SISTEMAS OPERACIONAIS

Chamadas ao Sistema

E

Processos

ESTRUTURA DO SISTEMA OPERACIONAL

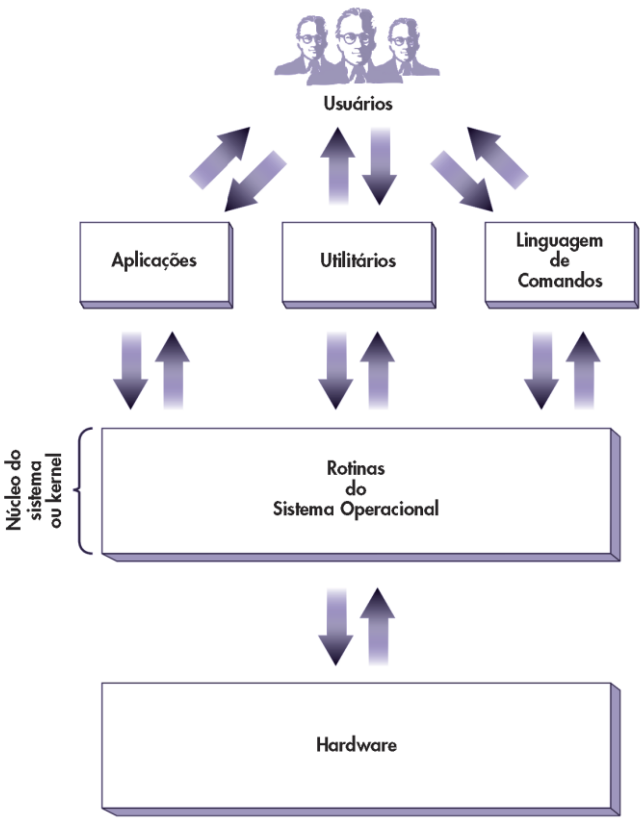


Fig. 4.1 Estrutura do sistema operacional.

Há três maneiras distintas de os usuários se comunicarem com o kernel do sistema operacional. Uma delas é por intermédio das chamadas rotinas do sistema realizadas por aplicações. Além disso, os usuários podem interagir com o núcleo mais amigavelmente por meio de utilitários ou linguagem de comandos. Cada sistema operacional oferece seus próprios utilitários, como compiladores e editores de texto. A linguagem de comandos também é particular de cada sistema, com estruturas e sintaxe próprias.

Fonte: Livro Arquitetura de Sistemas Operacionais, Fracis B.Machado e Luiz P. Maia – LTCEditora

Modo de Acesso do Processador e Rotinas do S. O.

Uma preocupação que surge nos projetos de sistemas operacionais é a implementação de mecanismos de proteção ao núcleo do sistema e de acesso aos seus serviços. Caso uma aplicação, que tenha acesso ao núcleo, realize uma operação que altere sua integridade, todo o sistema poderá ficar comprometido e inoperante. Muitas das principais implementações de segurança de um sistema operacional utilizam um mecanismo presente no hardware dos processadores, conhecido como *modo de acesso*.

Modo de Acesso:
Existem dois modos de acesso ao processador:
Modo usuário;
Modo Kernel
Quando o processador trabalha no modo usuário, uma aplicação só pode executar instruções conhecidas como não privilegiadas.
Já no modo kernel, a aplicação pode ter acesso ao conjunto total de instruções do processador.

Rotinas do sistema operacional:
Compõem o núcleo do sistema oferecendo serviços aos usuários e suas aplicações.
Todas as funções do núcleo são implementadas por rotinas do sistema que possuem em seu código instruções privilegiadas.
Assim, essas rotinas só poderão ser executadas quando o processador estiver no modo kernel

Todo o controle de execução de rotinas do sistema operacional é realizado pelo mecanismo conhecido como **"system call".**
Toda a vez que uma aplicação desejar chamar uma rotina do sistema operacional, o mecanismo **"system call"** é acionado.

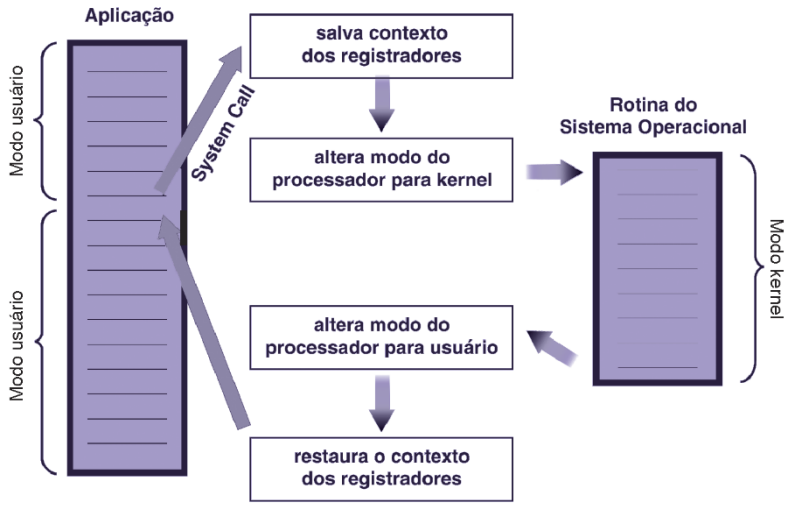


Fig. 4.2 Chamada a uma rotina do sistema (a).

