FACULDADE DE TECNOLOGIA - SENAC

SISTEMAS OPERACIONAIS

PROCESSOS

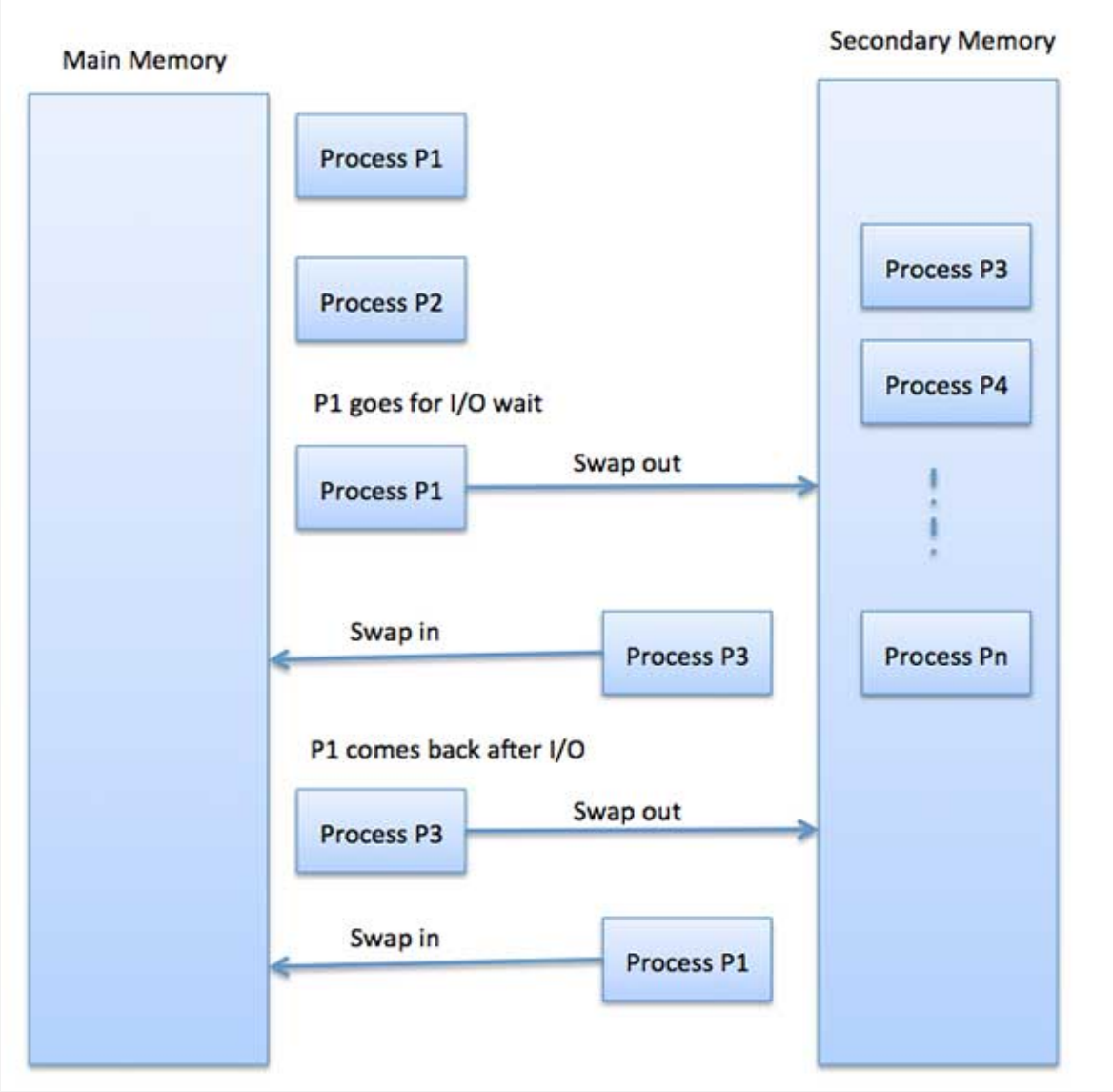
PROCESSO

- Conceito de processo
- Escalonamento de processos
- Operações com processos
- Comunicação entre processos

O QUE UM PROCESSO?

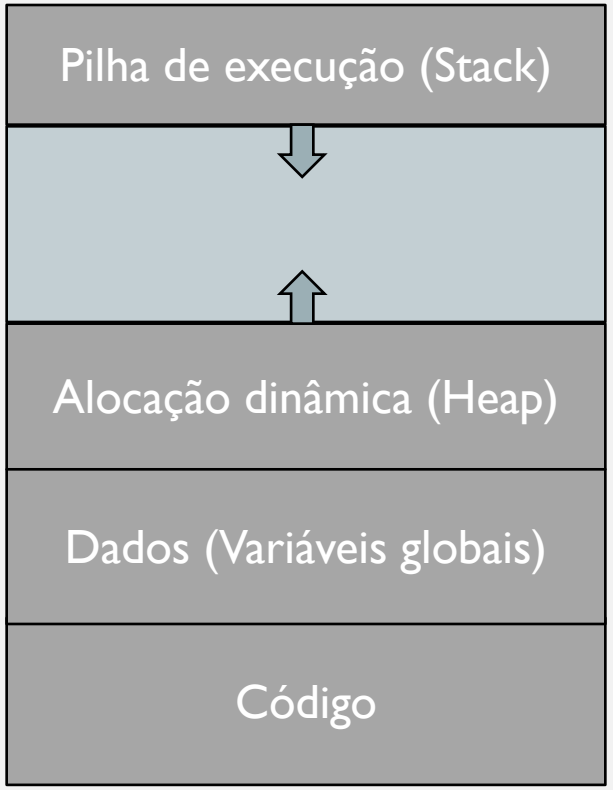
- Um sistema operacional executa uma variedade de programas:
  - Sistema Batch – *jobs*
  - Sistema Tempo Compartilhado (*Time-shared)* – programas do usuário ou tarefas
- Também conhecido como "tarefa" (task).
- **Processo** – um programa em execução na memória principal.
- Um processo inclui:
  - Contador de programa
  - Pilha
  - Seções de dados

O QUE UM PROCESSO?



O QUE UM PROCESSO?

