Aula 06 – Escalonamento de Processos

Norton Trevisan Roman
Clodoaldo Aparecido de Moraes Lima

24 de setembro de 2014

Sistemas Interativos

- Algoritmos para Escalonamento:
  - Round-Robin (Alternância Circular, ou Revezamento)
  - Prioridade
  - Múltiplas Filas
  - Shortest Process Next
  - Garantido
  - Lottery
  - Fair-Share
- Utilizam escalonamento em dois níveis (escalonador da CPU e memória)

Sistemas Interativos: Algoritmos

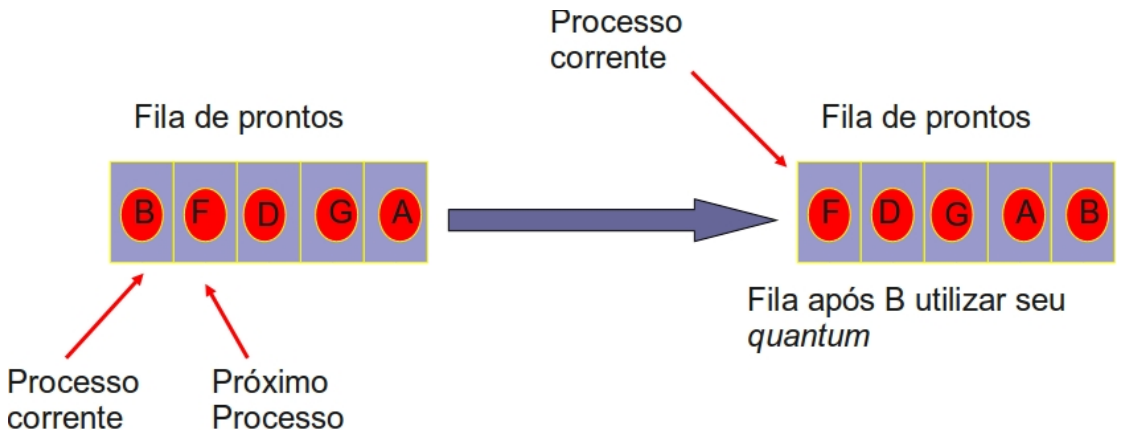
Round-Robin:

- Antigo, mais simples e mais utilizado
- Preemptivo
- Cada processo recebe uma fatia de tempo de execução chamado quantum
  - Os processos são então colocados em uma fila circular (de prontos) e executados um-a-um
  - Quando seu tempo acaba, o processo é suspenso e volta para o final da fila
  - Outro processo (primeiro da fila) é então colocado em execução

Sistemas Interativos: Algoritmos

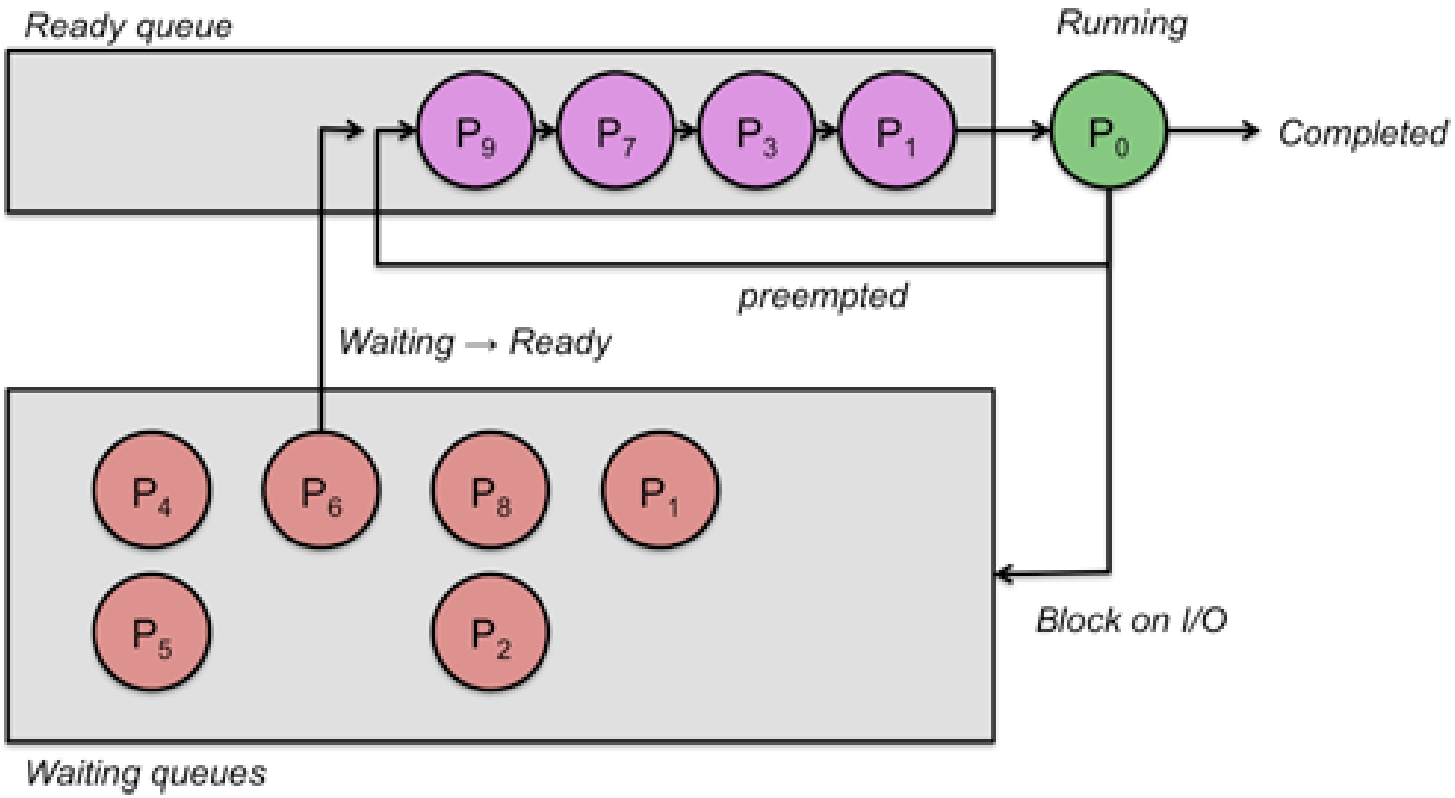
Round-Robin:

