PROCESSOS e THREADS

PROCESSOS

- A evolução do hardware e das necessidades do usuário cobra um preço dos SOs: o SO gerencia todos os softwares e hardware, que estão cada vez mais complexos.

- Antes apenas uma rotina era interpretada por vez, e hoje são muitas.

- Assim, o SO precisou evoluir.

- Para que o SO garanta a integridade e confiabilidade de cada tarefa solicitada pelo usuário, ele a trata como um “Processo”, alocando recursos e registradores específicos para cada tarefa.

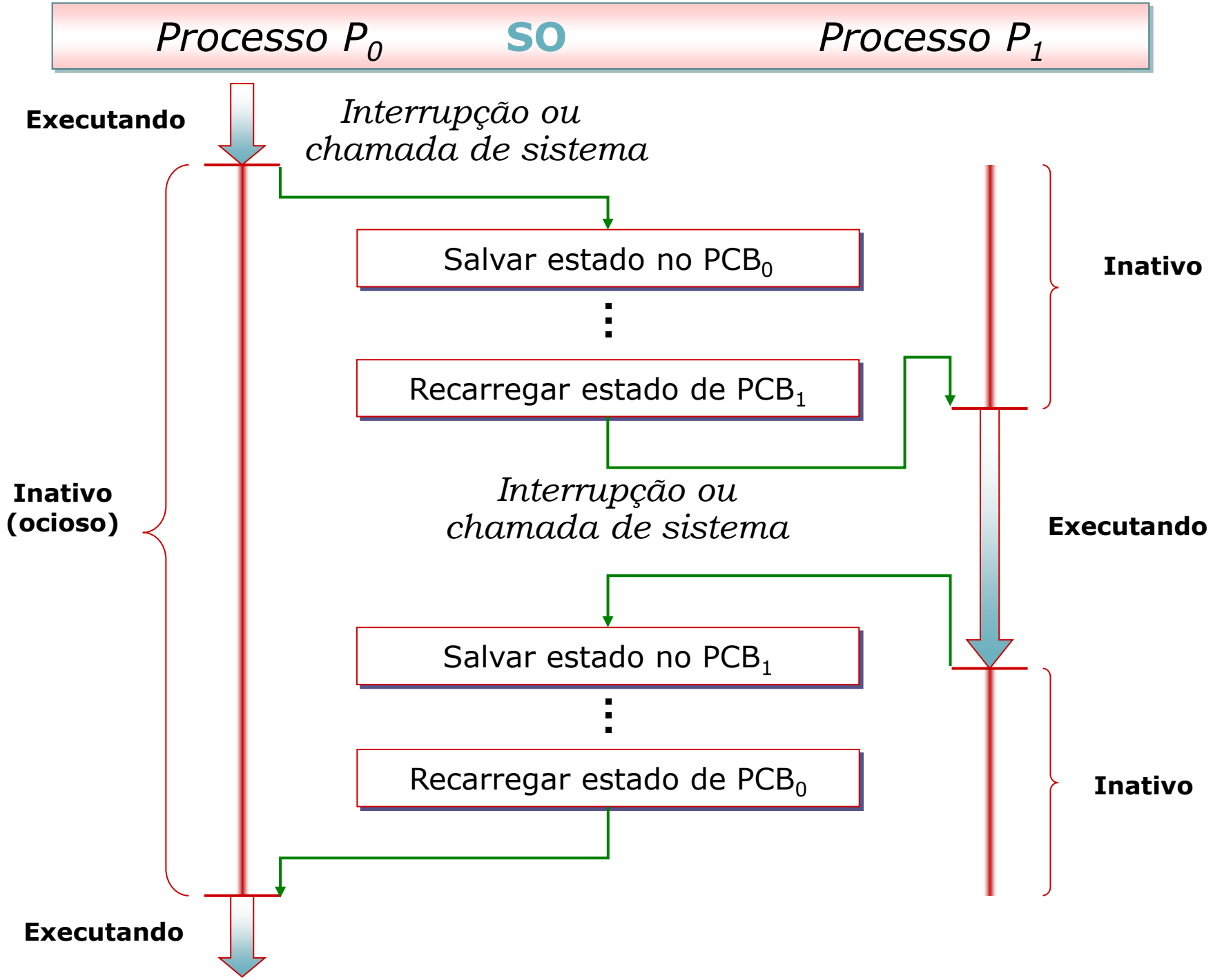
- Processo, de maneira simples, é um programa em execução, mas pode ser definido como **toda a estrutura necessária para que um programa seja executado (alocação de processador, memória e dispositivos de E/S).**

- Um processo é formado por três partes:
  - Contexto de hardware
  - Contexto de software
  - Espaço de endereçamento.

Estrutura de um Processo

- **Contexto de Hardware**
  - Esse contexto armazena o conteúdo dos registradores gerais do processador, além dos registradores de uso específico (Program counter, Stack Pointer e Registrador de Status).
  - O contexto de hardware é fundamental para a implementação dos sistemas multiprogramáveis, pois há várias tarefas sendo executadas simultaneamente, mas não ao mesmo tempo.
  - Para que isso seja possível, o sistema operacional trabalha com interrupções junto aos registradores.

- **Bloco de Controle de Processo - PCB**
  - O bloco de controle de processo (em inglês: **P**rocess **C**ontrol **B**lock ou **PCB**) é uma estrutura de dados no núcleo do sistema operacional que serve para armazenar a informação necessária para tratar um determinado processo.
  - Como o **PCB** contém informações críticas do processo ele deve ficar armazenado em uma área da memória protegida do acesso de usuários. Em alguns sistemas operacionais o **PCB** é alocado no início da pilha do núcleo do processo, já que é uma localização convenientemente protegida.


Processo P_0
SO
Processo P_1
Executando
Interrupção ou chamada de sistema
Salvar estado no PCB_0
Recarregar estado de PCB_1
Inativo
Inativo (ocioso)
Interrupção ou chamada de sistema
Executando
Salvar estado no PCB_1
Recarregar estado de PCB_0
Inativo
Executando

