



## **Sistemas Operacionais**

### **Introdução a Arquitetura**

#### **Parte 03**

#### **“Gerência de Processador”**

Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro  
Prof. Edwar Saliba Júnior  
Janeiro de 2026



## Nota

- Com exceção das figuras que possuem referência bibliográfica, todo o restante do conteúdo destes *slides* foram extraídos do livro dos autores “Francis Berenger Machado” e “Luiz Paulo Maia”. Todos referenciados no último *slide* desta apresentação.



## Processos do Sistema Operacional

- O conceito de processo, além de estar associado a aplicações de usuários, pode também ser implementado na própria arquitetura do sistema operacional.
- Quando processos são utilizados para a implementação de serviços do sistema, retira-se código de seu núcleo, tornando-o menor e mais estável.



## Processos do Sistema Operacional

- Alguns serviços que o sistema operacional pode implementar através de processos:
  - auditoria e segurança,
  - serviços de rede,
  - contabilização do uso de recursos,
  - contabilização de erros,
  - gerência de impressão,
  - gerência de *jobs batch*,
  - Temporização,
  - comunicação de eventos e
  - interface de comandos (*shell*).





## Gerência de Processador

- Com o surgimento dos sistemas multiprogramáveis, onde múltiplos processos poderiam permanecer na memória principal compartilhando o uso da CPU, a gerência do processador tornou-se uma das atividades mais importantes em um sistema operacional;
- a partir do momento em que diversos processos podem estar no estado de pronto, devem ser estabelecidos critérios para determinar qual processo será escolhido para fazer uso do processador e
- os critérios utilizados para esta seleção compõem a chamada **política de escalonamento**, que é a base da gerência do processador e da multiprogramação em um sistema operacional.

