Sistemas Operativos

Licenciatura Engenharia Informática
Licenciatura Engenharia Computacional

Ano letivo 2024/2025

Nuno Lau (nunolau@ua.pt)

Interrupções/Excepções

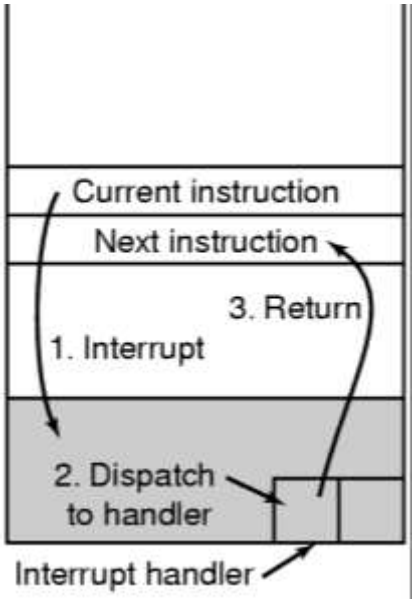
- Considerar um computador com apenas 1 CPU que está a executar o seguinte código:

```
while(1) {
        i++;
}
```

- Como pode o Sistema Operativo obter o controlo do computador?

Interrupções/Excepções

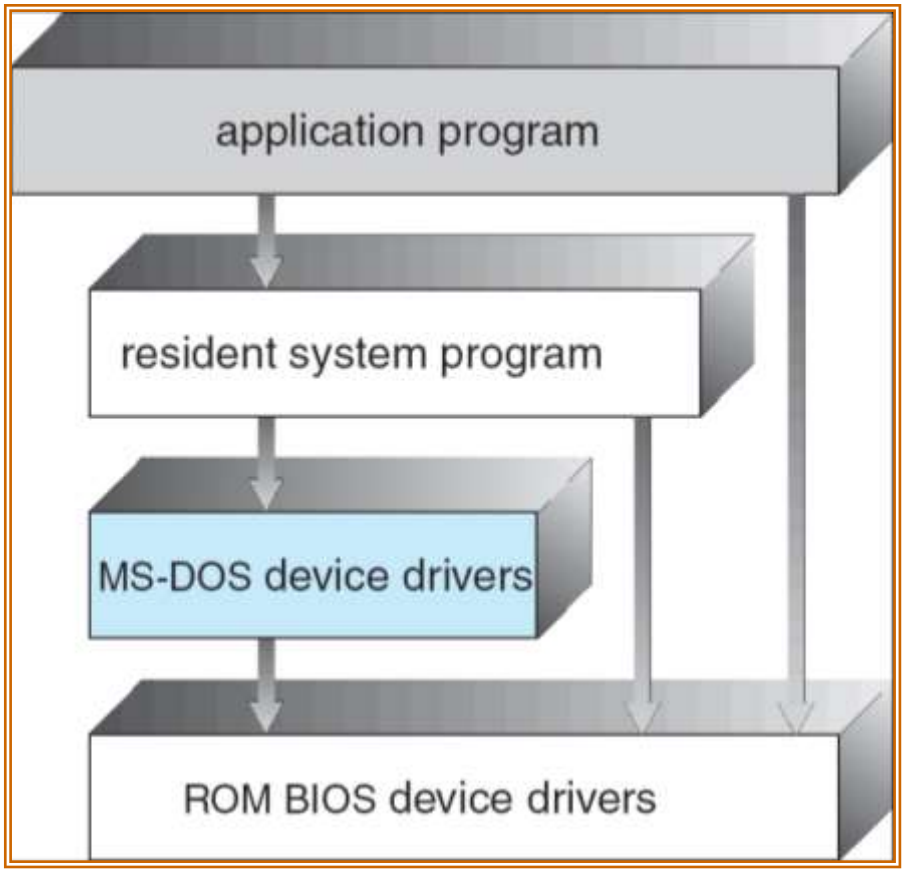
- Quando o CPU deteta uma interrupção **abandona o código** que está a executar e **transfere o controlo** para a **rotina de atendimento da interrupção**
- O endereço da instrução interrompida deve ser salvaguardado
  - Porquê?
- Durante a execução da rotina de atendimento as interrupções estão desativadas
  - Problema da interrupção perdida
- Um *trap* ou exceção é uma interrupção gerada por software
  - *Access violation*, *breakpoint*, *misaligned access*, *divide by 0*, *overflow*, *illegal instruction*, *previleged instruction*
- Nos Sistemas operativos as interrupções são fundamentais



