Sistemas Operacionais

Gerência de processos

Edson Moreno

edson.moreno@pucrs.br

http://www.inf.pucrs.br/~emoreno

Introdução

- Multiprogramação
  - Permite a execução de diversos processos concorrentemente

- Maior aproveitamento dos recursos
  - processador, periféricos: mais tempo sendo utilizado (diversos processos)
  - memória: diversos processos são carregados na memória

- Requisitos para multiprogramação
  - **Interrupções**: permite utilizar o processador enquanto está sendo realizada E/S (processo bloqueado)
  - **Gerência de memória**: permite carregar diversos programas em memória
  - **Proteção de memória** (dados/código)

Processos

- Conceito: abstração de um programa em execução
  - Um mesmo programa pode estar sendo executado por diversos usuários resultando em diversos processos
  - Um processo é uma estrutura gerenciável pelo SO
    - Incluem dados sobre PC, Segmento de dados, pilha, etc…

- Tipos de processos
  - Processos do **usuário** e processos de **sistema/núcleo/monitor**
  - Processos **cpu-bound** e processos **I/O bound**
  - Processos **pesados** e processos **leves**

- Nomenclatura
  - Sistemas em lote: Job
  - Sistemas de tempo compartilhado: Tarefa ou **Processo**
  - Nomenclatura muito intercambiável

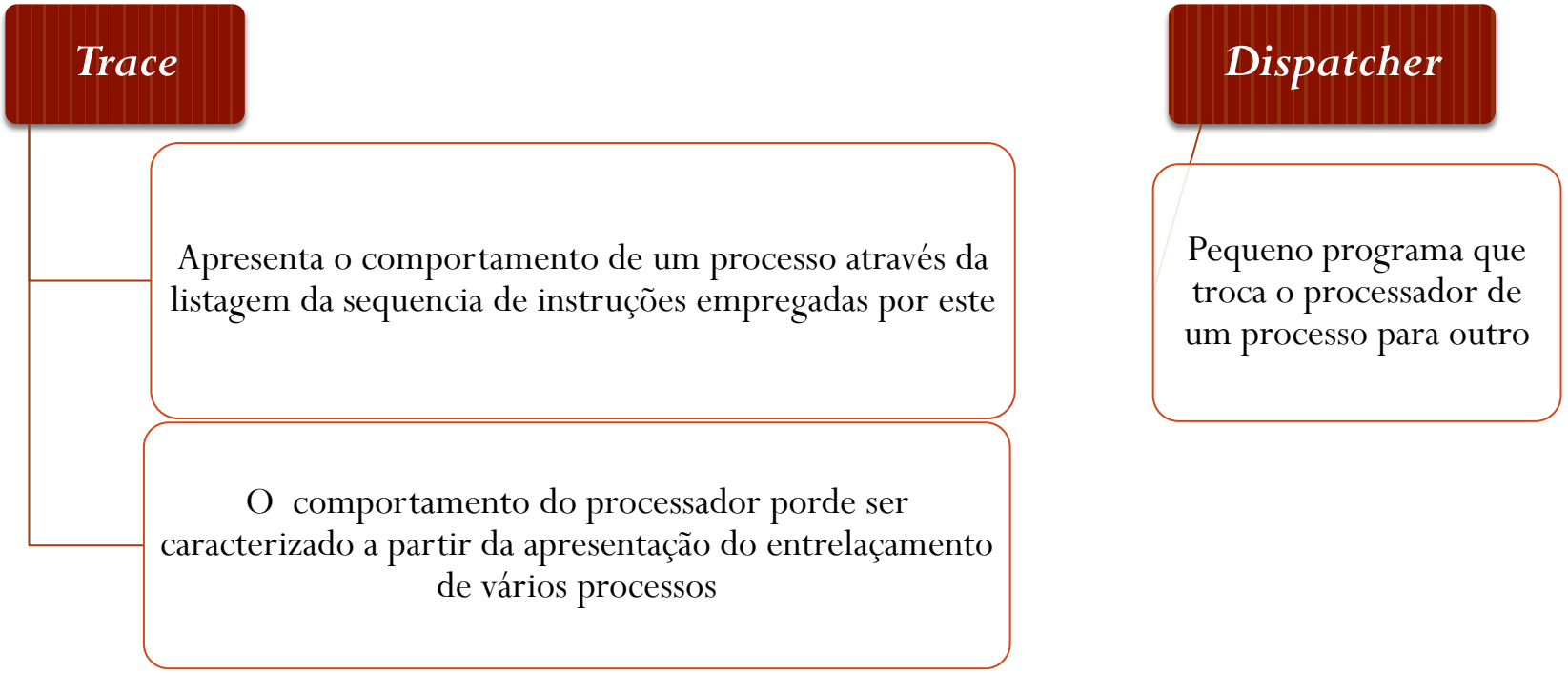
Processos

- Relacionamento entre processos
  - Organização dos processos de forma hierárquica (árvore)
  - Processos pais e processos filhos
  - Compartilhamento de estruturas de acesso, direitos, características
  - Quando um processo morre (abordagens):
    1. Todos os processos filhos são destruídos; ou
    2. Mantém processo até todos os filhos serem destruídos; ou
    3. Vincula processos filhos ao processo avô

- Exemplo: no Linux o processo init é a raiz da árvore
  - De onde são lançados os processos?

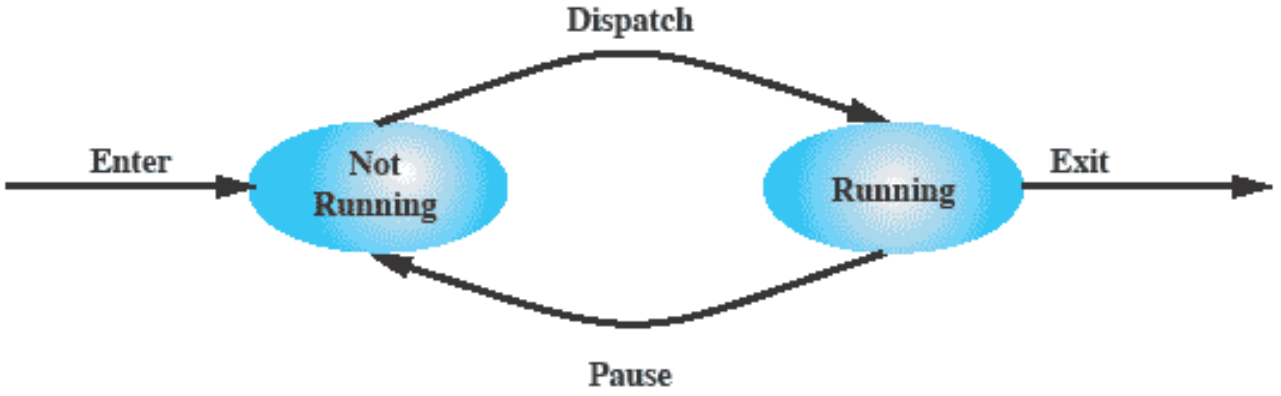
Processos

- Estados de um processo
  - Processador
    - Sempre executando instruções
    - Definida pelo PC
  - PC pode apontar para diferentes processos
    - Manipulação realizada pelo dispatcher (despachante)