## Gerência de Processador

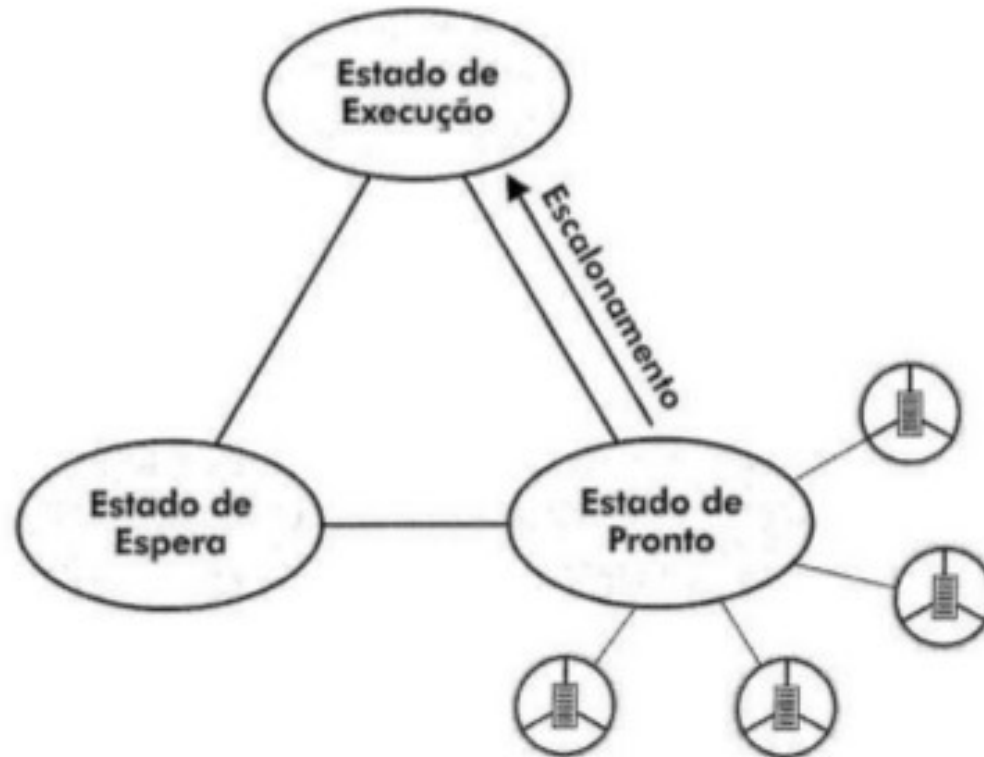


Figura 8.1: Escalonamento



## Critérios de Escalonamento

- **Throughput (taxa de transferência):** representa o número de processos executados em um determinado intervalo de tempo. Quanto maior o *throughput*, maior o número de tarefas executadas em função do tempo. A maximização do *throughput* é desejada na maioria dos sistemas.
- Comandos:
  - `$ iostat -dx 1` (estatísticas por dispositivo a cada 1 segundo)
    - `rkB/s, wkB/s` → throughput de leitura e escrita
    - `%util` → saturação/utilização do disco
  - `$ pidstat -u 1`
    - `%usr` (User CPU Time) - tempo de CPU gasto em modo usuário: execução do código-fonte do processo;
    - `%system` (System CPU Time) - tempo de CPU gasto em modo Kernel: leitura em disco, acesso à rede, alocação de memória, gerenciamento de arquivos, operações de drivers e etc. e
    - `%CPU` - uso total da CPU pelo processo: `(%usr + %system)`;
  - `C:\> typeperf "\PhysicalDisk(_Total)\Disk Read Bytes/sec" ^  
"\PhysicalDisk(_Total)\Disk Write Bytes/sec" ^  
"\PhysicalDisk(_Total)\% Disk Time"`



## Critérios de Escalonamento

- **Tempo de processador:** é o tempo que um processo leva no estado de execução durante seu processamento. As políticas de escalonamento não influenciam o tempo de processador de um processo, sendo este tempo em função apenas do código da aplicação e da entrada de dados e
- **tempo de espera:** é o tempo total que um processo permanece na fila de pronto durante seu processamento, aguardando para ser executado.
- **Comandos:**
  - \$ top
  - \$ htop
  - c:\> tasklist



## Critérios de Escalonamento

- **Tempo de “*turn around*”**: é o tempo que um processo leva desde a sua criação até ao seu término, levando em consideração todo o tempo gasto na espera para alocação de memória, espera na fila de pronto, processamento na CPU e na fila de espera, como nas operações de E/S e
- **tempo de resposta**: é o tempo decorrido entre uma requisição ao sistema ou à aplicação e o instante em que a resposta é exibida. Em sistemas interativos, podemos entender tempo de resposta como o tempo decorrido entre a última tecla digitada pelo usuário e o início da exibição do resultado no monitor.



## Escalonamento Preemptivo e Não-preemptivo

- O escalonamento preemptivo é caracterizado pela possibilidade do sistema operacional interromper um processo em execução e passá-lo para o estado de pronto, com o objetivo de alocar outro processo na CPU e
- no escalonamento não-preemptivo, quando um processo está em execução nenhum evento externo pode ocasionar a perda do uso do processador. O processo somente sai do estado de execução caso termine seu processamento ou execute instruções do próprio código que ocasionem uma mudança para o estado de espera.

## Algoritmos de Escalonamento

- FIFO (*First In, First Out*),
- o escalonamento FIFO é do tipo não-preemptivo e foi inicialmente implementado em sistemas monoprogramáveis com processamento *batch*, sendo ineficiente se aplicado na forma original em sistemas interativos de tempo compartilhado.





