

iZENELib Build Reference

Ian Yang

June 17, 2009

*iZENEs*oft(Shanghai) Co., Ltd

Abstract

This document represents how to build izenelib, how to use izenelib in other projects and how to maintain izenelib build scripts.

Contents

1	How to Build	3
1.1	Quick Start	3
1.2	Detailed Description	4
1.2.1	Shared CMake Modules	4
1.2.2	Install Prefix	4
1.2.3	Dependencies	5
2	How to Maintain	5
2.1	Project Version Control	5
2.2	Dependencies Resolve	5
2.3	How to Add New Library	6
3	How to Use izenelib	6
3.1	Install Tree Structure	6
3.2	Code Changes	7
3.3	Search Header and Library	7

1 How to Build

1.1 Quick Start

Assume that `izenelib` has been cloned to `/repo/izenelib`, and account name `first.last` is used to access repositories on `izenesoft.com`.

1. Create directory `myroot` in the home directory¹.
2. Clone `cmake.git` in `$HOME/myroot`.

```
cd ${HOME}/myroot
git clone first.last@izenesoft.com:/git/cmake.git
```

3. Configure. Some wanings may display if some dependencies cannot been found. You may install them or set some variables to help the build scripts find your installation.

```
cd /repo/izenelib/build
cmake ..
```

4. Compilation.

```
make
```

¹In Windows, create `myroot` in the directory specified by environment variable `'HOME'`.

5. Test (Optional).

```
make test
```

6. Install. Libraries and headers are installed in `${HOME}/myroot` by default.

```
make install
```

1.2 Detailed Description

1.2.1 Shared CMake Modules

CMake scripts can be encapsulated in modules and reused later. A repository `izenesoftware.com:/git/cmake.git` is created to share CMake modules among projects in iZENEssoft.

For example, the module `FindTokyoCabinet` has been submitted. It can be used to find library Tokyo Cabinet. You just need to append path of your cloned `cmake.git` to `CMAKE_MODULE_PATH` in the main `CMakeLists.txt`. Then use `FIND_PACKAGE`:

```
FIND_PACKAGE(TokyoCabinet 1.4.24)
IF(TokyoCabinet_FOUND)
    INCLUDE_DIRECTORIES(${TokyoCabinet_INCLUDE_DIRS})
    ADD_EXECUTABLE(myexe main.cpp)
    TARGET_LINK_LIBRARIES(myexe ${TokyoCabinet_LIBRARIES})
ELSE(TokyoCabinet_FOUND)
    MESSAGE(STATUS "Cannot build myexe without Tokyo Cabinet >= 1.4.24")
ENDIF(TokyoCabinet_FOUND)
```

The cloned `cmake.git` repository should be placed in `CMAKE_INSTALL_PREFIX`² in `izenelib`, and the name must be “`cmake`”. Otherwise you need to set the CMake cache variable `EXTRA_CMAKE_MODULES_DIRS`³, or environment variable with the same name.

1.2.2 Install Prefix

CMake command `INSTALL` copies files to directory specified by cache variable `CMAKE_INSTALL_PREFIX`. The default value is `/usr/local`, but it is changed to `${HOME}/myroot` in `izenelib`. The default value `${HOME}/myroot` can be changed by setting the environment variable `MY_INSTALL_PREFIX`. The cache variable `CMAKE_INSTALL_PREFIX` can be used to specify the prefix directly.

²It is `${HOME}/myroot` in `izenelib` by default.

³It is intended to be used on the command line with a `-DVAR=value`

1.2.3 Dependencies

Project `izenelib` depends on some external libraries.

- Boost

Boost is searched using CMake bundled module `FindBoost`. If your boost cannot be found, or the found version is not compatible, just set environment variable or cmake cache variable `BOOST_ROOT`. It should be set to the installation prefix of your specified boost.

- BZip2 (`libbz2`)

Manually set cache variable `BZIP2_INCLUDE_DIR` to header directory and `BZIP2_LIBRARIES` to the path of the library, if your installed BZip2 cannot be found.

- ZLIB (`libz`)

Manually set cache variable `ZLIB_INCLUDE_DIR` to header directory and `ZLIB_LIBRARIES` to the path of the library, if your installed ZLIB cannot be found.

