# Sistemas Operativos

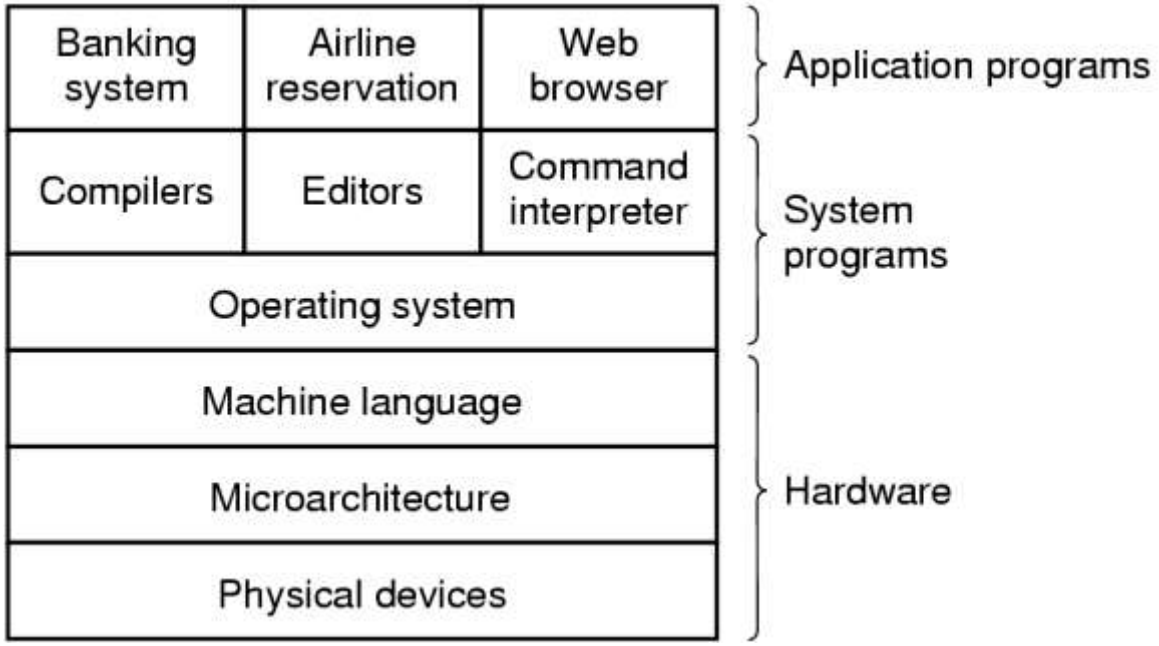
Licenciatura Engenharia Informática
Licenciatura Engenharia Computacional

Ano letivo 2024/2025

Nuno Lau (nunolau@ua.pt)

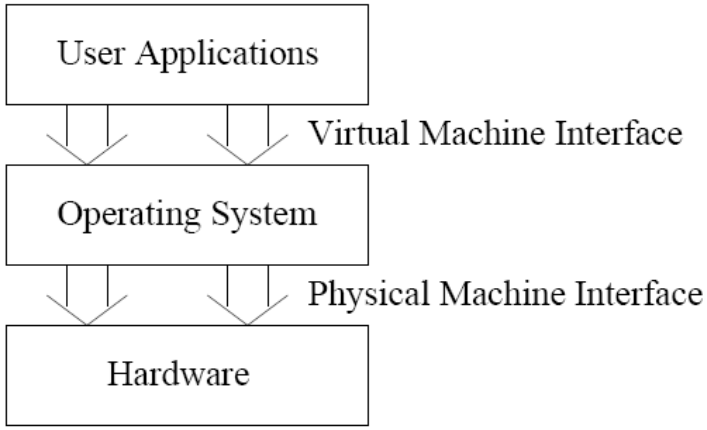
# O que é um Sistema Operativo?

- O Sistema Operativo é o programa base que estabelece a interface entre os programas de aplicação e o hardware.



# Objectivos do Sistema Operativo

- Executar os programas de aplicação
- Tornar o hardware mais fácil de usar
  - O SO cria um nível de abstração que esconde muitos dos pormenores da utilização de dispositivos específicos (usando *device drivers*)
- Usar o hardware de forma eficiente
  - O SO gere os recursos de hardware do sistema de forma a tornar a sua utilização mais eficiente, justa e segura



