Sistemas Operacionais

Prof. Jó Ueyama

Apresentação baseada nos slides da Profa. Kalinka Castelo Branco, do Prof. Dr. Antônio Carlos Sementille e da Profa. Dra. Luciana A. F. Martimiano e nas transparências fornecidas no site de compra do livro "Sistemas Operacionais Modernos"

Aula de Hoje (conteúdo detalhado)

1. **Conceitos Básicos - Chamadas de Sistemas**
2. **Processos**
3. **Criando Processos**
4. **Finalizando Processos**
5. **Estados do Processo**
6. **Implementação de Processos**
7. **Escalonamento de Processo**

Processos

- Multiprogramação:
  - Pseudoparalelismo: coleção de processos sendo executados alternadamente na CPU;
- Um processo é caracterizado por um programa em execução, mas existe uma diferença sutil entre processo e programa:
  - Um processo pode ser composto por vários programas, dados de entrada, dados de saída e um estado (executando, bloqueado, pronto)

Aula de Hoje (conteúdo detalhado)

1. **Conceitos Básicos - Chamadas de Sistemas**
2. **Processos**
3. **Criando Processos**
4. **Finalizando Processos**
5. **Estados do Processo**
6. **Implementação de Processos**
7. **Escalonamento de Processo**

Criando Processos

- Processos precisam ser criados e finalizados a todo o momento:
  - Inicialização do sistema;
  - Execução de uma chamada de sistema para criação de processo realizada por algum processo em execução;
  - Requisição de usuário para criar um novo processo;
  - Inicialização de um processo em *batch* – *mainframes* com sistemas em *batch*;

Criando Processos

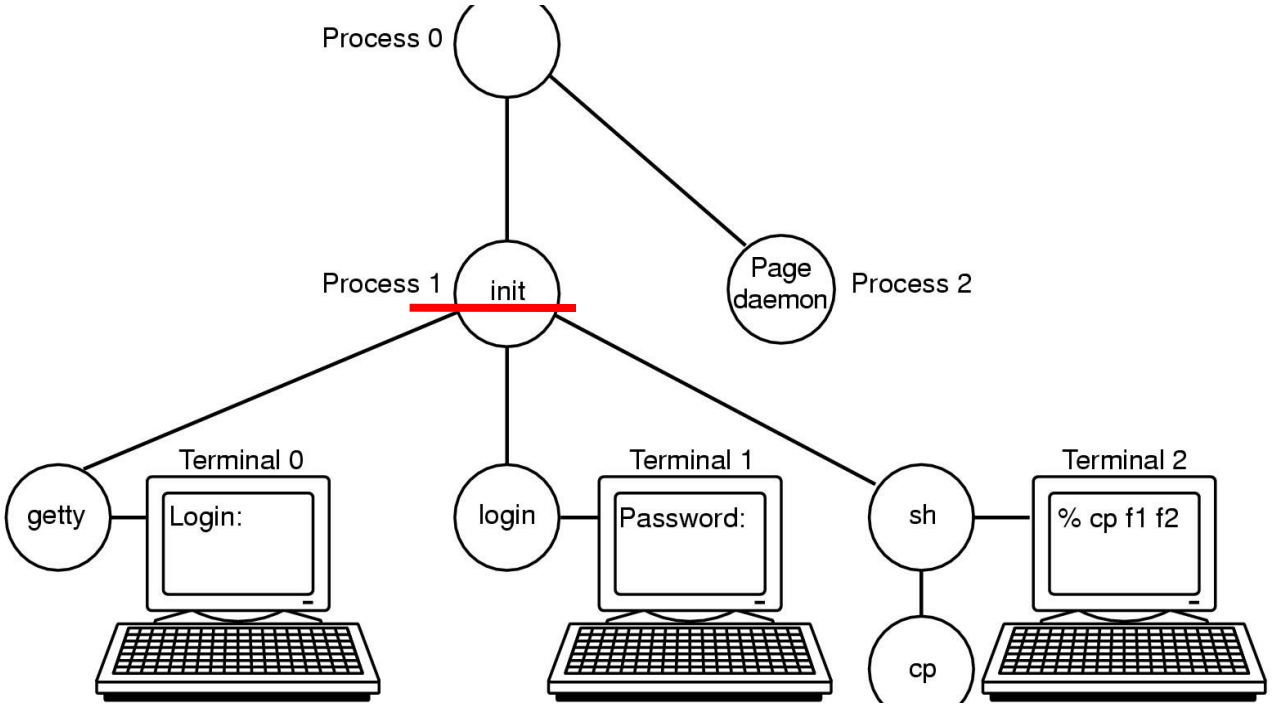
- Processos criados pelos usuários:
  - Iniciar um programa (linha de comando ou um duplo clique no mouse);
- Processos com funções específicas que independem de usuários – chamados de *daemons* (ou BG que não há interação direta com usuários):
  - Recepção e envio de emails;
  - Serviços de Impressão;
  - Usando o ‘&’ no Linux