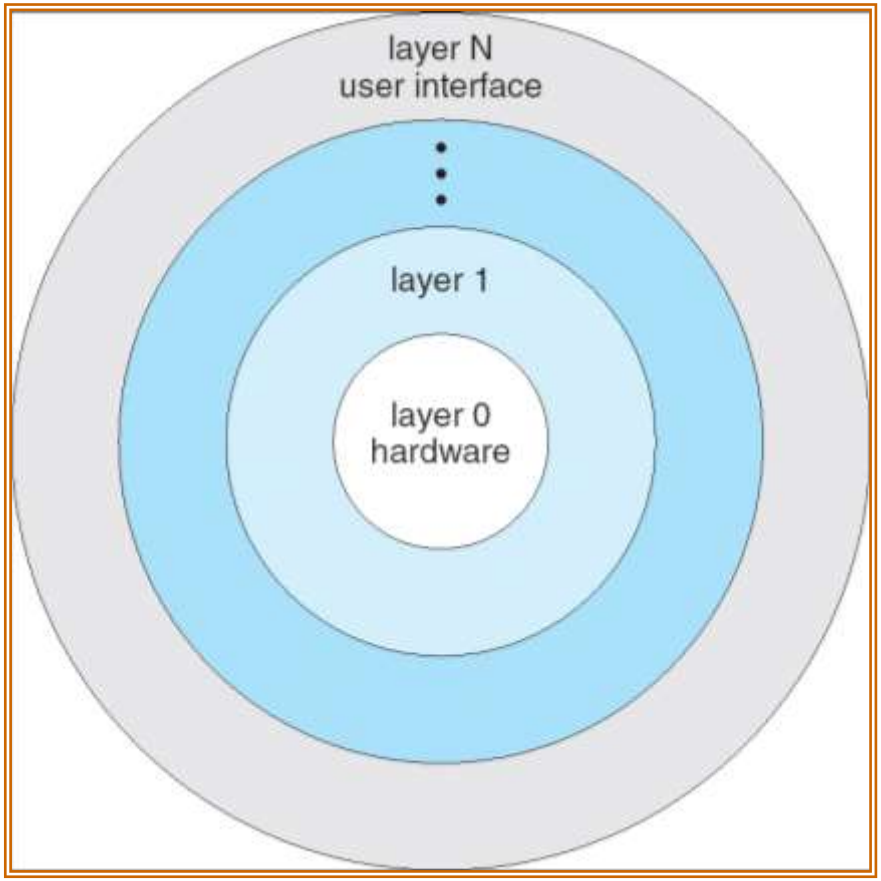
Estrutura do SO

- **Monolítico**
  - Sistema operativo contém todas as funcionalidades de forma estática
  - Permite código otimizado, pouco flexível, ocupa mais memória
  - Ex: MS-DOS; UNIX
- **Modular**
  - Sistema operativo permite adição/configuração de funcionalidades através de integração de módulos
  - Custo/*Overhead* da API, mais flexível, menos memória
  - Ex: Solaris, Linux
- ***Microkernel***
  - *Kernel* apenas com serviços básicos: *thread*, *address space*, *ipc*
  - Várias funcionalidades associadas ao SO correm em modo utilizador
  - Pouca memória, verificável, mudanças de *kernel mode* para *user mode* são frequentes
  - Ex: MACH, MINIX, QNX

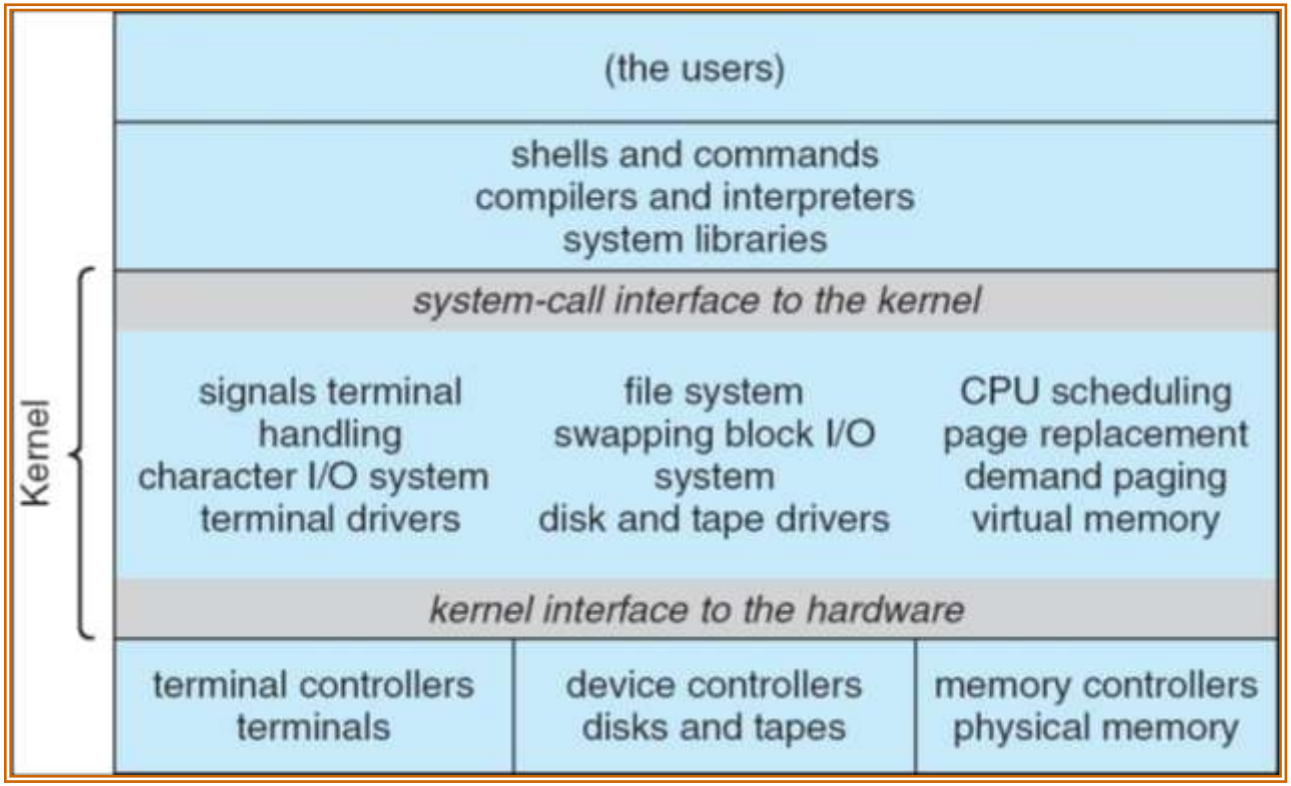
MS-DOS


application program
resident system program
MS-DOS device drivers
ROM BIOS device drivers

