chapter1.5 cmake设置构建类型

1. 文件构成：

```unix
.
├── CMakeLists.txt
├── build
└── main.cpp
```

2. 文件填充：

2.1 main.cpp

```cpp
#include <iostream>

int main(int argc, char *argv[])
{
   std::cout << "Hello Build Type!" << std::endl;
   return 0;
}
```

2.2 CMakeLists.txt

```
cmake_minimum_required(VERSION 3.5)
#如果没有指定则设置默认编译方式
if(NOT CMAKE_BUILD_TYPE AND NOT CMAKE_CONFIGURATION_TYPES)
  #在命令行中输出message里的信息
  message("Setting build type to 'RelWithDebInfo' as none was specified.")
  #不管CACHE里有没有设置过CMAKE_BUILD_TYPE这个变量，都强制赋值这个值为RelWithDebInfo
  set(CMAKE_BUILD_TYPE RelWithDebInfo CACHE STRING "Choose the type of build." FORCE)

  # 当使用cmake-gui的时候，设置构建级别的四个可选项
  set_property(CACHE CMAKE_BUILD_TYPE PROPERTY STRINGS "Debug" "Release"
    "MinSizeRel" "RelWithDebInfo")
endif()

project (build_type)
add_executable(cmake_examples_build_type main.cpp)

#命令的具体解释在二  CMake解析中，这里的注释只说明注释后每一句的作用
```

3. 文件解析：

3.1 构建级别

CMake具有许多内置的构建配置，可用于编译工程。 这些配置指定了代码优化的级别，以及调试信息是否包含在二进制文件中。

这些优化级别，主要有：

- Release —— 不可以打断点调试，程序开发完成后发行使用的版本，占的体积小。 它对代码做了优化，因此速度会非常快。在编译器中使用命令： `-O3 -DNDEBUG` 可选择此版本。
- Debug ——调试的版本，体积大。在编译器中使用命令： `-g` 可选择此版本。
- MinSizeRel—— 最小体积版本。在编译器中使用命令： `-Os -DNDEBUG` 可选择此版本。
- RelWithDebInfo—— 既优化又能调试。在编译器中使用命令： `-O2 -g -DNDEBUG` 可选择此版本。

3.2 设置级别的方式

在命令行运行CMake的时候， 使用cmake命令行的-D选项配置编译类型：

```
cmake .. -DCMAKE_BUILD_TYPE=Release
```

3.3 CMake中设置默认的构建级别

CMake提供的默认构建类型是不进行优化的构建级别。 对于某些项目，需要自己设置默认的构建类型，以便不必记住进行设置。

3.4 set()命令

该命令可以为普通变量、缓存变量、环境变量赋值：
( )处 可以设置零个或多个参数。多个参数将以分号分隔的列表形式加入，以形成要设置的实际变量值。零参数将导致未设置普通变量。见 `unset()` 命令显式取消设置变量。

所以此处学习SET命令需要分为设置普通变量，缓存变量以及环境变量三种类别来学习。

3.4.1 正常变量

```
set(<variable> <value>... [PARENT_SCOPE])
```

设置的变量值 作用域属于整个 CMakeLists.txt 文件。(一个工程可能有多个CMakeLists.txt)
当这个语句中加入PARENT_SCOPE后，表示要设置的变量是父目录中的CMakeLists.txt设置的变量。

比如有如下目录树：

```
├── CMakeLists.txt
└── src
    └── CMakeLists.txt
```

并且在 顶层的CMakeLists.txt中包含了src目录： `add_subdirectory(src)`
那么，顶层的CMakeLists.txt就是父目录，
如果父目录中有变量 `Bang` ,在子目录中可以直接使用（比如用message输出 `Bang` ，值是父目录中设置的值）并且利用set()修改该变量 `Bang` 的值，但是如果希望在离开 该子CMakeLists.txt后 对该变量做出的修改能够得到保留，那么就需要在set()命令中加入Parent scope这个变量。当然，如果父目录中本身没有这个变量，子目录中仍然使用了parent scope，那么出了这个作用域后，该变量仍然不会存在。

这里举一个实际的例子:

```
test:
    build
    sub:
        build
        CmakeLists.txt
    CmakeLists.txt
```

我们建立一个项目结构如上:

```
# test/sub/CMakeLists.txt
cmake_minimum_required (VERSION 3.5)
project (subtest)

set (val sub_hello)
set (val par_hello PARENT_SCOPE)
```

```
message (">>>>>> in sub level, value = ${val}")
```

```
# test/CMakeLists.txt
cmake_minimum_required (VERSION 3.5)
project (partest)

add_subdirectory (sub)

message (">>> in parent , value = ${val}")
```

执行如下:

```
#在项目test/build下执行cmake ..
>>>>>> in sub level, value = sub_hello
>>> in parent , value = par_hello
```

```
#在项目test/sub/build下执行cmake ..
>>>>>> in sub level, value = sub_hello
```

