Testing in Bash

aka Bash is Testing

Matt Turner
@mt165pro
mt165.co.uk

DevOps Days London
September 2018

```
# A command has has a stdout, a stderr, and an exit code



#include <stdio.h>
#include <stdlib.h>

void main( ) {
    printf( "don't bash me\n" );

    exit( 0 );
}
```

```
# All commands have a return value
# No matter what they write to stdout or stderr

$ ls
foo bar

$ echo $?
0
```

```
# Usually commands succeed
# Even when they don’t, you often don’t care

$ grep “needle” haystack
grep: haystack: No such file or directory

$ echo $?
2
```

```
# How do you test success / failure in a script?

$ if true; then echo “yes”; else echo “no”; fi
yes
```

```
# if is a piece of bash syntax

# it “takes precisely one argument: a command that returns 0 or non-0”

# … a command that returns 0 or non-0
```

```
$ which true
/bin/true

# true: “does nothing, successfully” [man(1) true, 1991]

$ file /bin/true
/bin/true: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV),
dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for
GNU/Linux 3.2.0, BuildID[sha1]=cae1cea2c8b5f2d151aceb44bcbe1d8415bc06c5,
stripped
```

```
$ if which flibble; then echo “yes”; echo echo “no” fi
No

# which is a command. A unary one
$ which which
/usr/bin/which
```

```
# These commands can be combined with boolean operators
# which are part of the bash syntax

# they are: ! && ||

$ if which foo || which bar; then echo “yes”; fi
```

```
# you may have seen script like this:
$ if [ “a” == “b” ]; then echo “hello”; fi
$ which [
/usr/bin/[
$ if [ -x foo ]; then echo “yes”; fi

# /usr/bin/[ is the command
# its arguments are: -x, foo, … and ]
# what does this print?
$ if [ a && b ]; then echo “yes”; else echo “no”; fi
```

```
no
```

```
# what does this print?
$ if true -a ls; then echo “yes”; else echo “no”; fi
```

```
yes
```

```
# what about this one?
$ if [ false ]; then echo “yes”; else echo “no”; fi
```

```
yes
```

```
# what about this one?
$ if [ false ]; then echo “yes”; else echo “no”; fi
```

```
yes
```

```
# syntaces can be combined
$ if [ -n “$foo” -a -f bar ] && which baz; then echo “yes”; fi

# but remember that code’s primary purpose is to be read...
# turns out, && and || are short-cct
$ if true && rm -rf /; then echo “you’re fine”; fi

# no if statement required
$ grep “needle” haystack && echo “found it”

# commonly used with /bin/[
$ [ -x foo ] && ./foo

# || even binds less tightly
$ [ -x foo ] && ./foo || echo “foo not executable”

# yay BODMAS
# turns out, && and || are short-cct
$ if true && rm -rf /; then echo “you’re fine”; fi

# no if statement required
$ grep “needle” haystack && echo “found it”

# commonly used with /bin/[
$ [ -x foo ] && ./foo

# || even binds less tightly
$ [ -x foo ] && ./foo || echo “foo not executable”

# yay BODMAS
```

Bash is Testing