- Cada processo recebe uma fatia de tempo de execução chamado quantum
  - Quando um processo solicita E/S, vai para a fila de bloqueados e, ao terminar a operação, volta para o final da fila de prontos
  - Se o processo terminar antes do fim do quantum, o próximo na fila é escolhido



Sistemas Interativos: Algoritmos

Round-Robin:



Sistemas Interativos: Algoritmos

Round-Robin:

- Quando um processo é removido da CPU e outro escalonado, quanto tempo este tem para rodar?
- Se o hardware do clock permitir programação de tempo
  - O SO pode estabelecer um novo quantum para o novo processo – ele terá um quantum inteiro
  - A interrupção do clock será gerada somente após esse tempo passar

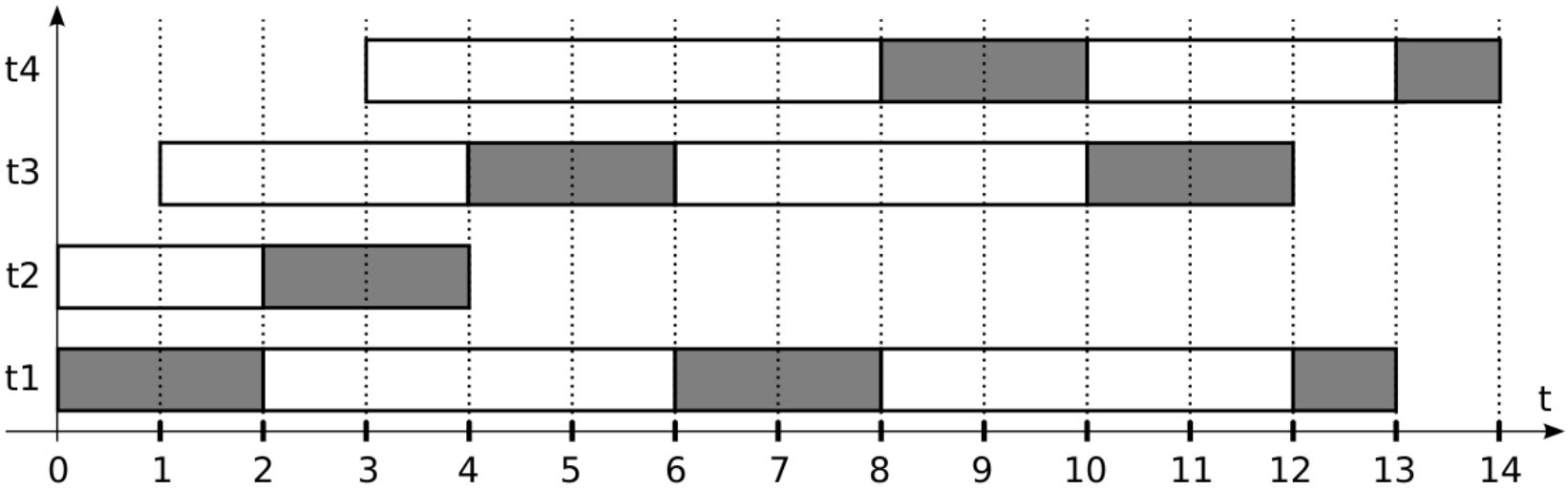
Sistemas Interativos: Algoritmos

Round-Robin:

- Se ele não permitir programação
  - O SO não tem controle sobre as interrupções do clock
  - O processo é escalonado e será interrompido quando vier a interrupção do clock – receberá somente o resto do quantum do processo anterior

Sistemas Interativos: Algoritmos

Round-Robin: Exemplo (quantum = 2s)

