Sistemas Operacionais

Módulo 02 - Processos

Prof. Daniel Sundfeld
daniel.sundfeld@unb.br

Modelo de Processo

- Todos os computadores modernos são capazes de fazer múltiplas tarefas ao mesmo tempo
- O conceito de processo é um dos mais importantes para qualquer Sistema Operacional
- Um processo é um programa em execução acompanhando de valores de tempo de execução

Modelo de Processo

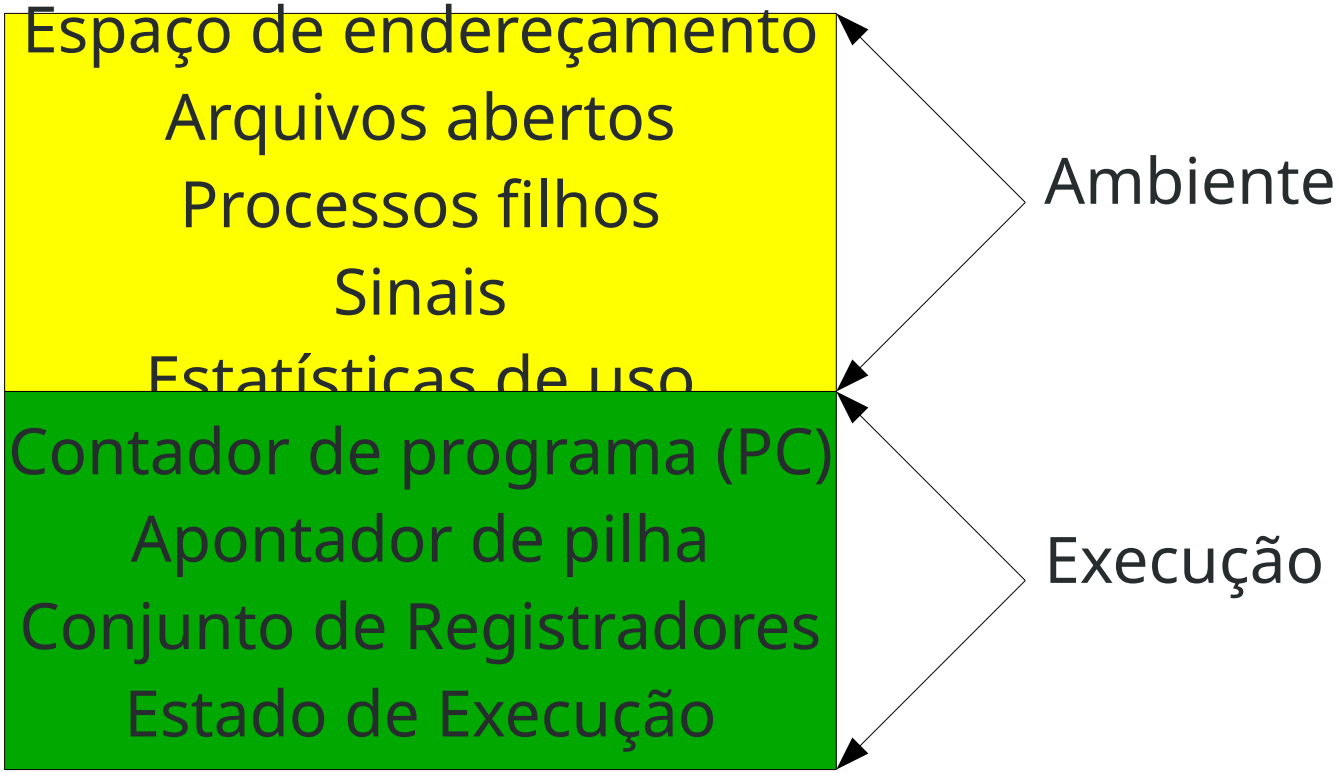
- Processo:
  - Código executável
  - Pilha de execução: contendo valores de variáveis locais
  - Apontador para pilha: registrador da CPU que armazena em qual área de memória está a pilha
  - Contador de programa: registrador da CPU que armazena a próxima instrução a ser executada
  - Valores dos registradores gerais da máquina

Modelo de Processo

- Diferença entre um processo e um programa:
  - Programa: receita de bolo (passivo)
  - Processo: ato de fazer o bolo (ativo)
  - Cozinheiro: CPU
  - Ingredientes do bolo: dados de entrada
  - Bolo: dados de saída
- Processos envolvem uma atividade
- Programas envolvem algoritmos

Modelo de Processo

- Informações relevantes a respeito de um processo:



Modelo de Processo

- Machado e Maia, dividem o processo em 3 partes: Espaço de Endereçamento, Contexto de Software e Contexto de Hardware
- Execução: Contexto de Hardware
- Ambiente: Espaço de Endereçamento + Contexto de Software

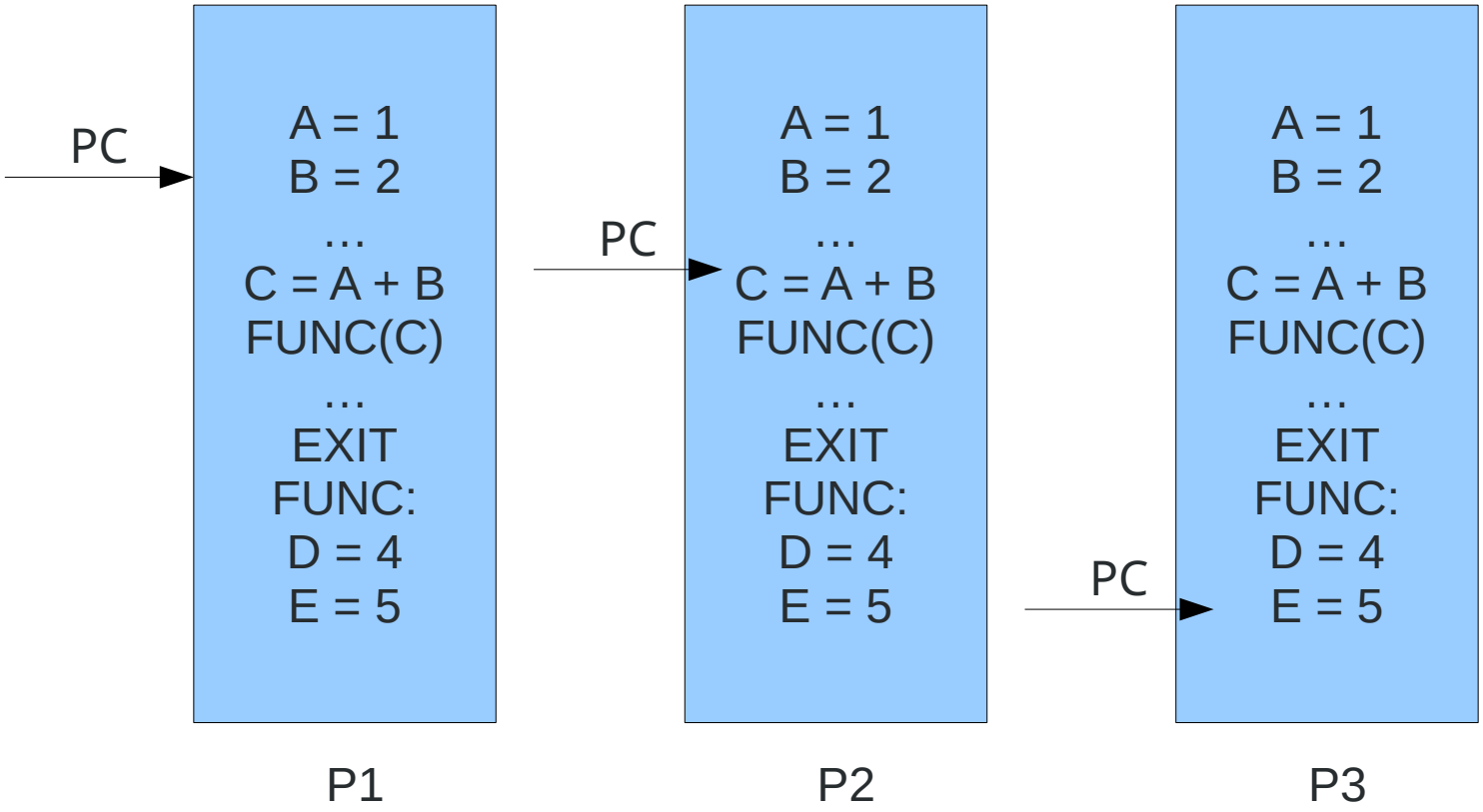
Modelo de Processo

- Classificação dos modelos de processos quanto ao custo de troca de contexto e de manutenção
  - Heavyweight (processo tradicional)
  - Lightweight (threads)

Modelo de Processo Tradicional

- Heavyweight
- O processo é composto tanto pelo ambiente como pela execução
- Cada processo possui um único fluxo de controle (contador de programa) e roda de forma independente dos demais
- Em um determinado instante, há vários processos ativos ao mesmo tempo, o processador é chaveado entre diversos processos

Modelo de Processo Tradicional



Criação de Processos

- Sistemas de um único propósito possuem um esquema mais simples e previsível de processos em execução
- Contudo, sistemas de propósito geral, que englobam os computadores pessoais atuais, não são previsíveis pois os computadores possuem diferentes programas instalados devido:
  - Hardware diferente
  - Necessidade dos usuários

