Gerência de Processador

Prof. Edwar Saliba Júnior
Junho de 2009

Introdução

- 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;
- 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.

Escalonamento de Processos

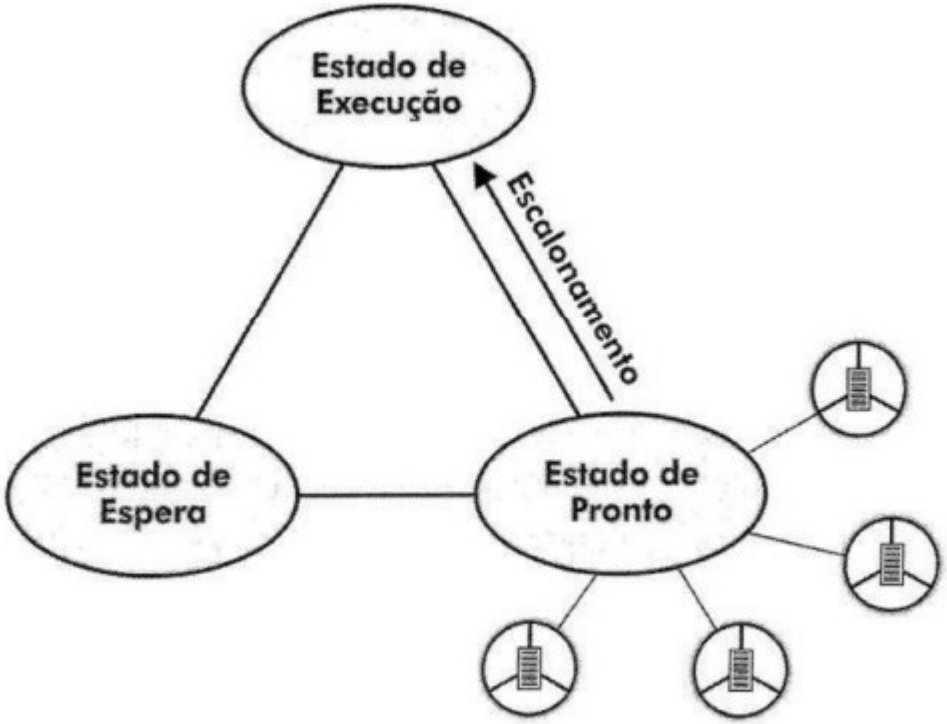


Figura 8.1: Escalonamento

Funções Básicas

- Manter o processador ocupado a maior parte do tempo;
- Balancear o uso da CPU entre processos;
- Privilegiar a execução de aplicações críticas;
- Maximizar o *throughput* do sistema;
- Oferecer tempos de respostas razoáveis para usuários interativos.

Considerações

- **A política de escalonamento varia de acordo com o propósito do sistema operacional (Sistemas de Tempo Real x Sistemas de Tempo Compartilhado x Sistemas *Batch*);**
- **O escalonador (*scheduler*) implementa a política de escalonamento;**
- ***Dispatcher*: rotina que realiza a troca do processo em execução após a decisão do escalonador;**
- **Latência do *dispatcher*: tempo gasto para a troca de processos;**
- **Em ambientes *multithread*, a unidade escalonada é a thread (somente as que estiverem no estado "pronto");**
- **Para simplificar o entendimento, chamamos sempre a unidade escalonada de processo.**

Preempção

- **Interrupção de um processo em execução para ser substituído por outro;**
- **O escalonamento de processos pode ser:**
  - **Não preemptivo: processamento *batch*;**
  - **Preemptivo: possibilidade de interrupção de um processo em execução.**

Escalonamento Preemptivo

- **Sistemas mais complexos;**
- **Priorização de tarefas;**
- **Sistemas de Tempo Real ou Tempo Compartilhado.**

Definições Importantes

- ***Throughput*:** 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;

- **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 função apenas do código da aplicação e da entrada de dados;

- **Tempo de espera:** é o tempo total que um processo permanece na fila de pronto durante seu processamento, aguardando para ser executado.

Definições Importantes

- **Tempo de *turnaround*:** 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;

- **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.

Sistemas Operacionais - Características

- As características de cada sistema operacional determinam quais são os principais aspectos para a implementação de uma política de escalonamento adequada;
- Exemplos:
  - sistemas de tempo compartilhado exigem que o escalonamento trate todos os processos de forma igual, evitando, assim, a ocorrência de *starvation*, ou seja, que um processo fique indefinidamente esperando pela utilização do processador;
  - Já em sistemas de tempo real, o escalonamento deve priorizar a execução de processos críticos em detrimento da execução de outros processos.

Política de Escalonamento - Principais Critérios

- **Ser justo:** um algoritmo de escalonamento é justo se todos os processos são tratados igualmente e nenhum processo pode sofrer postergação indefinida;
- **Maximizar o desempenho do sistema:** um algoritmo de escalonamento deve tentar servir o maior número possível de processos por unidade de tempo;
- **Maximizar o número de usuários interativos:** cada qual recebendo tempos de resposta aceitáveis (no máximo alguns segundos);
- **Ser previsível:** uma dada tarefa deveria rodar a mesma quantidade de tempo e sobre o mesmo custo independente da carga do sistema.