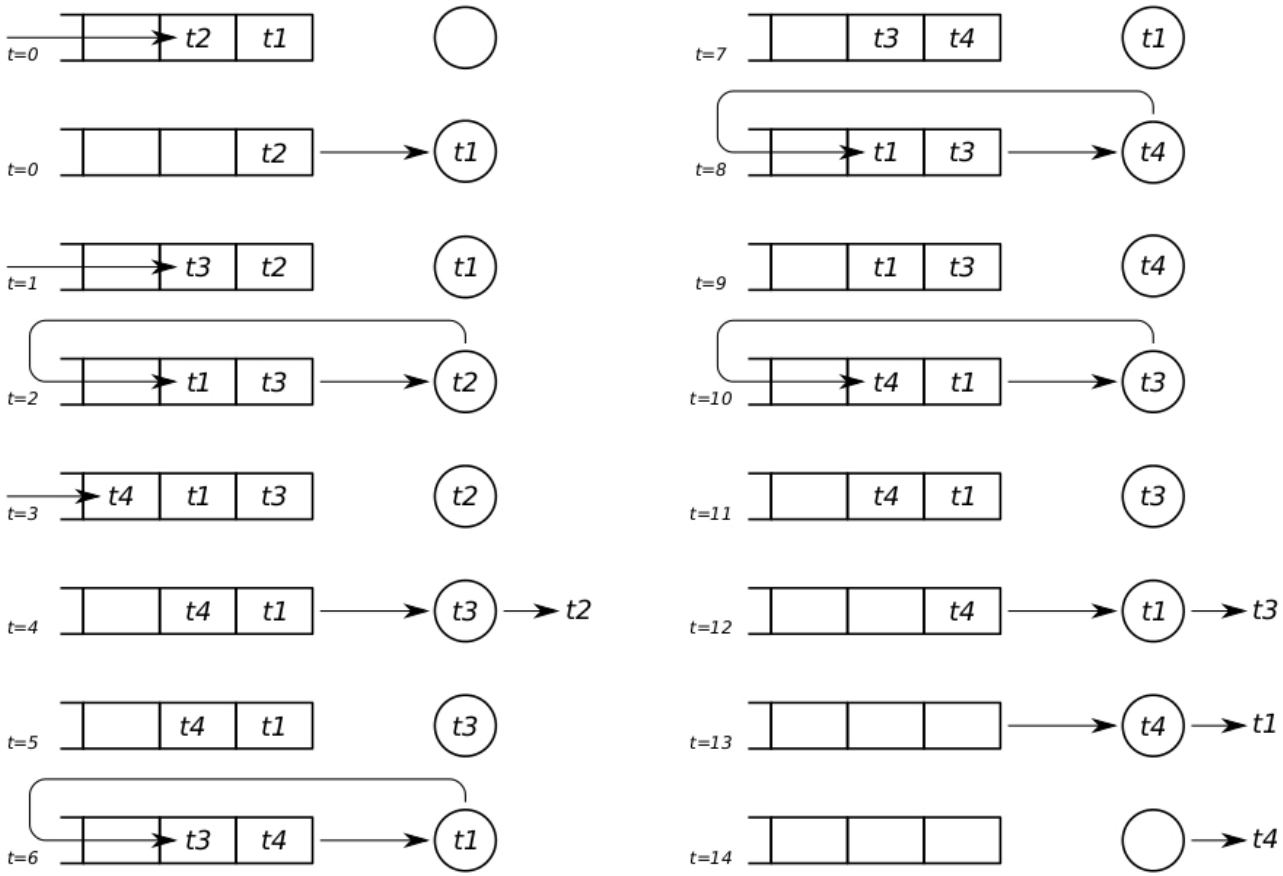


- Tempo médio de execução (*turnaround* médio):

$$T = \frac{t_1 + t_2 + t_3 + t_4}{4} = \frac{(13-0) + (4-0) + (12-1) + (14-3)}{4} = \frac{39}{4} = 9,75s$$

Sistemas Interativos: Algoritmos

Round-Robin: Exemplo (quantum = 2s)



Evolução da fila de prontos

Sistemas Interativos: Algoritmos

Round-Robin:

- Problemas:
  - Tempo de chaveamento de contexto (salvar e carregar registradores, mapas de memória etc.)
  - Processos IO-Bound (que realizam operações de E/S) são prejudicados – usam pouco a CPU e quando retornam de uma operação vão para o final da fila
  - Como E/S é demorada, processos IO-Bound podem não voltar na mesma posição e até mesmo perder várias voltas

Sistemas Interativos: Algoritmos

Round-Robin:

- Qual deve ser o tamanho do quantum?
  - Se for muito pequeno, ocorrem muitas trocas $\rightarrow$ diminui a eficiência da CPU
  - Se for muito longo, será como o first come, first served
- Overhead
  - $overhead = \frac{t_{contexto}}{t_{quantum} + t_{contexto}}$

Sistemas Interativos: Algoritmos

Round-Robin:

- Qual o tamanho do quantum?
  - Quantum = 4ms
    - Se o chaveamento levar 1ms, dos 5ms totais, 20% de tempo de CPU é perdido na troca de processos (ou seja, tarefas administrativas)
  - Quantum = 99 ms
    - Com 1ms de chaveamento gastamos 1% do tempo total de CPU no chaveamento $\rightarrow$ em 10 processos, o décimo levará quase 1s para iniciar
    - Tempo de espera dos processos, pela vez de rodar, é maior, já que a fatia de tempo dada a cada um deles é grande
  - Quantum razoável: 20-50ms

