

# Sistemas Operativos

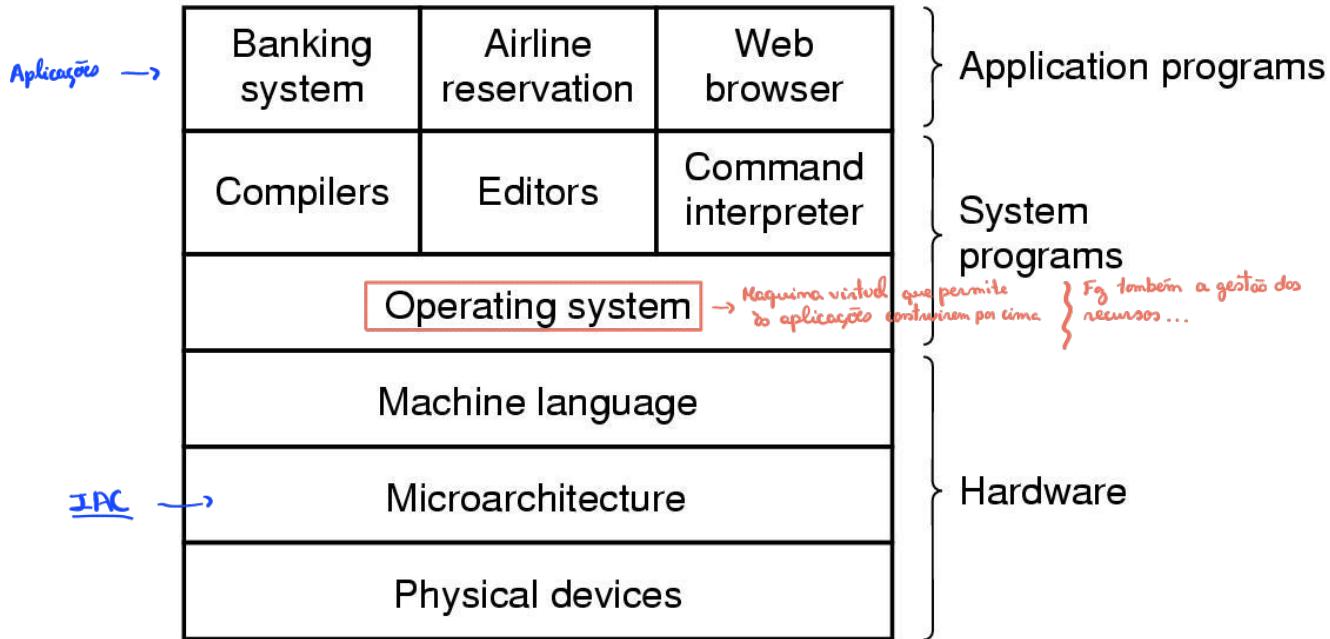
Licenciatura Engenharia Informática  
Licenciatura Engenharia Computacional

Ano letivo 2022/2023

Nuno Lau ([nunolau@ua.pt](mailto:nunolau@ua.pt))

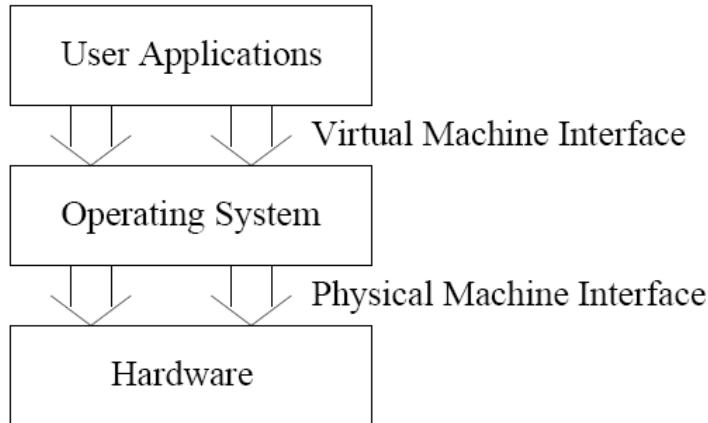
# O que é um Sistema Operativo?

- 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 (Maquina Virtual)
  - 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

# Gestão de processos

- Um processo é um programa em execução → Processo é o recurso de SO que suporta a execução de um programa
  - Programa é uma entidade passiva, enquanto que o processo é activo
- Processo necessita de recursos
  - CPU, memória, I/O, ficheiros, etc.
    - Computacional → Precisa sempre!
    - Input/Output
  - 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
  - Mais do que um thread a executar em paralelo
  - Processo requisita os recursos!
- Tipicamente o sistema tem muitos processos, alguns do utilizador outros do SO que correm em concorrência em 1 ou mais CPUs
  - estão a disputar pelos recursos
  - estamos a falar dos "cores"!,,
- Concorrência obtém-se através da multiplexagem no tempo dos CPUs entre os vários processos/*threads*
  - Placas gráficas de hoje em dia são usadas também para computação...
    - Normalmente para tarefas paralelizáveis!,,
  - O SO vai alternar o processo A a fazer espera um pouco, então o processo B ... , depois o C, todo na mesma CPU (basicamente ele vai rodando entre processos)

CPU + Memória  
Processo executa o programa  
(podem estar 2 processos a carregar o mesmo programa)

↳ Dinamicamente! // Quando com um processo carregamos um programa, os variáveis aparecem na memória  
↳ O SO garante que os recursos são libertados!  
(mesmo em caso de erros)  
e.g.: StackOverflow

Cada thread terá  
a sua própria  
stack

Em termos de programação é muito útil

↳ O SO vai alternar o processo A a fazer espera um pouco, então o processo B ... , depois o C, todo na mesma CPU (basicamente ele vai rodando entre processos)

# Gestão de memória (RAM)

Estamos a falar essencialmente de RAM. Recurso escasso e caro. Preço de um portátil depende muito da sua RAM!