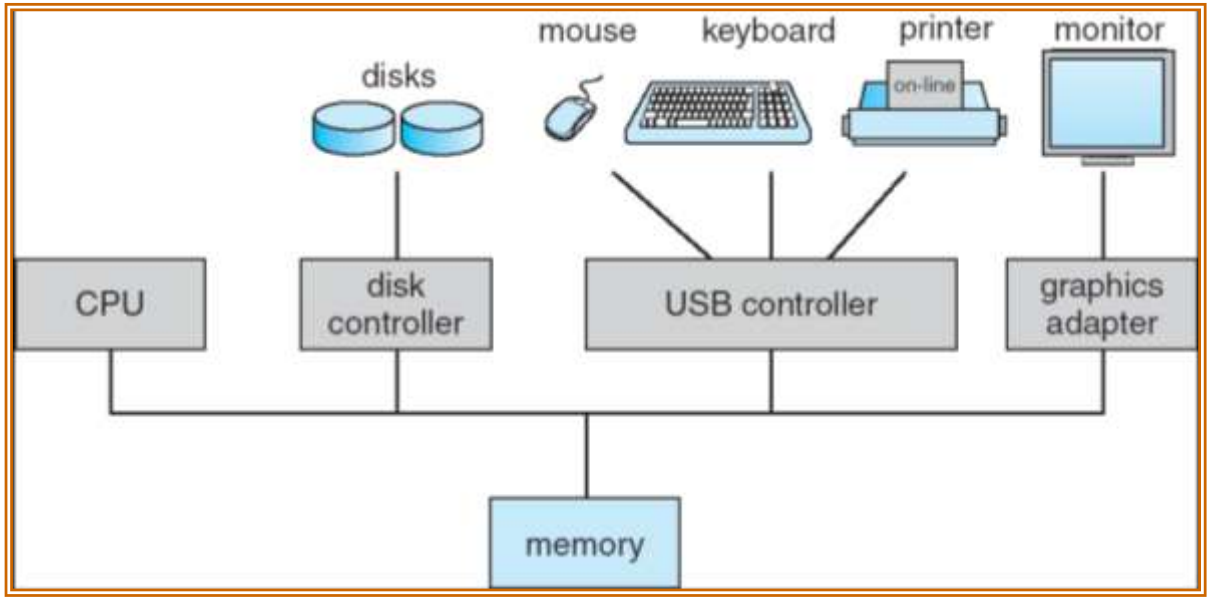

Os 2 últimos objectivos podem facilmente entrar em conflito

# Arranque do computador

- Programa de *bootstrap* é carregado quando o computador arranca ou é reinicializado
  - Tipicamente armazenado em ROM ou EPROM (*firmware - BIOS*)
  - Inicializa vários dispositivos do sistema (teclado, etc.)
  - Carrega o núcleo (*kernel*) do sistema operativo, ou um *bootloader*, e começa a sua execução

# Organização do computador

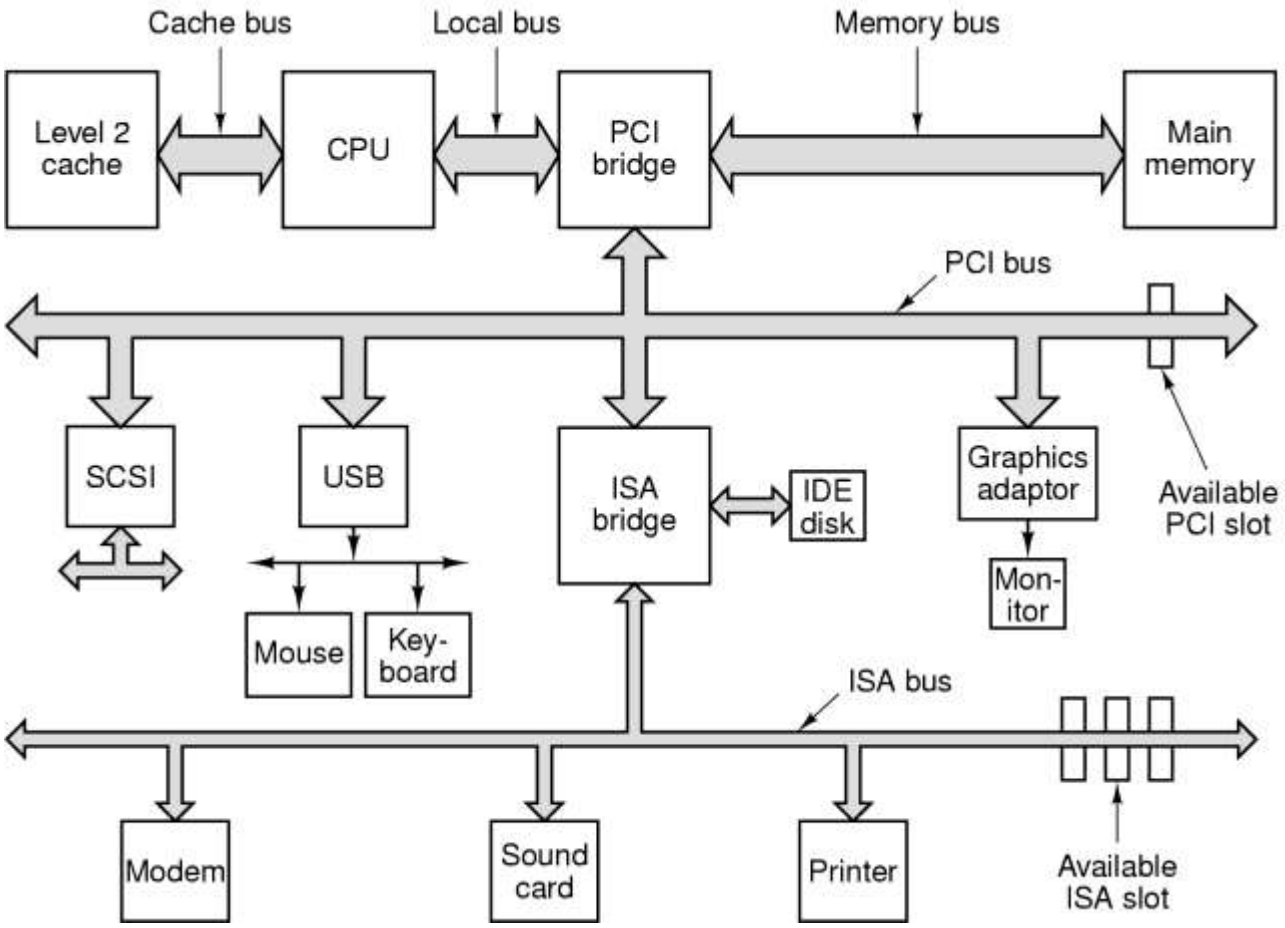
- Um ou mais CPUs e controladores de dispositivos ligados à memória através de barramento
- Execução concorrente de CPU e dispositivos origina conflitos no acesso à memória



