

最简单的基于FFmpeg的移动端例子：IOS HelloWorld

2015年07月27日 20:18:17 阅读数：29879

=====

最简单的基于FFmpeg的移动端例子系列文章列表：

[最简单的基于FFmpeg的移动端例子：Android HelloWorld](#)

[最简单的基于FFmpeg的移动端例子：Android 视频解码器](#)

[最简单的基于FFmpeg的移动端例子：Android 视频解码器-单个库版](#)

[最简单的基于FFmpeg的移动端例子：Android 推流器](#)

[最简单的基于FFmpeg的移动端例子：Android 视频转码器](#)

[最简单的基于FFmpeg的移动端例子附件：Android 自带播放器](#)

[最简单的基于FFmpeg的移动端例子附件：SDL Android HelloWorld](#)

[最简单的基于FFmpeg的移动端例子：IOS HelloWorld](#)

[最简单的基于FFmpeg的移动端例子：IOS 视频解码器](#)

[最简单的基于FFmpeg的移动端例子：IOS 推流器](#)

[最简单的基于FFmpeg的移动端例子：IOS 视频转码器](#)

[最简单的基于FFmpeg的移动端例子附件：IOS自带播放器](#)

[最简单的基于FFmpeg的移动端例子：Windows Phone HelloWorld](#)

=====

本文记录IOS平台下基于FFmpeg的HelloWorld程序。该示例C语言的源代码来自于《 [最简单的基于FFMPEG的Helloworld程序](#) 》。相关的概念就不再重复记录了。



IOS程序使用FFmpeg类库的说明

IOS应用程序使用FFmpeg类库的流程如下所示。

1. 编译FFmpeg类库

编译IOS的FFmpeg类库需要支持5种架构：armv7、armv7s、arm64、i386、x86_64。其中前面3个是给真机使用的，后面2个是给模拟器使用的。本文记录的FFmpeg类库还支持第三方类库libx264和libfaac，所以在编译之前还要先编译libx264和libfaac的源代码。总体说来，IOS下的类库需要编译成两个版本：thin和fat。每种架构对应一个thin版本的类库，将这些不同架构thin版本的类库合成起来之后，就形成了fat版本的类库。下面简单记录一下编译步骤。编译过程中IOS SDK版本为8.3，FFmpeg版本为2.7.1，faac和x264分别使用了最新版本的源代码。

(1) 第三方库libx264的编译

这一步用于生成支持armv7、armv7s、arm64、i386、x86_64几种架构的fat版本的libx264.a。下面这个脚本可以首先编译生成上面5种架构的thin版本的libx264.a，分成5个文件夹存储于thin-x264文件夹中；然后将这些类库合并成为1个fat版本的libx264.a，存储于fat-x264文件夹中。

build_x264.sh

[plain]  

```

1.  #!/bin/sh
2.  # LXH,MXY
3.  #
4.  # directories
5.  SOURCE="x264"
6.  FAT="fat-x264"
7.
8.  SCRATCH="scratch-x264"
9.  # must be an absolute path
10. THIN=`pwd`/"thin-x264"
11.
12. #This is decided by your SDK version.
13. SDK_VERSION="8.3"
14.
15. cd ./x264
16.
17. #===== simulator =====
18. PLATFORM="iPhoneSimulator"
19.
20. #i386
21. ARCHS="i386"
22.
23. export DEVROOT=/Applications/Xcode.app/Contents/Developer/Platforms/${PLATFORM}.platform/Developer
24. export SDKROOT=$DEVROOT/SDKs/${PLATFORM}${SDK_VERSION}.sdk
25. export CC=$DEVROOT/usr/bin/gcc
26. export LD=$DEVROOT/usr/bin/ld
27. export CXX=$DEVROOT/usr/bin/g++
28. export LIBTOOL=$DEVROOT/usr/bin/libtool
29. export HOST=i386-apple-darwin
30.
31. COMMONFLAGS="-pipe -gdwarf-2 -no-cpp-precomp -isysroot ${SDKROOT} -fPIC"
32. export LDFLAGS="${COMMONFLAGS} -fPIC"
33. export CFLAGS="${COMMONFLAGS} -fvisibility=hidden"
34.
35.
36. for ARCH in $ARCHS; do
37.
38. echo "Building $ARCH ....."
39.
40. make clean
41. ./configure \
42. --host=i386-apple-darwin \
43. --sysroot=$SDKROOT \
44. --prefix="$THIN/$ARCH" \
45. --extra-cflags="-arch $ARCH -miphoneos-version-min=6.0" \
46. --extra-ldflags="-L$SDKROOT/usr/lib/system -arch $ARCH -miphoneos-version-min=6.0" \
47. --enable-pic \
48. --enable-static \
49. --disable-asm \
50. make && make install && make clean
51.
52. echo "Installed: $DEST/$ARCH"
53. done
54.
55. #x86_64
56.
57. ARCHS="x86_64"
58.
59. unset DEVROOT
60. unset SDKROOT
61. unset CC
62. unset LD
63. unset CXX
64. unset LIBTOOL
65. unset HOST
66. unset LDFLAGS
67. unset CFLAGS
68.
69. make clean
70. for ARCH in $ARCHS; do
71.
72. echo "Building $ARCH ....."
73.
74. ./configure \
75. --prefix="$THIN/$ARCH" \
76. --enable-pic \
77. --enable-static \
78. --disable-asm \
79. make && make install && make clean
80.
81. echo "Installed: $DEST/$ARCH"
82. done
83.
84. #===== iphone =====
85.
86. export PLATFORM="iPhoneOS"
87.
88. ARCHS="arm64 armv7 armv7s "
89.
90. export DEVROOT=/Applications/Xcode.app/Contents/Developer
91. export SDKROOT=$DEVROOT/Platforms/${PLATFORM}.platform/Developer/SDKs/${PLATFORM}${SDK_VERSION}.sdk