Política de Escalonamento - Principais Critérios

- **Minimizar o *overhead*:** nem sempre este objetivo é considerado o mais importante. *Overhead* é comumente visto como excesso, seja de: processamento, memória, largura de banda e etc.;

- **Balancear o uso de recursos:** os mecanismos de escalonamento deveriam manter os recursos do sistema ocupados. Processos que usariam recursos sutis (simples, pequenos) deveriam ser favorecidos;

- **Utilização do processador:** na maioria dos sistemas, é desejável que o processador permaneça ocupado a maior parte do tempo. Uma utilização na faixa de 30% indica um sistema com uma carga de processamento baixa, enquanto que na faixa de 90% indica um sistema muito carregado, próximo de sua capacidade máxima.

Política de Escalonamento - Principais Critérios

- **Alcançar o equilíbrio entre resposta e utilização** – o melhor modo de garantir bons tempos de resposta é ter suficientes recursos disponíveis sempre que eles forem necessários. O preço a ser pago por esta estratégia é que a utilização total dos recursos será pobre. Em sistemas de tempo real, respostas rápidas são essenciais e a utilização dos recursos é menos importante. Em outros tipos de sistemas, a economia faz muitas vezes a efetiva utilização de recursos imperativa;

- **Evitar a espera indefinida (*starvation*):** em muitos casos, a espera indefinida pode ser tão ruim quanto o *deadlock*. Evitar a espera indefinida é melhor realizada pelo *aging* (envelhecimento – controla a postergação aumentando a prioridade do processo postergado);

- **Garantir prioridades:** em ambientes nos quais os processos recebem prioridades, o mecanismo de escalonamento deveria favorecer os processos de maior prioridade.

Algoritmos de Escalonamento

- **FIFO:**
  - **não preemptivo;**
  - **Simples.**

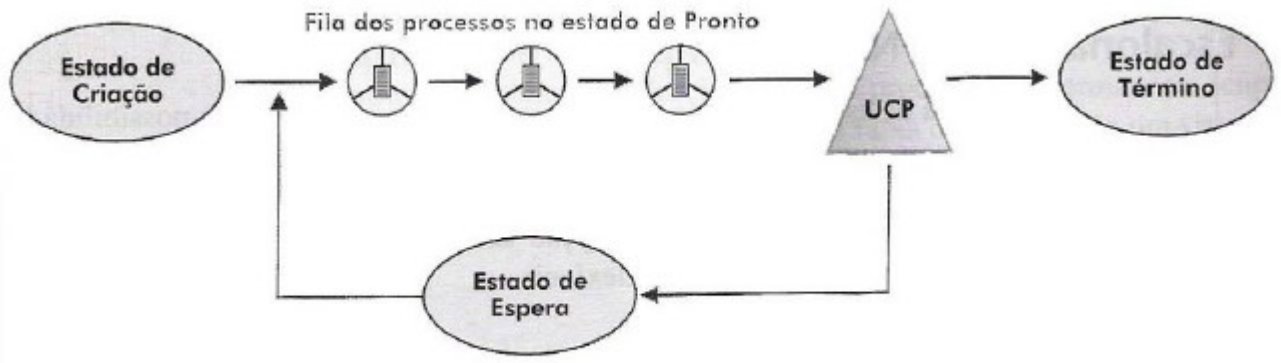
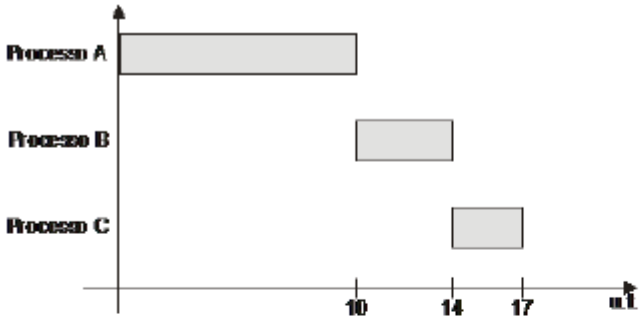
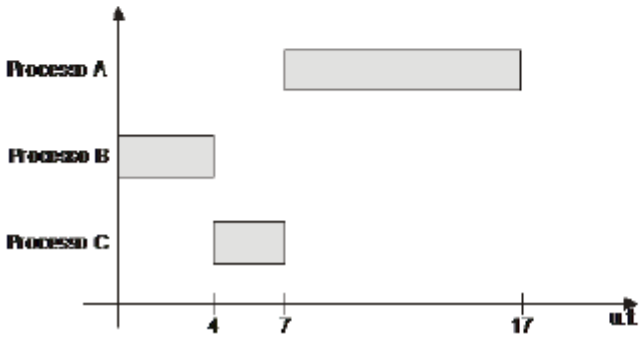


Figura 8.2: Escalonamento FIFO

FIFO – Tempo Médio de Espera





Processo	Tempo de processador (u.t.)
A	10
B	4
C	3

- Caso 1:

  (0+10+14)/ 3 = 8 u.t.

- Caso 2:

  (0+4+7)/3= 3.7 u.t.