## Cálculo do Tempo Médio de Espera

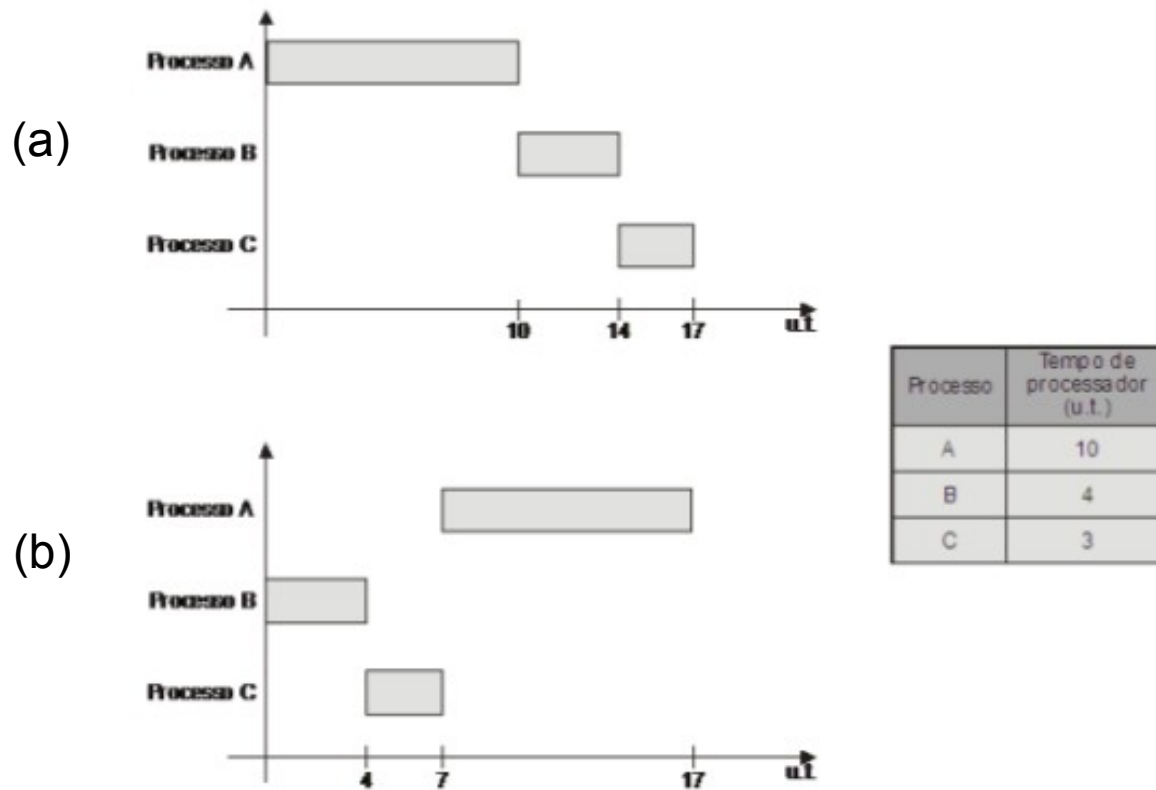


Figura 8.3: Exemplos de escalonamento FIFO



## Escalonamento Menor *Job* Primeiro

- No escalonamento *Shortest-Job-First* (SJF *scheduling*), também conhecido como *Shortest-Process-Next* (SPN *scheduling*), o algoritmo de escalonamento seleciona o processo:
  - que estiver no estado “pronto” e
  - que tiver o menor tempo de processador ainda por executar.

## Cálculo do Tempo Médio de Espera

- Calcule o tempo médio de espera, para os processos abaixo, utilizando o SJF.

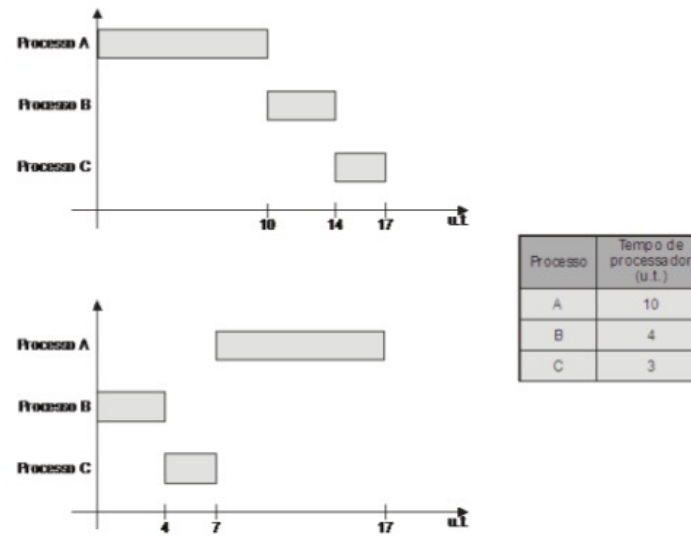


Figura 8.3: Exemplos de escalonamento FIFO



## Escalonamento pela Próxima Taxa de Resposta Mais Alta

- O escalonamento pela Próxima Taxa de Resposta Mais Alta (*Highest-Response-Ration-Next* - HRN) corrige algumas das fraquezas no SJF, particularmente o preconceito excessivo contra tarefas longas e o excessivo favoritismo para novas tarefas curtas;
- prioridades dinâmicas são calculadas de acordo com a fórmula:
  - $\text{prioridade} = (\text{tempo de espera} + \text{tempo de serviço}) / \text{tempo de serviço}$

## Cálculo do Tempo Médio de Espera

- Calcule o tempo médio de espera, para os processos abaixo, utilizando o HRN.

