Redes de Computadores

Fundamentos de Sistemas Operacionais - 2º Período

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL DE MINAS GERAIS

SUMÁRIO

Capítulo 8 - Gerência do Processador

8.1 Introdução

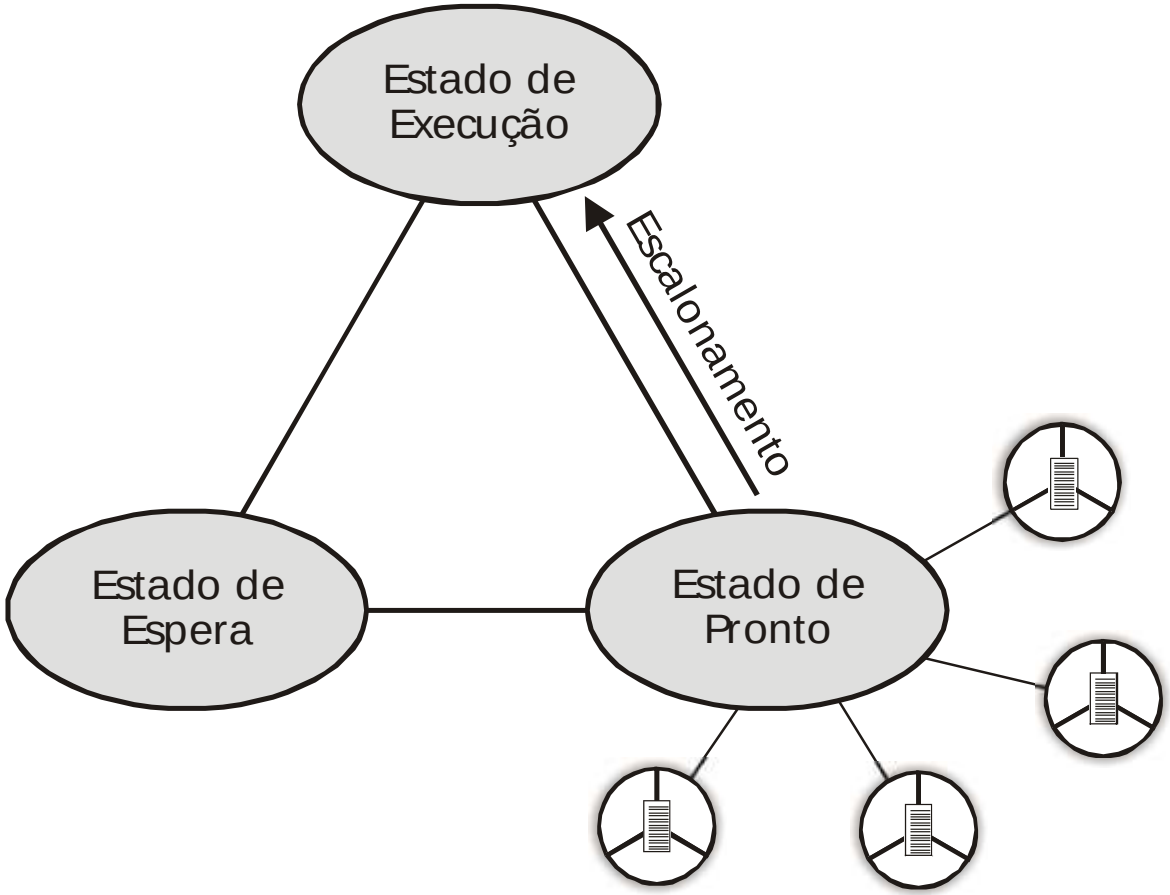
Com o surgimento dos sistemas multiprogramáveis, a gerência do processador tornou-se uma das atividades mais importantes em um SO.

A partir do momento em que diversos processos podem estar no estado de pronto, critérios devem ser estabelecidos para determinar qual processo será escolhido para fazer uso do processador.

Os critérios utilizados para essa seleção compõem a chamada política de escalonamento, que é a base da gerência do processador e da multiprogramação em um SO.

Capítulo 8 - Gerência do Processador

8.1 Introdução



8.2 Funções Básicas

A política de escalonamento de um SO tem diversas funções básicas, como:

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

A rotina do SO que tem como principal função implementar os critérios da política de escalonamento é denominada **escalonador** (***scheduler***).

A rotina conhecida como ***dispatcher*** é responsável pela troca de contexto dos processos após o escalonador determinar qual processo deve fazer uso do processador e o tempo gasto na troca é denominado **latência do *dispatcher***.

8.3 Critérios de Escalonamento

- **Utilização do Processador:** é desejável que o processador permaneça a maior parte do tempo ocupado.
- ***Throughput*:** representa o número de processos executados em um determinado instante de tempo.
- **Tempo de Processador / Tempo de UCP:** é o tempo que um processo leva no estado de execução durante o seu processamento.
- **Tempo de Espera:** é o tempo total que um processo permanece na fila de pronto durante seu processamento, aguardando para ser executado.
- **Tempo de *Turnaround*:** é o tempo que um processo leva desde a sua criação até seu término.

8.3 Critérios de Escalonamento

- **Tempo de Resposta:** é o tempo decorrido entre uma requisição ao sistema ou à aplicação e o instante em que a resposta é exibida.

De maneira geral, qualquer política de escalonamento busca otimizar a utilização do processador e o *throughput*, enquanto tenta diminuir os tempos de turnaround, espera e resposta.

Apesar disso, as funções que uma política de escalonamento deve possuir são muitas vezes conflitantes.

Dependendo do tipo do sistema operacional, um critério pode ter maior importância do que outros.

8.4 Escalonamentos Não-Preemptivos e Preemptivos

As políticas de escalonamento podem ser classificadas segundo a possibilidade de o sistema operacional interromper o processo em execução e substituí-lo por um outro (**preempção**).

