

Relatório

Projeto Gerenciador de Memória

Jhonatha Cordeiro - 5984, Viviane Renizia Silva - 5209

10 de dezembro de 2020



Introdução

1. Introdução
2. Funcionamento
3. Comparativo
4. Conclusão
5. Referências

Introdução

Neste projeto foi implementado em linguagem C algoritmo de substituição de páginas FIFO (First In, First Out) e comparado o desempenho do algoritmo implementado com o Random (Aleatório) disponibilizado pelo professor. Nas próximas seções será apresentado as principais funcionalidades do código e do algoritmo de substituição de páginas escolhido. Também abordaremos sobre o funcionamento do programa e ao final traremos uma conclusão em relação a comparação dos dois algoritmos implementados.

Funcionamento

Funcionamento do algoritmo FIFO: O sistema operacional mantém uma lista de todas as páginas atuais na memória, a página mais antiga fica no início da lista, já a página mais recente fica ao final da lista. Na ocorrência de uma falta de página a primeira página da lista é removida, e a nova página é adicionada ao final da lista.

Funcionamento do algoritmo Random: O algoritmo random, como o nome já diz, ele randomiza, sorteia, qual página será substituída da memória.

Principais funções do código:

Abaixo será listado as principais funções do código e seu respectivo funcionamento.

find_next_frame: Função responsável por buscar na memória de forma circular, um frame livre.

parse: Função responsável por converter as strings contidas no arquivo anomaly.dat em números inteiros.

run: Função principal do código, responsável por verificar se existe um endereço virtual mapeado e executar a simulação de page faults. Retorna ao final a quantidade de page faults.

simulate: Função responsável por simular o funcionamento de um algoritmo de substituição de páginas, verifica se ocorreu falta de página ou não, executa passos comuns em vários algoritmos, como exemplo, verificar se tem memória disponível, liberar memória, remover páginas e gravar endereço físico na tabela de páginas.

random_page: Função responsável por sortear a página que será mapeada na memória física.

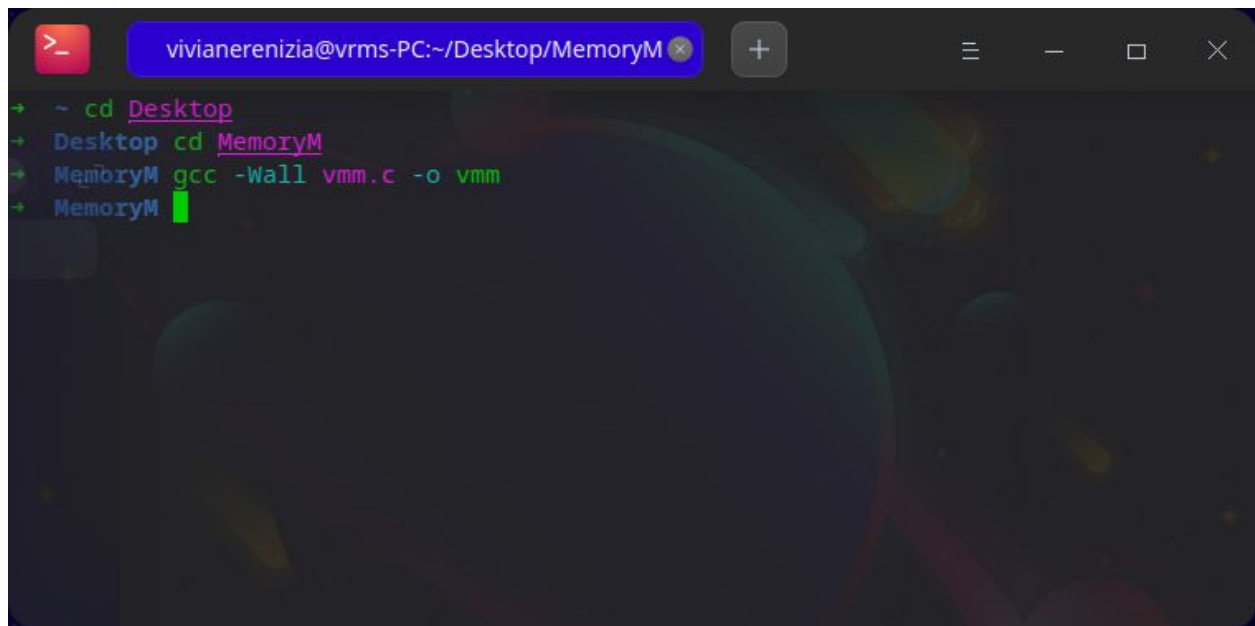
fifo: Função responsável por executar o algoritmo FIFO.

read_header: Função responsável por ler a quantidade de páginas virtuais e páginas físicas do arquivo anomaly.dat

main: Função mãe do programa, responsável por controlar a execução de todas as funções mencionadas acima.