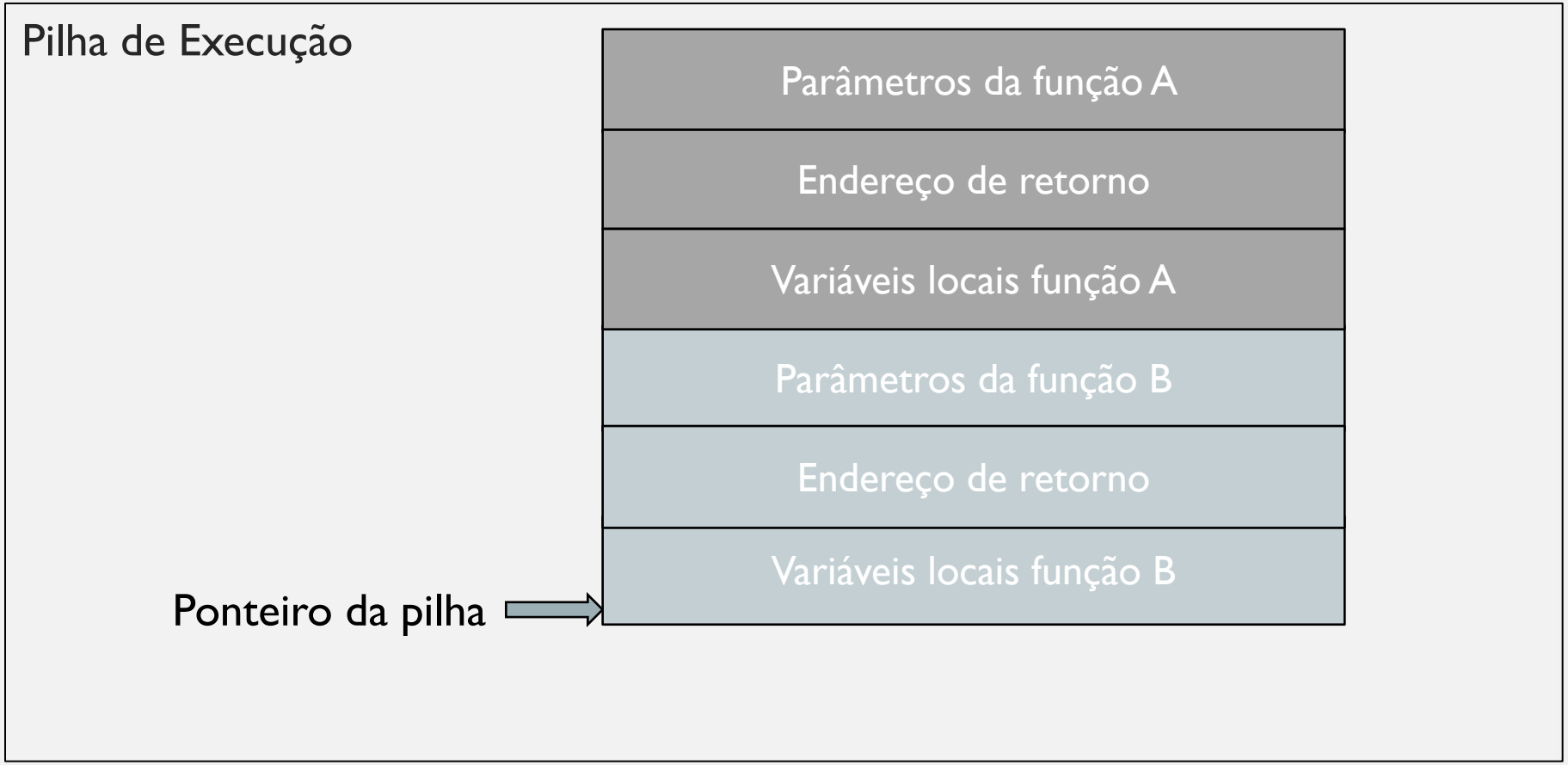
Mapa da memória



End. máximo

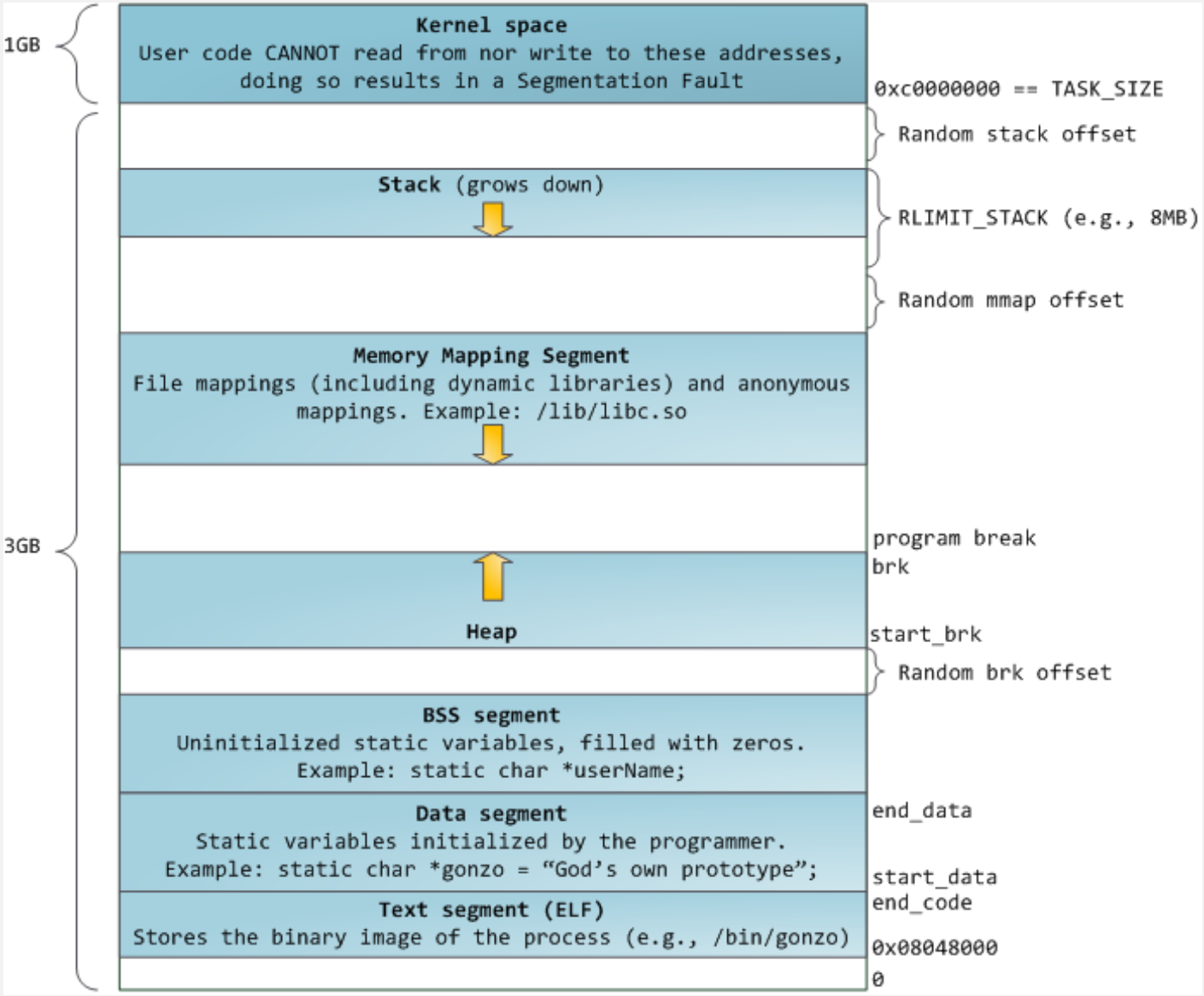
End. 0

O QUE UM PROCESSO?



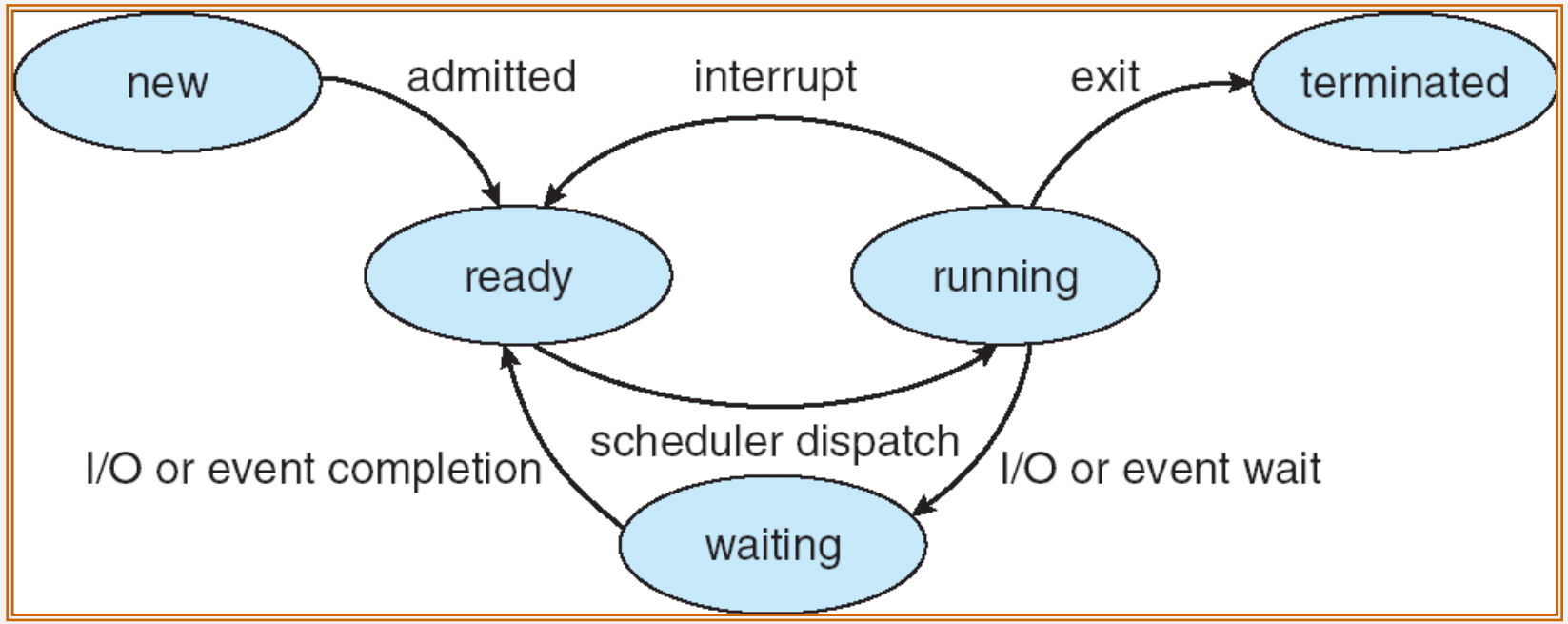
O QUE É UM PROCESSO?

Mapeamento da memória de um processo no Linux



ESTADO DE UM PROCESSO

- Durante a execução de um processo, ele altera seu *estado*
  - **Novo** (*new*): O processo está sendo criado.
  - **Executando** (*running*): instruções estão sendo executadas.
  - **Esperando** (*waiting*): O processo está esperando algum evento acontecer.
