

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

Departamento de Ciência da Computação

Sistemas Operacionais I

Professora: Valeria Bastos

Grupo: Gabriel Silva - DRE 115192431

Thamires Bessa - DRE 113032431



UFRJ

**Trabalho 2 - Simulação de Gerência de Memória Virtual**

## Introdução

Este trabalho tem por objetivo simular um escalonador de processos com gerência de memória virtual com paginação. A política de escalonamento é Round Robin com feedback e filas de prioridade. A estratégia de realocação é LRU (Least Recently Used) e tratamento de swapping. Os tempos de serviço, a quantidade de páginas virtuais de cada processo, e a quantidade de saídas para I/O de cada processo foram definidas de forma aleatória, utilizando uma distribuição uniforme para que não houvesse concentração de probabilidade dos números. Também definimos as estruturas de dados que seriam usadas para armazenar as informações de cada processo, assim como seus respectivos PCBs (Process Control Blocks): para as filas de prioridade e lista de processos escolhemos lista e iterando de forma FIFO; já para armazenar o PCB escolhemos dicionário; para a gerência de memória um array onde cada índice é um frame; definimos que a memória principal terá 64 frames. O working set limit para todos os processos é de 4 páginas virtuais por processo e este WSL é uma lista com apenas 4 posições.

## Estado da Arte

Dada a importância de uma base teórica sólida para a implementação prática deste trabalho, nos embasamos no livro-texto Operating Systems: Internals and Design Principles, de William Stallings para definir os conceitos utilizados.

Da gerência de memória e escalonamento de processos:

Frame - Um bloco de tamanho fixo na memória principal.

Página - Um bloco de dados de tamanho fixo que reside na memória secundária (como o disco). Uma página de dados pode ser temporariamente copiada para um frame da memória principal.

Processo-pai (Parent Process) - processo original, criador do novo processo.

Processo-filho (Child Process) - processo novo.

Process Control Block (PCB) - Estrutura de dados que armazena os elementos do processo e é criado e gerenciado pelo sistema operacional. Graças a ele é possível suporte a múltiplos processos. Para a finalidade da simulação implementada o PCB para cada processo conta com: PID (Process ID); PPID (Parent Process ID); status relativo a seu estado; prioridade; sua lista de pedidos de I/O. Os PIDs são incrementais a partir do PPID 8379 que é o primeiro a entrar na fila e possui tempo de serviço de 1u.t. (unidade de tempo); WSL (Working Set Limit); quantidade de páginas pertencentes àquele processo.

Gerência de Memória por paginação de memória virtual - A memória principal é dividida em um número de frames iguais e cada processo é dividido em um número de páginas e é carregado quando suas páginas são trazidas para frames disponíveis na memória principal. Ao utilizar memória virtual não é necessário carregar todas as suas páginas na memória principal, sendo assim, as páginas não residentes que são requisitadas são trazidas automaticamente depois. Dessa maneira temos um espaço de endereçamento virtual maior.

Tabela de páginas - é uma tabela mantida pelo sistema operacional que contém a localização do frame para cada página do processo e é utilizada pelo processador

para produzir um endereço físico. Para a finalidade da simulação implementada não há uma tabela de páginas de fato, mas há uma estrutura que, no controle do working set limit (WSL) do processo, gerencia o número da página, a sua localização na memória principal e o seu clock da última vez em que foi referenciado.

Fatia de tempo - Quantidade de tempo de processamento máximo que cada processo tem antes de ser preemptado.

Tempo de serviço - Tempo o qual o processo necessita de processamento para completar sua execução.

Saídas para I/O - Cada processo lista as suas saídas de I/O que necessita para completar sua execução.

Swapping - Ação de retirar as páginas de um processo da memória principal e passá-las para a memória secundária e vice-versa. Na nossa simulação quando um processo for swapped out da memória principal todas as suas páginas serão levadas. Ao ser swapped in para memória principal novamente todas as suas páginas também irão.

Process Spawning - Criação de processos.

Política de escalonamento Round Robin com feedback - Nessa política cada processo tem uma fatia de tempo máxima (fixa) antes de ser preemptado e funciona como uma fila circular. A estratégia de feedback separa os processos em filas de prioridade dando mais preferência aos que estão em filas mais prioritárias e organizando também a prioridade dos processos retornando de I/O para a fila de prontos.

Estados dos processos - Cada processo passa por pelo menos 4 (novo, pronto, executando, terminado) dos 7 estados durante sua execução, sendo eles:

- Novo - quando o processo é recém-criado e ainda não foi admitido pelo escalonador;

- Pronto - o processo está pronto para execução e aguarda ser despachado pelo sistema operacional;

- Executando - O processo está em execução pelo processador;

- Pronto/Suspenso - O processo está pronto para execução porém não há frame disponível na memória principal e ele se encontra na memória secundária.