# Organização do computador

- CPUs e controladores de dispositivos de I/O executam em paralelo
- Cada controlador de dispositivo trata um tipo particular
- Controladores de dispositivo têm *buffer* local
- CPU move dados de/para memória e de/para *buffers* locais
- Transferências de I/O são do dispositivo para o *buffer* local do respectivo controlador e depois para a memória
- Controlador do dispositivo informa CPU que terminou a operação através do envio de uma interrupção

# Organização do computador



# Gestão de processos

- Um processo é um programa em execução
  - Programa é uma entidade passiva, enquanto que o processo é activo
- Processo necessita de recursos
  - CPU, memória, I/O, ficheiros, etc.
  - Dados de inicialização
- Finalização de um processo deve libertar os recursos que forem reutilizáveis
- Processo com 1 *thread* tem um *Program Counter* (PC) que indica qual a próxima instrução a executar
  - Processo executa as instruções sequencialmente até terminar
- Processo *Multi-threaded* tem 1 PC por *thread*
- Tipicamente o sistema tem muitos processos, alguns do utilizador outros do SO que correm em concorrência em 1 ou mais CPUs
- Concorrência obtêm-se através da multiplexagem no tempo dos CPUs entre os vários processos/*threads*

# Gestão de memória

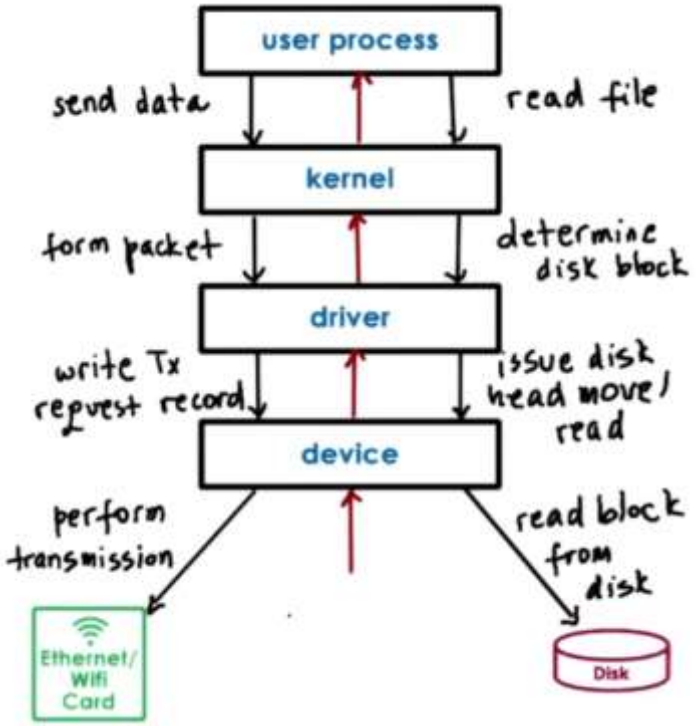
- Memória física é um recurso escasso (e caro)
- Todos os dados em memória antes e depois do processamento
- Todas as instruções em memória para poderem ser executadas
- Gestão da memória determina o que deve estar em memória em cada altura
  - Tentando maximizar a utilização do CPU
  - Disponibiliza uma utilização da memória mais transparente
  - Garante a segurança na utilização da memória pelos vários processos
  - Permite a partilha de memória entre processos
- Actividades de gestão de mémória
  - Conhecer quais as zonas de memória livres e ocupadas
  - Decidir que processos e dados mover para ou para fora da memória
  - Alocar e desalocar espaço de memória

