Licenciatura em Engenharia Informática

UC: Sistemas Operativos

Ficha 2 - O sistema operativo Linux

O Linux na perspetiva do utilizador (continuação)

Tópicos abordados:

– Substituição de comandos & utilitário xargs

– Comandos de manipulação de texto

– Comandos de extração de texto

– Comandos de *hashing* e assinaturas

– Codificação de ficheiros

– Extração de conteúdos em páginas *web*

– Editores de texto no Linux

– Programação em *bash*

– Exercícios

Duração prevista: 2 aulas

1 Substituição de Comandos e *xargs*

A substituição é uma forma de executar um comando como parâmetro de outro comando, ou seja, outra forma da utilização de *pipes*. Existem duas formas de o fazer: i) plicas invertidas e ii) $(comando).

1.1 Plicas invertidas[1] (`comando`)

```
$ comando1 `comando2 [-opções]`
```

O comando2 é executado e o seu resultado é o parâmetro de entrada do comando1.

[1] *Backtick* na designação Anglo-Saxónica.

Exemplos:

$ ls -la `which ps`

Mostra a listagem longa do ficheiro do comando *ps*.

$ id `whoami`

Mostra a informação sobre o utilizador que efetuou o *login*.

1.2 $(comando)

Este operador é a forma mais atual de substituição de comandos e também a mais legível.

$ comando1 **$(**comando2 [-opções]**)**

O comando2 é executado e o seu resultado é o parâmetro de entrada do comando1.

Exemplos:

$ ls -la $(which ps)

Mostra a listagem longa do ficheiro do comando *ps*.

$ id $(whoami)

Mostra a informação sobre o utilizador que efetuou o *login*.

Exercício 2a-1

Indique a linha de comando que permite a criação de um diretório com a data corrente no formato **ano.mes.dia**. Por exemplo, considerando que se está no dia 10 de março de 2019, o diretório a criar será **2019.03.10**.

Exercício 2a-2

O utilitário file devolve informação a respeito do tipo de ficheiro (executável, texto, imagem, etc.).

a) O que sucede quando executa: `file /bin/ls`?

b) O que sucede quando executa: `file $(find /etc)`?

c) O que sucede quando executa: `file $(find / )`?

d) Qual a diferença entre **`find /`** e **`find / 2> /dev/null`**?

1.3 Utilitário xargs

O utilitário **xargs** constrói, em tempo de execução, uma única linha de comando, fazendo uso do conteúdo que lhe é passado pela entrada padrão. Para cada elemento que recebe na entrada padrão, o **`xargs`** adiciona-o, como argumento, à linha de comando construída. Assim, a execução do utilitário `file` para todos os ficheiros do diretório `/etc` pode ser efetuada da seguinte forma:

```
find /etc | xargs file
```

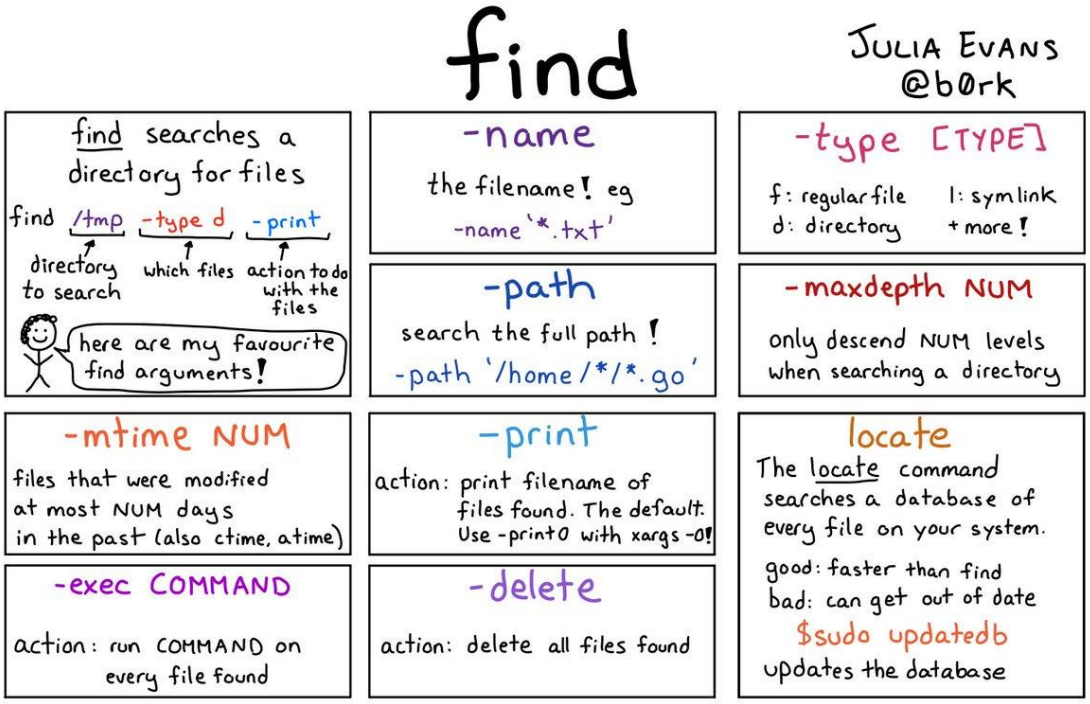
Neste caso, o **xargs** para cada nome ("NOME") de ficheiro que recebe da saída do **`find /etc`**, constrói a linha de comando **`file NOME1 NOME2`** e procede à respetiva execução.