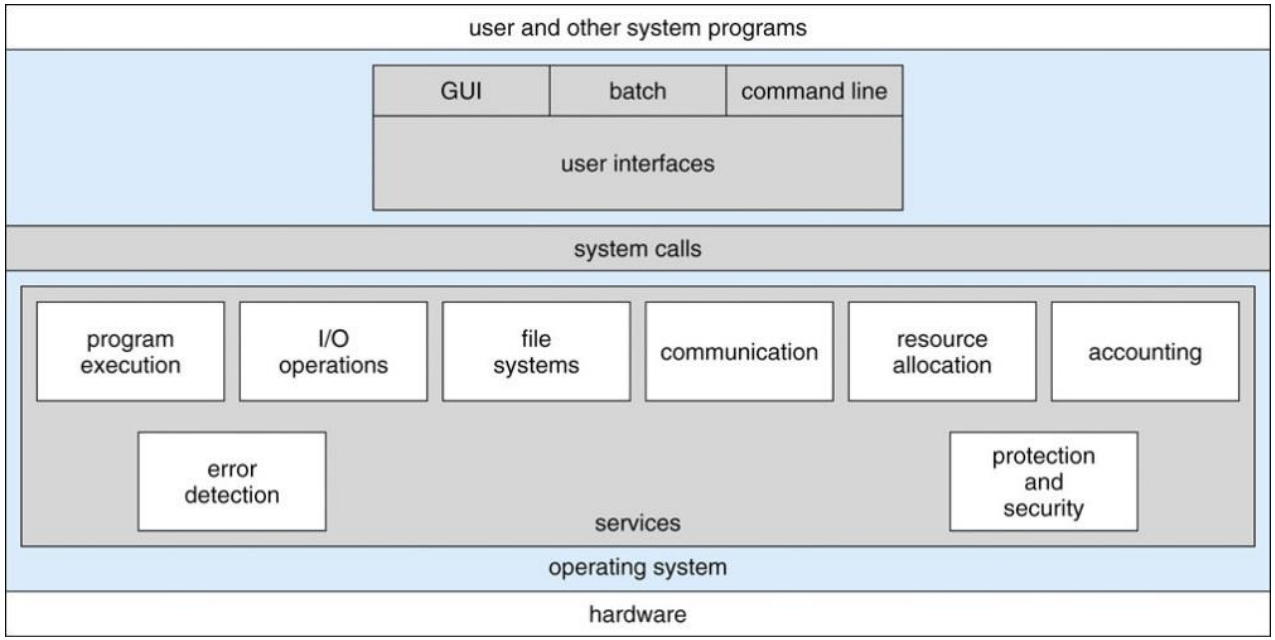
Ver Figura: Tendo a aplicação o devido privilégio para chamar a rotina do sistema, o S.O. salva o conteúdo corrente dos registradores, troca o modo de acesso do processador de usuário para kernel e realiza o desvio para a rotina alterando o registrador do PC com o endereço da rotina chamada. Ao término, o modo de acesso volta a ser "usuário". e os registradores restaurados.

Chamada ao Sistema (system calls)

Chamadas de sistemas são as portas de entrada para o modo Kernel;
É a interface entre os programas do usuário no modo usuário e o Sistema Operacional no modo kernel;
As chamadas diferem de SO para SO, no entanto, os conceitos relacionados às chamadas são similares independentemente do SO.

Grupos de system calls:

- Gerência de processos e threads
- Gerência de memória
- Gerência de sistema de arquivos
- Gerência de dispositivos



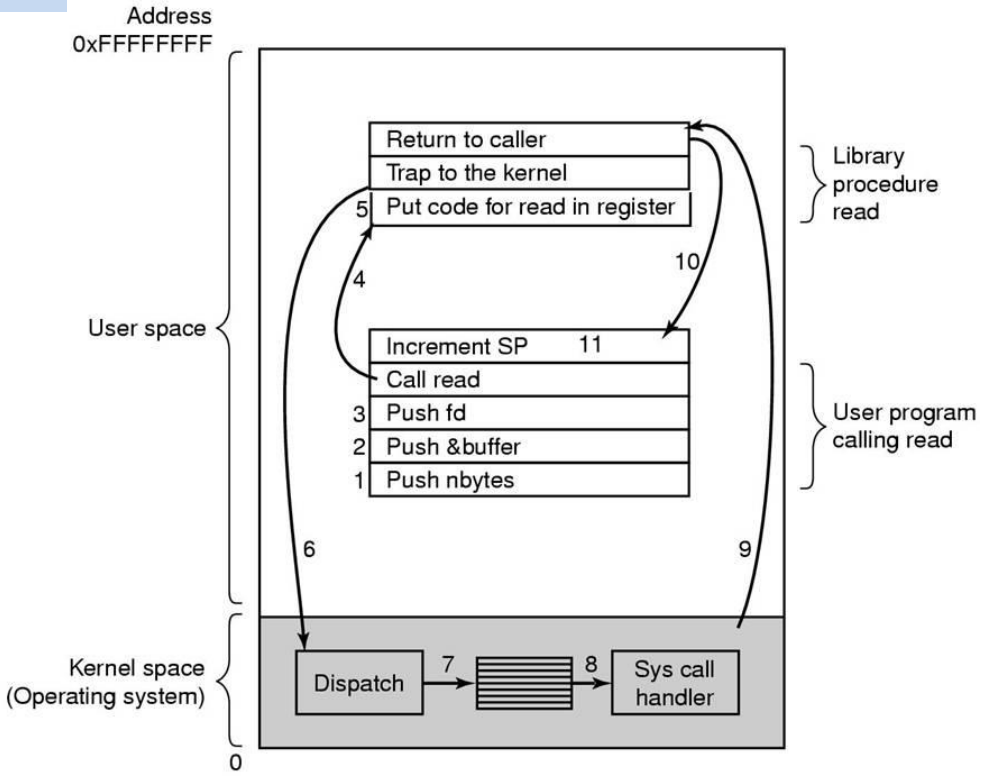
Exemplo de uma System Call

Por exemplo: a instrução:
count = read(fd,buffer,nbytes)