Criação de Processos

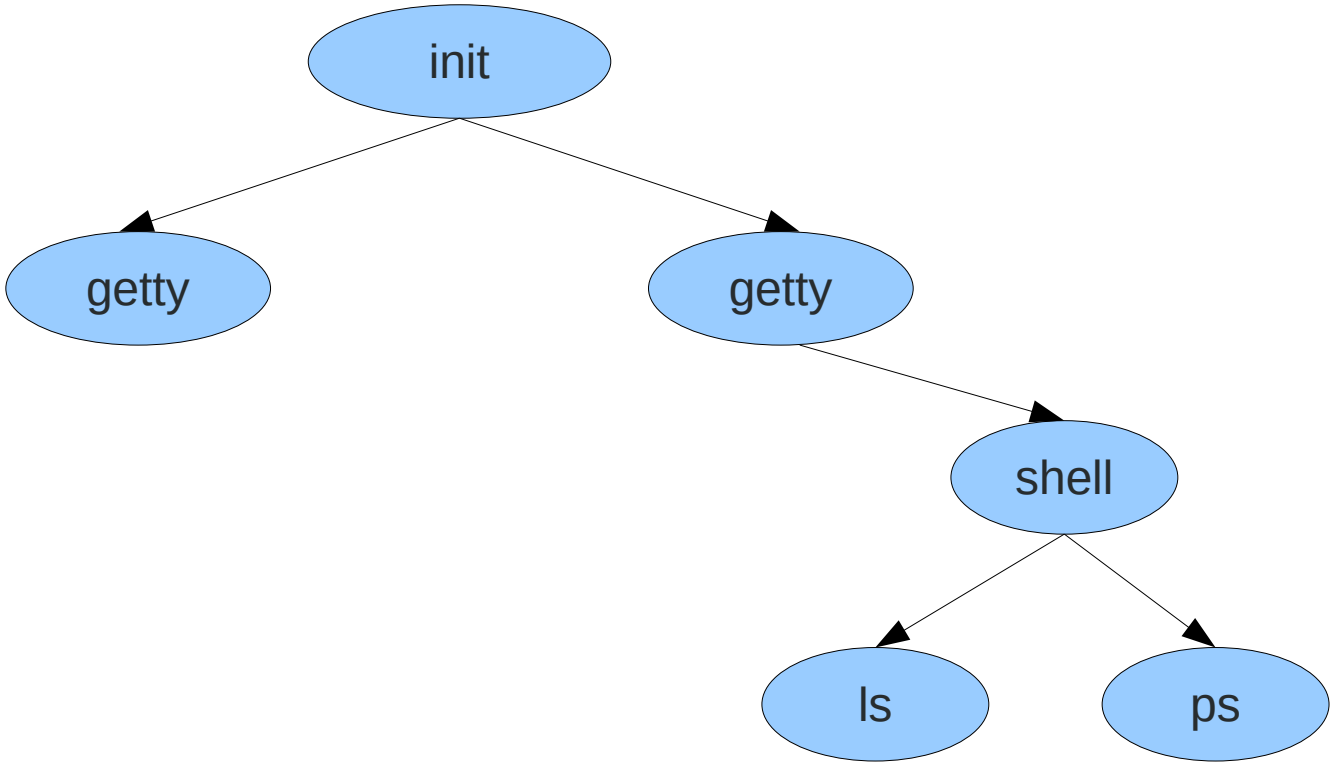
- Desta forma são necessários mecanismos para permitir a criação de processos.
- Os processos são criados:
  - Início do sistema
  - Execução de uma chamada ao sistema de criação de processo por um processo em execução
  - Requisição do usuário
  - Início de um job de lote

Criação de Processos

- Quando um sistema operacional é carregado, são criados diversos processos
- Alguns executam em primeiro plano e interagem com usuários
- Outros executam em segundo plano: chamados de **daemons**
- Normalmente os processos são criados por outro

Criação de Processos

- Hierarquia em árvore



Criação de Processos

- No Unix há apenas uma forma de criar processos, através de clonagem:
  - fork();
    - Clone idêntico ao processo que chamou
    - Normalmente, executa execve depois da chamada para mudar o "programa" em execução
- No Windows, há uma função que trata o processo de criação quanto da carga do programa:
  - CreateProcess

Término de Processos

- Condições de término de processos:
  - Saída normal (voluntária)
  - Saída por erro (voluntária)
  - Erro fatal (involuntário)
  - Cancelamento por outro processo (involuntário)

Hierarquia de Processos

- Quando um processo é criado por outro, eles continuam associados
- Em alguns sistemas, é possível listar o ppid (parent pid)
- Todos os processos são filhos ou descendentes do processo init (atualmente, há uma alternativa: systemd)
- O Windows não apresenta hierarquia de processos

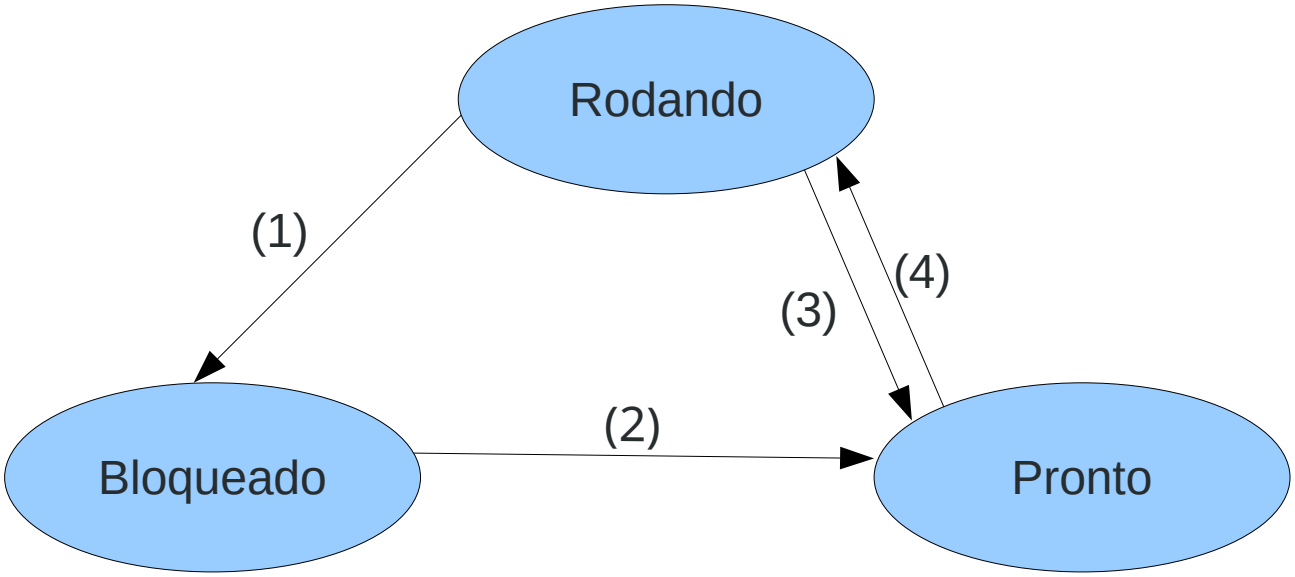
Estados de Processos

- Apesar de processos serem autossuficientes, muitas vezes necessitam acessar recursos ou comunicar com um processo
- Quando um processo está esperando um evento, dizemos que está bloqueado:
  - Leitura em disco
  - Leitura de rede
  - Esperando entrada do usuário

Estados de Processos

- Os processos podem estar:
  - Rodando
  - Bloqueado
  - Pronto
- Sistemas monoprocessados são aqueles que possuem apenas um único processo rodando

Estados de Processos



(1) O processo bloqueia-se aguardando uma entrada
(2) O evento aguardado pelo processo ocorreu, pode-se iniciar a executar.
(3) O tempo de posse do processador esgotou-se
(4) O processo é escolhido pelo escalonador para executar

CPU Bound e I/O Bound

- Processos podem ser classificados como CPU-bound ou I/O-bound
- Processos CPU-bound (afinidade à CPU) passam a maior parte do tempo usando a CPU, no estado rodando ou no estado pronto
- Processos I/O-bound (afinidade à Entrada e Saída) passam a maior parte do tempo em bloqueado por causarem muitas operações de entrada e saída

CPU Bound e I/O Bound

- Muitas aplicações de computadores pessoais que rodam em primeiro plano são exemplos de aplicações I/O Bound
- Essas aplicações passam muito tempo em estado bloqueado, aguardando as inforamações dos usuários (digitar um dado de entrada, por exemplo.)

Implementação de Processos

- Todas as informações sobre um processo são mantidas na tabela de processos
- Visualizar a tabela: “ps aux/CTRL+Alt+Del”
- Contém campos que dizem respeito à gerência do processo, à gerencia da memória e arquivos
- Uma entrada por processo, variam entre os diferentes sistemas operacionais

Tabela de Processos

- Cada processo tem um identificador único conhecido como pid (process id)
- As informações sobre um processo estão na **tabela de processos** (também chamada de **bloco de controle de processo**)
- Durante a execução, o processo compartilha o processador com outros processos em execução

Implementação de Processos: Tabela de Processos

Gerenciamento de processos	Gerenciamento de memória	Gerenciamento de arquivos
• Registradores Contador de programa • Palava de estado do programa • Ponteiro de pilha • Estado do processo • Prioridade • Parâmetros de escalonamento • Identificador do processo (PID) • Processo pai (PPID) • Grupo do processo • Sinais • Momento em que iniciou • Tempo de uso de CPU • Tempo de CPU do filho	• Ponteiro para o segmento de código • Ponteiro para o segmento de dados • Ponteiro para o segmento de pilha	• Diretório-raiz • Diretório de trabalho • Descritores de arquivos • Identificador do usuário • Identificador do grupo

Implementação de Processos

- **Troca de contexto:** operação de salvamento dos registradores de um processo e posterior restauração de registradores de outro processo.
- A troca de contexto permite a troca de processador entre os processos
- É a operação básica da multiprogramação

Implementação de Processos

- Como manter a ilusão de múltiplos processos sequenciais em uma máquina com uma CPU e muitos dispositivos de E/S
- Cada dispositivo de E/S possui possui uma área de memória chamada **vetor de interrupções**
- Esse vetor contém os endereços de procedimentos dos serviços de interrupção

Implementação de Processos

- O processo P1 solicita a operação
- O SO altera o estado de P1 para bloqueado
- O SO escolhe um dos processos prontos (P2) e o coloca para executar
- A solicitação de P1 é atendida pelo hardware
- O hardware interrompe P2, salvando seu estado de execução na pilha

Implementação de Processos

- O hardware acesso um endereço de memória física específico que contém o vetor de interrupções
- O vetor de interrupções contém funções de tratamento geradas por cada tipo de dispositivo

Implementação de Processos

- A rotina de tratamento de interrupção é executada pelo SO
- Os registradores que foram empilhados pelo hardware são salvos na tabela de processo do P2
- A interrupção é tratada
- O processo P1 é colocado na fila de prontos
- O SO acessa a entrada da tabela de processos escolhido e carrega o conteúdo da tabela nos registradores de máquina (restauração)
- O processo escolhido reinicia a execução

