

原 FFMpeg示例程序合集-批量编译脚本

2015年02月23日 15:20:04 阅读数：10272

此前做了一系列有关FFmpeg的示例程序，组成了《[最简单的FFmpeg示例程序合集](#)》，其中包含了如下项目：

simplest ffmpeg player:	最简单的基于FFmpeg的视频播放器
simplest ffmpeg audio player:	最简单的基于FFmpeg的音频播放器
simplest ffmpeg pic encoder:	最简单的基于FFmpeg的图像编码器
simplest ffmpeg video encoder:	最简单的基于FFmpeg的视频编码器
simplest ffmpeg audio encoder:	最简单的基于FFmpeg的音频编码器
simplest ffmpeg streamer:	最简单的基于FFmpeg的推流器（推送RTMP）
simplest ffmpeg mem handler:	最简单的基于FFmpeg的内存读写例子
simplest ffmpeg device:	最简单的基于FFmpeg的AVDevice例子
simplest ffmpeg format:	最简单的基于FFmpeg的封装格式方面的例子
simplest ffmpeg video filter:	最简单的基于FFmpeg的AVfilter例子（水印叠加）
simplest ffmpeg swscale:	最简单的基于FFmpeg的libswscale的示例

开始的时候，我只是在VC2010下调试通过了这些程序。去年就有不少朋友跟我提建议希望能把代码改成跨平台的。后来我调查了一下也发现FFmpeg确实在各个平台都有广泛的应用，因此也产生了把代码改成跨平台的意愿。但是改成跨平台代码的工作量比较大，所以一直也没有做出实际行动。今年春节前夕可算是得到了一阵相对自由的时间，于是果断进行了一阵子“连续战斗”，终于实现了这些工程在VC++，cl.exe，MinGW，Linux GCC以及MacOS GCC上面顺利的编译。在这个过程中，也写了各种编译器的编译脚本，在这里记录一下。

本文记录以下几种编译脚本：

VC++：simplest_ffmpeg_demos_compile_devenv_all.bat
CL.exe：simplest_ffmpeg_demos_compile_cl_all.bat
MinGW：simplest_ffmpeg_demos_compile_mingw_all.sh
GCC (Linux/MacOS)：simplest_ffmpeg_demos_compile_gcc_all.sh



下载地址：

【Github】<https://github.com/leixiaohua1020/leixiaohua1020.github.io/tree/master/batch>

准备工作

在记录具体的编译脚本之前，首先简单记录一下在不同平台下编译这些FFmpeg工程需要做的准备工作。注意这一步骤针对的是一台完全空白未做任何配置的“裸机”。如果已经编译安装过FFmpeg，就可以直接跳过这一步骤。

PS：一些示例程序需要安装SDL2，方法类似，不再记录。

VC++

VC++的编译是最简单的，相关的include和lib都已经配置好了，可以直接编译运行。

CL.exe

基本上等同于使用VC++编译，可以直接编译运行。

MinGW

两种方法：直接安装和源代码编译

直接安装

- (1)从FFmpeg Windows Build (<http://ffmpeg.zeranoe.com/>) 网站下载最新的shared 和dev版本的FFmpeg。
- (2)在Msys安装目录下创建“local”文件夹，“local”文件夹下创建“include”和“lib”文件夹。
- (3)将FFmpeg的dev版本下的include拷贝至{msys}/local/include；lib拷贝至{msys}/local/lib。
- (4)将FFmpeg的shared版本下的DLL拷贝至{mingw}/bin。

源代码编译

- (1)安装Yasm

从官网下载yasmXXX.exe，然后重命名为yasm.exe，拷贝至{mingw}/bin

PS:也可以下载源代码自己编译，稍微麻烦些，不再记录。

(2)编译安装SDL

PS：注意SDL的安装路径如果没有安装在{msys}/local目录下的话，configure的时候设置一下。

(3)编译安装libx264

(3)编译安装libfaac

(4)编译安装FFmpeg

```
[plain]
1. ./configure --enable-shared --enable-libfaac --enable-libx264 --enable-gpl --enable-nonfree
2. make
3. make install
```

Linux

(0)前期准备

某些Linux没有安装gcc/g++，需要先安装gcc/g++

进入超级管理员“su”

(1)安装相关的类库

[Debian/Ubuntu]

```
[plain]
1. apt-get -y install yasm libfaac-dev libx264-dev libsdl1.2-dev
```

