cmake(三)CMake输出自定义信息

一 message函数

① 清理上次的构建产物

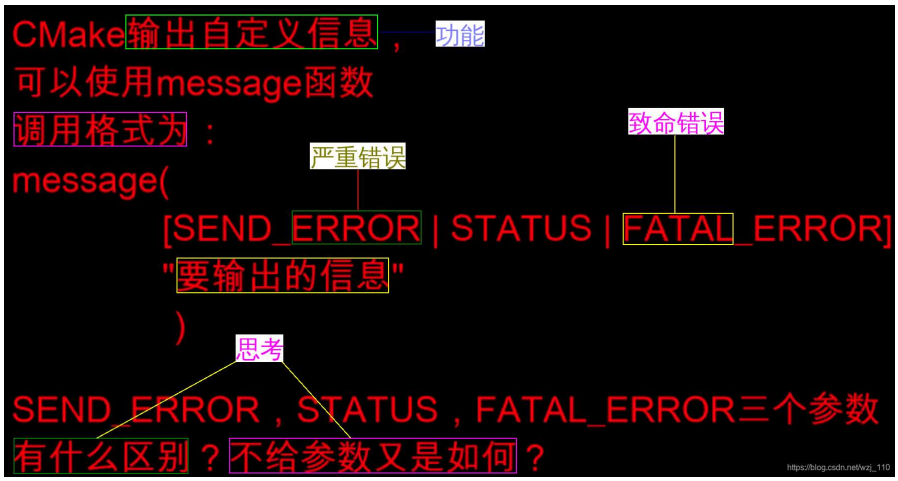
```
kiosk@k8s CmakeProjects $ ls
HelloCmake
kiosk@k8s CmakeProjects $ cd HelloCmake/
kiosk@k8s HelloCmake $ ls
CMakeCache.txt  CMakeFiles  cmake_install.cmake  CMakeLists.txt  hello_cmake  HelloCmake.cpp  Makefile
kiosk@k8s HelloCmake $ rm -fr  CMakeCache.txt CMakeFiles cmake_install.cmake hello_cmake Makefile
kiosk@k8s HelloCmake $ ls
CMakeLists.txt  HelloCmake.cpp
kiosk@k8s HelloCmake $ cd ../../
kiosk@k8s test $ tree CmakeProjects/
CmakeProjects/
└── HelloCmake
    ├── CMakeLists.txt
    └── HelloCmake.cpp

1 directory, 2 files
```

只保留这两个

恢复到cmake3之前的模样

② message函数

```
1）CMake 的命令行工具会在 'stdout' 上显示 'STATUS' 消息，在 stderr 上显示'其他所有'消息

2）CMake '警告和错误消息'的文本显示使用的是一种'简单的标记语言';文本'没有缩进',超过长度的行会'回卷'，段落之间以'新行'做为'
```



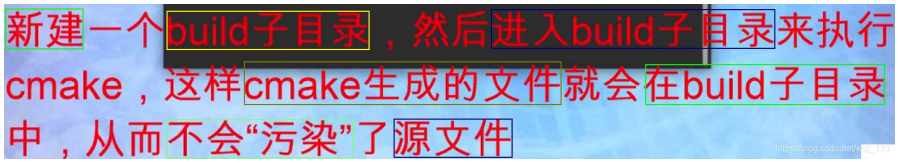

```
+++++++++'消息不同级别的行为'+++++++++

 (无) = '重要'消息;
 STATUS = '非重要'消息; -->'常用'
 WARNING = CMake '警告', 会继续执行;
 AUTHOR_WARNING = CMake 警告 (dev), 会'继续'执行;
 SEND_ERROR = CMake 错误, '继续执行', 但是会'跳过生成的步骤'; -->'常用'
 FATAL_ERROR = CMake 错误, '终止所有'处理过程;  -->'常用'
```

cmake的内置变量

```
'典型场景':  使用message函数作为 "调试输出",获取内置变量的'精准值'
```

③ 做如下的操作

使用命令"cmake .." -->'习惯'将cmake输出到'build目录'




```
kiosk@k8s test $ tree CmakeProjects/
CmakeProjects/
└── HelloCmake
    ├── CMakeLists.txt
    └── HelloCmake.cpp

1 directory, 2 files
kiosk@k8s test $ cd CmakeProjects/HelloCmake/
kiosk@k8s HelloCmake $ ls
CMakeLists.txt  HelloCmake.cpp
kiosk@k8s HelloCmake $ mkdir build
kiosk@k8s HelloCmake $ cd build/
kiosk@k8s build $
```

