Sistemas Operacionais I

Escalonamento de CPU

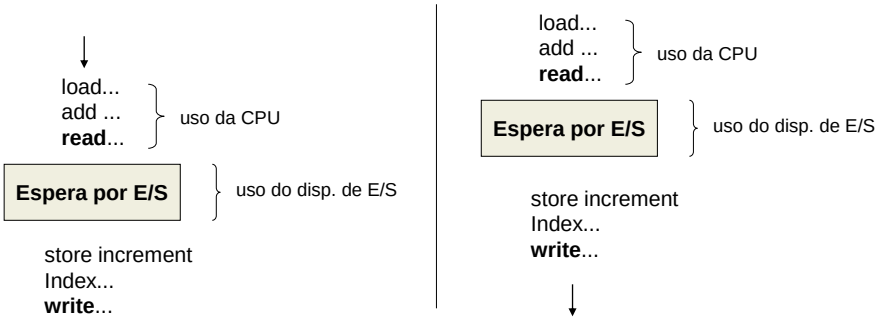
Prof. Carlos Eduardo de B. Paes
Departamento de Ciência da Computação
PUC-SP

Conceitos Básicos
Fases de uso da CPU e de E/S

- Propriedade que determina o grau de sucesso de mecanismos de alocação de CPU (Escalonamento):
  - A execução de um processo se dá em um ciclo, no qual se alternam a execução pela CPU e a espera de E/S → estados do processo se alternam entre um e outro

Conceitos Básicos
Fases de uso da CPU e de E/S

- Processo → geralmente começa com uma atividade que usa CPU intensamente e, em seguida, tem-se em geral muitas operações de E/S



Conceitos Básicos
Fases de uso da CPU e de E/S

- Estas fases podem ser medidas, mas podem variar de processo para processo e de computador para computador
- Geralmente tem-se o seguinte cenário:
  - grande número de fases em que a CPU é usada por um período muito curto de tempo
  - pequeno número de fases longas

Conceitos Básicos
Fases de uso da CPU e de E/S

- **CPU-Burst** : trecho do programa em que apenas operações internas a CPU são executadas
- **I/O-Burst** : trecho do programas em que apenas operações de E/S são executadas
- **Processos**: seqüência alternada entre de CPU-bursts e I/O bursts, começando e terminado com um CPU-burst

Conceitos Básicos
Fases de uso da CPU e de E/S

- **Processo I/O Bound** → processo cujo tempo de processamento depende mais do tempo de processamento de operações de E/S do que o tempo de processamento de CPU.
- **Processo CPU Bound** → processo cujo tempo de processamento depende mais da CPU, tem em geral algumas fases em que a CPU é usada por um período muito longo de tempo.
- *Esta distribuição pode ser importante na seleção de um algoritmo adequado para escalonamento de processos (alocação da CPU)*

Conceitos Básicos
Decisões de Escalonamento

- Decisões de alocação da CPU podem acontecer nos seguintes casos:
  - 1. Quando um processo muda do estado "em execução" para "em espera"
  - 2. Quando um processo muda do estado "em execução" para "pronto"
  - 3. Quando um processo muda do estado "em espera" para "pronto"
  - 4. Quando o processo termina

Conceitos Básicos
Alocação Preemptiva

- Casos 1 e 4 → alocação não-preeemptiva
  - não há escolha a ser feita em termos de alocação. Um novo processo dever ser selecionado para execução
- Casos 2 e 3 → alocação preemptiva
  - é preciso que uma decisão seja tomada, no projeto de um mecanismo de alocação de CPU

Conceitos Básicos
Alocação Preemptiva

- Alocação **Preemptiva** → problemas (custo deste tipo de alocação) de sincronização ao acesso a dados compartilhados
- Preempção → tem efeito sobre o projeto do núcleo (kernel) do SO
  - Tratamento de uma chamada de sistema/operação de I/O pelo kernel do SO precisa ser atômica (caso das maiorias das versões do UNIX)
  - Rotinas do *kernel* são executadas de forma atômica
  - **Este modelo é pobre para suporte a multiprocessamento e computação em tempo real.**

Dispatcher (Despachante)

- Módulo responsável por dar o controle da CPU para o processo selecionado pelo escalonador de curto prazo; Isto envolve:
  - Troca de contexto
  - Mudança para o modo usuários
  - Retomar a execução no ponto onde parou no programa do usuário e recomeçar
- Latência do Despachante: tempo gasto para parar um programa e começar outro

Critérios de Escalonamento

- Algoritmos de escalonamento → possuem características diferentes e podem favorecer uma ou outra classe de processos.
- Critérios → usados para comparação entre algoritmos de escalonamento.

