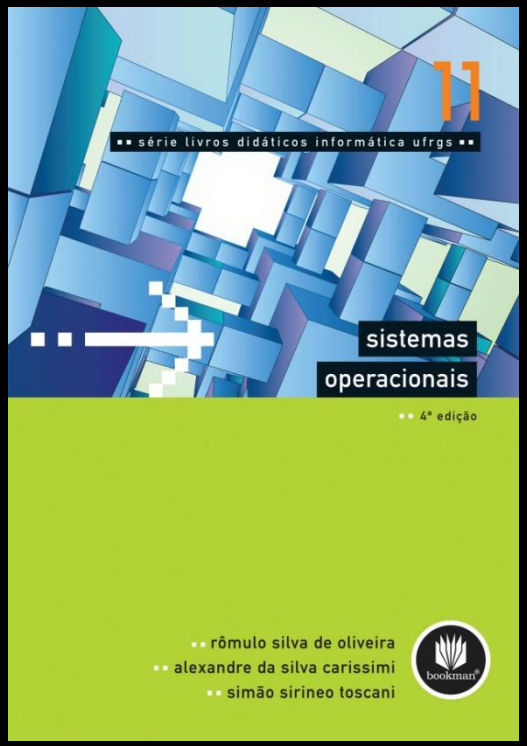
série de livros didáticos informática ufrgs

bookman®
EMPRESA DO GRUPO ARTMED
www.bookman.com.br

.inf
INSTITUTO DE INFORMÁTICA UFRGS



Sistemas Operacionais

Rômulo Silva de Oliveira
Alexandre da Silva Carissimi
Simão Sirineo Toscani

Sumário

GERÊNCIA DO PROCESSADOR

- Escalonamento
  - Escalonadores não preemptivos
- Escalonamento
  - Escalonamento preemptivos

Escalonamento

- O escalonador é a entidade do sistema operacional responsável por selecionar um processo apto para executar no processador
- O objetivo é dividir o tempo do processador de forma justa entre os processos aptos a executar
- Típico de sistemas multiprogramados: *batch, time-sharing*, multiprogramado ou tempo real
  - Requisitos e restrições diferentes em relação a utilização da CPU
- Duas partes:
  - Escalonador: política de seleção
  - *Dispatcher:* efetua a troca de contexto

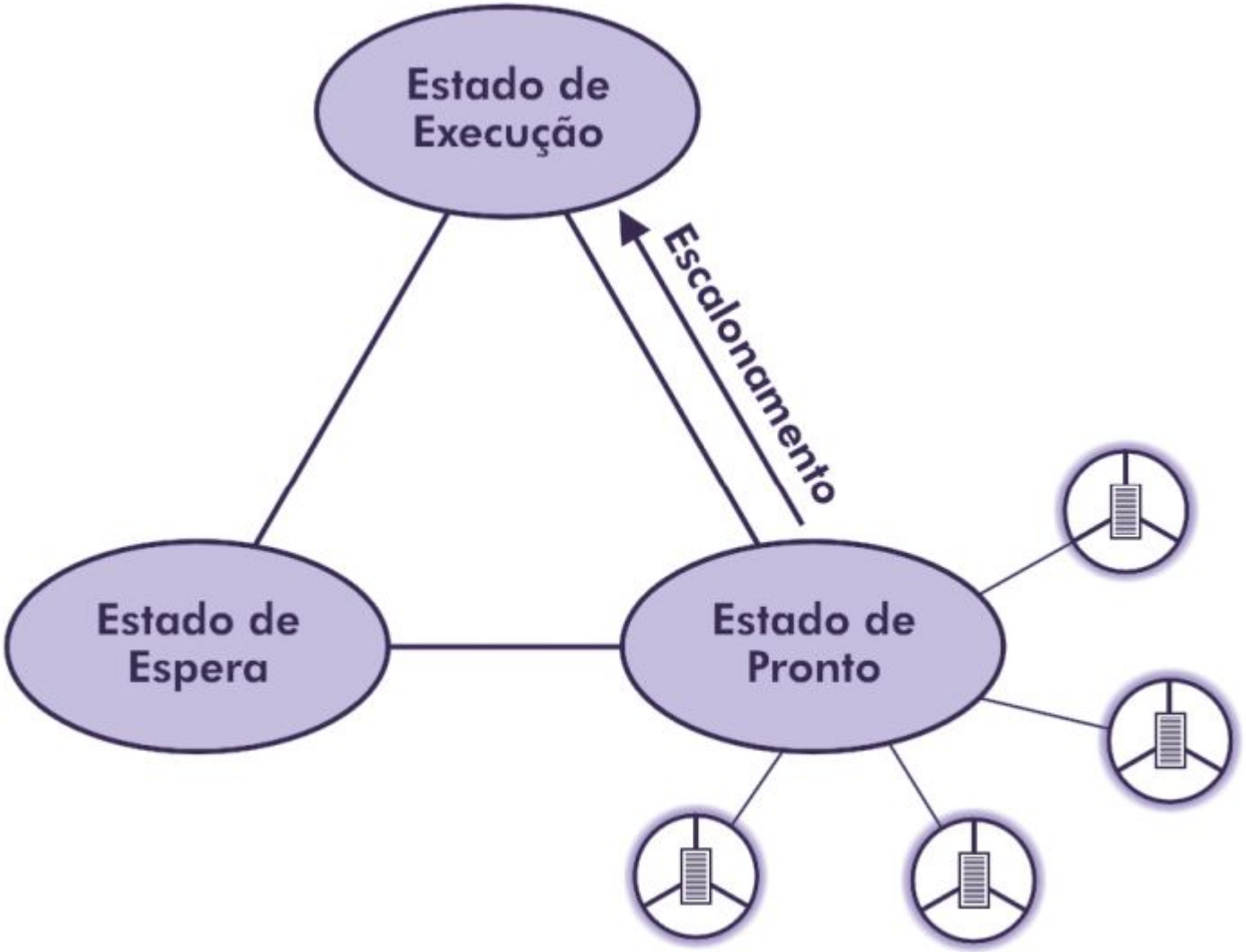


Fig. 8.1 Escalonamento.

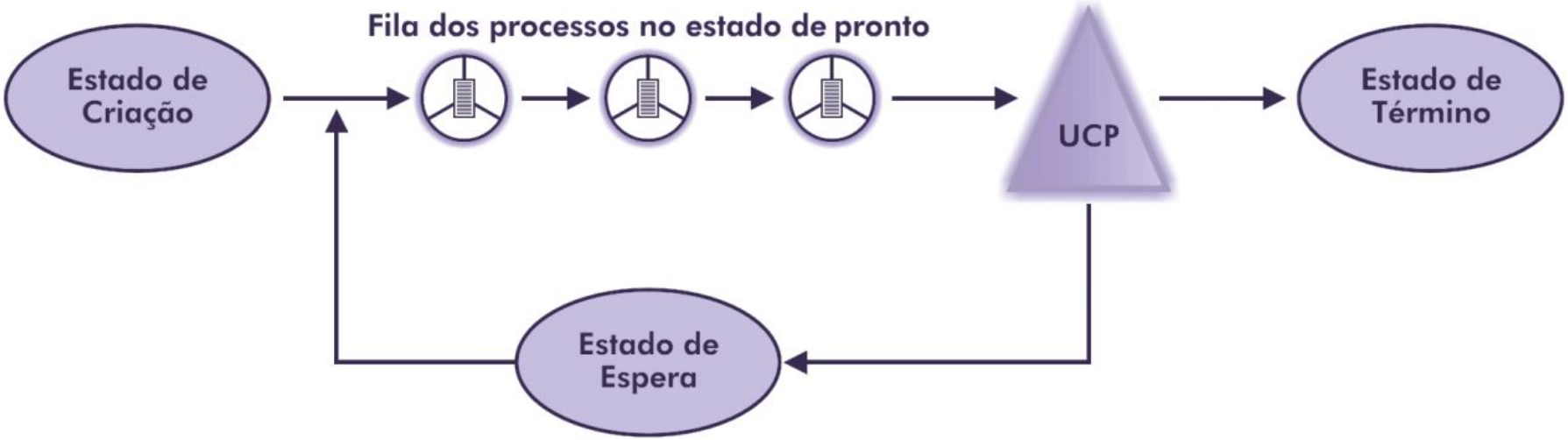


Fig. 8.2 Escalonamento FIFO.

