Shell Script Best Practices

27 Oct 2022

Published: *27 Oct 2022*
Author: *Shrikant Sharat Kandula*

Thank you for printing from [sharats.me](). Do check back regularly for more such awesome content. Subscribe to at sharats.me/posts/index.xml to stay updated.

This article is about a few quick thumb rules I use when writing shell scripts that I've come to appreciate over the years. Very opinionated.

Things

1. Use `bash`. Using `zsh` or `fish` or any other, will make it hard for others to understand / collaborate. Among all shells, `bash` strikes a good balance between portability and DX.

2. Just make the first line be `#!/usr/bin/env bash`, even if you don't give executable permission to the script file.

3. Use the `.sh` (or `.bash`) extension for your file. It may be fancy to not have an extension for your script, but unless your case explicitly depends on it, you're probably just trying to do clever stuff. Clever stuff are hard to understand.

4. Use `set -o errexit` at the start of your script.

   - So that when a command fails, `bash` exits instead of continuing with the rest of the script.

5. Prefer to use `set -o nounset`. You *may* have a good excuse to not do this, but, my opinion, it's best to always set it.

   - This will make the script fail, when accessing an unset variable. Saves from horrible unintended consequences, with typos in variable names.
   - When you want to access a variable that may or may not have been set, use `"${VARNAME-}"` instead of `"$VARNAME"`, and you're good.

6. Use `set -o pipefail`. Again, you may have good reasons to not do this, but I'd recommend to always set it.

- This will ensure that a pipeline command is treated as failed, even if one command in the pipeline fails.

7. Use `set -o xtrace`, with a check on `$TRACE` env variable.

   - For copy-paste: `if [[ -n "${TRACE-}" ]]; then set -o xtrace; fi`.
   - This helps in debugging your scripts, a lot. Like, really lot.
   - People can now *enable* debug mode, by running your script as `TRACE=1 ./script.sh` instead of `./script.sh`.

8. Use `[[ ]]` for conditions in `if` / `while` statements, instead of `[ ]` or `test`.

   - `[[ ]]` is a bash builtin, and is more powerful than `[ ]` or `test`.

9. Always quote variable accesses with double-quotes.

   - One place where it's *okay* not to is on the *left-hand-side* of an `[[ ]]` condition. But even there I'd recommend quoting.
   - When you need the unquoted behaviour, using `bash` arrays will likely serve you much better.

10. Use `local` variables in functions.

11. Accept multiple ways that users can ask for help and respond in kind.

    - Check if the first arg is `-h` or `--help` or `help` or just `h` or even `-help`, and in all these cases, print help text and exit.
    - Please. For the sake of your future-self.

12. When printing error messages, please redirect to stderr.

    - Use `echo 'Something unexpected happened' >&2` for this.

13. Use long options, where possible (like `--silent` instead of `-s`). These serve to document your commands explicitly.

    - Note though, that commands shipped on some systems like macOS don't always have long options.

14. If appropriate, change to the script's directory close to the start of the script.

    - And it's usually always appropriate.
    - Use `cd "$(dirname "$0")"`, which works in *most* cases.

15. Use `shellcheck`. Heed its warnings.

Template

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

set -o errexit
set -o nounset
set -o pipefail
if [[ -n "${TRACE-}" ]]; then
    set -o xtrace
fi

if [[ "$1" =~ ^-*h(elp)?$ ]]; then
    echo 'Usage: ./script.sh arg-one arg-two

This is an awesome bash script to make your life better.

'
    exit
fi

cd "$(dirname "$0")"

main() {
    echo do awesome stuff
}

main "$@"
```

Conclusion

I try to follow these rules in my scripts, and they're known to have made at least my own life better. I'm still not consistent though, unfortunately, in following my own rules. So perhaps writing them down this way will help me improve there as well.

Do you have anything you think I should add to this? Please share in the comments!

Discuss on: Hacker News.

Comments