

PROJETO PRÁTICO - GERENCIADOR DE MEMÓRIA

RELATÓRIO DO PROJETO

Projeto desenvolvido para a disciplina de Sistemas Operacionais, do curso de Sistemas de Informação da Universidade Federal de Viçosa - Campus Rio Paranaíba.

[Link do repositório no GitHub](#)

1. Execução do Algoritmo

Existem dois arquivos importantes para execução do código: "vmm.c" e "anomaly.dat". Para executar, digite o comando:

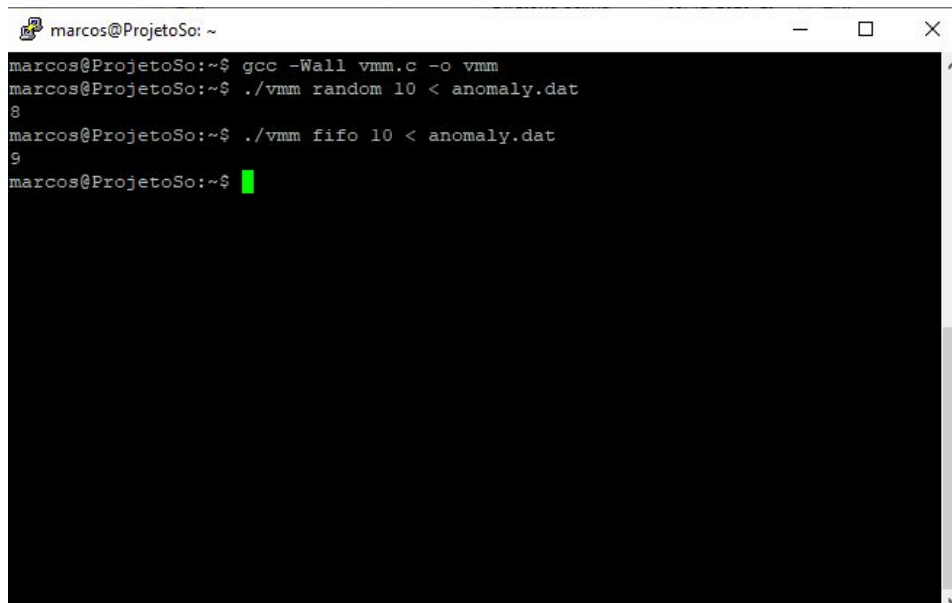
```
$ gcc -Wall vmm.c -o vmm
```

O comando acima, gera o arquivo para ser executado. Para acessar o programa, utilizaremos o código:

```
$ ./vmm [FUNCAO] 10 < anomaly.dat
```

Ao inserir esse código, estamos passando para a função os parâmetros necessários para serem analisados. Em [FUNCAO] substitua por:

- **random:** caso queira utilizar a função fornecida pelo professor;
- **fifo:** para utilizar a função implementada pelos autores do projeto.



```
marcos@ProjetoSo: ~  
marcos@ProjetoSo:~$ gcc -Wall vmm.c -o vmm  
marcos@ProjetoSo:~$ ./vmm random 10 < anomaly.dat  
8  
marcos@ProjetoSo:~$ ./vmm fifo 10 < anomaly.dat  
9  
marcos@ProjetoSo:~$
```

2. Função FIFO implementada pelos autores

```
//A pagina mais antiga pode ser removida
int fifo(int8_t** page_table, int num_pages, int prev_page,
        int fifo_frm, int num_frames, int clock) {

    int i; //Incrementador

    do {
        //Se o endereço físico é o mais velho para ser retornado
        if(page_table[i][PT_FRAMEID] == 0){
            return i;
        }

        ++i; //Incrementa para o próximo

    } while (i < num_pages); //Roda até encontrar a página vítima

    return -1;
}
```

O algoritmo de troca de página FIFO - First in, First out, tem como objetivo a remoção das páginas de memória mais antigas, fazendo com que o primeiro que entre seja o primeiro que será removido. Uma característica deste algoritmo o seu baixo custo é a simplicidade em sua implementação, por outro lado é que a página mais antiga pode ser usada frequentemente, causando um aumento de page fault

3. RANDOM *versus* FIFO

FIFO - First in First out

```
marcos@ProjetoSo: ~
marcos@ProjetoSo:~$ ./vmm fifo 10 < anomaly.dat
9
marcos@ProjetoSo:~$ ./vmm fifo 10 < anomaly.dat
9
marcos@ProjetoSo:~$ ./vmm fifo 10 < anomaly.dat
9
marcos@ProjetoSo:~$ ./vmm fifo 10 < anomaly.dat
9
marcos@ProjetoSo:~$ ./vmm fifo 10 < anomaly.dat
9
marcos@ProjetoSo:~$ ./vmm fifo 10 < anomaly.dat
9
marcos@ProjetoSo:~$ ./vmm fifo 10 < anomaly.dat
9
marcos@ProjetoSo:~$ ./vmm fifo 10 < anomaly.dat
9
marcos@ProjetoSo:~$ ./vmm fifo 10 < anomaly.dat
9
marcos@ProjetoSo:~$
```

RANDOM

```
marcos@ProjetoSo: ~
marcos@ProjetoSo:~$ ./vmm random 10 < anomaly.dat
7
marcos@ProjetoSo:~$ ./vmm random 10 < anomaly.dat
8
marcos@ProjetoSo:~$ ./vmm random 10 < anomaly.dat
10
marcos@ProjetoSo:~$ ./vmm random 10 < anomaly.dat
8
marcos@ProjetoSo:~$ ./vmm random 10 < anomaly.dat
10
marcos@ProjetoSo:~$ ./vmm random 10 < anomaly.dat
9
marcos@ProjetoSo:~$ ./vmm random 10 < anomaly.dat
8
marcos@ProjetoSo:~$ ./vmm random 10 < anomaly.dat
8
marcos@ProjetoSo:~$ ./vmm random 10 < anomaly.dat
9
marcos@ProjetoSo:~$ ./vmm random 10 < anomaly.dat
9
marcos@ProjetoSo:~$
```

Tabela de Resultados

Execução	Page Fault FIFO	Page Fault RANDOM
1	9	7
2	9	8
3	9	10
4	9	8
5	9	10
6	9	9
7	9	8
8	9	8
9	9	9
10	9	9
MÉDIA	9	9,5

Analisando a execução dos dois algoritmos, podemos perceber que o fifo apresenta uma constância no page fault, enquanto o random varia entre 7 a 10 page fault. Foram executadas 10 vezes os códigos tendo como entrada o arquivo anomaly.dat. Por não possuir uma variedade maior de entradas, o resultado não foi tão discrepante entre a média de page fault do algoritmo proposto pelo professor e o desenvolvido pelos autores.

Pelas análises no ambiente de máquina virtual Azure, com sistema operacional ubuntu, pudemos identificar que o fifo foi ligeiramente melhor que o random.