Arquivo a ser lido

Ponteiro para o buffer

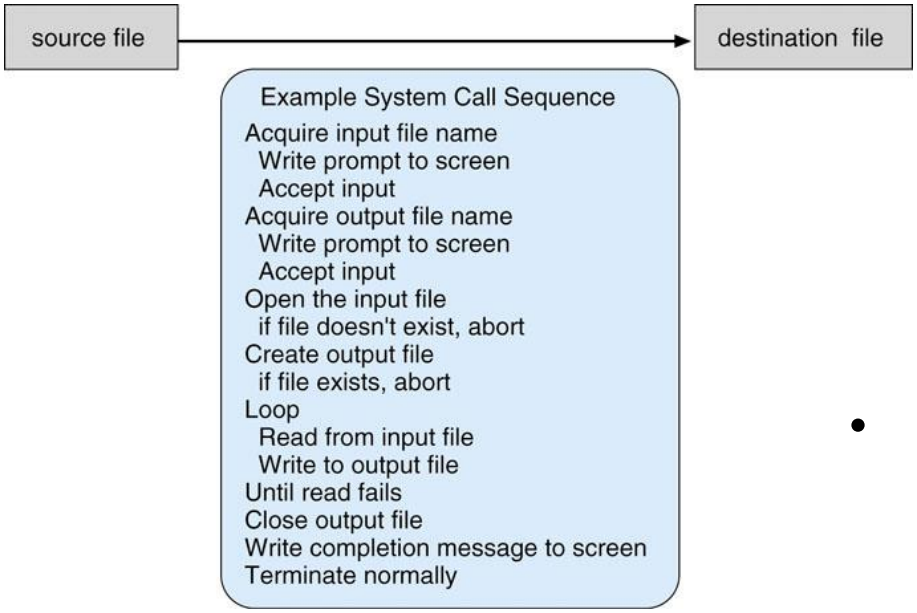
Bytes a serem lidos



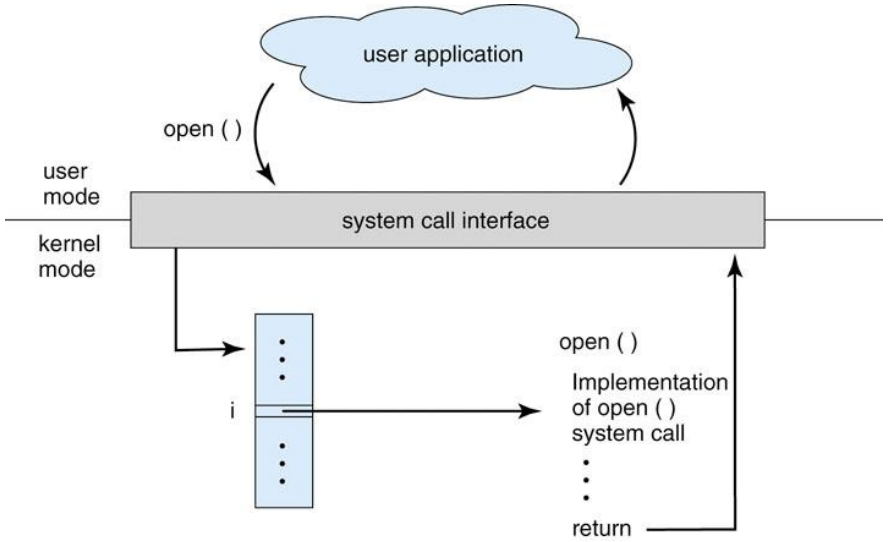
- Os 11 passos para fazer uma chamada ao sistema para o comando “read (arq, buffer, nbytes)”
- Após o passo 5, é executado um TRAP*(instrução que permite acesso ao modo kernel)*, passando do modo usuário para o modo sistema

Exemplo de Chamada ao Sistema

- Exemplo → transferir um arquivo



- Exemplo → abrir um arquivo



Processos : Introdução

Processo = Entidade Ativa, ao contrário do programa, que é uma entidade passiva

O processo constitui-se de:

- Código executável;
- Pilha de Execução;
- Dados;
- Estado;
- Registradores;
- Prioridades, arquivos abertos, quotas, etc.

Processo: características

Cada processo tem sua própria CPU virtual

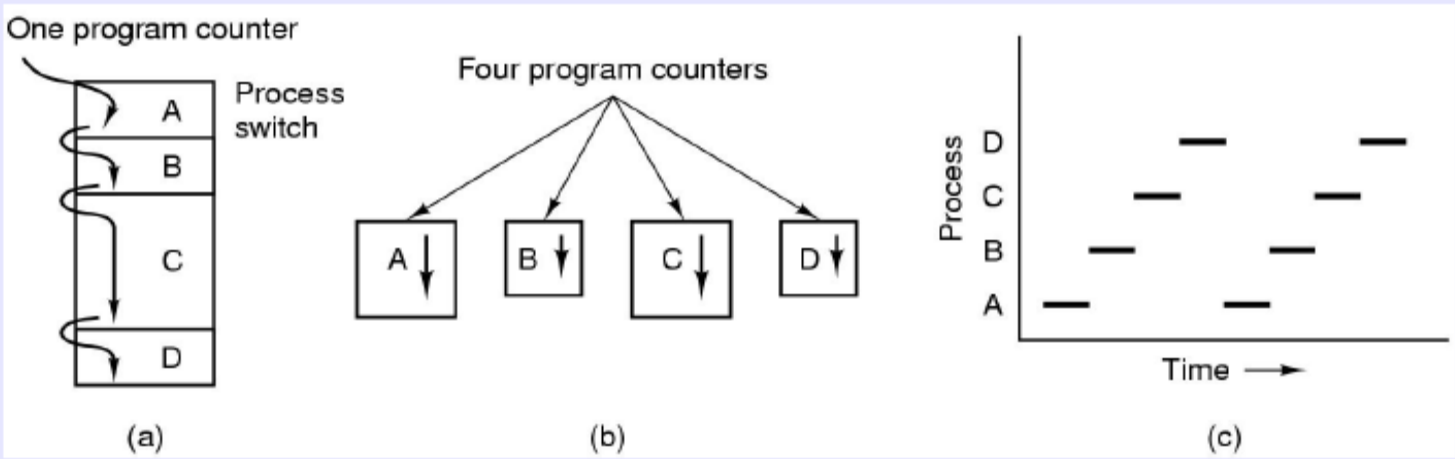
A CPU física é compartilhada por vários processos

