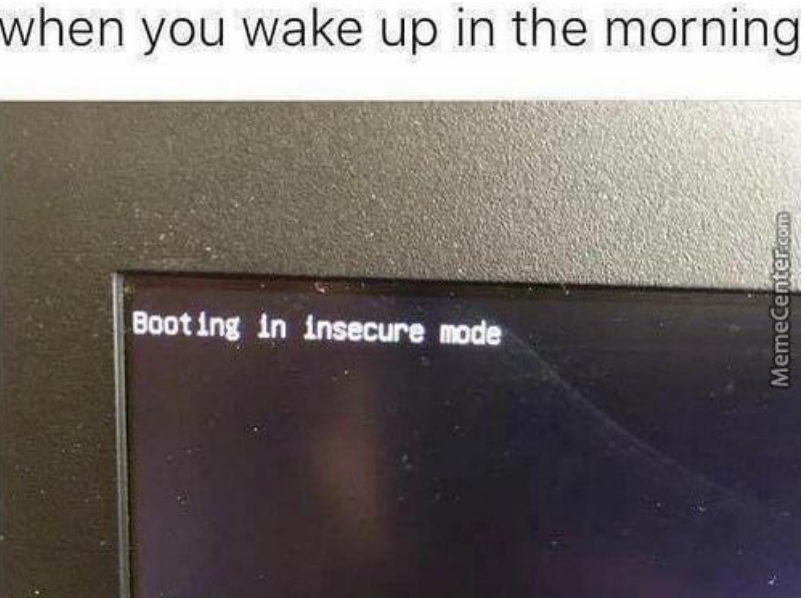
SISTEMAS OPERACIONAIS 1
21270 A

PROCESSO E GERENCIAMENTO DE PROCESSO



Departamento de Computação
Prof. Kelen Cristiane Teixeira Vivaldini

Departamento de Computação

Processos

Definição de Processo:

- Uma abstração de um programa em execução

Departamento de Computação

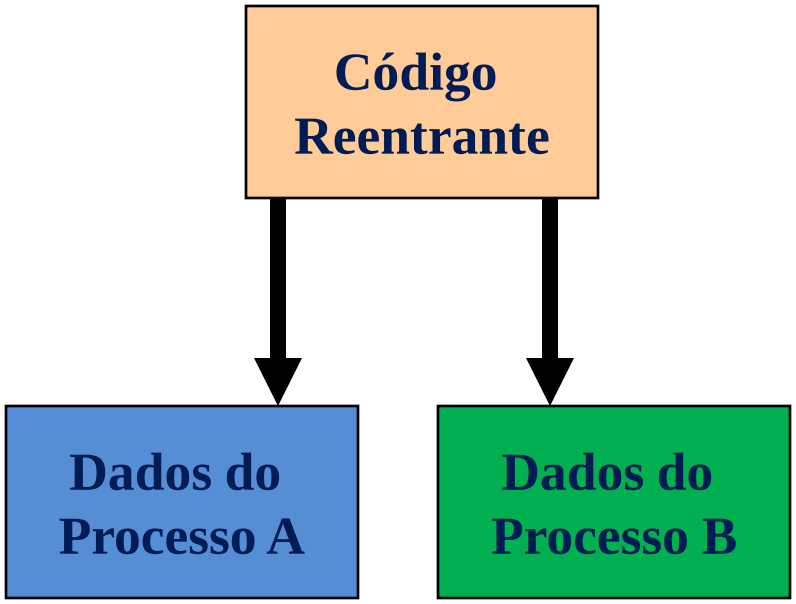
Processos

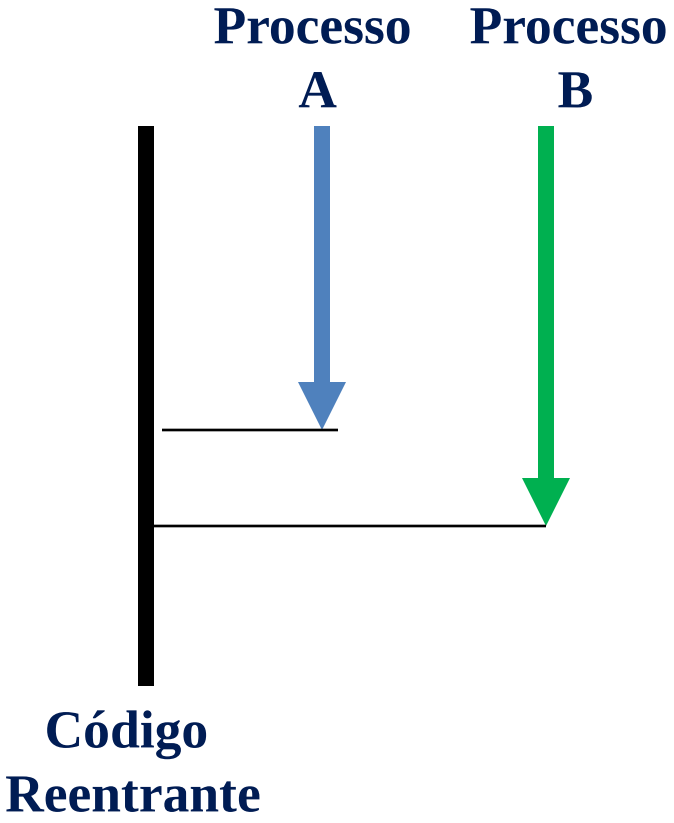
- É um programa em execução, incluindo os valores correntes do contador de programa, registradores, e variáveis. O conceito de processo é dinâmico, em contraposição ao conceito de programa, que é estático.

- Nem sempre um programa equivale a apenas um processo
- Em sistemas que permitem a reentrância, o código de um programa pode gerar diversos processos.

Processos

Exemplo de Reentrância





Processos

- **Reentrância**
  - É a capacidade de um código executável (código reentrante) ser compartilhado por diversos usuários, exigindo que apenas uma cópia do programa esteja na memória.
  - Permite que cada usuário possa estar em um ponto diferente do código reentrante, manipulando dados próprios, exclusivos de cada usuário.

Processos

Exemplo de Reentrância

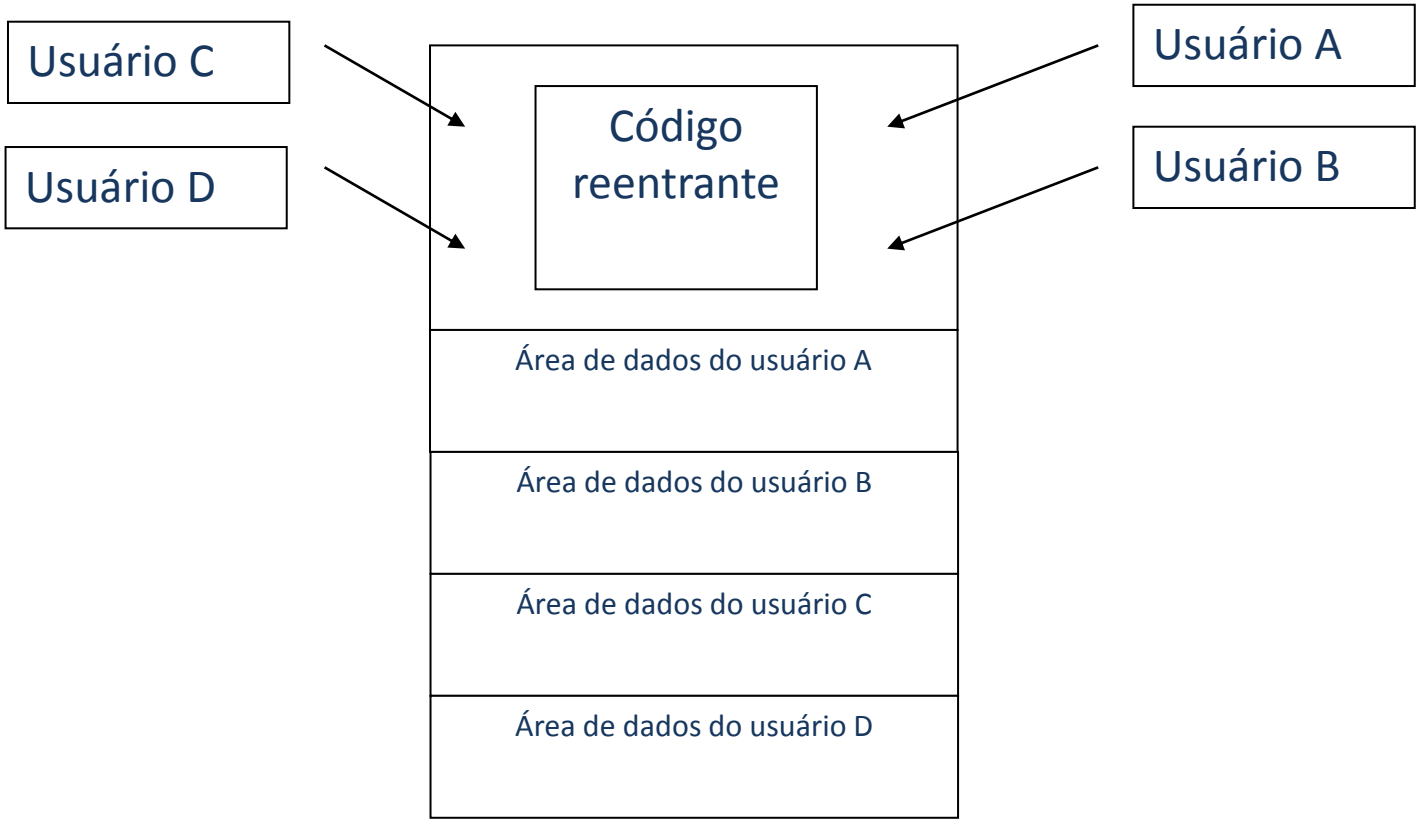
Não reentrante

```
int g_var = 1;

int f()
{
  g_var = g_var + 2;
  return g_var;
}

int g()
{
  return f() + 2;
}
```

Reentrante

```
int f(int i)
{
  return i + 2;
}

int g(int i)
{
  return f(i) + 2;
}
```

Processos



Memória principal

Criação de Processos

Principais eventos que causam a criação de um processo

1. Inicialização do Sistema;
2. Execução de uma chamada ao sistema de criação de processo por um processo em execução;
3. Uma requisição de Usuário para a criação de um novo processo ;
4. Início de um Job em lote.

