

# Workshop

## Introdução ao Linux



NEEEEC-FEUP



24/10/2020

Orador: André Campanhã

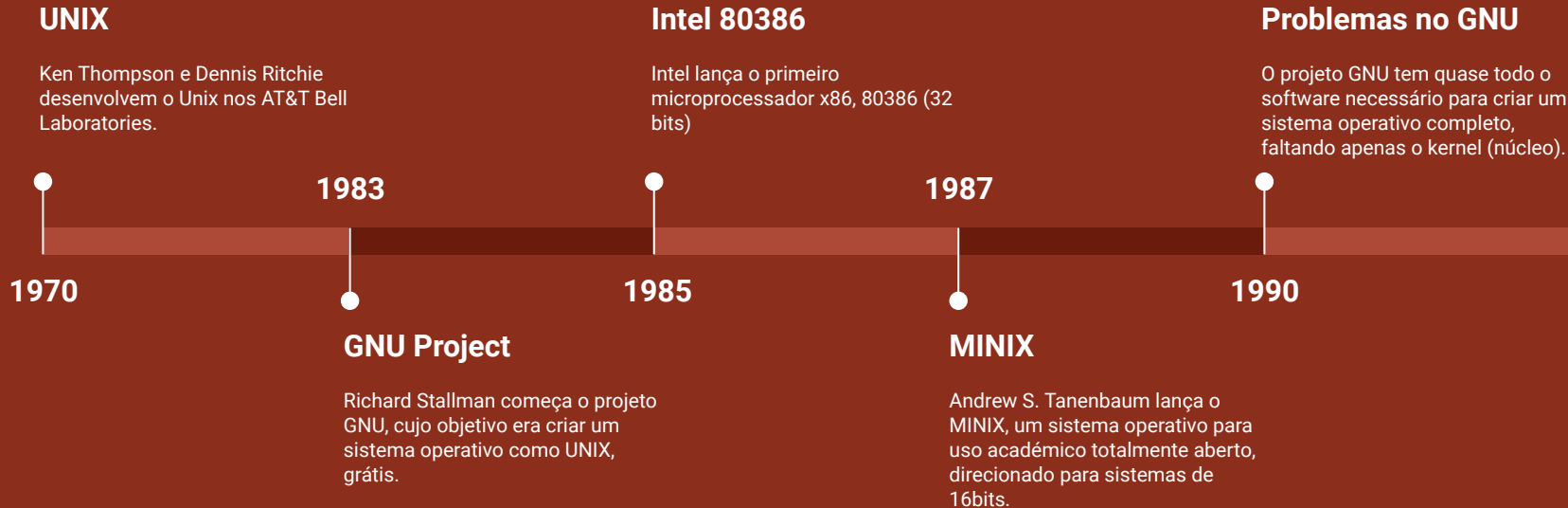
# Sistema operativo

Um sistema operativo (SO) é uma camada de *software* que fica entre o *hardware* e o *software* que o utilizador executa (aplicações).

Fundamentalmente, o SO permite que as aplicações comuniquem com o hardware de forma abstrata.



# Um pouco de história...



# 1991 - Linux



Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).

I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus ([torvalds@kruuna.helsinki.fi](mailto:torvalds@kruuna.helsinki.fi))

PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT portable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-).

— Linus Torvalds

# Linux



Linux é um kernel (núcleo). Juntamente com as aplicações do projeto GNU, forma o sistema operativo GNU/Linux, vulgarmente chamado apenas Linux.

A arquitetura é baseada no SO Unix, em que tudo é ou um ficheiro ou um processo, mantendo todos os seus componentes e programas modelares e capazes de se interligarem.

# Outline

1. Navegação
2. Trabalhar com ficheiros
3. Wildcards
4. Permissões
5. Filtros e Expressões Regulares
6. *Piping* e Redirecionamento
7. SSH