Exercício 2a-3

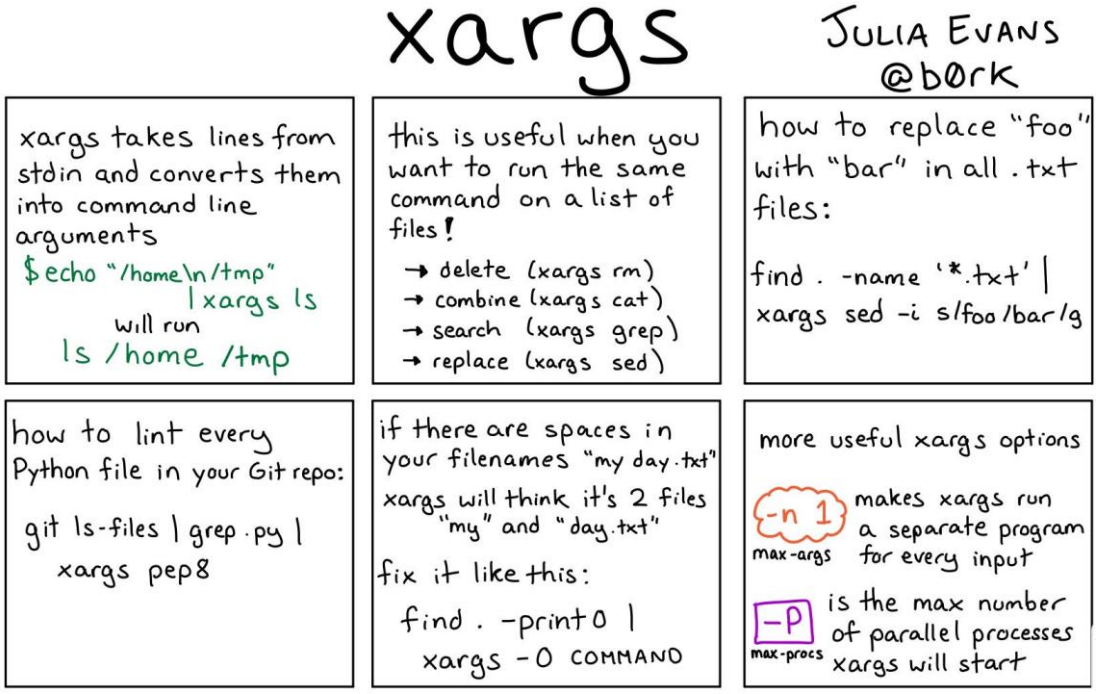
O que sucede quando executa a seguinte linha de comando?

```
find / 2> /dev/null | xargs file
```

1.4 Caso de uso: find



1.5 Caso de uso: xargs



2 Comandos de manipulação de texto

2.1 cut

$ **cut** <opt> ficheiro

Permite cortar partes de um ficheiro ou do conteúdo passado na entrada padrão (*stdin*).

Opções:

[-f] Especifica os campos a devolver;

[-d] Especifica o delimitador de campo (por omissão é o TAB);

[-c] Caracteres a devolver.

[-b] Byte subgroup

Exemplo:

```
$ cut -f1 -d: /etc/passwd
```

Mostrar todos os *logins* registados na máquina (campo número 1)

```
$ cut -b 8-16  /var/log/auth.log
```

Extrai a hora (baseado em bloco de caracteres) dos registos do ficheiro de log auth.log

Exercício 2b

Indique o comando que permita mostrar o caminho completo das diretorias *home*, de todos os utilizadores, registados no sistema.

2.2 Paste

$ **paste** <opt> ficheiro1 ficheiro2 … ficheiroN

Junta o conteúdo de ficheiros (utiliza a 1ª linha do "ficheiro1", a 1ª linha do "ficheiro2", ..., a 1ª linha do "ficheiroN" e junta-as numa só, com os campos separados por TAB (por omissão).

Opções:

-d Especifica o delimitador de campo (por omissão é o TAB);

-s Processa ficheiros de forma sequencial (e não em paralelo).