- Memória física é um recurso escasso (e caro)
- Todos os dados em memória antes e depois do processamento
 

*Os processos têm de estar em memória para serem executados...*
- Todas as instruções em memória para poderem ser executadas
- Gestão da memória determina o que deve estar em memória em cada altura
 

*existem coisas que não são mais utilizadas que outras...*

- Tentando maximizar a utilização do CPU

*Acesso à memória é MUITO mais lento que a frequência do CPU!*
  - Disponibiliza uma utilização da memória mais transparente

*cada processo pensa que a memória é infinita ... // eles não sabem a 100% o que sólido acontece*
  - Garante a segurança na utilização da memória pelos vários processos
 

*e.g.: todos os processos assumem que a stack está no mesmo sítio (mais fácil, mas o SO tem de garantir)*
  - Permite a partilha de memória entre processos

*Um processo não pode alterar/ler a memória de outro processo ...*
- temos de ter alguns cuidados ...*
- Actividades de gestão de memória !
 

*Dados para a memória: RAM  
Para fora da memória: Disco*
  - 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

*O processo também pode alugar memória durante a execução de um programa, mas HEAP, O SO tem de garantir isto!*

# Gestão de armazenamento (DISCO)

- SO disponibiliza uma vista lógica uniforme do espaço de armazenamento
  - A aplicação não quer saber que tipo de disco é! → Em hardware pode ser bastante diferente ...

→ A aplicação não quer saber que tipo de disco é!

Isto é gerido pelo SO // (máquina virtual)

ex.: para não só igual criar um ficheiro numa pen ou num disco, e o hardware é completamente diferente?

sequência ordenada de bytes que podemos controlar e que tem uma estrutura definida... → Tem também um cabeçalho; nome / medida...

- 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

Como organizar  
o Disco  
(memória de  
nossa)

- Ficheiros organizados em diretórios/pastas
- Permissões garantem acessos com segurança aos dados
- Atividades do SO

• O teclado é um ficheiro!  
• A memória na sua forma completa é um ficheiro!

• O utilizador não pode simplesmente fazer um programa que leia coisas que ele não tem acesso... (em código máquina) Existem coisas que o SO consegue. Para garantir isso quando o CPU está a executar código do SO ele está num diferente contexto - ele assume diferentes permissões! // não consegue

Contextos diferentes de execução (mesmo em código máquina)

- Criar e apagar ficheiros e diretórios
- Gerir diferentes Sistemas de Ficheiros de forma integrada
- Primitivas de manipulação de ficheiros e diretórios
- Mapeamento dos ficheiros na unidade de armazenamento
- Backups

Em unix: Todos os ficheiros têm permissões: Dono / Grupo / Outros

↳ Termos de ter sempre mapeado, não pode haver memory leak

- Podemos utilizar diferentes sistemas de ficheiros. (o SO faz isso de forma transparente)

# Sub-sistema de I/O (Input/Output)

- Um dos objetivos do SO é esconder os pormenores dos dispositivos de hardware do utilizador

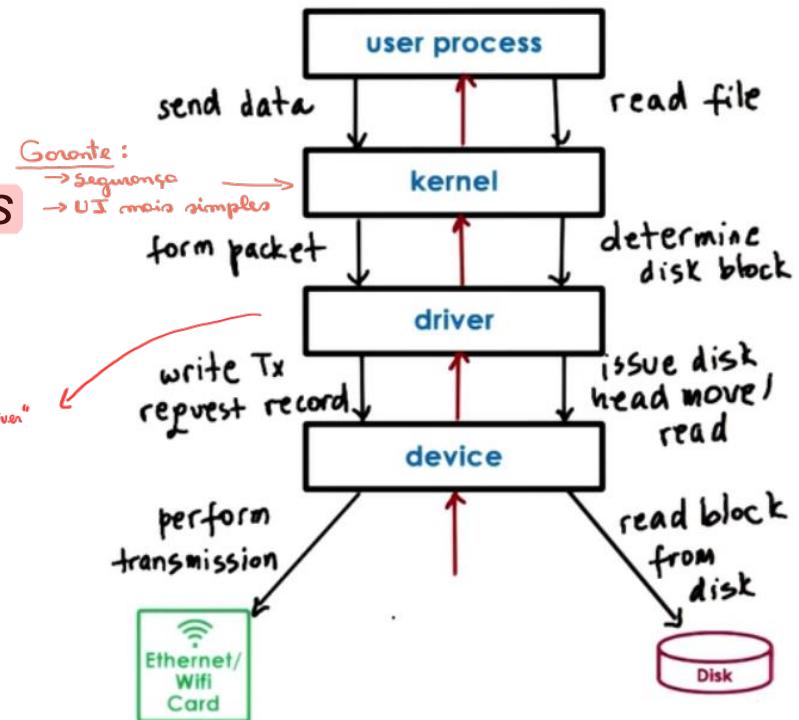
Conjunto de primitivas standard!

- SO é responsável por
  - Gestão da memória de I/O
  - Interface geral dos *device drivers*
  - Drivers para dispositivos específicos de hardware

• Ler um rato/teclado/disco/placa de rede  
... na forma mais básica

• Do ponto de vista da aplicação o acesso a diferentes lugares de rede é igual.

