cget Documentation

Release 0.1.0

Paul Fultz II

November 16, 2016

Contents

Contents:

CHAPTER 1

Introduction

Cmake package retrieval. This can be used to download and install cmake packages. The advantages of using `cget` are:

- Non-intrusive: There is no need to write special hooks in cmake to use `cget`. One cmake file is written and can be used to install a package with `cget` or standalone.
- Works out of the box: Since it uses the standard build and install of cmake, it already works with almost all cmake packages. There is no need to wait for packages to convert over to support `cget`. Standard cmake packages can be already installed immediately.
- Decentralized: Packages can be installed from anywhere, from github, urls, or local files.

1.1 Installing cget

`cget` can be simply installed using `pip` (you can get pip from here):

```
pip install cget
```

Or installed directly with python:

```
python setup.py install
```

On windows, you may want to install pkgconfig-lite to support packages that use pkgconfig.

1.2 Quickstart

1.2.1 Installing a package

Any library that uses cmake to build can be built and installed as a package with `cget`. A source for package can be from many areas (see *Package source*). We can simply install `zlib` with its URL:

```
cget install http://zlib.net/zlib-1.2.8.tar.gz
```

We can install the package from github as well, using a shorten form. For example, installing John MacFarlane's implementation of CommonMark in C called cmark:

```
cget install jgm/cmark
```

1.2.2 Removing a package

A package can be removed by using the same source name that was used to install the package:

```
cget install http://zlib.net/zlib-1.2.8.tar.gz
cget remove http://zlib.net/zlib-1.2.8.tar.gz
```

If an alias was specified, then the name of the alias must be used instead:

```
cget install zlib,http://zlib.net/zlib-1.2.8.tar.gz
cget remove zlib
```

1.2.3 Testing packages

The test suite for a package can be ran before installing it, by using the `--test` flag. This will either build the `check` target or run `ctest`. So if we want to run the tests for zlib we can do this:

```
cget install --test http://zlib.net/zlib-1.2.8.tar.gz
```

1.2.4 Setting the prefix

By default, the packages are installed in the local directory `cget`. This can be changed by using the `--prefix` flag:

```
cget install --prefix /usr/local zlib:http://zlib.net/zlib-1.2.8.tar.gz
```

The prefix can also be set with the `CGET_PREFIX` environment variable.

1.2.5 Integration with cmake

By default, cget creates a cmake toolchain file with the settings necessary to build and find the libraries in the cget prefix. The toolchain file is at `$CGET_PREFIX/cget.cmake`. If another toolchain needs to be used, it can be specified with the `init` command:

```
cget init --toolchain my_cmake_toolchain.cmake
```

Also, the C++ version can be set for the toolchain as well:

```
cget init --std=c++14
```

Which is necessary to use modern C++ on many compilers.

Package source

2.1 Directory

This will install the package that is located at the directory:

```
cget install ~/mylibrary/
```

There must be a `CMakeLists.txt` in the directory.

2.2 File

An archived file of the package:

```
cget install zlib-1.2.8.tar.gz
```

The archive will be unpacked and installed.

2.3 URL

An url to the package:

```
cget install http://zlib.net/zlib-1.2.8.tar.gz
```

The file will be downloaded, unpacked, and installed.

2.4 Github

A package can be installed directly from github using just the namespace and repo name. For example, John MacFarlane's implementation of CommonMark in C called cmark can be installed like this:

```
cget install jgm/cmark
```

A tag or branch can specified using the `@` symbol:

```
cget install jgm/cmark@0.24.1
```

2.5 Aliasing

Aliasing lets you pick a different name for the package. So when we are installing `zlib`, we could alias it as `zlib`:

```
cget install zlib,http://zlib.net/zlib-1.2.8.tar.gz
```

This way the package can be referred to as `zlib` instead of `http://zlib.net/zlib-1.2.8.tar.gz`.

CHAPTER 3

Requirements file

`cget` will install all packages listed in the `requirements.txt` file. Each requirement is listed on a new line.

`<package-source>`

This specifies the package source (see *Package source*) that will be installed.

`-H, --hash`
This specifies a hash checksum that should checked before installing the packaging. The type of hash needs to be specified with a colon first, and then the hash. So for md5, it would be something like `md5:6fc67d80e915e63aacb39bc7f7da0f6c`.

`-b, --build`
This is a dependency that is only needed for building the package. It is not installed as a dependent of the package, as such, it can be removed after the package has been installed.

`-t, --test`
`cget` will only install the dependency if the tests are going to be run. This dependency is also treated as a build dependency so the it can be removed after the package has been installed.

`-D, --define` `VAR=VALUE`
Extra configuration variables to pass to CMake.

`-X, --cmake`
This specifies an alternative cmake file to be used to build the library. This is useful for packages that don't have a cmake file.

CHAPTER 4

Commands

4.1 build

This will build a package, but it doesn't install it. This is useful over using raw cmake as it will use the cmake toolchain that was initialized by cget which sets cmake up to easily find the dependencies that have been installed by cget.