Algoritmos de Escalonamento

- ***Shortest Job First*:**
  - **Seleciona o processo em estado "pronto" que necessitar de menos tempo de execução;**
  - **O tempo de execução atribuído ao processo é uma estimativa baseada em execuções passadas;**
  - **Reduz o tempo médio de espera dos processos;**
  - **Problema: estimar tempo gasto por processos interativos (dependem da ação do usuário);**
  - **Pode ser não preemptivo ou preemptivo (se entrar na fila um processo com tempo de execução menor do que o que está sendo executado).**

Shortest Job First – Tempo de Médio de Espera

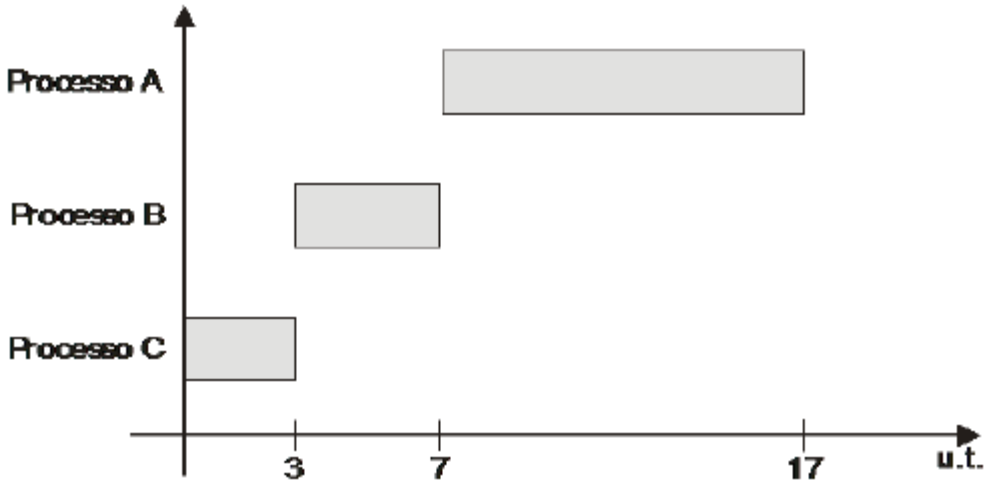


Figura 8.4: Exemplo de escalonamento SJF

- Caso único:

  (0+3+7)/3
  =3.3 u.t.

Algoritmos de Escalonamento

- **Escalonamento colaborativo ou cooperativo:**
  - **O próprio processo em execução determina se poderá ou não haver preempção, através da consulta a uma fila de mensagens enviadas por outros processos;**
  - **Não pode ser utilizado em Sistemas de Tempo Real;**
  - **Problema: um processo pode dominar o uso do processador caso não esteja apto a verificar a fila de mensagens e provocar a preenpção;**
  - **Primeiros sistemas operacionais da família Windows (conhecido como multitarefa cooperativo).**

Algoritmos de Escalonamento

- **Escalonamento circular ou *Round* Robin:**
  - Preemptivo;
  - Sistemas de Tempo Compartilhado com o uso de *time slice* (ou *quantum*);
  - Utiliza FIFO para escalonamento da fila;
  - *Time Slice*:
    - Grande: mesmos problemas do FIFO (tempo médio de espera alto);
    - Pequeno: alto custo de preempção, perde-se muito tempo com troca de contexto (10 a 100 milissegundos);
  - Utilizado em sistemas que executam processos interativos.

