

Fundamentos de Sistemas Operacionais

Professor: Cristiano Bonato Both



JESUÍTAS BRASIL

 UNISINOS

Somos infinitas possibilidades

Sumário

- Escalonamento
 - Objetivos
 - Níveis
 - Não-preemptivos
 - Preemptivos
 - Algoritmos
 - Estudos de caso
- Referências



JESUÍTAS BRASIL

UNISINOS

Somos infinitas possibilidades

Escalonamento

- Escalonador:
 - Responsável por selecionar um processo apto para executar no processador
- Objetivo:
 - Dividir o tempo do processador de forma justa entre os processos aptos a executar



JESUÍTAS BRASIL

UNISINOS

Somos infinitas possibilidades

Escalonamento

- Típico de sistemas multiprogramados:
 - *batch, time-sharing, tempo real*
 - Requisitos e restrições diferentes em relação a utilização da CPU
- Duas partes:
 - Escalonador: política de seleção
 - *Dispatcher*: efetua a troca de contexto

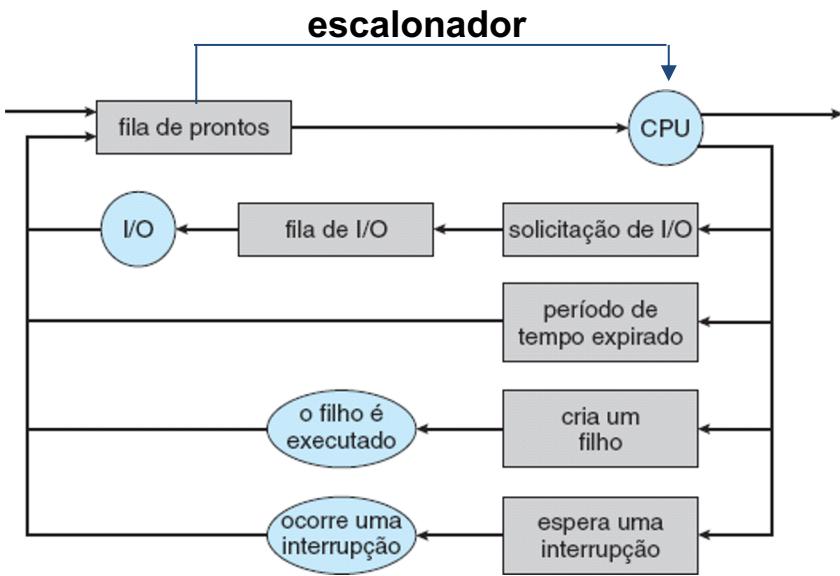


JESUÍTAS BRASIL

UNISINOS

Somos infinitas possibilidades

Escalonador da CPU



- Escolhe uma das tarefas da fila de aptos para executar na CPU
- Executa quando:
 - a CPU fica ociosa
 - criação e término de processos
 - um novo processo se torna apto (ou processo com maior prioridade)
 - interrupções (*timer*, dispositivos de I/O, erros, falta de página em memória)

- Ao escolher um processo da fila → despacha para execução:
 - troca de contexto
 - entra em modo usuário
 - atualiza PC (program counter)
 - executa



JESUÍTAS BRASIL

UNISINOS

Somos infinitas possibilidades

Escalonador da CPU

- Qual processo deve ser selecionado?
 - algoritmos/políticas de escalonamento
- Como os processos serão selecionados?
 - depende da estrutura da fila



JESUÍTAS BRASIL

 UNISINOS

Somos infinitas possibilidades

Objetivos do Escalonamento

- Maximizar a utilização do processador
- Maximizar a produção do sistema (*throughput*)
 - Número de processos executados por unidade de tempo
- Minimizar o tempo de execução (*turnaround*)
 - Tempo total para executar um determinado processo
- Minimizar o tempo de espera
 - Tempo que um processo permanece na lista de aptos
- Minimizar o tempo de resposta
 - Tempo decorrido entre uma requisição e a sua realização



JESUÍTAS BRASIL

UNISINOS

Somos infinitas possibilidades

Escalonador de longo prazo

- Executado quando um novo processo é criado
- Determina quando um novo processo, após sua criação, passa a ser apto
 - Controle de admissão
- Controla o grau de multiprogramação do sistema
 - + processos ativos == - tempo de uso do processador POR PROCESSO

