Tema 2

Introdução ao Ambiente Linux

Objetivos:

- Introdução ao Linux
- Interpretador de comandos
- Sistema de ficheiros
- Processos
- Utilizadores
- Gestão de aplicações

2.1 Introdução

O sistema operativo Linux, criado nos anos 90, é uma implementação livre de *UNIX*, um sistema operativo dos anos 70, que se popularizou devido à sua portabilidade para diferentes arquitecturas de hardware. Existem muitas distribuições de Linux disponíveis, mas todas têm em comum o *kernel* (núcleo), que foi desenvolvido por Linus Torvalds. Neste trabalho vão-se utilizar comandos *UNIX* para que o aluno se familiarize com o sistema operativo Linux. Por um lado, principalmente, vai-se trabalhar na linha de comandos; mas por outro lado, também iremos ver como alguns passos podem ser feitos quer na linha de comandos quer em ambiente gráfico. O sistema operativo Linux tem várias vantagens no seu kernel comparativamente com o sistema operativo Windows. O Linux é um sistema robusto e estável, que suporta tarefas de cálculo e actividades de peso com eficácia; é um sistema acessível, pois tem custo zero - é gratuito; é flexível, pois como o seu código fonte é aberto, o utilizador pode alterar variados aspectos à sua vontade, isto após ter várias distribuições por onde escolher; também é seguro, pois desde o kernel até à versão final a segurança é um aspecto trabalhado pelos co-autores de cada distribuição; este sistema operativo não precisa de anti-virus, raramente bloqueia e é muito rápido e eficaz.

2.2 Instalação de sistemas operativos

O processo normal de instalação de um sistema operativo é feito do seguinte modo:

1. O sistema instalador é disponibilizado num suporte móvel (Compact Disk (CD), Digital Versatile Disk (DVD), memória *flash*, etc.), total ou parcialmente. Quando é disponibilizado parcialmente, a parte em falta é obtida de repositórios da Internet.

2. O sistema instalador é executado logo após o arranque da máquina, sendo ativado pelo sistema de controlo do arranque da máquina (*boot loader*).

3. O sistema instalador escolhe um disco rígido da máquina, ou um conjunto de partições de discos da máquina, para aí criar os sistemas de ficheiros que irão ser usados pelo sistema operativo que irá ser instalado. Normalmente usam-se duas partições diferentes, uma com os ficheiros que normalmente vemos no sistema de ficheiros, outra designada como *swap* que serve para apoio à gestão da memória virtual. Esta última partição pode ser substituída por um ficheiro.

4. Após a instalação do sistema operativo no disco rígido, a partição de arranque dessa instalação é marcada como sendo de arranque (*boot*) e o sistema está pronto para ser reiniciado. Por vezes no arranque de um sistema instalado é iniciado primeiro um sistema de controlo dos sistema operativos a iniciar, de que é exemplo o `Grub`: arranque da BIOS → seleção do dispositivo de arranque → carregamento do módulo de arranque do dispositivo → seleção do sistema operativo a arrancar (opcional) → arranque do sistema operativo escolhido.

2.2.1 Sistemas *live*

Há, contudo, variantes a este processo base. Um deles consiste no arranque dos sistemas ditos *live* (ou distribuições *live*). Os sistema *live* são sistemas que arrancam como os demais mas não alteram nada na máquina de forma definitiva. Em particular, não usam qualquer repositório persistente da máquina (v.g. discos rígidos) para guardar qualquer informação. Portanto, estes sistemas podem-se executar em máquinas sem disco rígido.

Uma distribuição *live*, de que há inúmeros exemplos para Linux[1], é uma imagem de CD (ficheiro ISO) que pode ser usada para arrancar um sistema *live* numa máquina, a partir do seu leitor de CD.

Atualmente muitas das distribuições *live* possuem uma funcionalidade 2-em-1: permitem o arranque de uma versão *live* normal, mas essa permite depois criar uma instalação no disco rígido da máquina. Esta faceta será explorada neste trabalho.

