CMake: 工程构建工具

C++ 敏毅 2014-08-18 351 阅读

CMake: 工程构建工具

目录

1 前言

在 `Unix-Like` 操作系统上源码编译安装软件的过程:

```
$ ./configure
$ make
$ make install
```

这是基于 `autotools` 、GNUMake等工具链的，而 `CMake` 跨平台编译能够更好的管理编译过程，需要使用一个 配置文件 .

`CMake` 最初是由Kitware开发的，后来在KDE4开发中大量使用，证明在大型项目中的可用性：

> In addition to a build system, over the years `CMake` has evolved into a family of development tools: `CMake`, `CTest`, `CPack`, and `CDash`:
>
> - `CMake` is the build tool responsible for building software.
> - `CTest` is a test driver tool, used to run regression tests.
> - `CPack` is a packaging tool used to create platform-specific installers for software built with CMake.
> - `CDash` is a web application for displaying testing results and performing continuous integration testing.

2 Hello World

代码(test/t1/main.c):

```
#include <stdio.h>
int main()
{
    printf("Hello World from t1 Main!\n");
    return 0;
}
```

CMakeLists.txt:

```
PROJECT (HELLO)
SET(SRC_LIST main.c)
MESSAGE(STATUS "This is BINARY dir " ${HELLO_BINARY_DIR})
MESSAGE(STATUS "This is SOURCE dir "${HELLO_SOURCE_DIR})
ADD_EXECUTABLE(hello ${SRC_LIST})
```

构建工程:

```
$ cmake .

-- This is BINARY dir /path/to/t1
-- This is SOURCE dir /path/to/t1
-- Configuring done
-- Generating done
-- Build files have been written to: /path/to/t1

$ ls -1

CMakeCache.txt
CMakeFiles
CMakeLists.txt
Makefile
cmake_install.cmake
main.c
```

编译:

```
$ make

Scanning dependencies of target hello
[100%] Building C object CMakeFiles/hello.dir/main.c.o
Linking C executable hello
[100%] Built target hello

$ ./hello
Hello World from t1 Main!
```

清理工程:

```
$ make clean
```

3 CMakeLists.txt简单说明

3.1 PROJECT指令

PROJECT指令用来指定工程项目名称和使用的开发语言:

```
PROJECT(projectname [CXX] [C] [Java])
```

这条指令隐式的定义了两个CMake变量:

```
<projectname>_BINARY_DIR
<projectname>_SOURCE_DIR
```

同时，也定义了两个变量分别指向上面的变量:

```
PROJECT_BINARY_DIR
PROJECT_SOURCE_DIR
```

3.2 SET指令

SET指令用于定义变量:

```
SET(VAR [VALUE] [CACHE TYPE DOCSTRING [FORCE]])
```

3.3 MESSAGE指令

MESSAGE指令用于显示信息:

```
MESSAGE([SEND_ERROR | STATUS | FATAL_ERROR] "message to display" ...)
```

其中:

- SEND_ERROR: 发送错误，跳过过程
- STATUS: 打印信息
- FATAL_ERROR: 终止构建过程

3.4 ADD_EXECUTABLE指令

ADD_EXECUTABLE指令用于设定生成的可执行文件:

```
ADD_EXECUTABLE(hello main.c;func.c)
```

3.5 ADD_SUBDIRECTORY 指令

ADD_SUBDIRECTORY 指令可以指定源码目录，而不是一个个的源码文件:

```
ADD_SUBDIRECTORY(source_dir [binary_dir] [EXCLUDE_FROM_ALL])
```

但在具体的每个source_dir里面要添加CMakeLists.txt,如:

```
ADD_EXECUTABLE(hello main.c)
```

3.6 EXECUTABLE_OUTPUT_PATH和LIBRARY_OUTPUT_PATH变量

通过变量指定最终二进制文件的存放目录:

```
SET(EXECUTABLE_OUTPUT_PATH ${PROJECT_BINARY_DIR}/bin)
SET(LIBRARY_OUTPUT_PATH ${PROJECT_BINARY_DIR}/lib)
```

这两个变量会屏蔽ADD_SUBDIRECTORY指令中指定的bianry_dir

3.7 CMAKE_INSTALL_PREFIX变量

类似 `./configure --prefix=xxx` ,这个变量是关于安装的:

```
$ cmake -DCMAKE_INSTALL_PREFIX=/usr .
```

此变量默认是 `/usr/local`

3.8 INSTALL 指令

如果安装说明文件:

```
INSTALL(FILES COPYRIGHT README DESTINATION share/doc/cmake/t2)
```

安装程序:

```
INSTALL(PROGRAMS runhello.sh DESTINATION bin)
```

安装目录中的内容:

```
INSTALL(DIRECTORY doc/ DESTINATION share/doc/cmake/t2)
```

3.9 ADD_LIBRARY指令

建立共享库:

```
ADD_LIBRARY(libname [SHARED|STATIC|MODULE]
                    [EXCLUDE_FROM_ALL]
            source1 source2 ... sourceN)


SET(LIBHELLO_SRC hello.c)
ADD_LIBRARY(hello SHARED ${LIBHELLO_SRC})
```

3.10 SET_TARGET_PROPERTIES指令

SET_TARGET_PROPERTIES用来指定输出的名称:

```
SET_TARGET_PROPERTIES(target1 target2 ...
                      PROPERTIES prop1 value1
                                 prop2 value2 ...)

SET_TARGET_PROPERTIES(hello_static PROPERTIES OUTPUT_NAME "hello")
```

4 内部构建与外部构建

CMake推荐的是 外部构建 ，在HelloWorld里面的是 内部构建 方式。

对于 内部构建 ，会生成一些无法自动删除的中间文件；对于 外部构建 ，只不过是另外建立目录，将中间文件保存在 编译目录 而已。

还是以HelloWorld为例，进行 外部构建 :

```
$ ls -1     # 只保留这两个文件

CMakeLists.txt
main.c

$ mkdir build
$ cd build   # 这个就是编译目录, PROJECT_BINARY_DIR
$ cmake ../  # 这个就是源码目录, PROJECT_SOURCE_DIR
-- The C compiler identification is Clang 5.1.0
-- The CXX compiler identification is Clang 5.1.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- This is BINARY dir /path/to/t1/build
-- This is SOURCE dir /path/to/t1
-- Configuring done
-- Generating done
-- Build files have been written to: /path/to/t1/build

$ ls -1 .
CMakeCache.txt
CMakeFiles
Makefile
cmake_install.cmake
```