cmake(九)Cmake设置生成库的属性

原创 wzj_110 于 2021-04-19 00:13:34 发布 阅读量985 收藏 2 点赞数 1 版权

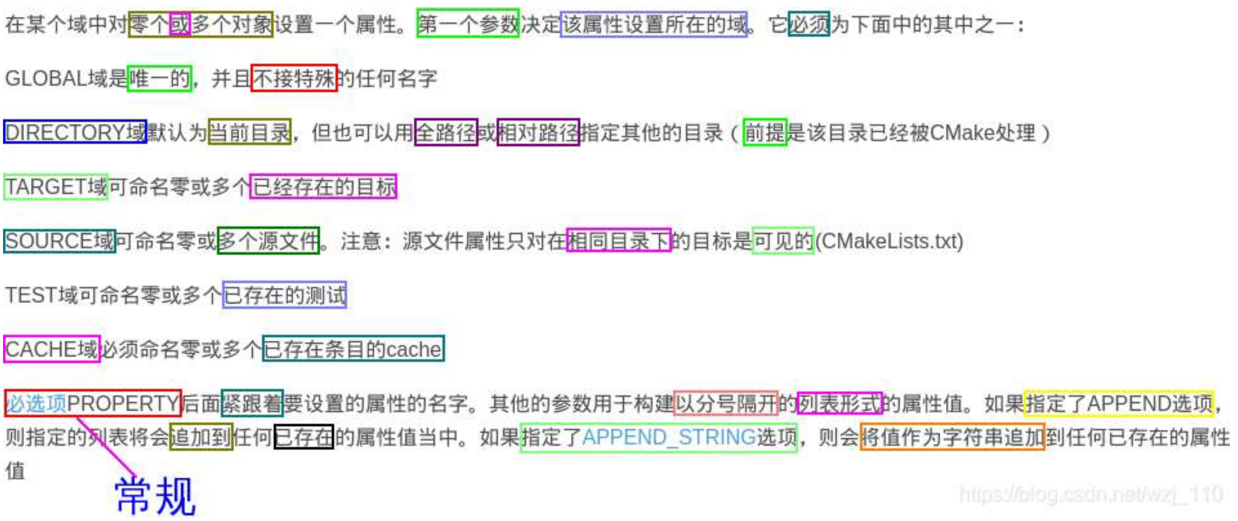
分类专栏： cmake DSL语言

cmake DSL语言 专栏收录该内容 38 篇文章 已订阅

一 基本语法

① 设置目标的属性

```
set_property(<GLOBAL                            |
             DIRECTORY [dir]                    |
             TARGET    [target1 [target2 ...]]  |
             SOURCE    [src1 [src2 ...]]        |
             TEST      [test1 [test2 ...]]      |
             CACHE     [entry1 [entry2 ...]]>
             [APPEND][APPEND_STRING]
             PROPERTY <name>[value1 [value2 ...]])
```



在某个域中对零个或多个对象设置一个属性。第一个参数决定该属性设置所在的域。它必须为下面中的其中之一：

GLOBAL域是唯一的，并且不接特殊的任何名字

DIRECTORY域默认为当前目录，但也可以用全路径或相对路径指定其他的目录（前提是该目录已经被CMake处理）

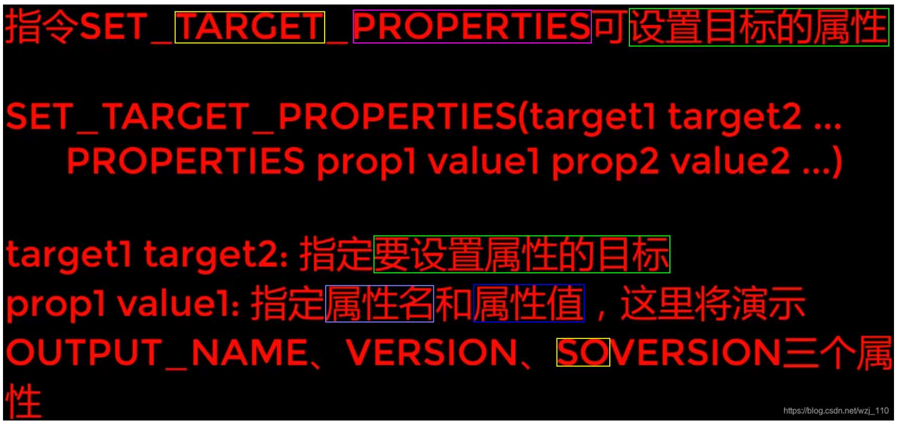
TARGET域可命名零或多个已经存在的目标

SOURCE域可命名零或多个源文件。注意：源文件属性只对在相同目录下的目标是可见的(CMakeLists.txt)

TEST域可命名零或多个已存在的测试

CACHE域必须命名零或多个已存在条目的cache

必选项PROPERTY后面紧跟着要设置的属性的名字。其他的参数用于构建以分号隔开的列表形式的属性值。如果指定了APPEND选项，则指定的列表将会追加到任何已存在的属性值当中。如果指定了APPEND_STRING选项，则会将值作为字符串追加到任何已存在的属性值