```

```

92. #DEVPATH=/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS${SDK_VERSION}.sdk
93. export CC=$DEVROOT/Toolchains/XcodeDefault.xctoolchain/usr/bin/clang
94. export AS=$DEVROOT/Toolchains/XcodeDefault.xctoolchain/usr/bin/as
95. COMMONFLAGS="-pipe -gdwarf-2 -no-cpp-precomp -isysroot ${SDKROOT} -marm -fPIC"
96. export LDFLAGS="${COMMONFLAGS} -fPIC"
97. export CFLAGS="${COMMONFLAGS} -fvisibility=hidden"
98. export CXXFLAGS="${COMMONFLAGS} -fvisibility=hidden -fvisibility-inlines-hidden"
99.
100.
101. for ARCH in $ARCHS; do
102.
103.     echo "Building $ARCH ....."
104.
105.     ./configure \
106.     --host=arm-apple-darwin \
107.     --sysroot=$DEVPATH \
108.     --prefix="$THIN/$ARCH" \
109.     --extra-cflags="-arch $ARCH" \
110.     --extra-ldflags="-L$DEVPATH/usr/lib/system -arch $ARCH" \
111.     --enable-pic \
112.     --enable-static \
113.     --disable-asm
114.
115.     make && make install && make clean
116.
117.     echo "Installed: $DEST/$ARCH"
118.
119.     done
120.
121.     cd ..
122.
123.     #===== fat lib =====
124.
125.     ARCHS="armv7 armv7s i386 x86_64 arm64"
126.
127.     echo "building fat binaries..."
128.     mkdir -p $FAT/lib
129.     set - $ARCHS
130.     CWD=`pwd`
131.     cd $THIN/$1/lib
132.     for LIB in *.a
133.     do
134.         cd $CWD
135.         lipo -create `find $THIN -name $LIB` -output $FAT/lib/$LIB
136.     done
137.
138.     cd $CWD
139.     cp -rf $THIN/$1/include $FAT

```

(2) 第三方库libfaac的编译

这一步用于生成支持armv7、armv7s、arm64、i386、x86_64几种架构的fat版本的libfaac.a。下面这个脚本可以首先编译生成上面5种架构的thin版本的libfaac.a，分成5个文件夹存储于fat-faac中；然后将这些类库合并成为1个fat版本的libfaac.a，存储于fat-faac中。

build_faac.sh

```

[plain]
1. #!/bin/sh
2. cd ./faac
3. make distclean
4. cd ..
5.
6. CONFIGURE_FLAGS="--enable-static --with-pic"
7.
8. ARCHS="arm64 armv7s x86_64 i386 armv7"
9.
10. # directories
11. SOURCE="faac"
12. FAT="fat-faac"
13.
14. SCRATCH="scratch-faac"
15. # must be an absolute path
16. THIN=`pwd`/"thin-faac"
17.
18. COMPILE="y"
19. LIPO="y"
20.
21. if [ "$*" ]
22. then
23.     if [ "$*" = "lipo" ]
24.     then
25.         # skip compile
26.         COMPILE=
27.     else
28.         ARCHS="$*"
29.         if [ $# -eq 1 ]
30.         then

```

```

31. # skip lipo
32. LIPO=
33. fi
34. fi
35. fi
36.
37. if [ "$COMPILE" ]
38. then
39. CWD=`pwd`
40. for ARCH in $ARCHS
41. do
42. echo "building $ARCH..."
43. mkdir -p "$SCRATCH/$ARCH"
44. cd "$SCRATCH/$ARCH"
45.
46. if [ "$ARCH" = "i386" -o "$ARCH" = "x86_64" ]
47. then
48. PLATFORM="iPhoneSimulator"
49. CPU=
50. if [ "$ARCH" = "x86_64" ]
51. then
52. SIMULATOR="-mios-simulator-version-min=7.0"
53. HOST=
54. else
55. SIMULATOR="-mios-simulator-version-min=5.0"
56. HOST="--host=i386-apple-darwin"
57. fi
58. else
59. PLATFORM="iPhoneOS"
60. if [ $ARCH = "armv7s" ]
61. then
62. CPU="--cpu=swift"
63. else
64. CPU=
65. fi
66. SIMULATOR=
67. HOST="--host=arm-apple-darwin"
68. fi
69.
70. XCRUN_SDK=`echo $PLATFORM | tr '[:upper:]' '[:lower:]'`
71. CC="xcrun -sdk $XCRUN_SDK clang -Wno-error=unused-command-line-argument-hard-error-in-future"
72. AS="/usr/local/bin/gas-preprocessor.pl $CC"
73. CFLAGS="-arch $ARCH $SIMULATOR"
74. CXXFLAGS="$CFLAGS"
75. LDFLAGS="$CFLAGS"
76.
77. CC=$CC CFLAGS=$CXXFLAGS LDFLAGS=$LDFLAGS CPPFLAGS=$CXXFLAGS CXX=$CC CXXFLAGS=$CXXFLAGS $CWD/$SOURCE/configure \
78. $CONFIGURE_FLAGS \
79. $HOST \
80. --prefix="$THIN/$ARCH" \
81. --disable-shared \
82. --without-mp4v2
83.
84. make clean && make && make install-strip
85. cd $CWD
86. done
87. fi
88.
89. #===== fat lib =====
90.
91. echo "building fat binaries..."
92. mkdir -p $FAT/lib
93. set - $ARCHS
94. CWD=`pwd`
95. cd $THIN/$1/lib
96. for LIB in *.a
97. do
98. cd $CWD
99. lipo -create `find $THIN -name $LIB` -output $FAT/lib/$LIB
100. done
101.
102. cd $CWD
103. cp -rf $THIN/$1/include $FAT

