

3

Programação em Bash Shell

(Bash Shell)

Os objectivos deste módulo são os seguintes:

- Programação do Bash Shell
- Estruturas de Controlo if e case
- Repetição

Referências

- [A quick guide to writing scripts using the bash shell](#) : [localcopy](#)
- [Advanced Bash Scripting Guide : the Linux Documentation Project](#) : [localcopy](#)
- [Vários documentos sobre Unix](#)

1 Introdução

O *bash shell* é antes de mais um interpretador de comandos. Efectua a leitura de comandos introduzidos na linha de comando e *interpreta-os*. A interpretação significa uma ou mais acções do sistema operativo, por exemplo executar o comando que foi escrito! Resumindo, os interpretadores de comandos de linha, tipo *Bash Shell*, são uma maneira de efectuar a interface entre o utilizador e o sistema operativo.

No caso dos sistemas Linux/Unix, o interpretador de comandos afecto a cada utilizador por defeito encontra-se no ficheiro `/etc/passwd`. Execute, por exemplo, o comando `finger` para ver a sua informação.

Um *Shell Script* é um conjunto de comandos escritos numa linguagem de *script*. Uma linguagem *script* permite uma interpretação mais complexa dos nossos comandos. Portanto um shell script poderá ser apenas uma sequência de comandos existentes ou através de uma linguagem própria podemos usar variáveis, ciclos de repetição e outras estruturas de controlo típicos de qualquer linguagem imperativa, até há linguagens criptas baseados em outras paradigmas de programação utilizando por exemplo objectos. Normalmente os *shell scripts* são utilizados na construção de pequenas aplicações de auxílio à administração de sistemas e aos programadores experientes. São utilizados para automatizar tarefas diárias tais como realização de cópias de segurança, gestão de contas, remoção automática de determinado tipo de recursos, pesquisa de informação, etc. Também podem ser utilizados para prototipagem rápida de aplicações mais complexas, gerir projectos de programação e como partes pequenas de sistemas sofisticados.

1.1 O Primeiro Script

Um *shell script* não é mais do que uma sequência de comandos interpretados um após o outro pelo sistema operativo. Existem varias linguagens cada uma com o seu próprio sintaxe e nomes pelos vários comandos e funcionalidades do sistema operativo.

A primeira linha dum ficheiro “script” indica a linguagem que será utilizada para interpretar o script. De facto indica ao interpretador actual qual a aplicação a utilizar. Na figura 1 a aplicação indicada é `/bin/bash` (**b**ourne **a**gain **s**hell) - o ficheiro *bash* no directório */bin*.

A segunda linha é apenas um comentário e será ignorado.

As restantes linhas são todas comandos que podiam ter sido introduzidos directamente na linha de comando.

```
#!/bin/bash
#primeiro script - ola.sh

echo ola $USER
pwd
date
```

Figura 1. Um Exemplo Simples - o ficheiro “ola.sh”

1.2 Atribuição de permissões de execução

Para executar o ficheiro é necessário atribuir permissões de execução, usando o comando *chmod*.

```
alunos:~/scripts crocker$ chmod +x <nome_ficheiro_shell_script>
ou
alunos:~/scripts crocker$ chmod 744 ola.sh
```

Figura 2 – Atribuição de permissões de execução

1.3 Execução.

Neste momento o ficheiro *ola.sh* pode ser executado, ou melhor dito “interpretado”, escrevendo simplesmente o seu nome. Ver o exemplo completo em baixo:

```
alunos:~/scripts crocker$ vi ola.sh           //criar o ficheiro
alunos:~/scripts crocker$ ls -l ola.sh        //ver detalhes do ficheiro
-rw-r--r-- 1 crocker staff 68 24 Feb 11:30 ola.sh
alunos:~/scriptscrocker$ ola.sh
-bash: ola.sh: command not found              //Problema com o path.
                                           //Não está incluído o directório actual o “ponto”
alunos:~/scripts crocker$ ./ola.sh
-bash:./ola.sh: Permission denied              //problema com a permissão
alunos:~/scripts crocker$ chmod 744 ola.sh    //mudar permissão
alunos:~/scripts crocker$ ./ola.sh
ola crocker
/Network/Servers/alunos/Volumes/alunos/Alunos/crocker/scripts
ola.sh exemplo.sh alunos.txt
ThuFeb 24 11:31:33 WET 2005
```

Finalmente vemos o resultado do nosso primeiro *shell script*.