PS：这些类库也可以下载源代码手动编译，稍微麻烦些，不再记录。

[RedHat/Fedora/CentOS]

需要手动编译每个类库

(2)编译安装FFmpeg

```
[plain]
1. ./configure --enable-shared --enable-libfaac --enable-libx264 --enable-gpl --enable-nonfree
2. make
3. make install
```

MacOS

两种方法：直接安装和源代码编译

直接安装

(1)安装Homebrew

```
[plain]
1. ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

(2)安装FFmpeg

```
[plain]
1. brew install ffmpeg
```

源代码编译

(1)安装Homebrew

```
[plain]
1. ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

(2)安装相关的类库

```
[plain]
1. brew install yasm faac x264 sdl
```

(3)编译安装FFmpeg

进入超级管理员“su”

```
[plain]
1. ./configure --enable-shared --enable-libfaac --enable-libx264 --enable-gpl --enable-nonfree
2. make
3. make install
```

VC++



使用devenv.exe进行编译是最简单的一种命令行编译方式。这种编译方式和使用VC++代开*.sln解决方案文件然后单击“编译”按钮的效果是一样的。由于项目解决方案中已经做过了include, lib以及相关选项的设置, 所以不需要做各种参数的配置。下面这条命令编译Debug版本程序：

```
[plain]
1. devenv.exe simplest_ffmpeg_player2.sln /rebuild Debug
```

下面这条命令编译生成Release版本程序, 同时将编译过程中的日志输出到“sf_player_release_compile_log.txt”文件中：

```
[plain]
1. devenv.exe simplest_ffmpeg_player2.sln /rebuild Release /Out sf_player_release_compile_log.txt
```

使用devenv进行编译的时候, 需要VC++运行环境, 有2种方法可以设置：

- (1) 批处理调用VC++设置环境的脚本, 例如对于VC2010来说, 位于“D:\Program Files\Microsoft Visual Studio 10.0\VC\vcvarsall.bat”
- (2) 在“Visual Studio 命令提示符”中运行批处理（或者编译命令）。“Visual Studio 命令提示符”位于伴随着VC++安装, 位于“Visual Studio Tools”目录下。