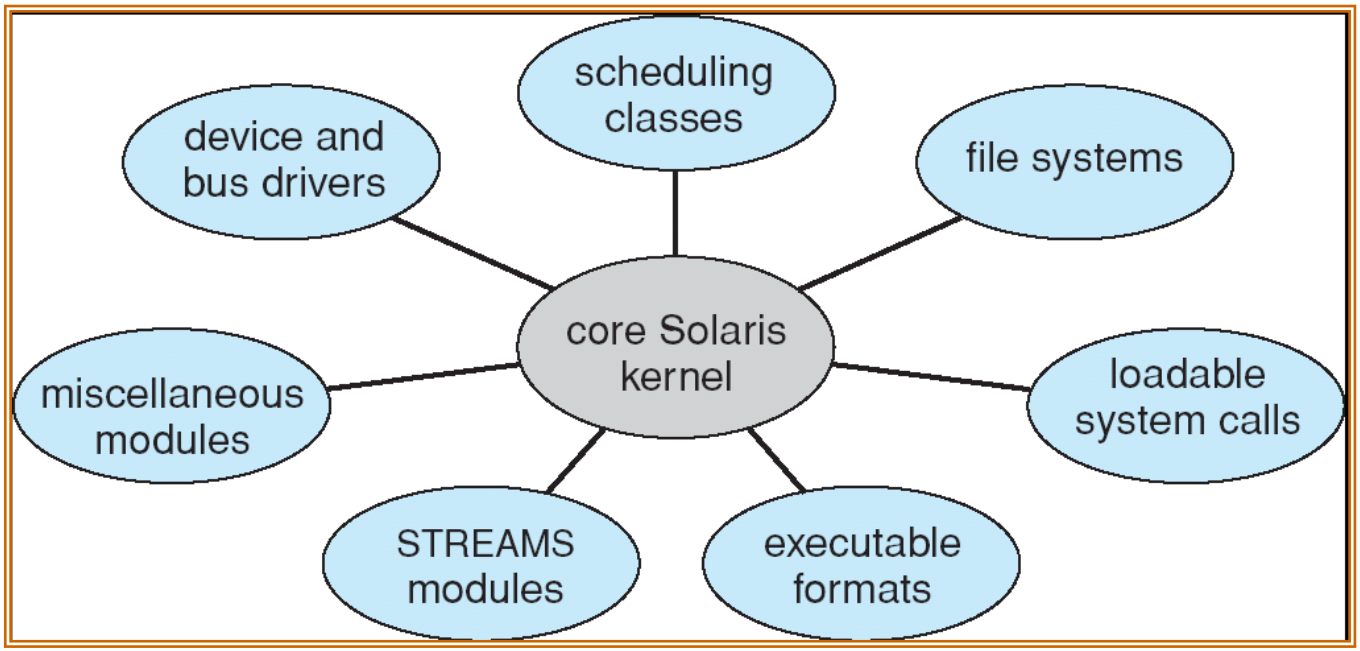
Organização em camadas



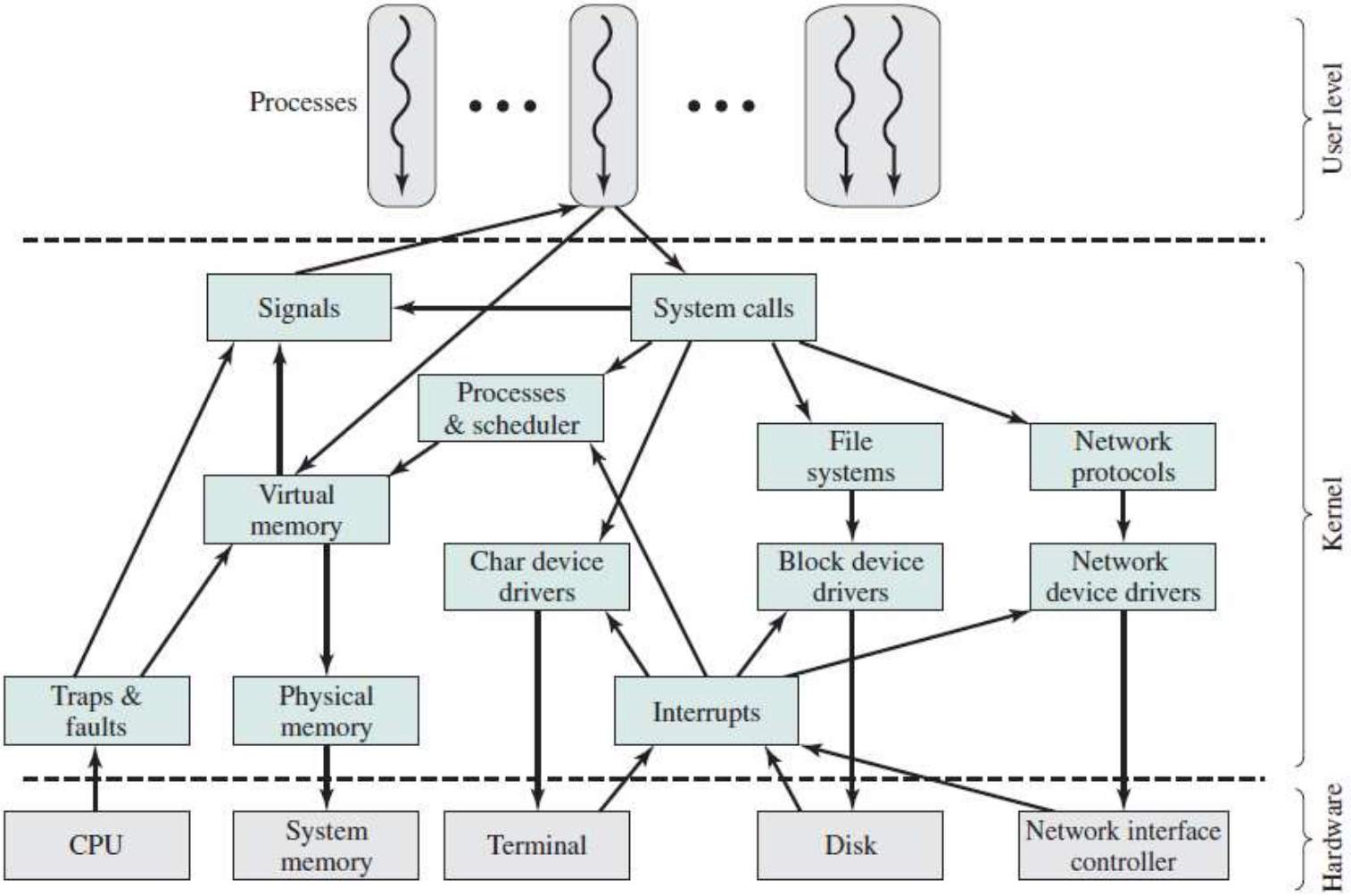
UNIX


(the users)
shells and commands
compilers and interpreters
system libraries
system-call interface to the kernel
signals terminal handling
character I/O system
terminal drivers
file system
swapping block I/O system
disk and tape drivers
CPU scheduling
page replacement
demand paging
virtual memory
kernel interface to the hardware
terminal controllers
terminals
device controllers
disks and tapes
memory controllers
physical memory
Kernel