- Multiprogramação
- Tempo Compartilhado

O ESCALONADOR seleciona qual processo deve usar a CPU a cada momento e por quanto tempo

Processo: características

- Dependendo do tipo de processo (modelo) podemos classificar o processamento como:
  - Multiprogramação
  - Programação sequencial
  - Programação escalonada



(a) Multiprogramação de 4 processos

(b) Modelo conceitual de quatro processos seqüenciais independentes

(c) Apenas um processo ativo em um dado instante

Processos: Criação

Principais eventos que causam a criação de um processo

- Início do Sistema
- Chamada de Sistema de criação de processo por um processo em execução
- Requisição do usuário para criação de um novo processo
- Início de um Job em lote

Processo: finalização

- Um processo pode ser finalizado devido a:

Saída Normal (voluntária)

Saída por erro (voluntária)

Erro fatal (involuntária)

Cancelamento por outro processo

Processos

Sistemas operacionais devem ser capazes de executar certas operações de processo, entre elas:

- criar um processo
- destruir um processo
- suspender um processo
- retomar um processo
- alterar a prioridade de um processo
- bloquear um processo
- acordar um processo
- despachar um processo
- habilitar um processo a se comunicar com outro (denominado comunicação interprocessos)

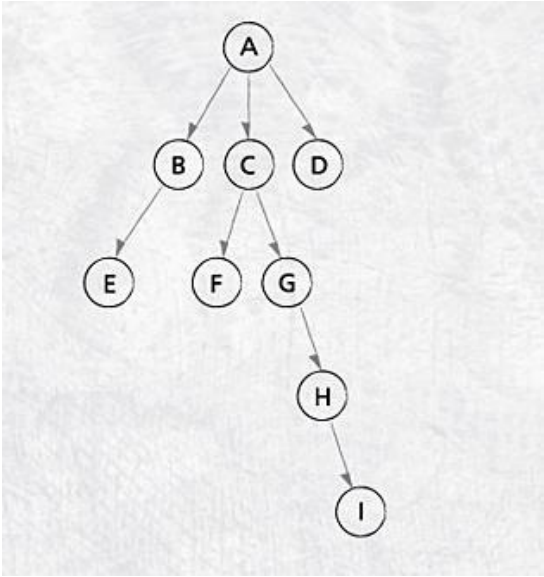
Processo: hierarquia

- Processos ao serem criados podem criar outros processos criando um hierarquia entre eles.
- Desta forma, chamamos de processo-pai aquele processo que criou outro e processo-filho ao processo que foi criado.

Um processo pai cria seu(s) processo(s) filho(s), um processo filho pode criar seus próprios processos

O Unix chama esta hierarquia de "grupo de processos"

O Windows não tem esse conceito, todos os processos estão no mesmo nível



Processos: características

- Cada processo possui seu próprio **espaço de endereço**.

- **Espaço de endereço** consiste em:
  - região de texto
    - Armazena o código que o processador executa.
  - região de dados
    - Armazena variáveis e memória alocada dinamicamente usadas pelo processo durante a execução.
  - região de pilha
    - Armazena instruções e variáveis locais para chamadas ativas ao procedimento

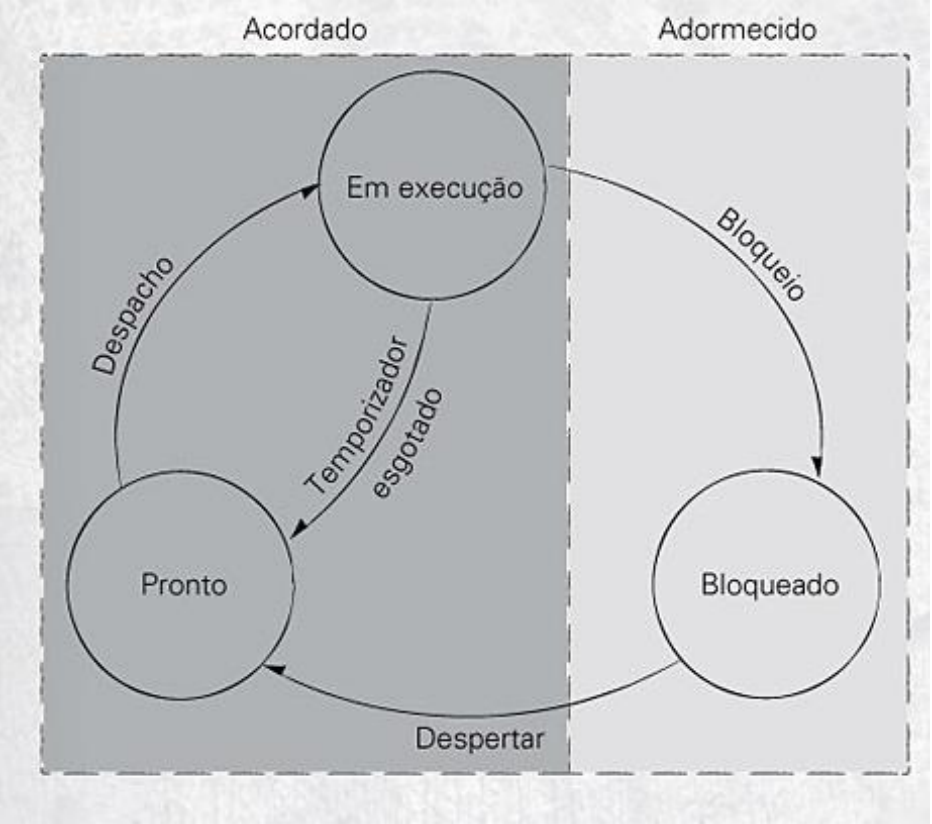
Processos: características

- Em um sistema uniprocessador, apenas um processo pode ser executado por vez.
  - Mas vários outros processos podem estar no estado pronto ou bloqueado.

