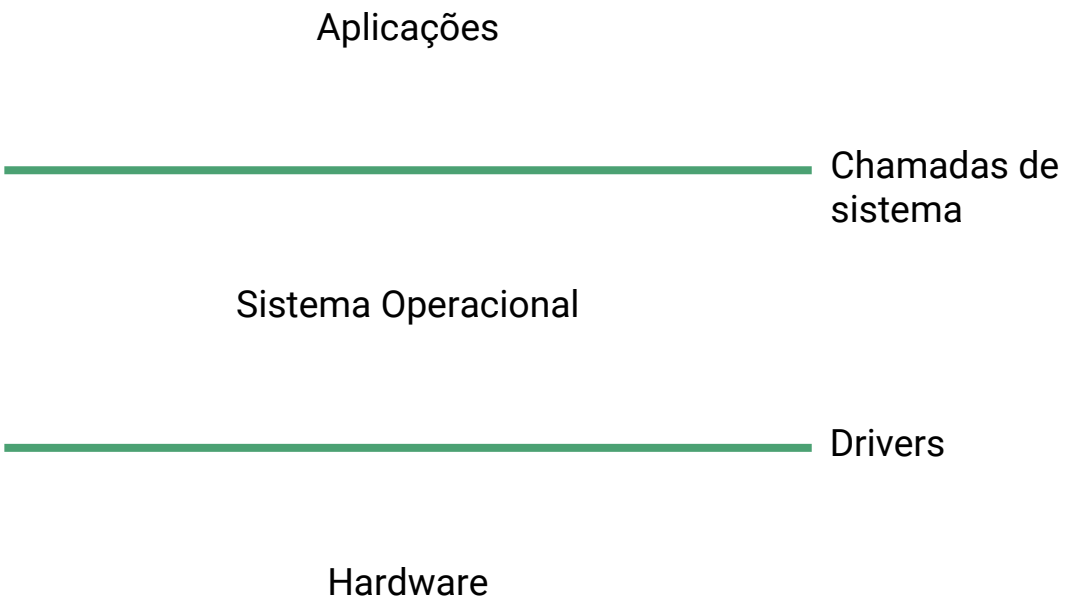
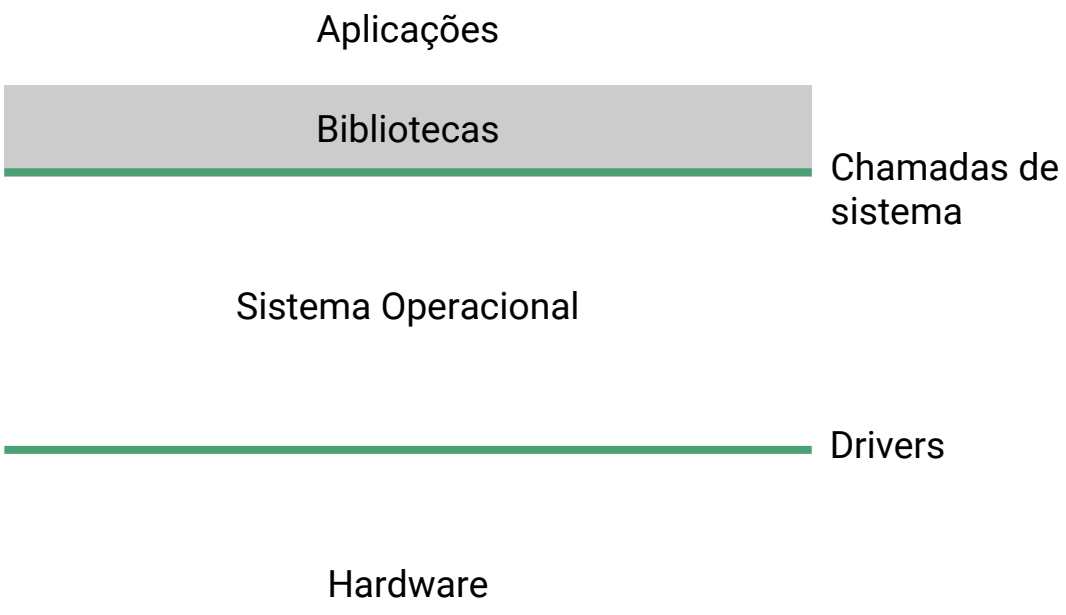
SO: Introdução - Parte 2