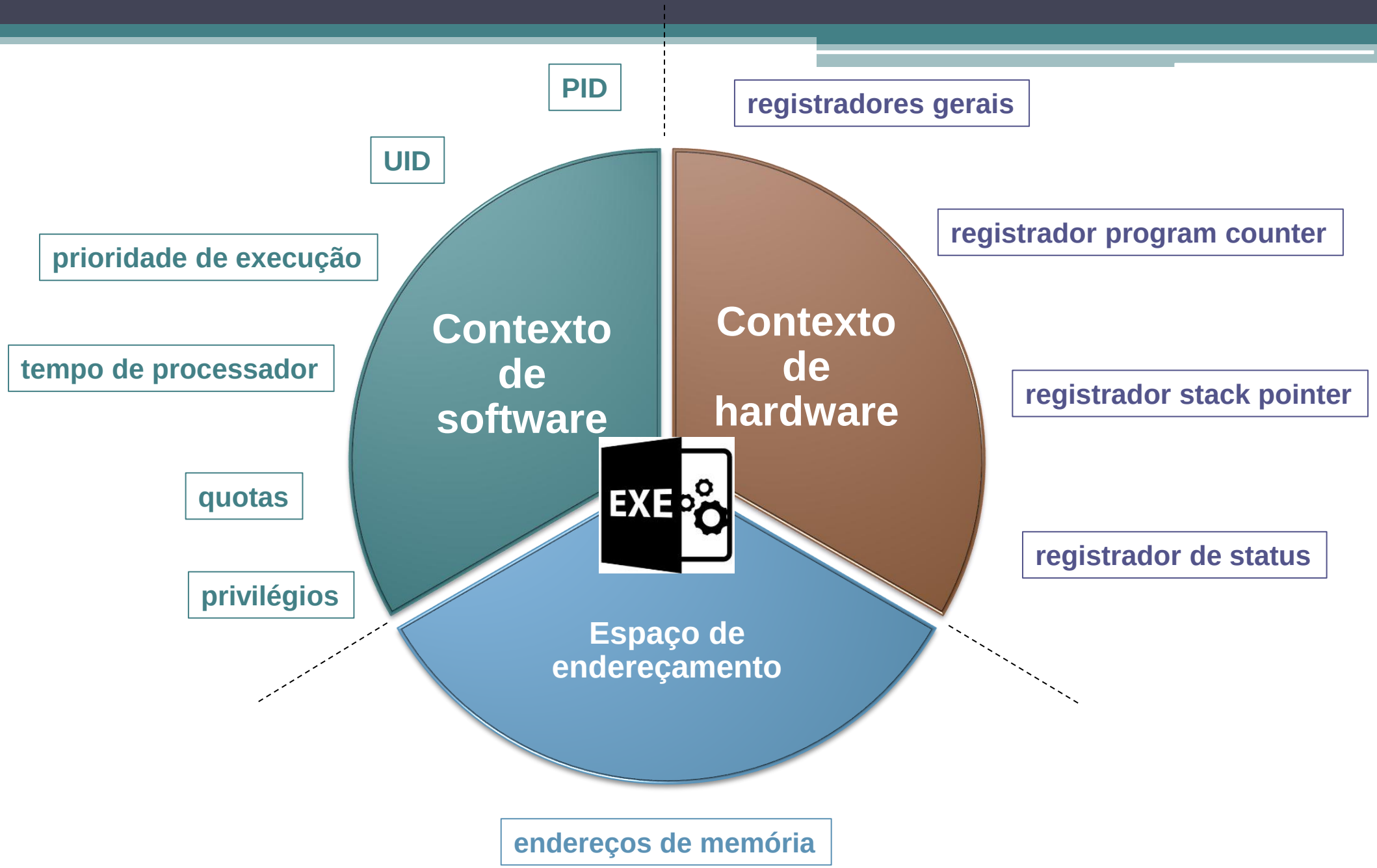
- **Contexto de software**
  - Nesse contexto são especificados limites e características dos recursos que podem ser alocados pelo processo (processador, memória, dispositivos de E/S).
  - Muitas dessas características são determinadas no momento da criação do processo, enquanto outras podem ser alteradas durante sua existência.
  - É composto por três grupos de informações:
    - Identificação;
    - Quotas;
    - Privilégios.

- **Identificação**: ao ser criado, o processo recebe duas identificações: **PID (Process Identification)**, representado por um número, e por meio dele que o SO faz o gerenciamento; há também o **UID (User Identification)**, que permite implementar o modelo de segurança.

- **Quotas**: refere-se aos limites de recursos a serem atribuídos para a tarefa. Caso não seja suficiente, o processo pode ser executado lentamente, ser interrompido ou não ser executado.
  - Exemplos:
    - Número máximo de arquivos abertos simultaneamente;
    - Tamanho máximo de memória principal e secundária que o processo pode alocar;
    - Número máximo de operações de E/S pendentes, etc.

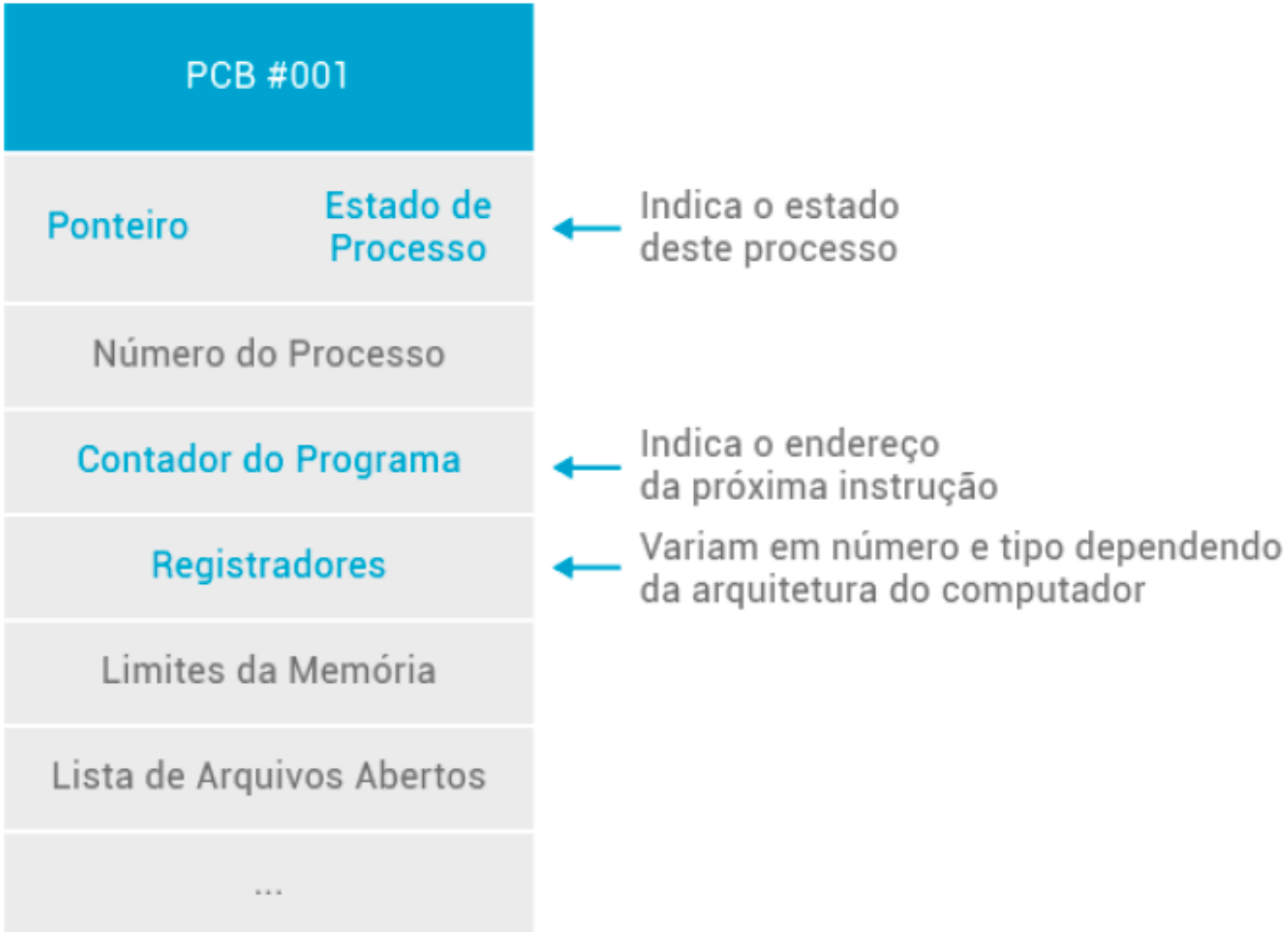
- **Privilégios**: definem ações que um processo pode fazer em relação a ele mesmo, aos demais processos e ao sistema operacional.
  - Privilégios que afetam o próprio processo permitem que suas características possam ser alteradas, como prioridade de execução, limites alocados na memória principal e secundária, etc.
  - Os privilégios que afetam os demais processos permitem, além da alteração de suas próprias características, alterar as de outros processos.
  - Privilégios que afetam o sistema são os mais amplos e poderosos, pois estão relacionados à operação e a gerência do ambiente.