Departamento de Computação

Criando Processos

- UNIX:
  - Fork;
    - Cria processo Pai e processo Filho com mesmo endereçamento;
    - Depois o processo Filho tem endereçamento separado;

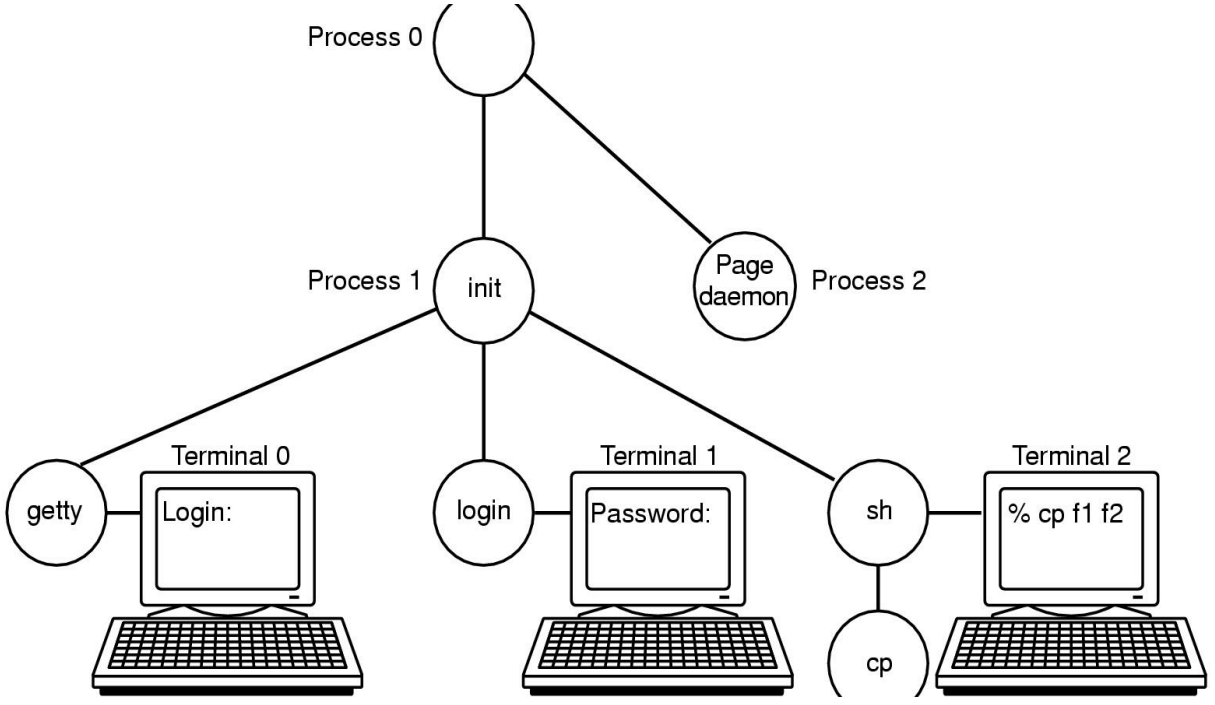
Departamento de Computação

Criando Processos

- UNIX:
  - Fork;
    - Cria processo Pai e processo Filho com mesmo endereçamento;
    - Depois o processo Filho tem endereçamento separado;
- Windows:
  - CreateProcess
    - Cria processo Pai e processo Filho com mesmo endereçamento sempre;

Criando Processos

- Exemplo UNIX:
  - Processo init: gera vários processos filhos para atender os vários terminais que existem no sistema;



Outros processos são gerados nos terminais

Término de um Processo

Condições que podem provocar o término de um processo

1. Saída normal (voluntária);
2. Saída por erro (voluntária);
3. Erro Fatal (involuntária);
4. Destruído por outro processo (involuntário).

Departamento de Computação

Término de um Processo

- Condições:
  - Término normal (voluntário):
    - A tarefa a ser executada é finalizada;
    - Chamadas: `exit` *(UNIX) e* `ExitProcess` *(Windows)*
  - Término com erro (voluntário):
    - O processo sendo executado não pode ser finalizado: gcc filename.c, o arquivo filename.c não existe;

Departamento de Computação

Término de um Processo

- Término com erro fatal (involuntário);
  - Erro causado por algum erro no programa (*bug*):
    - Divisão por 0 (zero)*;*
    - Referência à memória inexistente ou não pertencente ao processo;
    - Execução de uma instrução ilegal;
- Término causado por algum outro processo (involuntário):
  - Kill (UNIX) e TerminateProcess (Windows);

Hierarquias de Processos

Pai cria um processo filho, processo filho pode criar seus próprios processo

- Formação de hierarquia
  - UNIX : “grupo de processos” ("process group”)
  - Windows não possui conceito de hierarquia de processos
    Todos os processos são criados da mesma forma

Departamento de Computação

Processos

Um sistema operacional deve incluir funções para o gerenciamento de processos, como por exemplo:

- criação e remoção (destruição) de processos;
- controle do progresso dos processos;
- agir em condições excepcionais que acontecem durante a execução de um processo, incluindo interrupções e erros aritméticos;
- alocação de recursos de hardware entre os processos;
- fornecer um meio de comunicação através de mensagens ou sinais entre os processos.

Departamento de Computação

Processos

Existem 2 tipos de sistemas, com relação aos processos:

Sistemas Estáticos

- Geralmente são sistemas projetados para executar uma única aplicação. Permite que todos os processos necessários já estejam presentes quando o sistema é iniciado.

Sistemas Dinâmicos

- Possuem um número variável de processos. Necessita de uma forma de criar e destruir processos durante a execução.

Departamento de Computação

Bloco de Controle de Processos (PCB)

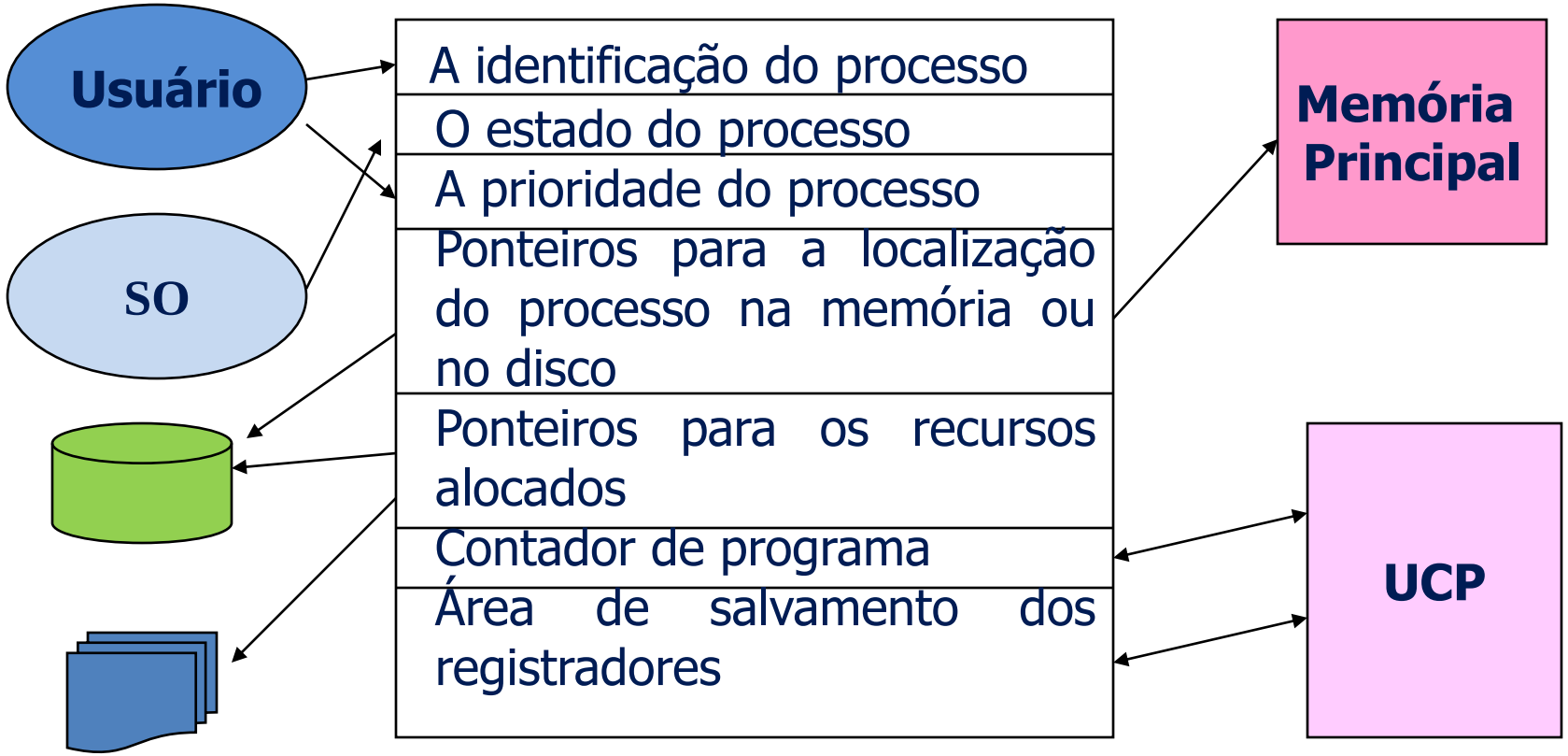
No sistema, cada processo será representado por seu resumo, que consiste no Bloco de Controle de Processo (BCP), também conhecido por Bloco de Controle de Programa ou Descritor de Processo.

Bloco de Controle de Processos (PCB)

O BCP consiste de uma estrutura de dados contendo informações importantes sobre o processo, incluindo:

- A identificação do processo;
- O estado do processo;
- A prioridade do processo;
- Ponteiros para a localização do processo na memória ou no disco;
- Ponteiros para os recursos alocados;
- Contador de programa;
- Área de salvamento dos registradores, etc.

Bloco de Controle de Processos (PCB)

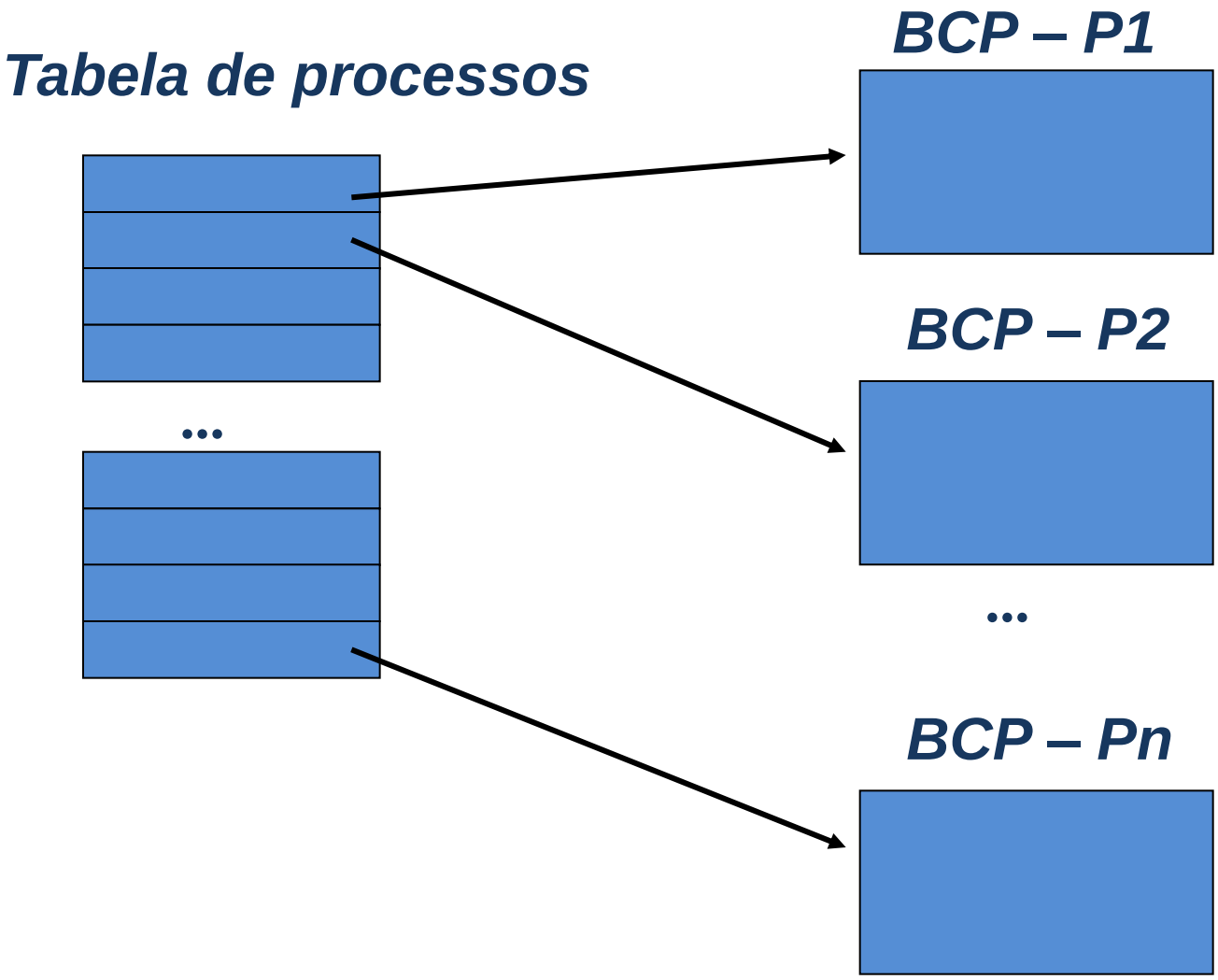


Departamento de Computação

Implementação de Processos

- Tabela de Processos:
  - Cada processo possui uma entrada;
  - Cada entrada possui um ponteiro para o bloco de controle de processo (BCP) ou descritor de processo;
  - BCP possui todas as informações do processo → contextos de hardware, software, endereço de memória;

Implementação de Processos



Implementação de Processos

Process management	Memory management	File management
Registers	Pointer to text segment	Root directory
Program counter	Pointer to data segment	Working directory
Program status word	Pointer to stack segment	File descriptors
Stack pointer		User ID
Process state		Group ID
Priority		
Scheduling parameters		
Process ID		
Parent process		
Process group		
Signals		
Time when process started		
CPU time used		
Children's CPU time		
Time of next alarm		

Algumas informações do BCP

Os Estados de um Processo

Desde o momento em que um processo codificado pelo programador for colocado em memória ou disco, até o momento de sua destruição, ele poderá passar por quatro estados:

Indefinido: quando o processo é desconhecido ao S.O. Um processo estará neste estado antes de ser criado e depois de ser destruído. (Neste ponto, ele será apenas um bloco de código no disco ou na memória).