Sistemas Interativos: Algoritmos

Escalonamento por Prioridades

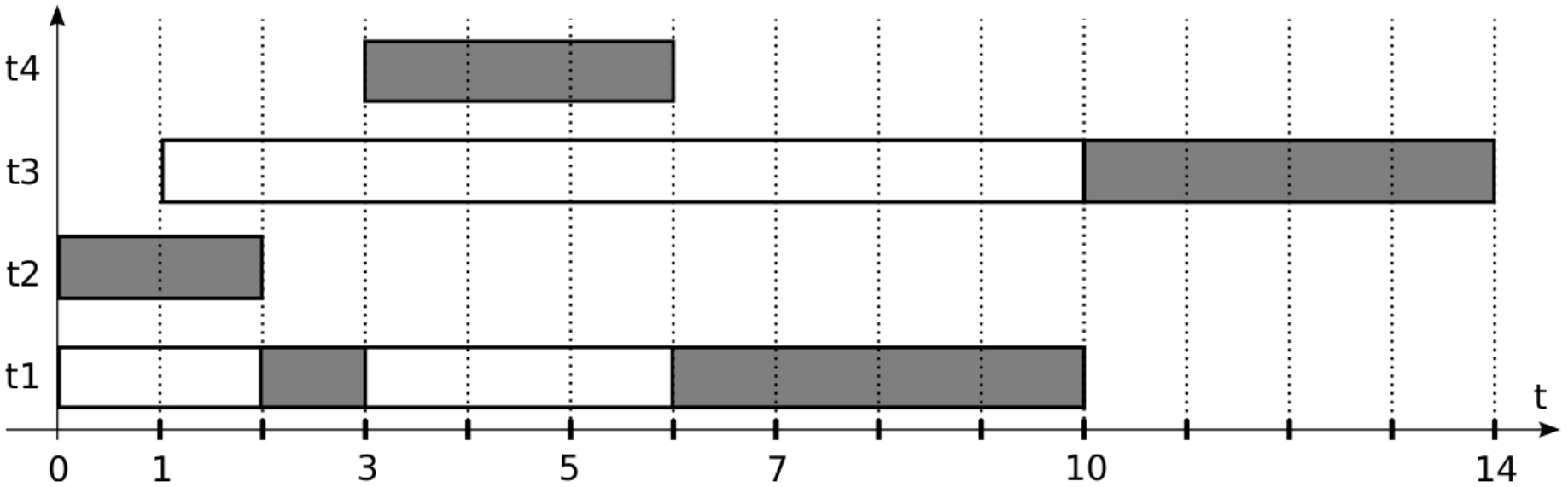
- Cada processo possui uma prioridade
  - Os processos prontos com maior prioridade são executados primeiro (a fila é ordenada por prioridade)
  - Round Robin pressupõe igual prioridade para todos os processos
- Prioridades são atribuídas estatica (pelo usuário. Ex: nice) ou dinamicamente (pelo sistema, ou pelo usuário. Ex: renice)
- Preemptivo

Sistemas Interativos: Algoritmos

Escalonamento por Prioridades – Ex:

tarefa	t_1	t_2	t_3	t_4
ingresso	0	0	1	3
duração	5	2	4	3
prioridade	2	3	1	4

(quanto maior, maior a prioridade)



Sistemas Interativos: Algoritmos

Escalonamento por Prioridades

- Como evitar que os processos com maior prioridade sejam executados indefinidamente?
  - Diminuir a prioridade do processo em execução a cada interrupção do clock
  - Se isso fizer dele o segundo na fila, trocá-lo pelo próximo processo com maior prioridade (chaveamento)
- Alternativamente:
  - Cada processo possui um quantum máximo (não necessariamente igual) para rodar
  - Quando terminar o quantum, o segundo mais alto em prioridade rodará

Sistemas Interativos: Algoritmos

Escalonamento por Prioridades

- Exemplo (decremento com interrupção do clock):

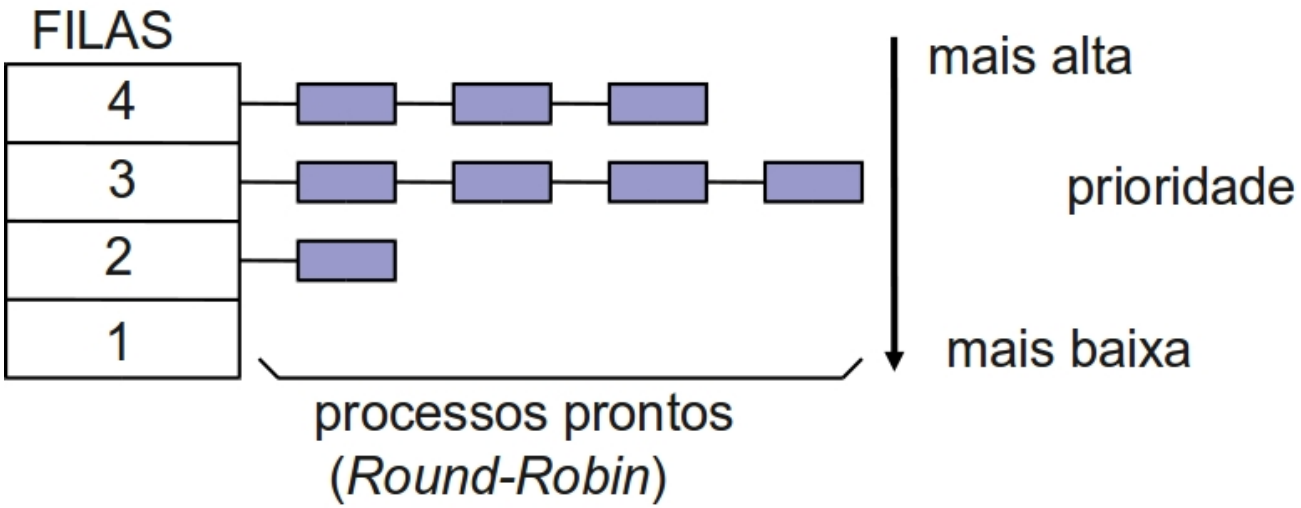
A(5)	B(4)	C(3)	D(2)	roda A
A(4)	B(4)	C(3)	D(2)	A já rodava, continua
A(3)	B(4)	C(3)	D(2)	A perde a prioridade
B(4)	A(3)	C(3)	D(2)	roda B
B(3)	A(3)	C(3)	D(2)	B já rodava, continua
B(2)	A(3)	C(3)	D(2)	B perde a prioridade
A(3)	C(3)	B(2)	D(2)	roda A novamente
A(2)	C(3)	B(2)	D(2)	A perde a prioridade
C(3)	A(2)	B(2)	D(2)	roda C

