



**UNIVERSIDADE FEDERAL DE UBERLÂNDIA**  
**FACULDADE DE COMPUTAÇÃO**  
**CURSO: CIÊNCIA DA COMPUTAÇÃO**

**GERÊNCIA DE PROCESSADOR**

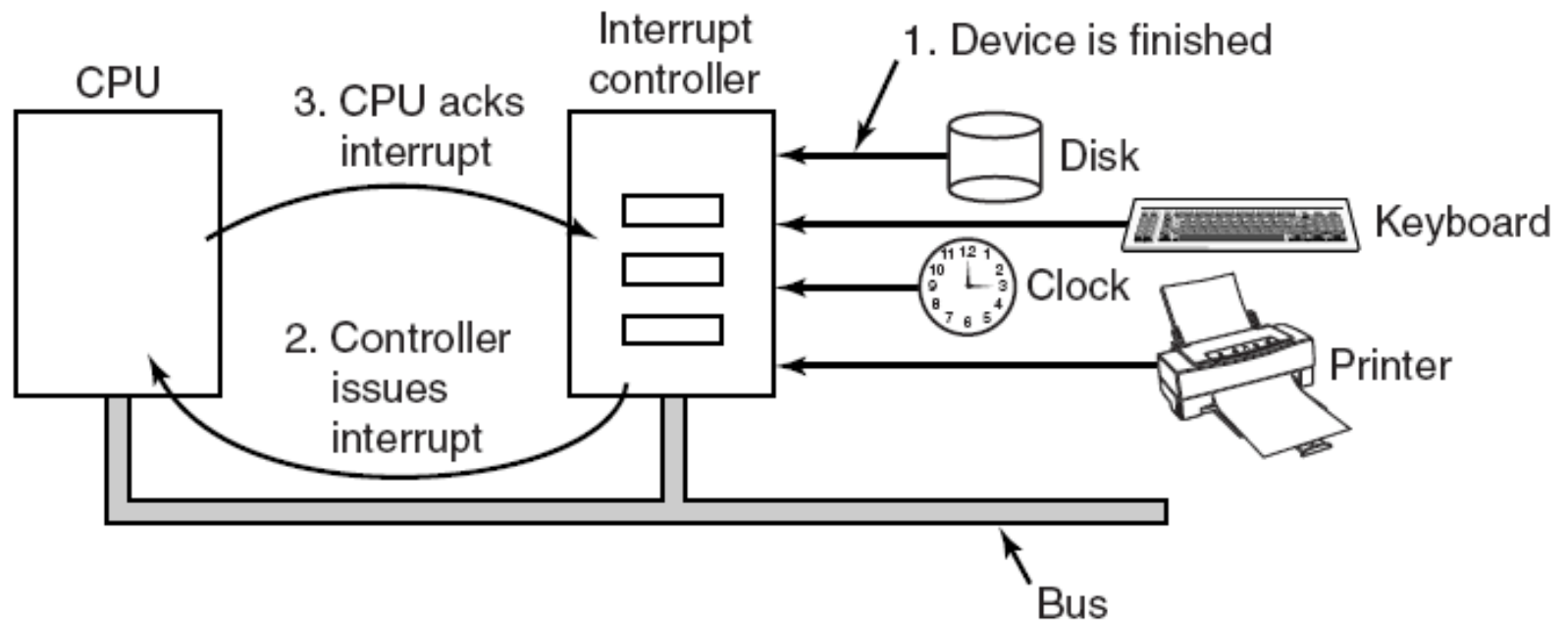
# **SISTEMAS OPERACIONAIS**

Prof. Rivalino Matias Jr

# AGENDA

- ✓ Mecanismo de Interrupção do Processador (*Timer Interrupt*)
- ✓ Tipos de Processos
- ✓ Escalonamento Não-Preemptivo
  - ✓ FIFO
  - ✓ SJF (*Shortest-Job-First*)
  - ✓ Escalonamento Cooperativo
- ✓ Escalonamento Preemptivo
  - ✓ Circular
  - ✓ Por Prioridade
  - ✓ Múltiplas filas
  - ✓ Múltiplas filas com realimentação
  - ✓ Tempo Real
- ✓ Escalonamento com Múltiplos Processadores

