Carlos E. Ferreira

cef@ime.usp.br

MAC 323 – Estruturas de Dados – 1998

Recuperação – Entrega: **11 de agosto de 1998**

Sistema de Multiprocessamento

A nota de recuperação da disciplina MAC 323 – Estruturas de Dados será dada através da entrega de um projeto sobre o funcionamento de um sistema de computação de multiprocessamento. O aluno deverá marcar comigo por e-mail uma data de apresentação do projeto, na qual darei a nota ao mesmo. A média final da disciplina será dada pela média ponderada entre a nota final do semestre (peso 1) e a nota dada a esta apresentação (peso 2).

O objetivo desse projeto é simular o funcionamento de um sistema de computação com multiprocessamento. Com este projeto teremos coberto uma (pequena) parte dos problemas enfrentados quando desejamos implementar um sistema de computação real, e teremos aplicado algumas das estruturas de dados vistas no curso para resolver os problemas envolvidos.

1 Filas das classes

Cada processo executado no computador pertence a uma determinada classe (1 ou 2). Cada classe tem permissões diferentes com relação ao limite máximo de tempo de execução na CPU (em UT: unidades de tempo), limite máximo de uso de memória e número máximo de linhas a serem impressas. Essas restrições estão descritas na tabela a seguir:

Classe	CPU (UT)	Memória (MB)	Linhas
1	10	2	500
2	40	16	10000

Quando um processo inicia sua execução, ele é colocado na fila de sua classe, e espera lá até que haja uma vaga na fila da CPU. Para efeito dessa simulação você deve "gerar" aleatoriamente um novo processo a cada UT, com probabilidade 70% que seja de classe 1 e 30% de classe 2. Ao "criar" o processo, gere aleatoriamente o tempo de CPU, a memória utilizada e o número de linhas que esse processo irá imprimir (dentro dos limites de sua classe).

As regras para um processo sair da fila de sua classe e ir para a fila da CPU são as seguintes. A fila da CPU comporta no máximo 10 processos. Um processo da classe 1 tem sempre prioridade. Um processo da classe 2 só entra na fila da CPU se a fila da classe 1 estiver vazia ou os últimos 4 processos que entraram na fila foram da classe 1.

2 Fila da CPU

Cada processo pode permanecer apenas um determinado tempo rodando na CPU. Quando esse tempo é ultrapassado, ele é suspenso e retorna para a última posição da fila da CPU. Com isso, um processo com grande tempo de execução necessita de vários ciclos da CPU para ser executado.

A fila da CPU comporta no máximo 10 processos. O ciclo da CPU será de 3 UT. A fila da CPU deverá ser implementada como uma lista circular duplamente ligada com cabeça de lista, como descrita em aula.

3 Fila das impressoras

O modelo a ser simulado segue o esquema de fila única para as impressoras (que são 5), cada impressora com velocidade de 10 linhas por UT. Quando uma impressora fica disponível, o primeiro processo da fila passará à impressão, a menos que algum processo já esteja no sistema há mais de 100 UT.

Os processos na fila da impressora estarão em duas filas ao mesmo tempo: uma por ordem de ingresso na fila e outra pelo tempo de permanência no sistema. Estas filas devem ser implementadas de modo a permitir que inserções em ambas as filas e remoção do primeiro elemento de cada fila possam ser feitas em tempo $O(\log n)$ (utilize fila de prioridade, como no EP do aeroporto).

4 Gerenciamento de memória

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. O objetivo aqui **não** é mostrar a estrutura utilizada nos sistemas mais modernos (e, de fato, a estrutura mostrada não é a mais utilizada, como vocês podem ver em livros de Sistemas Operacionais [2]). 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]).