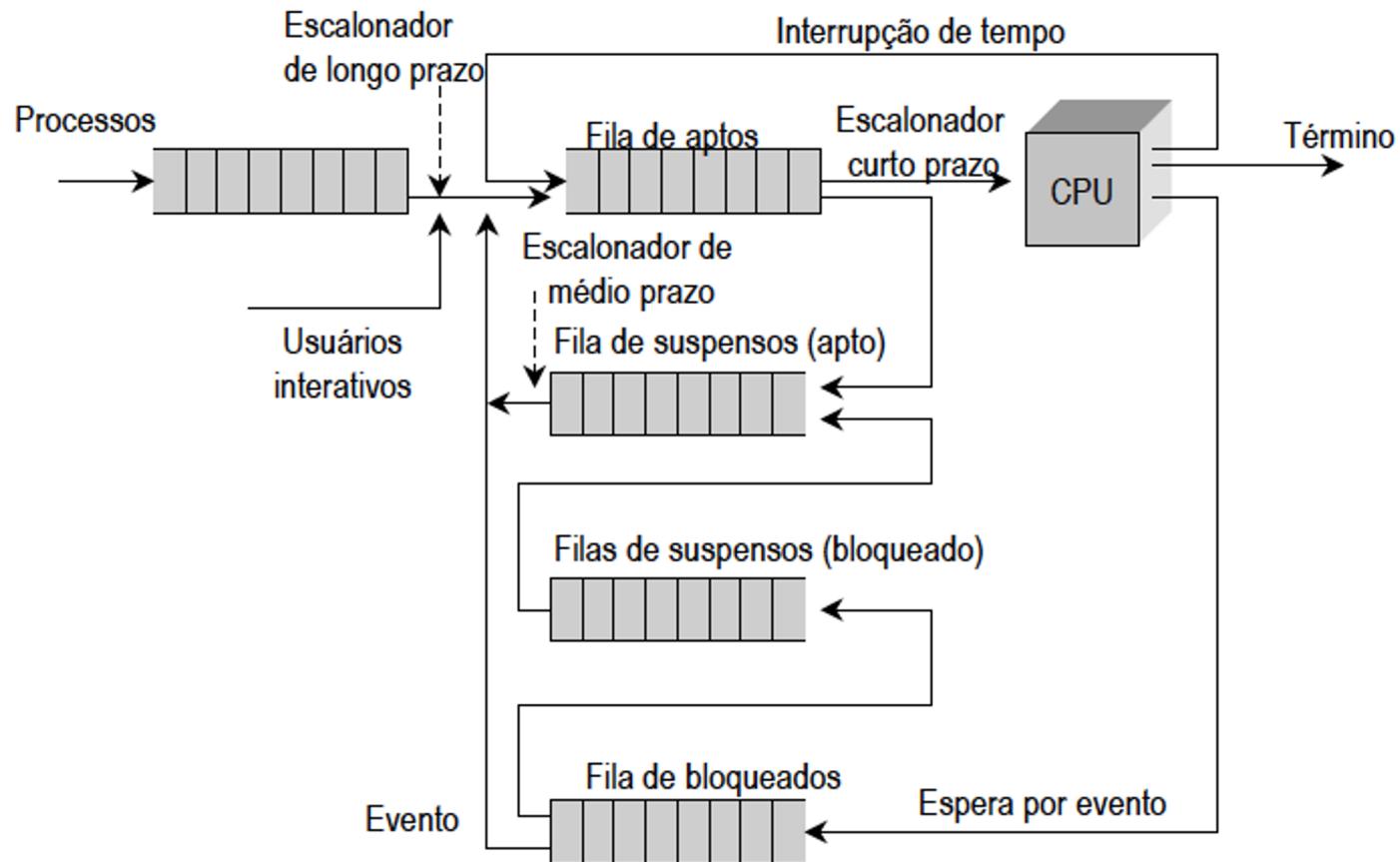


JESUÍTAS BRASIL

 UNISINOS

Somos infinitas possibilidades

Diagrama de escalonamento



JESUÍTAS BRASIL

UNISINOS

Somos infinitas possibilidades

Tipos de escalonadores

- Uma vez escalonado, o processo utiliza o processador até que:
 - **Não preemptivo:**