④ 演示message无参和STATUS级别的用法

1）修改CMakeLists.txt文件

```
1）'无参' -->'不指定'消息的'级别'

2）在'build'目录下'修改'CMakeLists文件
```

```
kiosk@k8s build $ vim ../CMakeLists.txt
```

```
project(HelloCmake)
set(SRC_LIST HelloCmake.cpp)
message("This is no paraments!")        不指定级别
message(STATUS "This is a status level message!")        status级别
add_executable(hello_cmake ${SRC_LIST})
```

2）测试

```
思考：  'cmake3 ..'会'只找'该级目录下的'一级'CMakeLists.txt文件,'还是'包括子目录下的'CMakeLists.txt'文件,'分别构建'
```

```
kiosk@k8s HelloCmake $ cd build/
kiosk@k8s build $ vim ../CMakeLists.txt
kiosk@k8s build $ cmake3 ..        两个点..代表上一层目录,告诉cmake3到上一层目录去寻找CmakeLists文件
-- 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          1）没有指定message级别的原样输出
-- Detecting CXX compile features                  2）指定STATUS的会在输出的信息前面加上--
-- Detecting CXX compile features - done
This is no paraments!
-- This is a status level message!          区别：staus多了--,可以看到正常情况下也是--,猜测也是status级别
-- Configuring done
-- Generating done
-- Build files have been written to: /var/ftp/pub/pub/cmake/test/CmakeProjects/HelloCmake/build
kiosk@k8s build $ ls          此时cmake生成的文件全部都在bubild目录,全部删除也不会影响源文件
CMakeCache.txt  CMakeFiles  cmake_install.cmake  Makefile
```

⑤ 测试SEND_ERROR

1）清理环境,修改CMakeLists.txt文件

```
kiosk@k8s build $ ls
CMakeCache.txt  CMakeFiles  cmake_install.cmake  Makefile
kiosk@k8s build $ rm -fr *
kiosk@k8s build $ vim ../CMakeLists.txt
```

2）修改后的内容

```
1 project(HelloCmake)
2 set(SRC_LIST HelloCmake.cpp)
3 message("This is no paraments!")
4 message(STATUS "This is a status level message!")
5 message(SEND_ERROR "CMake 错误，继续执行，但是会跳过生成的步骤")
6 add_executable(hello_cmake ${SRC_LIST})
```

添加一行

类似：shell中的set +e 某个命令报错,$?!=0继续向下执行

3）测试

从输出的结果来看： 'SEND_ERROR'发送了'错误'信息,而且'跳过生成(核心Makefile)'过程

```
kiosk@k8s build $ ls
CMakeCache.txt  CMakeFiles  cmake_install.cmake  Makefile
kiosk@k8s build $ rm -fr *
kiosk@k8s build $ vim ../CMakeLists.txt
kiosk@k8s build $ ls
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
This is no paraments!
-- This is a status level message!
CMake Error at CMakeLists.txt:5 (message):
  CMake 错误，继续执行，但是会跳过生成的步骤

-- Configuring incomplete, errors occurred!
See also "/var/ftp/pub/pub/cmake/test/CmakeProjects/HelloCmake/build/CMakeFiles/CMakeOutput.log".
kiosk@k8s build $ ls
CMakeCache.txt  CMakeFiles
```

正常和SEND_ERROR cmake3 ..区别

报错的位置

报错信息

跳过生成cmake_install.cmake Makefile

核心是这个

⑥ 测试FATAL_ERROR

细节： 把'FATAL_ERROR'放在'SEND_ERROR'的下面-->分析'二者顺序'对结果的'影响'

1）清理环境,修改文件

```
kiosk@k8s build $ rm -fr *
kiosk@k8s build $ vim ../CMakeLists.txt
```

2）内容1和测试1

备注： 演示'不是太完美',后续可以在'FATAL_ERROR'再加一个'无参'或者'STTAUS'级别的message

```
1 project(HelloCmake)
2 set(SRC_LIST HelloCmake.cpp)
3 message("This is no paraments!")
4 message(STATUS "This is a status level message!")
5 message(SEND_ERROR "CMake 错误，继续执行，但是会跳过生成的步骤")
6 message(FATAL_ERROR "Make 错误，终止所有处理过程-->类似shell 的 exit 非0,退出脚本")
7 add_executable(hello_cmake ${SRC_LIST})
```

