numpy / **numpy**

fortran extensions on OSX, issue with defining LDFLAGS #7427

New issue

Closed **ddale** opened this issue on Mar 17, 2016 · 8 comments

ddale commented on Mar 17, 2016 Contributor

I ran into a problem while trying to build a conda package (https://github.com/praxes/hedm-conda-recipes/tree/master/imaged11) for a project with fortran extensions. `python setup.py build` works fine, but if`conda build`` simply invokes that same command, I get a string of errors about f2py not finding python symbols:

```
[...]
compiling Fortran sources
Fortran f77 compiler: /Users/darren/.conda/envs/_build/bin/gfortran -Wall -g -ffixed-form -fno-
second-underscore -arch x86_64 -fPIC -O3 -funroll-loops
Fortran f90 compiler: /Users/darren/.conda/envs/_build/bin/gfortran -fopenmp -O2 -arch x86_64 -
fPIC -O3 -funroll-loops
Fortran fix compiler: /Users/darren/.conda/envs/_build/bin/gfortran -Wall -g -ffixed-form -fno-
second-underscore -fopenmp -O2 -arch x86_64 -fPIC -O3 -funroll-loops
creating build/temp.macosx-10.5-x86_64-2.7/fsrc
compile options: '-Ibuild/src.macosx-10.5-x86_64-2.7 -
I/Users/darren/.conda/envs/_build/lib/python2.7/site-packages/numpy/core/include -
I/Users/darren/.conda/envs/_build/include/python2.7 -c'
gfortran:f90: fsrc/fImageD11.f90
/Users/darren/.conda/envs/_build/bin/gfortran -Wall -g -Wl,-
rpath,/Users/darren/.conda/envs/_build/lib -arch x86_64 build/temp.macosx-10.5-x86_64-
2.7/build/src.macosx-10.5-x86_64-2.7/fImageD11module.o build/temp.macosx-10.5-x86_64-
2.7/build/src.macosx-10.5-x86_64-2.7/fortranobject.o build/temp.macosx-10.5-x86_64-
2.7/fsrc/fImageD11.o -L/Users/darren/.conda/envs/_build/lib/gcc/x86_64-apple-darwin11.4.2/4.8.5
-L/Users/darren/.conda/envs/_build/lib -lgomp -lpthread -lgfortran -o build/lib.macosx-10.5-
x86_64-2.7/ImageD11/fImageD11.so
Undefined symbols for architecture x86_64:
  "_PyArg_ParseTupleAndKeywords", referenced from:
      _f2py_rout_fImageD11_compute_xlylzl in fImageD11module.o
      _f2py_rout_fImageD11_assign in fImageD11module.o
      _f2py_rout_fImageD11_compute_gv in fImageD11module.o
  "_PyCObject_AsVoidPtr", referenced from:
      _initfImageD11 in fImageD11module.o
      _F2PyCapsule_AsVoidPtr in fortranobject.o
  "_PyCObject_FromVoidPtr", referenced from:
      _fortran_getattr in fortranobject.o
      _F2PyCapsule_FromVoidPtr in fortranobject.o
[...]
```

What I found was that `conda build` defines `LDFLAGS= -Wl,-rpath,/Users/darren/.conda/envs/_build/lib -arch x86_64`, and if I `unset LDFLAGS` in my conda recipe's build script, the package will compile. I asked about it on the conda mailing list, and someone replied that it appears f2py does not extend user-defined LDFLAGS, but instead let's user-defined LDFLAGS overwrite its own (https://lists.macosforge.org/pipermail/macports-users/2010-November/022574.html), which on osx include "-undefined dynamic_lookup -bundle". As it stands, it appears that anyone who needs to build a conda package on osx that includes fortran extensions will have to add the following to their build scripts:

Assignees

No one assigned

Labels

01 - Enhancement

Proposal

component: numpy.distutils

Projects

None yet

Milestone

1.16.0 release

Notifications

5 participants

```
LDFLAGS="$LDFLAGS -undefined dynamic_lookup -bundle"
```

Could f2py or numpy distutils extend user's LDFLAGS with these required settings?

👍 1

charris added **01 - Enhancement** **component: numpy.fft** **Proposal** labels on Mar 17, 2016

rgommers commented on Mar 17, 2016 Member

This is known `numpy.distutils` behavior for as long as I can remember.

Some digging would be needed to figure out why it was chosen to implement things this way. I can imagine there are usecases where one needs to get rid of something that `numpy.distutils` or `distutils` adds by default.

ddale commented on Mar 17, 2016 Contributor

Maybe a warning would be in order then, advising that if one is going to define their own LDFLAGS, it is highly advisable to include x, y, and z.

rgommers added **component: numpy.distutils** and removed **component: numpy.fft** labels on Jul 5, 2016

wlattner referenced this issue on Jul 19, 2016

ENH: add conda recipe #4 Closed

isuruf referenced this issue on Nov 15, 2017

Scipy build failures with PREFIX on include / library path #76 Closed

isuruf commented on Jul 7 Contributor

This is a really annoying behaviour, we see in conda-forge. Would you accept a patch to fix this?

> I can imagine there are usecases where one needs to get rid of something that numpy.distutils or distutils adds by default.

Appending is the normal behaviour for C extensions, so why not Fortran?

isuruf commented on Jul 7 Contributor

Also flags for `linker_so` and `linker_exe` are both overriden by the same flag `LDFLAGS`, which makes it impossible to have extra LDFLAGS since they need different sets of flags.

isuruf referenced this issue on Jul 7

Fix numpy distutils ldflags overriding behaviour #89 Merged

3 of 4 tasks complete

rgommers commented on Jul 7 Member

> This is a really annoying behaviour, we see in conda-forge. Would you accept a patch to fix this?

I'm a little hesitant here, because it will break existing build automation for people who use `LDFLAGS` now. So this will need discussion on the mailing list.

Question: is it just annoying, or does it actually prevent some things you want to do?

> Also flags for linker_so and linker_exe are both overriden by the same flag LDFLAGS, which makes it impossible to have extra LDFLAGS since they need different sets of flags.

This seems possible to change without backwards compatibility issues, by introducing some new envvars.

pv commented on Jul 7 Member

As a workaround, NPY_DISTUTILS_APPEND_FLAGS=1 env var could be a
backward-compatible option (or at least would offer time for
deprecation)...

isuruf commented on Jul 7 Contributor

> Question: is it just annoying, or does it actually prevent some things you want to do?

At conda-forge, we set some LDFLAGS for external libraries to be found (-L$PREFIX/lib) and flags like `-headerpad_max_install_names`. This is done for all packages using compilers, not just python packages. Flags in LDFLAGS env var is meant to be appended in every other build system including python's distutils, so that's why conda-forge sets it.

👍 1

rgommers commented on Jul 7 Member

> As a workaround, NPY_DISTUTILS_APPEND_FLAGS=1 env var could be a backward-compatible option (or at least would offer time for deprecation)...

That makes sense to me. Then we could add some noisy warnings if it's not set, and we'd be in no hurry to finally flip the switch, maybe leave that for 2.0 or 4-5 releases.

isuruf referenced this issue on Jul 7

Append *FLAGS instead of overriding #11525 Merged

rgommers closed this in #11525 on Jul 14

rgommers added this to the **1.16.0 release** milestone on Jul 14