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

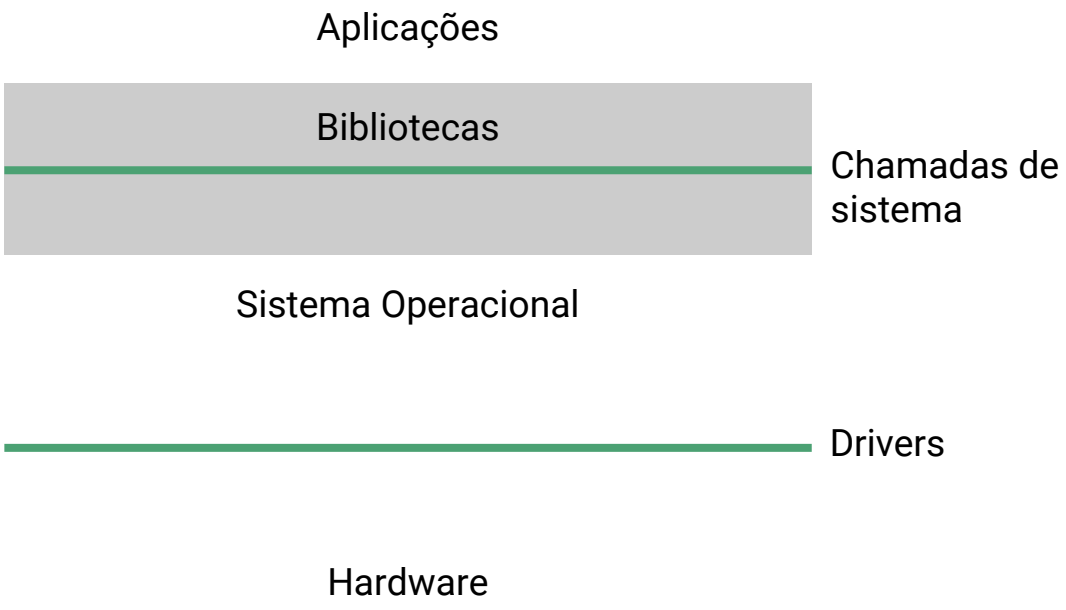
O que faz um sistema operacional?



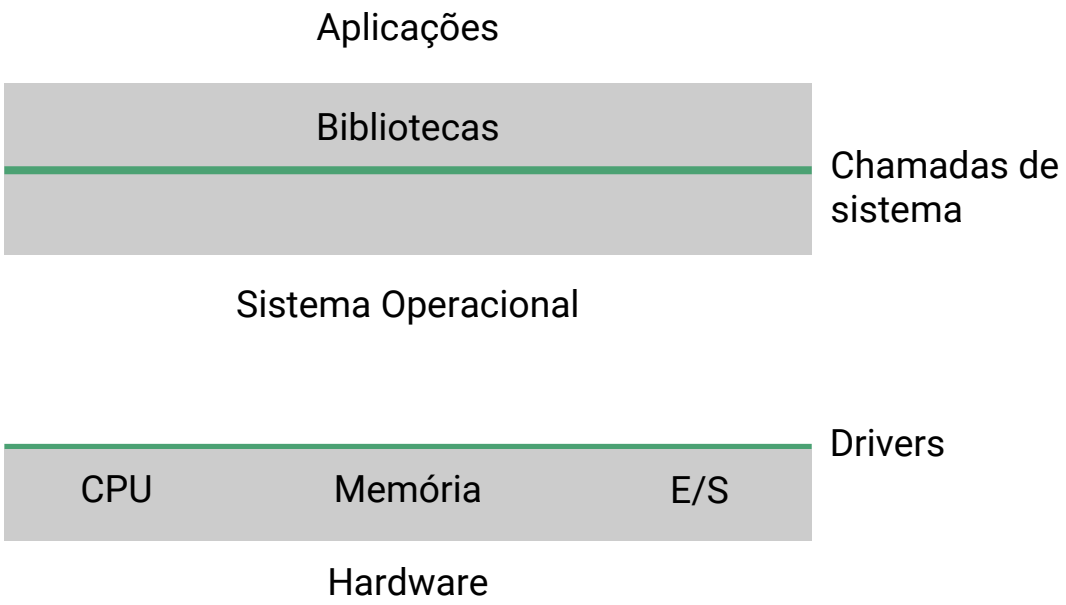
O que faz um sistema operacional?



