chapter1.8 cmake_Compiling_with_clang

1. 介绍：

使用CMake进行构建时，可以设置C和C ++编译器。 此示例与hello-cmake示例相同，不同之处在于它显示了将编译器从默认的gcc更改为clang的最基本方法

2. 文件结构：

```
$ tree
.
├── CMakeLists.txt
├── main.cpp
├── pre_test.sh
├── run_test.sh
|
|-- build.clang
```

- CMakeLists.txt - 包含了要运行的CMake命令
- main.cpp - 一个简单的 "Hello World" cpp 文件.
- pre_test.sh - 删除之前配置的build文件。
- run_test.sh - 找到具体clang编译器路径，并配置cmake使用clang编译器

3. 文件填充：

CMakeLists.txt

```
# 寻找CMake的最低版本，必须大于3.5
cmake_minimum_required(VERSION 3.5)

# 设置工程名称
project (hello_cmake)

# 生成可执行文件
add_executable(hello_cmake main.cpp)
```

main.cpp

```
#include <iostream>

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

pre_test.sh

```
#!/bin/bash
#pre_test脚本删除之前配置的build文件，run_test运行clang，生成这次的build.clang文件
#这个脚本的作用是如果存在build.clang这个文件夹，就把它删除掉
```

```
#ROOT_DIR=`pwd`#shell脚本的语法，pwd输出文件当前所在路径,赋值给ROOT_DIR这个变量
dir="01-basic/I-compiling-with-clang"
if [ -d "$ROOT_DIR/$dir/build.clang" ]; then
    echo "deleting $dir/build.clang"
    rm -r $dir/build.clang
fi


#if then fi是shell脚本里的判断语句，如果[]里的条件为真，则执行then后面的语句
#基本格式：
#       if [判断语句]; then
#           执行语句
#       fi
#-d与路径配合，路径存在则为真
#单纯的dir等价于ls -C -b; 也就是说，默认情况下，文件在列中列出，并垂直排序，特殊字符由反斜杠转义序列表示。
#也就是说只要当前历经下存在build.clang就删除掉
#本文dir是一个变量
```

run_test.sh

```
#!/bin/bash

#Ubuntu支持同时安装多个版本的clang。
#测试需要在调用cmake之前确定clang二进制文件
#这个脚本找到具体clang编译器路径，并配置cmake使用clang编译器

if [ -z $clang_bin ]; then
    clang_ver=`dpkg --get-selections | grep clang | grep -v -m1 libclang | cut -f1 | cut -d '-' -f2`
    clang_bin="clang-$clang_ver"#把版本号存到变量，把版本号添加到clangC编译器和clang编译器
    clang_xx_bin="clang++-$clang_ver"
fi

echo "Will use clang [$clang_bin] and clang++ [$clang_xx_bin]"
#echo用来输出信息

mkdir -p build.clang && cd build.clang && \
    cmake .. -DCMAKE_C_COMPILER=$clang_bin -DCMAKE_CXX_COMPILER=$clang_xx_bin && make

#相当于在shell中执行命令：whilch clang然后将返回的结果也就是路径，赋值给变量clang_bin
clang_bin=`which clang`
clang_xx_bin=`which clang++`
#which语句返回后面命令的路径
#  -z  指如果后面的路径为空则为真

#如果用which没有找到clang的二进制可执行文件，则用dpkg找到clang，并返回版本号
#dpkg –get-selections 罗列出所有包的名字并且给出了他们现在的状态比如已安装（ installed）已经卸载。（ deinstalled）
#grep clang从结果中查找到带有clang名字的
#grep -v   反转，选择不匹配的所有行。
#grep -m1  单纯的-m1表示输出1条匹配的结果之后就会停止
#grep -v -m1 libclang 输出包含clang的命令中，所有不包含libclang的一条介绍
#也就是去掉那些clang的库，找的是clang这个程序的版本。

#cut 命令从文件的每一行剪切字节、字符和字段并将这些字节、字符和字段写至标准输出。
#cut -f1  将这行按照空格？分隔之后选择第1个字段，就是clang-3.6
#cut -d '-' -f2  按照-分隔，选择第2个字段就是3.6   从而得到版本号
# ```shell
# $ dpkg --get-selections | grep clang | grep -v -m1 libclang
# clang-3.6                                       install
# $ dpkg --get-selections | grep clang | grep -v -m1 libclang | cut -f1
# clang-3.6
# $ dpkg --get-selections | grep clang | grep -v -m1 libclang | cut -f6
# install
# $ dpkg --get-selections | grep clang | grep -v -m1 libclang | cut -f1 | cut -d '-' -f2
```

```
# 3.6
# $ dpkg --get-selections | grep clang | grep -v -m1 libclang | cut -f1 | cut -d '-' -f1
# clang
# ```
#把每一步命令都运行一遍就知道用途了。
```

4. 文件解析：

（1）编译器选项

CMake提供了控制程序编译以及链接的选项，选项如下:

- CMAKE_C_COMPILER - 用于编译c代码的程序。
- CMAKE_CXX_COMPILER - 用于编译c++代码的程序。
- CMAKE_LINKER - 用于链接二进制文件的程序。

[注意1]
本例子需要事先使用命令 `sudo apt-get install clang-3.6 -y` 安装clang-3.6

[注意2]
这是调用clang的最基本，最简单的方法。 未来的示例将展示调用编译器的更好方法

（2）设置标志

就像 Build Type 描述的那样, 可以使用cmake gui或从命令行设置CMake的参数。

下面用了命令行：

```
cmake .. -DCMAKE_C_COMPILER=clang-3.6 -DCMAKE_CXX_COMPILER=clang++-3.6
```

注意，本文件中有两个脚本，可以自动执行脚本，通过脚本，指定编译器为clang并且编译工程。 上面通过设置关键字，表示运行make命令时，clang将用于编译二进制文件。 从make输出的以下几行可以看出这一点。

```
/usr/bin/clang++-3.6     -o CMakeFiles/hello_cmake.dir/main.cpp.o -c /home/matrim/workspace/cmake-
examples/01-basic/I-compiling-with-clang/main.cpp
Linking CXX executable hello_cmake
/usr/bin/cmake -E cmake_link_script CMakeFiles/hello_cmake.dir/link.txt --verbose=1
/usr/bin/clang++-3.6       CMakeFiles/hello_cmake.dir/main.cpp.o  -o hello_cmake -rdynamic
```

5. 总览：