  - **Pronto** (*ready*): O processo está esperando ser associado a um procesador.
  - **Terminado** (*terminated*): O processo terminou sua execução.



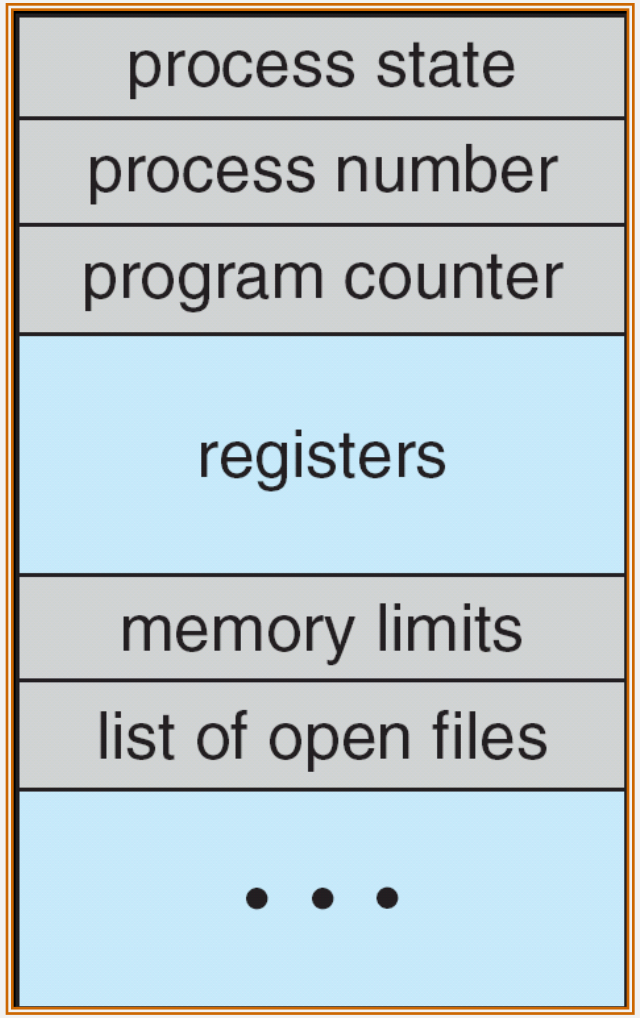
BLOCO DE CONTROLE DO PROCESSO

- A PCB ou **Bloco de Controle de Processos** armazena informações associada com cada processo.
  - Estado do Processo
  - Contador de Programas
  - Registradores da CPU
  - Informações de escalonamento da CPU
  - Informação de Gerenciamento de memória
  - Informação para Contabilidade
  - Informações do status de E/S