# Gestão de armazenamento

- SO disponibiliza uma vista lógica uniforme do espaço de armazenamento
  - Abstração das propriedades físicas da unidade de armazenamento – **ficheiro**
  - Controladores distintos para tipos de unidades diferentes (disco, tape, etc)
    - Propriedades muito diferentes (velocidade, capacidade, etc)
- Sistemas de Ficheiros
  - Ficheiros organizados em diretorias/pastas
  - Permissões garantem acessos com segurança aos dados
  - Atividades do SO
    - Criar  e apagar ficheiros e diretorias
    - Gerir diferentes Sistemas de Ficheiros de forma integrada
    - Primitivas de manipulação de ficheiros e diretorias
    - Mapeamento dos ficheiros na unidade de armazenamento
    - *Backups*

# Sub-sistema de I/O

- Um dos objetivos do SO é esconder os pormenores dos dispositivos de hardware do utilizador
- SO é responsável por
  - Gestão da memória de I/O
  - Interface geral dos *device drivers*
  - *Drivers* para dispositivos específicos de hardware




https://applied-programming.github.io/

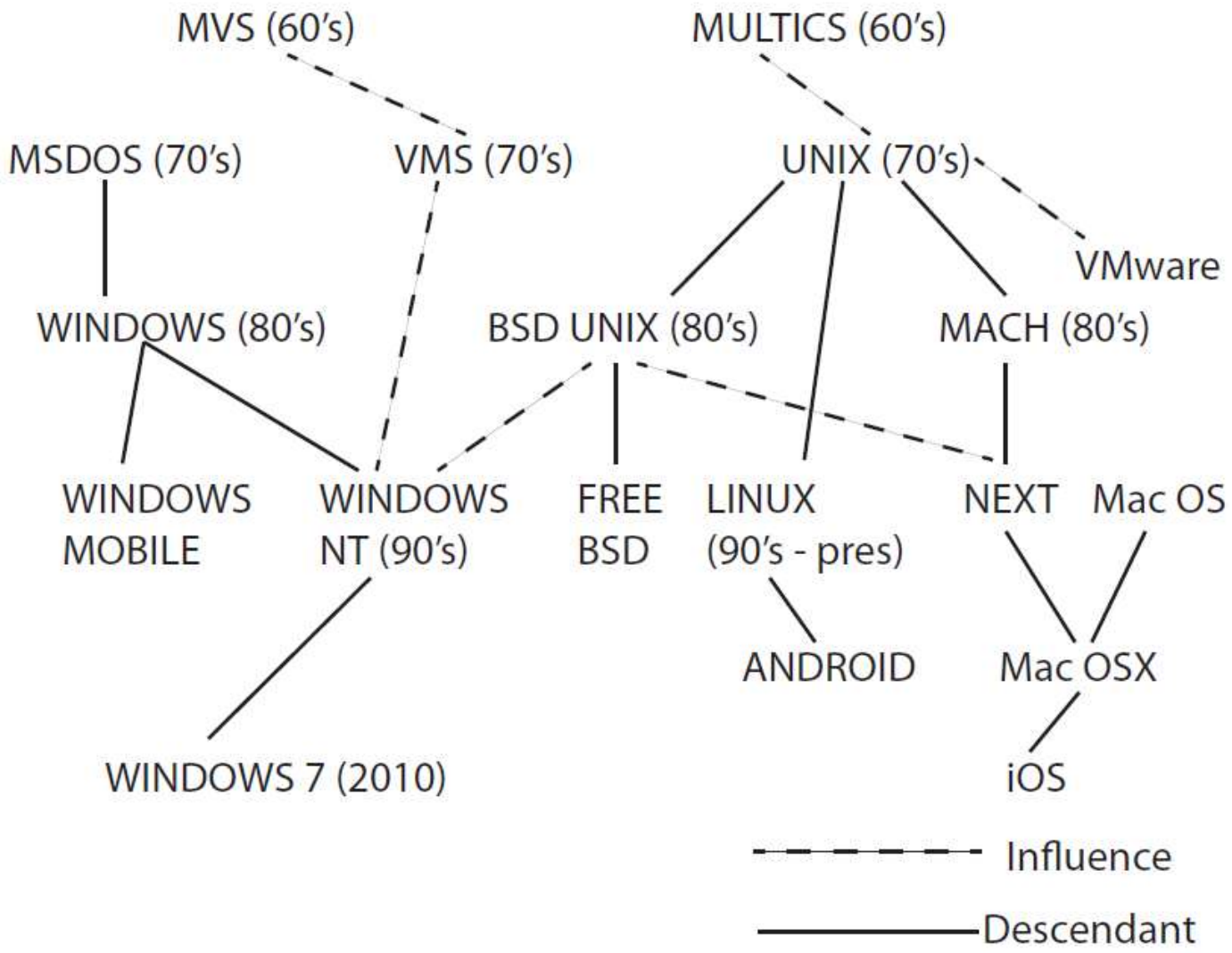