完整的脚本simplest_ffmpeg_demos_compile_devenv_all.bat如下所示。将脚本拷贝至《最简单的FFmpeg示例程序合集》所在目录运行, 就可以编译所有项目的Release版本和Debug版本, 并且输出相关的编译日志。

```
[plain]
1. @echo off
2. echo -----
3. echo simplest ffmpeg demos list ---- Devenv compile all
4. echo Lei Xiaohua
5. echo Communication University of China / Digital TV Technology
6. echo leixiaohua1020@126.com
7. echo http://blog.csdn.net/leixiaohua1020
8. echo -----
9.
10. set logfile=compile_log.txt
11. ::VS2010 Environment
12. call "D:\Program Files\Microsoft Visual Studio 10.0\VC\vcvarsall.bat"
13. ::=====
14. echo.
15. echo Compile simplest_ffmpeg_player...
16. cd simplest_ffmpeg_player
17. devenv.exe simplest_ffmpeg_player2.sln /rebuild Debug /Out ../sf_player_debug %logfile%
18. devenv.exe simplest_ffmpeg_player2.sln /rebuild Release /Out ../sf_player_release %logfile%
19. cd ..
20.
21. ::=====
22. echo.
23. echo Compile simplest_ffmpeg_audio_player...
24. cd simplest_ffmpeg_audio_player
25. devenv.exe simplest_ffmpeg_audio_player2.sln /rebuild Debug /Out ../sf_audio_player_debug %logfile%
26. devenv.exe simplest_ffmpeg_audio_player2.sln /rebuild Release /Out ../sf_audio_player_release %logfile%
27. cd ..
28.
29. ::=====
30. echo.
31. echo Compile simplest_ffmpeg_picture_encoder...
32. cd simplest_ffmpeg_picture_encoder
33. devenv.exe simplest_ffmpeg_picture_encoder.sln /rebuild Debug /Out ../sf_picture_encoder_debug %logfile%
34. devenv.exe simplest_ffmpeg_picture_encoder.sln /rebuild Release /Out ../sf_picture_encoder_release %logfile%
35. cd ..
36.
37. ::=====
38. echo.
39. echo Compile simplest_ffmpeg_video_encoder...
40. cd simplest_ffmpeg_video_encoder
41. devenv.exe simplest_ffmpeg_video_encoder.sln /rebuild Debug /Out ../sf_video_encoder_debug %logfile%
42. devenv.exe simplest_ffmpeg_video_encoder.sln /rebuild Release /Out ../sf_video_encoder_release %logfile%
43. cd ..
```

```

44.
45. ::=====
46. echo.
47. echo Compile simplest_ffmpeg_audio_encoder...
48. cd simplest_ffmpeg_audio_encoder
49. devenv.exe simplest_ffmpeg_audio_encoder.sln /rebuild Debug /Out ../sf_audio_encoder_debug_%logfile%
50. devenv.exe simplest_ffmpeg_audio_encoder.sln /rebuild Release /Out ../sf_audio_encoder_release_%logfile%
51. cd ..
52.
53. ::=====
54. echo.
55. echo Compile simplest_ffmpeg_streamer...
56. cd simplest_ffmpeg_streamer
57. devenv.exe simplest_ffmpeg_streamer.sln /rebuild Debug /Out ../sf_streamer_debug_%logfile%
58. devenv.exe simplest_ffmpeg_streamer.sln /rebuild Release /Out ../sf_streamer_release_%logfile%
59. cd ..
60.
61. ::=====
62. echo.
63. echo Compile simplest_ffmpeg_mem_handler...
64. cd simplest_ffmpeg_mem_handler
65. devenv.exe simplest_ffmpeg_mem_handler.sln /rebuild Debug /Out ../sf_mem_handler_debug_%logfile%
66. devenv.exe simplest_ffmpeg_mem_handler.sln /rebuild Release /Out ../sf_mem_handler_release_%logfile%
67. cd ..
68.
69. ::=====
70. echo.
71. echo Compile simplest_ffmpeg_device...
72. cd simplest_ffmpeg_device
73. devenv.exe simplest_ffmpeg_device.sln /rebuild Debug /Out ../sf_device_debug_%logfile%
74. devenv.exe simplest_ffmpeg_device.sln /rebuild Release /Out ../sf_device_release_%logfile%
75. cd ..
76.
77. ::=====
78. echo.
79. echo Compile simplest_ffmpeg_format...
80. cd simplest_ffmpeg_format
81. devenv.exe simplest_ffmpeg_format.sln /rebuild Debug /Out ../sf_format_debug_%logfile%
82. devenv.exe simplest_ffmpeg_format.sln /rebuild Release /Out ../sf_format_release_%logfile%
83. cd ..
84.
85. ::=====
86. echo.
87. echo Compile simplest_ffmpeg_video_filter...
88. cd simplest_ffmpeg_video_filter
89. devenv.exe simplest_ffmpeg_video_filter.sln /rebuild Debug /Out ../sf_video_filter_debug_%logfile%
90. devenv.exe simplest_ffmpeg_video_filter.sln /rebuild Release /Out ../sf_video_filter_release_%logfile%
91. cd ..
92.
93. ::=====
94. echo.
95. echo Compile simplest_ffmpeg_swscale...
96. cd simplest_ffmpeg_swscale
97. devenv.exe simplest_ffmpeg_swscale.sln /rebuild Debug /Out ../sf_swscale_debug_%logfile%
98. devenv.exe simplest_ffmpeg_swscale.sln /rebuild Release /Out ../sf_swscale_release_%logfile%
99. cd ..
100.
101. ::=====

```

CL.exe



cl.exe 是Microsoft C/C++编译器，和GCC属于同一个层面的东西。一个基本的调用cl.exe编译的命令如下所示：

```

[plain]
1. cl.exe helloworld.cpp

```

上述命令执行完后即可在同一目录下生成一个helloworld.exe的可执行程序。

编译包含类库的程序相对来说要复杂一些，下面以Simplest FFmpeg Player为例看一下它的编译脚本。Simplest FFmpeg Player使用CL.exe编译的脚本保存在“compile_cl.bat”文件中，如下所示。

```
[plain]
1.  ::最简单的基于FFmpeg的视频播放器 2----命令行编译
2.  ::Simplest FFmpeg Player 2----Compile in Cmd
3.  ::
4.  ::雷霄骅 Lei Xiaohua
5.  ::leixiaohua1020@126.com
6.  ::中国传媒大学/数字电视技术
7.  ::Communication University of China / Digital TV Technology
8.  ::http://blog.csdn.net/leixiaohua1020
9.  ::
10. ::VS2010 Environment
11. call "D:\Program Files\Microsoft Visual Studio 10.0\VC\vcvarsall.bat"
12. ::include
13. @set INCLUDE=include;%INCLUDE%
14. ::lib
15. @set LIB=lib;%LIB%
16. ::compile and link
17. cl simplest_ffmpeg_player.cpp /MD /link SDL2.lib SDL2main.lib avcodec.lib ^
18. avformat.lib avutil.lib avdevice.lib avfilter.lib postproc.lib swresample.lib swscale.lib ^
19. /SUBSYSTEM:WINDOWS /OPT:NOREF
20. exit
```