Exercício 2c

a) Execute os seguintes dois comandos, explicando a saída final.

```
$ cut -f1 -d: /etc/passwd > f1.txt; cut -d: -f6 /etc/passwd > f6.txt
$ paste -d: f1.txt  f6.txt
```

b) Execute os seguintes comandos, explicando a saída.

```
$ seq 10 > 10.txt; paste 10.txt 10.txt 10.txt
$ seq 10 > 10.txt; paste -s 10.txt 10.txt 10.txt
```

2.3 Translate (tr)

$ **tr** <opt> conjunto1 conjunto2

O comando **tr** (translate) permite a conversão de caracteres do "conjunto1" para o "conjunto2". Dado que, por omissão, o **tr** manipula a entrada padrão (*stdin*) e a saída padrão (*stdout*), torna-se necessário recorrer a redirecionamento de entrada/saída caso se pretenda efetuar a operação em ficheiros.

Opções:

[-d] (*delete*) apaga caracteres em vez de proceder à substituição;

[-s] (*squeeze*) elimina repetições.

Exemplos:

```
$ tr 'a' 'A'  < input.txt > output.txt
```

Converte todos os 'a' em "A", que possam existir no ficheiro "input.txt" e guarda o resultado no ficheiro "output.txt"

```
$ tr 'a-z' 'A-Z' < input.txt > output.txt
```

Converte todos as minúsculas em maiúsculas que possam existir no ficheiro "input.txt" e guarda o resultado no ficheiro "output.txt"

NOTA: não se deve usar '[' e ']' com o comando `tr`, exceto quando se quer manipular esses caracteres. O `tr` interpreta os caracteres '[' e ']' de forma idêntica a qualquer outro caracter.

Exercício 2d

Execute os seguintes comandos, explicando a saída.

```
$ cat /etc/passwd  |  tr ':' '|'
$ ps -ef  |  tr -s ' ' '#'  |  cut -f1,2 -d'#'
$ seq 10  |  tr -s '\n' ','
```

3 Comandos de extração de texto

3.1 grep

$ **grep** [opt] <expressão> [<ficheiro>]

Procura expressões (palavras, etc.) num ficheiro.

Opções:

[-i] Ignora maiúsculas e minúsculas;

[-c] Número de linhas que verificam a condição;

[-n] Devolve os números das linhas (bem como as próprias linhas) que verificam a condição;

[-v] Inverte a pesquisa, exibindo apenas as linhas que não verificam a condição;

[-A n] Mostra "n" linhas após a linha em que a expressão foi encontrada e coloca "--" a separar as "n" ocorrências;

[-B n] Mostra "n" linhas Antes da linha em que a expressão foi encontrada e coloca "--" a separar as "n" ocorrências;

[-m n] Mostra as primeiras "n" ocorrências da expressão que se pretende encontrar.

[-e padrão] Pesquisa padrão

[-o] Pesquisa padrão, devolvendo apenas as ocorrências (não a linha toda) em linhas separadas.

Exemplos:

```
$ ps -ef | grep root
# devolve as linhas de "ps" que contém a palavra "root"
$ ps -ef | grep $(whoami)
# devolve as linhas de "ps" que contém o login de quem executa o comando
```

Exercício 2e

Execute os seguintes comandos, explicando a saída.

```
$ echo $$
$ ps -ef | grep $$
$ ps -ef | grep -o $(whoami)
```

3.1.1 Expressões regulares básicas

Em Unix (e em muitos outros ambientes), as expressões regulares (muitas vezes designadas por "*regex*" na literatura anglo-saxónica) são uma forma poderosa de definir padrões. Iremos de seguida abordar somente o simples mecanismo de "âncoras".

Âncoras: permitem indicar que o padrão, a pesquisar se encontra no início ou no fim.

O **grep** (e outros comandos com suporte para *regex*) suporta as seguintes âncoras:

Opções:

[^] Início de linha;

[$] Fim de linha.

Exemplos:

```
$ ps -ef | grep ^root
# devolve as linhas do "ps -ef" que **comecem** por "root"

$ seq 100 | grep 0$
# devolve os números de 1 a 100 que **terminam** em 0
```

Exercício 2f

Utilizando o ficheiro "/etc/passwd":

a) Mostre os utilizadores do sistema operativo cujo login começa por "r";

b) Diga quantos utilizadores têm *bash* como *Shell*.

4 Utilitários para "*message digest*"

A designação "*message digest*" (algoritmos de *hash*) refere-se à criação de um identificador unívoco (muitas vezes em formato hexadecimal) para a representação do conteúdo de um ficheiro. Existem vários algoritmos de *hash*, desde o MD5 (*Message Digest*) e o SHA (*Secure Hash Algorithm*). Para além da aplicação em criptografia, os *messages digests* podem ser empregues para a deteção de ficheiros idênticos. Em Linux existem os seguintes utilitários para o cálculo de *message digests*:

- md5sum
- sha1sum
- sha224sum, sha256sum, sha384sum, sha512sum

Exemplos:

```
$ who > who.txt
# criar um ficheiro "who.txt" para usar nos próximos comandos

$ md5sum who.txt
# calcula o "md5sum" do conteúdo do ficheiro "who.txt"

$ md5sum $(ls)
# calcula o md5sum para cada um dos ficheiros do diretório corrente

$ ps -ef | sha1sum
# calcula o sha1sum da saída produzida pelo comando "ps -ef"
```

Exercício 2g

Utilizado o *link* https://download.gnome.org/sources/gedit/3.22/, descarregue um ficheiro à sua escolha e verifique a integridade do ficheiro com os seguintes algoritmos MD5, SHA1 e SHA256. Apenas deve usar uma linha de comando.