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.

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

Aula de Hoje (conteúdo detalhado)

1. **Conceitos Básicos - Chamadas de Sistemas**
2. **Processos**
3. **Criando Processos**
4. **Finalizando Processos**
5. **Estados do Processo**
6. **Implementação de Processos**
7. **Escalonamento de Processo**

Finalizando Processos

- 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;

Finalizando Processos

- 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)*;*

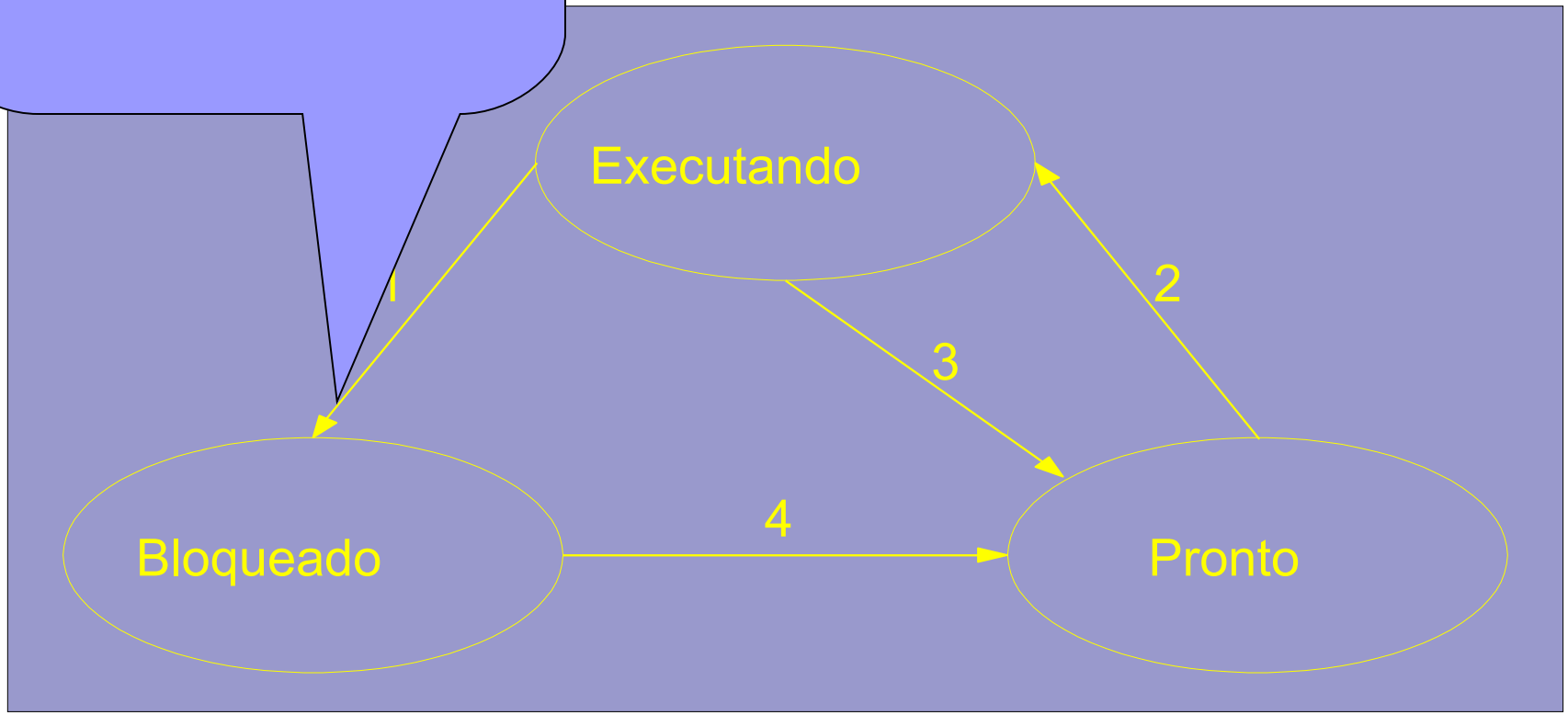
Aula de Hoje (conteúdo detalhado)