`<package-source>`

This specifies the package source (see *Package source*) that will be built.

`-p, --prefix` `PATH`
Set prefix where packages are installed. This defaults to a directory named `cget` in the current working directory. This can also be overidden by the `CGET_PREFIX` environment variable.

`-v, --verbose`
Enable verbose mode.

`-B, --build-path` `PATH`
Set the path for the build directory to use when building the package.

`-t, --test`
Test package after building. This will set the `BUILD_TESTING` cmake variable to true. It will first try to run the `check` target. If that fails it will call `ctest` to try to run the tests.

`-c, --configure`
Configure cmake. This will run either `ccmake` or `cmake-gui` so the cmake variables can be set.

`-C, --clean`
Remove build directory.

`-P, --path`
Show path to build directory.

`-D, --define` `VAR=VALUE`
Extra configuration variables to pass to CMake

`-T, --target` `TARGET`
Cmake target to build.

`-y, --yes`
Affirm all questions.

`-G, --generator` `GENERATOR`
Set the generator for CMake to use.

4.2 clean

This will clear the directory used by cget. This will remove all packages that have been installed, and any toolchain settings.

`-p, --prefix` `PATH`
Set prefix where packages are installed. This defaults to a directory named `cget` in the current working directory. This can also be overidden by the `CGET_PREFIX` environment variable.

`-v, --verbose`
Enable verbose mode.

`-y, --yes`
Affirm all questions.

4.3 init

This will initialize the cmake toolchain. By default, the `install` command will initialize a cmake toolchain if one doesn't exists. This allows setting different variable, such as setting C++ compiler or standard version.

`-p, --prefix` `PATH`
Set prefix where packages are installed. This defaults to a directory named `cget` in the current working directory. This can also be overidden by the `CGET_PREFIX` environment variable.

`-v, --verbose`
Enable verbose mode.

`-B, --build-path` `PATH`
Set the path for the build directory to use when building the package.

`-t, --toolchain` `FILE`
Set cmake toolchain file to use.

`--cxx` `COMPILER`
Set c++ compiler.

`--cxxflags` `FLAGS`
Set additional c++ flags.

`--ldflags` `FLAGS`
Set additional linker flags.

`--std` `TEXT`
Set C++ standard if available.

`-D, --define` `VAR=VALUE`
Extra configuration variables to pass to CMake.

`--shared`
Set toolchain to build shared libraries by default.

`--static`
Set toolchain to build static libraries by default.

4.4 install

A package can be installed using the `install` command. When a package is installed, `cget` configures a build directory with cmake, and then builds the `all` target and the `install` target. So, essentially, `cget` will run the equivalent of these commands on the package to install it:

```
mkdir build
cd build
cmake ..
cmake --build .
cmake --build . --target install
```

However, `cget` will always create the build directory out of source. It will also setup cmake to point to the correct prefix and install directories.

<package-source>

This specifies the package source (see *Package source*) that will be installed.

-p, --prefix PATH
Set prefix where packages are installed. This defaults to a directory named `cget` in the current working directory. This can also be overidden by the `CGET_PREFIX` environment variable.

-v, --verbose
Enable verbose mode.

-B, --build-path PATH
Set the path for the build directory to use when building the package.

-U, --update
Update package. This will rebuild the package even its already installed and replace it with the newly built package.

-t, --test
Test package before installing. This will set the `BUILD_TESTING` cmake variable to true. It will first try to run the `check` target. If that fails it will call `ctest` to try to run the tests.

--test-all
Test all packages including its dependencies before installing by running ctest or check target.

-f, --file FILE
Install packages listed in the file.

-D, --define VAR=VALUE
Extra configuration variables to pass to CMake.

-G, --generator GENERATOR
Set the generator for CMake to use.

-X, --cmake
This specifies an alternative cmake file to be used to build the library. This is useful for packages that don't have a cmake file.

4.5 list

This will list all packages that have been installed.

-p, --prefix `PATH`
Set prefix where packages are installed. This defaults to a directory named `cget` in the current working directory. This can also be overidden by the `CGET_PREFIX` environment variable.

-v, --verbose
Enable verbose mode.

4.6 pkg-config

This will run pkg-config, but will search in the cget directory for pkg-config files. This useful for finding dependencies when not using cmake.

-p, --prefix `PATH`
Set prefix where packages are installed. This defaults to a directory named `cget` in the current working directory. This can also be overidden by the `CGET_PREFIX` environment variable.

-v, --verbose
Enable verbose mode.

4.7 remove

This will remove a package. If other packages depends on the package to be removed, those packages will be removed as well.

<package-name>
This is the name of the package to be removed.

-p, --prefix `PATH`
Set prefix where packages are installed. This defaults to a directory named `cget` in the current working directory. This can also be overidden by the `CGET_PREFIX` environment variable.

-v, --verbose
Enable verbose mode.

-y, --yes
Affirm all questions.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Index