Sistemas Operativos 2019-20
Introdução a programação em *shell*
LEIC-A / LEIC-T / LETI
IST

As *shells* representam uma das principais interfaces para os sistemas operativos. Ao longo do tempo têm sido desenvolvidas em ambientes UNIX/Linux várias *shells*, e.g. Bourne shell (sh), Bourne Again shell (bash), Korn shell (ksh), e C shell (csh), que oferecem diferentes linguagens de *scripting*. Neste guião vamos aprender alguns elementos básicos de programação para uma das *shells* mais populares, o *bash*. Em particular, vamos focar-nos na utilização de variáveis e de construções para suportar iterações.

1. Utilização interativa e através de programas (*scripts*)

O *bash* é a *shell* que é executada por omissão quando é aberto um *terminal* nos PCs dos laboratórios. Para verificar qual é a *shell* que está a ser utilizada podem inspecionar a variável SHELL:

```
$ echo $SHELL
/bin/bash
```

Caso o terminal esteja a usar uma *shell* diferente, podem forçar a utilização da *shell bash* através deste comando:

```
$ bash
```

Tal como dá para executar comandos interactivamente em Python e mais tarde juntá-los num *script*, também é possível fazer o mesmo em *shell*. Para isso, basta criar um ficheiro com os comandos e invocar a *shell*.

Crie o seguinte ficheiro usando o seu editor de texto favorito:

```
#!/bin/bash
echo Nesta aula vamos aprender a
echo desenvolver os nossos primeiros scripts em bash!
```

Guarde o ficheiro com nome *meu_script.sh* e torne-o executável usando o comando *chmod*:

```
$ chmod +x meu_script.sh
```

e execute-o:

```
$ ./meu_script.sh
Nesta aula vamos aprender a
desenvolver os nossos primeiros scripts em bash!
```

Notas importantes:

- A extensão ".sh" não é obrigatória, mas é uma convenção útil pois permite identificar facilmente *scripts* de *shell*.
- A primeira linha do script indica ao sistema operativo qual é o interpretador que deve ser utilizado para executar o *script*. Mais em detalhe, os primeiros dois caracteres (#!, pronunciados "*she-bang*" ou "*hash-bang*" em Inglês) são lidos pelo sistema operativo durante a chamada de sistema *execve()* (ver *man execve*). Isto indica ao núcleo que se trata de um *script* cujo interpretador encontra-se no caminho absoluto especificado na restante parte da primeira linha do ficheiro (neste caso /bin/bash).
- Também é possível lançar um *shell script* usando esta sintaxe:

  ```
  $ bash meu_script.sh
  ```

  Note que, neste caso, o ficheiro *meu_script.sh* não tem de ser executável.

- É possível executar um *script* em modalidade *debug* passando a flag -x ao interpretador *bash*, quer quando isto é invocado explicitamente pelo terminal:

  ```
  $ bash -x meu_script.sh
  ```

  quer quando o interpretador é especificado na primeira linha do *script*:

  ```
  #!/bin/bash -x
  echo Nesta aula vamos aprender
  echo desenvolver os nossos primeiro scripts em bash!
  ```

2. Variáveis

Segue-se uma demonstração da utilização de duas variáveis, a e b.

```
$ a=10
$ echo $a
10
```

```
$ b=a
$ echo $b
a
$ b=$a
$ echo $b
10
$ b=batata
$ echo Temos $a ${b}s
Temos 10 batatas
$ echo Temos $a $bs
Temos 10
```

Notas importantes:

- Não se usam espaços à volta do operador de atribuição.
- Para ler o valor contido numa variável usa-se um $ imediatamente antes do nome.
- É boa prática, para evitar ambiguidades, usar chavetas ({}) à volta do nome da variável. Por exemplo, no último comando, ao usarmos $bs estaríamos a obter o valor da variável bs, o que não é o pretendido.

Expansão de comandos

Observe os exemplos indicados abaixo em que é usado o operador de *command expansion*, i.e., $(command), para capturar o output (stdout) dum comando e atribuí-lo a uma variável.

```
$ echo batata doce >terra
$ cat terra
batata doce
$ v=$(cat terra)
$ echo $v
batata doce
```

Nota: Em alternativa à sintaxe $(command), é possível usar também `command` (acento grave), como exemplificado de seguida.

```
$ myvar=`cat terra`
$ echo $myvar
batata doce
```

3. Iterações: ciclo *for*

O *bash* suporta vários operadores para implementar ciclos, e.g., *for*, *while* e *repeat*. Neste guião vamos introduzir apenas a sintaxe do operador *for*.

O ciclo *for* funciona com uma variável de iteração e uma lista de valores a iterar, tal como exemplificado por este *script*:

```
#!/bin/bash
echo Primeira demo do operador for
for i in 0 1 2
do
  echo $i
done

echo ------
echo Implementação equivalente usando expansão de comandos e o comando seq
echo Para compreender este exemplo, experimente executar seq 0 2 numa shell
echo Consulte o manual do comando seq (man seq)
for i in $(seq 0 2)
do
  echo $i
done

echo ------
echo O output do comando ls é expandido na variável de iteração f
for f in $(ls /usr/include/*.h)
do
  echo filename: $f
done

echo Neste exemplo vamos executar o programa Test passando-lhe
echo como input todos os ficheiros presentes na pasta inputs.
echo NOTA: Para que o script funcione, tem de ser executado
echo na mesma pasta onde se encontra o executável Test
for input in inputs/*.txt
do
  echo ==== ${input} ====
  ./Test <"$input"
done
```