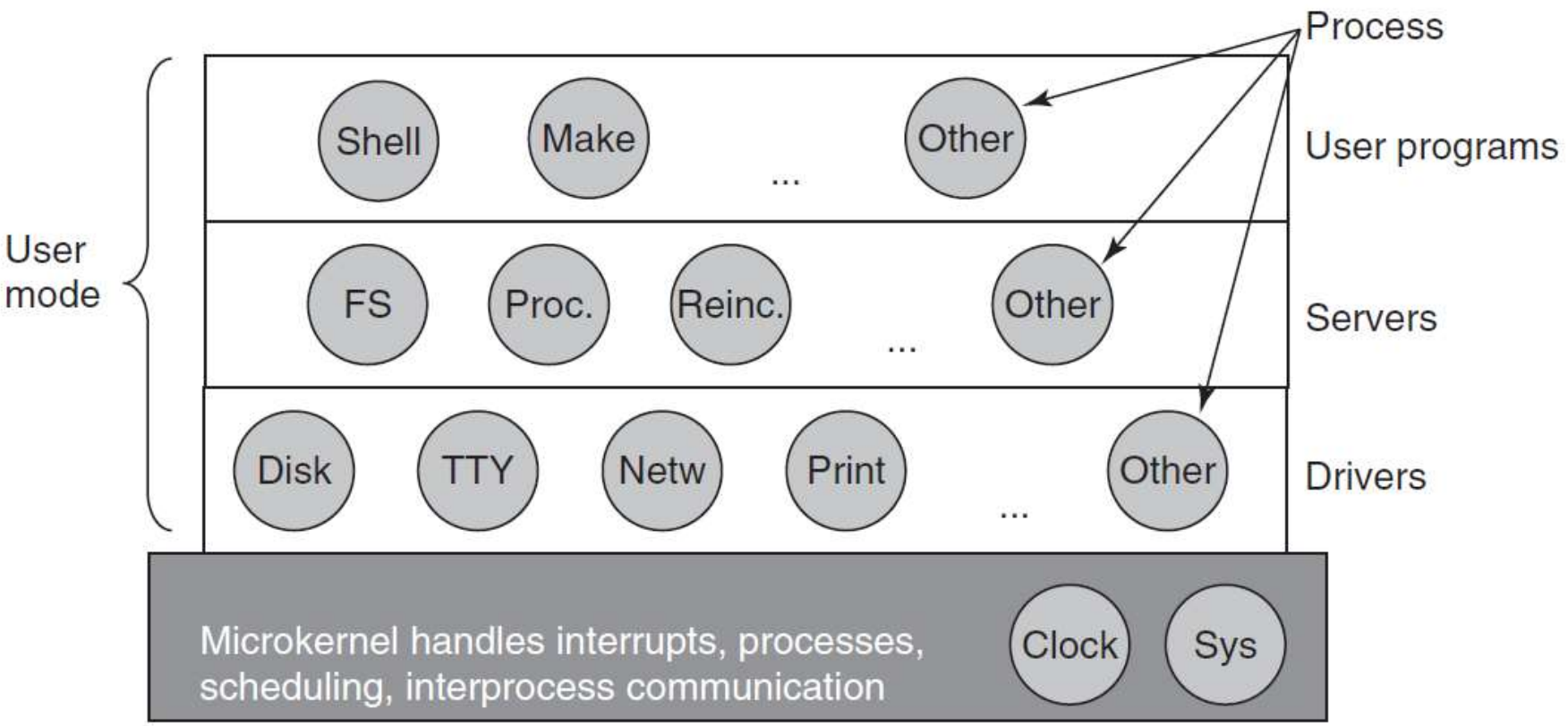
Abordagem Modular (Solaris)



Linux


Processes
User level
Signals
System calls
Processes & scheduler
File systems
Network protocols
Virtual memory
Char device drivers
Block device drivers
Network device drivers
Traps & faults
Physical memory
Interrupts
Kernel
CPU
System memory
Terminal
Disk
Network interface controller
Hardware

Microkernel (MINIX)



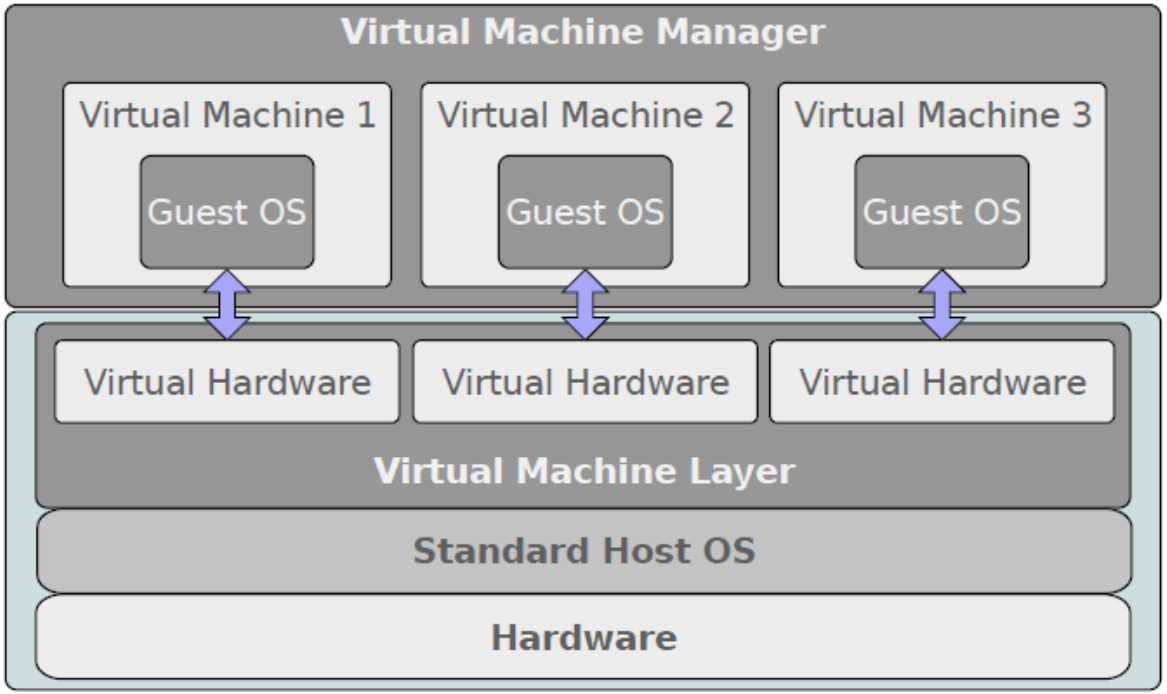
- Open Source project (MINIX 3 – www.minix3.org)
- *Kernel* has ~4k lines of code
  - Linux has 5M lines of code