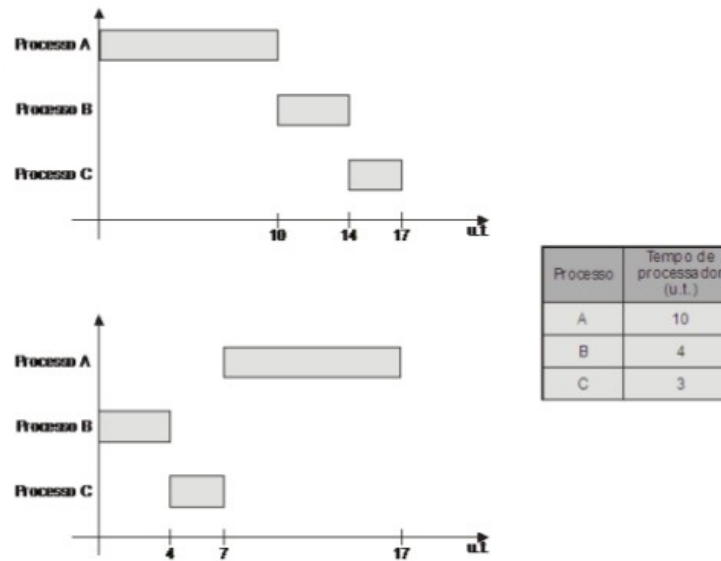


Figura 8.3: Exemplos de escalonamento FIFO



## Escalonamento Cooperativo

- A principal característica do escalonamento cooperativo está no fato da liberação do processador ser uma tarefa realizada exclusivamente pelo processo em execução, que de uma maneira cooperativa libera a CPU para um outro processo;
- neste mecanismo o processo em execução verifica, periodicamente, uma fila de mensagens para determinar se existem outros processos na fila de pronto e
- como a interrupção do processo em execução não é responsabilidade do sistema operacional, algumas situações indesejadas podem ocorrer.



## Escalonamento Circular

- Também conhecido como *Round Robin*,
- é um escalonamento do tipo **preemptivo**, projetado especialmente para sistemas de tempo compartilhado. É bastante semelhante com o FIFO, porém, quando um processo passa para o estado de “execução”, existe um tempo limite para o uso contínuo do processador, denominado “fatia de tempo” (*time slice*) ou “*quantum*” e
- neste escalonamento, toda vez que um processo é escalonado para execução, uma nova fatia de tempo é concedida. Caso a fatia de tempo expire, o sistema operacional interrompe o processo em execução, salva seu contexto e direciona-o para o final da fila de processos em estado “pronto”.
- Esse mecanismo é conhecido por preempção por tempo.



## Escalonamento Circular

- O valor da fatia de tempo depende da arquitetura de cada sistema operacional e, em geral, varia de 10 a 100 milissegundos. Este valor afeta diretamente o desempenho da política de escalonamento circular.
- Caso a fatia de tempo tenha um valor muito alto, este escalonamento tenderá a ter o mesmo comportamento do escalonamento FIFO.
- Caso o valor da fatia de tempo seja pequeno, a tendência é que haja um grande número de preempções, o que ocasionaria excessivas mudanças de contexto, prejudicando o desempenho do sistema e afetando o tempo de *turnaround* dos processos.

