

Aula 04 – Parte 1

Sistemas Operacionais I

Processos

Prof. Julio Cezar Estrella

jcezar@icmc.usp.br

Material adaptado de

Sarita Mazzini Bruschi

baseados no livro Sistemas Operacionais Modernos de A. Tanenbaum

Processos

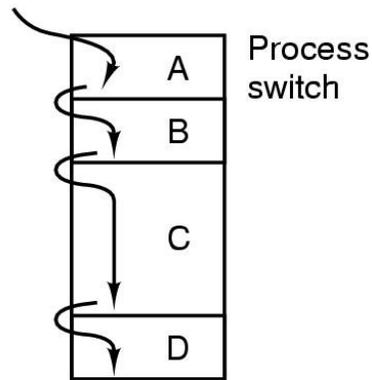
- **Introdução**
- Escalonamento de Processos
- Comunicação entre Processos
- Threads
- Deadlock

Processos

- Multiprogramação:

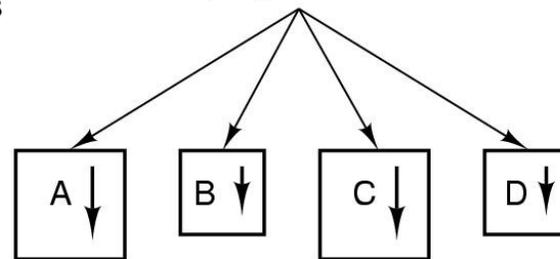
- Pseudoparalelismo: coleção de processos sendo executados alternadamente no CPU.

One program counter

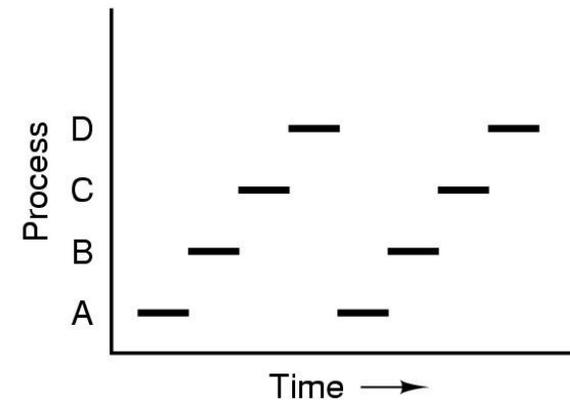


(a)

Four program counters



(b)



(c)

- (a) Modelo de multiprogramação com 4 processos na memória.
- (b) Modelo conceitual de 4 processos sequenciais e independentes.
- (c) Somente um programa está ativo de cada vez.

Processos

- 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)

Criando Processos

- Processos precisam ser criados e finalizados a todo o momento:
 - Inicialização do sistema;
 - Execução de uma chamada ao sistema de 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