```

(3) 编译armv7版本FFmpeg类库

这一步用于生成支持armv7架构的thin版本的FFmpeg类库，存储于thin-ffmpeg/armv7文件夹中。脚本如下所示。

build_ffmpeg_demo_armv7.sh

```

[plain]
1.  #!/bin/sh
2.  # LXH,MXY modified
3.
4.  cd ffmpeg
5.
6.  PLATFORM="iPhoneOS"
7.  INSTALL="thin-ffmpeg"
8.  SDK_VERSION="8.3"
9.
10. # libx264
11. export X264ROOT=../thin-x264/armv7
12. export X264LIB=$X264ROOT/lib
13. export X264INCLUDE=$X264ROOT/include
14. # libfaac
15. export FAACROOT=../thin-faac/armv7
16. export FAACLIB=$FAACROOT/lib
17. export FAACINCLUDE=$FAACROOT/include
18.
19. export DEVR00T=/Applications/Xcode.app/Contents/Developer
20.
21. export SDKROOT=$DEVR00T/Platforms/${PLATFORM}.platform/Developer/SDKs/${PLATFORM}/${SDK_VERSION}.sdk
22. export CC=$DEVR00T/Toolchains/XcodeDefault.xctoolchain/usr/bin/clang
23. export AS=$DEVR00T/Toolchains/XcodeDefault.xctoolchain/usr/bin/as
24.
25. COMMONFLAGS="-pipe -gdwarf-2 -no-cpp-precomp -isysroot ${SDKROOT} -marm -fPIC"
26. export LDFLAGS="${COMMONFLAGS} -fPIC"
27. export CFLAGS="${COMMONFLAGS} -fvisibility=hidden"
28. export CXXFLAGS="${COMMONFLAGS} -fvisibility=hidden -fvisibility-inlines-hidden"
29.
30.
31. echo "Building armv7..."
32.
33. make clean
34. ./configure \
35.     --cpu=cortex-a9 \
36.     --extra-cflags='-I$X264INCLUDE -I$FAACINCLUDE -arch armv7 -miphoneos-version-min=5.0 -mthumb' \
37.     --extra-ldflags='-L$X264LIB -L$FAACLIB -arch armv7 -miphoneos-version-min=5.0' \
38.     --enable-cross-compile \
39.     --arch=arm --disable-iconv \
40.     --target-os=darwin \
41.     --cc=${CC} --disable-asm \
42.     --sysroot=${SDKROOT} \
43.     --prefix=../${INSTALL}/armv7 \
44.     --enable-gpl --enable-nonfree --enable-version3 --disable-bzlib --enable-small --disable-vda \
45.     --disable-encoders --enable-libx264 --enable-libfaac --enable-encoder=libx264 --enable-encoder=libfaac \
46.     --disable-muxers --enable-muxer=flv --enable-muxer=mov --enable-muxer=ipod --enable-muxer=mpegts --enable-muxer=psp --enable-muxer=mp4 --enable-muxer=avi \
47.     --disable-decoders --enable-decoder=aac --enable-decoder=aac_latm --enable-decoder=h264 --enable-decoder=mpeg4 \
48.     --disable-demuxers --enable-demuxer=flv --enable-demuxer=h264 --enable-demuxer=mpegts --enable-demuxer=avi --enable-demuxer=mpc --enable-demuxer=mov \
49.     --disable-parsers --enable-parser=aac --enable-parser=ac3 --enable-parser=h264 \
50.     --disable-protocols --enable-protocol=file --enable-protocol=rtmp --enable-protocol=rtp --enable-protocol=udp \
51.     --disable-bsfs --enable-bsf=aac_adtstoasc --enable-bsf=h264_mp4toannexb \
52.     --disable-devices --disable-debug --disable-ffmpeg --disable-ffprobe --disable-ffplay --disable-ffserver --disable-debug
53.
54. make
55. make install
56.
57. cd ..