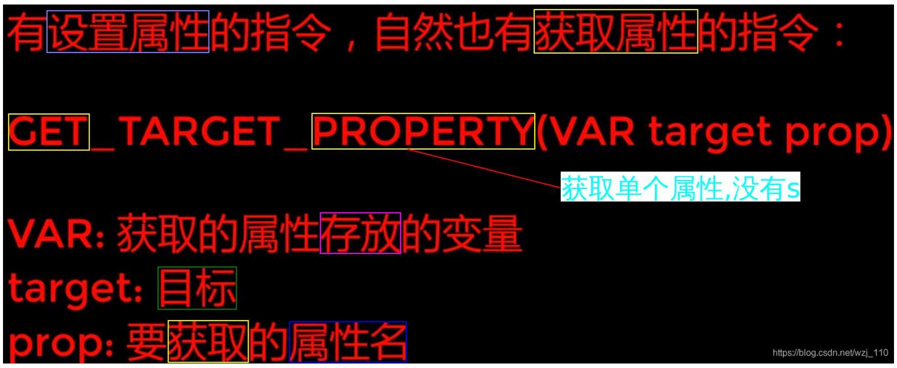
指令SET_TARGET_PROPERTIES可设置目标的属性

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

target1 target2: 指定要设置属性的目标
prop1 value1: 指定属性名和属性值，这里将演示OUTPUT_NAME、VERSION、SOVERSION三个属性

② 获取属性值

```
get_property: '获取'一个属性值

get_property(<variable>
               <GLOBAL             |
                DIRECTORY [dir]    |
                TARGET    <target> |
                SOURCE    <source> |
                TEST      <test>   |
                CACHE     <entry> |
                                              VARIABLE>
                PROPERTY <name>
                [SET | DEFINED |BRIEF_DOCS | FULL_DOCS])
```



有设置属性的指令，自然也有获取属性的指令：

GET_TARGET_PROPERTY(VAR target prop)

获取单个属性,没有s

VAR: 获取的属性存放的变量

target: 目标

prop: 要获取的属性名

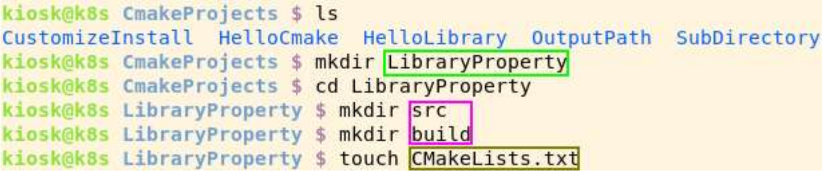
https://blog.csdn.net/wzj_110

必选项PROPERTY后面紧跟着要获取的属性的名字。如果指定了SET选项，则变量会被设置为一个布尔值，表明该属性是否已设置。如果指定了DEFINED选项，则变量也会被设置为一个布尔值，表明该属性是否已定义（如通过define_property）。如果定义了BRIEF_DOCS或FULL_DOCS选项，则该变量被设置为一个字符串，包含了对请求的属性的文档。如果该属性没有相关文件，则会返回NOTFOUND

二 实践

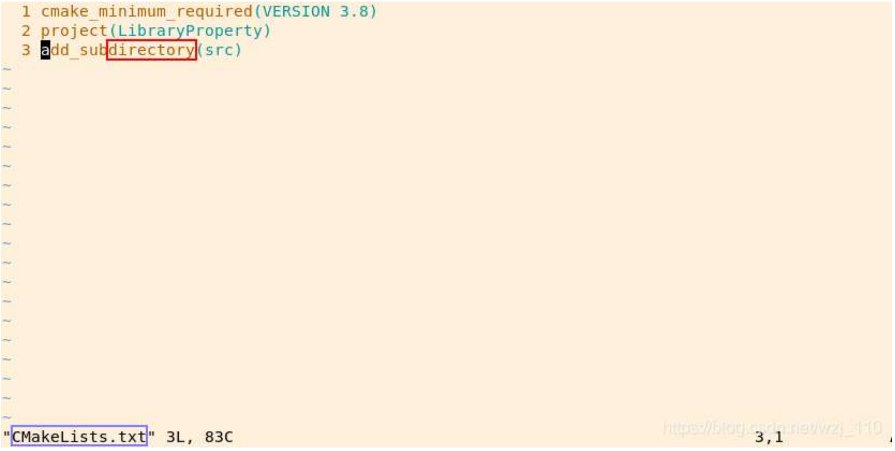
① 项目初始化

```
1） '新建'一个项目目录-->'LibraryProperty'-->'cd进入'

2） 创建'src'子目录用于'保存'源文件

3） 创建'build'子目录用于'外部编译'

4） 编写'项目根目录'下'CMakeLists.txt'文件
```



```
kiosk@k8s CmakeProjects $ ls
CustomizeInstall  HelloCmake  HelloLibrary  OutputPath  SubDirectory
kiosk@k8s CmakeProjects $ mkdir LibraryProperty
kiosk@k8s CmakeProjects $ cd LibraryProperty
kiosk@k8s LibraryProperty $ mkdir src
kiosk@k8s LibraryProperty $ mkdir build
kiosk@k8s LibraryProperty $ touch CMakeLists.txt
```

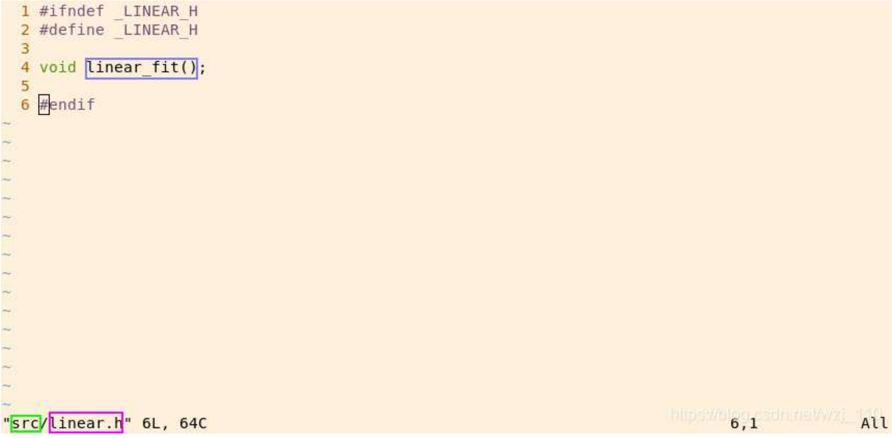
② 编写根目录的CMakeLists.txt文件



```
cmake_minimum_required(VERSION 3.8)
project(LibraryProperty)
add_subdirectory(src)
~
"CMakeLists.txt" 3L, 83C                                       3,1
```