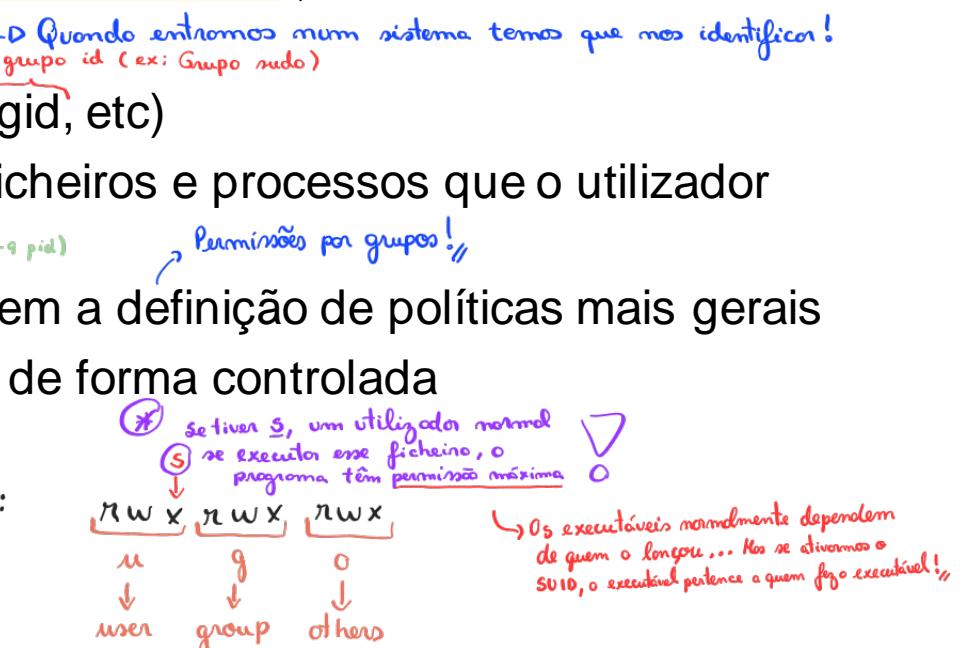
Cada módulo tem o seu "device driver"



# 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
    - Os processos também têm um dono!  
→ Isto acaba com um processo excludente (Kill -9 pid)  
mas precisamos de ser o dono do processo! //
  - Identificadores de grupo permitem a definição de políticas mais gerais
  - Utilizadores podem alterar o ID de forma controlada
    - Comando su → Muda de utilizador
    - Comando chown → Muda o dono
    - SUID bit permission

\$ su, mete o terminal com permissões Root



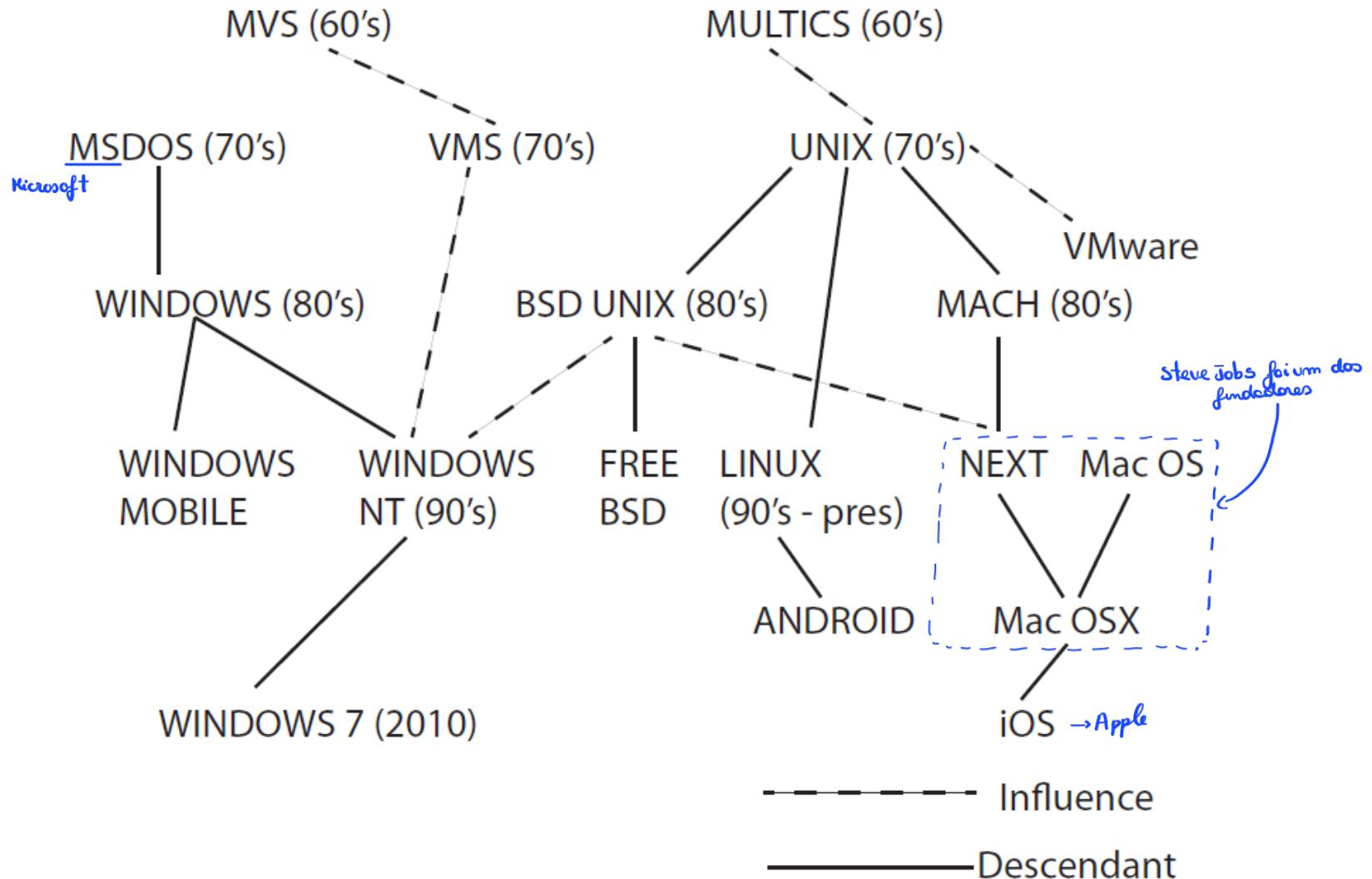
# SO atuais

- 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)