# Proteção e Segurança

- Proteção – mecanismo do SO para controlar o acesso de processos e utilizadores aos recursos
- Segurança – Defesa do sistema contra ataques internos e externos
  - Vasta gama: *denial-of-service*, vírus, roubo de identidade, etc.
- Sistemas permitem distinguir os utilizadores, definindo assim permissões diferentes
  - Identificação do utilizador (uid, gid, etc)
  - Associação da identificação a ficheiros e processos que o utilizador controla
  - Identificadores de grupo permitem a definição de políticas mais gerais
  - Utilizadores podem alterar o ID de forma controlada
    - Comando **`su`**
    - Comando **`chown`**
    - **`SUID`** bit permission

# SO atuais

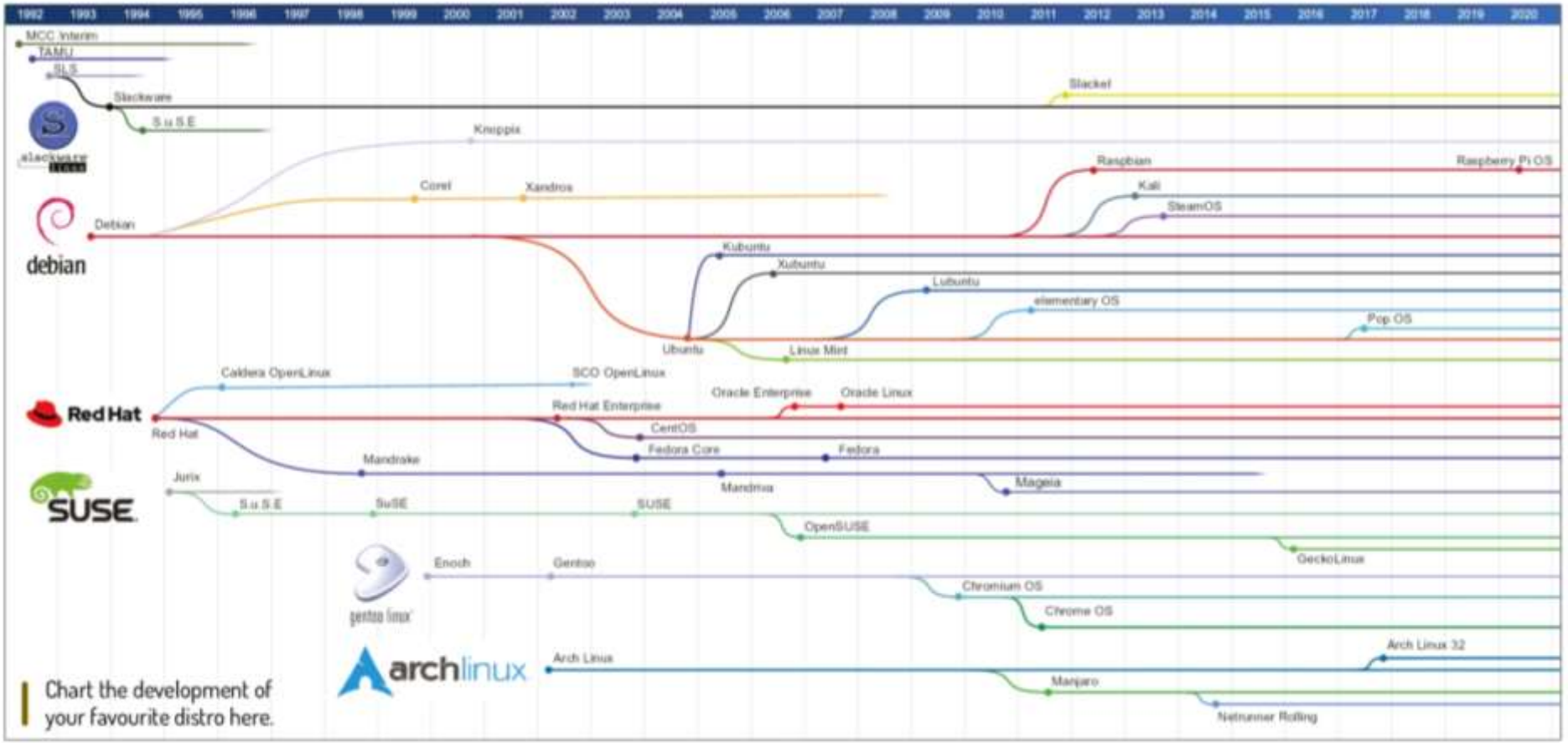
- Ambiente gráfico com o utilizador
- Multiutilizador e Multitarefa
- Memória virtual
- Sistemas de operação de rede
  - Acesso indistinto a ficheiros/dispositivos locais ou de rede
  - Aplicações de login remoto, correio electrónico, navegação internet, etc
- Enorme gama de *device drivers*
- Ligação dinâmica de dispositivos (*plug and play*)