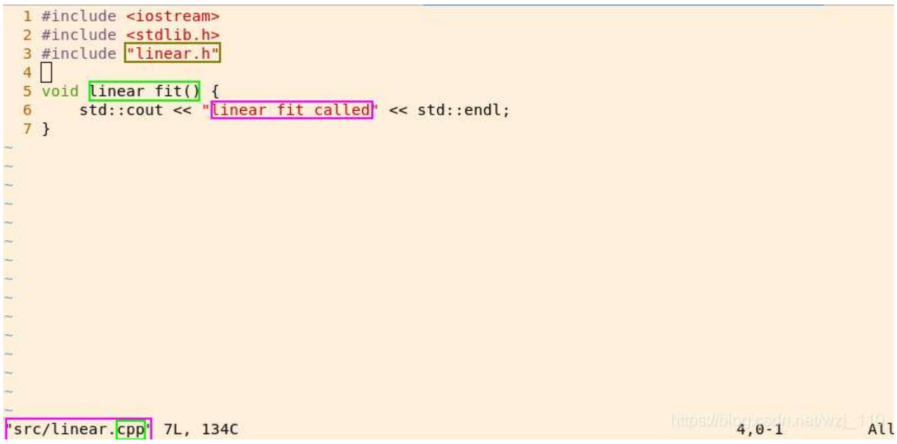
https://blog.csdn.net/wzj_110

③ src子目录添加要**生成库**的头文件和源文件



```
#ifndef _LINEAR_H
#define _LINEAR_H

void linear_fit();

#endif
```

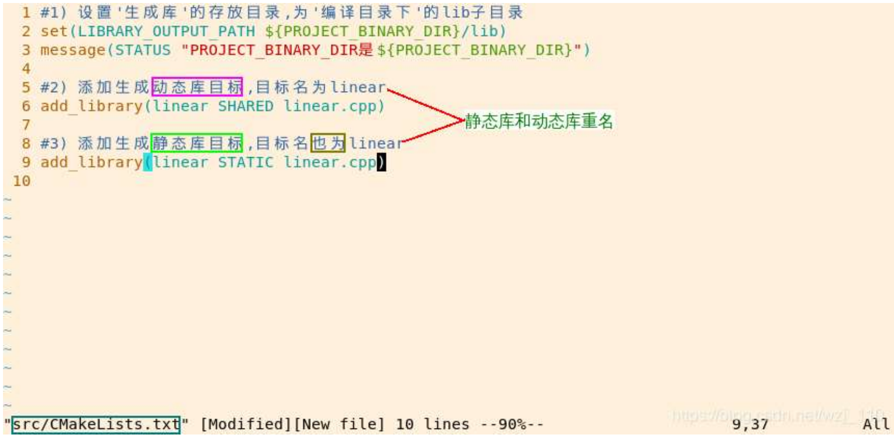
```
"src/linear.h" 6L, 64C                                          6,1           All
```



```
#include <iostream>
#include <stdlib.h>
#include "linear.h"

void linear_fit() {
    std::cout << "linear_fit_called" << std::endl;
}
```

```
"src/linear.cpp" 7L, 134C                                       4,0-1         All
```

④ 编写src子目录下的CMakeLists.txt文件

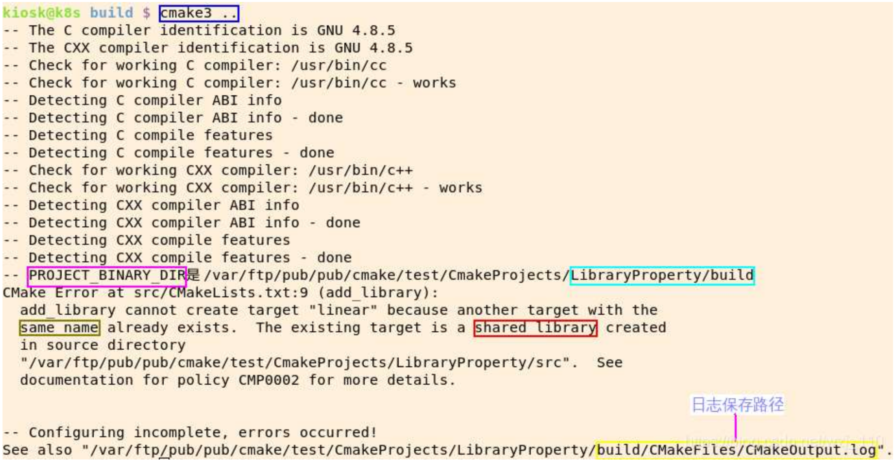


```
#1) 设置'生成库'的存放目录,为'编译目录下'的lib子目录
set(LIBRARY_OUTPUT_PATH ${PROJECT_BINARY_DIR}/lib)
message(STATUS "PROJECT_BINARY_DIR是${PROJECT_BINARY_DIR}")

#2) 添加生成动态库目标,目标名为linear
add_library(linear SHARED linear.cpp)

#3) 添加生成静态库目标,目标名也为linear
add_library(linear STATIC linear.cpp)

```

```
"src/CMakeLists.txt" [Modified][New file] 10 lines --90%--      9,37          All
```

