Escalonamento (Scheduling)

Problema de Escalonamento

- Dado um conjunto de processos executáveis, qual a ordem com que receberão o CPU (e durante quanto tempo)?

- Necessita da definição de uma politica

- ***Não é fácil***
  - Precisamos definir o que queremos otimizar
  - Como medir
  - Como atuar sobre os processos
  - Politicas complexas podem ter um custo muito elevado: quando o núcleo corre os processos não progridem…

Que métricas otimizar?

- **Débito (Throughput):** o número máximo de jobs por hora
- **Turn around time**: tempo entre a submissão do job e a obtenção do resultado
- **Utilização do processador**: % de tempo em que o processador esteve em uso útil
- **Tempo de resposta**: responder rapidamente aos eventos desencadeados pelos utilizadores
- **Cumprir metas temporais (deadlines)** para tratamento dos acontecimentos
- **Execução com desempenho previsível** (e.g., multimedia, máquinas virtuais na *cloud*)

Dependendo do contexto, alguns objetivos são mais relevantes que outros, sendo difícil compatibilizá-los

Algumas politicas de Escalonamento

- Batch – tratamento por lotes
  - Produtividade - Throughput – maximizar o número máximo de jobs por hora.
  - *Turn around time* – tempo entre a submissão do trabalho e a obtenção do resultado.
  - Utilização do processador – manter o processador com elevada ocupação
- Utilizadores interativos em tempo partilhado
  - Tempo de resposta – responder rapidamente aos eventos desencadeados pelos utilizadores
- Tempo real
  - Cumprir metas temporais (*deadlines*) para tratamento dos acontecimentos
  - Funcionamento com desempenho previsível (e.g., multimedia)

Escalonamento nos SO tradicionais

- Não vamos tratar do Batch e do Tempo Real estrito

- Vamos concentrar-nos no tempo partilhado, o mais usual em Unix, Windows, IOS, Android que nas versões mais recentes já têm possibilidade de escalonamento em tempo real relaxado

Classes de processos

- Tipicamente, há duas classes de processos:
  - CPU-intensivos
    - Uso intensivo do processador, raramente se bloqueiam
  - E/S-intensivos
    - Uso intensivo das E/S, quase sempre bloqueados
- Exemplos de cada classe?

Deve a classe do processo influenciar o acesso ao CPU?

- Num dado momento, há um processo CPU-intensivo e outro E/S-intensivo executáveis
- Qual deve ter maior prioridade no acesso ao CPU?
  - O processo E/S-intensivo!
- Ok, mas como saber a classe de cada processo?
  - Exigir que o programador/utilizador a indique?
  - Mais interessante: inferir a classe estudando o comportamento recente do processo

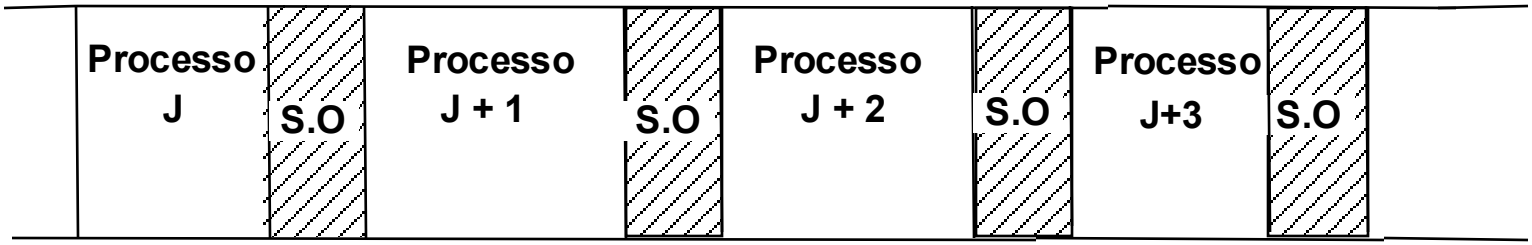
Políticas de Escalonamento em Sistemas de Tempo Partilhado

- Tempo de Execução Partilhado (*time-slices*)
- Prioridades
- Modificação dinâmica das prioridades
- Preempção

Tempo de Execução Partilhado: Time Slices

Objectivo:

Permitir que todos os processos executáveis tenham oportunidade de dispor do processador ciclicamente



Tempo de Execução Partilhado: Time Slices

- Como funciona:
  - Processos executáveis mantidos em fila FIFO
  - Processo no início da fila é colocado em execução até um tempo máximo *(time slice* ou *quantum)*
  - Processo perde o CPU para o próximo quando:
    - *Time slice* expira (via interrupção do *timer*)
    - Chama *sys call* que o bloqueia
  - Em ambos os casos, o processo retorna ao fim da fila (no caso de bloqueio, só depois de se desbloquear)

Desvantagens?

Duração do *time slice*

- Qual deve ser a duração do *time slice*?

