Processos e Algoritmos

Prof. Dr. João Carlos Lopes Fernandes
E-mail: joao.fernandes1@docente.unip.br

Prof. Dr. João Carlos Lopes Fernandes
E-mail: joao.fernandes1@docente.unip.br

Conceitos Básicos

- Processo: um programa de computador em execução.

- Espaço de Endereçamento: área de memória alocada a um processo. Dois endereços podem ser numericamente iguais, mas se referirem a locais diferentes se pertencerem a *espaços de endereçamento* diferentes.

- Proteção: manutenção da integridade dos dados de um processo na presença de outros processos. Implementada em conjunto pelo *hardware* e sistema operacional.

Escalonamento de processos

- Quando um ou mais processos estão prontos para serem executados, o sistema operacional deve decidir qual deles vai ser executado primeiro

- A parte do sistema operacional responsável por essa decisão é chamada escalonador, e o algoritmo usado para tal é chamado de algoritmo de escalonamento

- Para que um processo não execute tempo demais, praticamente todos os computadores possuem um mecanismo de relógio (clock) que causa uma interrupção, periodicamente

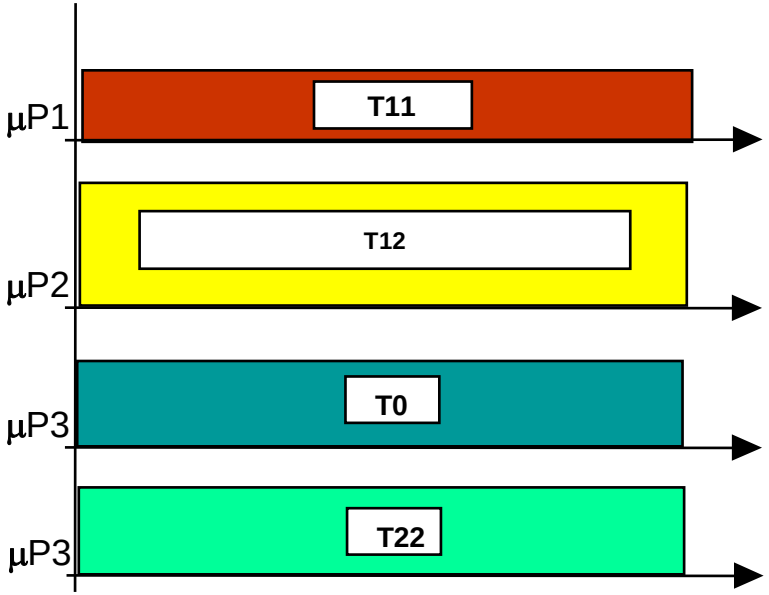
Características de Escalonamento

- Justiça (fairness)
  - Todos os processos têm chances iguais de uso dos processador
- Eficiência
  - Taxa de ocupação do processador ao longo do tempo
- Tempo de Resposta
  - Tempo entre a ocorrência de um evento e o término da ação correspondente
- Turnaround
  - “Tempo de resposta” para usuários em batch
- Throughput
  - Núm de “jobs” (processos) executados por unidade de tempo

Escalonamento de Processos

Abstração

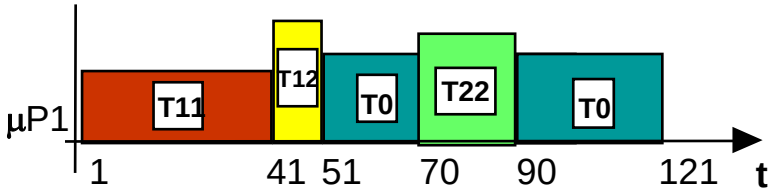
- Uma máquina para cada processo
- Paralelismo real



Escalonamento de Processos

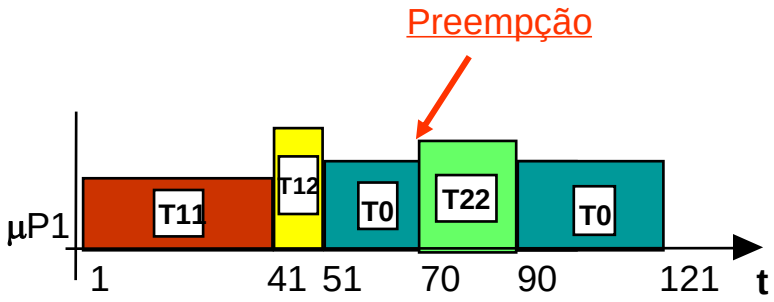
Realidade

- Compartilhamento do tempo
- Pseudoparalelismo



Escalonamento Preemptivo

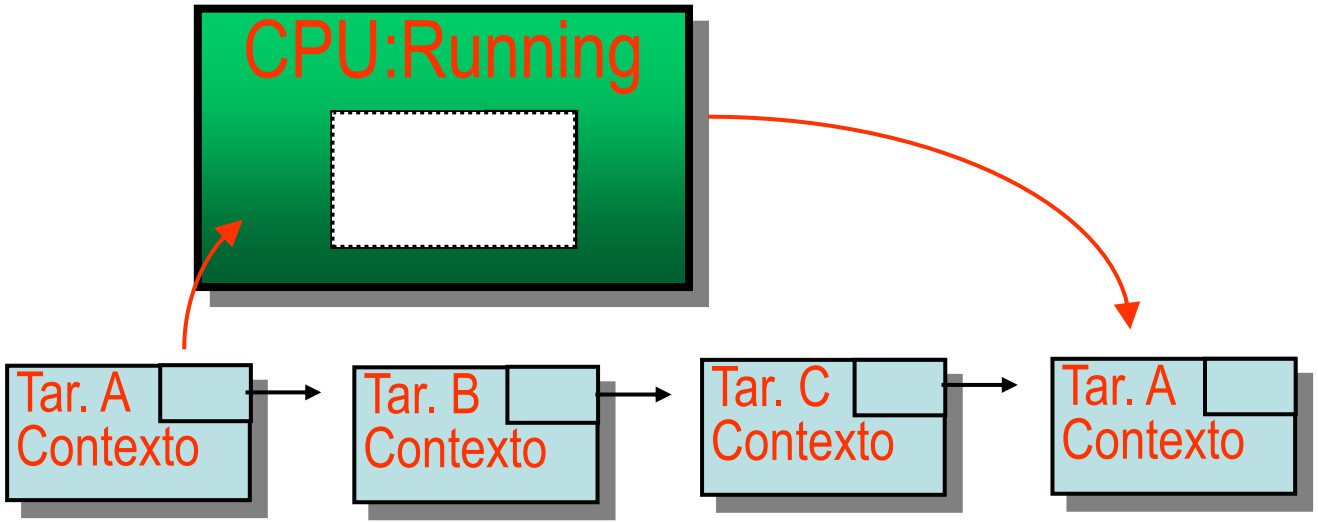
- Permite a suspensão temporária de processos
- Quantum ou time-slice: período de tempo durante o qual um processo usa o processador a cada vez