```

(4) 编译armv7s版本FFmpeg类库

这一步用于生成支持armv7s架构的thin版本的FFmpeg类库，存储于thin-ffmpeg/armv7s文件夹中。脚本如下所示。

build_ffmpeg_demo_armv7s.sh

```

[plain]
1.  #!/bin/sh
2.  # LXH,MXY modified
3.
4.  cd ffmpeg
5.
6.  PLATFORM="iPhoneOS"
7.  INSTALL="thin-ffmpeg"
8.  SDK_VERSION="8.3"
9.
10. # libx264
11. export X264ROOT=../thin-x264/armv7s
12. export X264LIB=$X264ROOT/lib
13. export X264INCLUDE=$X264ROOT/include
14. # libfaac
15. export FAACROOT=../thin-faac/armv7s
16. export FAACLIB=$FAACROOT/lib
17. export FAACINCLUDE=$FAACROOT/include
18.
19. export DEVROOT=/Applications/Xcode.app/Contents/Developer
20.
21. export SDKROOT=$DEVROOT/Platforms/${PLATFORM}.platform/Developer/SDKs/${PLATFORM}/${SDK_VERSION}.sdk
22. export CC=$DEVROOT/Toolchains/XcodeDefault.xctoolchain/usr/bin/clang
23. export AS=$DEVROOT/Toolchains/XcodeDefault.xctoolchain/usr/bin/as
24.
25. COMMONFLAGS="-pipe -gdwarf-2 -no-cpp-precomp -isysroot ${SDKROOT} -marm -fPIC"
26. export LDFLAGS="${COMMONFLAGS} -fPIC"
27. export CFLAGS="${COMMONFLAGS} -fvisibility=hidden"
28. export CXXFLAGS="${COMMONFLAGS} -fvisibility=hidden -fvisibility-inlines-hidden"
29.
30.
31. echo "Building armv7s..."
32.
33. make clean
34. ./configure \
35.     --cpu=cortex-a9 \
36.     --extra-cflags="-I$X264INCLUDE -I$FAACINCLUDE -arch armv7s -miphoneos-version-min=5.0 -mthumb" \
37.     --extra-ldflags="-L$X264LIB -L$FAACLIB -arch armv7s -miphoneos-version-min=5.0" \
38.     --enable-cross-compile \
39.     --arch=arm --disable-iconv \
40.     --target-os=darwin \
41.     --cc=${CC} --disable-asm \
42.     --sysroot=${SDKROOT} \
43.     --prefix=../${INSTALL}/armv7s \
44.     --enable-gpl --enable-nonfree --enable-version3 --disable-bzlib --enable-small --disable-vda \
45.     --disable-encoders --enable-libx264 --enable-libfaac --enable-encoder=libx264 --enable-encoder=libfaac \
46.     --disable-muxers --enable-muxer=flv --enable-muxer=mov --enable-muxer=ipod --enable-muxer=mpegts --enable-muxer=psp --enable-muxer=
mp4 --enable-muxer=avi \
47.     --disable-decoders --enable-decoder=aac --enable-decoder=aac_latm --enable-decoder=h264 --enable-decoder=mpeg4 \
48.     --disable-demuxers --enable-demuxer=flv --enable-demuxer=h264 --enable-demuxer=avi --enable-demuxer=mpegts --enable-demuxer=mpc --e
nable-demuxer=mov \
49.     --disable-parsers --enable-parser=aac --enable-parser=ac3 --enable-parser=h264 \
50.     --disable-protocols --enable-protocol=file --enable-protocol=rtmp --enable-protocol=rtp --enable-protocol=udp \
51.     --disable-bsfs --enable-bsf=aac_adtstoasc --enable-bsf=h264_mp4toannexb \
52.     --disable-devices --disable-debug --disable-ffmpeg --disable-ffprobe --disable-ffplay --disable-ffserver --disable-debug
53.
54. make
55. make install
56.
57. cd ..

```

(5) 编译arm64版本FFmpeg类库

这一步用于生成支持arm64架构的thin版本的FFmpeg类库，存储于thin-ffmpeg/arm64文件夹中。脚本如下所示。

build_ffmpeg_demo_arm64.sh

```

[plain]
1.  #!/bin/sh
2.  # LXH,MXY modified
3.
4.  cd ffmpeg
5.
6.  PLATFORM="iPhoneOS"
7.  INSTALL="thin-ffmpeg"
8.  SDK_VERSION="8.3"
9.
10. # libx264
11. export X264ROOT=../thin-x264/arm64
12. export X264LIB=$X264ROOT/lib
13. export X264INCLUDE=$X264ROOT/include
14. # libfaac
15. export FAACROOT=../thin-faac/arm64
16. export FAACLIB=$FAACROOT/lib
17. export FAACINCLUDE=$FAACROOT/include
18.
19. export DEVROOT=/Applications/Xcode.app/Contents/Developer
20.
21. export SDKROOT=$DEVROOT/Platforms/${PLATFORM}.platform/Developer/SDKs/${PLATFORM}/${SDK_VERSION}.sdk
22. export CC=$DEVROOT/Toolchains/XcodeDefault.xctoolchain/usr/bin/clang
23. export AS=$DEVROOT/Toolchains/XcodeDefault.xctoolchain/usr/bin/as
24.
25. COMMONFLAGS="-pipe -gdwarf-2 -no-cpp-precomp -isysroot ${SDKROOT} -marm -fPIC"
26. export LDFLAGS="${COMMONFLAGS} -fPIC"
27. export CFLAGS="${COMMONFLAGS} -fvisibility=hidden"
28. export CXXFLAGS="${COMMONFLAGS} -fvisibility=hidden -fvisibility-inlines-hidden"
29.
30.
31. echo "Building arm64..."
32.
33. make clean
34. ./configure \
35.     --extra-cflags='-I$X264INCLUDE -I$FAACINCLUDE -arch arm64 -miphoneos-version-min=5.0 -mthumb' \
36.     --extra-ldflags='-L$X264LIB -L$FAACLIB -arch arm64 -miphoneos-version-min=5.0' \
37.     --enable-cross-compile \
38.     --arch=arm --disable-iconv \
39.     --target-os=darwin \
40.     --cc=${CC} --disable-asm \
41.     --sysroot=${SDKROOT} \
42.     --prefix=../${INSTALL}/arm64 \
43.     --enable-gpl --enable-nonfree --enable-version3 --disable-bzlib --enable-small --disable-vda \
44.     --disable-encoders --enable-libx264 --enable-libfaac --enable-encoder=libx264 --enable-encoder=libfaac \
45.     --disable-muxers --enable-muxer=flv --enable-muxer=mov --enable-muxer=ipod --enable-muxer=mpegts --enable-muxer=psp --enable-muxer=m
46.     p4 --enable-muxer=avi \
47.     --disable-decoders --enable-decoder=aac --enable-decoder=aac_latm --enable-decoder=h264 --enable-decoder=mpeg4 \
48.     --disable-demuxers --enable-demuxer=flv --enable-demuxer=h264 --enable-demuxer=avi --enable-demuxer=mpegts --enable-demuxer=mpc --en
49.     able-demuxer=mov \
50.     --disable-parsers --enable-parser=aac --enable-parser=ac3 --enable-parser=h264 \
51.     --disable-protocols --enable-protocol=file --enable-protocol=rtmp --enable-protocol=rtp --enable-protocol=udp \
52.     --disable-bsfs --enable-bsf=aac_adtstoasc --enable-bsf=h264_mp4toannexb \
53.     --disable-devices --disable-debug --disable-ffmpeg --disable-ffprobe --disable-ffplay --disable-ffserver --disable-debug
54.
55. make
56. make install
57.
58. cd ..

