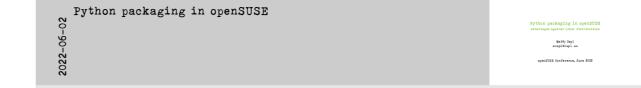
Python packaging in openSUSE Advantages against other distributions

> Matěj Cepl mcepl@cepl.eu

openSUSE Conference, June 2022



▲□▶ ▲圖▶ ▲≣▶ ▲≣▶ 三回 めんの

Problems in the Python non-specific packaging

Multiple interpreters

- Complicated commands
- Sheer number of packages
- Not enough checking

Python packaging in openSUSE

-02

90

2022-

Problems in the Python non-specific packaging

Multiple interpreters
 Complicated commands
 Sheer number of packages
 Not enough checking

Multiple interpreters

%build %if

%else

%endif

	Python	packaging	in openSUSE
02	·		
-90			
	1		
2022	L	-Multiple	interpreters
8			

The other solution is hard-coded, so whenever anything changes, everything
has to be changed again. Other solution is the same as what we used to use
for Python 2 (and we still use for Python2 in some enterprise channels), and
it is the same now when we use it for simultaneous support of Python 3.8, 3.9,
3.10 and possibly soon 3.11 on Tumbleweed, and it would be the same if somebody
finally fixed pypy3 or if a miracle happened and somebody got jython to the
functional state.

Multiple interpreters

Xbuild Xif X{with python3} Xpy3_build Xelse Xpy2_build Xendif

Multiple interpreters

%build %if

%else

%endif

```
Python packaging in openSUSE Relifie interpreters
```

The other solution is hard-coded, so whenever anything changes, everything
has to be changed again. Other solution is the same as what we used to use
for Python 2 (and we still use for Python2 in some enterprise channels), and
it is the same now when we use it for simultaneous support of Python 3.8, 3.9,
3.10 and possibly soon 3.11 on Tumbleweed, and it would be the same if somebody
finally fixed pypy3 or if a miracle happened and somebody got jython to the
functional state.

72build

Zowthon build

%build

Complicated commands

%check

%{buildroot}%{python_sitelib}

Python packaging in openSUSE

And this is still not correct, because it must be done for all Python versions separately (and those --ignore=build* must be arranged accordingly). And we need to be able to take into consideration existing PYTHONPATH.

Complicated commands

nutest -- impreadmild a -v

export PYTHONPATH+%{buildroot}%{python_sitelib} PYTHONDONTWRITEBYTECODE

Complicated commands

%check

%{buildroot}%{python_sitelib}

%check

-06-02	Python packaging in openSUSE	Complicated commands
		<pre>%chack export YTHONFATH-{(buildroot)}(python_sitelib) PYTRONDONTWRITESYTECCOD-1 pytestignore-build.* -v</pre>
2022	-Complicated commands	% Zpytest

And this is still not correct, because it must be done for all Python versions separately (and those --ignore=build* must be arranged accordingly). And we need to be able to take into consideration existing PYTHONPATH.

Example

%define python_module() python3-%{**}}

Name Version Release

Summary

License

URL

Source

%{version}

PATCH-FIX-UPSTREAM remove_mock.patch bsc#123456 mcepl@suse.com

we don't need stinking mock

Patch0

BuildRequires

BuildRequires

BuildRequires

BuildRequires

Requires

BuildArch

%description

Python packaging in openSUSE 02 ġ 2022--Example



 Eliminate as much boiler-plate as possible. We have only two lines now, which are same every time: that python_module definition and Release. And in both cases we are just forced to have them there by the mechanics of RPM.
 Note that macro python_subpackages, that's the root of all machinations. In effect this SPEC file is just a foundation, I don't want to use the term "template,, because that would be misleading, for multiple generated ones.

Exam	aple (cont.)
%pi	rep %{version}
%Ът	uild
%i1	nstall
	%{buildroot}
	heck %pyunittest discover -v
%f:	iles
	oc python_sitelib} python_sitelib} %{version}

Python packaging in openSUSE Each State (cont.)

Tchangelog

1. Using autosetup to avoid dealing with each patch again.

Toil to machines!

- Given the number of packages we maintain (over 2,500 in Factory) we need to put as much work as possible on machines.
- Packages are auto-generated by 'py2pack'.
- Automatic rebuilds
- All submissions are reviewed
- Every build in openSUSE ecosystem is checked by rpmlint and unless specifically permitted, failed rpmlint check means failed build.

Python packaging in openSUSE 02 ġ 2022--Toil to machines!

Given the number of packages we maintain (over 2,500 in Factory) we need to got as much work as possible on machines.
 Fackages are noto-generated by 'pylpack'.
 Astomatic retails
 All habitations are verticed

Toil to machines!

All submissions are reviewed
 Every build in openSUSE ecceptem is checked by rpmlint and unless specifically permitted, failed rpmlint check means failed build.

- They are auto-generated, but they are more like a ready-to-cook food, they need to be finished.
- 2. And yes, it is similar to the Fedora's pyp2rpm, but we have never managed to unify two code-bases. If anybody is willing to do the work, it would be lovely.
- 3. I may add an anecdote about

Thank you!

