UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Aula 6.1: Gerência do Processador

Conceitos, escalonamento e algoritmos

Prof. Rodrigo Campiolo
Prof. Rogério A. Gonçalves[1]

[1]Universidade Tecnológica Federal do Paraná (UTFPR)
Departamento de Computação (DACOM)
Campo Mourão, Paraná, Brasil

Ciência de Computação

BCC34G - Sistemas Operacionais

Introdução

- A **gerência de processador** depende de dois principais módulos:
  - **Escalonador (scheduler)**:
    - seleciona um processo.
    - considera uma política de seleção.
    - favorece tipos de processos (IO/CPU bound).
  - **Despachante (dispatcher)**:
    - realiza a troca de contexto.
    - grava o contexto do processo atual.
    - recupera o contexto do processo que executará.

Escalonamento

- O escalonador é classificado quanto ao seu nível:
  - Longo prazo (*long-term scheduler*).
  - Médio prazo (*medium-term scheduler*).
  - **Curto prazo** (*short-term/CPU scheduler*).

Escalonamento

- Escalonador de Longo Prazo
  - Executado quando um novo processo é criado.
  - Realiza o controle de admissão - quando um novo processo está apto para execução.
  - Controla o grau de multiprogramação do sistema, isto é, controla se o número de processos ativos afeta negativamente o tempo de uso do processador por processo.

Escalonamento

- Escalonador de Médio Prazo
  - Associado à Gerência de Memória.
    - mecanismo de *swapping*.
  - Suporte adicional à multiprogramação.
    - grau de multiprogramação efetiva (aptos e aptos-suspensos).

Escalonamento

- Escalonador de Curto Prazo
  - Seleciona o processo na lista de prontos para usar o processador.
  - Executado com alta frequência (curto prazo).
  - Aciona o módulo do despachante.

