SO: Introdução e Estrutura

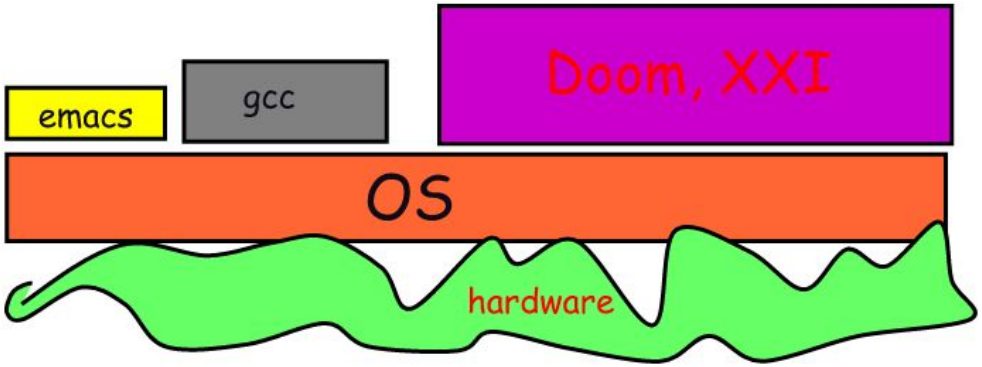
Sistemas Operacionais
2017-1
Flavio Figueiredo (http://flaviovdf.github.io)

O que é um Sistema Operacional?

Simplificando

- Uma interface entre o usuário e o hardware

Detalhando



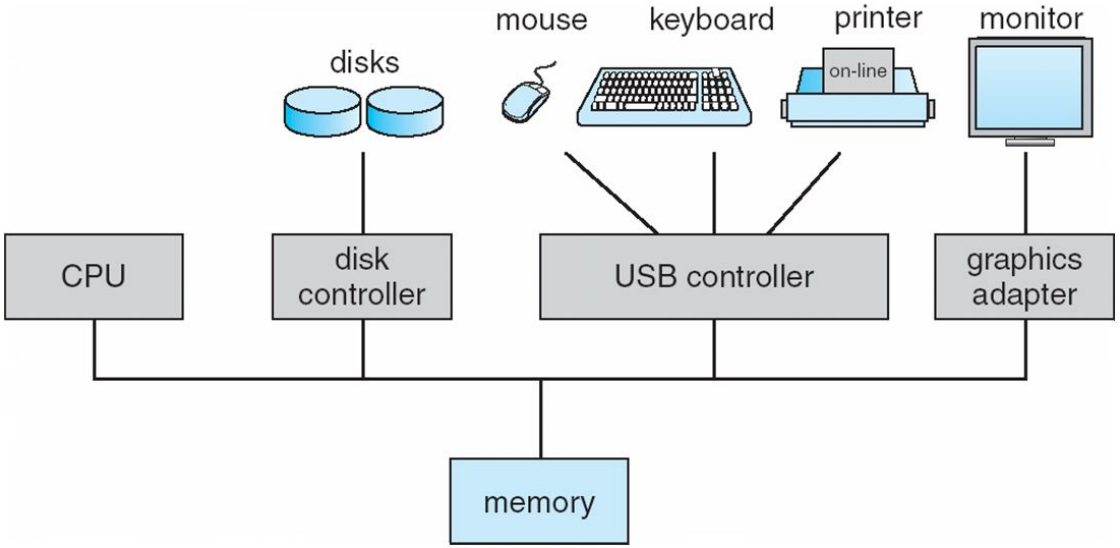
- Camada entre hardware e aplicativos
- Abstração do hardware
  - Esconde detalhes desnecessários (e.g., Disco SSD vs HDs)
  - Toma conta de “acessar o hardware” pelo usuário
- Outras funcionalidades
  - Tarefas de isolamento, segurança etc
- O SO é burocrático por você

Exemplos de Funcionalidade

- Facilitar a implementação de aplicações
  - Dar a impressão que cada aplicação tem memória ilimitada
- Execução de várias aplicações
  - Potencialmente de usuários diferentes
- Isolamento entre as aplicações e controle do hardware
  - Minha aplicação não pode atrapalhar a sua
  - Minha aplicação não pode formatar o disco rígido