Sugestão: uso de ";" para separar comandos.

5 Codificação de ficheiros

5.1 Codificação base64

A codificação base64 é empregue para transmitir conteúdo binário (e.g., ficheiro de imagem) por canais que apenas suportam texto em ASCII. Por exemplo, as imagens (e outros ficheiros binários) enviadas por email, são previamente codificadas em base64 antes do seu envio, sendo descodificadas para o formato original aquando da leitura do email. Note-se que, a codificação para base64 acarreta um aumento dos dados na ordem dos 33%.

No Linux, o utilitário `base64` permite a codificação/descodificação de conteúdo para/de base64.

$ **base64** [<opt>] ficheiro

Permite codificar ou descodificar, em base64, um ficheiro passado como parâmetro.

Opções:

[-d] (*decode data*) descodifica o ficheiro;

[-i] Ao descodificar, ignora caracteres não-alfabéticos.

Exemplos:

```
$ base64 logo_estg.png > logo_estg.base64
# Codifica o ficheiro (imagem) "logo_estg.png" e guarda o resultado no ficheiro "logo_estg.base64"

$ base64 -d logo_estg.base64 > logo_estg_decoded.png
# Descodifica o ficheiro "logo_estg.base64" e guarda o resultado no ficheiro (imagem) "logo_estg_decoded.png"
```

Exercício 2h

a) Crie o ficheiro `ps.ef.txt` que deve corresponder à saída padrão do comando `ps -ef`

b) Qual é o tamanho do ficheiro `ps.ef.txt`?

c) Recorrendo ao utilitário base64, codifique o ficheiro `ps.ef.txt` para base64, guardando o ficheiro codificado no ficheiro `ps.ef.txt.base64`

d) Qual é o tamanho do ficheiro `ps.ef.txt.base64`?

e) Visualize o conteúdo do ficheiro `ps.ef.txt.base64`, por exemplo, através do utilitário cat.

6 Extração de conteúdos em páginas “html”

Muitas vezes torna-se necessário a extração de conteúdos em páginas Web que se encontram em formato HTML, sendo essa operação designada por *web scraping*. A extração da informação pretendida faz-se muitas vezes com recurso a heurísticas, sendo o conteúdo descarregado através de um cliente HTTP que funcione em modo de texto. Para o efeito iremos considerar dois clientes HTTP: *wget* e o *curl*.

6.1 Utilitário wget

$ wget [opt] url

descarrega (*download*) conteúdos online via http, ftp ou https

O utilitário *wget* permite o acesso a conteúdos *online* (através de http, https e ftp).

Opções:

[-c] Continua operação de descarregamento (anteriormente interrompida);

[-O <ficheiro>] Grava conteúdo para “ficheiro”.

Exemplos:

$ wget http://www.estg.ipleiria.pt

Obtém e grava o conteúdo da página inicial de http://www.estg.ipleiria.pt

$ wget https://distrowatch.com/ -O osstats.html

Obtém e grava para “osstats.html” o conteúdo do URL https://distrowatch.com/

6.2 Utilitário curl

$ curl [opt] url

descarrega (*download*) ou transfere (*upload*) conteúdos online

O utilitário *curl* permite obter ou transferir conteúdos *online* (através de: DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMTP, SMTPS, TELNET). Este comando permite ser utilizado com ou sem interação com o utilizador.

Opções:

[-d <dados>] Envia dados via POST;

[-o <ficheiro>] Grava conteúdo para “ficheiro”.

Exemplos:

```
$ curl -o google.txt http://www.google.pt
```

Obtém e grava para "google.txt" o conteúdo do URL http://www.google.pt

6.2.1 Extração de conteúdo

A extração de conteúdo de páginas HTML – vulgo *web scrapping* – depende da "formatação" por que estas são elaboradas. Mas, conhecendo essa formatação é fácil obter a informação pretendida. Um exemplo para a extração da informação pode ser a obtenção das varáveis ambientais disponibilizadas na seguinte página: https://meteo.tecnico.ulisboa.pt/forecast/resume