1. **Conceitos Básicos - Chamadas de Sistemas**
2. **Processos**
3. **Criando Processos**
4. **Finalizando Processos**
5. **Estados do Processo**
6. **Implementação de Processos**
7. **Escalonamento de Processo**

Processos

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

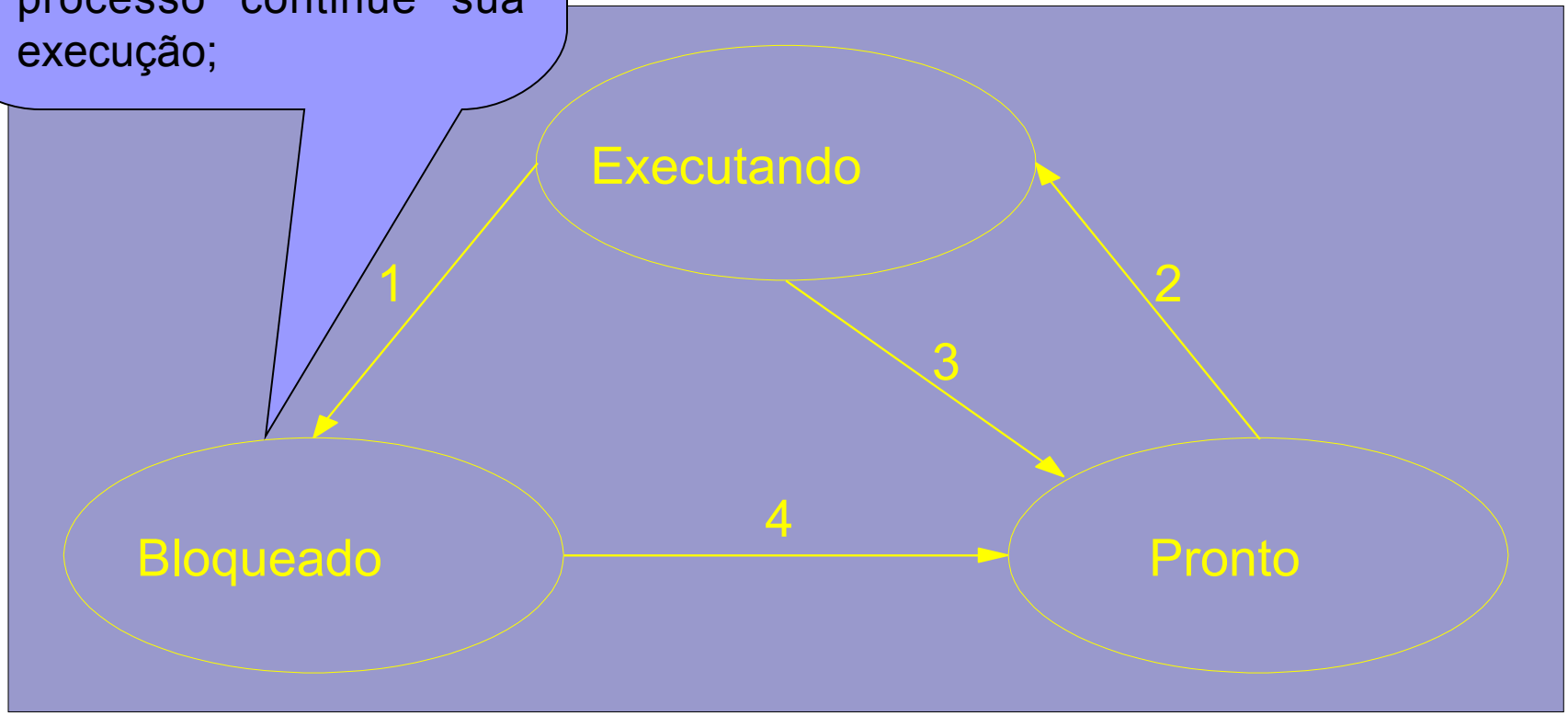
icos:



Processos

Um processo é bloqueado de duas maneiras:

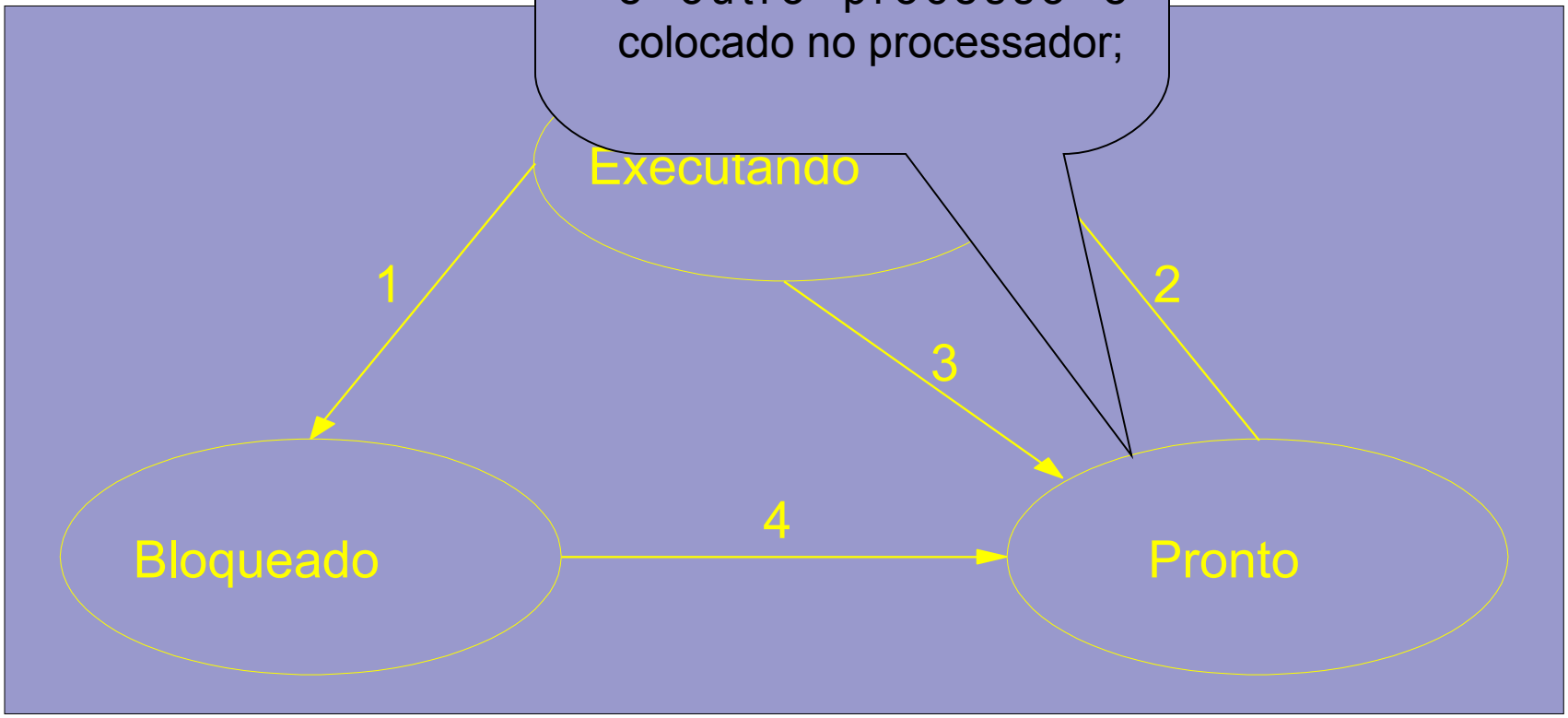
- chamada ao sistema: `block` ou `pause`;
- se não há entradas disponíveis para que o processo continue sua execução;



