㊐ **FFmpeg源代码简单分析：makefile**

===

FFmpeg的库函数源代码分析文章列表：

【架构图】

FFmpeg 源代码结构图 - 解码

FFmpeg 源代码结构图 - 编码

【通用】

FFmpeg 源代码简单分析：av_register_all()

FFmpeg 源代码简单分析：avcodec_register_all()

FFmpeg 源代码简单分析：内存的分配和释放（av_malloc()、 av_free() 等）

FFmpeg 源代码简单分析：常见结构体的初始化和销毁（AVFormatContext， AVFrame 等）

FFmpeg 源代码简单分析：avio_open2()

FFmpeg 源代码简单分析：av_find_decoder() 和 av_find_encoder()

FFmpeg 源代码简单分析：avcodec_open2()

FFmpeg 源代码简单分析：avcodec_close()

【解码】

图解 FFMPEG 打开媒体的函数 avformat_open_input

FFmpeg 源代码简单分析：avformat_open_input()

FFmpeg 源代码简单分析：avformat_find_stream_info()

FFmpeg 源代码简单分析：av_read_frame()

FFmpeg 源代码简单分析：avcodec_decode_video2()

FFmpeg 源代码简单分析：avformat_close_input()

【编码】

FFmpeg 源代码简单分析：avformat_alloc_output_context2()

FFmpeg 源代码简单分析：avformat_write_header()

FFmpeg 源代码简单分析：avcodec_encode_video()

FFmpeg 源代码简单分析：av_write_frame()

FFmpeg 源代码简单分析：av_write_trailer()

【其它】

FFmpeg 源代码简单分析：日志输出系统（av_log() 等）

FFmpeg 源代码简单分析：结构体成员管理系统 -AVClass

FFmpeg 源代码简单分析：结构体成员管理系统 -AVOption

FFmpeg 源代码简单分析：libswscale 的 sws_getContext()

FFmpeg 源代码简单分析：libswscale 的 sws_scale()

FFmpeg 源代码简单分析：libavdevice 的 avdevice_register_all()

FFmpeg 源代码简单分析：libavdevice 的 gdigrab

【脚本】

FFmpeg 源代码简单分析： makefile

FFmpeg 源代码简单分析： configure

【H.264】

FFmpeg 的 H.264 解码器源代码简单分析：概述

==

本文记录FFmpeg的Makefile的源代码。Makefile用于编译FFmpeg的源代码。通过分析Makefile文件，可以了解FFmpeg的源代码生成的流程。有关Makefile这部分我本身基础不算很牢，很多地方还在慢慢摸索，所以分析的内容不能保证完全准确。以后有时间和其他朋友多交流再慢慢完善这篇文章。

PS：make有一个很有用的命令："make -n"。该选项会显示命令，但不会执行命令，十分有助于分析Makefile

FFmpeg中与Makefile相关的文件

FFmpeg中与Makefile相关的文件主要有以下几个：
　　根目录Makefile：最基本的Makefile；
　　config.mak：由configure生成的Makefile，保存了Configure的设置信息；
　　libavXXXX/Makefile：每个类库的Makefile（仅仅设置了几个变量）；
　　library.mak：编译类库的Makefile（和libavXXXX/Makefile配合使用）；
　　common.mak：包含一些通用代码的Makefile；
它们之间的关系如下图所示。

图中除了画出了Makefile之外，还画出了和Makefile有关的一些文件：
　　XXX.c：C语言文件；
　　XXX.h：C语言文件用到的头文件；
　　XXX.o：C语言文件对应的目标文件；
　　XXX.d：C语言文件对应的依赖关系文件；

Make Install 之间的关系

简单分析了一下Makefile中的make install之间的关系，如下图所示（使用的是MinGW编译器）。

从图中可以看出，install伪目标依赖于4个伪目标：

　　install-progs：安装应用程序ffmpeg.exe，ffplay.exe，ffprobe.exe；
　　install-data：安装数据（*.ffpreset之类的文件，没研究过）
　　install-libs：安装类库（libavcodec.dll等文件）
　　install-headers：安装头文件（libavcodec/avcodec.h等文件）
install-progs依赖于ffmpeg.exe，ffplay.exe，ffprobe.exe以及install-progs-yes伪目标。在CONFIG_SHARED取值为yes的情况下，install-progs-yes依赖于install-libs。

install-libs依赖于伪目标install-libs-yes。install-libs-yes的依赖关系位于library.mak文件中。如果CONFIG_STATIC取值为yes，install-libs-yes依赖于install-lib$(NAME)-static（其中${NAME}为类库文件名，例如avformat）；如果CONFIG_SHARED取值为yes，install-libs-yes依赖于install-lib$(NAME)-shared。
install-headers依赖于伪目标install-lib$(NAME)-headers和install-lib$(NAME)-pkgconfig。

和install相对应，uninstall伪目标依赖于4个伪目标：
　　uninstall-progs：卸载应用程序；
　　uninstall-data：卸载数据；
　　uninstall-libs：卸载类库；
　　uninstall-headers：卸载头文件；
其中uninstall-libs和uninstall-headers的依赖关系位于library.mak中。