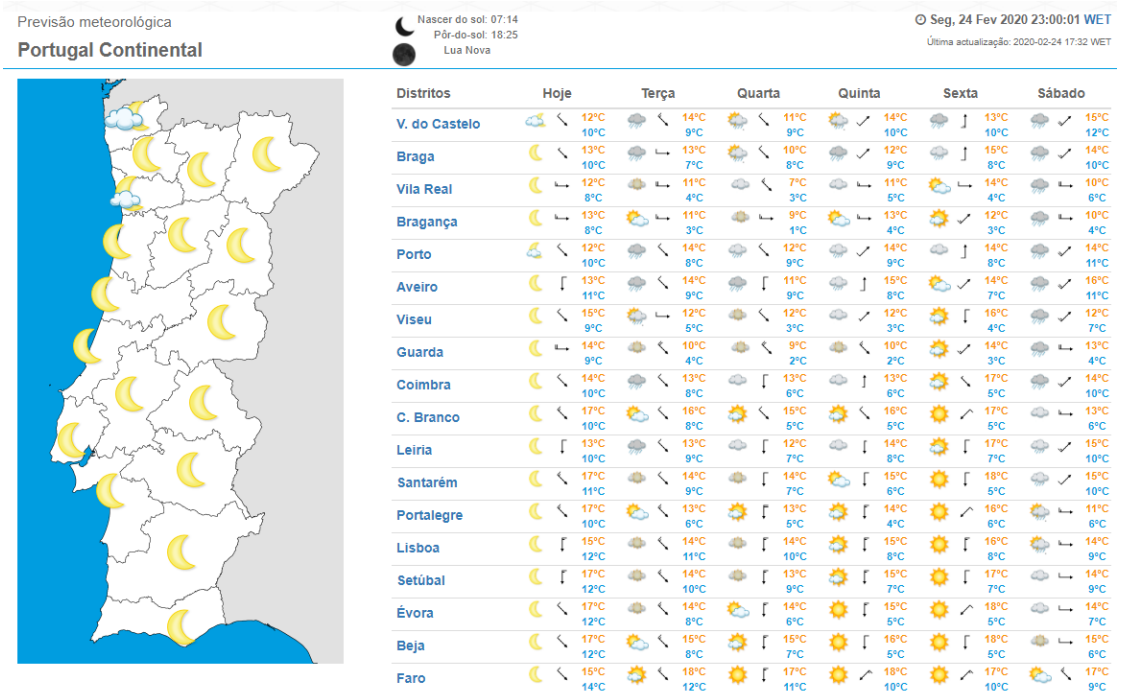


Figura 1 - sítio https://meteo.tecnico.ulisboa.pt/forecast/resume

Com os comandos de extração de texto é possível obter apenas partes do ficheiro HTML de resposta.

Por exemplo, se quiséssemos obter a temperatura máxima prevista para Leiria para o dia corrente, poder-se-ia executar a seguinte linha de comando:

```
curl -ks https://meteo.tecnico.ulisboa.pt/forecast/resume | grep -A2 'a title="Leiria"' | tail -n 1 | cut -d':' -f6 | cut -d'.' -f1 | tr -d ' '
```

No utilitário curl, a opção **`-s`** omite as mensagens de erro e a barra de progresso. Por sua vez, a opção **`-k`** ignora problemas que possam existir ao nível de certificados digitais necessários numa ligação HTTPS.

O mesmo resultado pode ser obtido usando primeiro o comando wget para guardar o ficheiro HTML.

```
$ wget -O meteo.html https://meteo.tecnico.ulisboa.pt/forecast/resume
```

Obtém e grava o conteúdo da páginade entrada no sítio "meteo" para o ficheiro "meteo.html"

```
$ grep -A2 'a title="Leiria"' meteo.html | tail -n 1 | cut -d':' -f6 | cut -d'.' -f1 | tr -d ' '
```

Obtém a temperatura máxima prevista para Leiria

Exercício:

O processo de *web scraping* é usualmente feito de forma sequencial. Para uma plena compreensão do processo de extração de dados, execute as seguintes linhas de comandos e interpretar os resultados

```
wget -O meteo.html https://meteo.tecnico.ulisboa.pt/forecast/resume
```

i)
```
cat meteo.html | grep -A2 'a title="Leiria"' meteo.html
```
 # Isola o conteúdo associado à palavra Leiria

ii)
```
cat meteo.html |grep -A2 'a title="Leiria"' | tail -n 1
```
 # Fica apenas com a última linha do conteúdo

iii)
```
cat meteo.html |grep -A2 'a title="Leiria"' | tail -n 1 | cut -d':' -f6
```
 # Usa o separador ":" para obter temperatura máxima

iv)
```
cat meteo.html |grep -A2 'a title="Leiria"' | tail -n 1 | cut -d':' -f6 | cut -d'.' -f1
```
 # Isola a temperature máxima

v)
```
cat meteo.html |grep -A2 'a title="Leiria"' | tail -n 1 | cut -d':' -f6 | cut -d'.' -f1 | tr -d ' '
```
 # Remove espaços que possam existir no resultado

Exercício 2i

Recorrendo ao sítio https://meteo.tecnico.ulisboa.pt/forecast/resume e a técnicas de web scraping, obtenha os seguintes elementos:

a) A temperatura mínima prevista para Leiria para o dia corrente

b) A hora do nascer do sol

c) A data/hora da última atualização do sítio

7 Editores de texto no Unix

De seguida serão mostrados alguns dos editores de texto, que podem ser utilizados para programar em *bash* ou *C*.