- Quantum grande:
  - Diminui núm. de mudanças de contexto e overhead do S.O.
  - Ruim para processos interativos

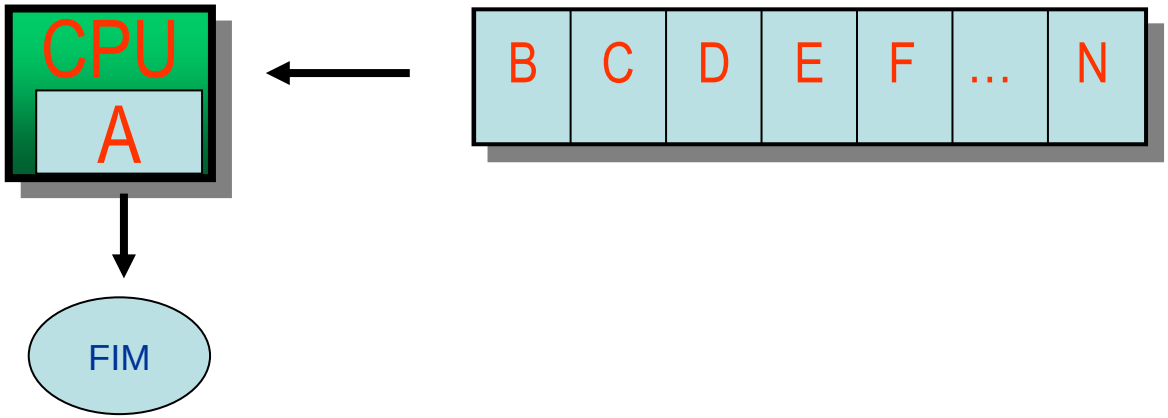
Escalonamento Round-Robin

- Uso de uma lista de processos sem prioridade
- Escalonamento preemptivo
- Simples e justo
- Bom para sistemas interativos



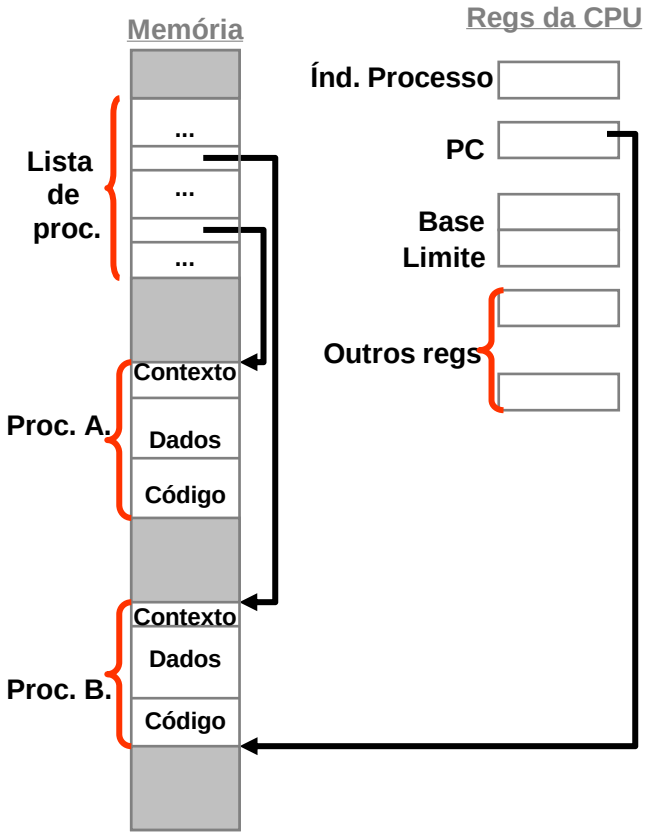
Escalonamento First-In First-Out (FIFO)

- Uso de uma lista de processos sem prioridade
- Escalonamento não-preemptivo
- Simples e justo
- Bom para sistemas em batch

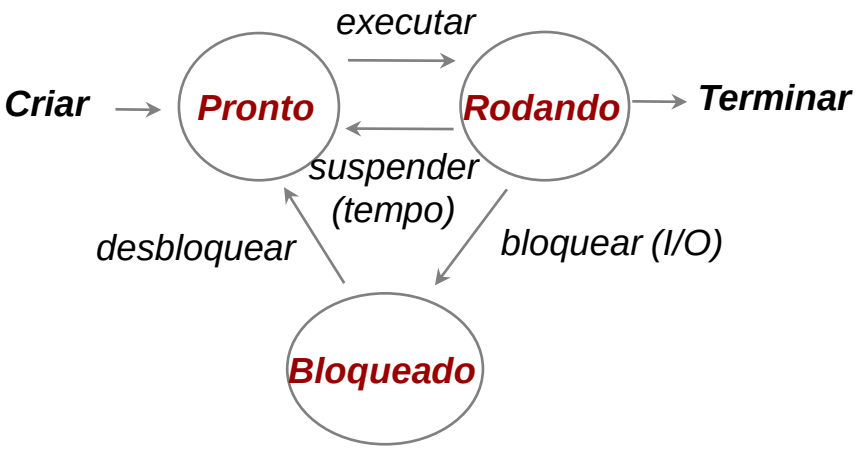


Multiprocessamento

- O índice do processo contem o apontador para a lista de processos
- Uma troca de processos consiste em trocar o valor dos registradores de contexto da CPU