- Término de execução do processo
- Execução de uma requisição de E/S ou sincronização
- Liberação voluntária do processador a outro processo



Situações que independem da política de escalonamento



JESUÍTAS BRASIL

UNISINOS

Somos infinitas possibilidades

Tipos de escalonadores

- Uma vez escalonado, o processo utiliza o processador até que:
 - **Preemptivo:**

- Interrupção de relógio
- Processo de mais alta prioridade esteja pronto para executar



ALÉM das situações que independem da política de escalonamento



JESUÍTAS BRASIL

 UNISINOS

Somos infinitas possibilidades

Algoritmos de escalonamento

- Algoritmos não preemptivos (cooperativos)
 - *First-In First-Out* (FIFO) ou *First-Come First-Served* (FCFS)
 - *Shortest Job First* (SJF) ou *Shortest Process Next* (SPN)
- Algoritmos preemptivos
 - *Round Robin* (circular)
 - Baseado em prioridades
- Existem outros algoritmos de escalonamento
 - *High Response Ratio Next* (HRRN)
 - *Shortest Remaining Time* (SRT)

v · e	Teoria das filas	[Esconder]
Nódulos de fila única	Fila D/M/1 · Fila M/D/1 · Fila M/D/c · Fila M/M/1 (Teorema de Burke) · Fila M/M/c · Fila M/M/ ∞ · Fila M/G/1 (Fórmula de Pollaczek–Khinchine · Método da matriz analítica) · Fila M/G/k · Fila G/M/1 · Fila G/G/1 (Fórmula de Kingman · Equação de Lindley) · Fila fork-join queue · Fila bulk	
Processos de chegada	Processo de Poisson · Processo de chegada markoviano · Processo de chegada racional	
Redes de filas	Rede de Jackson (Equações de tráfego) · Teorema de Gordon–Newell (Análise de valor médio · Algoritmo de Buzen) · Rede de Kelly · Rede-G · Rede BCMP	
Políticas de serviços	FIFO · LIFO · Processor sharing · Shortest job first · Shortest remaining time	
Conceitos chave	Corrente de Markov de tempo contínuo · Notação de Kendall · Lei de Little · Solução produto-forma (Equação de balanço · Quaserreversibilidade · Método de servidor flow-equivalent) · Teorema da chegada · Método da decomposição · Método de Beneš	
Teoremas de limite	Limite de fluido · Teoria de campo médio · Aproximação em tráfego pesado (Movimento browniano refletido)	
Extensões	Lista de fluidos · Rede de filas com camadas · Sistema de votação (teoria das filas) · Rede de filas adversárias · Perda de rede · Fila de novo julgamento	

FIFO – First-In First-Out

- *First-Come, First-Served (FCFS)*
 - Simples de implementar: fila
 - Processo executa até que:
 - Libere explicitamente o processador
 - Realize uma chamada de sistema (bloqueado)
 - Termine sua execução



JESUÍTAS BRASIL

 UNISINOS

Somos infinitas possibilidades

FIFO – First-In First-Out

- Desvantagem:
 - Prejudica processo I/O bound
- Tempo médio de **espera na fila** de execução:
 - Ordem A-B-C-D = $(0 + 12 + 20 + 35)/4 = 16.75$ u.t.
 - Ordem D-A-B-C = $(0 + 5 + 17 + 25)/4 = 11.7$ u.t.