Arquitetura de Computadores

- Vamos dar um passo para trás antes de entender os conceitos
- Bem alto nível



Uma Pergunta

- Você está fazendo um programa que lê n-bytes de um arquivo no disco.
- Para isto você tem uma biblioteca alto nível com uma função do tipo:
  - *read(file_struct *file, int n_bytes, int loc, byte *store_at)*
  - file é o arquivo
  - n_bytes indica o números de bytes a ser lido
  - loc indica de onde vamos iniciar a leitura (e.g., a partir do 5 byte)
  - store_at algum buffer para guardar o resultado
- Quais passos vocês acham que ocorrem nesta chamada?
  - Pense desde o seu programa, até a leitura no disco em si, até o retorno do resultado
  - O código já está compilado e executando
  - Lembre-se das aulas de arquitetura

Invertendo o Contexto

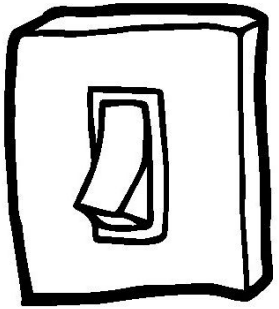
- Se ao invés de um programador lendo um arquivo
- Um usuário acaba de digitar uma tecla ('a') no teclado

- Quais são os passos até a letra 'a' aparecer na tela?
  - Não pense em detalhes de hardware (e.g., se o teclado é com ou sem fio)
  - Abstração dos passos

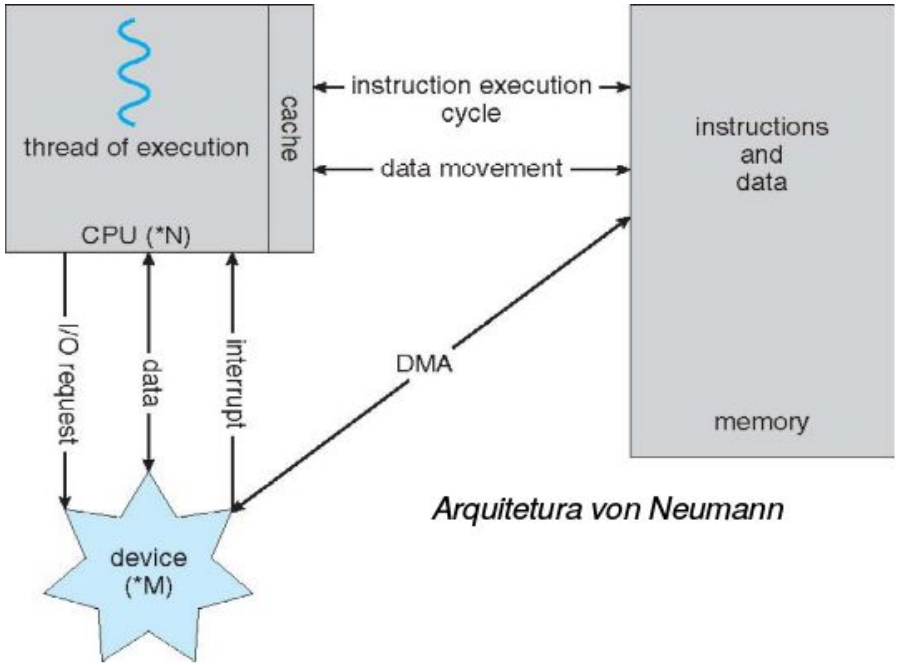
Dois Conceitos Importantes

- Interrupções
  - [Geralmente] Hardware tentando falar com o SO
  - [Geralmente] Ocasionado pelo usuário
    - Mover o mouse
    - Desligar o SO
    - etc
  - [Às Vezes] Não vai afetar o SO

- Chamadas de Sistema
  - Servem para o programador falar com SO
  - Muda o contexto para o espaço de *kernel/núcleo (veremos isto)*
  - [Às Vezes] Vai acabar gerando uma interrupção



Interrupções



Arquitetura von Neumann

S.O.s são movidos a interrrupções!