Estados de um Processo

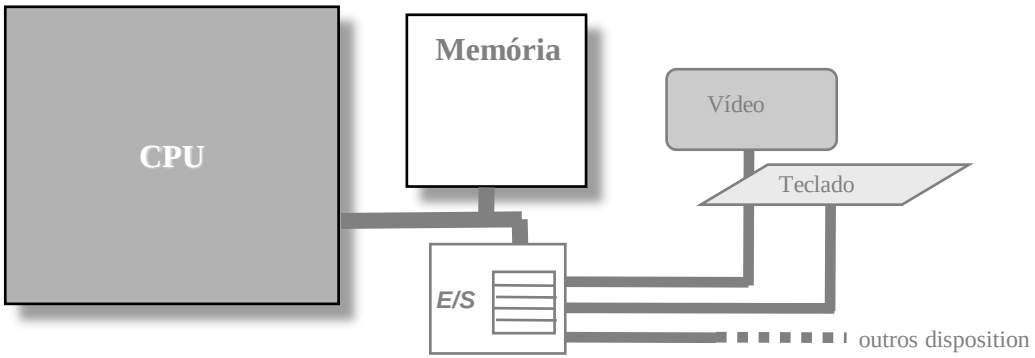


Contexto

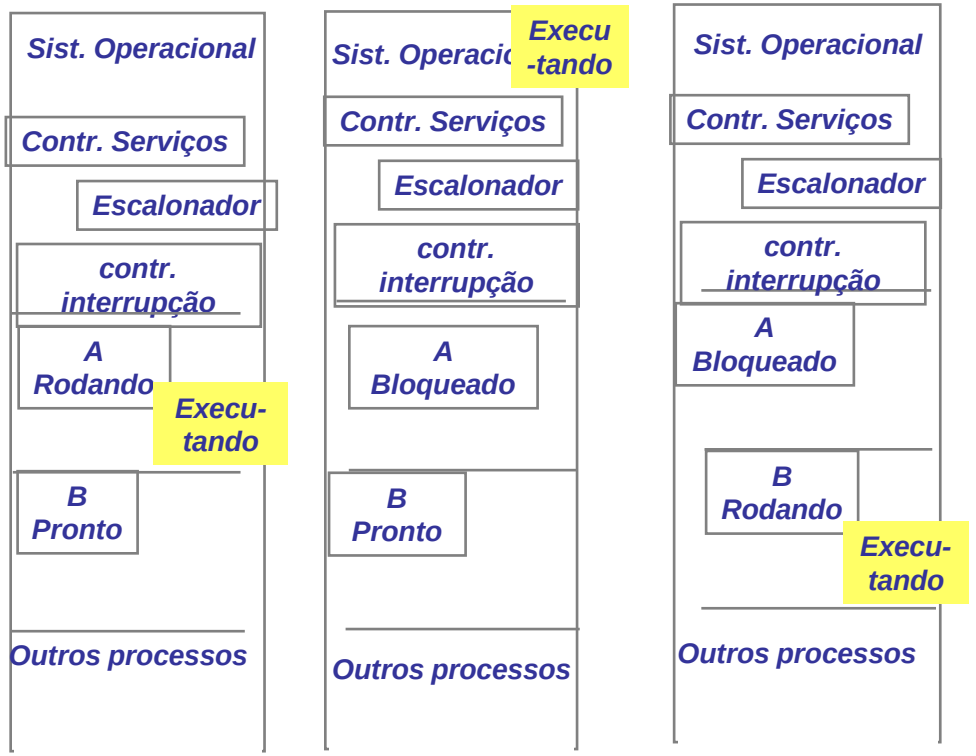
ID do Processo
Estado
Prioridade
Program Counter
Ponteiros da Memória
outros regs.
Status de E/S
Informações gerais   • ***tempo de CPU***   • ***limites, usuário, etc.***

Contexto de um Processo

- Informações
  - CPU: Registradores
  - Memória: Posições em uso
  - E/S: Estado das requisições
  - Estado do processo: Rodando, Bloqueado, Pronto
  - Outras



Exemplo



Processo A parou:

- ***Req. serviço ao S.O.***
- ***Interrupção de A Ex. erro***
- ***Interrupção de outra fonte. Ex. I/O***
- ***Tempo acabou***

Criação de Processos

- Principais eventos que levam à criação de processos
  - Início do sistema
  - Execução de chamada ao sistema de criação de processos
  - Solicitação do usuário para criar um novo processo
  - Início de um job em lote

Término de Processos

- Condições que levam ao término de processos
  - Saída normal (voluntária)
  - Saída por erro (voluntária)
  - Erro fatal (involuntário)
  - Cancelamento por um outro processo (involuntário)

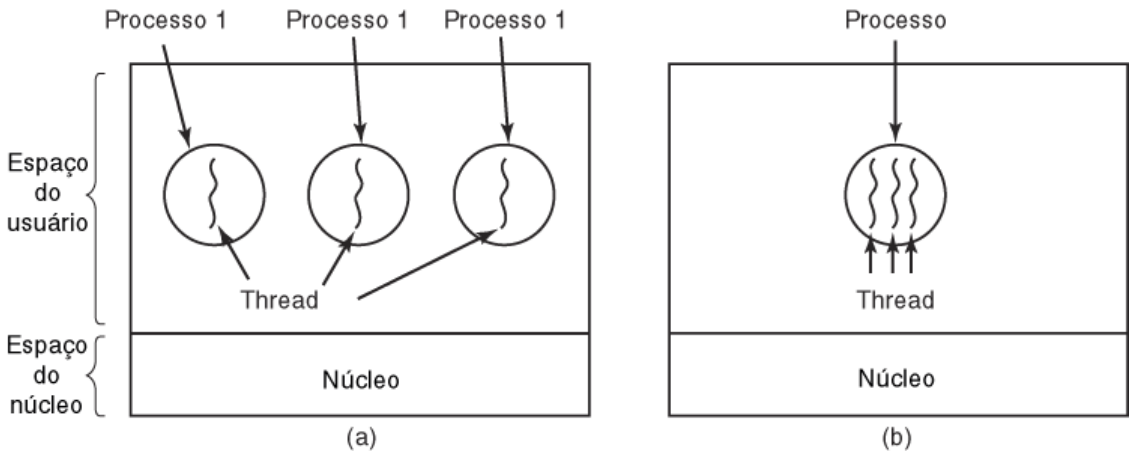
Hierarquias de Processos

- Processo “pai” cria um processo “filho”, processo filho pode criar seu próprio processo …
- Formando uma hierarquia
  - UNIX chama isso de “grupo de processos”
- Windows não possui o conceito de hierarquia de processos
  - Todos os processos são criados iguais (sem conceito de “pai” e “filho”)

Threads: Motivação Concorrência

- Problemas:
  - Programas que precisam de mais poder computacional
  - Dificuldade de implementação de CPUs mais rápidas
- Solução:
  - Construção de computadores capazes de executar várias tarefas simultaneamente

Threads
O Modelo de Thread (1)



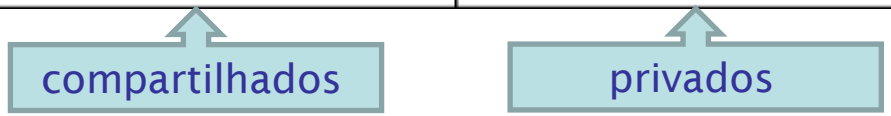
(a) Três processos, cada um com uma thread

