



### Docentes

João Paulo Barraca <jpbarraca@ua.pt>,

André Zúquete <andre.zuquete@ua.pt>, Bernardo Cunha <mbc@ua.pt>

# TEMA 2

## Introdução ao Ambiente Linux

### Objetivos:

- Introdução ao Linux
- Interpretador de comandos
- Sistema de ficheiros
- Utilizadores
- Gestão de aplicações
- Processos
- Acesso Remoto
- Bash Scripting

## 2.1 Introdução

### 2.1.1 Distribuição Lubuntu

Neste tema será utilizada uma distribuição denominada por *Lubuntu*. Tal como referido em [1]: O *Lubuntu* é uma variedade da distribuição *Ubuntu*, utilizando o ambiente de trabalho Lightweight X11 Desktop Environment (LXDE)[2] como o seu ambiente principal. O objectivo é fornecer uma distribuição muito leve, mantendo todas as vantagens do mundo *Ubuntu* (repositórios, suporte, etc.). O *Lubuntu* tem como alvo utilizadores de computadores pessoais “normais” e portáteis com hardware de baixos recursos. Estes utilizadores podem não saber utilizar a linha de comandos, e na maioria dos casos simplesmente não têm recursos suficientes para todos os efeitos da distribuição completa.

Neste caso, ela será utilizada pois utiliza a mesma estrutura da distribuição *Ubuntu*, mas com requisitos de recursos compatíveis com o ambiente disponível nas aulas.

De forma a acelerar a utilização, é fornecida uma imagem virtual com o *Lubuntu* pré-instalado, que pode ser encontrada na página da disciplina.

### 2.1.2 Criação de uma máquina virtual

Os passos para criar uma máquina virtual vão ser seguidamente indicados, acompanhados de capturas de ecrã exemplificativas:

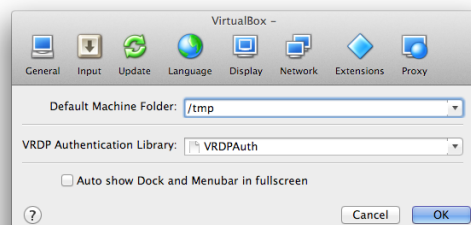
Inicie a execução do *VirtualBox*. Deverá surgir no ecrã uma janela como a indicada à direita. Antes de criar qualquer máquina virtual altere a definição que o *VirtualBox* possui quanto ao local onde guarda dados relativos s máquinas virtuais. Para tal, selecione **Ficheiro** e no menu selecione **Preferências...** Este menu encontra-se disponível se aproximar o ponteiro do rato do canto superior esquerdo do ecrã.



Considere apenas as definições associadas à classe **Geral**. Se estiver a trabalhar num Computador Pessoal (PC) do laboratório altere a definição de **Pasta pré-definida das Máquinas** para **/tmp**; se estiver a trabalhar no seu PC, altere para o local que considerar mais conveniente.

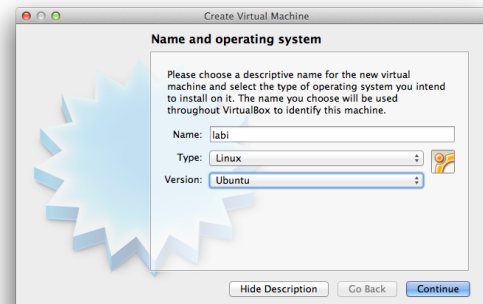
Devido a um problema com esta versão específica do *VirtualBox*, terá de ir às definições de proxy, activar a configuração, escrever algo para os 2 campos pedidos, e seguidamente desactivar a configuração de proxy.

Feita a alteração, feche a janela das definições.

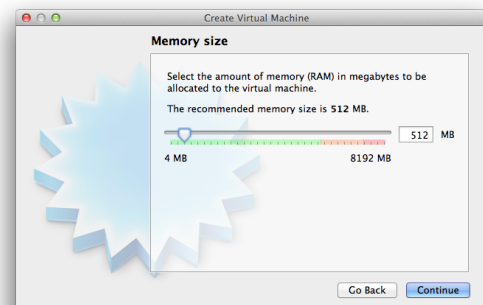


Selecione o botão **Novo** para indicar que deseja criar uma nova máquina virtual. Na janela seguinte selecione o botão **Avançar** (ou **Next**).

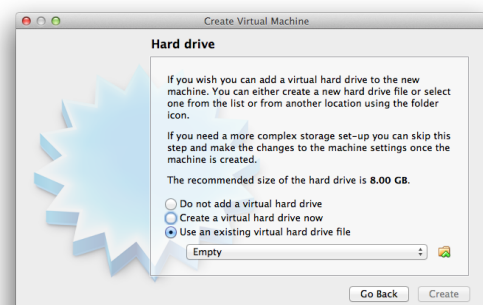
Escolha um nome para identificar a máquina virtual na lista de máquinas virtuais conhecidas localmente pelo *VirtualBox* (ex, labi). quanto ao tipo de sistema operativo, escolha *Linux* e *Ubuntu* (distribuição).



Indique a quantidade de memória Random Access Memory (RAM) de que disporá a máquina virtual. Escolha 512 Mibi Byte (MiB). Não escolha mais porque não será necessário e porque quanto mais escolher, menos memória terá o sistema hospedeiro. Esta configuração poderá ser alterada mais tarde, não é irreversível.



Indique que pretende arrancar de um disco rígido e que pretende utilizar um disco rígido (virtual) existente. Se estiver no do laboratório, descomprima (**bunzip2**) o disco que existe em `/usr/local/labi/Labi.vdi.bz2` para `/tmp` e utilize-o. Se estiver em casa, obtenha o disco a partir da página da disciplina, e utilize-o. Em qualquer um dos casos, ao adicionar o disco, selecione a opção *Use Host I/O Cache* no controlador do disco. Isto irá tornar o sistema bastante mais rápido.



Neste momento já foi recolhida toda a informação necessária para criar a máquina virtual com um disco virtual de suporte. Ao contrário da aula anterior, este disco já contém uma instalação de *Linux* pronta a utilizar. Deve-se salientar que, por motivos de desempenho das aulas, a quantidade de aplicações pré instalada é muito reduzida.

### 2.1.3 Arranque de uma máquina virtual

O arranque de uma máquina virtual é em tudo semelhante a uma máquina real, há uma passagem por um ponto onde é executada uma BIOS (virtual), onde se selecciona um dispositivo virtual de arranque (neste caso vai ser o Disco Virtual) e se carrega o mesmo para executar.

Após iniciar a máquina que acabou de criar, deverá ser-lhe apresentado uma janela a pedir as credenciais de acesso, tal como mostrado na Figura 2.1. O utilizador e a palavra passe são **labi**.