⑤ 报错1

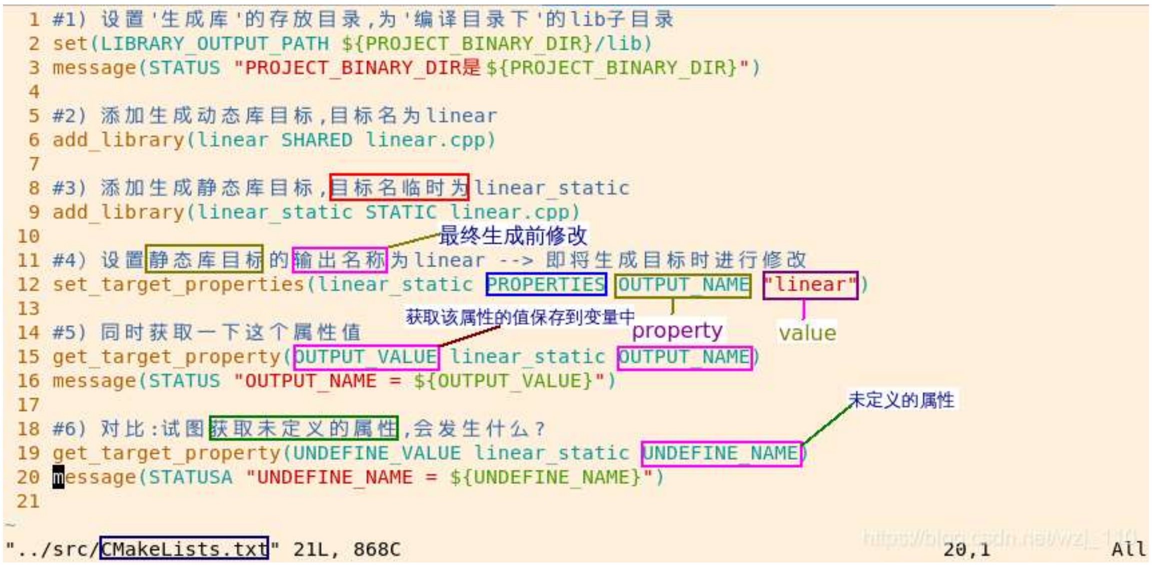
```
原因:  不能有'相同名称'的目标 -->'特制前缀'

疑惑:  理论上一个为'.so'结尾、一个为'.a'结尾的?
```



```
kiosk@k8s build $ cmake3 ..
-- The C compiler identification is GNU 4.8.5
-- The CXX compiler identification is GNU 4.8.5
-- 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
-- Detecting C compile features
-- Detecting C compile features - 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
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- PROJECT_BINARY_DIR是/var/ftp/pub/pub/cmake/test/CmakeProjects/LibraryProperty/build
CMake Error at src/CMakeLists.txt:9 (add_library):
  add_library cannot create target "linear" because another target with the
  same name already exists.  The existing target is a shared library created
  in source directory
  "/var/ftp/pub/pub/cmake/test/CmakeProjects/LibraryProperty/src".  See
  documentation for policy CMP0002 for more details.


-- Configuring incomplete, errors occurred!
See also "/var/ftp/pub/pub/cmake/test/CmakeProjects/LibraryProperty/build/CMakeFiles/CMakeOutput.log".
```

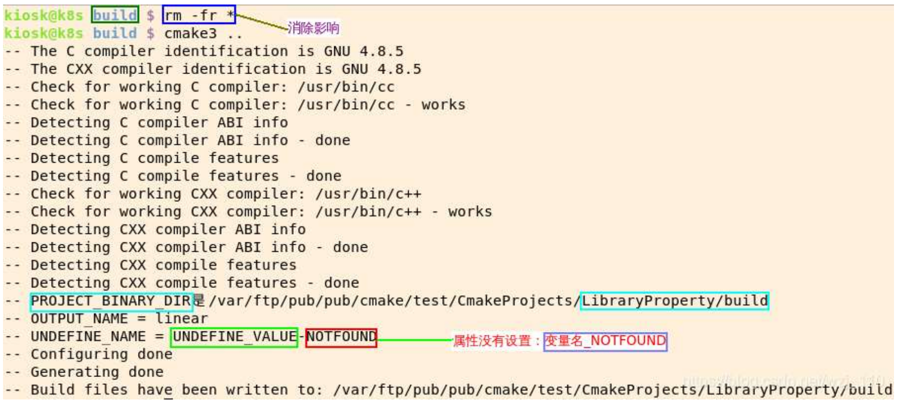
⑥ 如何解决上面的报错

```
方式1:  '随便'修改其中一个'库'的名称 --> '不再演示'

场景:  如果想让'静态库'和'动态库'同一前缀呢?

备注:  20行'修改'为--> message(STATUS "UNDEFINE_NAME = ${UNDEFINE_VALUE}")

体会:  '语法格式'
```