  - Quando todas as prioridades zerarem, elas são redistribuídas

Sistemas Interativos: Algoritmos

Escalonamento por Prioridades

- O que acontece se vários processos tiverem a mesma prioridade?
  - Podemos agrupa-los em classes de prioridade
  - Usa prioridade entre as classes e Round Robin dentro delas



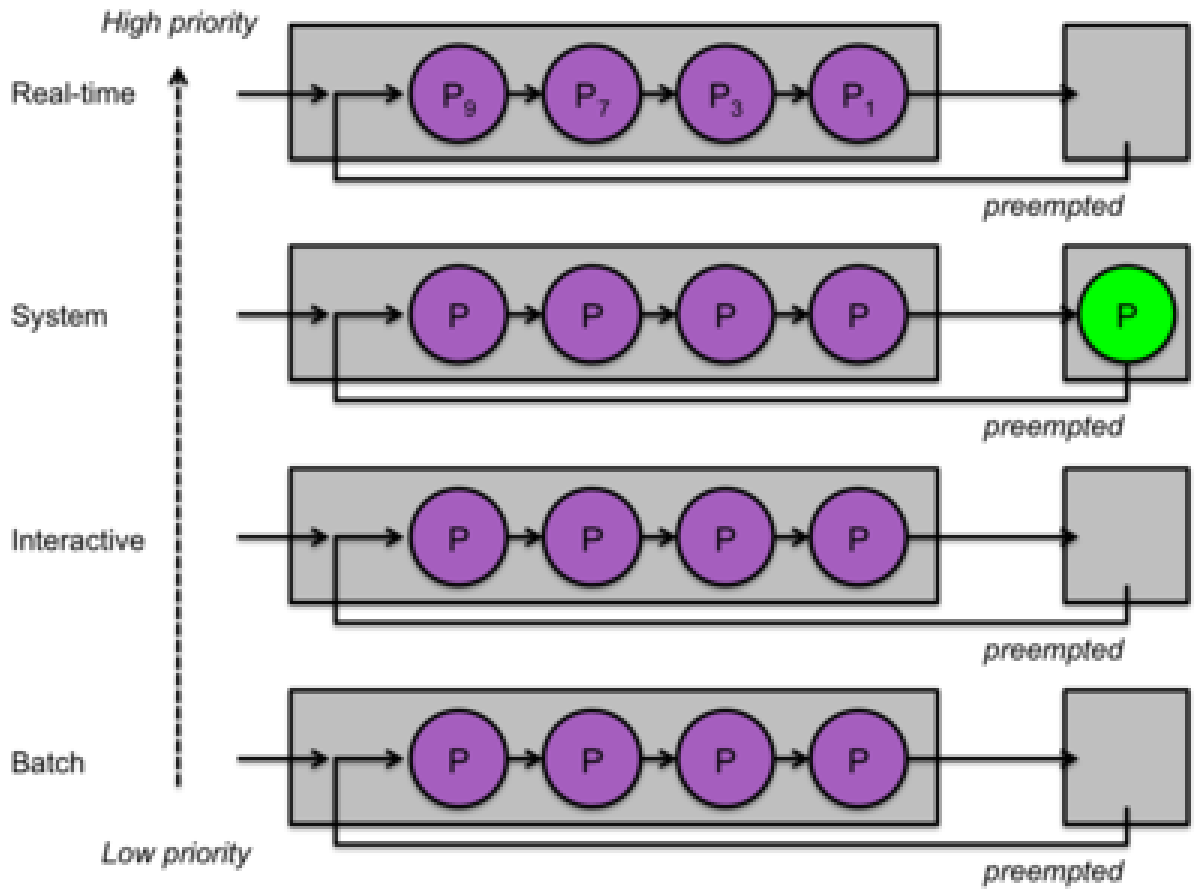
Sistemas Interativos: Algoritmos

Escalonamento por Prioridades

- Enquanto houver processos executáveis na classe maior:
  - Rode cada um de seus processos usando Round Robin (quantum fixo de tempo)
- Se essa classe não tiver mais processos para executar:
  - Passe à de menor prioridade
- Deve-se ajustar as prioridades ocasionalmente
  - Do contrário, as menos prioritárias morrerão de fome – *starvation*