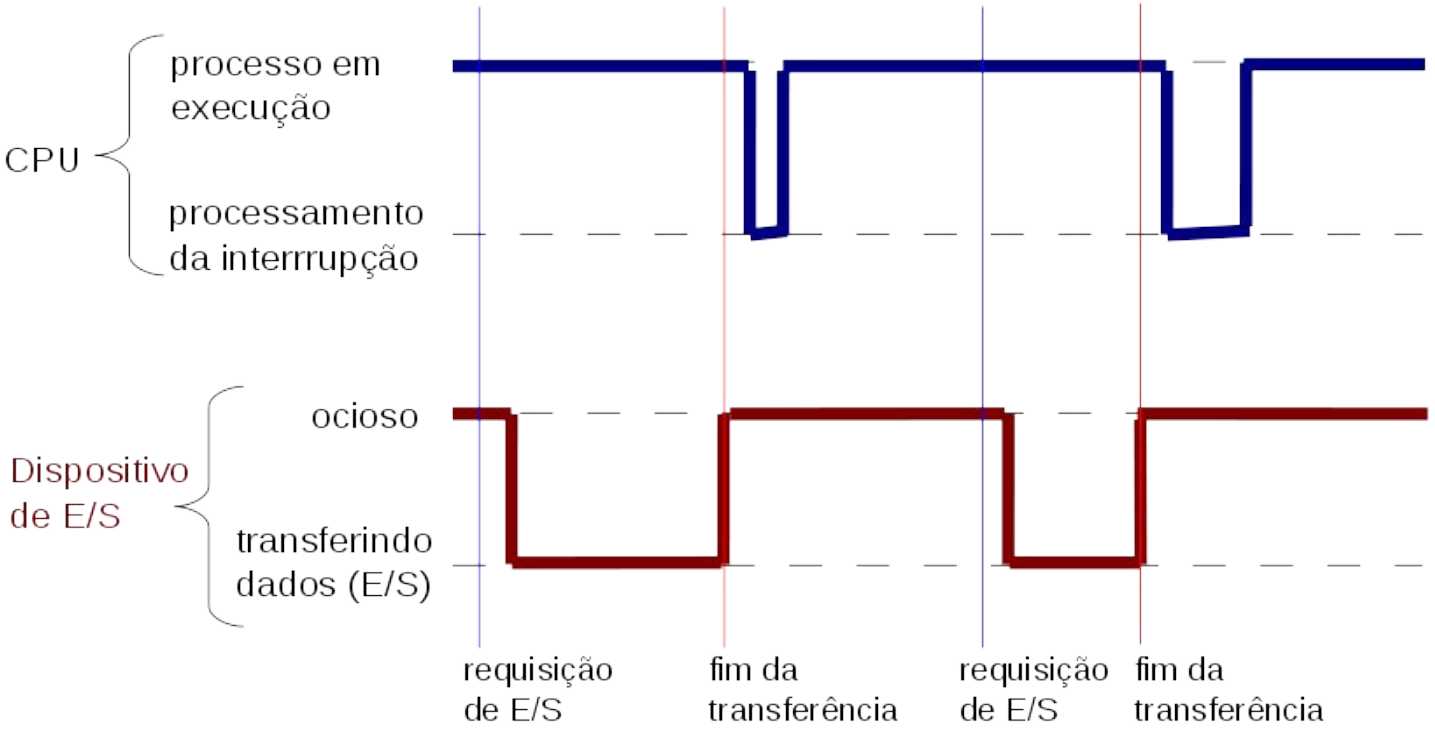
Interrupções

- Comunicação entre o Hardware e o SO
  - Passos intermediários
    - Controladores (e.g., controladora USB)
    - Processador
    - Barramentos
- Podem ser gerados pelo
  - Hardware (e.g., mouse)
  - Software (e.g., um processo através de um *system call*)
- Traps
  - Interrupções usadas para sinalizar que algo deu errado