根目录Makefile

根目录Makefile是最重要的。简单注释过的Makefile如下所示。

```python
1.  # FFmpeg Main Makefile
2.  #
```

```
3.   # 注释：雷霄骅
4.   # leixiaohua1020@126.com
5.   # http://blog.csdn.net/leixiaohua1020
6.   #
7.   # FFmpeg的 Main Makefile。最重要。
8.
9.   MAIN_MAKEFILE=1
10.  #重要：包含了configure信息
11.  include config.mak
12.
13.  #config.mak中：
14.  #SRC_PATH=.
15.  #在SRC_PATH搜索各种类型的文件
16.  vpath %.c    $(SRC_PATH)
17.  vpath %.cpp  $(SRC_PATH)
18.  vpath %.h    $(SRC_PATH)
19.  vpath %.S    $(SRC_PATH)
20.  vpath %.asm  $(SRC_PATH)
21.  vpath %.v    $(SRC_PATH)
22.  vpath %.texi $(SRC_PATH)
23.  vpath %/fate_config.sh.template $(SRC_PATH)
24.
25.  #CONFIG_XXX取值为yes
26.  #PROGS-yes= ffmpeg ffplay ffprobe
27.  PROGS-$(CONFIG_FFMPEG)   += ffmpeg
28.  PROGS-$(CONFIG_FFPLAY)   += ffplay
29.  PROGS-$(CONFIG_FFPROBE)  += ffprobe
30.  PROGS-$(CONFIG_FFSERVER) += ffserver
31.
32.  #config.mak中：
33.  #EXESUF=.exe
34.  #PROGSSUF=
35.  #$(var:%.c=%.o)意思是把.c 为结尾的变量替换成.o。
36.  #没有".c"的时候，代表匹配所有
37.  PROGS        := $(PROGS-yes:%=%$(EXESUF))
38.  INSTPROGS    = $(PROGS-yes:%=%$(PROGSSUF)$(EXESUF))
39.  OBJS         = $(PROGS-yes:%=%.o) cmdutils.o
40.  TESTTOOLS    = audiogen videogen rotozoom tiny_psnr base64
41.  HOSTPROGS    := $(TESTTOOLS:%=tests/%)
42.  TOOLS        = qt-faststart trasher
43.  TOOLS-$(CONFIG_ZLIB) += cws2fws
44.
45.  #PROGS= ffmpeg.exe ffplay.exe ffprobe.exe
46.  #INSTPROGS= ffmpeg.exe ffplay.exe ffprobe.exe
47.  #OBJS= ffmpeg.o ffplay.o ffprobe.o
48.
49.  BASENAMES    = ffmpeg ffplay ffprobe ffserver
50.  ALLPROGS     = $(BASENAMES:%=%$(PROGSSUF)$(EXESUF))
51.  ALLPROGS_G   = $(BASENAMES:%=%$(PROGSSUF)_g$(EXESUF))
52.  ALLMANPAGES = $(BASENAMES:%=%.1)
53.
54.  #ALLPROGS= ffmpeg.exe ffplay.exe ffprobe.exe ffserver.exe
55.  #ALLPROGS_G= ffmpeg_g.exe ffplay_g.exe ffprobe_g.exe ffserver_g.exe
56.  #ALLMANPAGES=ffmpeg.1 ffplay.1 ffprobe.1 ffserver.1
57.  FFLIBS-$(CONFIG_AVDEVICE) += avdevice
58.  FFLIBS-$(CONFIG_AVFILTER) += avfilter
59.  FFLIBS-$(CONFIG_AVFORMAT) += avformat
60.  FFLIBS-$(CONFIG_AVCODEC)  += avcodec
61.  FFLIBS-$(CONFIG_POSTPROC) += postproc
62.  FFLIBS-$(CONFIG_SWRESAMPLE)+= swresample
63.  FFLIBS-$(CONFIG_SWSCALE)  += swscale
64.
65.  #FFLIBS-yes= avdevice avfilter avformat avcodec postproc swresample swscale
66.  #一定需要libavutil
67.  FFLIBS := avutil
68.  #让通配符在变量中展开，需要使用wildcard关键字
69.  DATA_FILES := $(wildcard $(SRC_PATH)/presets/*.ffpreset) $(SRC_PATH)/doc/ffprobe.xsd
70.
71.  SKIPHEADERS = cmdutils_common_opts.h
72.  #重要
73.  include $(SRC_PATH)/common.mak
74.  #依赖的类库
75.  FF_EXTRALIBS := $(FFEXTRALIBS)
76.  FF_DEP_LIBS  := $(DEP_LIBS)
77.
78.  #伪目标
79.  #all是最关键的，生成最后的程序
80.  #all: ffmpeg.exe ffplay.exe ffprobe.exe
81.  #
82.  all: $(PROGS)
83.  #config.mak中：
84.  #EXESUF=.exe
85.  #PROGSSUF=
86.  #$@是一个自动化变量。可以简单理解为目标的集合。
87.  #$<是一个自动化变量。可以简单理解为依赖目标的集合。
88.  #%是通配符
89.  #两个冒号，"静态模式规则"。
90.  #ffmpeg_g.exe生成ffmpeg.exe；ffplay_g.exe生成ffplay.exe；ffprobe_g.exe生成ffprobe.exe
91.  #strip经常用来去除目标文件中的一些符号表、调试符号表信息，以减小程序的大小
92.  $(PROGS): %$(EXESUF): %$(PROGSSUF)_g$(EXESUF)
93.      $(CP) $< $@$(PROGSSUF)
```

```makefile
 94.         $(STRIP) $@$(PROGSSUF)
 95.
 96.    $(TOOLS): %$(EXESUF): %.o
 97.        $(LD) $(LDFLAGS) -o $@ $< $(ELIBS)
 98.
 99.    tools/cws2fws$(EXESUF): ELIBS = -lz
100.
101.    config.h: .config
102.    .config: $(wildcard $(FFLIBS:%=$(SRC_PATH)/lib%/all*.c))
103.        @-tput bold 2>/dev/null
104.        @-printf '\nWARNING: $(?F) newer than config.h, rerun configure\n\n'
105.        @-tput sgr0 2>/dev/null
106.    #给子目录中的Makefile使用的变量
107.    SUBDIR_VARS := OBJS FFLIBS CLEANFILES DIRS TESTPROGS EXAMPLES SKIPHEADERS \
108.                   ALTIVEC-OBJS MMX-OBJS NEON-OBJS X86-OBJS YASM-OBJS-FFT YASM-OBJS \
109.                   HOSTPROGS BUILT_HEADERS TESTOBJS ARCH_HEADERS ARMV6-OBJS TOOLS
110.
111.    define RESET
112.    $(1) :=
113.    $(1)-yes :=
114.    endef
115.
116.    #$(call <expression>,<parm1>,<parm2>,<parm3>...)
117.    #当make执行这个函数时，<expression>参数中的变量，如$(1)，$(2)，$(3)等，会被参数
118.    #<parm1>，<parm2>，<parm3>依次取代。而<expression>的返回值就是call函数的返回值。
119.
120.    #命令包==========================
121.    #用于编译每个库
122.    #$(1)取值为libavcodec，libavcodec等等
123.    define DOSUBDIR
124.    $(foreach V,$(SUBDIR_VARS),$(eval $(call RESET,$(V))))
125.    SUBDIR := $(1)/
126.    #每个库目录下的Makefile
127.    include $(SRC_PATH)/$(1)/Makefile
128.    #注：make一般情况下如果在中途检测到有执行出错的情况(返回非 0 状态)，那么就会放弃对当前规则后续命令的执行。
129.    #在命令前面加上"-"号之后，就算执行错误了，也会继续执行下去
130.    -include $(SRC_PATH)/$(1)/$(ARCH)/Makefile
131.    #编译类库
132.    include $(SRC_PATH)/library.mak
133.    endef
134.    #==============================
135.
136.    #$(foreach <var>,<list>,<text>)
137.    #把参数<list>中的单词逐一取出放到参数<var>所指定的变量中，
138.    #然后再执行<text>所包含的表达式。
139.    #$(eval text)
140.    #text的内容将作为makefile的一部分而被make解析和执行
141.    #
142.    #循环调用DOSUBDIR命令包
143.    #这一步会将libavcodec，libavformat等文件夹下的Makefile包含进来。
144.    $(foreach D,$(FFLIBS),$(eval $(call DOSUBDIR,lib$(D))))
145.    #ffplay需要SDL
146.    ffplay.o: CFLAGS += $(SDL_CFLAGS)
147.    ffplay_g$(EXESUF): FF_EXTRALIBS += $(SDL_LIBS)
148.    ffserver_g$(EXESUF): LDFLAGS += $(FFSERVERLDFLAGS)
149.    #链接生成ffmpeg_g.exe等等
150.    #FF_DEP_LIBS= libavcodec/libavcodec.a libavutil/libavutil.a ....
151.    %$(PROGSSUF)_g$(EXESUF): %.o cmdutils.o $(FF_DEP_LIBS)
152.        $(LD) $(LDFLAGS) -o $@ $< cmdutils.o $(FF_EXTRALIBS)
153.
154.    OBJDIRS += tools
155.
156.    -include $(wildcard tools/*.d)
157.
158.    VERSION_SH  = $(SRC_PATH)/version.sh
159.    GIT_LOG     = $(SRC_PATH)/.git/logs/HEAD
160.
161.    .version: $(wildcard $(GIT_LOG)) $(VERSION_SH) config.mak
162.    .version: M=@
163.
164.    version.h .version:
165.        $(M)$(VERSION_SH) $(SRC_PATH) version.h $(EXTRA_VERSION)
166.        $(Q)touch .version
167.
168.    # force version.sh to run whenever version might have changed
169.    -include .version
170.    #安装install
171.    #安装程序
172.    ifdef PROGS
173.    install: install-progs install-data
174.    endif
175.    #安装类库和头文件
176.    install: install-libs install-headers
177.    #install-libs-yes位于library.mak
178.    install-libs: install-libs-yes
179.
180.    install-progs-yes:
181.    install-progs-$(CONFIG_SHARED): install-libs
182.
183.    #config.mak中：
184.    #BINDIR=$(DESTDIR)${prefix}/bin
185.    #INSTALL=install
```

```
185.    #INSTALL=install
186.    #cp与install区别：
187.    #cp会先清空文件后往里写入新文件，而install则会先删除掉原先的文件然后写入新文件。
188.    install-progs: install-progs-yes $(PROGS)
189.        $(Q)mkdir -p "$(BINDIR)"
190.        $(INSTALL) -c -m 755 $(INSTPROGS) "$(BINDIR)"
191.
192.    install-data: $(DATA_FILES)
193.        $(Q)mkdir -p "$(DATADIR)"
194.        $(INSTALL) -m 644 $(DATA_FILES) "$(DATADIR)"
195.    #卸载
196.    uninstall: uninstall-libs uninstall-headers uninstall-progs uninstall-data
197.    #addprefix()用于加前缀
198.    #在这里获取ffmpeg.exe等的完整路径（用于删除）
199.    uninstall-progs:
200.        $(RM) $(addprefix "$(BINDIR)/", $(ALLPROGS))
201.
202.    uninstall-data:
203.        $(RM) -r "$(DATADIR)"
204.    #清空
205.    clean::
206.        $(RM) $(ALLPROGS) $(ALLPROGS_G)
207.        $(RM) $(CLEANSUFFIXES)
208.        $(RM) $(TOOLS)
209.        $(RM) $(CLEANSUFFIXES:%=tools/%)
210.        $(RM) coverage.info
211.        $(RM) -r coverage-html
212.
213.    distclean::
214.        $(RM) $(DISTCLEANSUFFIXES)
215.        $(RM) config.* .version version.h libavutil/avconfig.h
216.
217.    config:
218.        $(SRC_PATH)/configure $(value FFMPEG_CONFIGURATION)
219.
220.    # Without the sed genthml thinks "libavutil" and "./libavutil" are two different things
221.    coverage.info: $(wildcard *.gcda *.gcno */*.gcda */*.gcno */*/*.gcda */*/*.gcno)
222.        $(Q)lcov -c -d . -b . | sed -e 's#/./#/#g' > $@
223.
224.    coverage-html: coverage.info
225.        $(Q)mkdir -p $@
226.        $(Q)genhtml -o $@ $<
227.        $(Q)touch $@
228.
229.    include $(SRC_PATH)/doc/Makefile
230.    include $(SRC_PATH)/tests/Makefile
231.
232.    $(sort $(OBJDIRS)):
233.        $(Q)mkdir -p $@
234.
235.    # Dummy rule to stop make trying to rebuild removed or renamed headers
236.    %.h:
237.        @:
238.
239.    # Disable suffix rules.  Most of the builtin rules are suffix rules,
240.    # so this saves some time on slow systems.
241.    .SUFFIXES:
242.    #显示地指明一个目标是"伪目标"
243.    .PHONY: all all-yes alltools *clean config examples install*
244.    .PHONY: testprogs uninstall*
```

根目录Makefile代码一开始的时候包含了config.mak文件。这个文件是运行./configure的后生成的配置文件，包含了所有的配置信息。

随后代码定义了ffplay.exe，ffmpeg.exe，ffprobe.exe与ffplay_g.exe，ffmpeg_g.exe，ffprobe_g.exe的依赖关系。然后定义了ffplay_g.exe，ffmpeg_g.exe，ffprobe_g.exe与libavformat，libavcodec等这些类库的依赖关系。

根目录Makefile中也定义了all，install，uninstall，clean等等一系列的伪目标，这样可以通过给Makefile指定不同的目标来完成不同的事。

此外根目录的Makefile中有一个很重要的命令包DOSUBDIR。在该命令包通过包含libavXXX/Makefile和library.mak等文件，定义了FFmpeg类库（例如libavformat，libavcodec，libavutil等）的依赖关系。