- O S.O. mantém:
  - uma lista de prontos (processos no estado pronto) → organizada por ordem de prioridade.
  - uma lista de bloqueados (processos no estado bloqueado) → é tipicamente desordenada *(não há uma prioridade para um processo passar do estado "bloqueado" para o estado "pronto")*

Processo: fases (estados)

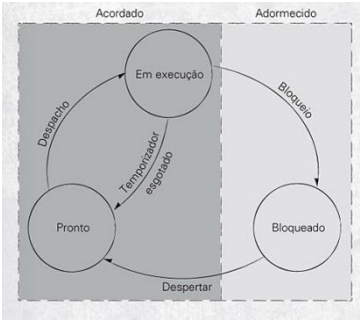
- Um processo, após ser criado, pode se encontrar em três estados:
  - Execução
  - Pronto
  - bloqueado



Durante seu tempo de vida um processo passa por uma série de **estados de processo** distintos. Vários eventos podem fazer que um processo mude de estado. Diz-se que um processo está ***executando*** (ou seja, no **estado *de execução***) se estiver executando em um processador. Diz-se que um processo está ***pronto*** (ou seja, em **estado *'de pronto'***) quando poderia executar em um processador se houvesse algum disponível. Diz-se que um processo está ***bloqueado*** (ou seja, no **estado *bloqueado***) se estiver esperando que algum evento aconteça (tal como um **evento de conclusão de E/S**, por exemplo) antes de poder prosseguir.

Processos: transição entre estados

Quando um usuário executa um programa, processos são criados e inseridos na lista de prontos. Um processo vai passando para o topo da lista à medida que outros concluem sua vez de usar o processador. Quando um processo chega ao topo da lista e há um processador disponível, aquele processo é designado a um processador e diz-se que ele fez uma **transição de estado**, passando do estado de *pronto* para o estado de *execução* (Figura A1). O ato de designar um processador ao primeiro processo da lista de prontos é denominado **despacho**, e é realizado por uma entidade do sistema denominada **despachante**. Diz-se que processos que estão nos estados de *pronto* ou de *execução* estão acordados porque disputam ativamente tempo de processador. O sistema operacional gerencia transições de estado para melhor servir aos processos no sistema. Para evitar que qualquer um dos processos monopolize o sistema, acidental ou maliciosamente, o sistema operacional estabelece um **relógio de interrupção em hardware** (também denominado **temporizador de intervalo**) que permite que o processo execute durante um intervalo de tempo específico ou **quantum**. Se o processo não devolver o processador voluntariamente antes que o intervalo de tempo expire, o relógio de interrupção gera uma interrupção, fazendo que o sistema operacional obtenha o controle do processador (interrupção síncrona). Então o sistema operacional muda o estado do processo, que estava anteriormente *em execução*, para *pronto* e despacha o primeiro processo da lista de prontos, mudando seu estado de *pronto* para *em execução*. Se um processo *em execução* iniciar uma operação de entrada/saída antes do seu quantum expirar e, conseqüentemente, tiver de esperar que a operação E/S seja concluída antes de poder usar o processador novamente, o processo *em execução* entregará voluntariamente o processador. Nesse caso, diz-se que o processo **bloqueou** a si mesmo, deixando em suspenso a conclusão da operação de E/S. Diz-se que processos no estado *bloqueado* estão adormecidos porque não podem executar mesmo que um processador fique disponível. O único outro estado de transição permitido em nosso modelo de três estados ocorre quando uma operação de E/S (ou algum outro evento pelo qual o processo esteja esperando) é concluído. Nesse caso o sistema operacional promove a transição do processo do estado *bloqueado* para o estado de *pronto*.

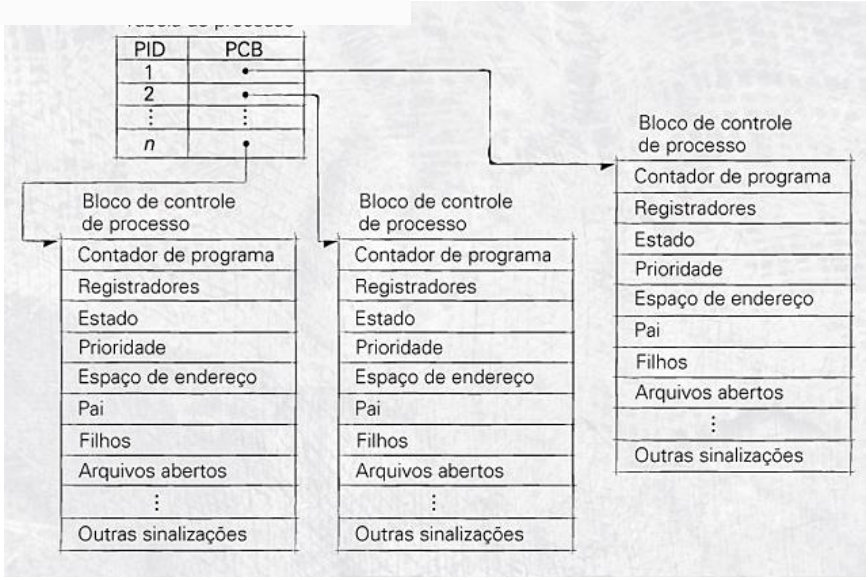


Processos: PCB (*process control block*)

O sistema operacional normalmente executa diversas operações quando cria um processo. Primeiro, deve ser capaz de identificar cada processo; portanto, designa ao processo um **número de identificação de processo** (*Process Identification Number* – **PID**). Em seguida o sistema operacional cria um **bloco de controle de processo** (*Process Control Block* – **PCB**), também denominado **descritor de processo**, que mantém as informações que o sistema operacional necessita para gerenciar o processo. Os PCBs comumente incluem informações como:

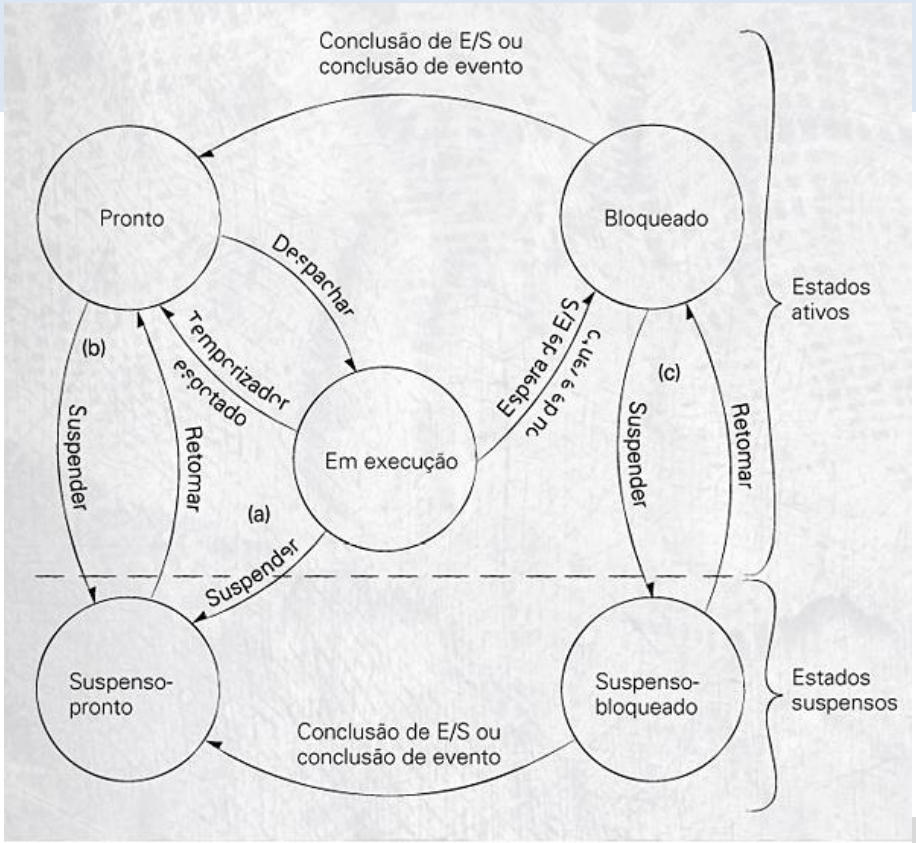
PID
estado do processo (por exemplo, *em execução, pronto* ou *bloqueado*)
contador de programa (um valor que determina qual instrução o processo deve executar em seguida)
prioridade de escalonamento
credenciais (dados que determinam os recursos que esse processo pode acessar)
um ponteiro para o **processo-pai** (o processo que criou esse processo)

ponteiros para os **processos-filho** (processos criados por esse processo), caso existam
ponteiros para localizar os dados e instruções do processo na memória
ponteiros para recursos alocados (tais como arquivos)



Processos: estados “suspensos”

- Muitos S.O. Permitem que administradores, usuários ou mesmo um processo suspedam um processo.
  - Um processo suspenso sai da disputa por tempo de um processador sem ser destruído.
  - O motivo é permitir que, por exemplo, um adm. que desconfie de um processo (questões de segurança) possa suspendê-lo e posteriormente analisá-lo para detetar possível ameaça a segurança do sistema ou com a finalidade de depuração de software.
- No diagrama a seguir pode-se ver a inclusão dos estados :
  - suspenso-pronto.
  - suspenso-bloqueado.



Interrupções

As interrupções ou exceções são eventos inesperados que podem ocorrer enquanto um programa roda (processo), este evento causa o desvio da execução

A interrupção é o mecanismo que tornou possível a implementação da concorrência nos SO

A interrupção é gerado por eventos assíncronos externos ao programa

Cada instrução é testada, quando ocorre uma interrupção o conteúdo da CPU é salvo, é identificada a interrupção, o tratador de interrupções é acionado e realiza as operações necessárias sobre aquela interrupção, quando ele acaba o programa deve retornar a sua execução normal, o dados da cpu são recuperados e programa volta a processar

As interrupções assíncronas podem ocorrer múltiplas vezes

Enquanto uma interrupção é tratada outras podem ocorrer, estas últimas não receberão atenção e são chamadas interrupções mascaráveis

Para que as interrupções sejam todas atendidas alguns processadores podem identificar prioridades nas interrupções

Exceção

A exceção ocorre devido a execução de uma instrução interna ao programa, como por exemplo uma divisão por zero.

Algumas linguagens de programação permitem que você tenha blocos para realizar o tratamento de instruções, ex: Delphi com o bloco “Try...Except...end”