Processos

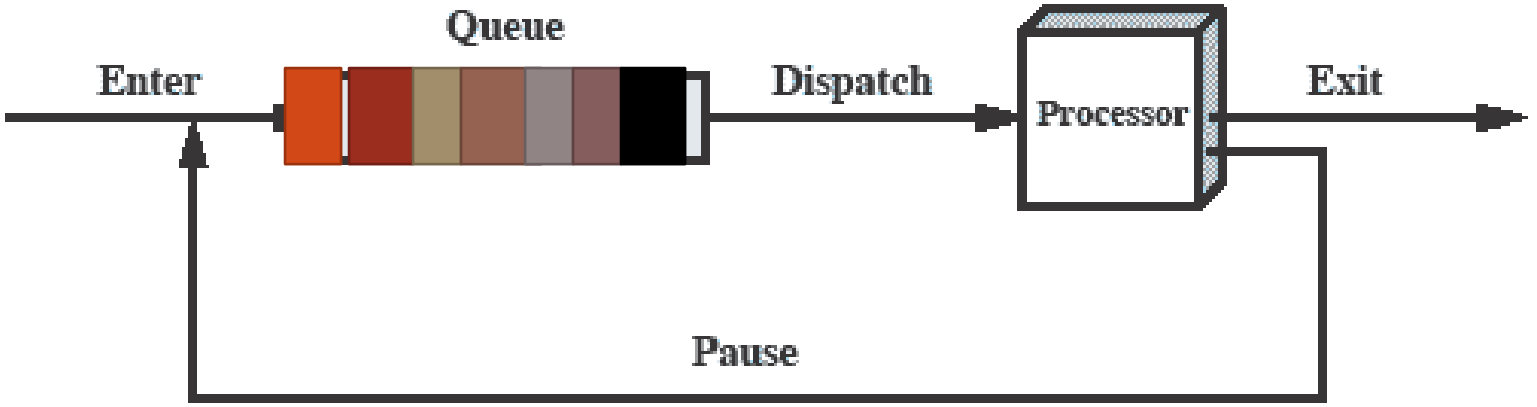
- Estados de um processo
  - Modelo básico de dois estados
    - Executando (*running*)
    - Não executando (*not-running*)



(a) State transition diagram

Processos

- Estados de um processo
  - Modelo básico de dois estados
    - Executando (*running*)
    - Não executando (*not-running*)
  - Diagrama de fila



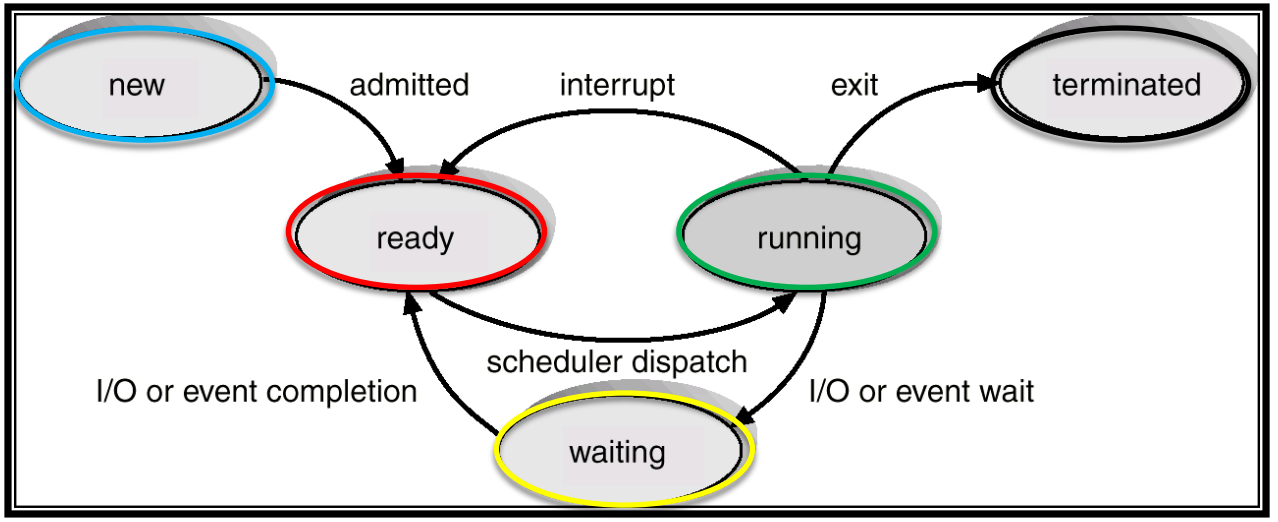
(b) Queuing diagram

Processos

- O que causa a criação de um processo?
  - Quando usuário cria uma sessão ou outro processo requer, ou …
- Mas quais os passos para criar um processo?
  - Basicamente
    - Cria estrutura de gerenciamento do processo
    - Aloca espaço em memória para o processo que chega
- E o que causa o fim de execução de um processo?
  - Um comando de halt (jobs), pedido de fim do processo feito pelo usuário, um erro, …
- O Modelo de dois estados é real, os processos ou estão em execução ou prontos para serem executados?

Processos

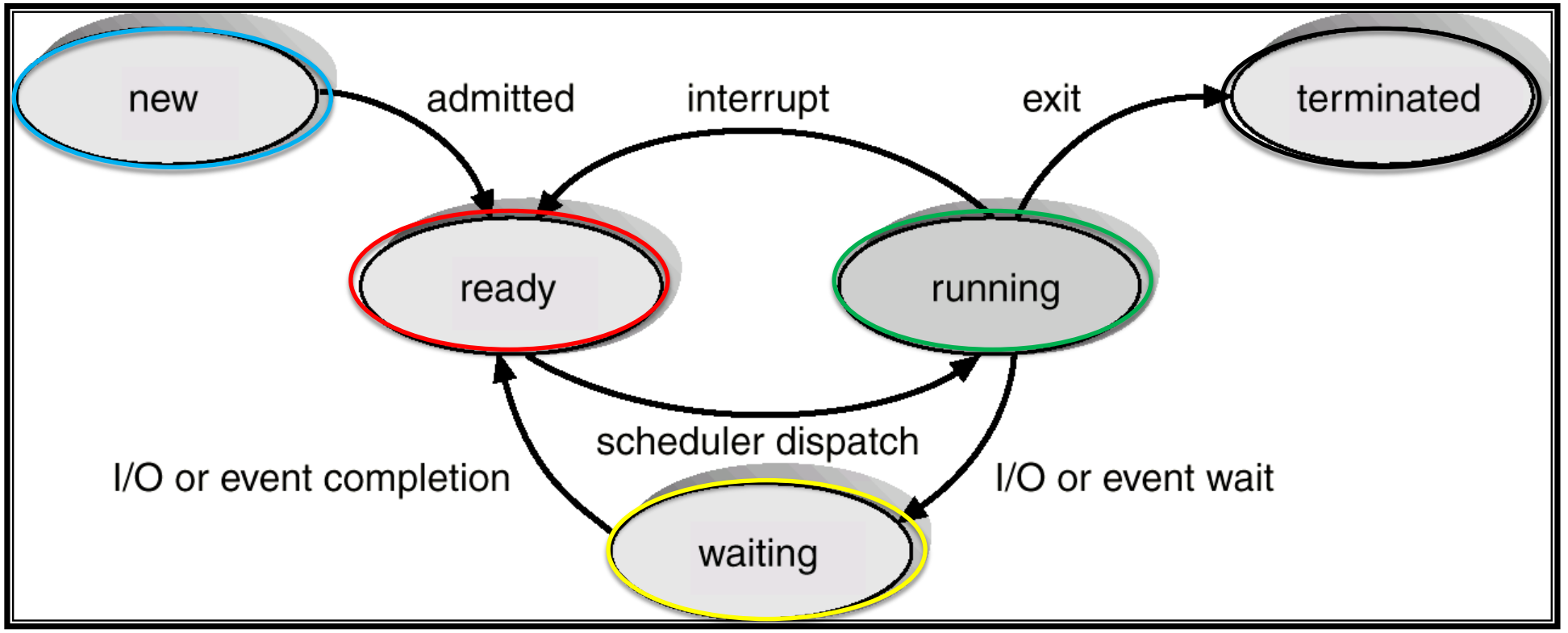
- Estados de um processo
  - Pronto (ready): Aguarda ser selecionado para utilizar o processador
  - Executando (running): Ocupando o processador
  - Bloqueado (blocked): Aguardando um evento (interrupção / ES)