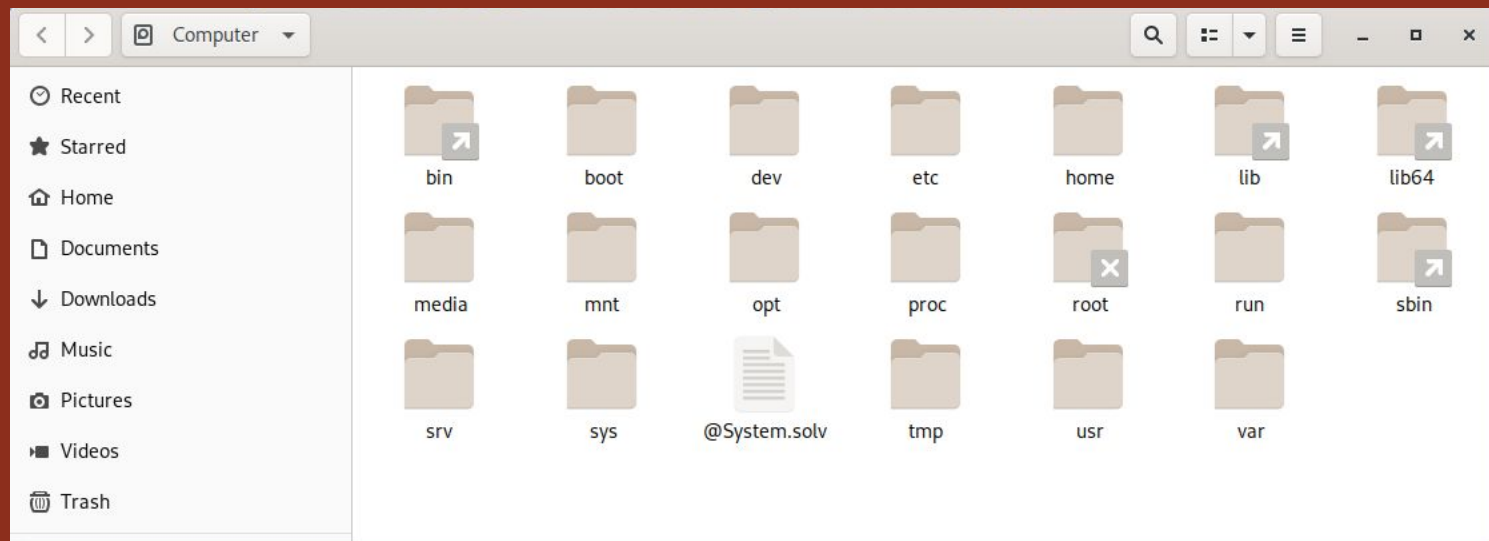
# Navegação



## Sistema de ficheiros

É **hierárquico** (em árvore). No topo existe a pasta raiz (ou **root**), denotada por uma barra “/”. Tem subpastas, que por sua vez podem ter mais subpastas, etc.

Cada uma destas pasta pode ter **ficheiros**.



Exemplo pasta **root**



# Navegação

## Caminho (*Path*)

Localização de um ficheiro ou pasta na árvore do sistema de ficheiros do sistema operativo.

Ex.: /home/andre/Documents/workshop.png

Existem dois tipos: absolutos - caminho a partir da pasta **root**, começa por / - e relativos - caminho a partir da pasta de atual.

# Navegação

## Onde estamos? (Pasta atual)

```
$ pwd
```

print working directory - imprimir a pasta atual

## O que está na localização atual?

```
$ ls [opções] [localização]
```

print working directory - imprimir a pasta atual

# Navegação

## Mudar localização

```
$ cd [localização]
```

change **d**irectory- mudar localização atual

# Navegação

## Caminhos

~ (til) - Atalho para a pasta do utilizador (normalmente /home/<utilizador>)

Ex.: ~/Documents -> /home/andre/Documents

. (ponto) - Referente à pasta atual. Apenas preciso em algumas circunstâncias.

.. (ponto ponto) - Referente à pasta “pai” da atual.

Ex.: (pasta atual /home/andre/Documents)

../Downloads -> /home/andre/Downloads

# Trabalhar com ficheiros

## Tudo é um ficheiro

Fundamentalmente, tudo no Linux é um ficheiro. Um ficheiro de texto é um ficheiro, uma pasta é um ficheiro, o teclado é um ficheiro (apenas de leitura), o monitor é um ficheiro (apenas de escrita), etc..

Os ficheiros são sensíveis a maiúsculas e minúsculas.