# Relações entre SOs




Operating Systems Principles and Practice, Anderson and Dahlin

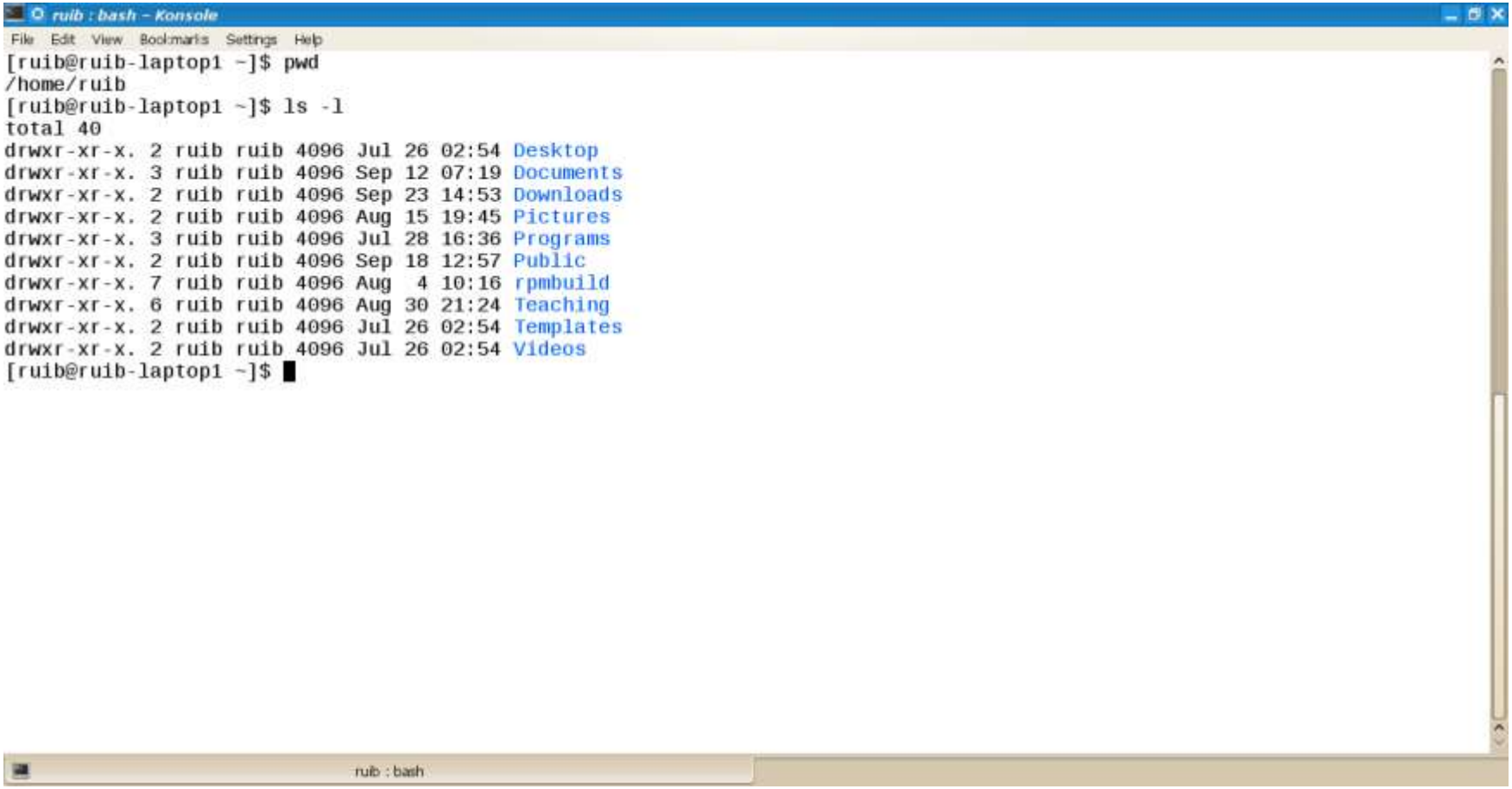
# Linux Chart




**CREDIT:** Based on the LinuxTimeLine, by fabiololix, GNU Free Documentation License v1.3, https://github.com/FabioLolix/LinuxTimeline/tree/master