  - **Contexto do kernel, contexto do processador e espaço de endereços.**

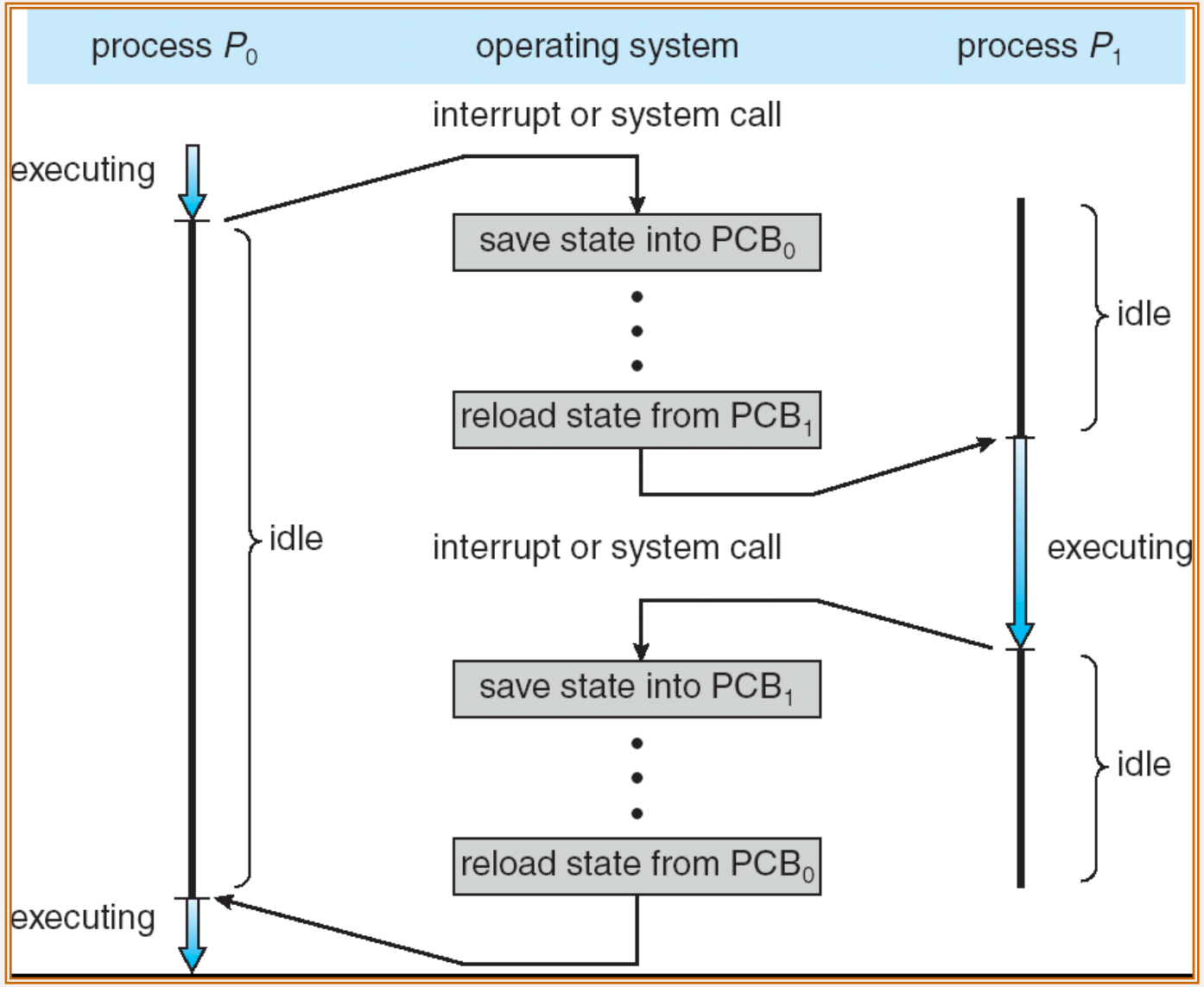
BLOCO DE CONTROLE DO PROCESSO

- Estrutura de dados no kernel para cada processo.
- Criado com o processo.
- Armazena o contexto do processador quando o processo é interrompido.
- Possui informações para o kernel sobre o processo: escalonamento, memória, recursos usados.



TROCA DE CONTEXTO NO PROCESSADOR

Possui suporte de hardware para otimizar.