[1] `http://en.wikipedia.org/wiki/List_of_live_CDs`

2.2.2 Distribuições populares de Linux

No mundo linux há várias distribuições populares, tais como as listadas na Figura 2.1: debian, a partir da qual se desenvolveram ubuntu, lubuntu, linuxmint, etc.; redhat a partir da qual se desenvolveram fedora, centOS, etc., slackware a partir da qual vem o openSUSE, etc.; archlinux e gentoo linux; entre muitas outras.

Nos computadores dos laboratórios de aulas está instalada uma versão de *Ubuntu*. O aluno também pode instalar o Ubuntu ou outra versão Linux em casa ou numa aula OT. Se optar pelo Ubuntu terá uma distribuição completamente funcional, com todas as vantagens do mundo *Ubuntu* (repositórios, suporte, etc.) e uma maior semelhança com o ambiente das aulas.

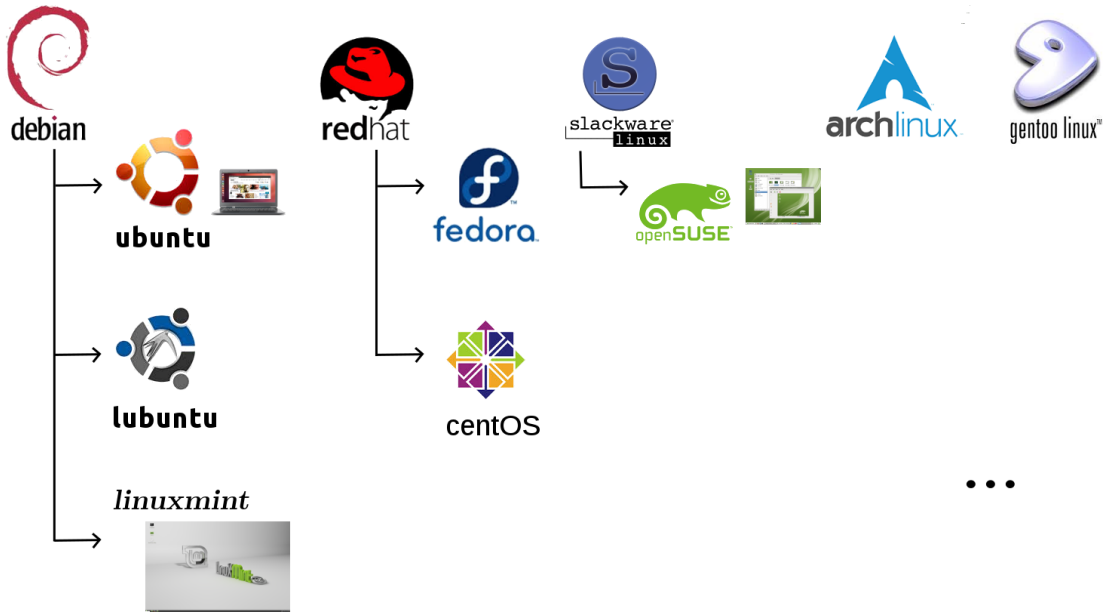

Figura 2.1: Algumas das distribuições mais populares de Linux.

2.3 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 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 que 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`[2], que está incluído em todas as distribuições usuais de Linux.

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.

2.3.1 Interfaces de texto e gráficas

Os sistemas *Linux* geralmente 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 → consola : Ctrl + Alt + F_n, onde F_n é uma das teclas F1, F2, $\cdots$, 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).

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

[2] `http://www.x.org`

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 combinação 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.

2.3.2 Interpretador de comandos

A maioria das distribuições *Linux* fornece uma Command Line Interface (CLI) através do programa **`bash`**.[3] Como não poderia deixar de ser, existem alternativas, tais como: **`ksh`**, **`tcsh`**, **`zsh`**, **`dash`** ou **`fish`**. Todos estes programas fornecem uma CLI ao utilizador, mas possuem funcionalidades ligeiramente distintas.