- Escalonador
  - Realiza a seleção de um processo pronto para ser executado

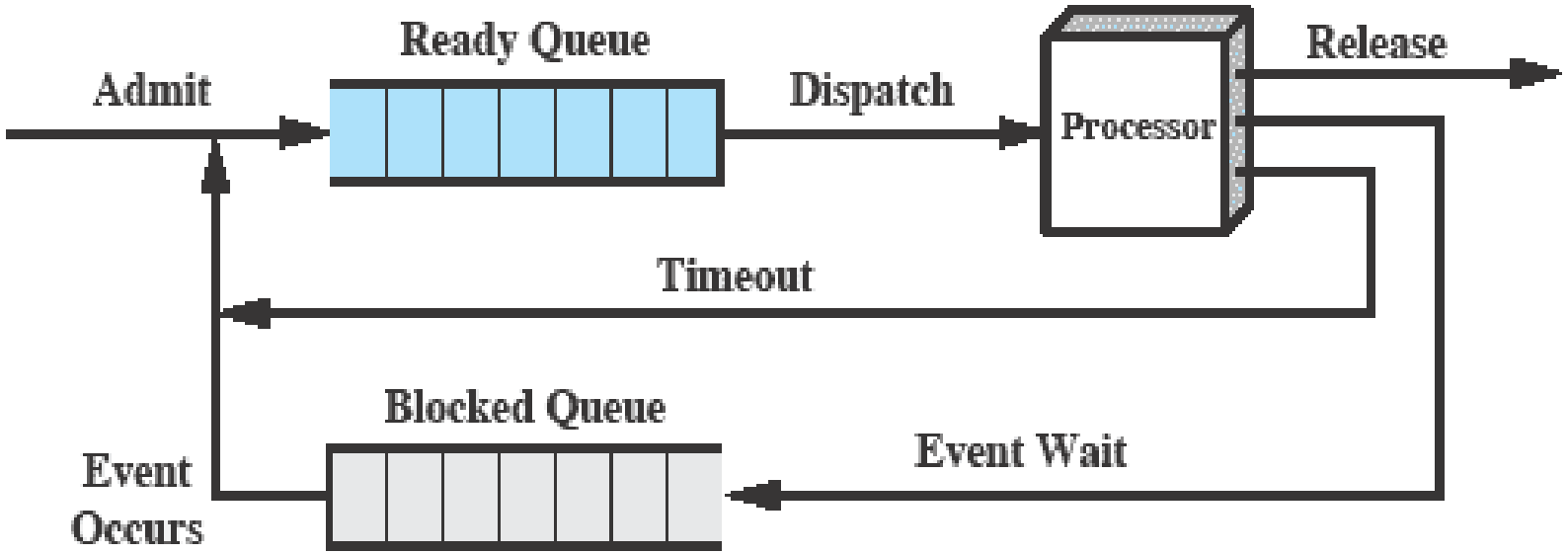
- Necessidade de gerência de filas por periférico
  - Para cada periférico existe uma fila para colocar os processos pendentes