Implementação de Processos

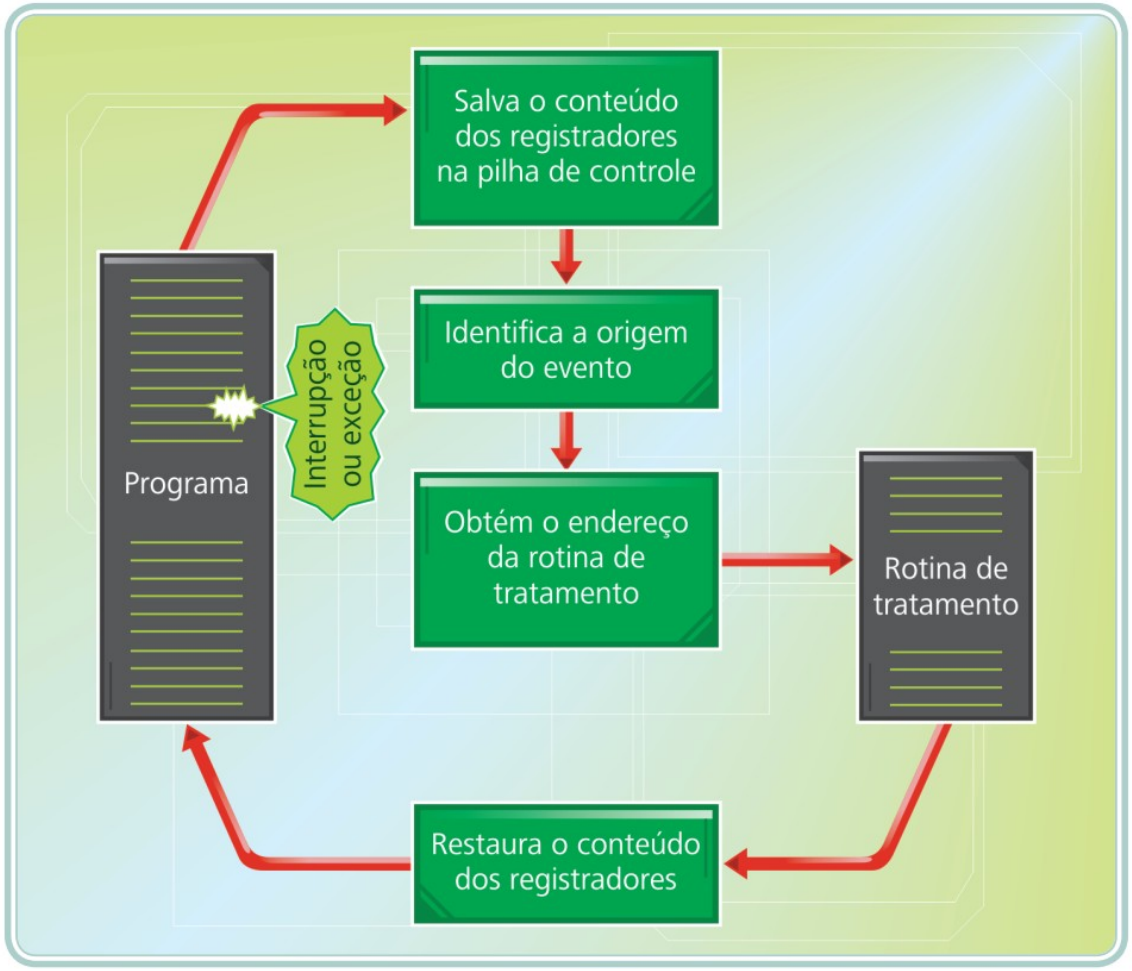


Figura 3.2: Mecanismo de interrupção e exceção

Fonte: Pereira, A. S., Vissotto, E. M., Franciscatto, R. Sistemas Operacionais. E-tec. 2015.

Escalonamento de Processos

- O Sistema Operacional é responsável por gerenciar o(s) recurso(s) de processamento de um computador
- Essa atividade é conhecida como **Escalonamento de Processador**
- A interação entre processos é realizada através de mecanismos de comunicação

Escalonamento de Processos

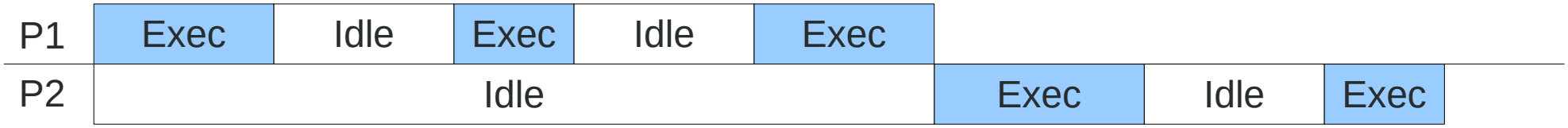
- Quando múltiplos processos encontram-se no estado pronto, é necessário eleger um processo para ser executado
- A parte do SO que faz essa eleição é chamada de **escalonador**
- O algoritmo utilizado chama-se **algoritmo de escalonamento**

Escalonamento de Processos

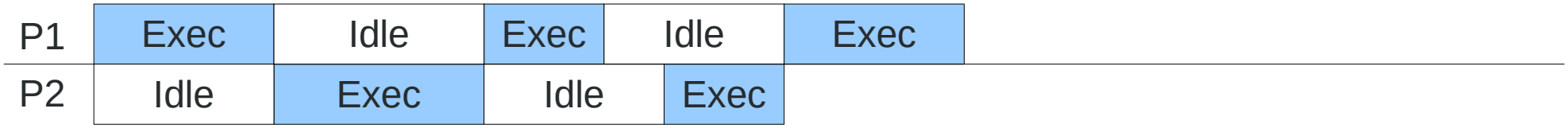
- O algoritmo de escalonamento é o responsável pela determinação de que processo vai rodar e por quanto tempo. Ele define a política de utilização do processador pelos processos
- Quando um processo solicita operações blocantes (E/S), sua execução fica suspensa até que o evento solicitado ocorra

Escalonamento de Processos

- 2 processos sem concorrência



- 2 processos com concorrência



- Executando concorrentemente pode-se obter uma melhor utilização da CPU, mesmo com apenas uma CPU. Especialmente verdade em I/O Bound

Classificação dos Escalonadores

- Os Escalonadores podem ser classificados em **Preemptivos** e **Não-Preemptivos**
- A preempção é a suspensão temporária da execução de um processo

Escalonamento Não-Preemptivo

- Escalonador não-preemptivo: Quando um processo obtém o processador, ele roda até o fim ou até que ele peça uma operação que ocasione o seu bloqueio
- Nenhuma entidade externa “tira a CPU à força” do processo

Escalonamento Preemptivo

- Cada processo possui um tempo (time-slice) de posse do processador
- Quando este tempo se esgota, o SO retira o processador deste processo e permite que outro processo se execute
- O controle de tempo de execução é feito por interrupção

Escalonamento Preemptivo

- Os processadores modernos possuem um clock que gera interrupções a uma frequência determinada
- O SO mantém um contador que é decrementado a cada clock tick
- Se o contador chegar a 0, o tempo de permanência do processo acabou
- O valor inicial deste contador corresponde ao tempo máximo de permanência do processo com a CPU

Escalonamento

- Os escalonadores não-preemptivos são de projeto simples, porém permite o abuso no tempo de CPU de um determinado programa
- Neste caso, um processo pode obter o monopólio do processador, impedindo outros processos de rodarem
- Isso viola vários critérios de um bom escalonador (justiça, tempo mínimo de resposta, etc)

Escalonamento

- Os escalonadores preemptivos asseguram um uso mais balanceado da CPU e são utilizados na maioria dos SO modernos
- Porém, o projeto de tais escalonadores, além de ser complexo introduz complicações na programação de processos concorrentes
- Os processos podem ser interrompidos em um tempo arbitrário, eles devem proteger suas estruturas de dados contra a interferência de outros processos (regiões críticas)

Critérios de Escalonamento

- Ao se projetar um escalonador, devemos observar vários critérios que devem estar presentes em um bom algoritmo de escalonamento:
- **Justiça**: garantir que todos os processos terão chances **justas** de uso de processador. Não são chances iguais
- **Eficiência**: quando existir trabalho a fazer, o processador deve estar ocupado

Critérios de Escalonamento

- **Minimizar o tempo de resposta**: reduzindo o tempo dos usuários interativos, reduz o tempo entre a entrada de usuário e a resposta dada (não considera tempo total de execução)
- **Minimizar o turnaround:** reduzir o tempo desde o lançamento do processo até seu término. Soma de: tempo de espera por recursos (memória, processador, E/S) e tempo de utilização da CPU. Mais utilizado em processamento batch

Critérios de Escalonamento

- **Minimizar waiting time:** Esse critério visa minimizar o tempo de espera pela CPU
- **Maximizar throughtput:** Maximizar o número de tarefas executados em uma unidade de tempo

Critérios de Escalonamento

- Não é possível atingir todos esses critérios de escalonamento, muito são contraditórios entre si
- Minimizar turnaround e minimizar tempo de resposta: para que usuários interativos obtenham um tempo de resposta pequeno, geralmente usuários em batch são penalizados com um tempo de resposta maior
- Um algoritmo que maximiza o throughput não é justo com processos de execução demorada

Algoritmos de Escalonamento

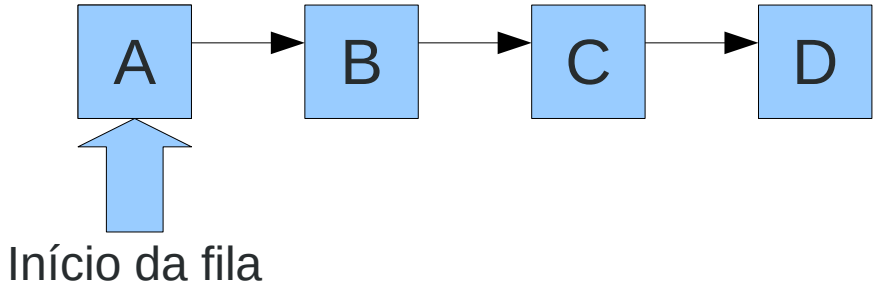
- O problema a ser resolvido pelos algoritmos de escalonamento: dado um conjunto de processos que devem ser executados, como dividir a utilização do processador entre estes processos?
- Estudar algoritmos clássicos de escalonamento:
  - First Come First Served
  - Round-Robin
  - Prioridades
  - Shortest Job First