Windows / Unix ↴

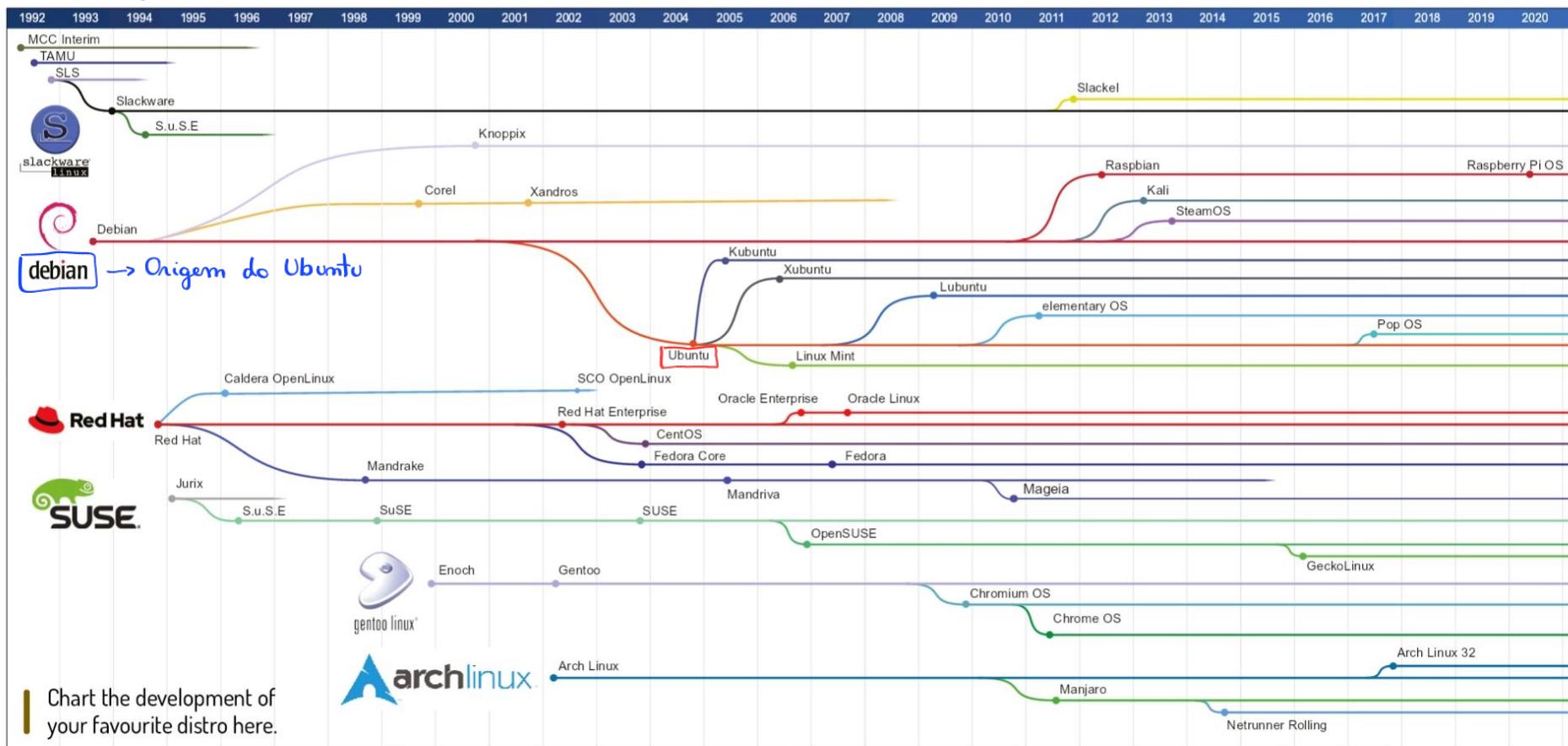
$\text{Ctrl} + \text{Alt} + \frac{\text{F1}}{\text{F2}}$  → Retina a interface gráfica  
• Util para resolver problemas gráficos

# Relações entre SOs



# Linux Chart

Grafico das distribuições!

