NOVEMBER 21, 2013

Best Practices for Writing Bash Scripts

This project now has its own homepage at bash3boilerplate.sh.

I recently tweeted a few best practices that I picked up over the years and got some good feedback. I decided to write them all down in a blogpost. Here goes

1. Use long options (`logger --priority` vs `logger -p`). If you're on cli, abbreviations make sense for efficiency. but when you're writing reusable scripts a few extra keystrokes will pay off in readability and avoid ventures into man pages in the future by you or your collaborators.
2. Use `set -o errexit` (a.k.a. `set -e`) to make your script exit when a command fails.
3. Then add `|| true` to commands that you allow to fail.
4. Use `set -o nounset` (a.k.a. `set -u`) to exit when your script tries to use undeclared variables.
5. Use `set -o xtrace` (a.k.a `set -x`) to trace what gets executed. Useful for debugging.
6. Use `set -o pipefail` in scripts to catch `mysqldump` fails in e.g. `mysqldump |gzip`. The exit status of the last command that threw a non-zero exit code is returned.
7. `#!/usr/bin/env bash` is more portable than `#!/bin/bash`.
8. Avoid using `#!/usr/bin/env bash -e` (vs `set -e`), because when someone runs your script as `bash ./script.sh`, the exit on error will be ignored.
9. Surround your variables with `{}`. Otherwise bash will try to access the `$ENVIRONMENT_app` variable in `/srv/$ENVIRONMENT_app`, whereas you probably intended `/srv/${ENVIRONMENT}_app`.
10. You don't need two equal signs when checking `if [ "${NAME}" = "Kevin" ]`.
11. Surround your variable with `"` in `if [ "${NAME}" = "Kevin" ]`, because if `$NAME` isn't declared, bash will throw a syntax error (also see `nounset`).
12. Use `:-` if you want to test variables that could be undeclared. For instance: `if [ "${NAME:-}" = "Kevin" ]` will set `$NAME` to be empty if it's not declared. You can also set it to `noname` like so `if [ "${NAME:-noname}" = "Kevin" ]`
13. Set magic variables for current file, basename, and directory at the top of your script for convenience.

Summarizing, why not start your next bash script like this:

```
#!/usr/bin/env bash
# Bash3 Boilerplate. Copyright (c) 2014, kvz.io

set -o errexit
set -o pipefail
set -o nounset
# set -o xtrace

# Set magic variables for current file & dir
__dir="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
__file="${__dir}/$(basename "${BASH_SOURCE[0]}")"
__base="$(basename ${__file} .sh)"
__root="$(cd "$(dirname "${__dir}")" && pwd)" # <-- change this as it depends on your

arg1="${1:-}"
```

If you have additional tips, please share and I'll update this post.

#Bash

Share: Twitter Facebook

About Kev

Kev is a dad, author of too many open source projects, and founder of Transloadit, a file encoding service for developers.

COMMENTS

Show comments

READ NEXT

SEPTEMBER 3, 2013

File Uploading Without a Server

#Amazon #Aws #Encoding #Html #Transloadit #Uploading

DECEMBER 2, 2013

Make Your MySQL Tables Strict

#Mysql