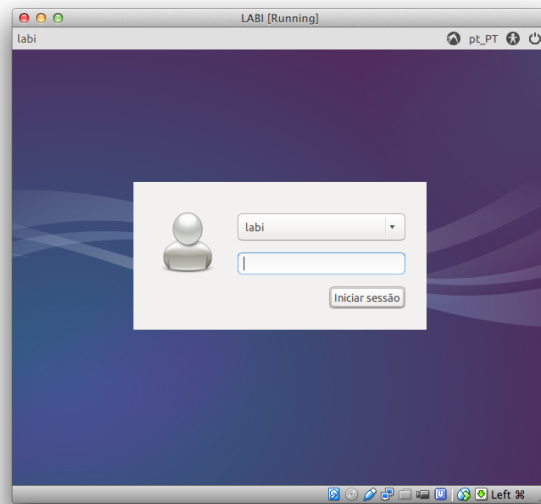


Figura 2.1: Pedido de entrada de credenciais *Linux*

O processo deverá depois prosseguir para um ambiente de trabalho com um menu de aplicações ao fundo do ecrã. No canto inferior esquerdo pode aceder às aplicações existentes. Não são muitas, mas poderá explorar algumas e verificar que são em tudo semelhantes a aplicações que conhece noutros sistemas operativos.

Este guião irá focar-se na utilização da consola. Não porque seja a única maneira de interagir com o sistemas, mas porque é o mais poderoso, permitindo aceder e manipular praticamente toda a informação existente no sistema.

## 2.2 A Linha de Comandos UNIX

Quando o sistema *UNIX* foi concebido, os computadores eram controlados essencialmente através de *consolas* ou *terminais* de texto: dispositivos dotados de um teclado e de um ecrã onde se podia visualizar somente texto. A interação com o sistema fazia-se tipicamente através da introdução de comandos escritos no teclado e da observação da resposta produzida no ecrã pelos programas executados.

Atualmente existem ambientes gráficos que correm sobre o *UNIX/Linux* e permitem visualizar informação de texto e gráfica, e interagir por manipulação virtual de objetos gráficos recorrendo a um rato e ao teclado. É o caso do Sistema de Janelas *X*, ou simplesmente *X* <sup>1</sup>, que está instalado nos PCs das salas de aula.

Apesar das novas formas de interação proporcionadas pelos ambientes gráficos, continua a ser possível e em certos casos preferível usar a interface de *linha de comandos* para muitas operações. No *X*, isto pode fazer-se usando um *emulador de terminal*, um programa que abre uma janela onde se podem introduzir comandos linha-a-linha e observar as respostas geradas tal como num terminal de texto à moda antiga.

### 2.2.1 Interfaces de texto e gráficas

Os sistemas *Linux* (tipicamente) simulam um ambiente de trabalho formado por 6 interfaces de texto (consolas) e 2 interfaces gráficas. Na prática todas estas interfaces coexistem sobre os mesmos equipamentos de interface com os utilizadores: ecrã, teclado, rato, etc. Muito embora estas interfaces estejam sempre ativas, elas não estão sempre visíveis: só uma delas está visível em cada instante.

Na maior parte dos casos os sistemas *Linux* apresentam uma interface gráfica quando iniciam, ou após a terminação da sessão de um utilizador. A comutação entre interfaces faz-se através das seguintes combinações de teclas:

**interface gráfica → consola** : Ctrl + Alt + F<sub>*n*</sub>, onde F<sub>*n*</sub> é uma das teclas F1, F2, ..., F6. O valor de *n* indica o número da consola que se pretende usar (F1 para a consola 1, etc.).

**consola → interface gráfica** : Alt + F7 (para a interface gráfica por omissão) ou Alt + F8 (para a interface gráfica secundária, normalmente inativa).

**consola → consola** : Alt + F<sub>*n*</sub>, onde F<sub>*n*</sub> é uma vez mais F1, F2, ..., F6.

**Importante!** Se executar este guião num anfitrião *Linux*, esta sequência de teclas irá alternar as sessões no anfitrião e não no convidado!

Após o arranque do *Linux*, cada consola apresenta uma interface muito simples de *login*, na qual são apresentadas algumas características do sistema (sistema operativo,

---

<sup>1</sup><http://www.x.org>

nome da máquina, nome da interface ( $tty_n$ ), e o que se pretende que o utilizador introduza: o *nome-de-utilizador* e a *senha*. Após um *login* bem sucedido, o utilizador pode continuar a trabalhar na linha de comandos que lhe é apresentada. Para terminar a sua sessão, o utilizador deverá fazer *logout* usando o comando **exit**. Após este comando, a consola voltará a apresentar a interface de *login* antes observada.

---

### Exercício 2.1

Mude para uma consola de texto, faça *login* na mesma, execute o comando **whoami**, e execute em seguida o comando **exit**. Repita o processo de *login* e faça desta vez *logout* carregando na conjunto de teclas Ctrl + D. Este conjunto de teclas é designado por EOF (*End-Of-File*).

Uma das consolas terá o ambiente gráfico ao qual pode voltar.

---

### 2.2.2 Interpretador de comandos

A maioria das distribuições *Linux* fornece uma Command Line Interface (CLI) através do programa **bash** <sup>2</sup>. Como não poderia deixar de ser, existem alternativas, tais como: **ksh**, **tcsh**, **zsh**, **dash** ou **fish**.

Todos estes programas fornecem um CLI ao utilizador, mas possuem funcionalidades ligeiramente diferentes.

Quando se utiliza um ambiente gráfico existem programas denominados por *emuladores de terminal*, responsáveis por na realidade apresentarem uma janela de interacção com o utilizador, enviando teclas para a *Shell* e apresentando o conteúdo ao utilizador. Existem vários emuladores de terminal, sendo que as distribuições fornecem sempre vários para escolha, tais como: **xterm**, **konsole**, **gnome-terminal** ou **lxterminal**. No sistema instalado, o emulador de terminal chama-se **lxterminal** e pode ser encontrado na secção *Acessórios* do menu de aplicações.

---

<sup>2</sup>O nome advém do facto de o programa **bash** ser uma versão melhorada do programa **Bourne shell**, desenvolvido por Steve Bourne.

## Exercício 2.2

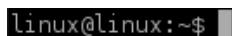
Procure o programa `lxterminal` e execute-o.

Um dos emuladores referidos anteriormente também está instalado, mas noutra secção. Encontre-o e execute-o.

Verifique que embora o texto apresentado seja semelhante, os emuladores apresentam aspectos diferentes para interacção.

Execute o comando `echo $SHELL` e verifique qual a *Shell* em utilização.

Ao executar o emulador de terminal, pode reparar que o interface é bastante simples, sempre apresentada apenas uma pequena informação tal representado na Figura 2.2.

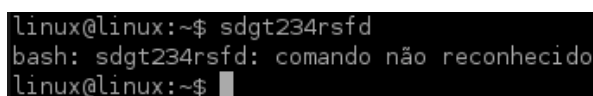


```
linux@linux:~$
```

Figura 2.2: *Prompt* apresentado pela `bash`.

O formato utilizado neste *Prompt* é o de utilizador @ nome da maquina: directorio actual \$. Ou seja, neste caso em particular, utilizador `linux`, máquina `linux`, directório actual `~`. Mais à frente, na Seção 2.3 iremos ver o que representa o `~`.

Se escrever qualquer coisa aleatória, exemplo `sdgt234rsfd`, seguido de ENTER, verá que a `bash` lhe indica uma situação de erro, tornando a pedir um comando válido.



```
linux@linux:~$ sdgt234rsfd
bash: sdgt234rsfd: comando não reconhecido
linux@linux:~$
```

Figura 2.3: Indicação de erro pela `bash`.