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`**.

Exercício 2.2

Usando o mecanismo de procura ou os menus do ambiente gráfico, procure um emulador de terminal e execute-o.

Procure um dos outros emuladores e execute-o.

Verifique que embora o texto apresentado seja semelhante, os emuladores apresentam aspectos ligeiramente diferentes e têm menus e funcionalidades de configuração diferentes. Descubra como pode mudar a cor de fundo e o tamanho de letra em cada um dos emuladores.

Execute o comando **`echo $SHELL`** para verificar qual a *Shell* em utilização em cada terminal.

[3] O nome advém do facto de o programa **`bash`** ser uma versão melhorada do programa **`Bourne shell`**, desenvolvido por Steve Bourne.

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.

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 **labi**, máquina **labi**, directório actual ∼. Mais à frente, na Secção 2.4 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.

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 que um comando faz, bem como a sintaxe e significado dos seus argumentos são decididos pelos seus programadores. Um utilizador médio tem de aprender a sintaxe e semântica de uma dezena ou duas de comandos e algumas das suas opções mais usuais, mas ninguém consegue memorizar os milhares de comandos disponíveis. Para uniformizar um pouco a sintaxe dos comandos e assim facilitar a aprendizagem, os programas seguem algumas convenções básicas:

1. Muitos programas aceitam opções de funcionamento quer no formato longo (`date --utc`), quer no formato curto, de uma letra só (`date -u`).

2. Várias opções no formato curto podem geralmente ser amalgamadas, e.g., `ls -l -a` equivale a `ls -la`.

3. Quase todos os comandos aceitam uma opção `--help`, por exemplo `date --help`, que serve apenas para mostrar um breve texto de ajuda.

4. Existe uma base de dados com manuais de utilização para todos os comandos disponíveis, acessível através do comando **`man`** seguido do nome do comando que se pretende consultar (e.g., `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`**.

2.4 Sistema de Ficheiros

Os ficheiros e directórios são organizados seguindo o príncipio de uma árvore com uma raíz (**`/`**) e directórios por baixo dessa raíz. 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, 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*.

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 diretório próprio nesta árvore, a partir do qual pode (e deve) criar e gerir toda a sua sub-árvore de diretórios e ficheiros: é o chamado *diretório do utilizador* ou *home directory*. Após a operação de *login* o sistema coloca-se nesse diretório.