# Interface Utilizador-SO

- Interpretador de comandos (CLI)

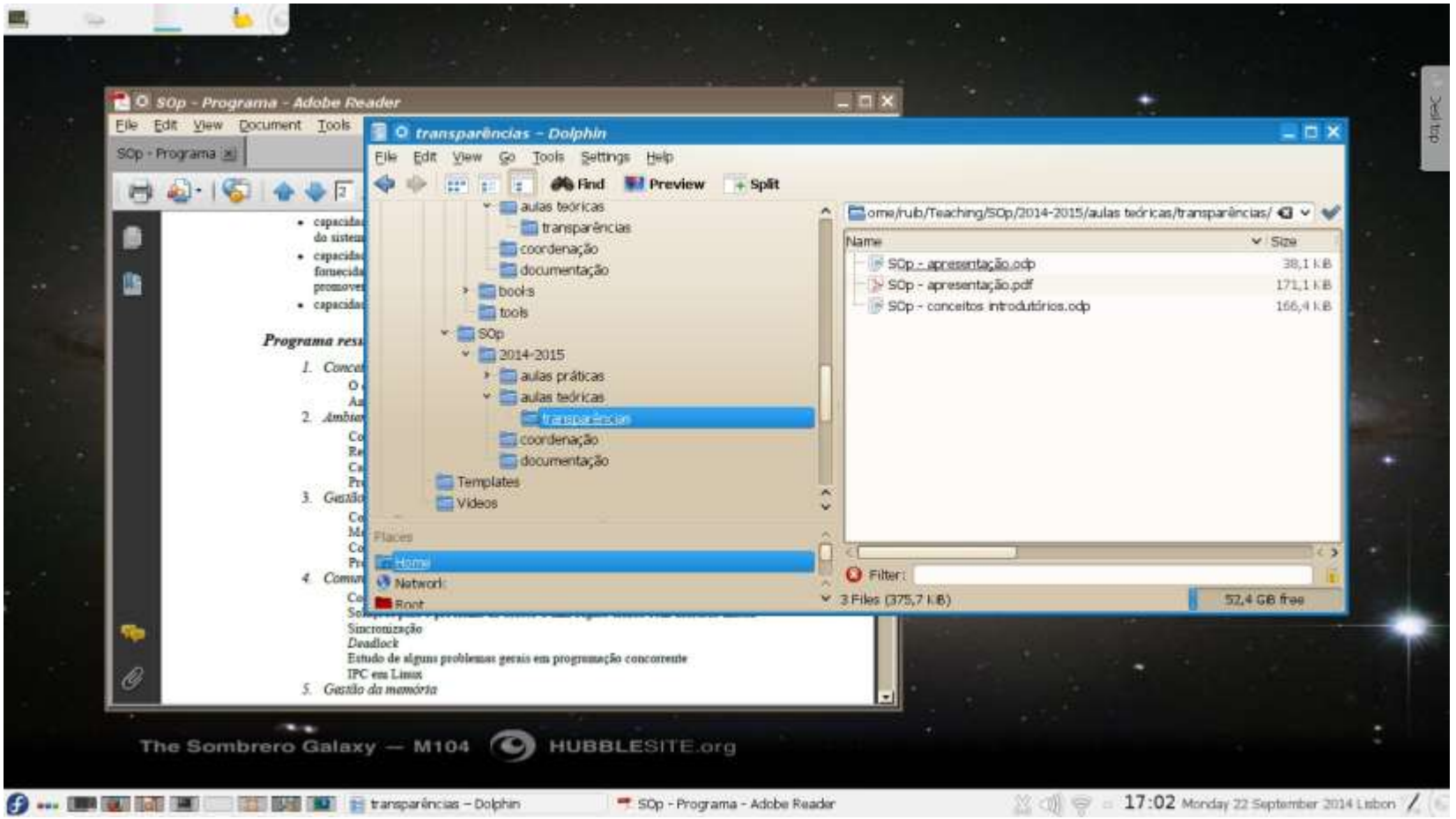


# Interface Utilizador-SO

- Interpretador de comandos (CLI)
  - Pode estar incluído no *kernel* ou funcionar como um programa de sistema
  - Se programa independente designado de *shell*
    - *Bourne shell, C shell, Korn shell, etc*
  - Função principal: ler e executar comandos do utilizador
  - Muitos destes comandos estão relacionados com a gestão de ficheiros
  - O código que efetivamente manipula os ficheiros pode estar integrado no interpretador de comandos ou usar programas independentes deste
    - Comandos internos e externos
      - Comando **`type`**
    - No caso de serem programas independentes a *shell* não necessita de perceber quais os efeitos do comando executado