- **Espaço de endereçamento**
  - Área de memória do processo em que instruções e dados do programa são armazenados para execução. Cada processo possui seu próprio espaço de endereçamento, que deve ser devidamente protegido do acesso dos demais processos.


PID
registradores gerais
UID
registrador program counter
prioridade de execução
Contexto de software
Contexto de hardware
registrador stack pointer
tempo de processador
EXE
quotas
registrador de status
privilégios
Espaço de endereçamento
endereços de memória

Bloco de Controle do Processo - PCB

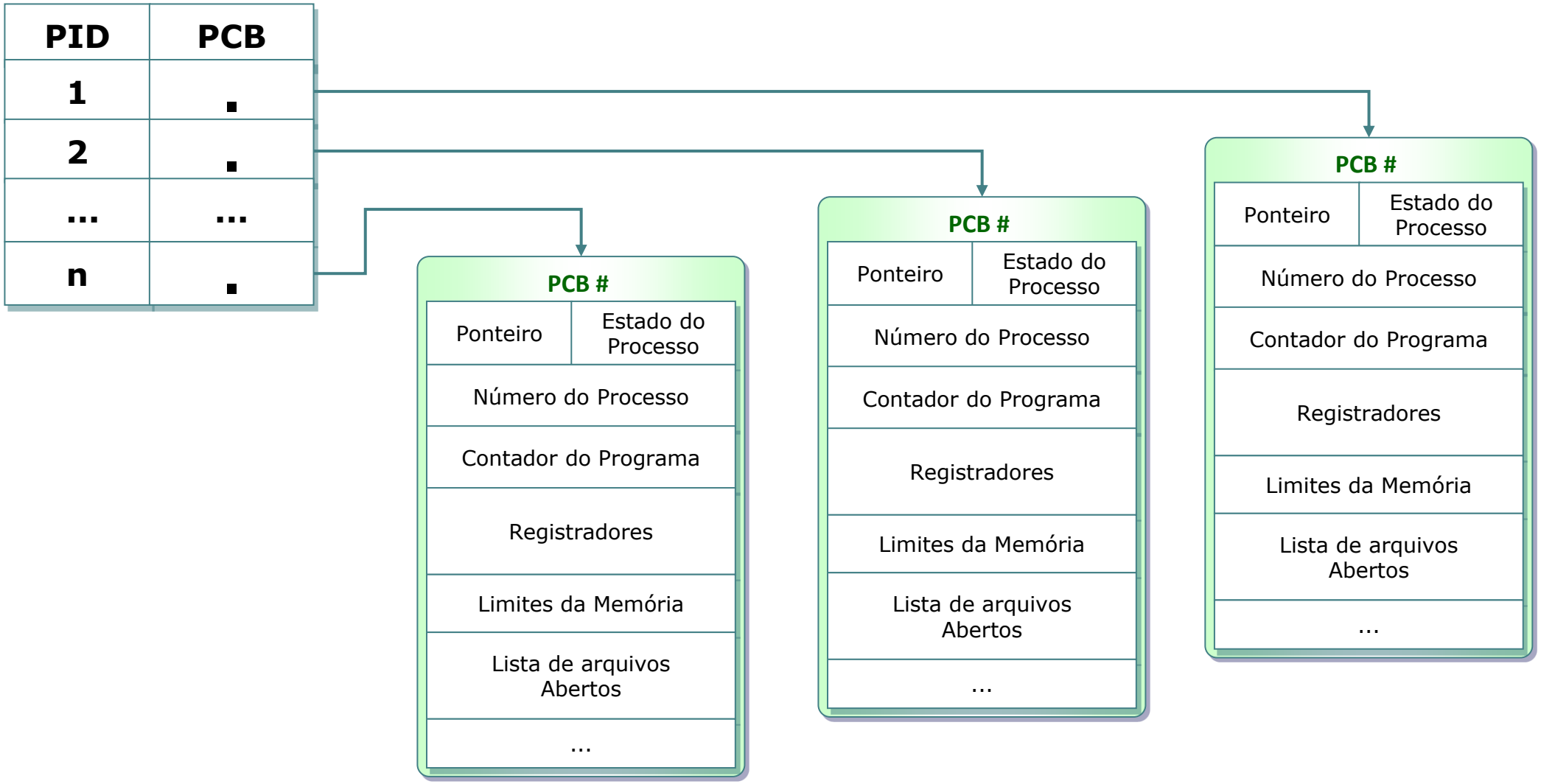
- Os processos são implementados pelo SO por meio de uma estrutura chamada PCB;
- A partir do PCB o SO mantém todas as informações sobre os processos em execução.
- Os PCBs de todos os processos ativos residem na memória principal, em uma área exclusiva do SO.
- O tamanho dessa área geralmente é limitado por um parâmetro do sistema operacional que permite especificar o número máximo de processos que podem ser suportados simultaneamente pelo sistema.


PCB #001
Ponteiro
Estado de Processo
Indica o estado deste processo
Número do Processo
Contador do Programa
Indica o endereço da próxima instrução
Registradores
Variam em número e tipo dependendo da arquitetura do computador
Limites da Memória
Lista de Arquivos Abertos
...