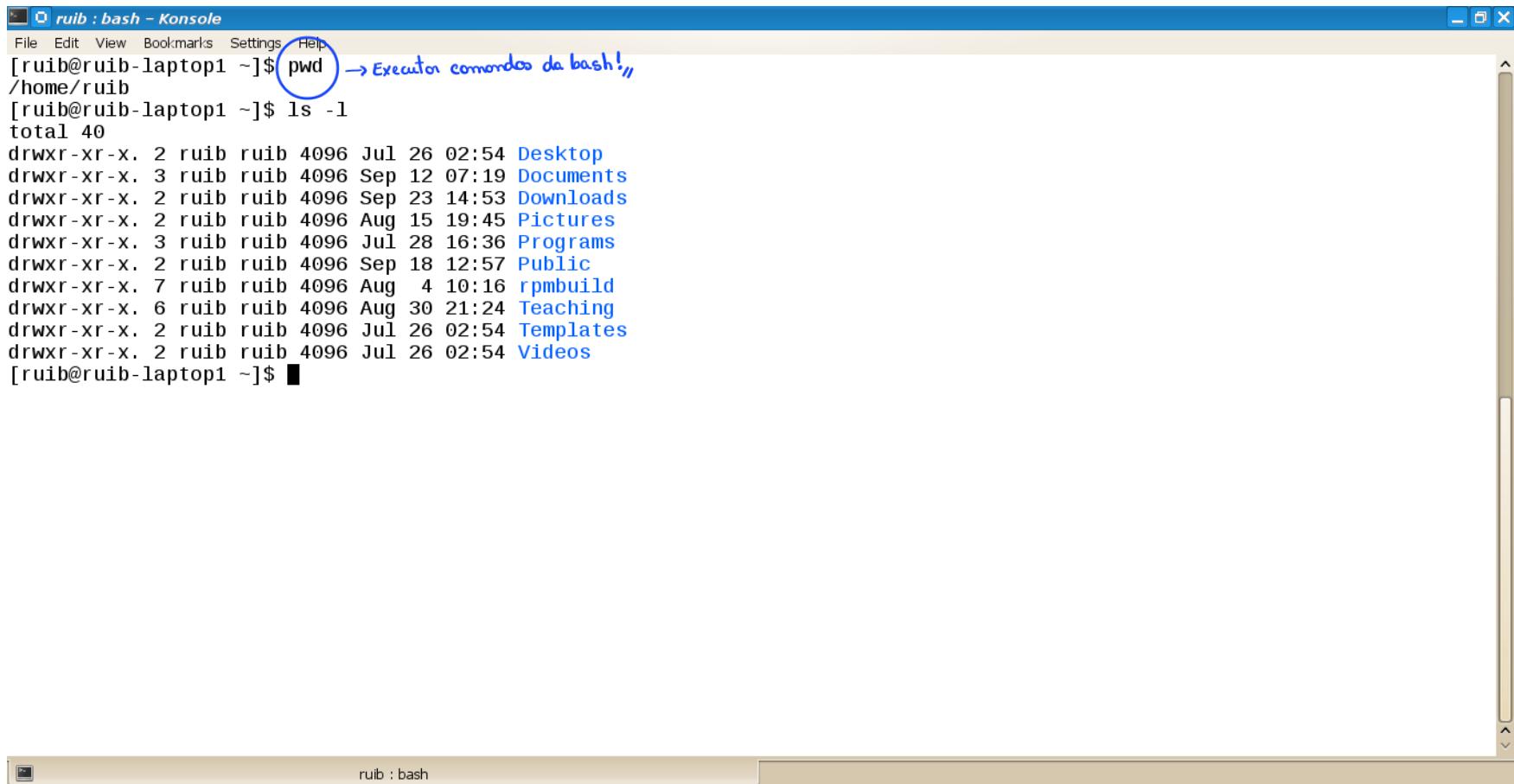


CREDIT: Based on the LinuxTimeline, by fabiololix, GNU Free Documentation License v1.3, <https://github.com/FabioLolix/LinuxTimeline/tree/master>

# Interface Utilizador-SO

## • Interpretador de comandos (CLI)

→ Baseado em linha de comandos! //



```
ruib : bash - Konsole
File Edit View Bookmarks Settings Help
[ruib@ruib-laptop1 ~]$ pwd → Executa comandos da bash! //
/home/ruib
[ruib@ruib-laptop1 ~]$ ls -l
total 40
drwxr-xr-x. 2 ruib ruib 4096 Jul 26 02:54 Desktop
drwxr-xr-x. 3 ruib ruib 4096 Sep 12 07:19 Documents
drwxr-xr-x. 2 ruib ruib 4096 Sep 23 14:53 Downloads
drwxr-xr-x. 2 ruib ruib 4096 Aug 15 19:45 Pictures
drwxr-xr-x. 3 ruib ruib 4096 Jul 28 16:36 Programs
drwxr-xr-x. 2 ruib ruib 4096 Sep 18 12:57 Public
drwxr-xr-x. 7 ruib ruib 4096 Aug 4 10:16 rpmbuild
drwxr-xr-x. 6 ruib ruib 4096 Aug 30 21:24 Teaching
drwxr-xr-x. 2 ruib ruib 4096 Jul 26 02:54 Templates
drwxr-xr-x. 2 ruib ruib 4096 Jul 26 02:54 Videos
[ruib@ruib-laptop1 ~]$ █
```