Bloqueado: quando o processo estiver parado à espera da ocorrência de um evento, equivalente a não estar em andamento progressivo. Ao ser criado, o processo passará por este estado.

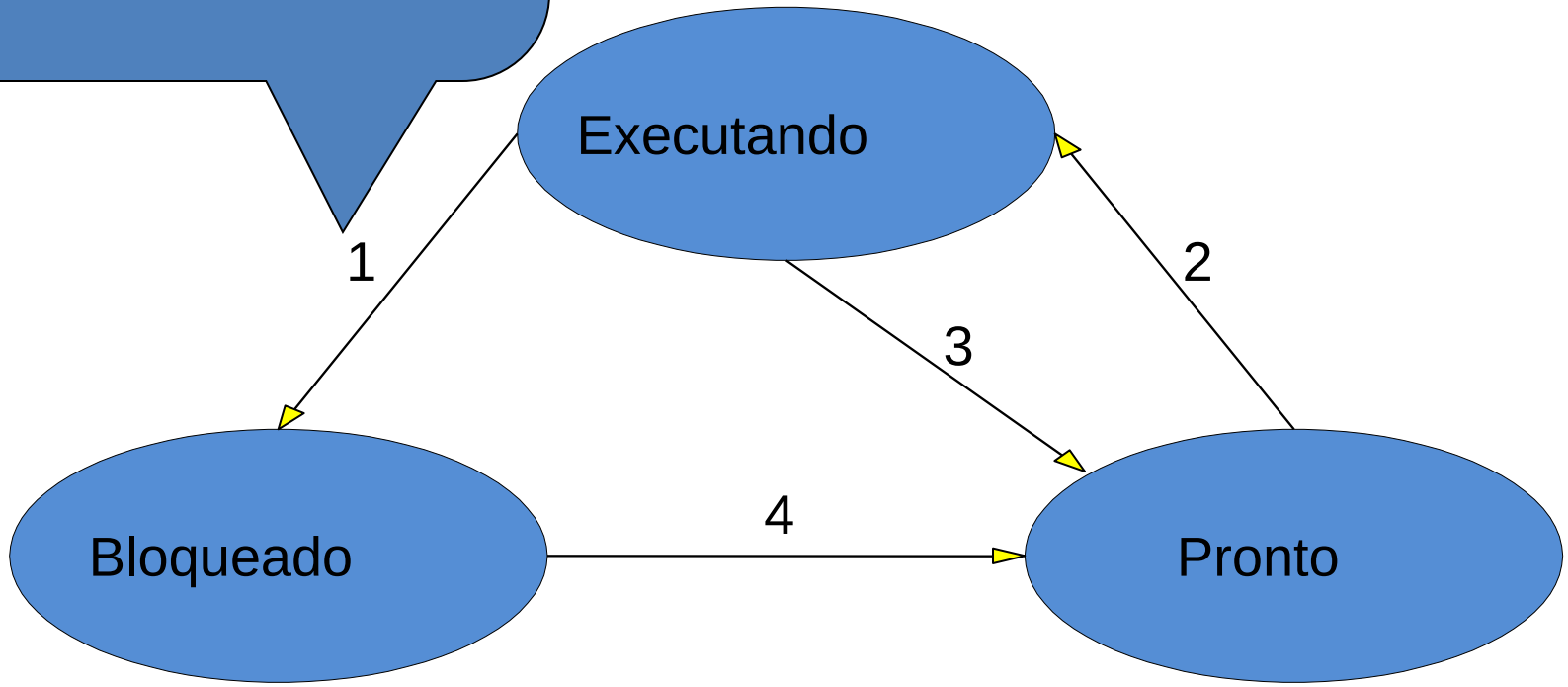
Pronto para a execução: quando o processo só não estiver em execução pelo fato da UCP estar sendo utilizada por outro processo.

Em execução: quando o processo estiver tendo andamento progressivo normal, usando a UCP.

Estados de Processos

Um processo sendo executado não pode continuar sua execução, pois precisa de algum evento (E/S ou semáforo) para continuar;

os:

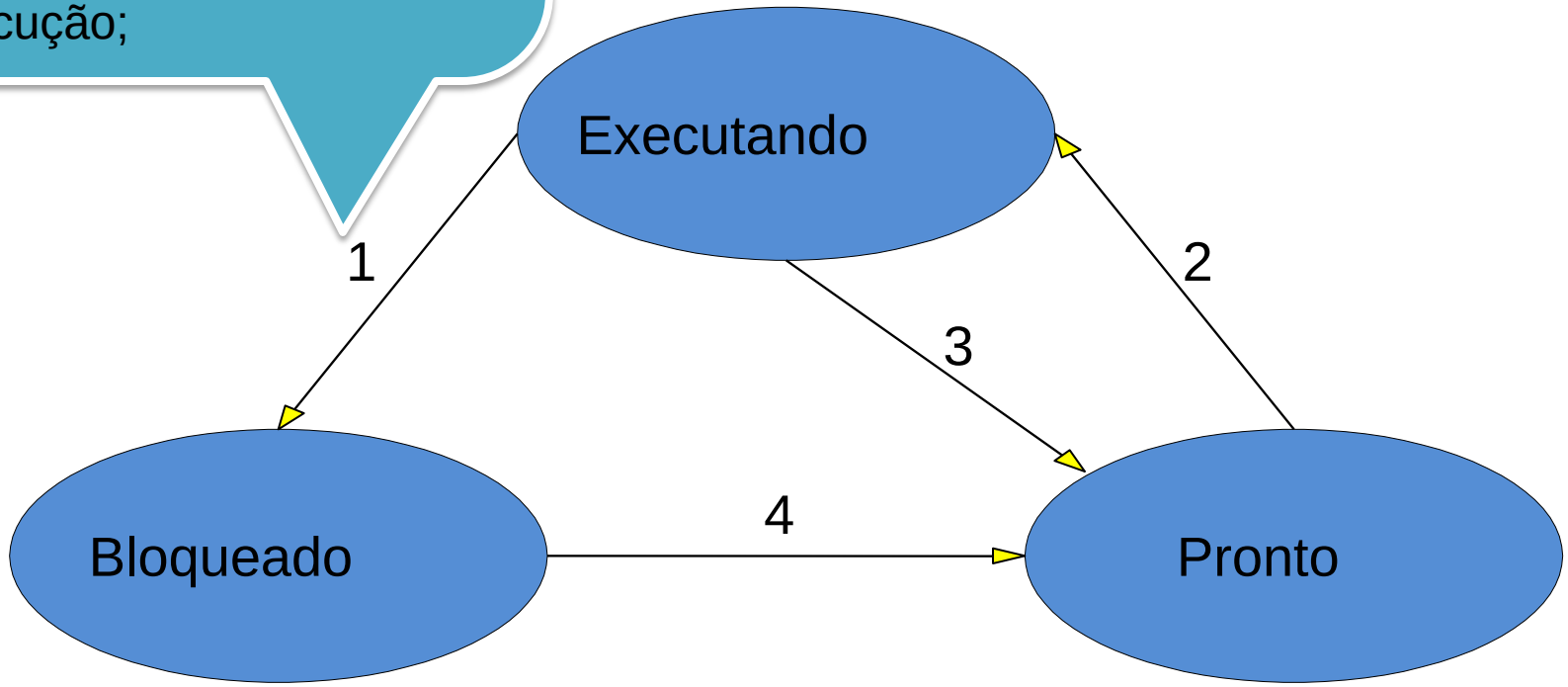


Estados de Processos

Um processo é bloqueado de duas maneiras:

- chamada ao sistema: `block` ou `pause`;
- se não há entradas disponíveis para que o processo continue sua execução;

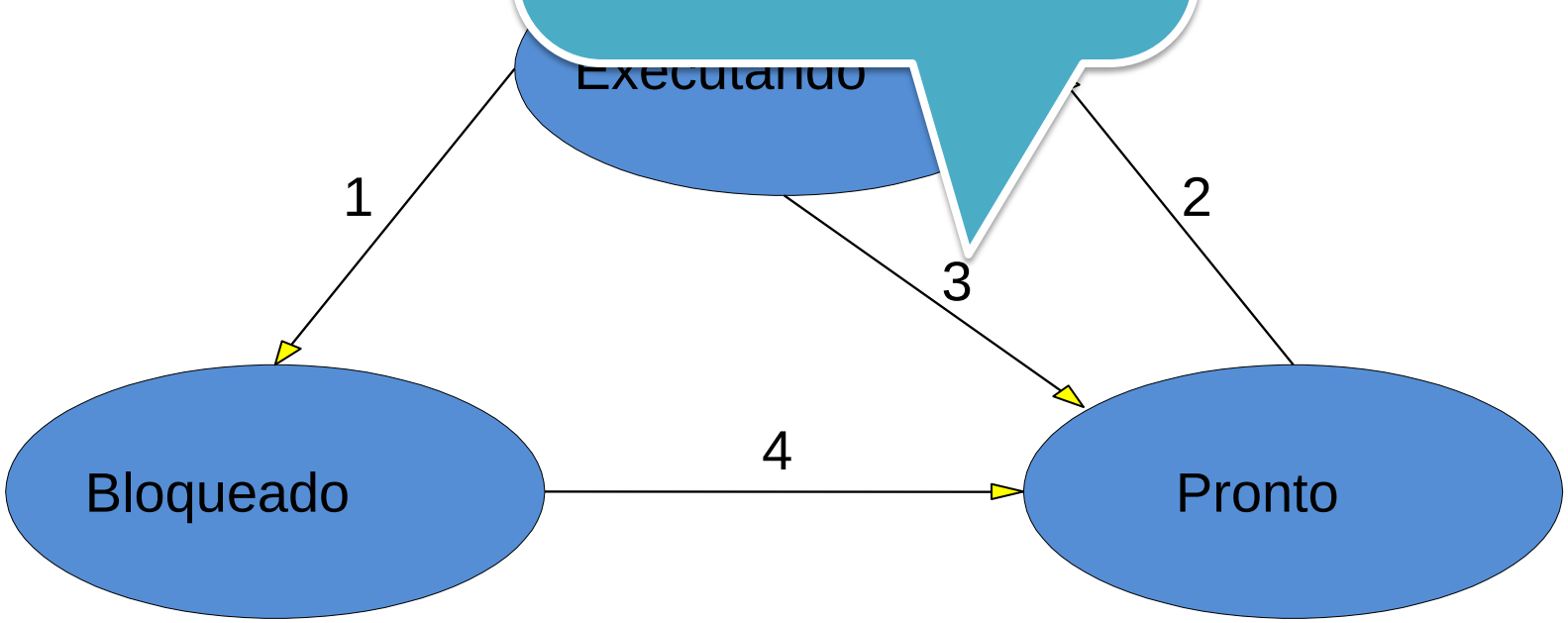
os:



Estados de Processos

As transições 2 e 3 ocorrem durante o escalonamento de processos:

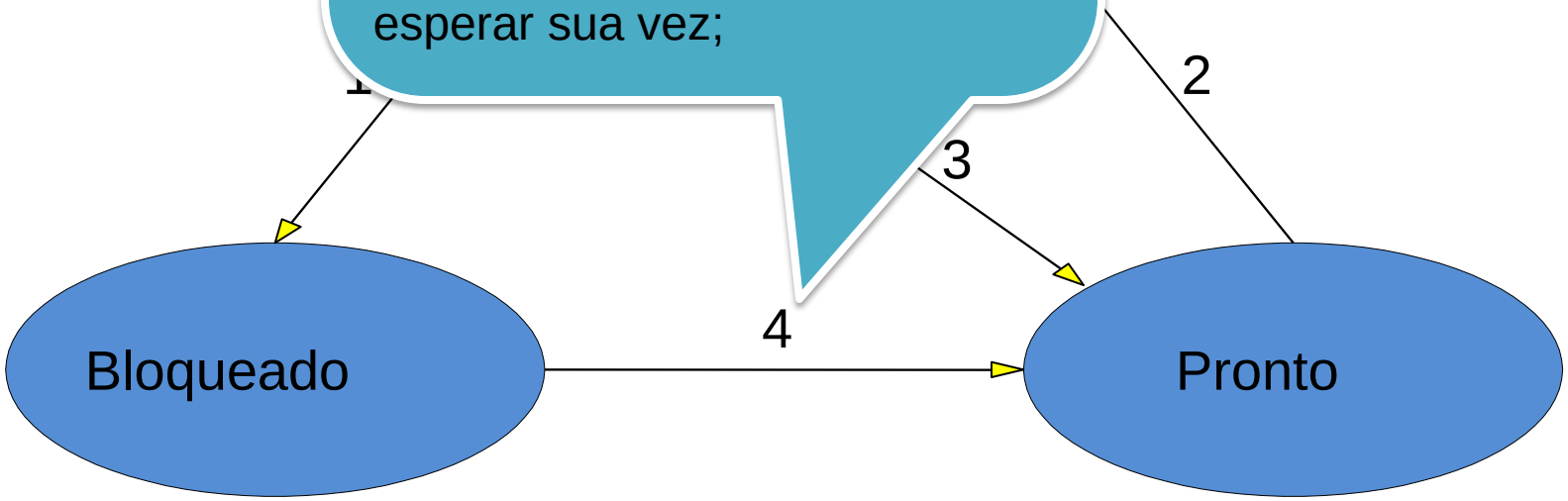
- o tempo destinado àquele processo acabou e outro processo é colocado no processador;



Estados de Processos

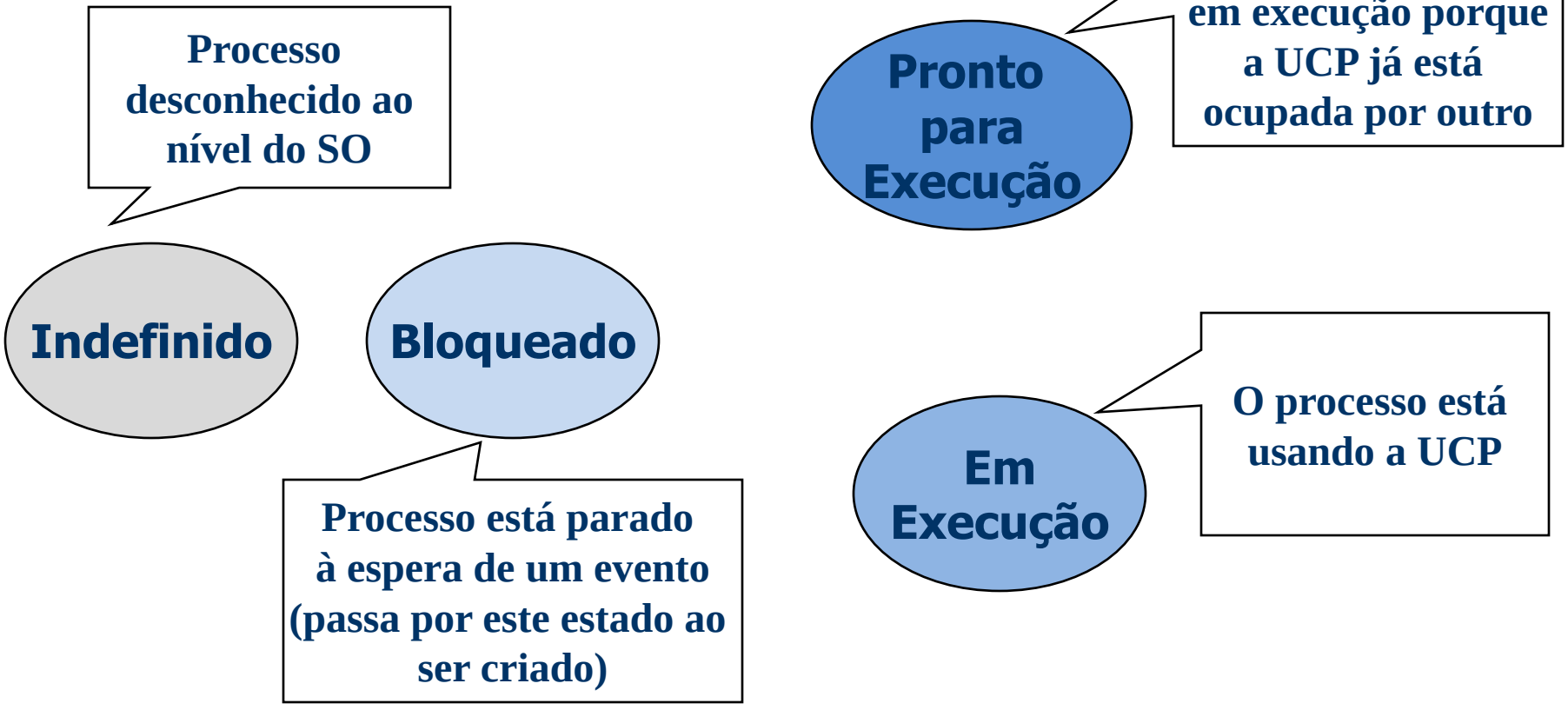
A transição 4 ocorre quando o evento esperado pelo processo bloqueado ocorre:

- se o processador está parado, o processo é executado imediatamente (2);
- se o processador está ocupado, o processo deve esperar sua vez;



Processos

- **Os Estados de um Processo**



As Ações que provocam mudanças de estado

As ações responsáveis pelas transições tem o seguinte significado:

Criar: colocar o processo a memória, tornando-o conhecido ao nível do sistema.

Acordar: liberar o andamento do processo. Esta ação será desempenhada quando ocorrer algum evento ou o disparo inicial do processo, fazendo com que haja a transição do estado bloqueado para o pronto para execução.

Despachar: dar seqüência imediata ao andamento do processo. O processo deverá estar em uma fila, e a ocorrência dessa ação resultará na retirada do processo da fila e sua colocação no estado em execução.

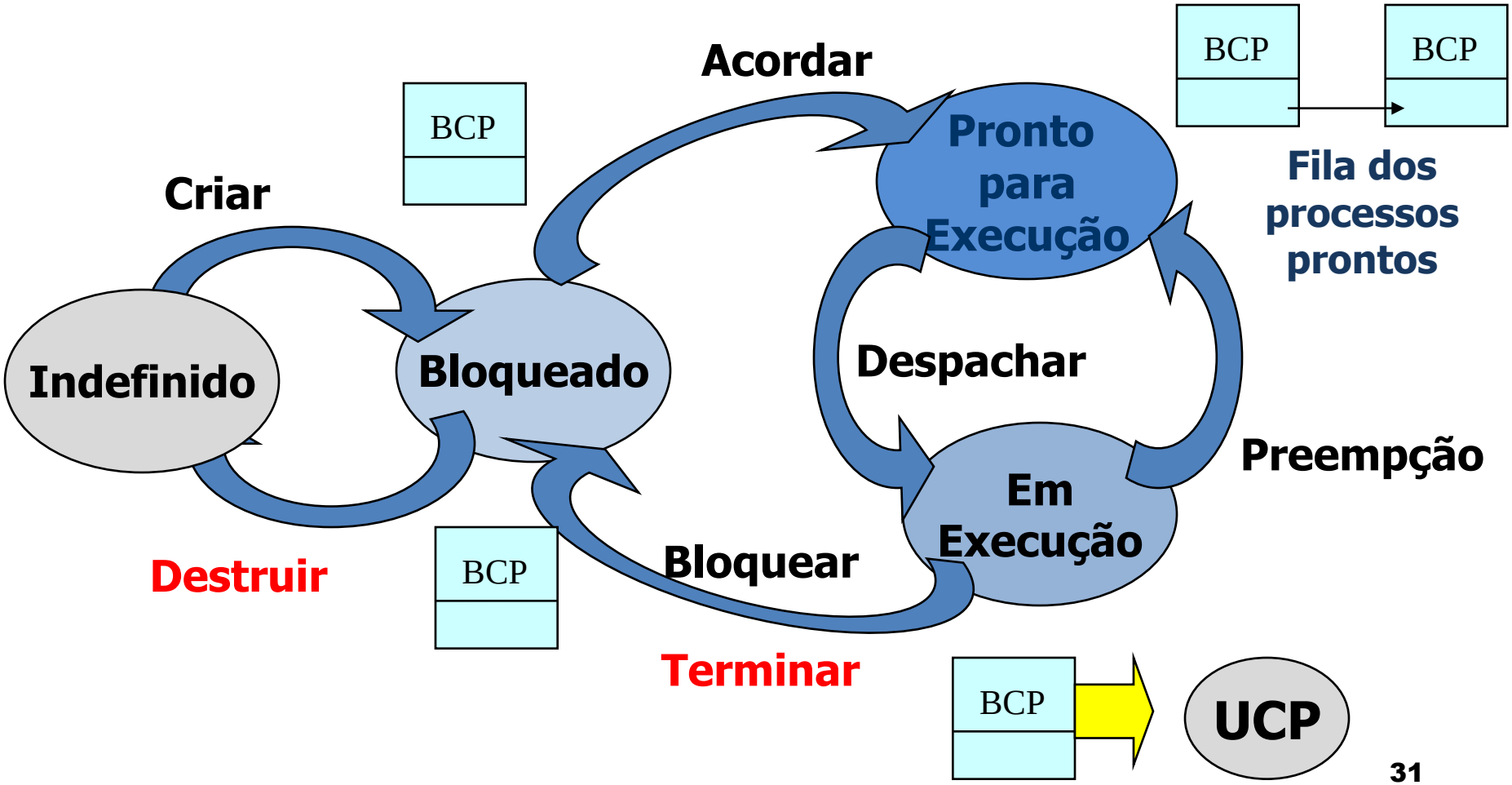
Bloquear: parar o andamento do processo para esperar a ocorrência de um evento. A ação bloquear será desempenhada quando uma determinada condição não for satisfeita, ou quando ocorrer o término do processo.

Preempção (Suspender): parar o andamento do processo por motivos alheios ao mesmo, como acontece na comutação forçada de fatias de tempo. Esta ação retirará o processo da UCP e o colocará numa fila, no estado pronto para a execução, liberando a utilização da UCP.

Destruir: liberar a área de memória ocupada pelo processo, tornando-o desconhecido ao nível do sistema.