Execução dos algoritmos:

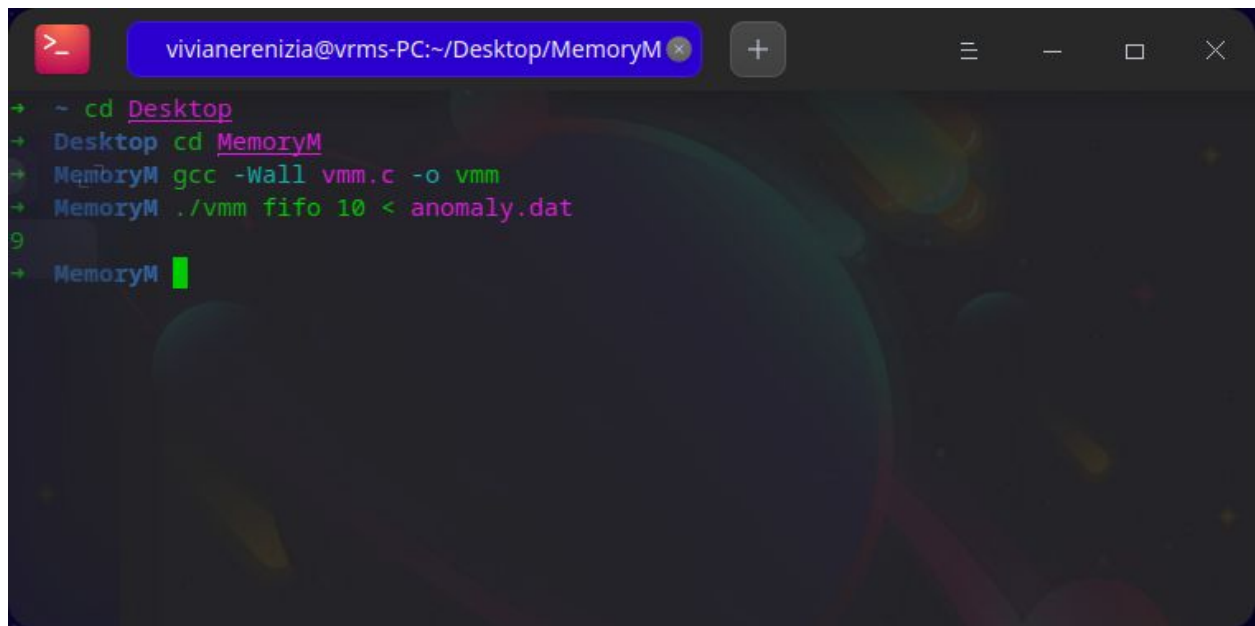
Passo 1: Compilar o código

A terminal window with a dark background and a space-themed wallpaper. The window title bar shows the user 'vivianerenizia@vrms-PC' and the current directory '~/Desktop/MemoryM'. The terminal content shows a sequence of commands: 'cd Desktop', 'cd MemoryM', and 'gcc -Wall vmm.c -o vmm'. The prompt changes from '~' to 'Desktop' and then 'MemoryM' as the user navigates through the directories. The final command 'gcc -Wall vmm.c -o vmm' is followed by a green cursor.

```
>_ vivianerenizia@vrms-PC:~/Desktop/MemoryM
→ ~ cd Desktop
→ Desktop cd MemoryM
→ MemoryM gcc -Wall vmm.c -o vmm
→ MemoryM
```