# Interface Utilizador-SO

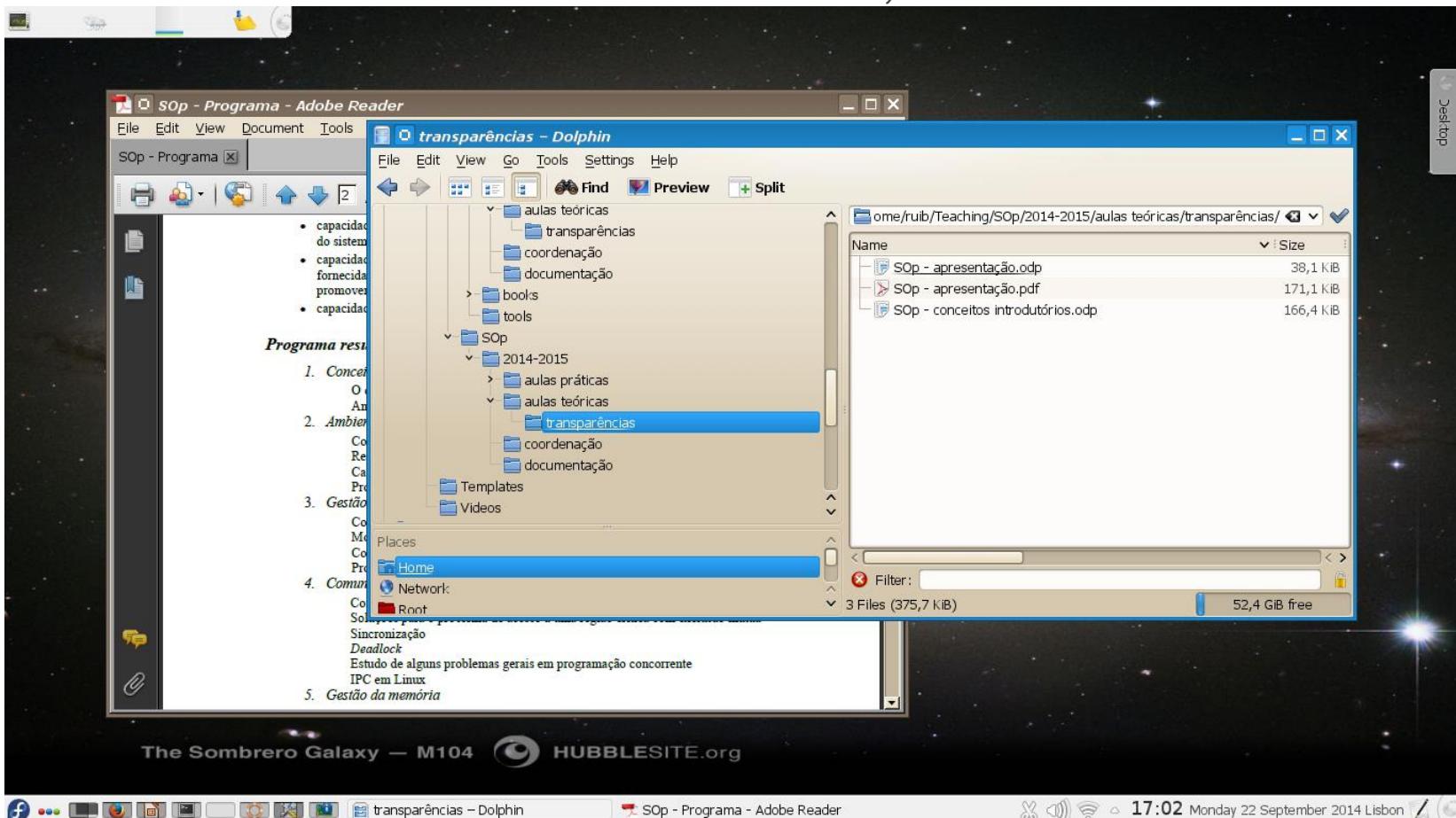
- Interpretador de comandos (CLI)
  - Pode estar incluído no kernel ou funcionar como um programa de sistema  
*Unix* ↑
  - Se programa independente designado de *shell*
    - *Bourne shell, C shell, Korn shell, etc*      *Bash* → Bourne Again Shell
  - **Função principal:** ler e executar comandos do utilizador      ↗ Alguns associados com a gestão de ficheiros
  - 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      { Própria shell fog...  
⇒ Ajuda externa... (clicar processos novos)
  - Comandos internos e externos
    - Comando *type / cd / history*
  - No caso de serem programas independentes a *shell* não necessita de perceber quais os efeitos do comando executado  
*Nem sabe se vai ou não terminar*  
*ex: \$ matlab*

# Interface Utilizador-SO

- Interface gráfica (GUI)

Com base em janelas!

Em linux é um pouco mais complicado, mas é o SO



# 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”  
⇒ Tempo de uma secretária ...
- A primeira interface gráfica apareceu em 1973, tendo sido desenvolvida no laboratório Xerox PARC (Silicon Valley)  
→ + do que uma impressora de impressores ...
- 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)  
↳ Muito "baixo nível"

# Sistema de Ficheiros FAT32

File Allocation Table

Nota: Um ficheiro ocupa K clusters, K intelecto!

→ Um cluster pode ter mais que um setor.

- Organização do disco

- Boot sector (boot loader, partition table, BIOS parameter block, etc.)  
onde é guardado o endereço
- 2 cópias da FAT  $\Rightarrow$  Serve para encontrar os ficheiros  
*Mais resiliente, é critico perder. ~ Perdemos tudo do disco...*
- Zona de dados (directória raiz no início da zona de dados)

- Como encontrar conteúdo de um ficheiro?

- Cluster inicial indicado na entrada de directória

Conjunto de blocos contidos no disco

- Clusters seguintes numa lista ligada através da FAT

- FAT

• O nome do ficheiro está na directória

- Array de números de clusters (cada um com 32 bits)

- Para cada cluster indica:

- Cluster seguinte; Final ou Livre

Guardados através de uma lista ligada da FAT

(\*)