Processos – Estados x eventos

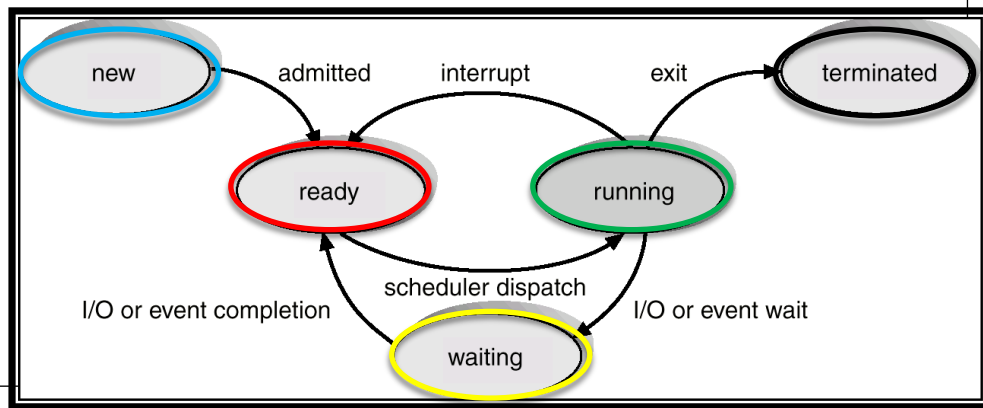


Processos – Estados x eventos

- Implementação com duas filas

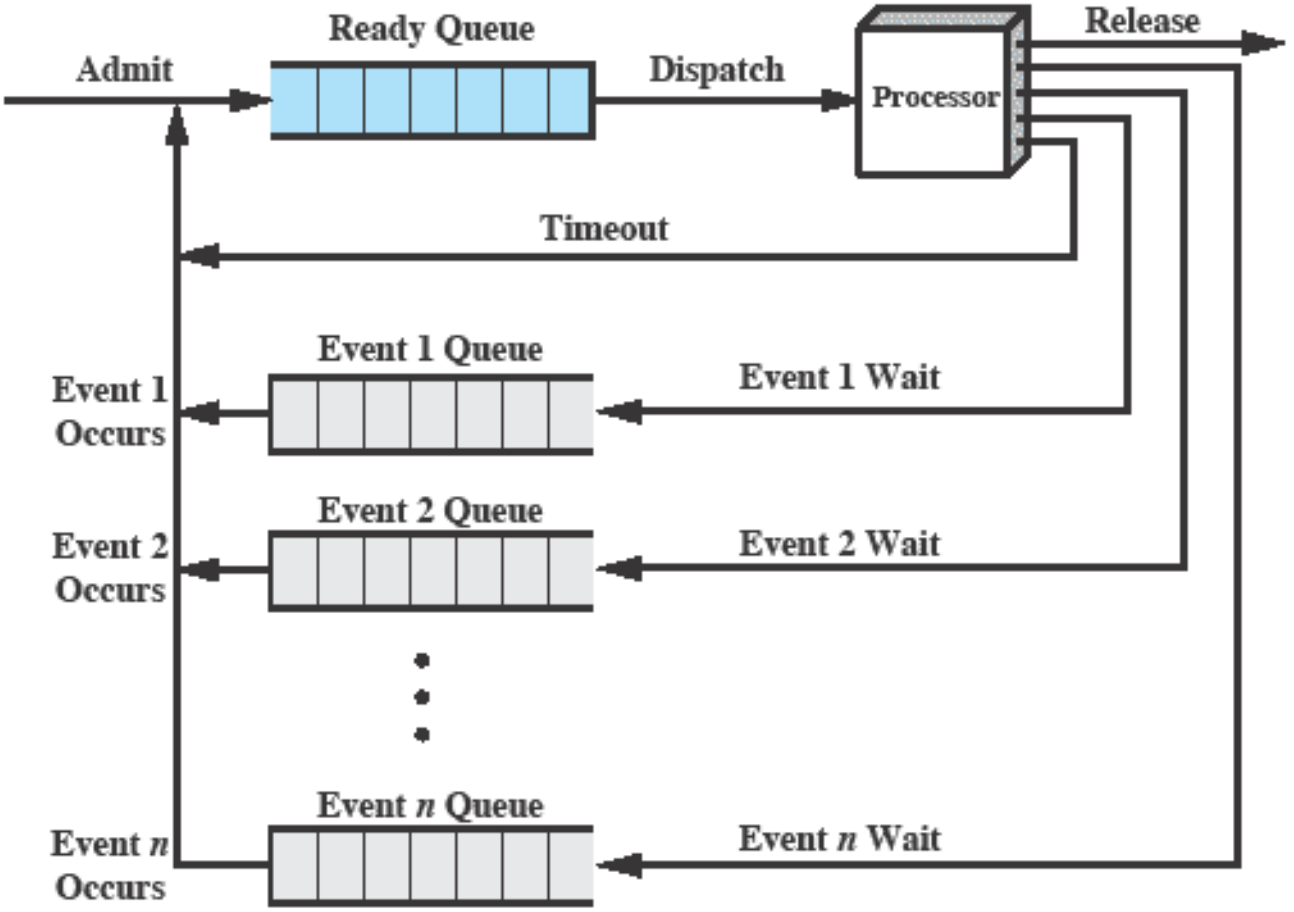


(a) Single blocked queue



Processos – Estados x eventos

– Implementação com n filas de bloqueamento específicas



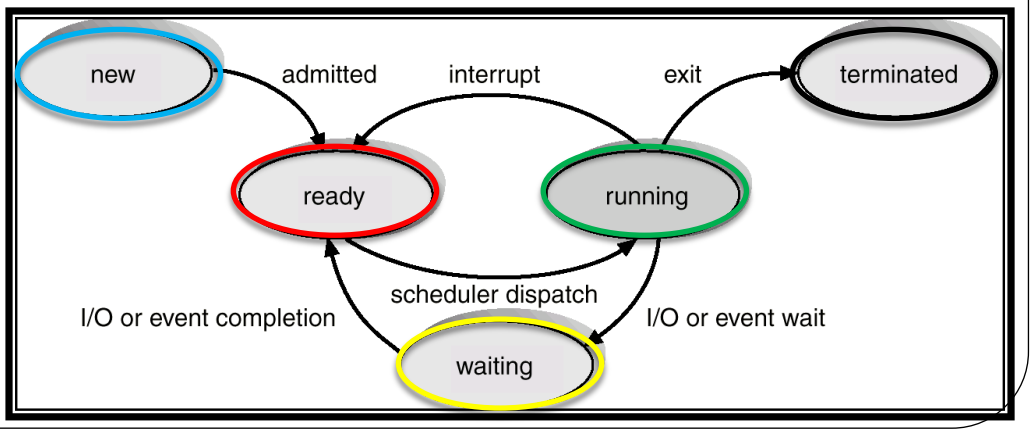
(b) Multiple blocked queues

Filas de escalonamento

- Sistemas monoprocessados
  - Um único processo executando (sendo executado por vez)
    - Recurso sob competição
  - Demais processos aptos a executar (ready) esperam pela CPU
    - **Fila de prontos (ready list)**: contém os processos aptos a rodar
    - **Filas de dispositivos**: contém os processos esperando I/O
    - **Filas de eventos**: contém os processos bloqueados a espera de eventos (ex. passagem de tempo)
- Sistemas multiprocessados
  - Memória comum
    - Uma única fila ready compartilhada pelos processadores
    - Uma fila ready identificada para cada processador
  - Memória distribuída:
    - Uma fila ready em cada processador

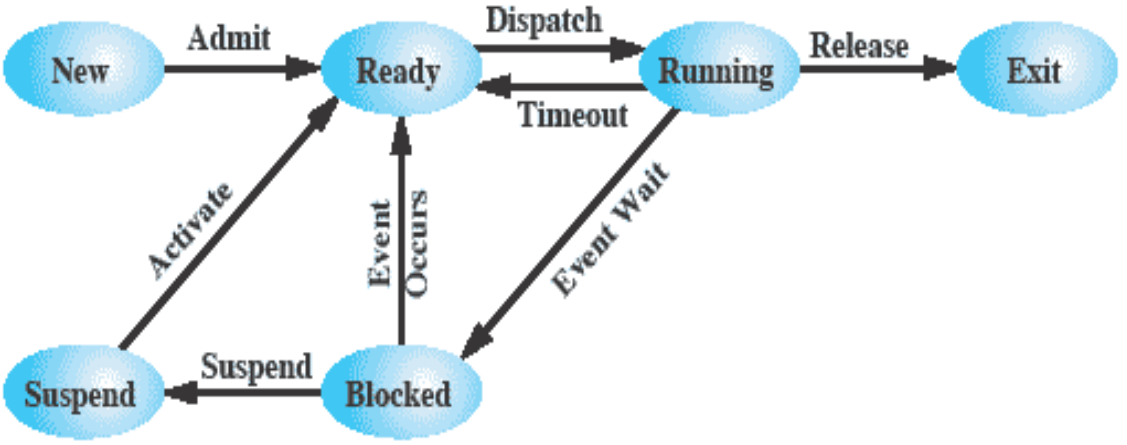
Processos

- Estados de um processo
  - O aumento do número de estados tem a ver com o uso eficiente do processador
    - Processos que requisitam ES ficam bloqueados até serem atendidos
  - Mas o que fazer se o número de processos crescer e sua grande maioria permanecer na fila de bloqueado, deixando o processador ocioso e a memória principal "totalmente" ocupada?
    - Solução está no emprego de swapping e a inclusão de um novo estado (i.e. Suspenso)



Processos – Estados x eventos

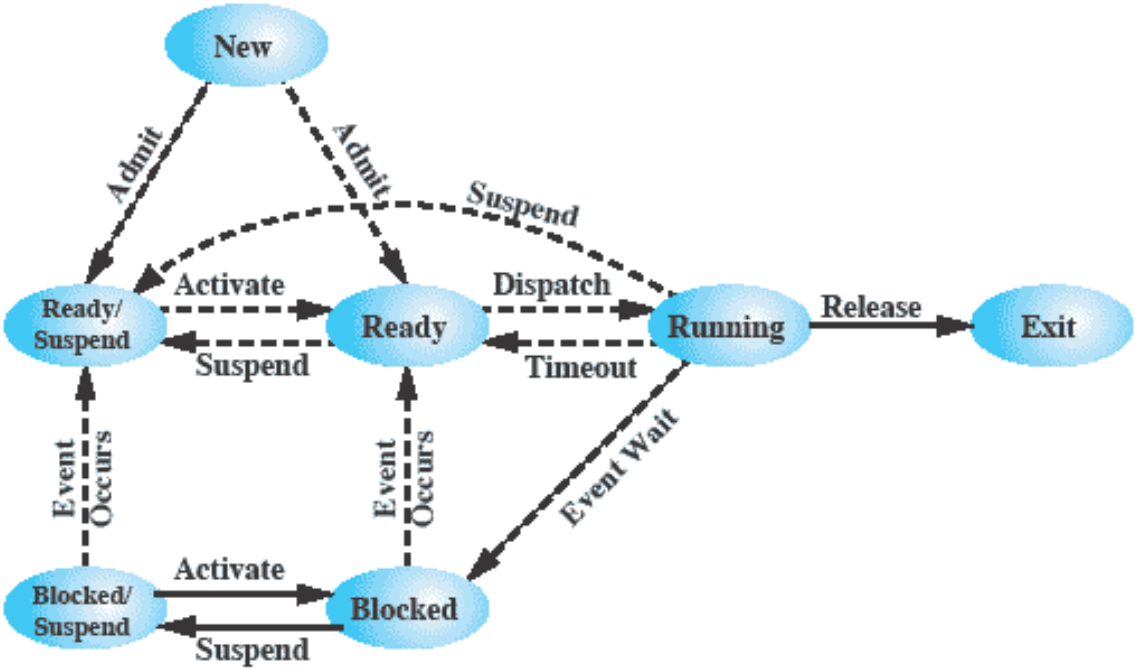
- Modelo de 6 estados
  - Características de um processo suspenso
    - Não está disponível imediatamente para execução
    - Seu estado pode ser estabelecido por um "agente":
      - O próprio processo, o processo pai ou ainda o SO
    - Pode estar aguardando a ocorrência de um evento
    - Sua mudança de estado deve ser compulsoriamente sinalizada pelo agente