Máquinas Virtuais

- As máquinas virtuais levam a organização em camadas até ao limite
- Ambientes de execução virtuais, possivelmente a executar SO distintos, em que os recursos do computador (CPU, memória, discos, etc) parecem ser exclusivos apesar de serem partilhados
- O SO cria a ilusão de que cada processo corre no seu processador com a sua memória
- Vários sistemas operativos podem executar concorrentemente em máquinas virtuais distintas tendo cada um o seu conjunto de processos em execução

Full Virtualization

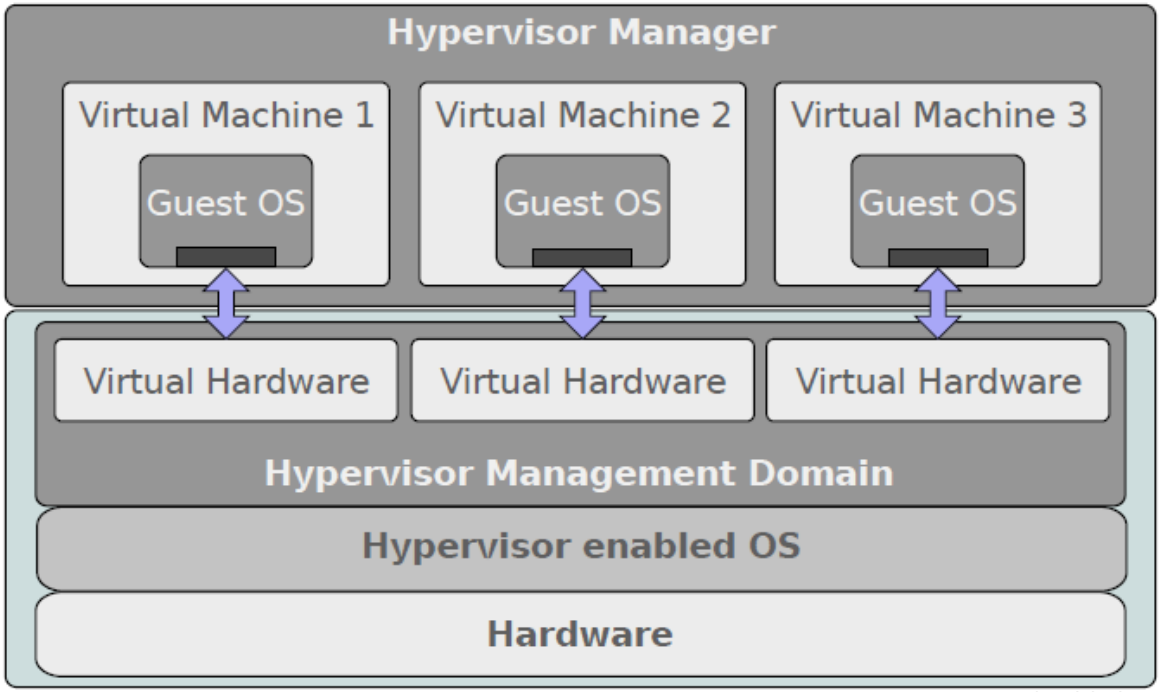
- Emulação do hardware
  - Através do *host* OS
- Transparente para o(s) *guest* SO(s)
- Exemplos: Virtual Box, VMWare



UPC, ASO, Serral

Paravirtualization

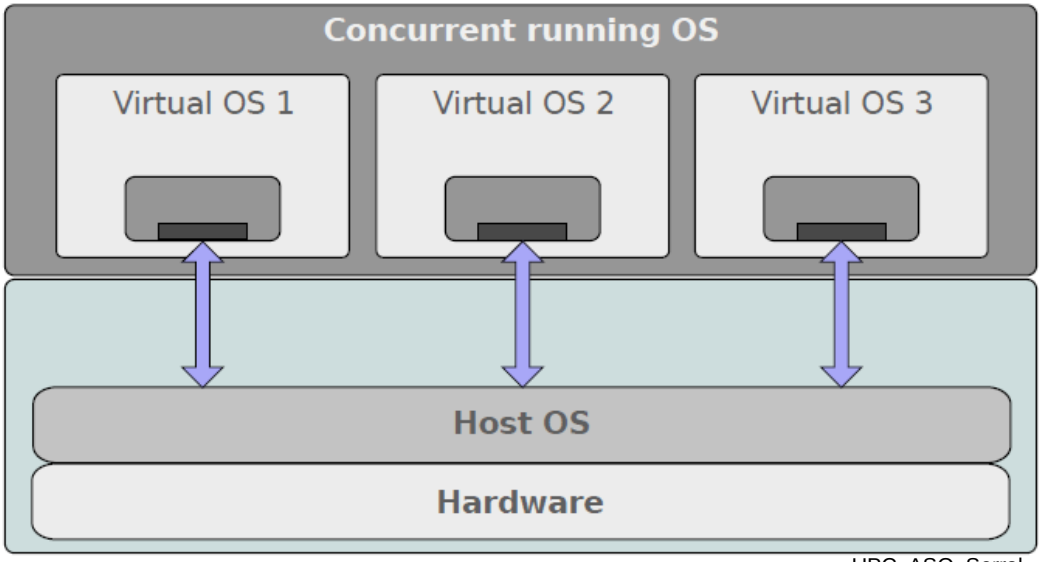
- Execução concorrente de vários SOs
- Necessita de suporte de hardware e dos SOs
- Exemplos: Xen, UML