O que faz um sistema operacional?



O que faz um sistema operacional?



Funcionalidades de um sistema operacional

Facilitar a implementação de aplicações

- Execução de várias aplicações
  - Escalonamento, sincronização, comunicação
- Utilização eficiente e mediação dos recursos
  - Concorrência, sistemas de arquivo
- Virtualização e proteção de memória

Isolamento

- Sistema operacional usa recursos do hardware para isolar processos

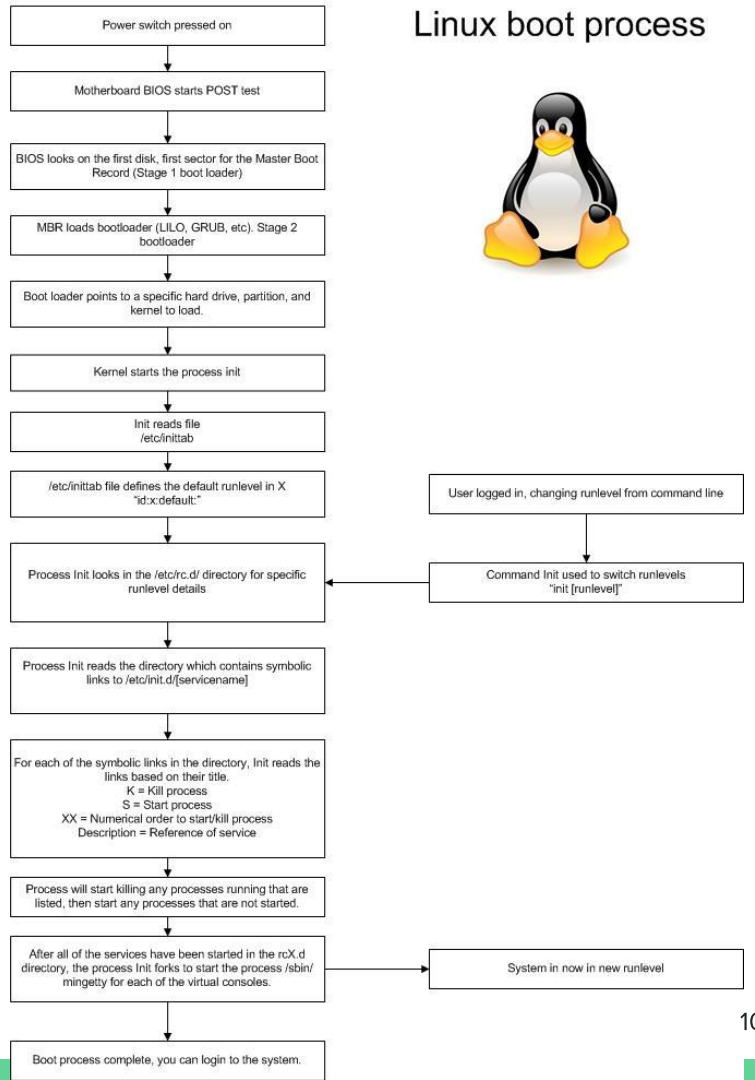
Isolamento

- Sistema operacional usa recursos do hardware para isolar processos
- Programas fazem requisições ao sistema operacional
  - Apenas o sistema operacional interage com o hardware

Quando o sistema operacional executa?

