Bash

J.João Almeida

Departamento de Informática
Universidade do Minho

3 de Dezembro de 2014

Mensagem subliminar

- linha de comando... (modo interactivo)
- composicionalidades...
- Bash scripting
- $\rightarrow$ pode ajudar à qualidade de vida...

Menu

Introdução

Manuais

man man
man apropos
help
help read
... tio-google("Bash scripting the hard way")

Ficheiros de configuração

- .bashrc – executado quando arranca nova bash
- .bash_profile – executado após login
- .bash_history –

Sugestão:

- `mkdir ~/bin`
- `export PATH=.:$HOME/bin:$PATH` em .bashrc

Convenções filosóficas

```
$                 → expande valores
$v                → valor da variável v

'text'            → strings
"text $v ..."     → strings com expansão
#...              → comentários
;                 → separador de commandos
\                 → protege metacarateres

<   >   >>        → redirecionamentos

comand | comand   → pipes
[[ ... ]]         → test expressions (Booleanos)
{ comandos }      → agrupa comandos num comando
( comando )       → cria uma subshell

(( exp arit ))    → aritmética
$((expression))   → ... para substituir em strings

`comando `        → expande para a stdout do comando
$(comando )       → ... idem
```

Lógica...

Comandos retornam um valor de sucesso

- $= 0$ – comando bem sucedido `int main()...  return 0`
- $\neq 0$ – houve problemas `...exit 1`
- makefile

```
||              → or   ( ou então...)
&&              → and  ( e depois...)
!               → not
```

Em caso de erros → stderr (=file desc. 2)

```
echo "erro" >&2 → print para STDERR
```

executar ou avisar de erro...

```
cat f || echo "cant find f" >&2
```

Sempre que se vai construir uma scritpt...

```
which bash
/bin/bash
```

```
#!/bin/bash
#
#  scriptname argument [argument] ...
#
#  A short explanation of your script's purpose.
#
#  Copyright [date], [name]
```

Redirecionamentos

```
date > a

(cat nonexisting-file ; date)  2> erros > saidas

(cat nonexisting-file ; date)  2>> erros >> saidas
```

```
wc < a

wc < a > b

wc <(ls)
```

Input e read

```
#!/bin/bash
{
    read l1
    read l2
    while read linha; do
       echo "."
       ultima=$linha
    done
} < $1

echo -e "1=$l1 \n 2=$l2 \n ultima=$ultima";
#fim
```

Parámetros

```
$0      → comando  argv[0]
$1 $2   → argv[1] argv[2]
$*      → "$1 $2 ... $n"
$@      → ($1, $2, ... $n)
$#      → argc

$?      → exit status of previous command
$$      → PID current process
```

Variáveis de ambiente

```
HOSTNAME:
PPID:   PID of the parent process
PWD:    Contains the current working directory.
RANDOM: Each time you expand this variable, a (pseudo)random [0..32767]
UID:    The ID of the current user.
HOME:
PATH: A colon-separated list of paths
PS1: define o prompt.
```

Types

```
Array              : declare -a variable              array of strings.
     var=("era" "uma" "gato")
     $(#var[*])    → 3
     ${var[2]}     → gato

Associative array  : declare -A variable
Integer            : declare -i variable
Read Only          : declare -r variable
Export             : declare -x variable         inherited by child Proc
default            : strings
```

If... (múltipla formatação possível)

```
if comando
then outro comando
fi

if comando
then
    outro comando
fi

if comando; then
    outro comando
fi

if [ -f ss ]; then echo 1; else echo 2; fi
```

Ciclos For, While, Until

```
while ! ping -c 1 -W 1 1.1.1.1; do
  echo "still waiting for 1.1.1.1"
  sleep 1
done


(( i=10 ))
while (( i > 0 ))
   do echo "$i empty cans of beer."
      (( i-- ))
   done

for (( i=10; i > 0; i-- ))
 do echo "$i empty cans of beer."
 done


for i in {10..1}
  do echo "$i empty cans of beer."
  done

for file in *.mp3; do rm "$file"; done
```

Estrutura de control Case

```
case $LANG in
    en*) echo 'Hello!'      ;;
    fr*) echo 'Salut!'      ;;
    de*) echo 'Guten Tag!' ;;
    nl*) echo 'Hallo!'      ;;
    it*) echo 'Ciao!'       ;;
    es*) echo 'Hola!'       ;;
    C|POSIX) echo 'hello world' ;;
    *)   echo 'I do not speak your language.' ;;
esac
```