(b) Um processo com três threads

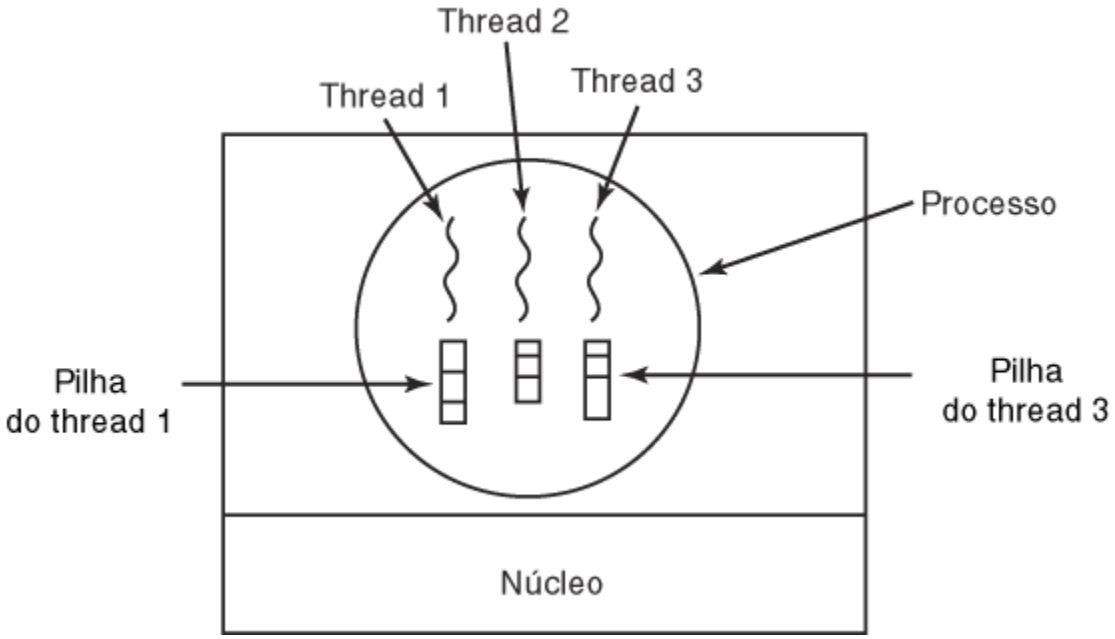
O Modelo de Thread (2)

Contexto

Itens por processo	Itens por thread
Espaço de endereçamento	Contador de programa
Variáveis globais	Registradores
Arquivos abertos	Pilha
Processos filhos	Estado
Alarmes pendentes	
Sinais e tratadores de sinais	
Informação de contabilidade	

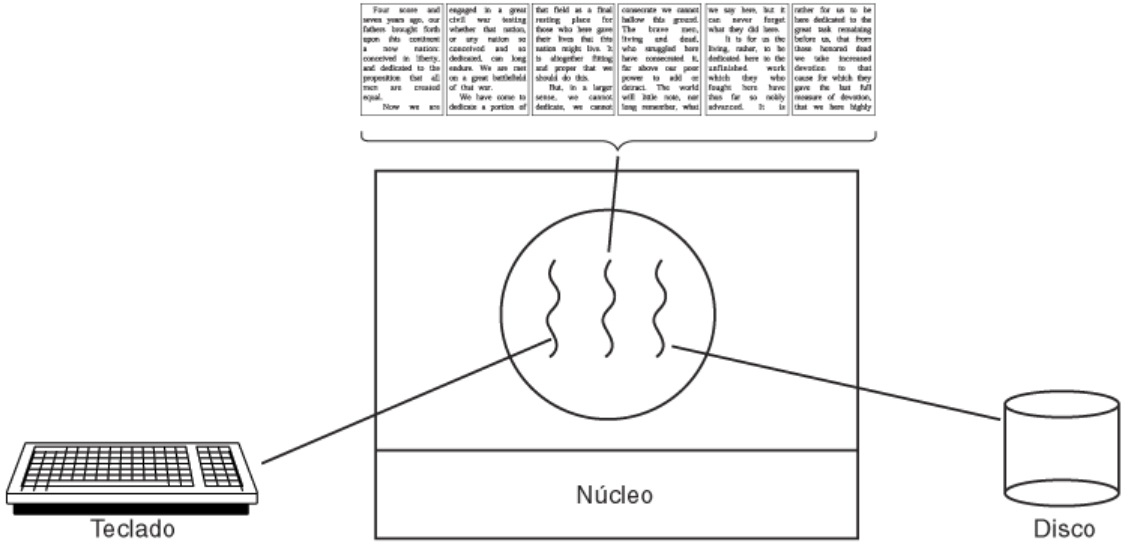


O Modelo de Thread



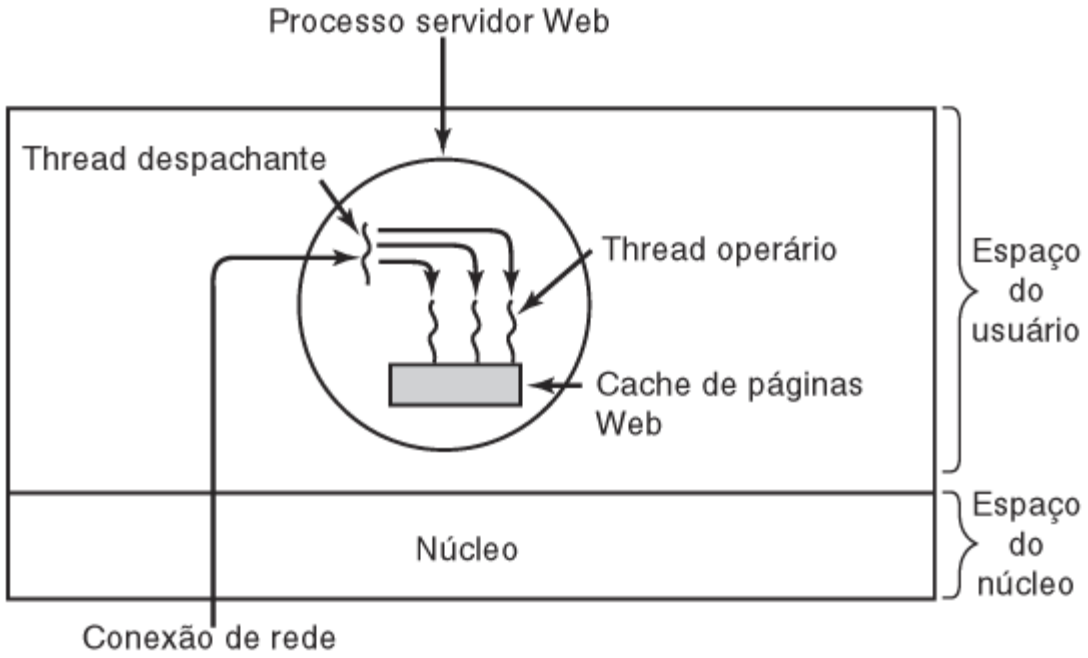
Cada thread tem sua própria pilha

Uso de Thread (1)