- Bloqueado - O processo encontra-se bloqueado para execução da sua chamada de I/O.

- Bloqueado/Suspenso - O processo encontra-se bloqueado para execução da sua chamada de I/O e não está na memória principal.

- Terminado - O processo completou sua execução e está dispensado.

Política de substituição de páginas - Quando um processo está com o máximo de páginas carregadas na memória principal definido pelo Working Set Limit, é necessário substituir as páginas antigas pelas novas que estão sendo requisitadas. Sendo assim, existem algoritmos de substituição de páginas, entre eles LRU, usado nesta simulação. Tal estratégia visa retirar do working set a página que foi referenciada há mais tempo, não sendo necessariamente a mais antiga, mas sim a que está a mais tempo sem uso na memória principal. Esta técnica, de acordo com o princípio da localidade, evita uma frequência de page faults alta pois a chance de uma página requisitada estar em uso frequente é maior.

Page fault - A página do processo requerida não encontra-se em memória principal, sendo assim, o sistema operacional tem de buscá-la na memória secundária e carregá-la em um frame para poder executar as instruções.

A partir dos conceitos definidos anteriormente podemos então pontuar as premissas deste trabalho:

Memória principal como uma lista de 64 frames; Spawn (criação) de processos a cada 3 segundos até atingir o número máximo de 18 processos; Working Set Limit de 4 páginas para cada processo, nos permitindo até 16 processos carregados na memória principal; Cada processo pode ter no máximo 64 páginas que será um número aleatório com distribuição uniforme; quando um processo é suspenso (swapped out) todas as suas páginas são levadas para a memória secundária e quando ele é trazido de volta para a memória (swapped in) todas as páginas retornam e não apenas a referenciada; A política de realocação de páginas é LRU; a fatia de tempo para o escalonador é 5 u.t.; tempo de serviço é um número aleatório entre 0 e 100 distribuído uniformemente; os tempos de I/O para impressora, disco e fita são respectivamente 15 u.t., 1 u.t. e 5 u.t.; a quantidade de pedidos de I/O por processo é também uma variável aleatória uniformemente distribuída entre 0 e 5; existem 3 filas, uma de alta prioridade, uma de baixa prioridade e uma de I/O; processos novos, retornando de I/O de fita e impressora vão para a fila de alta prioridade e processos preemptados e retornando de I/O de disco vão para a fila de baixa prioridade.

## Metodologia

A implementação do simulador de escalonamento de processos com gerência de memória foi feita em linguagem Python com uso de diversas bibliotecas e funções para emular o trabalho de um sistema operacional.

Para conseguir que a quantidade do tempo de serviço fosse aleatória, criamos a função *create\_process(amount)* onde além da criação do processo, temos também a hora randômica de chegada, o tempo de serviço e o seu respectivo PCB.

Já para os tempos de I/O tem-se a função *randomize\_ios(service\_time)* que nos retorna a quantidade de saídas para I/O aleatória entre 0 e 5, quais são os respectivos tipos de I/O e quando essa saída acontecerá durante o tempo de serviço do processo.

De forma análoga, para obter o número de páginas que um processo terá durante sua execução, temos a função *randomize\_pages()* que nos retorna um número aleatório uniformemente distribuído entre 1 e 64.

Para fazer a geração da sequência de páginas dado a quantidade de páginas existentes e o tempo de execução do processo temos a função *generate\_sequence\_of\_pages(q\_pages, s\_time)* nos retorna essas informações em formato de lista.

Para que os processos sejam devidamente despachados e executem durante sua fatia de tempo, temos a função *schedule\_next\_process()* que imprime a informação de qual processo está em execução e proveniente de qual fila de prioridade assim como seu respectivo status.

Assim que um processo encerra sua fatia de execução no processador chamamos a função *unschedule\_current\_process(pid)* para que ele tenha seu status alterado para pronto, bloqueado ou terminado.