2 Variáveis.

A sintaxe de declaração duma variável: <nome_variável>=<valor>

- Não são permitidos espaços antes nem depois do carácter =
- Apenas caracteres alfanuméricos podem ser utilizados como identificadores válidos de variáveis.
- Os valores do tipo string que contenham espaços devem ser especificados entre aspas como no exemplo em baixo

```
#!/bin/bash

var1=123
var2=Ola
var_ia_vel="Ola Mundo"
```

Figura 3: Exemplo de variáveis

Exercício 1: Faça um bash shell script para experimentar com variáveis como no exemplo em baixo.

```
#!/bin/bash

# variáveis de ambiente
echo $USER $HOME
echo $PATH
# ver todas as variáveis do ambiente com o comando "env"

#variaveis locais

ola="bom dia"
echo "$ola Paulo"
echo "$olaPaulo"          #Texto Pegado a variavel .. não funcione..
echo "${ola}Paulo"        #proteger a variável com as chavetas..ok
mesg="$ola $USER"
echo $mesg

# input usando read

echo "Introduza qualquer Coisa"
read var
echo "Introduziu $var"

#variaveis especiais

echo "Numero de Arguments para este script $#"
```

Figura 4: Exemplo de variáveis

2.1 Substituição de Comandos

Esta facilidade do Bash Shell permite atribuir o output dum comando a uma variável. Isto é feito usando plicas (acento) à volta do comando pretendido .. `comando` ..Veja o exemplo seguinte

```
data=`date`      #um exemplo simples
echo $data

#um exemplo misturando vários comandos e variáveis.
info=`echo $HOME ; echo " estamos no directorio "; pwd`
echo $info
```

2.2 Vectores (Arrays)

O Bash Shell permite a utilização de variáveis do tipo Array, apenas com uma dimensão (Vector).

Os elementos dum Array podem ser definidos usando a sintaxe `variable[índice]` - onde índice é um valor inteiro 0,1,2..etc.

Para obter o valor dum elemento dum array utilize-se a sinatxe `{variable[xx]}` .

```
v[2]=1
v[3]=ola
v[4]=12      #elementos dum array podem não ser consecutivas ou do mesmo tipo
v[7]="ola mundo" #pode deixar buracos no array
echo ${v[2]}

dias=( domingo segunda terca quarta )      #declaração e inicialização dum array
indice=0
echo "Hoje é ${dias[indice]}"
```

Figura 5: Exemplos de Arrays

```
files=(`ls`) #output do comando ls passado para um vector
echo ${files[2]}
echo ${#files[@]} -- numero de elementos do array
```

Figura 6: Exemplo de definição dum Array usando substituição dum comando.

3 Os Operadores

3.1 Operadores Aritméticos

+	Soma	-	Subtracção
*	Multiplicação	/	Divisão
**	Exponencial	%	Módulo (Resto da Divisão)

Para avaliar uma expressão aritmética utiliza-se a função `let` como no exemplo em baixo

```
#!/bin/bash
x=1
let x=x*2+3      #output5
echo "x=$x"
let x--          #output4
echo "x=$x"
y=2
let x=x + 3**y    #output 13
echo "x=$x"
```

Exercício 2: let Experimente com o comando `let`

3.2 Lógicos

&&	E
	Ou

3.3 Comparação inteira (it helps to know a little English ..)

-eq	Equal	Igual
-ne	Not Equal	Diferente
-gt	Greater than	Maior que
-ge	Greater or equal	Maior ou igual a
-lt	Less than	Menor que
-le	Less than or equal	Menor ou igual a

3.4 Comparação de strings

=	Igual
!=	Diferente
<	Menor que
>	Maior que
-z	String nula, ou seja, tamanho = 0
-n	String não é nula

3.5 Ficheiros

-e	Devolve verdade caso o ficheiro exista
-f	Devolve verdade se o ficheiro é regular e não uma directoria
-d	Devolve verdade caso se trate de uma directoria

Consultar as referências para obter a lista completa dos operadores

4. As Estruturas de Controlo de decisão “if” e “case”

4.1 A sintaxe do comando if

```
if [ condição1 ]
then
comandos no caso da condição1 ser verdadeira
elif [ condição2 ]
then
comandos no caso da condição2 ser verdadeira
else
comandos no caso de nenhuma das condições ser verdadeira
fi
```