JESUÍTAS BRASIL

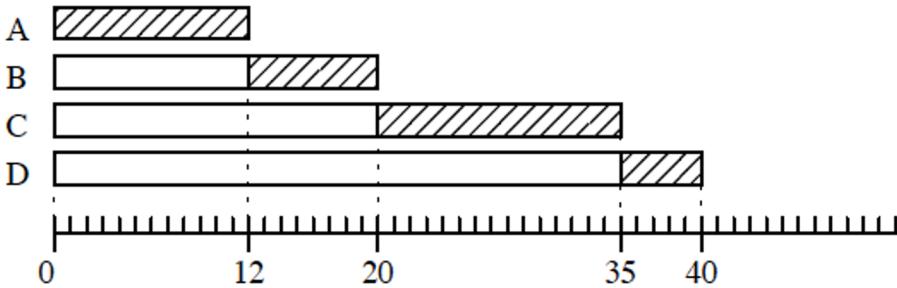
UNISINOS

Somos infinitas possibilidades

FIFO – First-In First-Out

- Tempo médio de processamento (*turnaround*):
$$= ((12 - 0.0) + (20 - 0.5) + (35 - 1.0) + (40 - 1.5)) / 4$$
$$= (12 + 19.5 + 34 + 38.5) / 4$$
$$= 104 / 4$$
$$= \mathbf{26} \text{ u.t}$$

Processo	Tempo
A	12
B	8
C	15
D	5



Processo	Tempo de Chegada na Fila
A	0.0
B	0.5
C	1.0
D	1.5



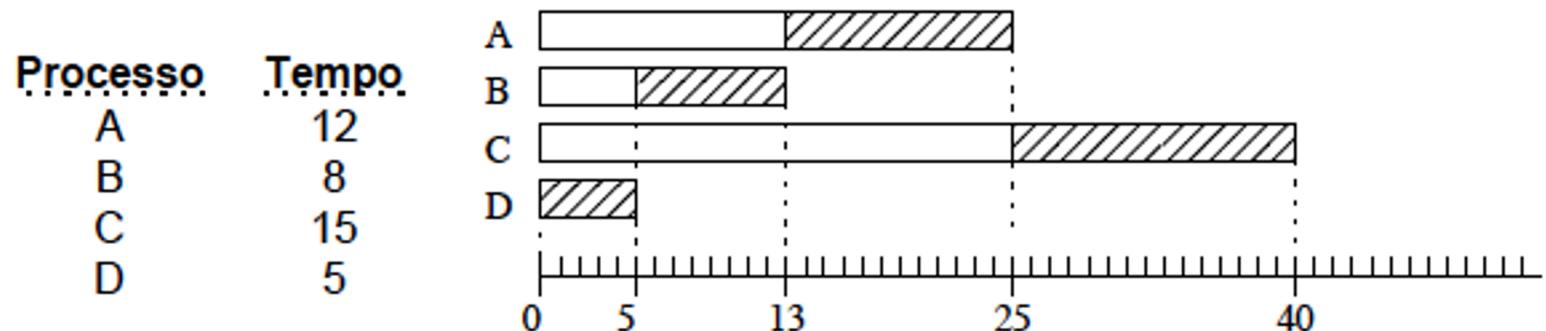
JESUÍTAS BRASIL

UNISINOS

Somos infinitas possibilidades

SJF – Shortest Job First

- Originário do fato que o menor tempo médio de espera é obtido quando se executa primeiro os processos de menor ciclo de processador (*I/O bound*)



$$\text{Tempo médio: } (0 + 5 + 13 + 25)/4 = 10.75 \text{ u.t}$$



JESUÍTAS BRASIL

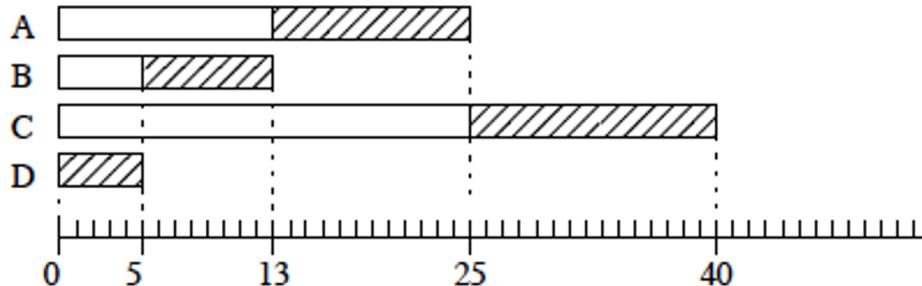
UNISINOS

Somos infinitas possibilidades

SJF – Shortest Job First

- Tempo médio de processamento (*turnaround*):
$$= ((12 - 0.0) + (17 - 1.5) + (25 - 0.5) + (40 - 1.0)) / 4$$
$$= (12 + 15.5 + 24.5 + 39) / 4$$
$$= \mathbf{22.75} \text{ u.t.}$$