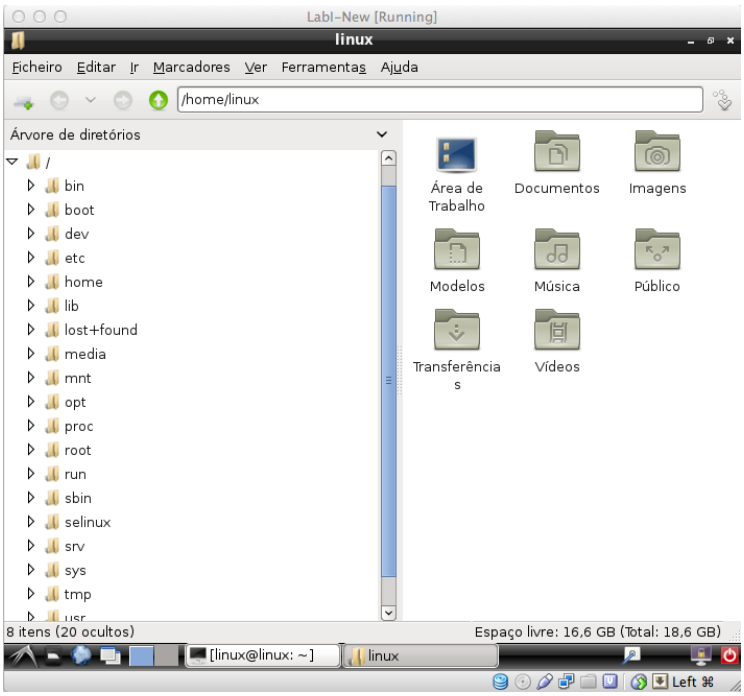

Figura 2.4: Estrutura do sistema de ficheiros *Linux*

Tabela 2.1: Principais directórios típicos do *Linux*

Directório	**Descrição**
`/`	Raiz do sistema de ficheiros
`/bin`	Executáveis essenciais do sistema.
`/boot`	Contem o *kernel* 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 *kernel*.
`/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).

Portanto neste momento deve ser esse o *diretório atual* (*current directory*). Tipicamente este directório é representado pelo caractere ∼.

Para saber qual é o diretório atual execute o comando **`pwd`**[4]. Deve surgir um nome como indicado na Figura 2.5, que indica que está no diretório `labi` que é um subdiretório de `home` que é um subdiretório direto da raiz `/`.

Para listar o conteúdo do diretório atual execute o comando **`ls`**[5]. Deve ver uma lista dos ficheiros (e subdiretórios) contidos no seu diretório neste momento, tal como referido na Figura 2.5.

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/labi* para ver o conteúdo do seu diretório 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 diretório, incluindo os escondidos, deve executar a variante **`ls -a`**[6]

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.

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.

[4] Acrónimo de *print working directory*.
[5] Abreviatura de *list*.
[6] A opção `-a` indica que se devem listar todos (*all*) os elementos do diretório.

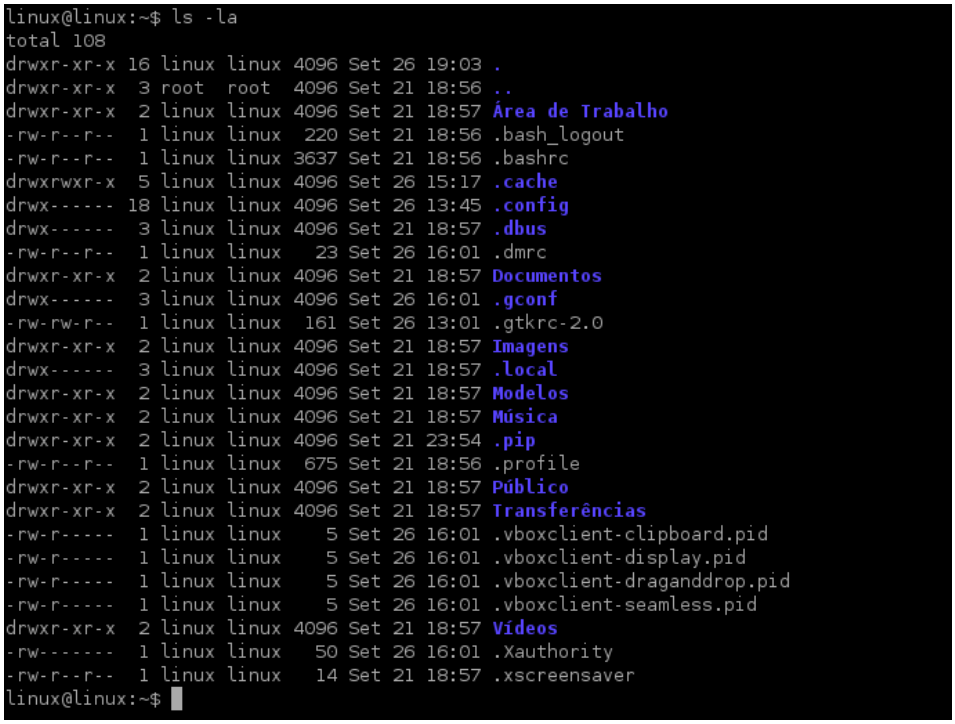

Figura 2.6: Listagem completa dos ficheiro na área pessoal

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*[7] `ll` equivalente ao comando `ls -l`.