Processos

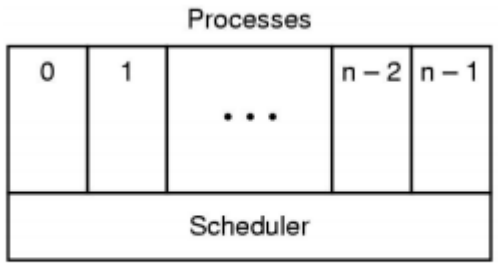
As Ações que provocam a transição entre estados



Processos

As Ações de Responsabilidade do Escalonador

- Quando um processo é criado
- Quando um processo termina
- Quando um processo faz uma operação de I/O
- Interrupção de relógio (sistemas preemptivos)

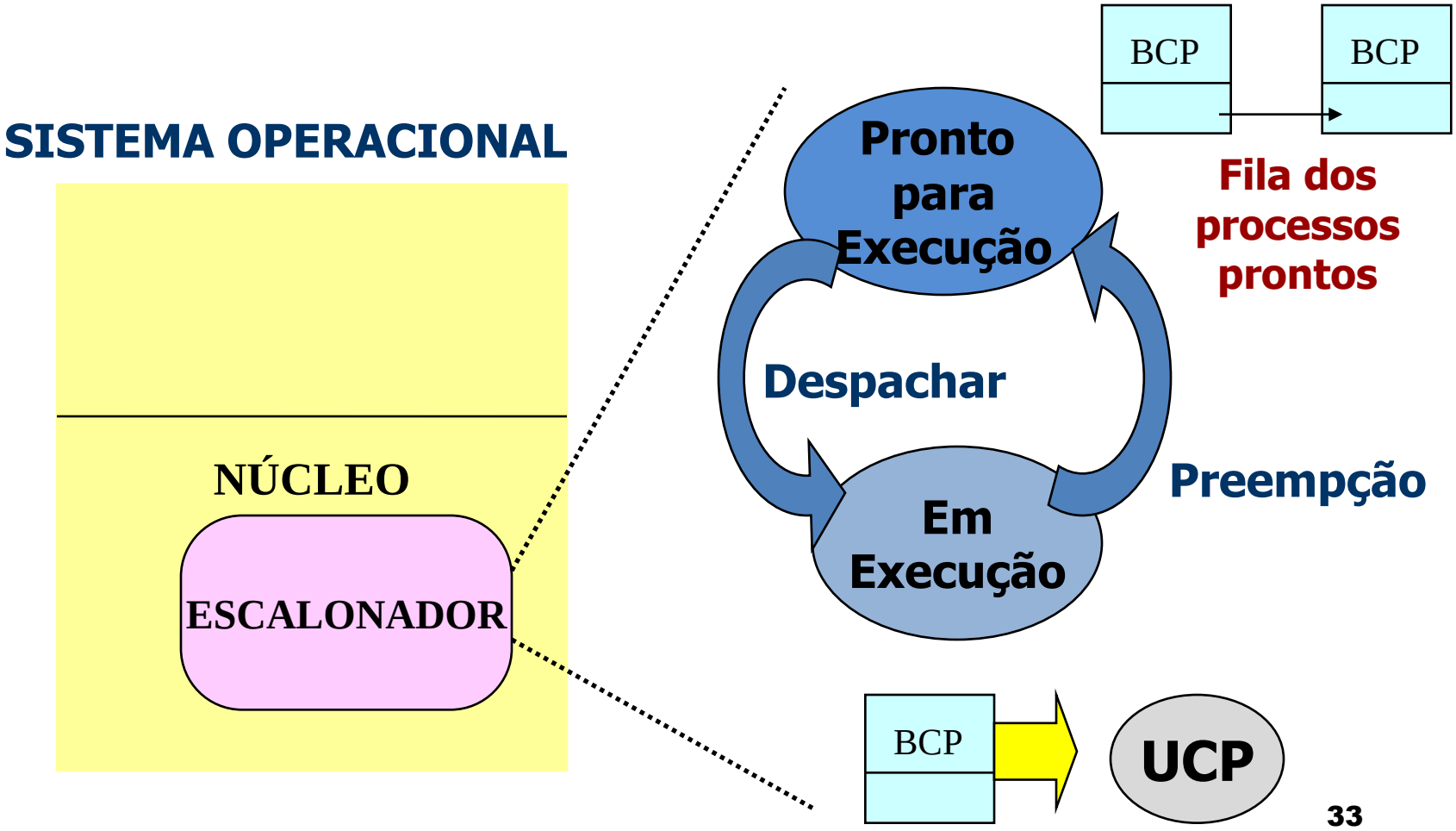


Tanenbaum: Figura 2.3

A função do escalonador é escolher qual deve ser o próximo processo a ser executado.

Processos

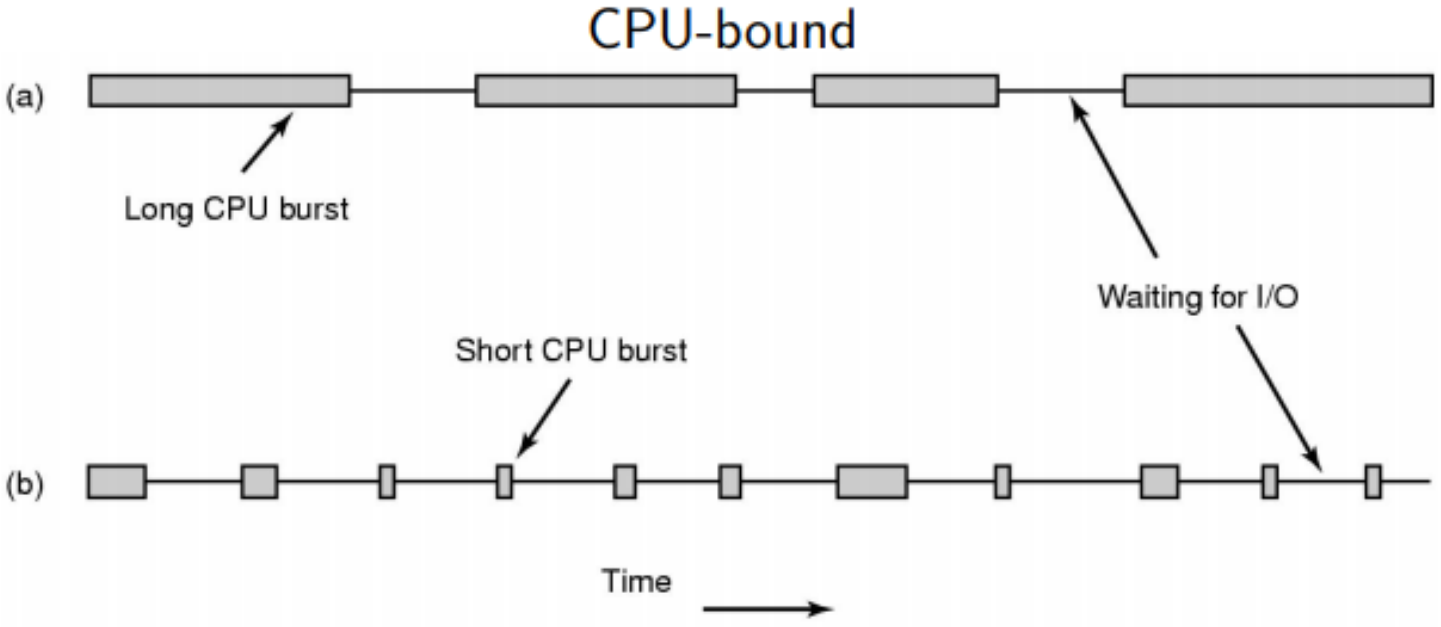
As Ações de Responsabilidade do Escalonador



Processos

- Processos *CPU-bound* (orientados à CPU): processos que utilizam muito o processador;
  - Tempo de execução é definido pelos ciclos de processador;
- Processos *I/O-bound* (orientados à E/S): processos que realizam muito E/S;
  - Tempo de execução é definido pela duração das operações de E/S;
- **IDEAL**: existir um balanceamento entre processos *CPU-bound* e *I/O-bound*;

Processos



Tanenbaum: Figura 2.37

Processos

Outras características dos processos

- Os processos não podem ser programados com suposições embutidas sobre temporização
- Em geral, um sistema multiplexa um só processador central entre vários processos. Os instantes em que o processador é realocado de um processo para outro são, em geral, imprevisíveis
- Um algoritmo de escalonamento determina quando parar o trabalho com um processo e atender um diferente

Escalonamento de Processos

- Escalonador de Processos escolhe o processo que será executado pela CPU;
- Escalonamento é realizado com o auxílio do hardware;
- Escalonador deve se preocupar com a eficiência da CPU, pois o chaveamento de processos é complexo e custoso:
  - Afeta desempenho do sistema e satisfação do usuário;
- Escalonador de processo é um processo que deve ser executado quando ocorre **mudança de contexto** (troca de processo);

Escalonamento de Processos

Mudança de Contexto

1. Hardware stacks program counter, etc.
2. Hardware loads new program counter from interrupt vector.
3. Assembly language procedure saves registers.
4. Assembly language procedure sets up new stack.
5. C interrupt service runs (typically reads and buffers input).
6. Scheduler decides which process is to run next.
7. C procedure returns to the assembly code.
8. Assembly language procedure starts up new current process.

Tanenbaum: Figura 2.5

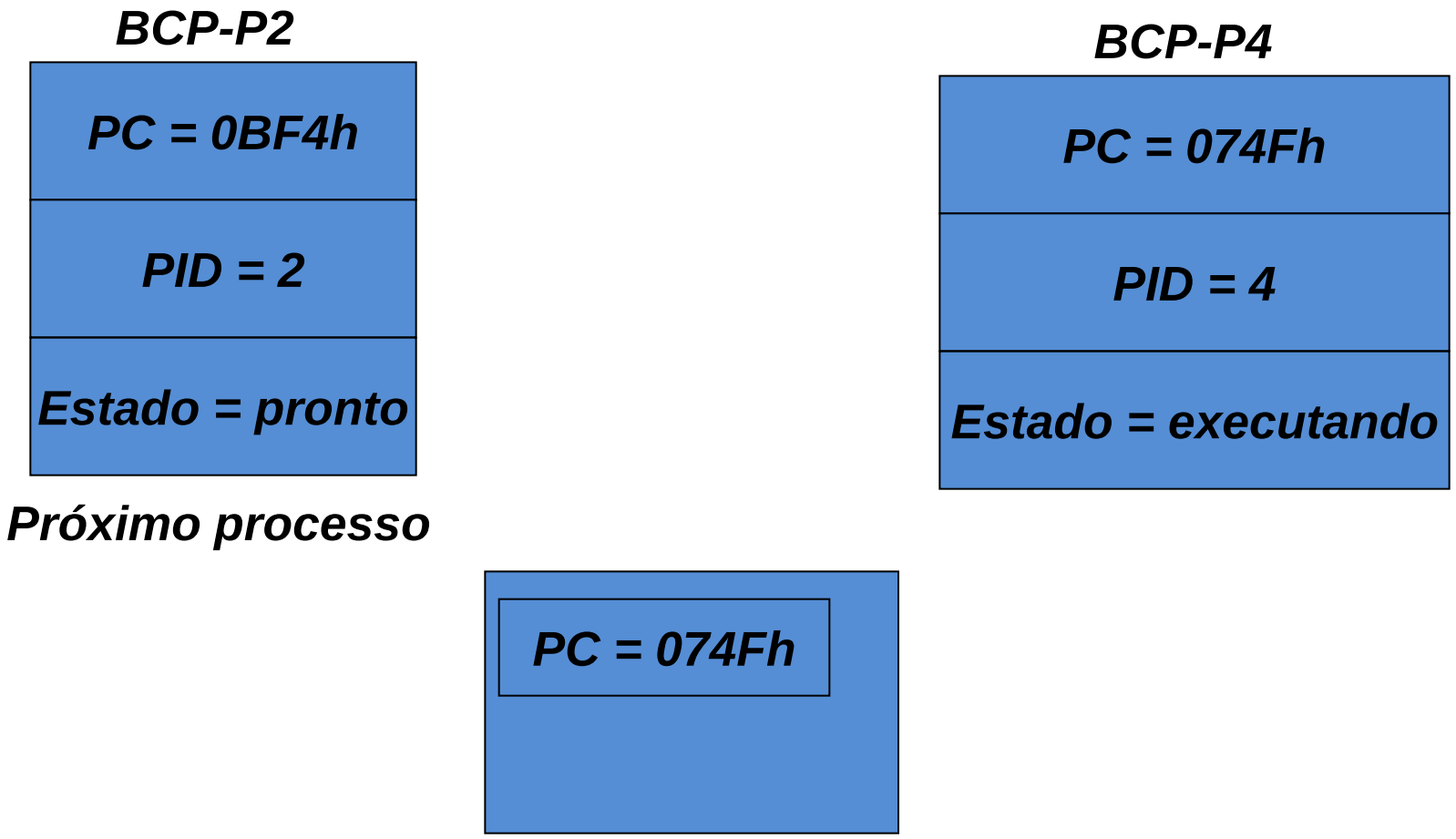
Departamento de Computação

Escalonamento de Processos

- Mudança de Contexto:
  - Overhead de tempo;
  - Tarefa cara:
    - Salvar as informações do processo que está deixando a CPU em seu BCP → conteúdo dos registradores;
    - Carregar as informações do processo que será colocado na CPU → copiar do BCP o conteúdo dos registradores;