config.mak

config.mak文件是运行./configure的后生成的配置文件，包含了所有的配置信息。简单注释过的config.mak的代码如下所示。

```python
1.    # FFmpeg config.mak
2.    #
3.    # 注释：雷霄骅
4.    # leixiaohua1020@126.com
5.    # http://blog.csdn.net/leixiaohua1020
6.    #
7.    # Configure脚本生成的Makefile，包含了各种配置信息。
8.    #
9.    # Automatically generated by configure - do not modify!
```

```makefile
#基本信息
ifndef FFMPEG_CONFIG_MAK
FFMPEG_CONFIG_MAK=1
FFMPEG_CONFIGURATION=
#各种路径=====================================
prefix=/usr/local
LIBDIR=$(DESTDIR)${prefix}/lib
SHLIBDIR=$(DESTDIR)${prefix}/bin
INCDIR=$(DESTDIR)${prefix}/include
BINDIR=$(DESTDIR)${prefix}/bin
DATADIR=$(DESTDIR)${prefix}/share/ffmpeg
MANDIR=$(DESTDIR)${prefix}/share/man
#是个相对路径
SRC_PATH=.
ifndef MAIN_MAKEFILE
SRC_PATH:=$(SRC_PATH:.%=..%)
endif
#工具集=====================================
CC_IDENT=gcc 4.6.2 (GCC)
#架构
ARCH=x86
#编译器
CC=gcc
CXX=g++
AS=gcc
#链接器
LD=gcc
DEPCC=gcc
#汇编器
YASM=yasm
YASMDEP=yasm
#生成静态库.a工具
AR=ar
RANLIB=ranlib
CP=cp -p
LN_S=ln -sf
STRIP=strip
#参数集=====================================
#编译器的参数
CPPFLAGS= -D_ISOC99_SOURCE -D_FILE_OFFSET_BITS=64 -D_LARGEFILE_SOURCE -U__STRICT_ANSI__
CFLAGS=   -std=c99 -fno-common -fomit-frame-pointer -I/include/SDL -D_GNU_SOURCE=1 -Dmain=SDL_main -g -Wdeclaration-after-statement
-Wall -Wno-parentheses -Wno-switch -Wno-format-zero-length -Wdisabled-optimization -Wpointer-arith -Wredundant-decls -Wno-pointer-si
gn -Wcast-qual -Wwrite-strings -Wtype-limits -Wundef -Wmissing-prototypes -Wno-pointer-to-int-cast -Wstrict-prototypes -O3 -fno-math
-errno -fno-signed-zeros -fno-tree-vectorize -Werror=implicit-function-declaration -Werror=missing-prototypes
CXXFLAGS=  -D__STDC_CONSTANT_MACROS
ASFLAGS=   -g
#目标文件有关的参数
AS_O=-o $@
CC_O=-o $@
CXX_O=-o $@
#链接器有关的参数
LDFLAGS= -Wl,--as-needed -Wl,--warn-common -Wl,-rpath-
link=libpostproc:libswresample:libswscale:libavfilter:libavdevice:libavformat:libavcodec:libavutil
FFSERVERLDFLAGS=-Wl,-E
SHFLAGS=-shared -Wl,--output-def,$$(@:$(SLIBSUF)=.def) -Wl,--out-implib,$(SUBDIR)lib$(SLIBNAME:$(SLIBSUF)=.dll.a) -Wl,--enable-runti
me-pseudo-reloc -Wl,--enable-auto-image-base -Wl,-Bsymbolic -Wl,--version-script,$(SUBDIR)lib$(NAME).ver
YASMFLAGS=-f win32  -DPREFIX
#前缀后缀=====================================
BUILDSUF=
PROGSSUF=
#${NAME}位于每个liavXXX/Makefile中，例如avformat
FULLNAME=$(NAME)$(BUILDSUF)
LIBPREF=lib
LIBSUF=.a
#例如libavformat.a
LIBNAME=$(LIBPREF)$(FULLNAME)$(LIBSUF)
SLIBPREF=
SLIBSUF=.dll
EXESUF=.exe
EXTRA_VERSION=
DEPFLAGS=$(CPPFLAGS) $(CFLAGS) -MM
CCDEP=
CXXDEP=$(DEPCC) $(DEPFLAGS) $< | sed -e "/^\#.*/d" -e "s,^[[:space:]]*$(*F)\\.o,$(@D)/$(*F).o," > $(@:.o=.d)
ASDEP=
CC_DEPFLAGS=-MMD -MF $(@:.o=.d) -MT $@
AS_DEPFLAGS=-MMD -MF $(@:.o=.d) -MT $@
HOSTCC=gcc
HOSTCFLAGS=-D_ISOC99_SOURCE -O3 -g -std=c99 -Wall
HOSTEXESUF=.exe
HOSTLDFLAGS=
HOSTLIBS=-lm
TARGET_EXEC=
TARGET_PATH=$(CURDIR)
#SDL
SDL_LIBS=-L/lib -lmingw32 -lSDLmain -lSDL -mwindows
SDL_CFLAGS=-I/include/SDL -D_GNU_SOURCE=1 -Dmain=SDL_main
LIB_INSTALL_EXTRA_CMD=$$(RANLIB) "$(LIBDIR)/$(LIBNAME)"
#链接
EXTRALIBS=-lavicap32 -lws2_32 -L/lib -lmingw32 -lSDLmain -lSDL -mwindows -lm -lz -lpsapi
INSTALL=install
LIBTARGET=i386
#例如libavformat.dll
```

```
96.  #例如libavformat.dll
97.  SLIBNAME=$(SLIBPREF)$(FULLNAME)$(SLIBSUF)
98.  #LIBVERSION变量位于library.mak
99.  #例如libavformat-53.dll
100. #生成的Dll似乎就是这个版本的
101. SLIBNAME_WITH_VERSION=$(SLIBPREF)$(FULLNAME)-$(LIBVERSION)$(SLIBSUF)
102. #例如libavformat-53.31.100.dll
103. SLIBNAME_WITH_MAJOR=$(SLIBPREF)$(FULLNAME)-$(LIBMAJOR)$(SLIBSUF)
104. SLIB_CREATE_DEF_CMD=
105. #生成导出库lib,会调用lib.exe
106. SLIB_EXTRA_CMD=-lib.exe /machine:$(LIBTARGET) /def:$$(@:$(SLIBSUF)=.def) /out:$(SUBDIR)$(SLIBNAME:$(SLIBSUF)=.lib)
107. SLIB_INSTALL_NAME=$(SLIBNAME_WITH_MAJOR)
108. SLIB_INSTALL_LINKS=
109. SLIB_INSTALL_EXTRA_LIB=lib$(SLIBNAME:$(SLIBSUF)=.dll.a) $(SLIBNAME_WITH_MAJOR:$(SLIBSUF)=.def)
110. SLIB_INSTALL_EXTRA_SHLIB=$(SLIBNAME:$(SLIBSUF)=.lib)
111. SAMPLES:=$(FATE_SAMPLES)
112. NOREDZONE_FLAGS=-mno-red-zone
113. #版本信息=====================================
114. libavcodec_VERSION=53.60.100
115. libavcodec_VERSION_MAJOR=53
116. libavdevice_VERSION=53.4.100
117. libavdevice_VERSION_MAJOR=53
118. libavfilter_VERSION=2.60.100
119. libavfilter_VERSION_MAJOR=2
120. libavformat_VERSION=53.31.100
121. libavformat_VERSION_MAJOR=53
122. libavutil_VERSION=51.34.101
123. libavutil_VERSION_MAJOR=51
124. libpostproc_VERSION=52.0.100
125. libpostproc_VERSION_MAJOR=52
126. libswresample_VERSION=0.6.100
127. libswresample_VERSION_MAJOR=0
128. libswscale_VERSION=2.1.100
129. libswscale_VERSION_MAJOR=2
130. #组件配置=====================================
131. #ARCH_
132. !ARCH_ALPHA=yes
133. !ARCH_ARM=yes
134. !ARCH_AVR32=yes
135. !ARCH_AVR32_AP=yes
136. !ARCH_AVR32_UC=yes
137. !ARCH_BFIN=yes
138. !ARCH_IA64=yes
139. !ARCH_M68K=yes
140. !ARCH_MIPS=yes
141. !ARCH_MIPS64=yes
142. !ARCH_PARISC=yes
143. !ARCH_PPC=yes
144. !ARCH_PPC64=yes
145. !ARCH_S390=yes
146. !ARCH_SH4=yes
147. !ARCH_SPARC=yes
148. !ARCH_SPARC64=yes
149. !ARCH_TOMI=yes
150. ARCH_X86=yes
151. ARCH_X86_32=yes
152. !ARCH_X86_64=yes
153. #HAVE_
154. !HAVE_ALTIVEC=yes
155. HAVE_AMD3DNOW=yes
156. HAVE_AMD3DNOWEXT=yes
157. !HAVE_ARMV5TE=yes
158. !HAVE_ARMV6=yes
159. !HAVE_ARMV6T2=yes
160. !HAVE_ARMVFP=yes
161. HAVE_AVX=yes
162. !HAVE_IWMMXT=yes
163. !HAVE_MMI=yes
164. HAVE_MMX=yes
165. HAVE_MMX2=yes
166. !HAVE_NEON=yes
167. !HAVE_PPC4XX=yes
168. HAVE_SSE=yes
169. HAVE_SSSE3=yes
170. !HAVE_VFPV3=yes
171. !HAVE_VIS=yes
172. !HAVE_BIGENDIAN=yes
173. HAVE_FAST_UNALIGNED=yes
174. !HAVE_PTHREADS=yes
175. HAVE_W32THREADS=yes
176. !HAVE_OS2THREADS=yes
177. HAVE_ALIGNED_STACK=yes
178. !HAVE_ALSA_ASOUNDLIB_H=yes
179. !HAVE_ALTIVEC_H=yes
180. !HAVE_ARPA_INET_H=yes
181. !HAVE_ASM_MOD_Y=yes
182. !HAVE_ASM_TYPES_H=yes
183. HAVE_ATTRIBUTE_MAY_ALIAS=yes
184. HAVE_ATTRIBUTE_PACKED=yes
185. HAVE_CBRTF=yes
186. HAVE_CLOSESOCKET=yes
187. !HAVE_CMOV=yes
```

```
188.  !HAVE_DCBZL=yes
189.  !HAVE_DEV_BKTR_IOCTL_BT848_H=yes
190.  !HAVE_DEV_BKTR_IOCTL_METEOR_H=yes
191.  !HAVE_DEV_IC_BT8XX_H=yes
192.  !HAVE_DEV_VIDEO_BKTR_IOCTL_BT848_H=yes
193.  !HAVE_DEV_VIDEO_METEOR_IOCTL_METEOR_H=yes
194.  !HAVE_DLFCN_H=yes
195.  !HAVE_DLOPEN=yes
196.  HAVE_DOS_PATHS=yes
197.  HAVE_EBP_AVAILABLE=yes
198.  HAVE_EBX_AVAILABLE=yes
199.  HAVE_EXP2=yes
200.  HAVE_EXP2F=yes
201.  !HAVE_FAST_64BIT=yes
202.  HAVE_FAST_CLZ=yes
203.  !HAVE_FAST_CMOV=yes
204.  !HAVE_FCNTL=yes
205.  !HAVE_FORK=yes
206.  !HAVE_GETADDRINFO=yes
207.  !HAVE_GETHRTIME=yes
208.  HAVE_GETPROCESSAFFINITYMASK=yes
209.  HAVE_GETPROCESSMEMORYINFO=yes
210.  HAVE_GETPROCESSTIMES=yes
211.  !HAVE_GETRUSAGE=yes
212.  HAVE_GNU_AS=yes
213.  !HAVE_IBM_ASM=yes
214.  !HAVE_INET_ATON=yes
215.  HAVE_INLINE_ASM=yes
216.  HAVE_ISATTY=yes
217.  HAVE_KBHIT=yes
218.  !HAVE_LDBRX=yes
219.  HAVE_LLRINT=yes
220.  HAVE_LLRINTF=yes
221.  HAVE_LOCAL_ALIGNED_16=yes
222.  HAVE_LOCAL_ALIGNED_8=yes
223.  !HAVE_LOCALTIME_R=yes
224.  HAVE_LOG2=yes
225.  HAVE_LOG2F=yes
226.  !HAVE_LOONGSON=yes
227.  HAVE_LRINT=yes
228.  HAVE_LRINTF=yes
229.  !HAVE_LZO1X_999_COMPRESS=yes
230.  !HAVE_MACHINE_IOCTL_BT848_H=yes
231.  !HAVE_MACHINE_IOCTL_METEOR_H=yes
232.  HAVE_MAKEINFO=yes
233.  HAVE_MALLOC_H=yes
234.  HAVE_MAPVIEWOFFILE=yes
235.  !HAVE_MEMALIGN=yes
236.  !HAVE_MKSTEMP=yes
237.  !HAVE_MMAP=yes
238.  HAVE_PEEKNAMEDPIPE=yes
239.  !HAVE_POLL_H=yes
240.  !HAVE_POSIX_MEMALIGN=yes
241.  HAVE_ROUND=yes
242.  HAVE_ROUNDF=yes
243.  !HAVE_SCHED_GETAFFINITY=yes
244.  HAVE_SDL=yes
245.  HAVE_SDL_VIDEO_SIZE=yes
246.  HAVE_SETMODE=yes
247.  !HAVE_SETRLIMIT=yes
248.  !HAVE_SNDIO_H=yes
249.  HAVE_SOCKLEN_T=yes
250.  !HAVE_SOUNDCARD_H=yes
251.  !HAVE_STRERROR_R=yes
252.  !HAVE_STRPTIME=yes
253.  HAVE_STRUCT_ADDRINFO=yes
254.  HAVE_STRUCT_IPV6_MREQ=yes
255.  !HAVE_STRUCT_RUSAGE_RU_MAXRSS=yes
256.  HAVE_STRUCT_SOCKADDR_IN6=yes
257.  !HAVE_STRUCT_SOCKADDR_SA_LEN=yes
258.  HAVE_STRUCT_SOCKADDR_STORAGE=yes
259.  !HAVE_STRUCT_V4L2_FRMIVALENUM_DISCRETE=yes
260.  HAVE_SYMVER=yes
261.  HAVE_SYMVER_ASM_LABEL=yes
262.  !HAVE_SYMVER_GNU_ASM=yes
263.  !HAVE_SYSCONF=yes
264.  !HAVE_SYSCTL=yes
265.  !HAVE_SYS_MMAN_H=yes
266.  HAVE_SYS_PARAM_H=yes
267.  !HAVE_SYS_RESOURCE_H=yes
268.  !HAVE_SYS_SELECT_H=yes
269.  !HAVE_SYS_SOUNDCARD_H=yes
270.  !HAVE_SYS_VIDEOIO_H=yes
271.  !HAVE_TERMIOS_H=yes
272.  HAVE_THREADS=yes
273.  HAVE_TRUNC=yes
274.  HAVE_TRUNCF=yes
275.  !HAVE_VFP_ARGS=yes
276.  HAVE_VIRTUALALLOC=yes
277.  HAVE_WINSOCK2_H=yes
278.  !HAVE_XFORM_ASM=yes
```

```
279.    !HAVE_XMM_CLOBBERS=yes
280.    HAVE_YASM=yes
281.    #CONFIG_
282.    CONFIG_BSFS=yes
283.    CONFIG_DECODERS=yes
284.    CONFIG_DEMUXERS=yes
285.    CONFIG_ENCODERS=yes
286.    CONFIG_FILTERS=yes