Escalonamento Circular

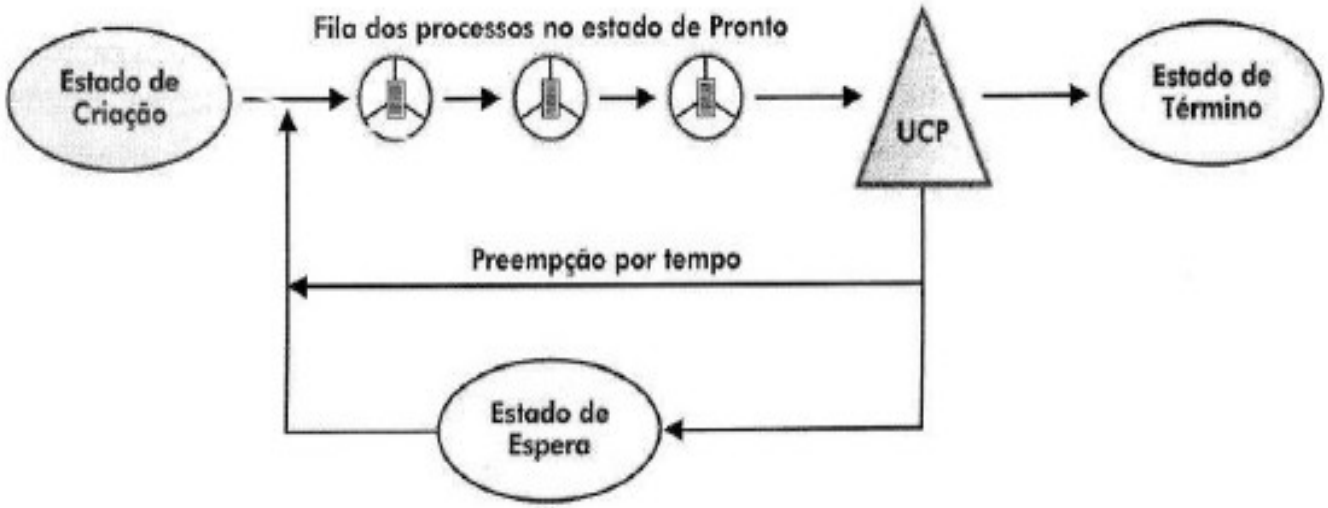


Figura 8.5: Escalonamento Circular

Esc. Circular – Exemplo

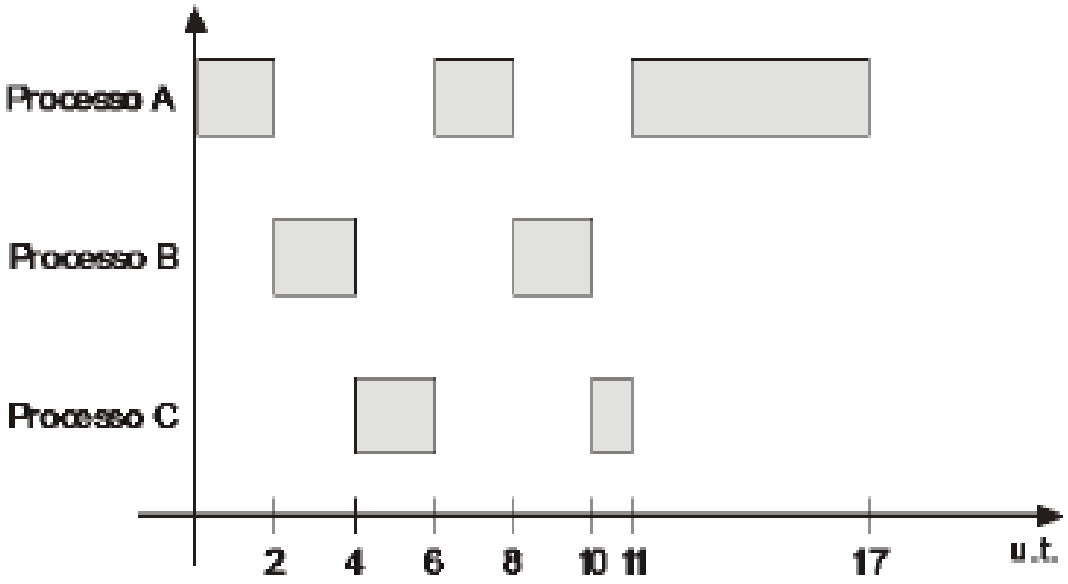
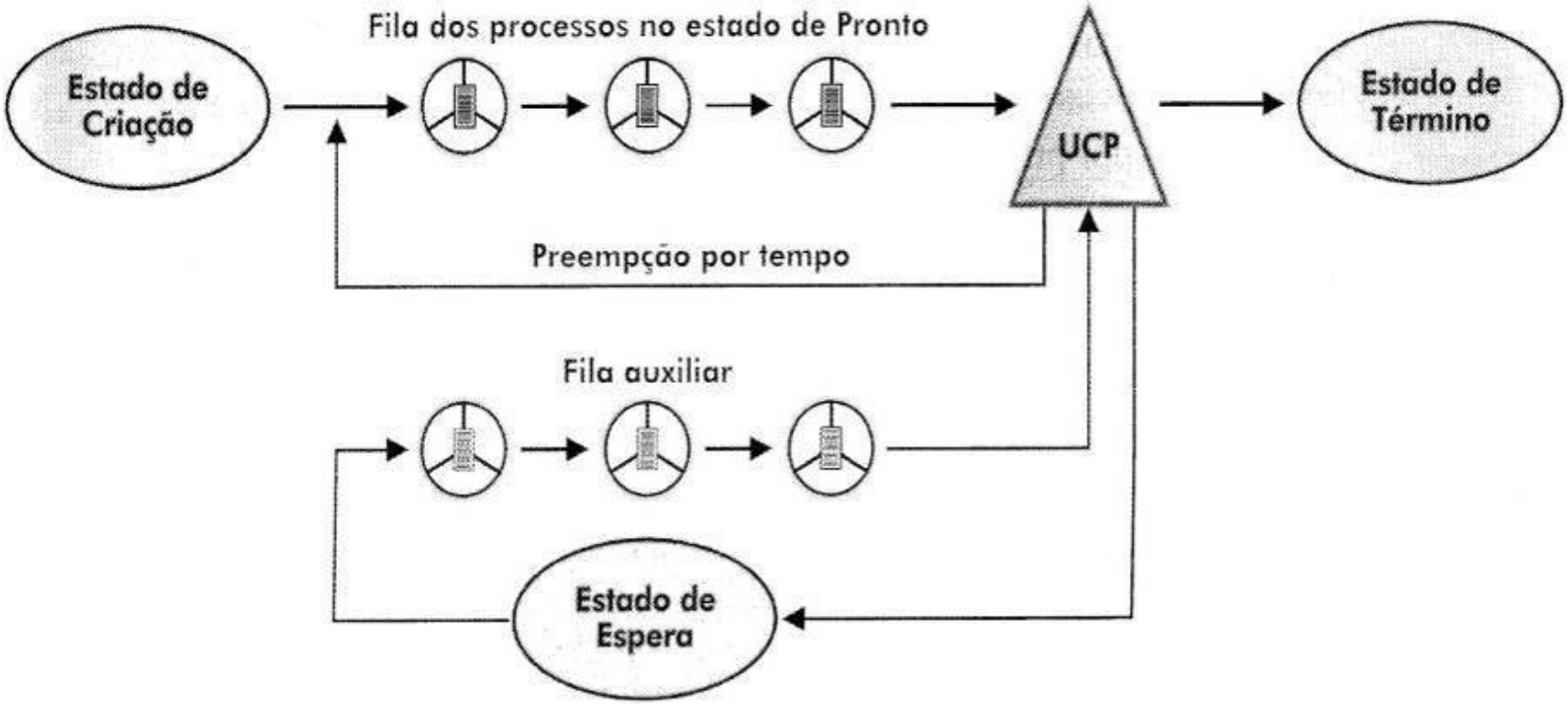


Figura 8.6: Exemplo de escalonamento circular

Algoritmos de Escalonamento

- **Escalonamento circular ou *Round* Robin:**
  - Problema: processos CPU-*Bound* são beneficiados no uso do processador em relação aos I/O-*Bound*. Devido as suas características os processos CPU-*Bound* tendem a usar integralmente as suas fatias de tempo de processamento. Enquanto os I/O-*Bound* têm mais chance de passar par a o estado de espera antes de sofrer preenpção por tempo;
  - Minimização do problema: conhecido como escalonamento circular virtual. Nesse esquema, processos que saem do estado de espera vão para uma fila de pronto auxiliar. Os processos da fila auxiliar possuem preferência no escalonamento em relação a fila de pronto, e o escalonador seleciona processos na fila de pronto quando a fila auxiliar estiver vazia.