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

[7]Um *alias* é um nome alternativo usado em representação de um determinado comando. São criados usando o comando interno `alias`.

`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.

Importante: Nos computadores das salas de aulas da UA, o subdiretório **`arca`** não é um diretório local do Computador Pessoal (PC) onde está a trabalhar; é na verdade a sua área privada de armazenamento no Arquivo Central de Dados (ARCA[8]), um servidor de

[8] https://arcaweb.ua.pt

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

Experimente mudar o diretório atual para o seu subdiretório `arca`. Liste o seu conteúdo. Reconhece algum dos ficheiros?

Exercício 2.15

Crie, no diretório `arca`, um subdiretório chamado `labi` e, dentro desse, um diretório chamado `aula01`. 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`.

`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 linha de comandos, crie um directório chamado **aula01** 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.

2.5 Edição de ficheiros de texto

Nos computadores do laboratório temos instaladas as aplicações **vim** e **gedit** para manipulação de ficheiros de texto na consola e num ambiente gráfico, respectivamente.

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. Um dos editores de ficheiros de texto mais usados em sistemas UNIX é o **vim**, uma versão melhorada do **vi**. Apesar da sua antiguidade, continua a ser preferido por muitos utilizadores, particularmente programadores, pela sua eficiência, pelas suas funções avançadas e por correr numa consola de texto. Atualmente também existem versões gráficas, como o **gvim** e o **vim-qt**[9] que é desenvolvido por membros da Universidade de Aveiro!

O **vim** é um editor com um modo de funcionamento 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 caráter "a" no texto, no local onde se encontra o cursor, ou a execução do comando associado à tecla *a*, dependendo do modo de trabalho actual no editor vim.

[9] https://bitbucket.org/equalsraf/vim-qt/wiki/Home

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 uma de entre várias teclas que indicam a posição onde se pretende introduzir texto. Algumas das mais usadas:

i — inserir no local do cursor.

o — inserir uma linha abaixo da atual.

O — inserir uma linha acima da atual.

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`.

:e fic — abre o ficheiro `fic` para edição.

:r fic — insere o conteúdo do ficheiro `fic` abaixo do cursor.

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 as operações do modo `ed` recorrendo aos menus da sua interface gráfica.

Caso não se sinta confortável com o `vim` pode usar outros editores mais convencionais, como o `gedit`, `kate`, `leafpad` ou outros. Porém, o `vim` tem a vantagem de se poder usar quase da mesma forma em ambientes gráficos e em consolas.

Todos estes editores possuem a função de realce de sintaxe, muito útil para programadores. Ou seja, sempre que editar um texto numa linguagem de programação ou de representação

conhecida, o editor usa cores diferentes para realçar blocos sintáticos distintos, de modo a facilitar a sua leitura e análise.

Para programação, também existem ambientes integrados de desenvolvimento (IDE) que além de um editor também incluem ferramentas de compilação, execução e depuração. Um IDE simples e versátil é o **geany**.

Exercício 2.17

Edite um ficheiro e escreva 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 alguns comandos do modo **ed**.

2.5.1 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).[10] 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 a aplicação desloca o cursor até ao texto encontrado. Para encontrar a ocorrência seguinte, usa-se o comando `n`. Para inverter o sentido e encontrar a ocorrência anterior, usa-se 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.

[10] Num teclado americano estes símbolos estão na mesma tecla física e num local muito conveniente.

Exercício 2.20

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

Exercício 2.21

Leia a página de manual do comando **`less`** para descobrir como se consegue saltar para o início e para o fim do texto e experimente esses comandos. Tenha em consideração que a deslocação para um ponto do texto é referida através da expressão "Go to" no manual.

2.6 Processos

Em *Linux* existe uma máxima que diz que tudo é um ficheiro ou um processo. Até agora 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 de texto que lhe é apresentado, assim como a *Shell* e muitos outros programas estão a executar simultaneamente. Um programa executado várias vezes dá origem a vários processos. No sistema, cada processo é identificado por um número chamado de Process IDentifier (PID).