# Mecanismo de Interrupção



Fonte: Tanenbaum, 2008

# Mecanismo de Interrupção

1. Salva os registradores não salvos pelo hardware
2. Configura o novo contexto do tratador da interrupção
3. Configura a pilha para o código de tratamento da interrupção
4. Confirma interrupção para o controlador de interrupções (APIC).
5. Copia os registradores salvos para a PCB do processo
6. Executa o código do tratador da interrupção (*interrupt handler*)
7. Escolha qual o próximo processo será executado
8. Configura o SO para assumir o contexto do novo processo (espaço de endereçamento, pilha, etc.)
9. Carrega os registradores com os dados do novo processo, incluindo o PC (*program counter*)
10. Inicia a execução do novo processo.

# Escalonamento

- Tipos de Processos:
  - CPU Bound
  - I/O Bound
- Critérios de Escalonamento
  - Utilização da CPU
  - *Throughput* ( $nTasks/unid. tempo$ )
  - *Turnaround*
  - Tempo de Resposta

# Escalonamento

- Escalonador de processos:
  - Invocado a cada *n clock ticks*

```
Scheduler
{
  for ( todos processos na fila de pronto)
  {
    atualiza contador de uso do processador
    verifica expiração de timers
    atualiza prioridade dos processos
  }
  if (existe processo com maior prioridade na fila)
  {
    remove o processo de maior prioridade da fila
    salva o contexto do processo corrente (executando)
    carrega o contexto do processo de maior prioridade
  }
}
```

# Tipos de Escalonamento

- Não-preemptivo
  - O Processo executa na CPU até seu término ou repasse voluntário do recurso;
  - Não existe interferência do SO
- Escalonamento preemptivo
  - A execução dos processos é interrompida pelo SO;
  - Cada troca de contexto gera um custo (*overhead*) para o sistema;
  - O mecanismo utilizado é normalmente a interrupção de clock;
  - Políticas de escalonamento fornecem parâmetros para os mecanismos de escalonamento (*algoritmos*).

# Tipos de Escalonamento

- Não-preemptivo

- FIFO: Utilização da CPU por ordem de chegada na fila;
- SJF (*shortest-job-first*): Processos com menor tempo de execução são selecionados primeiro;
- Cooperativo: Os processos liberam voluntariamente a CPU. Ex. MS-Windows 3.1/3.11.

- Preemptivo

- *Round-robin* (circular): Baseado em *time-slice/quantum*;
- Prioridades: Permite diferenciar processos *I/O bound* de *CPU bound*. A prioridade pode ser estática ou dinâmica.
- Múltiplas Filas: Existem várias filas de processos prontos. Processos na mesma fila são tratados de forma igual, contudo, as filas entre si são tratadas em nível de prioridade. Ex. Filas de processos *I/O bound* tem maior prioridade que as filas de processo *CPU bound*.