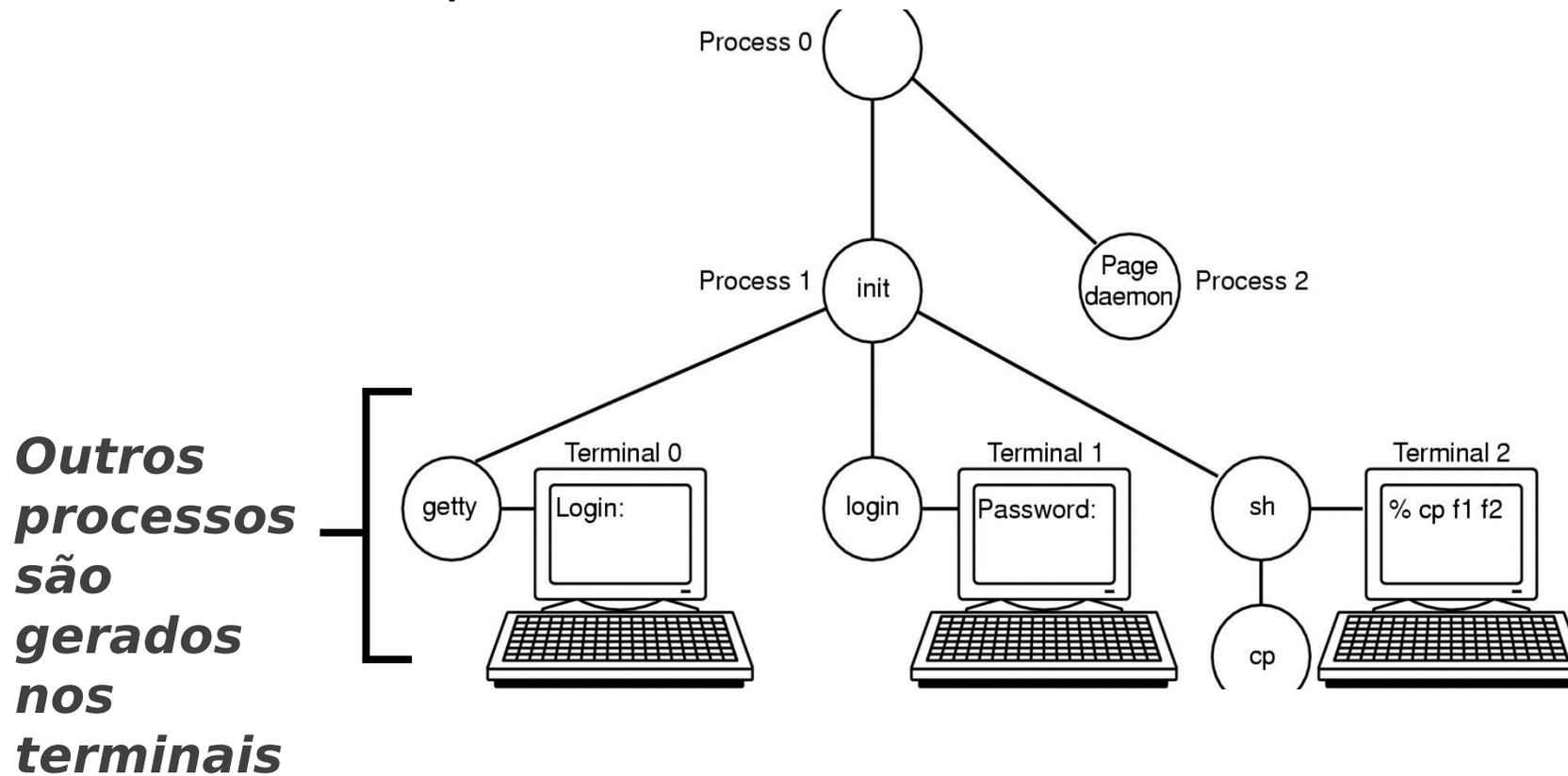
- Processos podem ser:
 - Específicos para usuários específicos:
 - Leitura de um arquivo;
 - Iniciar um programa (linha de comando ou um duplo clique no mouse);
 - Com funções específicas, que independem de usuários, que são criados pelo sistema operacional e que são processados em segundo plano (*daemons*):
 - Recepção e envio de emails;
 - Serviços de Impressão;

Criando Processos

- UNIX:
 - Fork;
 - Cria um processo idêntico (filho) ao processo que a chamou (pai), possuindo a mesma imagem de memória, as mesmas cadeias de caracteres no ambiente e os mesmos arquivos abertos;
 - Depois, o processo filho executa uma chamada para mudar sua imagem de memória e executar um novo programa
- Windows:
 - CreateProcess
 - Uma única função trata tanto do processo de criação quanto da carga do programa correto no novo processo

Criando Processos

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



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;