Sistemas Interativos: Algoritmos

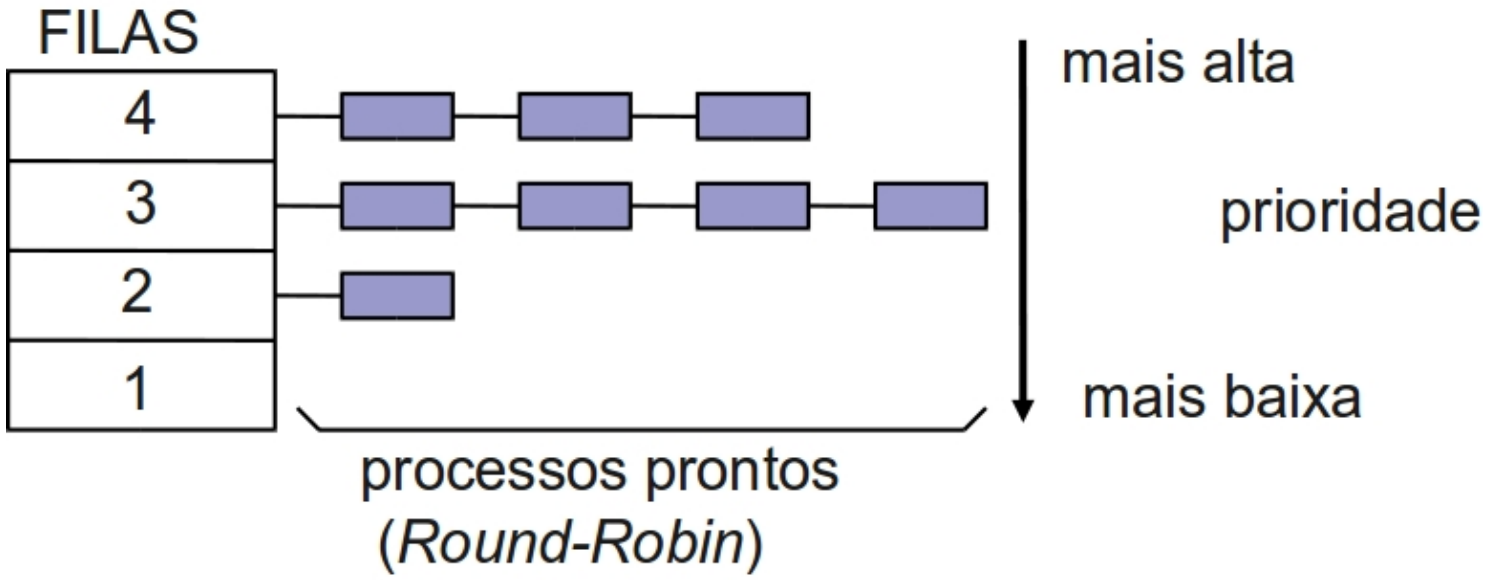
Escalonamento por Prioridades



Sistemas Interativos: Algoritmos

Escalonamento por Prioridades

- Mais fácil de atualizar a fila de prioridade com processos novos – menos comparações



Sistemas Interativos: Algoritmos

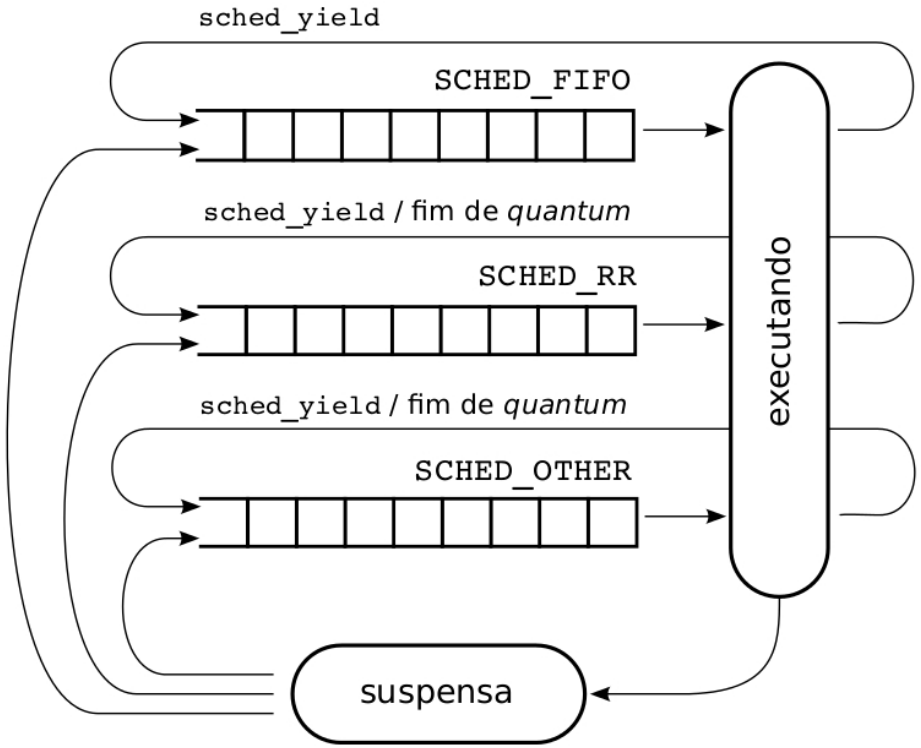
Escalonamento por Prioridades

- Windows (6 classes)
  - Baixa ou ociosa
  - Abaixo do normal
  - Normal
  - Acima do normal
  - Alta
  - Tempo-real
- Linux (2 escalas)
  - Tarefas interativas
  - Tarefas de tempo-real
- Não usam round robin, mas prioridade dentro de cada classe.

Sistemas Interativos: Algoritmos

Escalonamento por Prioridades: Linux

- SHED_FIFO e SHED_RR: tarefas de tempo-real
- SCHED_OTHER: demais tarefas



Sistemas Interativos: Algoritmos

Múltiplas Filas (IBM 7094):

- Preemptivo
  - Variação das classes de prioridades
- Útil quando se tem pouca memória
  - Gerando chaveamento constante para disco
- Define classes de prioridades
  - Processos pertencem a uma classe específica
  - Uma fila de processos por classe

Sistemas Interativos: Algoritmos

Múltiplas Filas (IBM 7094):

- Cada classe de prioridades possui quanta diferentes
  - Processos na classe mais alta, rodam um quantum
  - Os das classes seguintes, por 2, e então por 4, 8, 16 etc
  - Se um processo usar todos os quanta alocados a ele, desce uma classe
- A última classe roda com Round Robin

