

SIN 351 – Sistemas Operacionais

Júlia Mendonça Silva – 5956

Relatório do Projeto 2

Neste relatório será detalhado o funcionamento do código desenvolvido para o segundo projeto proposto pela disciplina de Sistemas Operacionais do curso de Sistemas de Informação da UFV-CRP, assim como comparação dos valores de saídas.

1. O projeto

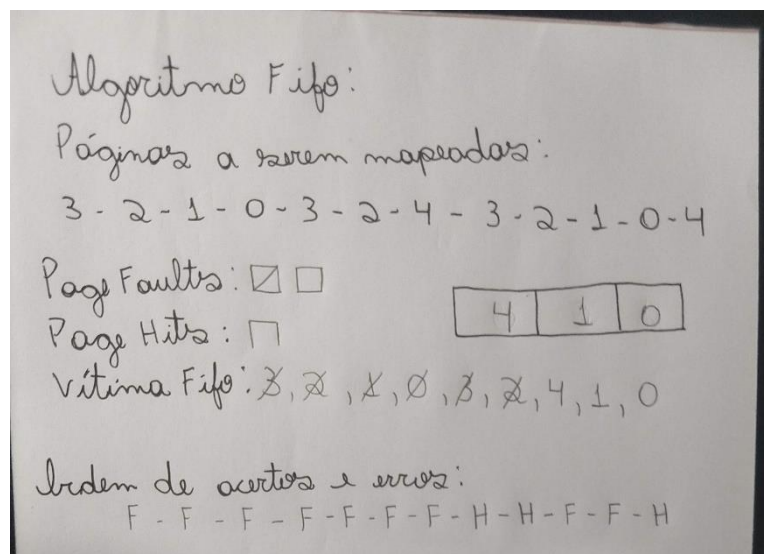
O projeto consistiu em implementar um algoritmo de substituição de páginas no contexto de gerenciamento de memória de um SO. O algoritmo escolhido para ser implementado foi o algoritmo FIFO.

2. O FIFO

O algoritmo FIFO é utilizado em diversas áreas da computação por ser um dos mais simples a serem implementados, porém nem sempre é o mais eficiente. Ele consiste em uma espécie de fila, em que o primeiro elemento, neste caso, página, a chegar ou ser mapeado é o primeiro a sair do mapeamento quando houver necessidade.

Para implementar o algoritmo fifo foi utilizado o código disponibilizado pelo professor e apenas a função intitulada “fifo” foi modificada de uma maneira que o algoritmo fosse executado. Durante o desenvolvimento do código foram encontrados problemas que já foram detalhados no README.md.

A partir do arquivo anomaly.dat disponibilizado como arquivo base pelo professor que será mais bem detalhado em breve podemos fazer a seguinte implementação “à mão” do algoritmo:



1 Demonstração do algoritmo FIFO feito à mão. Podemos ver o estado final do mapeamento da memória, assim como a quantidade e ordem em que ocorreram os erros e acertos do algoritmo.

3. Comparando Algoritmos

No código disponibilizado pelo professor já estava implementado o algoritmo Random para substituição de páginas, a seguir estão imagens mostrando a saída de ambos os algoritmos citados neste relatório, assim como uma tabela comparando seus resultados.

```
julia@julia-VirtualBox: ~/Downloads/Projeto2
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

julia@julia-VirtualBox:~/Downloads/Projeto2$ gcc -Wall vmm.c -o vmm
julia@julia-VirtualBox:~/Downloads/Projeto2$ ./vmm fifo 10 < anomaly.dat
Faults: 9
julia@julia-VirtualBox:~/Downloads/Projeto2$ ./vmm fifo 10 < anomaly.dat
Faults: 9
julia@julia-VirtualBox:~/Downloads/Projeto2$ ./vmm fifo 10 < anomaly.dat
Faults: 9
julia@julia-VirtualBox:~/Downloads/Projeto2$ ./vmm fifo 10 < anomaly.dat
Faults: 9
julia@julia-VirtualBox:~/Downloads/Projeto2$ ./vmm fifo 10 < anomaly.dat
Faults: 9
julia@julia-VirtualBox:~/Downloads/Projeto2$ ./vmm fifo 10 < anomaly.dat
Faults: 9
julia@julia-VirtualBox:~/Downloads/Projeto2$ ./vmm fifo 10 < anomaly.dat
Faults: 9
julia@julia-VirtualBox:~/Downloads/Projeto2$ ./vmm fifo 10 < anomaly.dat
Faults: 9
julia@julia-VirtualBox:~/Downloads/Projeto2$ ./vmm fifo 10 < anomaly.dat
Faults: 9
julia@julia-VirtualBox:~/Downloads/Projeto2$
```