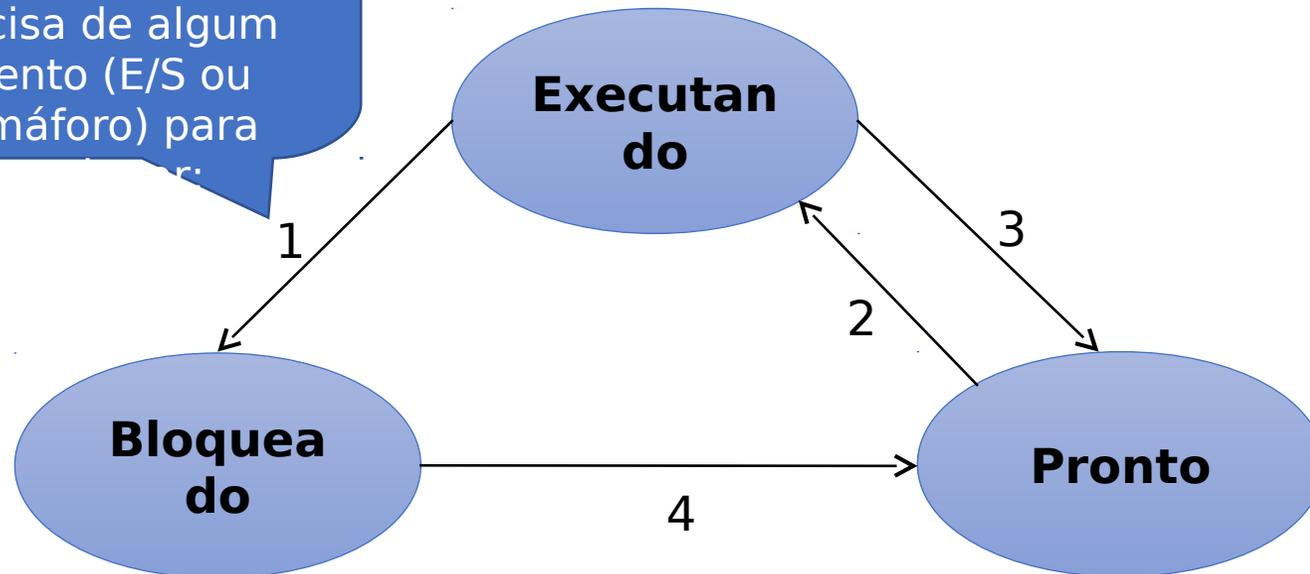
Finalizando Processos

- Condições (continuação):
 - 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);

Estados de Processos

- Os processos possuem 3 estados

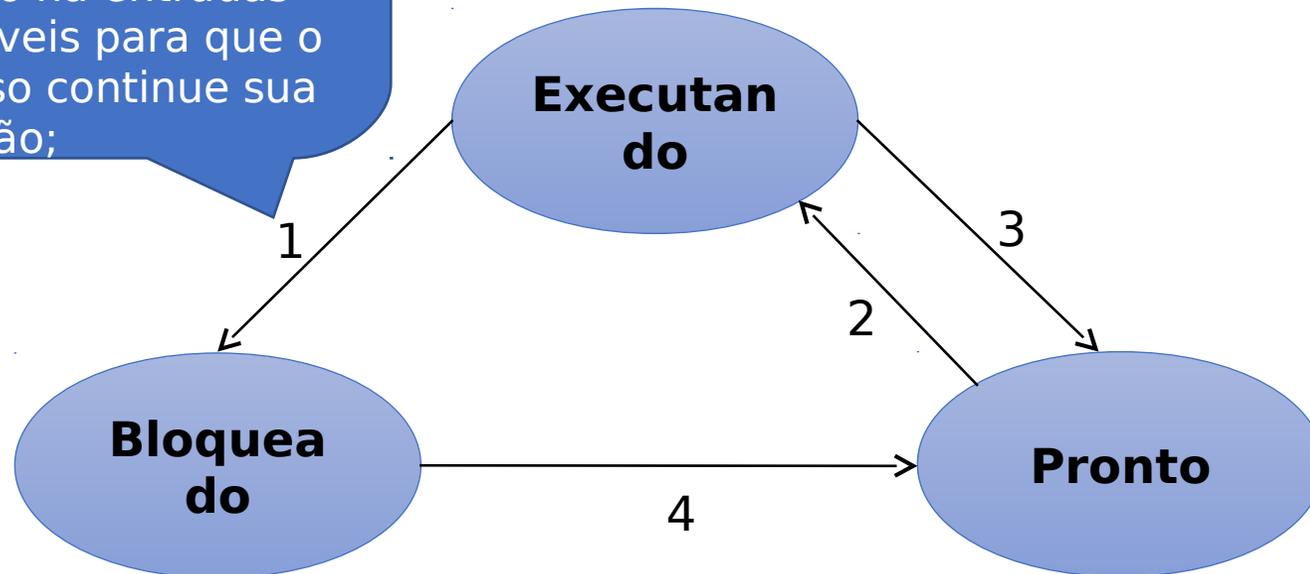
Um processo que não executado não pode continuar sua execução, pois precisa de algum evento (E/S ou semáforo) para