287.    !CONFIG_HWACCELS=yes
288.    CONFIG_INDEVS=yes
289.    CONFIG_MUXERS=yes
290.    CONFIG_OUTDEVS=yes
291.    CONFIG_PARSERS=yes
292.    CONFIG_PROTOCOLS=yes
293.    CONFIG_FFPLAY=yes
294.    CONFIG_FFPROBE=yes
295.    !CONFIG_FFSERVER=yes
296.    CONFIG_FFMPEG=yes
297.    !CONFIG_AVPLAY=yes
298.    !CONFIG_AVPROBE=yes
299.    !CONFIG_AVSERVER=yes
300.    CONFIG_AANDCT=yes
301.    CONFIG_AC3DSP=yes
302.    CONFIG_AVCODEC=yes
303.    CONFIG_AVDEVICE=yes
304.    CONFIG_AVFILTER=yes
305.    CONFIG_AVFORMAT=yes
306.    !CONFIG_AVISYNTH=yes
307.    !CONFIG_BZLIB=yes
308.    !CONFIG_CRYSTALHD=yes
309.    CONFIG_DCT=yes
310.    !CONFIG_DOC=yes
311.    CONFIG_DWT=yes
312.    !CONFIG_DXVA2=yes
313.    CONFIG_FASTDIV=yes
314.    CONFIG_FFT=yes
315.    !CONFIG_FREI0R=yes
316.    !CONFIG_GNUTLS=yes
317.    CONFIG_GOLOMB=yes
318.    !CONFIG_GPL=yes
319.    !CONFIG_GRAY=yes
320.    CONFIG_H264CHROMA=yes
321.    CONFIG_H264DSP=yes
322.    CONFIG_H264PRED=yes
323.    !CONFIG_HARDCODED_TABLES=yes
324.    CONFIG_HUFFMAN=yes
325.    !CONFIG_LIBAACPLUS=yes
326.    !CONFIG_LIBASS=yes
327.    !CONFIG_LIBCDIO=yes
328.    !CONFIG_LIBCELT=yes
329.    !CONFIG_LIBDC1394=yes
330.    !CONFIG_LIBDIRAC=yes
331.    !CONFIG_LIBFAAC=yes
332.    !CONFIG_LIBFREETYPE=yes
333.    !CONFIG_LIBGSM=yes
334.    !CONFIG_LIBMODPLUG=yes
335.    !CONFIG_LIBMP3LAME=yes
336.    !CONFIG_LIBNUT=yes
337.    !CONFIG_LIBOPENCORE_AMRNB=yes
338.    !CONFIG_LIBOPENCORE_AMRWB=yes
339.    !CONFIG_LIBOPENCV=yes
340.    !CONFIG_LIBOPENJPEG=yes
341.    !CONFIG_LIBPULSE=yes
342.    !CONFIG_LIBRTMP=yes
343.    !CONFIG_LIBSCHROEDINGER=yes
344.    !CONFIG_LIBSPEEX=yes
345.    !CONFIG_LIBSTAGEFRIGHT_H264=yes
346.    !CONFIG_LIBTHEORA=yes
347.    !CONFIG_LIBUTVIDEO=yes
348.    !CONFIG_LIBV4L2=yes
349.    !CONFIG_LIBVO_AACENC=yes
350.    !CONFIG_LIBVO_AMRWBENC=yes
351.    !CONFIG_LIBVORBIS=yes
352.    !CONFIG_LIBVPX=yes
353.    !CONFIG_LIBX264=yes
354.    !CONFIG_LIBXAVS=yes
355.    !CONFIG_LIBXVID=yes
356.    CONFIG_LPC=yes
357.    CONFIG_LSP=yes
358.    CONFIG_MDCT=yes
359.    CONFIG_MEMALIGN_HACK=yes
360.    !CONFIG_MLIB=yes
361.    CONFIG_MPEGAUDIODSP=yes
362.    CONFIG_NETWORK=yes
363.    !CONFIG_NONFREE=yes
364.    !CONFIG_OPENAL=yes
365.    !CONFIG_OPENSSL=yes
366.    !CONFIG_PIC=yes
367.    !CONFIG_POSTPROC=yes
368.    CONFIG_RDFT=yes
369.    CONFIG_RTPDEC=yes
```

```
370.   !CONFIG_RUNTIME_CPUDETECT=yes
371.   CONFIG_SAFE_BITSTREAM_READER=yes
372.   !CONFIG_SHARED=yes
373.   CONFIG_SINEWIN=yes
374.   !CONFIG_SMALL=yes
375.   !CONFIG_SRAM=yes
376.   CONFIG_STATIC=yes
377.   CONFIG_SWRESAMPLE=yes
378.   CONFIG_SWSCALE=yes
379.   CONFIG_SWSCALE_ALPHA=yes
380.   !CONFIG_THUMB=yes
381.   !CONFIG_VAAPI=yes
382.   !CONFIG_VDA=yes
383.   !CONFIG_VDPAU=yes
384.   !CONFIG_VERSION3=yes
385.   !CONFIG_X11GRAB=yes
386.   CONFIG_ZLIB=yes
387.   CONFIG_AVUTIL=yes
388.   !CONFIG_GPLV3=yes
389.   !CONFIG_LGPLV3=yes
390.   CONFIG_AAC_ADTSTOASC_BSF=yes
391.   CONFIG_CHOMP_BSF=yes
392.   CONFIG_DUMP_EXTRADATA_BSF=yes
393.   CONFIG_H264_MP4TOANNEXB_BSF=yes
394.   CONFIG_IMX_DUMP_HEADER_BSF=yes
395.   CONFIG_MJPEG2JPEG_BSF=yes
396.   CONFIG_MJPEGA_DUMP_HEADER_BSF=yes
397.   CONFIG_MP3_HEADER_COMPRESS_BSF=yes
398.   CONFIG_MP3_HEADER_DECOMPRESS_BSF=yes
399.   CONFIG_MOV2TEXTSUB_BSF=yes
400.   CONFIG_NOISE_BSF=yes
401.   CONFIG_REMOVE_EXTRADATA_BSF=yes
402.   CONFIG_TEXT2MOVSUB_BSF=yes
403.   CONFIG_AASC_DECODER=yes
404.   CONFIG_AMV_DECODER=yes
405.   CONFIG_ANM_DECODER=yes
406.   CONFIG_ANSI_DECODER=yes
407.   CONFIG_ASV1_DECODER=yes
408.   CONFIG_ASV2_DECODER=yes
409.   CONFIG_AURA_DECODER=yes
410.   CONFIG_AURA2_DECODER=yes
411.   CONFIG_AVRP_DECODER=yes
412.   CONFIG_AVS_DECODER=yes
413.   CONFIG_BETHSOFTVID_DECODER=yes
414.   CONFIG_BFI_DECODER=yes
415.   CONFIG_BINK_DECODER=yes
416.   CONFIG_BMP_DECODER=yes
417.   CONFIG_BMV_VIDEO_DECODER=yes
418.   CONFIG_C93_DECODER=yes
419.   CONFIG_CAVS_DECODER=yes
420.   CONFIG_CDGRAPHICS_DECODER=yes
421.   CONFIG_CINEPAK_DECODER=yes
422.   CONFIG_CLJR_DECODER=yes
423.   CONFIG_CSCD_DECODER=yes
424.   CONFIG_CYUV_DECODER=yes
425.   CONFIG_DFA_DECODER=yes
426.   CONFIG_DIRAC_DECODER=yes
427.   CONFIG_DNXHD_DECODER=yes
428.   CONFIG_DPX_DECODER=yes
429.   CONFIG_DSICINVIDEO_DECODER=yes
430.   CONFIG_DVVIDEO_DECODER=yes
431.   CONFIG_DXA_DECODER=yes
432.   CONFIG_DXTORY_DECODER=yes
433.   CONFIG_EACMV_DECODER=yes
434.   CONFIG_EAMAD_DECODER=yes
435.   CONFIG_EATGQ_DECODER=yes
436.   CONFIG_EATGV_DECODER=yes
437.   CONFIG_EATQI_DECODER=yes
438.   CONFIG_EIGHTBPS_DECODER=yes
439.   CONFIG_EIGHTSVX_EXP_DECODER=yes
440.   CONFIG_EIGHTSVX_FIB_DECODER=yes
441.   CONFIG_ESCAPE124_DECODER=yes
442.   CONFIG_ESCAPE130_DECODER=yes
443.   CONFIG_FFV1_DECODER=yes
444.   CONFIG_FFVHUFF_DECODER=yes
445.   CONFIG_FLASHSV_DECODER=yes
446.   CONFIG_FLASHSV2_DECODER=yes
447.   CONFIG_FLIC_DECODER=yes
448.   CONFIG_FLV_DECODER=yes
449.   CONFIG_FOURXM_DECODER=yes
450.   CONFIG_FRAPS_DECODER=yes
451.   CONFIG_FRWU_DECODER=yes
452.   CONFIG_GIF_DECODER=yes
453.   CONFIG_H261_DECODER=yes
454.   CONFIG_H263_DECODER=yes
455.   CONFIG_H263I_DECODER=yes
456.   CONFIG_H264_DECODER=yes
457.   #此处省略若干条…
458.   CONFIG_RTMP_PROTOCOL=yes
459.   CONFIG_RTMPT_PROTOCOL=yes
460.   CONFIG_RTMPE_PROTOCOL=yes
```

```
461.  CONFIG_RTMPTE_PROTOCOL=yes
462.  CONFIG_RTMPS_PROTOCOL=yes
463.  CONFIG_RTP_PROTOCOL=yes
464.  CONFIG_TCP_PROTOCOL=yes
465.  !CONFIG_TLS_PROTOCOL=yes
466.  CONFIG_UDP_PROTOCOL=yes
467.  #Test
468.  ACODEC_TESTS=ac3_fixed adpcm_adx adpcm_ima_qt adpcm_ima_wav adpcm_ms adpcm_swf adpcm_yam alac aref flac g722 g723_1 g726 mp2 pcm_alaw
      m_f32be pcm_f32le pcm_f64be pcm_f64le pcm_mulaw pcm_s16be pcm_s16le pcm_s24be pcm_s24daud pcm_s24le pcm_s32be pcm_s32le pcm_s8 pcm_u8
      av1 wmav2
469.  VCODEC_TESTS=amv asv1 asv2 cljr dnxhd_1080i dnxhd_720p dnxhd_720p_10bit dnxhd_720p_rd dv dv50 dv_411 error ffv1 flashsv flashsv2 flv
      1 h263 h263p huffyuv jpeg2000 jpegls ljpeg mjpeg mpeg mpeg1b mpeg2 mpeg2_422 mpeg2_idct_int mpeg2_ilace mpeg2_ivlc_qprd mpeg2thread m
      2thread_ilace mpeg4 mpeg4_adap mpeg4_qpel mpeg4_qprd mpeg4adv mpeg4nr mpeg4thread mpng msmpeg4 msmpeg4v2 msvideo1 prores qtrle qtrleg
       rc rgb roq rv10 rv20 snow snowll svq1 v210 vref wmv1 wmv2 yuv zlib zmbv
470.  LAVF_TESTS=aiff alaw asf au avi bmp caf dpx dv_fmt ffm flv_fmt gif gxf jpg mkv mmf mov mpg mulaw mxf mxf_d10 nut ogg pbmpipe pcx pgm
      pipe pixfmt png ppm ppmpipe rm rso sgi sox swf tga tiff ts voc voc_s16 wav wtv yuv4mpeg
471.  LAVFI_TESTS=crop crop_scale crop_scale_vflip crop_vflip null pixdesc pixfmts_copy pixfmts_crop pixfmts_hflip pixfmts_null pixfmts_pad
      xfmts_scale pixfmts_vflip scale200 scale500 vflip vflip_crop vflip_vflip
472.  SEEK_TESTS=seek_ac3_rm seek_adpcm_ima_wav seek_adpcm_ms_wav seek_adpcm_qt_aiff seek_adpcm_swf_flv seek_adpcm_yam_wav seek_alac_m4a se
      asv1_avi seek_asv2_avi seek_dnxhd_1080i_mov seek_dnxhd_720p_dnxhd seek_dnxhd_720p_rd_dnxhd seek_dv411_dv seek_dv50_dv seek_dv_dv seek
      ror_mpeg4_adv_avi seek_ffv1_avi seek_flac_flac seek_flashsv_flv seek_flv_flv seek_g726_wav seek_h261_avi seek_h263_avi seek_h263p_avi
      ek_huffyuv_avi seek_image_bmp seek_image_jpg seek_image_pcx seek_image_pgm seek_image_ppm seek_image_sgi seek_image_tga seek_image_ti
      seek_jpegls_avi seek_lavf_aif seek_lavf_al seek_lavf_asf seek_lavf_au seek_lavf_avi seek_lavf_dv seek_lavf_ffm seek_lavf_flv seek_lav
      if seek_lavf_gxf seek_lavf_mkv seek_lavf_mmf seek_lavf_mov seek_lavf_mpg seek_lavf_mxf seek_lavf_mxf_d10 seek_lavf_nut seek_lavf_ogg
      k_lavf_rm seek_lavf_swf seek_lavf_ts seek_lavf_ul seek_lavf_voc seek_lavf_wav seek_lavf_wtv seek_lavf_y4m seek_ljpeg_avi seek_mjpeg_a
      seek_mp2_mp2 seek_mpeg1_mpg seek_mpeg1b_mpg seek_mpeg2_422_mpg seek_mpeg2_idct_int_mpg seek_mpeg2i_mpg seek_mpeg2ivlc_qprd_mpg seek_m
      2reuse_mpg seek_mpeg2thread_mpg seek_mpeg2threadivlc_mpg seek_mpeg4_adap_avi seek_mpeg4_adv_avi seek_mpeg4_nr_avi seek_mpeg4_qprd_avi
      ek_mpeg4_rc_avi seek_mpeg4_thread_avi seek_msmpeg4_avi seek_msmpeg4v2_avi seek_odivx_mp4 seek_pbmpipe_pbm seek_pcm_alaw_wav seek_pcm_
      be_au seek_pcm_f32le_wav seek_pcm_f64be_au seek_pcm_f64le_wav seek_pcm_mulaw_wav seek_pcm_s16be_mov seek_pcm_s16le_wav seek_pcm_s24be
      v seek_pcm_s24daud_302 seek_pcm_s24le_wav seek_pcm_s32be_mov seek_pcm_s32le_wav seek_pcm_s8_mov seek_pcm_u8_wav seek_pgmpipe_pgm seek
      mpipe_ppm seek_rgb_avi seek_roqav_roq seek_rv10_rm seek_rv20_rm seek_snow53_avi seek_snow_avi seek_svq1_mov seek_wmav1_asf seek_wmav2
      f seek_wmv1_avi seek_wmv2_avi seek_yuv_avi
473.  endif # FFMPEG_CONFIG_MAK
```

