Guix quirks when packaging a C GNU Autotools program distributing also a Python package

Ten Years of Guix – September 16, 2022

Philippe SWARTVAGHER
Inria Bordeaux – Sud-Ouest

In 10 minutes

- Weird things with Guix packaging…
- …when packaging a weird software

- I will ask many questions
  - for which I didn’t find answers

- Disclaimer: feedback from work done in December 2021

- The whole feedback: https://gitlab.com/phsw/snippets/-/tree/master/packaging-c-and-python

Context

- C program
  - Generates some output
  - Autotools
- Python package and scripts
  - To analyze the program output
  - Python package managed (installed) with `pip`
- Autotools:
  - Builds and installs the C program
  - Launches `pip` commands for the Python package

```
.
├── python
│   ├── ph_prog # the Python package
│   │   ├── __init__.py
│   │   └── output_analyzer.py
│   └── ph_prog_analyze # the Python script
└── src
    └── ph_prog.c # the C program
```

```
./autogen.sh
cd build
../configure
make
sudo make install
ph_prog | ph_prog_analyze
sudo make uninstall
```

Guix

Let’s create a Guix package!

What could go wrong?

The plan

1) Use a `(build-system gnu-build-system)`

2) Add Python/`pip`-related `native-inputs`

   or: make two packages

Problem #1
How to use local sources?

Local sources

- My proof-of-concept: no upstream release, no Git repository, **only a local folder**
- What to put in `(source …)`?
  - It’s a mandatory field

Local sources

- Hack: package transformation
- Let’s use a dummy `(origin …)`
  - `--with-source=package=$(pwd)`

From template in https://guix.gnu.org/en/manual/devel/en/html_node/Defining-Packages.html

```
(define-public c-python
 (package
  (name "c-python")
  (version "0.1")
  (source (origin
           (method url-fetch)
           (uri (string-append "mirror://gnu/hello/hello-2.10.tar.gz"))
           (sha256
            (base32
             "0ssi1wpaf7plaswqqjwigppsg5fyh99vdlb9kzl7c9lng89ndq1i"))))
  (build-system gnu-build-system)
  (native-inputs (list python-pip python-setuptools python-wheel))
  (synopsis "Hello, GNU world: An example GNU package")
  (description "Guess what GNU Hello prints!")
  (home-page "https://www.gnu.org/software/hello/")
  (license gpl3+)))
```

Local sources

```
guix build -L . c-python --with-source=c-python=$(pwd)
```

Local sources

```
guix build -L . c-python --with-source=c-python=$(pwd)
```

```
guix build -L . c-python --with-source=c-python=.
guix build: erreur : invalid name: `.'
```

Problem #2
Why aren’t there « implicit » dependencies ?

Implicit dependencies

```
guix build -L . c-python -with-source=c-python=$(pwd)
…
checking for a Python interpreter with version >= 3.6... none
```

But I put `(native-inputs (list` **`python-pip python-setuptools python-wheel`**`))`!

Isn’t it obvious they require some Python interpreter?

Implicit dependencies

```
guix build -L . c-python -with-source=c-python=$(pwd)
…
checking for a Python interpreter with version >= 3.6... none
```

But I put `(native-inputs (list` **`python-pip python-setuptools python-wheel`**`))`!

Isn’t it obvious they require some Python interpreter?

Ok, let’s just complete the list:

`(native-inputs (list` **`python`** `python-pip python-setuptools python-wheel))`

Implicit dependencies (again)

```
guix build -L . c-python -with-source=c-python=$(pwd)
…
./autogen.sh: line 2: autoreconf: command not found
```

But I put `(build-system gnu-build-system)`!

Isn’t it obvious it requires some autotools?

Implicit dependencies (again)

```
guix build -L . c-python -with-source=c-python=$(pwd)
…
./autogen.sh: line 2: autoreconf: command not found
```

But I put `(build-system gnu-build-system)`!

Isn’t it obvious it requires some autotools?

Ok, let’s just complete the list:

```
(native-inputs (list python python-pip python-setuptools python-wheel
                     autoconf automake))
```

Problem #3 (fun fact?)
Guix really uses all sources!

Using all sources

- For debug purposes, had to add `-v` in a Makefile
- Strange: the `-v` doesn’t appear in Guix logs
- Why?

Using all sources

- Hint: I’m using `--with-source=$(pwd)`
- Counter intuitive hint: I’m using
  `(arguments '(#:out-of-source? #t))`

Using all sources

- Hint: I’m using `--with-source=$(pwd)`
- Counter intuitive hint: I’m using
  `(arguments '(#:out-of-source? #t))`

- All `autotools` generated files were already in `$(pwd)`
  - Not updated by Guix/autotools with changes in the original Makefile
  - `rm -rf autom4te.cache aclocal.m4 build* configure **/*.in`

Problem #...
And other problems

And other problems

- Autoconf version 2.71 available in Guix, but by default version 2.69 is used
- Prevent `pip` from accessing the network (ok, not Guix's fault)
- `ValueError: ZIP does not support timestamps before 1980`
- See https://gitlab.com/phsw/snippets/-/tree/master/packaging-c-and-python for the whole story

Conclusion

```
% guix shell -L . c-python --pure --with-source=c-python=$(pwd)
% ph_prog | ph_prog_analyze
Output of ph_prog: [1, 2, 3] sum computed by ph_prog_analyze=6
```

\o/