# Trabalhar com ficheiros

## Criar uma pasta

```
$ mkdir [opções] <pasta>
```

**make** **directory**

## Remover uma pasta

```
$ rmdir [opções] <pasta>
```

**remove** **directory** - apenas se a pasta estiver vazia

# Trabalhar com ficheiros



## Criar um ficheiro em branco

```
$ touch [opções] <nome>
```

# Trabalhar com ficheiros

## Copiar um ficheiro ou pasta

```
$ cp [opções] <origem> <destino>
```

Caso o destino seja uma pasta, copia o ficheiro de origem com o mesmo nome e coloca-o na pasta de destino. Caso seja um ficheiro, coloca-o no destino com o nome indicado.

Normalmente apenas copia um ficheiro. A opção -r copia recursivamente, permitindo copiar uma pasta inteira, incluindo os ficheiros e subpastas.



# Trabalhar com ficheiros

## Mover um ficheiro ou pasta

```
$ mv [opções] <origem> <destino>
```

Funciona de forma parecida ao **cp**.

Este comando pode ser também utilizado para renomear ficheiros ou pastas.

Ex.: `$ mv imagem.png img.png`

`$ mv subpasta pastarenomeada`

# Trabalhar com ficheiros

## Remover um ficheiro (e pastas não vazias)

```
$ rm [opções] <ficheiro>
```

Utilizando a opção -r (recursiva) e apagando uma pasta, o comando irá apagar todos os ficheiros, subpastas e ficheiros dentro dessas subpastas dentro.



NEEEEC-FEUP

# Trabalhar com ficheiros

## Editar ficheiros na linha de comando

GNU Nano, um editor de texto bastante simples, porém limitado nas suas funcionalidades.

Editores mais complexos (e muito mais ricos): emacs, vi, ...

```
GNU nano 4.3          New Buffer

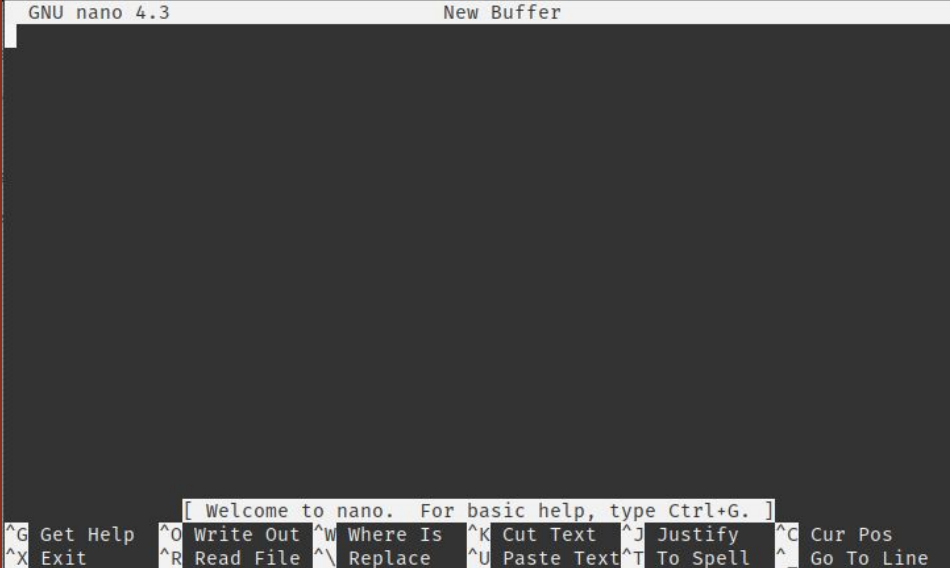
[ Welcome to nano.  For basic help, type Ctrl+G. ]
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace  ^U Paste Text ^T To Spell  _ Go To Line
```

# Trabalhar com ficheiros

## GNU nano

Barra inferior com atalhos (^ -> Ctrl).

Para ver a ajuda, carregar em Ctrl + G.



```
GNU nano 4.3          New Buffer

[ Welcome to nano.  For basic help, type Ctrl+G. ]
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace   ^U Paste Text ^T To Spell  _ Go To Line
```

# Trabalhar com ficheiros

## Visualizar conteúdo de um ficheiro

```
$ cat [ficheiro]
```

Imprime para a linha de comando o conteúdo do ficheiro.

Caso não seja fornecido um ficheiro, irá ler do STDIN (Standard Input), útil em conjunto com piping e redirecionamento (aprofundado mais à frente)

# Trabalhar com ficheiros

## Visualizar conteúdo de um ficheiro (ficheiros maiores)

```
$ less <ficheiro>
```

O cat é bom para ficheiros pequenos, mas quando o ficheiro é maior, pode ultrapassar o buffer do terminal, não sendo possível visualizar o conteúdo todo.

