Aula 3

Escalonamento

Escalonamento

- Porque?
- Quando?
- Como?
- Critérios?
- Políticas de escalonamento
- Como avaliar?

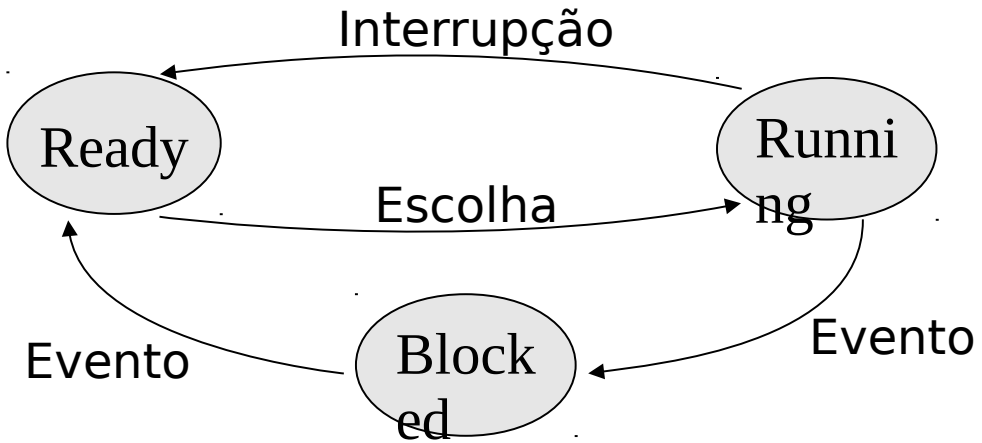
Referências:

- Capítulo 5: 5.1 a 5.3, 5.6

Porque Escalonar?

- Escalonamento controla compartilhamento de recursos
- Multiprogramação permite a utilização simultânea de vários usuários
- Maximizar utilização da CPU: Processos bloqueados não param a CPU

Quando escalonar?

