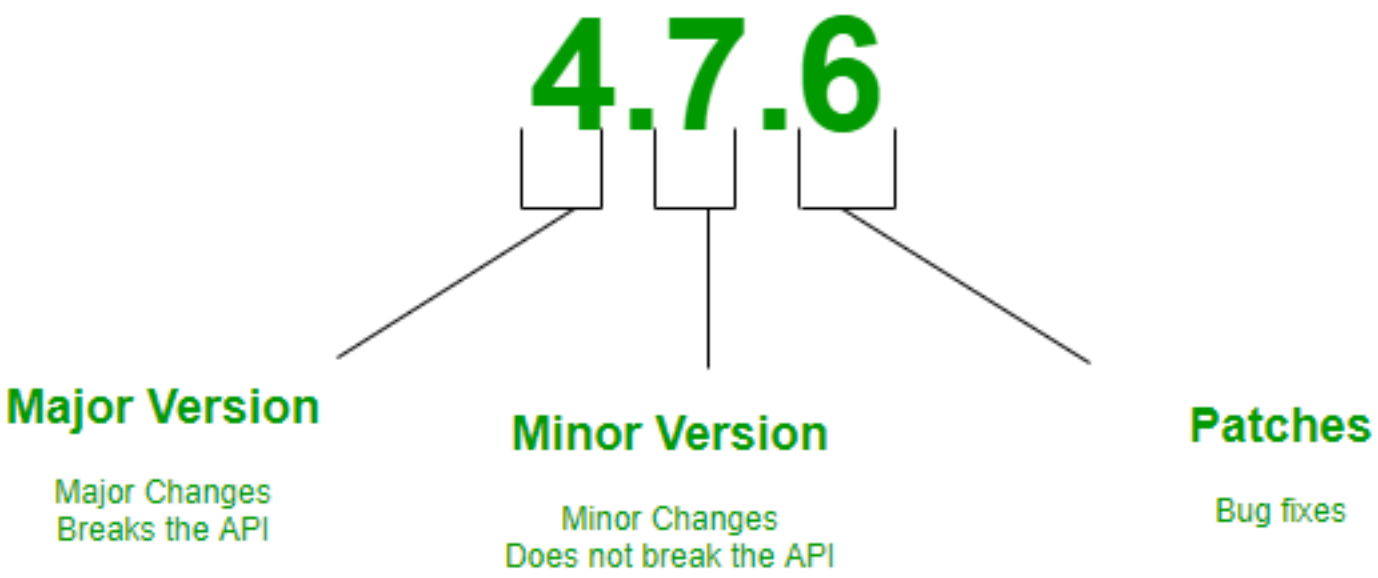
Semantic versioning, POSIX utility conventions and GNU extensions

Philip T.L.C. Clausen

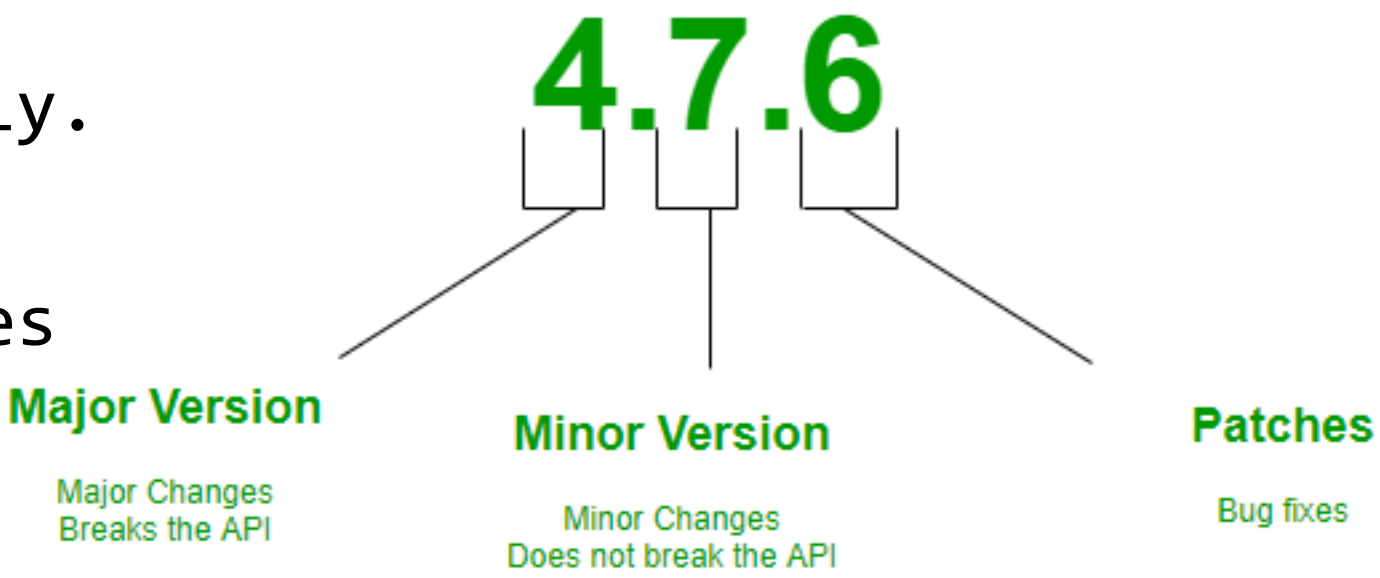
Semantic Versioning

- Comparability
- Stability
- Dependency hell
  - Locks and Promiscuity



Comparability

- Each number is to be interpreted numerically.
- Major > Minor > Patches
- Example:
  - 1.0.0 < 1.0.1 < 1.10.0 < 2.1.2



How to increment

- Major
  - Update that breaks the API
  - E.g., requiring a new dependency from the API
- Minor
  - New feature added
  - E.g., new option that enables novel treatment of data
- Patches
  - Bug fixes
  - E.g. fixed error occurring when parsing input

Stability

- Major version zero is considered unstable.
  - I.e., backwards compatibility might break at any update.
  - E.g. 0.1.1 and 0.2.0 might not work on the same machine.