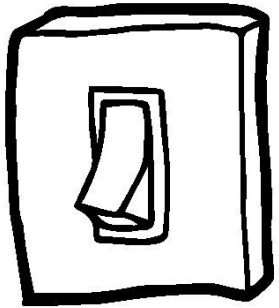
Booting

- Bios é carregada da ROM
  - Nem sempre é "read-only"
  - Pode ser re-programa *EPROM*
- A BIOS indica qual disco vamos dar boot
- Lê-se o Master Boot Record
  - Sempre fica em um bloco fixo do disco
  - Bloco 0 por exemplo
- O programa do Bloco 0 inicia o SO
  - O Grub por exemplo



Relembrando: Interrupções

- Sistemas operacionais são movidos pelas interrupções



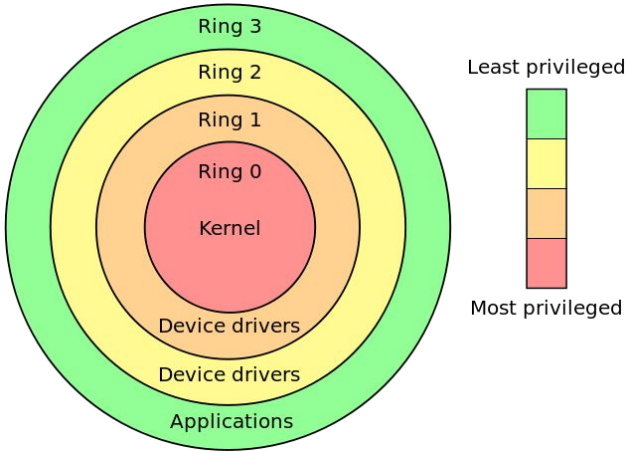
- Sinal entre o hardware e o SO

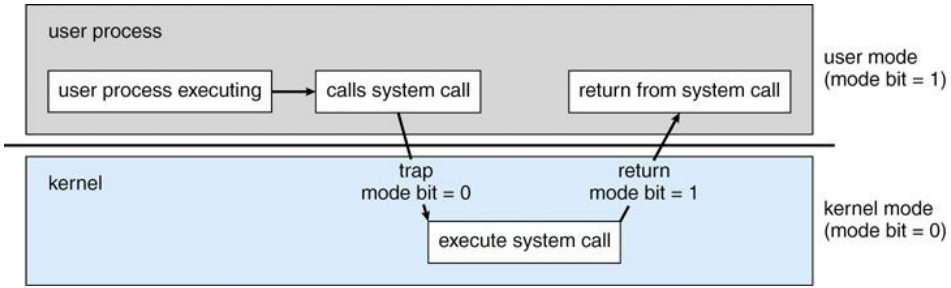
- Imagine o sistema operacional como um tratador de eventos
  - Interrupções são eventos
  - Cada evento tem um tratamento específico

Relembrando: Chamadas de Sistema

- Mudam o código de nível usuário para o nível kernel

- [Geralmente]
  Apenas o SO opera no nível 0
  - Drivers podem usar outros níveis