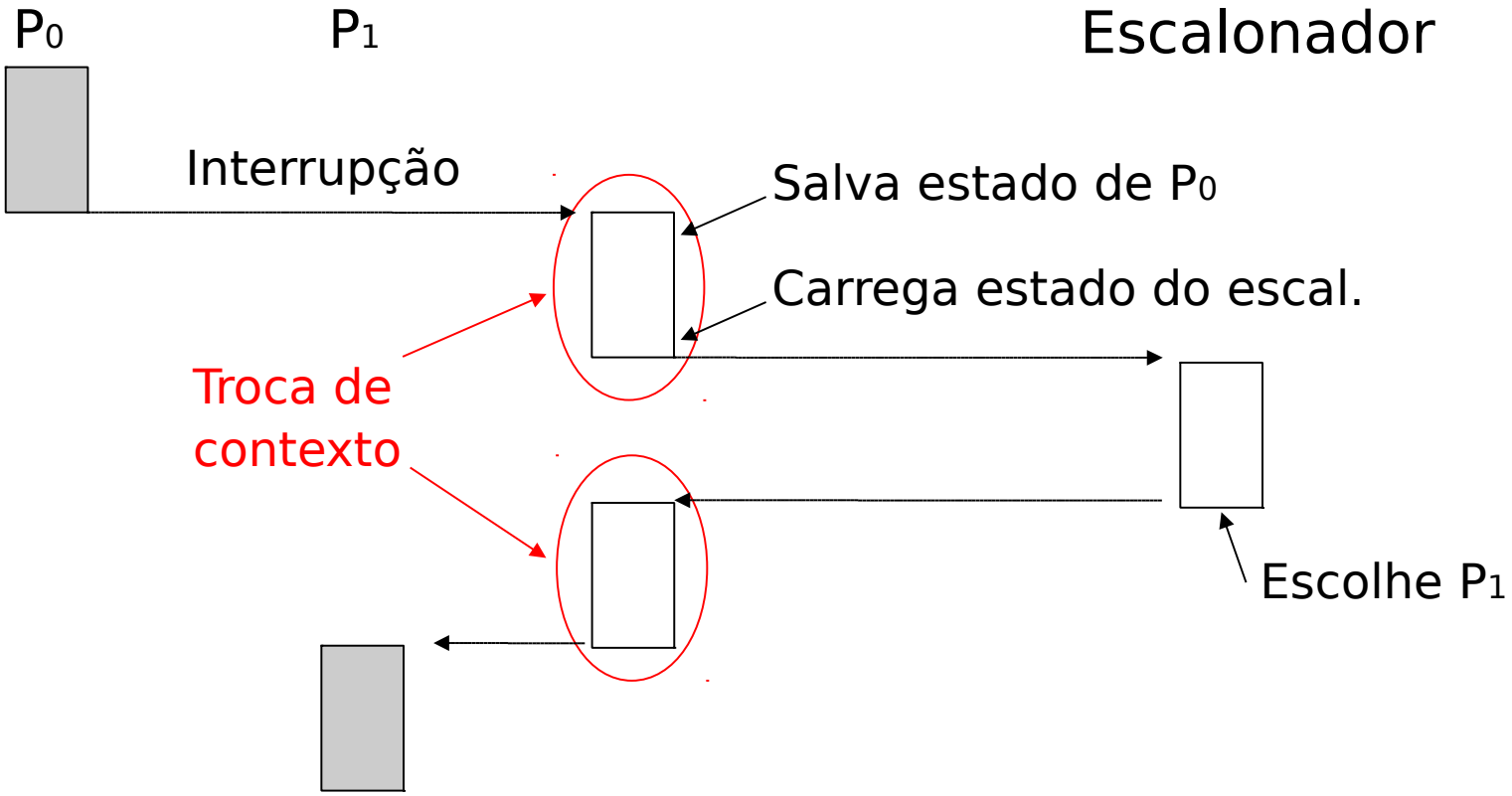


Escalonamento ocorre quando existem:

- Interrupções
  - Temporização
  - Erros (ex: divisão por zero)
- Eventos
  - E/S
  - Comunicação e sincronização

Ou sejam quando muda o processo que está executando

Troca de Contexto



Importante porque:

- Necessária para proteção
- Cara porque troca-se todo o modo de endereçamento

Critérios para Otimização

Políticas para escalonamento devem:

- Maximizar a utilização da CPU: mantê-la ocupada por mais tempo
- Maximizar *throughput*: número de *jobs* por segundo
- Minimizar o *turnaround*: tempo entre submissão e conclusão
- Minimizar o tempo de resposta: tempo entre submissão e início da execução
- Ser *justo* significa o que?
- Ser eficiente: minimizar o *overhead* tal como troca de contexto

Muitas políticas de escalonamento existem! Cada uma com um objetivo diferente

Preempção x Não Preempção

Políticas não preemptivas:

- Processo nunca é interrompido: execução só para quando o processo termina ou é bloqueado
- Mais simples

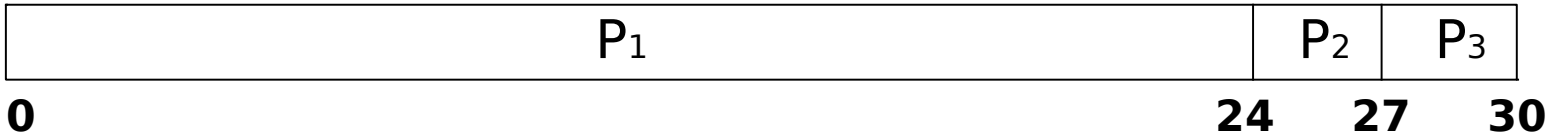
Políticas preemptivas:

- Processo pode ser interrompido antes de terminar ou sem ser bloqueado
- Cada processo tem um *quantum* de tempo de execução. Após este tempo ele é interrompido
- Melhor tempo de resposta

A partir de agora, *processos* e *threads* serão equivalentes a não ser quando citado.

Política de escalonamento – FIFO

- FIFO: First in First Out
- FCFS: First Come First Served

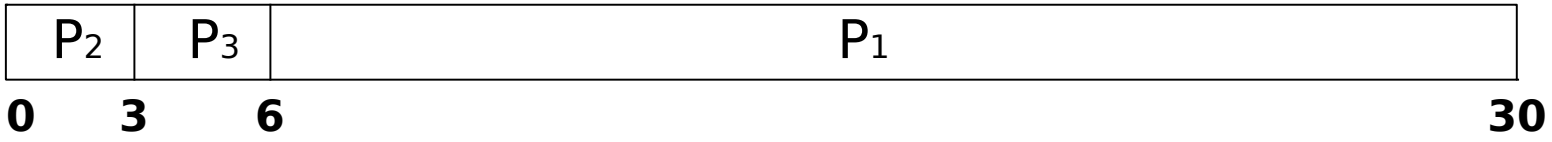


- Não preemptivo: processo executa até o fim
  Ou até bloquear por E/S ou sincronização
- Justo, quem pede primeiro leva vantagem

- Vantagem: simples
- Desvantagem: tempo de resposta alto. Já ficou atrás do cara com a “pastinha preta” na fila do banco?