```

(6) 编译i386版本FFmpeg类库

这一步用于生成支持i386架构的thin版本的FFmpeg类库，存储于thin-ffmpeg/i386文件夹中。脚本如下所示。



build_ffmpeg_demo_i386.sh

```
[plain]  
1.  #!/bin/sh
2.  # LXH,MYX modified
3.
4.  cd ffmpeg
5.
6.  PLATFORM="iPhoneSimulator"
7.  INSTALL="thin-ffmpeg"
8.  SDK_VERSION="8.3"
9.
10. # libx264
11. export X264ROOT=../thin-x264/i386
12. export X264LIB=$X264ROOT/lib
13. export X264INCLUDE=$X264ROOT/include
14. # libfaac
15. export FAACROOT=../thin-faac/i386
16. export FAACLIB=$FAACROOT/lib
17. export FAACINCLUDE=$FAACROOT/include
18.
19. export DEVR00T=/Applications/Xcode.app/Contents/Developer/Platforms/${PLATFORM}.platform/Developer
20. export SDKROOT=$DEVR00T/SDKs/${PLATFORM}/${SDK_VERSION}.sdk
21. export CC=$DEVR00T/usr/bin/gcc
22. export LD=$DEVR00T/usr/bin/ld
23.
24. export CXX=$DEVR00T/usr/bin/g++
25.
26. export LIBTOOL=$DEVR00T/usr/bin/libtool
27.
28. COMMONFLAGS="-pipe -gdwarf-2 -no-cpp-precomp -isysroot ${SDKROOT} -fPIC"
29. export LDFLAGS="${COMMONFLAGS} -fPIC"
30. export CFLAGS="${COMMONFLAGS} -fvisibility=hidden"
31.
32.
33. echo "Building i386..."
34. make clean
35. ./configure \
36.     --cpu=i386 \
37.     --extra-cflags='-I$X264INCLUDE -I$FAACINCLUDE -arch i386 -miphoneos-version-min=5.0' \
38.     --extra-ldflags='-L$X264LIB -L$FAACLIB -arch i386 -miphoneos-version-min=5.0' \
39.     --enable-cross-compile \
40.     --arch=i386 --disable-iconv \
41.     --target-os=darwin \
42.     --cc=${CC} \
43.     --sysroot=${SDKROOT} \
44.     --prefix=../${INSTALL}/i386 \
45.     --enable-gpl --enable-nonfree --enable-version3 --disable-bzlib --enable-small --disable-vda \
46.     --disable-encoders --enable-libx264 --enable-libfaac --enable-encoder=libx264 --enable-encoder=libfaac \
47.     --disable-muxers --enable-muxer=flv --enable-muxer=mov --enable-muxer=mpegts --enable-muxer=ipod --enable-muxer=psp --enable-muxer=m
p4 --enable-muxer=avi \
48.     --disable-decoders --enable-decoder=aac --enable-decoder=aac_latm --enable-decoder=h264 --enable-decoder=mpeg4 \
49.     --disable-demuxers --enable-demuxer=flv --enable-demuxer=h264 --enable-demuxer=mpegts --enable-demuxer=avi --enable-demuxer=mpc --en
able-demuxer=mov \
50.     --disable-parsers --enable-parser=aac --enable-parser=ac3 --enable-parser=h264 \
51.     --disable-protocols --enable-protocol=file --enable-protocol=rtmp --enable-protocol=rtp --enable-protocol=udp \
52.     --disable-bsfs --enable-bsf=aac_adtstoasc --enable-bsf=h264_mp4toannexb \
53.     --disable-devices --disable-debug --disable-ffmpeg --disable-ffprobe --disable-ffplay --disable-ffserver --disable-debug
54.
55. make
56. make install
57.
58. cd ..
```