Tipos de Chamadas de Sistema

- Controle de processos
  - encerrar, abortar
  - criar
  - esperar
- Gerenciamento de arquivos
  - criar
  - apagar
  - abrir
- Gerenciamento de dispositivos
  - solicitar dados
  - liberar
  - bloquear
- Comunicações
  - canais de comunicação entre proc.
  - enviar receber mensagens
  - conectar/desconectar
- Manutenção
  - gerenciar processos
  - hora
  - logging

xv6: um UNIX simplificado

- Recomendado para entender como um SO funciona
- Código simples e fácil de entender
- https://github.com/mit-pdos/xv6-public

xv6: chamadas de sistema

```
fork()
exit()
wait()
kill(pid)
getpid()
sleep(n)
exec(filename, *args)
sbrk(n)
```

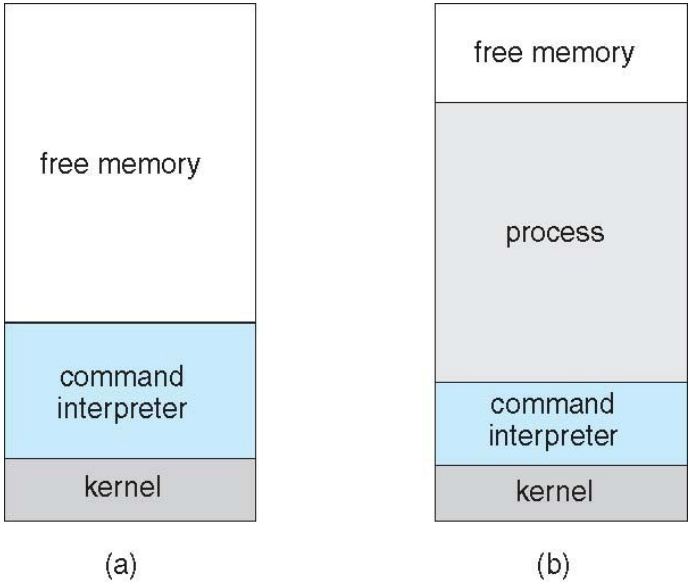
```
open(filename, flags)
read(fd, buf, n)
write(fd, buf, n)
close(fd)
dup(fd)
pipe(p)

chdir(dirname)
mkdir(dirname)

mknod(name, major, minor)
fstat(fd)
link(f1, f2)
unlink(filename)
```

Relembrando: Organização de sistemas operacionais

- Bibliotecas
- Kernel monolítico
- Microkernel
- Kernel modular
- Hypervisors

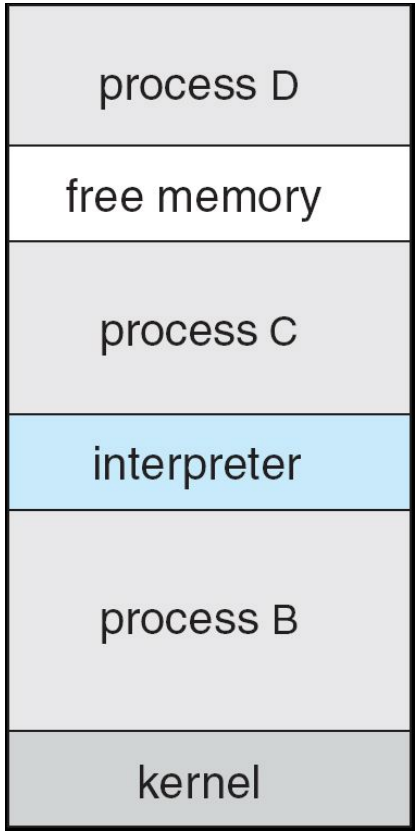
Single Tasking

- MS/DOS
  - Apenas uma tarefa executa por ver
- Cada processo novo pode ocupar a região que estiver livre na memória
- Kernel é protegido