O less permite andar para cima e para baixo no ficheiro com as **setas**, **barra de espaços** para avançar uma página, **b** para voltar uma página e **q** para sair.

# Wildcards

São um conjunto de componentes que podem ser utilizados para criar um padrão que define um conjunto de ficheiros ou pastas.

Quando é referido um caminho, podem ser usadas **wildcards** para o transformar num conjunto.

- \* - representa zero ou mais caracteres
- ? - representa um único carácter
- [ ] - representa um intervalo de caracteres

# Wildcards

## Exemplo

```
user@bash:$ ls  
andre blah.txt campanha exemplo.txt foo1 foo2
```

```
user@bash:$ ls f*  
foo1 foo2
```



# Wildcards



Ao contrário do que parece, é o bash (programa que fornece a interface de linha de comando) que faz a tradução da *wildcard* **b\***.

Quando é executado o comando **ls f\***, o bash traduz para **ls foo1 foo2** e executa o programa. **O programa nunca recebe as *wildcards*.**

# Wildcards

## Mais alguns exemplos

```
user@bash:$ ls  
andre  campanha  first foo2    video.mp4    blah.txt  exemplo.txt  foo1
```

```
user@bash:$ ls *.???  
blah.txt  exemplo.txt  video.mp4
```

```
user@bash:$ ls [fv]*  
first foo1 foo2 video.mp4
```

# Wildcards



## Mais alguns exemplos

```
user@bash:$ ls  
andre  campanha  first foo2    video.mp4    blah.txt exemplo.txt foo1
```

```
user@bash:$ ls *[0-9]  
foo1 foo2 video.mp4
```

```
user@bash:$ ls [^a-d]* # ^ inverte o intervalo  
exemplo.txt first foo1 foo2 video.mp4
```

# Permissões

Especificam o que cada utilizador pode ou não fazer no que diz respeito a um ficheiro ou pasta. São importantes para criar um ambiente com segurança.

As permissões ditam 3 ações possíveis:

- **r** read - ver conteúdo do ficheiro
- **w** write - modificar o ficheiro
- **x** execute - ficheiro: executar ou correr (caso seja um programa ou script);  
pasta: entrar dentro da pasta (**cd**)

# Permissões

Para cada ficheiro são definidos 3 conjuntos de pessoas a quem são definidas permissões:

- **owner (u)** - um utilizador único a quem o ficheiro pertence
- **group (g)** - o grupo de utilizadores a que o ficheiro pertence
- **others (o)** - todos os outros utilizadores

# Permissões

## Ver as permissões

```
user@bash:$ ls -l <ficheiro>  
-rwxr--r-- 1 user user 0 nov 24 15:49 <ficheiro>
```

Os primeiros 10 caracteres identificam as permissões do ficheiro

- 1º carácter, traço (-) se ficheiro, **d** se pasta.
- 3 caracteres seguintes: Permissões do dono (**owner**). A letra representa a presença da permissão, o traço (-) a ausência, na ordem leitura (**r**), escrita (**w**) e executar (**x**).
- 3 caracteres seguintes: Permissões do grupo (**group**).
- 3 últimos: Permissões dos outros utilizadores (**others**).

# Permissões

## Mudar permissões

```
$ chmod <permissões> <caminho>
```

O argumento de permissões é constituído por 3 elementos:

- Quem [ugoa] - user (ou owner), group, others, all
- Dar ou retirar permissão - mais (+) ou menos (-)
- A permissão a ser definida - read (r), write (w), execute (x)

# Permissões

## Exemplo chmod

```
user@bash:$ ls -l foo1
-rw-r--r-- 1 user user 0 dez  2 15:49 foo1
user@bash:$ chmod g+x foo1
user@bash:$ ls -l foo1
-rw-r-xr-- 1 user user 0 dez  2 15:49 foo1
user@bash:$ chmod go-rx foo1
user@bash:$ ls -l foo1
-rw----- 1 user user 0 dez  2 15:49 foo1
```



# Permissões

## Utilizador root

No Linux apenas dois utilizadores podem alterar as permissões de um ficheiro: o dono do ficheiro e o utilizador **root**.