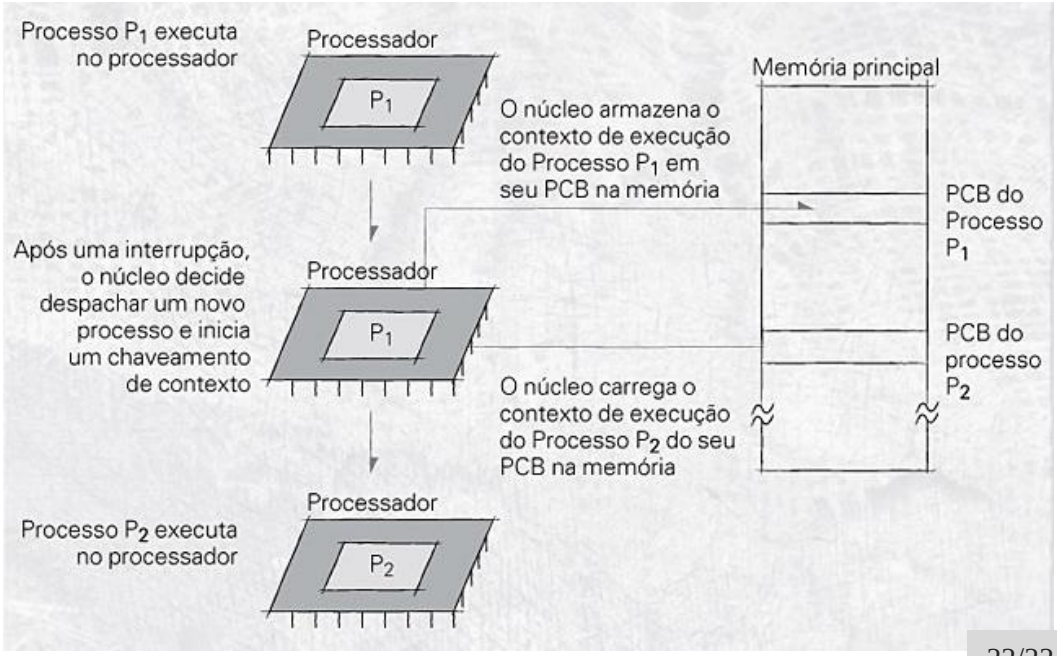
Processo: Chaveamento de contexto

- O S.O. realiza um chav. de contexto para interromper um processo em execução e começar a executar um processo previamente pronto.
- Para realizar o chav. de contexto o núcleo deve:
  - Salvar o **contexto de execução** do processo em *execução* em seu PCB ;
  - Carregar o **contexto de execução** anterior do processo *pronto* a partir do PCB desse último processo.
- Chav. de contexto é puro custo adicional e ocorre tão frequentemente que os S.O. devem minimizar o tempo de chaveamento de contexto. Isso pelo fato que o processador não pode realizar nenhuma computação útil (só executa tarefas essenciais para o S.O. mas não executa nenhuma instrução em nome de qquer.processo).

Na arquitetura Intel 32 (***IA-32***) o S.O. Despacha um novo processo especificando a localização do PCB na memória.

O processador realiza um chav.de contexto salvando o contexto de exec. do processo que estava em execução anteriormente.

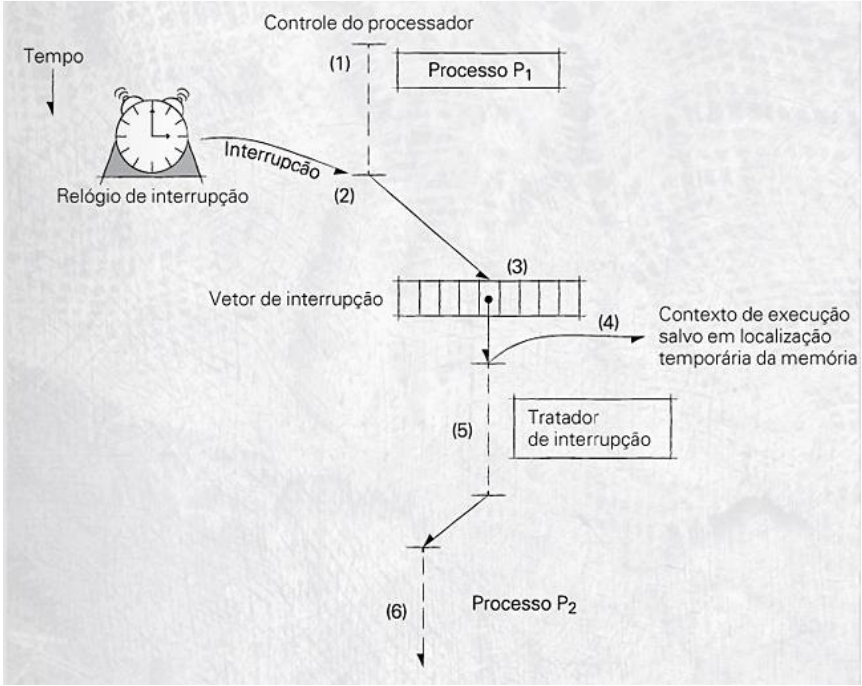
A ***IA-32*** não fornece instruções para salvar e restaurar o contexto de exec. De um processo.



Linha de Interrupções

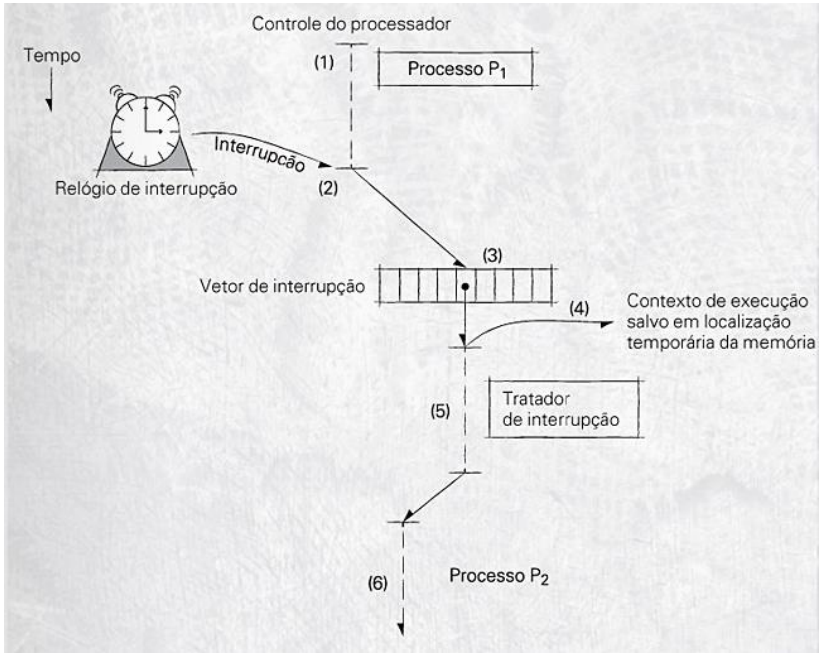
1. A linha de interrupção, uma conexão elétrica entre a placa principal e um processador, fica ativa — dispositivos como temporizadores, placas periféricas e controladores enviam sinais que ativam a linha de interrupção para informar a um processador que ocorreu um evento (por exemplo, um período de tempo passou ou uma requisição de E/S foi concluída). A maioria dos processadores contém um controlador de interrupção que organiza as interrupções segundo suas prioridades, para que as mais importantes sejam atendidas em primeiro lugar. Outras interrupções são enfileiradas até que todas as de prioridade mais alta tenham sido atendidas.