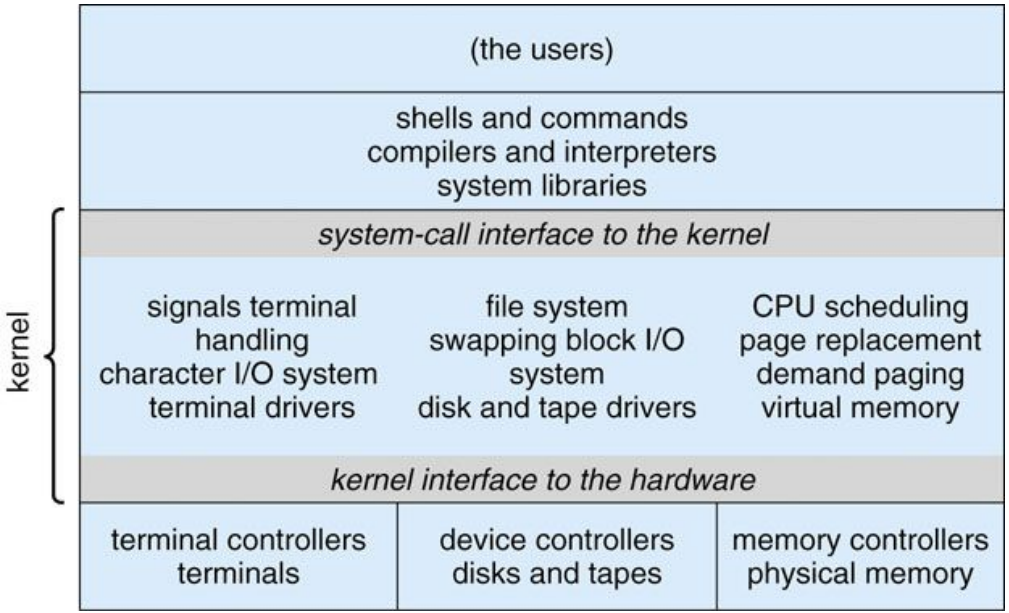


Multi Tasking (multiprocessado)

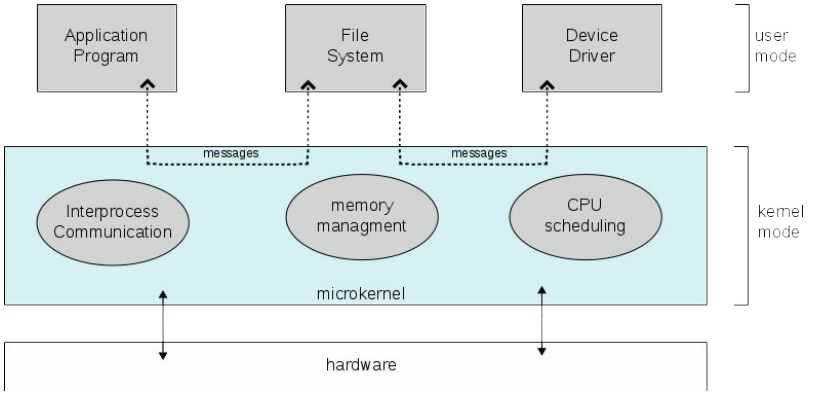
- UNIX Like
- Cada processo ocupa um espaço na memória
- SO isola cada um
- SO chavea a execução entre os processos



Relembrando: Estruturas de SOs



Unix Tradicional:
“Monolítico”



Mach: Microkernel

Um bicho diferente

Sistemas operacionais são diferentes de programas comuns

- Sem acesso à biblioteca padrão
- Escrito em linguagens de médio nível (assembler, C)
- Sem memória protegida
- Difícil acesso a instruções de ponto flutuante
- Pilha de tamanho fixo
- Assíncrono, preemptivo e concorrente
- Portabilidade

Para onde vamos...

- Passamos pelo Capítulo 1 e 2 do Livro
  - Rapidamente
- Capítulo 3:
  - Processos
- Referências
  - Silberschatz, Galvin, Gagne; Operating System Fundamentals, 8th Edition
    - Chapters 1 and 2
  - Tanenbaum; Modern Operating Systems; 5th Edition
    - Chapter 1
  - xv6 book (Sep. 2014 draft)
    - Chapter 0