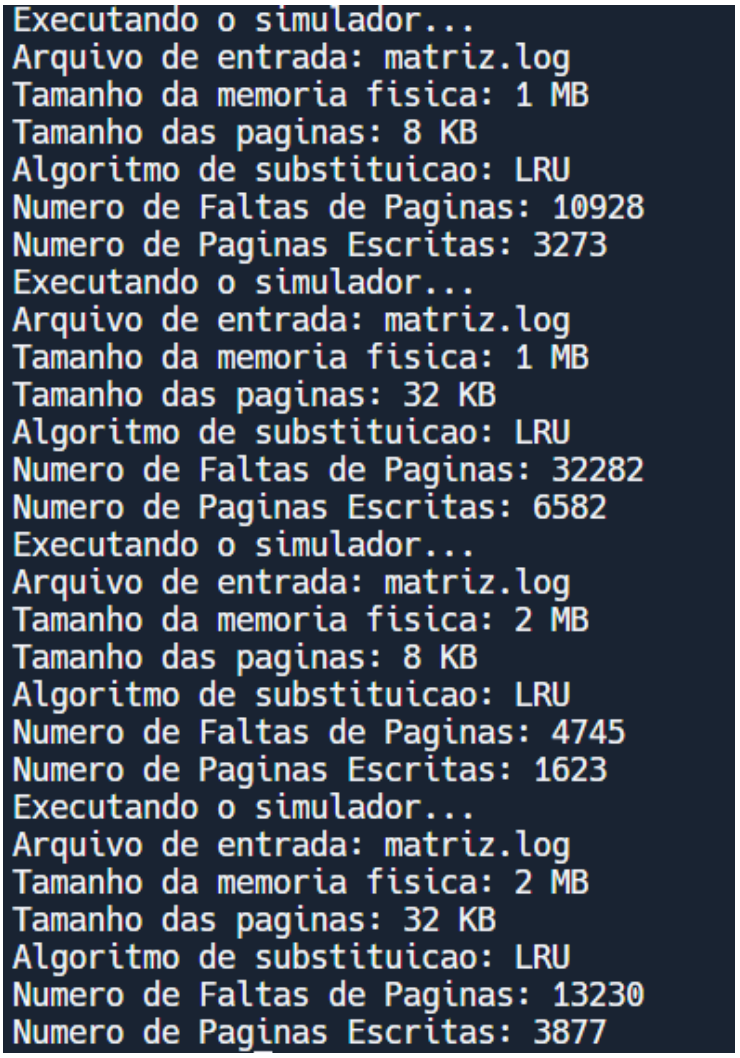
- O algoritmo Ótimo sempre remove a página que será usada mais tarde no futuro (ou nunca mais será usada), o que leva ao menor número possível de substituições.

- LRU teve desempenho melhor do que Segunda Chance e Clock, o que faz sentido, pois LRU tenta se aproximar do comportamento do ótimo, eliminando a página menos recentemente usada de forma precisa (via last_access_time).

- Já a Segunda Chance e o Clock apenas verificam o bit de referência (referenced). Se ele estiver ativo, a página ganha uma segunda chance, mesmo que ela não esteja realmente sendo usada de forma frequente — ou