Critérios de Escalonamento

- Critérios:
  - **Utilização da CPU** → deixar a CPU a maior parte do tempo ocupada. A utilização da CPU pode variar de 0 a 100 por cento. (40% = pouco usado, 90% = bastante usado)
  - **Throughput (produtividade)** → enquanto a CPU estiver ocupada, algum trabalho está sendo feito. Uma forma de medir a produtividade de um sistema é pelo número de processos finalizados por unidade de tempo.

Critérios de Escalonamento

- Critérios:
  - **Tempo de processamento(TURNAROUND)** → consiste do tempo que o processo leva para ser executado (entre o inicio e término da execução)
  - **Tempo de espera** → soma dos períodos em que um processo fica esperando na fila de prontos.
  - **Tempo de resposta** → intervalo de tempo entre o envio de uma requisição e a primeira resposta a requisição (em sistemas interativo, o tempo de processamento pode não ser o melhor critério a ser usado)

Critérios de Escalonamento

- Desejamos → maximizar a utilização da CPU e a produtividade e minimizar os tempos de processamento, de espera e de resposta.
- Vamos conseguir tudo isso???

Algoritmos de Escalonamento

- Trata do problema de decidir qual dos processos na fila de processos prontos vai ser executado pela CPU
- Existem vários algoritmos diferentes de escalonamento
- Vamos estudar alguns deles:

Algoritmos de Escalonamento
First-In, First-Out (FIFO)

- Primeiro a chega, primeiro a ser servido
- Muito simples
- O primeiro processo a entrar na fila de prontos será o primeiro a ser executado.
- Quando um processo entra na fila de prontos, seu PCB é colocado no final da fila.
- Quando a CPU está livre, ela é alocada ao processo descrito pelo PCB que está no inicio da fila.

Algoritmos de Escalonamento
First-In, First-Out (FIFO)

- Tempo de resposta é sempre muito longo
- Exmplo: três processo entram na fila de processos prontos na ordem P1, P2 e P3.

Processo	Tempo de CPU
P_1	24
P_2	3
P_3	3

DIAGRAMA DE GANTT

Algoritmos de Escalonamento
First-In, First-Out (FIFO)

- O tempo de espera é de 0 milissegundos para o processo P1, 24 milissegundos para o processo P2 e 27 milissegundos para o processo P3.
- Tempo de espera médio é de (0 + 24 + 27) / 3 = 17 milissegundos.
- Exercício: os processo entram na fila na ordem P2, P3 e P1. Qual o resultado?

Algoritmos de Escalonamento
First-In, First-Out (FIFO)

- Problemas com o FIFO:
  - Algoritmo não-preemptivo
  - Os processo podem ficar em um cenário de comboio… (processos I/O bound esperam pelos processo CPU bound)
  - Desastroso para sistemas de tempo compartilhado.

Algoritmos de Escalonamento
Menor-Primeiro

- Associa a cada processo duração de sua próxima fase de uso da CPU (chamamos de D_P)
- A CPU é alocada, quando possível, ao processo com menor duração da próxima fase de uso da CPU.
- Se dois processos têm o mesmo tempo D_P → FIFO é adotado

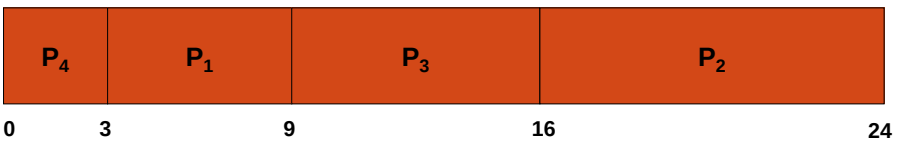
Algoritmos de Escalonamento
Menor-Primeiro

- Exemplo: considere os processos P_1, P_2, P_3 e P_4, com os seguintes valores para D_{P1}, D_{P2}, D_{P3} e D_{P4} dados em milissegundos:

Processo	Duração da próxima fase de uso da CPU
P_1	$D_{P1} = 6$
P_2	$D_{P2} = 8$
P_3	$D_{P3} = 7$
P_4	$D_{P4} = 3$

Algoritmos de Escalonamento
Menor-Primeiro

- Diagrama de Gantt do algoritmo MP:



Tempo de espera P_1 → 3 ms
Tempo de espera P_2 → 16 ms
Tempo de espera P_3 → 9 ms
Tempo de espera P_4 → 0 ms

Tempo de espera médio → (3+16+9+0)/4 = 7 ms

Algoritmos de Escalonamento
Menor-Primeiro

- Se estivéssemos usando o esquema FIFO, o tempo de espera médio seria de 10,25 ms
- O algoritmo MP é comprovadamente ótimo, fornecendo o menor tempo médio de espera para um determinado conjunto de processos
- Problema: como determinar o valor de D_P !!!
- Dificilmente pode ser implementado → não há como saber, para um dado processo, a duração de sua próxima fase de uso da CPU.

Algoritmos de Escalonamento
Menor-Primeiro (preemptivo)

- MP pode ser tanto preemptivo quanto não-preemptivo
- Fila de prontos → chega um processo com menor tempo de duração da fase de uso da CPU
- Neste caso, o processo em execução (com maior D_P) perde a CPU (ou seja, é interrompido)

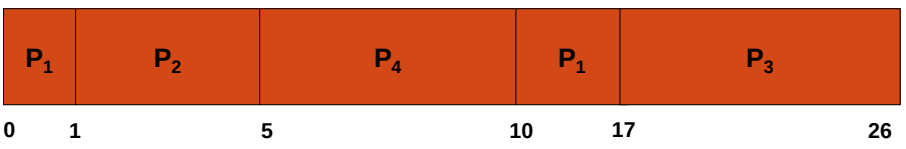
Algoritmos de Escalonamento
Menor-Primeiro (preemptivo)

- Exemplo: considere quatro processos com a duração da fase de uso da CPU em milissegundos:

Processo	Tempo de chegada	D_P
P_1	0	$D_{P1} = 8$
P_2	1	$D_{P2} = 4$
P_3	2	$D_{P3} = 9$
P_4	3	$D_{P4} = 5$

Algoritmos de Escalonamento
Menor-Primeiro

- Considerando a entrada dos processos na fila de processos pronto (tempo de chegada) e a duração de suas fases de uso da CPU, a alocação MP preemptiva é feita conforme o Diagrama de Gantt:



Algoritmos de Escalonamento
Menor-Primeiro

- O processo P_1 é iniciado no tempo 0, pois é o único processo na fila
- Processo P_2 chega no tempo 1
- O tempo restante para a execução de P_1 (7 ms) é maior do que o tempo requisitado por P_2 (4 ms) → P_1 é interrompido e a CPU alocada ao processo P_2
- Tempo de espera médio → ((10-1)+(1-1)+(17-2) + (5-3))/4 = 26/4 = 6,5

Algoritmos de Escalonamento
Prioridade

- MP → algoritmo de alocação por prioridade na qual a prioridade (p) é o inverso da duração prevista da próxima fase de uso da CPU. (maior a duração, mais baixa a prioridade e vice-versa)
- Prioridades → podem ser altas ou baixas
- São normalmente números em um determinado intervalo, com 0 a 7, ou 0 a 4095. (0 pode ser alta ou baixa !!!)

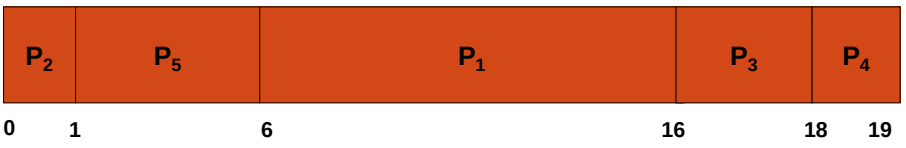
Algoritmos de Escalonamento
Prioridade

- Exemplo: considere o seguinte conjunto de processos, supondo que tenham entrado na fila de processos prontos em um tempo inicial 0, na ordem P_1,P_2, …, P_5