Escalonamento

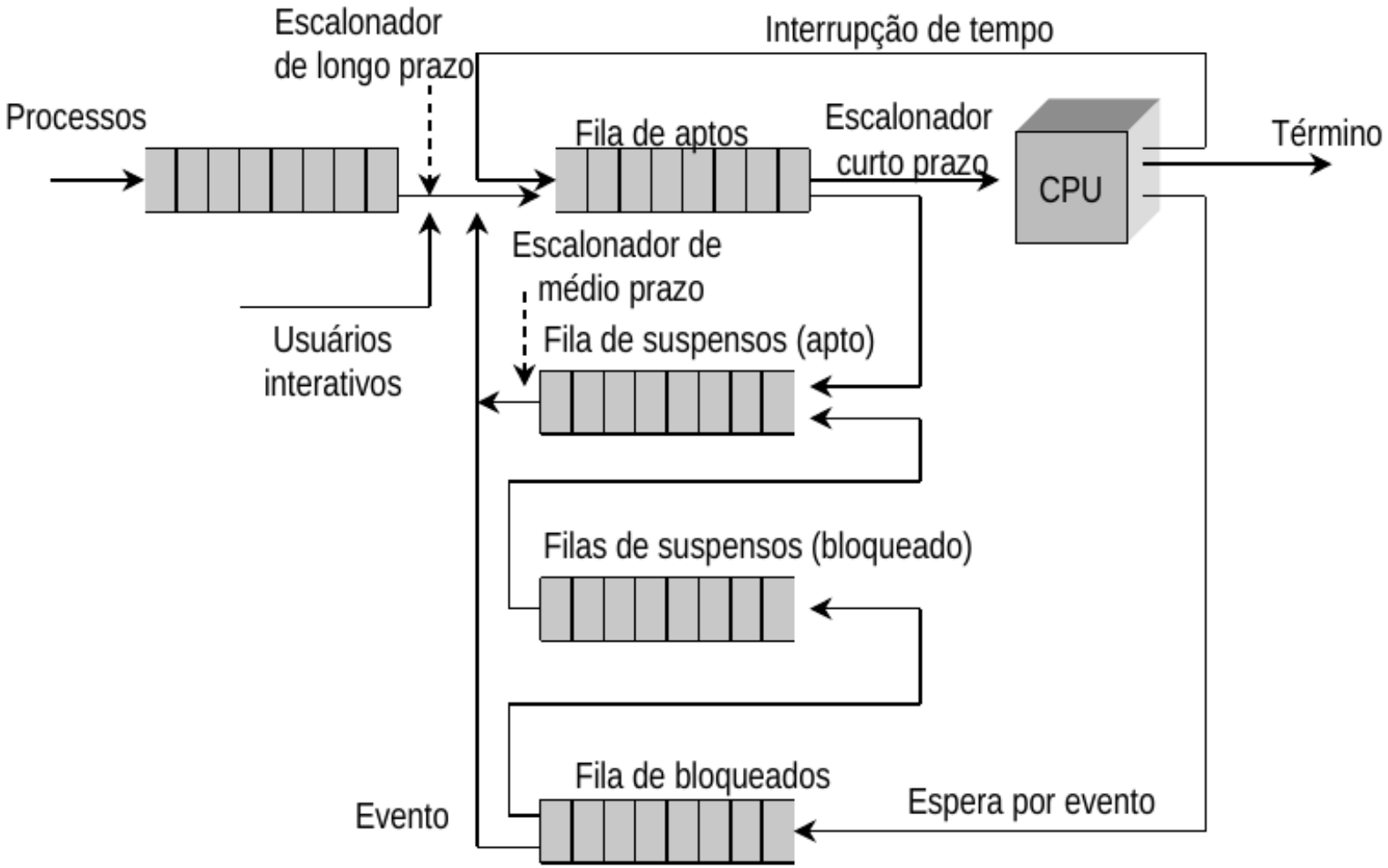

Figura 1: Escalonador de curto, médio e longo prazo [1].

Escalonador de CPU

- O **escalonador de CPU (CPU scheduler)** é a entidade responsável por selecionar um processo pronto para executar no processador (CPU).
- O principal objetivo é prover acesso à CPU de forma justa entre os processos prontos.
- SOs possuem requisitos diferentes para uso de tempo da CPU.
  - sistemas de lote;
  - tempo compartilhado;
  - tempo real;
  - portáteis.

Escalonador de CPU

- Objetivos do escalonamento
  - Maximizar o uso do processador (*CPU utilization*).
    - manter a CPU ocupada.
  - Maximizar a taxa de trabalho do sistema (*throughput*).
    - número de processos executados por unidade de tempo.
  - Minimizar o tempo de execução (*turnaround time*).
    - tempo total de execução de um processo.
  - Minimizar o tempo de espera (*waiting time*).
    - tempo total de espera na lista de prontos.
  - Minimizar o tempo de resposta (*response time*).
    - tempo entre uma requisição e seu início.

Escalonador de CPU

- Quando executar o escalonador?
  - Considerar as características do escalonador.
    - preemptivo ou não-preemptivo, uso ou não de prioridades, ...
  - CPU livre e processos aptos a executar.
  - Criação e término de processos.
  - Processos de alta prioridade estar na fila de aptos.
  - Interrupção de relógio (limite de tempo por processo).
  - Interrupção de hardware (E/S).
  - Interrupção de software (chamadas de sistema).
  - Interrupção de falta de página em memória.
  - Exceções.

Despachante (*Dispatcher*)