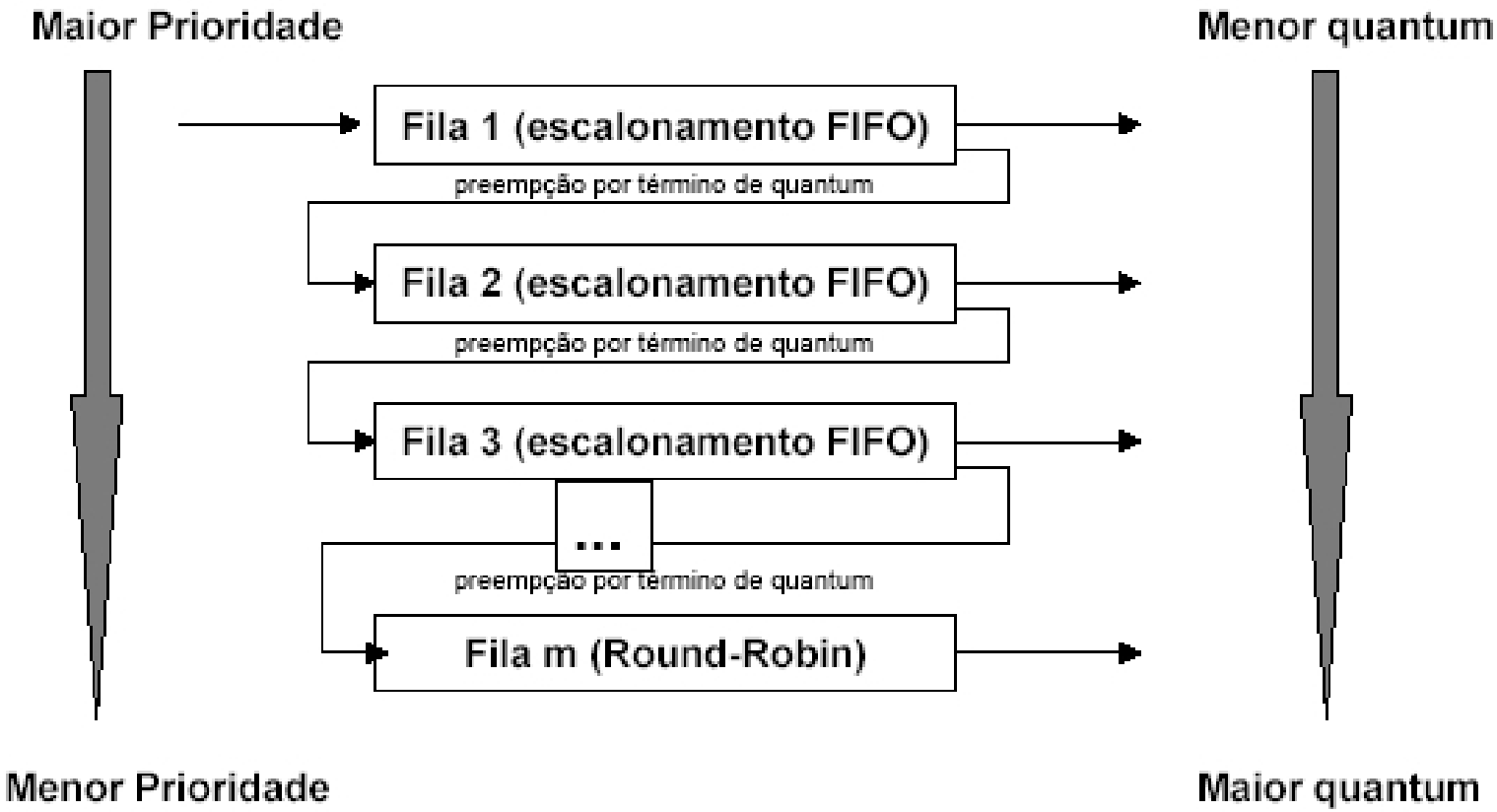
Sistemas Interativos: Algoritmos

Múltiplas Filas:

- Cada vez que um processo é executado e suspenso ele recebe mais tempo para execução
  - Inicialmente recebe 1 quantum, e é suspenso
  - Então muda de classe e recebe 2, sendo suspenso;
  - Muda novamente de classe e recebe 4, 8, 16 etc
- Reduz o número de trocas de processo
  - Os mais curtos terminam logo
  - Aos mais longos é dado mais tempo, progressivamente

Sistemas Interativos: Algoritmos

Múltiplas Filas:



Sistemas Interativos: Algoritmos

Múltiplas Filas:

- Ex.: um processo precisa de 100 quanta para ser finalizado
  - Inicialmente, ele recebe um quantum para execução
  - Das próximas vezes ele recebe, respectivamente, 2, 4, 8, 16, 32 e 64 quanta (7 chaveamentos) para execução (em vez de 100, com round robin)
- Quanto mais próximo de ser finalizado, menos frequente (embora mais demorado) é o processo na CPU
  - Mais ele desce na fila de prioridade
  - Eficiência – os pequenos e rápidos ainda tem vez

Sistemas Interativos: Algoritmos

Múltiplas Filas:

- As classes podem ser criadas de várias maneiras
- Ex: 4 classes (Berkeley – XDS 940)
  - Terminal (mais alta): processos esperando entrada de terminal (quando recebesse a entrada)
  - E/S: processos esperando E/S do disco (que ficou pronto)
  - Quantum curto: processos cujo quantum terminou
  - Quantum longo: processos que terminassem seu quantum várias vezes sem serem bloqueados pelo terminal ou outra E/S
  - Favorece usuários interativos mais que processos em segundo plano

Sistemas Interativos: Algoritmos

Shortest Process Next

- Mesma ideia do Shortest Job First
  - Executa-se a tarefa mais curta primeiro
- Problema: em processos interativos não se conhece o tempo necessário para execução
- Como empregar esse algoritmo: ESTIMATIVA de TEMPO!
  - Com base em execuções antigas da mesma tarefa
  - Verificar o comportamento passado do processo e estimar o tempo.