First Come First Served (FCFS)

- O processo obtém a CPU de acordo com a ordem da chegada das solicitações. O processo que pede a CPU primeiro obtém a CPU em primeiro lugar
- O escalonamento FCFS é não-preemptivo. Assim, um processo CPU/bound pode fazer com que vários processos esperem por um tempo indeterminado

First Come First Served

- Implementação: processos que solicitam a CPU são colocados em uma fila de prontos, gerenciada segunda a política FIFO (First-In First-Out)

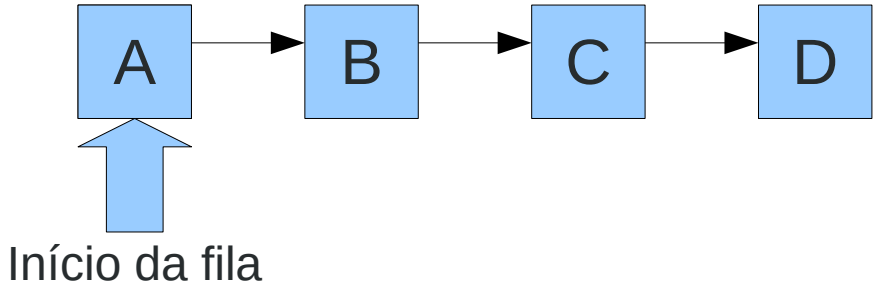


Em execução:

Tempo: 0

First Come First Served

- Primeiro processo da fila é selecionado

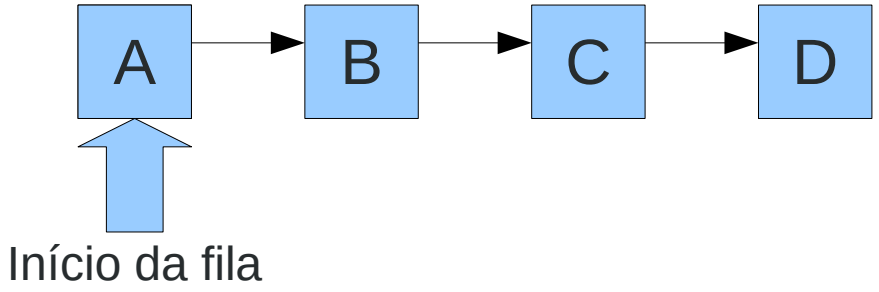


Em execução: A

Tempo: 1

First Come First Served

- O processo continua em execução

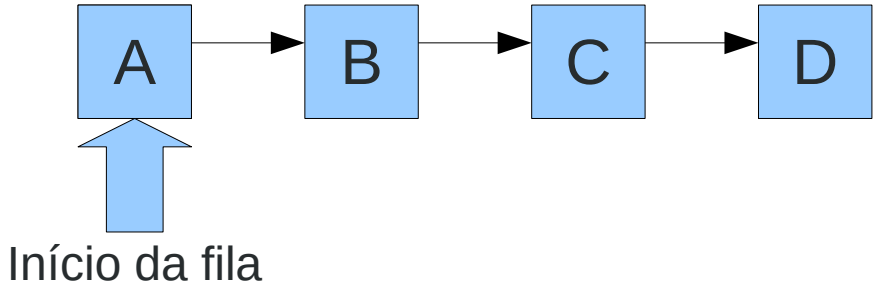


Em execução: A

Tempo: 2

First Come First Served

- O processo continua em execução até causar seu bloqueio



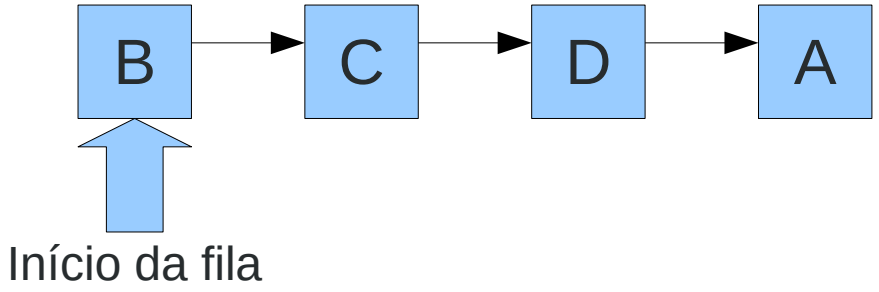
Em execução: A

Processo causou Seu bloqueio

Tempo: 3

First Come First Served

- O próximo processo da fila é selecionado

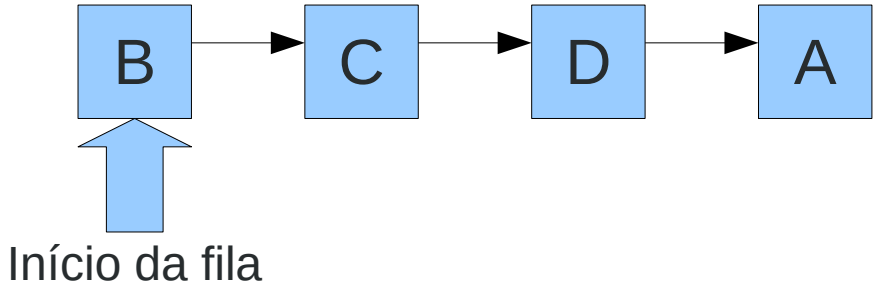


Em execução: B

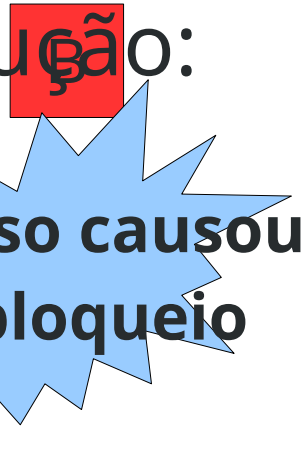
Tempo: 4

First Come First Served

- O processo B causa seu bloqueio



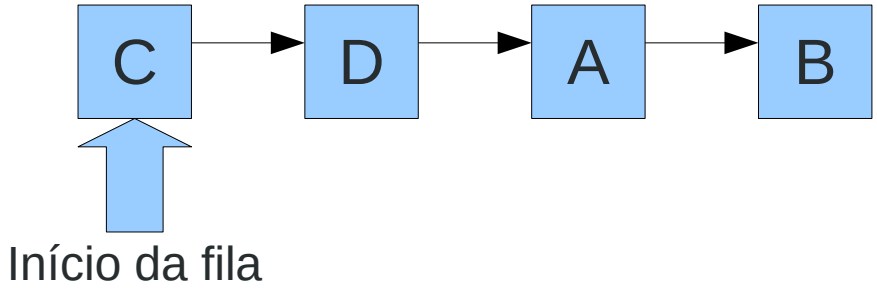
Em execução:



Tempo: 5

First Come First Served

- O próximo processo é selecionado



Em execução:

Tempo: 6

First Come First Served (FCFS)

- Vantagens:
  - Simples de ser implementado
  - Algoritmo **eficiente**: a CPU sempre é utilizada
- Desvantagens:
  - Impossibilidade de se prever quando um processo vai iniciar
  - Tempo média de espera de processos não é priorizado
  - Processos que usam muito a CPU levam vantagens sobre processos que causam muito seu bloqueio

First Come First Served

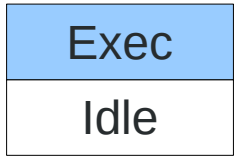
- Problema: dados os seguintes processos, tempo de execução e tempo de chegada. Calcule o tempo de serviço e tempo médio de espera
- Tempo de Espera = Tempo de Serviço – Tempo de Chegada

First Come First Served

- Tempo de Espera = Tempo de Serviço – Tempo de Chegada
- Nenhum dos processo gera chamada blocantes

Processo	Tempo de Chegada	Tempo de Execução	Tempo de Serviço	Tempo de Espera
P0	0	8		
P1	1	2		
P2	2	5		
P3	3	4		

Escalonamento de Processos



Executando
Esperando

Tempo: 0

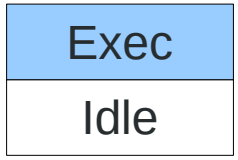
Fila de Processos: P1

Processo em execução:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

P1

Escalonamento de Processos



Executando
Esperando

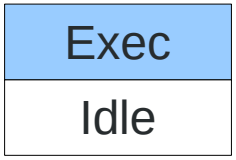
Tempo: 1

Fila de Processos: P2

Processo em execução: P1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P1	Exec																				
P2																					