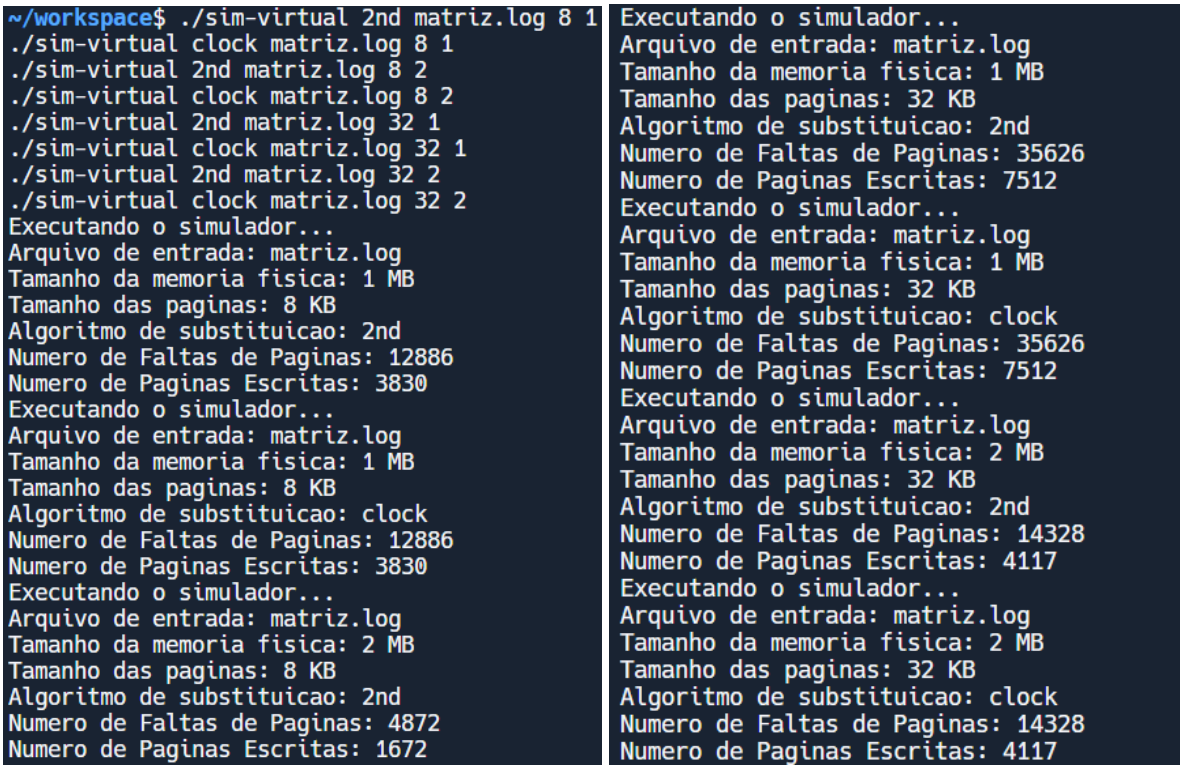
seja, elas podem manter páginas "velhas" na memória à toa, o que aumenta as faltas.

Teste modificando valores de página e modificando tamanho de memória disponível:



Nos testes com o programa matriz.log, foi possível observar claramente o impacto da memória disponível e do tamanho de página. A duplicação da memória física de 1MB para 2MB resultou em uma redução expressiva no número de faltas de página, evidenciando a importância do espaço disponível para manter páginas em RAM. Além disso, o uso de páginas de 8KB foi consideravelmente mais eficiente do que páginas de 32KB, especialmente para esta carga de trabalho com acessos intensivos e repetitivos a dados estruturados. Isso demonstra que páginas menores permitem maior flexibilidade na alocação e melhor aproveitamento do princípio da localidade, reduzindo a necessidade de substituições e, consequentemente, o número de páginas sujas escritas.