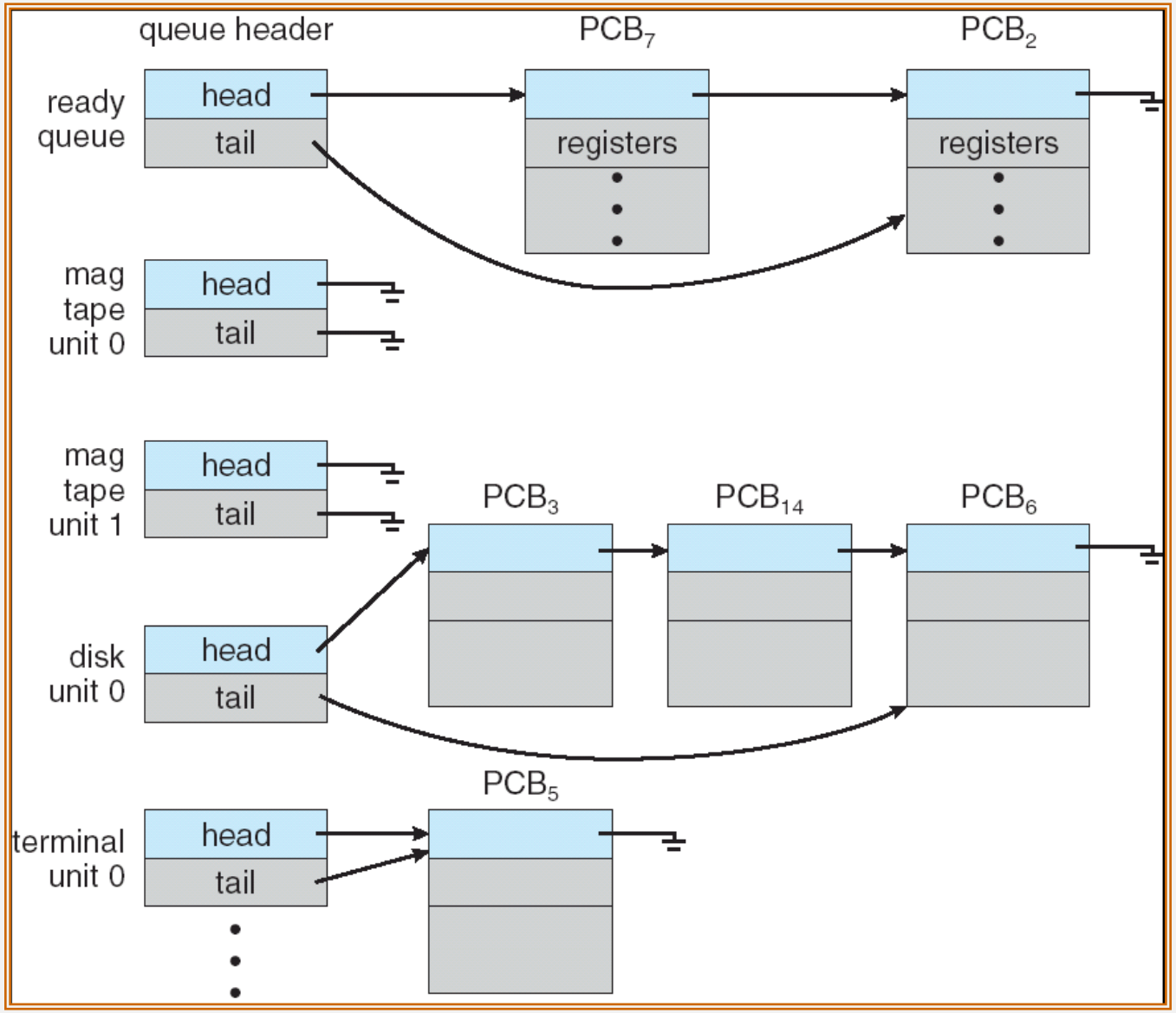
ESCALONAMENTO DE PROCESSOS

- Aumenta a utilização do processador
- Paralelismos aparente e transparente
- S.O. decide quem, como e quando.

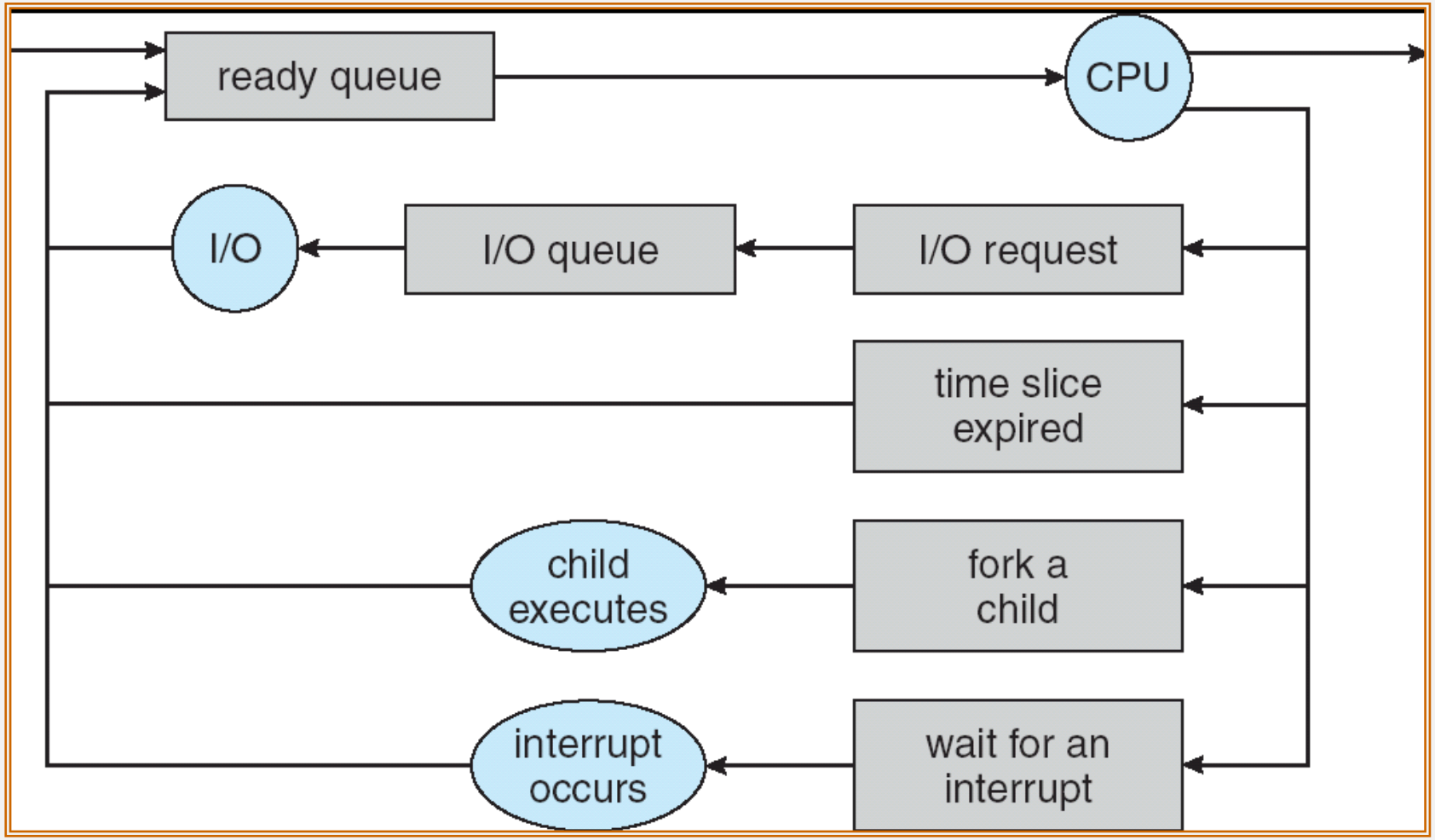
- **Fila de *Job*** – conjunto de todos os processos no sistema.
- **Fila de Processos prontos** (*Ready queue*) – conjunto de todos os processos residentes na memória principal, prontos e esperando para executar.
- **Fila de dispositivos** – conjunto dos processos esperando por um dispositivo de E/S.
- Migração de processos entre as várias filas.

