Sistemas Operacionais

Prof. Robson de Souza

Aulas 29 e 30

Conteúdo: Gerência de memória: Algoritmos de substituição de página e Segmentação

Sabendo que a memória secundária é maior que a memória principal e os dados dos programas devem ser trazidos das páginas da memória secundária para a memória principal, fica claro que, em determinado momento a memória pode estar cheia e ao tentar alocar uma nova página, não exista espaço para a mesma. Nesse caso, é necessário substituir alguma página da memória principal pela nova página a ser alocada.

Para decidir qual página deve sair da memória existem diversos algoritmos que podem ser utilizados, uma opção seria escolher aleatoriamente uma página e fazer a substituição, porém isso não seria eficiente.

O **algoritmo ótimo** de substituição de página substitui a página que demorará mais tempo para ser referenciada. Isso é bom porque se a página substituída for muito referenciada, é bem provável que ela tenha de retornar à memória em pouco tempo. O problema desse algoritmo é que é impossível saber o futuro, mas ele pode ser utilizado como parâmetro para comparação.

Algoritmo aleatório

Esse algoritmo escolhe e remove qualquer página, sem nenhum critério de decisão. A sua implementação é bastante simples, porém o resultado é pouco eficiente.

Algoritmo de substituição de página Não Recentemente Utilizada - Not Recently Used (NRU)

Esse algoritmo usa dois bits de status: bit R (referenciado) e bit M (modificado). Quando o processo inicia, suas páginas ainda não estão presentes na memória. Assim que uma delas é referenciada, o bit R é colocado em 1. Em seguida, se esta página é modificada, o bit M é colocado em 1. Periodicamente o bit R é limpo (Por exemplo, a cada interrupção de relógio) para distinguir páginas que não foram referenciadas recentemente das que foram. Ao ocorrer uma page fault (falha de página), o sistema operacional separa todas as páginas em quatro categorias:

Classe 0: não referenciada, não modificada.
Classe 1: não referenciada, modificada.
Classe 2: referenciada, não modificada.
Classe 3: referenciada, modificada.

Embora as páginas de classe 1 pareçam impossíveis à primeira vista, elas ocorrem quando uma página de classe 3 tem seu bit R limpo por uma interrupção de clock. A interrupção de clock não limpa o bit M porque essa informação é necessária para saber se a página tem de ser regravada no disco ou não.

Com essas informações, o NRU remove uma página aleatória da classe mais baixa que não
esteja vazia. Entre as vantagens está a baixa complexidade de entendimento e implementação, além da boa aproximação para o algoritmo ótimo.

Utilizando esse algoritmo fica implícito que é melhor remover uma página que foi modificada mas não foi referenciada em um período de tempo, do que remover uma página limpa que é constantemente referenciada.

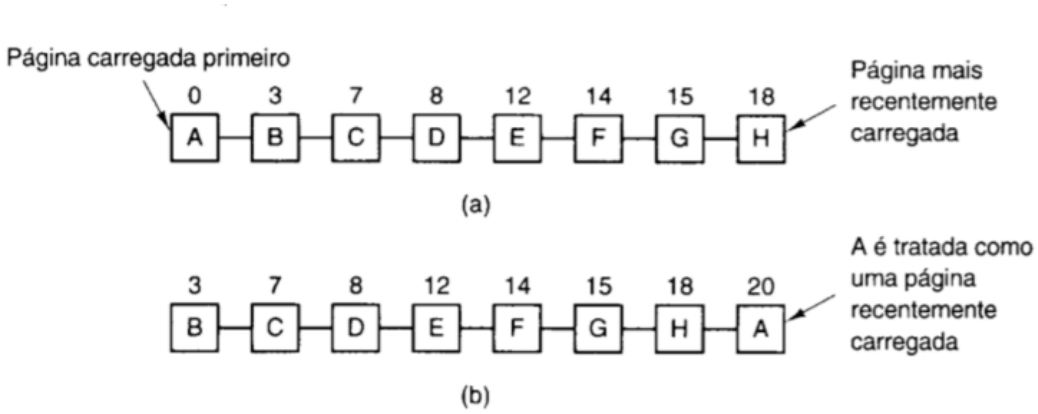
Primeira a Entrar, Primeira a Sair (FIFO)

Nesse algoritmo, o sistema operacional mantém uma lista de todas as páginas atualmente na memória, sendo que a página no topo da lista é a mais antiga e a página no fim é a mais recente. Em uma falha de página, a página no topo e removida e a nova página é adicionada no fim da lista.

Esse algoritmo é extremamente simples de entender e implementar, porém, pode ser que uma página muito referenciada esteja no início da lista e consequentemente seja substituída, tendo que ser alocada novamente em um curto período de tempo. Por essa razão, esse algoritmo na sua forma pura é raramente utilizado.

Algoritmo da segunda chance

Esse algoritmo utiliza o FIFO, porém, ao invés da sua forma pura, usa o bit R da página antes de substituí-la. O funcionamento se dá de modo que se inspeciona o bit R da página mais velha (a primeira da fila). Se for 0, ela é velha e não foi usada recentemente, com essa informação fica claro que pode ser trocada. Se o bit R for 1, significa que ela foi utilizada recentemente, logo, o bit é feito 0 e a página é colocada no final da fila, seu tempo de carga é modificado fazendo parecer que recém chegou na memória (recebe uma segunda chance). A busca continua normalmente.