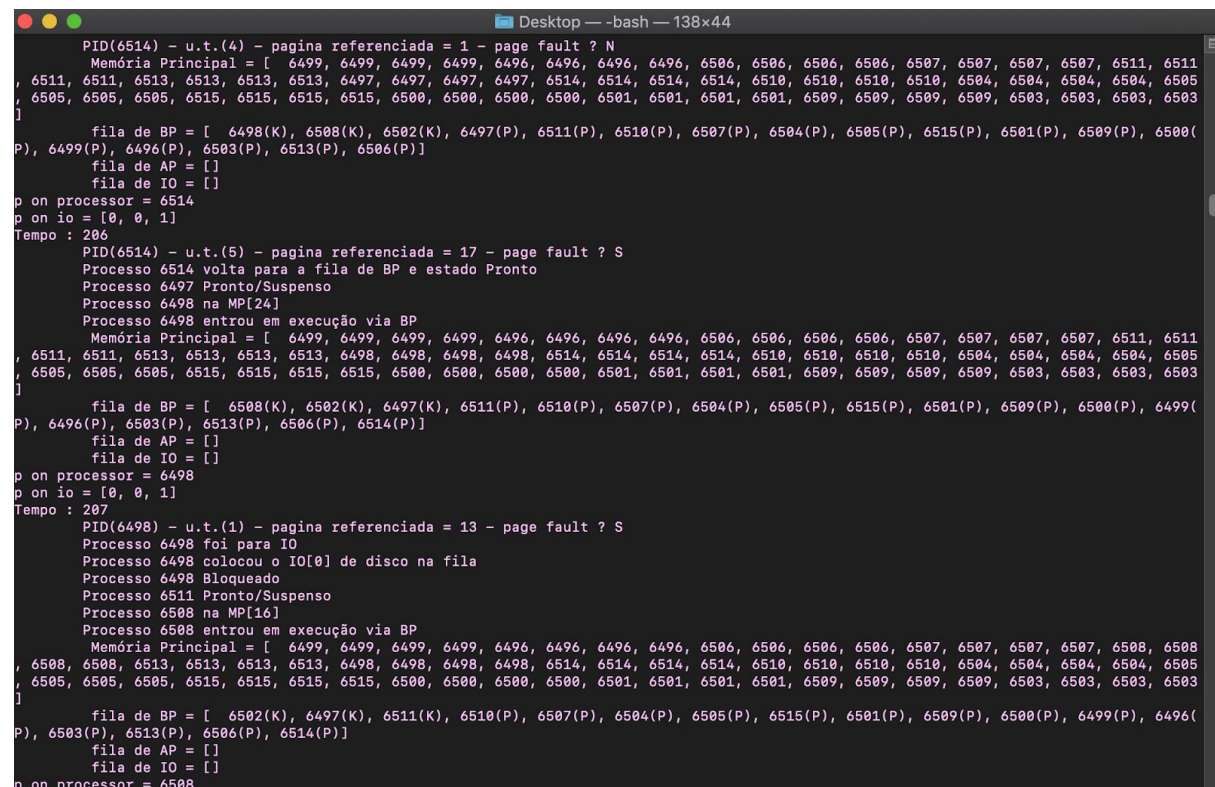
Analogamente, para que os processos sejam despachados para seus respectivos serviços de I/O pelo tempo de duração determinado, executa-se a função *dispatch\_process\_io(pid)*, já para que a fila de I/O seja atualizada e os processos possam ser escalonados para I/O, a função *update\_io\_queue()* é chamada.

Como a política de substituição de páginas é LRU, precisamos manter uma etiqueta para cada página a fim de saber quando foi referenciada por último. Sendo assim, temos a função *step\_current\_process(pid)* que atualiza o estado do processo dado o ciclo de clock e a função *check\_lru(kiwi, status)*. E, conseqüentemente, para fazer a realocação das páginas do working set list do processo, chamamos a função *update\_WSL(pid)*.

Para fazer a gerência de swap, chama-se a função *do\_swap\_in(pid)* que verifica através do algoritmo LRU quais páginas devem ser substituídas e realiza a ação de trazer as páginas para a memória principal. Já para retirar da memória por falta de espaço, chama-se a função *do\_swap\_out(pid)* que faz uma chamada a função *check\_if\_mem\_is\_full()* para checar o status da memória.

Finalmente, para verificar se todos os processos terminaram sua execução, chama-se a função *test\_services\_times()* que encerra a simulação.