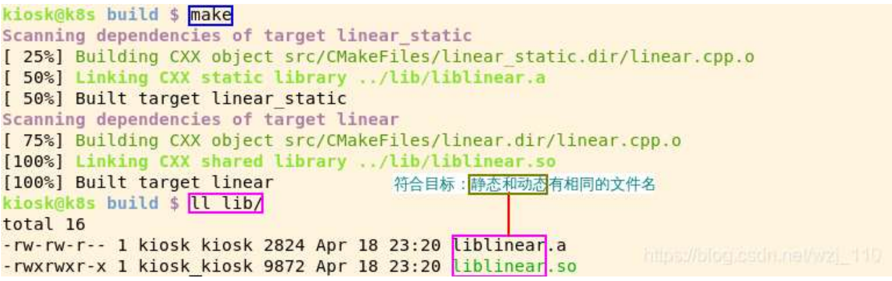


```
 1 #1) 设置'生成库'的存放目录,为'编译目录下'的lib子目录
 2 set(LIBRARY_OUTPUT_PATH ${PROJECT_BINARY_DIR}/lib)
 3 message(STATUS "PROJECT_BINARY_DIR是${PROJECT_BINARY_DIR}")
 4
 5 #2) 添加生成动态库目标,目标名为linear
 6 add_library(linear SHARED linear.cpp)
 7
 8 #3) 添加生成静态库目标,目标名临时为linear_static
 9 add_library(linear_static STATIC linear.cpp)
10
11 #4) 设置静态库目标的输出名称为linear --> 即将生成目标时进行修改
12 set_target_properties(linear_static PROPERTIES OUTPUT_NAME "linear")
13
14 #5) 同时获取一下这个属性值
15 get_target_property(OUTPUT_VALUE linear_static OUTPUT_NAME)
16 message(STATUS "OUTPUT_NAME = ${OUTPUT_VALUE}")
17
18 #6) 对比:试图获取未定义的属性,会发生什么?
19 get_target_property(UNDEFINE_VALUE linear_static UNDEFINE_NAME)
20 message(STATUSA "UNDEFINE_NAME = ${UNDEFINE_NAME}")
21
~
"../src/CMakeLists.txt" 21L, 868C                                20,1          All
```

⑦ 继续测试

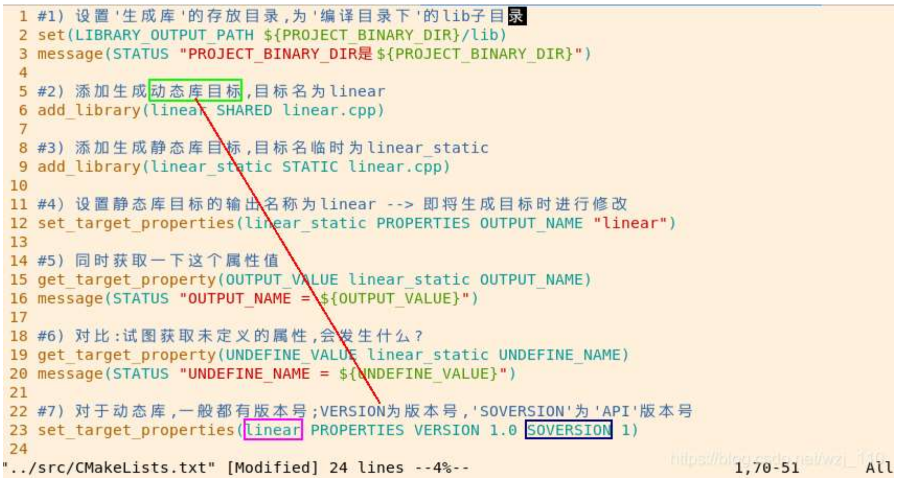


```
kiosk@k8s build $ rm -fr *
kiosk@k8s build $ cmake3 ..
-- The C compiler identification is GNU 4.8.5
-- The CXX compiler identification is GNU 4.8.5
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc - works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - 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
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- PROJECT_BINARY_DIR是/var/ftp/pub/pub/cmake/test/CmakeProjects/LibraryProperty/build
-- OUTPUT_NAME = linear
-- UNDEFINE_NAME = UNDEFINE_VALUE-NOTFOUND
-- Configuring done
-- Generating done
-- Build files have been written to: /var/ftp/pub/pub/cmake/test/CmakeProjects/LibraryProperty/build
```



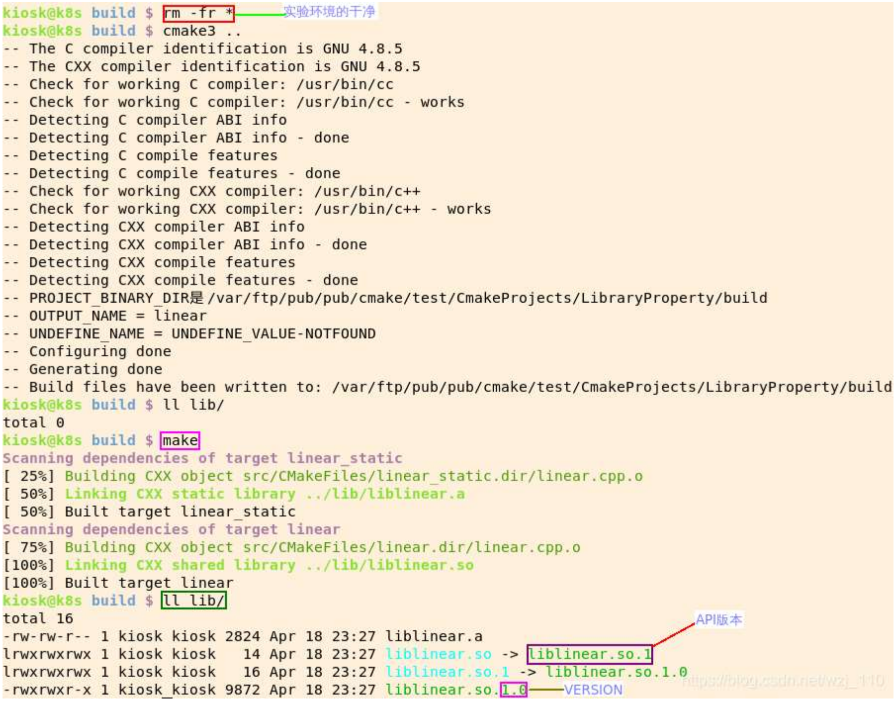
```
kiosk@k8s build $ make
Scanning dependencies of target linear_static
[ 25%] Building CXX object src/CMakeFiles/linear_static.dir/linear.cpp.o
[ 50%] Linking CXX static library ../lib/liblinear.a
[ 50%] Built target linear_static
Scanning dependencies of target linear
[ 75%] Building CXX object src/CMakeFiles/linear.dir/linear.cpp.o
[100%] Linking CXX shared library ../lib/liblinear.so
[100%] Built target linear
kiosk@k8s build $ ll lib/
total 16
-rw-rw-r-- 1 kiosk kiosk 2824 Apr 18 23:20 liblinear.a
-rwxrwxr-x 1 kiosk kiosk 9872 Apr 18 23:20 liblinear.so
```