## Escalonamento Circular



Figura 8.5: Escalonamento Circular

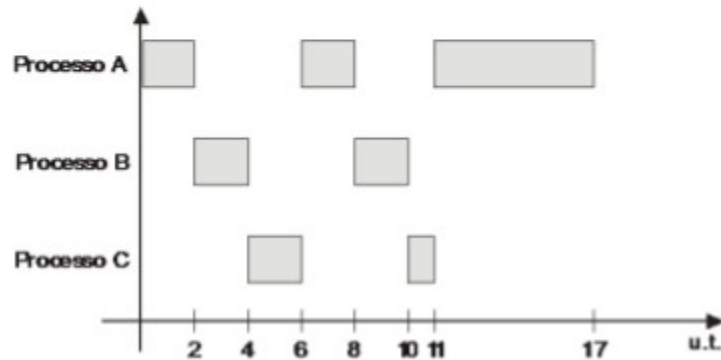


Figura 8.6: Exemplo de escalonamento circular



## Escalonamento com Prioridade

- É um escalonamento do tipo **preemptivo** realizado com base em um valor associado a cada processo denominado prioridade de execução. O processo com maior prioridade no estado de pronto é sempre o escolhido para execução, e processos com valores iguais são escalonados seguindo o critério do FIFO.
- Neste escalonamento, o conceito de fatia de tempo não existe; ou seja, um processo em execução não pode sofrer preempção por tempo.
- No escalonamento por prioridade, a perda do uso do processador só ocorrerá no caso de uma mudança voluntária para o estado de espera ou quando um processo de prioridade maior passa para o estado de pronto.