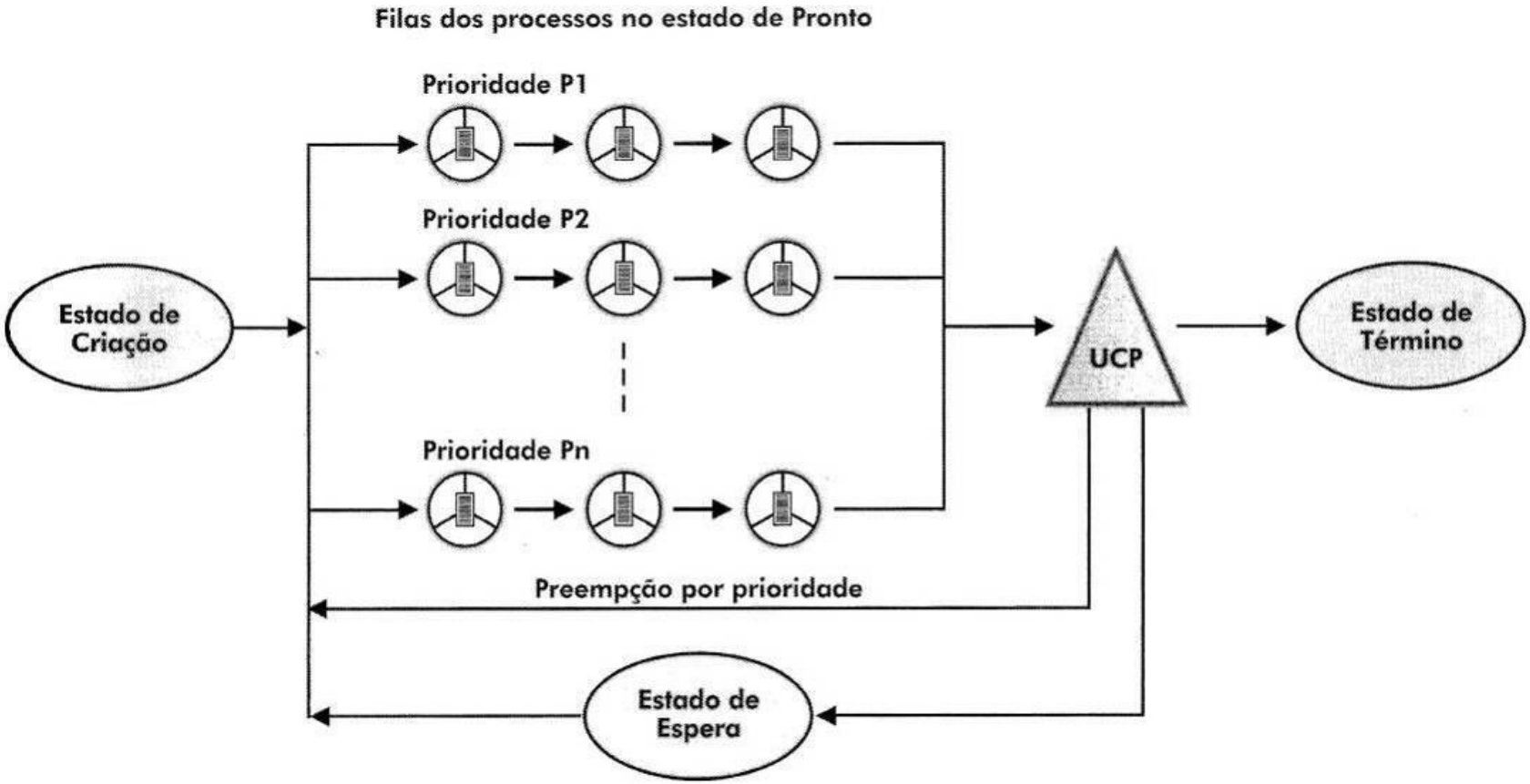
Esc. Circular - Refinamento



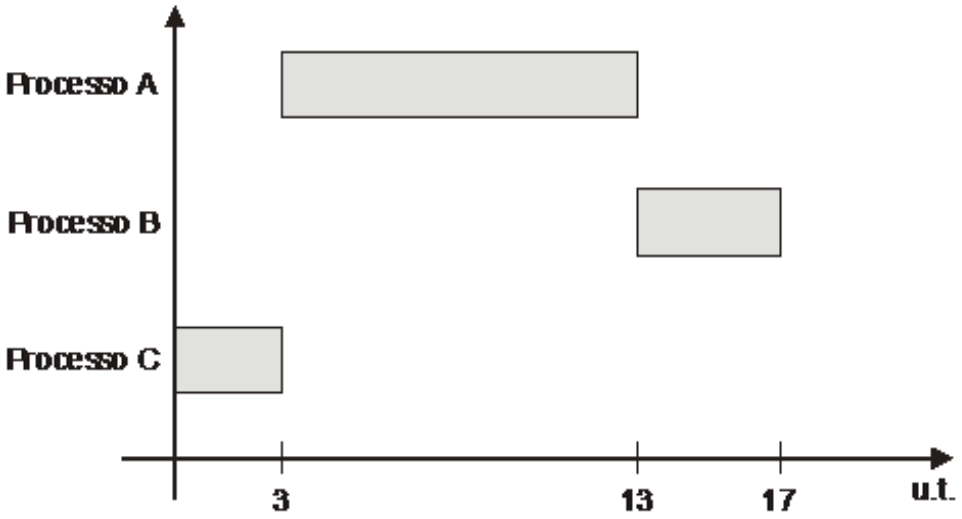
Algoritmos de Escalonamento

- **Escalonamento com prioridade:**
  - Prioridade de execução: valor associado ao processo (contexto de *software*);
  - Escalonador sempre escolhe o de maior prioridade (FIFO para desempate);
  - De tempos em tempos a rotina de escalonamento reavalia as prioridades dos processos em “pronto”. Se for detectado algum com prioridade de execução maior do que o que está sendo executado, pode ocorrer a preempção, ou simplesmente o rearranjo da fila de entrada;