⑧ 动态库的其它属性设置



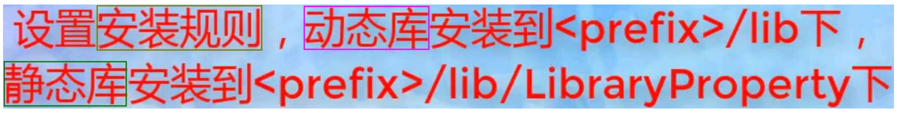
```
#1) 设置'生成库'的存放目录,为'编译目录下'的lib子目录
set(LIBRARY_OUTPUT_PATH ${PROJECT_BINARY_DIR}/lib)
message(STATUS "PROJECT_BINARY_DIR是${PROJECT_BINARY_DIR}")

#2) 添加生成动态库目标,目标名为linear
add_library(linear SHARED linear.cpp)

#3) 添加生成静态库目标,目标名临时为linear_static
add_library(linear_static STATIC linear.cpp)

#4) 设置静态库目标的输出名称为linear --> 即将生成目标时进行修改
set_target_properties(linear_static PROPERTIES OUTPUT_NAME "linear")

#5) 同时获取一下这个属性值
get_target_property(OUTPUT_VALUE linear_static OUTPUT_NAME)
message(STATUS "OUTPUT_NAME = ${OUTPUT_VALUE}")

#6) 对比:试图获取未定义的属性,会发生什么?
get_target_property(UNDEFINE_VALUE linear_static UNDEFINE_NAME)
message(STATUS "UNDEFINE_NAME = ${UNDEFINE_VALUE}")

#7) 对于动态库,一般都有版本号;VERSION为版本号,'SOVERSION'为'API'版本号
set_target_properties(linear PROPERTIES VERSION 1.0 SOVERSION 1)

"../src/CMakeLists.txt" [Modified] 24 lines --4%--                    1,70-51       All
```

```
效果： 动态库带'有版本号',并且有两个'链接文件'

1）带有'VERSION'的库

2）带有'SOVERSION'的库
```

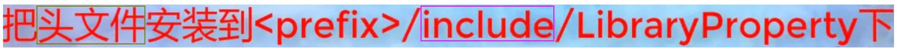


```
kiosk@k8s build $ rm -fr *
kiosk@k8s build $ cmake3 ..
-- The C compiler identification is GNU 4.8.5
-- The CXX compiler identification is GNU 4.8.5
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc - works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - 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
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- PROJECT_BINARY_DIR是/var/ftp/pub/pub/cmake/test/CmakeProjects/LibraryProperty/build
-- OUTPUT_NAME = linear
-- UNDEFINE_NAME = UNDEFINE_VALUE-NOTFOUND
-- Configuring done
-- Generating done
-- Build files have been written to: /var/ftp/pub/pub/cmake/test/CmakeProjects/LibraryProperty/build
kiosk@k8s build $ ll lib/
total 0
kiosk@k8s build $ make
Scanning dependencies of target linear_static
[ 25%] Building CXX object src/CMakeFiles/linear_static.dir/linear.cpp.o
[ 50%] Linking CXX static library ../lib/liblinear.a
[ 50%] Built target linear_static
Scanning dependencies of target linear
[ 75%] Building CXX object src/CMakeFiles/linear.dir/linear.cpp.o
[100%] Linking CXX shared library ../lib/liblinear.so
[100%] Built target linear
kiosk@k8s build $ ll lib/
total 16
-rw-rw-r-- 1 kiosk kiosk 2824 Apr 18 23:27 liblinear.a
lrwxrwxrwx 1 kiosk kiosk   14 Apr 18 23:27 liblinear.so -> liblinear.so.1
lrwxrwxrwx 1 kiosk kiosk   16 Apr 18 23:27 liblinear.so.1 -> liblinear.so.1.0
-rwxrwxr-x 1 kiosk kiosk 9872 Apr 18 23:27 liblinear.so.1.0
```