## Escalonamento com Prioridade

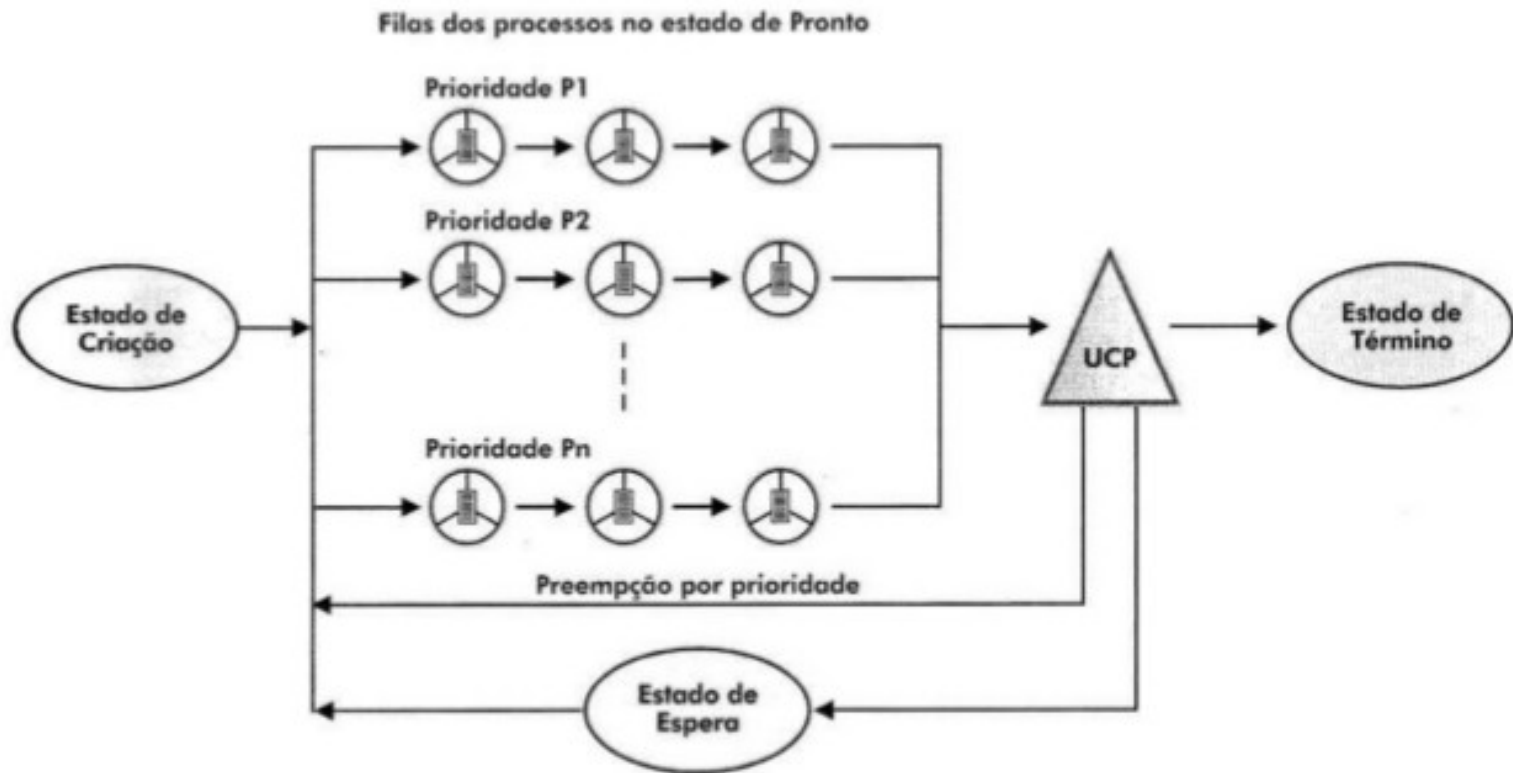


Figura 8.8: Escalonamento por prioridades



## Escalonamento com Prioridade

A figura 8.9 exemplifica o escalonamento por prioridades com três processos de prioridades diferentes, onde 5 é o mais alto valor de prioridade.

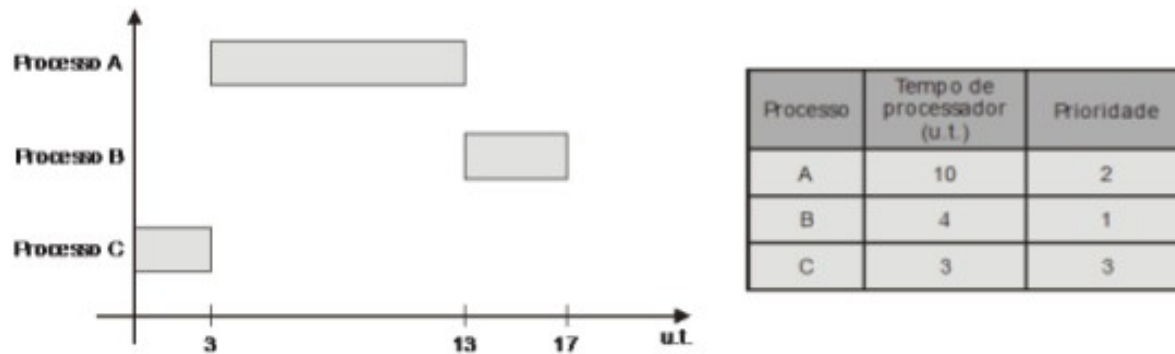


Figura 8.9: Exemplo de escalonamento por prioridades



## Escalonamento Circular Com Prioridade

- Este escalonamento implementa o conceito de fatia de tempo e de prioridade de execução associada a cada processo.
- Um processo permanece no estado de execução até que:
  - termine seu processamento, voluntariamente passe para o estado de espera ou
  - sofra uma preempção por tempo ou prioridade.