Estados de Processos

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

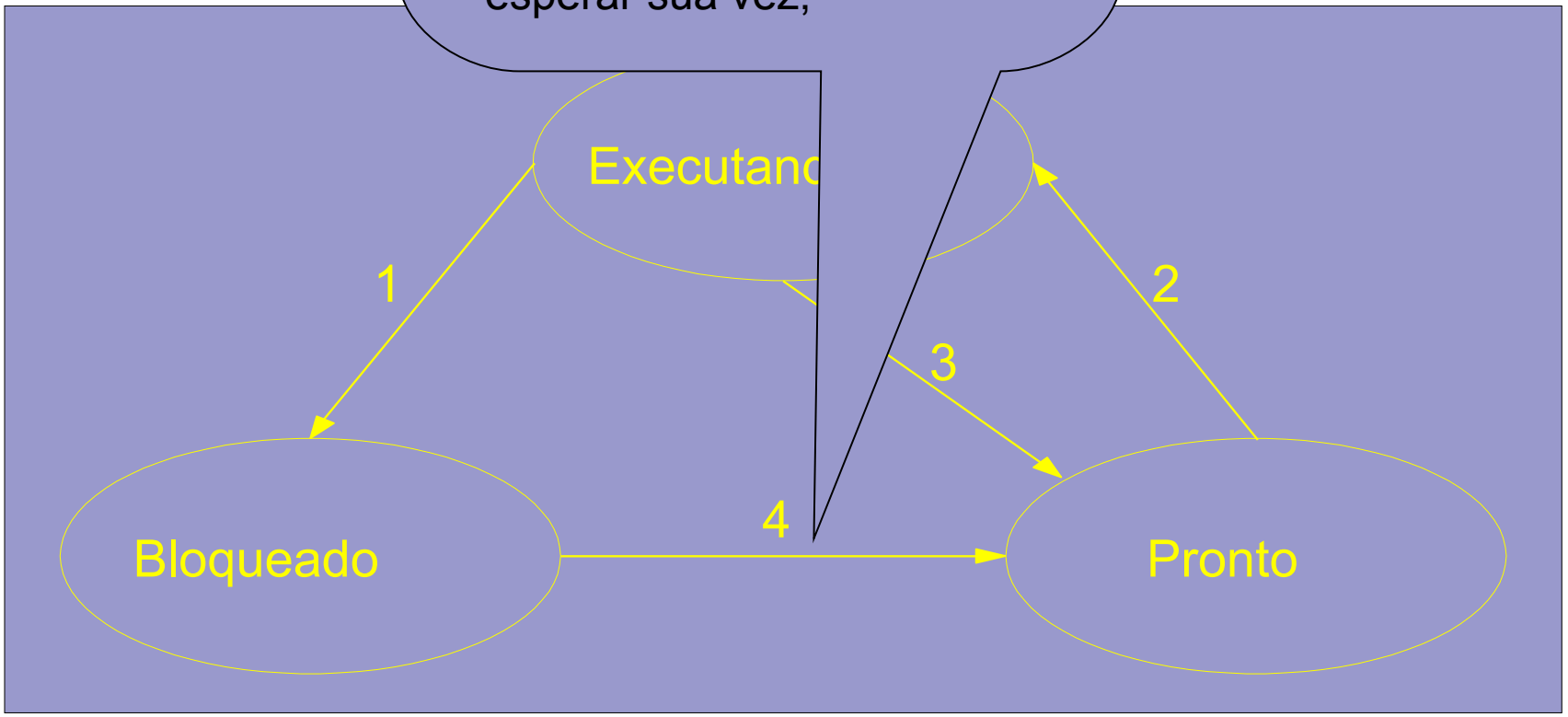
- o tempo destinado àquele processo acabou e outro processo é colocado no processador;