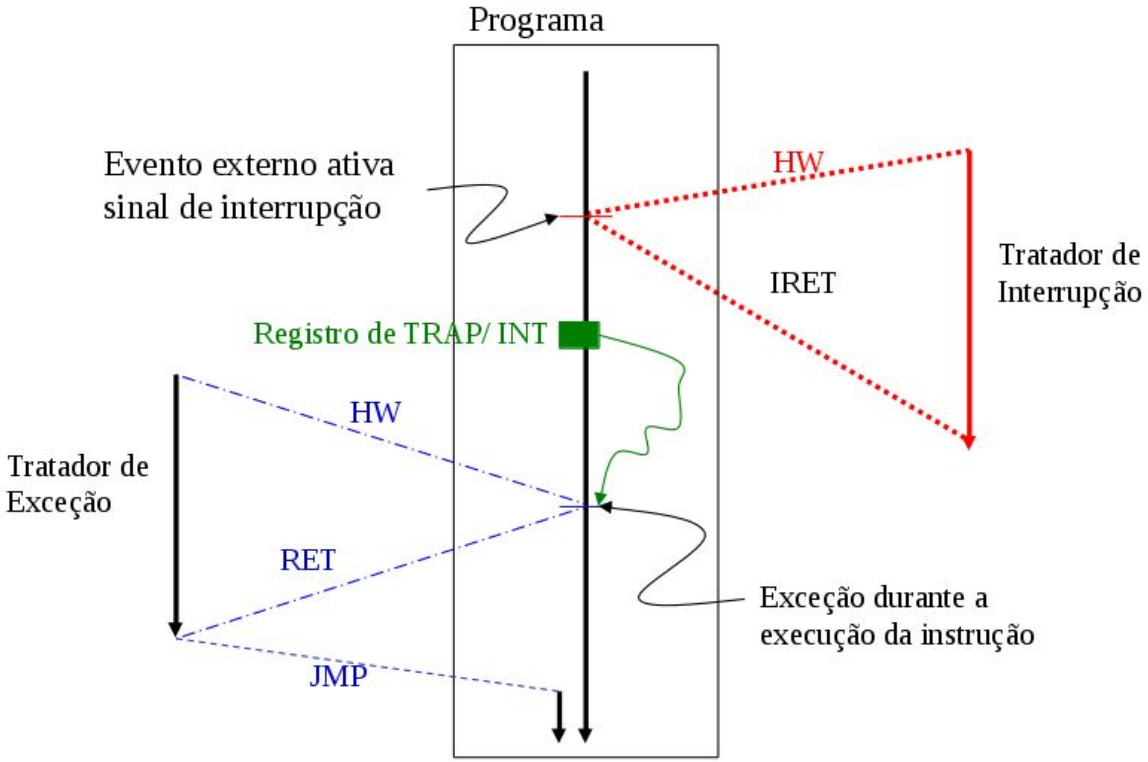
Passo a Passo

1. Um sinal externo em uma linha de interrupção inicia o processamento da mesma no HW
2. A CPU salva o estado de execução no momento da INT
    a. endereço da instrução interrompida
    b. valor dos registradores
    c. status
3. Novas interrupções são temporariamente desabilitadas
4. Diferentes trechos de código definem as ações para cada tipo de interrupção
5. O HW deve determinar o tipo (núm.) da INT
    a. polling – a CPU deve inspecionar o sistema e decidir
    b. vetor – o HW fornece um índice para um vetor de interrupções (inicializado previamente pelo S.O.)
6. Ao retornar a arquitetura restaura o estado da CPU automaticamente