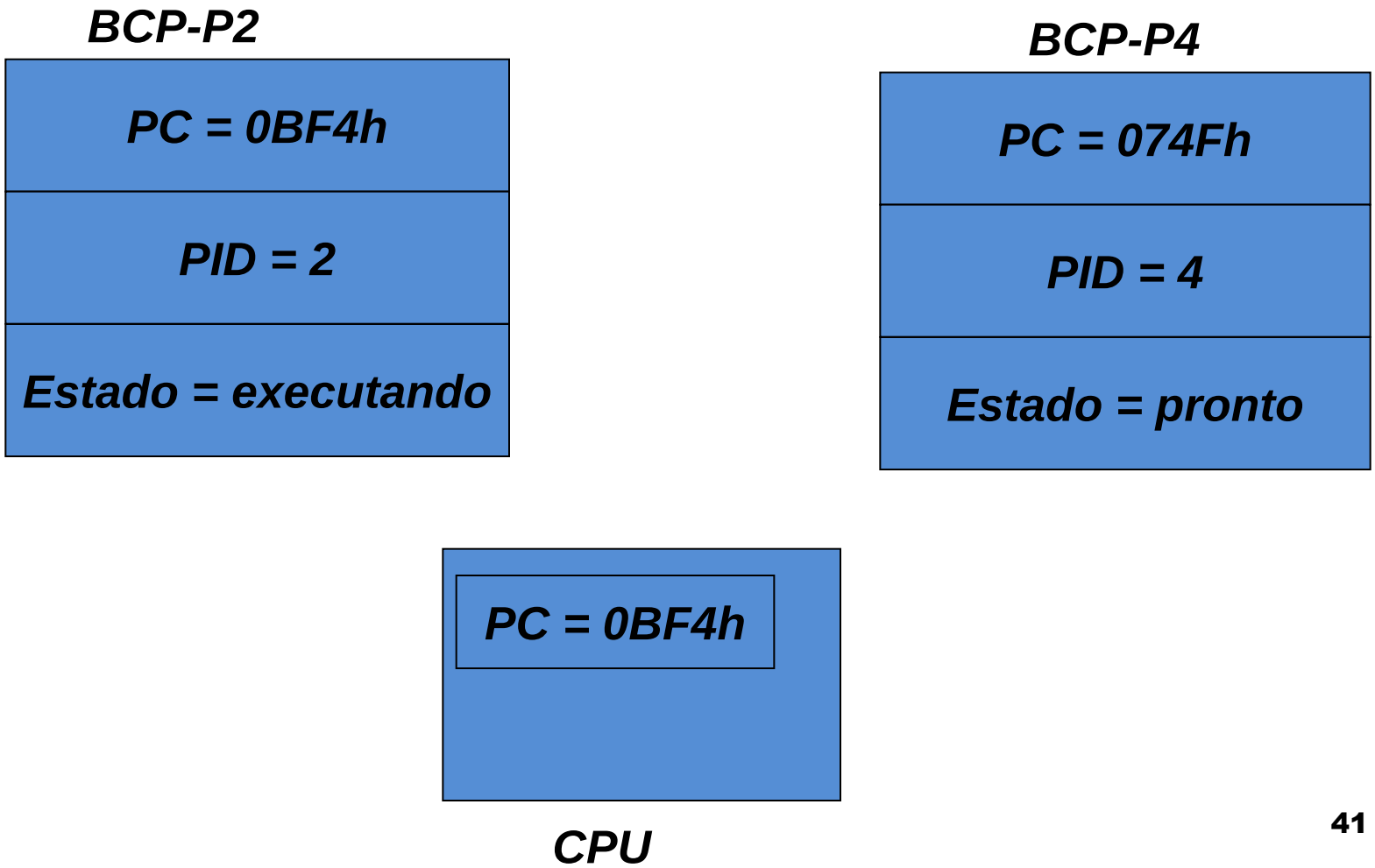
Escalonamento de Processos

Antes da Mudança de Contexto



Escalonamento de Processos

Depois da Mudança de Contexto



Escalonamento de Processos

- Situações nas quais escalonamento é necessário:
  - Um novo processo é criado;
  - Um processo terminou sua execução e um processo pronto deve ser executado;
  - Quando um processo é bloqueado (semáforo, dependência de E/S), outro deve ser executado;
  - Quando uma interrupção de E/S ocorre o escalonador deve decidir por: executar o processo que estava esperando esse evento; continuar executando o processo que já estava sendo executado ou executar um terceiro processo que esteja pronto para ser executado;

Escalonamento de Processos

- Tempo de execução de um processo é imprevisível:
  → CPU gera interrupções em intervalos entre 50 a 60 hz (ocorrências por segundo);
- Algoritmos de escalonamento podem ser divididos em duas categorias dependendo de como essas interrupções são tratadas:
  - Preemptivo: estratégia de suspender o processo sendo executado;
  - Não-preemptivo: estratégia de permitir que o processo sendo executado continue sendo executado até ser bloqueado por alguma razão (semáforos, operações de E/S-interrupção);

Escalonamento de Processos

- Categorias de Ambientes:
  - **Sistemas em Batch**: usuários não esperam por respostas rápidas; algoritmos preemptivos ou não-preemptivos;
  - **Sistemas Interativos**: interação constante do usuário; algoritmos preemptivos; Processo interativo → espera comando e executa comando;
  - **Sistemas em Tempo Real**: processos são executados mais rapidamente; tempo é crucial → sistemas críticos;

Escalonamento de Processos

- Características de algoritmos de escalonamento:
  - Qualquer sistema:
    - **Justiça** (*Fairness*)*:* cada processo deve receber uma parcela justa de tempo da CPU;
    - **Balanceamento**: diminuir a ociosidade do sistema;
    - **Políticas do sistema** – prioridade de processos;

Escalonamento de Processos

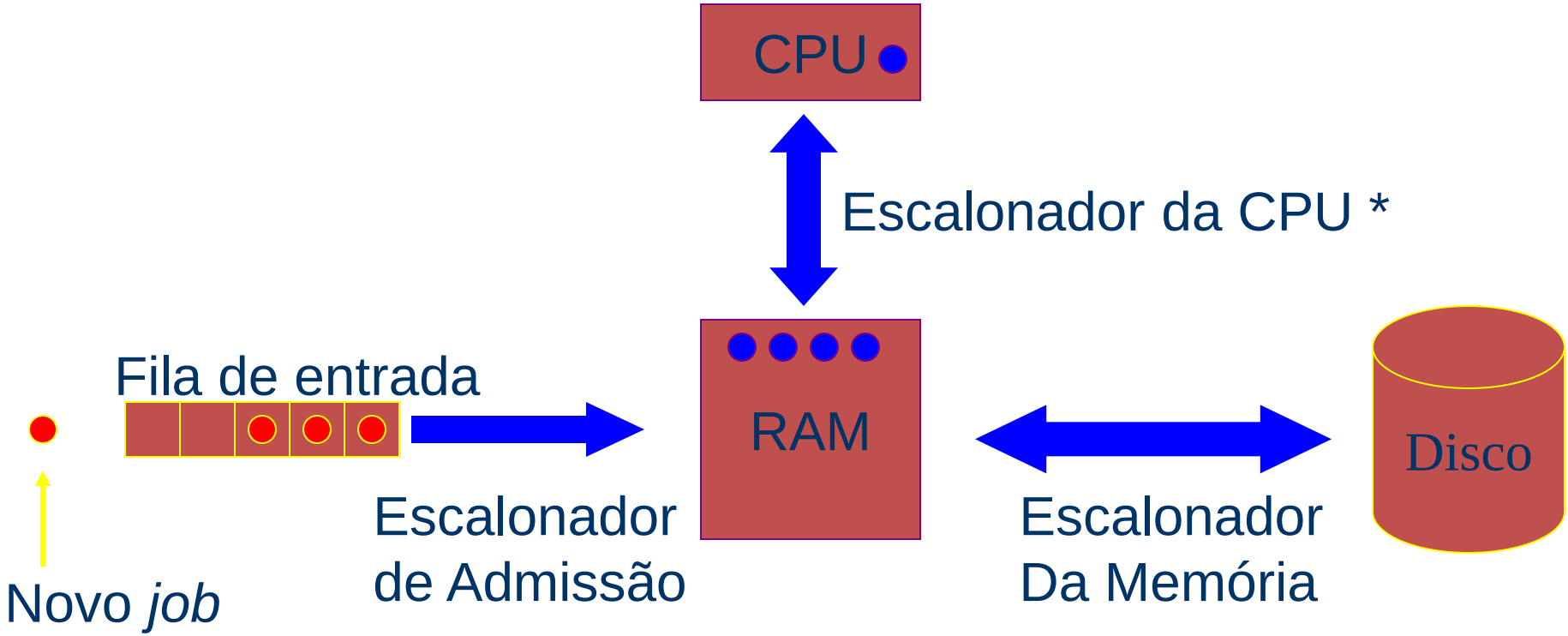
- Características de algoritmos de escalonamento:
  - Sistemas em *Batch*:
    - **Taxa de execução** (*throughput*): máximo número de *jobs* executados por hora;
    - ***Turnaround time*** (tempo de retorno): tempo no qual o processo espera para ser finalizado;
    - **Tempo de espera**: tempo gasto na fila de prontos;
    - **Eficiência**: CPU deve estar 100% do tempo ocupada;
  - Sistemas Interativos:
    - **Tempo de resposta**: tempo esperando para iniciar execução;
    - **Proporcionalidade:** Satisfação do usuários;

Escalonamento de Processos

- Características de algoritmos de escalonamento:
  - Sistemas em Tempo Real:
    - **Prevenir perda de dados**;
    - **Previsibilidade**: prevenir perda da qualidade dos serviços oferecidos;

Escalonamento de Processos
Sistemas em *Batch*

- Escalonamento *Three-Level*



Escalonamento de Processos
Sistemas em *Batch*

- Escalonamento *Three-Level*
  - **Escalonador de admissão**: processos menores primeiro; processos com menor tempo de acesso à CPU e maior tempo de interação com dispositivos de E/S;
  - **Escalonador da Memória**: decisões sobre quais processos vão para a MP:
    - A quanto tempo o processo está esperando?
    - Quanto tempo da CPU o processo já utilizou?
    - Qual o tamanho do processo?
    - Qual a importância do processo?
  - **Escalonador da CPU**: seleciona qual o próximo processo a ser executado;