Um processador de texto com três threads

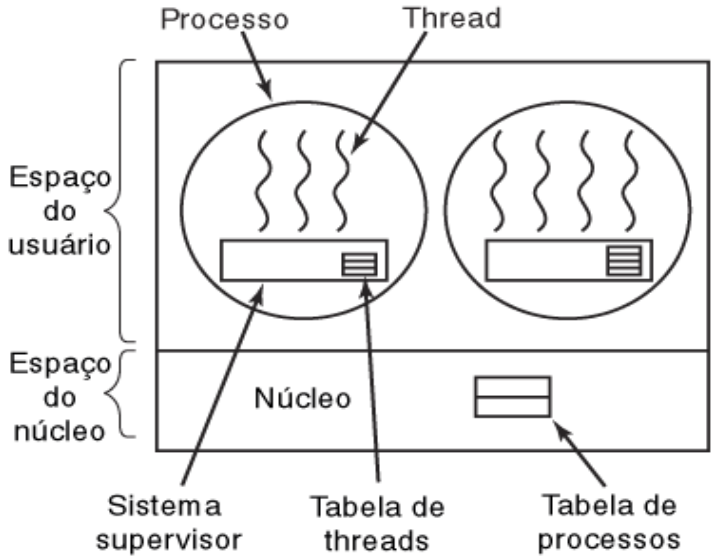
Uso de Thread (2)



Um servidor web com múltiplos threads

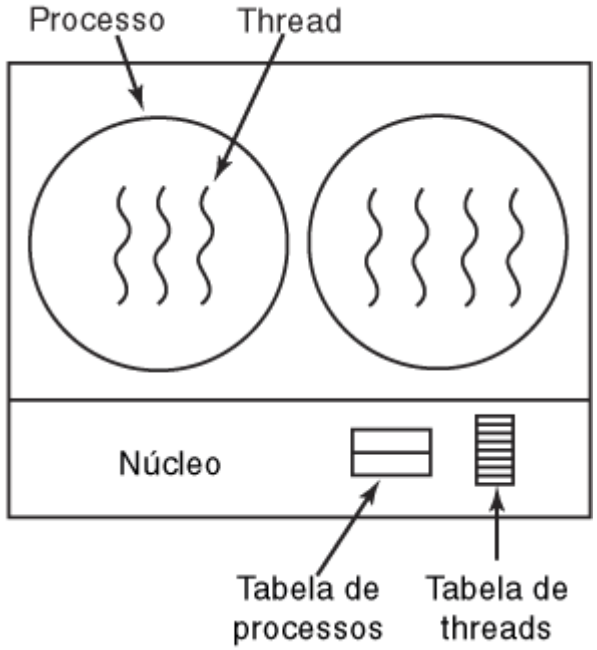
vs um serviço Web com múltiplos servidores (mais adiante – módulo II)