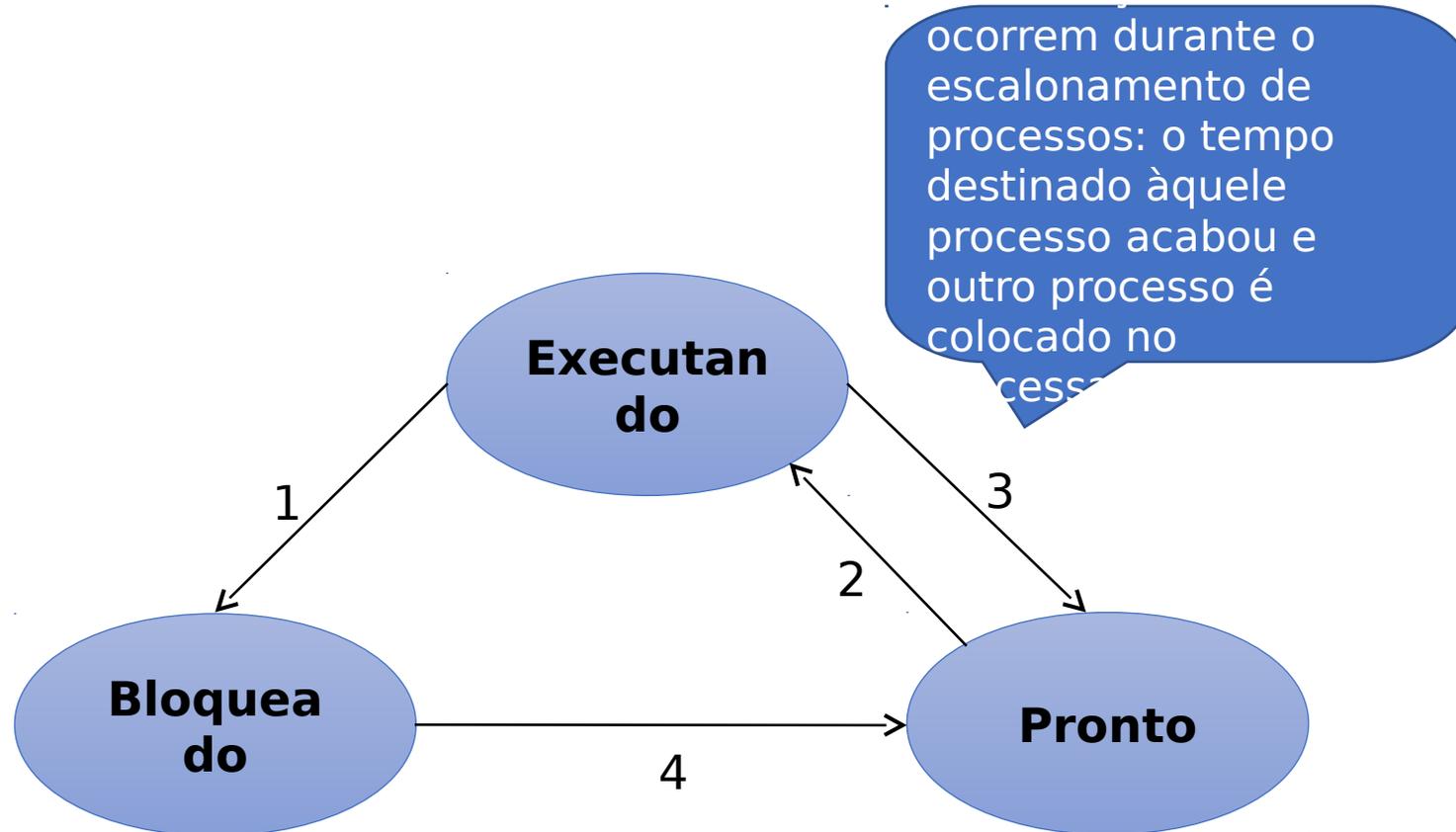
Estados de Processos

Um processo é bloqueado de duas maneiras:

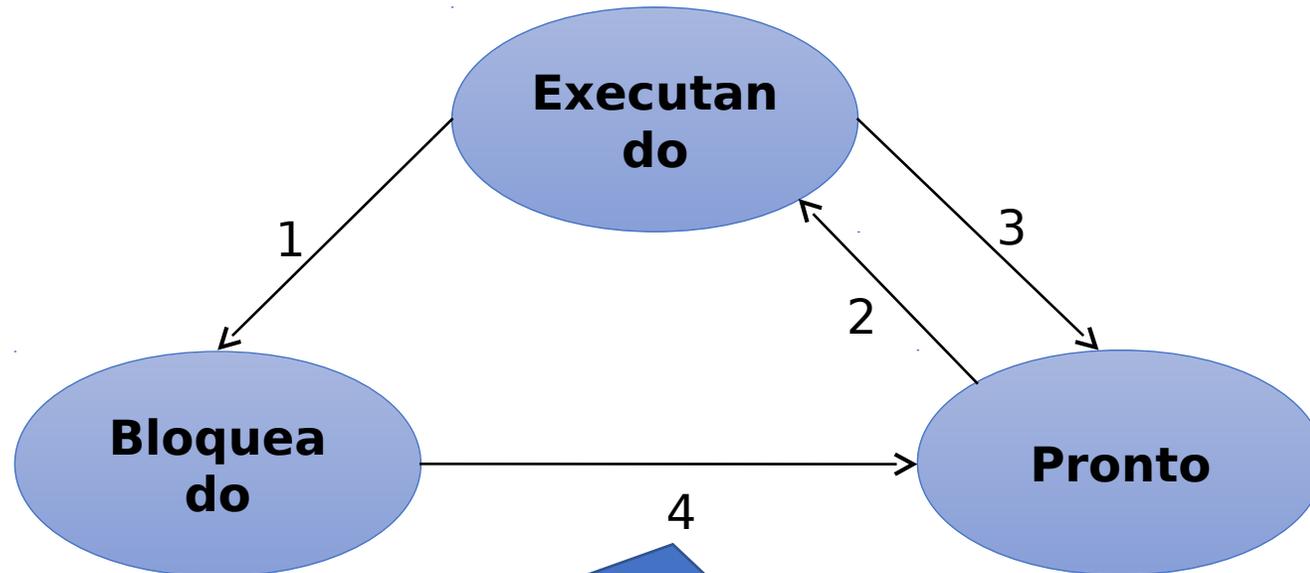
- chamada ao sistema: block ou pause;
- se não há entradas disponíveis para que o processo continue sua execução;