Como pode notar, por muitos comandos errados que introduza, a `bash` irá sempre pedir de novo um comando, sem que isso traga algum prejuízo para o sistema.

Observe também a resposta foi impressa imediatamente a seguir à linha do comando, de forma concisa, sem distrações nem grandes explicações. Este comportamento é usual em muitos comandos *UNIX* e é típico de um certo estilo defendido pelos criadores deste

sistema.  
Simples, mas eficaz.

### Exercício 2.3

Execute o comando `date` e observe o resultado.

### Exercício 2.4

Execute o comando `cal` e observe o resultado. Descubra em que dia da semana nasceu, passando o mês e o ano como *argumentos* ao comando `cal`, por exemplo:  
`cal jan 1981`.

## Formato dos comandos

Os comandos em *UNIX* têm sempre a forma:

---

```
comando argumento1 argumento2 ...
```

---

onde **comando** é o nome do programa a executar e os argumentos são cadeias de caracteres, que podem ser incluídas ou não, de acordo com a sintaxe esperada por esse programa.

Os argumentos podem ainda modificar o comportamento base do programa, designando-se nesse caso por opções (ou, por vezes, por *flags* em inglês, por pretenderem sinalizar algo de especial). Tipicamente as opções são indicadas de duas formas:

**-x** , onde *x* representa uma letra, maiúscula ou minúscula, ou um algarismo.

**--nome-da-opção** , onde *nome-da-opção* é um bloco de texto, sem espaços em branco, com a designação da opção.

### Exercício 2.5

Execute o comando `date -u` para ver a hora atual no fuso horário UTC e observe o resultado. Execute o comando `date --utc` para obter o mesmo resultado.

Na linha de comandos é possível recuperar um comando indicado anteriormente usando as teclas de direção ↑ e ↓. É possível depois editá-lo (alterá-lo) para produzir um novo comando (com argumentos diferentes, por exemplo). Outra funcionalidade muito útil



é a possibilidade de o sistema completar automaticamente comandos ou argumentos parcialmente escritos usando a tecla **Tab**.

Um interpretador de comandos também mantém uma lista numerada com todos os comandos executados durante a sua execução. Esta lista pode ser consultada através do comando **history**. O resultado produzido por este comando é uma listagem, numerada, dos comandos executados pela ordem cronológica da sua execução. Os interpretadores de comandos permitem usar os números usados nessa listagem para recuperar (e executar novamente) os respetivos comandos, através do comando **!n**, onde *n* é o número de ordem do comando que se pretende recuperar. A recuperação e execução rápida do comando exatamente anterior pode ser feita com o comando **!!**.

### Exercício 2.6

Execute o comando **history** e observe o resultado.

### Exercício 2.7

Execute o comando **!!** e observe o resultado. E se executar **!2**?

### Exercício 2.8

Recupere um comando anterior usando as teclas de direção, edite-o e execute-o.

### Exercício 2.9

Obtenha o número de ordem de um comando anterior e re-execute-o usando o comando que permite a indexação de um comando anterior (**!numero**).

Uma linha pode ser editada de forma elementar ou sofisticada. A forma elementar consiste em deslocar o cursor ao longo da linha, para trás ou para a frente, usando as teclas de direção **←** e **→**, adicionar novo texto à linha e apagar texto da linha, atrás do cursor, com a tecla **DELETE**.

## Ajuda dos comandos

O funcionamento dos comandos, as suas condicionantes e qual a sintaxe dos seus argumentos não é algo que se memorize de forma sistemática. Apenas os comandos mais utilizados, e os argumentos mais utilizados são possíveis de memorizar por um utilizador médio ou avançado. Desta forma, foi adoptada uma uniformização deste aspecto, resumindo-se aos seguintes pontos:

1. Todos os comandos podem ser executados utilizando **nome-do-comando -help** (ex., **date --help**), o que não irá produzir nenhuma acção além de mostrar uma breve ajuda.
2. Os programas normalmente aceitam opções de funcionamento nos formatos longos (ex., **--utc**) e curtos (**-u**).
3. Existe uma base de dados com a ajuda para todos os comandos disponíveis, acessível através do comando **man** seguido do nome do comando que se pretende consultar (ex., **man date**).

---

### Exercício 2.10

1. Considerando o comando **date** que vimos anteriormente, verifique o resultado de **date -u** e **date --utc**.
  2. Aceda à ajuda do comando **date** através dos métodos descritos.
  3. Aplique este método a outros comandos tais como **echo**, **true**, **false**.
- 

---

### Exercício 2.11

Verifique a utilidade do comando **apropos**. Pode usar como guia o parâmetro **successfully** e correlacionar o seu resultado com o comando **man**.

---

## 2.3 Sistema de Ficheiros

Os ficheiros e directórios são organizados seguindo o princípio de uma árvore com uma raiz (/) e directórios por baixo dessa raiz. Isto é semelhante ao utilizado no sistema operativo *Windows*, com a grande diferença que, em quanto o *Windows* considera que cada dispositivo (ex, Disco Rígido, Compact Disk (CD)) é uma unidade distinta, em *Linux* tal

como na maioria dos restantes sistemas operativos, os diversos dispositivos encontram-se mapeados em directórios da mesma raíz. A Figura 2.4 demonstra a estrutura do sistema de ficheiros se pode encontrar na máquina virtual fornecida, e típica de um qualquer *Linux*.

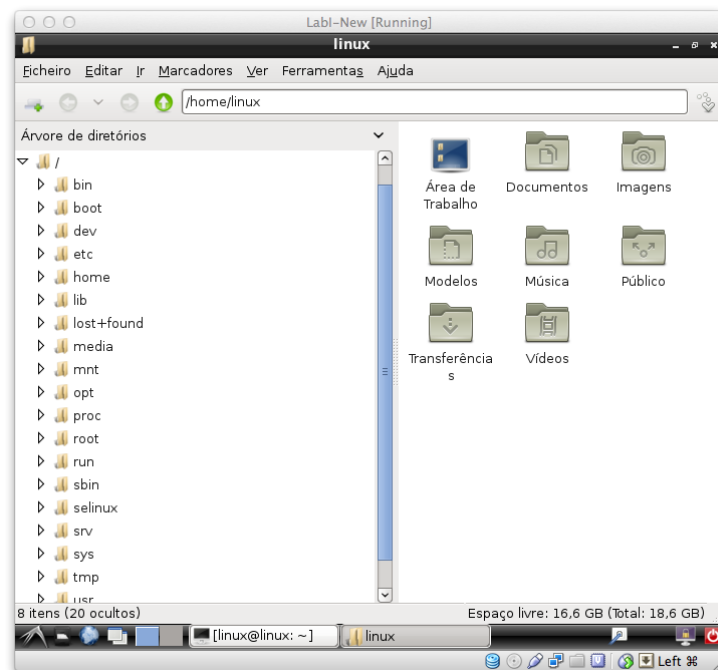


Figura 2.4: Estrutura do sistema de ficheiros *Linux*

Embora o *Linux* não obrigue a existência de uma estrutura específica, tipicamente o mesmo modelo é sempre utilizado. A Tabela 2.1 descreve o propósito de alguns destes directórios.