Existe uma tabela de processos em execução, que o SO mantém com ponteiros para cada PCB.

Essa tabela permite acesso rápido aos PCBs.

Os processos são retirados dela quando encerrados.


PID
PCB
1
2
...
n
PCB #
Ponteiro
Estado do Processo
Número do Processo
Contador do Programa
Registradores
Limites da Memória
Lista de arquivos Abertos
...

Estado dos Processos

- Como já vimos, nos SOs atuais, um programa (e seus processos) não alocam o processador exclusivamente. Há o uso compartilhado do mesmo, assim como dos demais recursos do computador.

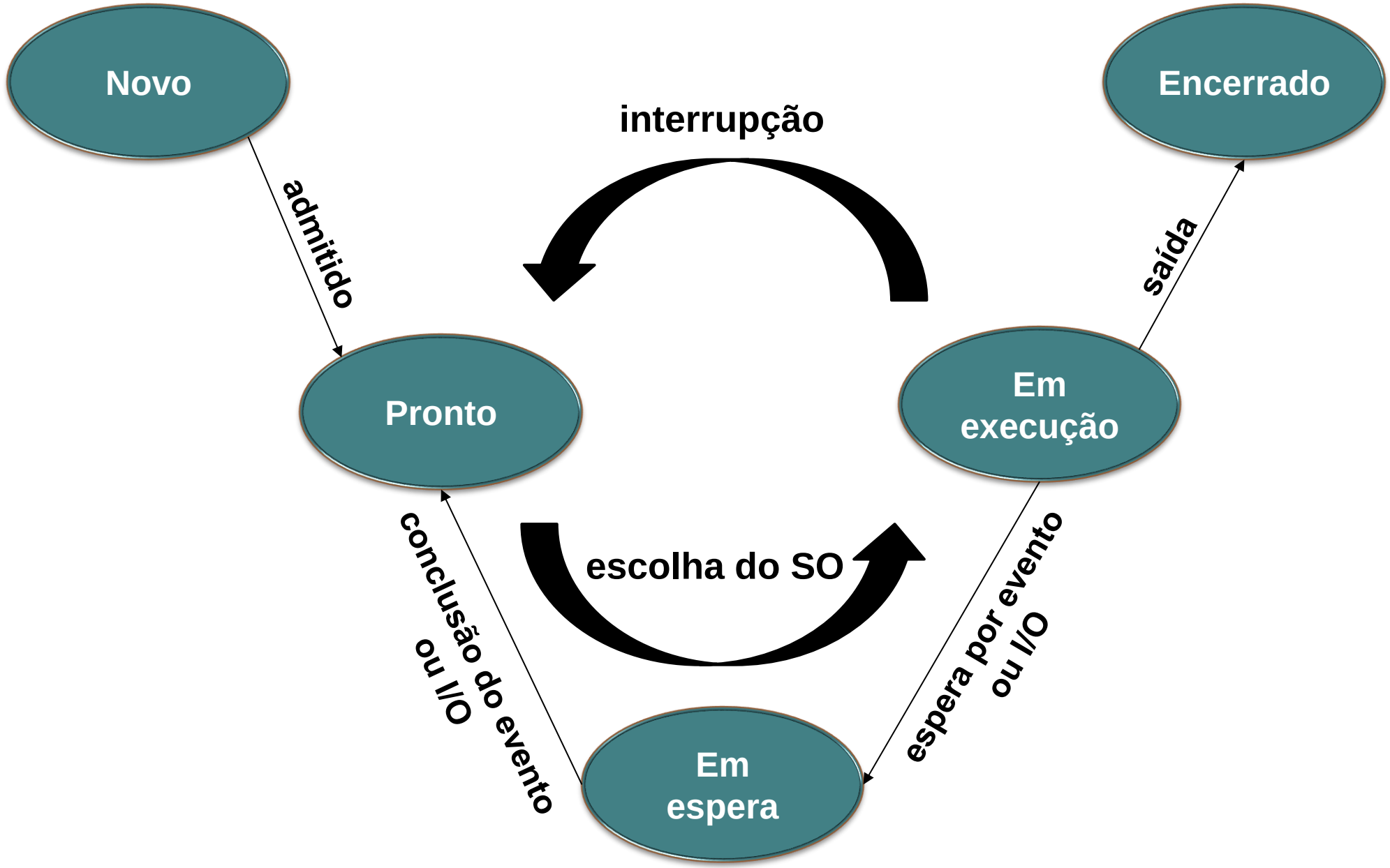
- Por isso, há a alternância entre processos em execução, e essa alternância faz com que os processos estejam em diferentes estados, de acordo com o momento de execução.

- Existem os seguintes estados dos processos:
  - Novo
  - Pronto
  - Em execução
  - Em espera
  - Encerrado

Apenas um processo pode estar em Execução

Vários processos podem estar em Espera ou Prontos

- **Novo**: quando o processo é criado;
- **Pronto**: quando o processo aguarda para ser executado. O SO é quem determina a ordem e os critérios para que o processo saia do estado de pronto;
- **Em execução**: quando o processo está fazendo uso do processador;
- **Em espera**: quando o processo aguarda por algum evento externo ou por algum recurso computacional de I/O para seguir com o seu processamento;
- **Encerrado**: quando o processo é finalizado e destruído pelo SO.


Novo
admitido
interrupção
Encerrado
saída
Pronto
Em execução
escolha do SO
conclusão do evento ou I/O
espera por evento ou I/O
Em espera

Mudanças de Estado do Processo

- Todo processo muda de estado durante sua execução e seu ciclo de vida (desde quando é criado até não ser mais processado) em razão de eventos:
  - Criados por ele próprio;
  - Criados pelo SO.

Criados por ele próprio: eventos voluntários como, por exemplo, a leitura de arquivos para entrada de informações e gravação de informações em arquivos de saída;

Criados pelo SO: eventos involuntários como uma chamada para execução, previamente escalonada no SO (chamada automática).

- As mudanças de estado que podem acontecer com um processo são:
  - Novo → Pronto
  - Pronto → Execução
  - Execução → Espera
  - Espera → Pronto
  - Execução → Pronto
  - Execução → Encerrado

Pronto → **Execução**: após a criação de um processo, o SO coloca em uma lista de processos no estado de pronto, em que aguarda para ser executado.

Execução → **Espera**: um processo em execução passa para o estado de espera por eventos gerados pelo próprio processo (eventos voluntários), ou por eventos externos (gerados pelo SO).

Espera → Pronto: um processo passa no estado de espera para o estado de pronto quando a solicitação é atendida ou o recurso que aguarda é concedido.