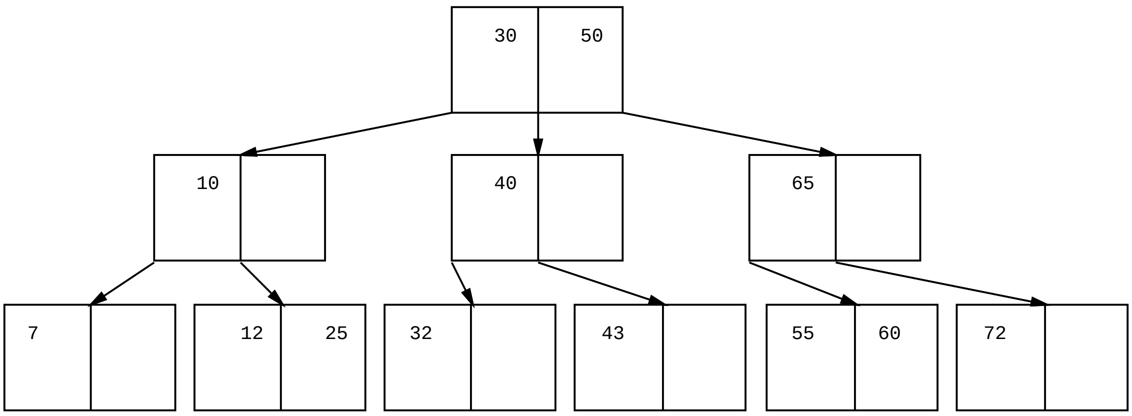
A seguir descrevemos o esquema de gerenciamento de memória a ser implementado.

O sistema será de multiprocessamento, ou seja, diversos processos compartilharão a memória. No início do processamento toda a memória (256MB) está disponível. 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 (volta para o fim da fila da CPU) até que seja possível alocar uma área daquele tamanho. Os tamanhos dos blocos requisitados podem variar (são estabelecidos pelos processos – gerados aleatoriamente). Assim, na fila da CPU teremos processos que estão executando (e portanto ocupando uma parte da memória) e outros que estão suspensos até que haja memória suficiente para executar.

Os pedaços disponíveis de memória estarão 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). Assim, cada nó da árvore 2-3 terá o seguinte formato:

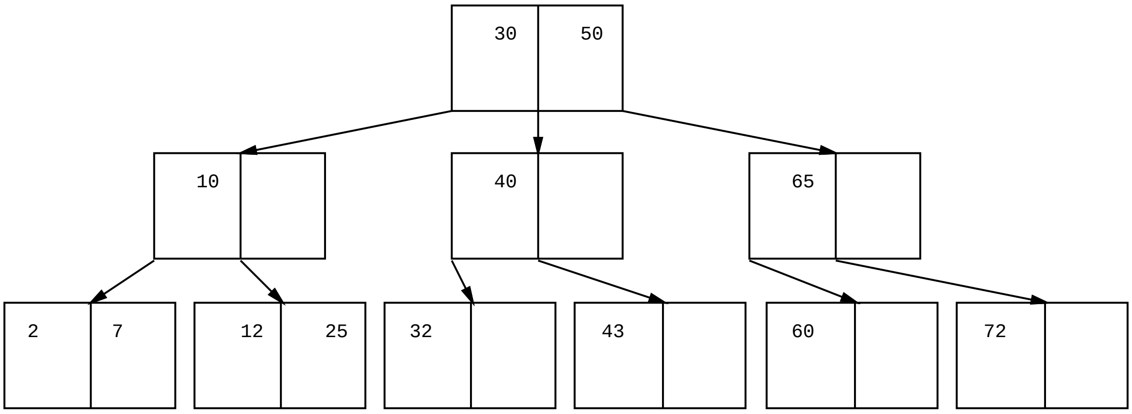
P_1	K_1	$[I_1, F_1]$	P_2	K_2	$[I_2, F_2]$	P_3

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, um processo requisita 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 projeto 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.

5 Saída do programa

A saída do seu programa deverá ser clara. A cada instante deverá ser fácil ver quantos processos estão em cada uma das filas, quais processos estão ocupando a CPU e as impressoras, qual o estado da memória, etc.

Ao finalizar a execução de um processo, as seguintes informações deverão ser impressas:

(a) características do processo (classe, tempo de CPU, número de linhas);

(b) tempo total de permanência no sistema;

(c) tempo gasto nas filas;

(d) razão entre o tempo útil (tempo de execução na CPU) e tempo total de permanência no sistema;

Ao término da execução, imprima o número total de processos executados, assim como média dos itens (b), (c) e (d) em cada classe. E execução deve continuar até que uma tecla "Pare" seja acionada.

Lembre-se que a saída e documentação do seu programa também serão avaliadas para efeito de nota.

Referências

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

[2] Tanenbaum, *Operating Systems: Design and Implementation*, Prentice-Hall, Englewood Cliffs (1987).