ESCALONAMENTO DE PROCESSOS

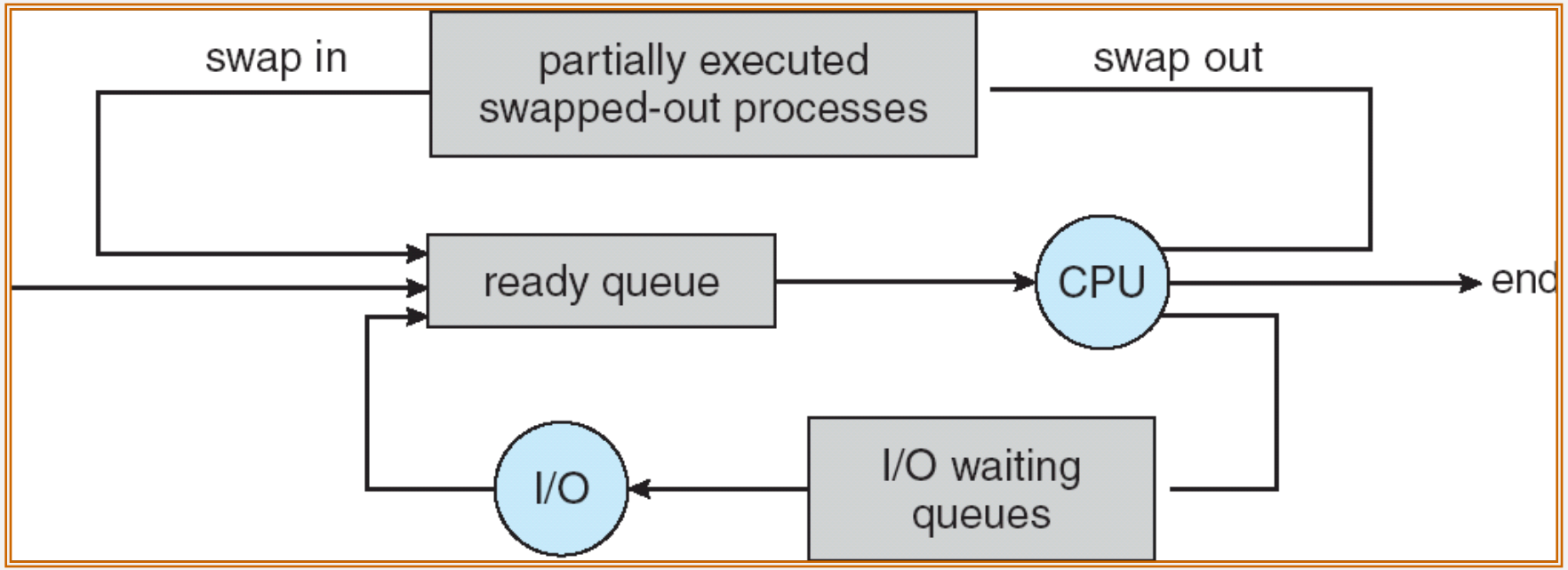
Filas de escalonamento



ESCALONAMENTO DE PROCESSOS



ESCALONAMENTO DE PROCESSOS

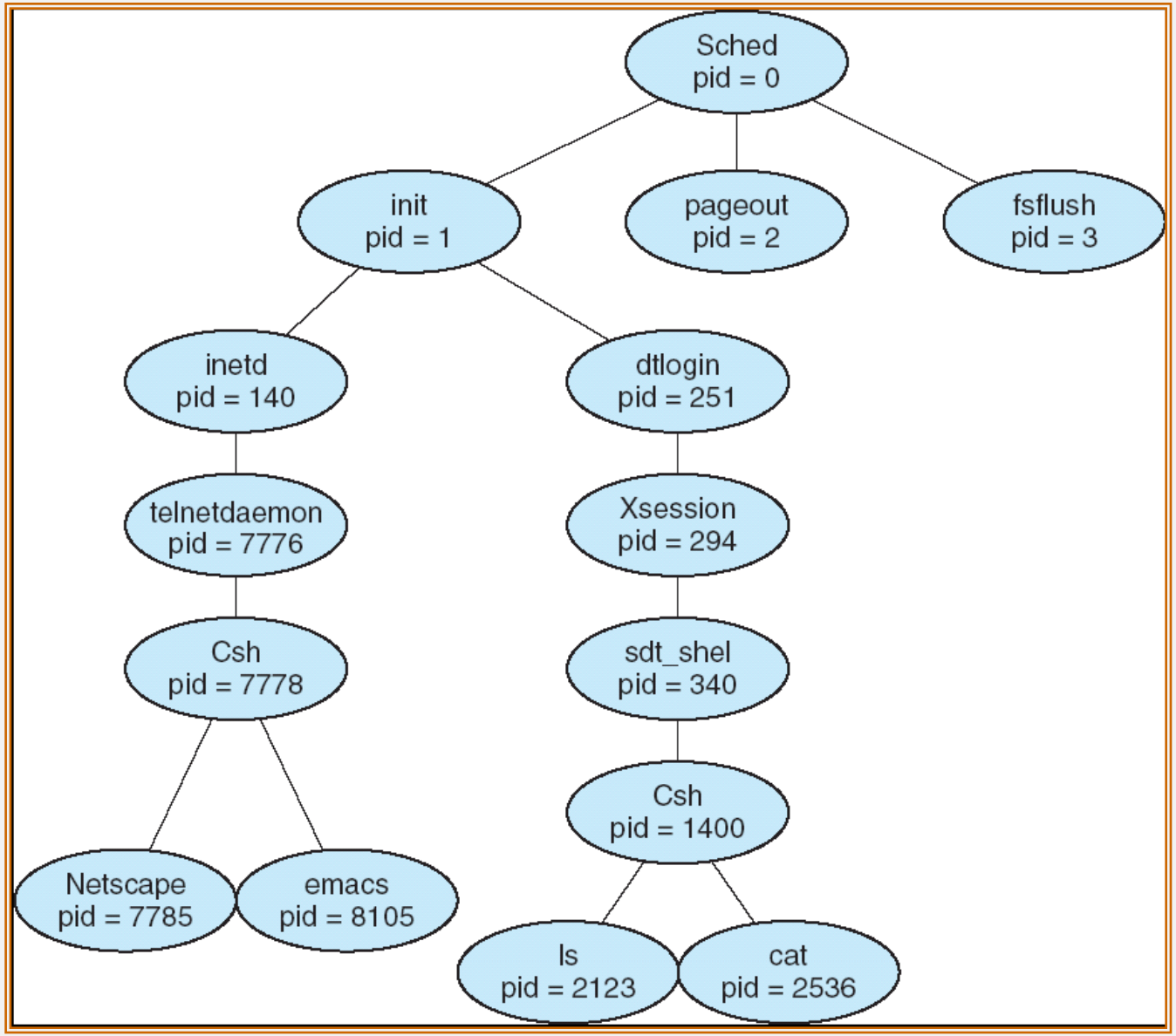


ESCALONAMENTO DE PROCESSOS

- Escalonador é invocado muito freqüentemente (milisegundos) ⇒ (deve ser rápido).
- Processos podem ser descritos como:
  - **Processos com E/S predominante** (I/O-*bound process)* – gasta mais tempo realizando E/S do que computando, muitos ciclos curtos de CPU.
  - **Processos com uso de CPU predominante** (CPU-*bound process)* – gasta mais tempo realizando computações; poucos ciclos longos de CPU.

CRIAÇÃO DE PROCESSOS

Hierarquia de processos
(S.O. Solaris)

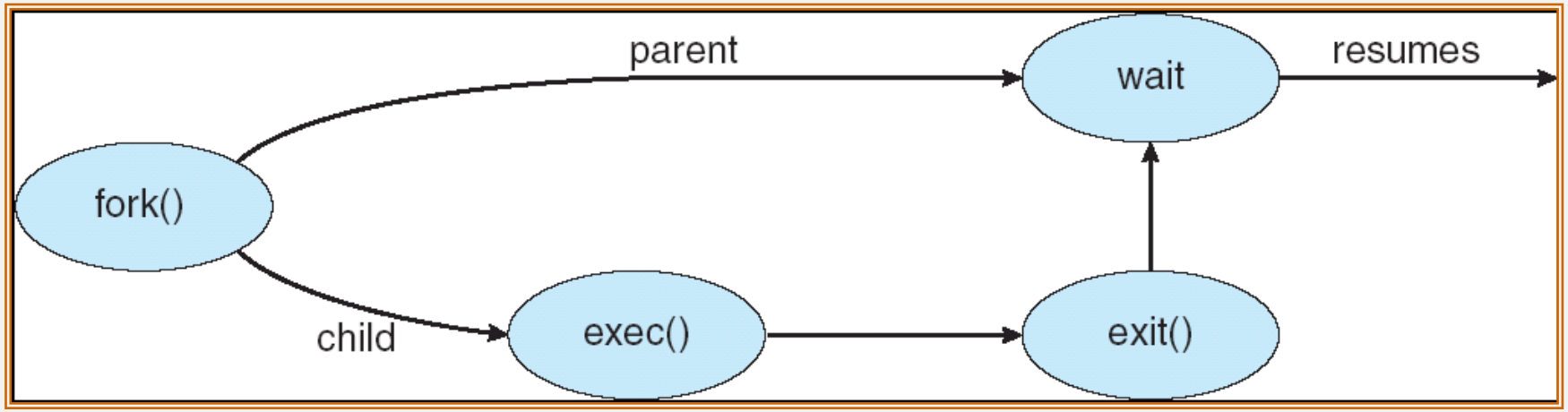


CRIAÇÃO DE PROCESSOS

- Processo pai cria processo filho, o qual, por sua vez, pode criar outros processos, formando uma árvore de processos.
- Geralmente, processos são identificados e gerenciados via um **Identificador de Processos** (*Process IDentifier* - ***PID***)
- Compartilhamento de Recursos:
  - Pai e filho compartilham todos os recursos; ou
  - Filho compartilha um subconjunto dos recursos do pai; ou
  - Pai e filho não compartilham recursos.
- Execução:
  - Pai e filho executam concorrentemente; ou
  - Pai espera até filho terminar.

CRIAÇÃO DE PROCESSOS

- Espaço de endereçamento
  - Filho duplica espaço do pai.
  - Filho tem um programa carregado no seu espaço.
- Exemplos no UNIX
  - Chamada de sistemas ***fork*** cria um novo processo.
  - Chamada de sistemas ***exec*** é usada após o ***fork*** para sobrescrever o espaço de memória do processo com um novo programa.



CRIAÇÃO DE PROCESSOS

Programa em C criando um novo processo

```
int main()
{
  Pid_t  pid;
                    /* cria outro processo */
  pid = fork();
  if (pid < 0) {    /* ocorrência de erro*/
        fprintf(stderr, “Criação Falhou");
        exit(-1);
  }                 /* processo filho*/
  else if (pid == 0) {
        execlp("/bin/ls", "ls", NULL);
