Linux Packages

Or: How to package your Python project

Quick recap

- setuptools provides entry points for console scripts
- Add “entry_points” section in setup.py:

```
entry_points={
    "console_scripts": [
        "msdp-hello=msdp.hello:sys_main",
    ]
}
```

- Format:

```
    script_name=package.module:method
```

Method

- Must return an *int*, to be used as exit code
- Can call the same method that “__main__” uses
- Example:

```
def sys_main():
    try:
        main()  # actual method doing the arg parsing/work
        return 0
    except Exception:
        return 1
```

Example code

```python
import argparse
import traceback

def main(args=None):
    parser = argparse.ArgumentParser()
    parser.add_argument("--text", help="the text to output", required=True)
    parsed = parser.parse_args(args=args)
    print(parsed.text)

def sys_main():
    try:
        main()
        return 0
    except Exception:
        print(traceback.format_exc())
        return 1

if __name__ == '__main__':
    main()
```

Packaging

- Demoing
  - stdeb (Debian)
  - py2deb (Debian)
  - fpm (Debian/RPM)
- Other tools
  - dh-virtualenv (more work involved with templates)
  - pypi2deb (old version didn’t work)

stdeb

- https://github.com/astraw/stdeb
- Uses distutils “sdist_dsc” for source packages
- “bdist_deb” for binary Debian packages
- Other commands:
  - install_deb
  - debianize

py2deb

- https://py2deb.readthedocs.io/en/latest/
- Uses pip-accel which is based on pip (< 7.2)
- Cannot handle Python 3.8+
- Does not handle “python_requires” meta-data
- Uses requirements.txt or setup.py

fpm (Effing package management!)

- https://github.com/jordansissel/fpm
- Conversion utility from one format to another
- Sources:

  gem, python, pear, dirs, tar, rpm, deb, npm, pacman
- Targets:

  deb, rpm, solaris, freebsd, tar, dirs, Mac pkg, pacman