This is a `test(1)`

A shell scripter's guide to ubiquitous assumption testing

Michael Dexter

Portland Linux/Unix Group
February 3rd, 2022

This is a follow-up

Lowest Common Denominator Coding
With `vi(1)` and `sh(1)`

Michael Dexter

Portland Linux/Unix Group
July 3rd, 2014

Slides are available at pdxlinux.org

`sh(1)` Scripting 101

```
# vi myscript.sh
<i> echo Hello World <ESC>
:wq
# sh myscript.sh
Hello World
```

(Note the (type)`script(1)` command)

`sh(1)` 👉 ‘`man 1 sh`’

`#` 👉 Shell prompt. Any of ‘em.
`<i>` 👉 Type “i” on the keyboard

(type)`script(1)`
Like magic, it records what flies by on the console until you type “`exit`”
BUT… “`man <command>`” is usually fine

`sh(1)` Scripting 101

```
# vi myscript.sh
<i> echo Hello World <ESC>
:wq
# sh myscript.sh
Hello World
```

(Note the (type)`script(1)` command)

Behind the Scenes: Exit Status Codes

```
# sh
# foo=beer ; echo $foo
beer
# echo $?
0
```

`0` = Success

`1` or greater = Fail

Behind the Scenes: Exit Status Codes

```
# echo $? 👈
```

Variable with the “success” or “failure” exit code/return value

Behind the Scenes: Exit Status Codes

`$?` – Most Recent Exit Status

`$1 $2 $3` – Arguments to the Command

`$*` – All Arguments to the Command

`$0` – The Name of the Command

`$#` – The Number of Arguments

Behind the Scenes: Exit Status Codes

```
# cat foo
cat: foo: No such file or
directory
# echo $?
1
```

Fail!

Behind the Scenes: Exit Status Codes

```
# date
Wed Feb  2 21:07:16 PST 2022
# echo $?
0
# date && echo Success
Wed Feb  2 21:07:18 PST 2022
Success
```

(Report Success of the test)

Behind the Scenes: Exit Status Codes

```
# cat foo || echo Failure
cat: foo: No such file or
directory
Failure
```

(Report Failure of the test)

Behind the Scenes: Exit Status Codes

```
# cat foo > /dev/null 2>&1👈 \👈
    👈|| echo Failure
Failure
```

Fancy!

“[“, Pronounced “test”

```
# foo=bar
# [ "$foo" = "bar" ]
# echo $?
0
```

Super Fancy!

man test

```
...
-d file  True if file exists and is a directory.
-e file  True if file exists (regardless of type).
-f file  True if file exists and is a regular file.
...
```

In Practice

```
#!/bin/sh
if [ ! -d ~/mydir ] ; then
  mkdir ~/mydir || \
  { echo Failed to create! ; exit 1 ; }
  echo created ~/mydir
else
  echo ~/mydir already exists
fi
```

Super mega fancy idempotence!

Idempotence (UK: /ˌɪdɛmˈpoʊtəns/, US: /ˌaɪdəm-/) is the property of certain operations in mathematics and computer science whereby they can be applied multiple times without changing the result beyond the initial application.

In Practice

```
mkdir ~/mydir || \
   { echo Failed to create! ; exit 1 ; }
```

ABORT EARLY – PLEASE!

ABORT EARLY – PLEASE!

ABORT when you are missing that ISO the whole process depends on

ABORT when that disk you think you have does not exist

Thank you for coming to my TED talk

RECAP

A few seconds of adding tests
will save you hours,
if not days,
of debugging

You’re probably thinking…

UM. OKAY. THANKS.

RABBIT HOLES

OMG! The `-q` flag!

Verified on FreeBSD...

```
# grep -q root /etc/passwd
# echo $?
0
```

Translation: Found it!

```
kldstat -q -m vmm
```

Translation: Is vmm.ko loaded?

`make(1)` manual page

```
-q        Do not execute any commands, but exit 0 if the
specified targets are up-to-date and 1, otherwise.
```

How are the `rc(8)` tests?

Put “`set -x`” in `/etc/rc.conf`

Boot, shut down

40,000+ lines of output!

Search for “not found”, “missing”, “error”, “cannot”…

```
# grep already 40818-lines-of-boot-and-shutdown-output.txt
add host 127.0.0.1: gateway lo0 fib 0: route already in table
add host ::1: gateway lo0 fib 0: route already in table
```

How are the `rc(8)` tests?

```
/etc/network.subr
ifexists()
{
    [ -z "$1" ] && return 1
    ${IFCONFIG_CMD} -n $1 > /dev/null 2>&1
}
```

If the network interface driver is not present in the kernel then **ifconfig** will attempt to load it.

The **-n** flag disables this behavior.

So one day…

Interface scalability tests on FreeBSD, OmniOS (illumos), and GNU/Linux (Debian)

Unchartered Territory

Unchartered Territory

Pop quiz!

How many `tap(4)` interfaces can FreeBSD 13.0 support?

Bonus! Is this a factual or emotional question?

Unchartered Territory

Mostly correct: “Who cares?”

But… *Never prevent someone from doing something* ***you*** *have not thought of*

Taxes & computers were “not thought of”

Unchartered Territory

FreeBSD

Creating 1024 `tap` devices 5 seconds
Testing `tap0` with `ifconfig(8)` 0.07s
Testing `tap0` with `arp(8)` 0.04s
Testing `tap0` with `if_exists`* 0.00s – Think `ifconfig -q <nic>`
Testing the time to run `ifconfig -l` 0.07s
Tearing down 1024 `tap` devices 282 seconds

* `if (if_nametoindex(argv[i]) == 0) return 1;`

Unchartered Territory

FreeBSD

Creating 10,000 `tap` devices 647 seconds

Testing `tap0` with `ifconfig(8)` 4.52s

Testing `tap0` with `arp(8)` 0.43s

Testing `tap0` with `if_exists` 0.00s

Testing the time to run `ifconfig -l` 4.58s

Tearing down 10,000 `tap` devices 1h39m9.64s

Reboot? Faster! Unload a kernel module? What the heck?

Unchartered Territory

FreeBSD

To make a long story short…
Testing the existence of *one* of 32k `tap` devices took two minutes

On an EPYC 7402p

Unchartered Territory

Debian! What was the question? (TrueNAS SCALE)

Creating 10,000 `tap` devices 107 seconds
Tearing down 10,000 `tap` devices 461 seconds
`ip a s <device>` 0.001s seconds

OmniOS CE (illumos)

Creating 10,000 `tap` devices 435 seconds
Tearing down 10,000 `tap` devices 419 seconds
`dladm show-link <dev>` 0m0.02s seconds

Chartered Territory

Think about “ifconfig -q”, “ip -q”, and “dladm -q”

Go find the scaling issues of your favorite OS

And have efficient `test(1)`*s at every step*

Peace! Questions?

@MichaelDexter on Twitter
editor@callfortesting.org

You want to give a PLUG talk!