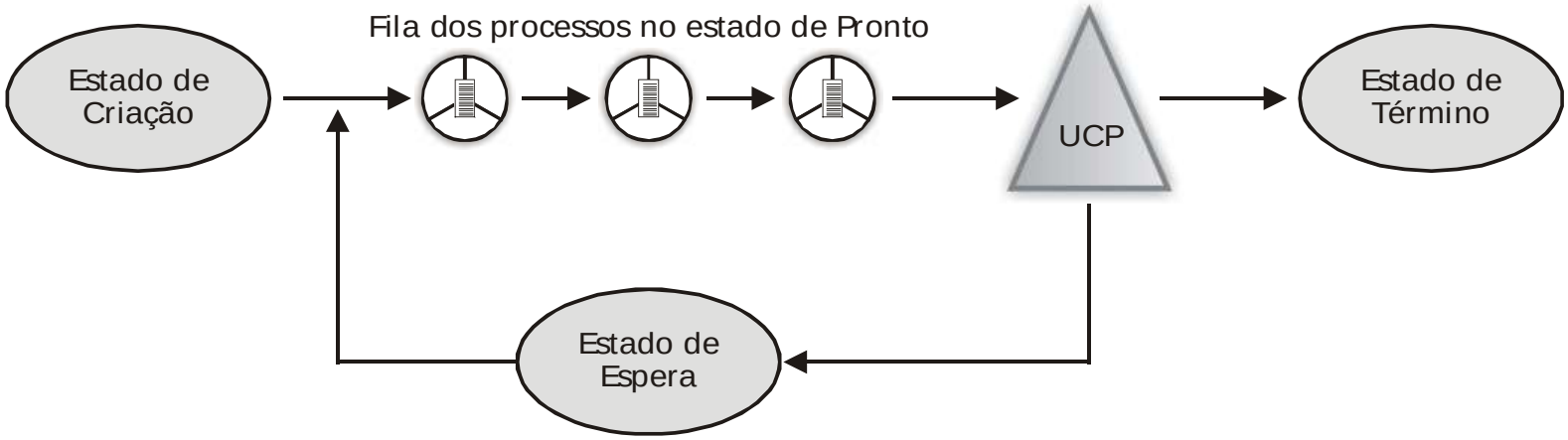
Escalonamento não-preemptivo: quando um processo está em execução nenhum evento externo pode ocasionar a perda do uso do processador.

Escalonamento preemptivo: o sistema operacional pode interromper um processo em execução e passá-lo para o estado de pronto, com o objetivo de alocar outro processo na UCP com prioridade maior do que o anterior.

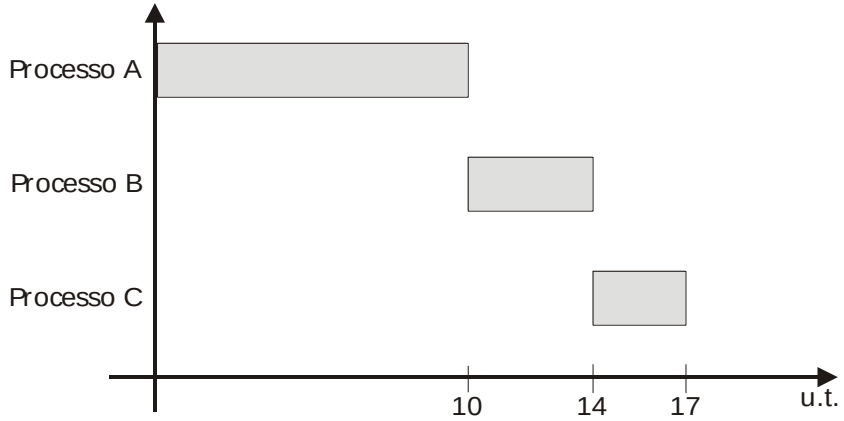
Atualmente, a maioria dos SOs implementa políticas de escalonamento preemptivas.

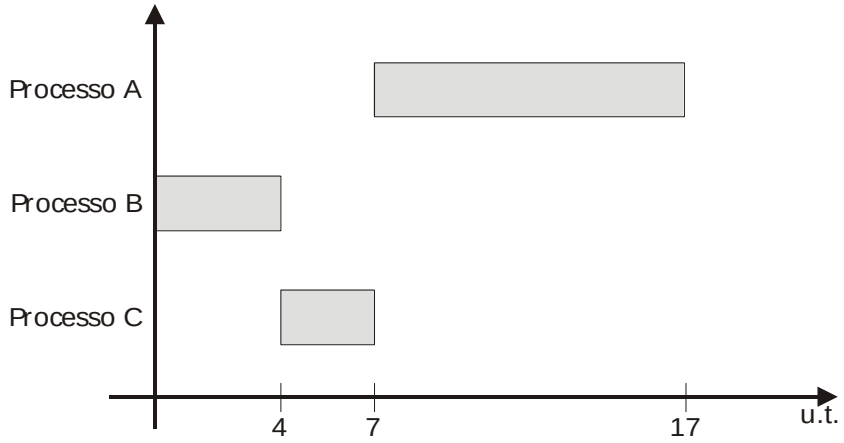
8.5 Escalonamento *First-In-First-Out* (FIFO *scheduling*)

Também conhecido como *first-come-first-served* (FCFS *scheduling*).



8.5 Escalonamento *First-In-First-Out* (FIFO *scheduling*)





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

8.5 Escalonamento *First-In-First-Out* (FIFO *scheduling*)

Apesar de simples, o escalonamento FIFO apresenta algumas deficiências:

- Impossibilidade de prever-se quando um processo terá a sua execução iniciada, já que isso varia em função do tempo de execução dos demais processos na fila de pronto;

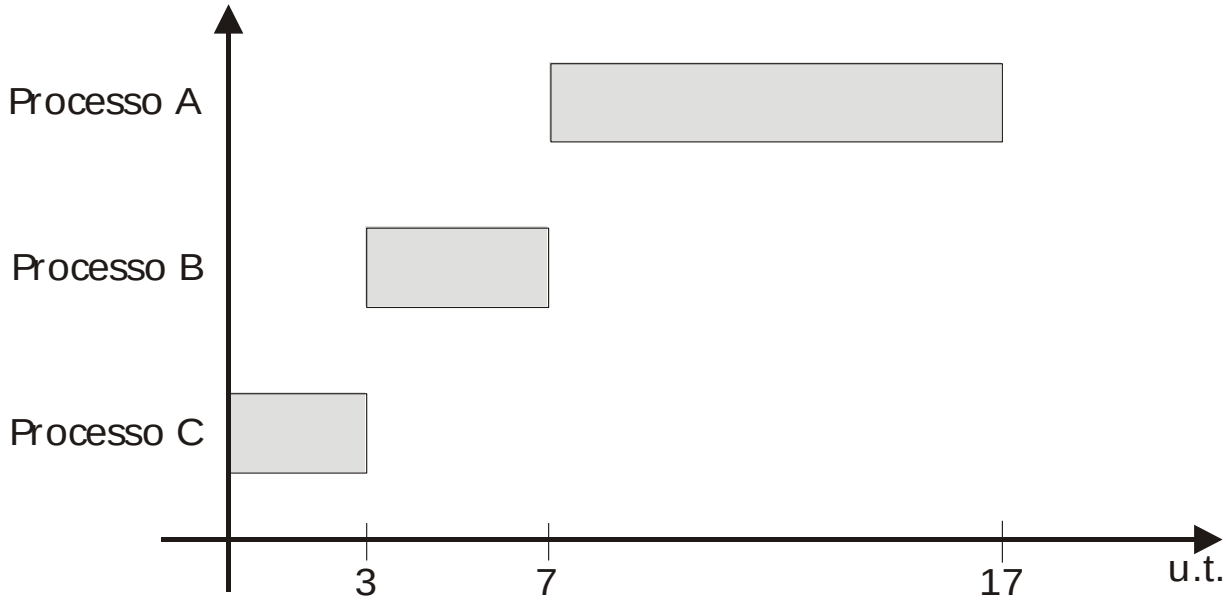
- Processos CPU-bound levam vantagem no uso do processador sobre processos I/O-bound.