(a) With One Suspend State

Processos – Estados x eventos

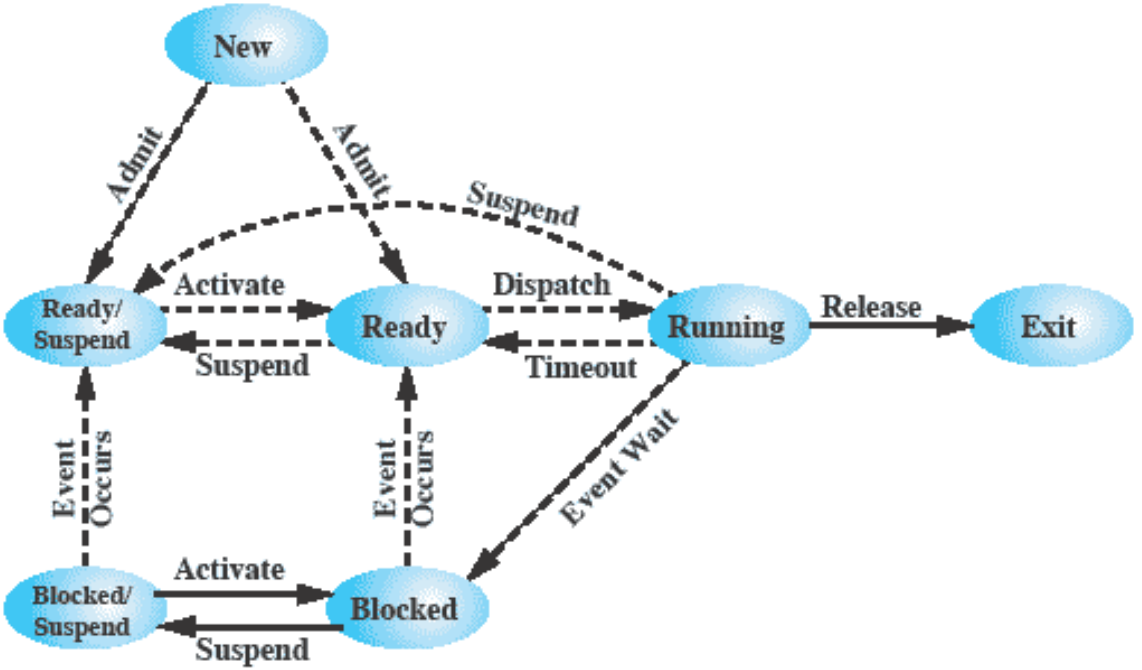
- Modelo de 7 estados
  - Blocked -> Blocked/Suspend: Se não há processo na lista de pronto então ao menos um processo bloqueado deve ser "swapped out" para dar lugar a outro processo não bloqueado



(b) With Two Suspend States

Processos – Estados x eventos

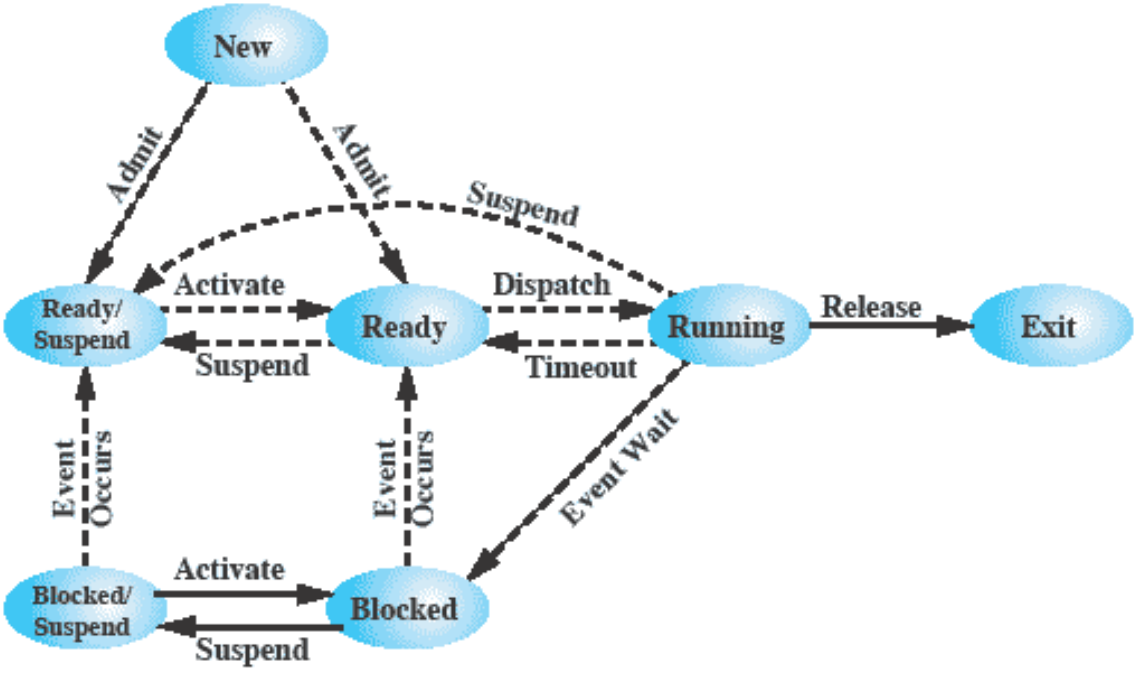
- Modelo de 7 estados
  - Blocked/Suspend → Ready/Suspend: Um processo é movido para o estado Ready/Suspend quando o evento esperado ocorre.



(b) With Two Suspend States

Processos – Estados x eventos

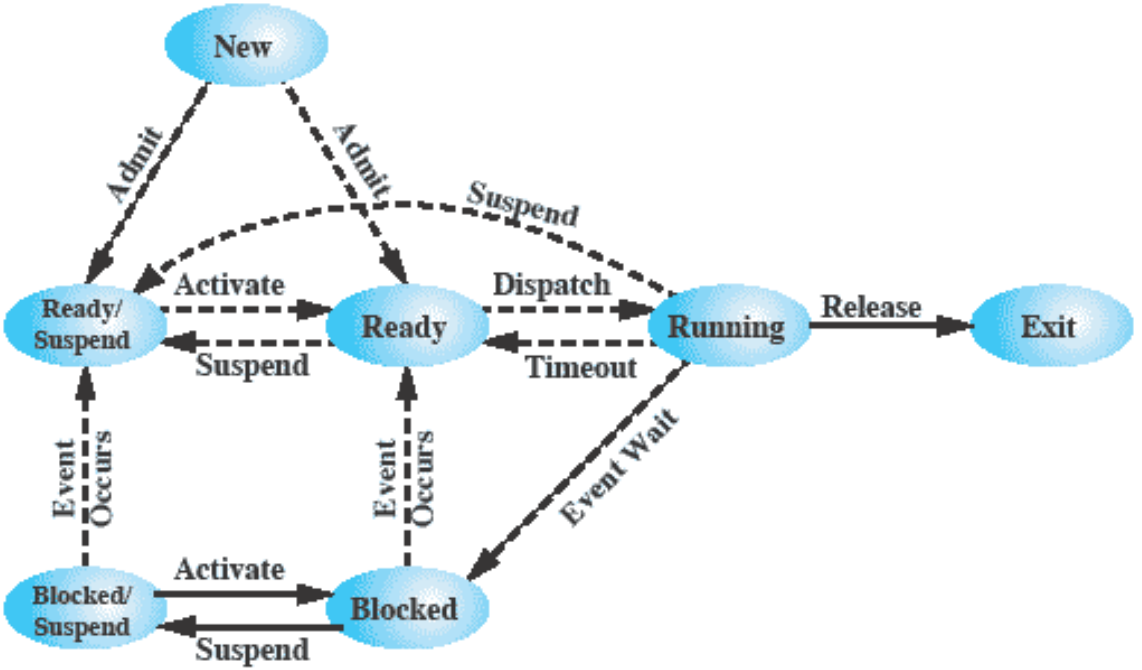
- Modelo de 7 estados
  - Ready/Suspend → Ready: Transição ocorre não há processos na memória principal ou um processo de maior prioridade encontra-se na lista de processos "ready/suspense".
  - O SO terá de buscar um dos processos para retomada da execução



(b) With Two Suspend States

Processos – Estados x eventos

- Modelo de 7 estados
  - Ready → Ready/Suspend: SO normalmente escolhe o processo bloqueado para tal transição. Escolha pode ser feita se este é o único jeito de liberar muito espaço em memória ou ainda garantir espaço para um processo de maior prioridade.