Implementação de Threads de Usuário



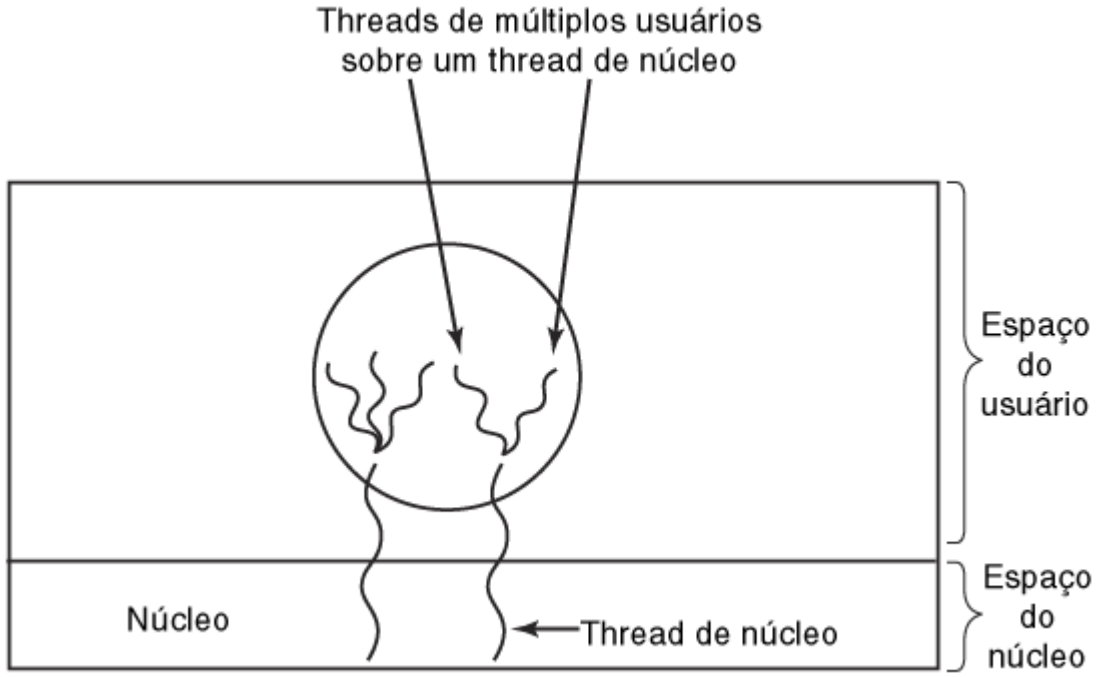
Um pacote de threads de usuário

Implementação de Threads de Núcleo



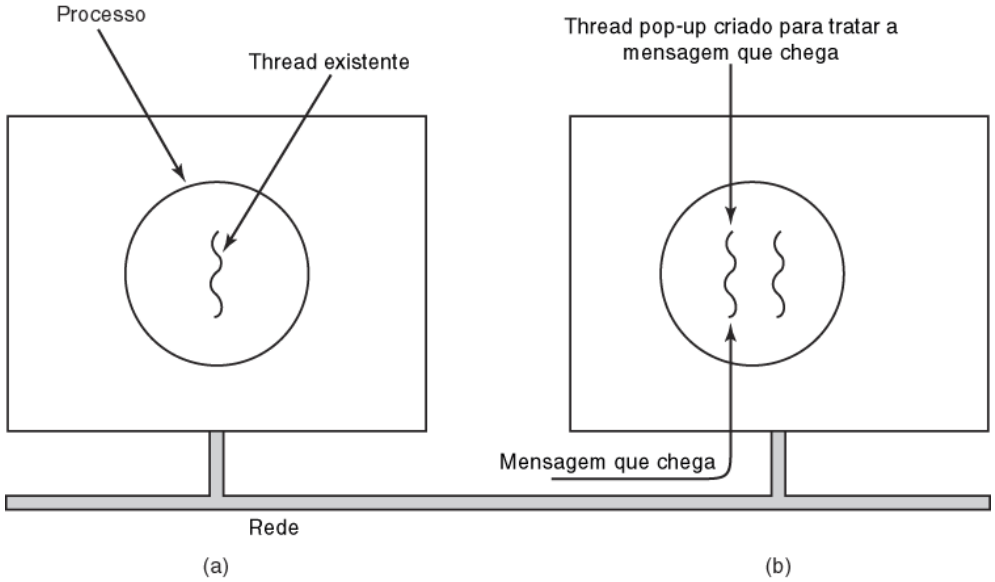
Um pacote de threads gerenciado pelo núcleo

Implementações Híbridas



Multiplexação de threads de usuário sobre threads de núcleo

Criação de um novo thread quando chega uma mensagem



Processos X Threads

- Tipos de sistemas
  - 1 processo X 1 thread: MSDOS
  - N processos X 1 thread: OS/386, VAX/VMS, Windows 3.1, UNIX antigo
  - 1 processo X N threads: kernels para sist. embarcados.
  - N processos X N threads: Windows e UNIX (Linux)

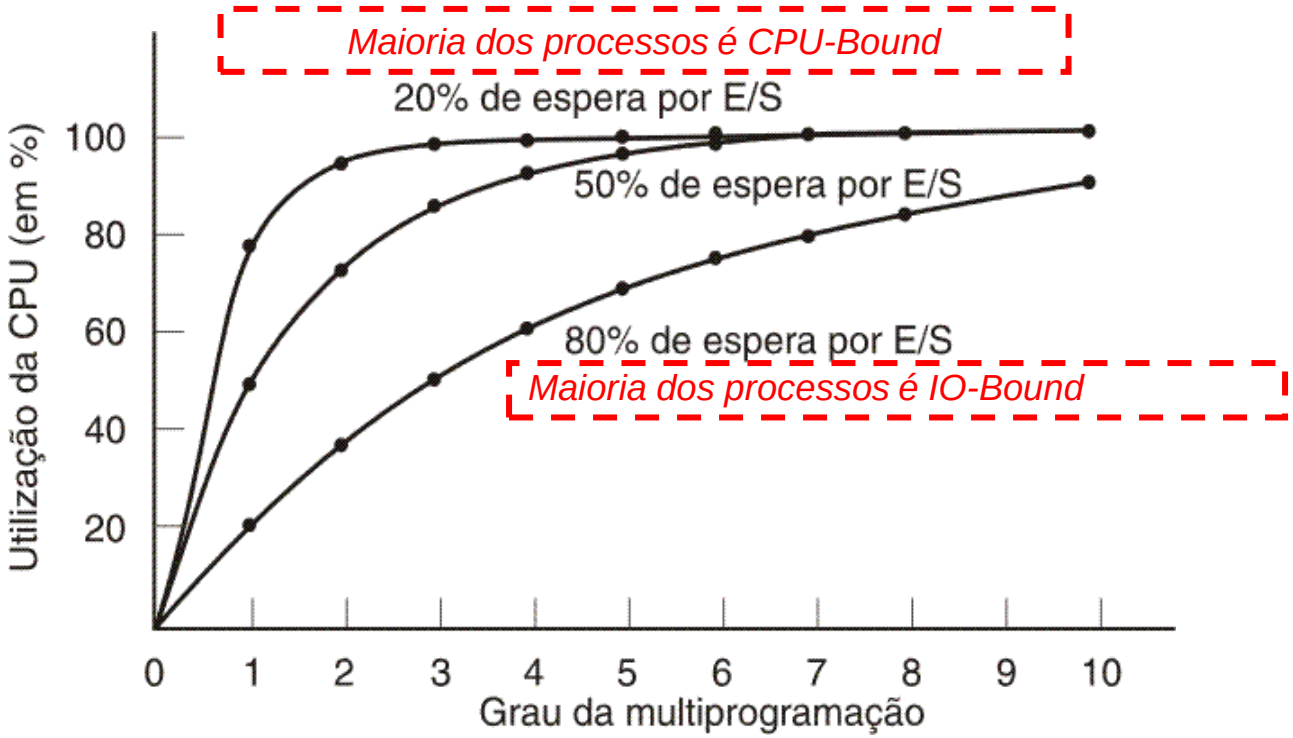
Conceitos Básicos: Tipos de S.O.

O que é necessário para havermultiprocessamento?

- Suporte do Hardware
  - Temporizadores (timers)
  - Interrupções
  - Gerenciamento de memória
  - Proteção de memória

- Suporte do S.O.
  - Escalonamento dos processos
  - Alocação de memória
  - Gerenciamento dos periféricos

Tipos de Processos Vs. Utilização da CPU



Utilização da CPU como uma função do número de processos na memória

Conceitos Básicos:

A importância da Interrupção

- Para obter entrada e saída de dados, **não é interessante** que a CPU tenha que ficar continuamente monitorando o status de dispositivos como discos ou teclados

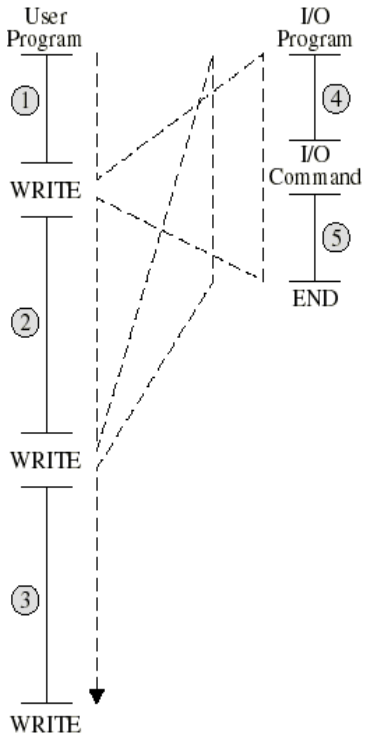
- O mecanismo de interrupções permite que o hardware "chame a atenção" da CPU quando há algo a ser feito

Conceitos Básicos:

A importância da Interrupção

- Numa sistema simples, CPU deve esperar a execução do comando de E/S
  - A cada chamada do comando *write* a CPU fica esperando o dispositivo executar o comando.

