Crossenv

Release 0.7

Benjamin Fogle

Aug 23, 2023

CONTENTS:

1 Introduction **3**
- 1.1 Requirements . . . 3
- 1.2 Vocabulary . . . 3
- 1.3 How it works . . . 3
- 1.4 Installation . . . 4

2 "Quick" Start **5**
- 2.1 Build build-python . . . 5
- 2.2 Build or obtain host-python . . . 5
- 2.3 Make the cross environment . . . 6
- 2.4 Use the resulting packages . . . 6

3 Cross-Compiling Environment **7**
- 3.1 Environment variables . . . 7
- 3.2 The `sys` module . . . 7
- 3.3 The `os` module . . . 7
- 3.4 The `platform` module . . . 8

Index **9**

Note: This documentation is in the early stages.

Porting a Python app to an embedded device can be complicated. Once you have Python built for your system, you may find yourself needing to include many third-party libraries. Pure-Python libraries usually just work, but many popular libraries rely on compiled C code, which can be challenging to build.

This package is a tool for cross-compiling extension modules. It creates a special virtual environment such that `pip` or `setup.py` will cross compile packages for you, often with no further work on your part.

It can be used to:

- Build binary wheels, for installation on target.
- Install packages to a directory for upload or inclusion in a firmware image.

Note: While this tool can cross-compile *most* Python packages, it can't solve all the problems of cross-compiling, and it can't make cross-compiling a completely pain-free process. In some cases manual intervention may still be necessary.

This tool requires Python 3.5 or higher (host and build). Significant work has gone into cross-compiling Python in newer versions, and many of the techniques needed to do the cross compilation properly are not available on older releases.

This tool currently only supports Linux build machines.

CHAPTER

ONE

INTRODUCTION

1.1 Requirements

Crossenv requires Python 3.5 or higher (host and build). Significant work has gone into cross-compiling Python in newer versions, and many of the techniques needed to do the cross compilation properly are not available on older releases.

Crossenv currently only supports Linux build machines. Other operating systems may work, but they are untested and unsupported.

1.2 Vocabulary

There is no standard vocabulary for the pieces that go into cross-compiling, and different resources will often use contradictory terms. To prevent confusion we use the GNU terminology exclusively, which is used by Python itself.

Host	The machine you are building **for**. (Android, iOS, other embedded systems.)
Build	The machine you are building **on**. (Probably your desktop.)
Host-python	The compiled Python binary and libraries that run on Host
Build-python	The compiled Python binary and libraries that run on Build.
Cross-python	Build-python, configured specially to build packages that can be run with Host-python. This tool creates Cross-python.

1.3 How it works

Python makes a note of the compiler and compiler flags used when it was built. (This information can be viewed by running `python3 -m sysconfig`.) When `distutils` or `setuptools` attempts to create an extension module, they compile the extension using these recorded values along with reported information about the currently running system.

Cross-python creates a virtual environment that, when activated, tricks Build-Python into reporting all system information exactly as Host-python would. When done correctly, a side effect of this is that `distutils` and `setuptools` will cross-compile when building packages. All of the normal packaging machinery still works correctly, so dependencies, ABI tags, and so forth all work as expected.

1.4 Installation

Crossenv can be installed using pip (using build-python):

```
$ pip install crossenv
```

CHAPTER

TWO

"QUICK" START

Cross compiling can be challenging, and crossenv is focused only on one particular piece. As such, this section is not a complete guide.

2.1 Build build-python

Don't trust a build-python that you didn't build yourself. The version of python that comes with a Linux distribution is usually patched by the maintainers in ways that are subtly incompatible with the stock Python source. Normally this isn't an issue, but when using crossenv, build-python will end up running some of host-python's (unpatched) code. The end result is one of many obscure errors.

Build-python and host-python must be the exact same version. As above, one may need to execute the other's bytecode, which only works if they have the same version.

For general build instructions refer to the Python Developer's Guide. You don't need a debug build, so `configure --prefix=/path/to/build-python` is usually enough.