Importante : Os espaços entre as palavras chaves são importantes ... if_[_condição1_]

4.2 A sintaxe do comando case

A estrutura *case* é similar à estrutura *switch* da linguagem C.

```
case valor_duma_variável in
    constante 1) comando1;;
    constante 2) comando2
                comando3 ;;
    *)           comando_default ;;
esac
```

Exercício 3: Menus de Calendários

- Faça um bash shell script que apresente ao utilizador um menu onde o utilizador pode escolher entre duas opções. A primeira opção deverá mostrar o calendário do mês actual e a segunda a data actual Utilize a estrutura de decisão `if..else` e os comandos `date` e `cal`. Se o utilizador escolher outra opção então o programa dever responder com o texto “Opção desconhecida”
- Mudar a estrutura de decisão para “**case**”
- Adicione mais três opções, “calendário do ano actual”, “o numero de segundos desde 1970” e a “o calendário do ano 1752”. Se o utilizador escolhe rum a opção inválida deverá aparecer a mensagem “opção invalida”. Terá que investigar antes os comandos `cal` e `date` usando o comando `man` ou `info`
- Adicione mais uma opção – “A data em português” .. “Hoje é quarta-feira, dia 15 de Março de 2007”

Exemplo 4: O script mywho

O script em baixo indicará se existe um utilizador actualmente ligado ao computador cujo username contem o string lido para a variável `nome`. Se for o caso o script executará o comando ***echo*** para imprimir uma mensagem no ecrã.

```
#!/bin/bash
# script mywho : versão inicial

echo "Introduza Nome (userid) da Pessoa "
read nome

echo "Procurando  "$nome
# comando who seguido por um filtro.
who| grep $nome

# Nota : exit status of previous command is stored in  $?
# Valor por defeito é zero. Zero indique sucesso.

if [ $? -eq 0 ] ; then
    echo "$nome  Foi Encontrado "
else
    echo "$nome  Não Foi Encontrado "
fi
echo
```

Exercício 5: mywho

Implemente e Experimente o script “mywho”. A seguir modifique-o para aceitar um string como parâmetro em vez de ser lido. Se o username (string) for encontrado então o script deverá adicionalmente executar o comando ***finger*** filtrando o output pelo valor do string.

Solução: Em vez de ler o “nome” com o comando ***read*** utilizar-se-á a variável \$1.

5 Estruturas de Repetição

Existem os seguinte ciclos de repetição:

- ciclos for
- ciclos while
- ciclos until

5.1 O comando for

A sintaxe do comando for: `for variável in Lista_de_Valores do ..done`

```
#!/bin/bash
for X in red green blue 11 21 23
do
    echo $X
done
```

Exemplo : imprimir cores e números numa lista simples

```
#!/bin/bash
for i in *.c
do
    cp $i ~/backup/
done
```

Exemplo: Copiar todos os ficheiro de C para um directório de backup

```
#!/bin/bash
y="ola bom dia"
for i in $y
do
    echo $y
done
```

Exemplo: Copiar todos os ficheiro de C para um directório de backup

```
#!/bin/bash
y="ola bom dia"
for i in $y
do
    echo $y
done
```

A lista neste caso é a variável y que contenha 3 strings

Exemplo: Calcular a soma duma coluna de números inteiros contido no ficheiro valores.txt (um valor por LINHA) passando primeiro os valores para um vector para ilustrar o uso dum vector ao mesmo tempo.

```
vec=( `cat valores.txt` )
sum=0
for elemento in "${vec[@]}"
do
    echo $elemento
    let sum=sum+elemento
done
echo "soma = $sum"
```

A lista neste caso é os valores do vector convertidos numa lista usando as aspas.

