

# Sistemas Operativos

## Escalonamento de Processos

3º ano - ESI e IGE (2010/2011)

Engenheiro Anilton Silva Fernandes  
([afernandes@ipiaget.net](mailto:afernandes@ipiaget.net))

# Escalonamento de Processos

- Acto de realizar o chaveamento de processos prontos para executar de acordo com regras bem estabelecidas
  - de forma a que todos os processos tenham a sua chance de utilizar a CPU
- A parte do SO responsável por este chaveamento de processos é o *escalador*
  - usa um *algoritmo de escalonamento* para decidir, a cada instante, qual o próximo processo a ser executado

# Escalonamento de Processos

- Um bom algoritmo de escalonamento tem que incluir os seguintes critérios
  - Justiça - cada processo tem que ter uma parte justa do tempo de CPU;
  - Eficiência - garantir uma ocupação de 100% do tempo de CPU;
  - Tempo de Resposta Mínimo - minimizar o tempo de resposta para os utilizadores interativos;
  - Minimizar o intervalo de tempo entre a submissão de um trabalho e a obtenção dos resultados de volta;
  - Maximizar o número de trabalhos processados por hora.

# Escalonamento de Processos

- Garantir esses critérios não é fácil
  - qualquer algoritmo de escalonamento que favoreça alguma classe de serviços, certamente vai estar prejudicando uma outra;
  - cada processo é único e imprevisível
    - uns gastam muito tempo esperando E/S
    - outros usariam a CPU por horas seguidas se isto fosse permitido
- Com o objetivo de impedir que um processo rode por tempo excessivo, a maioria dos computadores dispõe de um relógio
  - que é um dispositivo de temporização que interrompe a CPU periodicamente
    - é comum uma frequência de 50 a 60 vezes por segundo - 50 ou 60 Hertz, mas em vários computadores o próprio SO pode escolher a frequência de tempo que quiser

# Escalonamento de Processos

- A estratégia de interrupção de processos em execução é designado de “**escalonamento de esvaziamento**” (preemptive scheduling)
  - que contrasta com a estratégia de deixar um processo executar até seu término (non preemptive scheduling);

# Escalonamento de Processos

- **Escalonamento "Round-Robin"**
  - este algoritmo é um dos mais antigos, mais justos e mais amplamente utilizado e consiste do seguinte
    - A cada processo é atribuído um intervalo de tempo (**quantum**), no qual se permite que ele execute.
    - Se o processo ainda estiver rodando no final de seu quantum, a CPU é "esvaziada", ou seja, tomada e dada a um outro processo.
    - Se o processo bloquear ou terminar antes de seu tempo ter expirado, a CPU vai ser atribuída a outro processo.

# Escalonamento de Processos

- **Escalonamento "Round-Robin"**

- a implementação é bem simples:

- O escalonador mantém uma lista de processos executáveis (isto é, processos prontos).
- Quando um quantum expira sem o processo acabar, o mesmo é colocado no final desta lista.
- O escalonador seleciona então o primeiro processo desta lista para execução.



# Escalonamento de Processos

- Escalonamento "Round-Robin"
  - O valor do quantum deve ser cuidadosamente escolhido, uma vez que o chaveamento de um processo para outro, por si só, requer uma certa quantidade de tempo para administração
    - salvar e carregar registradores e mapas de memória, atualizar várias listas, tabelas, etc
  - Deste modo, se o quantum escolhido for muito pequeno, a CPU pode gastar mais tempo fazendo chaveamento de um processo para outro do que propriamente executar os processos.

# Escalonamento de Processos

- Escalonamento "Round-Robin" - Exemplo
  - suponha que o chaveamento de um processo para outro leve 5 ms. Suponha também que o quantum atribuído a cada processo seja de 20 ms. Com esses parâmetros, depois de 20 ms de trabalho útil, a CPU vai ter que gastar 5 ms no chaveamento do processo, o que significa 20% do tempo de CPU sendo gasto em administração.
  - Por outro lado, se o valor do quantum for muito grande, o tempo de resposta para utilizadores interativos pode tornar-se insuportável

# Escalonamento de Processos

- Escalonamento por Prioridade
  - O esquema do *round-robin* assume que todos os processos são igualmente importantes, sejam eles sem urgência ou de pouca importância. Para poder oferecer um tratamento especializado a diversos processos, introduziremos o conceito de prioridade
  - Quando um processo é criado, uma prioridade é atribuída a ele
  - Quando o escalonador tiver que escolher, entre os processos prontos, qual o próximo processo a executar, o escolhido vai ser o que tiver prioridade mais alta

# Escalonamento de Processos

- Escalonamento por Prioridade
  - O problema com este esquema é que um processo com prioridade alta pode rodar indefinidamente, impedindo que outros processo, com prioridades mais baixas, cheguem a executar
  - Este problema pode ser resolvido decrementando-se a prioridade do processo ativo a cada interrupção de relógio.
  - Eventualmente, a prioridade deste processo ativo vai ficar menor que a próxima mais alta prioridade de um outro processo. Neste momento, ocorre um chaveamento entre os processos.