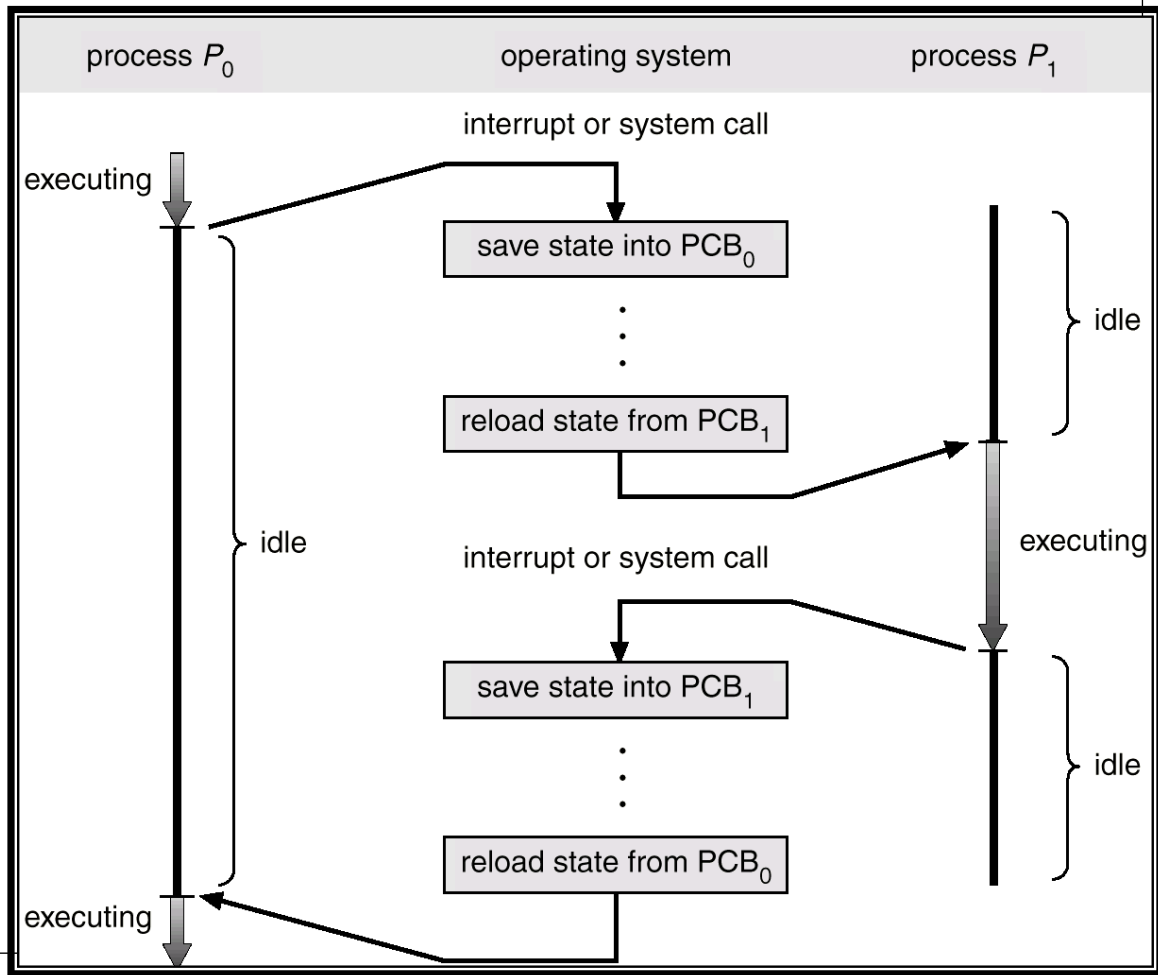
(b) With Two Suspend States

Escalonador

- Mecanismo que implementa uma política de escalonamento
  - Garante o compartilhamento de um recurso central de processamento
- Tipos de escalonador
  - Longo termo (Jobs)
    - Programas submetidos são armazenados em disco
    - Escalonador
      - Seleciona programas do disco para executar
      - Carrega o programan na memória para executação
      - Disponibiliza na fila de pronto
    - Utilizados normalmente em sistemas batch (não interativos)
      - Linux / Windows não empregam tal política
  - Pequeno termo (escalonadores da CPU)
    - Seleciona um processo entre os prontos
    - Submete ao *dispatcher* para ser operado pelo processador
  - DIFERENÇA entre os tipos de escalonador: freqüência de execução

Dispatcher

- Entrega a CPU para o processo selecionado pelo Escalonador de Pequeno Termo (Escalonador da CPU)
  - Carrega os registradores gerais (da pilha, do descritor,..) nos registradores de máquina (restaura contexto)
  - Carrega o registrador PC (Program Counter)
    - dispara a execução



Algoritmos de escalonamento (scheduling)

- Objetivos
  - Minimizar tempo de espera (waiting time)
    - Tempo de espera: qtde de tempo que um processo aguardou na fila de pronto
  - Minimizar tempo de resposta (sistemas interativos)
    - Tempo de resposta: qtde de tempo até o primeiro resultado do processo
  - Minimizar tempo de turnaround (tempo total de execução)
    - Tempo de turnaround: qtde de tempo desde a criação até o término
  - Maximizar o número de programas executados por unidade de tempo
    - Vazão: número de processos operados/fnalizados por intervalo de tempo
  - Distribuir uniformemente o tempo de CPU
    - Aplicar algum critério de justiça

Algoritmo FCFS (First-Come First-Served)

- Característica
  - Primeiro processo que requisita a CPU é o primeiro que a recebe
    - processo pronto para executar entra no final da lista de prontos
    - quando a CPU é liberada, ela é alocada para o primeiro da lista

- Problemas
  - Inadequado para sistemas interativos
  - Processos CPU bound -> monopolização da CPU

Algoritmo FCFS (First-Come First-Served)

- Chegada dos processos
  - P1: 12
  - P2: 2
  - P3: 4
- Tempos de espera na fila (Waiting Time)
  - P1: 0 (primeiro a chegar)
  - P2: 12
  - P3: (12+2) = 14
- Turnaround time
  - P1: 12
  - P2: 14
  - P3: 18
- Média do Turnaround time

  (12+14+18)/3 = 14,67

Algoritmo FCFS (First-Come First-Served)

- Exercício 1
  - Calcule waiting time, turnaround time e média para os seguintes casos:
    - P1:4 P2:7 P3:2 P4:10
    - Ordem de chegada na fila dada pela identificação do processo

- Exercício 2

P	***Proc. Time***	***Arrival Time***
P1	10	0
P2	5	5
P3	8	10
P4	3	12
P5	10	16

Algoritmo SJF (Shortest Job First)

- Característica
  - Sempre executa o processo da fila que demanda o menor tempo
  - Cada processo é relacionado ao seu tempo de execução
  - Quanto menor o tempo de execução, maior a prioridade dada pelo escalonador para execução

- Problema
  - Determinar o tempo de execução de um processo antes de executá-lo (pode ser feito através de análise histórica de execução do processo)

Algoritmo SJF (Shortest Job First)