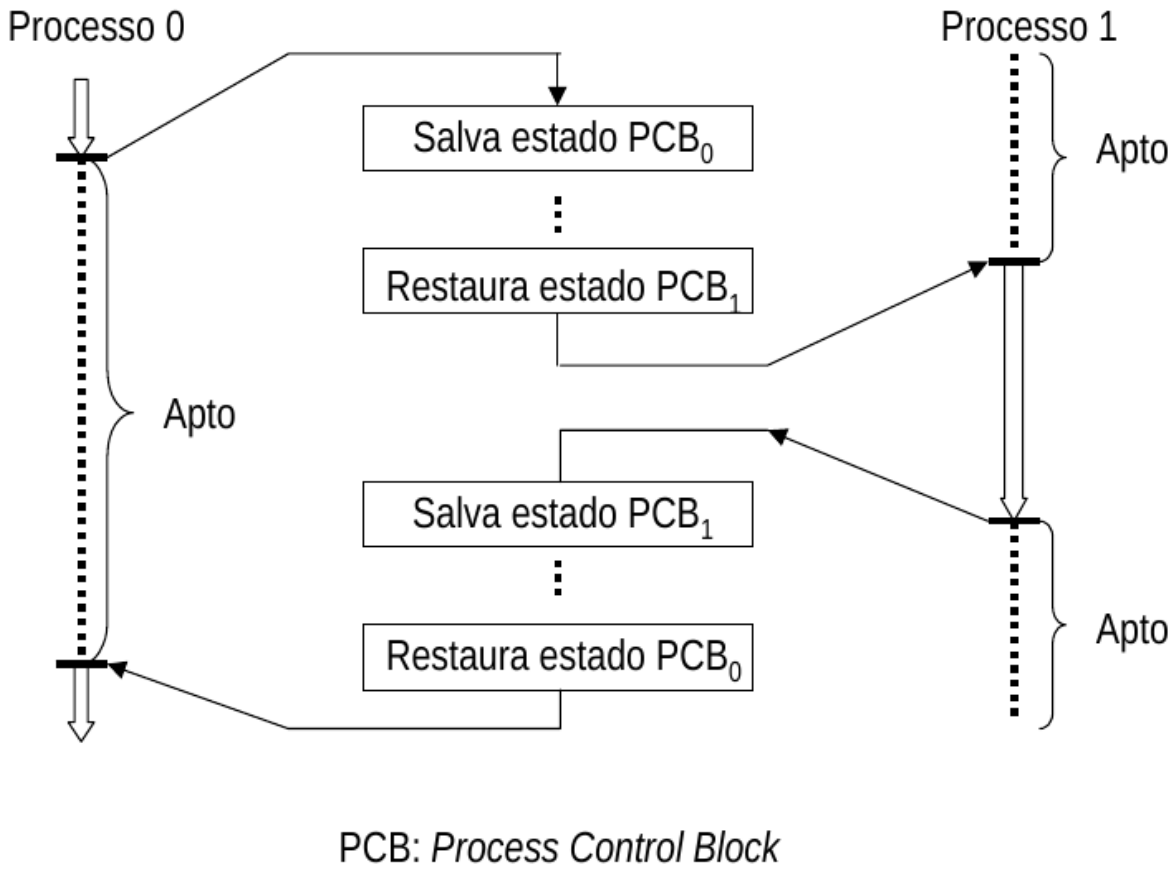

Figura 2: O **despachante** realiza a troca de contexto[1].

Atividades

1. Pesquisar as principais informações que o Despachante deve salvar durante a troca de contexto.

Algoritmos de Escalonamento

- Um **algoritmo de escalonamento** seleciona um processo apto para executar na CPU.
- Algoritmos não preemptivos:
  - *First-In First-Out (FIFO)* ou *First-Come First-Served (FCFS)*.
  - *Shortest Job First (SJF)* ou *Shortest Process Next (SPN)*.
- Algoritmos preemptivos:
  - *Shortest Remaining Time First (SRTF)*.
  - *Round Robin* (Circular).
  - Prioridades.

Algoritmos de Escalonamento

- FIFO - First In First Out.
  - Também denominado de *First-Come, First-Served (FCFS)*.
  - Simplicidade de implementação (fila simples).
  - Favorece processos CPU-bound.
  - Operação:
    - Processos que se tornam aptos são inseridos no final da fila.
    - O processo no início da fila é o próximo a executar.
    - O processo executa até que:
      (a) finalize a sua execução.
      (b) libere o processador explicitamente (*yield*).
      (c) faça uma chamada de sistema (estado = bloqueado).

Algoritmos de Escalonamento

- FIFO - First In First Out.
  - Análise de operação - Tempo de Espera:
    - 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.

Processo	Tempo
A	12
B	8
C	15
D	5

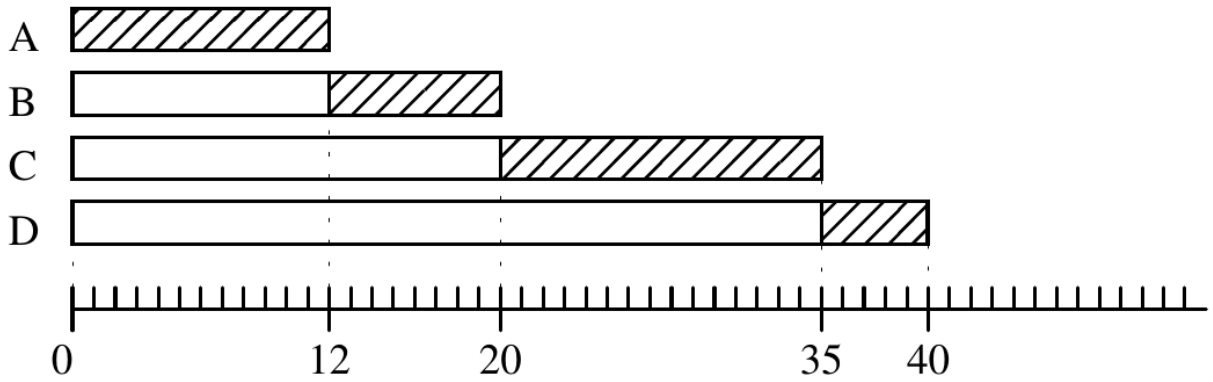

Figura 3: Diagrama de Gantt - FIFO [1]

Algoritmos de Escalonamento

- SJF - Shortest Job First
  - Também denominado de *Shortest Process Next (SPN)*.
  - Provê o menor tempo médio de espera para um conjunto de processos (algoritmo ótimo).
  - Favorece processos IO-bound.
  - Operação:
    - Seleciona o processo apto com o menor tempo para uso de CPU (ciclo de processador).
  - Problema: determinar o tempo do próximo ciclo de CPU de cada processo.
  - Uso: processos *batch* ou fazer uso de previsão.