Exemplo

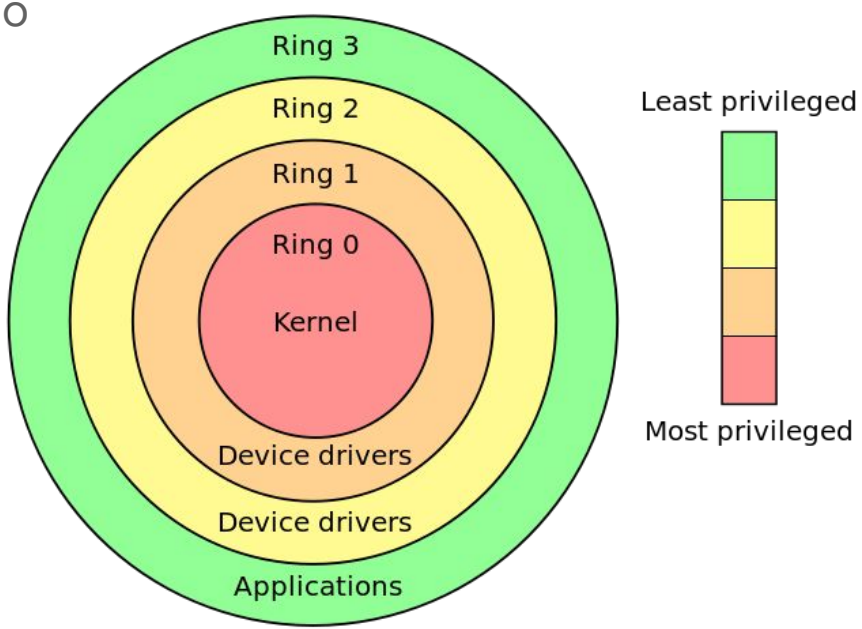

CPU
processo em execução
processamento da interrrupção
Dispositivo de E/S
ocioso
transferindo dados (E/S)
requisição de E/S
fim da transferência
requisição de E/S
fim da transferência

Exemplo


Programa
Evento externo ativa
sinal de interrupção
HW
IRET
Tratador de
Interrupção
Registro de TRAP/ INT
HW
Tratador de
Exceção
RET
Exceção durante a
execução da instrução
JMP

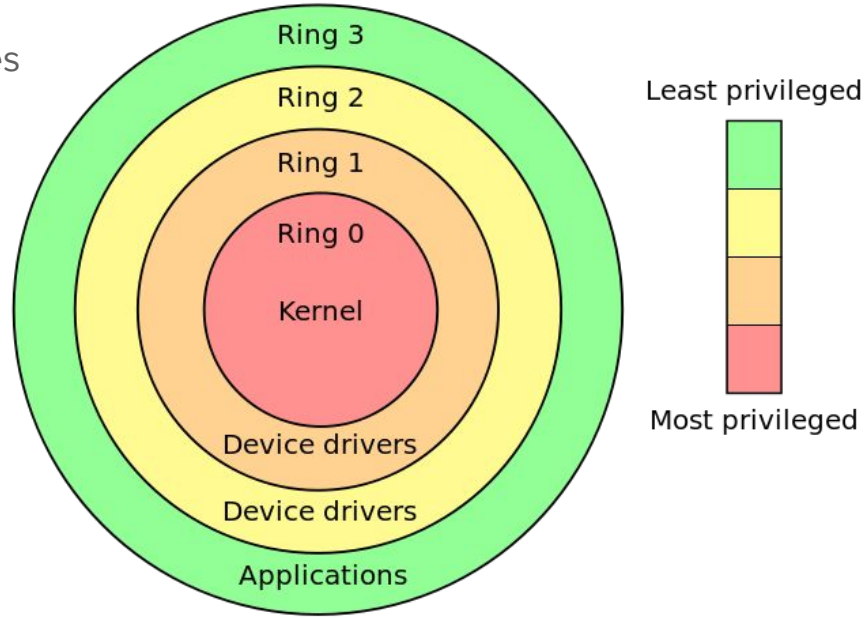
Chamadas de Sistema

- Servem para mudar o nível de operação
  - Como assim nível de operação?
- Modo Usuário (verde)
  - Funções e rotinas que são “seguras”
  - e.g., função do quicksort
  - e.g., somar 2 números
- Kernel Model
  - Tratar com o hardware
  - Apenas o SO pode fazer isto
  - Na verdade são várias camadas