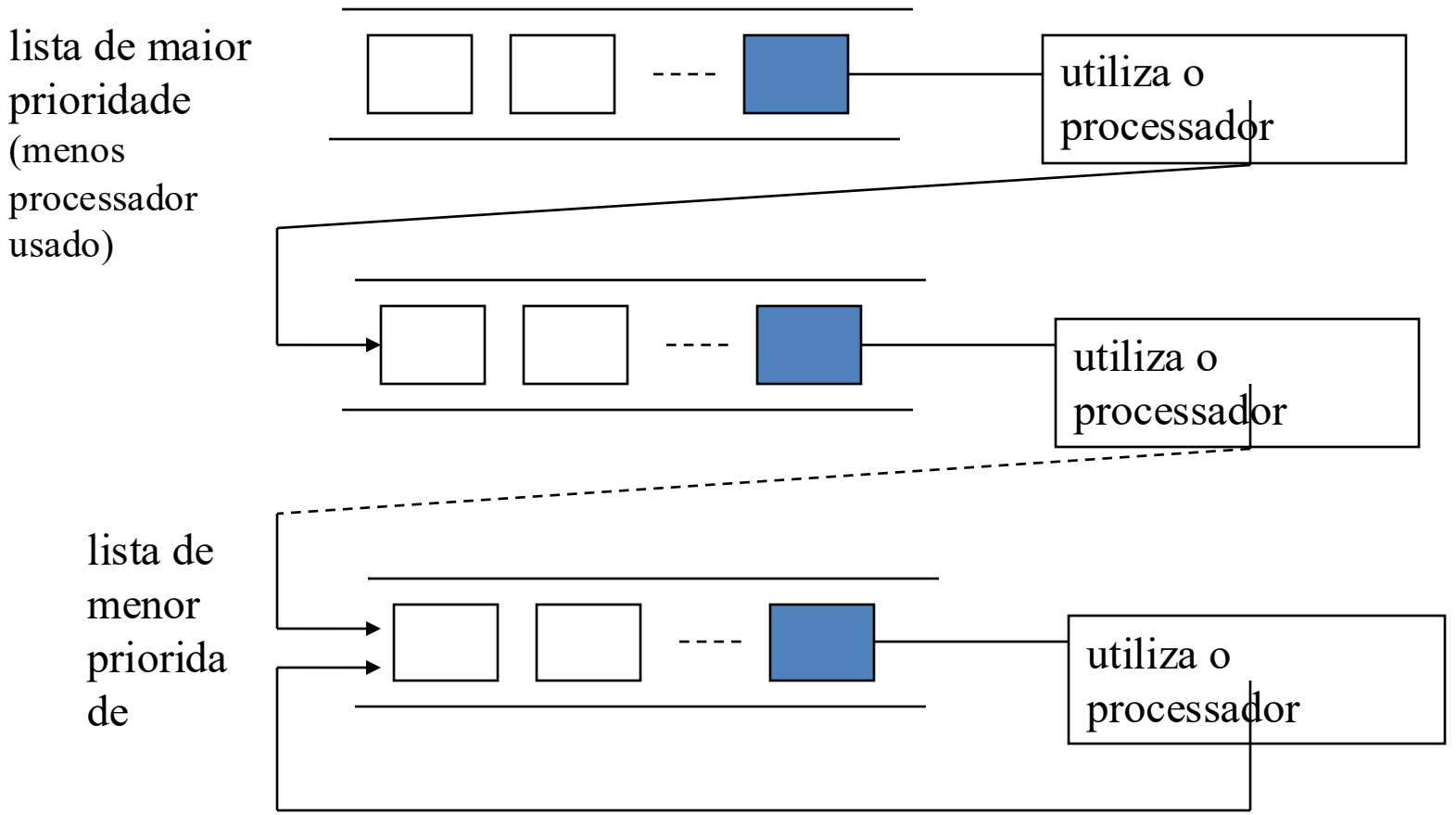
- Muito pequeno: bom para processos interativos, mas aumenta muito o tempo de multiplexagem

- Duração significativa (ex: 200 ms): bom para CPU-intensivos, mas pode retirar a interatividade ao sistema