- SJF - Shortest Job First
  - Análise de operação - Tempo de Espera:
    - Ordem D-B-A-C $(0 + 5 + 13 + 25)/4 = 10{,}75$ u.t.

Processo	Tempo
A	12
B	8
C	15
D	5

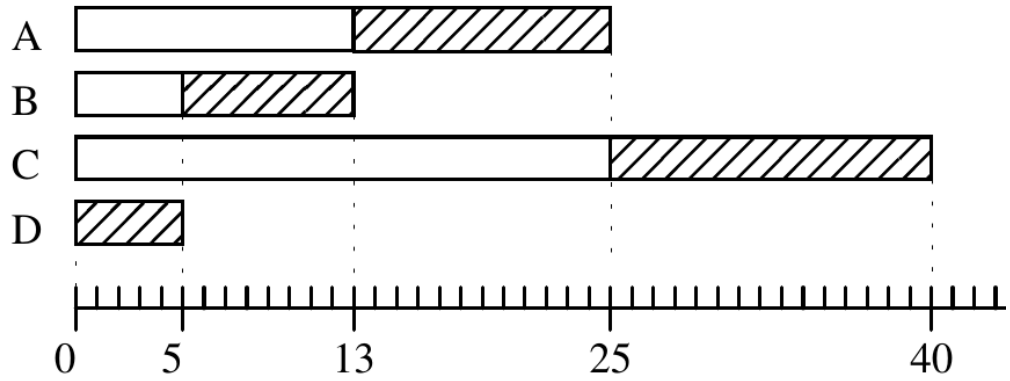

Figura 4: Diagrama de Gantt - SJF [1]

Algoritmos de Escalonamento

- SJF com Previsão
  - Fazer uma previsão usando informações de execuções passadas.
  - Fazer o uso de uma média exponencial.
    t_n = tempo do enésimo ciclo de CPU.
    τ_{n+1} = valor previsto para o próximo ciclo de CPU.
    τ_n = informação dos ciclos passados $(n-1)$.
    α = ponderação dos ciclos anteriores $(0 \leq \alpha \leq 1)$.

    $$\tau_{n+1} = \alpha t_n + (1-\alpha)\tau_n$$

- SJF com Previsão

Ciclo de cpu:

Real:		6	4	6	4	13	13	13
Previsto:	10	8	6	6	5	9	11	12

Parâmetros: α=0.5 τ_0=10

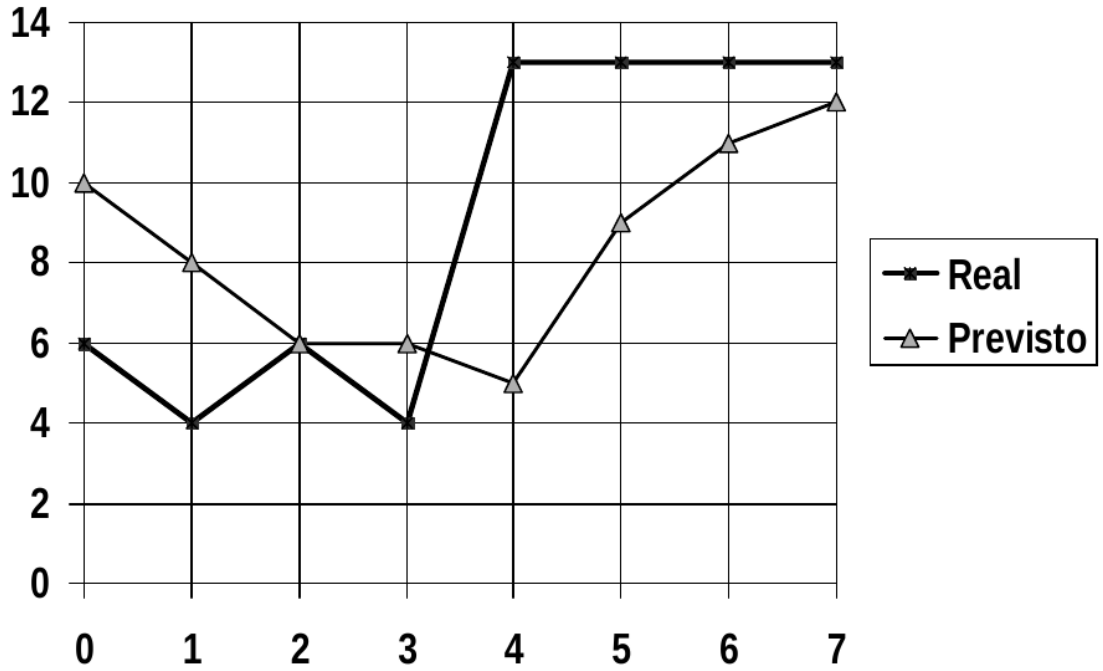

Figura 5: Gráfico de comparação do real com a previsão do SJF [1].