(7) 编译x86_64版本FFmpeg类库

这一步用于生成支持x86_64架构的thin版本的FFmpeg类库，存储于thin-ffmpeg/x86_64文件夹中。脚本如下所示。

build_ffmpeg_demo_x86_64.sh


```
[plain]  
1.  #!/bin/sh
2.  # LXH,MXY modified
3.
4.  cd ./ffmpeg
5.
6.  INSTALL="thin-ffmpeg"
7.
8.  # libx264
9.  export X264ROOT=../thin-x264/x86_64
10. export X264LIB=$X264ROOT/lib
11. export X264INCLUDE=$X264ROOT/include
12. # libfaac
13. export FAACROOT=../thin-faac/x86_64
14. export FAACLIB=$FAACROOT/lib
15. export FAACINCLUDE=$FAACROOT/include
16.
17. unset DEVR00T
18. unset SDKR00T
19. unset CC
20. unset LD
21. unset CXX
22. unset LIBTOOL
23. unset HOST
24. unset LDFLAGS
25. unset CFLAGS
26.
27. echo "Building x86_64..."
28.
29. make clean
30. ./configure \
31.     --extra-cflags='-I$X264INCLUDE -I$FAACINCLUDE' \
32.     --extra-ldflags='-L$X264LIB -L$FAACLIB' \
33.     --disable-iconv \
34.     --disable-asm \
35.     --prefix=../${INSTALL}/x86_64 \
36.     --enable-gpl --enable-nonfree --enable-version3 --disable-bzlib --enable-small --disable-vda \
37.     --disable-encoders --enable-libx264 --enable-libfaac --enable-encoder=libx264 --enable-encoder=mpeg2video --enable-encoder=libfaac \
38.     --disable-muxers --enable-muxer=flv --enable-muxer=mov --enable-muxer=ipod --enable-muxer=mpegts --enable-muxer=psp --enable-muxer=m
39.     p4 --enable-muxer=avi \
40.     --disable-decoders --enable-decoder=aac --enable-decoder=mpeg2video --enable-decoder=aac_latm --enable-decoder=h264 --enable-decoder
41.     =mpeg4 \
42.     --disable-demuxers --enable-demuxer=flv --enable-demuxer=h264 --enable-demuxer=avi --enable-demuxer=mpegts --enable-demuxer=mpc --en
43.     able-demuxer=mov \
44.     --disable-parsers --enable-parser=aac --enable-parser=ac3 --enable-parser=h264 \
45.     --disable-protocols --enable-protocol=file --enable-protocol=rtmp --enable-protocol=rtp --enable-protocol=udp \
46.     --enable-bsf=aac_adtstoasc --enable-bsf=h264_mp4toannexb \
47.     --disable-devices --disable-debug --disable-ffmpeg --disable-ffprobe --disable-ffplay --disable-ffserver --disable-debug
48.
49. make
50. make install
51. cd ..
```

(8) 打包类库

这一步用于将上述步骤中生成的5个版本的FFmpeg打包生成fat版本的FFmpeg类库。这一步骤执行完毕后，将thin-ffmpeg中几个thin版本的类库合并为一个fat版本的类库，并存储于fat-ffmpeg文件夹中。脚本如下所示。

build_ffmpeg_fat.sh

```
[plain]
1.  #!/bin/sh
2.
3.
4.  # directories
5.  THIN=`pwd`/"thin-ffmpeg"
6.  FAT=`pwd`/"fat-ffmpeg"
7.  CWD=`pwd`
8.  # must be an absolute path
9.
10.
11.  ARCHS="armv7s i386 armv7 arm64 x86_64"
12.
13.
14.  echo "building fat binaries..."
15.
16.  mkdir -p $FAT/lib
17.  set - $ARCHS
18.  cd thin-ffmpeg/$1/lib
19.  for LIB in *.a
20.  do
21.  cd $CWD
22.  lipo -create `find $THIN -name $LIB` -output $FAT/lib/$LIB
23.  done
24.
25.  cd $CWD
26.  cp -rf $THIN/$1/include $FAT
```

生成完fat版本的类库后，可以在命令行使用lipo命令查看类库的架构，如下所示。

```
[plain]
1.  lipo -info libavcodec.a
```

2. 编写iOS程序

编写包含FFmpeg类库支持的iOS程序分成两步：配置Xcode环境，编写C语言代码。

(1) 配置Xcode环境

下面以Xcode的iOS中的Single View Application为例，记录一下配置步骤：

(a)

拷贝头文件所在的include文件夹和fat版本的FFmpeg类库（包括libavformat.a, libavcodec.a, libavutil.a, libavdevice.a, libavfilter.a, libpostproc.a, libswresample.a, libswscale.a；以及第三方fat版本类库libx264.a, libfaac.a）至项目文件夹。并将它们添加至项目中。

(b)

项目属性->Build Settings中配置以下3个选项。
Linking->Other Linker Flags中添加下面内容：

```
[plain]
1.  -lavformat
2.  -lavcodec
3.  -lavutil
4.  -lavdevice
5.  -lavfilter
6.  -lpostproc
7.  -lswresample
8.  -lswscale
9.  -lx264
10. -lfaac
```

Search Paths->Header Search Paths添加下面内容

```
[plain]
1.  $(PROJECT_DIR)/include
```

Search Paths->Library Search Paths添加下面内容

```
[plain]
1.  $(PROJECT_DIR)
```

其它的一些配置。这些配置随着FFmpeg版本的不同而有略微的不同（在某些情况下也可能不需要配置）。我目前使用的2.7.1版本的FFmpeg需要配置下面的选项。
项目属性->General->Linked Frameworks and Libraries中添加两个类库：AVFoundation.framework和libz.dylib。

(2) 编写C语言代码

做好上面配置后，就可以在项目中编写代码测试一下FFmpeg是否正确配置了。由于iOS使用的Objective-C是兼容C语言的，所以可以直接写C语言代码调用FFmpeg。