这个脚本按照顺序做了以下几点工作：

- (1) 设置VC++运行环境。这一步通过call "D:\Program Files\Microsoft Visual Studio 10.0\VC\vcvarsall.bat"实现。
- (2) 设置include目录。这一步设置FFmpeg以及SDL2的头文件所在的目录，通过修改include环境变量实现（在include环境变量前面加上项目文件夹中的"include"目录）。
- (3) 设置lib目录。这一步设置FFmpeg以及SDL2的库文件所在的目录，通过修改lib环境变量实现（在lib环境变量前面加上项目文件夹中的"lib"目录）。
- (4) 编译和链接。这一步用于将simplest_ffmpeg_player.cpp编译生成simplest_ffmpeg_player.exe。在这里需要注意几点：
 - a) 链接类库使用/link
 - b) 使用SDL类库的时候，务必设置/MD选项（使用动态链接的库）
 - c) 使用SDL类库的时候，务必设置/SUBSYSTEM:WINDOWS
 - d) 使用FFmpeg类库的时候，务必设置/OPT:NOREF

上述脚本运行完后，生成simplest_ffmpeg_player.exe。

完整的脚本simplest_ffmpeg_demos_compile_cl_all.bat如下所示。将脚本拷贝至《最简单的FFmpeg示例程序合集》所在目录运行，就可以编译所有的示例程序。注意这个脚本只是分别调用了各个程序目录下的compile_cl.bat文件。

```
[plain]
1.  @echo off
2.  echo -----
3.  echo simplest_ffmpeg_demos list ---- CL compile all
4.  echo Lei Xiaohua
5.  echo Communication University of China / Digital TV Technology
6.  echo leixiaohua1020@126.com
7.  echo http://blog.csdn.net/leixiaohua1020
8.  echo -----
9.  ::=====
10.
11. echo.
12. echo Compile simplest_ffmpeg_player...
13. cd simplest_ffmpeg_player
14. cd simplest_ffmpeg_player
15. start /wait compile_cl.bat
16. cd ..
17. cd simplest_ffmpeg_player_su
18. start /wait compile_cl.bat
19. cd ..
20. cd simplest_ffmpeg_decoder_pure
21. start /wait compile_cl.bat
22. cd ..
23. cd ..
24.
25. ::=====
26. echo.
27. echo Compile simplest_ffmpeg_audio_player...
28. cd simplest_ffmpeg_audio_player
29. cd simplest_ffmpeg_audio_player
30. start /wait compile_cl.bat
31. cd ..
32. cd ..
33.
34. ::=====
35. echo.
36. echo Compile simplest_ffmpeg_picture_encoder...
37. cd simplest_ffmpeg_picture_encoder
38. cd simplest_ffmpeg_picture_encoder
39. start /wait compile_cl.bat
40. cd ..
41. cd ..
42.
43. ::=====
44. echo.
45. echo Compile simplest_ffmpeg_video_encoder...
```

```

46. cd simplest_ffmpeg_video_encoder
47. cd simplest_ffmpeg_video_encoder
48. start /wait compile_cl.bat
49. cd ..
50. cd simplest_ffmpeg_video_encoder_pure
51. start /wait compile_cl.bat
52. cd ..
53. cd ..
54.
55. ::=====
56. echo.
57. echo Compile simplest_ffmpeg_audio_encoder...
58. cd simplest_ffmpeg_audio_encoder
59. cd simplest_ffmpeg_audio_encoder
60. start /wait compile_cl.bat
61. cd ..
62. cd ..
63.
64. ::=====
65. echo.
66. echo Compile simplest_ffmpeg_streamer...
67. cd simplest_ffmpeg_streamer
68. cd simplest_ffmpeg_streamer
69. start /wait compile_cl.bat
70. cd ..
71. cd ..
72.
73. ::=====
74. echo.
75. echo Compile simplest_ffmpeg_mem_handler...
76. cd simplest_ffmpeg_mem_handler
77. cd simplest_ffmpeg_mem_player
78. start /wait compile_cl.bat
79. cd ..
80. cd simplest_ffmpeg_mem_transcoder
81. start /wait compile_cl.bat
82. cd ..
83. cd ..
84.
85. ::=====
86. echo.
87. echo Compile simplest_ffmpeg_device...
88. cd simplest_ffmpeg_device
89. cd simplest_ffmpeg_grabdesktop
90. start /wait compile_cl.bat
91. cd ..
92. cd simplest_ffmpeg_readcamera
93. start /wait compile_cl.bat
94. cd ..
95. cd ..
96.
97. ::=====
98. echo.
99. echo Compile simplest_ffmpeg_format...
100. cd simplest_ffmpeg_format
101. cd simplest_ffmpeg_demuxer
102. start /wait compile_cl.bat
103. cd ..
104. cd simplest_ffmpeg_demuxer_simple
105. start /wait compile_cl.bat
106. cd ..
107. cd simplest_ffmpeg_muxer
108. start /wait compile_cl.bat
109. cd ..
110. cd simplest_ffmpeg_remuxer
111. start /wait compile_cl.bat
112. cd ..
113. cd ..
114.
115. ::=====
116. echo.
117. echo Compile simplest_ffmpeg_video_filter...
118. cd simplest_ffmpeg_video_filter
119. cd simplest_ffmpeg_video_filter
120. start /wait compile_cl.bat
121. cd ..
122. cd ..
123.
124. ::=====
125. echo.
126. echo Compile simplest_ffmpeg_swscale...
127. cd simplest_ffmpeg_swscale
128. cd simplest_ffmpeg_swscale
129. start /wait compile_cl.bat
130. cd ..
131. cd simplest_pic_gen
132. start /wait compile_cl.bat
133. cd ..
134. cd ..
135.
136. ::=====