Processo	D_P	Prioridade
P_1	$D_{P1} = 10$	3
P_2	$D_{P2} = 1$	1
P_3	$D_{P3} = 2$	3
P_4	$D_{P4} = 1$	4
P_5	$D_{P5} = 5$	2

Algoritmos de Escalonamento
Prioridade

- Diagrama de Gantt:



Tempo de espera médio é de 8,2 ms

Algoritmos de Escalonamento
Prioridade

- **Prioridades** → podem ser definidas tanto interna quanto externamente
- **Prioridades interna** → são definidas de acordo com alguma ou algumas quantidades mensuráveis (limites de tempo, requisitos de memória, número de arquivos abertos e a razão entre uso de E/S e uso de CPU)

Algoritmos de Escalonamento
Prioridade

- **Prioridades externas** → são determinadas por critérios externos ao SO (importância do processo, tipo e a quantidade de capital usado para pagar o computador …)
- **Alocação por prioridade** → pode ser tanto preemptiva quanto não-preemptiva

Algoritmos de Escalonamento
Prioridade

- **Alocação preemptiva** → interrompe um processo de baixa prioridade quando um outro processo de alta prioridade chega no sistema (entra na fila de processos prontos para executar)
- Problema → um processo pode ser abandonado !!!
- Este algoritmo pode deixar processos de baixa prioridade esperando indefinidamente pela CPU. (sistema com vários processos de alta prioridade)

Algoritmos de Escalonamento
Prioridade

- Solução para este problema do abandono → técnica de envelhecimento
- Conhecida como prioridade dinâmica → consiste em aumentar a prioridade dos processos que ficam em espera no sistema por um longo tempo
- Prioridade estática X dinâmica !!!

Algoritmos de Escalonamento
Round-Robin (Circular)

- Projetado especialmente para sistemas de tempo compartilhado (*time sharing*)
- Similar ao sistema de FIFO, mas adiciona-se preempção ao algoritmo.
- Define-se um intervalo de tempo → quantum de tempo que varia geralmente de 10 a 100 milissegundos

Algoritmos de Escalonamento
Round-Robin (Circular)

- Fila de processo prontos → FIFO
- Novos processos são adicionados no final da fila
- O escalonador da CPU escolhe o primeiro processo da fila de prontos, prepara o temporizador para interromper depois do quantum de tempo e inicia a execução do processo

Algoritmos de Escalonamento
Round-Robin (Circular)

- Duas situações podem ocorrer:
  - Processo termina sua execução ante do fim do ser quantum de tempo → processo libera a CPU de forma voluntária e escalonador escolhe um outro processo
  - Processo tem fase de uso de CPU muito longa, ou seja, maior que o quantum de tempo determinado pelo escalonador de processos → temporizador causará uma interrupção (timer) tratada pelo SO (troca de contexto será executada e o processo será colocado no final da fila de processos prontos, e o escalonador selecionará então o próximo processo a ganhar a CPU

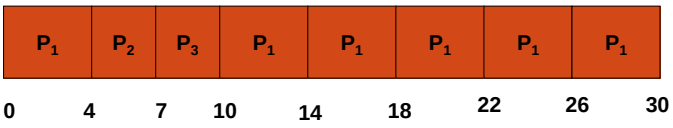
Algoritmos de Escalonamento
Round-Robin (Circular)

- Exemplo: considere o seguinte conjunto de processos que entram na fila de processos prontos em um tempo inicial 0

Processo	Duração da próxima fase de uso da CPU
P_1	$D_{P1} = 24$
P_2	$D_{P2} = 3$
P_3	$D_{P3} = 3$

Algoritmos de Escalonamento
Round-Robin (Circular)

- Vamos usar um quanto de tempo de 4ms
- Diagrama de Gantt



Tempo de espera médio → 17/3 = 5,66 ms

Algoritmos de Escalonamento
Round-Robin (Circular)

- Desempenho do RR depende fortemente do valor do quantum de tempo
- Tempo de troca de contexto → fator importante !!!
- Este tempo de troca mais o quantum de tempo são fatores importantes neste tipo de alocação (*time sharing*)
- Se este for muito baixo, diversas trocas de contexto entre processos ocorrerão tornando a CPU bastante ociosa.

