

Bash quick reference

Fabio Viola

Command-line arguments

Get the program name:

```
echo $0
```

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
```

Arithmetic with `bc`:

```
echo "2+2" | bc -l
```

Generating a random number between 0 and 1:

```
echo $(( ${RANDOM}%2 ))
```

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
```

Substrings

Getting a substring of Y characters, starting from char X:

```
echo ${MYSTRING:X:Y}
```

Getting from character X to the end of the string:

```
echo ${MYSTRING:X}
```

Getting the first X characters:

```
echo ${MYSTRING:0:X}
```

Resources

```
$ man bash
```

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'
```

Wait

Wait for a process to finish:

```
./myprocess
PID_MYPROCESS=$!
tail --pid=PID_MYPROCESS -f /dev/null
```

Dates

Getting a date with a specific format:

```
$ date +"%Y/%m/%d_%dH:%M"
```

where format strings can be:

- %Y year with four digits
- %y year with two digits
- %m month with two digits
- %d day with two digits
- %H hour with two digits
- %M minute with two digits

Getting date of a specific day:

```
$ date -d "2020/02/01"
```

Getting a date with a specific offset:

```
$ date -d "2020/02/01_+4_days"
```

Redirection

Redirect both standard input and output:

```
$ some_command 2>&1 file.log
```

Redirect to file but also print on stdout:

```
$ some_command | tee file.log
```