Using the shell

Johan Öhlin

9 November 2021

Yabs

Scripts

```
#!/bin/bash
echo -n "Hello $USER. "
echo "Script $0 started"
```

Save as `script.bash`.

Executing scripts

```
#!/bin/bash
echo -n "Hello $USER. "
echo "Script $0 started"
```

Execute method:

```
$ bash script.bash
```

Make executable and run:

```
$ chmod +x script.bash
$ ./script.bash
```

read

```
#!/bin/bash
echo "What is your name?"
read NAME
echo -n "Hello $NAME. "
echo "Script $0 started"
```

What if the user doesn't provide any name?

if-then-else

```
if [[ $A -gt $B ]]; then
  ...
elif ls | grep "file"; then
  ...
else
  ...
fi
```

Conditions

Arithmetic

`$A -gt $B`	Greater than
`$A -ge $B`	Greater than or equal to
`$A -eq $B`	Equal to
`$A -le $B`	Lesser than or equal to
`$A -lt $B`	Lesser than

Conditions

Strings

`-z $string`	String has zero length
`-n $string`	String has non-zero length
`$string1 == $string2`	Exactly the same value
`$string1 != $string2`	Not the same values
`$string1 < $string2`	`$string1` goes before `$string2`
`$string1 > $string2`	`$string1` goes after `$string2`

Conditions

Files

`-a $file`	File exist
`-d $file`	File exist and is a directory
`-e $file`	File exist
`-f $file`	File exist and is a regular file
`-r $file`	File exist and is readable
`-w $file`	File exist and is writable
`-x $file`	File exist and is executable

Conditions

More can be found in the `man` page of `bash`.

Conditions

Can be logically combined:

```
 ( [[ $A -gt $B ]] || [[ $C == $D ]]) && [[ -r $FILE
]]
```

Can be written directly in the shell:

```
$ [[ -e $FILE ]] && rm -v $FILE
```

Conditions

```
#!/bin/bash
echo "What is your name?"
read NAME
if [[ -z $NAME ]]; then
  NAME=$USER
fi
echo -n "Hello $NAME. "
echo "Script $0 started"
```

Arguments

```
#!/bin/bash
echo "Number of arguments:  $#"
echo "And they are.  $@"
```

Loops

```
#!/bin/bash
echo "Number of arguments:  $#"
echo "And they are:  $@"
for arg in $@; do
  echo $arg
done
for _ in {1..10}; do
  echo "Hello"
done
for i in {1..10..3}; do
  echo "$i"
done
for((i=0; i<20; i++)); do
  echo "$i"
done
```

Variables

Assign variables on single line without spaces.

```
A=20
B="Hello World"
```

Derefer variables with `$`. Use with double quotes.

```
$ echo "$B"
Hello World
$ echo '$B'
$B
```

Variables

Simple substitution:

```
$ A="Abbas"
echo "${A#Ab}
bas
echo "${A%as}
Abb
echo "${A:3}
as
echo "${A:1:4}
bbas
echo "${A/b/x
Axbas
echo "${A//b/x
Axxas
echo "${PWD#*/
```

Array

```
#!/bin/bash
ARRAY=(a b c 1 2 3)
for e in ${ARRAY[*]}; do
  echo $e
done
echo "${ARRAY[0]}"
echo "${ARRAY[20]}"
echo "${ARRAY[-1]}"
B=${A[@]}
echo "$B"
```

Maths

```
#!/bin/bash
A=$((1+2))
echo $A
((A=A+RANDOM))
echo $A
((A++))
echo $A
```

Common programming operations are supported.

while

```
#!/bin/bash
while [[ $A -le $# ]]; do
   ((A+=1+A))
done
echo $A while true; do
  ((A++))
  if [[ $A -gt 20 ]]; then
    break
  elif [[ $A -lt 5 ]]; then
    continue
  fi
done
echo $A
```

`while` can be substituted with `util`.

switch-case

```
#!/bin/bash
case $@ in
  bar)
    command1
  ;;
  foo)
    command2
  ;;
  *)
    command3
  ;;
esac
```

getopts

`getopts` parses command options. Value is set to `$OPTARG` variable.

```
#!/bin/bash
while getopts ":h?:a" option; do
  case $@ in
    a)
      AOPTION=$OPTARG ;;
    h|?)
      AOPTION=$OPTARG ;;
    :)
      echo "No flag" ;;
    *)
      command3
    ;;
  esac
```

getopts

Tip: To unset a variable, to ensure it doesn't already exist, use `unset`.

Tip: To have a default value of a variable, you can use `${VAR:-default}`

You can iterate through variables by shifting.

```
#!/bin/bash
while [[ $1 ]]; do
  echo $1
  shift
done
```

functions

```
#!/bin/bash
function help {
  cat << eof
use:  cmd [blah blah]
eof
}
```

functions

```
#!/bin/bash
function find_txt {
  find .  -type f -name "*.txt"
}
function pretty_print {
  cat | nl
}
find_txt | pretty_print
```

alias

```
#!/bin/bash
shopt -s expad_aliases
alias find_txt='find .  -type f -name "*.txt"'
alias pretty_print="cat | nl"
find_txt | pretty_print
```

`shopt` has a bunch of other more-or-less useful options.

Run `shopt -p` to see all options set or unset.

select

```
$ select s in a b c d; do
  echo "Your favourite letter is $s!"
done
1) a
2) b
3) c
4) d
#?  4
Your favourite letter is d!
#?  2
Your favourite letter is b!
#?  x
Your favourite letter is !
#?
```