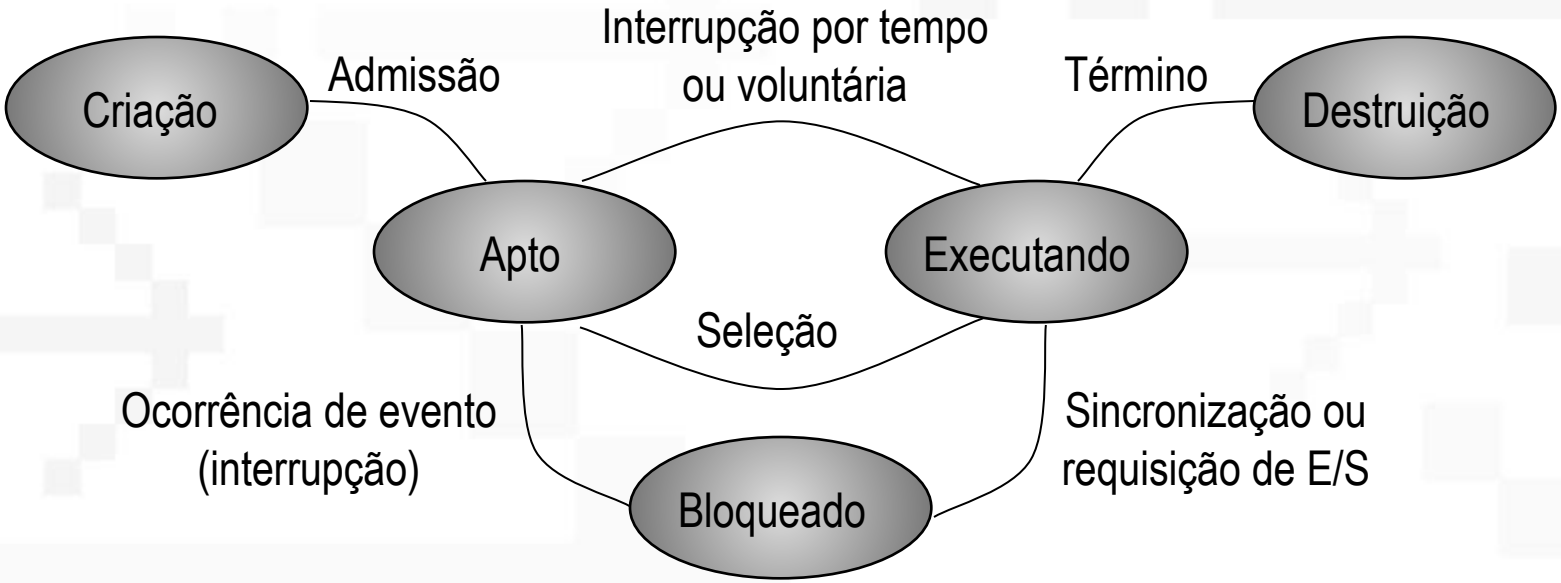
Objetivos do escalonamento

- Maximizar a utilização do processador
- Maximizar a produção do sistema (*throughput*)
  - Número de processos executados por unidade de tempo
- Minimizar o tempo de execução (*turnaround*)
  - Tempo total para executar um determinado processo
- Minimizar o tempo de espera
  - Tempo que um processo permanece na lista de aptos
- Minimizar o tempo de resposta
  - Tempo decorrido entre uma requisição e a sua realização

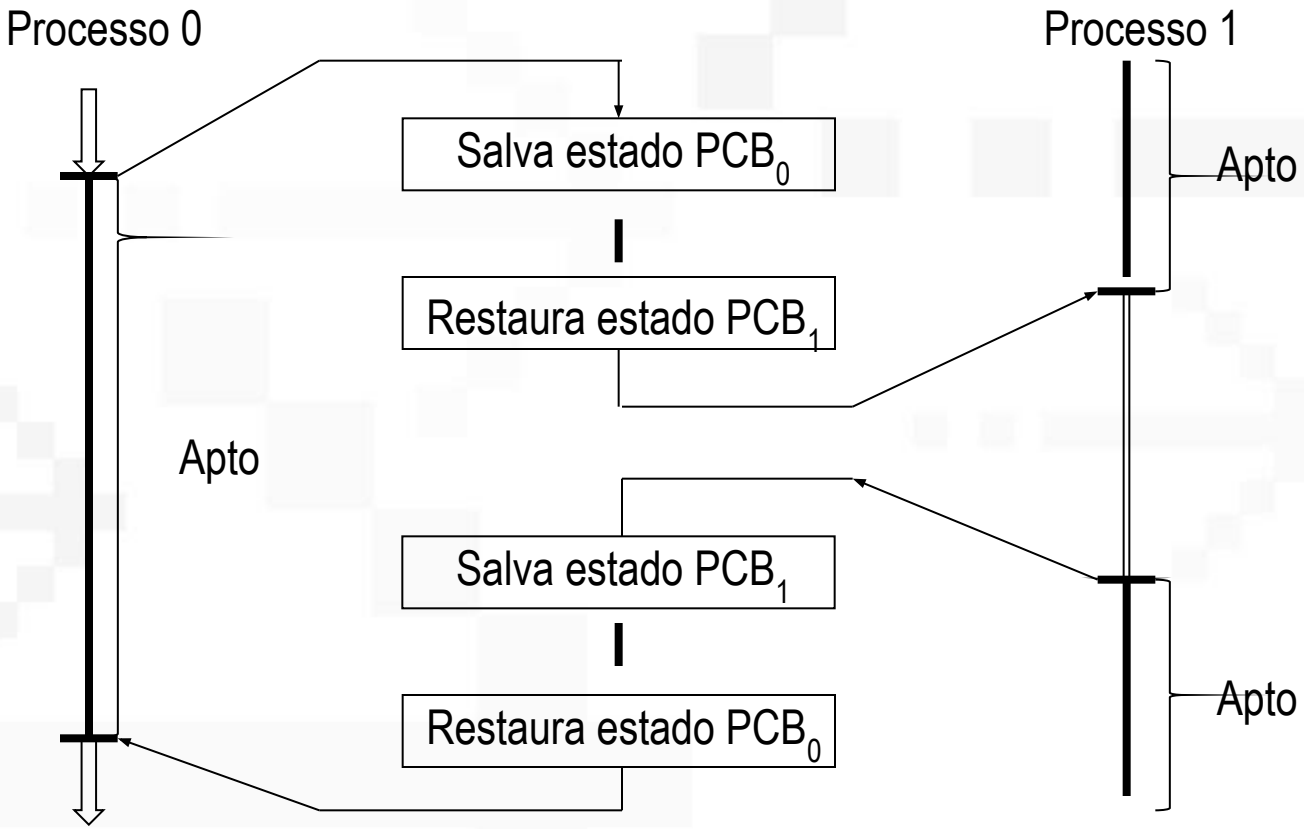
Situações típicas para execução do escalonador