Teste comparação 2nd e clock



```
~/workspace$ ./sim-virtual 2nd matriz.log 8 1
./sim-virtual clock matriz.log 8 1
./sim-virtual 2nd matriz.log 8 2
./sim-virtual clock matriz.log 8 2
./sim-virtual 2nd matriz.log 32 1
./sim-virtual clock matriz.log 32 1
./sim-virtual 2nd matriz.log 32 2
./sim-virtual clock matriz.log 32 2
Executando o simulador...
Arquivo de entrada: matriz.log
Tamanho da memoria fisica: 1 MB
Tamanho das paginas: 8 KB
Algoritmo de substituicao: 2nd
Numero de Faltas de Paginas: 12886
Numero de Paginas Escritas: 3830
Executando o simulador...
Arquivo de entrada: matriz.log
Tamanho da memoria fisica: 1 MB
Tamanho das paginas: 8 KB
Algoritmo de substituicao: clock
Numero de Faltas de Paginas: 12886
Numero de Paginas Escritas: 3830
Executando o simulador...
Arquivo de entrada: matriz.log
Tamanho da memoria fisica: 2 MB
Tamanho das paginas: 8 KB
Algoritmo de substituicao: 2nd
Numero de Faltas de Paginas: 4872
Numero de Paginas Escritas: 1672
Executando o simulador...
Arquivo de entrada: matriz.log
Tamanho da memoria fisica: 1 MB
Tamanho das paginas: 32 KB
Algoritmo de substituicao: 2nd
Numero de Faltas de Paginas: 35626
Numero de Paginas Escritas: 7512
Executando o simulador...
Arquivo de entrada: matriz.log
Tamanho da memoria fisica: 1 MB
Tamanho das paginas: 32 KB
Algoritmo de substituicao: clock
Numero de Faltas de Paginas: 35626
Numero de Paginas Escritas: 7512
Executando o simulador...
Arquivo de entrada: matriz.log
Tamanho da memoria fisica: 2 MB
Tamanho das paginas: 32 KB
Algoritmo de substituicao: 2nd
Numero de Faltas de Paginas: 14328
Numero de Paginas Escritas: 4117
Executando o simulador...
Arquivo de entrada: matriz.log
Tamanho da memoria fisica: 2 MB
Tamanho das paginas: 32 KB
Algoritmo de substituicao: clock
Numero de Faltas de Paginas: 14328
Numero de Paginas Escritas: 4117
```

Comentários do motivo dos resultados de segunda chance e clock serem similares será feito nas observações e conclusões do trabalho. Nesse momento, gostaria de observar que ambos tiveram o mesmo desempenho, tanto em número de faltas de página quanto em páginas sujas escritas. Isso era esperado, já que no simulador implementado ambos compartilham a mesma lógica de substituição. Ao dobrar a memória física de 1 MB para 2 MB, a redução no número de faltas de página foi expressiva, reforçando o impacto direto da quantidade de quadros disponíveis na eficiência do sistema. Além disso, foi possível observar que o uso de páginas menores (8 KB) levou a um desempenho significativamente superior em comparação com páginas maiores (32 KB), principalmente no cenário com menor quantidade de memória. Isso se deve à maior granularidade e melhor aproveitamento do princípio da localidade espacial, típico de cargas de trabalho científicas. Por fim, os testes também evidenciaram que o aumento de substituições acarreta em mais páginas sujas sendo escritas.

Testes para o algoritmo ótimo

```
~/workspace$ ./sim-virtual otimo simulador.log 8 1
./sim-virtual otimo simulador.log 8 2
./sim-virtual otimo simulador.log 32 1
./sim-virtual otimo simulador.log 32 2
Executando o simulador...
Arquivo de entrada: simulador.log
Tamanho da memoria fisica: 1 MB
Tamanho das paginas: 8 KB
Algoritmo de substituicao: otimo
Numero de Faltas de Paginas: 9179
Numero de Paginas Escritas: 3527
Executando o simulador...
Arquivo de entrada: simulador.log
Tamanho da memoria fisica: 2 MB
Tamanho das paginas: 8 KB
Algoritmo de substituicao: otimo
Numero de Faltas de Paginas: 4748
Numero de Paginas Escritas: 2159
Executando o simulador...
Arquivo de entrada: simulador.log
Tamanho da memoria fisica: 1 MB
Tamanho das paginas: 32 KB
Algoritmo de substituicao: otimo
Numero de Faltas de Paginas: 28328
Numero de Paginas Escritas: 7581
Executando o simulador...
Arquivo de entrada: simulador.log
Tamanho da memoria fisica: 2 MB
Tamanho das paginas: 32 KB
Algoritmo de substituicao: otimo
Numero de Faltas de Paginas: 11870
Numero de Paginas Escritas: 4421
```

A execução do simulador utilizando o algoritmo ótimo reforçou o comportamento esperado: mesmo com conhecimento total da sequência de acessos, o número de faltas de página ainda é fortemente influenciado pela quantidade de memória física disponível e pelo tamanho da página. Quando a memória foi dobrada de 1 MB para 2 MB, o número de faltas foi reduzido pela metade em ambos os tamanhos de página, confirmando que o aumento de quadros disponíveis melhora significativamente o desempenho. No entanto, o uso de páginas maiores (32 KB) resultou em um número significativamente maior de faltas, mesmo com memória abundante. Isso mostra que o aumento no tamanho da página pode levar à perda de eficiência na reutilização de quadros e ao aumento da fragmentação interna. O algoritmo ótimo conseguiu limitar o impacto desses fatores, mas não os anula.