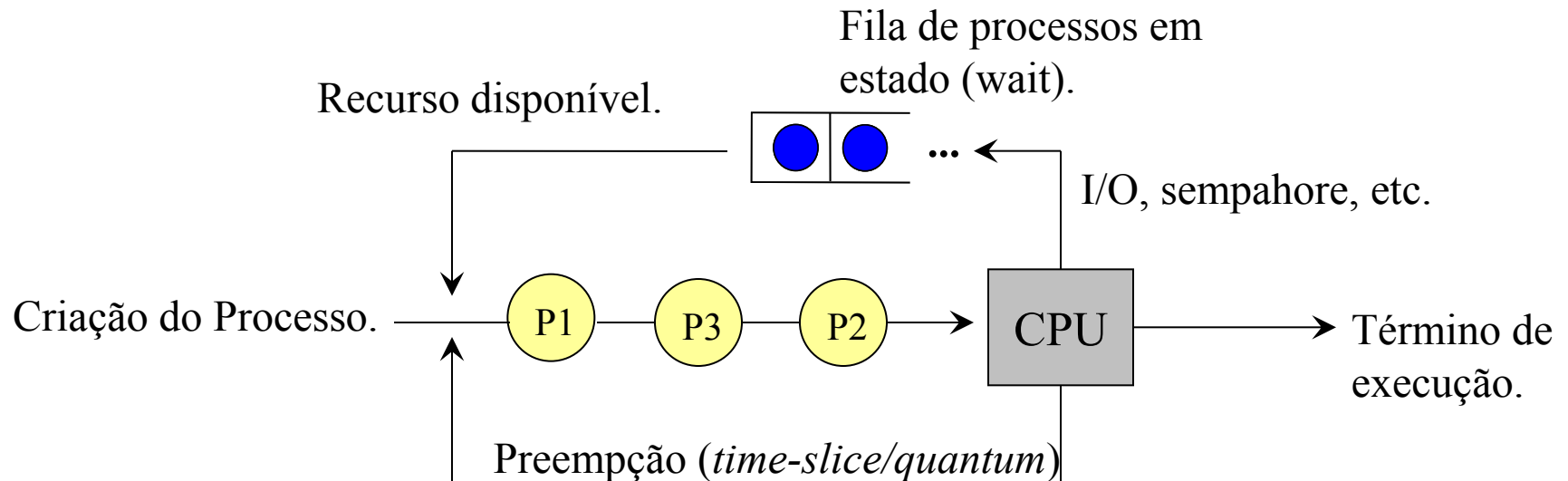


# Tipos de Escalonamento

- Preemptivo
  - *Round-robin* (circular):
    - Garante a execução das tarefas pela intervenção do SO, sendo esta sua principal vantagem em relação aos modelos não preemptivos;
    - Sua deficiência reside no tratamento não diferenciado entre processos *CPU-Bound* e *IO-Bound*;

# Preemptivo

- Round-robin/circular:



# Tipos de Escalonamento

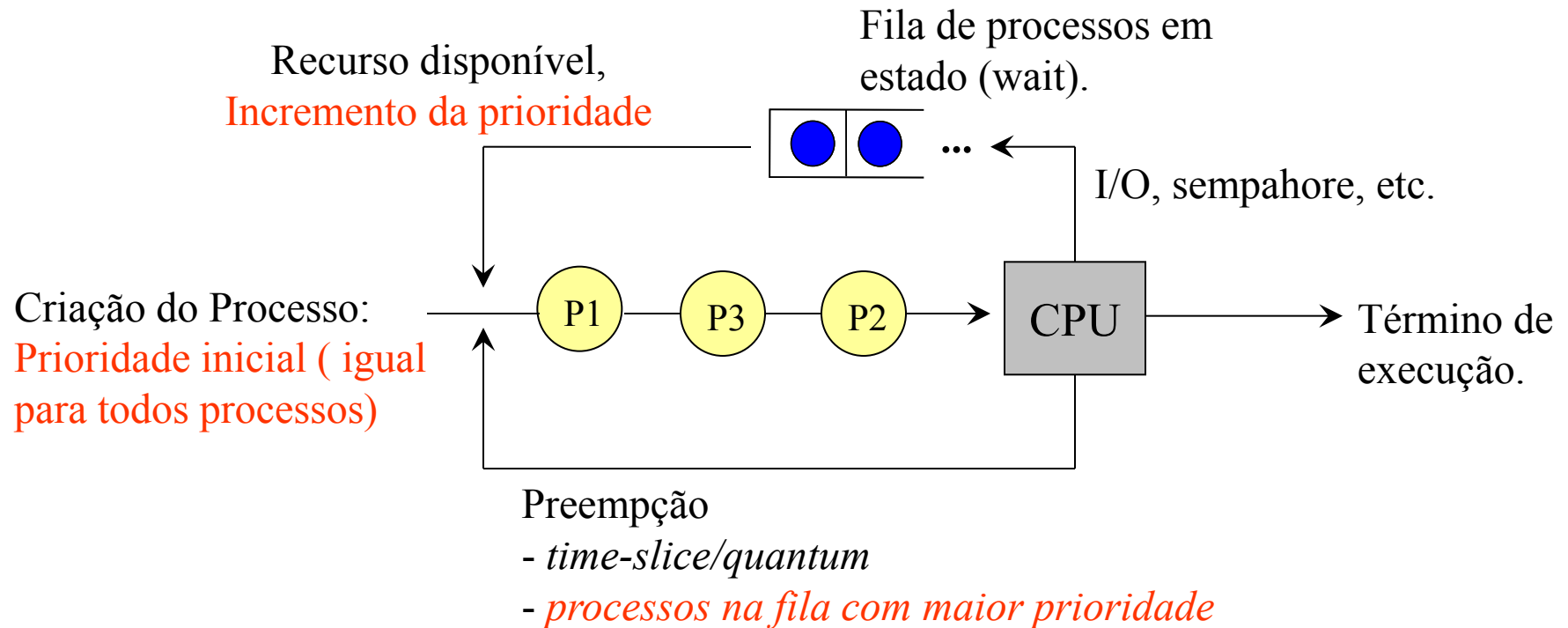
- Preemptivo

- Baseado em Prioridades:

- Processos *IO-Bound* possuirão maior prioridade em relação a processos *CPU-Bound*;
    - As prioridades podem ser:
      - Estáticas: Atribuídas na criação do processo permanecendo durante toda a vida do process;
      - Dinâmicas: Mudam de acordo com o comportamento do processo;
    - Exemplo:  $priority = (1 * quantum) / f$ , onde  $f$  = fração de tempo utilizada do *quantum*.
      - *PID (100 – CPUbound):*  $priority = (1 * 300) / 300 = 1$
      - *PID (340 – IObound):*  $priority = (1 * 300) / 002 = 150$
    - Em muitos sistemas *UNIX-like*, o comando *nice* é utilizado para alterar a prioridade de processos.

# Preemptivo

- Baseado em Prioridades:

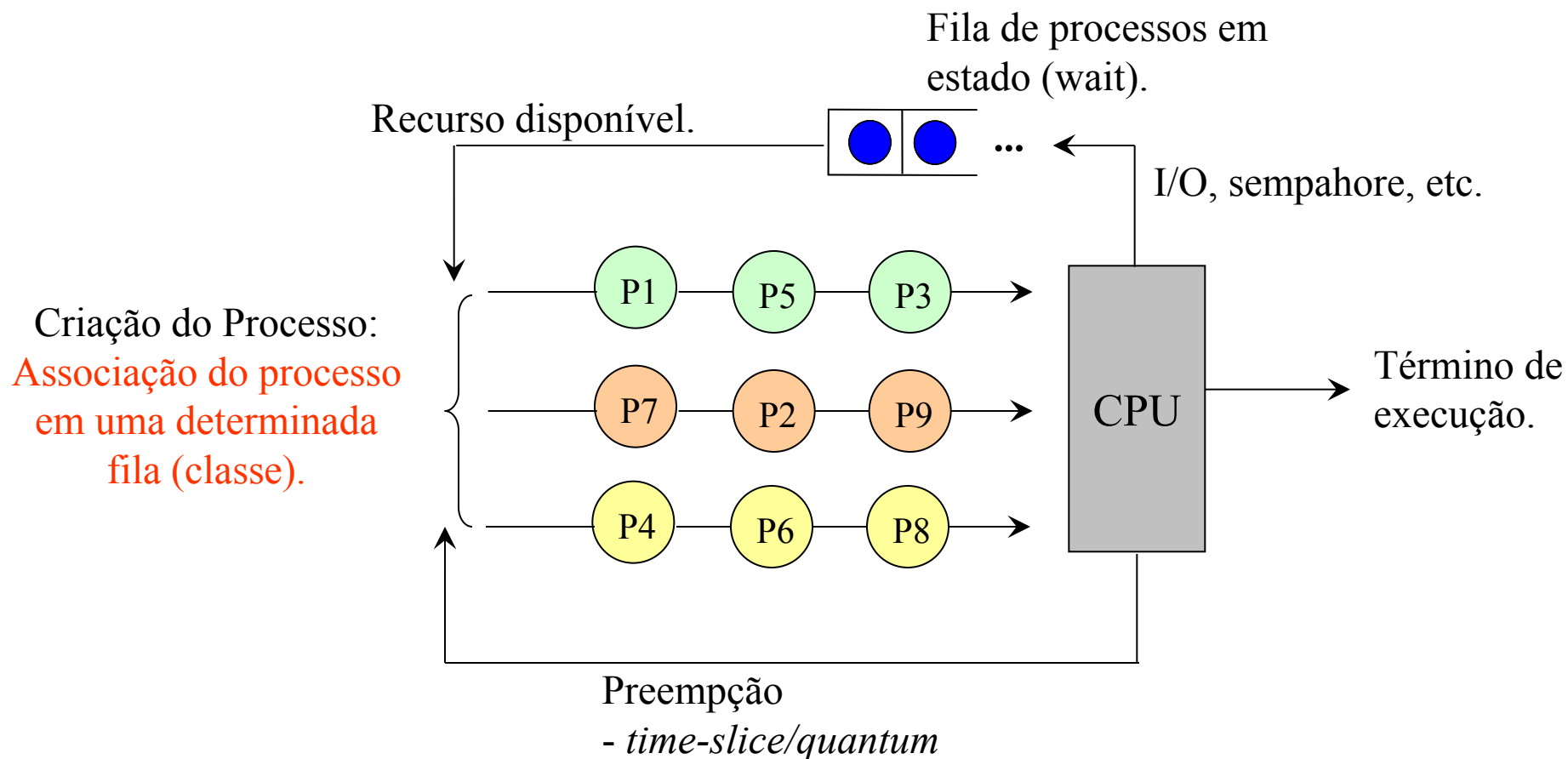


# Tipos de Escalonamento

- Preemptivo
  - Múltiplas filas (Corbat et al. 1962):
    - Existem várias filas de processos prontos para executar;
    - Cada processo é associado a uma fila de forma exclusiva;
    - Para cada fila é associada uma prioridade específica;
    - A execução de processos de uma fila inicia-se após o esvaziamento das filas de prioridades superiores;
    - Exemplo de classes de processos:
      - Sistema
      - Interativos
      - *Batch*

# Preemptivo

- Múltiplas Filas:

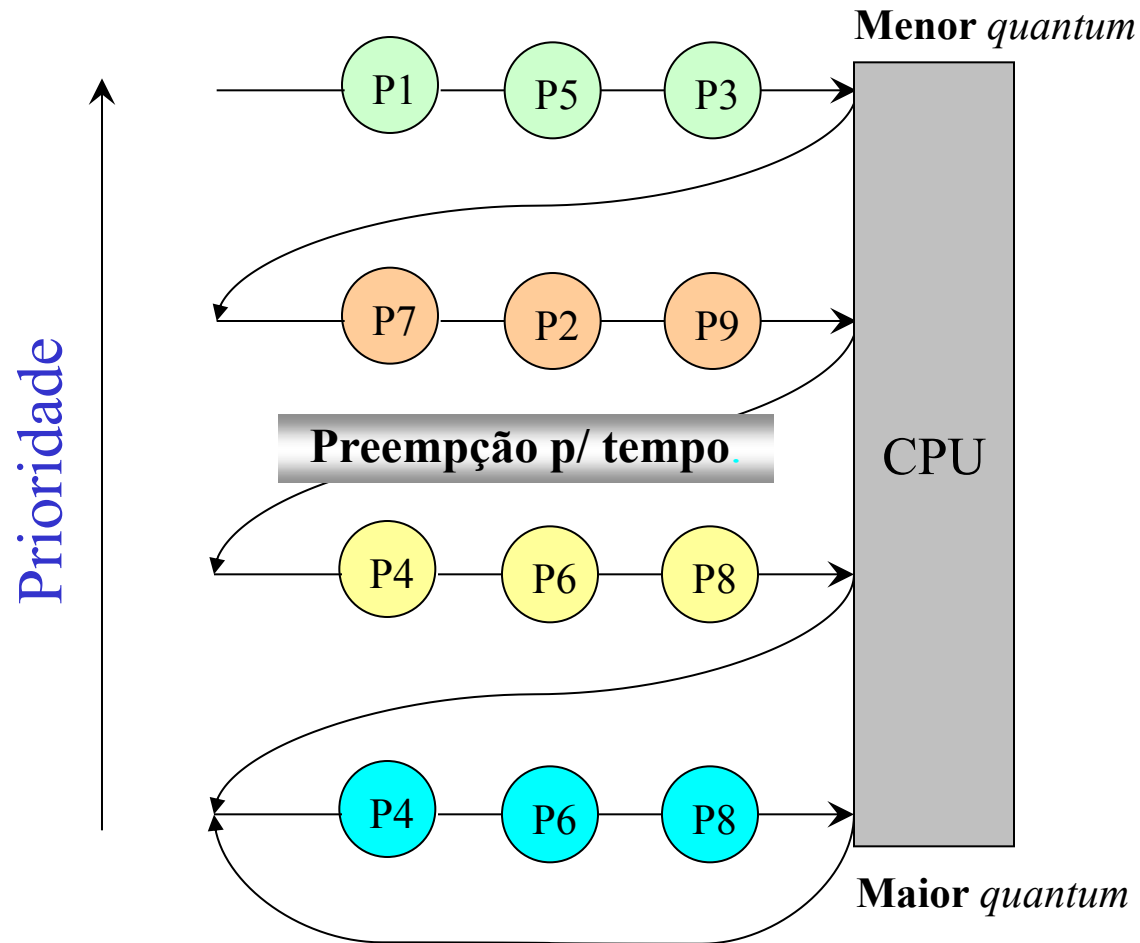


# Tipos de Escalonamento

- Preemptivo
  - Múltiplas filas com realimentação (*multi-level feedback*):
    - Existem várias filas de processos prontos para executar;
    - Para cada fila é associada uma prioridade específica;
    - A execução de processos de uma fila inicia-se após o esvaziamento das filas de prioridades superiores;
    - Os processos migram de uma fila para a outra, dependendo de seu comportamento.

# Preemptivo

- Múltiplas Filas c/ realimentação:





# Tipos de Escalonamento

- Preemptivo
  - Tempo Real (*Real Time - RT*):
    - Normalmente aplicados para controle de processos;
    - O principal requisito são os limites rígidos de tempo;
    - Utiliza prioridades e não *quantum/time slice*;
    - Pode se associar níveis de prioridade entre *threads* de controle;

# Tipos de Escalonamento

- Múltiplos Processadores
  - Fracamente acoplados:
    - SO local;
    - Balanceamento de carga;
    - Migração de processos;
  - Fortemente acoplados:
    - Fila única de processos prontos para todos os processadores;
    - O código do escalonador deve implementar exclusão mútua;
    - O processo pode executar em qualquer processador disponível (SMP);
    - Em caso de utilização de *threads* mapeadas no *kernel*, suas execuções podem ser em processadores distintos.