Exemplos do programa rodando:



```
PID(6514) - u.t.(4) - pagina referenciada = 1 - page fault ? N
Memória Principal = [ 6499, 6499, 6499, 6499, 6496, 6496, 6496, 6496, 6506, 6506, 6506, 6506, 6507, 6507, 6507, 6507, 6511, 6511
, 6511, 6511, 6513, 6513, 6513, 6513, 6497, 6497, 6497, 6497, 6514, 6514, 6514, 6514, 6510, 6510, 6510, 6510, 6504, 6504, 6504, 6504, 6505
, 6505, 6505, 6505, 6515, 6515, 6515, 6515, 6500, 6500, 6500, 6500, 6501, 6501, 6501, 6501, 6509, 6509, 6509, 6509, 6503, 6503, 6503, 6503
]
fila de BP = [ 6498(K), 6508(K), 6502(K), 6497(P), 6511(P), 6510(P), 6507(P), 6504(P), 6505(P), 6515(P), 6501(P), 6509(P), 6500(P)
P), 6499(P), 6496(P), 6503(P), 6513(P), 6506(P)]
fila de AP = []
fila de IO = []
p on processor = 6514
p on io = [0, 0, 1]
Tempo : 206
PID(6514) - u.t.(5) - pagina referenciada = 17 - page fault ? S
Processo 6514 volta para a fila de BP e estado Pronto
Processo 6497 Pronto/Suspensão
Processo 6498 na MP[24]
Processo 6498 entrou em execução via BP
Memória Principal = [ 6499, 6499, 6499, 6499, 6496, 6496, 6496, 6496, 6506, 6506, 6506, 6506, 6507, 6507, 6507, 6507, 6511, 6511
, 6511, 6511, 6513, 6513, 6513, 6513, 6498, 6498, 6498, 6498, 6514, 6514, 6514, 6514, 6510, 6510, 6510, 6510, 6504, 6504, 6504, 6504, 6505
, 6505, 6505, 6505, 6515, 6515, 6515, 6515, 6500, 6500, 6500, 6500, 6501, 6501, 6501, 6501, 6509, 6509, 6509, 6509, 6503, 6503, 6503, 6503
]
fila de BP = [ 6508(K), 6502(K), 6497(K), 6511(P), 6510(P), 6507(P), 6504(P), 6505(P), 6515(P), 6501(P), 6509(P), 6500(P), 6499(P)
P), 6496(P), 6503(P), 6513(P), 6506(P), 6514(P)]
fila de AP = []
fila de IO = []
p on processor = 6498
p on io = [0, 0, 1]
Tempo : 207
PID(6498) - u.t.(1) - pagina referenciada = 13 - page fault ? S
Processo 6498 foi para IO
Processo 6498 colocou o IO[0] de disco na fila
Processo 6498 Bloqueado
Processo 6511 Pronto/Suspensão
Processo 6508 na MP[16]
Processo 6508 entrou em execução via BP
Memória Principal = [ 6499, 6499, 6499, 6499, 6496, 6496, 6496, 6496, 6506, 6506, 6506, 6506, 6507, 6507, 6507, 6507, 6508, 6508
, 6508, 6508, 6513, 6513, 6513, 6513, 6498, 6498, 6498, 6498, 6514, 6514, 6514, 6514, 6510, 6510, 6510, 6510, 6504, 6504, 6504, 6504, 6505
, 6505, 6505, 6505, 6515, 6515, 6515, 6515, 6500, 6500, 6500, 6500, 6501, 6501, 6501, 6501, 6509, 6509, 6509, 6509, 6503, 6503, 6503, 6503
]
fila de BP = [ 6502(K), 6497(K), 6511(K), 6510(P), 6507(P), 6504(P), 6505(P), 6515(P), 6501(P), 6509(P), 6500(P), 6499(P), 6496(P)
P), 6503(P), 6513(P), 6506(P), 6514(P)]
fila de AP = []
fila de IO = []
p on processor = 6508
```

## Conclusão

A partir dos resultados obtidos podemos concluir que o nosso simulador retornou resultados satisfatórios ao emular um Sistema Operacional escalonando processos com política Round Robin com feedback e com gerência de memória virtual por paginação. Os processos seguiram a política correta de mudança de estados devido às saídas para I/O e por preempção e também seguindo a ordem circular e suas respectivas filas. Já a gerência

de memória executou como esperado fazendo as mudanças de estados pertinentes e respeitando as restrições impostas pelo projeto da simulação.

## **Referências**

Operating Systems: Internals and Design Principles, de William Stallings

Sobre velocidade da impressora

<https://hp-laserjet-4100-printer-series.printerdoc.net/en/control-panel-menus-2/i-o-menu/>

Sobre velocidade do disco

[https://en.wikipedia.org/wiki/Hard\\_disk\\_drive\\_performance\\_characteristics](https://en.wikipedia.org/wiki/Hard_disk_drive_performance_characteristics)

Sobre velocidade da fita magnética

<https://www.techopedia.com/definition/8213/magnetic-tape-drive>

Bibliotecas

[https://matplotlib.org/3.1.0/gallery/lines\\_bars\\_and\\_markers/broken\\_barh.html#sphx-glr-gallery-lines-bars-and-markers-broken-barh-py](https://matplotlib.org/3.1.0/gallery/lines_bars_and_markers/broken_barh.html#sphx-glr-gallery-lines-bars-and-markers-broken-barh-py)

<https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.uniform.html>

<https://docs.python.org/2/library/collections.html>

<https://docs.python.org/2/tutorial/inputoutput.html#reading-and-writing-files>