O escalonamento FIFO é do tipo não-preemptivo e foi inicialmente implementado em sistemas monoprogramáveis com processamento *batch*.

Atualmente, sistemas de tempo compartilhado utilizam FIFO com variações.

8.6 Escalonamento *Shortest-Job-First* (SJF *scheduling*)

Também conhecido como *shortest-process-next* (SPN *scheduling*).



8.6 Escalonamento *Shortest-Job-First* (SJF *scheduling*)

Esta implementação foi utilizada nos primeiros sistemas operacionais com processamento exclusivamente batch.

Uma maneira de implementar o escalonamento SJF em sistemas interativos foi considerar o comportamento do processo neste ambiente.

Um problema existente nesta implementação é não ser possível ao SO saber quanto tempo um processo irá permanecer utilizando a UCP na próxima vez em que for escalonado.

Na sua concepção inicial, o escalonamento SJF é não-preemptivo.

8.6 Escalonamento *Shortest-Job-First* (SJF *scheduling*)

Vantagem sobre o FIFO: redução do tempo médio de *turnaround* dos processos, porém no SJF é possível haver *starvation* para processos com tempo de processador muito longo ou do tipo CPU-bound.

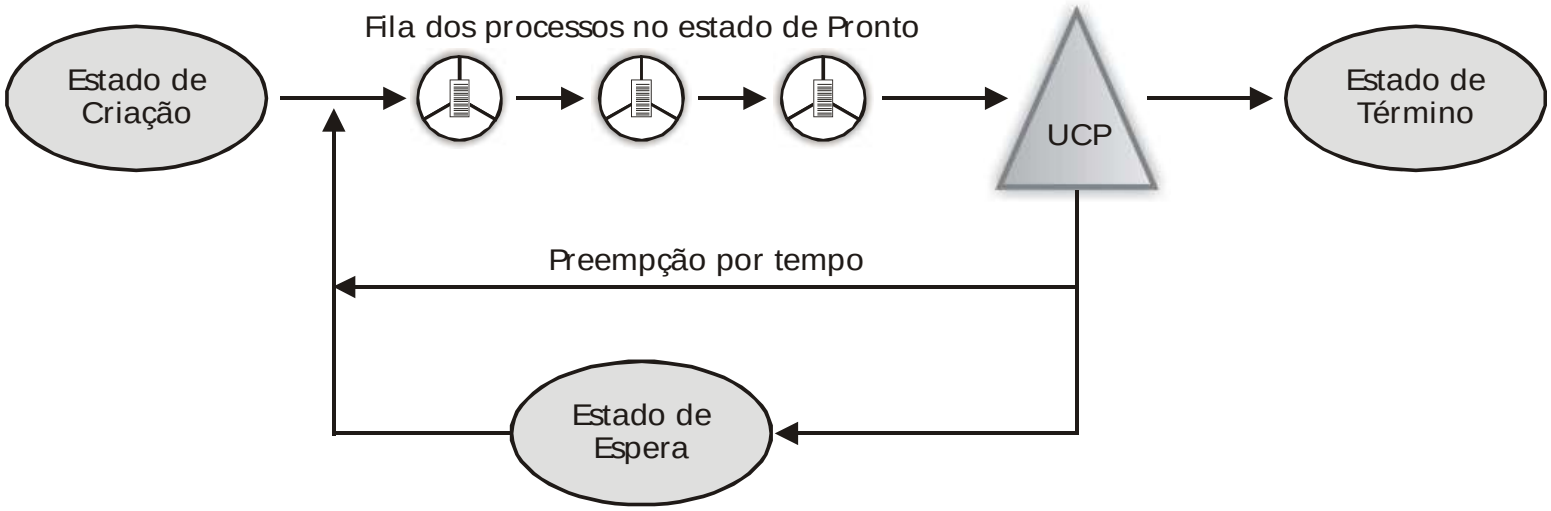
Uma implementação do escalonamento SJF com preempção é conhecida como escalonamento *shortest-remaining-time* (SRT *scheduling*).

8.7 Escalonamento Cooperativo

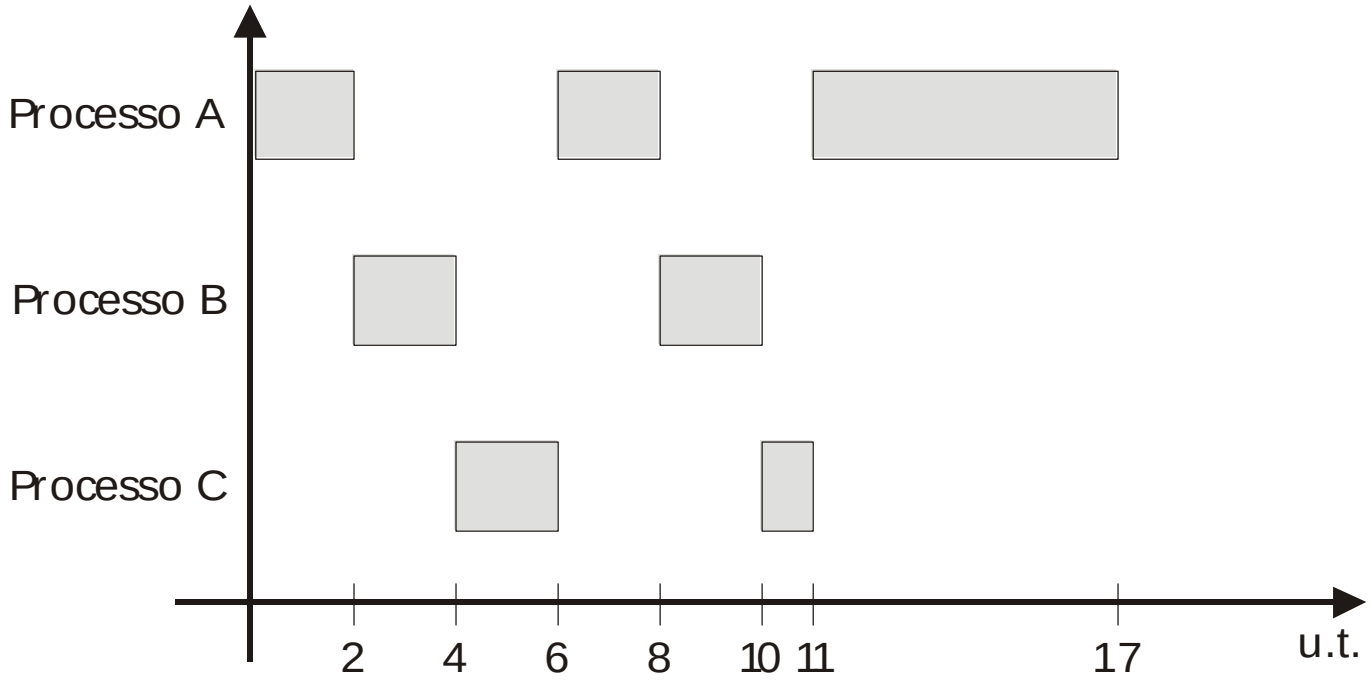
Um processo em execução pode voluntariamente liberar o processador, retornando à fila de pronto e possibilitando que um novo processo seja escalonado, permitindo assim uma melhor distribuição no uso do processador.