UPC, ASO, Serral

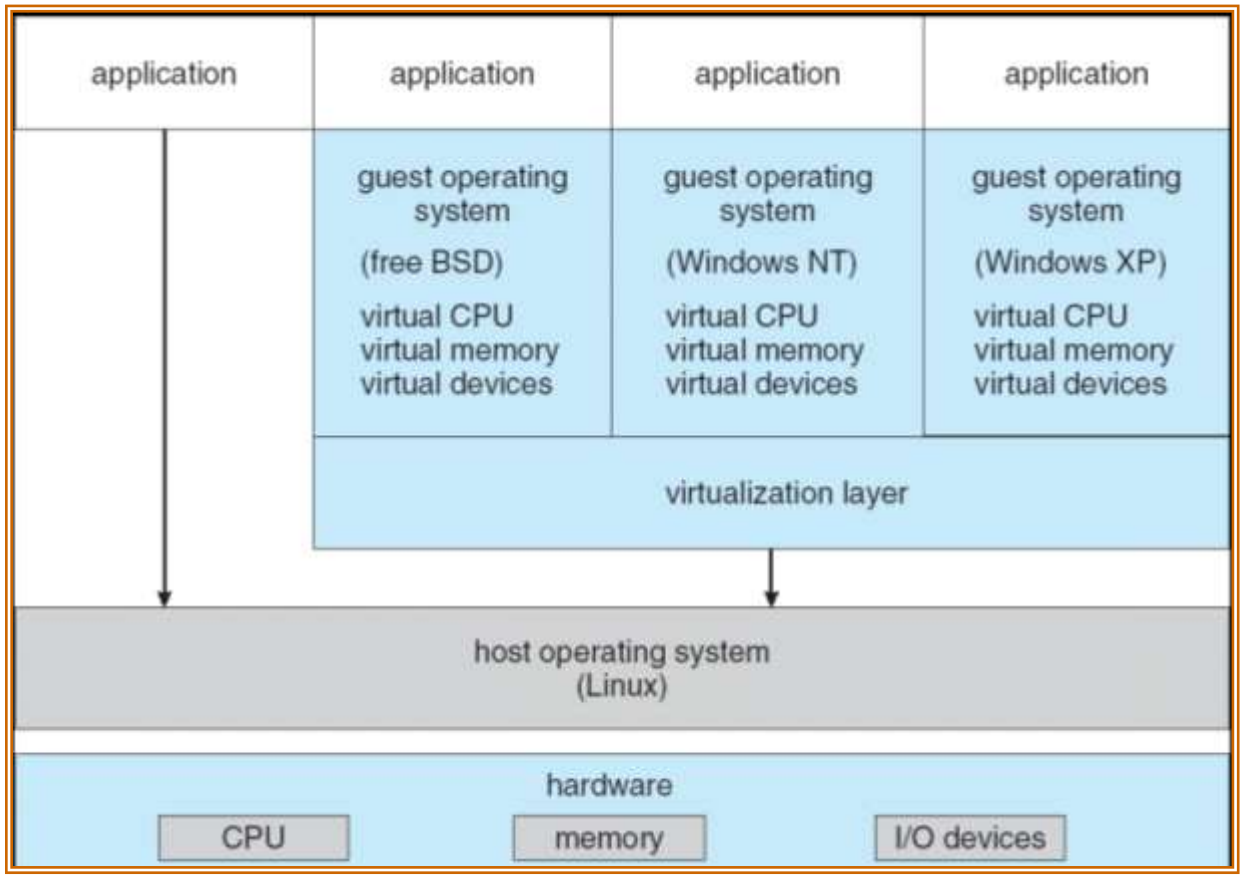
OS Virtualization

- Colaboração entre *host* e *guests*
  - Acesso direto ao hardware dos *guests*
  - Pode executar em *userspace*
- Necessita de suporte do SO
  - *Host* e *guests* usam o mesmo SO
- Exemplos: Docker.io, Solaris containers

