Projeto Escalonador

1. Linguagem

Esse projeto deve ser desenvolvido em linguagem Java.

2. Interface Gráfica

É obrigatório a implementação da interface gráfica.

3. Tipos de Processo

Dois tipos de processo. **CPU Bounds e I/O Bounds.** Os processos Cpu Bounds executam durante o tempo definido no quantum. Os processos I/O Bounds executam X quantum do tempo definido no quantum até serem bloqueados. Depois, entram no processo de I/O por Y tempo. O sistema deve oferecer mecanismos para o usuário definir o valor X e o Y na interface gráfica.

4. Configurações

Devem permitir ajustes na configuração: tempo do quantum, tempo de chaveamento e percentual de tempo para I/O Bounds.

5. Algoritmos de Escalonamento

Round-Robin: Um dos algoritmos mais antigos, simples, justos e amplamente usados é o circular (round-robin). A cada processo é designado um intervalo, chamado de seu quantum, durante o qual ele é deixado executar. Se o processo ainda está executando ao fim do quantum, a CPU sofrerá uma preempção e receberá outro processo. Se o processo foi bloqueado ou terminado antes de o quantum ter decorrido, o chaveamento de CPU será feito quando o processo bloquear, é claro. O escalonamento circular é fácil de implementar. Tudo o que o escalonador precisa fazer é manter uma lista de processos executáveis, como mostrado na Figura 2.42(a). Quando o processo usa todo o seu quantum, ele é colocado no fim da lista, como mostrado na Figura 2.42(b).

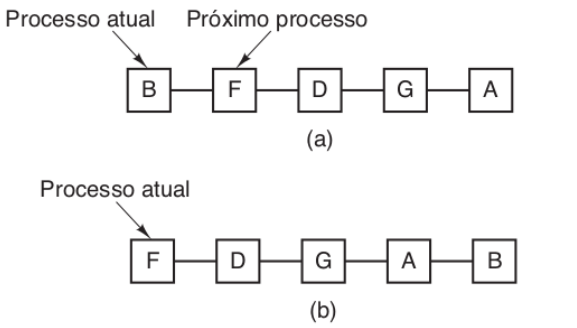
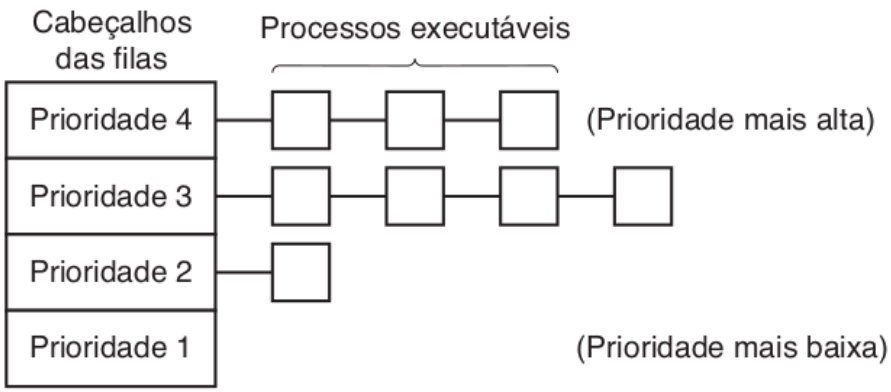


FIGURA 2.42 Escalonamento circular. (a) A lista de processos executáveis. (b) A lista de processos executáveis após *B* usar todo seu quantum.

Escalonamento de prioridades entre as classes: A Figura 2.43 mostra um sistema com quatro classes de prioridade. O algoritmo de escalonamento funciona do seguinte modo: desde que existam processos executáveis na classe de prioridade 4, apenas execute cada um por um quantum, estilo circular, e jamais se importe com classes de prioridade mais baixa. Se a classe de prioridade 4 estiver vazia, então execute os processos de classe 3 de maneira circular. Se ambas as classes — 4 e 3 — estiverem vazias, então execute a classe 2 de maneira circular e assim por diante. Se as prioridades não forem ajustadas ocasionalmente, classes de prioridade mais baixa podem todas morrer famintas.

FIGURA 2.43 Um algoritmo de escalonamento com quatro classes de prioridade.



Múltiplas Filas: Processos na classe mais alta seriam executados por dois quanta. Processos na classe seguinte seriam executados por quatro quanta etc. Sempre que um processo consumia todos os quanta alocados para ele, era movido para uma classe inferior. Como exemplo, considere um processo que precisasse computar continuamente por 100 quanta. De início ele receberia um quantum, então seria trocado. Da vez seguinte, ele receberia dois quanta antes de ser trocado. Em sucessivas execuções ele receberia 4, 8, 16, 32 e 64 quanta, embora ele tivesse usado apenas 37 dos 64 quanta finais para completar o trabalho. Apenas 7 trocas seriam necessárias (incluindo a carga inicial) em vez de 100 com um algoritmo circular puro. Além disso, à medida que o processo se aprofundasse nas filas de prioridade, ele seria usado de maneira cada vez menos frequente, poupando a CPU para processos interativos curtos.

Escalonamento por loteria: A ideia básica é dar bilhetes de loteria aos processos para vários recursos do sistema, como o tempo da CPU. Sempre que uma decisão de escalonamento tiver de ser feita, um bilhete de loteria será escolhido ao acaso, e o processo com o bilhete fica com o recurso. Quando aplicado ao escalonamento de CPU, o sistema pode realizar um sorteio 50 vezes por segundo, com cada vencedor recebendo 20 ms de tempo da CPU como prêmio.

6. Algoritmos a serem implementados

Para esse projeto é obrigatório a implementação do algoritmo **Múltiplas Filas** por todos os grupos. O segundo algoritmo será sorteado na aula do dia 26/10/2018.

7. Relatórios

O sistema deve apresentar as seguintes informações no relatório da simulação:

- Tempo Total de Execução (Em ciclos)
- Utilização da CPU: %
- Trocas de Contexto