- Dependem se o escalonador é preemptivo ou não, se considera prioridades ou não, etc...
  - Sempre que a CPU estiver livre e houver processos aptos a executar
  - Criação e término de processos
  - Um processo de mais alta prioridade ficar apto a executar
  - Interrupção de tempo
    - Processo executou por um período de tempo máximo permitido
  - Interrupção de dispositivos de entrada e saída
  - Interrupção por falta de página (segmento) em memória
    - Endereço acessado não está carregado na memória (memória virtual)
  - Interrupção por erros

Eventos de transição de estados



Chaveamento de contexto (*dispatcher*)



PCB: *Process Control Block*

Níveis de escalonamento

- Longo prazo
- Médio prazo
- Curto prazo

Escalonador longo prazo

- Executado quando um novo processo é criado
- Determina quando um processo novo passa a ser considerado no sistema, isto é, quando após sua criação ele passa a ser apto
  - Controle de admissão
- Controla o grau de multiprogramação do sistema
  - Quanto maior o número de processos ativos, menor a porcentagem de tempo de uso do processador por processo

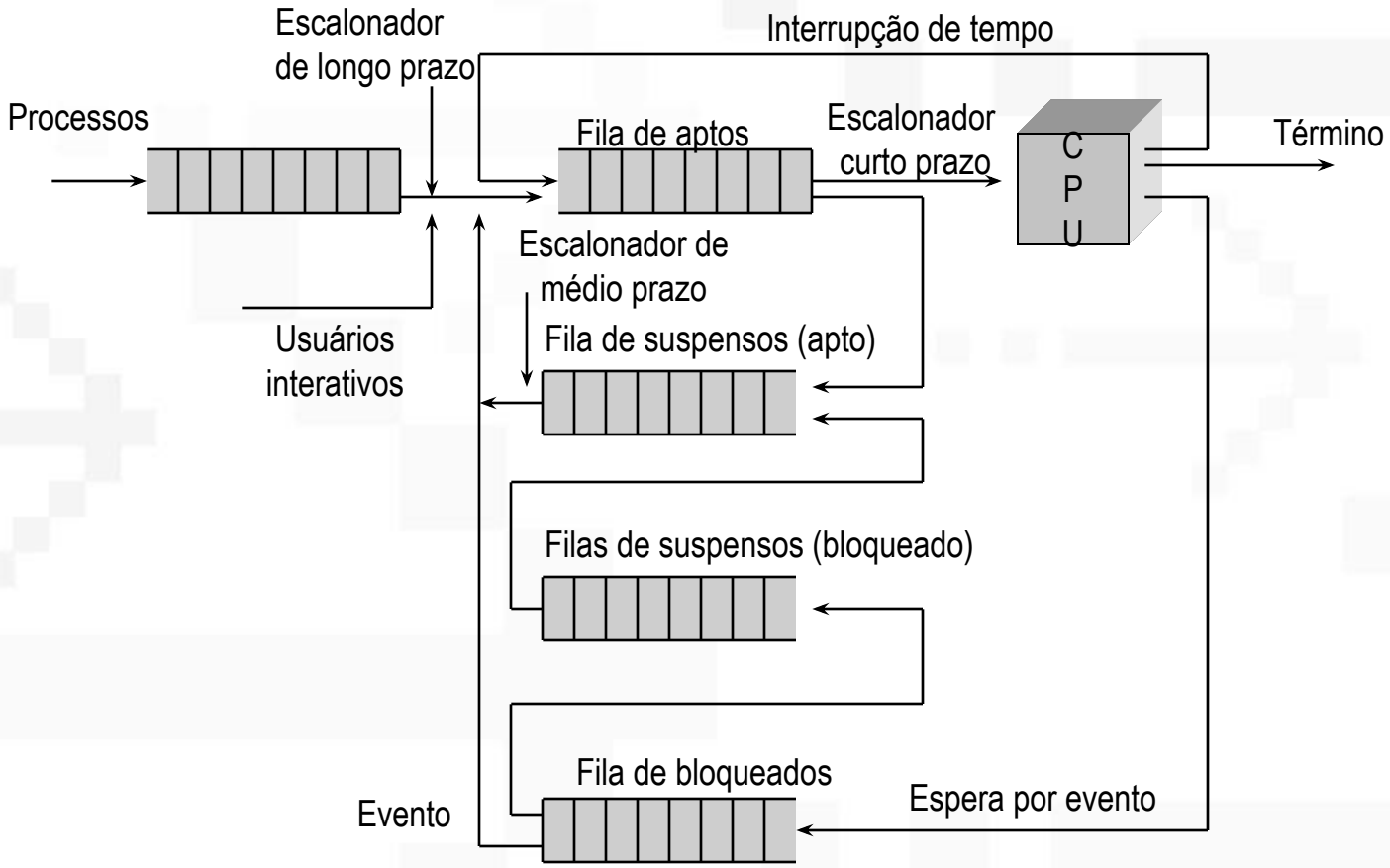
Escalonador médio prazo

- Associado a gerência de memória
  - Participa do mecanismo de *swapping*
- Suporte adicional a multiprogramação
  - Grau de multiprogramação efetiva (diferencia aptos dos aptos-suspensos)

Escalonador de curto prazo

- Mais importante
- Determina qual processo apto deverá utilizar o processador
- Executado sempre que ocorre eventos importantes:
  - Interrupção de relógio
  - Interrupção de entrada/saída
  - Chamadas de sistemas
  - Sinais (interrupção software)

Diagrama de escalonamento



Tipos de escalonador

- Um vez escalonado, o processo utiliza o processador até que:
  - **Não preemptivo**:
    - Término de execução do processo
    - Execução de uma requisição de entrada/saída ou sincronização
    - Liberação voluntária do processador a outro processo (*yield*)
  - **Preemptivo**:
    - Término de execução do processo
    - Execução de uma requisição de entrada/saída ou sincronização
    - Liberação voluntária do processador a outro processo (*yield*)
    - Interrupção de relógio
    - Processo de mais alta prioridade esteja pronto para executar

Algoritmos de escalonamento (1)

- Algoritmo de escalonamento seleciona qual processo deve executar em um determinado instante de tempo
- Existem vários algoritmos para atingir os objetivos do escalonamento
- Os algoritmos buscam:
  - Obter bons tempos médios invés de maximizar ou minimizar um determinado critério
  - Privilegiar a variância em relação a tempos médios

Algoritmos de escalonamento (2)

- Algoritmos não preemptivos (cooperativos)
  - *First-In First-Out* (FIFO) ou *First-Come First-Served* (FCFS)
  - *Shortest Job First* (SJF) ou *Shortest Process Next* (SPN)
- Algoritmos preemptivos
  - Round robin (circular)
  - Baseado em prioridades
- Existem outros algoritmos de escalonamento
  - *High Response Ratio Next* (HRRN)
  - *Shortest Remaining Time* (SRT)
  - etc...

