MAC–IME–USP

Carlos Eduardo Ferreira
Sala 108C Tel.: 3091 6079
e-mail cef@ime.usp.br
monitor Leonardo B. Facci
e-mail lbfacci@ime.usp.br

MAC 323 - Estruturas de Dados

Primeiro semestre de 2007

Gerenciamento de memória – **Entrega: 1/6/2007**

Em um sistema operacional o gerenciamento da memória é um dos aspectos mais fundamentais, e organizá-lo de forma eficiente é muito importante para o desempenho do sistema. Nesse exercício–programa vamos implementar um sistema de gerenciamento de memória. O objetivo **não** é mostrar a estrutura utilizada nos sistemas mais modernos (e, de fato, a estrutura mostrada não é a mais utilizada, como vocês vão aprender na disciplina de Sistemas Operacionais). Bons algoritmos de gerenciamento de memória são tema de pesquisa interessante, envolvendo problemas como partilhamento da memória, coleta de lixo, etc. (veja o capítulo 12 do livro [1]).

A seguir descrevemos o esquema de gerenciamento de memória a ser implementado.

O sistema será de multiprocessamento, ou seja, diversos processos compartilharão a memória. O sistema não é capaz de manipular pedaços não contíguos de memória. Assim, se um processo deseja uma quantidade de memória que não está disponível no momento, fica suspenso até que seja possível alocar uma área daquele tamanho. Os tamanhos dos blocos requisitados podem variar (são estabelecidos pelos processos).

A entrada do seu programa deverá ser uma lista de pedidos da seguinte forma (use este padrão para a entrada a fim de facilitar a correção):

```
N <tamanho>
A <tamanho>
D <endereço> <tamanho>
I
```

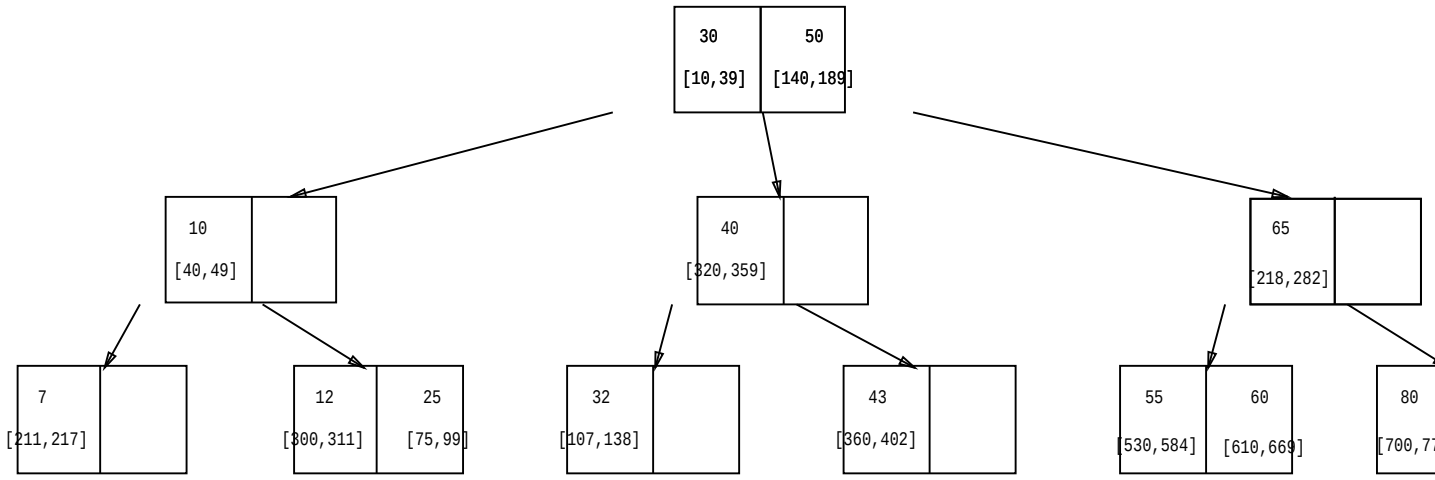
Se o pedido foi um `A` o processo está alocando `<tamanho>` bytes, se for um `D`, ele está devolvendo a porção de memória de `<tamanho>` bytes a partir de `<endereço>`. O comando `N` (que deve ser o primeiro na entrada) indica o tamanho total de memória disponível inicialmente, e o comando `I` causa a impressão dos blocos de memória, ordenados por tamanho e depois pelo endereço no seguinte formato:

```
<tamanho do bloco> <endereço de início> <endereço de fim>
```

No final da impressão deverá ter uma linha em branco.