```

MinGW



MinGW是Windows下的GCC/G++编译器。使用MinGW编译需要运行其安装目录下的msys.bat设置其环境变量。一个基本的调用MinGW gcc编译的命令如下所示：

```
1. gcc helloworld.cpp -o helloworld.exe
```

上述命令执行完后即可在同一目录下生成一个helloworld.exe的可执行程序。

编译包含类库的程序相对来说要复杂一些，下面以Simplest FFmpeg Player为例看一下它的编译脚本。Simplest FFmpeg Player使用MinGW GCC编译的脚本保存在“compile_mingw.sh”文件中，如下所示。

```
1. #!/bin/sh
2. #最简单的基于FFmpeg的视频播放器 2----MinGW命令行编译
3. #Simplest FFmpeg Player 2----Compile in MinGW
4. #
5. #雷霄骅 Lei Xiaohua
6. #leixiaohua1020@126.com
7. #中国传媒大学/数字电视技术
8. #Communication University of China / Digital TV Technology
9. #http://blog.csdn.net/leixiaohua1020
10. #
11. #compile
12. g++ simplest_ffmpeg_player.cpp -g -o simplest_ffmpeg_player.exe \
13. -I /usr/local/include -L /usr/local/lib \
14. -lmingw32 -lSDL2main -lSDL2 -lavformat -lavcodec -lavutil -lswscale
```

这个脚本使用MinGW中的g++完成编译。在这里要注意一点，并不是gcc只能编译c代码，g++只能编译c++代码，而是gcc和g++都可以编译C和C++代码。源代码文件后缀为.c的，gcc把它当作是C程序，而g++当作是c++程序；后缀为.cpp的，两者都会认为是c++程序。上述编译命令有以下几点需要注意：

- (a) include目录设置为/usr/local/include，要确保SDL2和FFmpeg的头文件都安装在这个目录里（SDL2有可能没有安装在这里）
- (b) lib目录设置为/usr/local/lib，要确保SDL2和FFmpeg的库文件都安装在这个目录里（SDL2有可能没有安装在这里）
- (c) 使用SDL类库的时候，务必链接-lmingw32