8.8 Escalonamento Circular

O escalonamento circular (*round robin scheduling*) é um escalonamento do tipo preemptivo, projetado especialmente para sistemas de tempo compartilhado.



8.8 Escalonamento Circular



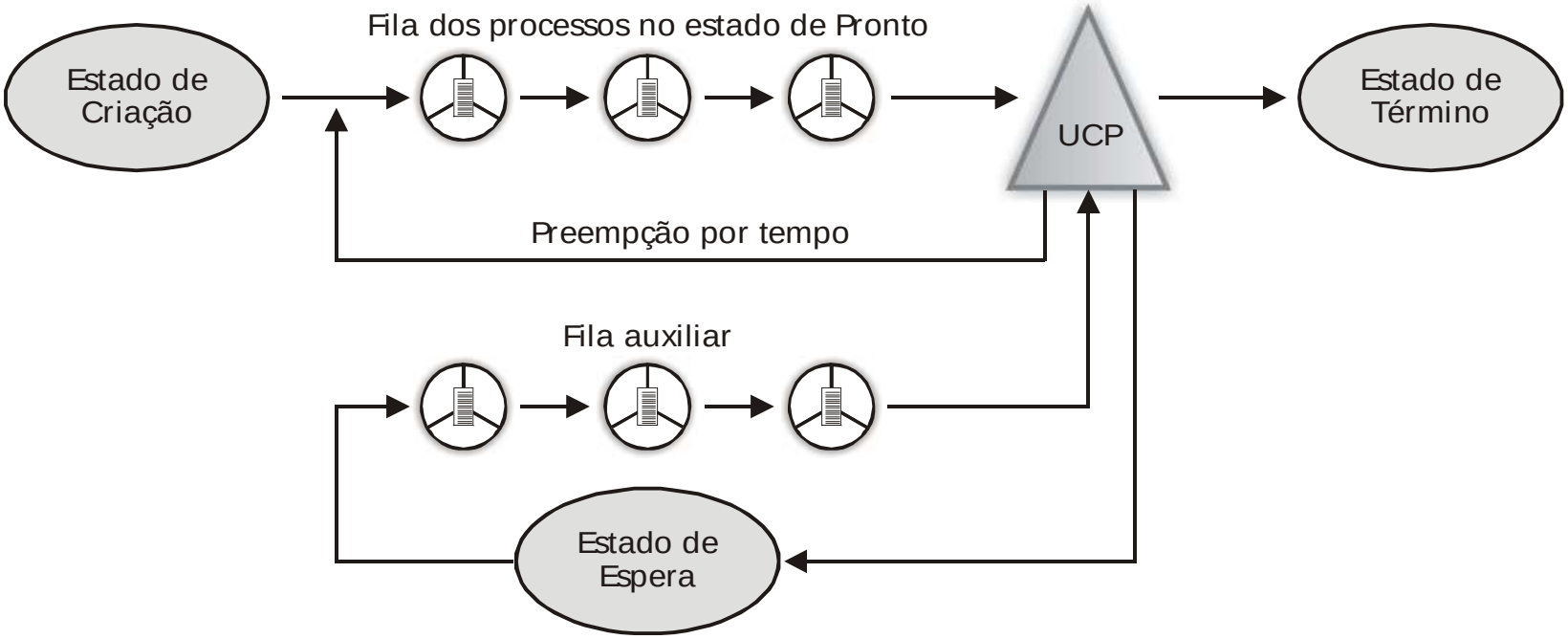
8.8 Escalonamento Circular

A principal **vantagem** do escalonamento circular é não permitir que um processo monopolize a UCP, sendo o tempo máximo alocado continuamente igual à fatia de tempo definida no sistema.

Um **problema** presente nesta política é que processos CPU-bound são beneficiados no uso do processador em relação aos processos I/O-bound, o que provoca um balanceamento desigual no uso do processador.

Um refinamento do escalonamento circular, que busca reduzir esse problema, é conhecido como **escalonamento circular virtual**.