  - Para que processos de menor prioridade não deixem de ser executados, alguns sistemas podem incrementar a prioridade dos processos que estão a mais tempo na fila.

Escalonamento com Prioridade



Esc. com Prioridade - Exemplo

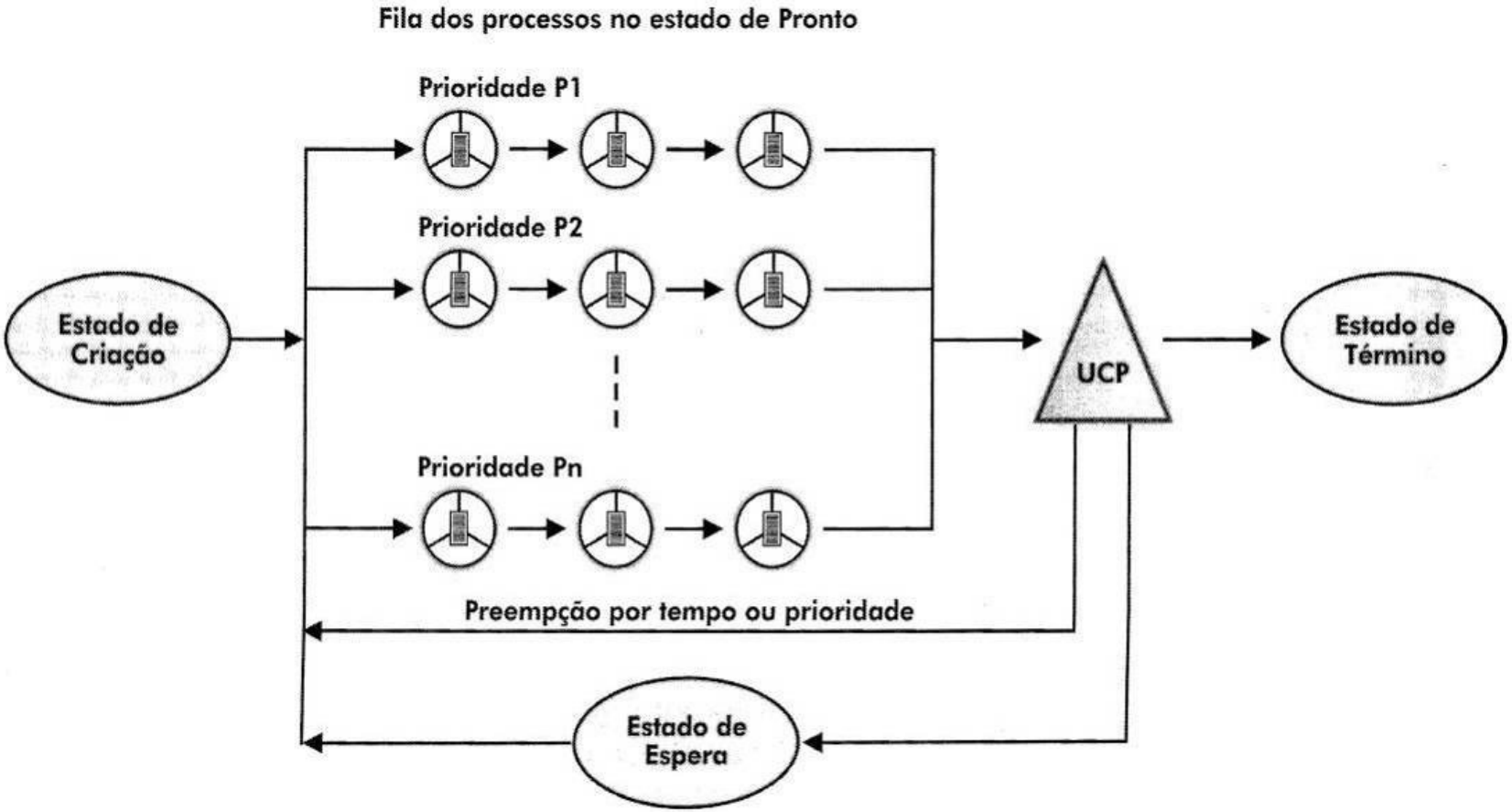


Processo	Tempo de processador (u.t.)	Prioridade
A	10	2
B	4	1
C	3	3

Algoritmos de Escalonamento

- **Escalonamento circular com prioridade:**
  - Fatia de tempo + prioridade de execução;
  - Utilizado na maioria dos sistemas de tempo compartilhado;
  - O Windows 2000 utiliza 32 níveis de prioridade.