Para sair do estado de espera e passar para o estado de execução, ele obrigatoriamente vai para o estado de pronto, pois o SO só seleciona processos para execução dos que estiverem no estado de pronto.

Execução → Pronto: um processo em execução passa para o estado de pronto por conta de eventos externos, como por exemplo, se o tempo estimado para execução do processo expirar.

Se isso ocorrer, o processo volta para a fila de pronto, em que irá esperar por uma nova oportunidade para continuar seu processamento.

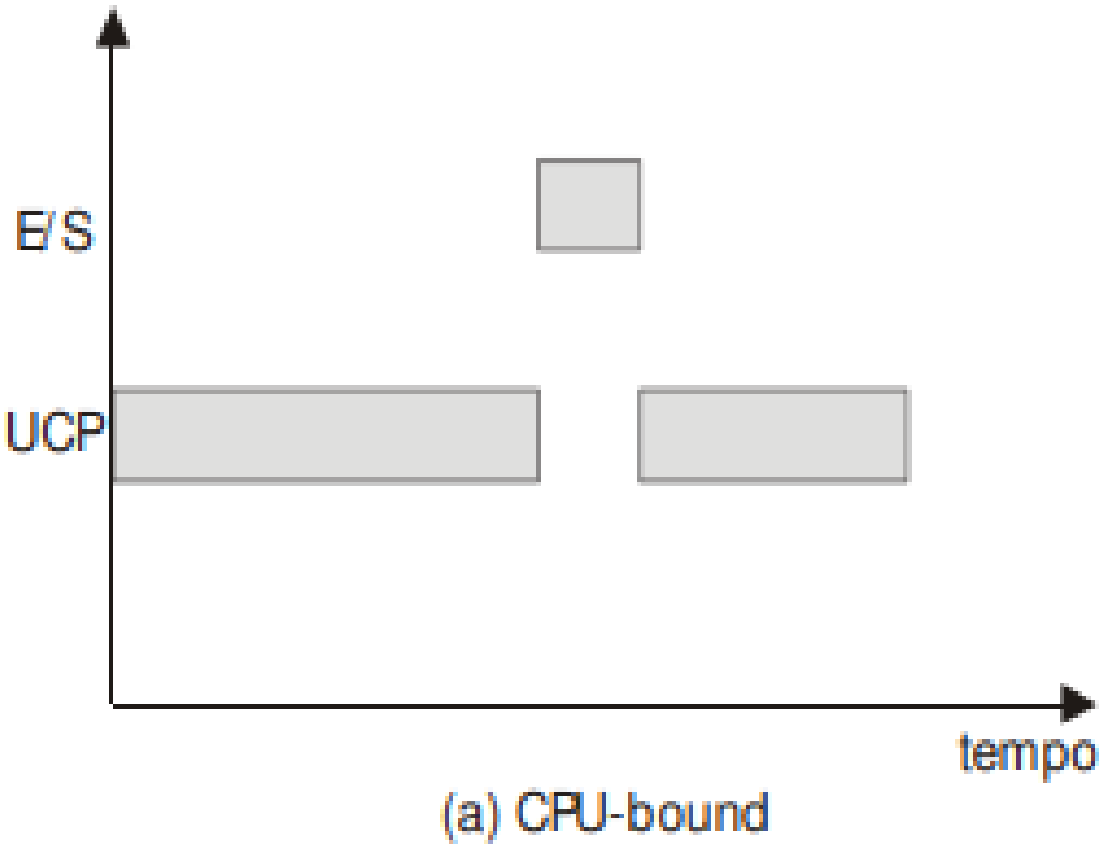
Criação e Eliminação de Processos

- Os processos são criados e eliminados por diversas razões.
- A criação ocorre a partir do momento em que o SO adiciona um novo PCB à sua estrutura e atribui um espaço de endereçamento na memória para uso.
- Após a criação do PCB, o SO já detecta a existência do processo. Sendo assim é possível gerenciar e associar programas ao seu contexto para serem executados.
- Para eliminar o processo, todos os recursos associados a ele deverão ser retirados e o PCB deletado pelo sistema operacional.

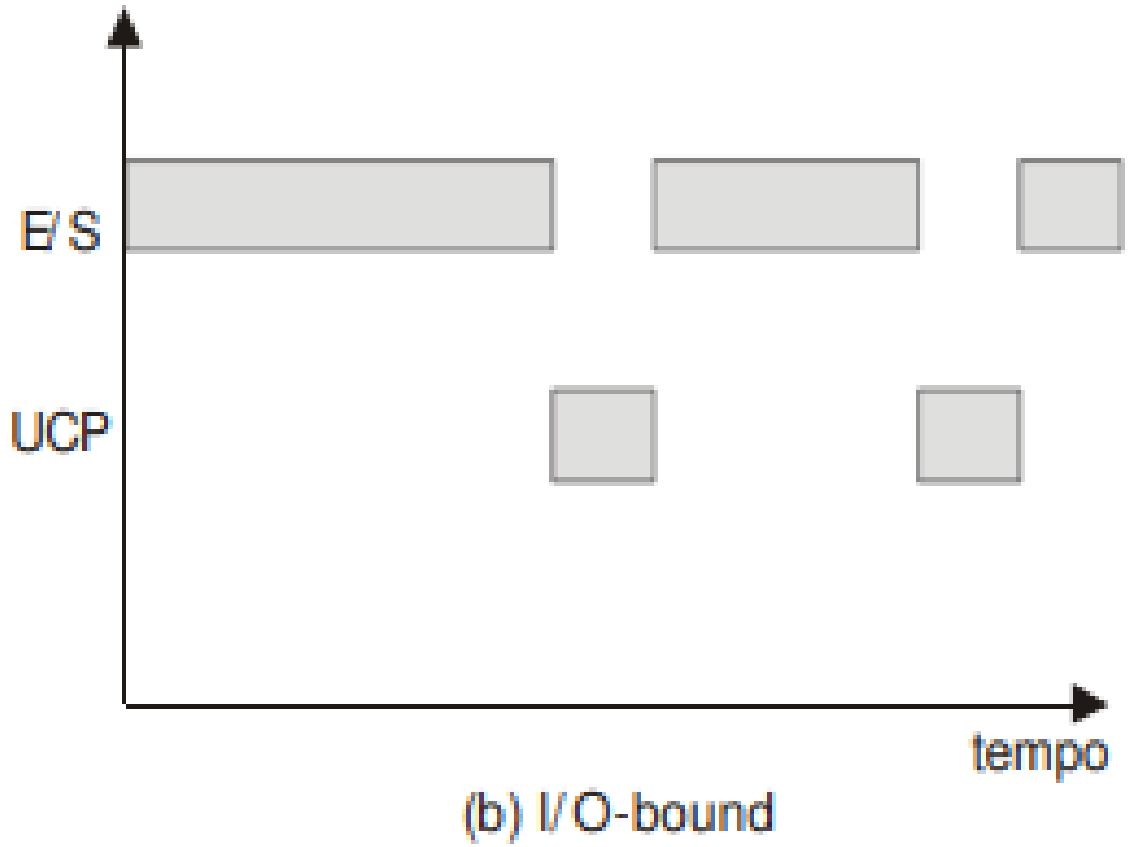
Processos CPU-bound e I/O bound

- Processos podem ser classificados como CPU bound ou I/O bound de acordo com a utilização do processador e dos dispositivos de E/S.
  - **CPU-bound:** quando passa a maior parte do tempo no estado de execução, utilizando o processador, ou pronto.
    - Normalmente encontrado em aplicações científicas.
  - **I/O-bound:** quando passa a maior parte do tempo no estado de espera, pois realiza um elevado número de operações de E/S.
    - Normalmente encontrado em aplicações comerciais.