8.8 Escalonamento Circular



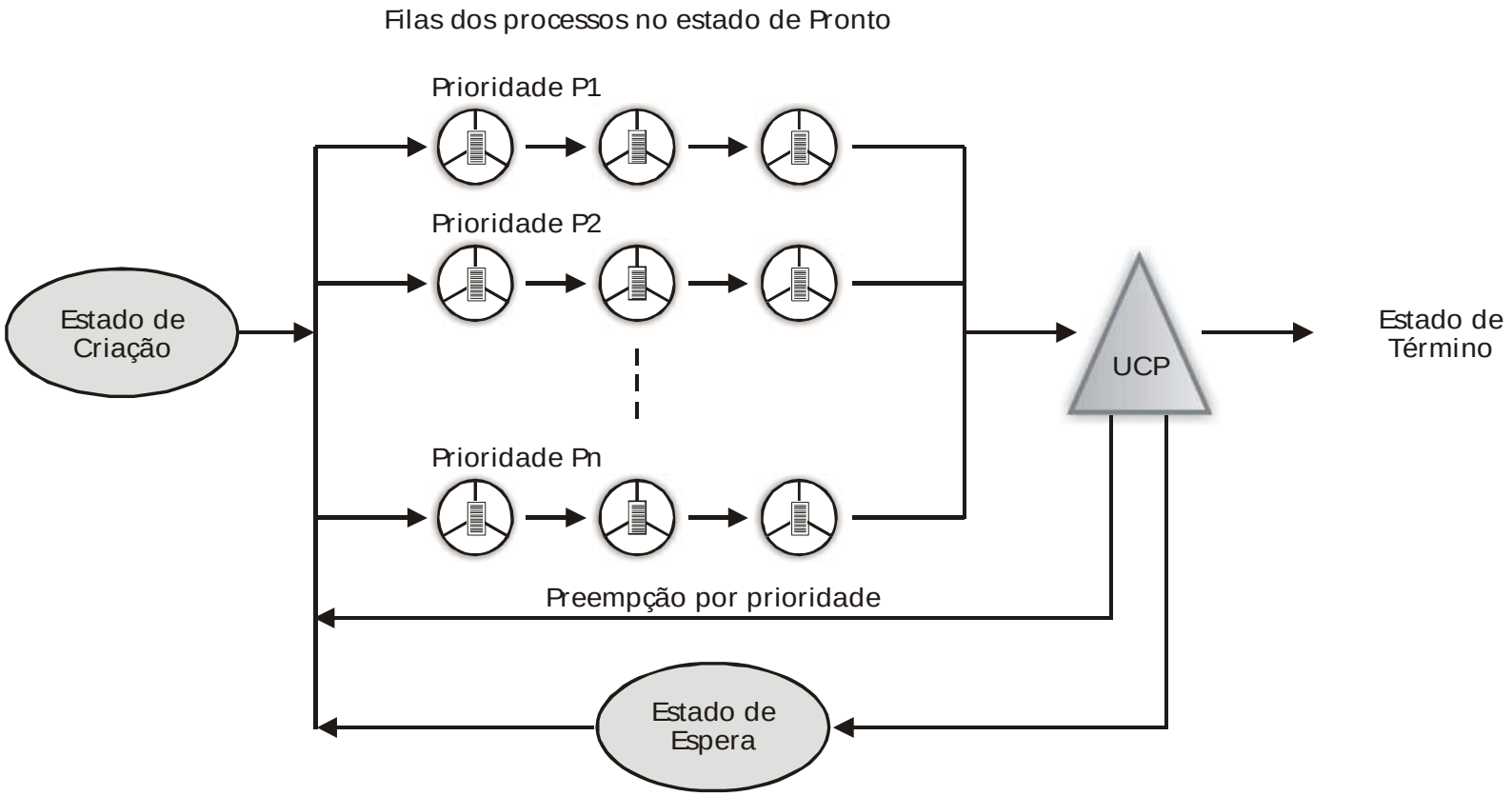
8.9 Escalonamento por Prioridades

É um escalonamento do tipo **preemptivo** realizado com base em um valor associado a cada processo denominado **prioridade de execução**.

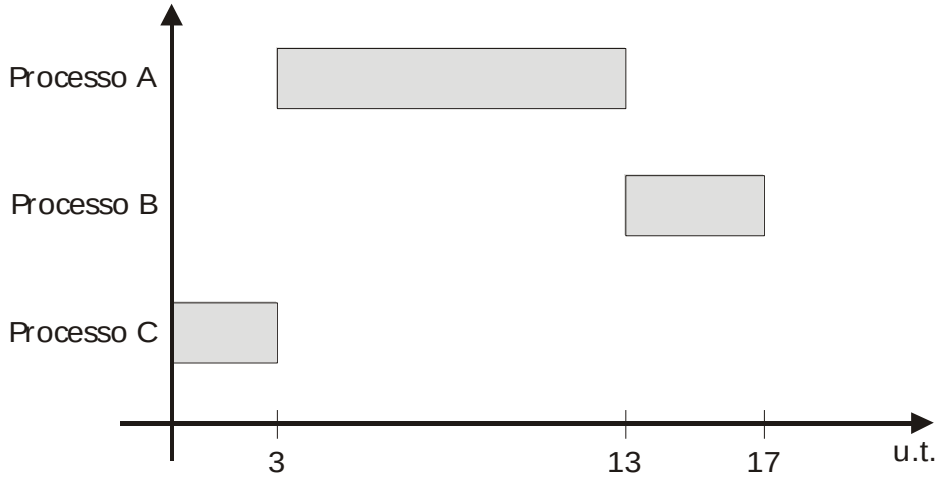
A perda do uso do processador só ocorrerá no caso de uma mudança voluntária para o estado de espera ou quando um processo de prioridade maior passa para o estado de pronto (**preempção por prioridade**).

A rotina de escalonamento deve reavaliar as prioridades dos processos no estado de pronto, para verificar se há processos com maior prioridade do que o processo em execução.

8.9 Escalonamento por Prioridades



8.9 Escalonamento por Prioridades



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

8.9 Escalonamento por Prioridades

O escalonamento por prioridades também pode ser implementado de uma maneira **não-preemptiva**.

A prioridade de execução é uma característica do contexto de software de um processo, e pode ser classificada como **estática** e **dinâmica**.

Um dos principais problemas do escalonamento circular é o ***starvation***. Uma solução para esse problema é a técnica de ***aging***.

O escalonamento por prioridades possibilita diferenciar os processos segundo critérios de importância.