# Escalonamento de Processos

- Escalonamento por Prioridade
  - As prioridades podem ser associadas aos processos de um modo estático ou dinâmico. Uma prioridade é estática quando ela é associada a um processo no momento da criação do processo (mesmo que depois ela seja decrementada para impedir o processo de monopolizar a CPU)
  - Uma prioridade é dinâmica quando o sistema é quem decide o seu valor baseado em estatísticas sobre a execução deste processo
  - Eventualmente, a prioridade deste processo ativo vai ficar menor que a próxima mais alta prioridade de um outro processo. Neste momento, ocorre um chaveamento entre os processos.
  - Um exemplo de prioridades estáticas é o seguinte: Imagine que num sistema militar, os processos iniciados pelos generais tenham prioridade sobre os processos dos coronéis, que por sua vez têm prioridade sobre os majores, e assim por diante por toda a hierarquia.

# Escalonamento de Processos

- Escalonamento por Prioridade
  - Um algoritmo simples de atribuição de prioridade para processos com muita carga de E/S consiste em setar a prioridade para  $1/f$ , onde "f" é a fração do último quantum que o processo usou.
  - Um processo que só usou 2 ms de seus 100 ms de quantum, conseguiria prioridade 50, enquanto que um processo que executasse 50 ms antes de bloquear conseguiria prioridade 2.
  - Um processo que usasse todo o seu quantum receberia prioridade 1.

# Escalonamento de Processos

- Escalonamento por Prioridade
  - Normalmente é conveniente agrupar processos em classes de prioridades, aplicando dois níveis de escalonamento:
    - Escalonamento de prioridades entre as diversas classes, executando todos os processos das classes de prioridades mais altas antes das classes de prioridades mais baixas.
    - Escalonamento "round-robin" entre os processos dentro de uma mesma classe.

# Escalonamento de Processos

- Filas Múltiplas

- Um dos primeiros escalonadores por prioridade foi utilizado no CTSS. O problema no CTSS é que o chaveamento de processos era muito lento pois apenas um processo podia estar presente na memória por vez, fazendo com que os processos tivessem que ser lidos do disco e escritos no disco a cada chaveamento
- A prática demonstrou que era mais eficiente dar aos processos com grande utilização de CPU, um quantum razoavelmente grande de tempos em tempos, ao invés de ficar atribuindo um quantum pequeno frequentemente
  - Isto reduziria o chaveamento
  - Por outro lado, como foi visto acima, o fato de dar um quantum grande a todos os processos pode degradar o tempo de resposta

# Escalonamento de Processos

- Filas Múltiplas
  - A solução encontrada foi o estabelecimento de classes de prioridade.
    - Os processos de classes mais altas são escolhidos para execução com mais frequência que os processos das classes mais baixas.
    - Por outro lado, os processos das classes mais baixas recebem um quanta maior para processamento.
    - Desta forma, os processos da classe de mais alta prioridade recebem por exemplo 1 quantum;
      - os processos da classe imediatamente abaixo recebem 2 quanta;
      - os processos da classe seguinte recebem 3 quanta e assim sucessivamente.
    - Quando um processo usa todo o seu quantum, ele é movido para a classe imediatamente inferior.

# Escalonamento de Processos

- Filas Múltiplas
  - O problema com este método é que se um processo começa com uma grande quantidade de cálculos, mas em seguida se torna interativo (isto é, exige comunicação com o usuário), o tempo de resposta seria muito pobre. Este problema pode ser eliminado do seguinte modo: cada vez que um <RET> é digitado no terminal de um processo, este processo é transferido para a classe de prioridade mais alta, assumindo deste modo, que o processo vai se tornar interativo.

# Escalonamento de Processos

- Filas Múltiplas - Exemplo
  - Considere, como exemplo, um processo que necessite de 100 quantum para executar continuamente.
  - Este processo receberia inicialmente 1 quantum. Nas execuções sucessivas, ele receberia 2, 4, 8, 16, 32 e 64 quantum, apesar de usar somente 37 dos 64 quantum finais para completar seu trabalho.
  - Deste modo, apenas 7 chaveamentos seriam necessários ao invés dos 100 que ele necessitaria se utilizasse o "round-robin" puro.

# Escalonamento de Processos

- Filas Múltiplas
  - Esta solução funcionou bem até que um usuário descobriu que teclando <RET> aleatoriamente, o seu tempo de resposta melhorava bastante. Em seguida ele contou o que descobriu aos amigos. A moral da história é que a implementação de uma estratégia eficiente é mais difícil na prática do que em princípio.