Escalonamento de Processos
Sistemas em *Batch*

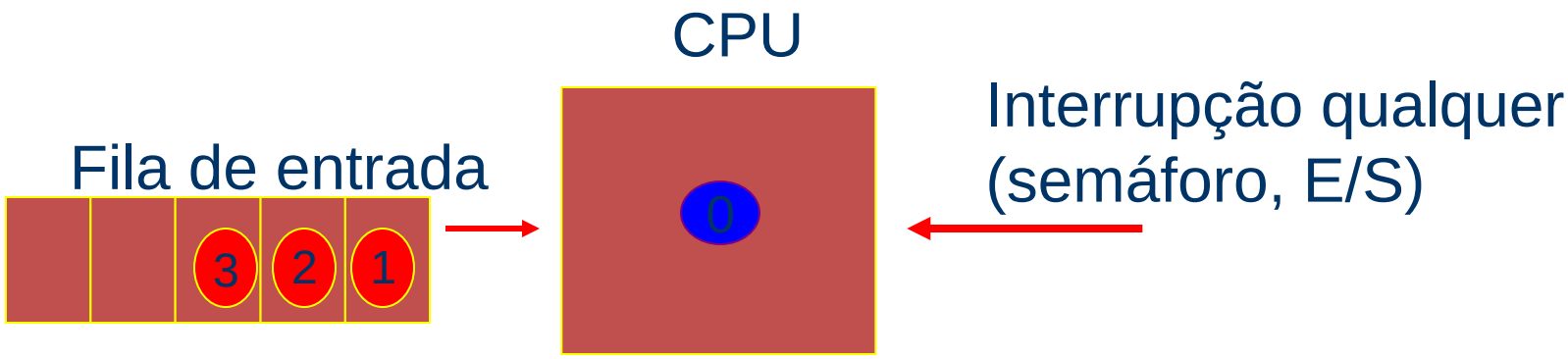
- Algoritmos para Sistemas em *Batch*:
  - *First-Come First-Served (ou FIFO)*;
  - *Shortest Job First (SJF)*;
  - *Shortest Remaining Time Next (SRTN)*;

Escalonamento de Processos Sistemas em *Batch*

- Algoritmo *First-Come First-Served*
  - Não-preemptivo;
  - Processos são executados na CPU seguindo a ordem de requisição;
  - Fácil de entender e programar;
  - Desvantagem:
    - Ineficiente quando se tem processos que demoram na sua execução;

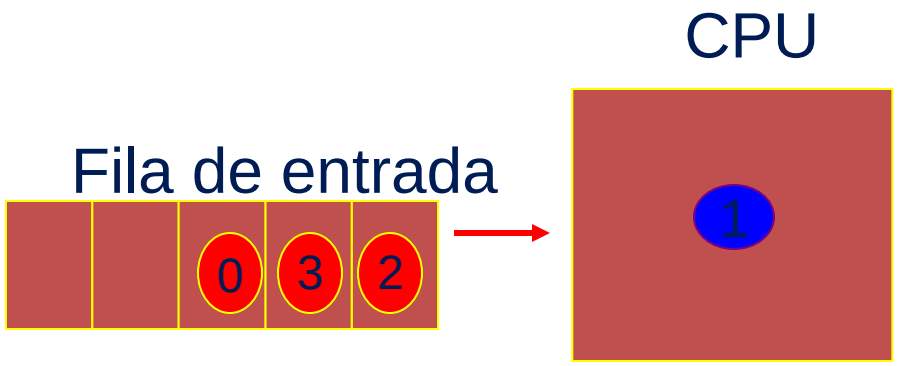
Escalonamento de Processos
Sistemas em *Batch*

Algoritmo *First-Come First-Served*



Escalonamento de Processos
Sistemas em *Batch*

Algoritmo *First-Come First-Served*

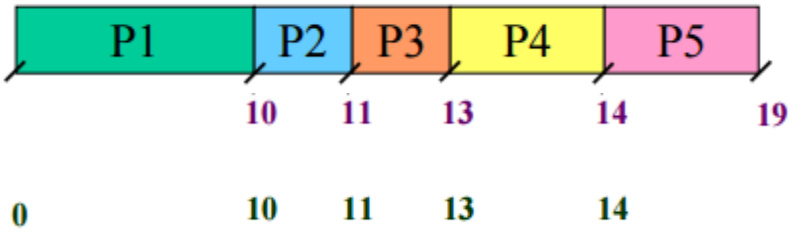


CPU não controla o tempo dos processos!
(não-preemptivo)

Escalonamento de Processos
Sistemas em *Batch*

Algoritmo *First-Come First-Served*

Calculate AVERAGE turnaround time and AVERAGE waiting time



Turnaround Time, Ta

Waiting Time, Tw

Job	Runtime [ms]
P1	10
P2	1
P3	2
P4	1
P5	5

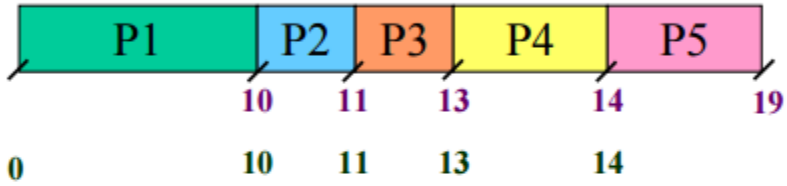
Average Ta :

Average Tw

Escalonamento de Processos
Sistemas em *Batch*

Algoritmo *First-Come First-Served*

Calculate AVERAGE turnaround time and AVERAGE waiting time



Job	Runtime [ms]
P1	10
P2	1
P3	2
P4	1
P5	5

Average Ta = Total Ta / Total no of jobs

Average Ta = (10 +11 + 13 + 14 + 19) / 5 = **13.4 ms**

Average Tw = Total Tw / Total no of jobs

Average Tw = (0 +10 + 11 + 13 + 14) / 5 = **9.6 ms**

Escalonamento de Processos
Sistemas em *Batch*

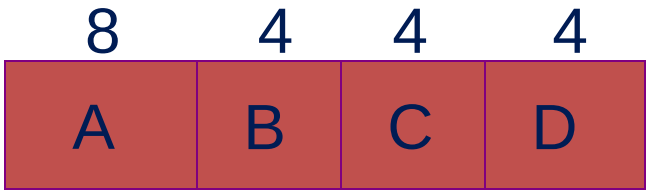
- Algoritmo *Shortest Job First*
  - Não-preemptivo;
  - Possível prever o tempo de execução do processo;
  - Menor processo é executado primeiro;
  - Menor *turnaround*;
  - Desvantagem:
    - Baixo aproveitamento quando se tem poucos processos prontos para serem executados;

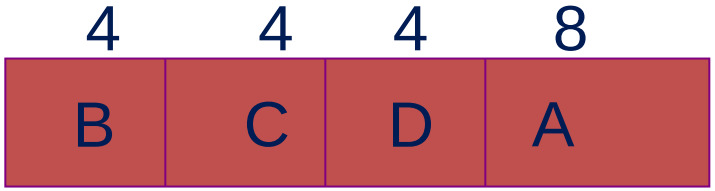
Escalonamento de Processos
Sistemas em *Batch*

- Algoritmo *Shortest Job First*

Escalonamento de Processos
Sistemas em *Batch*

Algoritmo *Shortest Job First*





Em ordem:
Turnaround A = 8
Turnaround B = 12
Turnaround C = 16
Turnaround D = 20
Turnaround time → 56/4 = 14

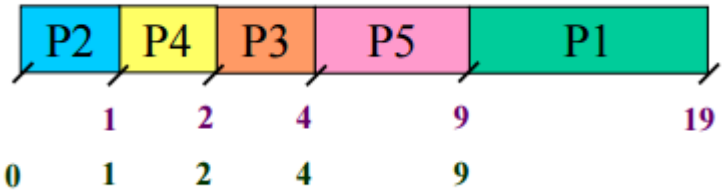
Menor *job* primeiro:
Turnaround B = 4
Turnaround C = 8
Turnaround D = 12
Turnaround A = 20
Turnaround time → 44/4 = 11

$(4a+3b+2c+d)/4$ ← Número de Processos

Escalonamento de Processos
Sistemas em *Batch*

Algoritmo *Shortest Job First*

Calculate AVERAGE turnaround time and AVERAGE waiting time



Turnaround Time, Ta

Waiting Time, Tw

Job	Runtime [ms]
P1	10
P2	1 *
P3	2
P4	1 *
P5	5

* FCFS for jobs with the same runtime

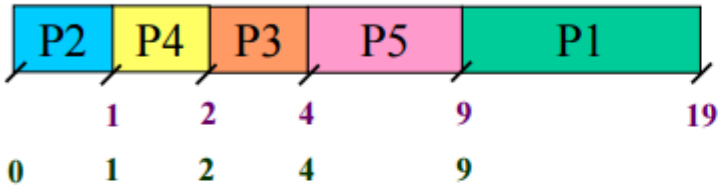
Average Ta =

Average Tw =

Escalonamento de Processos
Sistemas em *Batch*

Algoritmo *Shortest Job First*

Calculate AVERAGE turnaround time and AVERAGE waiting time



Turnaround Time, Ta

Waiting Time, Tw

Job	Runtime [ms]
P1	10
P2	1 *
P3	2
P4	1 *
P5	5

* FCFS for jobs with the same runtime

Average Ta = Total Ta / Total no of jobs

Average Ta = (1 + 2 + 4 + 9 + 19) / 5 = **7 ms**

Average Tw = Total Tw / Total no of jobs

Average Tw = (0 +1 + 2 + 4 + 9) / 5 = **3.2 ms**

Escalonamento de Processos
Sistemas em *Batch*

Algoritmo *Shortest Job First – Non-simultaneous arrival time*

Calculate AVERAGE turnaround time and AVERAGE waiting time

Job	Runtime [ms]	Arrival
P1	10	0
P2	1	0
P3	2	3
P4	1	3
P5	5	3

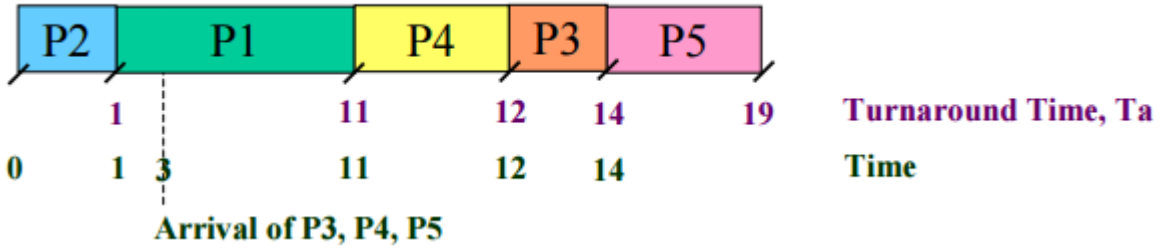
Average Ta =

Average Tw =

Escalonamento de Processos
Sistemas em *Batch*

Algoritmo *Shortest Job First – Non-simultaneous arrival time*

Calculate AVERAGE turnaround time and AVERAGE waiting time



Job	Runtime [ms]	Arrival
P1	10	0
P2	1	0
P3	2	3
P4	1	3
P5	5	3

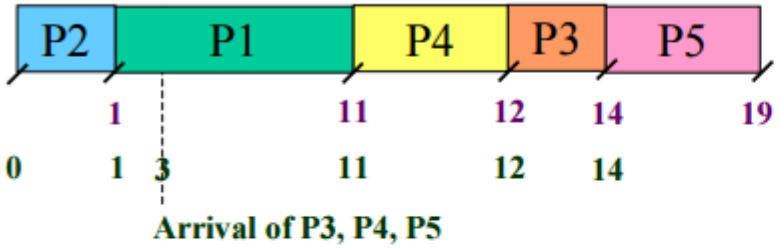
Average Ta =

Average Tw =

Escalonamento de Processos Sistemas em *Batch*

Algoritmo *Shortest Job First – Non-simultaneous arrival time*

Calculate AVERAGE turnaround time and AVERAGE waiting time



Job	Runtime [ms]	Arrival
P1	10	0
P2	1	0
P3	2	3
P4	1	3
P5	5	3

Average Ta = Total Ta / Total no of jobs

Average Ta = (1 + 11 + 12 + 14 + 19) / 5 = **11.4 ms**

Average Tw = Total Tw / Total no of jobs

Average Tw = (0 +1 + 8 + 9 + 11) / 5 = **5.8 ms**

Escalonamento de Processos Sistemas em *Batch*

- Algoritmo *Shortest Remaining Time Next*
  - Preemptivo;
  - Processos com menor tempo de execução são executados primeiro;
  - Se um processo novo chega e seu tempo de execução é menor do que do processo corrente na CPU, a CPU suspende o processo corrente e executa o processo que acabou de chegar;
  - Desvantagem: processos que consomem mais tempo podem demorar muito para serem finalizados se muitos processos pequenos chegarem!