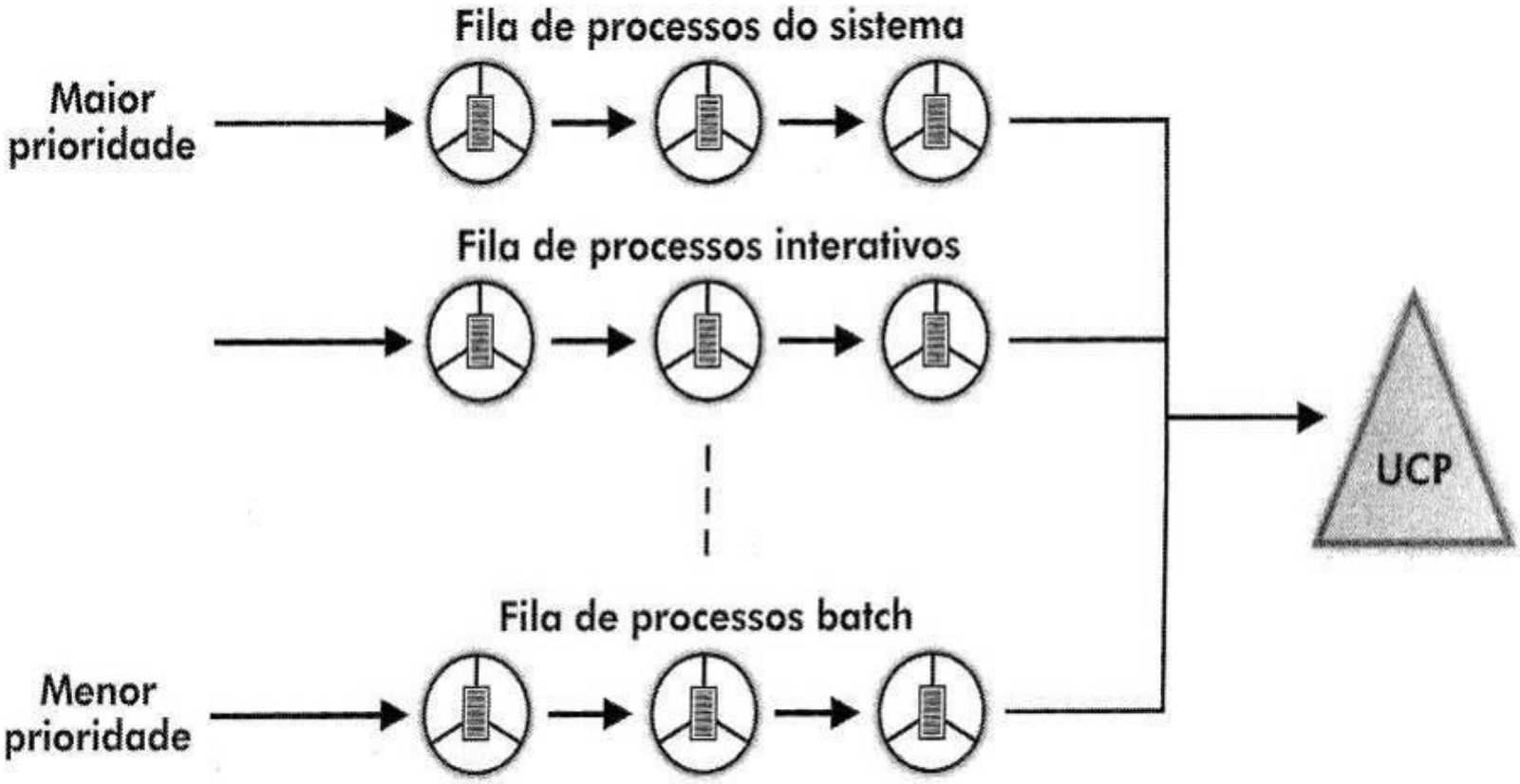
Esc. Circular com Prioridade



Algoritmos de Escalonamento

- **Escalonamento por Múltiplas Filas:**
  - Existem diversas filas de processos no estado de pronto, cada qual com uma prioridade específica;
  - Processos são associados as filas em função de características próprias: importância, tipo de processamento ou área de memória necessária;
  - Como os processos possuem características de processamento distintas, é difícil que um único mecanismo de escalonamento seja adequado a todos. A principal vantagem de múltiplas filas é a possibilidade da convivência de mecanismos de escalonamento distintos em um mesmo sistema operacional.