At a minimum, you need zlib and openssl to build. Since build-python only exists to build packages, you can often get away with leaving most optional components disabled. It's usually sufficient to build just enough to get pip working, which requires the `ssl` and `zlib` modules. (More complicated builds may require more. It depends very much on your specific requirements.)

2.2 Build or obtain host-python

In this quick start we assume you are building host-python yourself. In other cases you may be targeting a pre-built system image. A pre-built image has it's own challenges, which are covered elsewhere.

You will need to build any host dependencies beforehand. So, for example, if you want host-python to be able to communicate over a network, you may need to cross-compile OpenSSL. Building these dependencies is beyond the scope of this project.

Building host-python requires a working build-python. We recommend putting build-python in your `$PATH` for the configure script to find. Here is an example of a configure command used for testing crossenv against an ARM host:

```
$ PATH=/path/to/build-python/bin:$PATH \
    ./configure --prefix=/path/to/host-python \
                --host=arm-linux-musleabihf \
                --build=x86_64-linux-gnu \
                --without-ensurepip \
                ac_cv_buggy_getaddrinfo=no \
                ac_cv_file__dev_ptmx=yes \
```

(continues on next page)

(continued from previous page)

```
                ac_cv_file__dev_ptc=no
$ make
$ make install
```

The *–host* option specifies the host triplet, such as `aarch64-linux-gnu`. Python will expect a compiler in you `$PATH` named `aarch64-linux-gnu-gcc`, but this can be overridden by passing `CC=/path/to/cc` on the command line. You can use `CFLAGS` and `LDFLAGS` to point Python to any dependencies it needs.

The `ac_cv_*` arguments are to set information about the system that configure isn't able to determine when cross compiling. The first, `ac_cv_buggy_getaddrinfo=no` allows IPv6, and the other two are for the benefit of os.openpty. You may not need any of this functionality, but you still need to supply the parameters.

2.3 Make the cross environment

First install crossenv:

```
$ /path/to/build-python/bin/pip3 install crossenv
```

Build the cross-virtual environment:

```
$ /path/to/build-python/bin/python3 -m crossenv \
    /path/to/host-python/bin/python3 \
    cross_venv
```

Activate the cross-virtual environment:

```
$ . ./cross_venv/bin/activate
```

Build something:

```
(cross) $ pip install numpy
(cross) $ python setup.py install
```

Packages that you need at *build* time are best installed in build-python explicitly:

```
(cross) $ build-pip install cffi
(cross) $ pip install bcrypt

(cross) $ build-pip install wheel
(cross) $ python setup.py bdist_wheel
```

2.4 Use the resulting packages

It's up to you to incorporate your cross-compiled module into your project. It might be easiest to create a wheel, and then unzip it at the right location. If you did a *pip install* you can find the installed libraries at `cross_venv/cross/lib/python`*`VERSION`*`/site-packages`.

CHAPTER

THREE

CROSS-COMPILING ENVIRONMENT

This section documents the changes to the Python environment beyond just making `cross-python` behave like `host-python`.

3.1 Environment variables

Crossenv sets `PYTHON_CROSSENV` to a non-empty value.

3.2 The `sys` module

`sys.`**`cross_compiling`**
: Set to `True` in `cross-python`. May be used like so:

```
if getattr(sys, 'cross_compiling', False):
    ...
```

`sys.`**`build_path`**
: Analagous to `sys.path`, but applies when importing packages from `build-python`. This path is searched just before the entries on `sys.path` that point to the Python standard library. This means that *`sys.build_path`* is preferred when loading modules from the standard library, but prepending to `sys.path` still works as expected.

3.3 The `os` module

`os.`**`uname`**`()`
: In addition to returning `host-python`'s information, it always reports the `node` as "`build`".

3.4 The `platform` module

`platform.`**`uname`**`()`
In addition to returning `host-python`'s information, it always reports the `node` as "`build`".

INDEX

Symbols

$PATH, 5, 6

E

environment variable
    $PATH, 5, 6

O

os.uname() (*built-in function*), 7

P

platform.uname() (*built-in function*), 8

S

sys.build_path (*built-in variable*), 7
sys.cross_compiling (*built-in variable*), 7