Cada utilizador possui um directório próprio nesta árvore, a partir do qual pode (e deve) criar e gerir toda a sua sub-árvore de directórios e ficheiros: é o chamado *directório do utilizador* ou *home directory*. Após a operação de *login* o sistema coloca-se nesse directório. Portanto neste momento deve ser esse o *directório atual* (*current directory*). Tipicamente este directório é representado pelo caractere `~`.

Para saber qual é o directório atual execute o comando `pwd`<sup>3</sup>. Deve surgir um nome como indicado na Figura 2.5, que indica que está no directório `linux` que é um subdirectório de `home` que é um subdirectório direto da raíz `/`.

---

<sup>3</sup>Acrónimo de *print working directory*.

Tabela 2.1: Principais directórios típicos do *Linux*

Directório	Descrição
/	Raiz do sistema de ficheiros
/bin	Executáveis essenciais do sistema.
/boot	Contem o <i>kernel</i> para iniciar o sistema.
/etc	Configurações.
/home	Áreas dos utilizadores.
/mnt	Pontos de montagem temporários.
/media	Pontos de montagem de unidades como os CDs.
/lib	Bibliotecas essenciais e módulos do <i>kernel</i> .
/usr	Bibliotecas e aplicações tipicamente usadas por utilizadores.
/sbin	Programas tipicamente utilizados pelo super-utilizador.
/tmp	Ficheiros temporários.
/var	Ficheiros frequentemente modificados (bases de dados, impressões).

Para listar o conteúdo do directório atual execute o comando `ls`<sup>4</sup>. Deve ver uma lista dos ficheiros (e subdirectórios) contidos no seu directório neste momento, tal como referido na Figura 2.5.

```
linux@linux:~$ pwd
/home/linux
linux@linux:~$ ls
Área de Trabalho  Imagens  Música  Transferências
Documentos        Modelos  Público  Vídeos
linux@linux:~$
```

Figura 2.5: Resultado dos comandos `ls` e `pwd`

Dependendo da configuração do sistema, os nomes nesta listagem poderão aparecer com cores diferentes e/ou com uns caracteres especiais (`/`, `@`, `*`) no final, que servem para indicar o tipo de ficheiro mas de facto não fazem parte do seu nome.

(Num ambiente gráfico a mesma informação está disponível numa representação mais visual. Experimente, por exemplo, escolher *Árvores de directórios/linux* para ver o conteúdo do seu directório pessoal.)

Ficheiros cujos nomes começam por “.” não são listados por omissão, são ficheiros *escondidos*, usados geralmente para guardar informações de configuração de diversos programas. Para listar todos os ficheiros de um directório, incluindo os escondidos, deve executar a variante `ls -a`<sup>5</sup>

<sup>4</sup> Abreviatura de *list*.

<sup>5</sup> A opção `-a` indica que se devem listar todos (*all*) os elementos do directório.

Por vezes é necessário listar alguns atributos dos ficheiros para além do nome. Pode fazê-lo executando as variantes `ls -l` ou `ls -la`, sendo que o resultado deverá ser semelhante ao apresentado na Figura 2.6.

```
linux@linux:~$ ls -la
total 108
drwxr-xr-x 16 linux linux 4096 Set 26 19:03 .
drwxr-xr-x  3 root  root  4096 Set 21 18:56 ..
drwxr-xr-x  2 linux linux 4096 Set 21 18:57 Área de Trabalho
-rw-r--r--  1 linux linux  220 Set 21 18:56 .bash_logout
-rw-r--r--  1 linux linux 3637 Set 21 18:56 .bashrc
drwxrwxr-x  5 linux linux 4096 Set 26 15:17 .cache
drwx----- 18 linux linux 4096 Set 26 13:45 .config
drwx-----  3 linux linux 4096 Set 21 18:57 .dbus
-rw-r--r--  1 linux linux   23 Set 26 16:01 .dmrc
drwxr-xr-x  2 linux linux 4096 Set 21 18:57 Documentos
drwx-----  3 linux linux 4096 Set 26 16:01 .gconf
-rw-rw-r--  1 linux linux  161 Set 26 13:01 .gtkrc-2.0
drwxr-xr-x  2 linux linux 4096 Set 21 18:57 Imagens
drwx-----  3 linux linux 4096 Set 21 18:57 .local
drwxr-xr-x  2 linux linux 4096 Set 21 18:57 Modelos
drwxr-xr-x  2 linux linux 4096 Set 21 18:57 Música
drwxr-xr-x  2 linux linux 4096 Set 21 23:54 .pip
-rw-r--r--  1 linux linux   675 Set 21 18:56 .profile
drwxr-xr-x  2 linux linux 4096 Set 21 18:57 Público
drwxr-xr-x  2 linux linux 4096 Set 21 18:57 Transferências
-rw-r-----  1 linux linux    5 Set 26 16:01 .vboxclient-clipboard.pid
-rw-r-----  1 linux linux    5 Set 26 16:01 .vboxclient-display.pid
-rw-r-----  1 linux linux    5 Set 26 16:01 .vboxclient-draganddrop.pid
-rw-r-----  1 linux linux    5 Set 26 16:01 .vboxclient-seamless.pid
drwxr-xr-x  2 linux linux 4096 Set 21 18:57 Videos
-rw-----  1 linux linux   50 Set 26 16:01 .Xauthority
-rw-r--r--  1 linux linux   14 Set 21 18:57 .xscreensaver
linux@linux:~$
```

Figura 2.6: Listagem completa dos ficheiro na área pessoal

Os principais atributos mostrados nestas listagens longas são:

**Tipo de ficheiro** identificado pelo primeiro carácter à esquerda, sendo `d` para diretório, `-` para ficheiro normal, `l` para *soft link*, etc.

**Permissões** representadas por 3 conjuntos de 3 caracteres. Indicam as permissões de leitura `r`, escrita `w` e execução/pesquisa `x` relativamente ao dono do ficheiro, aos outros elementos do mesmo grupo e aos restantes utilizadores da máquina.

**Propriedade** indica a que utilizador e a que grupo pertence o ficheiro.

**Tamanho** em número de bytes.

**Data e hora** da última modificação.

**Nome** do ficheiro.

Normalmente existe um *alias*<sup>6</sup> 11 equivalente ao comando `ls -l`.

Além do `ls` e variantes, existem outros comandos importantes para a observação e manipulação de diretórios, por exemplo:

`cd` --- o diretório atual passa a ser o diretório do utilizador.

`cd nome-do-dir` --- o diretório atual passa a ser o diretório `dir`.

`mkdir nome-do-dir` --- cria um novo diretório chamado `dir`.

`rmdir nome-do-dir` --- remove o diretório `dir`, desde que esteja vazio.

O argumento `dir` pode ser dado de uma forma absoluta ou relativa. Na forma absoluta, `dir` identifica o caminho (*path*) para o diretório pretendido a partir da raiz de todo o sistema de ficheiros; tem a forma `/subdir1/.../subdirN`. Na forma relativa, `dir` indica o caminho para o diretório pretendido a partir do diretório atual; tem a forma `subdir1/.../subdirN`.

