Developing cross-platform CPython extensions

1. Create simple examplemodule/CMakeLists.txt describing the extension

```
cmake_minimum_required(VERSION 2.8.9)
project(examplemodule)
find_package(Python REQUIRED CONFIG)
include_directories(${PYTHON_INCLUDE_DIRS})
add_library(example MODULE examplemodule.cxx)
target_link_libraries(example ${PYTHON_LIBRARIES})
set_target_properties(example PROPERTIES PREFIX "")
```

2. Create examplemodule.cxx implementing the extension

```
3. Configure and build
 mkdir examplemodule && cd $_
 cmake -DPython_DIR=${HOME}/scratch/python-build ../examplemodule
 make -j4
```

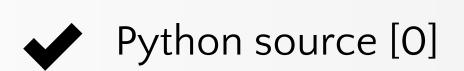
Require CMake >= 2.8.9

github.com/jcfr/python-cmake-custom-extension

Motivation

- Maintainable build system
- Easy embedding of CPython
- Built-in support for cross-compilation
- First class support for Visual Studio







Python CMake build system [2]

CMake build system for CPython Simple with built-in support for cross-compilation.

Jean-Christophe Fillion-Robin, Matt McCormick



davidsansome/python-cmake-buildsystem

Install

Generation of pkg-config file

cmake \

make install

Configurable install prefix

Install tree layout similar to "Autoconf" one

-DCMAKE_INSTALL_PREFIX=/path/to/python-install

One simple language for all platforms

Generates native build system

Kitware

What is CMake?

- Cross-platform
- Open-source BSD-like license
- Self-contained No dependencies
- Large community

CMake generators

A CMake Generator is responsible for writing the input files for a native build system.

Use cmake - G option to specify the generator for a new build tree.

Extra Generators for auxiliary IDE

CodeBlocks CodeLite Eclipse CDT4 KDevelop3 Kate

Sublime Text 2

Command-Line Build Tool Generators

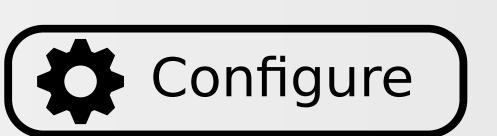
Borland Makefiles MSYS Makefiles MinGW Makefiles NMake Makefiles NMake Makefiles JOM Ninja

Unix Makefiles

Watcom WMake

IDE Build Tool Generators

Visual Studio 6 Visual Studio 7 Visual Studio 7 .NET 2003 Visual Studio 8 2005 Visual Studio 9 2008 Visual Studio 10 2010 Visual Studio 11 2012 Visual Studio 12 2013 Xcode



Support for cross-compilation

libpython: shared and/or static

Python modules: shared or built-in

Dependencies: system or explicit

Support for in or out of source build

Configurable install prefix



Cross-compiling for RaspberryPi

1. Build the toolchain using crosstool-ng [5]

2. Create Toolchain-RaspberryPi.cmake [6] set(CMAKE_C_COMPILER \${toolchain}/bin/arm-unknown-linux-gnueabi-gcc)

set(CMAKE_FIND_ROOT_PATH \${toolchain}/arm-unknown-linux-gnueabi/sysroot)

3. Configure

mkdir python-install-pi && mkdir python-build-pi && cd \$_ cmake -DCMAKE_TOOLCHAIN_FILE=/path/to/Toolchain-RaspberryPi.cmake \ -DCMAKE_INSTALL_PREFIX=/home/jchris/sratch/python-install-pi \ ../python-cmake-buildsystem

4. Edit TryRunResults.cmake with expected values

5. Re-configure

cmake -C TryRunResults.cmake -DCMAKE_TOOLCHAIN_FILE=Toolchain-RaspberryPi.cmake -DCMAKE_INSTALL_PREFIX=/home/jchris/sratch/python-install-pi ../python-cmake-buildsystem

6. Cross-compile 7. Upload to target

make install

Ubuntu 13.10 / CMake 2.8.

Build

Unix/Ubuntu build (see ReadMe for Windows and others)

Install build tools sudo apt-get install build-essential cmake git

Create directory mkdir ~/scratch && cd \$_

Download python source wget python.org/ftp/python/2.7.3/Python-2.7.3.tgz tar -xzvf Python-2.7.3.tgz

Download buildsystem git clone git@github.com:\ davidsansome/python-cmake-buildsystem.git

Configure

mkdir python-install && mkdir python-build && cd \$_ -DCMAKE_INSTALL_PREFIX=\${HOME}/scratch/python-install

Build make -j4

Install make install

Future work

Support 2.7.8 and 3.x

Document CMake buildsystem using sphinx.

Setup Travis CI

Setup dashboard for RaspberryPi

First class support for frozen module [7].

Integrate SetupTools with CMake

Contributing

Follow the Github flow [7]:

Create a branch

Open a Pull Request

→ Test

Discuss and review

Merge

Run tests

\$ ctest -D Experimental -j10 Test project /home/jchris/scratch/python-build Start 1: test_site [...]

391/392 Test #374: test_poll Passed 10.16 sec 392/392 Test #255: test_io Passed 38.07 sec 100% tests passed, 0 tests failed out of 392

Test

Total Test time (real) = 66.09 sec

Test results submitted to CDash [3]

Website similar to buildbot [4] with built-in support for cmake and ctest

Acknowledgments

Build system based on the original work of David Sansome, Alex Neundorf and David DeMarle.

RaspberryPi cross compilation based on work of Luis Ibañez [5][6].

Much of this work was supported by the National Institutes of Health

Thanks to David Thompson for his poster feedback, and thanks to Mysha Sissine for her support.

References

[0] http://www.python.org

- [1] http://www.cmake.org
- [2] https://github.com/davidsansome/python-cmake-buildsystem
- [3] http://open.cdash.org/index.php?project=CPython
- [4] http://buildbot.python.org/all/waterfall
- [5] http://www.kitware.com/blog/home/post/426
- [6] http://www.kitware.com/blog/home/post/428
- [7] http://conference.scipy.org/scipy2013/presentation_detail.php?id=129 [8] https://guides.github.com/introduction/flow/index.html

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 License.

Based on template from Felix Breuer - http://blog.felixbreuer.net/2010/10/24/poster.html