File Location Table

Fich1: 5

Fich2: 8, 4

2 clusters

 $\Rightarrow$  Hair: 62 Kbytes

Continua no Cluster 4

Vazio

... 8

9: 0x4

10: 0x00000000

11: 0x00000000

12: 0x00000000

13: 0x00000000

14: 0x00000000

15: 0x00000000

16: 0x00000000

17: 0x00000000

18: 0x00000000

19: 0x00000000

20: 0x00000000

21: 0x00000000

22: 0x00000000

23: 0x00000000

24: 0x00000000

25: 0x00000000

26: 0x00000000

27: 0x00000000

28: 0x00000000

29: 0x00000000

30: 0x00000000

31: 0x00000000

32: 0x00000000

33: 0x00000000

34: 0x00000000

35: 0x00000000

36: 0x00000000

37: 0x00000000

38: 0x00000000

39: 0x00000000

40: 0x00000000

41: 0x00000000

42: 0x00000000

43: 0x00000000

44: 0x00000000

45: 0x00000000

46: 0x00000000

47: 0x00000000

48: 0x00000000

49: 0x00000000

50: 0x00000000

51: 0x00000000

52: 0x00000000

53: 0x00000000

54: 0x00000000

55: 0x00000000

56: 0x00000000

57: 0x00000000

58: 0x00000000

59: 0x00000000

60: 0x00000000

61: 0x00000000

62: 0x00000000

63: 0x00000000

64: 0x00000000

65: 0x00000000

66: 0x00000000

67: 0x00000000

68: 0x00000000

69: 0x00000000

70: 0x00000000

71: 0x00000000

72: 0x00000000

73: 0x00000000

74: 0x00000000

75: 0x00000000

76: 0x00000000

77: 0x00000000

78: 0x00000000

79: 0x00000000

80: 0x00000000

81: 0x00000000

82: 0x00000000

83: 0x00000000

84: 0x00000000

85: 0x00000000

86: 0x00000000

87: 0x00000000

88: 0x00000000

89: 0x00000000

90: 0x00000000

91: 0x00000000

92: 0x00000000

93: 0x00000000

94: 0x00000000

95: 0x00000000

96: 0x00000000

97: 0x00000000

98: 0x00000000

99: 0x00000000

100: 0x00000000

101: 0x00000000

102: 0x00000000

103: 0x00000000

104: 0x00000000

105: 0x00000000

106: 0x00000000

107: 0x00000000

108: 0x00000000

109: 0x00000000

110: 0x00000000

111: 0x00000000

112: 0x00000000

113: 0x00000000

114: 0x00000000

115: 0x00000000

116: 0x00000000

117: 0x00000000

118: 0x00000000

119: 0x00000000

120: 0x00000000

121: 0x00000000

122: 0x00000000

123: 0x00000000

124: 0x00000000

125: 0x00000000

126: 0x00000000

127: 0x00000000

128: 0x00000000

129: 0x00000000

130: 0x00000000

131: 0x00000000

132: 0x00000000

133: 0x00000000

134: 0x00000000

135: 0x00000000

136: 0x00000000

137: 0x00000000

138: 0x00000000

139: 0x00000000

140: 0x00000000

141: 0x00000000

142: 0x00000000

143: 0x00000000

144: 0x00000000

145: 0x00000000

146: 0x00000000

147: 0x00000000

148: 0x00000000

149: 0x00000000

150: 0x00000000

151: 0x00000000

152: 0x00000000

153: 0x00000000

154: 0x00000000

155: 0x00000000

156: 0x00000000

157: 0x00000000

158: 0x00000000

159: 0x00000000

160: 0x00000000

161: 0x00000000

162: 0x00000000

163: 0x00000000

164: 0x00000000

165: 0x00000000

166: 0x00000000

167: 0x00000000

168: 0x00000000

169: 0x00000000

170: 0x00000000

171: 0x00000000

172: 0x00000000

173: 0x00000000

174: 0x00000000

175: 0x00000000

176: 0x00000000

177: 0x00000000

178: 0x00000000

179: 0x00000000

180: 0x00000000

181: 0x00000000

182: 0x00000000

183: 0x00000000

184: 0x00000000

185: 0x00000000

186: 0x00000000

187: 0x00000000

188: 0x00000000

189: 0x00000000

190: 0x00000000

191: 0x00000000

192: 0x00000000

193: 0x00000000

194: 0x00000000

195: 0x00000000

196: 0x00000000

197: 0x00000000

198: 0x00000000

199: 0x00000000

200: 0x00000000

