universidade de aveiro deti departamento de electrónica, telecomunicações e informática

Aula Prática Nº 2

Objetivos

Programação básica usando a *bash* shell

Comandos e argumentos

Caracteres especiais

Variáveis e parâmetros

Expansão de parâmetros

Padrões

Testes e condições

Agrupamento de instruções

Guião

BASH is an acronym for Bourne Again Shell. It is based on the Bourne shell and it is mostly compatible with its features.

Shells are command interpreters. They are applications that provide users with the ability to give commands to their operating system interactively, or to execute batches of commands quickly. In no way are they required for the execution of programs; they are merely a layer between system function calls and the user.

Think of a shell as a way for you to speak to your system. Your system doesn't need it for most of its work, but it is an excellent interface between you and what your system can offer. It allows you to perform basic math, run basic tests and execute applications. More importantly, it allows you to combine these operations and connect applications to each other to perform complex and automated tasks.

Scripts are basically lists of commands (just like the ones you can type on the command line), but stored in a file. When a script is executed, all these commands are (generally) executed sequentially, one after another.

http://mywiki.wooledge.org/BashGuide

http://www.gnu.org/software/bash/manual/bash.html

1. Crie o *script* **`aula02e01.sh`** com o seguinte conteúdo:

```
#!/bin/bash
# Este é o meu primeiro script
msg="O meu primeiro script em bash!"
echo $msg
```

   a) Utilizando o comando **`chmod`**, atribua permissão de execução ao utilizador do ficheiro.

   b) Execute o *script*: **`./aula02e01.sh`**

   c) Altere a linha `msg="O meu primeiro script em bash!"` para `msg="O meu segundo script em bash!"` e a linha `echo $msg` para `echo msg`. Execute novamente o *script*.

2. Crie o *script* **`aula02e02.sh`** com o seguinte conteúdo:

```
#!/bin/bash
# Atenção aos espaços
echo Este       e      um       teste
echo "Este       e      um       teste"
```

   Execute-o e interprete o resultado.

3. A *bash* consegue interpretar vários tipos de comandos: *aliases, functions, builtins, keywords* e *executables*. Na aula passada vimos a diferença entre *builtins* e *executables.* Crie o *script* **`aula02e03.sh`** com o seguinte conteúdo, execute-o e anote o tipo de cada comando:

```
#!/bin/bash -i
# O comando type permite saber o tipo de um comando
type rm
type cd
type for
type ll
type dequote
```

4. Há diversos caracteres que têm significados especiais na *bash*. Esta processa-os antes de passar ao SO os comandos em que são usados, o que tem impacto muito significativo. Consulte e explore a página *http://mywiki.wooledge.org/BashGuide/SpecialCharacters*. A lista seguinte mostra os mais importantes (alguns já foram explorados na aula 1):

- *Whitespace*
- **$**
- *'texto'*
- *"texto"*
- **#**
- *;*
- ****
- **> *e* <**
- **|**
- **[[** *expressão* **]]**
- **{** *comandos;* **}**
- *`comando`* *ou* **$(***comando***)**
- **(***comando***)**
- **((***expressão***))**
- **$((***expressão***))**

   a) Crie o *script* **`aula02e04.sh`** como segue, execute-o e interprete o resultado:

```
#!/bin/bash
echo "O meu editor por omissão BASH $BASH \$BASH"
echo 'O meu editor por omissão BASH $BASH \$BASH'
echo $(( 5 + 5 ))
(( 5 > 0 )) && echo "cinco é maior do que zero"
today=$(date); echo $today
```

b) Construa os comandos que permitem listar, de entre os ficheiros e directorias em **`/etc`**:

   i. Todos.
   ii. Todos cujo nome começa por **`a`**.
   iii. Todos cujo nome começa por **`a`** e têm mais que 3 caracteres.
   iv. Todos os que têm **`conf`** no nome.