Processo	Tempo
A	12
B	8
C	15
D	5



Tempo médio: $(0 + 5 + 13 + 25)/4 = 10.75 \text{ u.t}$

Processo	Tempo de Chegada na Fila
A	0.0
B	0.5
C	1.0
D	1.5



JESUÍTAS BRASIL

UNISINOS

Somos infinitas possibilidades

SJF – Shortest Job First

- Algoritmo ótimo, isto é, fornece o menor tempo de espera para um conjunto de processos
- Processos I/O *bound* são favorecidos
- Dificuldade é determinar o tempo do próximo ciclo de CPU de cada processo, porém:
 - Pode ser empregado em processos batch (*long term scheduler*)
 - Prever o futuro com base no passado



JESUÍTAS BRASIL

UNISINOS

Somos infinitas possibilidades

Prevendo o futuro

- Pode ser feito utilizando os tempos de ciclos já passados e realizando uma média exponencial
 1. t_n = tempo do enésimo ciclo de CPU
 2. T_{n+1} = valor previsto para o próximo ciclo de CPU
 3. T = armazena a informação dos ciclos passados ($n - 1$)
 4. α , $0 \leq \alpha \leq 1$
 5. Define-se: $T_{n+1} = \alpha t_n + (1 - \alpha) T_n$
- Fator α tem o efeito de considerar, de forma ponderada, os ciclos anteriores de processador



JESUÍTAS BRASIL

UNISINOS

Somos infinitas possibilidades

Prevendo o futuro

- Não considera o último ciclo de processador, só o passado ($\alpha = 0$)
 - $T_{n+1} = t_n$
 - Considera apenas o último ciclo de processador ($\alpha = 1$)
 - $T_{n+1} = t_n$
 - Tipicamente se emprega $\alpha = 0.5$
 - Tem o efeito de considerar o mesmo peso para a história atual e a história passada
- $$T_{n+1} = \alpha t_n + (1 - \alpha) \alpha t_{n-1} + \dots + (1 - \alpha)^j \alpha t_{n-j} + \dots + (1 - \alpha)^{n+1} \alpha t_0$$



JESUÍTAS BRASIL

UNISINOS

Somos infinitas possibilidades

Exemplo de “previsão do futuro”

Ciclo de cpu:

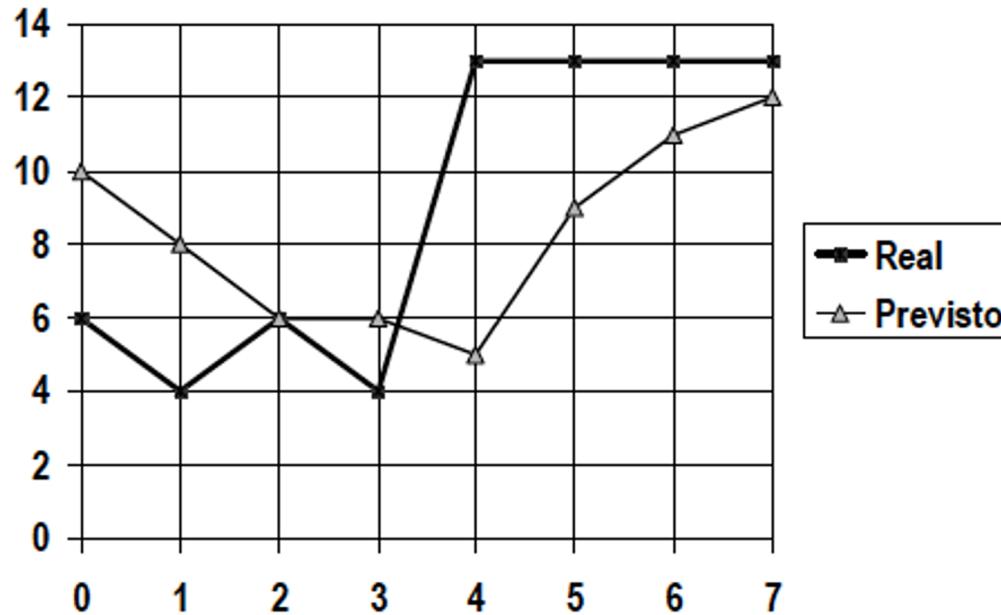
Real:

6 4 6 4 13 13 13

Previsto:

10 8 6 6 5 9 11 12

Parâmetros: $\alpha=0.5$ $\tau_0=10$



JESUÍTAS BRASIL

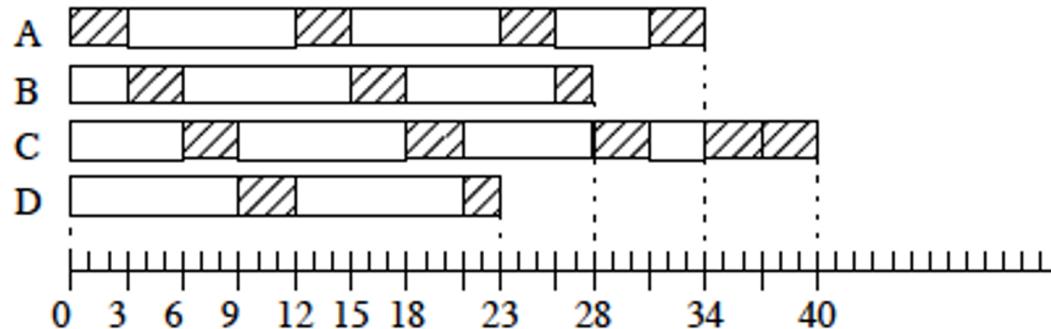
UNISINOS

Somos infinitas possibilidades

Escalonadores preemptivos

Round Robin

- Similar ao algoritmo FIFO, só que:
 - Cada processo recebe uma fatia de tempo (*time-slice, quantum*) para executar um ciclo de processador
- Fila circular
- Necessário delimitar de quanto serão as fatias de tempo
 - Interrupção de tempo



JESUÍTAS BRASIL

UNISINOS

Somos infinitas possibilidades

Escalonadores preemptivos

Round Robin

- Por ser preemptivo, um processo perde o processador quando?
 - 4 situações vistas antes!!
- Se $quantum \rightarrow \infty$ obtém-se o comportamento de um escalonador FIFO



JESUÍTAS BRASIL

UNISINOS

Somos infinitas possibilidades

Escalonadores preemptivos

Round Robin - Problemas

- Problema 1: Dimensionamento do *quantum*
 - Compromisso entre:
 - *overhead* e tempo de resposta em função do número de usuários ($1/k$ na presença de k usuários)
 - tempo de chaveamento e tempo do ciclo de processador (*quantum*)
- Problema 2: Processos I/O bound são prejudicados
 - Esperam como processos CPU *bound*, mas provavelmente não utilizam todo o seu *quantum*
 - Solução:
 - Prioridades: associar prioridades mais altas aos processos I/O *bound*



JESUÍTAS BRASIL

UNISINOS

Somos infinitas possibilidades

Escalonadores preemptivos com prioridade

- Se um processo de maior prioridade entra em estado apto, o processo em execução é preemptado
 - É possível haver prioridade não-preemptiva
- Escalonador deve sempre selecionar o processo de mais alta prioridade segundo uma política
 - Round-Robin
 - FIFO (FCFS)
 - SJF (SPN)