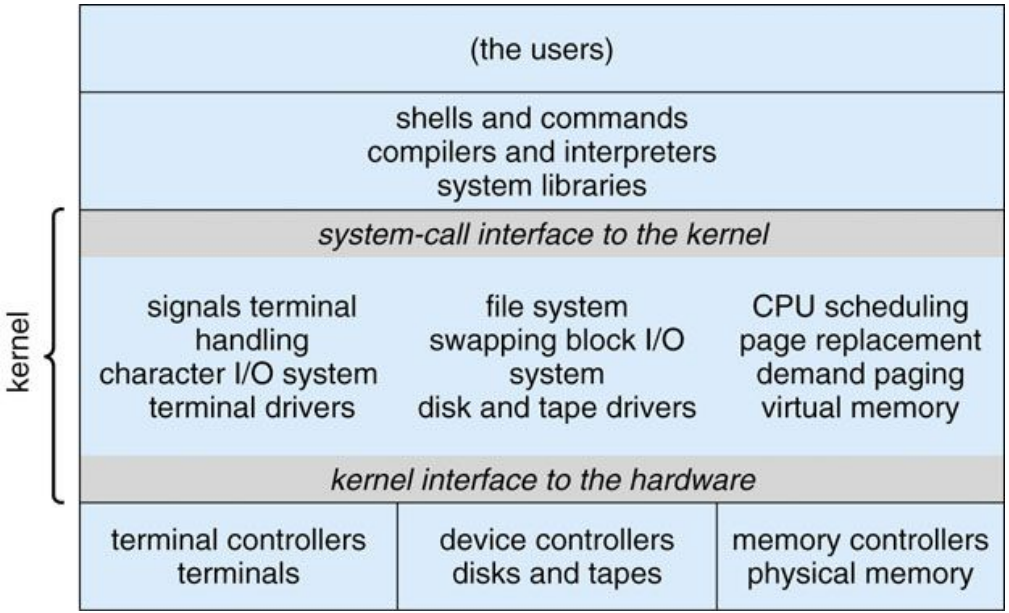


Níveis de Operações

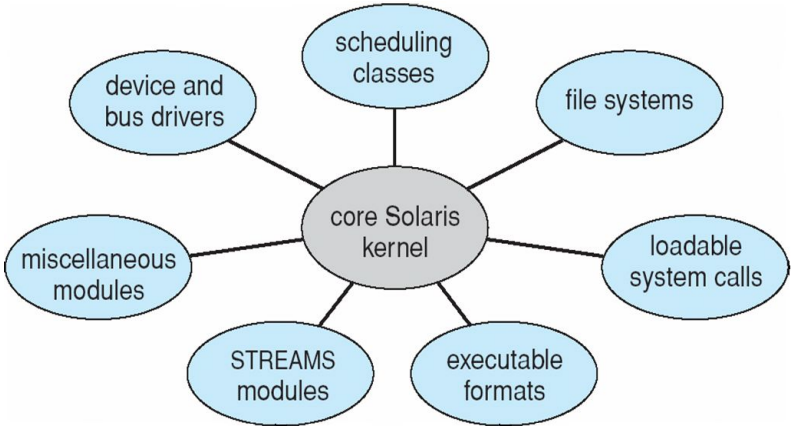
- Definidos pelo Hardware
  - Lembre-se que o hardware define instruções
  - Cada uma tem um nível
- Apenas o SO opera no nível 0
  - Drivers podem usar outros níveis
- Este é o modelo mais adotado hoje
  - Kernel + Drivers (módulos)
  - Existem Outros
    - Microkernel
    - Monolítico
      - Detalhes no Livro