Escalonamento de Processos

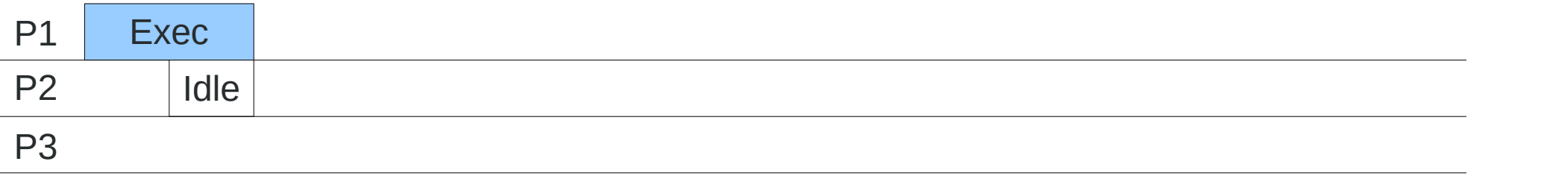


Executando
Esperando

Tempo: 2

Fila de Processos: P2, P3

Processo em execução: P1



Escalonamento de Processos

Executando
Esperando

Tempo: 3

Fila de Processos: P2, P3, P4

Processo em execução: P1

Escalonamento de Processos

Executando
Esperando

Tempo: 4

Fila de Processos: P2, P3, P4

Processo em execução: P1

Escalonamento de Processos

Executando
Esperando

Tempo: 5

Fila de Processos: P2, P3, P4

Processo em execução: P1

Escalonamento de Processos

Executando
Esperando

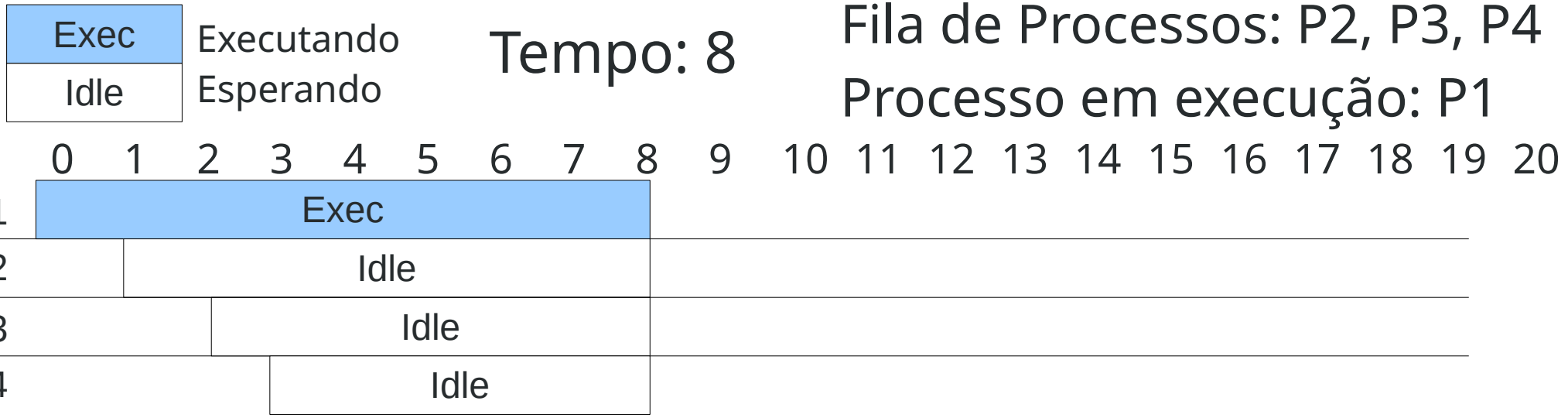
Tempo: 6

Fila de Processos: P2, P3, P4

Processo em execução: P1

Escalonamento de Processos

Escalonamento de Processos



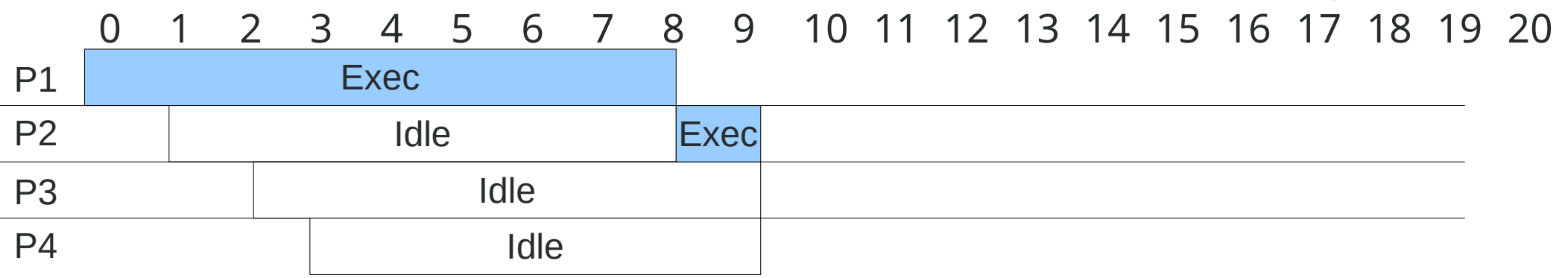
Escalonamento de Processos

Executando
Esperando