8.10 Escalonamento Circular com Prioridades

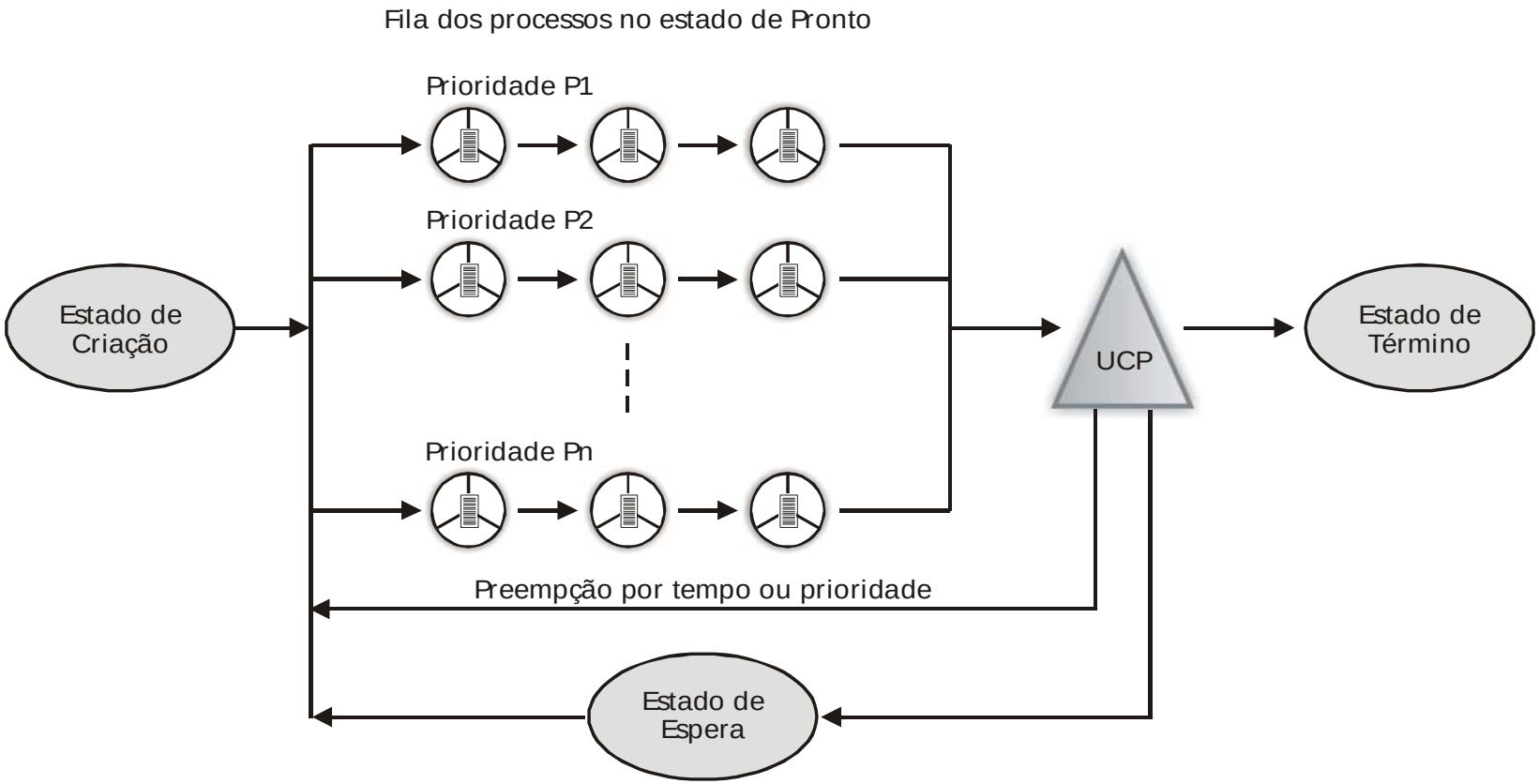
Implementa o conceito de fatia de tempo e de prioridade de execução associada a cada processo.

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.

A **vantagem** deste tipo de escalonamento é permitir o melhor balanceamento no uso do processador em sistemas de tempo compartilhado.

Possui **duas** variações: circular com prioridades estáticas e circular com prioridades dinâmicas.

8.10 Escalonamento Circular com Prioridades



8.11 Escalonamento por Múltiplas Filas

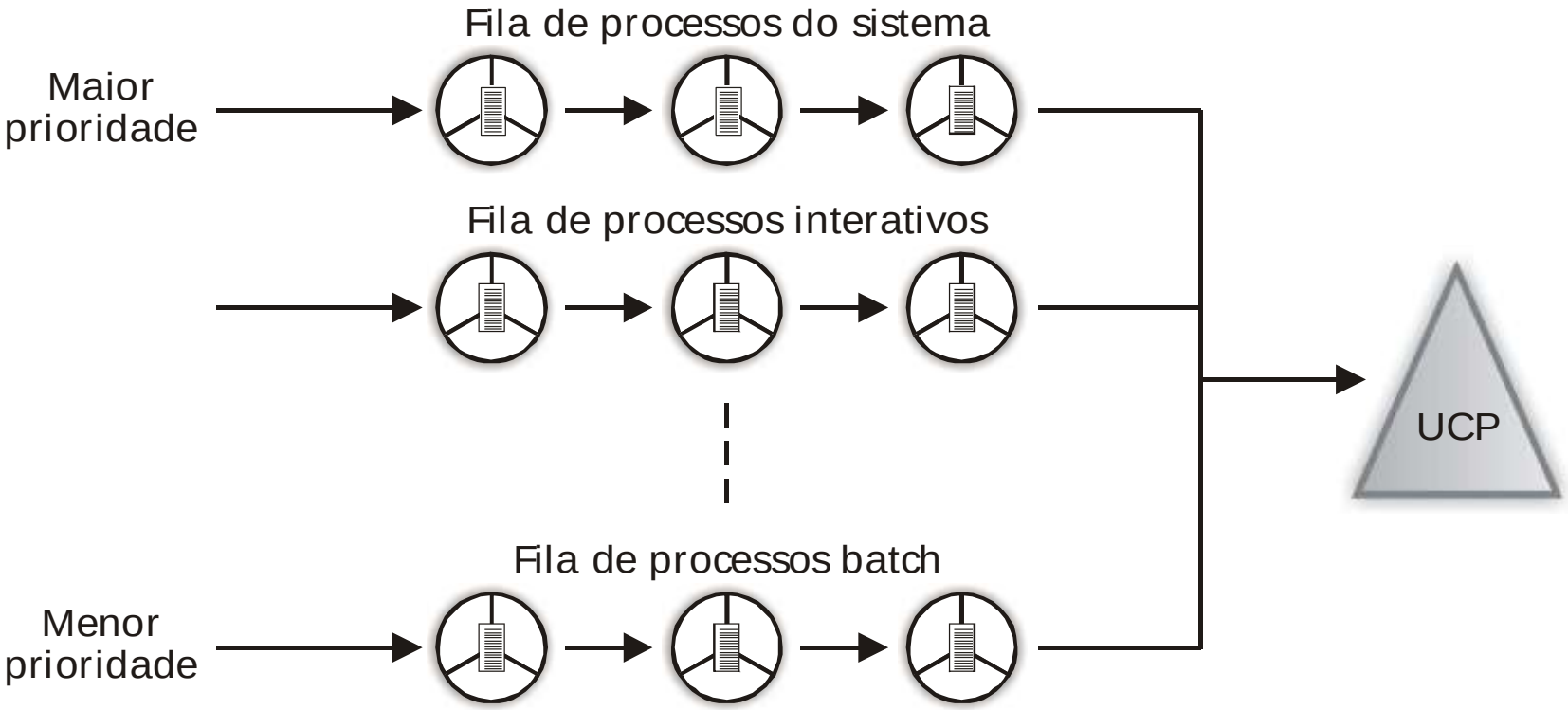
Neste tipo de escalonamento, os processos são associados às filas em função de características próprias, como importância para a aplicação, tipo de processamento ou área de memória necessária.