Processos CPU-bound X I/O bound



(a) CPU-bound



(b) I/O-bound

Processos Foreground e Background

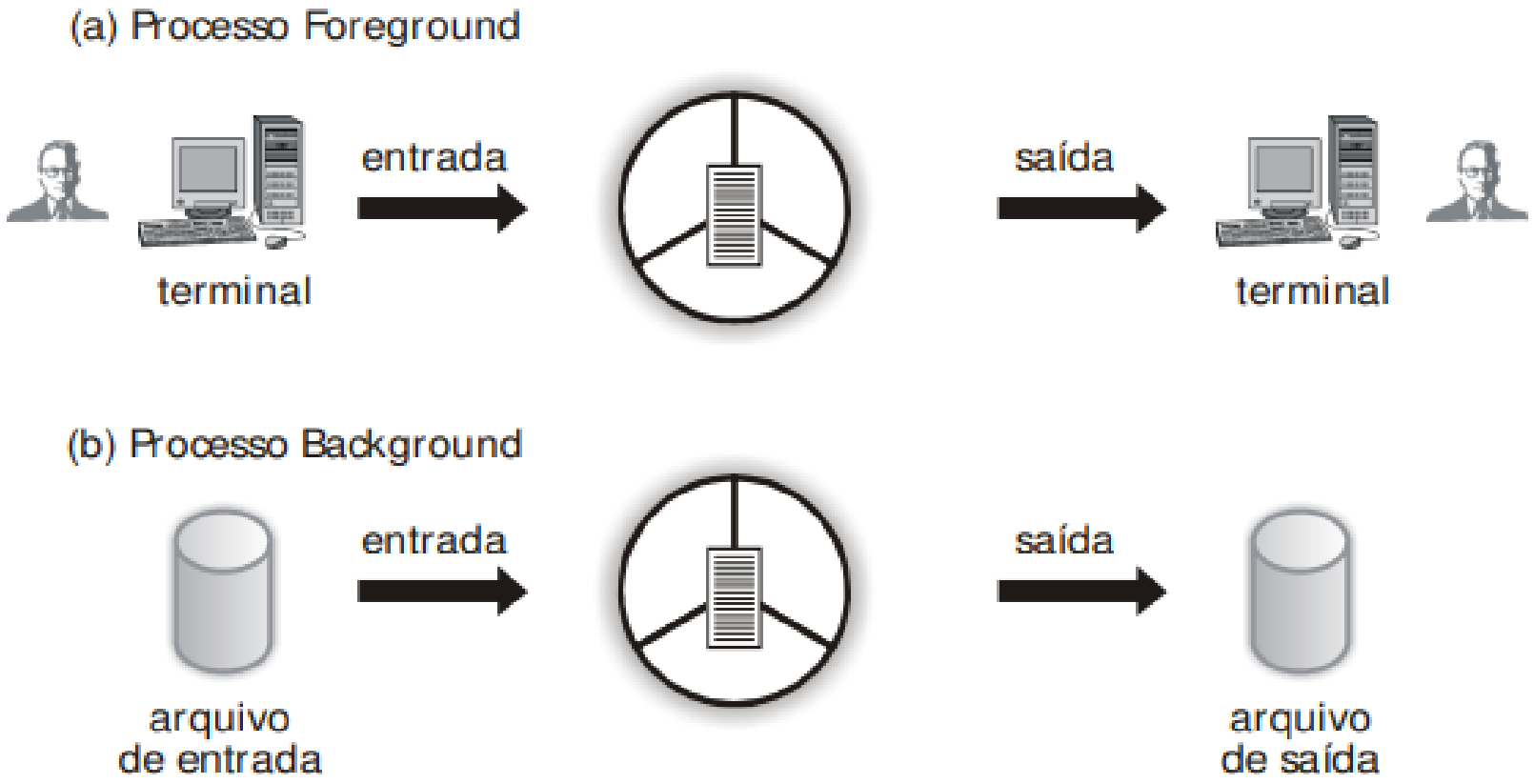
- Um processo possui sempre pelo menos dois canais de comunicação associados a sua estrutura, pelos quais são realizadas todas as entradas e saídas de dados ao longo do seu processamento.

- Os canais de entrada (input) e saída (output) de dados podem estar associados a terminais, arquivos, impressoras e até mesmo a outros processos.

Processos Foreground e Background

- Um processo foreground é aquele que permite a comunicação direta do usuário com o processo durante o seu processamento.
- Um processamento background é aquele que não existe a comunicação com o usuário durante o seu processamento.
  - Nesse caso, os canais de E/S não estão associados a nenhum dispositivo de E/S interativo, mas em geral a arquivos de E/S

Processos Foreground e Background



Formas de Criação de Processos

- Nos sistemas operacionais, como acabamos de ver um processo é uma instância em execução de um programa.
- A criação de processos pode ocorrer de diferentes formas, dependendo do contexto e do objetivo do sistema.
- As principais formas de criação de processos incluem:

- **Inicialização do Sistema (Bootstrapping):** O sistema operacional cria processos essenciais ao iniciar o computador, como o processo do kernel e serviços do sistema.
- **Criação por um Processo Pai:** Um processo pode criar outro através de chamadas do sistema, formando uma hierarquia de processos. O processo pai pode controlar a execução do processo filho.

- **Execução de um Comando pelo Usuário:** Quando um usuário executa um programa na interface gráfica ou no terminal, um novo processo é criado para essa aplicação.
  - **Exemplo:** Abrir o Bloco de Notas no Windows ou executar um script em Python no Linux.
- **Criação por um Serviço do Sistema (Daemon/Background Process):** Alguns processos são criados automaticamente pelo sistema para gerenciar tarefas em segundo plano, como servidores web, impressoras e monitoramento de dispositivos.
- **Criação por um Programa de Aplicação:** Alguns softwares criam novos processos para executar tarefas separadas, como navegadores da web, que geram processos individuais para abas e extensões.