Observação e conclusões:

O simulador implementado conseguiu reproduzir com fidelidade o funcionamento dos principais algoritmos de substituição de páginas estudados na disciplina. Ao executar o simulador com os algoritmos LRU, Segunda Chance e

Clock, tornou-se evidente que cada política possui características próprias que influenciam tanto o número de faltas de página quanto a quantidade de páginas sujas escritas.

O algoritmo ótimo mostrou-se interessante como base de comparação teórica, evidenciando o limite inferior de faltas de página. Já o LRU, mesmo sem conhecimento do futuro, apresentou resultados próximos do ótimo em muitas situações, confirmando sua eficácia como algoritmo realista.

Algo visível também que ocorreu, foi a similaridade entre o algoritmo clock e segunda chance. Essa similaridade pode parecer errada, entretanto, isso ocorre pelo fato de ambos utilizarem a mesma lógica de verificação do bit de referência e substituição com ponteiro circular. Na implementação deste trabalho, a função select_frame_Clock() é uma chamada direta à lógica da Segunda Chance, tornando os dois algoritmos equivalentes no simulador. Essa equivalência reforça a ideia de que o algoritmo Clock nada mais é do que uma otimização prática da Segunda Chance tradicional. Sendo assim, ambos percorrem os quadros com um ponteiro circular, ambos verificam o bit referenced, ambos zeram esse bit se ele for 1 e seguem em frente até encontrar um quadro com R = 0 e por fim ambos substituem o primeiro quadro encontrado com R = 0. Percebe-se então que ambos têm estratégias similares para marcar o R, zerando-o ou substituindo-o, gerando resultados extremamente similares.

Existem certos padrões que podemos perceber ao alterar os argumentos passados e nos “divertimos” com o código. A variação dos parâmetros do simulador afeta diretamente o comportamento e o desempenho do gerenciamento de memória. O algoritmo de substituição escolhido determina a estratégia para remoção de páginas da memória física quando ocorre um page fault. O algoritmo Ótimo, por conhecer o futuro, apresenta o menor número possível de faltas. O LRU aproxima esse desempenho ao considerar o histórico de uso recente. Já os algoritmos Segunda Chance e Clock tendem a ter desempenho ligeiramente inferior, pois podem manter páginas pouco utilizadas por conta da lógica de segunda oportunidade.

O tamanho da página influencia na granularidade da alocação de memória. Páginas menores (como 8 KB) permitem mais quadros disponíveis, o que favorece o aproveitamento da memória, mas aumenta o número de páginas a serem gerenciadas, potencialmente elevando o número de page faults. Por outro lado, páginas maiores (como 32 KB) reduzem o número total de quadros, o que pode diminuir as trocas, mas aumenta a fragmentação interna.

Por fim, o tamanho da memória física teórica impacta diretamente o espaço disponível para manter páginas ativas. Ao dobrar a memória (de 1 MB para 2 MB, por exemplo), o número de quadros disponíveis também dobra, reduzindo significativamente o número de faltas de página e, consequentemente, a necessidade de substituições e escritas de páginas sujas.

Ao todo, o trabalho proporcionou uma compreensão prática profunda do impacto das políticas de substituição de páginas no desempenho de sistemas operacionais, além de reforçar conceitos de paginação, bit de referência, bit de

modificação e simulação de memória virtual.

Como esse é o último trabalho, gostaríamos (Felipe e Pedro) de agradecer ao professor Seibel e ao monitor Enzo pelo ótimo período. Boas férias!