Algoritmos de Escalonamento
Round-Robin (Circular)

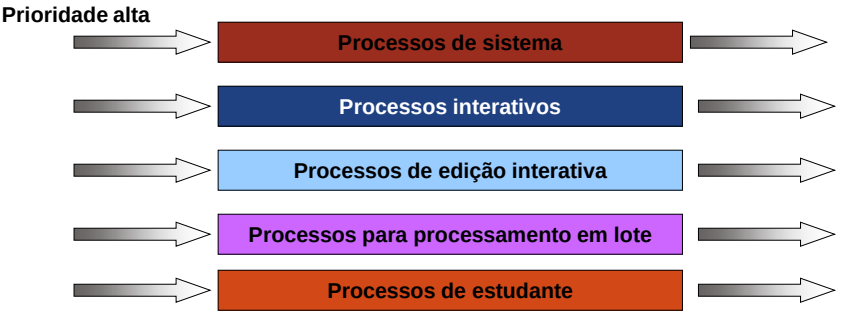
- Se for muito grande, o tempo de resposta irá aumentar degradando o atendimento de serviços interativos.
- Neste tipo de alocação todos os processos possuem igual importância !!!

Algoritmos de Escalonamento
Vária Filas (Multiples Queues)

- Processos podem ser classificados facilmente em diferentes grupos.
- Processos interativos (***foreground***) são normalmente separados de processos não-interativos (***background***)
- Processos interativos e I/O-boud → usam pouco a CPU (justiça !!!)
- Processos não-interativos → normalmente consomem mais CPU (CPU-bound)

Algoritmos de Escalonamento
Vária Filas (Multiples Queues)

- Neste algoritmos a fila de processos prontos é dividida em vário níveis, constituindo filas separadas.



Algoritmos de Escalonamento
Vária Filas (Multiples Queues)

- Processos entram em uma determinada fila de acordo com alguma prioridade (tamanho de memória necessário, prioridade do processo ou tipo) e ficam nesta fila até serem selecionados para execução
- Cada fila usa um determinado algoritmo para seleção (por exemplo FIFO e Round-Robin)

Algoritmos de Escalonamento
Vária Filas (Multiples Queues)

- Exemplo: cinco filas conforme a figura apresentada
- Cada fila tem prioridade absoluta sobre as filas de menor prioridade
- Os tempos poderiam ser repartidos entre cada fila → cada fila teria um quantum de tempo

Algoritmos de Escalonamento
Vária Filas Com Realimentação

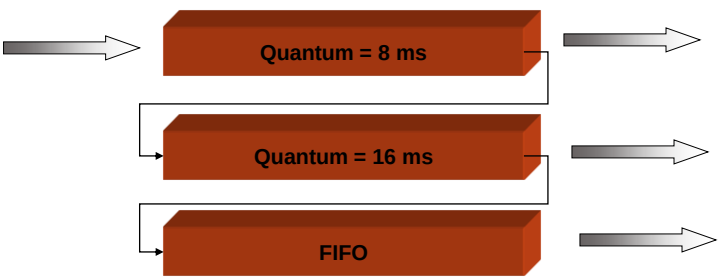
- Algoritmo anterior → processo entra em uma determinada fila e aí permanecem (esquema não é flexível)
- A alocação com várias filas com realimentação permite que os processos sejam transferidos entre as diversas filas
- Idéia → separar os processos com características diferentes de uso de CPU

Algoritmos de Escalonamento
Vária Filas Com Realimentação

- Processo quer usa CPU por muito tempo poder ser transferido para um fila de mais baixa prioridade
- Este processo mantém nas filas de prioridades mais altas os processos interativos e os processos que dependem mais de E/S
- Processo que fica muito tempo em uma fila de baixa prioridade pode ser promovido a uma fila de alta prioridade → evita que os processo fiquem abandonados !!!