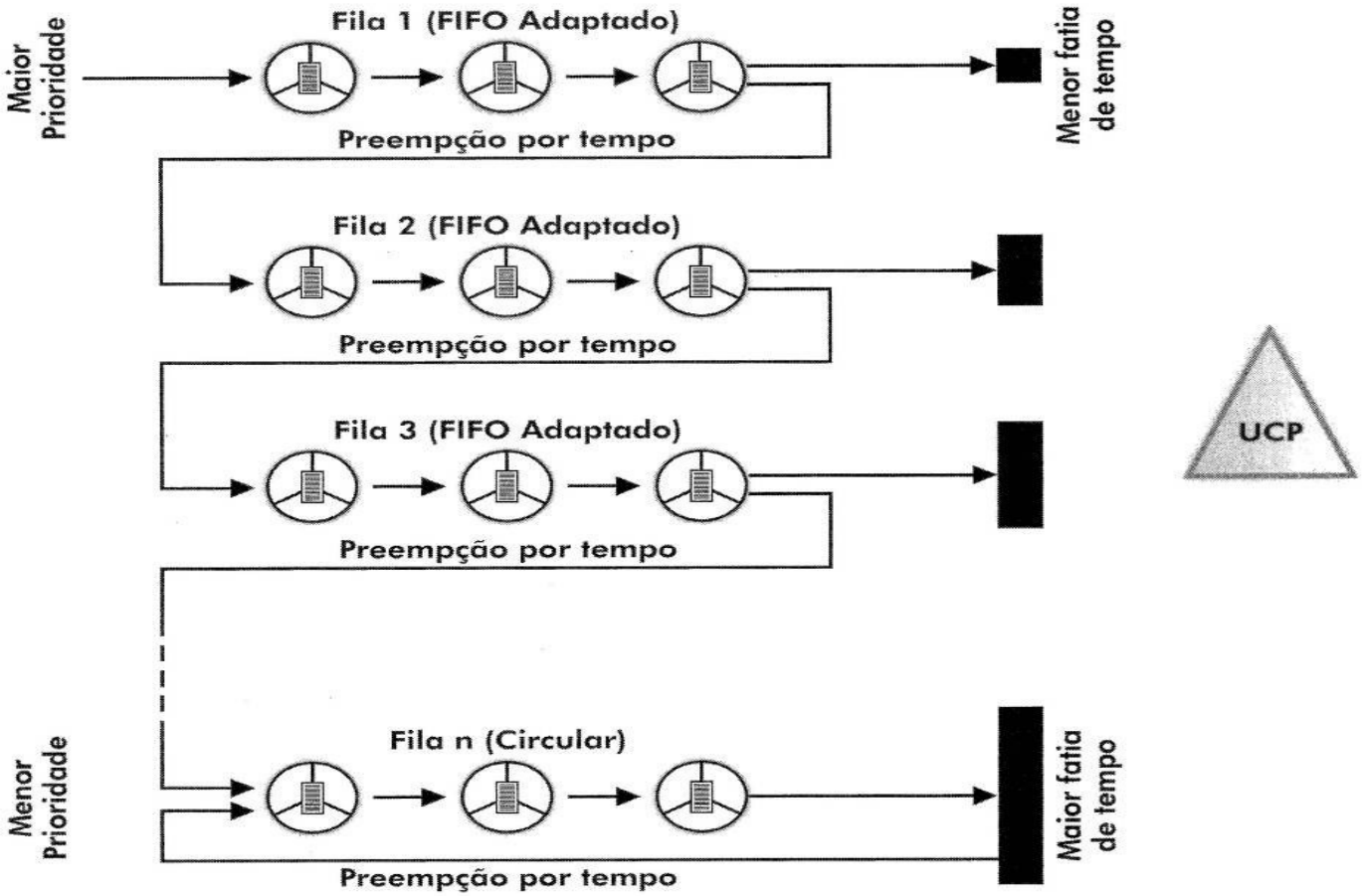
Esc. por Múltiplas Filas



Algoritmos de Escalonamento

- **Escalonamento por Múltiplas Filas com Realimentação:**
  - Semelhante ao escalonamento por múltiplas filas, porém os processos podem trocar de fila durante seu processamento;
  - Vantagem: permite ao sistema operacional identificar dinamicamente o comportamento de cada processo, direcionando-o para fila com prioridade de execução e mecanismo de escalonamento mais adequado ao longo de seu processamento.

Esc. Por Múltiplas Filas com Realimentação



Escalonamento em Dois Níveis

- Até agora temos assumido que todos os processos estão na memória principal. Se não houver memória disponível para todos, alguns de tais processos devem ser mantidos em disco. Esta situação tem grande impacto sobre o escalonador, uma vez que o tempo gasto na troca de contexto envolvendo o disco, é algumas ordens de magnitude maior do que quando ambos os processos estão na memória principal;

- Uma forma mais prática de tratar com o *swap* de processos é usando um escalonador de dois níveis; onde um subconjunto dos processos prontos é carregado inicialmente na memória principal. O escalonador então se limita a escolher processos deste subconjunto;

- Periodicamente, um escalonador de mais alto nível entra em cena, para remover processos que já tenham ficado tempo suficiente na memória principal, carregando nesta, processos que estão há muito no disco.

Escalonamento em Dois Níveis

- Uma vez completada a troca, o escalonador de baixo nível entra novamente em cena, limitando suas escolhas aos processos armazenados na memória principal;

- O escalonador de baixo nível coloca para rodar um dos processos que estiverem na memória no momento da ativação do escalonador, enquanto que cabe ao de alto nível movimentar periodicamente processos entre a memória principal e o disco.

Escalonamento em Dois Níveis

- O escalonador de alto nível pode usar os seguintes critérios na sua tomada de decisão:
  - quanto tempo se passou desde que o processo foi colocado no disco ou na memória principal;
  - quanto tempo de processador o processo usou recentemente;
  - qual o tamanho do processo;
  - qual a prioridade do processo.

Bibliografia

- MACHADO, F. B.; MAIA, L. P. **Arquitetura de Sistemas Operacionais,** 3ª Ed., Rio de Janeiro: LTC Editora, 2002.
- SILVA, Guilherme Baião S. *Slides* da disciplina de Sistemas Operacionais de Arquitetura Fechada. Faculdade INED, 2005.