Exemplos de Estrutura

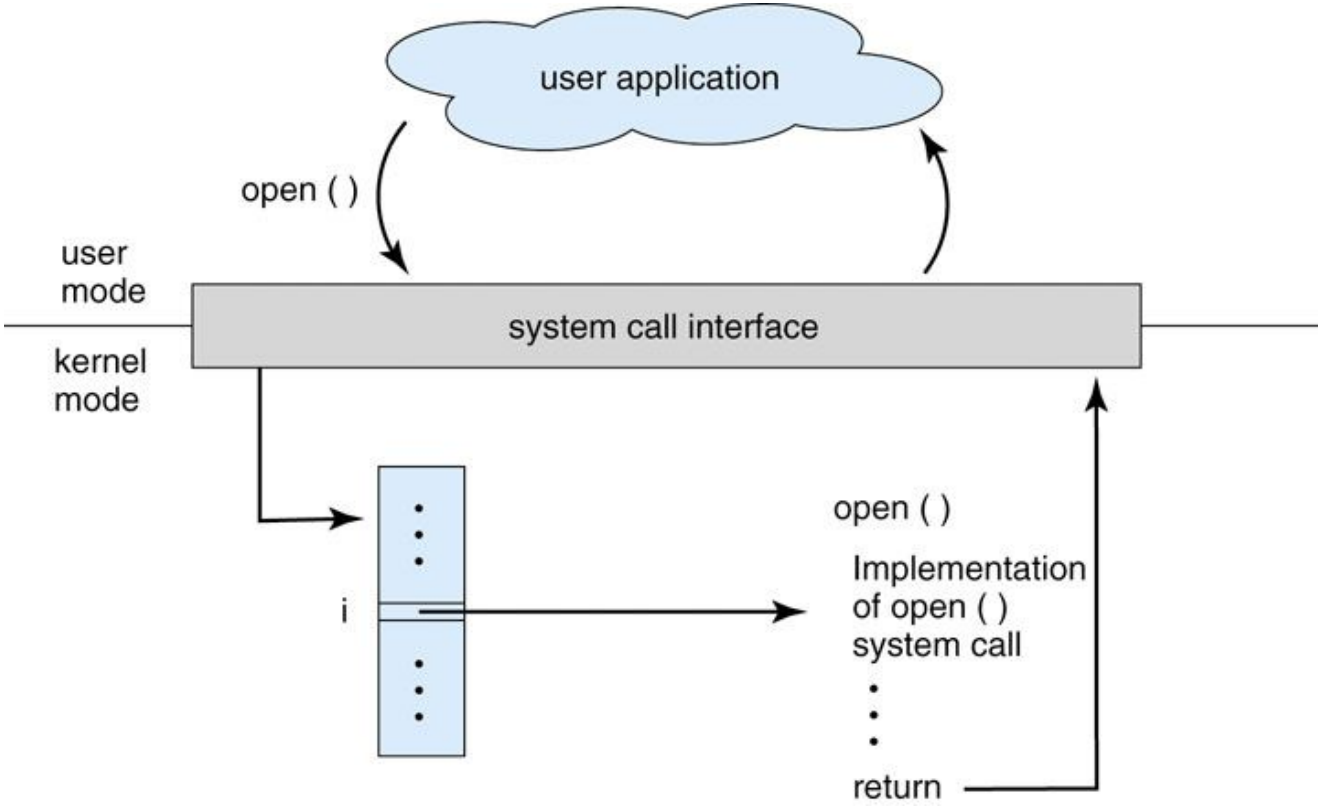


Unix Tradicional:
“Monolítico”

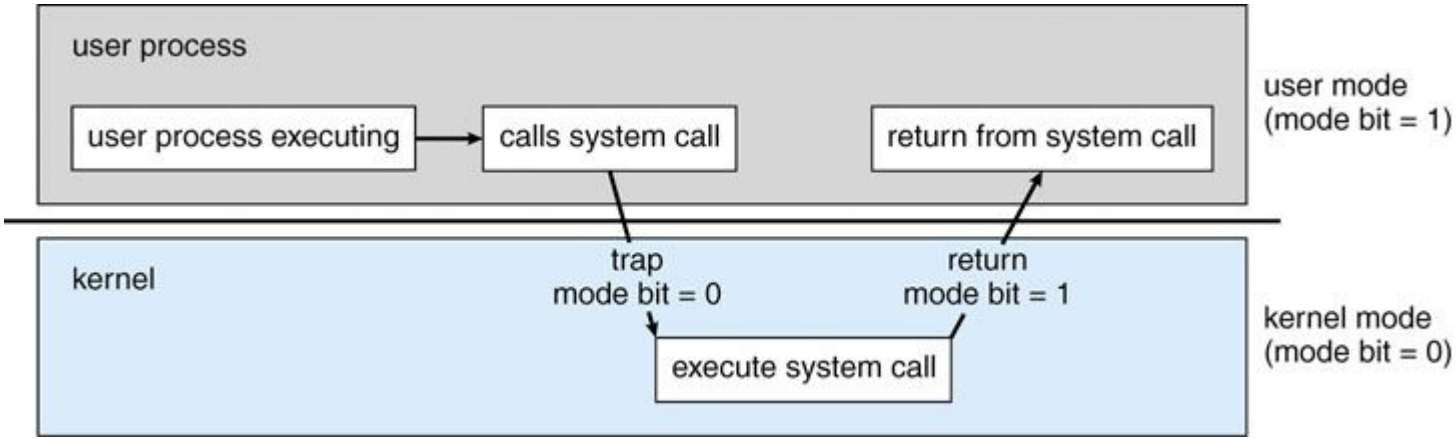


Solaris: Módulos

Como Ocorre a System Call?



Exemplificando



Aonde Estamos

- Não falei de
  - Histórico de SOs
  - Exemplos de SOs
  - Assuntos interessantes que estão nos capítulos iniciais do livro
  - Prefiro ir direto ao assunto em si

- Estamos vendo
  - Boa parte do Capítulo 2
  - Mais exemplos no livro

- Para onde vamos
  - Processos (Capítulo 3)