NAME

dh_pypy - calculates PyPy dependencies, adds maintainer scripts to byte compile files, etc.

SYNOPSIS

dh_pypy -p PACKAGE [-V [X.Y][-][A.B]] DIR [-X REGEXPR]

DESCRIPTION

QUICK GUIDE FOR MAINTAINERS

- build-depend on pypy and dh-python,
- add {*pypy:Depends*} to Depends
- build module/application using its standard build system,
- install files to the standard locations,
- add *pypy* to dhs --with option, or:
- call **dh_pypy** in the *binary-** target,

NOTES

dependencies

dh_pypy tries to translate Python dependencies from requires.txt file to Debian dependencies. Use debian/pypydist-overrides or --no-guessing-deps option to override it if the guess is incorrect. If you want dh_pypy to generate more strict dependencies (f.e. to avoid ABI problems) create debian/pypy-foo.pydist file. See /usr/share/doc/dh-python/README.PyDist for more information. If the pydist file contains PEP386 flag or set of (uscan like) rules, dh_pypy will make the depedency versioned (version requirements are ignored by default).

private dirs

/usr/share/foo, */usr/share/games/foo*, */usr/lib/foo* and */usr/lib/games/foo* private directories are scanned for Python files by default (where *foo* is binary package name). If your package is shipping Python files in some other directory, add another dh_pypy call in debian/rules with directory name as an argument - you can use different set of options in this call. If you need to change options for a private directory that is checked by default, invoke dh_pypy with --skip-private option and add another call with a path to this directory and new options.

debug packages

In binary packages which name ends with *-dbg*, all files in */usr/lib/pypy/dist-packages/* directory that have extensions different than *so* or *h* are removed by default. Use --no-dbg-cleaning option to disable this feature.

overriding supported / default PyPy versions

If you want to override systems list of supported PyPy versions or the default one (f.e. to build a package that includes symlinks for older version of PyPy or compile .py files only for given interpreter version), you can do that via *DEBPYPY_SUPPORTED* and/or *DEBPYPY_DEFAULT* env. variables.

OPTIONS

--version
show programs version number and exit

-h, **--help**
show help message and exit

--no-guessing-deps
disable guessing dependencies

--no-dbg-cleaning
do not remove any files from debug packages

--no-ext-rename do not add magic tags nor multiarch tuples to extension file names

--no-shebang-rewrite
do not rewrite shebangs

--skip-private
dont check private directories

-v, **--verbose**
turn verbose mode on

-i, **--indep**
act on architecture independent packages

-a, **--arch**
act on architecture dependent packages

-q, **--quiet**
be quiet

-p *PACKAGE*, **--package**=*PACKAGE*
act on the package named PACKAGE

-N *NO_PACKAGE*, **--no-package**=*NO_PACKAGE*
do not act on the specified package

-X *REGEXPR*, **--exclude**=*REGEXPR*
exclude items that match given REGEXPR. You may use this option multiple times to build up a list of things to exclude.

--compile-all
compile all files from given private directory in postinst/rtupdate not just the ones provided by the package (i.e. do not pass the --package parameter to py3compile/py3clean)

--depends=*DEPENDS*
translate given requirements into Debian dependencies and add them to ${pypy:Depends}. Use it for missing items in requires.txt

--recommends=*RECOMMENDS*
translate given requirements into Debian dependencies and add them to ${pypy:Recommends}

--suggests=*SUGGESTS*
translate given requirements into Debian dependencies and add them to ${pypy:Suggests}

--requires=*FILENAME*
translate requirements from given file(s) into Debian dependencies and add them to ${pypy:Depends}

--shebang=*COMMAND*
use given command as shebang in scripts

--ignore-shebangs
do not translate shebangs into Debian dependencies

SEE ALSO

- /usr/share/doc/dh-python/README.PyDist
- pybuild(1)
- *http://deb.li/dhpy* - most recent version of this document

AUTHOR

Piotr Ożarowski, 2013