Universidade Federal de Viçosa – Campus Florestal
Ciência da Computação – Sistemas Operacionais
Professor: Daniel Mendes Barbosa
Trabalho Prático 3 – Data de entrega: ver PVANet

Este trabalho deverá ser feito pelos mesmos grupos do trabalho prático anterior. Cada grupo deverá entregar um **relatório** através do PVANet, contendo uma **breve documentação e decisões importantes de cada projeto (tarefas A e B)**, além de todo os códigos dos projetos, implementados na linguagem C e no sistema operacional Linux. Deve também conter uma **análise** e **discussão** dos resultados obtidos com as simulações. A forma de analisar e discutir os resultados é de responsabilidade do grupo.

Tarefa A: simulação de alocação de memória principal

Objetivo: Simular e avaliar diferentes técnicas de alocação/desalocação de memória: *first fit, next fit, best fit e worst fit*, quando uma lista encadeada é usada para manter o registro de uso de memória.

Suponha que a memória seja de 256 KB e que esteja dividida em unidades de 2 KB cada. Um processo pode requisitar entre três e dez unidades de memória. A sua simulação consiste de três componentes: componente de memória que implementa uma técnica específica de alocação/desalocação; componente de geração de requisição que gera requisições de alocação/desalocação; e componente de relatórios estatísticos que imprime as estatísticas relevantes. Componente de memória fornece no mínimo as seguintes funções:

1 . `int allocate_mem (int process_id, int num_units)`: aloca `num_units` unidades de memória para o processo, cujo identificador é `process_id`. Se for bem-sucedida, a função retorna o número de nós percorridos na lista encadeada, caso contrário retorna -1.

2. `int deallocate_mem (int process_id)`: desaloca a memória alocada para o processo, cujo identificador é `process_id`. Se bem-sucedida, retorna 1, caso contrário, retorna -1.

3. `int fragment_count()`: retorna o número de buracos (fragmentos de tamanho de uma ou duas unidades).

Você irá implementar um componente de memória separado para cada técnica de alocação/desalocação de memória. O componente de geração de requisição gera requisições de alocação e desalocação. Para requisições de alocação, o componente especifica o identificador de processo do processo para o qual a memória está sendo requisitada, assim como o número de unidades de memória sendo requisitadas. Para essa simulação, suponha que a memória é requisitada para cada processo, apenas uma vez. Para requisições de desalocação, o componente especifica o identificador de processo do

processo, cuja memória deve ser desalocada. Para essa simulação, suponha que toda a memória alocada para o processo é desalocada durante uma requisição de desalocação. Você pode gerar essas requisições com base em algum critério específico, por exemplo, de forma aleatória ou a partir de alguma marcação de alocação/desalocação de memória de alguma fonte.

Existem três parâmetros de desempenho que a sua simulação deve calcular para todas as quatro técnicas:

- número médio de fragmentos externos;

- tempo médio de alocação em termos de número médio de nós percorridos na alocação;

- o percentual de vezes que uma requisição de alocação é negada.

Gere 10 mil requisições, usando o componente de geração de requisições, e para cada requisição invoque a função apropriada do componente de memória para cada uma das técnicas de alocação/desalocação de memória. Após cada requisição, atualize os três parâmetros de desempenho para cada uma das técnicas.

O componente de relatórios estatísticos imprime o valor dos três parâmetros para todas as quatro técnicas no final.

Tarefa B: simulação rudimentar de memória virtual.

Objetivo: Simular e avaliar 4 diferentes algoritmos de substituição de páginas.

Considere em sua simulação simplificada que não há segmentação e que há um número M de molduras de páginas, numeradas de 0 a M – 1, e um número P de páginas virtuais, numeradas de 0 a P-1.

Haverá um arquivo de entrada para cada simulação a ser feita, contendo em sua primeira linha os valores de M e P, nesta ordem, e separados por um espaço. A partir da segunda linha, o arquivo terá um comando por linha para o simulador, onde cada comando irá simular os acessos à memória. Esses comandos são R (leitura) e W (escrita), seguidos de um número, que será o número da página virtual a ser acessada (também uma simplificação, não tendo o endereço exato da palavra a ser acessada).

Com base em cada comando, seu simulador deverá trazer a página para uma moldura (memória principal), caso ela não esteja. Se todas as molduras estiverem ocupadas, deverá ser usado um algoritmo de substituição de página para escolher a vítima (página a ser substituída). O algoritmo a ser usado, deverá ser escolhido no próprio programa, antes do início da simulação. Ao final da simulação deverá ser exibido o número de faltas de página ocorridas durante a simulação.

Você deverá deixar o mais claro possível na sua documentação as estruturas de dados utilizadas, e os motivos de cada decisão tomada.

Deverão ser criados arquivos de teste (pode ser pensado inclusive um programa auxiliar gerador destes arquivos, que pode ser documentado) e mostrados os resultados para cada algoritmo.