201: 0x00000000

202: 0x00000000

203: 0x00000000

204: 0x00000000

205: 0x00000000

206: 0x00000000

207: 0x00000000

208: 0x00000000

209: 0x00000000

210: 0x00000000

211: 0x00000000

212: 0x00000000

213: 0x00000000

214: 0x00000000

215: 0x00000000

216: 0x00000000

217: 0x00000000

218: 0x00000000

219: 0x00000000

220: 0x00000000

221: 0x00000000

222: 0x00000000

223: 0x00000000

224: 0x00000000

225: 0x00000000

226: 0x00000000

227: 0x00000000

228: 0x00000000

229: 0x00000000

230: 0x00000000

231: 0x00000000

232: 0x00000000

233: 0x00000000

234: 0x00000000

235: 0x00000000

236: 0x00000000

237: 0x00000000

238: 0x00000000

239: 0x00000000

240: 0x00000000

241: 0x00000000

242: 0x00000000

243: 0x00000000

244: 0x00000000

245: 0x00000000

246: 0x00000000

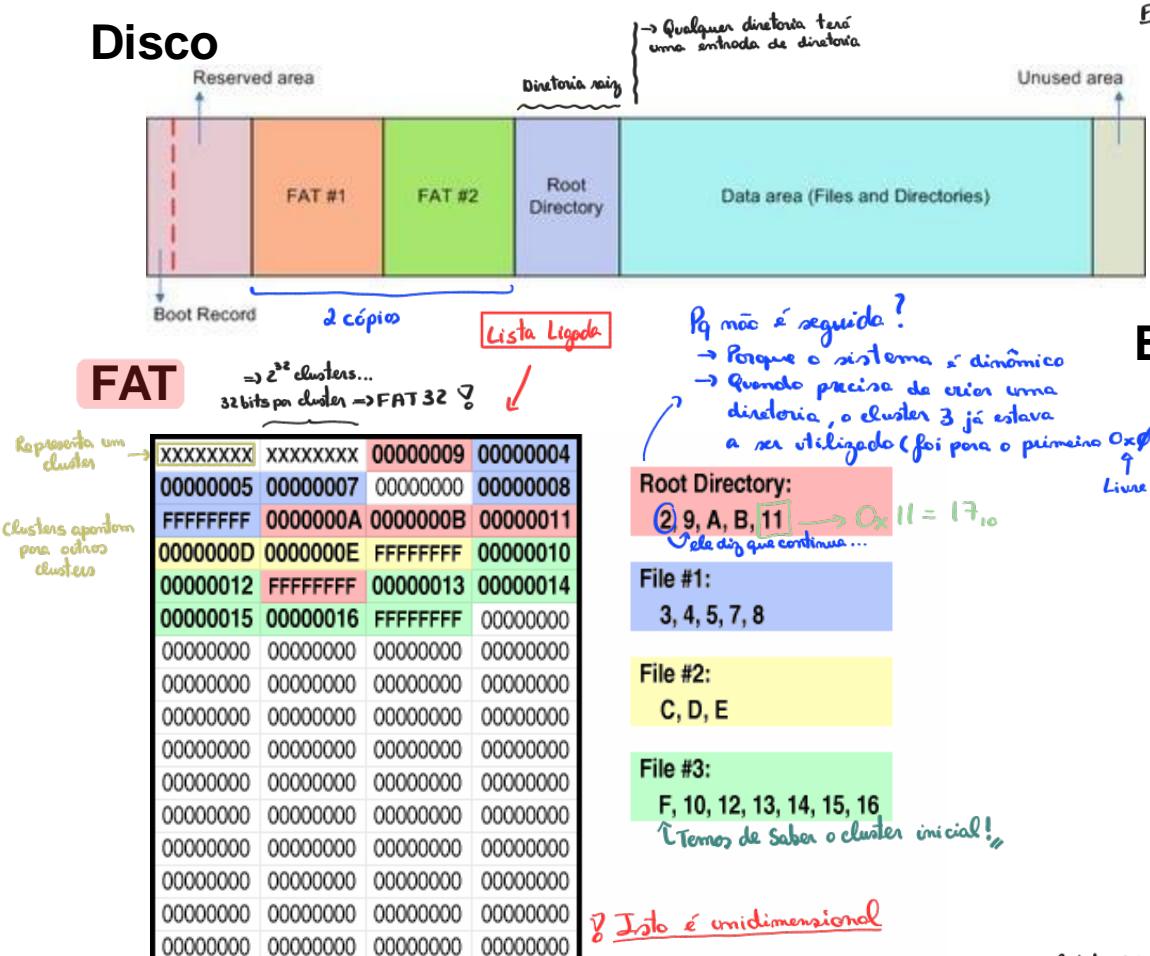
247: 0x00000000

248: 0x00000000

# FAT32

- Sistema de Ficheiros FAT32

**Disco** { → Qualquer diretório terá uma estrutura de diretórios.



mais é seguida?

- Porque o sistema é dinâmico
  - Quando precisa de criar uma diretoria, o elástico Z já estava a ser utilizado (foi para o prime

## **Root Directory:**

② 9, A, B, 11 → O<sub>x</sub> II = 17<sub>10</sub>  
Tele dir que continua ...

File #1:

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

### **File #3:**

F, 10, 12, 13, 14, 15, 16

↑ Temos de Saber o cluster inicial!,,

? Isso é unidimensional

DETI/UA

↑ Disco é dividido em clusters,

2.9.

- tomorrow do SF = 2GB = 2.147.483.648 B

- terms  $2^{16}$  clusters

$$\Rightarrow \text{Tomonth per cluster} = \frac{2 \times 2^{30}}{16} = \underline{\underline{32768}} \text{ bytes}$$

FAT Assimindo clusters com 1024B  
Cluster Id próximo/ultimo/livre cluster

0	XXXX
1	XXXX
2	9
3	4
4	5
5	7
6	0
7	8
8	FFFF
9	a
10	b
11	11
12	d
	:
13	FFFF

- 0 byte 1023 da raiz está  
no cluster 2, ( $0 \rightarrow 1023$ )
  - 0 byte 1024 da raiz está  
no cluster 9, ( $1023 \rightarrow 2097$ )

Qualquer ficheiro teria pelo menos ~32KB de espaço ... (problemático)

→ Qual o cluster onde estiver o byte 3000 do File?

R: cluster 5,  
 $\lfloor 3000/1024 \rfloor = 2$   
 $\lfloor 1023/1024 \rfloor = 0$   
 $\lfloor 1024/1024 \rfloor = 1$

3	4	5
0 byte	3000	952 do

$3000 \times 1024 = 952$  byte 3000 está  
 $1023 \times 1024 = 1023$  no byte 952 do  
 $1024 \times 1024 = 1$  cluster 5,,

## **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)

Pode ter 32 entradas por cluster

32 bytes

Aqui guardavamos o primeiro cluster

File 1: 0x00 00 00 03

bytes 20-21: 0x00 00

bytes 26-27: 0x00 03  
~~~~~ byte menos significativo do primeiro cluster

→ Para eliminar o File 1 basta mudar o byte 0 da entrada de diretório para 0xE5 e na FAT é meter os respetivos clusters a 0x00000000 livre

→ Para criar o File 2 com 1500 bytes: precisamos de 2 clusters!

→ Root Directory: escolher uma entrada livre (allocated/deleted)  
→ preencher os atributos...  
→ e vimos na FAT o primeiro cluster vazio (0x00000000) e colocámos o na entrada de diretório o cluster id.

→ FAT: No primeiro cluster colocámos o cluster id do segundo cluster e nesse o (0xFFFFFFF)

File 4: 6, 0x17  
base16

|      |      |
|------|------|
| 0x17 | FFFF |
| 6    | 23   |

→ Disco: somos ao cluster 6 e colocámos os primeiros 1024 bytes e ao cluster 23 os outros 476 bytes (0 → 1023) (1024 → 1499)