Algoritmos de Escalonamento
Vária Filas Com Realimentação

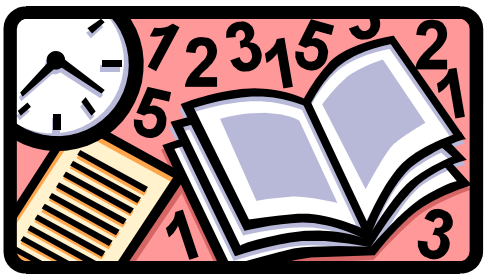
- Filas com vários níveis e transferências entre elas:



Algoritmos de Escalonamento
Vária Filas Com Realimentação

- Parâmetros que definem este algoritmo:
  - Números de filas
  - Algoritmo de alocação da CPU usado para cada fila
  - Método usado para determinar quando transferir um processo para uma fila de prioridades mais alta
  - Método usado para determinar quando transferir um processo para uma fila de prioridades mais baixa
  - Método usado para determinar em que fila um processo deve ser colocado, quando precisar usar a CPU

Exercícios



Alocação de Vários Processadores

- Até este momento → sistemas monoprocessados
- Várias CPU's → maior complexidade !!!
- Sistemas multiprocessados → temos várias CPU's de forma homogênea ou heterogênea
- Sistemas homogêneos → os processadores são idênticos no que diz respeito à sua funcionalidade

Alocação de Vários Processadores

- Modelo homogêneo → qualquer CPU que está disponível pode ser alocada
- Limitação: dispositivo de E/S ligado a um barramento reservado a um determinado processador
  - Os processos que querem usar este dispositivo só podem ser alocados neste processador

Alocação de Vários Processadores

- Vários processadores idênticos → compartilhamento de tarefas
- **Modelo 1** → cada CPU têm a sua fila de processos e seu próprio algoritmo de alocação

Alocação de Vários Processadores

- **Modelo 1** → Problemas !!
  - Uma das CPU's pode ficar ociosa, com fila vazia, enquanto uma CPU está com a fila cheia
  - Deve ser programado com cuidado → as estruturas de dados do processo são compartilhadas e, assim, devem garantir que dois algoritmos não escolham o mesmo processo e que nenhum processo seja incorretamente removido da filha

Alocação de Vários Processadores

- **Modelo 2** → Todos os processos entram em uma única fila de processos e são selecionados para execução em qualquer CPU disponível
  - Este modelo evita o problema do segundo
  - Neste modelo, o algoritmo de escalonamento é executado por uma CPU reservada para este fim → estrutura Mestre/Escravo

Alocação de Vários Processadores

- Multiprocessamento Assimétrico é mais simples do que Multiprocessamento Simétrico (symmetric multiprocessor – SMP)
  - O acesso a estrutura de dados do sistema é feito em apenas uma CPU, diminuindo o compartilhamento de dados

Alocação em Tempo Real
Real Time

- Sistemas de tempo real → SO utilizado com um propósito específico
- RTOS (*Real Time Operating System*) → usado quando existem requisitos rígidos relativos ao tempo sobre a operação de um processador ou sobre o fluxo de dados
- Freqüentemente usado como um dispositivo de controle em uma aplicação dedicada a um propósito específico

Alocação em Tempo Real
Real Time

- Algumas aplicações:
  - Experimentações científicas
  - Sistemas de geração de imagem para medicina
  - Sistemas de controle industrial
  - Sistema de injeção de combustível no motor
  - Embedded Systems
  - Etc …

Alocação em Tempo Real
Real Time

- Tipos de Sistemas Operacionais de Tempo Real (RTOS):
  - Sistemas de tempo real com restrições rígidas (*Hard Real Time*)
  - Sistemas de tempo real flexíveis (*Soft Real Time*)

Alocação em Tempo Real
Hard Real Time

- Princípio básico → limite rígido de tempo
- Especificação do processo:
  - Tempo total necessário para o seu processamento ou;
  - Tempo total necessário para realização de operações de E/S
- Escalonador pode permitir ou rejeitar a execução do processo → reserva de recurso
- Garantia do escalonador:
  - Tempo de execução de cada função do SO

Alocação em Tempo Real
Hard Real Time

- Problema: para alguns dispositivos fica praticamente impossível definir este tempo !!!
- Na prática: temos programas de propósitos específicos, sendo executados em processadores dedicados à sua execução, e sem a funcionalidade completa de computadores e SO's modernos

Alocação em Tempo Real
Soft Real Time

- Princípio básico → limite de tempo menos restritivo
- Existem processos críticos que tem mais prioridade sobre os outros processos
- Sistemas *Time-Sharing* → incorporam funcionalidades de um sistema *Soft Real Time*