Processos Independentes

- O uso de processos independentes é a maneira mais simples de implementar a concorrência em sistemas multiprogramáveis.
  - Não existe vínculo do processo criado com o seu criador.
  - A criação de um processo independente exige a alocação de um PCB, possuindo contextos de hardware, contextos de software e espaços de endereçamento próprios.

Subprocessos

- Subprocessos são processos criados dentro de uma estrutura hierárquica.
  - Neste modo, o processo criador é denominado de processo pai, enquanto no novo processo é chamado processo-filho.
  - O processo filho fica subordinado ao processo pai.
  - Também possui seu próprio contexto de hardware, software e espaço de endereçamento.

THREADS

- Os processos nada mais são que os programas em execução. Um processo possui um único fluxo de controle.

- Esses processos podem ser divididos em instruções ou grupo de instruções e, cada uma delas ser executada em processadores diferentes. Esse é o conceito de Thread.

- Uma thread é a divisão de um processo em partes menores, também chamada de tarefa.

- A execução de um programa acontece ou por meio de uma ação do SO ou por meio de uma intervenção do usuário (que “abre” o software).

- Quando o usuário abre um software, o SO interpreta a ação e requisita tudo o que é necessário para que esse software seja executado, nesse momento ele é carregado na memória RAM.

- O conceito de thread foi introduzido na tentativa de reduzir o tempo gasto em criação, eliminação e troca de contexto nas aplicações concorrentes, além de economizar recursos do sistema como um todo.

- Threads compartilham o processador da mesma maneira que um processo, ou seja, enquanto uma thread espera por uma operação de E/S, outra thread pode ser executado.

- Todo programa em execução gera um processo e, alguns programas possuem uma árvore de processos, ou seja, cada qual com uma lista de instruções a serem executadas pelo processador.

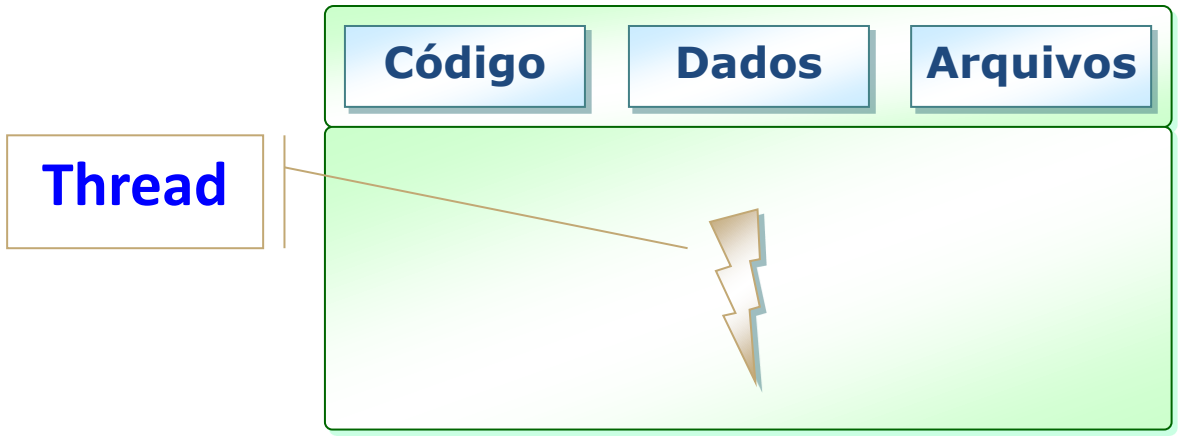
- A lista é composta por linhas de instruções executáveis que informam ao processador os passos a serem executados.

- Threads compartilham o processador da mesma forma que processos. Também passam por mudanças de estados (execução, espera e pronto). Por exemplo, quando uma thread espera uma operação de leitura de arquivo uma outra thread pode ser executada.

- Threads são implementadas internamente na memória principal através de uma estrutura de dados chamada bloco de controle de thread (**Thread control block - TCB**). O TCB armazena mais algumas informações relacionadas ao thread como prioridade, estado de execução e bits de estado.

Monothread

- Nesse ambiente um processo suporta apenas um programa no seu espaço de endereçamento.



Processo tradicional - Uma thread

Multithread

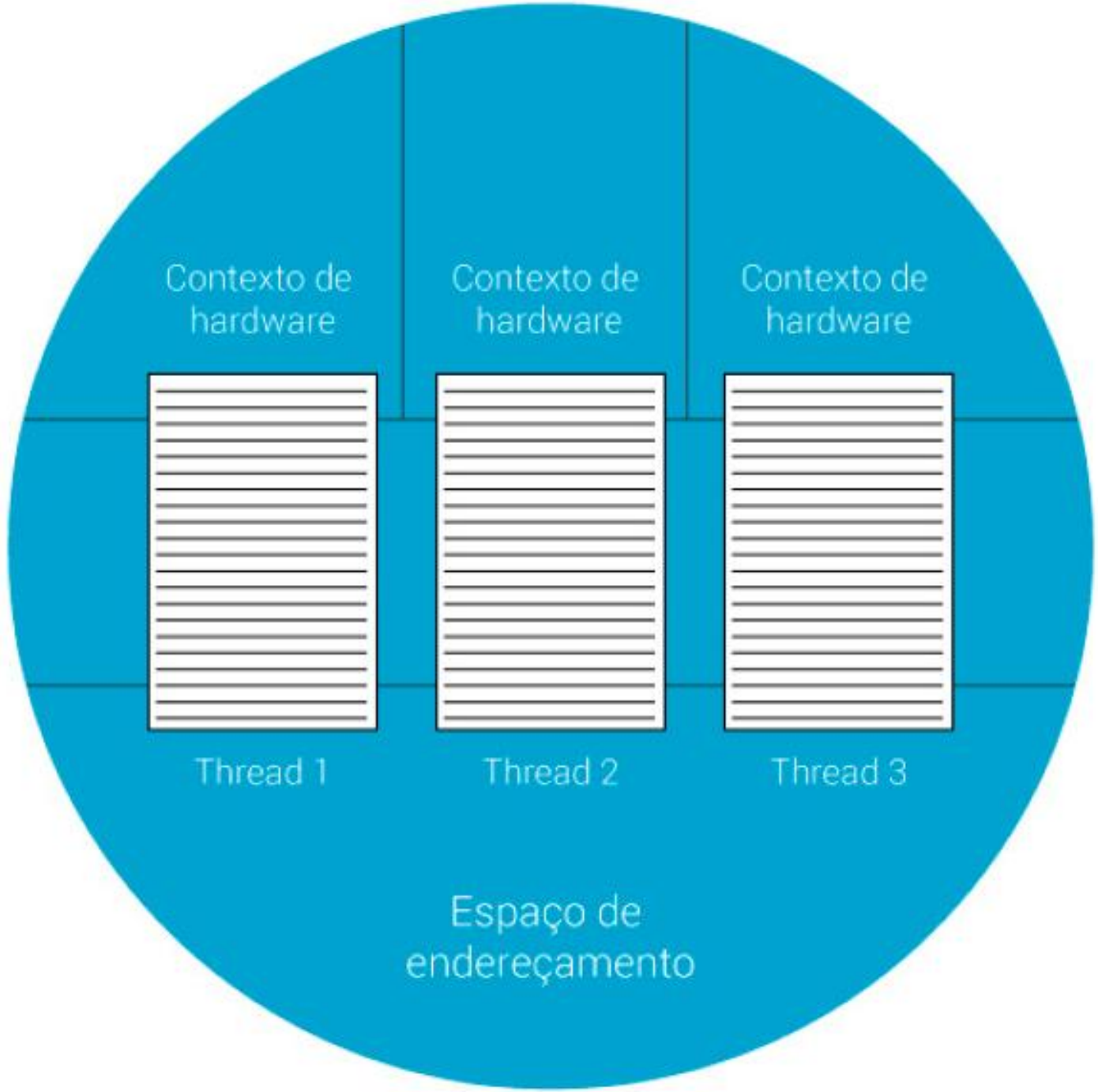
- Nessa configuração o processo tem pelo menos uma thread de execução, mas pode compartilhar o seu espaço de endereçamento com outras threads.