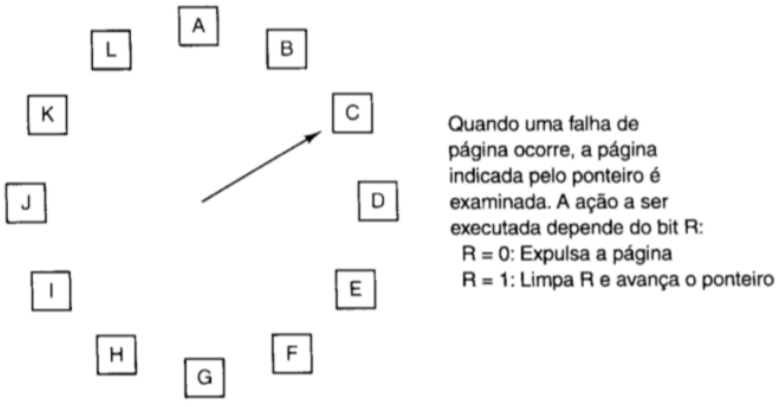


Basicamente, o que esse algoritmo está fazendo é procurar uma página antiga que não tenha sido recentemente referenciada. Se todas as páginas foram referenciadas, o algoritmo da segunda chance se torna um algoritmo FIFO puro.

Algoritmo do relógio

O algoritmo da segunda chance é ineficiente pois ele frequentemente fica deslocando elementos na lista. Uma abordagem melhor é manter todas as páginas em uma lista circular na forma de um relógio. Assim, quando houver page fault, a página indicada pelo ponteiro é examinada. Se o bit R estiver em 0, ela é substituída e o ponteiro avança. Se o bit R for 1, é zerado e o ponteiro avança fazendo uma nova busca. Quando chega ao fim sem encontrar bit R = 0, o ponteiro retorna ao início e essa página então é eliminada.

Fica claro que, embora esse algoritmo pareça diferente do algoritmo da segunda chance, os dois se diferem apenas na implementação e no fato de que o algoritmo do relógio não “desloca” as páginas. Fora isso é a mesma coisa.



Algoritmo de substituição de página Menos Recentemente Utilizada - Least Recently Used (LRU)

O LRU (Least Recently Used - usada menos recentemente) parte do princípio que as páginas usadas com mais frequência nas ultimas execuções provavelmente serão muito utilizadas novamente.
Desse modo, quando houver page fault o algoritmo elimina a página não utilizada pelo período de tempo mais longo.

Embora esse algoritmo seja possível de implementar e aparente ter um bom desempenho, ele possui um custo computacional muito alto, pois para implementar completamente o LRU, é necessário manter uma lista encadeada de todas as páginas na memória, com as páginas mais recentemente utilizadas na frente e as menos recentemente utilizadas no fundo. A dificuldade é que a lista deve ser atualizada a cada referência de memória. A operação de localizar uma página na lista, excluí-la e então movê-la para frente consome muito tempo.

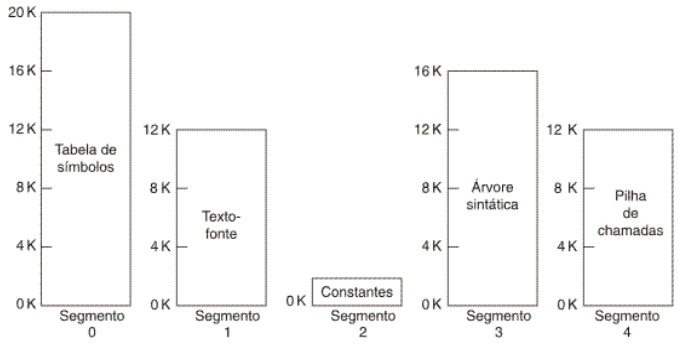
O modo mais simples de se implementar o algoritmo LRU é usar um um contador, que é incrementado a cada instrução. A página referenciada necessita de um novo campo, que recebe o valor do contador no momento de sua referência. Quando houver page fault, o hardware examina as páginas em busca do menor valor de contador, a página “usada menos recentemente”.

Segmentação

No sistema de paginação, todas as páginas possuem exatamente o mesmo tamanho, o que significa que se uma página tiver 512 KB, e um processo necessitar de apenas 32 KB, ele desperdiçará mais memória que um processo que precisa de 512 KB.

O sistema operacional pode utilizar um outro mecanismo, chamado de **segmentação**, nesse caso, o espaço de endereçamento virtual é dividido em blocos de tamanhos diferentes chamados segmentos. Na segmentação existe uma relação entre a lógica do programa e sua alocação na memória principal.

Nesse caso se utilizam as tabelas de segmentação e essa técnica permite que as tabelas aumentem ou encolham conforme a necessidade.



Campos da entrada da tabela de segmentação:

Campo	Descrição
Tamanho	Especifica o tamanho do segmento.
Bit de validade	Indica se o segmento está na memória principal.
Bit de modificação	Indica se o segmento foi alterado.
Bit de referência	Indica se o segmento foi recentemente referenciado, sendo utilizado pelo algoritmo de substituição.
Proteção	Indica a proteção do segmento.

Referências bibliográficas:

TANENBAUM, Andrew. 2ª ed. **Sistemas Operacionais Modernos,** Editora Pearson, 2003.

SILBERSCHATZ, Abraham. **Sistemas Operacionais com JAVA,** 6ª ed. Editora Campus

MACHADO, Francis B. **Arquitetura de Sistemas Operacionais,** 4ª ed, LTC, 2007.