Algoritmos de Escalonamento

- Shortest Remaining Time First (SRTF)
  - Também denominado de SJF Preemptivo.
  - Favorece processos IO bound.
  - Pode causar inanição (*starvation*).
  - Operação:
    - Seleciona o processo apto com o menor tempo.
    - Se o processo em execução não for o de menor tempo, é retirado e colocado na fila de aptos.

- Shortest Remaining Time First (SRTF)
  - Análise de operação - Tempo de Espera:
    - Considere a tabela:

Processos	Tempo de Chegada	Duração de Ciclo
P1	0	8
P2	1	4
P3	2	9
P4	3	5

Tabela 1: Ordem de chegada e duração de ciclo.

  - Tempo médio de espera:
    $[(10-1)+(1-1)+(17-2)+(5-3)]/4 = 26/4 = 6,5$ u.t. (SJF: 7,75 ut)

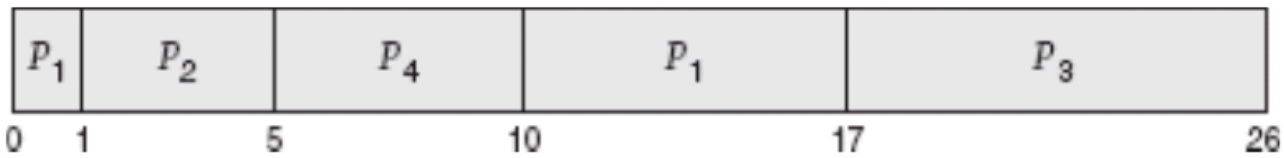

Figura 6: Diagrama de Gantt - SRTF [2]

Algoritmos de Escalonamento

- Round Robin (RR).
  - Similar ao algoritmo FIFO.
    - tempo limite máximo de CPU (*time-slice*, **quantum**).
  - A fila de processos aptos é uma fila circular.
  - Mecanismo para delimitar as fatias de tempo.
    - Interrupção de tempo
  - Operação:
    - Processos que se tornam aptos são inseridos no final da fila.
    - O processo no início da fila é o próximo a executar.
    - O processo executa até que:
      (a) finalize a sua execução.
      (b) libere o processador explicitamente (*yield*).
      (c) faça uma chamada de sistema (estado = bloqueado).
      (d) fatia de tempo (*quantum*) é esgotada.

Algoritmos de Escalonamento

- Round Robin (RR)
  - Análise de operação - Tempo de Espera:
    - Duração dos processos (A=12, B=8, C=15, D=5)
    - Quantum = 3 ut
    - Ordem inicial: A-B-C-D.

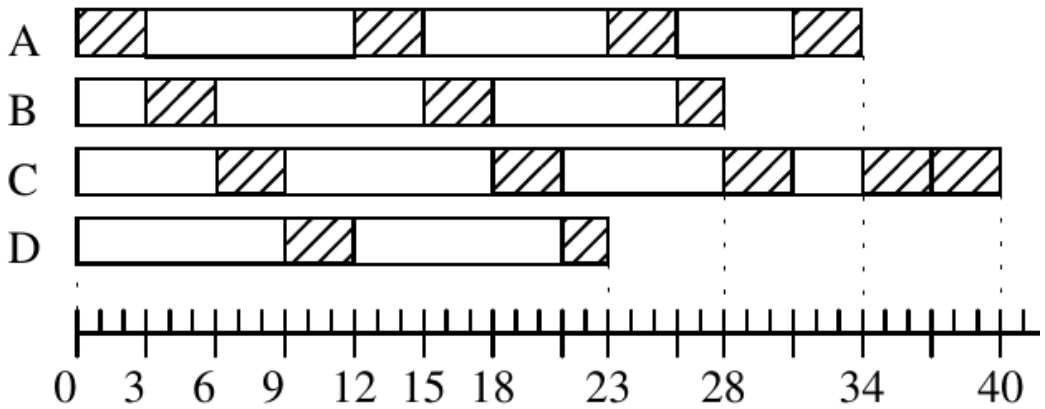

Figura 7: Diagrama de Gantt - Round Robin [1]

- Tempo de espera médio:
  A (9+8+5), B (3+9+8), C (6+9+7+3), D (9+9) = 85/4 = 21,25 ut

Algoritmos de Escalonamento

- Round Robin (RR).
  - Questões sobre o **quantum**:
    - Pequeno ou grande?
    - Fixo ou variável?
    - Idêntico ou não para todos os processos?
  - Problema de dimensionamento do quantum:
    (a) sobrecarga e tempo de resposta.
    (b) tempo de chaveamento e tempo de ciclo de processador.
  - Problema de favorecer processos CPU-bound:
    (a) IO-bound não usam todo o seu quantum.
    (b) Há como compensar essa desigualdade?