Alocação em Tempo Real
Soft Real Time

- Problema → Estes sistemas podem gerar as seguintes anomalias:
  - Alocação injusta de recursos
  - Tempo de espera
  - Abandono de processo
- Resultado positivo: tem-se um sistema de propósito geral que suporta também processamento de imagem e som, gráficos interativos de alta velocidade e tarefas *Soft Real Time*

Alocação em Tempo Real
Soft Real Time

- Implementação → requer um projeto cuidadoso do escalonador e aspectos relacionados a ele
- Aspectos gerais (primeira condição):
  - Sistema deve ter alocação por prioridade e os processos RT devem ter prioridades mais altas
  - Processo RT → não pode ter suas prioridades diminuídas
- Aspecto geral (segunda condição):
  - **O tempo de latência do *dispatcher*** (tempo gasto para iniciar a execução de processos que está pronto) deve ser o menor possível

Alocação em Tempo Real
Soft Real Time

- Técnica de envelhecimento (primeira condição) →não pode ser aplicada para processos RT
- Segunda condição → mais problemática
- Problema: Muitos SOs e a maioria das versões do UNIX
  - São forçados a esperar o término de uma chamada de sistema ou aguardar até que o processo seja bloqueado ao requisitar um operação de E/S, antes de realizar uma mudança de contexto → **tempo de latência muito grande !!!**

Alocação em Tempo Real
Soft Real Time

- Tentativa de reduzir este tempo:
  - Permitir que as chamadas de sistema possam ser interrompidas
  - Uma forma é definir pontos de preempção no núcleo (kernel) do SO (colocando apenas em locais "seguros")
  - Na prática: existem poucos pontos de preempção do kernel do SO
  - Conclusão: mesmo com pontos de preempção o tempo de latência pode ser grande

Alocação em Tempo Real
Soft Real Time

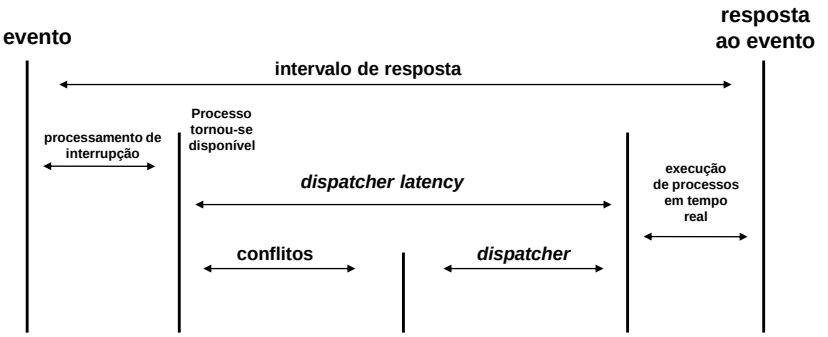
- Outro método:
  - Chamadas do sistema podem ser interrompidas em qualquer ponto
  - As estruturadas de dados dos processo (ou compartilhadas) devem ser protegidas usando mecanismos de sincronização
  - Este método é usado pelo Sistema Operacional SOLARIS da Sun
  - Problema: Inversão de prioridade → processo de baixa prioridade está alterando a ED do núcleo e um outro processo de alta prioridade (RT) deseja fazer a mesma coisa

Alocação em Tempo Real
Soft Real Time

- Forma de solucionar este problema de inversão de prioridade:
  - Uso de protocolo de herança de prioridade →todos os processos (aqueles que estejam usando recursos de que o processo de alta prioridade necessita) recebem prioridades alta até que sejam terminadas todas as operações realizadas pelo processo
  - Depois de realizar a operação → a prioridade do processo volta ao seu valor inicial

Alocação em Tempo Real
Soft Real Time

- Tempo de latência para inicialização de um processo



Alocação em Tempo Real
Soft Real Time

- Fase de conflito do tempo de latência têm três componentes:
  - Preempção de qualquer processo em execução
  - Processos de menor prioridade liberando recursos necessário para processo de alta prioridade
  - Mudança de contexto do processo que está executando para o processo de alta prioridade

Alocação em Tempo Real
Soft Real Time

- Exemplo: Solaris 2
  - Tempo de latência com preempção desabilitada é maior que 100 milissegundos
  - Com preempção habilitada é normalmente reduzido para 2 milissegundos