

START HERE

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

CMake build system for CPython

Simple with built-in support for cross-compilation.

Jean-Christophe Fillion-Robin, Matt McCormick



What is CMake ?

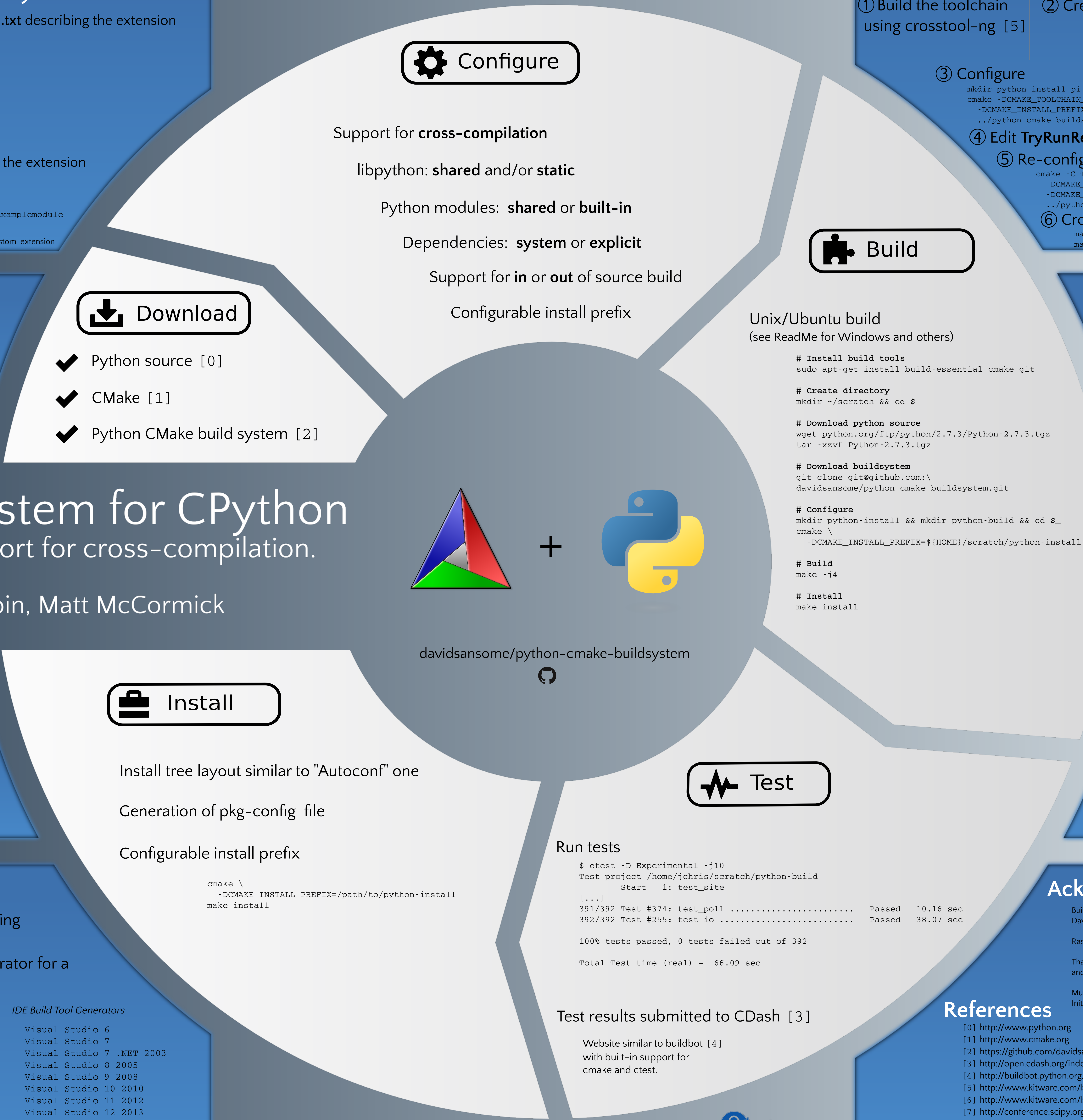
- One simple language for all platforms
- Generates native build system
- Cross-platform
- Open-source – BSD-like license
- Self-contained – No dependencies
- Large community

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	Command-Line Build Tool Generators	IDE Build Tool Generators
CodeBlocks	Borland Makefiles	Visual Studio 6
CodeLite	MSYS Makefiles	Visual Studio 7
Eclipse CDT4	MinGW Makefiles	Visual Studio 7 .NET 2003
KDevelop3	NMake Makefiles	Visual Studio 8 2005
Kate	NMake Makefiles JOM	Visual Studio 9 2008
Sublime Text 2	Ninja	Visual Studio 10 2010
	Unix Makefiles	Visual Studio 11 2012
	Watcom WMake	Visual Studio 12 2013
		Xcode



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/scratch/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/scratch/python-install-pi \
      ../python-cmake-buildsystem
```

⑥ Cross-compile

```
make -j4
make install
```

⑦ Upload to target

Ubuntu 13.10 / CMake 2.8.9

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 [8]:

- 🔗 Create a branch
- 🔗 Open a Pull Request
- 🔗 Test
- 🔗 Discuss and review
- 🔗 Merge

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].

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

Much of this work was supported by the National Institutes of Health, Roadmap Initiative for Medical Research under grant U54 EB005149.

References

- [0] <http://www.python.org>
- [1] <http://www.cmake.org>
- [2] <https://github.com/dauidsansome/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>