04 – The bonus shebang

CS 2043: Unix Tools and Scripting, Spring 2019 [1]

Matthew Milano
January 30, 2019

Cornell University
Table of Contents
Caution About Shebang

- The shebang **must** be the first line.
- Generally speaking, best approach is to use `env`:
  ```
  #!/usr/bin/env bash
  #!/usr/bin/env python
  ```
- Generally, it is “wrong” to hard-code say `#!/bin/bash`
  - If I have a custom installation of `bash` that I want to use, your script will ignore me and use the default system `bash`.
  - There times **ARE** you do this, but they are very uncommon.
    - Example: program that interfaces with the operating system.
    - Then you **do** want to hard-code paths to `/bin` or `/usr/bin`.
  - Not a `#` commentable language?
    - Official answer: just don’t use a shebang.
    - Unofficial answer: technically it doesn’t matter, since the shebang is a hack on the first 8 bits, but this would render the file useless except for when it is executed by a shell.
Consider the tool `gnome-tweak-tool`. It's purpose is to alter system configurations of the desktop manager Gnome.

Their shebang:

```
#!/usr/bin/env python
```

This is “wrong”. My operating system uses `/usr/bin/python` behind the scenes for displaying windows etc.

I have a `custom` python installation that I use for daily hacking.

- `gnome-tweak-tool` uses my `custom` python, instead of using the `system` python.
- Should be using `/usr/bin/python`.

Why is it “wrong”? The `gi.repository` library imported refers to my `custom` python, not the `system` python.

This “bug” has been around for years with no change. There has to be a reason?
Shebang Details

• The Shebang does not need a space, but can have it if you want. The following all work:

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

• The `#!` is the *magic* (yes, that is the technical term):
  • The `#!` must be the very first two characters, and
  • the executable separated by whitespace *on the same line*.

• Recall that starts `#` is a comment in *bash*.
  • Technically this line is never “executed” *by the script*.
  • The *shell launching the script* to determine *how* to launch.

• In general, you will see either one space or no spaces.
  • Best to stick with one of those for consistency ;)

6
Shebang Limitations

- Generally, only safe to use two arguments in shebang:
  1. The interpreter.
  2. An optional set of arguments.

- So when you do `/usr/bin/env`, technically
  1. `/usr/bin/env` is the “interpreter”
  2. `bash` is the argument.

- This means that if you want to use `perl` or `awk` or something, you are limited to single letter flags. E.g. if you want `-a, -b, -c`, you would have to do `/usr/bin/perl -abc`.

  - `/usr/bin/env` cannot be used!
  - [Interesting mail thread][04_env_mail].

- [Amusing hacks available][04_shebang_hacks].
[1] Stephen McDowell, Bruno Abrahao, Hussam Abu-Libdeh, Nicolas Savva, David Slater, and others over the years. “Previous Cornell CS 2043 Course Slides”.