上述脚本运行完城后，生成simplest_ffmpeg_player.exe。

完整的脚本simplest_ffmpeg_demos_compile_mingw_all.sh如下所示。将脚本拷贝至《最简单的FFmpeg示例程序合集》所在目录运行，就可以编译所有的示例程序。注意这个脚本只是分别调用了各个程序目录下的compile_mingw.sh文件。

```
1. echo "=====
2. echo "simplest ffmpeg demos list ---- MinGW compile all"
3. echo "Lei Xiaohua"
4. echo "Communication University of China / Digital TV Technology"
5. echo "leixiaohua1020@126.com"
6. echo "http://blog.csdn.net/leixiaohua1020"
7. echo "=====
8. #=====
9.
10. echo ""
11. echo "Compile simplest_ffmpeg_player..."
12. cd simplest_ffmpeg_player
13. cd simplest_ffmpeg_player
14. sh compile_mingw.sh
15. cd ..
16. cd simplest_ffmpeg_player_su
17. sh compile_mingw.sh
18. cd ..
19. cd simplest_ffmpeg_decoder_pure
20. sh compile_mingw.sh
21. cd ..
22. cd ..
23.
24. #=====
25. echo ""
26. echo "Compile simplest_ffmpeg_audio_player..."
27. cd simplest_ffmpeg_audio_player
28. cd simplest_ffmpeg_audio_player
29. sh compile_mingw.sh
30. cd ..
31. cd ..
32.
33. #=====
34. echo ""
```

```
35. echo "Compile simplest_ffmpeg_picture_encoder..."
36. cd simplest_ffmpeg_picture_encoder
37. cd simplest_ffmpeg_picture_encoder
38. sh compile_mingw.sh
39. cd ..
40. cd ..
41.
42. #=====
43. echo ""
44. echo "Compile simplest_ffmpeg_video_encoder..."
45. cd simplest_ffmpeg_video_encoder
46. cd simplest_ffmpeg_video_encoder
47. sh compile_mingw.sh
48. cd ..
49. cd simplest_ffmpeg_video_encoder_pure
50. sh compile_mingw.sh
51. cd ..
52. cd ..
53.
54. #=====
55. echo ""
56. echo "Compile simplest_ffmpeg_audio_encoder..."
57. cd simplest_ffmpeg_audio_encoder
58. cd simplest_ffmpeg_audio_encoder
59. sh compile_mingw.sh
60. cd ..
61. cd ..
62.
63. #=====
64. echo ""
65. echo "Compile simplest_ffmpeg_streamer..."
66. cd simplest_ffmpeg_streamer
67. cd simplest_ffmpeg_streamer
68. sh compile_mingw.sh
69. cd ..
70. cd ..
71.
72. #=====
73. echo ""
74. echo "Compile simplest_ffmpeg_mem_handler..."
75. cd simplest_ffmpeg_mem_handler
76. cd simplest_ffmpeg_mem_player
77. sh compile_mingw.sh
78. cd ..
79. cd simplest_ffmpeg_mem_transcoder
80. sh compile_mingw.sh
81. cd ..
82. cd ..
83.
84. #=====
85. echo ""
86. echo "Compile simplest_ffmpeg_device..."
87. cd simplest_ffmpeg_device
88. cd simplest_ffmpeg_grabdesktop
89. sh compile_mingw.sh
90. cd ..
91. cd simplest_ffmpeg_readcamera
92. sh compile_mingw.sh
93. cd ..
94. cd ..
95.
96. #=====
97. echo ""
98. echo "Compile simplest_ffmpeg_format..."
99. cd simplest_ffmpeg_format
100. cd simplest_ffmpeg_demuxer
101. sh compile_mingw.sh
102. cd ..
103. cd simplest_ffmpeg_demuxer_simple
104. sh compile_mingw.sh
105. cd ..
106. cd simplest_ffmpeg_muxer
107. sh compile_mingw.sh
108. cd ..
109. cd simplest_ffmpeg_remuxer
110. sh compile_mingw.sh
111. cd ..
112. cd ..
113.
114. #=====
115. echo ""
116. echo "Compile simplest_ffmpeg_video_filter..."
117. cd simplest_ffmpeg_video_filter
118. cd simplest_ffmpeg_video_filter
119. sh compile_mingw.sh
120. cd ..
121. cd ..
122.
123. #=====
124. echo ""
125. echo "Compile simplest_ffmpeg_swscale..."
```



```
126. cd simplest_ffmpeg_swscale
127. cd simplest_ffmpeg_swscale
128. sh compile_mingw.sh
129. cd ..
130. cd simplest_pic_gen
131. sh compile_mingw.sh
132. cd ..
133. cd ..
134.
135. #=====
```