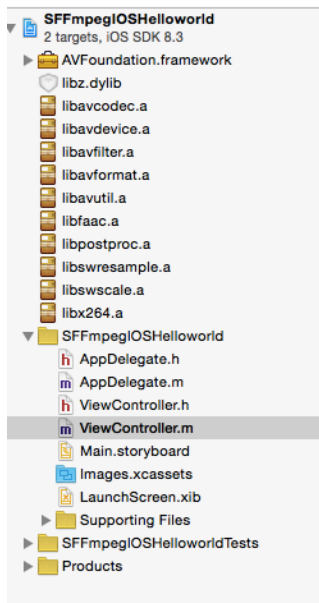
可以在ViewController.m中的viewDidLoad()函数中添加一行printf()代码打印FFmpeg类库的版本信息，如下所示。

```
[objc] ⓘ ⓘ  
1.  #import "ViewController.h"  
2.  #include <libavcodec/avcodec.h>  
3.  
4.  @interface ViewController ()  
5.  
6.  @end  
7.  
8.  @implementation ViewController  
9.  
10. - (void)viewDidLoad {  
11.     [super viewDidLoad];  
12.     // Do any additional setup after loading the view, typically from a nib.  
13.     printf("%s",avcodec_configuration());  
14. }  
15.  
16. - (void)didReceiveMemoryWarning {  
17.     [super didReceiveMemoryWarning];  
18.     // Dispose of any resources that can be recreated.  
19. }  
20.  
21. @end
```

如果类库编译无误，启动iOS程序的时候会在控制台打印版本信息。

源代码

项目的目录结构如图所示。



C代码位于ViewController.m文件中，内容如下所示。

```
[objc] ⓘ ⓘ  
1.  /**  
2.   * 最简单的基于FFmpeg的HelloWorld程序 — IOS  
3.   *  Simplest FFmpeg HelloWorld — IOS  
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.  *  本程序可以获得FFmpeg类库相关的信息。  
12.  *  This software can get information about FFmpeg libraries.  
13.  *  
14.  */  
15.  
16.  #import "ViewController.h"  
17.  #include <libavcodec/avcodec.h>  
18.  #include <libavformat/avformat.h>  
19.  #include <libavfilter/avfilter.h>  
20.  
21.  @interface ViewController ()  
22.  
23.  @end
```

```

23. @implementation ViewController
24.
25.
26.
27. - (void)viewDidLoad {
28.     [super viewDidLoad];
29.     // Do any additional setup after loading the view, typically from a nib.
30.
31.     av_register_all();
32.     char info[10000] = { 0 };
33.     printf("%s\n", avcodec_configuration());
34.     sprintf(info, "%s\n", avcodec_configuration());
35.     NSString * info_ns = [NSString stringWithFormat:@"%s", info];
36.     self.content.text=info_ns;
37. }
38.
39.
40. - (void)didReceiveMemoryWarning {
41.     [super didReceiveMemoryWarning];
42.     // Dispose of any resources that can be recreated.
43. }
44.
45. - (IBAction)clickProtocolButton:(id)sender {
46.     //Alert
47.     /*
48.     UIAlertView *alter = [[UIAlertView alloc] initWithTitle:@"Title" message:@"This is content" delegate:nil cancelButtonTitle:@"Close" otherButtonTitles:nil];
49.     [alter show];
50.     */
51.     char info[40000]={0};
52.     av_register_all();
53.
54.     struct URLProtocol *pup = NULL;
55.     //Input
56.     struct URLProtocol **p_temp = &pup;
57.     avio_enum_protocols((voidvoid **)p_temp, 0);
58.     while ((*p_temp) != NULL){
59.         sprintf(info, "%s[In ][%10s]\n", info, avio_enum_protocols((voidvoid **)p_temp, 0));
60.     }
61.     pup = NULL;
62.     //Output
63.     avio_enum_protocols((voidvoid **)p_temp, 1);
64.     while ((*p_temp) != NULL){
65.         sprintf(info, "%s[Out][%10s]\n", info, avio_enum_protocols((voidvoid **)p_temp, 1));
66.     }
67.     //printf("%s", info);
68.     NSString * info_ns = [NSString stringWithFormat:@"%s", info];
69.     self.content.text=info_ns;
70. }
71.
72. - (IBAction)clickAVFormatButton:(id)sender {
73.     char info[40000] = { 0 };
74.
75.     av_register_all();
76.
77.     AVInputFormat *if_temp = av_iformat_next(NULL);
78.     AVOutputFormat *of_temp = av_oformat_next(NULL);
79.     //Input
80.     while(if_temp!=NULL){
81.         sprintf(info, "%s[In ][%10s]\n", info, if_temp->name);
82.         if_temp=if_temp->next;
83.     }
84.     //Output
85.     while (of_temp != NULL){
86.         sprintf(info, "%s[Out][%10s]\n", info, of_temp->name);
87.         of_temp = of_temp->next;
88.     }
89.     //printf("%s", info);
90.     NSString * info_ns = [NSString stringWithFormat:@"%s", info];
91.     self.content.text=info_ns;
92. }
93.
94.
95. - (IBAction)clickAVCodecButton:(id)sender {
96.
97.     char info[40000] = { 0 };
98.
99.     av_register_all();
100.
101.     AVCodec *c_temp = av_codec_next(NULL);
102.
103.     while(c_temp!=NULL){
104.         if (c_temp->decode!=NULL){
105.             sprintf(info, "%s[Dec]", info);
106.         }
107.         else{
108.             sprintf(info, "%s[Enc]", info);
109.         }
110.         switch (c_temp->type){
111.             case AVMEDIA_TYPE_VIDEO:
112.                 sprintf(info, "%s[Video]", info);
113.                 break:

```

```

114.         case AVMEDIA_TYPE_AUDIO:
115.             sprintf(info, "%s[Audio]", info);
116.             break;
117.         default:
118.             sprintf(info, "%s[Other]", info);
119.             break;
120.     }
121.     sprintf(info, "%s%10s\n", info, c_temp->name);
122.
123.
124.     c_temp=c_temp->next;
125. }
126. //printf("%s", info);
127. NSString * info_ns = [NSString stringWithFormat:@"%s", info];
128. self.content.text=info_ns;
129. }
130.
131. - (IBAction)clickAVFilterButton:(id)sender {
132.     char info[40000] = { 0 };
133.     avfilter_register_all();
134.     AVFilter *f_temp = (AVFilter *)avfilter_next(NULL);
135.     while (f_temp != NULL){
136.         sprintf(info, "%s%10s\n", info, f_temp->name);
137.     }
138.     //printf("%s", info);
139.     NSString * info_ns = [NSString stringWithFormat:@"%s", info];
140.     self.content.text=info_ns;
141. }
142.
143. - (IBAction)clickConfigurationButton:(id)sender {
144.     char info[10000] = { 0 };
145.     av_register_all();
146.
147.     sprintf(info, "%s\n", avcodec_configuration());
148.
149.     //printf("%s", info);
150.     //self.content.text=@"Lei Xiaohua";
151.     NSString * info_ns = [NSString stringWithFormat:@"%s", info];
152.     self.content.text=info_ns;
153. }
154.
155.
156. @end

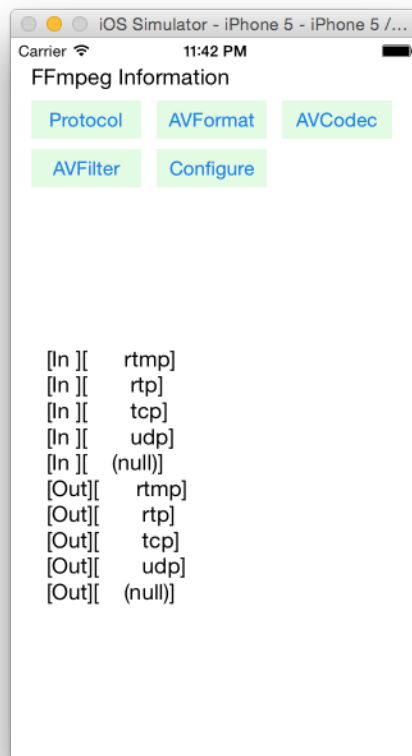
```

运行结果

App在手机上运行后的结果如下图所示。



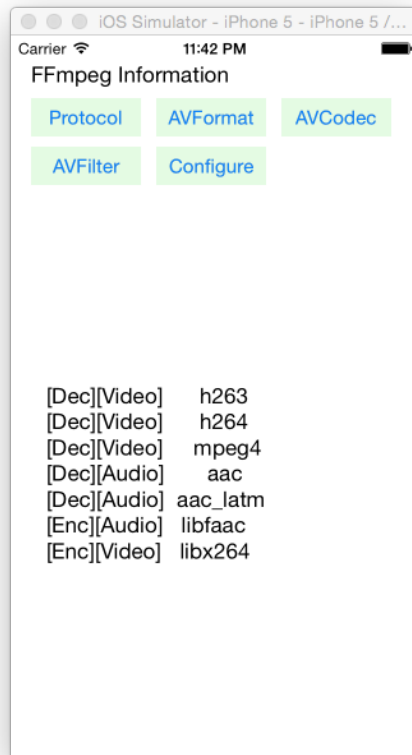
单击不同的按钮，可以得到类库不同方面的信息。单击“Protocol”按钮内容如下所示。



单击“AVFormat”按钮内容如下所示。



单击“AVCodec”按钮内容如下所示。



单击“Configure”按钮即为程序开始运行时候的内容。

下载

simplest ffmpeg mobile

项目主页

Github：https://github.com/leixiaohua1020/simplest_ffmpeg_mobile

开源中国：https://git.oschina.net/leixiaohua1020/simplest_ffmpeg_mobile

SourceForge：<https://sourceforge.net/projects/simplestffmpegmobile/>

CSDN工程下载地址：<http://download.csdn.net/detail/leixiaohua1020/8924391>

本解决方案包含了使用FFmpeg在移动端处理多媒体的各种例子：

[Android]

simplest_android_player: 基于安卓接口的视频播放器

simplest_ffmpeg_android_helloworld: 安卓平台下基于FFmpeg的HelloWorld程序

simplest_ffmpeg_android_decoder: 安卓平台下最简单的基于FFmpeg的视频解码器

simplest_ffmpeg_android_decoder_onelib: 安卓平台下最简单的基于FFmpeg的视频解码器-单库版

simplest_ffmpeg_android_streamer: 安卓平台下最简单的基于FFmpeg的推流器

simplest_ffmpeg_android_transcoder: 安卓平台下移植的FFmpeg命令行工具

simplest_sdl_android_helloworld: 移植SDL到安卓平台的最简单程序

[IOS]

simplest_ios_player: 基于IOS接口的视频播放器

simplest_ffmpeg_ios_helloworld: IOS平台下基于FFmpeg的HelloWorld程序

simplest_ffmpeg_ios_decoder: IOS平台下最简单的基于FFmpeg的视频解码器

simplest_ffmpeg_ios_streamer: IOS平台下最简单的基于FFmpeg的推流器

simplest_ffmpeg_ios_transcoder: IOS平台下移植的ffmpeg.c命令行工具

simplest_sdl_ios_helloworld: 移植SDL到IOS平台的最简单程序

文章标签：[FFmpeg](#) [IOS](#) [编译](#) [arm](#)

个人分类：[FFMPEG](#) [IOS多媒体](#)

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