Tempo: 9

Fila de Processos: P3, P4

Processo em execução: P2



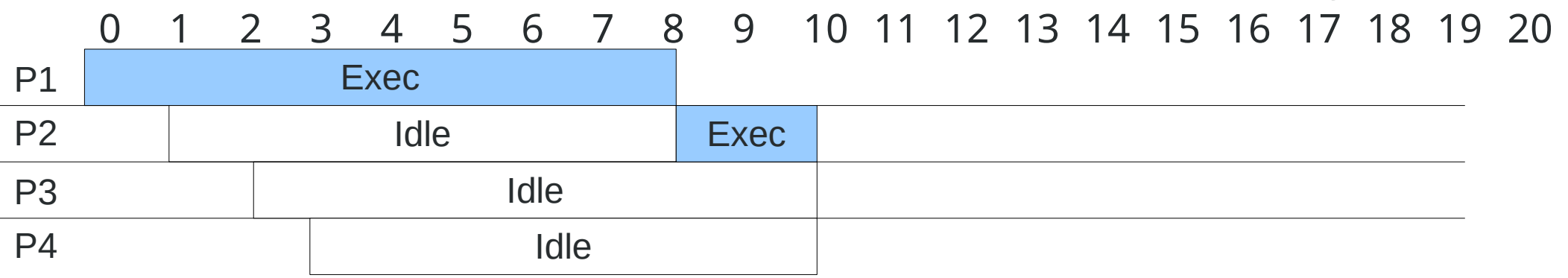
Escalonamento de Processos

Executando
Esperando

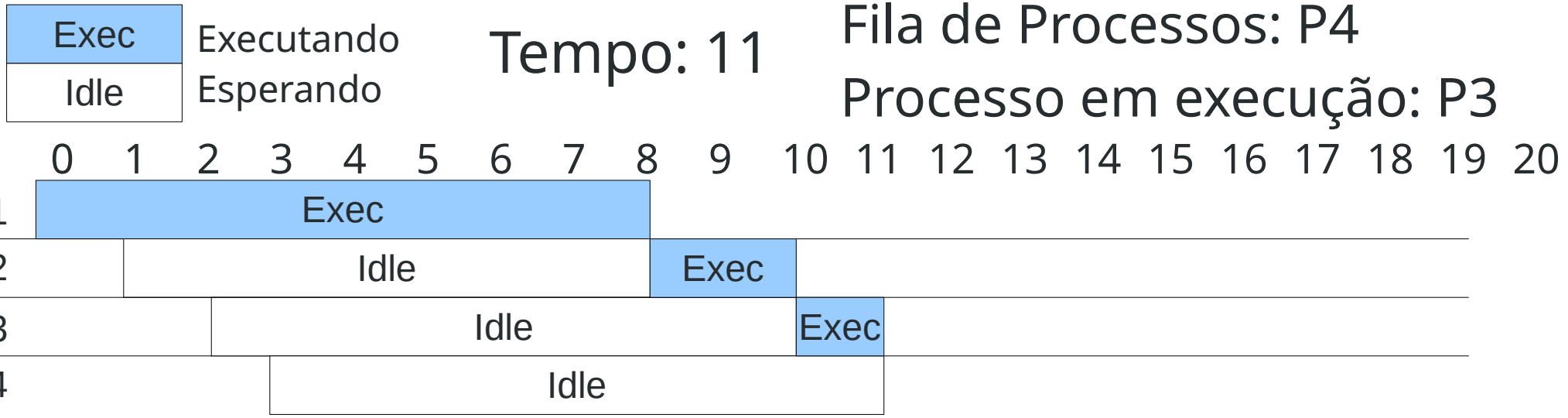
Tempo: 10

Fila de Processos: P3, P4

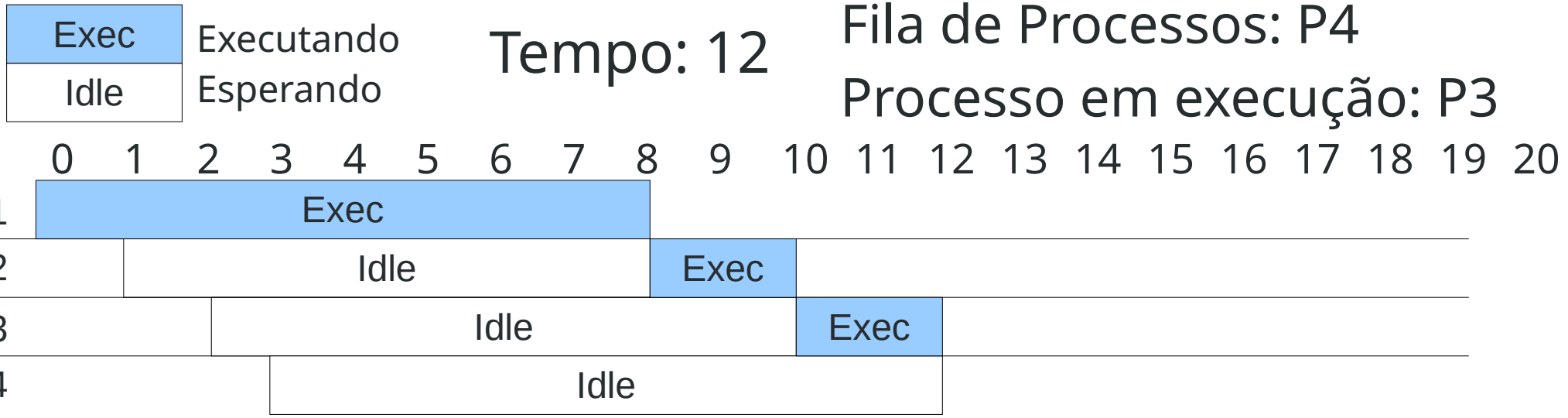
Processo em execução: P2



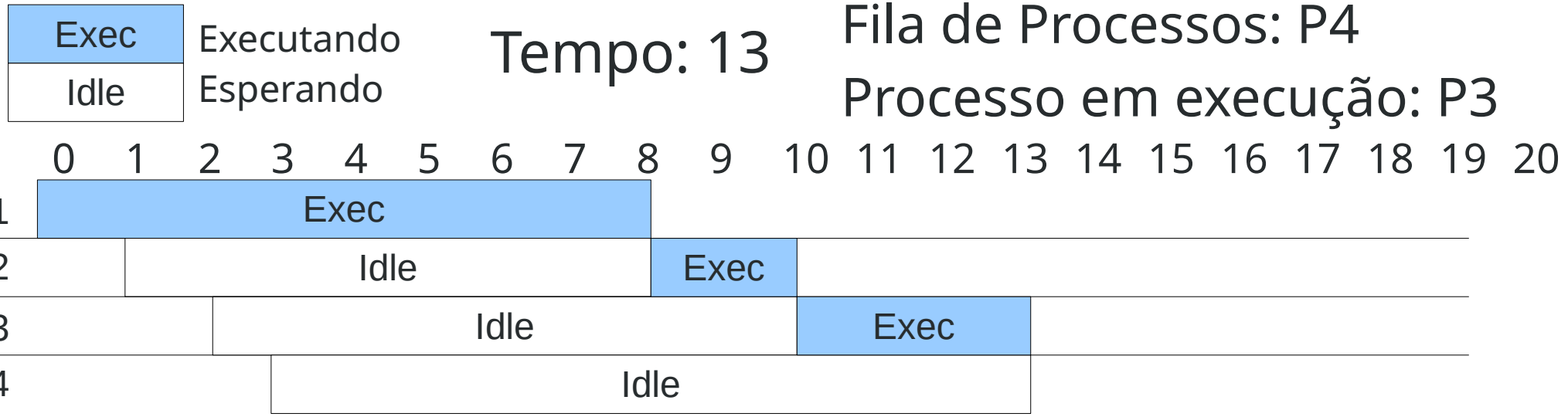
Escalonamento de Processos



Escalonamento de Processos



Escalonamento de Processos



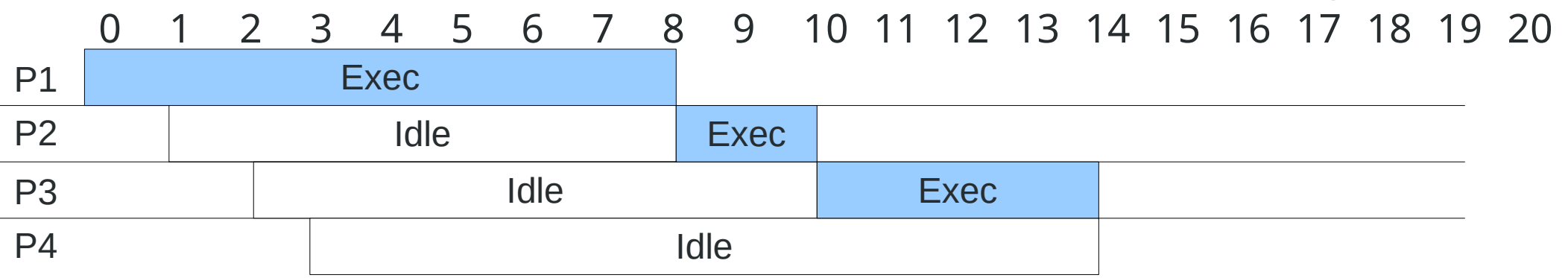
Escalonamento de Processos

Executando
Esperando

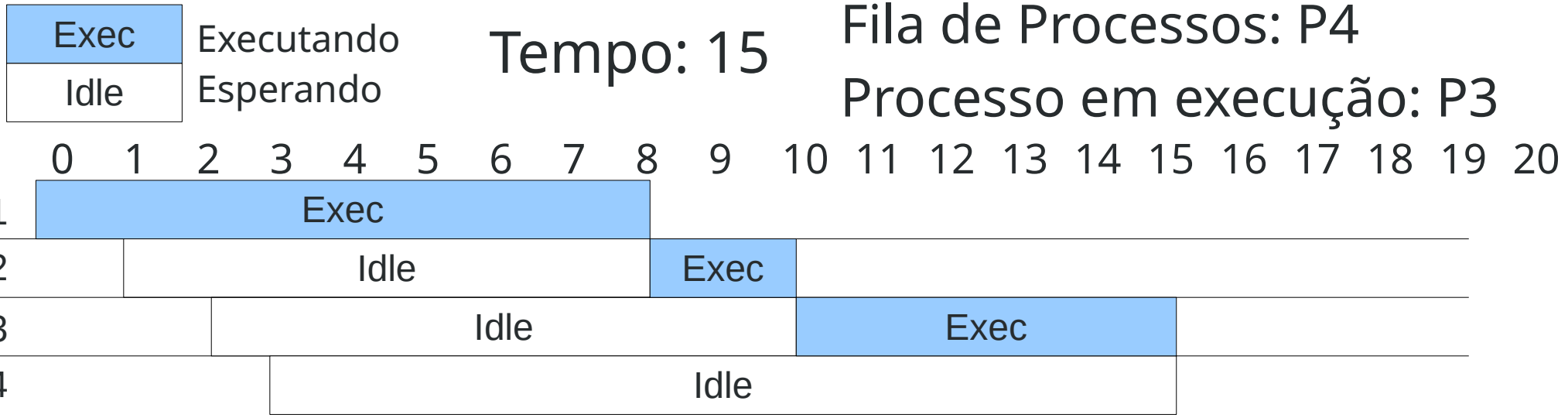
Tempo: 14

Fila de Processos: P4

Processo em execução: P3



Escalonamento de Processos



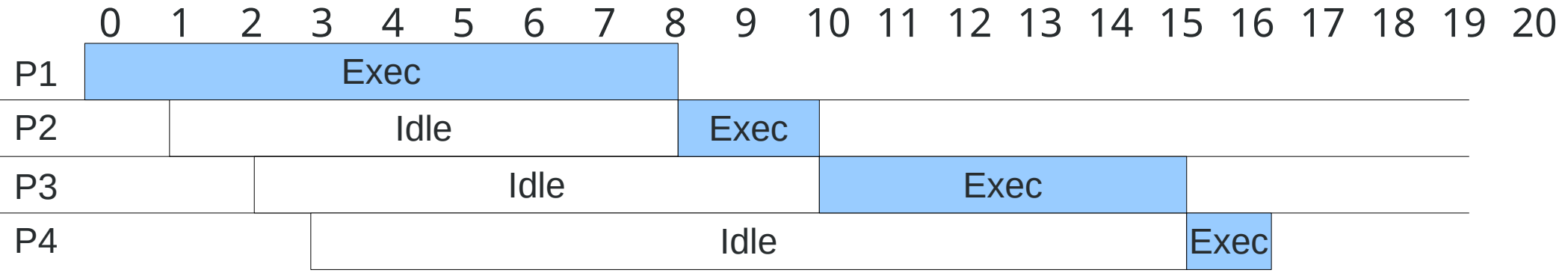
Escalonamento de Processos

Executando
Esperando

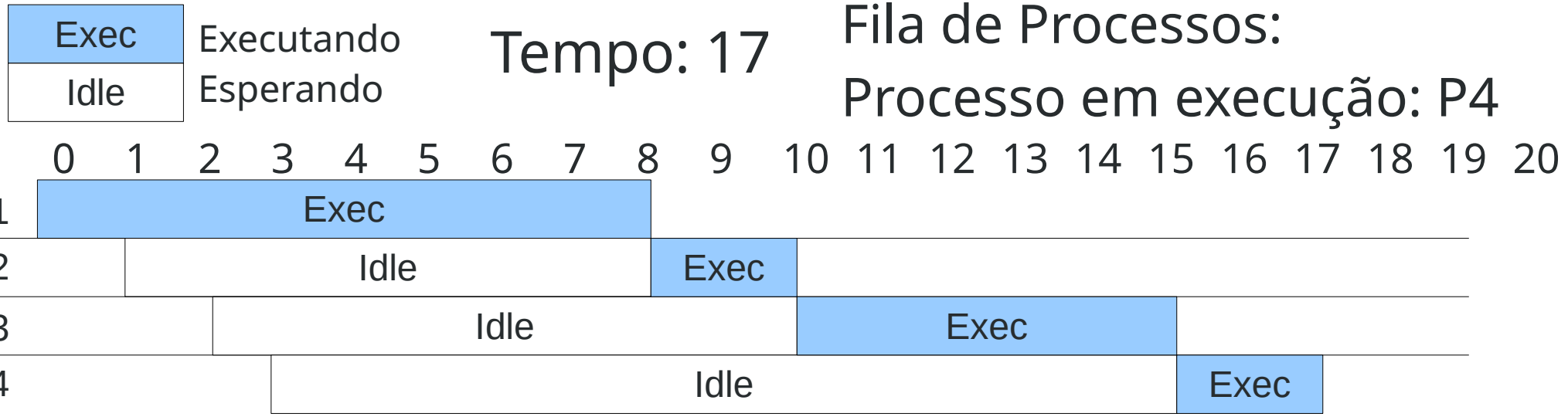
Tempo: 16

Fila de Processos:

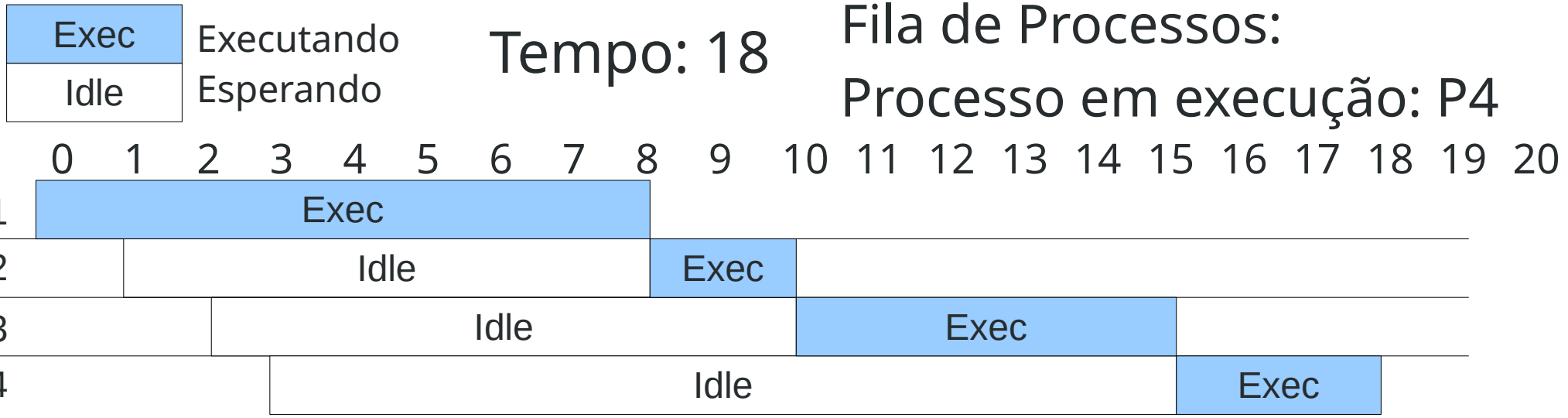
Processo em execução: P4



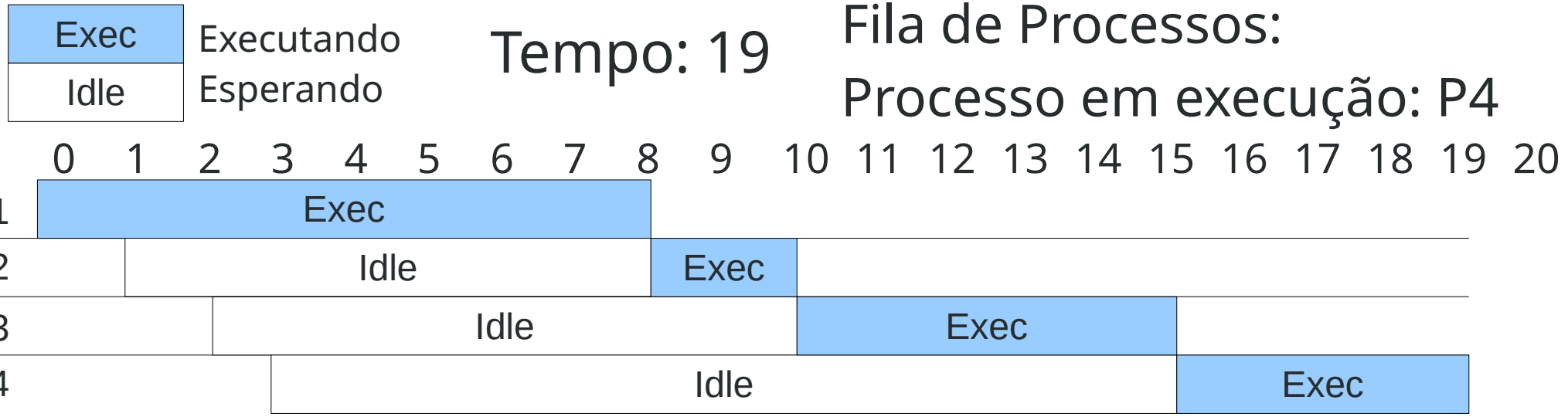
Escalonamento de Processos



Escalonamento de Processos



Escalonamento de Processos



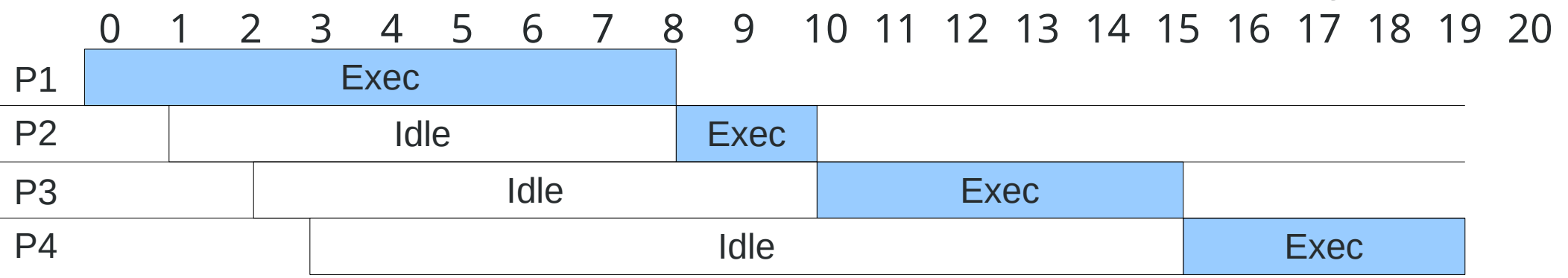
Escalonamento de Processos

Executando
Esperando

Tempo: 20

Fila de Processos:

Processo em execução:



First Come First Served

- Tempo de Espera = Tempo de Serviço – Tempo de Chegada
- Nenhum dos processo gera chamada blocantes

Processo	Tempo de Chegada	Tempo de Execução	Tempo de Serviço	Tempo de Espera
P0	0	8	0	0
P1	1	2	8	7
P2	2	5	10	8
P3	3	4	15	12

First Come First Served

- **Solução**: Tempo médio de espera
  $(0 + 7 + 8 + 12) \ / \ 4 = 6{,}75$

First Come First Served

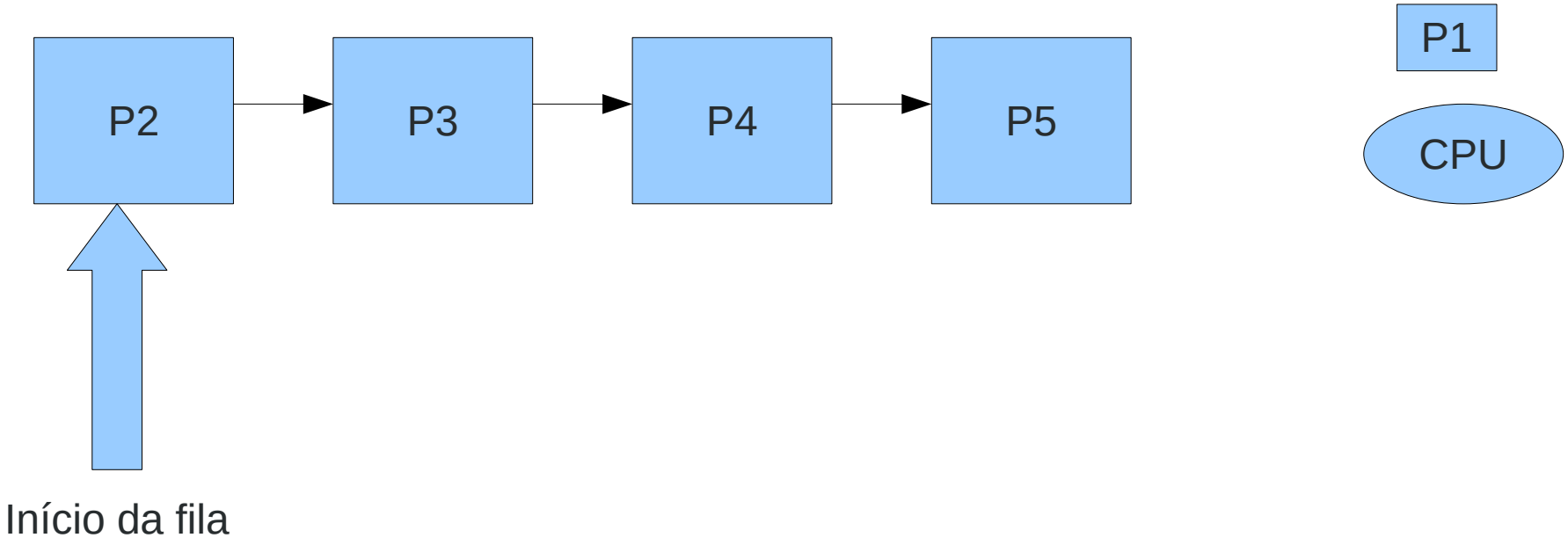
- **Justiça:** Não. Um processo pode monopolizar
- **Eficiência**: OK. A CPU sempre é utilizada, se existe trabalho
- **Minimizar o tempo de resposta**: Não. Caso um processo monopolize
- **Minimizar o turnaround:** Tarefas longas sendo executadas primeiro, aumenta turnaround (Ver exemplo do SJF a frente)
- **Minimizar waiting time:** O tempo de espera não é considerado durante o escalonamento
- **Maximizar throughtput:** Se a primeira tarefa for longa, o tempo de término fica alta

Round-Robin

- Cada processo tem o direito de usar o processador por um intervalo de tempo pré-definido. Este intervalo de tempo é denominado quantum
- Quando o quantum se esgota, o processador é dado a outro processo
- Algoritmo por alternância circular
- Algoritmo justo

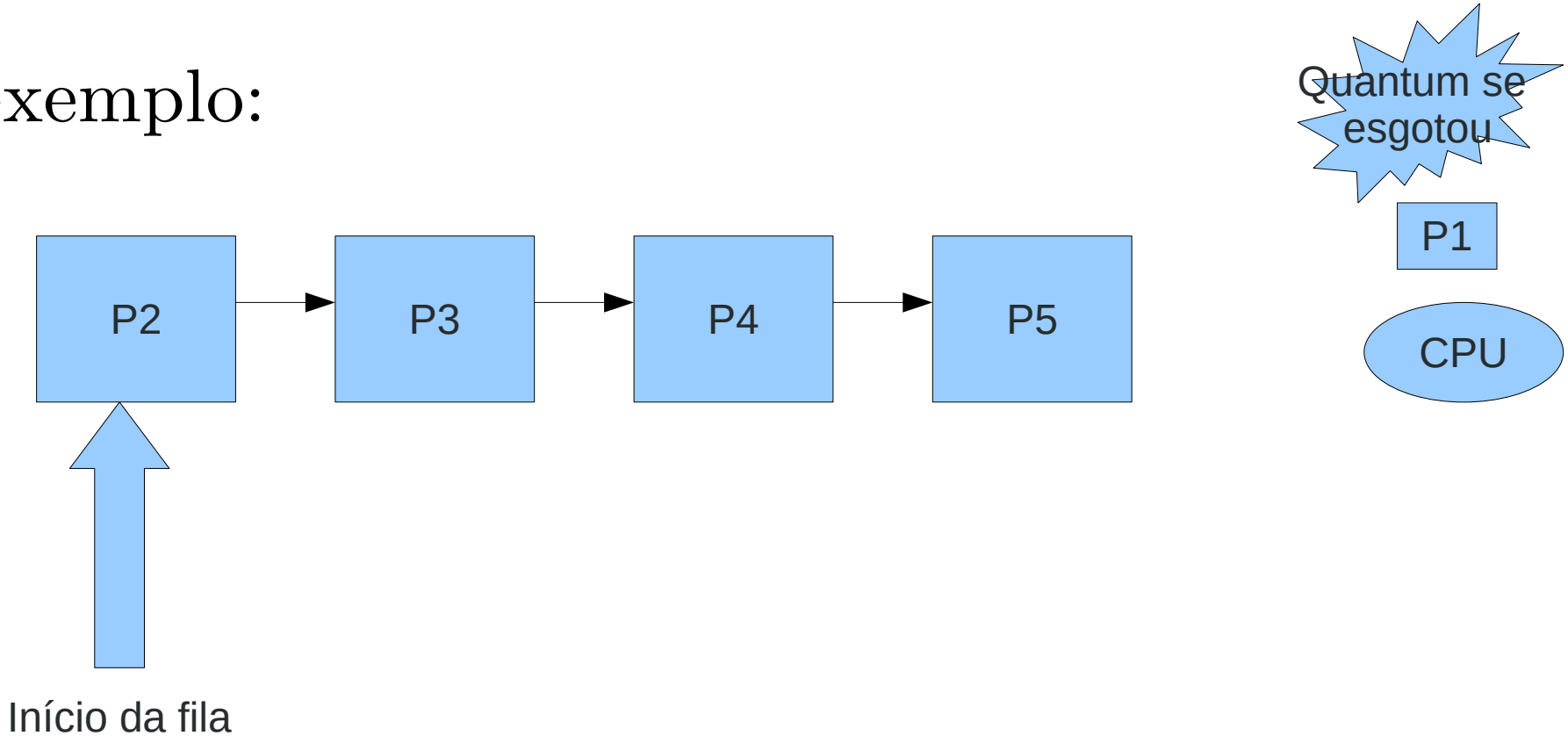
Round-Robin

- Exemplo:



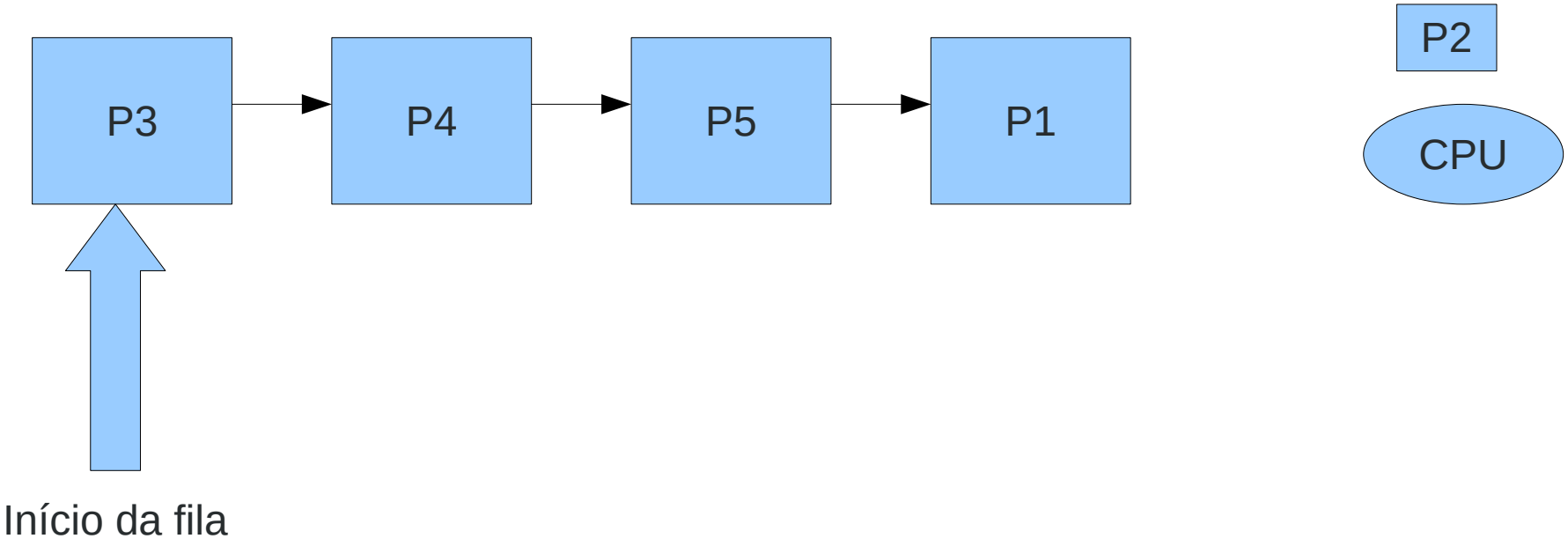
Round-Robin

- Exemplo:



Round-Robin

- Exemplo:



Round-Robin

- Um dos maiores problemas do algoritmo de escalonamento round-robin diz a respeito da determinação de um bom valor a ser atribuído ao quantum
- Para determinar esse valor, deve-se considerar o tempo médio da troca de contexto e o tempo de resposta desejado
  - Qual a influência que o tempo da troca de contexto?
- Quantum padrão 100 ms

Round-Robin

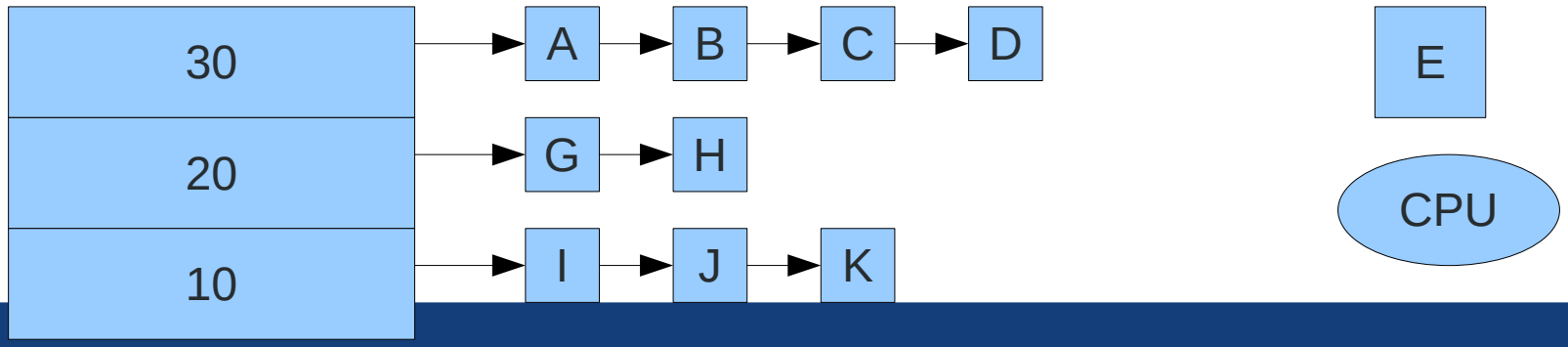
- Quantum muito longo: FCFS
- Quantum muito curto: perda de vários requisitos, tais como eficiência
- Para o algoritmo ser eficiente é necessário que o tempo de troca de contexto seja consideravelmente menor que a execução do programa
  - Verdade para os sistemas computacionais modernos

Escalonamento com Prioridades

- Baseia-se no fato que alguns processos são prioritários e devem ser executados antes dos outros
- Sistemas Operacionais em Tempo Real
- Cada processo é atribuída uma prioridade. Processos com prioridade maior rodam primeiro

Escalonamento com Prioridades

- As prioridades podem ser atribuídas de duas formas: estática ou dinâmica
- **Estática**: os processos são divididos em classes e a cada classe é atribuída uma prioridade. A cada prioridade existe uma fila de prontos associada



Escalonamento com Prioridades

- **Dinâmica:** o sistema analisa o comportamento dos processos e atribui prioridades favorecendo um certo tipo de comportamento
  - Processos I/O devem possuir prioridade alta
  - Prioridade dinâmica: 1/f, onde f é a fração do quantum de tempo usada na última rodada do processo

Shortest Job First

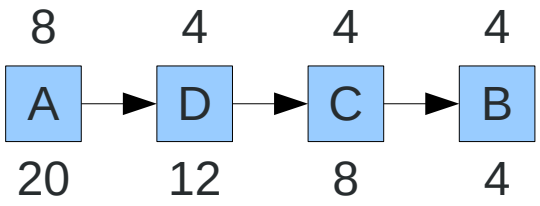
- Algoritmo projetado para sistemas em lote
- Reduzir o tempo de turnaround (tempo de lançamento do processo até seu término)
- Requer que o tempo total de execução do processo seja conhecido antes do seu início
- Algoritmo: Dado um conjunto de processos, execute os de menor tempo de execução antes

Shortest Job First

- Exemplo:

Início

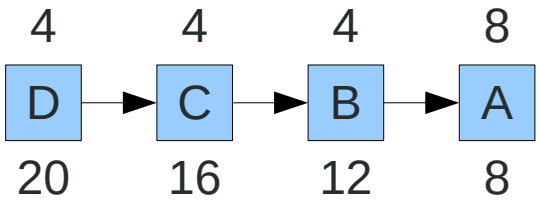
Tempo de execução:



SJF

Turnaround:

Tempo de execução:



FCFS

Turnaround:

Shortest Job First

- Aplicação para sistemas interativos
- Processo interativo:
  - Espera comando
  - Executa comando
- Se considerarmos cada "executa comando" como um trabalho, podemos aplicar o SJF para processos interativos

Shortest Job First

- Desvantagem: como determinar o tempo de execução do comando?
- **Problema da parada é indecidível**
- Necessário combinar técnicas para estimar valores baseados em execuções anteriores (aging)
- Quando um valor fica antigo, ele praticamento não influencia na estimativa

Escalonamento em Dois Níveis

- Um caso típico de escalonamento em dois níveis é o algoritmo que considera tanto os processos que estão em memória como os processos que estão em disco
  - Primeiro nível: manipula os processos que estão carregados em memória (FCFS, SJF, ...)
  - Segundo Nível: examina periodicamente o tempo de execução dos processos e os tira ou carrega em memória
- Quando o modelo de processos inclui threads, podemos ter também em 2 níveis. O primeiro determina qual processo irá rodar e o segundo determina qual thread do processo selecionado irá executar

Referências

- Capítulo 2 – TANENBAUM, A. S. *Sistemas Operacionais Modernos*. 4ª ed. Prentice Hall, 2016.
- Capítulo 5 e 8 – MACHADO, F. B; MAIA, L. P. *Arquitetura de Sistemas Operacionais*. 5ª ed. Rio de Janeiro: LTC, 2013.