2 Execução do algoritmo FIFO

```
julia@julia-VirtualBox: ~/Downloads/Projeto2
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

julia@julia-VirtualBox:~/Downloads/Projeto2$ gcc -Wall vmm.c -o vmm
julia@julia-VirtualBox:~/Downloads/Projeto2$ ./vmm random 10 < anomaly.dat
Faults: 9
julia@julia-VirtualBox:~/Downloads/Projeto2$ ./vmm random 10 < anomaly.dat
Faults: 10
julia@julia-VirtualBox:~/Downloads/Projeto2$ ./vmm random 10 < anomaly.dat
Faults: 10
julia@julia-VirtualBox:~/Downloads/Projeto2$ ./vmm random 10 < anomaly.dat
Faults: 9
julia@julia-VirtualBox:~/Downloads/Projeto2$ ./vmm random 10 < anomaly.dat
Faults: 8
julia@julia-VirtualBox:~/Downloads/Projeto2$ ./vmm random 10 < anomaly.dat
Faults: 9
julia@julia-VirtualBox:~/Downloads/Projeto2$ ./vmm random 10 < anomaly.dat
Faults: 8
julia@julia-VirtualBox:~/Downloads/Projeto2$ ./vmm random 10 < anomaly.dat
Faults: 9
julia@julia-VirtualBox:~/Downloads/Projeto2$ ./vmm random 10 < anomaly.dat
Faults: 8
julia@julia-VirtualBox:~/Downloads/Projeto2$ ./vmm random 10 < anomaly.dat
Faults: 9
julia@julia-VirtualBox:~/Downloads/Projeto2$
```

3 Execução do algoritmo Random

Como podemos ver a saída do algoritmo fifo, ou seja, o número de *page faults* da memória se manteve constante durante todas as 10 execuções, já no algoritmo random, esse número variou.

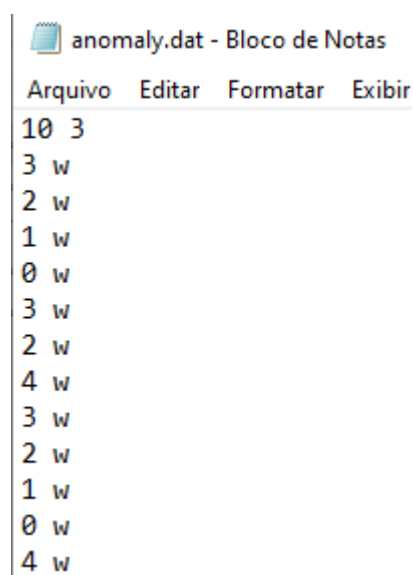
Podemos ver na tabela uma comparação direta e a média de erros dos dois algoritmos para o arquivo base utilizado (anomaly.dat).

Execução	FIFO	Random
1	9	9
2	9	10
3	9	10
4	9	9
5	9	8
6	9	9
7	9	8
8	9	9
9	9	8
10	9	9
Média	9	8,9

Podemos ver que o algoritmo Random se mostrou melhor do que o Fifo nessas 10 execuções seguidas, porém por se tratar de um algoritmo que seleciona randomicamente a página a ser retirada do mapeamento da memória, pode haver casos em que o algoritmo Random pode ter um desempenho pior.

4. O arquivo anomaly.dat

O arquivo anomaly.dat é essencial para a execução dos algoritmos uma vez que ele dita qual a estrutura da memória a ser gerenciada. Em sua primeira linha o arquivo contém o número de páginas virtuais disponível e logo em seguida o número de molduras disponíveis na memória. As linhas subsequentes possuem o nome da página a ser mapeada (um número inteiro) seguidas do tipo de acesso a memória que cada página utilizará. Podemos ver um exemplo de arquivo anomaly.dat abaixo.

The image shows a screenshot of a text editor window titled "anomaly.dat - Bloco de Notas". The window has a menu bar with "Arquivo", "Editar", "Formatar", and "Exibir". The text content of the file is as follows:

```
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
```

4 Exemplo do arquivo anomaly.dat