Há dois nomes especiais para diretórios: “.” e “..” que representam respetivamente o diretório atual e o diretório pai, ou seja, o diretório ao qual o atual pertence.

### Exercício 2.12

Execute os comandos seguintes e interprete os resultados:

```
cd /
pwd
cd /usr
cd ~
pwd
cd /usr/sbin
pwd
cd -
pwd
```

### Exercício 2.13

Experimente utilizar o programa gráfico gestor de ficheiros para navegar pelos mesmos diretórios que no exercício anterior: `/`, `/usr`, `/usr/local/src`, etc. Acessível no menu *Acessórios/Gestor de ficheiros PcManFM*.

<sup>6</sup>Um *alias* é um nome alternativo usado em representação de um determinado comando. São criados usando o comando interno `alias`.

**Importante:** Nos PCs dos laboratórios, o subdiretório **arca** não é um diretório local do PC onde está a trabalhar; é na verdade a sua área privada de armazenamento no Arquivo Central de Dados (ARCA<sup>7</sup>), um servidor de ficheiros da Universidade de Aveiro. Esta área também é acessível a partir do ambiente Windows e através da Web. É neste diretório que deve gravar os ficheiros e diretórios que criar no decurso das aulas práticas. Os computadores das salas de aulas foram programados para apagarem o diretório de utilizador (e.g. `/homermt/a1245/`) sempre que são reiniciados. Só o conteúdo do subdiretório **arca** é salvaguardado. É portanto aí que deve colocar todo o seu trabalho.

### Exercício 2.14

Também pode executar estes comandos no computador da aula. Experimente mudar o diretório atual para o seu subdiretório **arca**. Liste o seu conteúdo. Reconhece algum dos ficheiros?

### Exercício 2.15

No PC da aula (e não na máquina virtual), crie, no diretório **arca**, um subdiretório chamado **labi** e, dentro desse, um diretório chamado **aula02**. Guarde neste diretório este guião.

## Manipulação de ficheiros

O Linux dispõe de diversos comandos de manipulação de ficheiros. Eis alguns:

`cat fic` --- imprime no dispositivo de saída *standard* (por omissão o ecrã) o conteúdo do ficheiro `fic`. O nome deste comando é uma abreviatura de *concatenate*, porque permite concatenar o conteúdo de vários ficheiros num só.

`rm fic` --- remove (apaga) o ficheiro `fic`.

`mv fic1 fic2` --- muda o nome do ficheiro `fic1` para `fic2`.

`mv fic dir` --- move o ficheiro `fic` para dentro do diretório `dir`.

`cp fic1 fic2` --- cria uma cópia do ficheiro `fic1` chamada `fic2`.

`cp fic dir` --- cria uma cópia do ficheiro `fic` dentro do diretório `dir`.

---

<sup>7</sup><https://arca.ua.pt>

`head fic` --- mostra as primeiras linhas do ficheiro (de texto) `fic`.

`tail fic` --- mostra as últimas linhas do ficheiro (de texto) `fic`.

`more fic` --- imprime no dispositivo de saída padrão (por omissão o ecrã), página a página, o conteúdo do ficheiro `fic`.

`less fic` --- comando similar ao `more`, mas que permite uma navegação mais sofisticada para trás e para diante nas linhas apresentadas.

`grep padrão fic` --- seleciona as linhas do ficheiro (de texto) `fic` que satisfazem o critério de seleção `padrão`.

`wc fic` --- conta o número de linhas, palavras e caracteres do ficheiro `fic`. O nome deste comando é um acrónimo de *word count*

`sort fic` --- ordena as linhas do ficheiro `fic` de acordo com um critério (alfanumérica crescente, por omissão).

`find dir -name fic` --- procura um ficheiro com o nome `fic` a partir do diretório `dir`.

Todos estes comandos podem ser invocados usando argumentos opcionais que configuram o seu modo de funcionamento.

---

### Exercício 2.16

Utilizando a consola, crie um directório chamado `aula02` no seu Ambiente de Trabalho e copie o ficheiro `/etc/passwd` para este directório. Imprima o seu conteúdo no ecrã.

Experimente outros comandos da lista acima.

---

## 2.4 Edição de ficheiros de texto

**Importante!** - Antes de iniciar esta secção, instale a aplicação `vim` usando os comandos `sudo apt-get update` e depois `sudo apt-get install vim`

Um ficheiro de texto é um ficheiro constituído por octetos (bytes) que representam caracteres alfanuméricos, sinais de pontuação, espaços em branco e caracteres invisíveis de mudança de linha. O editor primordial do Linux para ficheiros de texto é o `vi`, ou a sua versão mais atual `vim` (`gvim` a versão gráfica deste último). Também existe o `vim-qt`<sup>8</sup>

---

<sup>8</sup><https://bitbucket.org/equalsraf/vim-qt/wiki/Home>



que é desenvolvido por membros da Universidade de Aveiro!

O **vim** é um editor que suscita ódios e paixões entre os utilizadores do Linux. O seu modo de funcionamento é único, porque permite usar o teclado completo como fonte de caracteres (para o texto que se pretende introduzir) e como fonte de comandos. Assim, se durante uma edição de um texto se carregar na tecla “a”, o resultado pode ser a introdução do carácter “a” no texto, no local onde se encontra o cursor, ou a execução do comando associado a tecla *a*. Sendo assim, como se sabe exatamente o que irá suceder quando se carrega numa tecla?

O **vim** trabalha em dois modos, ditos de *comando* e de *inserção*. No modo de comando, todas as teclas representam comandos; no modo de inserção, todas as teclas representam algo que se pretende inserir no texto. A mudança entre estes dois modos faz-se com as seguintes teclas:

**Modo de comando** → **modo de inserção** : através de várias teclas que indicam várias formas como e onde se pretende introduzir texto. As mais diretas e usuais são:

**i** -- inserir no local do cursor.

**I** -- inserir no início da linha.

**a** -- inserir à frente do cursor.

**I** -- inserir após o final da linha.

**o** -- inserir uma linha abaixo da atual.

**O** -- inserir uma linha acima da atual.

**R** -- sobrepor à linha atual a partir do local do cursor.

**C** -- apagar linha atual a partir do local do cursor e inserir a partir desse ponto.

modo de inserção → modo de comando: através da tecla ESC (*escape*).

Para além destes dois modos, o **vi** possui ainda dois outros modos: editor (**ed**) e visual. O modo **ed** é acionado quando no modo de comando se carrega na tecla “:”. Este modo serve para executar comandos relacionados com a salvaguarda do conteúdo do ficheiro e com o abandono da edição:

**:q** -- terminar a edição atual sem salvaguardar o conteúdo do ficheiro.

**:q!** -- terminar a edição atual sem salvaguardar o conteúdo do ficheiro, ignorando avisos caso tenha sido alterado.

**:x** -- terminar a edição atual com a salvaguarda do conteúdo do ficheiro.

**:w** -- salvaguardar o conteúdo do ficheiro.

**:w fic** -- salvaguardar o conteúdo editado no ficheiro **fic**.

**:r fic** -- carregar para o editor o conteúdo do ficheiro **fic**.

O modo visual é o modo por omissão, ao qual o **vim** volta após se ter executado um comando no modo **ed**. Na versão gráfica do **vim** (**gvim**) pode fazer estas operações do modo **ed** recorrendo apenas aos menus da sua interface gráfica.