Ex: escrita em disco

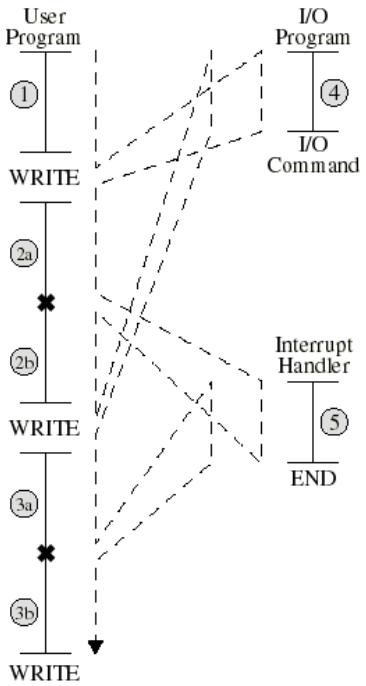


Conceitos Básicos:

A importância da Interrupção

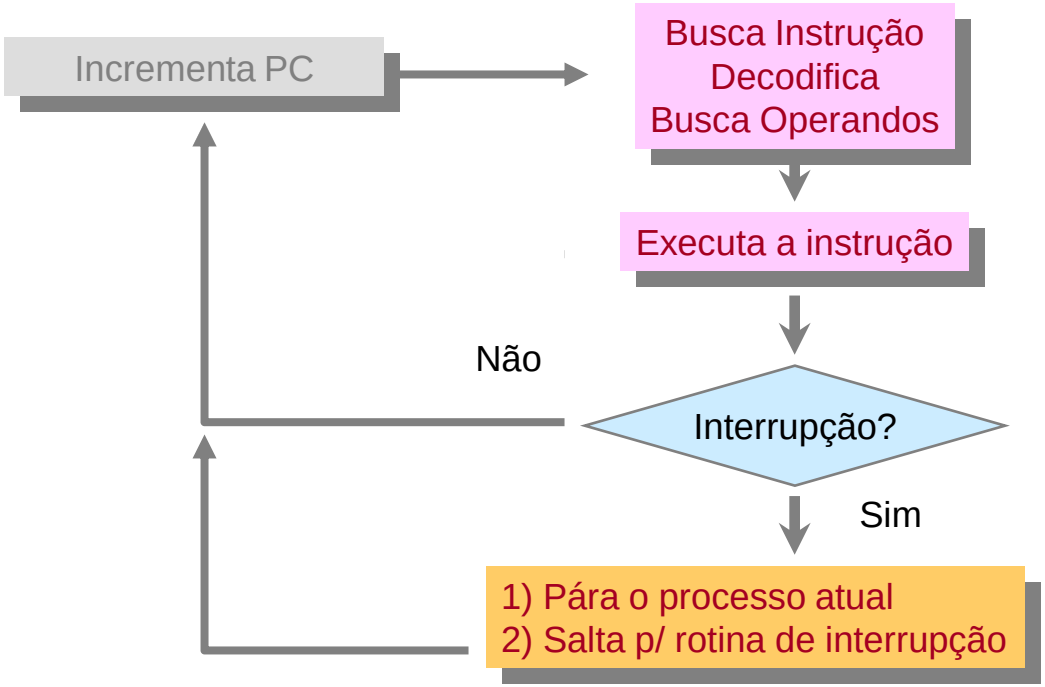
- Um sistema com interrupção não fica esperando
  - A CPU solicita o *write* e fica exectando outras tarefas até ser interrompida pelo disco.

Ex: escrita em disco



Conceitos Básicos

Operação Básica da CPU



Interrupção de Relógio

(Um tipo de Interrupção de HW)

- O sistema operacional atribui *quotas de tempos de execução* (*quantum* ou *time slice* – fatias de tempo) para cada um dos processos em um sistema com *multiprogramação*

- A cada interrupção do relógio, o tratador verifica se a fatia de tempo do processo em execução já se esgotou e, se for esse o caso, suspende-o e aciona o *escalonador* para que esse escolha outro processo para colocar em execução

Interrupções Síncronas ou *Traps*

- *Traps* ocorrem em consequência da instrução sendo executada [no programa em execução]

- Algumas são geradas pelo hardware, para indicar, por exemplo, *overflow* em operações aritméticas ou acesso a regiões de memória não permitidas
  - Essas são situações em que o programa não teria como prosseguir
  - O hardware sinaliza uma interrupção para passar o controle para o tratador da interrupção (no SO), que tipicamente termina a execução do programa

Traps (cont.)

- *Traps* também podem ser geradas, explicitamente, por instruções do programa
  - Essa é uma forma do programa acionar o sistema operacional, por exemplo, para requisitar um serviço de entrada ou saída
    - Um exemplo de programa é a própria interface de usuário (shell ou GUI) do SO
  - Um programa não pode chamar diretamente uma rotina do sistema operacional, já que o SO é um processo a parte, com seu próprio espaço de endereçamento...
    - Através do mecanismo de interrupção de software, um processo qualquer pode ativar um tratador que pode "encaminhar" uma chamada ao sistema operacional

