UNIVERSIDADE FEDERAL DE VIÇOSA
***CAMPUS* RIO PARANAÍBA**
INSTITUTO DE CIÊNCIAS EXATAS
GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO
DISCIPLINA: SISTEMAS OPERACIONAIS (SIN351)
PROFESSOR MS. RODRIGO MOREIRA

RELATÓRIO TRABALHO PRÁTICO II: IMPLEMENTANDO UM GERENCIADOR DE MEMÓRIA

Amanda Oliveira Nascimento; 5149
Thalita Mendonça Antico; 5141
Vinicius Hideyuki Ikehara; 5191

RIO PARANAÍBA - MG
DEZEMBRO DE 2020

1. COMPILANDO E EXECUTANDO O CÓDIGO

Este projeto é utilizado para introduzir os conceitos de gerenciamento de memória, nesta pasta estão incluídos os seguintes arquivos: vmm.c, anomaly.dat, gerador.c, Readme e o relatório em PDF.

Como compilar e executar os códigos:

1º Passo: Abrir o terminal na pasta com todos os arquivos;

2º Passo: Escrever o código "gcc -Wall vmm.c -o vmm" e apertar *Enter*;

3º Passo: Para executar o código do algoritmo random, escrever o código "./vmm random 10 <anomaly.dat" e apertar *Enter*;

4º Passo: Para executar o código do algoritmo fifo, escrever o código "./vmm fifo 10 <anomaly.dat" e apertar *Enter*.

2. EXPLICANDO O CÓDIGO FIFO

O código do FIFO foi adicionado ao código-base disponibilizado pelo professor;

O método do FIFO tem seu funcionamento pelo preenchimento da RAM como forma de uma fila, ou seja, o primeiro que chegou, ocupará o primeiro lugar da RAM, o segundo na fila, o segundo lugar, e assim sucessivamente, e desta forma, quando a RAM estiver completa, a substituição da página será para o que está mais tempo na memória, como o próprio nome dá a entender First In First Out - " primeiro que entra, primeiro que sai".

O sistema operacional preserva e mantém uma determinada fila de páginas correntes na memória. Como já foi descrito acima, as páginas iniciais da fila corresponderão as páginas mais antigas, e as páginas ao final corresponderão as mais novas. Na ocorrência de um page fault, ou seja, quando uma página que não está na RAM é referenciada, a página contida no início é removida e a nova página é adicionada ao final da fila.

```
40
41 int fifo(int8_t** page_table, int num_pages, int prev_page, int fifo_frm, int num_frames, int clock) {
42         for(int page = 0; page < num_pages; page++){
43                 if(page_table[page][PT_FRAMEID] == fifo_frm){
44                         //printf("Pagina %d foi retirada\n",page);
45                 return page;
46         }
47           }
48
49     return -1;
50 }
```

Imagem 1: Código FIFO adicionado.

3. ANÁLISE DOS RESULTADOS

Foram realizados com os algoritmos de gerenciamento de memória FIFO e RANDOM, anomaly.dat disponibilizado pelo professor, é possível observar além do número de *page-faults* de cada algoritmo para cada uma das 10 execuções, foi observado pelos alunos o uso da biblioteca time.h que também permite o cálculo do tempo de execução dos algoritmos.

Tabela 1: Resultados dos testes realizados com os algoritmos de gerenciamento de memória FIFO e RANDOM.

Execuções	***Page Fault* FIFO**	**Tempo FIFO (s)**	***Page Fault* RANDOM**	**Tempo RANDOM (s)**
1	9	0.000078	7	0.000077
2	9	0.000083	9	0.000078
3	9	0.000060	8	0.000071
4	9	0.000062	8	0.000072
5	9	0.000071	6	0.000078
6	9	0.000069	7	0.000060
7	9	0.000071	6	0.000060
8	9	0.000075	9	0.000071
9	9	0.000077	9	0.000068
10	9	0.000069	7	0.000106

Fonte: os autores.

Observando a tabela 1 é possível analisar que o FIFO possui um número estável de *page-faults* até por conta da natureza do algoritmo, enquanto o random possui uma variação com média de 8 *page-faults* para a entrada padrão do anomaly.dat disponibilizado pelo professor. Em relação ao tempo de execução o FIFO novamente é a opção mais estável, mas isso não significa que tenha os melhores resultados, contendo uma média de 0.000071 segundos e 0.000007 segundos de desvio padrão. O tempo para o algoritmo Random possui 0.000071, ou seja, a mesma média do FIFO, porém 0.0000129 segundos de desvio padrão, o que é quase o dobro comparado ao FIFO.

Considerando o anomaly.dat disponibilizado pelo professor, o conjunto de páginas referenciadas do FIFO funcionou de maneira mais eficiente, podendo possivelmente evitar o *trashing.*

4. ADICIONANDO NOVOS ACESSOS

Como já foi dito na seção 1, o arquivo gerador.c desenvolvido pelos alunos, também está disponível, e este código será utilizado para criar um novo arquivo de anomalias, o grupo entrou em consenso de manter 10 páginas e 3 *frames* e aumentar apenas os acessos, então para compilar e executar o código gerador é necessário seguir os seguintes passos:

Como compilar e executar os códigos:
1º Passo: Abrir o terminal na pasta com todos os arquivos;
2º Passo: Escrever o código "gcc -Wall vmm.c -o vmm" e apertar *Enter*;
3º Passo: Escrever o código “gcc -Wall gerador.c -o gerador” e apertar *Enter*;
4º Passo: Para executar o gerador, escrever o código “./gerador” e apertar *Enter*;
5º Passo: O programa vai pedir para inserir o número de acessos, então insira o número de acessos e aperte *Enter*.
6º Passo: Para executar o código do algoritmo random, escrever o código "./vmm random 10 <anomalygerado.dat" e apertar *Enter*;
7º Passo: Para executar o código do algoritmo fifo, escrever o código "./vmm fifo 10 <anomalygerado.dat" e apertar *Enter*.

Tabela 2: Resultados dos testes realizados com os algoritmos de gerenciamento de memória FIFO e RANDOM com o anomalygerado.dat com 100 acessos.

Execuções	*Page Fault* FIFO	Tempo FIFO (s)	*Page Fault* RANDOM	Tempo RANDOM (s)
1	80	0.000122	75	0.000110
2	80	0.000119	81	0.000137
3	80	0.000117	80	0.000141
4	80	0.000133	78	0.000104
5	80	0.000101	77	0.000111
6	80	0.000127	76	0.000139
7	80	0.000112	71	0.000072
8	80	0.000127	75	0.000137
9	80	0.000133	74	0.000132
10	80	0.000112	72	0.000148

Fonte: os autores.

Observando a tabela 2 é possível analisar semelhanças ainda com relação a tabela 1, o random possui uma variação com média de 75.5 *page-faults* para a entrada do anomalygerado.dat com 100 acessos, sendo assim temos uma média inferior de *page-faults*

comparado ao FIFO. Em relação ao tempo de execução o FIFO novamente é a opção mais estável, mas isso não significa que tenha os melhores resultados, contendo uma média de 0.00012 segundos e 0.00001 segundos de desvio padrão. O tempo para o algoritmo Random possui 0.000013, ou seja, a mesma média do FIFO, porém 0.00002 segundos de desvio padrão, o que é novamente assim como foi observado na tabela 1, quase o dobro comparado ao FIFO.