You Suck At Shell Scripting

$ whoami

Overview

- This focuses on bash, but most topics are generally true over all shells.
- (proper) Shells are great
- Therefore shell scripts are great.
- . . . until they aren't anymore

First of all:

It puts the variables in quotes, or else it gets the hose again

. . . and this is why

```
touch "old backup" backup
cleanup="old backup"
rm $cleanup
```

Q: What happens?

This:

```
$ touch "old backup" backup
$ cleanup="old backup"
$ rm $cleanup
rm: cannot remove 'old': No such file or
    directory
```

You deleted the one thing you didn't want to.

And got shouted at.

Exceeeept. . .

. . . when you need to access individual “parts” of a variable

```
oida="1 2 3"
for dings in "$oida"; do
  echo "$dings"
done
```

Q: What happens?

This:

You get:

```
1  2  3
```

Instead of the expected:

```
1
2
3
```

Improve readability of more complex variables-in-strings-substitution constructs

. . . with curly braces

- Example:

```
awful$example
```

vs

```
nicer${example}
```

Put global flags at the beginning

- Start your commands with the globally valid flags, move towards the more specific ones towards the end.

DON'T:

```
$ kubectl get -o json pod oida --namespace
    neinneinnein
```

Obviously the official kubernetes docs show you a totally wrong example. Which is to be expected.

DO:

```
$ kubectl -n jajaja get pod oida -o json
```

Write the long version of a commands flags in a script

```
yq e -i blablabla
```

vs

```
yq eval --inplace moareblabla
```

Do you need POSIX compability?

- If not, use bash-isms without shame

Set a very specific shebang

- Otherwise you can't be certain with which shell you're dealing.
- /bin/bash might not exist
- /bin/sh might just be whatever.
- bash invoked with /bin/sh will start in posix compability mode.
  From the bash man page:
  If bash is invoked with the name sh, it tries to mimic the startup behavior of historical versions of sh as closely as possible, while conforming to the POSIX standard as well.

- This is what you want:

```
#!/usr/bin/env bash
```

Use double brackets

```
[[ if "$foo" == "$bar" ]]
```

instead of single brackets

```
 [ if "$foo" = "$bar" ]
```

- The single bracket is actually an executable/alias for `test` (see `man [`).
- Double brackets are a keyword in bash (and other shells likes ksh, etc)
- Double bracket give you a couple of nice features like `=~` and enables you to use `==`, `&&`, `||`, etc...

  More info: https://mywiki.wooledge.org/BashFAQ/031

Fail for everything, always

Do:

```
  set -eu -o pipefail
```

Otherwise errors from inside pipelines will go unnoticed:

```
set -eu
bam="$(echo "foo" | grep "bar" | sort)"
echo "i survived!"
```

Arrays suck.

- If you want arrays, use a proper programming language.
- Or does this look nice?

```
oida[0]="bam"
oida[666]="windows"
oida[23]="lizard king"
```

(yes, this is totally valid)

- . . . and using them is a good indicator your script is doing too much

Includes suck

- They make scripts hard to read.
- If you think “a library would be nice”, use a proper programming language!
- . . . and using them is a good indicator your script is doing too much.

Don't be a smartass. You'll have to debug this in a year.

- Yes, i'm looking at your overengineered jq statement.

If you need to hand over more than one argument to your script. . .

- Use `getopt` so you can have readable CLI flags and the order doesn't matter anymore.
- But the syntax isn't exactly very pretty.
- You might just want to use Python and argparse.

How to suck less

- Obey shellcheck (mostly)
  - It makes sense to store the exit code of a more complex command in a variable, instead of putting it directly in the if-statement
  - `sed 's/this/that/'` is more obvious than `${oida//this/that}`
- Keep it simple.
- Work in bash. Because then you're used to it.

Learning resources

- `man bash`
- https://tldp.org/LDP/abs/html/
- https://mywiki.wooledge.org/
  - Especially https://mywiki.wooledge.org/BashPitfalls

Questions?

Contact/Whatever

- https://mstdn.social/@kinderstampfer
- https://codeberg.org/flart