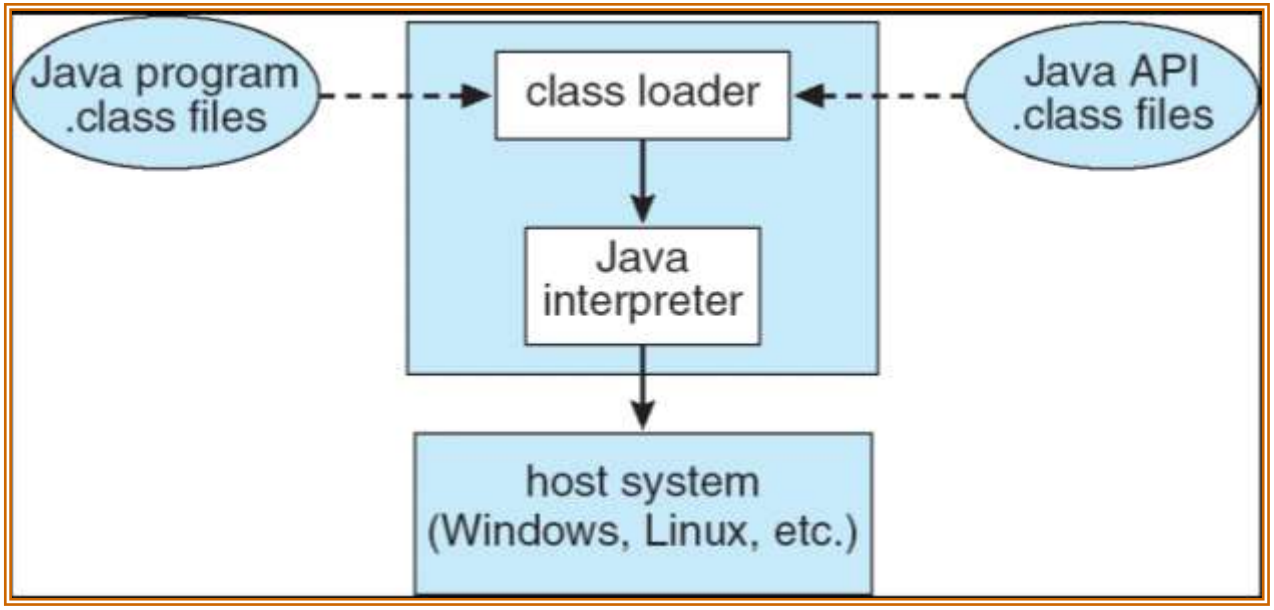


UPC, ASO, Serral

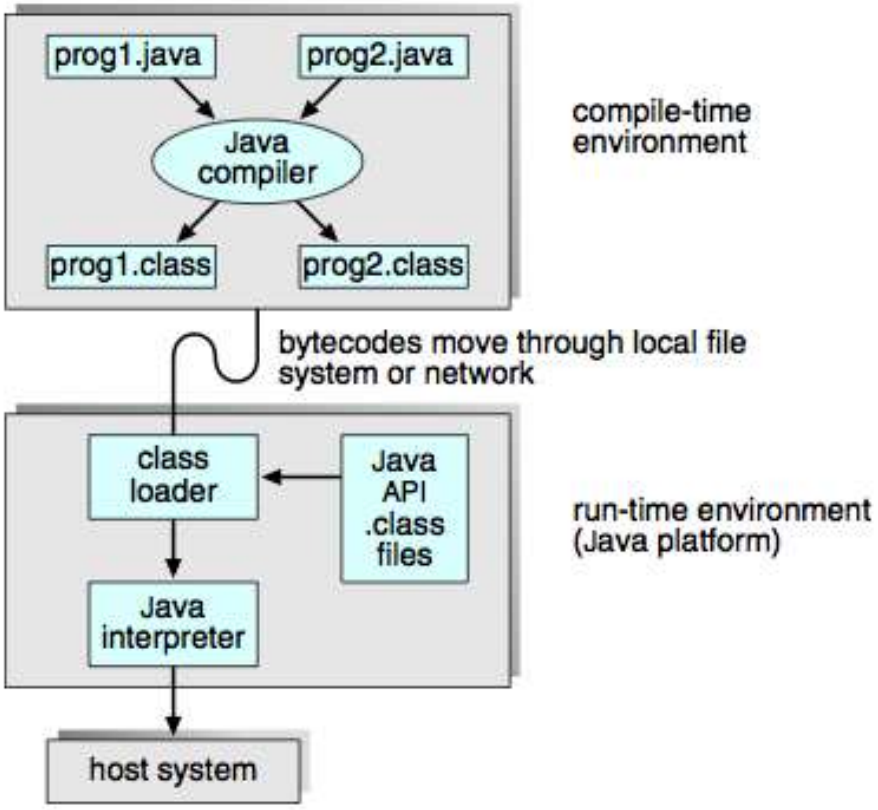
VMware


application
application
application
application
guest operating system
(free BSD)
virtual CPU
virtual memory
virtual devices
guest operating system
(Windows NT)
virtual CPU
virtual memory
virtual devices
guest operating system
(Windows XP)
virtual CPU
virtual memory
virtual devices
virtualization layer
host operating system
(Linux)
hardware
CPU
memory
I/O devices

Java Virtual Machine



Desenvolvimento em Java



Módulos em Linux

- O Linux pode carregar e remover módulos do *kernel* durante a sua execução
  - São necessárias permissões de *superuser*
  - Comandos principais:
    - **`lsmod`**, **`modinfo`**, **`modprobe`**, **`rmmod`**

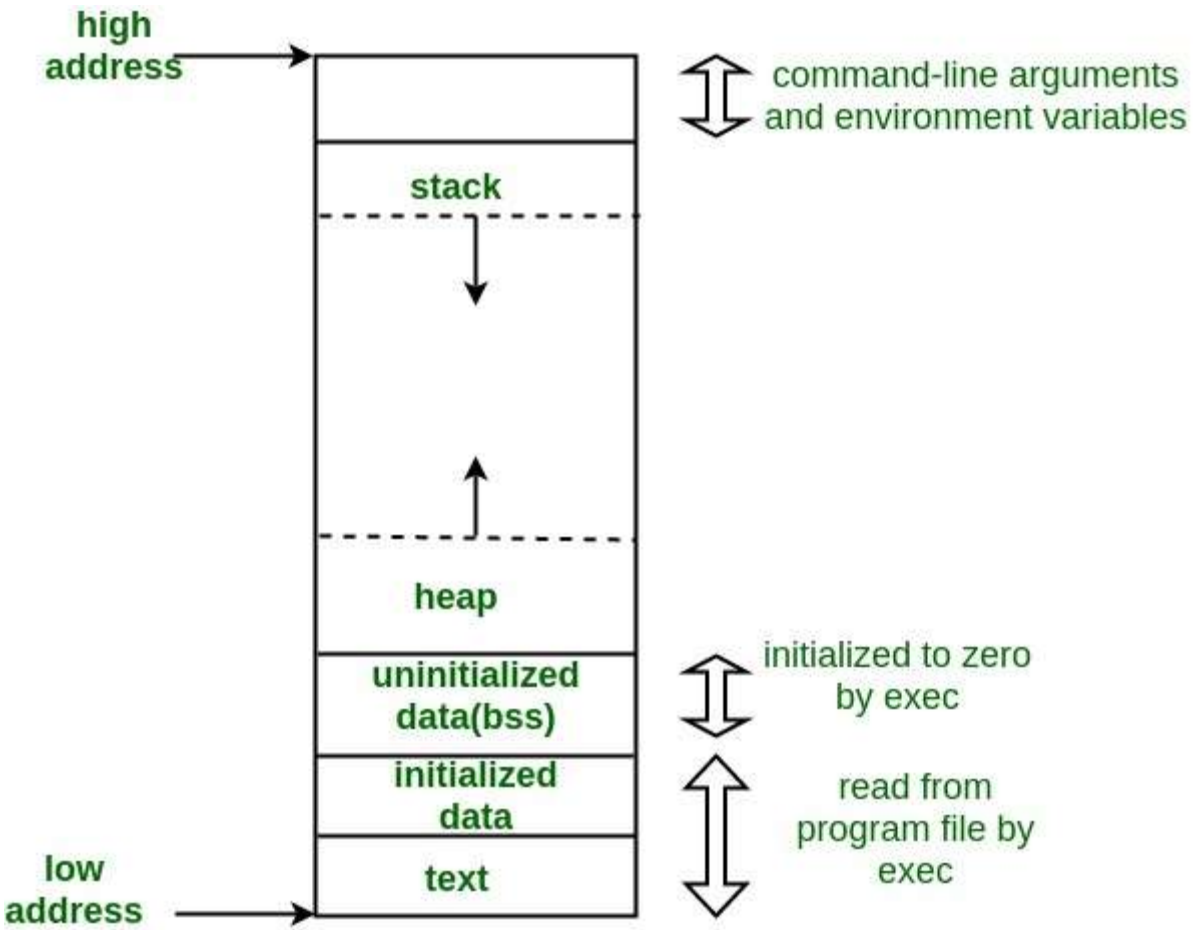
Processo

- Programa em execução
- Criar um processo
  - Inicialização do Sistema
  - Execução de chamada ao sistema por processo em execução
  - Pedido do utilizador para criar novo processo
  - Início de um *batch script*
- Processos podem correr em:
  - *foreground*: interage com utilizador
  - *background*: executa sem interação, *daemon*