Caso não se sinta confortável com o **vim** pode usar outros editores mais convencionais, como o **leafpad** ou o **geany**.

Note, porém, que a grande vantagem do **vim** face aos demais é o facto de se poder usar quase da mesma forma em ambientes gráficos e em consolas.

Finalmente, importa referir que tanto o **vim** como o **geany** possuem a função de realce de sintaxe. Ou seja, sempre que se estiver a editar algo que possui uma estrutura linguística conhecida do editor (ex, *Java*), ele tem capacidade de usar cores diferentes para blocos diferentes de texto, de modo a facilitar a sua leitura e análise.

---

### Exercício 2.17

Edite um ficheiro e escreva no mesmo os aspetos do sistema *Linux* que mais o surpreenderam, tanto positivamente como negativamente. Experimente fazê-lo com o **vim**, e exercite as diferentes formas de entrada em modo de inserção, bem como os vários comandos do modo **ed**.

---

#### 2.4.1 Procura de texto

O modo de procura de texto do **vim** é o mesmo que se usa em vários outros comandos, como o **man**, o **more**, o **less**, etc.

A pesquisa de um bloco de texto num ficheiro é feita em modo comando com a tecla “/” (pesquisa para diante do cursor) ou “?” (pesquisa para trás do cursor). Num teclado americano estas letras estão na mesma tecla física e num local muito conveniente, o que não acontece nos teclados portugueses...

Após a escrita de um destes caracteres deve-se escrever o texto a procurar e terminar com a tecla **Enter** (ou **Return**), após o que a aplicação em causa desloca o texto que apresenta para o local onde se encontra uma instância do texto que se pretende encontrar. Para encontrar a instância seguinte, mantendo o mesmo sentido de pesquisa (para diante

ou para trás do cursor), deve-se usar o comando “n”. Para realizar novamente a pesquisa mas invertendo o sentido anterior deve ser usado o comando “N”.

### Exercício 2.18

Edite o ficheiro anterior com o `vim` e realize pesquisas de texto no mesmo. Note que para o fazer terá de estar em modo de comando.

### Exercício 2.19

Apresente o conteúdo do ficheiro anterior com o comando `less` e realize pesquisas de texto no mesmo.

### Exercício 2.20

Execute o comando `man less` e realize pesquisas de texto do manual, tanto para diante como para trás.

### Exercício 2.21

Descubra como se consegue ir para o início e para o fim do texto do manual apresentado pelo comando `man`. Tenha em consideração que o texto do manual é produzido pelo comando `man` mas apresentado pelo comando `less` e que a deslocação para um ponto do texto é referida através da expressão “Go to” no manual.

## 2.5 Gestão de utilizadores

O sistema *Linux* assume uma gestão baseada em utilizadores e grupos. Já deve ter reparado que assim era, pois para aceder a qualquer sistema é sempre necessário fornecer uma senha associada a um utilizador (*login*).

A utilização deste sistema permite que múltiplos utilizadores façam uso do sistema apenas após autenticação, respeitando a privacidade de cada utilizador, e evitando que um dado utilizador danifique os dados de um outro utilizador.

Este modelo foi generalizado de forma a considerar tanto utilizadores reais, como é o caso do utilizador `linux`, como outros que servem para confinar aplicações a definições

de segurança específicos. A título ilustrativo pode considerar que o programa que serve páginas Web apenas deverá ter acesso aos ficheiros das páginas que serve, e não aos ficheiros de um outro programa.

Um utilizador em *Linux* tem um caracter especial: o **root**. Este utilizador corresponde ao super-utilizador, ou administrador da máquina, tendo permissões para aceder e modificar qualquer recurso.

Pode verificar qual o seu utilizador através do comando **whoami**, ou em mais detalhe através do comando **id**<sup>9</sup>. O resultado deverá ser o demonstrado na Figura 2.7. Experimente o mesmo comando no PC da aula.

```
linux@linux:~$ whoami
linux
```

Figura 2.7: Resultado da execução do comando **whoami**

No PC da sala, os alunos terão apenas acesso à sua área pessoal e não à de outros, nem a ficheiros críticos para o funcionamento correcto do sistema.

### Exercício 2.22

Utilizando os comandos **mkdir** e **rmdir** verifique se possui permissões para escrever nos seguintes locais: **/home/linux**, **/etc**, **/tmp**, **/bin**.

Na máquina virtual, é possível assumir outros utilizadores, o que é especialmente importante para tarefas de gestão. Através do comando **sudo** é possível ao utilizador **linux** assumir temporariamente as funções de super-utilizador (isto é: **root**).

A primeira vez, numa sessão, que o utilizador **linux** utilizar o comando **sudo**, será pedida a sua palavra passe para confirmação. Execuções seguintes irão lembrar-se do resultado desta validação e não será pedida qualquer outra informação.

A Figura 2.8 demonstra a utilização do comando **sudo** nas suas duas formas mais comuns. No primeiro caso (**sudo whoami**), o comando **whoami** é executado como super-utilizador, sendo que o comando **whoami** seguinte já irá voltar a utilizar o utilizador **linux**. É possível executar o comando no formado **sudo -s**, o que irá criar uma nova *Shell* no contexto do super-utilizador. Todos os comandos inseridos até que seja introduzido **exit**, irão ser executados com os privilégios do utilizador **root**.

---

<sup>9</sup>O comando **id** irá mostrar o utilizador e todos os grupos aos quais ele pertence

```

linux@linux:~$ whoami
linux
linux@linux:~$ sudo whoami
root
linux@linux:~$ whoami
linux
linux@linux:~$ sudo -s
root@linux:~# whoami
root
root@linux:~# exit
exit
linux@linux:~$ whoami
linux
linux@linux:~$ █

```

Figura 2.8: Resultado da execução do comando `sudo`, inspeccionando qual o utilizador em uso.

### Exercício 2.23

Verifique se possui permissões para visualizar (`cat`) o ficheiro `/etc/shadow`. Se não possuir, visualize o seu conteúdo recorrendo ao comando `sudo`.

Consegue identificar qual o conteúdo deste ficheiro? Pode recorrer ao comando `man`.

Como não poderia deixar de ser, os utilizadores não são tratados pelo sistema operativos com os nomes que temos utilizado (ex, `linux`). Na realidade tudo são números, e tanto os utilizadores como grupos existente são mapeados para números. Isso explica o porquê dos números obtidos quando executa o comando `id`, tal como representado na Figura 2.9.

```

linux@linux:~$ id
uid=1000(linux) gid=1000(linux) grupos=1000(linux),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),108(lpadmin),124(sambashare)

```

Figura 2.9: Resultado da execução do comando `id`, inspeccionando qual o utilizador em uso.

São de relevância dois valores apresentados:

`uid` - User Identifier, ou seja, o número que identifica o utilizador.