Sistemas Interativos: Algoritmos

Shortest Process Next

- E se tarefas curtas não pararem de chegar? Inanição
- Solução: envelhecimento (aging)
  - T_0 – tempo estimado para um determinado processo
  - T_1 – tempo medido de sua execução
  - Nova estimativa para o mesmo processo: $aT_0 + (1 - a)T_1$, com $a = \frac{1}{2}$, por exemplo
  - Estima o valor seguinte da série a partir da média ponderada do valor sendo medido e a estimativa anterior

Sistemas Interativos: Algoritmos

Escalonamento Garantido

- Garantias são dadas aos processos dos usuários
- n processos $\longrightarrow \frac{1}{n}$ do tempo de CPU para cada processo

  - Mantém registro do total de CPU que cada processo já usou

Sistemas Interativos: Algoritmos

Escalonamento Garantido

- n processos $\longrightarrow \frac{1}{n}$ do tempo de CPU para cada processo
  - Calcula a quantidade destinada a cada um (o quanto teoricamente deveria ter usado)
    - Ex: tempo desde a criação $\div$ n
  - Compara com o tempo que efetivamente usou
    - Se um processo já teve mais que deveria, um outro, que teve menos, será escolhido

Sistemas Interativos: Algoritmos

Escalonamento por Loteria

- Cada processo recebe "bilhetes" que lhe dão direito a recursos do sistema (inclusive processador)
  - Fatias de processamento iguais por bilhete
  - Quando um escalonamento deve ser feito, escolhe-se aleatoriamente um bilhete, e seu dono conseguirá o recurso
  - Processos mais importantes podem receber mais bilhetes
  - Processos que detenham uma fração f dos bilhetes obterão uma fração f dos recursos (aproximadamente)
  - Processos podem doar bilhetes para colaboração com outros (e recebê-los de volta depois, quando o outro processo doar a ele)

Sistemas Interativos: Algoritmos

Escalonamento por Loteria

- Precisa garantir que todos terão sua vez de rodar
  - Um modo é manter duas listas:
    - Bilhetes já sorteados
    - Bilhetes ainda não sorteados
  - Quando a lista de não sorteados se esvazia, os bilhetes da lista de sorteados são transferidos a ela, reiniciando o processo
- São ideais quando se quer "fracionar" o uso da CPU entre vários processos, a taxas diferentes
  - Basta dar a mesma proporção em bilhetes

Sistemas Interativos: Algoritmos

Escalonamento por Fração Justa (Fair Share)

- À exceção do fair share, em geral o dono do processo não é levado em conta:
  - Se um usuário A possui 9 processos, e um usuário B 1, o usuário A ganha 90% do uso da CPU, com round-robin
- Fair share: O tempo de CPU é alocado ao usuário
  - Se a um usuário foi prometido uma certa fatia de tempo, ele a receberá, independentemente do número de processos
- Extensão do algoritmo garantido
  - Não necessariamente usa 1/n

Sistemas Interativos: Algoritmos

Escalonamento por Fração Justa (Fair Share)

- Exemplo:
  - Usuário 1 $\rightarrow$ A, B, C, D
  - Usuário 2 $\rightarrow$ E
  - Foi prometido 50% da CPU a cada um, e é usado Round Robin:
    - A, E, B, E, C, E, D, E, A, E, ... (possível escolha)
  - Se 2/3 deve ir ao Usuário 1:
    - A, B, E, C, D, E, A, B, E, ...
  - Tudo depende da noção de "justiça" do sistema
    - Se é proporção do tempo, número de tarefas etc

Sistemas de Tempo Real

- Tempo é um fator crítico
  - Tipicamente, um ou mais aparatos externos ao computador geram estímulos
  - O computador deve reagir apropriadamente dentro de um intervalo fixo de tempo
  - Múltiplos processos competindo pela CPU, cada qual com seu próprio trabalho e prazo
- Sistemas críticos:
  - Piloto automático de aviões
  - Monitoramento de pacientes em hospitais

Sistemas de Tempo Real

- Sistemas críticos:
  - Controle de automação em fábricas
  - Monitoramento de reatores em usinas nucleares
  - Transações em bancos
  - Multimídia (música em um cd ou video em dvd):
    - O tempo de processamento deve ser tal que não haja interrupções
- Ponto importante:
  - Obter respostas com atraso é tão ruim quanto não obter respostas

Sistemas de Tempo Real

- Tipos de STR:
  - Hard Real Time (críticos): atrasos não são tolerados
    - Aviões, usinas nucleares, hospitais
  - Soft Real Time (não-críticos): atrasos ocasionais são tolerados
    - Bancos; Multimídia
- Para obter comportamento em tempo real:
  - Programas são divididos em vários processos
  - Cada um previsível e geralmente rápido

Sistemas de Tempo Real

- Eventos causam a execução de processos:
  - Quando um evento externo é detectado, cabe ao escalonador arrumar os processos de modo que todos os prazos sejam cumpridos
- Eventos podem ser classificados como:
  - Periódicos: ocorrem em intervalos regulares de tempo
  - Aperiódicos: acontecem de modo imprevisível
  - Dependendo do tempo necessário para cada evento, pode não ser possível processar todos em tempo $\rightarrow$ sistema não escalonável