Estados

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;



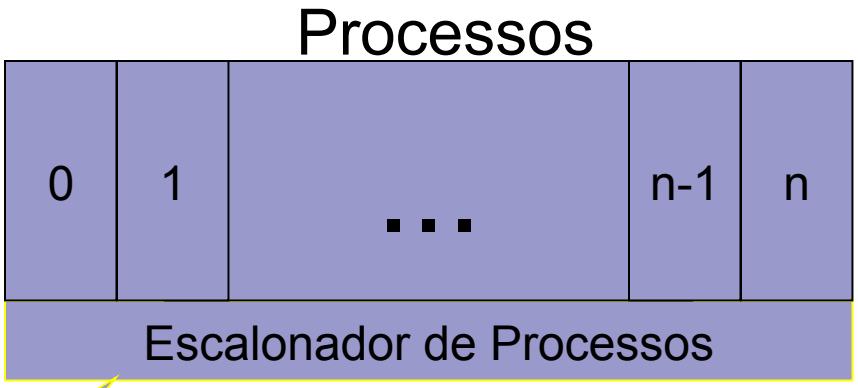
Aula de Hoje (conteúdo detalhado)

1. **Características de Processos**
2. **Implementação de Processos**
3. **Escalonamento de Processo**
4. **Escalonamento em Batch**
5. **Algoritmos de Escalonamento em Sistemas Batch**

Características - 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*;

Escalonador de Processos



- Nível mais baixo do SO;
- Manipulação de interrupções e processos;

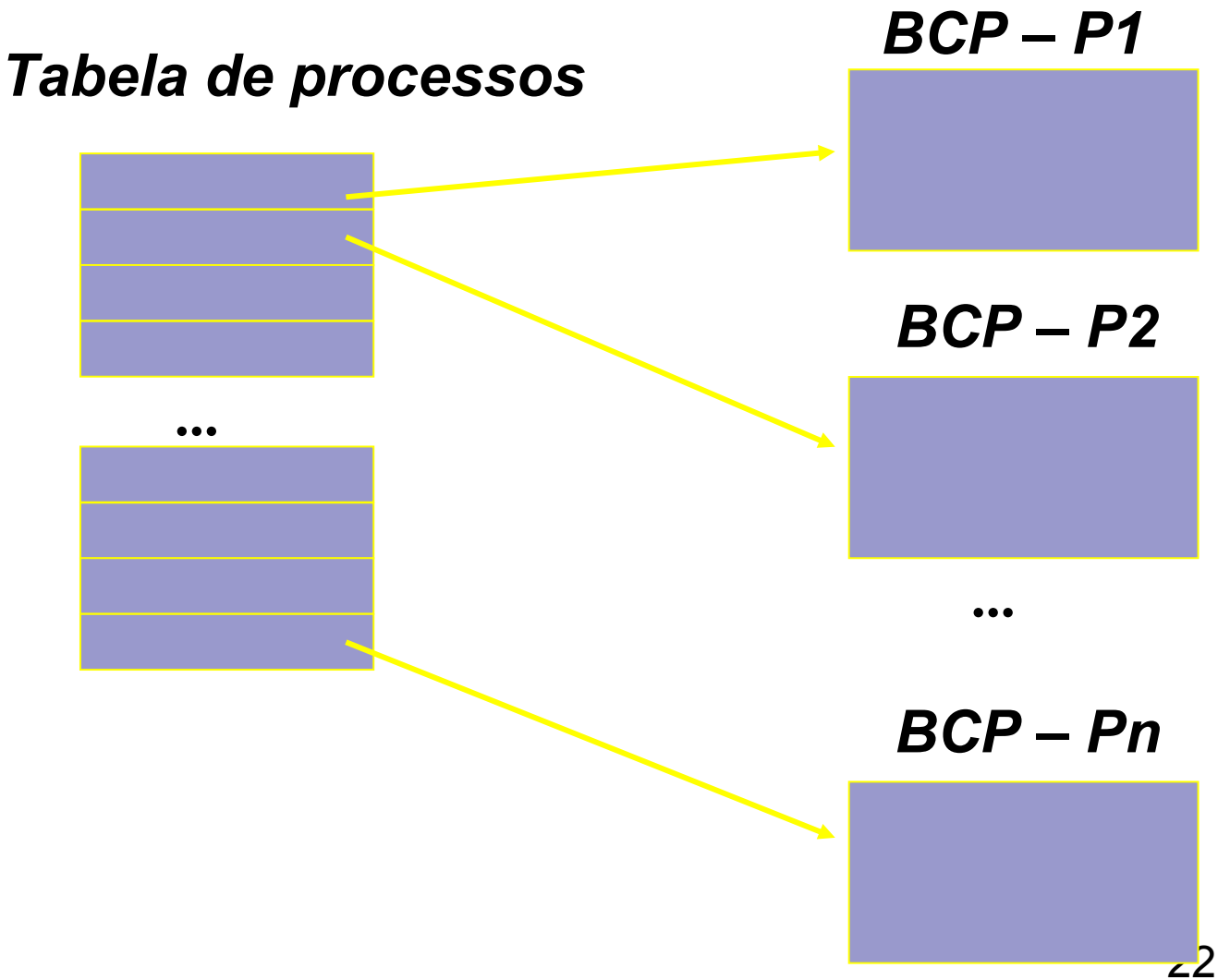
Aula de Hoje (conteúdo detalhado)