Escalonamento de Processos
Sistemas Interativos

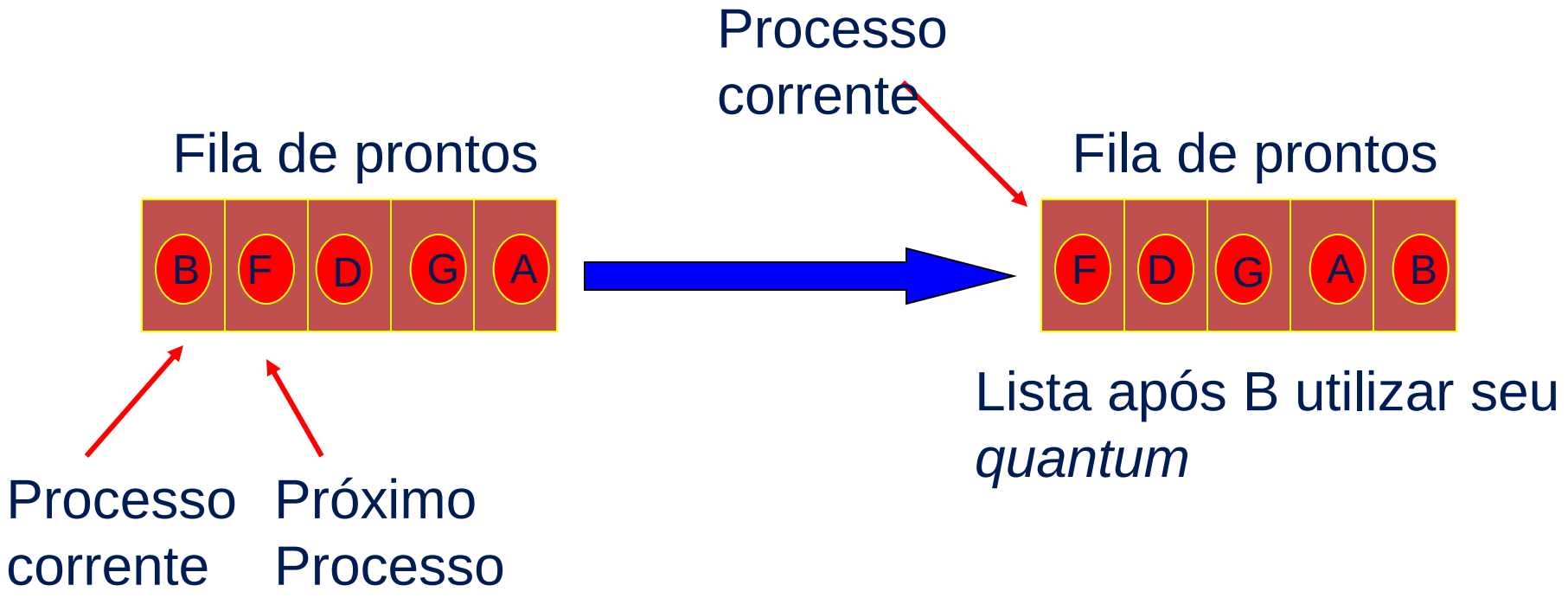
- Algoritmos para Sistemas Interativos:
  - *Round-Robin*;
  - Prioridade;
  - Múltiplas Filas;
  - *Shortest Process Next*;
  - Garantido;
  - *Lottery*;
  - *Fair-Share;*
- Utilizam escalonamento em dois níveis (escalonador da CPU e memória);

Escalonamento de Processos Sistemas Interativos

- Algoritmo *Round-Robin*
  - Antigo, mais simples e mais utilizado;
  - Preemptivo;
  - Cada processo recebe um tempo de execução chamado *quantum*; ao final desse tempo, o processo é suspenso e outro processo é colocado em execução;
  - Escalonador mantém uma lista de processos prontos;

Escalonamento de Processos
Sistemas Interativos

Algoritmo *Round-Robin*



Escalonamento de Processos Sistemas Interativos

- Algoritmo *Round-Robin*
  - Tempo de chaveamento de processos;
  - *quantum*: se for muito pequeno, ocorrem muitas trocas diminuindo, assim, a eficiência da CPU; se for muito longo o tempo de resposta é comprometido;

Escalonamento de Processos
Sistemas Interativos

- Algoritmo *Round-Robin:*

Exemplos:

Δt = 4 mseg

x = 1mseg

quantum

→ 25% de tempo de CPU é perdido → menor eficiência

Δt = 100 mseg

x = 1mseg

→ 1% de tempo de CPU é perdido → Tempo de espera dos processos é maior

chaveamento

Escalonamento de Processos
Sistemas Interativos

- Algoritmo *Round-Robin:*

Exemplos:

$\Delta t = 4$ mseg

x = 1mseg → 25% de tempo de CPU é perdido → menor eficiência

$\Delta t = 100$ mseg

x = 1mseg → 1% de tempo de CPU é perdido → Tempo de espera dos processos é maior

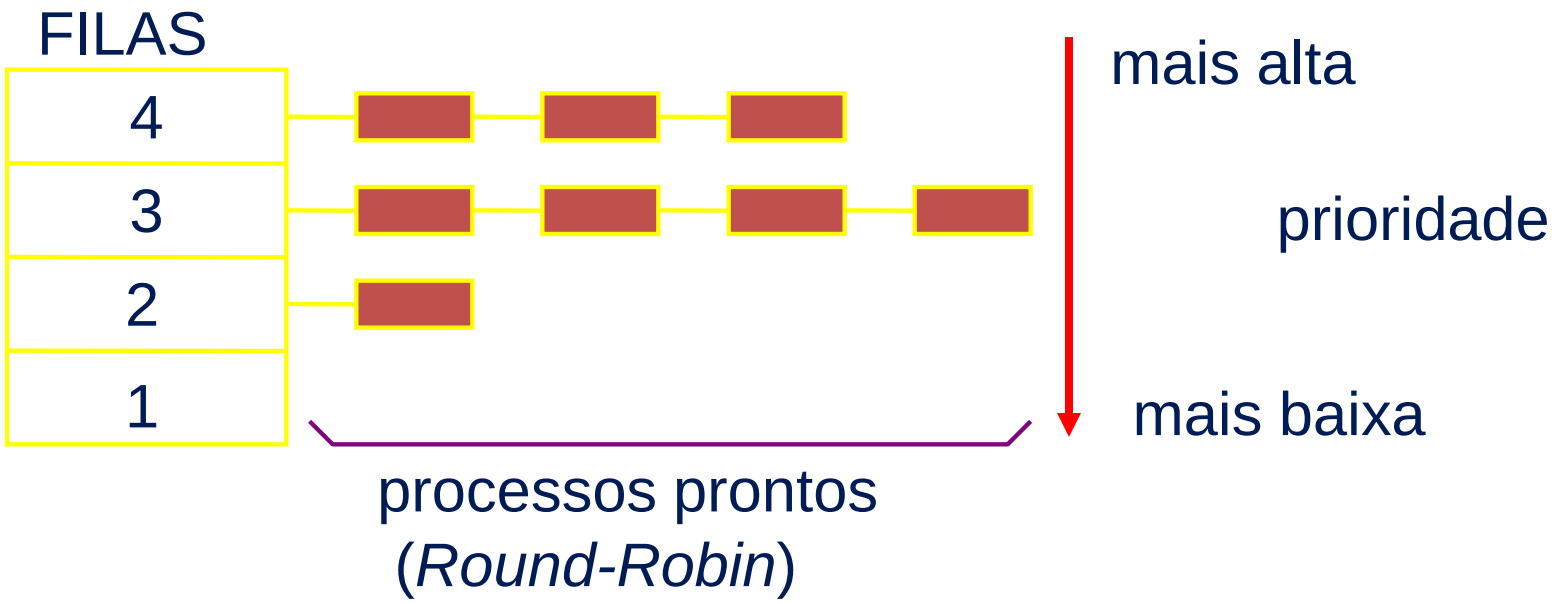
quantum razoável: 20-50 mseg

Escalonamento de Processos Sistemas Interativos

- Algoritmo com Prioridades
  - Cada processo possui uma prioridade → os processos prontos com maior prioridade são executados primeiro;
  - Prioridades são atribuídas dinâmica ou estaticamente;
  - Classes de processos com mesma prioridade;
  - Preemptivo;

Escalonamento de Processos
Sistemas Interativos

- Algoritmo com Prioridades



Escalonamento de Processos Sistemas Interativos

- Algoritmo com Prioridades
  - Como evitar que os processos com maior prioridade sejam executado indefinidamente?
    - Diminuir a prioridade do processo corrente e troca-lo pelo próximo processo com maior prioridade (chaveamento);
    - Cada processo possui um *quantum*;

Escalonamento de Processos Sistemas Interativos

- Múltiplas Filas:
  - CTSS (*Compatible Time Sharing System*);
  - Classes de prioridades;
  - Cada classe de prioridades possui *quanta* diferentes;
  - Assim, a cada vez que um processo é executado e suspenso ele recebe mais tempo para execução;
  - Preemptivo;

Escalonamento de Processos Sistemas Interativos

- Múltiplas Filas:
  - Ex.: um processo precisa de 100 *quanta* para ser executado;
    - Inicialmente, ele recebe um *quantum* para execução;
    - Das próximas vezes ele recebe, respectivamente, 2, 4, 8, 16, 32 e 64 *quanta* (7 chaveamentos) para execução;
    - Quanto mais próximo de ser finalizado, menos frequente é o processo na CPU → eficiência

Escalonamento de Processos Sistemas Interativo

- Algoritmo *Shortest Process Next*
  - Mesma idéia do *Shortest Job First*;
  - Processos Interativos: não se conhece o tempo necessário para execução;
  - Como empregar esse algoritmo: ESTIMATIVA de TEMPO!

Escalonamento de Processos Sistemas Interativo

- Outros algoritmos:
  - Algoritmo Garantido:
    - Garantias são dadas aos processos dos usuários:
      - **n** usuários → **1/n** do tempo de CPU para cada usuário;
  - Algoritmo *Lottery*:
    - Cada processo recebe "*tickets*" que lhe dão direito de execução;
  - Algoritmo *Fair-Share*:
    - O dono do processo é levado em conta;
    - Se um usuário A possui mais processos que um usuário B, o usuário A terá prioridade no uso da CPU;

Escalonamento de Processos Sistemas em Tempo Real

- Tempo é um fator crítico;
- Sistemas críticos:
  - Aviões;
  - Hospitais;
  - Usinas Nucleares;
  - Bancos;
  - Multimídia;
- Ponto importante: obter respostas em atraso e tão ruim quanto não obter respostas;

Escalonamento de Processos Sistemas em Tempo Real

- Tipos de STR:
  - ***Hard Real Time***: atrasos não são tolerados;
    - Aviões, usinas nucleares, hospitais;
  - ***Soft Real Time***: atrasos são tolerados;
    - Bancos; Multimídia;
- Programas são divididos em vários processos;
- Eventos causam a execução de processos:
  - **Periódicos**: ocorrem em intervalos regulares de tempo;
  - **Aperiódicos**: ocorrem em intervalos irregulares de tempo;

Escalonamento de Processos Sistemas em Tempo Real

- Algoritmos podem ser estáticos ou dinâmicos;
  - **Estáticos**: decisões de escalonamento antes do sistema começar;
    - Informação disponível previamente;
  - **Dinâmicos**: decisões de escalonamento em tempo de execução;