Bash quick reference

Fabio Viola

Command-line arguments

Read the arguments:

```
echo $1 $2 ...
```

Number of arguments:

```
echo $#
```

Variables

Create a variable:

```
VAR1="value"                  # global
local VAR2="other value"      # local
```

Arrays:

```
MYLIST[0]="value1"        # on the fly
${MYLIST[i]}              # access element i
${MYLIST[*]}              # access whole list
${#MYLIST[*]}             # size of the list
$MYLIST+="value"          # add an element
unset $MYLIST[i]          # remove element i
```

Arithmetic

Simple arithmetic with `let` (output is on a variable):

```
let a=2+2
let a++
```

Arithmetic with `expr` (output is on stdout):

```
expr 2 + 2
expr 2 \* 2 # note the escape char!
VAR2=$(expr 2 - $VAR1) # assignment
```

Test command for conditions

Logic comparison:

```
[ CONDITION1 -a CONDITION2 ] # and
[ CONDITION1 -o CONDITION2 ] # or
[ ! CONDITION ] # not
```

String comparison:

```
[ STRING1 = STRING2 ]    # equal
[ STRING1 != STRING2 ]   # different
```

Integer comparison:

```
[ INTEGER1 -eq INTEGER2 ]    # equal
[ INTEGER1 -ne INTEGER2 ]    # not equal
[ INTEGER1 -ge INTEGER2 ]    # greater/equal
[ INTEGER1 -gt INTEGER2 ]    # greater than
[ INTEGER1 -le INTEGER2 ]    # less/equal
[ INTEGER1 -lt INTEGER2 ]    # less than
```

File comparison:

```
[ -f FILE ] # file exists and is a regular file
[ -d FILE ] # file exists and is a directory
```

ℹ For more type `man test`.

Conditions

If-then-else:

```
if CONDITION;
    then ... ;
    elif CONDITION ; then ... ;
    else ... ;
fi
```

Switch case construct:

```
case EXPRESSION in
    VALUE1 )
        ... ;;
    VALUE2 )
        ... ;;
    ...
esac
```

Loops

For loop:

```
for VARIABLE in RANGE; do ... ; done
```

While loop:

```
while CONDITION; do ... ; done
```

Until loop:

```
until CONDITION; do ... ; done
```

Functions

Declare a function:

```
functionName () { ...; }
```

Invoke a function:

```
functionName ARG1 ARG2 ...
```

Arguments:

```
$0          # function name
$#          # number of arguments provided
$* / $@     # list of all the arguments provided
"$@"        # as above, with separated items
```

Ranges

Two ways:

```
{START..END}
seq START INCREMENT END
```

Examples with for loop:

```
for I in {1..3} ; do echo $I ; done
for I in $(seq 0 1 10) ; do echo $I ; done
```

Subshell

Assign the output of a command to a variable:

```
VAR1=$(command)
VAR2=`command`
```