Exercício 6: Argumentos Implemente e experimente o seguinte Script *args*

```
#!/bin/bash
#script args

num=1

for x in $*
do
    echo "Argumento $num = $x"
    let num++
done
```

Execução:

```
$ args 12 in ola

Argumento1 = 12
Argumento2 = in
Argumento3 = ola
```

Exercício 7: mywho

Generalizar o script anterior “mywho” para aceitar N parâmetros. Precisar da estrutura de repetição “for”

Exercício 8:fichas

Faça um bash shell script, **fichas**, que faça uma listagem dos ficheiros no directório actual na seguinte forma

```
[crocker@penhas p2]$ fichas
Ficheiro 1  ex1.c
Ficheiro 2  fichas
Ficheiro 3  gl.c
Ficheiro 4  tmp
Ficheiro 5  stack.c
```

Solução :

Utilizar uma variável para guardar os nomes dos ficheiro `f=`ls` ...`
seguido por um ciclo **for variavel in \$f do.. done**

Exercício 9: Enquanto

Implemente o script “enquanto” em baixo que ilustre o sintaxe do comando while.

```
#!/bin/bash
#script contar ou not-quite-enquanto
cnt=1
while [ $cnt -le 10 ]
do
    echo "cnt $cnt"
    let cnt++
done
```

Modifique o shell script, *enquanto*, que faça uma contagem apartir do valor do seu primeiro argumento até o valor do segundo. Deverá estudar e modificar o script em baixo

```
[bash]$ enquanto 2 5
cnt 2
cnt 3
cnt 4
cnt 5
```

Exercicio 10: wcnovo

- (a) Faça um bash shell script, *wcnovo*, que mostre apenas o numero de palavras e linhas de cada ficheiro no directório actual.
- (b) Altere o script para que no fim apresente o número total de palavras e linhas de todos os ficheiros do directório.

8 Funções

Funções são blocos de comandos que podem ser executados em qualquer parte do código.
Ver o exemplo em baixo:

```
#!/bin/bash
#script backup

Listar()
{
  echo "Opcao 1 :listar ficheiros .c"
  echo "Opcao 2 :listar ficheiros .txt"
  read op
  cd ~/backup
  if [ op -eq 1 ]
    ls -l *.c
  else
    ls -l *.txt
  fi
}

Main_Menu()
{
  opcao=1
  while [ $opcao -ne 0 ]
  do
    echo "1. Backup dos ficheiros"
    echo "2. Listagem da pasta ~/backup"
    echo "0. Sair"
    echo
    echo -n "Introduza a sua escolha "
    read opcao
    case $opcao in
      1) Backup ;;
      2) Listar ;;
      0) exit ;;
      *) "Opcao desconhecida"
    esac
  done
}

date
Main_Menu
```

9 Finalizar

Os seus scripts devem ser acessíveis a partir de qualquer directório do seu login shell.

Por isso coloca os seus scripts no seu directório ~/bin e altere o seu path para incluir este directório.

Nota

No seu ambiente de trabalho, devem configurar a variável "Path" para incluir o directório actual (o símbolo ".") e o seu directório \$HOME/bin. Também devem conformar que tem um ficheiro ".alias", onde devem colocar comandos tipos aliases, e que este ficheiro é lido durante o processo de logon.

10 Utilidades

10.1 Formatação

Poderá formatar output no bash shell usando o comando builtin do shell “printf” ..
o sintaxe é muito parecido com o sintaxe do printf da linguagem C.

Exemplo

```
alunos:$ printf "%s\t%f\n%f\n" "ola" 12.2 13.3
ola 12.200000
13.300000
```

10.2 Ordenação

O comando sort (ordenar) é muito útil para ordenar usando critérios alfabéticos ou numéricos

Exemplo : Usando o ficheiro test.txt para experimentar que tem quatro campos

```
alunos:~/scripts$ cat test.txt
a:1:3.2
c:9:1.2
d:1:0.2
b:2:0.3
```

Ordenar alfabeticamente o primeiro campo (key -k 1) -t é para indicar o separador dos campos, neste caso o “:”

```
alunos:~/scripts$ sort -t ":" -k 1 test.txt
a:1:3.2
b:2:0.3
c:9:1.2
d:1:0.2
```

Ordenar numericamente (-n) usando o campo 2

```
alunos:~/scripts$ sort -n -t ":" -k 2 test.txt
a:1:3.2
d:1:0.2
b:2:0.3
c:9:1.2
```

Ordenar numericamente (-n) usando o campo 3

```
alunos:~/scripts$ sort -n -t ":" -k 3 test.txt
d:1:0.2
b:2:0.3
c:9:1.2
a:1:3.2
```

10.3 Cálculos Numéricos

COM INTEIROS Usando os comandos `expr` e `let`

```
$expr 1 + 3
4

$ expr 10 / 2
5

$ let x=3*5
$ echo $x
15
$ let x=15/3+2
$ echo $x
7
```

COM REAIS

Devem ter visto que os comandos do shell `"let"` e `"expr"` apenas permitem operações do tipo inteiro. E para efectuar cálculos numéricos com floats ?