A principal **vantagem** de múltiplas filas é a possibilidade da convivência de mecanismos de escalonamento distintos em um mesmo sistema operacional.

Além disso, neste mecanismo o processo não possui prioridade, ficando essa característica associada à fila. O processo sofre preempção caso um outro processo entre em uma fila de maior prioridade.

Desvantagem: no caso de um processo alterar seu comportamento no decorrer do tempo, não poderá ser redirecionado para outra fila.

8.11 Escalonamento por Múltiplas Filas



8.12 Escalonamento por Múltiplas Filas com Realimentação

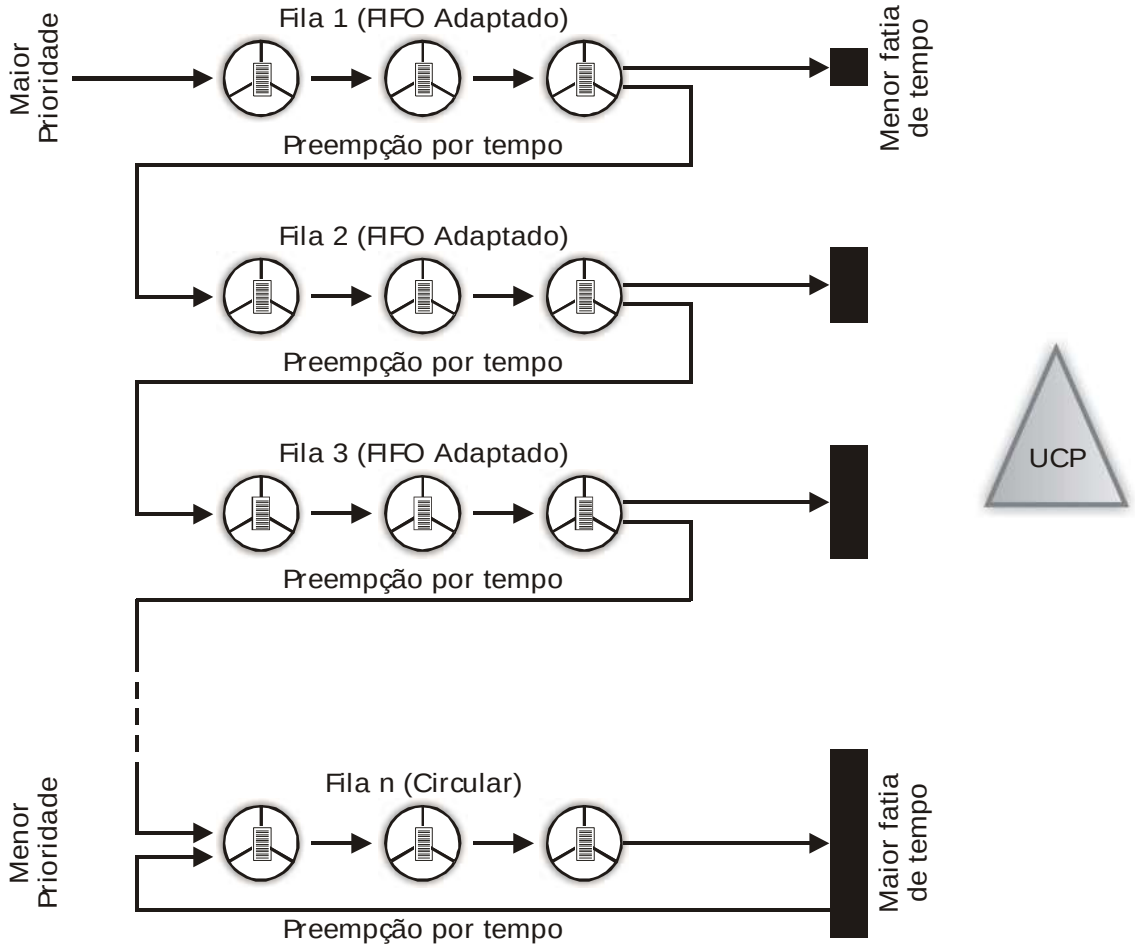
Neste tipo de escalonamento, os processos podem trocar de filas durante o seu processamento.

Vantagem: permitir ao SO identificar dinamicamente o comportamento de cada processo, direcionando-o para filas com prioridade de execução e mecanismo de escalonamento mais adequados.

O escalonamento de um processo em uma fila ocorre apenas quando todas as outras filas de prioridades mais altas estiverem vazias.

Problemas: complexidade de implementação; mudança de comportamento de um processo CPU-bound para I/O-bound pode comprometer seu tempo de resposta.

8.12 Escalonamento por Múltiplas Filas com Realimentação



8.13 Política de Escalonamento em Sistemas de Tempo Compartilhado

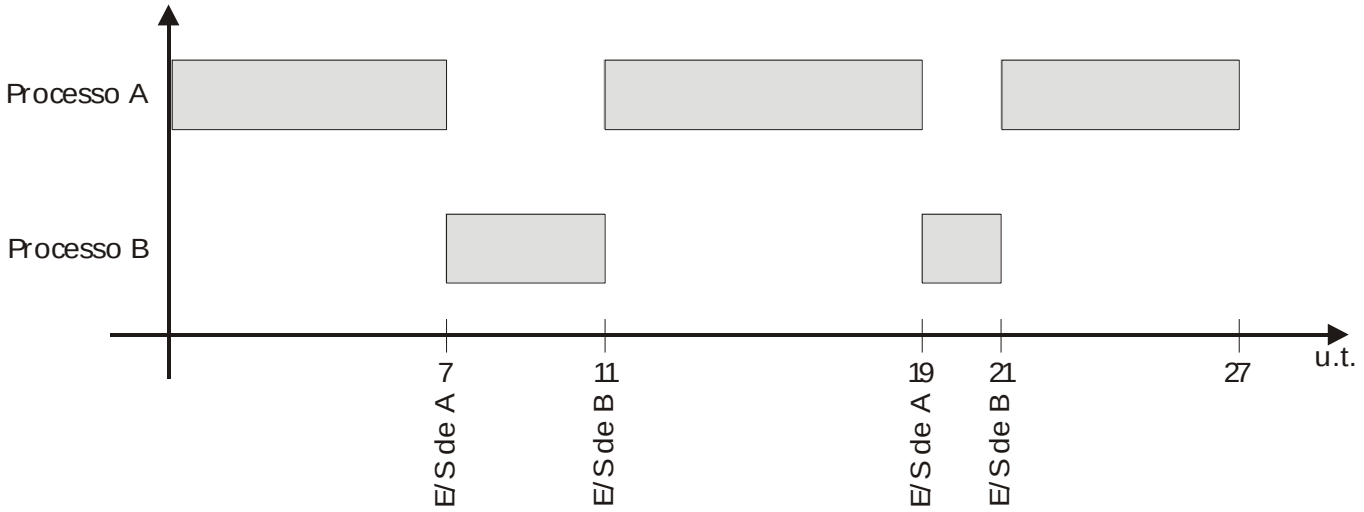
Em geral, sistemas de tempo compartilhado caracterizam-se pelo processamento interativo, no qual os usuários interagem com as aplicações exigindo tempos de respostas baixos.

