Driving 3D Printers with Python

Lessons Learned

Gina Häußge // @foosel
EuroPython 2021
Gina Häußge

- Software engineer
- Maker
- Hobby baker
- 100% Nerd
- Full time OSS dev
- Creator & maintainer of OctoPrint
OctoPrint?

- “The snappy web interface for your 3D printer!”
- Open Source, AGPLv3
- ~100k confirmed users, unknown actual number
- Backend: Python
- Frontend: HTML & CSS & JS
- 👉 octoprint.org
Target audience & platform

- OctoPrint’s audience: Owners of 3D Printers = makers & tinkerers, some businesses
  - end users
  - rarely developers themselves
- Platform: Small single PCB Linux computers
  - e.g. Raspberry Pi
  - but: platform agnostic!
Challenge #1
Initial installation and keeping things up to date
Initial installation

• Initial installation happens by the end user
  • 😱 Bootstrap Python & pip install octoprint
  • 😊 Preconfigured RaspberryPi image: download, flash, wizard, done

• And on something that is not a Pi?
  • There be dragons 🐉 but it’s supported
  • Huge variety of encountered environments (Linux, Windows, Mac, FreeBSD, …)
  • Code needs to anticipate that
    • E.g. IPv6 dual stack on Windows
    • E.g. file locking differences
Keeping things up to date

- OctoPrint
  - 😱 pip install -U octoprint
  - 😊 Built-in update mechanism for OctoPrint & plugins via pip
- pip
  - 😱 You are using pip version X, however version Y is available. You should consider upgrading via the 'python -m pip install --upgrade pip' command.
  - 😊 Scary message hidden, optional auto-update
- Operating system?
  - Some do, some don’t
  - Result: unknown environment, sometimes broken environment, hard to test, additional support overhead
- “Never touch a running system!” 😞
Lessons Learned

1. Try to control runtime environment but stay flexible.

2. Make updates easy.

3. Use features as motivator for touching a running system.
Challenge #2
Maintaining backwards compatibility for a whole ecosystem
Maintaining backwards compatibility

• Plugin system since 2015 (1.2.0), over 300 plugins
• Established plugin ecosystem → this is suddenly a platform that needs stability
  • Disgruntled authors = no ecosystem = disgruntled users
• Problem: Dependency updates with breaking changes
• Problem: Code refactoring
• Problem: Python 2 vs 3 ⚡
Extraction of compatibility info from code (with some workflow debugging)

Python 3 migration script makes switching of existing instances easier

OctoPi 0.18.0 gets released & defaults to Python 3

2020-03-04 OctoPrint 1.4.0 with Python 3 compatibility
Lessons Learned

1. Read changelogs of dependencies (if you can find them) & use version pinning to your advantage.

2. Be prepared to work around breaking changes.

3. Most of all: Keep the eco system worth living in.
Challenge #3
Performance
Performance

- Webinterface + several clients + keeping a printer running + plugins + ...
- Locked to one thread at a time (GIL)
- CPU bound tasks (message parsing, checksums, file analysis, ...)
- Multiple processes non trivial due to needed data sharing (plugins) & cross platform difficulties
- Underlying platform often limited (RPi)
Lessons Learned

1. Extract CPU bound tasks if possible.

2. Be very conservative with resources.

3. Explore IPC options.
Thank you for your attention!

Twitter: @foosel
Slides: octoprint.org/slides/europython2021/