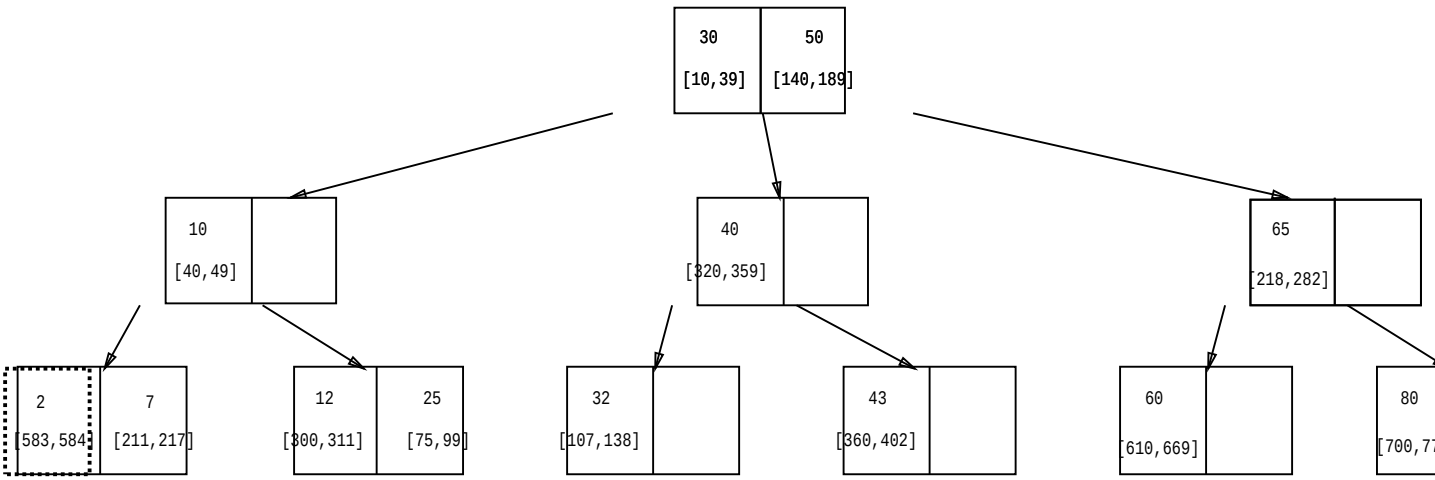
Os pedaços disponíveis de memória serão armazenados em uma árvore 2-3 (uma B-árvore em que armazenamos no máximo 2 elementos por nó), em que a chave será o tamanho em bytes do segmento (no caso em que dois segmentos têm o mesmo tamanho, use o menor endereço de início do segmento para desempate).

Observe o exemplo abaixo.



Quando requisitamos uma porção de memória, que é possível de se satisfazer no momento, uma remoção e, em geral, uma inserção serão necessárias. Veja o exemplo a seguir, em que alocamos o bloco requerido do menor bloco disponível maior que o pedido feito.

Ex.: Usando a árvore do exemplo anterior, pedimos um bloco de 53 kbytes.



Quando um bloco é devolvido, em geral é muito ineficiente tentar juntá-lo com blocos vizinhos, pois isso acarreta remoções e inserções na árvore. Assim, faremos esse rearranjo apenas quando tivermos algum pedido que não pôde ser satisfeito.

Seu exercício–programa pode implementar outras estratégias para escolha do bloco de que alocar a memória requisitada, e pode utilizar a estrutura que desejar para facilitar o rearranjo de memória.

Você deverá fazer um pequeno relatório mostrando o desempenho do seu sistema para os testes que você fez (por exemplo, quantas vezes foi necessário fazer o rearranjo, quantos processos ficaram suspensos esperando memória, etc). Durante a execução emita mensagens informando o usuário qual o estado do gerenciador.

Referências

[1] A.V. Aho, J.E. Hopcroft, J.D. Ullman, *Data Structures and Algorithms*, Addison-Wesley (1985).