7.1 Editor "vi"

Uma das ferramentas mais usadas em Linux para editar texto, é o "vi". Presentemente é mais comum o uso da versão avançada, designada de "VIM" (*VI improved*)

```
$ vi <nome_ficheiro>
```

O *vi* tem dois modos de funcionamento: modo de comando, em que as teclas pressionadas correspondem a comandos e o modo de edição propriamente dito. Para passar ao modo de comando é necessário premir a tecla "Esc". Para passar a modo de edição, basta utilizar um comando como o "i" ou o "a" que ative este modo. O modo de comando disponibiliza ainda o modo de *última linha*, em que os comandos aparecem na última linha. São exemplos de comando de última linha, os comandos que se iniciam por ":". De seguida são apresentados, alguns dos muitos comandos possíveis:

:q	Sair sem gravar (desde que não tenha sido feita nenhuma alteração)
:q!	Sair sem gravar (caso tenha sido feita alguma alteração no ficheiro, a opção :q não é suficiente)
:x	Sair e gravar o ficheiro
:w	Gravar
:w ficheiro	Gravar com um determinado nome
I	Passar a modo de edição (inserir texto antes do cursor)
A	Passar a modo de edição (inserir texto depois do cursor)
x	Apagar letra
dd	Apagar linha
H	Mover o cursor para a esquerda
J	Mover o cursor para baixo
K	Mover o cursor para cima
L	Mover o cursor para a direita
:N	Vai para a linha N (N deve ser um número)
Yy	Copia a linha onde se encontra o cursor
y$	Copia desde o cursor até ao fim da linha
Nyy	Copia N linhas para baixo do cursor (N deve ser um número)
P	Cola informação copiada
u	Undo
/palavra	Procurar palavra
N	Procurar a próxima palavra

Nota: Mais informação em: http://vimdoc.sourceforge.net/

7.2 Editor “pico”

Outro editor, em modo texto, é o “pico”.

```
$ pico <nome_ficheiro>
```

Na próxima figura estão apresentados os principais atalhos:

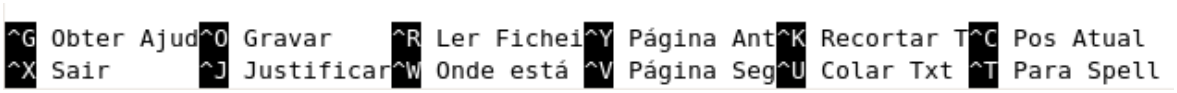


Figura 2 - Menu do editor “pico”

7.3 gedit

Para abrir o editor de texto “Gedit” na linha de comandos:

```
$ gedit <nome_ficheiro>
```

Ou, através da interface gráfica:

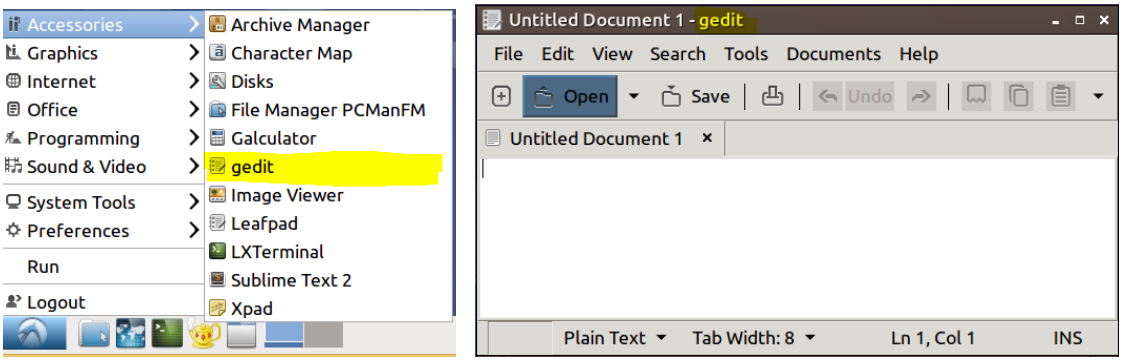


Figura 3 - Editor “Gedit”

7.4 Geany

Para abrir o editor de texto “Geany” na linha de comandos:

```
$ geany <nome_ficheiro>
```

Ou, através da interface gráfica:

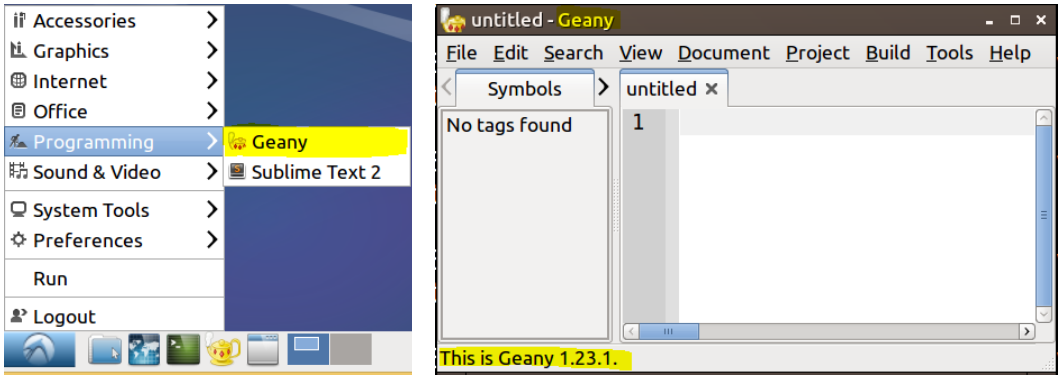


Figura 4 - Editor “Geany”

7.5 Sublime

Para abrir o editor de texto “Sublime” na linha de comandos:

```
$ sublime-text <nome_ficheiro>
```

Ou através da interface gráfica:

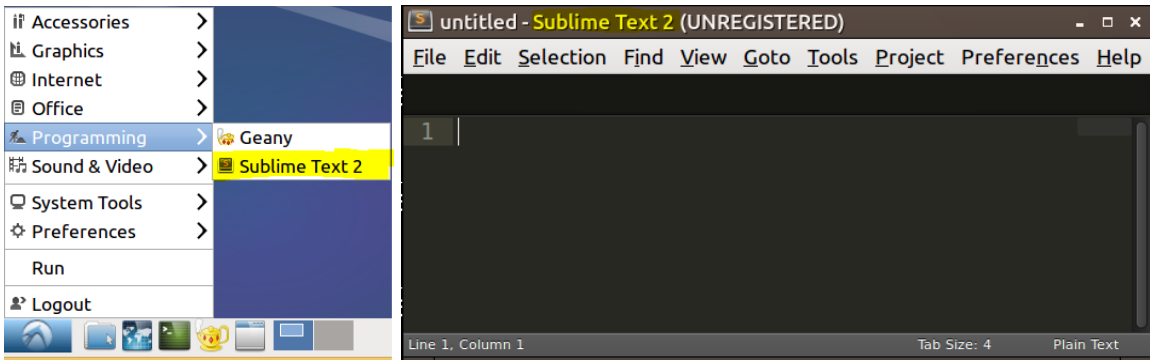


Figura 5 - Editor “Sublime”

Por não se tratar de software Open Source, o editor Sublime não faz parte dos repositórios base do Ubuntu. Para instalar o Sublime é necessário adicionar um repositório extra. As instruções encontram-se no site oficial:

https://www.sublimetext.com/docs/3/linux_repositories.html

8 Programação em Bash

A *bash* é um programa, interpretador de comandos, que nos permite interagir (através da *shell*) com o sistema o sistema operativo. Todos os comandos que foram mostrados até ao momento são exemplos disso. Existem duas formas de poder executar esses comandos:

1. Escrevendo diretamente os comandos na *shell*, como se tem feito até agora;
2. Criando ficheiros com um ou vários comandos, que formam uma sequência de comandos, podendo esta sequência ser um programa com o objetivo de realizar determinada tarefa, como é exemplo a criação de utilizadores. A este tipo de ficheiro dá-se o nome de *script* (conjunto de instruções que são interpretadas e executadas sequencialmente).

Este capítulo mostra como abordar a segunda opção: *scripts* em *bash*. De seguida será mostrada, de forma muito resumida, como criar e executar *scripts*, mostrando-se também como se pode controlar o fluxo de um programa, recorrendo a estruturas de decisão e repetição.

8.1 Estrutura de um ficheiro bash

Para criar um programa em *bash* é necessário:

Criar um ficheiro de texto. A primeira linha desse ficheiro deverá ser (obrigatoriamente):

```
#!/usr/bin/env bash
```

A linha anterior é normalmente apelidada de "shebang"[2].