config.mak代码大致可以分为以下几类信息：

　　（1）各种路径（prefix等）

　　（2）工具集（arch、cc、ld、yasm等）

　　（3）参数集（cppflag、cflag、ldflag等）

　　（4）前缀后缀（.a、.dll、.exe等）

　　（5）类库版本（libavXXX_version信息）

　　（6）组件配置。这一部分信息使用{组件名}=yes的方式进行书写。对于不支持的组件，则在该组件所在行的前面标记上"！"号（感叹号似乎在Makefile语法中并没有什么特殊的用意，此处可能仅仅是作为一种标记？）。这一部分可以分为3类信息：

　　　　a）

　　　　　　ARCH_信息

　　　　b）

　　　　　　HAVE_信息

　　　　c）

　　　　　　CONFIG_信息。这一部分内容最多，将近有1000行。

　　（7）Test信息（测试组件的结果？还没研究）

libavXXXX/Makefile

libavXXXX/Makefile指的是FFmpeg类库（libavformat、libavcodec、libavutil等）所在的文件夹下的Makefile。例如libavformat文件夹下的Makefile代码如下所示。

```python
1.   # FFmpeg Libavformat Makefile
2.   #
3.   # 注释：雷霄骅
4.   # leixiaohua1020@126.com
5.   # http://blog.csdn.net/leixiaohua1020
6.   #
7.   # FFmpeg中libavformat的Makefile。
8.   # 注意该Makefile并没有定义类库的编译规则（这一部分统一在library.mak中完成）。
9.   # 该Makefile中只是赋值了几个重要的字符串：
10.  #      NAME, FFLIBS, HEADERS, OBJS, OBJS-yes
11.
12.  #重要：包含了configure信息，位于上一级目录
13.  include $(SUBDIR)../config.mak
14.  #名称
15.  NAME = avformat
16.  #用到的库？
17.  FFLIBS = avcodec avutil
18.  #SDK中的头文件
19.  HEADERS = avformat.h avio.h version.h
20.
21.  #OBJS存储的是必须的目标文件
22.  OBJS = allformats.o           \
23.         cutils.o               \
24.         id3v1.o                \
25.         id3v2.o                \
26.         metadata.o             \
```

```
26.            metadata.o           \
27.            options.o            \
28.            os_support.o         \
29.            sdp.o                \
30.            seek.o               \
31.            utils.o              \
32.
33.  #OBJS-yes存储的是可选的目标文件
34.  OBJS-$(CONFIG_NETWORK)                    += network.o
35.
36.  # muxers/demuxers
37.  OBJS-$(CONFIG_A64_MUXER)                  += a64.o
38.  OBJS-$(CONFIG_AAC_DEMUXER)                += aacdec.o rawdec.o
39.  OBJS-$(CONFIG_AC3_DEMUXER)                += ac3dec.o rawdec.o
40.  OBJS-$(CONFIG_AC3_MUXER)                  += rawenc.o
41.  OBJS-$(CONFIG_ACT_DEMUXER)                += act.o
42.  OBJS-$(CONFIG_ADF_DEMUXER)                += bintext.o sauce.o
43.  OBJS-$(CONFIG_ADX_DEMUXER)                += adxdec.o
44.  OBJS-$(CONFIG_ADX_MUXER)                  += rawenc.o
45.  OBJS-$(CONFIG_ADTS_MUXER)                 += adtsenc.o
46.  OBJS-$(CONFIG_AEA_DEMUXER)                += aea.o pcm.o
47.  OBJS-$(CONFIG_AIFF_DEMUXER)               += aiffdec.o riff.o pcm.o isom.o
48.  OBJS-$(CONFIG_AIFF_MUXER)                 += aiffenc.o riff.o isom.o
49.  OBJS-$(CONFIG_AMR_DEMUXER)                += amr.o
50.  OBJS-$(CONFIG_AMR_MUXER)                  += amr.o
51.  OBJS-$(CONFIG_ANM_DEMUXER)                += anm.o
52.  OBJS-$(CONFIG_APC_DEMUXER)                += apc.o
53.  OBJS-$(CONFIG_APE_DEMUXER)                += ape.o apetag.o
54.  OBJS-$(CONFIG_APPLEHTTP_DEMUXER)          += applehttp.o
55.  OBJS-$(CONFIG_ASF_DEMUXER)                += asfdec.o asf.o asfcrypt.o \
56.                                               riff.o avlanguage.o
57.  OBJS-$(CONFIG_ASF_MUXER)                  += asfenc.o asf.o riff.o
58.  OBJS-$(CONFIG_ASS_DEMUXER)                += assdec.o
59.  OBJS-$(CONFIG_ASS_MUXER)                  += assenc.o
60.  OBJS-$(CONFIG_AU_DEMUXER)                 += au.o pcm.o
61.  OBJS-$(CONFIG_AU_MUXER)                   += au.o
62.  OBJS-$(CONFIG_AVI_DEMUXER)                += avidec.o riff.o
63.  OBJS-$(CONFIG_AVI_MUXER)                  += avienc.o riff.o
64.  OBJS-$(CONFIG_AVISYNTH)                   += avisynth.o
65.  OBJS-$(CONFIG_AVM2_MUXER)                 += swfenc.o
66.  OBJS-$(CONFIG_AVS_DEMUXER)                += avs.o vocdec.o voc.o
67.  OBJS-$(CONFIG_BETHSOFTVID_DEMUXER)        += bethsoftvid.o
68.  OBJS-$(CONFIG_BFI_DEMUXER)                += bfi.o
69.  OBJS-$(CONFIG_BINK_DEMUXER)               += bink.o
70.  OBJS-$(CONFIG_BINTEXT_DEMUXER)            += bintext.o sauce.o
71.  OBJS-$(CONFIG_BIT_DEMUXER)                += bit.o
72.  OBJS-$(CONFIG_BIT_MUXER)                  += bit.o
73.  OBJS-$(CONFIG_BMV_DEMUXER)                += bmv.o
74.  OBJS-$(CONFIG_C93_DEMUXER)                += c93.o vocdec.o voc.o
75.  OBJS-$(CONFIG_CAF_DEMUXER)                += cafdec.o caf.o mov.o mov_chan.o \
76.                                               riff.o isom.o
77.  OBJS-$(CONFIG_CAF_MUXER)                  += cafenc.o caf.o riff.o isom.o
78.  OBJS-$(CONFIG_CAVSVIDEO_DEMUXER)          += cavsvideodec.o rawdec.o
79.  OBJS-$(CONFIG_CAVSVIDEO_MUXER)            += rawenc.o
80.  OBJS-$(CONFIG_CDG_DEMUXER)                += cdg.o
81.  OBJS-$(CONFIG_CRC_MUXER)                  += crcenc.o
82.  OBJS-$(CONFIG_DAUD_DEMUXER)               += daud.o
83.  OBJS-$(CONFIG_DAUD_MUXER)                 += daud.o
84.  OBJS-$(CONFIG_DFA_DEMUXER)                += dfa.o
85.  OBJS-$(CONFIG_DIRAC_DEMUXER)              += diracdec.o rawdec.o
86.  OBJS-$(CONFIG_DIRAC_MUXER)                += rawenc.o
87.  OBJS-$(CONFIG_DNXHD_DEMUXER)              += dnxhddec.o rawdec.o
88.  OBJS-$(CONFIG_DNXHD_MUXER)                += rawenc.o
89.  OBJS-$(CONFIG_DSICIN_DEMUXER)             += dsicin.o
90.  OBJS-$(CONFIG_DTS_DEMUXER)                += dtsdec.o rawdec.o
91.  OBJS-$(CONFIG_DTS_MUXER)                  += rawenc.o
92.  OBJS-$(CONFIG_DV_DEMUXER)                 += dv.o
93.  OBJS-$(CONFIG_DV_MUXER)                   += dvenc.o
94.  OBJS-$(CONFIG_DXA_DEMUXER)                += dxa.o riff.o
95.  OBJS-$(CONFIG_EA_CDATA_DEMUXER)           += eacdata.o
96.  OBJS-$(CONFIG_EA_DEMUXER)                 += electronicarts.o
97.  OBJS-$(CONFIG_EAC3_DEMUXER)               += ac3dec.o rawdec.o
98.  OBJS-$(CONFIG_EAC3_MUXER)                 += rawenc.o
99.  OBJS-$(CONFIG_FFM_DEMUXER)                += ffmdec.o
100. OBJS-$(CONFIG_FFM_MUXER)                  += ffmenc.o
101. OBJS-$(CONFIG_FFMETADATA_DEMUXER)         += ffmetadec.o
102. OBJS-$(CONFIG_FFMETADATA_MUXER)           += ffmetaenc.o
103. OBJS-$(CONFIG_FILMSTRIP_DEMUXER)          += filmstripdec.o
104. OBJS-$(CONFIG_FILMSTRIP_MUXER)            += filmstripenc.o
105. OBJS-$(CONFIG_FLAC_DEMUXER)               += flacdec.o rawdec.o \
106.                                              oggparsevorbis.o \
107.                                              vorbiscomment.o
108. OBJS-$(CONFIG_FLAC_MUXER)                 += flacenc.o flacenc_header.o \
109.                                              vorbiscomment.o
110. OBJS-$(CONFIG_FLIC_DEMUXER)               += flic.o
111. OBJS-$(CONFIG_FLV_DEMUXER)                += flvdec.o
112. OBJS-$(CONFIG_FLV_MUXER)                  += flvenc.o avc.o
113. OBJS-$(CONFIG_FOURXM_DEMUXER)             += 4xm.o
114. OBJS-$(CONFIG_FRAMECRC_MUXER)             += framecrcenc.o
115. OBJS-$(CONFIG_FRAMEMD5_MUXER)             += md5enc.o
116. OBJS-$(CONFIG_GIF_MUXER)                  += gif.o
117. OBJS-$(CONFIG_GSM_DEMUXER)                += gsmdec.o
```

```
118.  OBJS-$(CONFIG_GXF_DEMUXER)                 += gxf.o
119.  OBJS-$(CONFIG_GXF_MUXER)                   += gxfenc.o audiointerleave.o
120.  OBJS-$(CONFIG_G722_DEMUXER)                += rawdec.o
121.  OBJS-$(CONFIG_G722_MUXER)                  += rawenc.o
122.  OBJS-$(CONFIG_G723_1_DEMUXER)              += g723_1.o
123.  OBJS-$(CONFIG_G723_1_MUXER)                += rawenc.o
124.  OBJS-$(CONFIG_G729_DEMUXER)                += g729dec.o
125.  OBJS-$(CONFIG_H261_DEMUXER)                += h261dec.o rawdec.o
126.  OBJS-$(CONFIG_H261_MUXER)                  += rawenc.o
127.  OBJS-$(CONFIG_H263_DEMUXER)                += h263dec.o rawdec.o
128.  OBJS-$(CONFIG_H263_MUXER)                  += rawenc.o
129.  OBJS-$(CONFIG_H264_DEMUXER)                += h264dec.o rawdec.o
130.  OBJS-$(CONFIG_H264_MUXER)                  += rawenc.o
131.  OBJS-$(CONFIG_ICO_DEMUXER)                 += icodec.o
132.  OBJS-$(CONFIG_IDCIN_DEMUXER)               += idcin.o
133.  OBJS-$(CONFIG_IDF_DEMUXER)                 += bintext.o
134.  OBJS-$(CONFIG_IFF_DEMUXER)                 += iff.o
135.  OBJS-$(CONFIG_IMAGE2_DEMUXER)              += img2.o
136.  OBJS-$(CONFIG_IMAGE2_MUXER)                += img2.o
137.  OBJS-$(CONFIG_IMAGE2PIPE_DEMUXER)          += img2.o