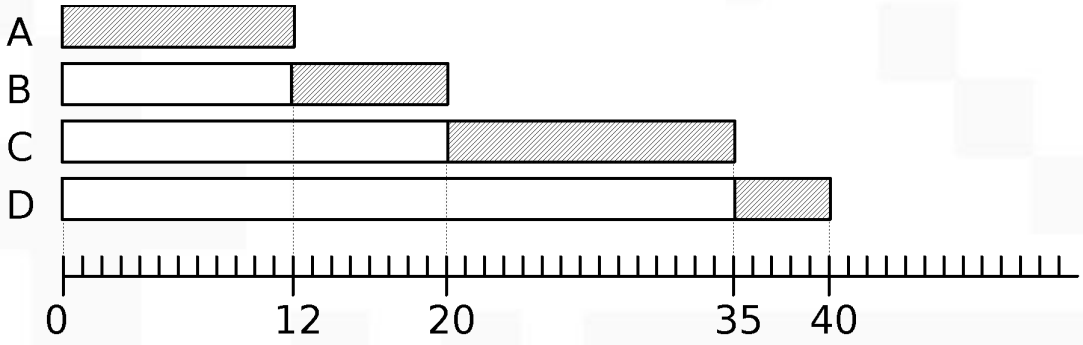
FIFO - *First In First Out* (1)

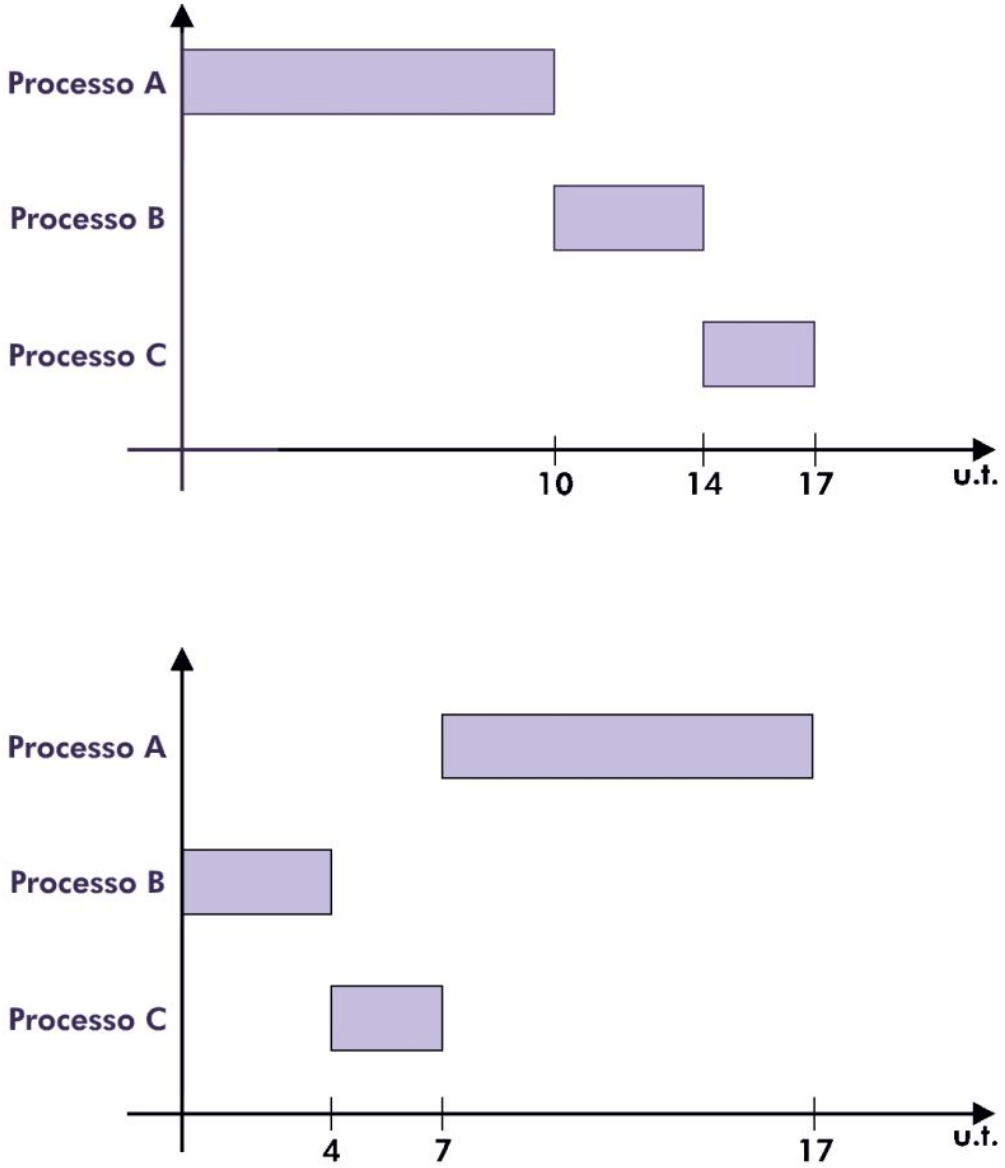
- *First-Come, First-Served* (FCFS)
- Simples de implementar
  - Fila
- Funcionamento:
  - Processos que se tornam aptos são inseridos no final da fila
  - Processo que está no início da fila é o próximo a executar
  - Processo executa até que:
    - Libere explicitamente o processador
    - Realize uma chamada de sistema (bloqueado)
    - Termine sua execução

FIFO - *First In First Out* (2)

- Desvantagem:
  - Prejudica processos *I/O bound*
- Tempo médio de espera na fila de execução:
  - Ordem A-B-C-D = (0 + 12 + 20 + 35) / 4 = 16.75 u.t.
  - Ordem D-A-B-C = (0 + 5 + 17 + 25) / 4 = 11.7 u.t.

Processo	Tempo
A	12
B	8
C	15
D	5





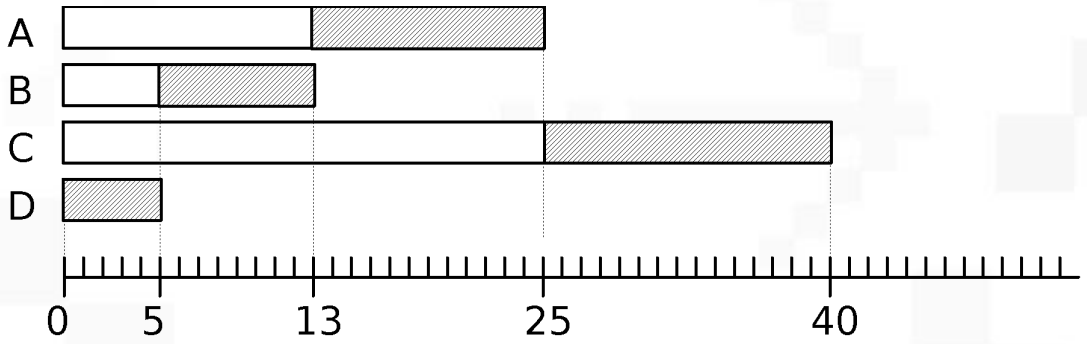
Processo	Tempo de processador (u.t.)
A	10
B	4
C	3

Fig. 8.3 Escalonamento FIFO (exemplo).

SJF - *Shortest Job First* (1)

- Originário do fato que o menor tempo de médio é obtido quando se executa primeiro os processos de menor ciclo de processador (*I/O bound*)