# Interface Utilizador-SO

- Interface gráfica (GUI)



# Interface Utilizador-SO

- Interface gráfica (GUI)
  - Sistema baseado em janelas e menus e preparado para ser manipulado através do rato
  - Usando o rato o utilizador pode seleccionar ficheiros, executar programas e abrir vários tipos de menus
  - Metáfora do “ambiente de trabalho”
  - A primeira interface gráfica apareceu em 1973, tendo sido desenvolvida no laboratório Xerox PARC (Silicon Valley)
  - Vários sistemas disponibilizam GUI e CLI
    - Windows é GUI, mas tem CLI como *shell* de comandos
    - Apple Mac OS X tem Aqua como GUI e várias *shells* disponíveis
    - Sistemas UNIX são em geral baseados em CLI, mas com várias GUI disponíveis (KDE, Gnome, etc)

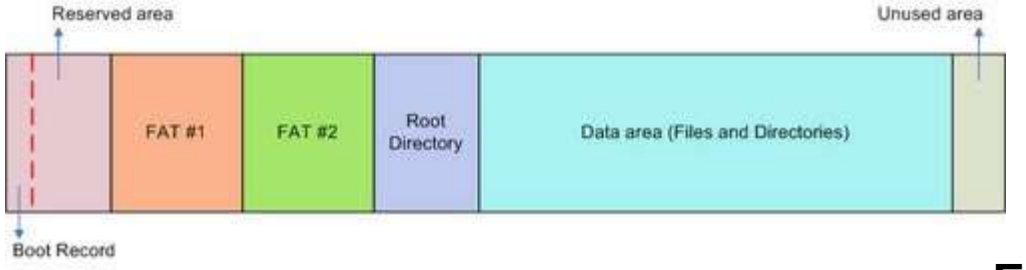
# FAT32

- Sistema de Ficheiros FAT32
  - Organização do disco
    - Boot sector (boot loader, partition table, BIOS parameter block, etc.)
    - 2 cópias da FAT
    - Zona de dados (directoria raiz no início da zona de dados)
  - Como encontrar conteúdo de um ficheiro?
    - Cluster inicial indicado na entrada de diretoria
    - Clusters seguintes **numa lista ligada** através da FAT
  - FAT
    - Array de números de clusters (cada um com 32 bits)
    - Para cada cluster indica:
      - Cluster seguinte; Final ou Livre
  - Directorias
    - Constituídas por entradas de diretoria de tamanho fixo
    - Entradas definem nome e metadados de ficheiros e diretorias
    - Entradas definem cluster inicial

# FAT32

- Sistema de Ficheiros FAT32

**Disco**




**FAT**

| | | | |
|---|---|---|---|
| XXXXXXXX | XXXXXXXX | 00000009 | 00000004 |
| 00000005 | 00000007 | 00000000 | 00000008 |
| FFFFFFFF | 0000000A | 0000000B | 00000011 |
| 0000000D | 0000000E | FFFFFFFF | 00000010 |
| 00000012 | FFFFFFFF | 00000013 | 00000014 |
| 00000015 | 00000016 | FFFFFFFF | 00000000 |
| 00000000 | 00000000 | 00000000 | 00000000 |
| 00000000 | 00000000 | 00000000 | 00000000 |
| 00000000 | 00000000 | 00000000 | 00000000 |
| 00000000 | 00000000 | 00000000 | 00000000 |
| 00000000 | 00000000 | 00000000 | 00000000 |
| 00000000 | 00000000 | 00000000 | 00000000 |
| 00000000 | 00000000 | 00000000 | 00000000 |
| 00000000 | 00000000 | 00000000 | 00000000 |
| 00000000 | 00000000 | 00000000 | 00000000 |

**Root Directory:**
2, 9, A, B, 11

**File #1:**
3, 4, 5, 7, 8

**File #2:**
C, D, E

**File #3:**
F, 10, 12, 13, 14, 15, 16

**Entrada de diretoria**

| Root Directory SFN Entry Data Structure | |
|---|---|
| Bytes | Purpose |
| 0 | First character of file name (ASCII) *or* allocation status (0x00=unallocated, 0xe5=deleted) |
| 1-10 | Characters 2-11 of the file name (ASCII); the "." is implied between bytes 7 and 8 |
| 11 | File attributes (see File Attributes table) |
| 12 | Reserved |
| 13 | File creation time (in tenths of seconds)* |
| 14-15 | Creation time (hours, minutes, seconds)* |
| 16-17 | Creation date* |
| 18-19 | Access date* |
| 20-21 | High-order 2 bytes of address of first cluster (0 for FAT12/16)* |
| 22-23 | Modified time (hours, minutes, seconds) |
| 24-25 | Modified date |
| 26-27 | Low-order 2 bytes of address of first cluster |
| 28-31 | File size (0 for directories) |