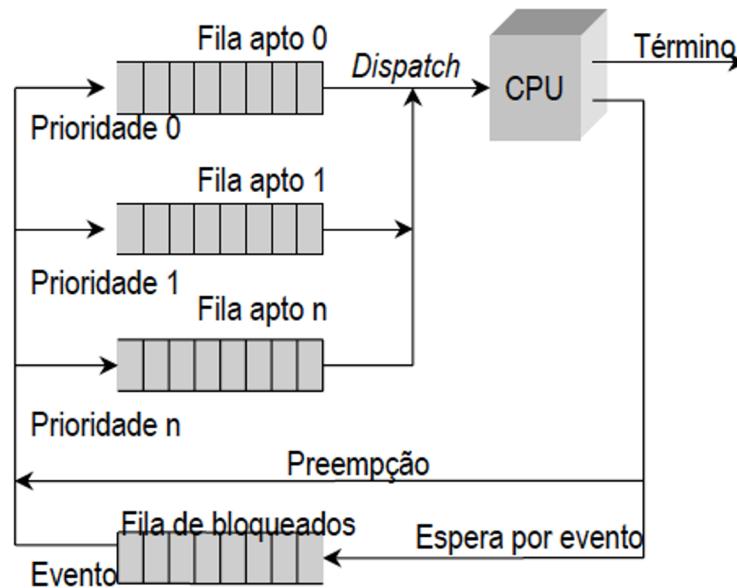
JESUÍTAS BRASIL

UNISINOS

Somos infinitas possibilidades

Escalonadores preemptivos com prioridade

- Múltiplas filas associadas ao estado apto
- Prioridades diferentes para cada fila
 - Pode ter sua própria política de escalonamento (FIFO, SJF, RR)



JESUÍTAS BRASIL

UNISINOS

Somos infinitas possibilidades

Como definir a prioridade de um processo?

- Prioridade estática:
 - Processo mantém sua prioridade desde a sua criação até o fim do seu ciclo de vida
- Prioridade dinâmica:
 - Ajustada de acordo com o estado de execução do processo e/ou do sistema
 - *E.g.* ajustar a prioridade em função da fração do *quantum* que foi realmente utilizada pelo processo
 - *quantum* = 100ms
 - Processo A utilizou 2ms -> nova prioridade = $1/0.02 = 50$
 - Processo B utilizou 50ms -> nova prioridade = $1/0.5 = 2$



JESUÍTAS BRASIL

UNISINOS

Somos infinitas possibilidades

Problemas com prioridades

- Um processo de baixa prioridade pode não ser executado (*starvation*)
- Processo com prioridade estática pode ficar mal classificado e ser penalizado ou favorecido em relação aos demais
 - Típico de processos que trocam de padrão de comportamento durante a execução (*CPU bound* a *I/O bound* e vice-versa)
- Solução
 - Múltiplas filas com realimentação



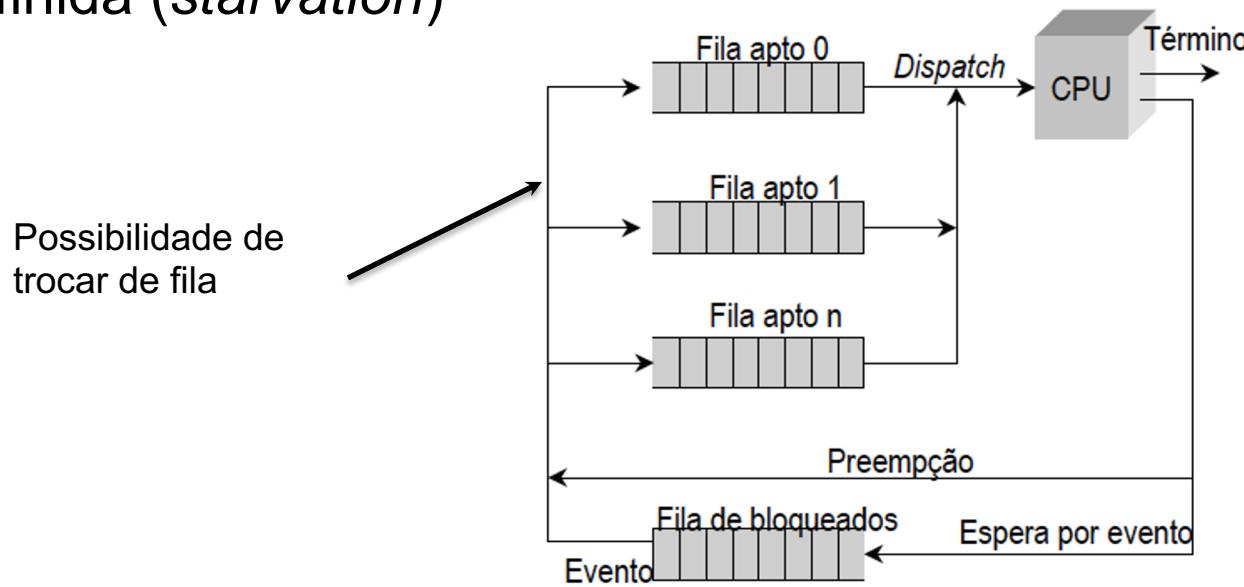
JESUÍTAS BRASIL

UNISINOS

Somos infinitas possibilidades

Escalonadores preemptivos - Múltiplas filas com realimentação

- Baseado em prioridades dinâmicas
- Em função do tempo de uso da CPU a prioridade do processo aumenta e diminui
- Esquema de envelhecimento (*aging*) evita postergação indefinida (*starvation*)