Algoritmos de Escalonamento

- Escalonamento com Prioridades
  - Processos aptos de maior prioridade têm preferência.
  - Ocorre preempção se um processo apto for de maior prioridade que atualmente em execução.
  - Existência de prioridades implicam em preempção, mas podemos ter prioridade não-preemptiva.
  - Escalonar seleciona o processo de mais alta prioridade considerando uma política (RR, FIFO, SJF).

Algoritmos de Escalonamento

- Escalonamento com Prioridades
  - Múltiplas filas associadas ao estado apto.
  - Cada fila com uma prioridade.
  - Cada fila com política própria de escalonamento (FIFO, RR).

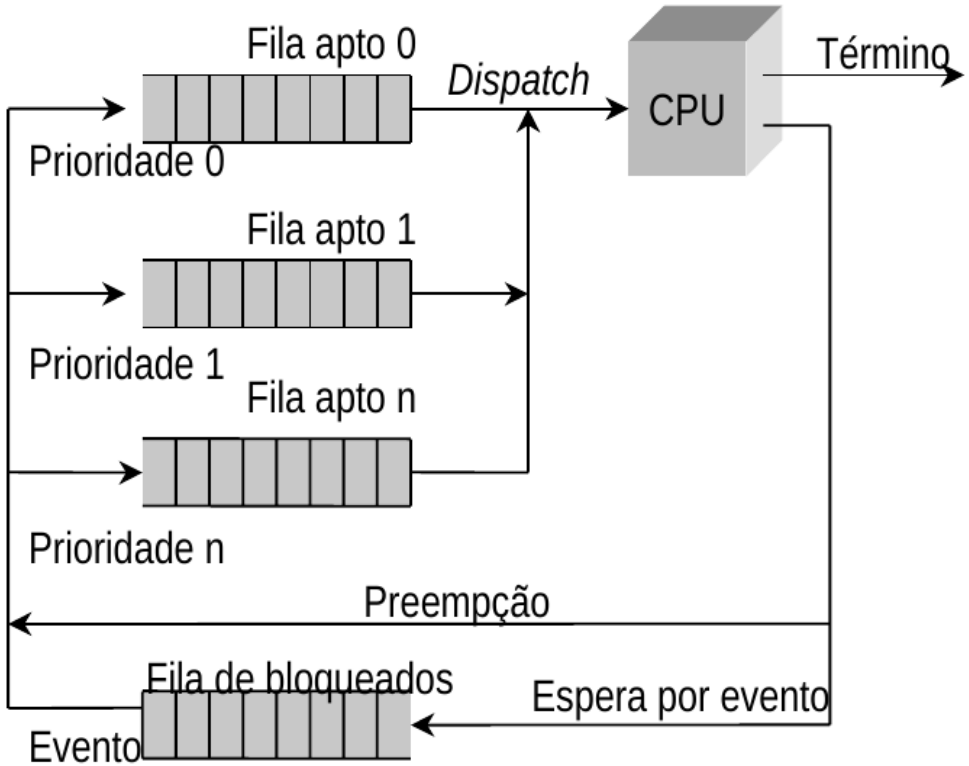

Figura 8: Implementação de escalonamento com prioridades [1].

Algoritmos de Escalonamento

- Escalonamento com Prioridades - política FIFO
  - processo é preemptado – insere no início de sua fila de prioridade.
  - processo de bloqueado para apto – insere no final da fila de sua prioridade.
  - processo troca de prioridade – insere no final da fila de sua nova prioridade.
  - processo cede processador – insere no final da fila de sua prioridade.

Algoritmos de Escalonamento

- Escalonamento com Prioridades
  - Prioridade estática:
    - prioridade não se modifica durante a vida do processo.
  - Prioridade dinâmica:
    - prioridade ajustada considerando o estado da execução do processo e/ou do sistema.
  - Exemplo:
    (a) ajustar em função do uso efetivo do quantum:

    quantum = 100ms
    Processo A usou 10ms – nova prioridade = 1/0.1 = 10
    Processo B usou 50ms – nova prioridade = 1/0.5 = 2

Algoritmos de Escalonamento

- Escalonamento com Prioridades
  - Processos com prioridade baixa podem não ser executados (*starvation*).
  - Processos com prioridade estática podem ser favorecidos ou penalizados em relação aos demais (ex: mudança de comportamento durante a execução).
  - Como considerar a mudança de comportamento (IO bound para CPU bound ou vice-versa) durante a execução?

