**[MacOS10.8.3+Xcode4.6+IOS6.1 编译FFmpeg，简单使用](http://www.cnblogs.com/dyllove98/archive/2013/06/07/3125183.html)**

先说一下我的编译环境：MacOS10.8.3、Xcode4.6(4H127)、IOS6.1

文档提供了编译FFmpeg i386(模拟器)版本、armv7版本（iPhone 3GS以上）、armv7s(iPhone5)版本等lib库的方法，最后还提供了一种把这三个版本合为一种公共lib库的脚本，模拟器和真机只需要一套库文件即可。

1.提前准备Command Line Tools，在Xcode-Preference-Downloads-Components下载。

2.从官网下载最新版本的 ffmpeg-0.8.14，解压到桌面或者下载里面，方便编译。从下图可以看到有很多链接可以下载

3.下载最新版本的[gas-preprocessor](https://github.com/mansr/gas-preprocessor)，解压，拷贝gas-preprocessor.pl到 /usr/bin 目录中。注意这个目录是个系统目录，默认隐藏了，直接查看不到。可以先点击一下Finder，上面菜单 前往-前往文件夹，或者直接选中了Finder后按command+shift+g快捷键，粘贴/usr/bin后“前往”就可以打开这个目录了。

4.打开终端，使用cd命令切换到第二步解压后的ffmpeg目录，比如我的是 **cd Desktop/ffmpeg-0.8.14/，**然后在这个文件夹建立几个目录，方便编译时生成lib库，也方便查找，或者使用以下命令在终端里面执行生成，一行一行来。执行完成后，可以在ffmpeg-1.2/文件夹里面看到armv7、armv7s、i386、universal/lib等文件夹。

1. mkdir armv7
2. mkdir armv7s
3. mkdir i386
4. mkdir -p universal/lib

5.编译**ffmpeg armv7（iPhone 3GS以上）**版本，复制以下命令到终端里面执行。注意如果你的SDK是IOS6.0，只需要把iPhoneOS6.1换成iPhoneOS6.0即可

./configure  --prefix=armv7  --cc=/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/usr/bin/gcc  --as='gas-preprocessor.pl /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/usr/bin/gcc'  --sysroot=/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS6.1.sdk  --extra-ldflags=-L/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS6.1.sdk/usr/lib/system  --target-os=darwin  --arch=arm  --cpu=cortex-a8  --extra-cflags='-arch armv7'  --extra-ldflags='-arch armv7'  --enable-pic  --enable-cross-compile  --enable-ffmpeg  --disable-ffplay  --disable-ffserver  --disable-asm  --disable-encoders  --disable-decoders  --enable-decoder=h264  --disable-doc

复制以下命令在终端里面回车执行：

make clean && make && make install

6.编译**ffmpeg armv7s（iPhone5）**版本，复制以下命令到终端里面回车执行：

./configure  --prefix=armv7s --sysroot="/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS6.1.sdk"  --target-os=darwin  --cc="/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/usr/bin/gcc"  --extra-cflags="-arch armv7s -mfpu=neon -miphoneos-version-min=6.0"  --extra-ldflags="-arch armv7s -isysroot /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS6.1.sdk -miphoneos-version-min=6.0"  --arch=arm  --cpu=cortex-a9  --enable-pic --enable-cross-compile  --enable-ffmpeg  --disable-ffplay  --disable-ffserver  --disable-asm  --disable-encoders  --disable-decoders  --enable-decoder=h264  --disable-doc

复制以下命令在终端里面回车执行：

make clean && make && make install

7.编译**ffmpeg i386（模拟器）**版本，复制以下命令到终端里面回车执行：

./configure  --prefix=i386 --sysroot="/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator6.1.sdk"  --target-os=darwin  --cc="/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/usr/bin/gcc"  --extra-cflags="-arch i386"  --extra-ldflags="-arch i386 -isysroot /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator6.1.sdk"  --arch=i386  --cpu=i386 --enable-pic  --enable-cross-compile  --enable-ffmpeg  --disable-ffplay  --disable-ffserver  --disable-asm  --disable-encoders  --disable-decoders  --enable-decoder=h264

复制以下命令在终端里面回车执行：

make clean && make && make install

8.创建公共lib库，就是把上面生成的三个版本的.a文件合并为一个，以后到真机或者模拟器只需要这一套就ok。复制以下命令到终端里面回车执行：

lipo -create -arch armv7k armv7s/lib/libavcodec.a -arch armv7 armv7/lib/libavcodec.a -arch i386 i386/lib/libavcodec.a -output universal/libavcodec.a  
lipo -create -arch armv7k armv7s/lib/libavdevice.a -arch armv7 armv7/lib/libavdevice.a -arch i386 i386/lib/libavdevice.a -output universal/libavdevice.a  
lipo -create -arch armv7k armv7s/lib/libavformat.a -arch armv7 armv7/lib/libavformat.a -arch i386 i386/lib/libavformat.a -output universal/libavformat.a  
lipo -create -arch armv7k armv7s/lib/libavutil.a -arch armv7 armv7/lib/libavutil.a -arch i386 i386/lib/libavutil.a -output universal/libavutil.a  
lipo -create -arch armv7k armv7s/lib/libswscale.a -arch armv7 armv7/lib/libswscale.a -arch i386 i386/lib/libswscale.a -output universal/libswscale.a  
lipo -create -arch armv7k armv7s/lib/libavfilter.a -arch armv7 armv7/lib/libavfilter.a -arch i386 i386/lib/libavfilter.a -output universal/libavfilter.a