从这里来看我们发现在执行父级CmakeLists.txt的内容时，会输出子目录的内容，而在执行子目录的CmakeLists.txt时则只会输出自己的内容。

3.4.2 CACHE变量

完整语句如下：

```
set(<variable> <value>... CACHE <type> <docstring> [FORCE])
```

- 首先什么是CACHE变量，就是在运行cmake的时候，变量的值可能会被缓存到一份文件里即build命令下的CMakeCache.txt，当你重新运行cmake的时候，那些变量会默认使用这个缓存里的值。这个变量是全局变量，整个CMake工程都可以使用该变量。
- 在这个文件里，只要运行cmake ..命令，自动会出现一些值，比如 CMAKE_INSTALL_PREFIX ，如果设置 set(CMAKE_INSTALL_PREFIX "/usr") ，虽然CACHE缓存文件里还有这个CMAKE_INSTALL_PREFIX 变量，但是因为我们显示得设置了一个名为CMAKE_INSTALL_PREFIX 的正常变量，所以之后使用CMAKE_INSTALL_PREFIX ，值是我们设置的正常变量的值。
- 如果加上CACHE关键字，则设置的这个变量会被写入缓存文件中（但如果本身缓存文件中有这个变量，则不会覆盖缓存中的变量）。
- 只有加上**FORCE**关键字，这个被写入文件的值会覆盖之前文件中存在的同名变量。

3.4.3 环境变量

```
set(ENV{<variable>} [<value>])
```

设置一个 Environment Variable 到给定值。随后的调用 $ENV{<varible>} 将返回此新值。
此命令仅影响当前的CMake进程，不影响调用CMake的进程，也不影响整个系统环境，也不影响后续构建或测试过程的环境。
如果在空字符串之后 ENV{} 或如果没有参数，则此命令将清除环境变量的任何现有值。
之后的参数将被忽略。如果发现其他参数，则会发出作者警告。

4. 总览：

```
 huluobo@huluobodeMacBook-Pro ▶ ~/cmake-examples/myCmake/chapter1.5 ▶ ᛘ main ± ▶ cd build
 huluobo@huluobodeMacBook-Pro ▶ ~/cmake-examples/myCmake/chapter1.5/build ▶ ᛘ main ± ▶ cmake ..
Setting build type to 'RelWithDebInfo' as none was specified.
-- The C compiler identification is AppleClang 15.0.0.15000040
-- The CXX compiler identification is AppleClang 15.0.0.15000040
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
```

```
-- Check for working C compiler: /Library/Developer/CommandLineTools/usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /Library/Developer/CommandLineTools/usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done (0.4s)
-- Generating done (0.0s)
-- Build files have been written to: /Users/huluobo/cmake-examples/myCmake/chapter1.5/build
 huluobo@huluobodeMacBook-Pro ▶ ~/cmake-examples/myCmake/chapter1.5/build ▶ ᛘ main ± ▶ make VERBOSE=1
/Applications/CMake.app/Contents/bin/cmake -S/Users/huluobo/cmake-examples/myCmake/chapter1.5 -
B/Users/huluobo/cmake-examples/myCmake/chapter1.5/build --check-build-system CMakeFiles/Makefile.cmake 0
/Applications/CMake.app/Contents/bin/cmake -E cmake_progress_start /Users/huluobo/cmake-
examples/myCmake/chapter1.5/build/CMakeFiles /Users/huluobo/cmake-
examples/myCmake/chapter1.5/build//CMakeFiles/progress.marks
/Library/Developer/CommandLineTools/usr/bin/make  -f CMakeFiles/Makefile2 all
/Library/Developer/CommandLineTools/usr/bin/make  -f CMakeFiles/cmake_examples_build_type.dir/build.make
CMakeFiles/cmake_examples_build_type.dir/depend
cd /Users/huluobo/cmake-examples/myCmake/chapter1.5/build && /Applications/CMake.app/Contents/bin/cmake -E
cmake_depends "Unix Makefiles" /Users/huluobo/cmake-examples/myCmake/chapter1.5 /Users/huluobo/cmake-
examples/myCmake/chapter1.5 /Users/huluobo/cmake-examples/myCmake/chapter1.5/build /Users/huluobo/cmake-
examples/myCmake/chapter1.5/build /Users/huluobo/cmake-
examples/myCmake/chapter1.5/build/CMakeFiles/cmake_examples_build_type.dir/DependInfo.cmake "--color="
/Library/Developer/CommandLineTools/usr/bin/make  -f CMakeFiles/cmake_examples_build_type.dir/build.make
CMakeFiles/cmake_examples_build_type.dir/build
[ 50%] Building CXX object CMakeFiles/cmake_examples_build_type.dir/main.cpp.o
/Library/Developer/CommandLineTools/usr/bin/c++   -O2 -g -DNDEBUG -arch arm64 -isysroot
/Library/Developer/CommandLineTools/SDKs/MacOSX14.0.sdk -MD -MT
CMakeFiles/cmake_examples_build_type.dir/main.cpp.o -MF
CMakeFiles/cmake_examples_build_type.dir/main.cpp.o.d -o
CMakeFiles/cmake_examples_build_type.dir/main.cpp.o -c /Users/huluobo/cmake-
examples/myCmake/chapter1.5/main.cpp
[100%] Linking CXX executable cmake_examples_build_type
/Applications/CMake.app/Contents/bin/cmake -E cmake_link_script
CMakeFiles/cmake_examples_build_type.dir/link.txt --verbose=1
/Library/Developer/CommandLineTools/usr/bin/c++ -O2 -g -DNDEBUG -arch arm64 -isysroot
/Library/Developer/CommandLineTools/SDKs/MacOSX14.0.sdk -Wl,-search_paths_first -Wl,-
headerpad_max_install_names CMakeFiles/cmake_examples_build_type.dir/main.cpp.o -o
cmake_examples_build_type
[100%] Built target cmake_examples_build_type
/Applications/CMake.app/Contents/bin/cmake -E cmake_progress_start /Users/huluobo/cmake-
examples/myCmake/chapter1.5/build/CMakeFiles 0
 huluobo@huluobodeMacBook-Pro ▶ ~/cmake-examples/myCmake/chapter1.5/build ▶ ᛘ main ± ▶
```