5. No Exercício 1, vimos como se atribuem valores a variáveis na *bash* e como se utilizam depois. É também possível passar argumentos a um *script*. O seu acesso dentro do *script* é feito utilizando o que se designa por variáveis especiais (**`$1`**, **`$2`**, … **`$`***n*, **`$*`**, **`$@`**, **`$#`**, etc.). Crie o *script* **`aula02e05.sh`** com o seguinte conteúdo:

```
#!/bin/bash
echo "$#"
echo "Arg 1: $1"
echo "Arg 2: $2"
echo "$*"
echo "$@"
```

   a) Execute o *script* com diversos números e tipos de argumentos e observe os resultados.
   b) Crie o *script* **`px.sh`** que tem como parâmetro o nome de um ou mais ficheiros e dá ao utilizador permissão de execução de todos eles.
   c) Desenvolva o *script* **`soma2.sh`** que recebe como argumentos dois números e escreve o valor da sua soma na consola.
   d) Crie o *script* **`aula02e05d.sh`** com o seguinte conteúdo e execute-o com vários argumentos, alguns contendo espaços. Interprete os resultados.

```
#!/bin/bash
echo using '$*:' $*
for i in $*; do echo "$i"; done

echo using '$@:' $@
for i in $@; do echo "$i"; done

echo using '"$*":' "$*"
for i in "$*"; do echo "$i"; done

echo using '"$@":' "$@"
for i in "$@"; do echo "$i"; done
```

6. É também possível fazer expansão de parâmetros na *bash* (vide detalhes em *http://www.gnu.org/software/bash/manual/bashref.html#Shell-Parameter-Expansion*). Crie o *script* **`aula02e06.sh`** com o seguinte conteúdo, execute-o e interprete o resultado:

```
#!/bin/bash
# Parameter Expansion
file="$HOME/.bashrc"
echo "File path: $file"
echo "File name: ${file##*/}"
echo "Directory name: ${file%/*}"
```

7. Os símbolos **`*`**, **`?`** e **`[…]`** são usados para criar padrões, por exemplo para aceder a múltiplos ficheiros. Na *bash* podemos ter outras formas de criar padrões, tais como as expressões regulares e a expansão de chavetas. Crie o *script* **`aula02e07.sh`** com o seguinte conteúdo, execute-o e interprete o resultado:

```
#!/bin/bash
# Brace Expansion
echo {1..9}
echo {0,1}{0..9}
```

   a) Utilizando o comando **`touch`**, escreva o *script* **`c10xpto.sh`** para criar 10 ficheiros com os nomes **`xpto00.dat`** até **`xpto09.dat`**.

   b) Copie o *script* anterior para **`c10param.sh`**. Edite-o de modo que crie 10 ficheiros cujo nome-base (segmento antes dos números) seja um parâmetro do *script*. (Exemplo: **`c10param.sh abc`** criará os ficheiros **`abc00.dat`** até **`abc09.dat`**).

8. Para elaborar *scripts* mais complexos, para além de sequências de comandos, é necessário dispor de testes e instruções de decisão e repetição. Um ponto essencial a ter em consideração é que todos os comandos geram um código de saída quando terminam. Este código é normalmente usado para saber como correu o comando executado. Foi convencionado que o valor ***0*** significa 'tudo bem'.

   a) Crie o *script* **`aula02e08.sh`** listado a seguir, execute-o e interprete o resultado:

```
#!/bin/bash
# Exit status
ping -c 1 www.ua.pt
echo "Exit code: $?" # $? gives exit code of last process
ping -c 1 wwwww.ua.pt
echo "Exit code: $?"
```

   b) Crie o *script* **`exitfile.sh`** que recebe como argumento o nome de um ficheiro, que pode existir ou não, e imprime o código de saída do comando **`file`** para esse ficheiro.

9. É possível criar listas de comandos em *bash*. Para isso podem ser usados os seguintes operadores: **`;`**, **`&`**, **`&&`**, ou **`||`**. Para explorar o seu significado, consulte a página *http://www.gnu.org/software/bash/manual/bashref.html#Lists*.

   a) Crie o *script* **`aula02e09.sh`** listado a seguir, execute-o e interprete o resultado:

```
#!/bin/bash
mkdir d && cd d && pwd
echo "----------------"
pwd && rm xpto || echo "I couldn't remove the file"
```

   b) O que acontece quando repete a execução do *script*?

   c) Em que diretoria de trabalho se encontra quando executa o último comando (considere em separado a execução do *script* na alínea *a* e na alínea *b*)?