1. **Características de Processos**
2. **Implementação de Processos**
3. **Escalonamento de Processo**
4. **Escalonamento em Batch**
5. **Algoritmos de Escalonamento em Sistemas Batch**

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;
  - BCP e vetor de interrupção.

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

Aula de Hoje (conteúdo detalhado)

1. **Características de Processos**
2. **Implementação de Processos**
3. **Escalonamento de Processo**
4. **Escalonamento em Batch**
5. **Algoritmos de Escalonamento em Sistemas Batch**

Escalonamento de Processos

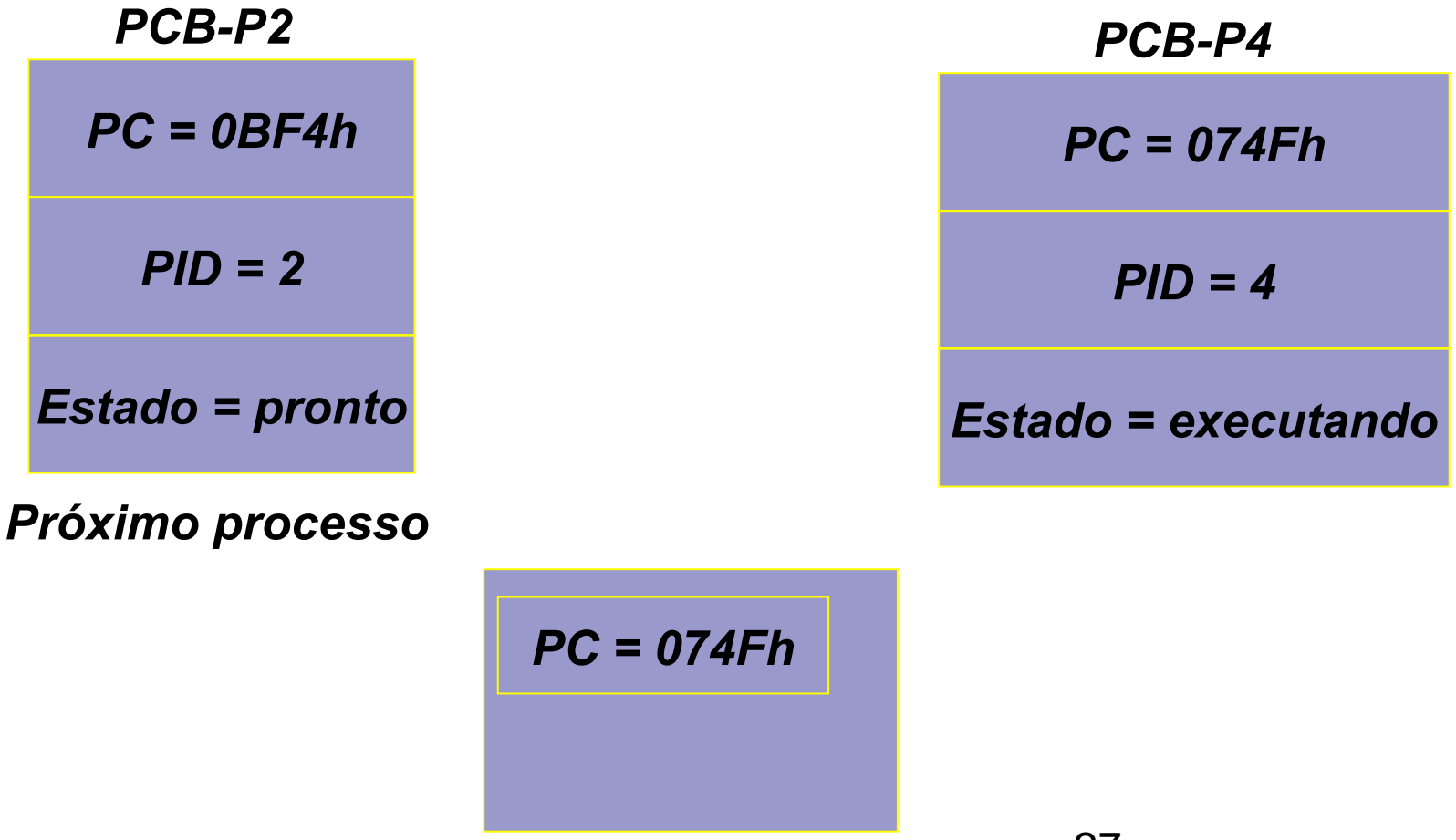
- Escalonador de Processos escolhe o processo que será executado pela CPU;
- Escalonador deve se preocupar com a eficiência da CPU, pois o chaveamento de processos é complexo e custoso:
  - Afeta desempenho do sistema e satisfação do usuário;
- Escalonador de processo é um processo que deve ser executado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

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

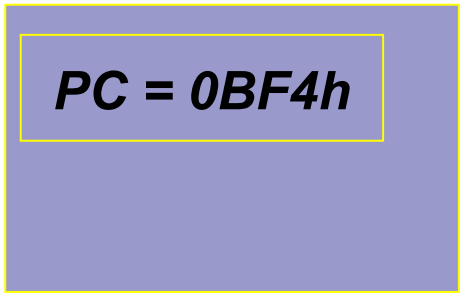
<u>Depois</u> da Mudança de Contexto

PCB-P2

PC = 0BF4h
PID = 2
Estado = executando

PCB-P4

PC = 074Fh
PID = 4
Estado = pronto



CPU

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: i) executar o processo que estava esperando esse evento; ii) continuar executando o processo que já estava sendo executado ou; iii) 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**: sistemas que processa um lote de tarefas enfileiradas e só executa o outro após o término do primeiro; 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 dentro do prazo estipulado previamente; tempo é crucial → sistemas críticos (o que são?); não-preemptivos, mas podem ser preemptivos também (e.g. desde que dentro do *deadline*)

Escalonamento de Processos

- Critérios adotados pela maioria dos 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 unidade de tempo (e.g. hora);
    - ***Turnaround time*** (tempo de retorno): tempo no qual o processo espera para ser finalizado; submissão até o fim da chegada de todo output (tipicamente, tempo de espera + tempo de execução).
    - **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;
    - **Satisfação do usuários**; QoE (*Quality of Experience*).

Escalonamento de Processos

- Características de algoritmos de escalonamento:
  - Sistemas em Tempo Real:
    - **Prevenir perda de dados** (e.g. aplicações de multimídia)
    - **Previsibilidade**: prevenir perda da qualidade dos serviços oferecidos (e.g. a consistência na transmissão é melhor do que a rajada)
      - Diferentemente das aplicações elásticas

Perguntas?