Select

```
select choice in Apples Pears Crisps Lemons Kiwis; do
   if [[ $choice = Crisps ]]
     then echo "Correct! Crisps are not fruit."; break; fi
   echo "Errr... no. Try again."
done
```

Tests and Locical expressions

```
-e FILE: file exists.
-f FILE: file is a regular file.
-d FILE: file is a directory.
-h FILE: file is a symbolic link.
-r FILE: file is readable by you.
-w FILE: the file is writable by you.
-O FILE: the file is effectively owned by you.
FILE -nt FILE: file is newer than the second.
FILE -ot FILE: file is older than the second.
-z STRING: the string is empty (it's length is zero).
-n STRING: the string is not empty (it's length is not zero).
STRING = STRING
STRING != STRING
STRING < STRING
STRING > STRING
EXPR -a EXPR
EXPR -o EXPR
! EXPR: (logical NOT).
INT -eq INT:
INT -ne INT:
INT -lt INT:
INT -gt INT:
INT -le INT:
INT -ge INT:
```

Regular expressions

```
re = '(..)[_-](..)'

if [[ $LANG =~ $re ]]
   then
        echo "Your country code  is ${BASH_REMATCH[2]}."
        echo "Your language code  is ${BASH_REMATCH[1]}."
   else
        echo "Your locale was not recognised"
   fi

## BASH_REMATCH  -- array do que fez match ()...
```

Funções

```
function fun1(){
  echo 34
}

function fun2(){
  local res=$(fun1)
  echo $res
}
```

Arrays

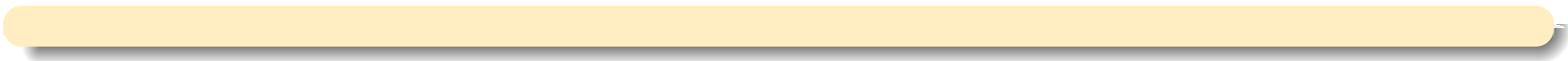
```
names=("Bob" "Peter" "$USER" "Big Bad John")

names[0]="Rui"

files=(~/*.jpg)
cp "${files[@]}" /backups/


for file in "${myfiles[@]}"; do
   cp "$file" /backups/
done

${#array[@]}       #cardinal
```

Um menu simples...

```
while true; do
    echo "Welcome to the Menu"
    echo "  1. Say hello"
    echo "  2. Say good-bye"

    read -p "-> " response
    case $response in
        1) echo 'Hello there!' ;;
        2) echo 'See you later!'; break ;;
        *) echo 'What was that?' ;;
    esac
done
```

Exemplo: open

```
#!/bin/bash

 open() {
    case "$1" in
      *.mp3|*.ogg|*.wav|*.flac|*.wma) vlc "$1"     ;;
      *.jpg|*.gif|*.png|*.bmp)        eog "$1"     ;;
      *.avi|*.mpg|*.mp4|*.wmv)        mplayer "$1" ;;
      *.pdf|*.ps)                     evince "$1" ;;
      *)                              echo "what is $1 ?" ;;
    esac
 }

 for file in $* ; do       ### $@
    open "$file"
 done
```



Demos variados

ferramenta para

abrir (EXEM/ex1)
abrir (EXEM/ex2)
abrir (EXEM/ex3)

→ Scriptização de template

Outras scripts...

Evitar usar Bash para tudo!!!
Para fazer:

- sed // streams, filtros
- awk // excelente para tabelas, filtros
- até o vi(m)!!
- perl excelente para texto
- perl / python / ruby (dão para tudo!)

... Outras scripts...

```
#!/usr/bin/perl -s
our($l,$c) ;          ## command line options
$l ||= 30;            ## ex4 -l=300   default=30
$c ||= 3;             ## ex4 -c=6     default=3

use utf8::all;

my @a= 1..6


sub die_face{ $a[int(rand(6))] }

for (1..$l){
  print die_face()," "  for(1..$c-1);
  print die_face(),"\n"
}

__END__
#From unicode table misc. symb. #U+2680   &#x2680;    DIE FACE-1
```

Vi scripting

```
#!/usr/bin/vim -s
:%s/ *$//
1Go----------------------^[
G
:r ! echo "\#last changed by $USER in :" `date`
:x
```

Conclusões

FixMe: Falta concluir

Fim