- Processos
  - P1: 12 -> 3o.
  - P2: 2 -> 1o.
  - P3: 4 -> 2o.
- Tempos de espera na fila
  - P2: 0 (menor tempo de execução)
  - P3: 2 (segundo menor tempo de execução)
  - P1: (4+2) = 6
- Turnaround time
  - P1: 18
  - P2: 2
  - P3: 6
- Média (somatório dos turnaround times/nro de procs)

  (18+2+6)/3 = 8,67

Algoritmo SJF (Shortest Job First)

- Exercício 3
  - Calcule waiting time, turnaround time e média para os seguintes casos:
    - P1:4 P2:7 P3:2 P4:10

- Exercício 4

P	***Proc. Time***	***Arrival Time***
P1	10	0
P2	5	5
P3	8	10
P4	3	12
P5	10	16

Algoritmos baseados em prioridades

- **Características**
  - Cada processo possui uma prioridade
  - Processos com maior prioridade são executados antes de processos com menor prioridade
- **Problema**
  - Postergação indefinida: um processo pode nunca ser executado
  - **Solução**: incrementar periodicamente a prioridade dos processos ***(aging)***
- Algoritmos preemptivos vs não preemptivos
  - **preemptivos**: se o processo submetido tiver prioridade maior que a do processo em execução, então o processo em execução perde a CPU para o processo com maior prioridade
  - **não preemptivos**: se o processo submetido tiver prioridade maior que a do processo em execução, então o processo com maior prioridade entra na ready-list e espera acabar a utilização da CPU pelo processo em execução

Algoritmos baseados em prioridades

- Exercício 5
  - SJF **com preempção**
    - Prioridade é para processo com menor tempo de processamento
    - Calcular:
      - Waiting time
      - Turnaround time
      - Média de turnaround time dos processos

P	**PT**	**AT**
P1	10	0
P2	3	5
P3	10	7
P4	5	17

Algoritmo Round-robin

- Característica
  - Lista de processos é implementada como uma fila circular
  - **quantum**: tempo máximo no qual um processo pode utilizar a CPU ininterruptamente -> evita monopólio da CPU por um único processo
  - Funcionamento:
    1. pega o 1o. processo da ready list e o executa
    2. processo executa por uma fatia de tempo, é preemptado e reinserido no final da ready list
    3. volta para o 1o. passo
- Problemas
  - overhead para salvamento e restauração de contexto
  - Definição do quantum:
    - **muito grande:** se aproxima do FCFS
    - **muito pequeno**: aumenta overhead de troca de contexto (valores comuns: 10-100 miliseg)

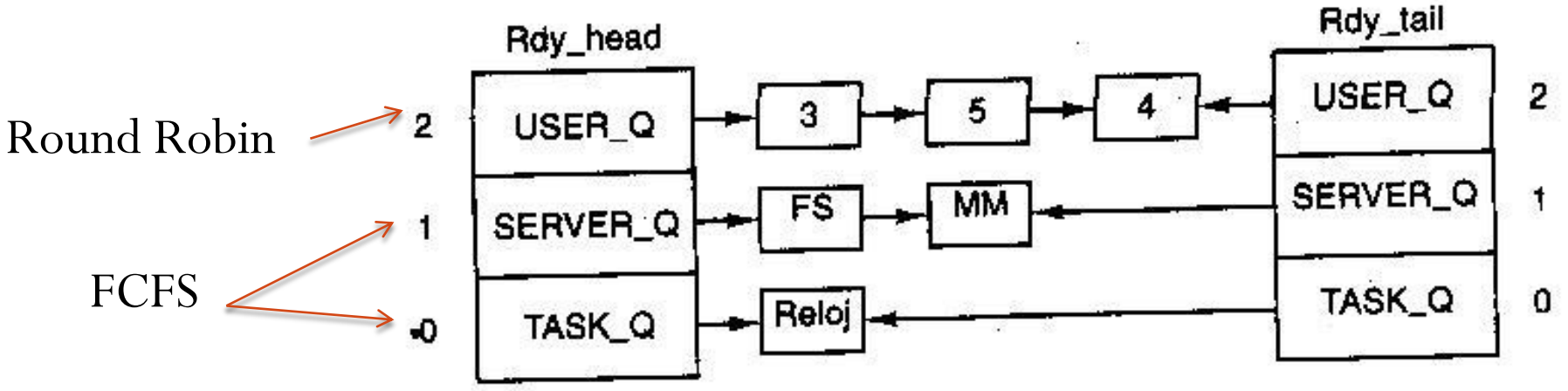
Algoritmo Round-robin

- Exercício 6:
  - Quantum = 5
  - Calcular:
    - Waiting time,
    - Turnaround time
    - Média de turnaround time

P	**PT**	**AT**
P1	10	0
P2	5	0
P3	8	4
P4	3	10

Algoritmos com múltiplas filas

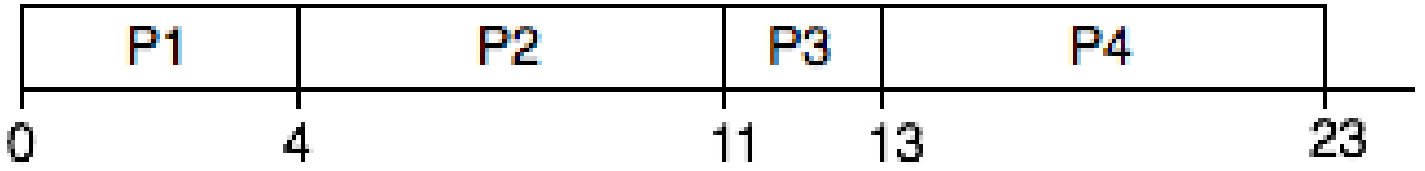
- Características
  - Processos são classificados em grupos
  - Existe uma fila para cada grupo
  - Cada fila pode ter seu próprio algoritmo
  - Pode existir relação de prioridade entre filas
  - Para evitar postergação indefinida
    - Cada fila executa durante uma determinada fatia de tempo, ou os processos se movimentam pelas filas