  }
  else {            /* processo pai */
                   /* pai irá esperar o filho completar execução */
        wait (NULL);
        printf (“Filho Completou Execução");
        exit(0);
  }
}
```

TÉRMINO DE PROCESSOS

- Processo executa última declaração e pede ao sistema operacional para decidir (***exit***).
  - Dados de saída passam do filho para o pai (via ***wait***).
  - Recursos do processo são desalocados pelo sistema operacional.
- Pai pode terminar a execução do processo filho (***abort***).
  - Filho se excedeu alocando recursos.
  - Tarefa delegada ao filho não é mais necessária.
  - Pai está terminando:
    - Sistema operacional não permite que um filho continue sua execução se seu pai terminou.
    - Todos os filhos terminam - Terminação em cascata.

CRIAÇÃO DE PROCESSOS EM JAVA

- Quando um programa Java começa a execução, uma instância da máquina virtual Java é criada. Na maioria dos sistemas, a JVM aparece como um aplicativo comum em execução como um processo separado no sistema operacional do host. Cada instância da JVM fornece suporte para vários encadeamentos de controle; mas o Java não suporta um modelo de processo, o que permitiria que a JVM criasse vários processos dentro da mesma máquina virtual.

- É possível criar um processo externo para a JVM, no entanto, usando a classe ProcessBuilder, que permite que um programa Java especifique um processo que é nativo para o sistema operacional (como /usr/bin/ls ou C:\WINDOWS\system32\mspaint.exe).

- Executar este programa envolve passar o nome do programa que deve ser executado como um processo externo na linha de comando. Criamos o novo processo invocando o método start () da classe ProcessBuilder, que retorna uma instância de um objeto Process. Esse processo será executado externamente à máquina virtual e não poderá afetar a máquina virtual - e vice-versa. A comunicação entre a máquina virtual e o processo externo ocorre através do InputStream e OutputStream do processo externo.