É um superutilizador, que tem permissões totais sobre todo o computador.

Se o utilizador for administrador de sistema, pode executar qualquer comando como root, precedendo o comando com sudo (super user do):

```
$ sudo <comando>
```

# Filtros e Expressões Regulares

## Imprimir as primeiras linhas

```
$ head [-número de linhas] [caminho]
```

Por predefinição imprime as 10 primeiras linhas

## Imprimir as últimas linhas

```
$ tail [-número de linhas] [caminho]
```

Por predefinição imprime as 10 últimas linhas

# Filtros e Expressões Regulares

## Ordenar um ficheiro

```
$ sort [-opções] [caminho]
```

Por predefinição ordena alfabeticamente

## Filtrar conteúdo organizado em colunas

```
$ cut [-opções] [caminho]
```

Ex.: `cut -f 1,2 -d ' ' mysampleddata.txt`

Apenas imprime as colunas 1 e 2

# Filtros e Expressões Regulares

## Pesquisar e substituir

```
$ sed <expressão> [caminho]
```

Uma expressão básica tem o seguinte formato: **s/a\_pesquisar/a\_substituir/g**

Ex.: sed 's/maçã/pêra/g' -> substitui todas as ocorrências de maçã por pêra.

O termo de pesquisa é na verdade algo chamado expressão regular, que define um padrão (parecido com as *wildcards*).

# Filtros e Expressões Regulares

## Pesquisar e imprimir

```
$ egrep <expressão> [caminho]
```

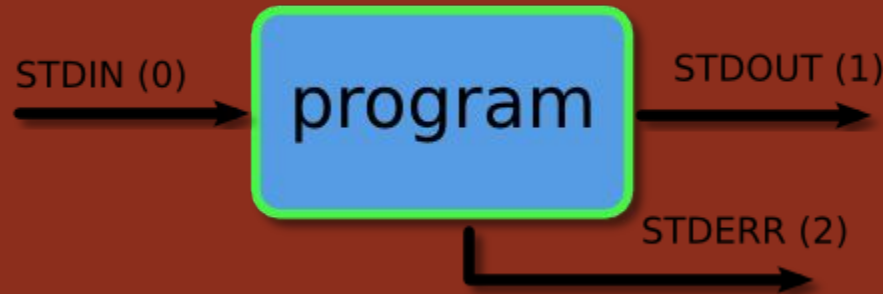
Ex.: egrep 'mellon' mysampledada.txt

# Piping e Redirecionamento

**STDIN (0)** - Standard input (dados fornecidos ao programa)

**STDOUT (1)** - Standard output (dados imprimidos pelo programa, por predefinição para a linha de comandos)

**STDERR (2)** - Standard error (mensagens de erro, por predefinição imprimidos na linha de comandos)



# Piping e Redirecionamento

## Redirecionar para um ficheiro

```
user@bash:$ cat fruta.txt
```

```
Banana Laranja Tomate Maçã Pêra Uva Pêssego Limão Lima Ananás Abacate
```

```
user@bash:$ sed 's/Banana/Kiwi/g' fruta.txt > fruta2.txt
```

```
user@bash:$ cat fruta2.txt
```

```
Kiwi Laranja Tomate Maçã Pêra Uva Pêssego Limão Lima Ananás Abacate
```

# Piping e Redirecionamento

## Redirecionar de ficheiro

```
user@bash:$ cat fruta.txt
```

```
Banana Laranja Tomate Maçã Pêra Uva Pêssego Limão Lima Ananás Abacate
```

```
user@bash:$ cat < fruta.txt
```

```
Banana Laranja Tomate Maçã Pêra Uva Pêssego Limão Lima Ananás Abacate
```

Útil quando o programa não aceita um ficheiro como parâmetro



# Piping e Redirecionamento

## Piping

```
user@bash:$ cat fruta.txt
```

Banana

Laranja

Tomate

Maçã

```
user@bash:$ cat fruta.txt | head -2 | tail -1
```

Laranja

# SSH



## Secure Shell

Protocolo de rede criptográfico para operar serviços de rede seguramente em redes não seguras.

Qualquer serviço pode ser protegido com SSH.

Aplicações usuais: Linha de comandos remota, execução de comandos remotamente.

# SSH

## Ligar a servidor remoto

```
$ ssh <utilizador>@<servidor>
```



**NEEEC-FEUP**