注意二者的顺序

```
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
This is no paraments!
-- This is a status level message!
CMake Error at CMakeLists.txt:5 (message):
  CMake 错误，继续执行，但是会跳过生成的步骤


CMake Error at CMakeLists.txt:6 (message):
  Make 错误，终止所有处理过程-->类似shell 的 exit
  非0,退出脚本


-- Configuring incomplete, errors occurred!
See also "/var/ftp/pub/pub/cmake/test/CmakeProjects/HelloCmake/build/CMakeFiles/CMakeOutput.log".
kiosk@k8s build $ ls
CMakeCache.txt  CMakeFiles
```

执行完SEND_ERROR还会继续执行FATAL_ERROR

3）内容2和测试2

操作： '掉换'二者的'顺序'

```
project(HelloCmake)
set(SRC_LIST HelloCmake.cpp)
message("This is no paraments!")
message(STATUS "This is a status level message!")
message(FATAL_ERROR "Make 错误，终止所有处理过程-->类似shell 的 exit 非0,退出脚本")
message(SEND_ERROR "CMake 错误，继续执行，但是会跳过生成的步骤")
add_executable(hello_cmake ${SRC_LIST})
```

致命错误

```
kiosk@k8s build $ rm -fr *
kiosk@k8s build $ vim ../CMakeLists.txt
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
This is no paraments!
-- This is a status level message!
CMake Error at CMakeLists.txt:5 (message):
  Make 错误，终止所有处理过程-->类似shell 的 exit
  非0,退出脚本


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

FATAL_ERROR执行完就退出了,后续的不再执行了

⑦ message打印内置变量

```
cmake_minimum_required(VERSION 3.18)

project(show_vars VERSION 1.0.1)

# 为了分行确定输出内容
message("")

message("1.PROJECT_BINARY_DIR = ${PROJECT_BINARY_DIR}")
message("2.PROJECT_SOURCE _DIR = ${_DIR}")
message("3.CMAKE_CURRENT_SOURCE_DIR = ${CMAKE_CURRENT_SOURCE_DIR}")
message("4.CMAKE_CURRRENT_BINARY_DIR = ${CMAKE_CURRRENT_BINARY_DIR}")
message("5.CMAKE_CURRENT_LIST_FILE = ${CMAKE_CURRENT_LIST_FILE}")
message("6.CMAKE_CURRENT_LIST_LINE = ${CMAKE_CURRENT_LIST_LINE}")
message("7.CMAKE_MODULE_PATH = ${CMAKE_MODULE_PATH}")
message("8.CMAKE_SOURCE_DIR = ${CMAKE_SOURCE_DIR}")
message("9.EXECUTABLE_OUTPUT_PATH = ${EXECUTABLE_OUTPUT_PATH}")
message("10.LIBRARY_OUTPUT_PATH = ${LIBRARY_OUTPUT_PATH}")
message("11.PROJECT_NAME = ${PROJECT_NAME}")
message("12.PROJECT_VERSION_MAJOR = ${PROJECT_VERSION_MAJOR}")
message("13.PROJECT_VERSION_MINOR = ${PROJECT_VERSION_MINOR}")
```

```
message("14.PROJECT_VERSION_PATCH = ${PROJECT_VERSION_PATCH}")
message("15.CMAKE_SYSTEM = ${CMAKE_SYSTEM}")
message("16.CMAKE_SYSTEM_NAME = ${CMAKE_SYSTEM_NAME}")
message("17.CMAKE_SYSTEM_VERSION = ${CMAKE_SYSTEM_VERSION}")
message("18.BUILD_SHARED_LIBS = ${BUILD_SHARED_LIBS}")
message("19.CMAKE_C_FLAGS = ${CMAKE_C_FLAGS}")
message("20.CMAKE_CXX_FLAGS = ${CMAKE_CXX_FLAGS}")
message("21.CMAKE_SYSTEM_PROCESSOR   = ${CMAKE_SYSTEM_PROCESSOR}")
# 为了分行确定输出内容
message("")
```