GCC (Linux/MacOS)



GCC是Linux/MacOS下的C/C++编译器。在Linux/MacOS下可以使用GCC编译C/C++程序，一个简单的GCC编译命令：

```
1. gcc helloworld.cpp -o helloworld.out
```

上述命令执行完后即可在同一目录下生成一个名称为helloworld.out的可执行程序。

编译包含类库的程序相对来说要复杂一些，下面以Simplest FFmpeg Player为例看一下它的编译脚本。Simplest FFmpeg Player使用GCC编译的脚本保存在“compile_gcc.sh”文件中，如下所示。

```
1. #!/bin/sh
2. #最简单的基于FFmpeg的视频播放器 2----命令行编译
3. #Simplest FFmpeg Player 2----Compile in Shell
4. #
5. #雷霄骅 Lei Xiaohua
6. #leixiaohua1020@126.com
7. #中国传媒大学/数字电视技术
8. #Communication University of China / Digital TV Technology
9. #http://blog.csdn.net/leixiaohua1020
10. #
11. #compile
12. gcc simplest_ffmpeg_player.cpp -g -o simplest_ffmpeg_player.out \
13. -I /usr/local/include -L /usr/local/lib -lSDL2main -lSDL2 -lavformat -lavcodec -lavutil -lswscale
```

这个脚本使用gcc完成编译。上文中已经提到过一次，并不是gcc只能编译c代码，g++只能编译c++代码，而是gcc和g++都可以编译C和C++代码。源代码文件后缀为.c的，gcc把它当作是C程序，而g++当作是c++程序；后缀为.cpp的，两者都会认为是c++程序。上述编译命令有以下几点需要注意：

- (a) include目录设置为/usr/local/include，要确保SDL2和FFmpeg的头文件都安装在这个目录里
- (b) lib目录设置为/usr/local/lib，要确保SDL2和FFmpeg的库文件都安装在这个目录里
- (c) 有些版本中的Linux可能没有安装gcc/g++编译器，需要先装好这两个编译器

上述脚本运行完城后，生成simplest_ffmpeg_player.out。

注意事项：Linux和MacOS的不同

Linux和MacOS在SDL1.2的使用上有很大的不同。在MacOS下使用SDL1.2必须加上“-framework Cocoa”参数，否则编译会出现错误。因此在MacOS下编译使用SDL1.2的程序的时候，编译命令如下所示（以simplest_ffmpeg_grabdesktop为例，该程序采用了SDL1.2）。

```
1. gcc simplest_ffmpeg_grabdesktop.cpp -g -o simplest_ffmpeg_grabdesktop.out \
2. -framework Cocoa -I /usr/local/include -L /usr/local/lib -lSDLmain -lSDL -lavformat -lavcodec -lavutil -lavdevice -lswscale
```

对于这些采用SDL1.2的程序，保存了一个MacOS下专有的脚本“compile_gcc_mac.sh”。

此外，Linux和MacOS在显示上也有较大的不同。此前发现Windows和MacOS下可以正常显示的程序在Ubuntu下却会出现“绿屏”现象。不过随着代码的调整这一情况已经被消除了。