Pode utilizar o comando **`ps`** para verificar os processos da sua sessão, ou combinar com as opções **`-ax`** para ver que processos estão activos, independentemente do utilizador. Se adicionar a opção **`-u`** irá igualmente ver 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 2.7), é mostrado o utilizador, o PID, a quantidade de processador e memória utilizados, o seu estado, há 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 2.7: Resultado da execução do comando **`ps Tu`** para mostrar os processos associados com a sessão actual.

Exercício 2.22

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 processes activos?

Exercício 2.23

Repita o exercicio anterior utilizando o comando **top**. Pode utilizar as teclas < e > para seleccionar qual a coluna escolhida para ordenar os processos. Use **q** para terminar.

Um aspecto importante dos processos é a sua gestão, nomeadamente a possibilidade de controlar o seu estado de execução e, eventualmente terminá-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 envia um sinal a todos os processos com um determinado nome:

```
killall [-sinal] nome-do-processo
```

Se não se indicar o sinal, é enviado o sinal **SIGTERM**, que geralmente provoca a terminação do processo de forma relativamente segura. No entanto, em certas condições, um processo pode ficar num estado em que não reage ao sinal. Nesse caso, pode usar-se o sinal **SIGKILL** ou **9**, que não pode ser ignorado, mas corre-se maior risco de perder dados que não tenham sido gravados. Existem sinais com outras funcionalidades. Pode consultar uma lista com o comando **man 7 signal**.

Exercício 2.24

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

2.7 Administração básica

(Pderá saltar esta secção numa primeira leitura.)

Com certeza já reparou que muitos diretórios e ficheiros do sistema pertencem a um utilizador chamado `root`, e só ele tem permissão de os modificar. Isto impede que um erro de outro utilizador possa destruir o sistema, mas também implica que só o `root`, também chamado de *super-utilizador* ou *administrador* do sistema, consegue fazer certas tarefas como instalar software nos diretórios do sistema ou registar novos utilizadores, por exemplo.

Nos sistemas modernos é usual permitir que um ou mais utilizadores do sistema possam usar o comando `sudo` para assumir temporariamente o papel de super-utilizador e assim realizar tarefas de administração. O primeiro utilizador criado numa instalação de *Linux* tem geralmente essa permissão.

Como é óbvio, nos computadores da sala de aula, um utilizador normal não tem essa permissão, pelo que não poderá fazer estas tarefas de administração. Por esta razão, os exercícios desta secção terão de ser feitos numa máquina diferente, onde tenha poderes de administrador.

Uma hipótese é usar uma *máquina virtual*: um computador simulado dentro de outro. Estudaremos máquinas virtuais noutra aula, mas por agora pode usar uma máquina virtual que preparámos para usar no computador da sala. Para isso:

1. extraia o conteúdo do arquivo `/usr/local/labi/labi-vb-setup.tbz2` para a sua pasta pessoal;

2. extraia o conteúdo do arquivo `/usr/local/labi/labi-lubuntu.vdi.tbz2` para a sua pasta pessoal (isto poderá demorar um bocado);

3. execute o programa VirtualBox;

4. na janela do VirtualBox, escolha Start.

Deve aparecer uma janela onde arranca o seu computador virtual. Faça login com o utilizador `labi` e password `labi`. Este utilizador tem permissões de administração e permite-lhe avançar para os exercícios das sub-secções seguintes.

2.7.1 Gestão de utilizadores

O sistema *Linux* assume uma gestão baseada em utilizadores e grupos. 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 que além de utilizadores reais, como é o caso do utilizador **`labi`**, também há *utilizadores de sistema* que servem apenas para confinar aplicações a definições de segurança específicas. Por exemplo, um processo que serve páginas Web pertence a um certo utilizador de sistema para ter acesso aos ficheiros das páginas que serve, mas não aos ficheiros de qualquer outro utilizador.