Escalonamento com prioridades

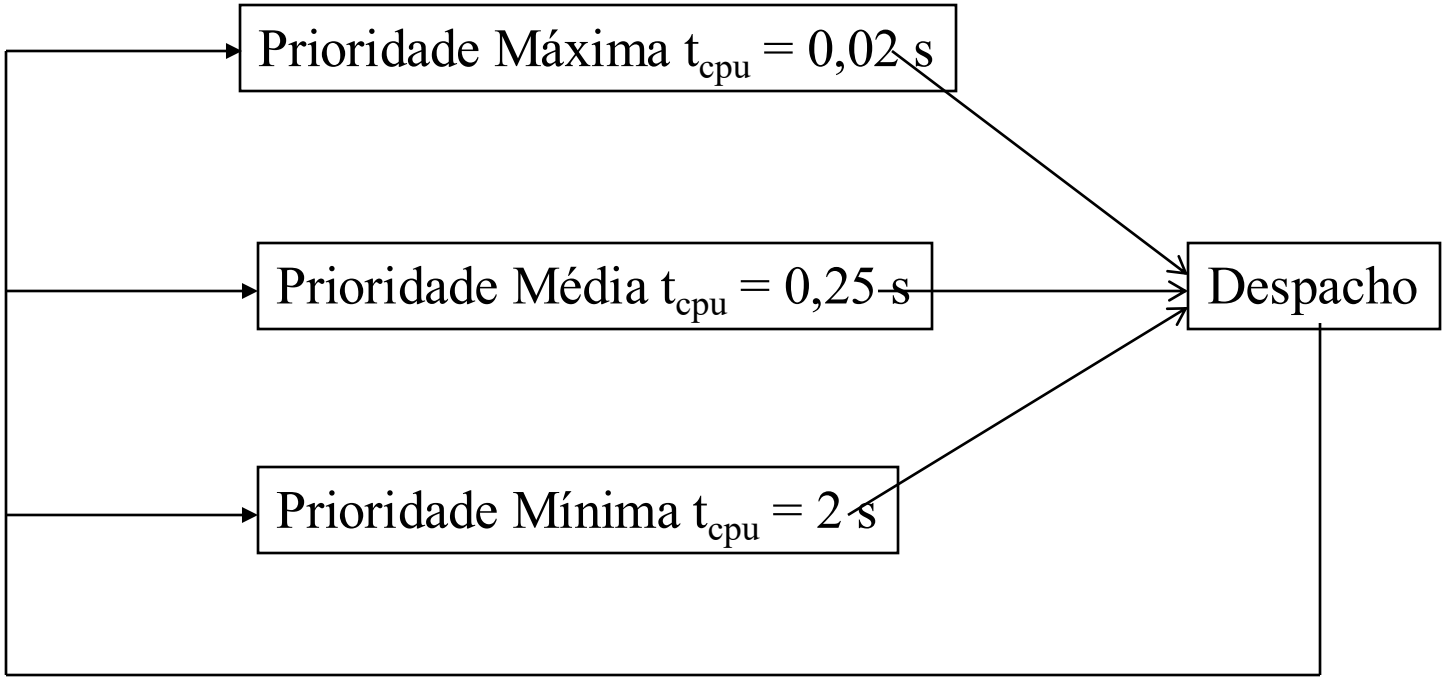
- As prioridades permitem definir a importância de um processo no processo de escalonamento
- Um processo mais prioritário tem maior probabilidade de dispor do processador
- A prioridade pode ser
  - Fixa, usual em processos de tempo real
  - Dinámica consoante o comportamento do processo:
    - usual nos sistemas de tempo virtual e normalmente privilegiando os processo interactivos (I/O intensivos).

Gestão Multilista



Em que situações um processo deve ser promovido? E relegado?

Gestão Multilista com *Quantum* Variável



- Adaptar o valor do quantum ao comportamento dos processos
- Diferentes estratégias possíveis
  - Alguns escalonadores aumentam quantum de processos de menor prioridade
  - Outros dão maior quantum a processos de maior prioridade

Preempção em teoria

- **O que é?** Retirar o processador a um processo em execução **assim que** existe outro processo mais prioritário executável
- **Objectivo?** permite que os processos mais prioritários reajam rapidamente a um dado acontecimento
  - Por exemplo, mais prioritário estava bloqueado mas acontecimento externo desbloqueou-o
  - Assim que processo receber execução, pode reagir ao acontecimento

Preempção na prática

- Em teoria, o despacho deve ser chamado na sequência de qualquer ação suscetível de modificar os estado dos processos
  - Desvantagem: Pode levar a mudanças frequente de contexto
- Na prática, "pseudo-preempção":
  - Assegurar que processo menos prioritário só perde processador após tempo mínimo
  - Em algumas situações, processo menos prioritário não perde CPU (por exemplo quando está a executar rotina do núcleo)

Escalonadores de hoje em dia (I)

- Multi-fila com prioridades variáveis + fixas, pseudo-preemptivos, quantum variável
- Escalonador gere tarefas (não processos), cada uma com a sua prioridade
- Afinidade entre tarefas do mesmo processo é explorada
  - Comutação para tarefa do mesmo processo é priorizada

Escalonadores de hoje em dia (II)

- Escalonamento multi-core
  - Tarefas agrupadas por CPU, tentando:
    - balancear carga entre CPUs
    - colocar tarefas do mesmo processo no mesmo CPU ou CPUs próximos
  - Cada CPU tem a sua instância de escalonador (e respetiva multi-fila)
  - Ocasionalmente, tarefas migram para outro CPU
    - Para rebalancear o Sistema