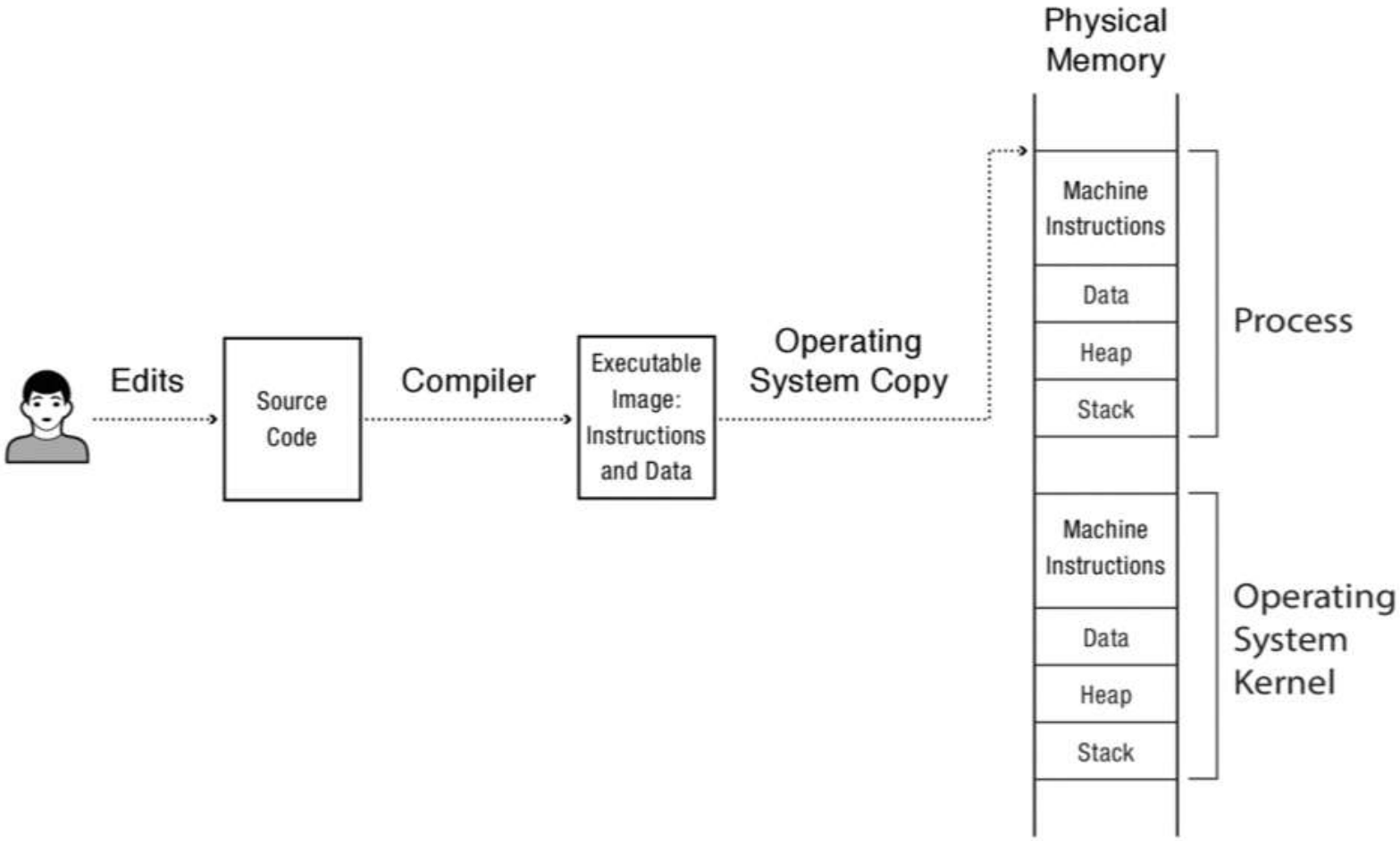
Processo

- Programa em execução
- Criar um processo
  - Inicialização do Sistema
  - Execução de chamada ao sistema por processo em execução
  - Pedido do utilizador para criar novo processo
  - Início de um *batch script*
- Processos podem correr em:
  - *foreground*: interage com utilizador
  - *background*: executa sem interação, *daemon*

Memória de um processo



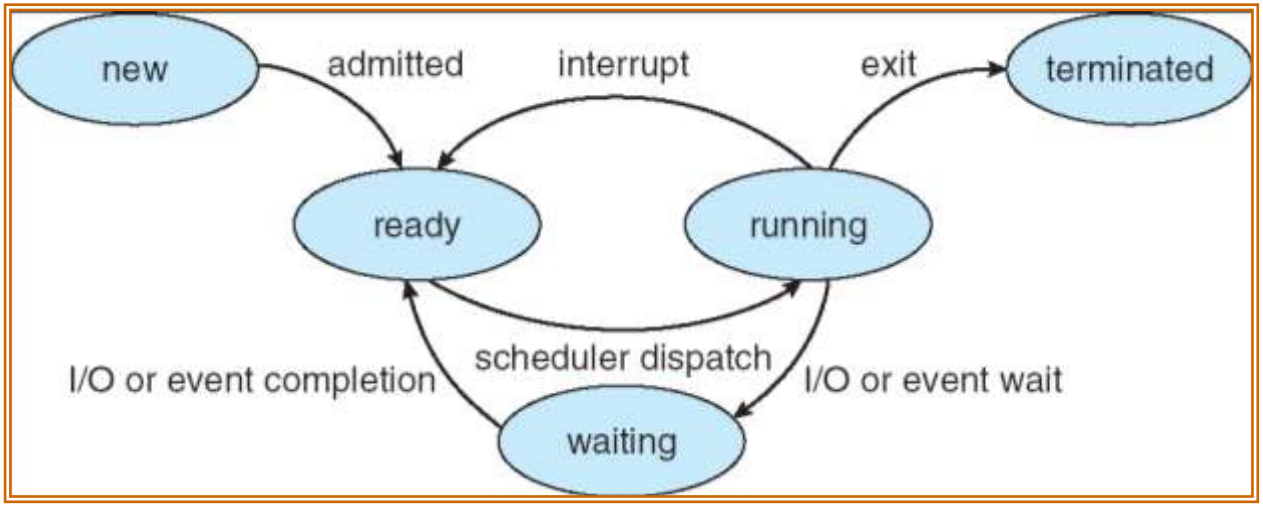
Memória processo/OS



Estados de um processo

- Um processo em execução pode estar nos seguintes estados
  - *New*
  - *Running*
  - *Waiting*
  - *Ready*
  - *Terminated*

Estados de um processo



Estados de um processo