- TokyoCabinet (`libtokyocabinet`)

It is not required to build `izenelib`, but it is recommended to install it because a header only library in `izenelib` is based on Tokyo Cabinet with version at least 1.4.24. You can set cache variable `TokyoCabinet_ROOT_DIR` to the installation prefix of TokyoCabinet if your installation cannot be found.

2 How to Maintain

2.1 Project Version Control

It is required to update the project version in the main `CMakeLists.txt` after updating the source codes.

- `PROJECT_VERSION_PATCH`

Minor code changes affecting interface.

- `PROJECT_VERSION_MINOR`

Significant interface changes, such as a new feature is added.

- `PROJECT_VERSION_MAJOR`

Essential changes such as a rully refactoring.

2.2 Dependencies Resolve

All dependencies must be configured smart and automatically. Libraries should be searched using `FIND_PACKAGE`. Non-portable header must be checked using module `CHECK_INCLUDE_FILE`. We should make less assumption about the build environment.

2.3 How to Add New Library

Assume that you want to add a library “hello”.

1. Place header files in directory `include`, 3rd party directory should be placed in `include/3rdparty`. The directory hierarchy is important, the header file `include/xxx/yyy.h` will be used as `<xxx/yyy.h>` and `include/3rdparty/iii/jjj.h` will be used as `<iii/jjj.h>`.
2. Add sub-directory in directory `source`, and create a `CMakeLists.txt` for the source code.
3. If your library has dependencies, add corresponding `FIND_PACKAGE` or other commands in `ConfigureChecks.cmake`.
4. Check whether all dependencies have been found.
5. If the library can be compiled, append the name “hello” to global property `BUILD_COMPONENTS`. If the library cannot because of missing dependencies, append “hello” to global property `NON_BUILD_COMPONENTS`.
6. Add two library targets “hello_static” and “hello_shared” for static library and dynamic library respectively. Set `VERSION` property of “hello_shared”.
7. Install “hello_static” and “hello_shared”. Set “ARCHIVE DESTINATION” to `${VERSION_INSTALL_PREFIX}/lib` and “LIBRARY DESTINATION” to `shared`. Also export two targets as name “hello”.
8. Install the export “hello” in `${VERSION_INSTALL_PREFIX}/cmake`

3 How to Use izenelib

3.1 Install Tree Structure

It is assumed that izenelib has been installed in `/path/to/myroot`.

- Header files
`/path/to/myroot/izenelib-x.y.z/include`
- Static libraries
`/path/to/myroot/izenelib-x.y.z/lib`
- Package config (used in `FIND_PACKAGE`)
`/path/to/myroot/izenelib-x.y.z/cmake`
- Shared libraries
`/path/to/myroot/shared`

3.2 Code Changes

You may need to update the header file name in your project. For example boost memory header is <boost/memory.hpp>, udt3 header is <udt3/udt.h>.

3.3 Search Header and Library

1. Append izenelib install prefix⁴ to CMAKE_PREFIX_PATH. It can be cmake cache variable or environment variable.
2. Use FIND_PACKAGE to find. see some examples below.

```
# example 1, use header only library
FIND_PACKAGE(izenelib REQUIRED)
INCLUDE_DIRECTORIES(${izenelib_INCLUDE_DIRS})

# example 2, use library febird and sdb_btree >= 1.0.1
FIND_PACKAGE(izenelib REQUIRED COMPONENTS febird sdb_btree)
INCLUDE_DIRECTORIES(${izenelib_INCLUDE_DIRS})
ADD_EXECUTABLE(test test.cpp)
TARGET_LINK_LIBRARIES(test ${izenelib_SHARED_LIBRARIES})
# same with
# TARGET_LINK_LIBRARIES(test febird_shared)
# TARGET_LINK_LIBRARIES(test sdb_btree)

# example 3, not required
FIND_PACKAGE(izenelib COMPONENTS udt)
IF(izenelib_FOUND AND izenelib_udt_FOUND)
# include header directory and link library here
ELSE(izenelib_FOUND AND izenelib_udt_FOUND)
    MESSAGE(STATUS "Cannot build xxx without izenelib udt, skipped")
ENDIF(izenelib_FOUND AND izenelib_udt_FOUND)
```

File “izenelib-config.cmake.in” has detailed descriptions about the usages. And “source/build_demo” is a demo.

⁴directory containing shared, izenelib-x.y.z sub-directories.