Pode verificar qual o seu utilizador através do comando **`whoami`**, ou em mais detalhe através do comando **`id`**.[11] O resultado deverá ser o demonstrado na Figura 2.8. Experimente o mesmo comando no computador da sala (fora da máquina virtual).

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

Figura 2.8: Resultado da execução do comando **`whoami`**

Exercício 2.25

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

Na máquina virtual, o utilizador **`labi`** consegue assumir temporariamente as funções de super-utilizador (**`root`**) através do comando **`sudo`**.

Na primeira vez que um utilizador usa o comando **`sudo`**, é pedida a sua palavra passe para confirmação. Execuções seguintes na mesma sessão "lembram-se" da validação durante alguns minutos e não a pedem de novo.

A Figura 2.9 demonstra a utilização do comando **`sudo`** nas suas duas formas mais comuns. Na primeira forma, (**`sudo whoami`**), o comando **`whoami`** é executado como super-utilizador,

[11] O comando **`id`** irá mostrar o utilizador e todos os grupos aos quais ele pertence

mas o comando **whoami** seguinte é novamente executado pelo utilizador **labi**. A segunda forma, **sudo -s**, cria uma nova *Shell* no contexto do super-utilizador (repare no prompt). Todos os comandos inseridos serão executados com os privilégios de **root**, até que se termine esta shell com **exit** ou **Ctrl-D**.

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

Exercício 2.26

Verifique se possui permissões para visualizar (**cat**) o ficheiro **/etc/shadow**. Se não possuir, visualize o seu conteúdo recorrendo ao comando **sudo**.
Consegue perceber para que serve este ficheiro? Pode recorrer ao comando **man**.

Na verdade, os utilizadores não são tratados internamente pelo sistema operativo com os nomes que temos utilizado (ex., **labi**). Na realidade tanto os utilizadores como os grupos existentes são mapeados para números. São esses números que se vê no reultado do comando **id**, tal como representado na Figura 2.10.

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

Figura 2.10: Resultado da execução do comando **id**, inspeccionando qual o utilizador em uso (neste exemplo o utilizador tem o nome **linux**).

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 pelo sistema para controlar as permissões. Os nomes são utilizados para facilitar a gestão aos administradores (humanos).

Exercício 2.27

Verifique qual o `uid` e `gid` do utilizador `root`.

Exercício 2.28

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.

2.7.2 Gestão de Aplicações

Uma característica importante das distribuições de *Linux* é o facto de utilizarem um sistema integrado para a pesquisa, instalação, actualização e remoção de aplicações. É um conceito semelhante (e anterior) ao das lojas de aplicações tipicamente disponíveis nos telemóveis actuais. As grandes diferenças é que as aplicações disponibilizadas por estes meios são geralmente de utilização livre e gratuita, 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 podem variar dependendo da distribuição utilizada.

No caso das distribuições que derivam da distribuição *Debian*[12], a gestão de aplicações é realizada através da família de comandos `apt-`, ou através de aplicações mais intuitivas como o `synaptic`, o `aptitude`, ou mesmo o `Ubuntu Software Center`.

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`.

Os principais comandos a utilizar são:

[12]Para ver como se organizam as distribuições, consultar `http://futurist.se/gldt/`.

- **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. Este processo é importante pois permite obter actualizações para as aplicações instaladas. Este processo de actualização deve ser repetido periodicamente e geralmente os sistemas já vêm configurados para o fazerem de forma automática.

Exercício 2.29

Utilizando o comando **apt-get update** actualize a base de dados de aplicações. (Lembre-se que precisa de correr o comando dentro de um **sudo**.)
Utilizando o comando **apt-get upgrade** actualize as aplicações do sistema.