A escolha de uma política de escalonamento para atingir esse propósito deve levar em consideração o compartilhamento dos recursos de forma equitativa para possibilitar o uso balanceado da UCP entre os processos.

Atualmente, a maioria dos sistemas operacionais de tempo compartilhado utiliza o escalonamento circular com prioridades dinâmicas.

8.13 Política de Escalonamento em Sistemas de Tempo Compartilhado

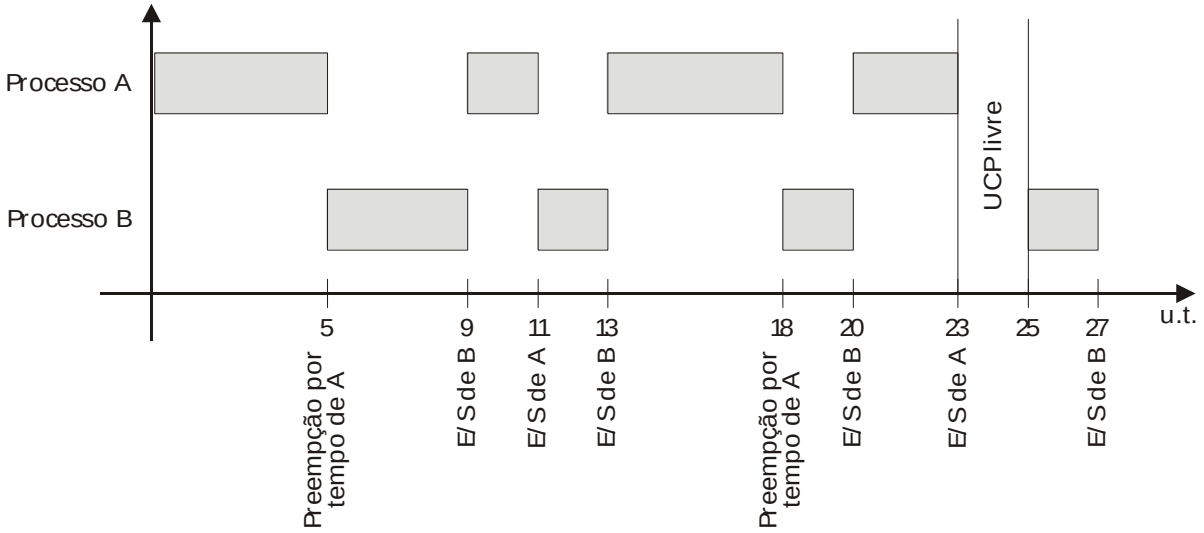
Exemplo: escalonamento FIFO.



Processo	Tempo de processador (u.t.)	Característica
A	21	CPU-bound
B	6	I/O-bound

8.13 Política de Escalonamento em Sistemas de Tempo Compartilhado

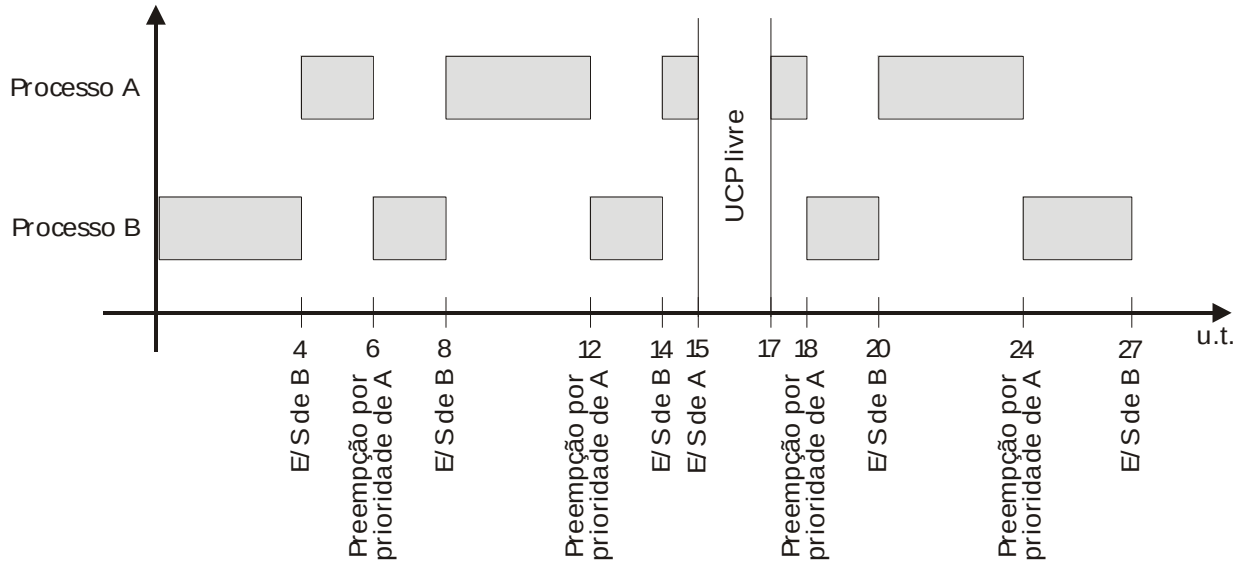
Exemplo: escalonamento circular com fatia de tempo igual a 5 u.t.



Processo	Tempo de processador (u.t.)	Característica
A	15	CPU-bound
B	10	I/O-bound

8.13 Política de Escalonamento em Sistemas de Tempo Compartilhado

Exemplo: escalonamento circular com prioridades.



Processo	Tempo de processador (u.t.)	Característica	Prioridade
A	12	CPU-bound	Baixa
B	13	I/O-bound	Alta

8.14 Política de Escalonamento em Sistemas de Tempo Real

Algumas aplicações específicas exigem respostas imediatas para a execução de terminadas tarefas. Nesse caso, a aplicação deve ser executada em sistemas de tempo real.

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 para sistemas de tempo real. Não deve existir o conceito de fatia de tempo, e a prioridade de cada processo deve ser **estática**.