Algoritmos de Escalonamento

- Escalonamento usando Múltiplas Filas com Realimentação
  - Considera prioridades dinâmicas.
  - A prioridade aumenta ou diminui dependendo do uso de CPU.
  - Evita a postergação indefinida (*starvation*) por meio de mecanismo de envelhecimento (*aging*).

processos trocam de filas

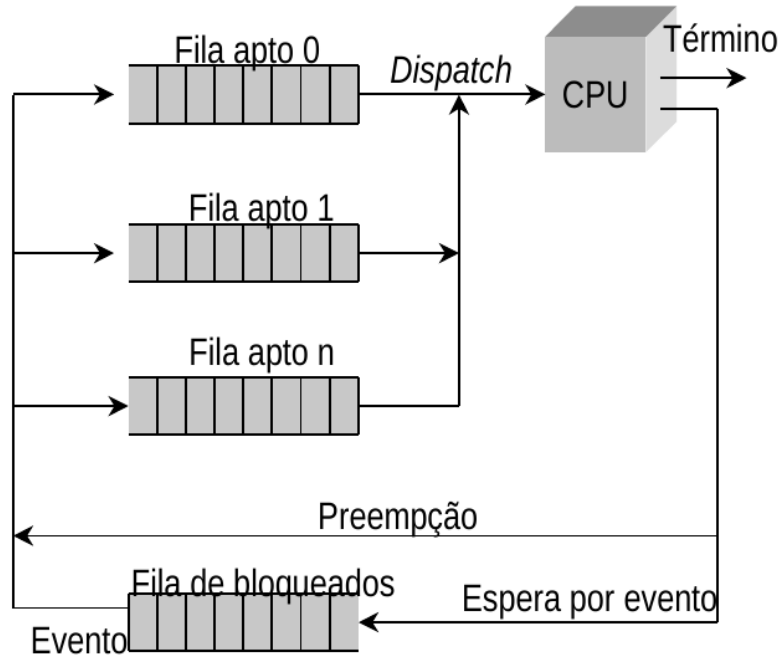

Figura 9: Escalonamento usando múltiplas filas com realimentação [1].

Algoritmos de Escalonamento

- Escalonamento Garantido [3]
  - Ideia: fazer promessas reais aos usuários.
  - Exemplos: n usuários ganham $\frac{1}{n}$ da CPU; n processos ganham $\frac{1}{n}$ da CPU.
- Escalonamento por Loteria [3]
  - Ideia: dar bilhetes de loteria aos processos para tempo de CPU.
  - Escalonador realiza sorteios de bilhetes (ex.: 50 vezes/s).
  - Processos podem receber bilhetes extras para aumentar as chances.
- Escalonamento por Fração Justa [3]
  - Considera o dono do processo no escalonamento.
  - Ideia: alocar uma fração justa por usuário.

Atividades: Algoritmos de Escalonamento

1. Considere um escalonador de curto prazo com duas filas: Fila A (processos TI) e Fila B (processos Alunos). O algoritmo entre as filas é de fatia de tempo. De cada 11 unidades de tempo de CPU, 7 são para a Fila A e 4 para a B. Em ambas filas é usado o algoritmo RR com quantum 2. Mostre a ordem de execução considerando a ordem da tabela 1 (crescente pelo id do processo). Se um processo for preemptado por acabar a fatia de tempo da fila, ele deve retornar ao início da fila. Ao ganhar a CPU novamente, processará somente o restante do quantum não consumido anteriormente [1].

Fila	Processo	Duração
A	P1	6
A	P2	5
A	P3	7
B	P4	3
B	P5	8
B	P6	4

Tabela 2: Processos e tempo de execução.

Escalonamento SMP

- Afinidade de processador
  - Manter o processo em execução no mesmo processador.
  - Afinidade leve ou rígida (ex: *taskset* no Linux).
  - Gerenciamento de cache.
- Balanceamento de carga
  - Duas estratégias de migração:
    (a) por impulsão (push): verifica os processadores e move tarefas para os menos sobrecarregados.
    (b) por expulsão (pull): extrai uma tarefa aguardando para um processador ocioso.

Escalonamento de Tempo Real (TR)

- Sistemas de tempo real não crítico
  - O escalonador não provê garantias de quando uma tarefa será executada.
  - Garante preferência em relação a tarefas não críticas.
- Sistemas de tempo real crítico
  - O escalonador deve garantir que as tarefas sejam atendidas segundo requisitos mínimo e máximo de tempo.