Os comandos mais usuais para gerir a instalação de uma nova aplicação, por exemplo a aplicação **cowsay**, são os seguintes (ver Figura 2.11):

1. **apt-cache search cowsay** — Procura pelo nome e indica pacotes relacionados.
2. **apt-get install cowsay** — Instala o pacote **cowsay**, que tem a aplicação.
3. **apt-get remove cowsay** — Para remover a aplicação, se já não a quisermos.

Exercício 2.30

Instale e verifique o funcionamento das aplicações **cowsay** e **fortune**.

2.8 Para aprofundar o tema

Exercício 2.31

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

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

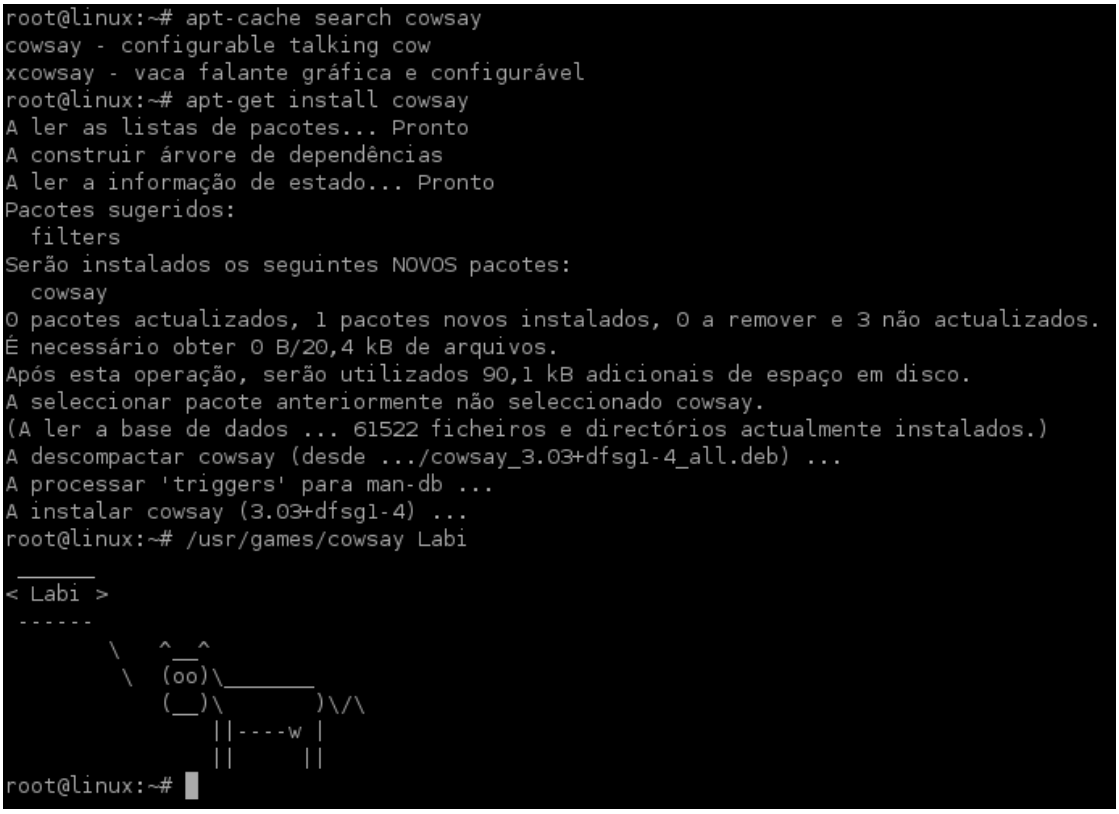

Figura 2.11: Processo de instalação da aplicação **cowsay**

Exercício 2.32

Explore a utilização dos operadores "**|**" e "**>**".
Por exemplo:

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

ou

```
ls -la |wc
```

Glossário

CD Compact Disk

CLI	Command Line Interface
DVD	Digital Versatile Disk
ISO	Imagem de Arquivo de CD
PID	Process IDentifier
PC	Computador Pessoal