- Major version above zero shall be stable.
  - I.e., if 1.3.0 works for your purpose, then 1.4.2 will work as well, 1.1.3 might not work.

- Dependency requirements may be specified as:

  KMA >= 1.4.1

  1.4.1 <= KMA < 2.0.0

Version out of control

- Won’t semantic versioning end up at version 64.0.0 quickly, when adding backwards incompatible changes are not allowed.
  - This is why we have major version zero.
  - Backwards incompatibility should not be added lightly.
  - If so, you probably do not have a lot of dependent software.

Dependency hell

The dreadful pit your software end up in, when there are not paid attention to semantic versioning. Rendering the software as un-updatable without changing the major version frequently.

- Version locks
  - Dependency versions are locked too tightly.

- Dependency promiscuity
  - Dependency versions are specified too loosely.
  - Too many dependencies.

Version locks

- If a dependency requires a specific version, your software will break whenever that dependency is updated at the API. I.e., git commits are a nogo.
- If two different dependencies require different versions of a dependency in common.
- Remember that version locks are intended when publishing results, not methods.

Solutions to version locks

- Remember that only backwards incompatible dependencies should break your software.

- Limit the number of dependencies, sharing dependencies. I.e., do not make pipelines of pipelines, as these tend to add dependencies quickly, some that might be mutually exclusive.

- Not all major version changes of dependencies breaks compatibility for your specific use case.
  E.g. CSI Phylogeny works despite changes of BWA

Dependency promiscuity

- If a dependency has a major version change, it is your responsibility to ensure that your software works with the new release as well, if a limit was not specified.

- Keeping track of a lot of dependencies quickly becomes unmanageable, especially if a few of them are at major version zero.

Solutions to dependency promiscuity

- Keep your dependencies contained, and not externally linked.
- Consider if all dependencies are needed, some might be avoided easily.
- If you have a lot of dependencies, you should probably rethink your solution entirely.

Why is Semantic Versioning important

- Without it we have no idea about when things are going to break.
- Reproducibility would be next to impossible.
- Software advancements would be tedious, slow and unstable.


v1.

POSIX Utility Conventions

Portable Operating System Interface (for unix-like systems).

General convention on how to use command line programs, how to add arguments to them and how these should be interpreted.

Currently POSIX utility conventions describe the minimal effort of how command line arguments should be interpreted, and hence used.

Options and non-options/operands

utility_name [options] [non-options/operands]

- Options SHALL be preceded by a hyphen ‘-’
- Option names are single alphanumeric characters (0-9, A-Z, a-z or check ‘man isalnum’).
- When an option takes an argument, it SHALL be parsed as: -ifoo or -i foo
- Multiple options MAY follow a hyphen.
  - I.e., ‘-ifa’ is equivalent to ‘–i –f –a’ or ‘-i fa’.
- The first ‘--’ SHALL indicate end of options.

Options and non-options/operands (MAY or SHOULD)

- Non-options MAY precede options.
  - I.e. ‘grep –v [pattern] [file]’ is equivalent to ‘grep [pattern] [file] -v’ (if accepted)
- Options SHOULD precede non-options on the command line.
- Options MAY be repeated, the options SHOULD be interpreted in the order they are given. But MAY be interpreted as the programmer sees fit (e.g. ssh -vvv).

GNU extensions

Gnu’s Not Unix

The GNU project aims to provide

FREEDOM to the user, while these extensions are mostly contained in C99, and thus considered standard today.

- Upon the POSIX utility conventions, GNU adds long options.

Long options

- Allowing only single character names for options, may give rise to un-intuitive option names.

- With long options the one-character limit is removed, where long options are denoted with a double hyphen ‘--’. E.g. --distance.

Long option conventions

- SHALL only contain alphanumeric characters.
- Words SHOULD be separated by hyphens ‘-’.
- Long options SHOULD consist of one to three words, but there is no limit as long as these are unique.
- Long options SHOULD consist of lowercase characters only.
- Arguments SHALL be given as ‘--long-option=arg’ or ‘--long-option arg’.

The most important MAY and SHOULD

- Why let non-options/operands precede options?.
- Why ‘--long-option’, and not ‘--long_option’ or ‘--longOption’.
- Why SHOULD long options be lowercase and limited to one to three words.

Take home message

- **ALWAYS** obey semantic versioning.

- Remember to adhere to the POSIX utility conventions and GNU long options.
  - If you are going to release a stable version.
  - When you increment the major version.

- For the python users ‘argparse’ has been created and takes care of most of it.
  - Does not warn about multicharacter short options.
  - Requires fixed number of arguments.

Most options and arguments are intuitive.

“thefuck” are you talking about.

```
(base) dtu-c02w41xbg8wn:~ plan$ eco Hello
-bash: eco: command not found
(base) dtu-c02w41xbg8wn:~ plan$ fuck
echo Hello [enter/↑/↓/ctrl+c]
Hello
(base) dtu-c02w41xbg8wn:~ plan$ eco hello
-bash: eco: command not found
(base) dtu-c02w41xbg8wn:~ plan$ fuck --yes
           echo hello
hello
(base) dtu-c02w41xbg8wn:~ plan$
```

Keep it simple,
Stick to the standards