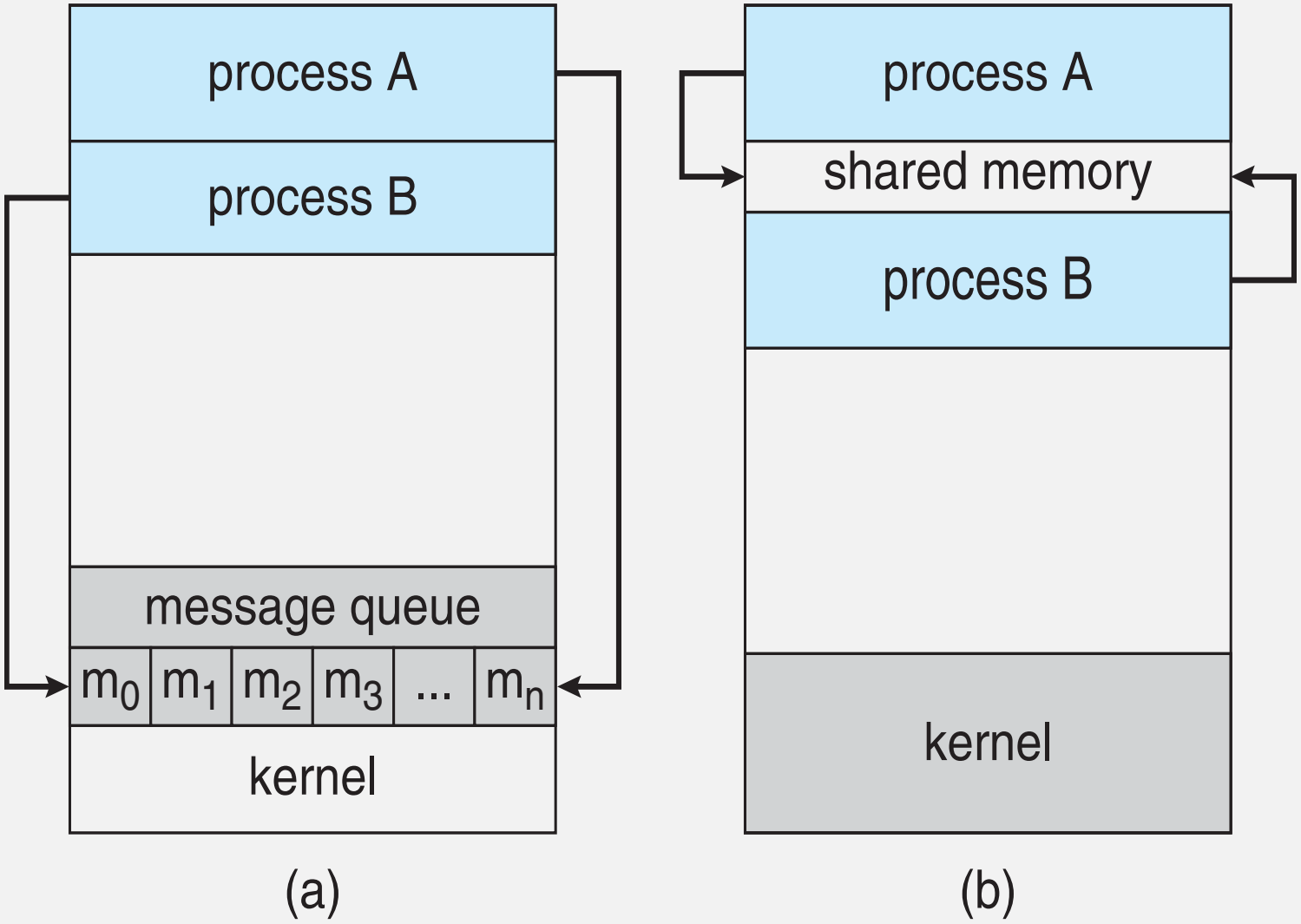
CRIAÇÃO DE PROCESSOS EM JAVA

```
import java.io.*;
public class OSProcess
{
    public static void main(String[] args) throws IOException {
        if (args.length != 1) {
            System.err.println("Usage: java OSProcess <command>");
            System.exit(0);
        }
       // args[0] is the command that is run in a separate process
        ProcessBuilder pb = new ProcessBuilder(args[0]);
        Process process = pb.start();
      // obtain the input stream
        InputStream is = process.getInputStream();
        InputStreamReader isr = new InputStreamReader(is);
        BufferedReader br = new BufferedReader(isr);
       // read the output of the process
        String line;
        while ( (line = br.readLine()) != null)
        System.out.println(line);
        br.close();
    }
}
```

COMUNICAÇÃO ENTRE PROCESSOS

- Processos em um sistema podem ser **Independentes** ou **Cooperantes**
- Processos **Independentes** não podem afetar ou ser afetados pela execução de outro processo.
- Processos **Cooperantes** podem afetar ou ser afetados pela execução de outro processo
- Razões para cooperação entre processos:
  - Compartilhamento de Informações
  - Aumento na velocidade da computação
  - Modularidade
  - Conveniência
- Processos cooperantes precisam de **Comunicação entre Processos** (**IPC** – *interprocess communication*)
- Dois modelos de IPC: memória compartilhada e troca de mensagens

COMUNICAÇÃO ENTRE PROCESSOS



COMUNICAÇÃO: TROCA DE MENSAGENS

- Mecanismo para processos se comunicarem e sincronizarem suas ações.
- Sistema de mensagens – processos se comunicam uns com os outros sem utilização de variáveis compartilhadas.
- Suporte a IPC (*InterProcess Communication*) provê duas operações uma para envio outra para recebimento:
  - **send**(*mensagem*) – tamanho da mensagem fixo ou variável
  - **receive**(*mensagem*)
- Se *P* e *Q* querem se comunicar, eles necessitam:
  - Estabelecer um *link de comunicação* entre eles
  - Trocar mensagens via send/receive
- Implementação de links de comunicação
  - Físico (ex. Memória compartilha, barramento de hardware)
  - Lógico (ex. Propriedades lógicas)

PROCESSOS

- Revisão:
  - Vimos como o S.O. gerencia processos e cria abstrações relacionadas com processos:
    - Máquina virtual (espaço de endereços, cpu, recursos)
    - Proteção
    - Mecanismos: troca de contexto, escalonamento, comunicação entre processos.