138.  OBJS-$(CONFIG_IMAGE2PIPE_MUXER)            += img2.o
139.  OBJS-$(CONFIG_INGENIENT_DEMUXER)           += ingenientdec.o rawdec.o
140.  OBJS-$(CONFIG_IPMOVIE_DEMUXER)             += ipmovie.o
141.  OBJS-$(CONFIG_ISS_DEMUXER)                 += iss.o
142.  OBJS-$(CONFIG_IV8_DEMUXER)                 += iv8.o
143.  OBJS-$(CONFIG_IVF_DEMUXER)                 += ivfdec.o riff.o
144.  OBJS-$(CONFIG_IVF_MUXER)                   += ivfenc.o
145.  OBJS-$(CONFIG_JV_DEMUXER)                  += jvdec.o
146.  OBJS-$(CONFIG_LATM_DEMUXER)                += rawdec.o
147.  OBJS-$(CONFIG_LATM_MUXER)                  += latmenc.o
148.  OBJS-$(CONFIG_LMLM4_DEMUXER)               += lmlm4.o
149.  OBJS-$(CONFIG_LOAS_DEMUXER)                += loasdec.o
150.  OBJS-$(CONFIG_LXF_DEMUXER)                 += lxfdec.o
151.  OBJS-$(CONFIG_M4V_DEMUXER)                 += m4vdec.o rawdec.o
152.  OBJS-$(CONFIG_M4V_MUXER)                   += rawenc.o
153.  OBJS-$(CONFIG_MATROSKA_DEMUXER)            += matroskadec.o matroska.o \
154.                                                riff.o isom.o rmdec.o rm.o
155.  OBJS-$(CONFIG_MATROSKA_MUXER)              += matroskaenc.o matroska.o \
156.                                                riff.o isom.o avc.o \
157.                                                flacenc_header.o avlanguage.o
158.  OBJS-$(CONFIG_MD5_MUXER)                   += md5enc.o
159.  OBJS-$(CONFIG_MICRODVD_DEMUXER)            += microdvddec.o
160.  OBJS-$(CONFIG_MICRODVD_MUXER)              += microdvdenc.o rawenc.o
161.  OBJS-$(CONFIG_MJPEG_DEMUXER)               += rawdec.o
162.  OBJS-$(CONFIG_MJPEG_MUXER)                 += rawenc.o
163.  OBJS-$(CONFIG_MLP_DEMUXER)                 += rawdec.o
164.  OBJS-$(CONFIG_MLP_MUXER)                   += rawenc.o
165.  OBJS-$(CONFIG_MM_DEMUXER)                  += mm.o
166.  OBJS-$(CONFIG_MMF_DEMUXER)                 += mmf.o pcm.o
167.  OBJS-$(CONFIG_MMF_MUXER)                   += mmf.o riff.o
168.  OBJS-$(CONFIG_MOV_DEMUXER)                 += mov.o riff.o isom.o mov_chan.o
169.  OBJS-$(CONFIG_MOV_MUXER)                   += movenc.o riff.o isom.o avc.o \
170.                                                movenchint.o rtpenc_chain.o \
171.                                                mov_chan.o
172.  OBJS-$(CONFIG_MP2_MUXER)                   += mp3enc.o rawenc.o
173.  OBJS-$(CONFIG_MP3_DEMUXER)                 += mp3dec.o
174.  OBJS-$(CONFIG_MP3_MUXER)                   += mp3enc.o rawenc.o id3v2enc.o
175.  OBJS-$(CONFIG_MPC_DEMUXER)                 += mpc.o apetag.o
176.  OBJS-$(CONFIG_MPC8_DEMUXER)                += mpc8.o
177.  OBJS-$(CONFIG_MPEG1SYSTEM_MUXER)           += mpegenc.o
178.  OBJS-$(CONFIG_MPEG1VCD_MUXER)              += mpegenc.o
179.  OBJS-$(CONFIG_MPEG2DVD_MUXER)              += mpegenc.o
180.  OBJS-$(CONFIG_MPEG2VOB_MUXER)              += mpegenc.o
181.  OBJS-$(CONFIG_MPEG2SVCD_MUXER)             += mpegenc.o
182.  OBJS-$(CONFIG_MPEG1VIDEO_MUXER)            += rawenc.o
183.  OBJS-$(CONFIG_MPEG2VIDEO_MUXER)            += rawenc.o
184.  OBJS-$(CONFIG_MPEGPS_DEMUXER)              += mpeg.o
185.  OBJS-$(CONFIG_MPEGTS_DEMUXER)              += mpegts.o isom.o
186.  OBJS-$(CONFIG_MPEGTS_MUXER)                += mpegtsenc.o adtsenc.o
187.  OBJS-$(CONFIG_MPEGVIDEO_DEMUXER)           += mpegvideodec.o rawdec.o
188.  OBJS-$(CONFIG_MPJPEG_MUXER)                += mpjpeg.o
189.  OBJS-$(CONFIG_MSNWC_TCP_DEMUXER)           += msnwc_tcp.o
190.  OBJS-$(CONFIG_MTV_DEMUXER)                 += mtv.o
191.  OBJS-$(CONFIG_MVI_DEMUXER)                 += mvi.o
192.  OBJS-$(CONFIG_MXF_DEMUXER)                 += mxfdec.o mxf.o
193.  OBJS-$(CONFIG_MXF_MUXER)                   += mxfenc.o mxf.o audiointerleave.o
194.  OBJS-$(CONFIG_MXG_DEMUXER)                 += mxg.o
195.  OBJS-$(CONFIG_NC_DEMUXER)                  += ncdec.o
196.  OBJS-$(CONFIG_NSV_DEMUXER)                 += nsvdec.o
197.  OBJS-$(CONFIG_NULL_MUXER)                  += nullenc.o
198.  OBJS-$(CONFIG_NUT_DEMUXER)                 += nutdec.o nut.o riff.o
199.  OBJS-$(CONFIG_NUT_MUXER)                   += nutenc.o nut.o riff.o
200.  OBJS-$(CONFIG_NUV_DEMUXER)                 += nuv.o riff.o
201.  OBJS-$(CONFIG_OGG_DEMUXER)                 += oggdec.o           \
202.                                                oggparsecelt.o   \
203.                                                oggparsedirac.o  \
204.                                                oggparseflac.o   \
205.                                                oggparseogm.o    \
206.                                                oggparseskeleton.o \
207.                                                oggparsespeex.o  \
208.                                                oggparsetheora.o \
```

```
209.                                          oggparsevorbis.o \
210.                                          riff.o \
211.                                          vorbiscomment.o
212.  OBJS-$(CONFIG_OGG_MUXER)            += oggenc.o \
213.                                          vorbiscomment.o
214.  OBJS-$(CONFIG_OMA_DEMUXER)          += omadec.o pcm.o oma.o
215.  OBJS-$(CONFIG_OMA_MUXER)            += omaenc.o rawenc.o oma.o id3v2enc.o
216.  OBJS-$(CONFIG_PCM_ALAW_DEMUXER)     += pcmdec.o pcm.o rawdec.o
217.  OBJS-$(CONFIG_PCM_ALAW_MUXER)       += pcmenc.o rawenc.o
218.  OBJS-$(CONFIG_PCM_F32BE_DEMUXER)    += pcmdec.o pcm.o rawdec.o
219.  OBJS-$(CONFIG_PCM_F32BE_MUXER)      += pcmenc.o rawenc.o
220.  OBJS-$(CONFIG_PCM_F32LE_DEMUXER)    += pcmdec.o pcm.o rawdec.o
221.  OBJS-$(CONFIG_PCM_F32LE_MUXER)      += pcmenc.o rawenc.o
222.  OBJS-$(CONFIG_PCM_F64BE_DEMUXER)    += pcmdec.o pcm.o rawdec.o
223.  OBJS-$(CONFIG_PCM_F64BE_MUXER)      += pcmenc.o rawenc.o
224.  OBJS-$(CONFIG_PCM_F64LE_DEMUXER)    += pcmdec.o pcm.o rawdec.o
225.  OBJS-$(CONFIG_PCM_F64LE_MUXER)      += pcmenc.o rawenc.o
226.  OBJS-$(CONFIG_PCM_MULAW_DEMUXER)    += pcmdec.o pcm.o rawdec.o
227.  OBJS-$(CONFIG_PCM_MULAW_MUXER)      += pcmenc.o rawenc.o
228.  OBJS-$(CONFIG_PCM_S16BE_DEMUXER)    += pcmdec.o pcm.o rawdec.o
229.  OBJS-$(CONFIG_PCM_S16BE_MUXER)      += pcmenc.o rawenc.o
230.  OBJS-$(CONFIG_PCM_S16LE_DEMUXER)    += pcmdec.o pcm.o rawdec.o
231.  OBJS-$(CONFIG_PCM_S16LE_MUXER)      += pcmenc.o rawenc.o
232.  OBJS-$(CONFIG_PCM_S24BE_DEMUXER)    += pcmdec.o pcm.o rawdec.o
233.  OBJS-$(CONFIG_PCM_S24BE_MUXER)      += pcmenc.o rawenc.o
234.  OBJS-$(CONFIG_PCM_S24LE_DEMUXER)    += pcmdec.o pcm.o rawdec.o
235.  OBJS-$(CONFIG_PCM_S24LE_MUXER)      += pcmenc.o rawenc.o
236.  OBJS-$(CONFIG_PCM_S32BE_DEMUXER)    += pcmdec.o pcm.o rawdec.o
237.  OBJS-$(CONFIG_PCM_S32BE_MUXER)      += pcmenc.o rawenc.o
238.  OBJS-$(CONFIG_PCM_S32LE_DEMUXER)    += pcmdec.o pcm.o rawdec.o
239.  OBJS-$(CONFIG_PCM_S32LE_MUXER)      += pcmenc.o rawenc.o
240.  OBJS-$(CONFIG_PCM_S8_DEMUXER)       += pcmdec.o pcm.o rawdec.o
241.  OBJS-$(CONFIG_PCM_S8_MUXER)         += pcmenc.o rawenc.o
242.  OBJS-$(CONFIG_PCM_U16BE_DEMUXER)    += pcmdec.o pcm.o rawdec.o
243.  OBJS-$(CONFIG_PCM_U16BE_MUXER)      += pcmenc.o rawenc.o
244.  OBJS-$(CONFIG_PCM_U16LE_DEMUXER)    += pcmdec.o pcm.o rawdec.o
245.  OBJS-$(CONFIG_PCM_U16LE_MUXER)      += pcmenc.o rawenc.o
246.  OBJS-$(CONFIG_PCM_U24BE_DEMUXER)    += pcmdec.o pcm.o rawdec.o
247.  OBJS-$(CONFIG_PCM_U24BE_MUXER)      += pcmenc.o rawenc.o
248.  OBJS-$(CONFIG_PCM_U24LE_DEMUXER)    += pcmdec.o pcm.o rawdec.o
249.  OBJS-$(CONFIG_PCM_U24LE_MUXER)      += pcmenc.o rawenc.o
250.  OBJS-$(CONFIG_PCM_U32BE_DEMUXER)    += pcmdec.o pcm.o rawdec.o
251.  OBJS-$(CONFIG_PCM_U32BE_MUXER)      += pcmenc.o rawenc.o
252.  OBJS-$(CONFIG_PCM_U32LE_DEMUXER)    += pcmdec.o pcm.o rawdec.o
253.  OBJS-$(CONFIG_PCM_U32LE_MUXER)      += pcmenc.o rawenc.o
254.  OBJS-$(CONFIG_PCM_U8_DEMUXER)       += pcmdec.o pcm.o rawdec.o
255.  OBJS-$(CONFIG_PCM_U8_MUXER)         += pcmenc.o rawenc.o
256.  OBJS-$(CONFIG_PMP_DEMUXER)          += pmpdec.o
257.  OBJS-$(CONFIG_PVA_DEMUXER)          += pva.o
258.  OBJS-$(CONFIG_QCP_DEMUXER)          += qcp.o
259.  OBJS-$(CONFIG_R3D_DEMUXER)          += r3d.o