# Escalonamento de Processos

- Menor Serviço (Job) Primeiro
  - A maioria dos algoritmos acima foram criados para sistemas interativos.
  - Um algoritmo que é especialmente apropriado para serviços em lote, para os quais sabe-se de antemão o tempo de execução, é o *menor serviço primeiro*.
    - Nestes sistemas, muitas vezes os utilizadores já têm uma boa estimativa dos tempos de execução do programa - numa empresa de seguros, por exemplo, pode-se prever com grande precisão o tempo de execução de um lote de 1000 apólices, uma vez que este tipo de serviço é executado todos os dias.

# Escalonamento de Processos

- Menor Serviço (Job) Primeiro - Exemplo
  - 4 serviços: A, B, C e D levam 8, 4, 4, e 4 minutos para executar, respectivamente.
    - Executando-os nesta ordem, o tempo de resposta para A é de 8 minutos, para B é de 12 minutos ( $8 + 4$ ), para C de 16 minutos ( $8 + 4 + 4$ ) e para D de 20 minutos ( $8 + 4 + 4 + 4$ ), com um tempo médio de resposta de 14 minutos.
  - Considere agora a execução destes serviços usando o algoritmo do *menor serviço primeiro*.
    - O tempo de resposta vai ser 4, 8 ( $4 + 4$ ), 12 ( $4 + 4 + 4$ ) e 20 minutos ( $4 + 4 + 4 + 8$ ), respectivamente, com um tempo médio de resposta de 11 minutos - o *menor serviço primeiro* é comprovadamente melhor

# Escalonamento de Processos

- Escalonamento Orientado à Política
  - Uma abordagem completamente diferente de escalonamento é fazer promessas de desempenho aos utilizadores e procurar cumpri-las.
    - Uma promessa simples que é fácil de cumprir é esta: se houverem "n" utilizadores utilizando o sistema num dado instante, cada um vai receber  $1/n$  do potencial da CPU.

# Escalonamento de Processos

- Escalonamento Orientado à Política
  - Para cumprir esta promessa, o sistema vai proceder da seguinte forma:
    - registra quanto tempo de CPU cada usuário teve desde que entrou no sistema
      - registra por quanto tempo cada usuário usou o sistema
      - calcula a quantidade de CPU que cada usuário tem por direito (tempo decorrido desde que o usuário entrou no sistema dividido pelo número de utilizadores)
      - calcula a razão entre o tempo de CPU que o usuário realmente obteve pelo tempo de CPU que o usuário tem por direito
      - executa o processo do usuário com a menor razão até que esta ultrapasse a razão do usuário mais próximo

# Escalonamento de Processos

- Escalonamento em Dois Níveis
  - Até agora, temos assumido que todos os processos executáveis estão em memória principal, mas se não houver memória suficiente disponível, alguns dos processos executáveis terão de ser mantidos em disco.
  - Isto traz implicações importantes para o escalonamento, uma vez que o tempo de chaveamento de um processo que está em disco é muito maior que o tempo de chaveamento de um processo em memória (uma ou duas ordens de magnitude a mais).

# Escalonamento de Processos

- Escalonamento Orientado à Política
  - Uma idéia similar pode ser aplicada aos sistemas de tempo real, onde os processos têm prazos para execução que têm que ser cumpridos.
  - Neste caso, o processo cujo prazo de execução estiver mais próximo de expirar roda primeiro (um processo que tem que terminar em 10 segundos consegue prioridade maior que um processo que tem que terminar em 10 minutos).

# Escalonamento de Processos

- Escalonamento Orientado à Política

- Uma solução para isso é o uso de um escalonador em dois níveis:
  - Um subconjunto dos processos executáveis é carregado inicialmente na memória principal;
  - Um outro subconjunto é mantido em disco;
  - Um escalonador, denominado escalonador de nível mais baixo, se restringe a escolher processos somente deste subconjunto, utilizando qualquer uma das políticas de escalonamento descritas no texto acima;
  - Periodicamente, um outro escalonador, denominado de escalonador de nível mais alto, é invocado para remover processos que estão na memória há bastante tempo, e trazer para a memória aqueles processos que estão em disco há muito tempo;
  - Quando esta troca foi feita, o escalonador de nível mais baixo se restringe, novamente, a executar processos que estão na memória.

# Escalonamento de Processos

- Escalonamento Orientado à Política
  - Esta solução faz com que o escalonador de baixo nível se preocupa em escalar processos que estão na memória principal enquanto que o escalonador de alto nível se preocupa em trazer e levar processos entre a memória e o disco.
  - Um escalonador de alto nível pode usar os seguintes critérios de escolha de processos:
    - Qual o tempo decorrido desde que o processo foi trazido para memória e levado de volta para o disco?
    - Quanto tempo de CPU o processo tem tido recentemente?
    - Qual o tamanho do processo?
    - Qual a prioridade do processo?