`gid` - Group Identifier, ou seja, o número do grupo principal do utilizador.

Estes valores são utilizados para atribuir as permissões. E os nomes utilizados para facilitar a gestão aos administradores (humanos).

### Exercício 2.24

Recorrendo ao comando `sudo` verifique qual o `uid` e `gid` do utilizador `root`.

### Exercício 2.25

Analise o resultado do comando `ls -la` sobre várias localizações, como por exemplo: `/etc`, `/tmp` e verifique a que utilizadores pertencem os ficheiros e quais as permissões.

## 2.6 Gestão de Aplicações

Um facto importante das distribuições de *Linux* é o de utilizarem um sistema integrado para a pesquisa, instalação, actualização e remoção de aplicações. Pode-se comparar à loja de aplicações tipicamente disponíveis nos telemóveis actuais. As grandes diferenças é que todas as aplicações disponibilizadas por estes meios são livres de se utilizar, e existe um conjunto de ferramentas para a sua gestão.

**Importante!** Todas as distribuições usam o seu sistema de gestão de pacotes, sendo que algumas partilham as ferramentas utilizadas. No entanto, não existe um método universal, comum a todas as distribuições para a gestão de pacotes. Os comandos necessários para esta gestão irão variar para cada distribuição utilizada. A título de exemplo, na distribuição *SliTaz* é usado o `tazpkg`.

No caso das distribuições que possuem como seu ancestral a distribuição *Debian*<sup>10</sup>, a gestão de aplicações é realizada através da família de aplicações `apt-`, sendo que em algumas distribuições existem ambientes mais compostos como o `aptitude`, o `synaptic`, ou mesmo o *Ubuntu One*.

Através da linha de comandos, e recorrendo aos comandos `apt-*`, é possível realizar uma gestão completa. De realçar que a gestão de aplicações é uma operação privilegiada. Portanto só disponível a utilizadores com permissões para o efeito, normalmente através de `sudo`.

---

<sup>10</sup>Para ver como se organizam as distribuições, consultar <http://futurist.se/gldt/>

São dois os principais comandos a utilizar:

- `apt-get` -- Permite actualizar, instalar e remover aplicações.
- `apt-cache` -- Permite procurar por aplicações.

Para a utilização deste sistema é necessário em primeiro lugar actualizar a lista de aplicações. Isto irá transferir a lista de aplicações nos servidores da distribuição, para o computador local. Depois torna-se possível pesquisar sobre essa lista e assim seleccionar aplicações a instalar. Periodicamente este processo de actualização tem de ser repetido.

Este processo é importante pois permite obter actualizações para as aplicações instaladas.

### Exercício 2.26

Utilizando o comando `apt-get update` actualize a base de dados de aplicações.

Utilizando o comando `apt-get upgrade` actualize as aplicações do sistema.

O processo para instalar a aplicação `cowsay` será seguinte (ver Figura 2.10):

1. `apt-cache search cowsay` -- Procura pelo nome exacto da aplicação `cowsay`.
2. `apt-get install cowsay` -- Instala a aplicação `sl`.
3. `apt-get remove cowsay` -- (caso necessário), remove a aplicação `cowsay`.

### Exercício 2.27

Instale e verifique o funcionamento das aplicações `fortune`, `sl`. Estas aplicações serão instaladas para o directório `/usr/games`.

## 2.7 Pontos de Montagem

Se estiver habituado ao sistema *Windows*, esperará que uma unidade remota, como por exemplo, uma PEN USB, apareça como uma letra adicional no sistema de ficheiros (ex, D:). Em *Linux* isso não acontece, sendo que os dispositivos são representados nos

```

root@linux:~# apt-cache search cowsay
cowsay - configurable talking cow
xcowsay - vaca falante gráfica e configurável
root@linux:~# apt-get install cowsay
A ler as listas de pacotes... Pronto
A construir árvore de dependências
A ler a informação de estado... Pronto
Pacotes sugeridos:
  filters
Serão instalados os seguintes NOVOS pacotes:
  cowsay
0 pacotes actualizados, 1 pacotes novos instalados, 0 a remover e 3 não actualizados.
É necessário obter 0 B/20,4 kB de arquivos.
Após esta operação, serão utilizados 90,1 kB adicionais de espaço em disco.
A seleccionar pacote anteriormente não seleccionado cowsay.
(A ler a base de dados ... 61522 ficheiros e directórios actualmente instalados.)
A descompactar cowsay (desde ../cowsay_3.03+dfsg1-4_all.deb) ...
A processar 'triggers' para man-db ...
A instalar cowsay (3.03+dfsg1-4) ...
root@linux:~# /usr/games/cowsay Labi

  ____
< Labi >
  -----
      \      ^__^
         (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||

root@linux:~# █

```

Figura 2.10: Processo de instalação da aplicação `cowsay`

chamados *pontos de montagem*, ou *mount points*.

Estes pontos não são mais do que directórios onde se define que qualquer escrita ou leitura dentro dele é transformada numa leitura ou escrita num outro dispositivo.

De forma a definir e consultar os pontos de montagem, poderá recorrer ao comando `mount`. Se o executar sem qualquer parâmetro, irá ver que existem diversos pontos de montagem já definidos (ver Figura 2.11).

Cada linha apresentada irá indicar qual o dispositivo real utilizado, qual o sistema de ficheiros, onde está montado o dispositivo e quais as opções. Como exemplo, poderá verificar que existe um dispositivo chamado `/dev/sda1`, montado em `/`, que é do tipo `ext4`. Isto corresponde ao sistema de ficheiros principal da máquina que está a utilizar. Um outro interessante é o que está montado em `/proc`. Este dispositivo (`proc`) é virtual e criado pelo núcleo *Linux* de forma a permitir observar e configurar parâmetros de execução.



```

root@linux:~# mount
/dev/sda1 on / type ext4 (rw,noatime,errors=remount-ro)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
none on /sys/fs/cgroup type tmpfs (rw)
none on /sys/fs/fuse/connections type fusectl (rw)
none on /sys/kernel/debug type debugfs (rw)
none on /sys/kernel/security type securityfs (rw)
udev on /dev type devtmpfs (rw,mode=0755)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=0620)
tmpfs on /run type tmpfs (rw,noexec,nosuid,size=10%,mode=0755)
none on /run/lock type tmpfs (rw,noexec,nosuid,nodev,size=5242880)
none on /run/shm type tmpfs (rw,nosuid,nodev)
none on /run/user type tmpfs (rw,noexec,nosuid,nodev,size=104857600,mode=0755)
gvfsd-fuse on /run/user/linux/gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,user=linux)
root@linux:~# █

```

Figura 2.11: Resultado da execução do comando `mount`.

### Exercício 2.28

Inspeccione os ficheiros presentes em `/proc`. Pode recorrer a comandos como o `cat` para mostrar o conteúdo destes ficheiros.

Por exemplo, são relevantes as entradas: `cpuinfo`, `diskstats`, `filesystems`, `interrupts`, `meminfo`, `partitions`, `uptime` ou `version`. Também pode utilizar o comando `man` para obter informação sobre cada uma das entradas.