完整的脚本simplest_ffmpeg_demos_compile_gcc_all.sh如下所示。将脚本拷贝至《最简单的FFmpeg示例程序合集》所在目录运行，就可以编译所有的示例程序。注意这个脚本只是分别调用了各个程序目录下的compile_gcc.sh文件。

```
1. echo "=====
2. echo "simplest ffmpeg demos list ---- GCC compile all"
3. echo "Lei Xiaohua"
4. echo "Communication University of China / Digital TV Technology"
5. echo "leixiaohua1020@126.com"
6. echo "http://blog.csdn.net/leixiaohua1020"
7. echo "=====
8. #=====
```

```

9. #05 kernel
10. kernel=$(uname -s)
11.
12. #change the access permissions (--recursive)
13. chmod -R 777 ./
14.
15. echo ""
16. echo "Compile simplest_ffmpeg_player..."
17. cd simplest_ffmpeg_player
18. cd simplest_ffmpeg_player
19. sh compile_gcc.sh
20. cd ..
21. cd simplest_ffmpeg_player_su
22. sh compile_gcc.sh
23. cd ..
24. cd simplest_ffmpeg_decoder_pure
25. sh compile_gcc.sh
26. cd ..
27. cd ..
28.
29. #=====
30. echo ""
31. echo "Compile simplest_ffmpeg_audio_player..."
32. cd simplest_ffmpeg_audio_player
33. cd simplest_ffmpeg_audio_player
34. sh compile_gcc.sh
35. cd ..
36. cd ..
37.
38. #=====
39. echo ""
40. echo "Compile simplest_ffmpeg_picture_encoder..."
41. cd simplest_ffmpeg_picture_encoder
42. cd simplest_ffmpeg_picture_encoder
43. sh compile_gcc.sh
44. cd ..
45. cd ..
46.
47. #=====
48. echo ""
49. echo "Compile simplest_ffmpeg_video_encoder..."
50. cd simplest_ffmpeg_video_encoder
51. cd simplest_ffmpeg_video_encoder
52. sh compile_gcc.sh
53. cd ..
54. cd simplest_ffmpeg_video_encoder_pure
55. sh compile_gcc.sh
56. cd ..
57. cd ..
58.
59. #=====
60. echo ""
61. echo "Compile simplest_ffmpeg_audio_encoder..."
62. cd simplest_ffmpeg_audio_encoder
63. cd simplest_ffmpeg_audio_encoder
64. sh compile_gcc.sh
65. cd ..
66. cd ..
67.
68. #=====
69. echo ""
70. echo "Compile simplest_ffmpeg_streamer..."
71. cd simplest_ffmpeg_streamer
72. cd simplest_ffmpeg_streamer
73. sh compile_gcc.sh
74. cd ..
75. cd ..
76.
77. #=====
78. echo ""
79. echo "Compile simplest_ffmpeg_mem_handler..."
80. cd simplest_ffmpeg_mem_handler
81. cd simplest_ffmpeg_mem_player
82. if [ "$kernel" == "Darwin" ];then
83. sh compile_gcc_mac.sh
84. else
85. sh compile_gcc.sh
86. fi
87. cd ..
88. cd simplest_ffmpeg_mem_transcoder
89. sh compile_gcc.sh
90. cd ..
91. cd ..
92.
93. #=====
94. echo ""
95. echo "Compile simplest_ffmpeg_device..."
96. cd simplest_ffmpeg_device
97. cd simplest_ffmpeg_grabdesktop
98. if [ "$kernel" == "Darwin" ];then
99. sh compile_gcc_mac.sh

```

```
100. else
101. sh compile_gcc.sh
102. fi
103. cd ..
104. cd simplest_ffmpeg_readcamera
105. if [ "$kernel" == "Darwin" ];then
106. sh compile_gcc_mac.sh
107. else
108. sh compile_gcc.sh
109. fi
110. cd ..
111. cd ..
112.
113. #####
114. echo ""
115. echo "Compile simplest_ffmpeg_format..."
116. cd simplest_ffmpeg_format
117. cd simplest_ffmpeg_demuxer
118. sh compile_gcc.sh
119. cd ..
120. cd simplest_ffmpeg_demuxer_simple
121. sh compile_gcc.sh
122. cd ..
123. cd simplest_ffmpeg_muxer
124. sh compile_gcc.sh
125. cd ..
126. cd simplest_ffmpeg_remuxer
127. sh compile_gcc.sh
128. cd ..
129. cd ..
130.
131. #####
132. echo ""
133. echo "Compile simplest_ffmpeg_video_filter..."
134. cd simplest_ffmpeg_video_filter
135. cd simplest_ffmpeg_video_filter
136. if [ "$kernel" == "Darwin" ];then
137. sh compile_gcc_mac.sh
138. else
139. sh compile_gcc.sh
140. fi
141. cd ..
142. cd ..
143.
144. #####
145. echo ""
146. echo "Compile simplest_ffmpeg_swscale..."
147. cd simplest_ffmpeg_swscale
148. cd simplest_ffmpeg_swscale
149. sh compile_gcc.sh
150. cd ..
151. cd simplest_pic_gen
152. sh compile_gcc.sh
153. cd ..
154. cd ..
155.
156. #####
```

版权声明：本文为博主原创文章，未经博主允许不得转载。 <https://blog.csdn.net/leixiaohua1020/article/details/43898125>

文章标签： FFmpeg 编译 gcc Mac linux

个人分类： FFMPEG

所属专栏： FFmpeg

此PDF由spygg生成,请尊重原作者版权!!!

我的邮箱:liushidc@163.com