Escalonamento de Tempo Real (TR)

- Dois tipos de latência (atrasos) afetam o desempenho de sistemas TR.
  - Latência de interrupção.
    - Período entre a chegada e início de atendimento da interrupção.
    - Questão importante: o tempo de interrupções desabilitadas deve ser curto.
  - Latência de despacho
    - Período entre a parada de um processo e início de outro.
    - Uso de núcleos preemptivos; liberação de processos em execução e de recursos.

Escalonamento de Tempo Real (TR)

- Escalonamento de taxa monotônica.
  - Execução de tarefas periódicas com prioridade estática e preempção.
  - Prioridade inversamente proporcional ao período de execução (favorece os mais frequentes).
  - Pressupõe que o pico de CPU sempre é o mesmo para cada tarefa. .
- Escalonamento *Earliest Deadline First (EDF)*.
  - Prioridade dinâmica de acordo com limite de tempo.
  - Quanto mais cedo, maior a prioridade.

Escalonamento de Tempo Real (TR)

- Escalonamento das Cotas Proporcionais
  - Alocam T cotas de tempo entre todas as aplicações.
  - Uma aplicação ganha um percentual de cotas.
  - Devem ser usados em conjunto com política de admissão.
- Escalonamento de TR do POSIX
  - Duas classes: `SCHED_FIFO` e `SCHED_RR`.

Escalonamento no Linux

- Algoritmo **O(1)**.
  - Usado no kernel 2.5.
  - Suporte a afinidade de processador e balanceamento de carga.
  - Insatisfatório em sistemas interativos.
- Algoritmo **Completely Fair Scheduler (CFS)**.
  - Usado a partir do kernel 2.6.23.
  - Considera classes de escalonamento.
  - Cada classe com prioridade específica.
  - Flexibilidade para lidar com características específicas dos sistemas.

Escalonamento no Linux

- Classe de escalonamento padrão – CFS.
  - Proporção de tempo CPU a cada tarefa.
  - Baseado em um valor denominado **nice**.
  - O *nice* varia de -20 a +19 (alta baixa) e o valor padrão é 0. Indica *grau de gentileza*.
    `# ps ax -o pid,ni,cmd`
  - Trabalha com o conceito de latência-alvo, intervalo de tempo que todas as tarefas precisam ser executadas uma vez.
  - Latência-alvo possui padrão e mínimo, mas pode aumentar dependendo da carga do sistema.

Escalonamento no Linux

- Classe de escalonamento padrão – CFS.
  - Não atribui prioridades diretamente.
  - Mantém tempo de execução da tarefa (**vruntime**).
  - **vruntime** é associado a um fator de decaimento baseado na prioridade (menor prioridade, maior decaimento). Seleciona-se a tarefa de menor vruntime. Prioridade padrão, vruntime é igual ao tempo de execução.
  - Exemplo: I/O bound x CPU bound: se mesma prioridade, I/O bound acabará tendo precedência.
  - Implementado com uma árvore rubro-negra, mas mantém cache do nó mais à esquerda.

Escalonamento no Windows

- Algoritmo de escalonamento
  - Preempção baseado em prioridades.
  - Esquema de prioridade em 32 níveis.
  - *Classe variável:* 1 a 15.
  - *Classe tempo real:* 16 a 31.
  - *Classe 0:* gerenciamento de memória.
  - Esquema de filas para cada prioridade.
  - Prioridade da classe e prioridade relativa dentro da classe.
  - Quando tempo do *quantum* expira, tarefa é interrompida. Se prioridade variável, prioridade é diminuída (nunca abaixo da prioridade base).

Escalonamento no Solaris

- Algoritmo de escalonamento
  - Baseado em prioridades.
  - Seis classes com prioridades e algoritmos específicos: tempo compartilhado, interativa, tempo real, sistema, compartilhamento justo, prioridade fixa.
  - Padrão: tempo compartilhado. Prioridades alteradas dinamicamente e quantum variável. Usa filas multiníveis com retroalimentação.

Atividades

1. Fazer a lista de exercícios *L06 - Gerência do Processador* disponível na plataforma Moodle.

Referências

[1] Oliveira, R. S. d., Carissimi, A. d. S., and Toscani, S. S. (2010). *Sistemas operacionais*. Série Livros Didáticos. Bookman.

[2] Silberschatz, A., Galvin, P. B., and Gagne, G. (2015). *Fundamentos de sistemas operacionais*. LTC, 9 edition.

[3] Tanenbaum, A. S. and Bos, H. (2016). *Sistemas operacionais modernos*. Pearson Education do Brasil, 4 edition.