O conceito de ponto de montagem é o que permite partilhar ficheiros entre o anfitrião e a máquina virtual. Depois de activar uma pasta partilhada no *VirtualBox*, poderemos usar o comando `mount` da seguinte forma: `mount -t vboxsf labi /mnt`. Isto quer dizer que iremos montar um dispositivo chamado `labi`, que possui um sistema de ficheiros do tipo `vboxsf` e que o ponto de montagem será o `/mnt`. A Figura 2.12 mostra o resultado deste comando. Para desassociar um ponto de montagem, usamos o comando `umount`.

### Exercício 2.29

Recorrendo ao comando `mount`, monte uma pasta partilhada no directório `/mnt`. Verifique que o ponto de montagem aparece listado e transfira um ficheiro entre a máquina virtual e o anfitrião.

```

root@linux:~# mount -t vboxsf labi /mnt
root@linux:~# mount
/dev/sdal on / type ext4 (rw,noatime,errors=remount-ro)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
none on /sys/fs/cgroup type tmpfs (rw)
none on /sys/fs/fuse/connections type fusectl (rw)
none on /sys/kernel/debug type debugfs (rw)
none on /sys/kernel/security type securityfs (rw)
udev on /dev type devtmpfs (rw,mode=0755)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=0620)
tmpfs on /run type tmpfs (rw,noexec,nosuid,size=10%,mode=0755)
none on /run/lock type tmpfs (rw,noexec,nosuid,nodev,size=5242880)
none on /run/shm type tmpfs (rw,nosuid,nodev)
none on /run/user type tmpfs (rw,noexec,nosuid,nodev,size=104857600,mode=0755)
gvfsd-fuse on /run/user/linux/gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,user=linux)
labi on /mnt type vboxsf (rw)
root@linux:~# ls /mnt
conteudo_do_diretorio_labi.txt
root@linux:~#

```




Figura 2.12: Resultado da execução do comando `mount` para montar os documentos partilhados.

Para não se repetirem estes comandos sempre que se pretende aceder a um ponto de montagem. Isto é, de forma a tornar um ponto de montagem mais permanente, é possível utilizar o ficheiro `/etc/fstab` para registar a informação de montagem. A partir do momento que este ficheiro contém informação relativa aos pontos de montagem, passa também a ser possível montar dispositivos de forma mais rápida, executando por exemplo `mount /mnt`. Este comando irá montar o directório `/mnt` de acordo com a informação presente no ficheiro `/etc/fstab`.

### Exercício 2.30

Adicione a informação necessária ao ficheiro `/etc/fstab` de forma a automatizar a montagem da pasta partilhada. Seguidamente, monte a pasta usando a forma rápida descrita anteriormente.

Verifique se também pode desmontar o directório da mesma forma.

## 2.8 Processos

Em linux existe uma máxima que diz que tudo é um ficheiro ou um processo. Até agora nós vimos como funcionam os ficheiros, mas ignorámos os processos.

Os processos não são nada mais do que os programas em execução na máquina. Todo o ambiente gráfico ou texto que lhe é apresentado, assim como a *Shell* e muitos outros programas estão a executar. Como não poderia deixar de ser, todos os processo são identificados por um número chamado de Process IDentifier (PID).

Pode utilizar o comando `ps` para verificar os processos da sua sessão, ou usar combinar com as opções `-ax` para ver que processos estão activos na máquina virtual, independentemente do utilizador. Se adicionar a opção `-u` irá igualmente verificar detalhes dos processos como por exemplo a memória utilizada.

O comando `ps Tu` irá mostrar os processos associados com o terminal e os seus detalhes. Como pode ver (Figura 2.13), é mostrado o utilizador, o PID, a quantidade de processador e memória utilizados, o seu estado, à quanto tempo foram iniciados e qual o seu nome.

```
linux@linux:~$ ps Tu
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
linux    1784  0.0  0.5  6372  2736 pts/0    Ss   07:08   0:00 /bin/bash
linux    4638  0.0  0.2  5276  1220 pts/0    R+   17:17   0:00 ps Tu
linux@linux:~$
```

Figura 2.13: Resultado da execução do comando `ps Tu` para montar os processos associados com a sessão actual.

### Exercício 2.31

Compare o resultado de `ps`, `ps -a`, `ps -ax` e `ps -aux`.

Qual o processo que consome mais memória? Qual o processo que consome mais CPU? Quantos utilizadores têm processos activos?

### Exercício 2.32

Repita o exercício anterior utilizando o comando `top`. Pode utilizar as teclas “<” e “>” para seleccionar qual a coluna escolhida para ordenar os processos.

Um aspecto importante dos processos é a sua gestão, nomeadamente a possibilidade de controlar o seu estado de execução e, eventualmente termina-la de forma forçada.

Os comandos `kill` e `killall` permitem enviar sinais aos programas. A diferença entre estes comandos reside no facto de o primeiro (`kill`) enviar um sinal a um processo específico identificado pelos seu identificador:

```
kill sinal identificador-do-processo
```

Enquanto o segundo (`killall`) enviar um sinal a todos os processos com um determinado nome:

```
killall sinal nome-do-processo
```

Um destes sinais, o `SIGKILL` ou `-9` permite terminar qualquer programa <sup>11</sup>

### Exercício 2.33

Inicie e termineis. No primeiro inicie um processo `top`. No segundo terminal, recorrendo aos comandos `ps` e `kill/killall` termine a execução do processo `top`.

## 2.9 Para aprofundar o tema

### Exercício 2.34

Explore os restantes ficheiros e directórios nos directórios `/proc` e `/sys`.

Recorra ao comando `man` para obter informação sobre cada uma das entradas.

### Exercício 2.35

Utilizando o comando `adduser` adicione um outro utilizador ao sistema. Pode depois apagar este utilizador através do comando `deluser`.

Depois de criado o utilizador, tente aceder a uma sessão com esse utilizador. Verifique quais os grupos a que pertence se se pode executar o comando `sudo`.

---

<sup>11</sup>Para consultar a lista de todos os sinais, pode utilizar o comando `man 7 signals`.

### Exercício 2.36

Utilizando o comando `apt-get` instale a aplicação `synaptic` e experimente-a. Pode agora, de uma forma gráfica, explorar todas as aplicações disponíveis para o sistema *Ubuntu*.

### Exercício 2.37

Explore a utilização dos operadores “|” e “>”.  
Por exemplo:

```
ls -la > ls.txt
```

ou

```
ls -la |wc
```



## Glossário

<b>CD</b>	Compact Disk
<b>CLI</b>	Command Line Interface
<b>LXDE</b>	Lightweight X11 Desktop Environment
<b>MiB</b>	Mibi Byte
<b>PID</b>	Process IDentifier
<b>PC</b>	Computador Pessoal
<b>RAM</b>	Random Access Memory

## Referências

- [1] Lubuntu Team, *Lubuntu / lightweight, fast, easier*, <http://www.lubuntu.net>, [Online; acedido em 24 de Setembro de 2014], 2013.
- [2] LXDE Project, *Lightweight x11 desktop environment*, <http://lxde.org>, [Online; acedido em 24 de Setembro de 2014], 2013.