Estados de Processos



Estados de Processos



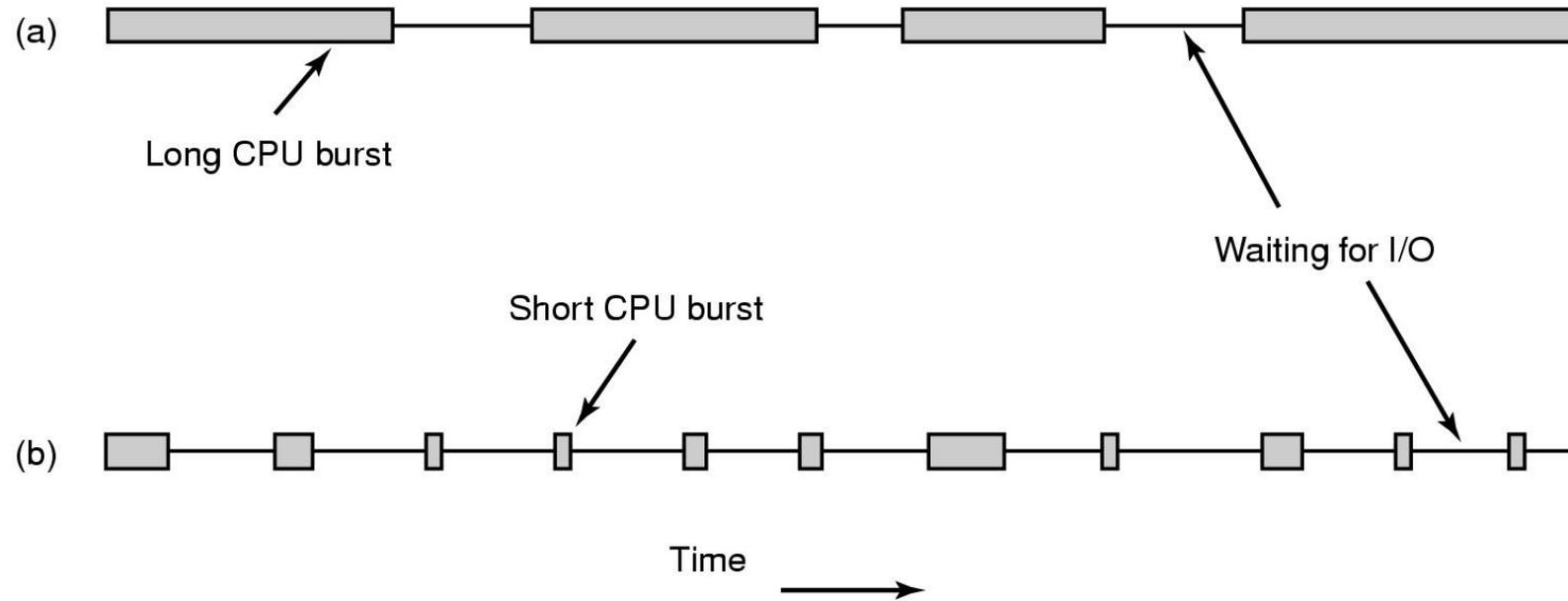
A transição 4 ocorre quando o evento esperado pelo processo bloqueado ocorre:

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

Processos

- 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



(a) Processos CPU-bound

(b) Processos I/O-bound

Escalonador de Processos

Processos

0	1	...	n-1	n
---	---	-----	-----	---

Escalonador de Processos

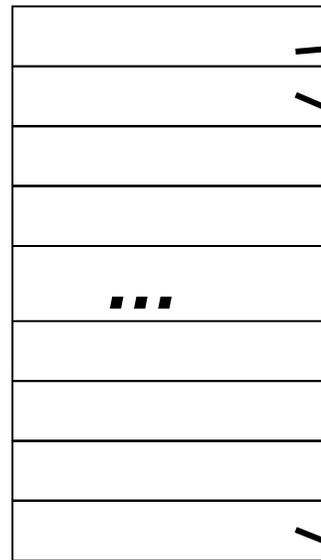
- Nível mais baixo do SO;
- Manipulação de interrupções e processos;

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

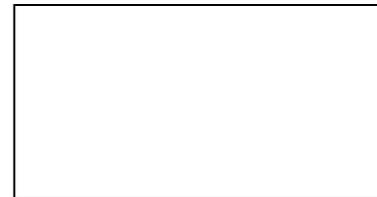
Tabela de processos



BCP - P1



BCP - P2



...

BCP - Pn



Implementação de Processos

Process management	Memory management	File management
Registers Program counter Program status word Stack pointer Process state 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	Pointer to text segment Pointer to data segment Pointer to stack segment	Root directory Working directory File descriptors User ID Group ID

Algumas informações do BCP