universidade de aveiro deti departamento de electrónica, telecomunicações e informática

Aula Prática Nº 4

Objetivos

Funções em *bash*

Listas (*arrays*)

Guião

1. A criação de funções em *bash* é uma forma de agrupar comandos para poderem ser executados mais tarde num *script* através de uma única chamada ao nome da função. As funções são executadas na mesma *bash* onde são declaradas. A declaração de funções é feita com a seguinte sintaxe:

```
function FUNCTION { COMMANDS; }
```

ou

```
FUNCTION () { COMMANDS; }
```

a) Crie o *script* **`aula04e01.sh`** com o seguinte conteúdo, execute-o e interprete o resultado:

```
#!/bin/bash
function imprime_msg()
{
      echo "A minha primeira funcao"
      return 0
}
imprime_msg
```

b) Acrescente ao *script* anterior uma função que permita mostrar a data de hoje, o nome do PC onde está a trabalhar e o nome do utilizador.

c) Existem várias formas de importar funções para um *script*, quando estas foram declaradas num ficheiro separado:

```
. /path/to/functions #atenção ao espaço.

source /path/to/functions
```

Altere o *script* anterior de modo a implementar as funções num ficheiro separado.

d) defina a função `dw()`, na própria bash do terminal (sem criar um *script*), que executa os comandos `date` e `who` em sequência. Execute essa função em diferentes diretorias.

2. As funções podem ser consideradas como pequenos *scripts*, recebendo argumentos e devolvendo um estado de saída para processamento futuro. Os argumentos nas funções são tratados da mesma forma que nos *scripts*.

a) Crie o *script* **`aula04e02.sh`** com o seguinte conteúdo, execute-o passando um número entre 1 e 5 e interprete o resultado:

```
#!/bin/bash
function numeric_to_string()
{
    case "$1" in
        1)
            echo "Um"
            ;;
        2)
            echo "Dois"
            ;;
        3)
            echo "Três"
            ;;
        *)
            echo "Outro numero"
    esac
    return 0
}
numeric_to_string $1
```

b) Com base no que já aprendeu sobre a utilização do valor de retorno de um comando (`$?`), como acha que pode avaliar o valor de retorno de uma função? Altere o *script* anterior de modo a devolver um número igual ao valor introduzido através do valor de retorno e mostre como acede a esse valor após a chamada à função.

c) Desenvolva um *script* que receba dois números como argumento e escreva uma mensagem que indique se os números são iguais ou, caso contrário, que indique o maior deles. Deve implementar uma função para a comparação dos dois números. Essa função não deve imprimir qualquer mensagem no ecrã.

d) Altere o *script* anterior de modo a não ter argumentos mas a pedir ao utilizador para introduzir dois números (explore o comando `read`).

3. A *bash* permite a declaração de *arrays* através da notação `variavel[pos]`. Também é possível declarar um *array* com a seguinte notação: `variavel=( elemento1 elemento2 elemento3 … )`. O acesso a uma posição do *array* é feito com base na utilização de chavetas: `${variavel[pos]}`. Explore o conteúdo da seguinte página: http://tldp.org/LDP/abs/html/arrays.html onde encontra muitos detalhes sobre a utilização de *arrays* em Bash.

a) Crie o *script* **`aula04e03a.sh`** com o seguinte conteúdo, execute-o e interprete o resultado:

```
#!/bin/bash

lista=( {1..10} )

for i in "${lista[@]}"; do
      echo "$i"
done
```

b) Crie o *script* **`aula04e03b.sh`** com o seguinte conteúdo, execute-o e interprete o resultado:

```
#!/bin/bash

lista=( $(seq 2 3 15) )

echo vals  ${lista[@]}
echo count ${#lista[@]}
echo index ${!lista[@]}

for ((i = 0; i < ${#lista[@]}; i++)); do
      lista[i]=$((${lista[i]}+1))
      echo "$i: ${lista[i]}"
done

unset lista[1]
unset lista[3]

echo count ${#lista[@]}

for i in ${!lista[@]}; do
      echo "$i: ${lista[i]}"
done
```

c) Desenvolva um *script* que permita ordenar um conjunto de números armazenado num ficheiro utilizando o algoritmo de ordenação por seleção. O ficheiro deve ser passado por argumento ao *script*. Efetue as validações que considere necessárias.