Processo	Tempo
A	12
B	8
C	15
D	5



Tempo médio: (0 + 5 + 13 + 25)/4 = 10.75 u.t

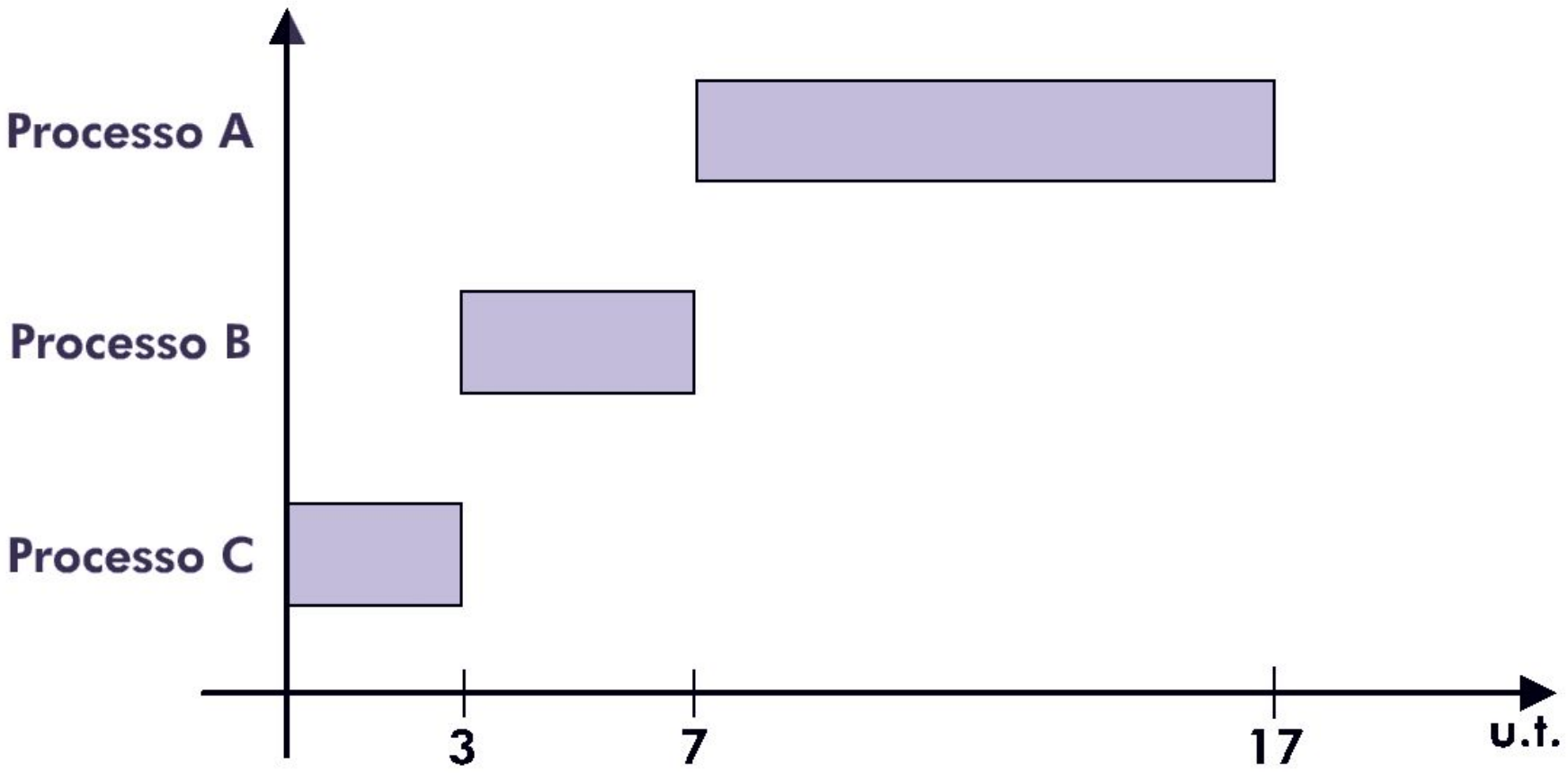


Fig. 8.4 Escalonamento SJF (exemplo).

SJF - *Shortest Job First* (2)

- Algoritmo ótimo, isto é, fornece o menor tempo médio de espera para um conjunto de processos
- Processos *I/O bound* são favorecidos
- Dificuldade é determinar o tempo do próximo ciclo de CPU de cada processo, porém:
  - Pode ser empregado em processos *batch* (*long term scheduler*)
  - Prever o futuro com base no passado

Tipos de escalonador (lembrando...)

- Um vez escalonado, o processo utiliza o processador até que:
  - Não preemptivo:
    - Término de execução do processo
    - Execução de uma requisição de entrada/saída ou sincronização
    - Liberação voluntária do processador a outro processo (*yield*)
  - Preemptivo:
    - Término de execução do processo
    - Execução de uma requisição de entrada/saída ou sincronização
    - Liberação voluntária do processador a outro processo (*yield*)
    - Interrupção de relógio
    - Processo de mais alta prioridade esteja pronto para executar

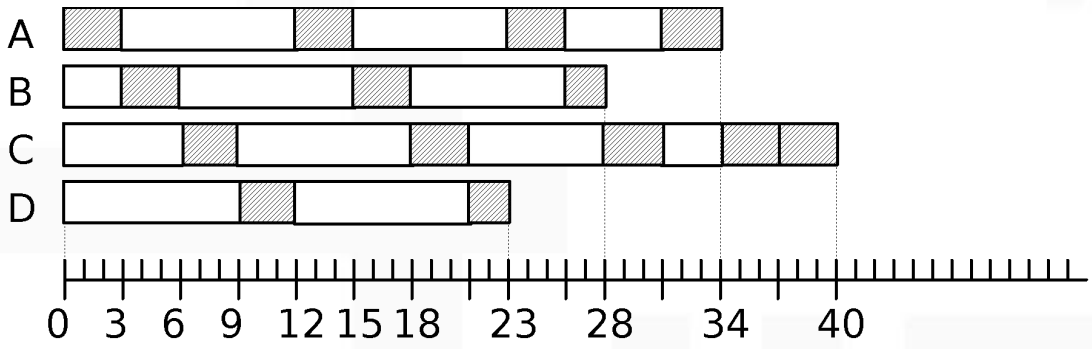
Escalonadores preemptivos

- Por interrupção de tempo
  - Round robin (circular)
- Por prioridades

Um processo é dito preemptivo, se o mesmo pode perder o processador por algum motivo que não seja o término de seu ciclo de processador

RR - *Round Robin* (1)

- Similar ao algoritmo FIFO, só que:
  - Cada processo recebe um tempo limite máximo (*time-slice*, *quantum*) para executar um ciclo de processador
- Fila de processos aptos é uma fila circular
- Necessidade de um relógio para delimitar as fatias de tempo
  - Interrupção de tempo



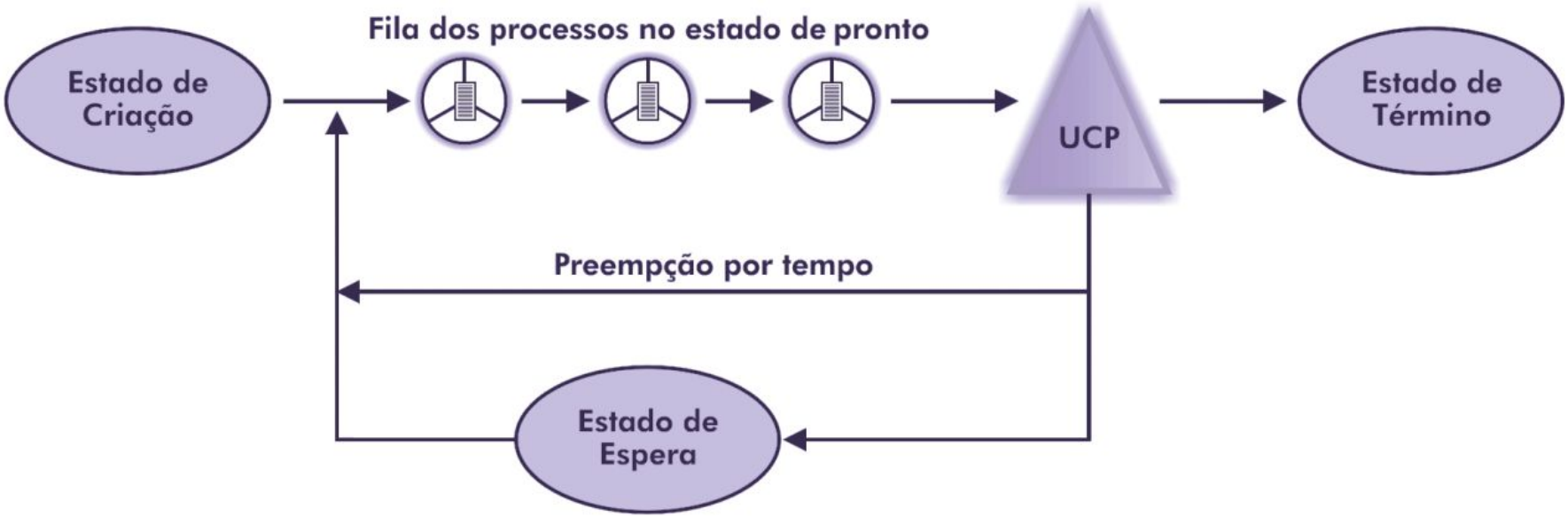


Fig. 8.5 Escalonamento circular.