JESUÍTAS BRASIL

UNISINOS

Somos infinitas possibilidades

Estudo de caso: UNIX

- Múltiplas filas com realimentação empregando *Round Robin* em cada uma das filas
- Prioridades são reavaliadas uma vez por segundo em função de:
 - prioridade atual
 - prioridade do usuário
 - tempo recente de uso da CPU
 - outros fatores
- Prioridades são divididas em faixas de acordo com o tipo de usuário
- A troca dinâmica das prioridades respeita os limites da faixa



JESUÍTAS BRASIL

UNISINOS

Somos infinitas possibilidades

Estudo de caso: Linux

- Linux implementa dois tipos de prioridade
 - Estática: exclusivamente para processos de tempo real (prioridade mais alta)
 - Dinâmica: processos interativos e *batch*
- Prioridade é definida pelo usuário (com privilégios especiais) e não é modificada pelo escalonador
- Seleciona processos de maior prioridade para executar
 - Executa segundo a política selecionada: SCHED_FIFO ou SCHED_RR



JESUÍTAS BRASIL

UNISINOS

Somos infinitas possibilidades

Estudo de caso: Linux

- Políticas de escalonamento do Linux (padrão POSIX)
 - SCHED_FIFO: FIFO com prioridade estática
 - Válido apenas para processos de tempo real
 - SCHED_RR: Round-Robin com prioridade estática
 - Válido apenas para processos de tempo real
 - SCHED_OTHER: Filas multinível com prioridades dinâmicas (*time-sharing*)
 - Processos interativos e *batch*



JESUÍTAS BRASIL

 UNISINOS

Somos infinitas possibilidades

Estudo de caso: Windows

- Unidade de escalonamento é a *thread*
- Escalonador preemptivo com prioridades
- Prioridades organizadas em duas classes:
 - Tempo real: prioridade estática (níveis 16-31)
 - Variável: prioridade dinâmica (níveis 0-15)
- Cada nível é implementado por uma fila em uma política *Round-Robin*:
 - Múltiplas filas: classe de tempo real
 - Múltiplas filas com realimentação: classe de tempo variável
 - Classe de tempo real tem precedência sobre as da classe variável



JESUÍTAS BRASIL

UNISINOS

Somos infinitas possibilidades

Estudo de caso: Windows

- Dois parâmetros definem a prioridade de uma *thread*:
 - Valor de prioridade de base do processo
 - Prioridade inicial que indica sua prioridade relativa dentro do processo
- Prioridade da *thread* varia de acordo com uso do processador
 - Preempta por esgotar o *quantum*: prioridade reduzida
 - Preempta por operações de E/S: prioridade aumentada
 - Nunca assume valor inferior a sua prioridade de base, em superior a 15
- Fator adicional em máquinas multiprocessadas:
 - Tentativa de escalaronar uma *thread* no processador que ela executou mais recentemente
 - Princípio: reaproveitamento de dados na memória cache



JESUÍTAS BRASIL

UNISINOS

Somos infinitas possibilidades

Referências Bibliográficas

- SILBERSCHATZ, A.; GALVIN, Peter; GAGNE Greg, Operating System Concepts Essentials. John Wiley & Sons, Inc. 2th edition, 2013.
- TANENBAUM, Andrew S. Sistemas operacionais modernos. 3a. ed. São Paulo: Pearson, 2009-2013. p. 653.
- OLIVEIRA, Rômulo; CARÍSSIMI, Alexandre; TOSCANI, Simão. Sistemas Operacionais. Porto Alegre: Bookman, 4a. ed. 2010.
- <https://www.kernel.org/doc/html/latest/scheduler/index.html>



JESUÍTAS BRASIL

 UNISINOS

Somos infinitas possibilidades