Opção 1 : Utilizar o programa `bc` - arbitrary precision calculator language

```
y=2.2
x=2.7
echo "$x/$y"| bc -l
1.22727272727272727272
```

Opção 2 : Utilizar o programa `awk`

```
x=2.2
y=3.3
echo $x $y | awk '{print $1+$2}'
```

Opção 3 : Escrever (em `c`) um comando próprio !!!!! chamado por exemplo `mylet`

Utilização seria :

```
mylet 2.2 2.4 +
OU
mylet 2.3 4.2 *
```

O programa depois será qualquer coisa como ...

```
#include ..
main( int argc , char**argv )
{
    float a,b;
    a= atof(argv[1] );
    b= atof( argv[1] );
    if ( 0==strcmp(argv[3],"+")) printf("%f\n",a+b);
    if ( 0==strcmp(argv[3],"/")) if (b!=0) printf("%f\n",a/b); else printf("NaN\n");
}
```

```
cc -o mylet mylet.c
```

```
mv mylet $HOME/bin - disponibilizar comando, copiando para um directoria do seu path!
```

11 Exercícios Suplementares

NOTA: não utilizem chamadas ao sistema **system** apenas os comandos e linguagem do shell.

EXERCÍCIO Nº1 (SCRIPT)

- (a) Implemente um comando Unix, designado por **wh**, que envia para o écran o path completo do comando passado como parâmetro; por exemplo, **wh ls** devolverá `/bin/ls`.
- (b) Generalize o comando para *n* parâmetros.

dica which

EXERCÍCIO Nº2 (SCRIPT)

Implemente um comando Unix, designado por **cpy**, que copie a primeira metade dum ficheiro para outro ficheiro. O ficheiro destino é um ficheiro criado pelo próprio comando e passado como parameter

dica . primeiro numero de linhas no ficheiro(wc) depois head/tail

EXERCÍCIO Nº3 (SCRIPT)

Implemente um comando Unix, designado por **ch**, que iniba recursivamente todas as permissões, excepto as do proprietário, das directorias – a partir dum directório dado como parameter. Se for dado um segundo parameter o valor "read" então as directorias deviam também ter a permissão "rx" para o grupo.

dica chmod

EXERCÍCIO Nº4 (SCRIPT)

Implemente um comando Unix, designado por **us**, que faça uma listagem de todos os utilizadores autorizados do sistema, incluindo os usernames, nomes completos, e estado de utilização (IN ou OUT).

dica ficheiro /etc/passwd e o comando finger

EXERCÍCIO Nº5 (SCRIPT)

Implemente um comando Unix, designado por **matar**, que faça uma listagem de todos os processos do utilizador corrente (PID, nome do utilizador, e o nome do processo). A seguir deverá pedir um "nome" e depois remover todos os processos do utilizador com este nome.

dica ps, kill

Referências	1
1 Introdução.....	2
1.1 O Primeiro Script	2
1.2 Atribuição de permissões de execução.....	3
1.3 Execução.	3
2 Variáveis.....	4
Exercício 1: Faça.....	4
2.1 Substituição de Comandos	5
2.2 Vectors (Arrays)	5
3 Os Operadores	5
3.1 Operadores Aritméticos	5
Exercício 2: let Experimente com o comando let 3.2 Lógicos	5
3.2 Lógicos	6
3.3 Comparação inteira (it helps to know a little English ..).....	6
3.4 Comparação de strings	6
3.5 Ficheiros	6
4. As Estruturas de Controlo de decisão “if” e “case”	6
4.1 A sintaxe do comando if.....	6
4.2 A sintaxe do comando case	7
Exercício 3: Menus de Calendarios.....	7
Exemplo 4: O script mywho.....	7
Exercício 5: mywho	7
5 Estruturas de Repetição	8
Exercício 6: Argumentos Implemente.....	9
Exercício 7:	9
Exercício 8:fichas	9
Exercício 9: Enquanto	9
Exercício 10: wcnovo.....	10
8 Funções	11
9 Finalizar.....	11
10 Utilidades	12
10.1 Formatação	12
10.2 Ordenação.....	12
10.3 Calculos Numericos	13
11 Exercícios Suplementares	14