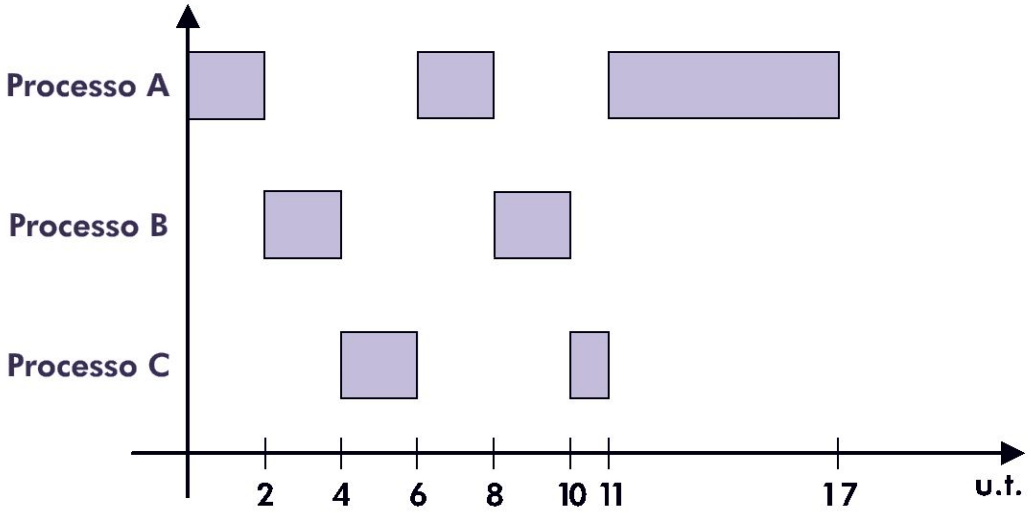


Fig. 8.6 Escalonamento circular (exemplo).

RR - *Round Robin* (2)

- Por ser preemptivo, um processo perde o processador quando:
  - Libera explicitamente o processador (*yield*)
  - Realize uma chamada de sistema (bloqueado)
  - Termina sua execução
  - Quando sua fatia de tempo é esgotada
- Se *quantum* → ∞ obtém-se o comportamento de um escalonador FIFO

Problemas com o *Round Robin*

- Problema 1: Dimensionamento do *quantum*
  - Compromisso entre *overhead* e tempo de resposta em função do número de usuários (*1/k* na presença de *k* usuários)
  - Compromisso entre tempo de chaveamento e tempo do ciclo de processador (*quantum*)
- Problema 2: Processos *I/O bound* são prejudicados
  - Esperam da mesma forma que processos *CPU bound* porém muito provavelmente não utilizam todo o seu *quantum*
  - Solução:
    - Prioridades: Associar prioridades mais altas aos processos *I/O bound* para compensar o tempo gasto no estado de espera (apto)

Escalonamento com prioridades

- Sempre que um processo de maior prioridade que o processo atualmente em execução entrar no estado apto deve ocorrer uma preempção
  - A existência de prioridades pressupõem a preempção
  - É possível haver prioridade não-preemptiva
- Escalonador deve sempre selecionar o processo de mais alta prioridade segundo uma política:
  - Round-Robin
  - FIFO (FCFS)
  - SJF (SPN)

Implementação de escalonador com prioridades

- Múltiplas filas associadas ao estado apto
- Cada fila uma prioridade
  - Pode ter sua própria política de escalonamento (FIFO, SJF, RR)

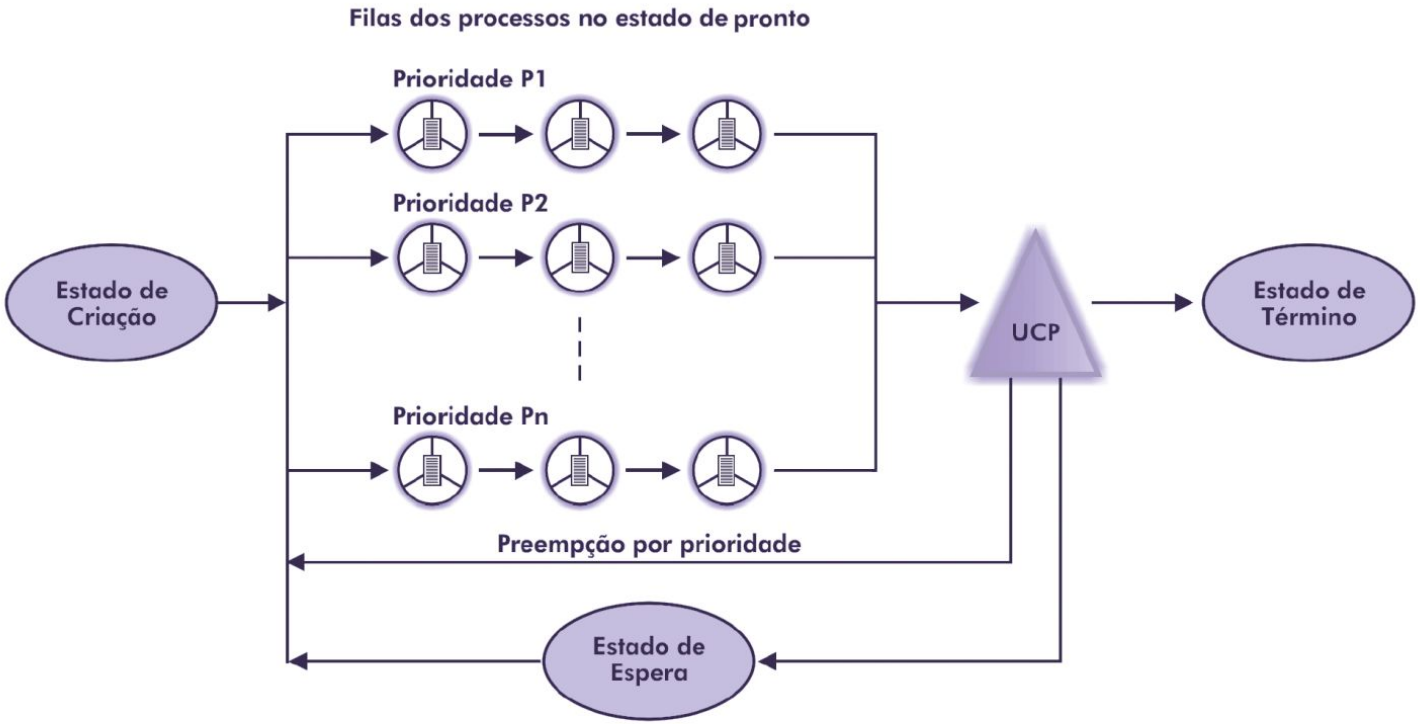


Fig. 8.8 Escalonamento por prioridades.