260.  OBJS-$(CONFIG_RAWVIDEO_DEMUXER)     += rawvideodec.o rawdec.o
261.  OBJS-$(CONFIG_RAWVIDEO_MUXER)       += rawenc.o
262.  OBJS-$(CONFIG_RL2_DEMUXER)          += rl2.o
263.  OBJS-$(CONFIG_RM_DEMUXER)           += rmdec.o rm.o
264.  OBJS-$(CONFIG_RM_MUXER)             += rmenc.o rm.o
265.  OBJS-$(CONFIG_ROQ_DEMUXER)          += idroqdec.o
266.  OBJS-$(CONFIG_ROQ_MUXER)            += idroqenc.o rawenc.o
267.  OBJS-$(CONFIG_RSO_DEMUXER)          += rsodec.o rso.o pcm.o
268.  OBJS-$(CONFIG_RSO_MUXER)            += rsoenc.o rso.o
269.  OBJS-$(CONFIG_RPL_DEMUXER)          += rpl.o
270.  OBJS-$(CONFIG_RTP_MUXER)            += rtp.o             \
271.                                          rtpenc_aac.o     \
272.                                          rtpenc_latm.o    \
273.                                          rtpenc_amr.o     \
274.                                          rtpenc_h263.o    \
275.                                          rtpenc_mpv.o     \
276.                                          rtpenc.o         \
277.                                          rtpenc_h264.o \
278.                                          rtpenc_vp8.o  \
279.                                          rtpenc_xiph.o \
280.                                          avc.o
281.  OBJS-$(CONFIG_RTPDEC)               += rdt.o             \
282.                                          rtp.o             \
283.                                          rtpdec.o          \
284.                                          rtpdec_amr.o  \
285.                                          rtpdec_asf.o  \
286.                                          rtpdec_g726.o \
287.                                          rtpdec_h263.o \
288.                                          rtpdec_h264.o \
289.                                          rtpdec_latm.o \
290.                                          rtpdec_mpeg4.o \
291.                                          rtpdec_qcelp.o \
292.                                          rtpdec_qdm2.o \
293.                                          rtpdec_qt.o   \
294.                                          rtpdec_svq3.o \
295.                                          rtpdec_vp8.o  \
296.                                          rtpdec_xiph.o
297.  OBJS-$(CONFIG_RTSP_DEMUXER)         += rtsp.o rtspdec.o httpauth.o
298.  OBJS-$(CONFIG_RTSP_MUXER)           += rtsp.o rtspenc.o httpauth.o \
299.                                          rtpenc_chain.o
```

```makefile
300.    OBJS-$(CONFIG_SAP_DEMUXER)              += sapdec.o
301.    OBJS-$(CONFIG_SAP_MUXER)                += sapenc.o rtpenc_chain.o
302.    OBJS-$(CONFIG_SBG_DEMUXER)              += sbgdec.o
303.    OBJS-$(CONFIG_SDP_DEMUXER)              += rtsp.o
304.    OBJS-$(CONFIG_SEGAFILM_DEMUXER)         += segafilm.o
305.    OBJS-$(CONFIG_SEGMENT_MUXER)            += segment.o
306.    OBJS-$(CONFIG_SHORTEN_DEMUXER)          += rawdec.o
307.    OBJS-$(CONFIG_SIFF_DEMUXER)             += siff.o
308.    OBJS-$(CONFIG_SMACKER_DEMUXER)          += smacker.o
309.    OBJS-$(CONFIG_SMJPEG_DEMUXER)           += smjpegdec.o smjpeg.o
310.    OBJS-$(CONFIG_SMJPEG_MUXER)             += smjpegenc.o smjpeg.o
311.    OBJS-$(CONFIG_SOL_DEMUXER)              += sol.o pcm.o
312.    OBJS-$(CONFIG_SOX_DEMUXER)              += soxdec.o pcm.o
313.    OBJS-$(CONFIG_SOX_MUXER)                += soxenc.o
314.    OBJS-$(CONFIG_SPDIF_DEMUXER)            += spdif.o spdifdec.o
315.    OBJS-$(CONFIG_SPDIF_MUXER)              += spdif.o spdifenc.o
316.    OBJS-$(CONFIG_SRT_DEMUXER)              += srtdec.o
317.    OBJS-$(CONFIG_SRT_MUXER)                += rawenc.o
318.    OBJS-$(CONFIG_STR_DEMUXER)              += psxstr.o
319.    OBJS-$(CONFIG_SWF_DEMUXER)              += swfdec.o
320.    OBJS-$(CONFIG_SWF_MUXER)                += swfenc.o
321.    OBJS-$(CONFIG_THP_DEMUXER)              += thp.o
322.    OBJS-$(CONFIG_TIERTEXSEQ_DEMUXER)       += tiertexseq.o
323.    OBJS-$(CONFIG_MKVTIMESTAMP_V2_MUXER)    += mkvtimestamp_v2.o
324.    OBJS-$(CONFIG_TMV_DEMUXER)              += tmv.o
325.    OBJS-$(CONFIG_TRUEHD_DEMUXER)           += rawdec.o
326.    OBJS-$(CONFIG_TRUEHD_MUXER)             += rawenc.o
327.    OBJS-$(CONFIG_TTA_DEMUXER)              += tta.o
328.    OBJS-$(CONFIG_TTY_DEMUXER)              += tty.o sauce.o
329.    OBJS-$(CONFIG_TXD_DEMUXER)              += txd.o
330.    OBJS-$(CONFIG_VC1_DEMUXER)              += rawdec.o
331.    OBJS-$(CONFIG_VC1T_DEMUXER)             += vc1test.o
332.    OBJS-$(CONFIG_VC1T_MUXER)               += vc1testenc.o
333.    OBJS-$(CONFIG_VMD_DEMUXER)              += sierravmd.o
334.    OBJS-$(CONFIG_VOC_DEMUXER)              += vocdec.o voc.o
335.    OBJS-$(CONFIG_VOC_MUXER)                += vocenc.o voc.o
336.    OBJS-$(CONFIG_VQF_DEMUXER)              += vqf.o
337.    OBJS-$(CONFIG_W64_DEMUXER)              += wav.o riff.o pcm.o
338.    OBJS-$(CONFIG_WAV_DEMUXER)              += wav.o riff.o pcm.o
339.    OBJS-$(CONFIG_WAV_MUXER)                += wav.o riff.o
340.    OBJS-$(CONFIG_WC3_DEMUXER)              += wc3movie.o
341.    OBJS-$(CONFIG_WEBM_MUXER)               += matroskaenc.o matroska.o \
342.                                               riff.o isom.o avc.o \
343.                                               flacenc_header.o avlanguage.o
344.    OBJS-$(CONFIG_WSAUD_DEMUXER)            += westwood_aud.o
345.    OBJS-$(CONFIG_WSVQA_DEMUXER)            += westwood_vqa.o
346.    OBJS-$(CONFIG_WTV_DEMUXER)              += wtvdec.o wtv.o asfdec.o asf.o asfcrypt.o \
347.                                               avlanguage.o mpegts.o isom.o riff.o
348.    OBJS-$(CONFIG_WTV_MUXER)                += wtvenc.o wtv.o asf.o asfenc.o riff.o
349.    OBJS-$(CONFIG_WV_DEMUXER)               += wv.o apetag.o
350.    OBJS-$(CONFIG_XA_DEMUXER)               += xa.o
351.    OBJS-$(CONFIG_XBIN_DEMUXER)             += bintext.o sauce.o
352.    OBJS-$(CONFIG_XMV_DEMUXER)              += xmv.o riff.o
353.    OBJS-$(CONFIG_XWMA_DEMUXER)             += xwma.o riff.o
354.    OBJS-$(CONFIG_YOP_DEMUXER)              += yop.o
355.    OBJS-$(CONFIG_YUV4MPEGPIPE_MUXER)       += yuv4mpeg.o
356.    OBJS-$(CONFIG_YUV4MPEGPIPE_DEMUXER)     += yuv4mpeg.o
357.
358.    # external libraries
359.    OBJS-$(CONFIG_LIBMODPLUG_DEMUXER)       += libmodplug.o
360.    OBJS-$(CONFIG_LIBNUT_DEMUXER)           += libnut.o riff.o
361.    OBJS-$(CONFIG_LIBNUT_MUXER)             += libnut.o riff.o
362.
363.    # protocols I/O
364.    OBJS+= avio.o aviobuf.o
365.
366.    OBJS-$(CONFIG_APPLEHTTP_PROTOCOL)       += applehttpproto.o
367.    OBJS-$(CONFIG_CACHE_PROTOCOL)           += cache.o
368.    OBJS-$(CONFIG_CONCAT_PROTOCOL)          += concat.o
369.    OBJS-$(CONFIG_CRYPTO_PROTOCOL)          += crypto.o
370.    OBJS-$(CONFIG_FILE_PROTOCOL)            += file.o
371.    OBJS-$(CONFIG_GOPHER_PROTOCOL)          += gopher.o
372.    OBJS-$(CONFIG_HTTP_PROTOCOL)            += http.o httpauth.o
373.    OBJS-$(CONFIG_HTTPPROXY_PROTOCOL)       += http.o httpauth.o
374.    OBJS-$(CONFIG_HTTPS_PROTOCOL)           += http.o httpauth.o
375.    OBJS-$(CONFIG_MMSH_PROTOCOL)            += mmsh.o mms.o asf.o
376.    OBJS-$(CONFIG_MMST_PROTOCOL)            += mmst.o mms.o asf.o
377.    OBJS-$(CONFIG_MD5_PROTOCOL)             += md5proto.o
378.    OBJS-$(CONFIG_PIPE_PROTOCOL)            += file.o
379.
380.    # external or internal rtmp
381.    RTMP-OBJS-$(CONFIG_LIBRTMP)             = librtmp.o
382.    RTMP-OBJS-$(!CONFIG_LIBRTMP)            = rtmpproto.o rtmppkt.o
383.    OBJS-$(CONFIG_RTMP_PROTOCOL)            += $(RTMP-OBJS-yes)
384.
385.    OBJS-$(CONFIG_RTP_PROTOCOL)             += rtpproto.o
386.    OBJS-$(CONFIG_TCP_PROTOCOL)             += tcp.o
387.    OBJS-$(CONFIG_TLS_PROTOCOL)             += tls.o
388.    OBJS-$(CONFIG_UDP_PROTOCOL)             += udp.o
389.
390.    SKIPHEADERS-$(CONFIG_NETWORK)           += network.h rtsp.h
```

```
391.   TESTPROGS = seek
392.   TOOLS    = aviocat ismindex pktdumper probetest
```