## Escalonamento Circular Com Prioridade

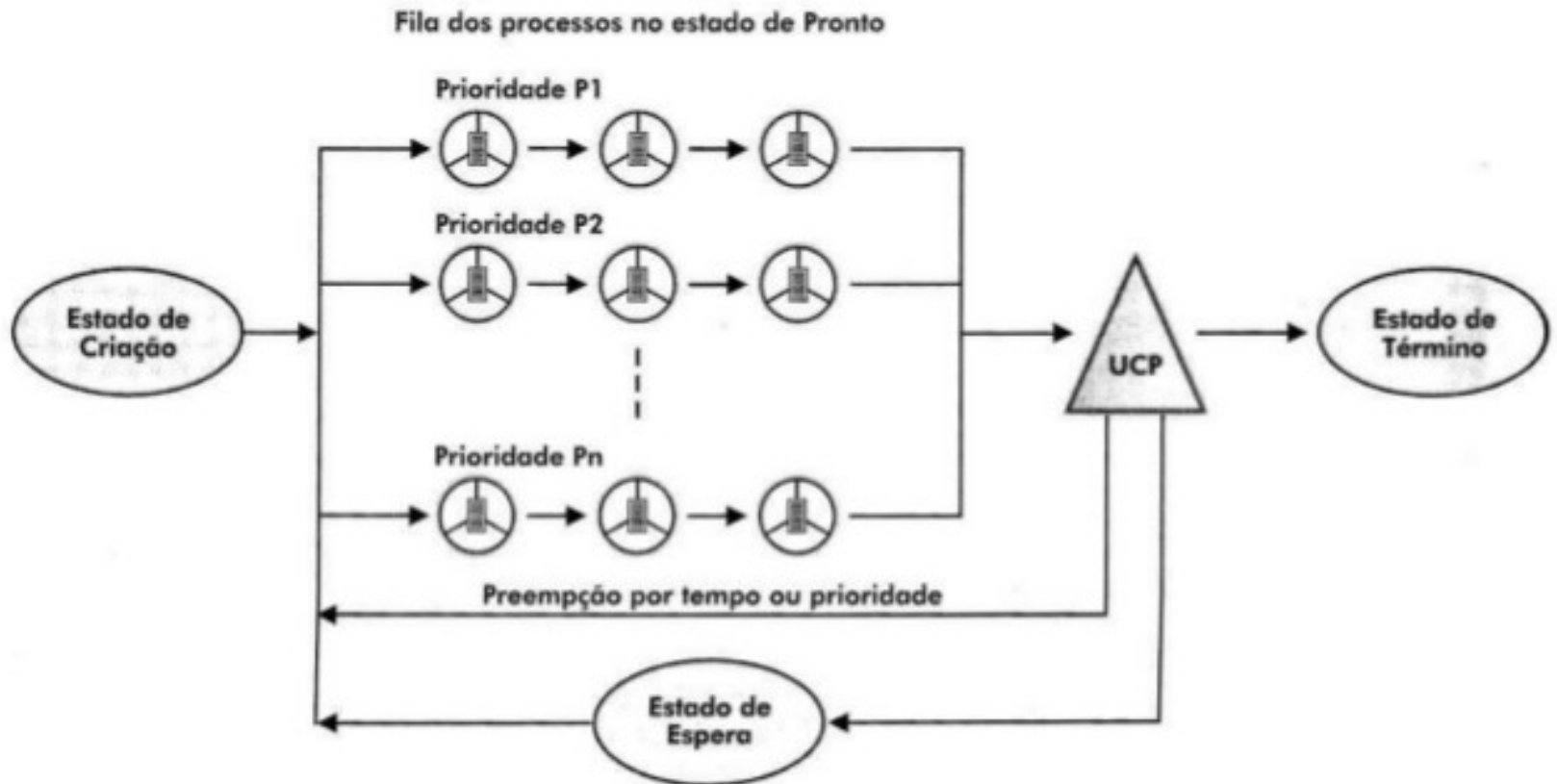


Figura 8.10: Escalonamento circular com prioridades



## Escalonamento em Sistemas de Tempo Real

- Diferentemente dos sistemas de tempo compartilhado, onde a aplicação não é prejudicada pela variação no tempo de resposta, algumas aplicações específicas exigem respostas imediatas para a execução de determinadas tarefas.
- O escalonamento em sistemas de tempo real deve levar em consideração a importância relativa de cada tarefa na aplicação. Em função disso, o escalonamento por prioridades é o mais adequado, já que para cada processo uma prioridade é associada em função da importância do processo dentro da aplicação.



## Referências

- ERDEI, Jacob. **Basic Structure for a Pentium Microprocessor**. Disponível em: <<https://www.pctechguide.com/cpu-architecture/basic-structure-of-a-pentium-microprocessor>>. Acesso em: 13 jul. 2017.
- MACHADO, F. B.; MAIA, L. P. **Arquitetura de Sistemas Operacionais**. 3. ed. Rio de Janeiro: LTC, 2010.
- PANATTA, B.; SANTI, D. de; MOERSCHBACHER, G.; LIMA, L. G. de. **Esquema Geral de Funcionamento do Processador**. Disponível em: <[http://sca.unioeste-foz.br/~habib/x/trabalhos/grupoa4/public\\_html/processador.html](http://sca.unioeste-foz.br/~habib/x/trabalhos/grupoa4/public_html/processador.html)>. Acesso em: 14 jul. 2017.
- UNIVERSIDADE FEDERAL FLUMINENSE. **Organização de Computadores II - processadores**. Disponível em: <<http://orgcomp2.ic.uff.br/processadores.php>>. Acesso em: 14 jul. 2017.