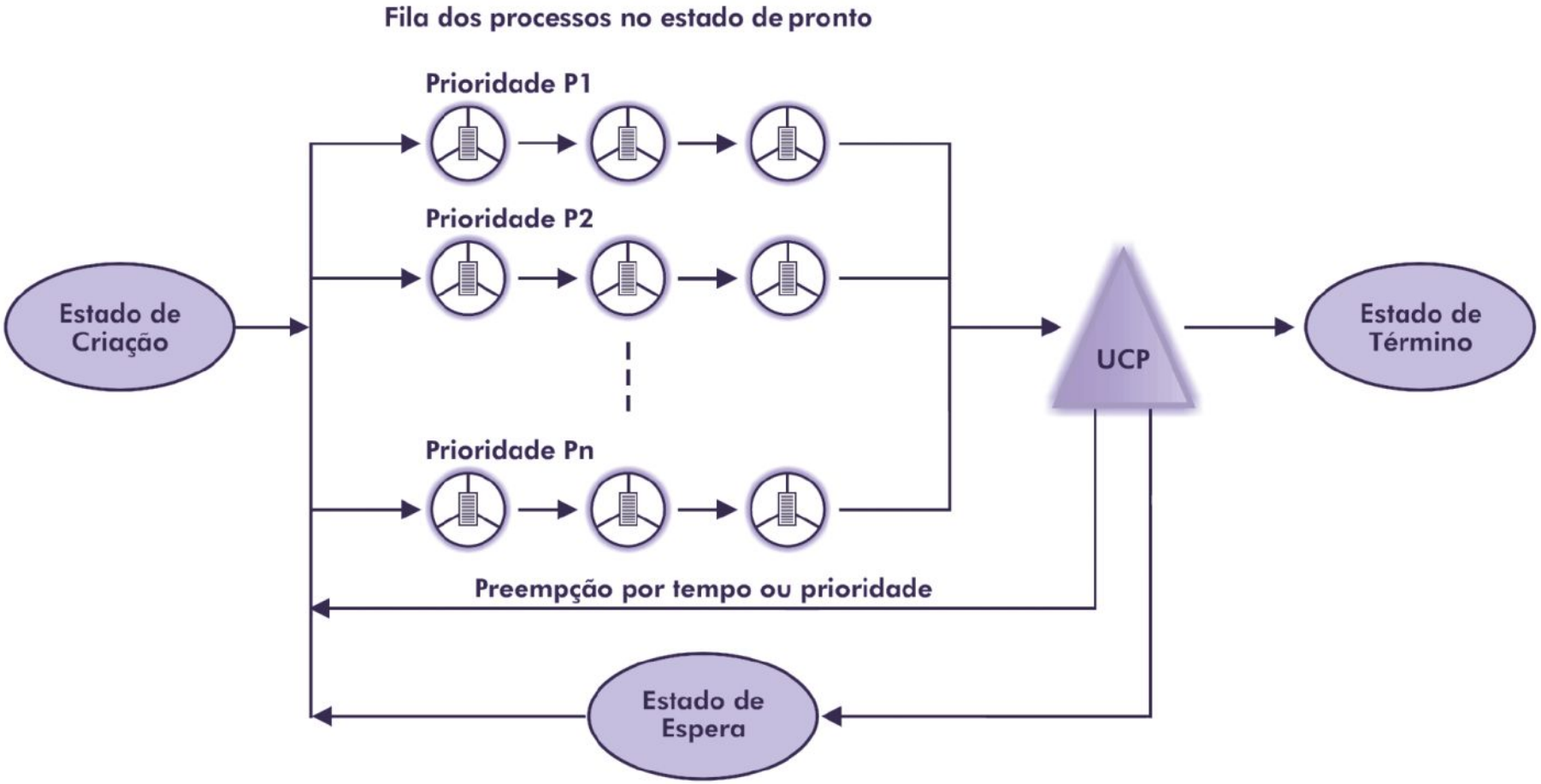


Fig. 8.10 Escalonamento circular com prioridades.

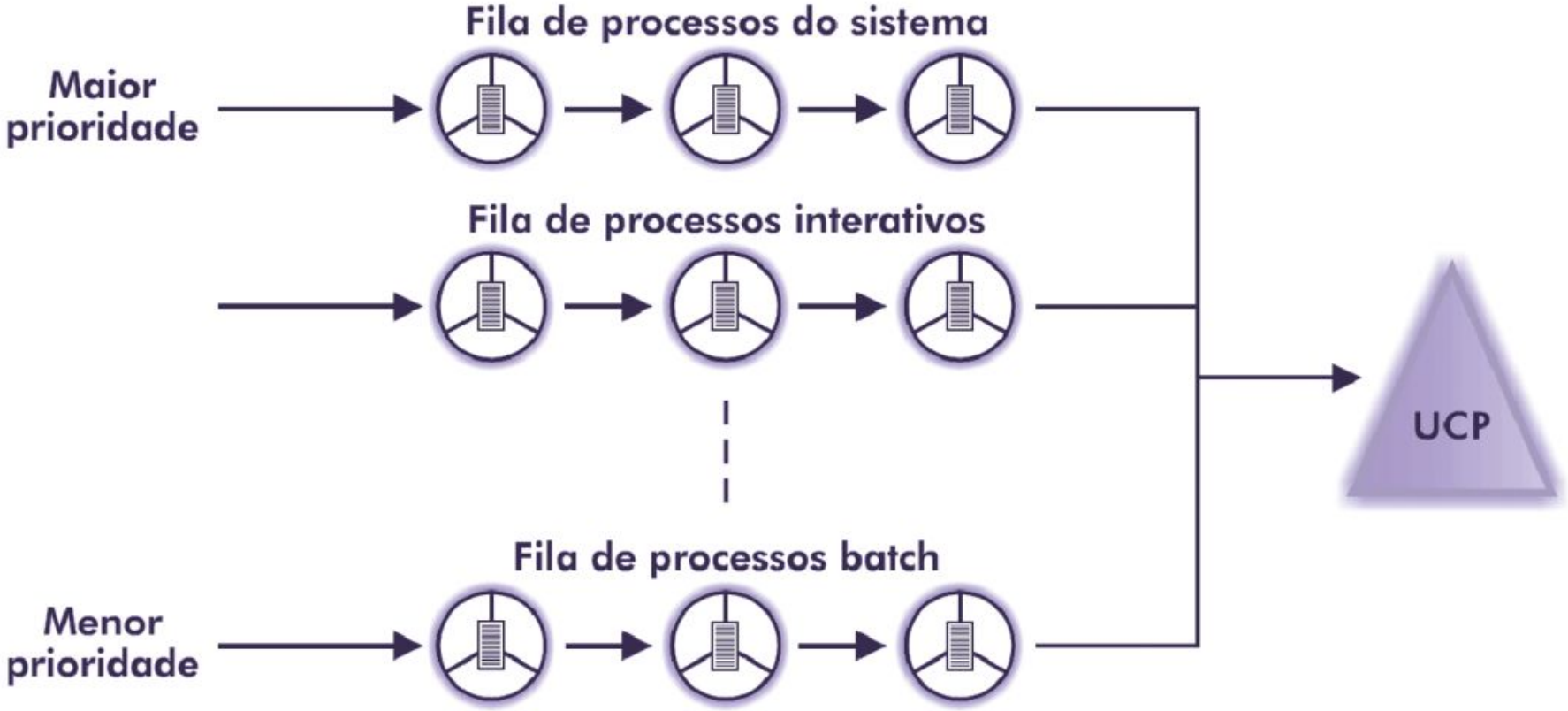


Fig. 8.11 Escalonamento por múltiplas filas.