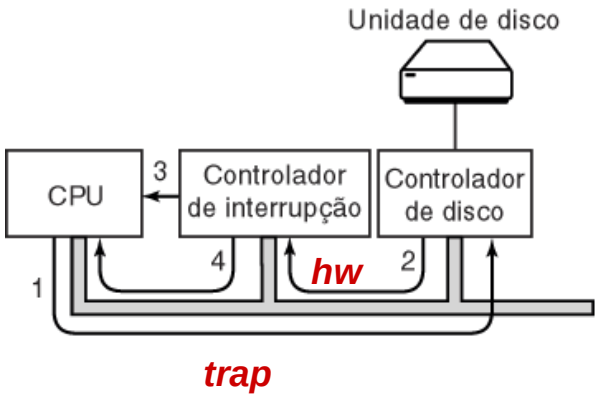
Chamadas ao Sistema

Unix	Win32	Descrição
fork	CreateProcess	Crie um novo processo
waitpid	WaitForSingleObject	Pode esperar um processo sair
execve	(none)	CrieProcesso = fork + execve
exit	ExitProcess	Termine a execução
open	CreateFile	Crie um arquivo ou abra um arquivo existente
close	CloseHandle	Feche um arquivo
read	ReadFile	Leia dados de um arquivo
write	WriteFile	Escreva dados para um arquivo
lseek	SetFilePointer	Mova o ponteiro de posição do arquivo
stat	GetFileAttributesEx	Obtenha os atributos do arquivo
mkdir	CreateDirectory	Crie um novo diretório
rmdir	RemoveDirectory	Remova um diretório vazio
link	(none)	Win32 não suporta ligações (link)
unlink	DeleteFile	Destrua um arquivo existente
mount	(none)	Win32 não suporta mount
umount	(none)	Win32 não suporta mount
chdir	SetCurrentDirectory	Altere o diretório de trabalho atual
chmod	(none)	Win32 não suporta segurança (embora NT suporte)
kill	(none)	Win32 não suporta sinais
time	GetLocalTime	Obtenha o horário atual

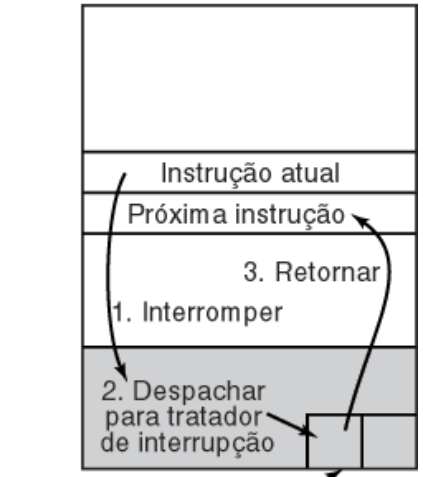
Algumas chamadas da interface API Windows

Conceitos Básicos

Interrupção do Programa



(a)



(b)

Interrupções

Assíncronas (hardware)

- geradas por algum dispositivo externo à CPU
- ocorrem independentemente das instruções que a CPU está executando
- não há qualquer comunicação entre o programa interrompido e o tratador
- Exemplos:
  - interrupção de relógio, quando um processo esgotou a sua fatia de tempo (*time slice*) no uso compartilhado do processador
  - teclado, para uma operação de E/S (neste caso, de Entrada)

Síncronas (*traps*)

- Geradas pelo programa em execução, em consequência da instrução sendo executada
- Algumas são geradas pelo hardware em situações em que o programa não teria como prosseguir
- Como as interrupções síncronas ocorrem em função da instrução que está sendo executada, nesse caso o programa passa algum parâmetro para o tratador
- Exs.: READ, *overflow* em operações aritméticas ou acesso a regiões de memória não permitidas

Interrupção: Passo-a-passo

1. O hardware empilha o contador de programa etc.
2. O hardware carrega o novo contador de programa a partir do vetor de interrupção.
3. O procedimento em linguagem de montagem salva os registradores.
4. O procedimento em linguagem de montagem configura uma nova pilha.
5. O serviço de interrupção em C executa (em geral lê e armazena temporariamente a entrada).
6. O escalonador decide qual processo é o próximo a executar.
7. O procedimento em C retorna para o código em linguagem de montagem.
8. O procedimento em linguagem de montagem inicia o novo processo atual.

Esqueleto do que o nível mais baixo do SO faz quando ocorre uma interrupção

Proteção de Recursos

Proteção de recursos (dados etc.)

- Segurança e privacidade
- Necessidade de comunicação
- Métodos
  - Prevenção (Ling. alto-nível)
  - Detecção e Resolução (Hardware: exceções)

Proteção de Recursos via Hardware

- Análise de operações ilegais
- Proteção via opcode
  - Instruções privilegiadas (E/S ...)
  - Detecção do nível de operação
    - supervisor
    - usuário
- Proteção via operando
  - Privilégios de acesso
  - Análise de domínio
    - área de memória
    - objeto definido pelo usuário

Proteção de Área de Memória

- Domínio único
  - Cada processo possui espaço de memória definido
- Domínio duplo
  - Parte pública: bibliotecas (compartilhamento total)
  - Parte privada: programa de usuário
- Múltiplos domínios
  - Conjunto compartilhado
    - Ex: Sa = {a1,a2,a3}, Sb = {a3,b4,b5}
    - Como implementar ?

Sincronização de Processos

- Execução paralela de processos
  - Real (multiprocessador) ou virtual (time sharing)
  - Problemas de sincronização

```
Process observer
  repeat
    observe um evento
    cont = cont + 1
  until false
```

```
Process reporter
  repeat
    if ((cont mod 10) = 0)
      print (cont)
  until false
```

Situação: reporter testou cont = 10
observer cont = 11
reporter imprimiu cont = 11

- velocidade imprevisível dos processos
- compartilhamento de dados

condição de corrida

controle no acesso às variáveis compartilhadas

Problema da Exclusão Mútua

- Garantir que apenas um processo acesse um dado compartilhado por vez:
  - definição de seções críticas
- Problemas:
  - deadlock
  - starvation (oposto de fairness)

Resolvendo a exclusão mútua

- Busy-waiting
  - Controle de acesso a região
    - variável Lock

Lock = 0 pode entrar
lock = 1 bloqueado

```
loop: test lock
      bne loop
      add #1, loop
```

e se houver Interrupção?

- Desabilitar interrupções

  Simplicidade

  Regiões devem ser pequenas

  Inviável em multiprocessadores

Suporte da arquitetura

- Operações atômicas: Ex. Test-and-set (tas)
  - tas <dst>

```
if dst = 0
then dst = 1
```

Indivisível

```
wait: tas lock
      bmi wait
      critical section
      clr.b lock
```

- Semáforos
  P e V
- Monitores

  Programa que engloba e controla o recurso compartilhado e que tem acesso restrito

Conceitos

- Processo: um programa em execução
- Página: parte de um programa capaz de caber na memória
- Memória virtual: espaço de armazenamento de páginas em disco
- Espaço de endereçamento e proteção
- Escalonamento: quando um ou mais processos estão prontos para serem executados, o sistema operacional deve decidir qual deles vai ser executado

- Interrupção
  - Por hardware
    - Algum dispositivo externo à CPU (ex. teclado)
    - Relógio (para suspender um processo)
  - Por software (trap)
    - Execução de intrução de programa (ex. READ)
    - situações em que o programa não teria como prosseguir (ex. *overflow* em operações aritméticas)
- Chamadas ao sistema formam a interface entre o SO e os programas de usuário