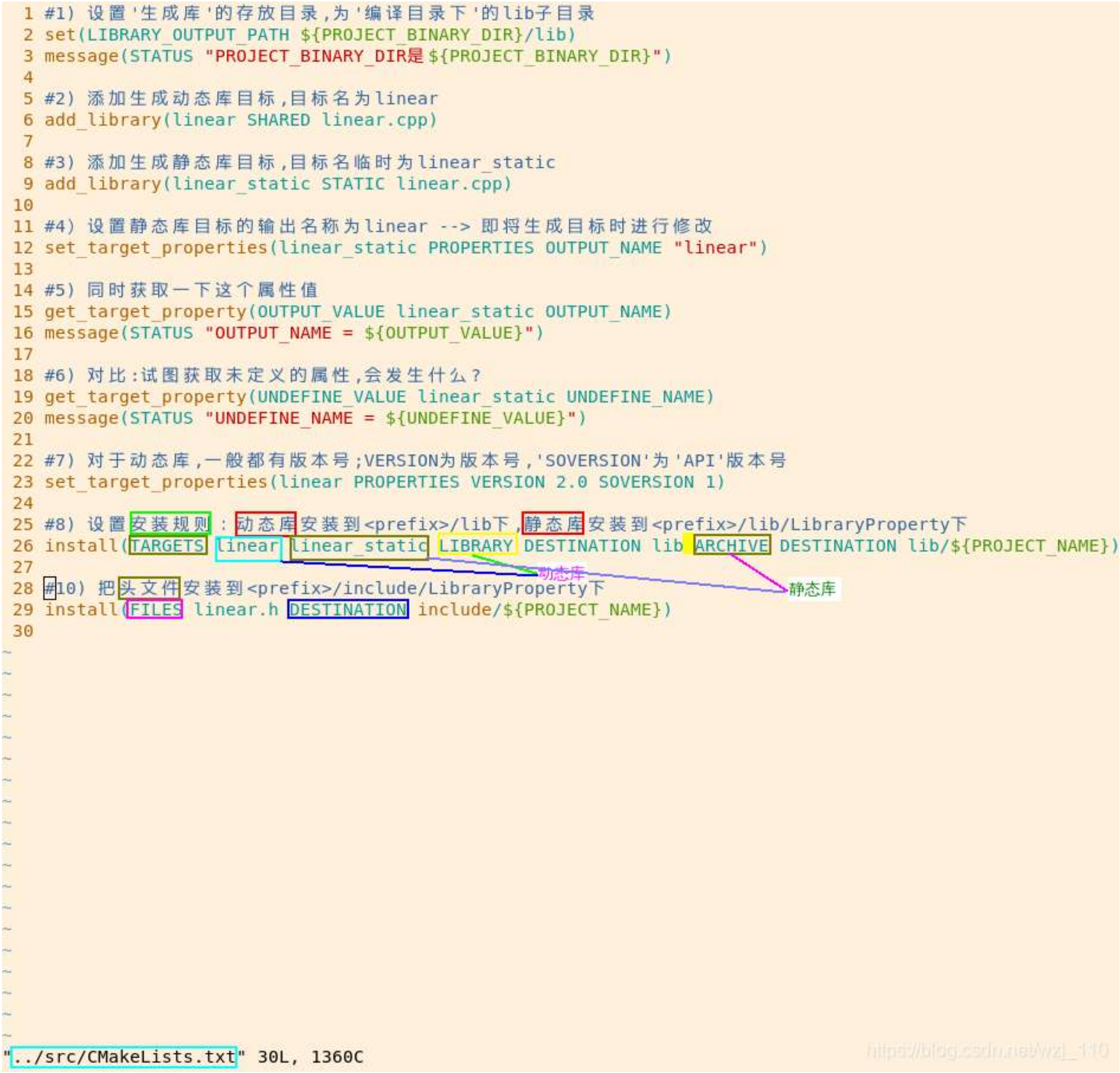
⑨ 继续修改



设置安装规则，动态库安装到<prefix>/lib下，静态库安装到<prefix>/lib/LibraryProperty下



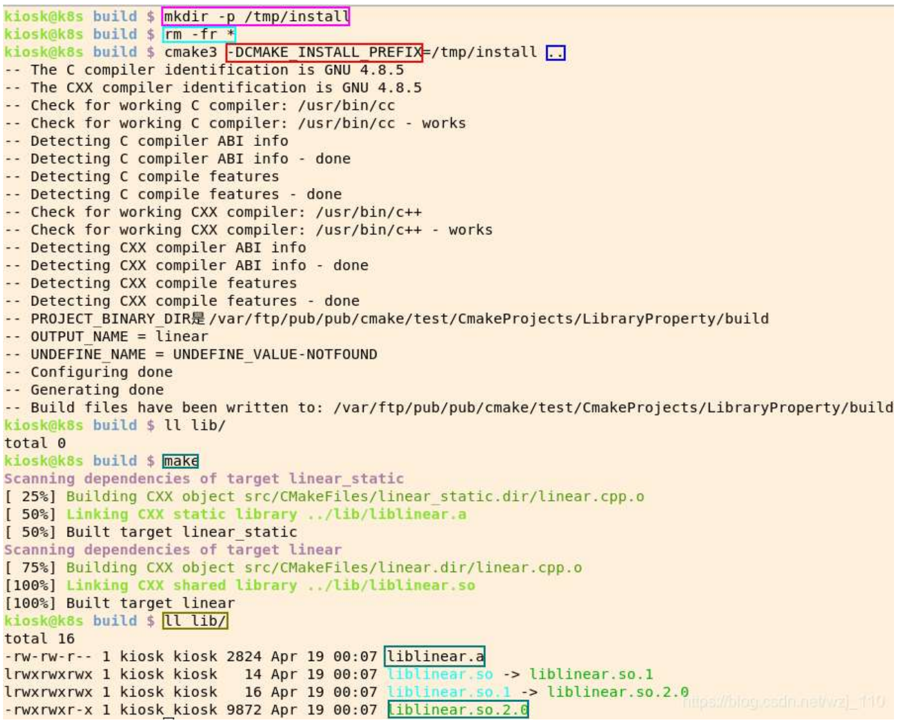
把头文件安装到<prefix>/include/LibraryProperty下



```
#1) 设置'生成库'的存放目录,为'编译目录下'的lib子目录
set(LIBRARY_OUTPUT_PATH ${PROJECT_BINARY_DIR}/lib)
message(STATUS "PROJECT_BINARY_DIR是${PROJECT_BINARY_DIR}")

#2) 添加生成动态库目标,目标名为linear
add_library(linear SHARED linear.cpp)

#3) 添加生成静态库目标,目标名临时为linear_static
add_library(linear_static STATIC linear.cpp)

#4) 设置静态库目标的输出名称为linear --> 即将生成目标时进行修改
set_target_properties(linear_static PROPERTIES OUTPUT_NAME "linear")

#5) 同时获取一下这个属性值
get_target_property(OUTPUT_VALUE linear_static OUTPUT_NAME)
message(STATUS "OUTPUT_NAME = ${OUTPUT_VALUE}")

#6) 对比:试图获取未定义的属性,会发生什么?
get_target_property(UNDEFINE_VALUE linear_static UNDEFINE_NAME)
message(STATUS "UNDEFINE_NAME = ${UNDEFINE_VALUE}")

#7) 对于动态库,一般都有版本号;VERSION为版本号,'SOVERSION'为'API'版本号
set_target_properties(linear PROPERTIES VERSION 2.0 SOVERSION 1)

#8) 设置安装规则 : 动态库安装到<prefix>/lib下,静态库安装到<prefix>/lib/LibraryProperty下
install(TARGETS linear linear_static LIBRARY DESTINATION lib ARCHIVE DESTINATION lib/${PROJECT_NAME})

#10) 把头文件安装到<prefix>/include/LibraryProperty下
install(FILES linear.h DESTINATION include/${PROJECT_NAME})

"../src/CMakeLists.txt" 30L, 1360C
```