Exemplo: *pthreads*

- A política de escalonamento FIFO com prioridade considera:
  - Quando um processo em execução é preemptado ele é inserido no ínicio de sua fila de prioridade
  - Quando um processo bloqueado passa a apto ele é inserido no final da fila de sua prioridade
  - Quando um processo troca de prioridade ele é inserido no final da fila de sua nova prioridade
  - Quando um processo em execução “passa a vez” para um outro processo ele é inserido no final da fila de sua prioridade

Como definir a prioridade de um processo?

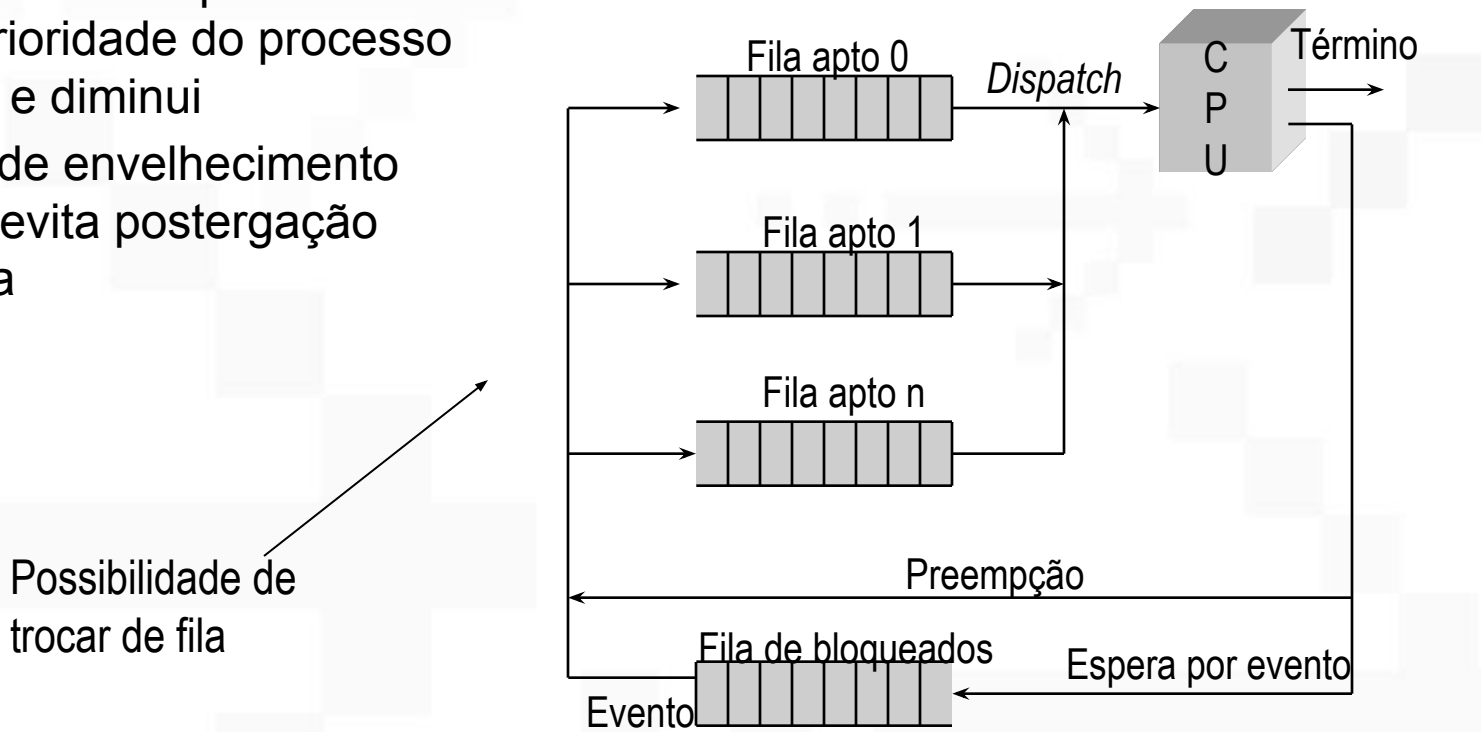
- Prioridade estática:
  - Um processo é criado com uma determinada prioridade e esta prioridade é mantida durante todo o tempo de vida do processo
- Prioridade dinâmica:
  - Prioridade do processo é ajustada de acordo com o estado de execução do processo e/ou do sistema
    - e.g; ajustar a prioridade em função da fração do *quantum* que foi realmente utilizada pelo processo:
      - q = 100 ms
      - Processo A utilizou 2ms □ nova prioridade = 1/0.02 = 50
      - Processo B utilizou 50ms □ nova prioridade = 1/0.5 = 2

Problemas com prioridades

- Um processo de baixa prioridade pode não ser executado
  - Postergação indefinida (*starvation*)
- Processo com prioridade estática pode ficar mal classificado e ser penalizado ou favorecido em relação aos demais
  - Típico de processos que durante sua execução trocam de padrão de comportamento (*CPU bound* a *I/O bound* e vice-versa)
- Solução:
  - Múltiplas filas com realimentação

Múltiplas filas com realimentação

- Baseado em prioridades dinâmicas
- Em função do tempo de uso da CPU a prioridade do processo aumenta e diminui
- Sistema de envelhecimento (*agging*) evita postergação indefinida



Estudo de caso: escalonamento Linux

- Duas classes em função do tipo de processos (*threads*)
  - Processos interativos e *batch*
  - Processos de tempo real
- Políticas de escalonamento do linux (padrão POSIX)
  - SCHED_FIFO: FIFO com prioridade estática
    - Válido apenas para processos de tempo real
  - SCHED_RR: Round-robin com prioridade estática
    - Válido apenas para processos de tempo real
  - SCHED_OTHER: Filas multinível com prioridades dinâmicas (*time-sharing*)
    - Processos interativos e *batch*

Escalonamento linux (*timesharing*)

- Baseado no uso de créditos e prioridade
- Sistema de créditos:
  - Cada processo executa um certo número de créditos
  - O processo com maior crédito é o selecionado
  - Cada interrupção de tempo o processo em execução perde um crédito
  - Processo que atinge zero créditos é suspenso (escalonador médio prazo)
  - Se no estado apto não existir processos com créditos é realizado uma redistribuição de créditos para todos os processos (qualquer estado)

$$Créditos = \frac{Créditos}{2} + prioridade$$

Escalonamento não preemptivo com prioridades

- SJF é um forma de priorizar processos
  - A prioridade é o inverso do próximo tempo previsto para ciclo de CPU
- Processos de igual prioridade são executados de acordo com uma política FIFO
- Problema de postergação indefinida (*starvation*)
  - Processo de baixa prioridade não é alocado a CPU por sempre existir um processo de mais alta prioridade a ser executado
  - Solução:
    - Envelhecimento
- O conceito de prioridade é mais “consistente” com preempção
  - Processo de maior prioridade interrompe a execução de um menos prioritário

SIMULADOR DE ALGORITMOS

http://cpuburst.com/ganttcharts.html

Leituras complementares

- R. Oliveira, A. Carissimi, S. Toscani; *Sistemas Operacionais*. Editora Sagra-Luzzato, 2001.
  - Capítulo 4, Capítulo 9 (seção 9.4), Capítulo 10 (seção10.4)
- A. Silberchatz, P. Galvin; *Operating System Concepts*. (4th edition) Addison-Wesley, 1994.
  - Capítulo 5
- A. Silberchatz, P. Galvin, G. Gane; *Applied Operating System Concepts*. (1st edition). Addison-Wesley, 2000.
  - Capítulo 4, 5 e 6
- W. Stallings; *Operating Systems*. (4th edition). Prentice Hall, 2001.
  - Capítulo 9