1. LOCAL\_PATH := $(call my-dir)//一个Android.mk文件必须以LOCAL\_PATH定义开头，它用来定位源码。这当前这个例子中，宏“call my-dir”由构建内部系统提供，用来返回当前的路径,即包含Android.mk本身的当前目录.  
  2. include $(CLEAR\_VARS)//这个CLEAR\_VARS变量由NDK构建系统内部提供，它指向一个GNU Makefile文件，用来清除一个以LOCAL\_开头的变量的值，但LOCAL\_PATH除外  
  3. LOCAL\_MODULE := hello-jni//LOCAL\_MOUDLE用来定义你的模块名，它的名字必须是唯一的。不能包含空格。需要注意地是，NDK构建系统会自动为所生成的目标模块加上前缀和扩展名，比如foo模块，最终终会生成libfoo.so文件.需要注意地是,如果你为你的模块名加上lib前缀，则NDK构建系统则不会再加上lib前缀,也就是说，生成目标库时，NDK创建系统会自动判断是否需要加上lib前缀  
  4. LOCAL\_SRC\_FILES := hello-jni.c//LOCAL\_SRC\_FILES必须包含C源码或C++源码。注：你不必列出所需要的头文件，NDK构建系统会自动为你找到对应的头文件。另一个需要注意地是，默认C++的扩展名为cpp,但也可以通过变量LOCAL\_CPP\_EXTENSION来定义不同的扩展名,用这个变量时设置其值时千万别忘了第一个字符是点。如.cxx将生效，而cxx则不会.   
  5. LOCAL\_WHOLE\_STATIC\_LIBRARIES ：该cocos2d-x在android 环境下需要信赖的静态库，一般不需要更改这里，除非自己要添加自己的功能需要信赖到第三方库才需要添加到这里来  
  
  6. LOCAL\_C\_INCLUDES ：对应Classes目录的cocos2d-x的h(头)文件  
  7. include $(BUILD\_SHARED\_LIBRARY)//BUILD\_SHARED\_LIBARARY由NDK构建系统内容提供，它指向一个GNU Makefile脚本，负责收集此上次执行'include $(CLEAR\_VARS)'以来的所有LOCAL\_XXX变量信息，和决定最终生成什么，以及如何精确地构建动态库。BUILD\_STATIC\_LIBRARY与此类似，它则生成静态库.它指向一个编译脚本，将收集所有LOCAL\_变量定义的信息，并决定如何生成目标动态库文件。在include 它之前，你必须先使用LOCAL\_MODULE,LOCAL\_SRC\_FIELS定义模块名和源文件列表。使用方法如下:  
  
  
include $(BUILD\_SHARED\_LIBRARY)注:生成的文件为 lib$(LOCAL\_MODULE).so  
  8.   
  
  
编译预定的变量名拥有以下规则:  
  
- 以前缀LOCAL\_开头  (如： LOCAL\_MODULE)- 以前缀PRIVATE\_, NDK\_ or APP\_开头  (系统内容使用)- 小写字母(内容使用, 如. 'my-dir’)   9.BUILD\_STATIC\_LIBRARY:BUILD\_SHARED\_LIBRARY的一个变种，用来生成静态库。静态库不会被拷贝到你的apk内，但它可以用来生成so文件，参考接下来将要介绍的变量LOCAL\_STATIC\_LIBRARIES 和 LOCAL\_WHOLE\_STATIC\_LIBRARIES,使用方法如下:include $(BUILD\_STATIC\_LIBRARY)注：使用它最终将生成lib$(LOCAL\_MODULE).a文件  10.$(call import-module,相对路径)导入外部模块的.mk文件 ，和 include基本一样概念上的区别是include导入的是由我们自己写的.mk。而import-module导入的是外部库、外部模块提供的.mk。用法上：include的路径是.mk文件的绝对路径。而import是设置的路径指定到模块的.mk所在目录，是相对于NDK\_MODULE\_PATH中的路径列表的相对路径   11.import-module的使用注意:     1、设置路径时，注意与NDK\_MODULE\_PATH中的路径相互配合。  
      2、导入模块的.mk中如果也有import-module，则注意其相对路径也要在NDK\_MODULE\_PATH中。  
      3、上面说了import-module和include一样。如果import-module和Include包含了同一个.mk,会报重复包含的错误。  
  
  
  12.NDK\_MODULE\_PATH的作用:  
       NDK\_MODULE\_PATH是一个很重要的变量，当android.mk中使用了$(call import-module,XXX)函数引入外部库文件时会用到，用以指示该往哪里去找这个文件。  
       NDK\_MODULE\_PATH 是一个环境变量，不是android.mk中设置的变量。NDK\_MODULE\_PATH多个路径用冒号分割。  
       设置NDK\_MODULE\_PATH的方法:  
  
   1、在系统环境里手动添加这个环境变量，  
    2、在build\_native.sh中 运行ndk-build之前使用export命令定义环境变量NDK\_MODULE\_PATH。  
    如：export NDK\_MODULE\_PATH=路径1:路径2:路径3  
    3、直接将NDK\_MODULE\_PATH=路径1:路径2 加到 ndk-build命令的参数后面。ndk-build的参数最终会直接传给make.  
    如：$NDK\_ROOT\_LOCAL/ndk-build -C $HELLOWORLD\_ROOT NDK\_MODULE\_PATH=路径1:路径2  
(命令 make aaa=213 //在编译makefile之前将aaa当作环境变量设置为213.)s  
    4、还可以在android.mk中设置NDK\_MODULE\_PATH  
    在import语句之前加入，  
$(call import-add-path,$(LOCAL\_PATH)/platform/third\_party/android/prebuilt)  
    将一个新的路径加入NDK\_MODULE\_PATH变量  
  13.LOCAL\_LDLIBS := -lGLESv2 \ 需要链接的系统默认库  
  14.$(call import-add-path,$(LOCAL\_PATH))添加外部导入库目录    
  15.LOCAL\_CFLAGS := -DUSE\_FILE32API  
     LOCAL\_EXPORT\_CFLAGS := -DUSE\_FILE32API  
      编译C,CPP源代码的一组选项。通常用来添加外部宏定义或编译选项，也可以在LOCAL\_CFLAGS中添加头文件搜索路径  
     LOCAL\_CFLAGS := \  
       '-D\_STLP\_USE\_NEWALLOC’ //Android NDK r6b使用stl遇到undefined reference to   `std::\_\_node\_alloc::\_M\_allocate(unsigned int&)” 的终极解决办法