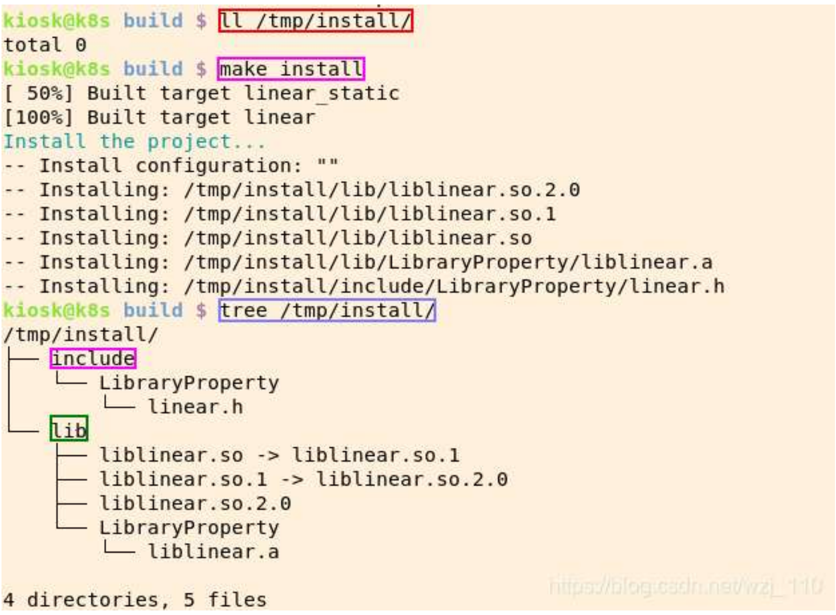
install参数

⑩ 测试



```
kiosk@k8s build $ mkdir -p /tmp/install
kiosk@k8s build $ rm -fr *
kiosk@k8s build $ cmake3 -DCMAKE_INSTALL_PREFIX=/tmp/install ..
-- The C compiler identification is GNU 4.8.5
-- The CXX compiler identification is GNU 4.8.5
-- 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
-- Detecting C compile features
-- Detecting C compile features - 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
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- PROJECT_BINARY_DIR是/var/ftp/pub/pub/cmake/test/CmakeProjects/LibraryProperty/build
-- OUTPUT_NAME = linear
-- UNDEFINE_NAME = UNDEFINE_VALUE-NOTFOUND
-- Configuring done
-- Generating done
-- Build files have been written to: /var/ftp/pub/pub/cmake/test/CmakeProjects/LibraryProperty/build
kiosk@k8s build $ ll lib/
total 0
kiosk@k8s build $ make
Scanning dependencies of target linear_static
[ 25%] Building CXX object src/CMakeFiles/linear_static.dir/linear.cpp.o
[ 50%] Linking CXX static library ../lib/liblinear.a
[ 50%] Built target linear_static
Scanning dependencies of target linear
[ 75%] Building CXX object src/CMakeFiles/linear.dir/linear.cpp.o
[100%] Linking CXX shared library ../lib/liblinear.so
[100%] Built target linear
kiosk@k8s build $ ll lib/
total 16
-rw-rw-r-- 1 kiosk kiosk 2824 Apr 19 00:07 liblinear.a
lrwxrwxrwx 1 kiosk kiosk   14 Apr 19 00:07 liblinear.so -> liblinear.so.1
lrwxrwxrwx 1 kiosk kiosk   16 Apr 19 00:07 liblinear.so.1 -> liblinear.so.2.0
-rwxrwxr-x 1 kiosk kiosk 9872 Apr 19 00:07 liblinear.so.2.0
```

备注： '库'被成功安装到'指定目录'



```
kiosk@k8s build $ ll /tmp/install/
total 0
kiosk@k8s build $ make install
[ 50%] Built target linear_static
[100%] Built target linear
Install the project...
-- Install configuration: ""
-- Installing: /tmp/install/lib/liblinear.so.2.0
-- Installing: /tmp/install/lib/liblinear.so.1
-- Installing: /tmp/install/lib/liblinear.so
-- Installing: /tmp/install/lib/LibraryProperty/liblinear.a
-- Installing: /tmp/install/include/LibraryProperty/linear.h
kiosk@k8s build $ tree /tmp/install/
/tmp/install/
├── include
│   └── LibraryProperty
│       └── linear.h
└── lib
    ├── liblinear.so -> liblinear.so.1
    ├── liblinear.so.1 -> liblinear.so.2.0
    ├── liblinear.so.2.0
    └── LibraryProperty
        └── liblinear.a

4 directories, 5 files
```

显示推荐内容