Respostas - exercícios

Algoritmo FCFS

- Exercício 1
  - P1:4 P2:7 P3:2 P4:10



P1:4
P2:7
P3:2
P4:10

P2:7
P3:2
P4:10

P3:2
P4:10

P4:10

	PT	WT	TT
P1	4	0	4
P2	7	4	11
P3	2	11	13
P4	10	13	23

Total=51

Média=12,75

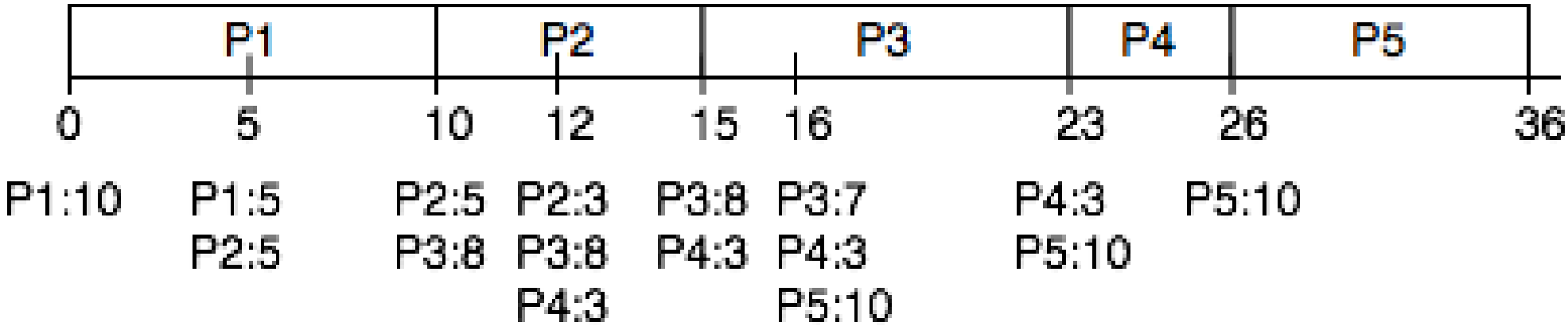
Algoritmo FCFS

- Exercício 2

P	AT	PT	WT	TT
P1	0	10	0	10
P2	5	5	5	10
P3	10	8	5	13
P4	12	3	11	14
P5	16	10	10	20

Total=67

Média=13,4



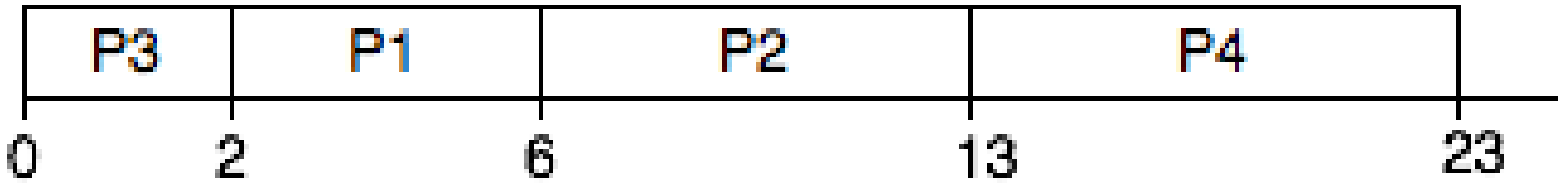
Algoritmo SJF

- Exercício 3

	PT	WT	TT
P1	4	2	6
P2	7	6	13
P3	2	0	2
P4	10	13	23

Total=44

Média=11



P1:4 P2:7 P3:2 P4:10

P1:4 P2:7 P4:10

P2:7 P4:10

P4:10

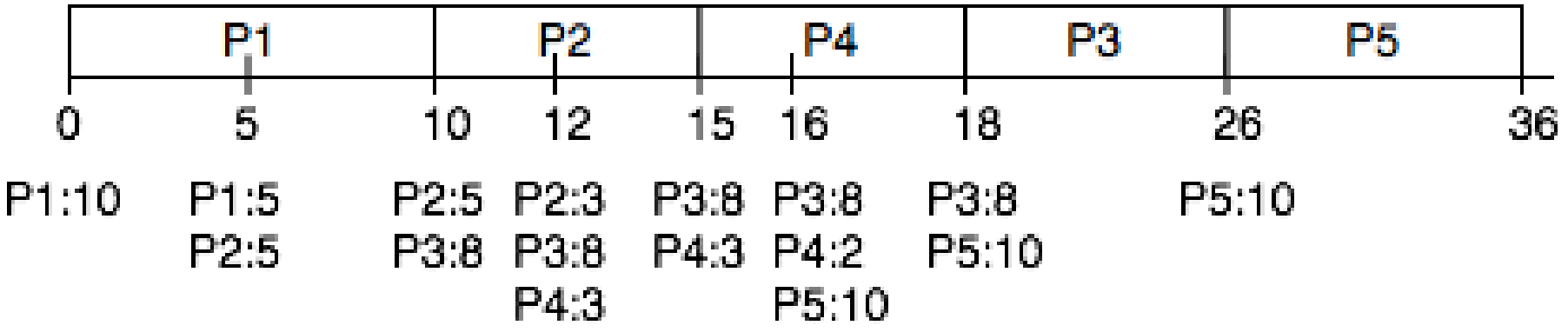
Algoritmo SJF

- Exercício 4

P	AT	PT	WT	TT
P1	0	10	0	10
P2	5	5	5	10
P3	10	8	8	16
P4	12	3	3	6
P5	16	10	10	20

Total=62

Média=12,4



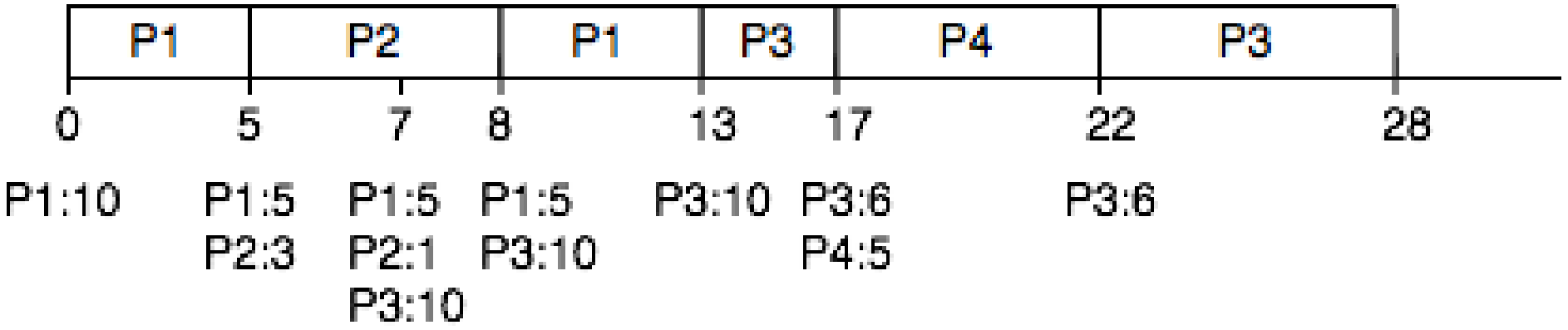
Algoritmo SJF com preempção

- Exercício 5

P	AT	PT	WT	TT
P1	0	10	3	13
P2	5	3	0	3
P3	7	10	6+5	21
P4	17	5	0	5

Total=42

Média=10,5



Algoritmo Round-Robin

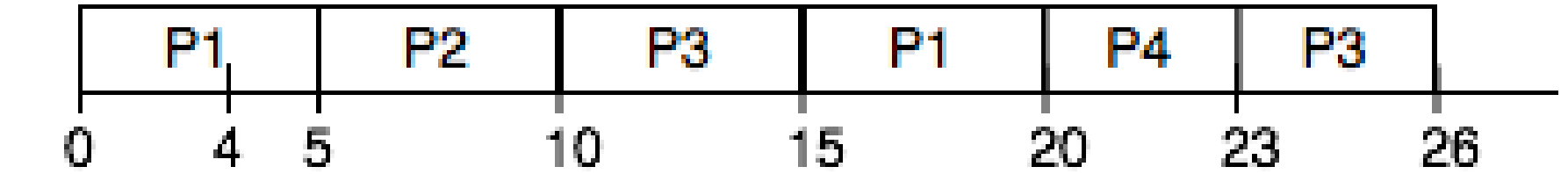
- Exercício 6

Q=5

P	AT	PT	WT	TT
P1	0	10	10	20
P2	0	5	5	10
P3	4	8	6+8	22
P4	10	3	10	13

Total=65

Média=16,25



P1:10 P1:6 P2:5 P3:8 P1:5 P4:3 P3:3
P2:5 P2:5 P3:8 P1:5 P4:3 P3:3
P3:8 P1:5 P4:3 P3:3

Exercícios

- Calcule waiting time, turnaround time e média de turnaround time utilizando os algoritmos FCFS, SJF, SJF com preempção e round-robin (quantum=4) para os seguintes casos

a)

P	**PT**	**AT**
P1	7	0
P2	3	2
P3	5	8

b)

P	**PT**	**AT**
P1	12	5
P2	8	10
P3	5	14
P4	8	14

c)

P	**PT**	**AT**
P1	5	0
P2	10	7
P3	7	10
P4	4	12
P5	8	12
P6	5	20
P7	5	27

Bibliografia

- Silberschatz, G. "***Operating System Concepts***". Capítulos 4 e 5

- Tanenbaum, A. "***Sistemas Operacionais: projeto e implementação***". Capítulo 2