2. Após a linha de interrupção tornar-se ativa, o processador conclui a execução da interrupção corrente e, então, faz uma pausa na execução do processo corrente. Para fazer essa pausa, o processador precisa salvar informações suficientes para que o processo possa ser retomado no lugar exato e com as informações do registrador corretas. Nos antigos sistemas da IBM, esses dados ficavam contidos em uma estrutura de dados denominada palavra de estado de programa (*Program Status Word* — PSW). Na arquitetura Intel IA-32, esse estado de processo é denominado segmento de estado de tarefa (*Task State Segment* — TSS). O TSS é tipicamente armazenado no PCB de um processo.[17]



Linha de Interrupções

3. O processador, então, passa o controle ao tratador de interrupção apropriado. Para cada tipo de interrupção é designado um único valor o qual o processador usa como um índice no **vetor de interrupção**, que se caracteriza por um conjunto (array) de ponteiros para tratadores de interrupção. O vetor de interrupção localiza-se na memória a qual os processos não podem acessar, portanto, processos sujeitos a erro não podem modificar seu conteúdo.
4. O tratador de interrupção executa as ações apropriadas com base no tipo de interrupção.
5. Após o tratador de interrupção concluir, o estado do processo interrompido (ou de algum outro 'processo seguinte', se o núcleo iniciar um chaveamento de contexto) é restaurado.
6. O processo interrompido (ou qualquer outro 'processo seguinte') é executado. É responsabilidade do sistema operacional determinar se é o processo interrompido ou algum outro 'processo seguinte' que é executado. Essa decisão importante, que pode causar impacto significativo sobre o nível de serviço que cada aplicação recebe, é discutida no "Escalonamento de processador". Por exemplo, se a interrupção sinalizar a conclusão de um evento de E/S que provocou a transição de um processo de alta prioridade de *bloqueado* para *pronto*, o sistema operacional poderá preterir o processo interrompido e despachar o processo de alta prioridade.



Tipos de Interrpções e Exceções

Tipo de interrupção	*Descrição das interrupções de cada tipo*
E/S	São iniciadas pelo hardware de entrada/saída. Notificam ao processador que o status de um canal ou dispositivo mudou. Interrupções de E/S são causadas quando uma operação de E/S é concluída, por exemplo.
Temporizador	Um sistema pode conter dispositivos que geram interrupções periodicamente. Essas interrupções podem ser usadas para tarefas como controle de tempo e monitoração de desempenho. Temporizadores também habilitam o sistema operacional a determinar se o quantum de um processo expirou.
Interrupções interprocessadores	Essas interrupções permitem que um processador envie uma mensagem a outro em um sistema multiprocessador.

Classes de exceções	*Descrição das exceções de cada classe*
Falha	Causadas por uma vasta gama de problemas que podem ocorrer, enquanto as instruções em linguagem de máquina de um programa são executadas. Entre esses problemas, estão a divisão por zero, dados (que estão sendo utilizados) no formato errado, tentativa de executar um código de operação inválido, tentativa de referência a uma localização de memória que está fora dos limites da memória real, tentativa de um processo de usuário de executar uma instrução privilegiada, e tentativa de referir-se a um recurso protegido.
Desvio	Gerados por exceções como transbordamento (quando o valor armazenado por um registrador excede à capacidade do registrador) e quando o controle do programa chega a um ponto de parada no código.
Aborto	Ocorrem quando o processador detecta um erro do qual o processo não pode se recuperar. Por exemplo, quando a própria rotina de tratamento de exceções causa uma exceção, o processador pode não conseguir tratar ambos os erros seqüencialmente, o que é denominado exceção de dupla falha, que extingue o processo que lhe deu início.

Exercícios

Por que o espaço de endereço de um processo é dividido em várias regiões?

(V/F) Os termos 'processo' e 'programa' são sinônimos.

(V/F) A qualquer dado instante somente um processo pode executar instruções em um computador.

Um processo entra no estado de *bloqueado* quando está esperando que um evento ocorra. Cite diversos eventos que podem fazer um processo entrar em estado de *bloqueado*.

Como o sistema operacional impede que um processo monopolize um processador?

Qual a diferença entre processos que estão acordados e processos que estão adormecidos?

Qual a finalidade da tabela de processos?

(V/F) A estrutura de um PCB depende da implementação do sistema operacional.

(V/F) Um processo pode ter um número zero de processos-pai.

Por que é vantajoso criar uma hierarquia de processos em vez de uma lista encadeada?

Quais são as três maneiras pelas quais um processo pode chegar ao estado *suspenso-pronto*?

Em qual cenário é melhor suspender um processo em vez de abortá-lo?

Por que um sistema operacional deve minimizar o tempo requerido para realizar um chaveamento de contexto?

O que significa interrupção síncrona?

Cite uma alternativa para interrupção e explique por que raramente é usada.

Por que as localizações dos tratadores de interrupção geralmente não são armazenadas em uma lista encadeada?

Por que o contexto de execução do processo é salvo na memória enquanto o tratador de interrupção executa?