O "código *bash*" (comandos) do script é escrito nas linhas seguintes. Exemplo:

```
echo "olá mundo!!!"
```

Gravar o ficheiro. Neste ponto deve-se ter em atenção que para identificar os ficheiros de *script* em *bash* a extensão do ficheiro deve ser ".sh". Exemplo:

```
first.sh
```

Temos, também, que garantir que este ficheiro tem a permissão de execução, ou seja, que o utilizador tem a permissão "x" associada. Para isso deve-se alterar a permissão do ficheiro. Exemplo:

```
chmod u+x first.sh
```

Executar o ficheiro, ou seja, chamar o ficheiro na linha de comandos:

```
./first.sh
```

E ver o resultado da execução do programa:

```
"Olá mundo !!!"
```

[2] Como alternativa (menos portável), poderá utilizar a shebang #!/bin/bash

Exercício 2j

Execute os passos anteriores e teste a execução do *script*.

8.2 Declaração de variáveis

Para declarar variáveis não é necessário definir tipos. Podem-se usar para os nomes das variáveis números e letras. Para aceder ao valor da variável é necessário utilizar o símbolo "$".

Para exemplificar a utilização de variáveis vamos utilizar o exemplo anterior:

```
#!/usr/bin/env bash
var="olá mundo!!!"
echo $var
```

Nota: não devem existir espaços entre o operador de atribuição (símbolo =) e qualquer um dos operandos, isto é, entre o valor atribuído (lado direito) e a variável a ser atribuída (lado esquerdo).

Exercício 2l

Execute o programa anterior.

8.3 Variáveis de ambiente

Para utilizar as variáveis de ambiente existentes basta chamá-las de "forma normal", isto é, pelo seu nome. Alguns exemplos de variáveis de ambiente comuns são:

- $USER – Nome do utilizador atual
- $PATH – Caminho para as diretorias onde estão os comandos do sistema operativo;
- $HOME – Caminho da *home directory* do utilizador.

Exercício 2m

a) Crie um *script* que mostre o valor das variáveis mostradas anteriormente.

b) Execute o comando "env", comentando a saída do mesmo.

8.4 Ciclo “for”

Para poder efetuar ciclos existem várias estruturas, mas apenas se apresenta a estrutura “for”. A sintaxe é:

```
for var in lista; do
    comandos_a_repetir
done
```

Um exemplo da utilização deste ciclo pode ser mostrar uma lista de 5 elementos:

```
for i in {1..5}; do
   echo $i
done
```

Outro exemplo, para a conversão de todos os ficheiros de imagem “jpg” de uma diretoria:

```
cont=0
for file in *.jpg; do
    echo $file
    let cont=$cont+1  # ou: cont=$((cont+1))
    S=$(printf "%s_1920x_%02d.jpg" $file $cont)
    echo $S
    convert -resize 1920x $file $S
done
```

Nota: o utilitário *convert* faz parte do conjunto de software *imagemagick* que permite, via linha de comando, o processamento de imagens. No caso em apreço, o *convert* efetua o redimensionamento para o tamanho 1920x (valor horizontal, o valor vertical é calculado pelo *convert*, por forma a manter o *aspect* ratio) da imagem que lhe é indicada na linha de comando como primeiro parâmetro da linha de comando. Caso necessite de efetuar a instalação do *imagemagick*, deverá recorrer ao comando **sudo apt install imagemagick**.

Exercícios:

1. Transcreva a listagem acima indicada para o ficheiro **resize.sh**. De seguida, crie o diretório *ImagemJPG* e coloque nesse diretório duas imagens em formato JPEG e cujos respetivos nomes terminem em **.jpg**. Poderá efetuar o download de imagens de teste a partir do endereço http://lorempixel.com/ (e.g. `wget -O imagem1.jpg http://lorempixel.com/400/200/`). Efetue ainda uma cópia do ficheiro **resize.sh** para o diretório *ImagemJPG*. Seguidamente, execute o programa **resize.sh** da seguinte forma: **bash resize.sh**. O que sucedeu?
2. Execute os comandos indicados na listagem anterior, mas recorrendo apenas à linha de comando, isto é, o conteúdo de **resize.sh** deve ser colocado numa única linha de comando (sugestão: uso do separador “;” para separar múltiplos comandos)

9 Exercícios

1. Utilizado o ficheiro que se encontra em http://tinyurl.com/zfsbfu2, obtenha:
   a) A temperatura máxima;
   b) A temperatura média;
   c) A precipitação entre as 0:00 e as 6:00.

2. Indique uma linha de comando, que permita calcular o "md5sum" de todos os ficheiros do diretório corrente, cujo nome termina por "**.txt**". Os resultados devem ser guardados, no ficheiro "YYYYMMDD_DiaSemana.txt", em que YYYY representa o ano, MM o mês (01 a 12), DD o dia do mês (01 a 31) e DiaSemana o dia da semana (0 a 6 [domingo a sábado]).

3. Utilizado o link http://ipinfo.io/, obtenha informação sobre:
   a) O IP público que está a utilizar;
   b) A cidade e código postal do IP da Google (http://ipinfo.io/8.8.8.8).

   Atenção:

   Informação retirada de: http://ipinfo.io/developers

   "You are limited to 1,000 API requests per day. If you need to make more requests, or need SSL support, see our paid plans."