Shortest Job First – SJF

- Solução: rodar os processos menores primeiro

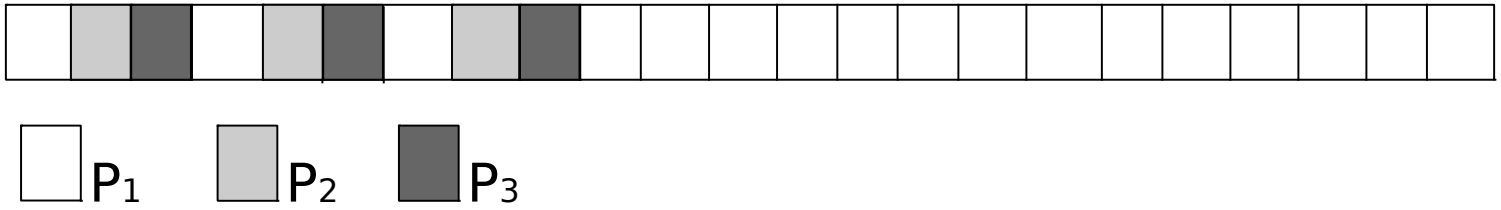


- Algoritmo com menor tempo de espera: ótimo!

- Contudo, é difícil calcular-se o tamanho do processo:
  - Principalmente o tamanho do próximo *burst*
  - Pode-se estimar o tamanho do *burst* usando-se o tamanho do anterior: nem sempre preciso

Round-Robin

Outra solução: adotar um *quantum* de tempo. Quando o processo esgota seu *quantum* ele é interrompido e colocado numa fila de *Ready*.



- Justo: todos tem um pedaço igual
- Virtualmente todos os SOs usam round-robin
- Tamanho do *quantum*:
  - Muito pequeno: troca de contexto cara
  - Muito grande: tempo de resposta alto
- Desvantagem: nenhum processo termina antes de todo o mundo rodar um pouco

Prioridades

Cada processo tem uma prioridade:

- Processos com prioridade maior rodam primeiro
- Alocação de prioridades:
  - Estática: ex – processos que monitoram a temperatura do urânio rodam primeiro
  - Dinâmica: ex – processos que estão mais perto do seu prazo rodam primeiro
- Vantagem: O urânio não derrete!
- Desvatagem: Quem não cuida do urânio pode não rodar nunca!!!

Filas múltiplas

- São usadas várias filas de processos prontos para executar. Cada processo é colocado em uma fila
- Cada fila tem uma política de escalonamento
- Existe uma política de escalonamento *entre* filas

- Vantagem:
  - Pode-se implementar políticas sofisticadas
- Desvantagem:
  - Complicado de prever o comporamento
  - Processos nunca mudam de fila

Filas múltiplas com realimentação

Processos podem mudar de filas:

- Desvantagem: mais complicado ainda! Agora tem que ter políticas de *promoção* e *rebaixamento*
- Vantagem: flexibilidade

Exemplo: 4.3 BSD Unix

Filas múltiplas com realimentação:

- Entre filas: prioridade fixa
- Detro da fila: *round-robin*

Prioridades (i.e., a *qual fila* o processo pertence) mudam baseado em:

- Processo usou todo o *quantum*? PRI--
- Usou CPU por muito tempo? PRI--
- Está esperando por muito tempo? PRI++ *aging*!
- ...

Efeito: processos interativos rodam mais rápido. Processos com muita CPU rodam depois. Mas as prioridades mudam dinamicamente.

Avaliação Analítica

Dada uma certa política de escalonamento, como ela se comporta na prática?

- Assume um conjunto fixo de processos.
- Calcula-se como cada política escalona este conjunto.
- Determina-se o tempo de resposta, etc.

- Vantagem: fácil de calcular.

- Desvantagem: conjunto fixo de processos.

Teoria das Filas

- Assume-se que processos vão chegar segundo uma certa distribuição;
- Métodos estatísticos (teoria das filas) permitem concluir que, se a chegada de processos obedecem à distribuição, então podemos calcular:
  - Tempo de resposta médio
  - Tamanho médio das filas
  - Utilização de CPU
  - ...
- Vantagem: Produz muita informação útil
- Desvantagem: Nem todos os processos podem ser modelados de acordo com distribuições estatísticas; resultados são “médios”

Simulação

- Implementa-se um modelo e roda-se para um número grande de processos
- Calcula-se tempo de resposta, etc, a partir dos resultados da execução

- Vantagem: Qualquer política pode ser modelada; eficiente
- Desvantagem: Resultados não são garantidos

Verificação automática

- Modela-se o sistema como um grafo
- Usam-se ferramentas que visitam o grafo e calculam propriedades do mesmo, incluindo tempo de resposta, etc

- Vantagem: Resultados garantidos; qualquer política pode ser modelada
- Desvantagem: busca em grafos é cara

Escalonamento: Resumo

- Controla o compartilhamento de recursos
- Ocorre quando o processo executando muda
- Critérios para otimização de políticas: tempo de resposta, throughput, utilização, etc
- Políticas:
  - Preemptivas x não preemptivas
  - FIFO
  - SJF
  - Round-robin
  - Priodidades
  - Filas múltiplas
- Métodos de avaliação:
  - Analíticos
  - Teoria das filas
  - Simulação
  - Verificação automática