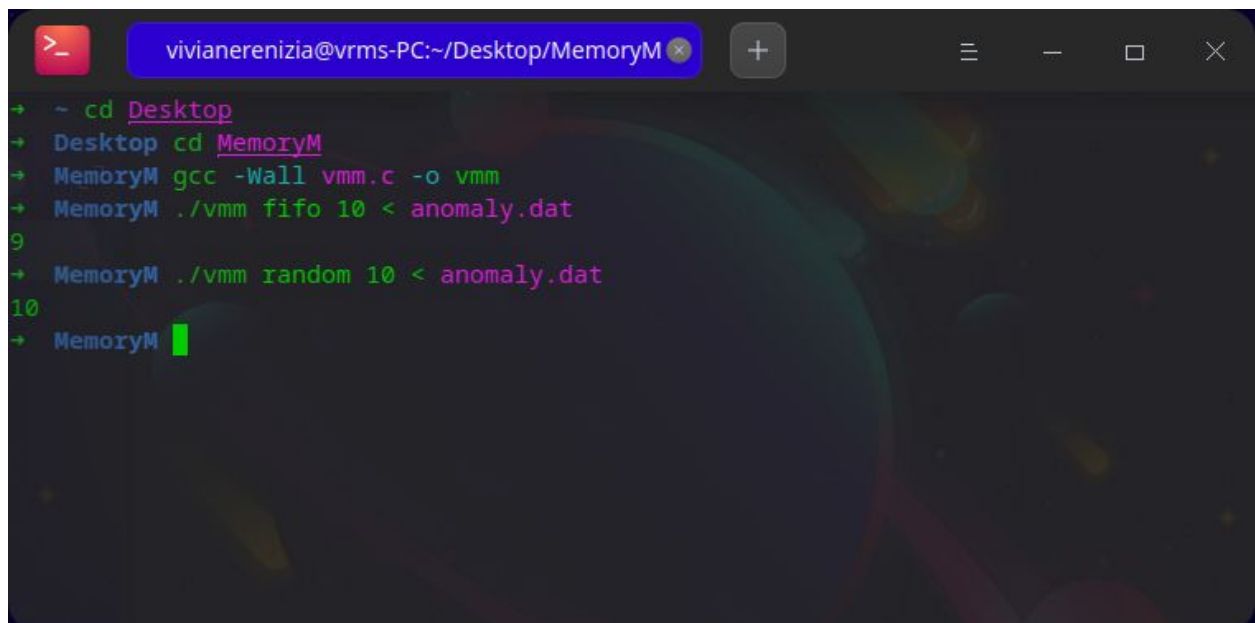
Passo 2: Execução

FIFO :



```
vivianerenizia@vrms-PC:~/Desktop/MemoryM
~ cd Desktop
Desktop cd MemoryM
MemoryM gcc -Wall vmm.c -o vmm
MemoryM ./vmm fifo 10 < anomaly.dat
9
MemoryM
```

Random:



```
vivianerenizia@vrms-PC:~/Desktop/MemoryM
~ cd Desktop
Desktop cd MemoryM
MemoryM gcc -Wall vmm.c -o vmm
MemoryM ./vmm fifo 10 < anomaly.dat
9
MemoryM ./vmm random 10 < anomaly.dat
10
MemoryM
```

Comparativo

A tabela abaixo representa os resultados obtidos através da comparação de execução dos algoritmos FIFO e Random.

Execução	FIFO	Random
1	9	10
2	9	8
3	9	9
4	9	8
5	9	7
6	9	8
7	9	9
8	9	7
9	9	10
10	9	9
11	9	8
12	9	8
13	9	8
14	9	9
15	9	10
Média	9	8,5

A tabela acima traz o resultado de 15 execuções de FIFO e RANDOM. Assim, pode-se observar:

- O FIFO sempre dá o mesmo resultado;
- O RANDOM, por ser aleatório, traz resultados diversos;

- O algoritmo RANDOM possui performance duvidosa já que depende da aleatoriedade;
- As médias de Page Fault foram próximas uma da outra;
- Ambos os algoritmos têm desempenho duvidoso em relação aos outros algoritmos de troca de página, existem melhores.

Conclusão

É possível concluir que ambos os algoritmos são ineficientes, o algoritmo random, depende da aleatoriedade para ser eficiente, já o algoritmo fifo se torna ineficiente pelo fato de remover páginas pouco referenciadas e páginas que são referenciadas frequentemente. Uma abordagem mais importante seria utilizar o algoritmo FIFO com o bit R evitando a substituição de páginas frequentemente usadas. Um ponto importante a se destacar é a complexidade do esqueleto disponibilizado. Código um pouco extenso, complexo e com poucos comentários. A descrição do projeto também nos levou a entender que algo mais, além da implementação do algoritmo escolhido, deveria ser feito. Porém acreditamos que foi apenas uma interpretação equivocada do grupo. Outro ponto é que não conseguimos solucionar um bug no projeto. Nosso algoritmo retorna corretamente o número de page faults, mas printando as variáveis `to_free` e `fifo_frm` é possível perceber que elas não estão apontando o mesmo endereço, possuem valores diferentes. Com outras implementações testadas retornava um bug de mapeamento e não obtinhamos o resultado esperado. Em conclusão, retornamos o resultado correto, porém, a página retirada da memória pode não ser a página correta. Não conseguimos "driblar" o assert.

Referências

TANENBAUM, A. S.; Sistemas Operacionais Modernos. Editora Pearson Brasil, 2a. edição, 2003.

Aulas da Disciplina de Sistemas Operacionais - SIN 351, ministradas pelo professor Rodrigo Moreira, UFV-CRP, 2020.