Múltiplas threads

- Em ambiente com múltiplos processadores, cada processo pode responder a várias solicitações (processos) simultaneamente.

- A vantagem no uso de multithreads é a possibilidade de minimizar a alocação de recursos do sistema, tais como processador e memória.

- As threads compartilham contexto de software e espaço de endereçamento, mas o contexto de hardware cada uma tem o seu.


Contexto de hardware
Contexto de hardware
Contexto de hardware
Thread 1
Thread 2
Thread 3
Espaço de endereçamento

- Threads compartilham o espaço dentro de um único processo. Assim, permitem que o compartilhamento de dados entre threads de um mesmo processo seja mais simples e rápido.

- Threads de um único processo compartilham o mesmo espaço de endereçamento, portanto, não há proteção no acesso à memória, permitindo que uma thread possa alterar facilmente dados alheios.

- Por isso, é fundamental que a aplicação implemente mecanismos de comunicação e sincronização entre threads, com o objetivo de garantir o acesso seguro aos dados compartilhados na memória.

- O uso de threads deixa as aplicações mais rápidas.

- Threads dentro de um processo dividem o mesmo espaço de endereçamento, logo, a comunicação entre elas pode ser realizada de forma rápida, causando menos interrupção no SO.

- A grande diferença entre aplicações monothread e multithread está no uso do espaço de endereçamento. Processos são independentes e portanto cada processo possui seu próprio espaço de endereçamento, enquanto que as threads compartilham o mesmo espaço de endereçamento de um processo.

- Este compartilhamento de memória permite que a troca de dados entre threads de um mesmo processo seja simples e rápida se comparado a comunicação de processos em um ambiente monothread.

- Existem diversas abordagens na implementação de threads em um SO, o que influenciará no desempenho, na concorrência e na modularidade das aplicações multithread.

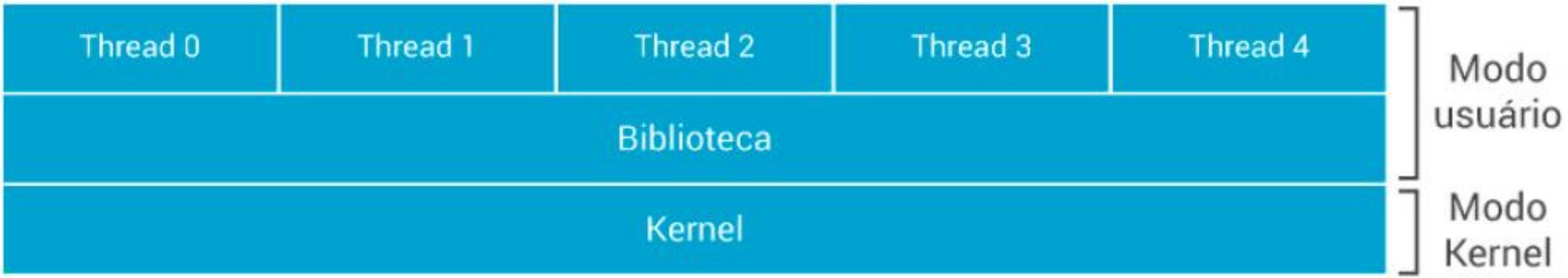
- Threads podem ser oferecidos:
  - Por uma biblioteca de rotinas fora do núcleo do sistema operacional (modo usuário);
  - Pelo núcleo do sistema operacional (modo kernel);
  - Por uma combinação de ambos (modo híbrido).

Thread de Usuário

- É implementada pelos aplicativos, e não pelo SO.

- Para isso, deve haver uma biblioteca de rotinas que permita à aplicação realizar a criação / eliminação de threads, a comunicação e o escalonamento. O SO não tem qualquer responsabilidade sobre essas threads.

- As threads de modo usuário são mais rápidas porque não necessitam acessar o núcleo do SO, evitando assim, a mudança de modo de acesso (usuário-kernel-usuário).



Threads de Kernel

- São implementadas pelo kernel (núcleo do SO) com as system calls (chamadas do sistema), que disponibilizam todo o gerenciamento e comunicação.

- Possuem execução mais lenta, pois precisam fazer a mudança de modo de acesso (usuário-kernel-usuário). São necessárias chamadas a rotinas do sistema (system calls) e, consequentemente, várias mudanças no modo de acesso.

Threads de Kernel

Threads em Modo Híbrido

- A arquitetura de threads em modo híbrido combina as vantagens de threads implementados em modo usuário e modo kernel.

- Um processo pode ter várias threads de modo usuário e, por sua vez, uma thread de modo kernel pode ter várias threads de modo usuário. O kernel reconhece as threads de modo kernel e pode escaloná-las individualmente.

Processos do Sistema Operacional

- Um processo pode ser associado a aplicações dos usuários, no entanto, também pode estar associado ao próprio S.O.

- **Quando processos são utilizados para a implementação de serviços do sistema, estamos retirando código do seu núcleo, tornando-o menor e mais estável.**

- Alguns serviços que o S.O. pode implementar através de processos:
  - Auditoria e segurança;
  - Serviços de rede;
  - Contabilização do uso de recursos;
  - Contabilização de erros;
  - Gerência de impressão;
  - Gerência de Jobs Batch;
  - Temporização;
  - Comunicação de eventos;
  - Interface de comandos (shell).