从代码可以看出，libavformat文件夹下的Makefile的规则十分简单，并不包含文件之间的依赖关系（依赖关系位于library.mak中），仅仅是设置了几个变量的值：

NAME：类库名称。注意不包含类库前面的"lib"以及类库的后缀。在这里是"avformat"。

FFLIBS：该类库依赖的类库名称。在这里用到了"avcodec"和"avutil"。

HEADERS：该类库导出的头文件。在这里是"avformat.h"，"avio.h"，"version.h"。

OBJS：该类库依赖的目标文件（必须的）。在这里是"utils.o"等等。

OBJS-yes：该类库依赖的目标文件（可选的）。在这里是"flvdec.o"、"flvenc.o"等等。

library.mak

library.mak专门用于存储编译类库的规则，是和libavXXX/Makefile配合使用的。它的源代码如下所示。

```python
1.    # FFmpeg library.mak
2.    #
3.    # 注释：雷霄骅
4.    # leixiaohua1020@126.com
5.    # http://blog.csdn.net/leixiaohua1020
6.    #
7.    # 编译类库(libavformat等)专用的Makefile，其中包含了编译类库的规则。
8.
9.    # 【NAME位于每个类库的Makefile】，可以取avcodec，avformat等等
10.   SRC_DIR := $(SRC_PATH)/lib$(NAME)
11.
12.   include $(SRC_PATH)/common.mak
13.
14.   #这些信息都位于config.mak中
15.   #例如：
16.   # libavformat_VERSION=53.31.100
17.   # libavformat_VERSION_MAJOR=53
18.
19.   LIBVERSION := $(lib$(NAME)_VERSION)
20.   LIBMAJOR   := $(lib$(NAME)_VERSION_MAJOR)
21.   INCINSTDIR := $(INCDIR)/lib$(NAME)
22.   THIS_LIB   := $(SUBDIR)$($(CONFIG_SHARED:yes=S)LIBNAME)
23.
24.   all-$(CONFIG_STATIC): $(SUBDIR)$(LIBNAME)
25.   all-$(CONFIG_SHARED): $(SUBDIR)$(SLIBNAME)
26.
27.
28.   $(SUBDIR)%-test.o: $(SUBDIR)%-test.c
29.       $(COMPILE_C)
30.
31.   $(SUBDIR)%-test.o: $(SUBDIR)%.c
32.       $(COMPILE_C)
33.   #汇编？
34.   $(SUBDIR)x86/%.o: $(SUBDIR)x86/%.asm
35.       $(YASMDEP) $(YASMFLAGS) -I $(<D)/ -M -o $@ $< > $(@:.o=.d)
36.       $(YASM) $(YASMFLAGS) -I $(<D)/ -o $@ $<
37.
38.   $(OBJS) $(OBJS:.o=.s) $(SUBDIR)%.ho $(TESTOBJS): CPPFLAGS += -DHAVE_AV_CONFIG_H
39.   $(TESTOBJS): CPPFLAGS += -DTEST
40.
41.   # 【OBJS来自于每个类库的Makefile】
42.   #$@   表示规则中的目标文件集
43.   #$^   所有的依赖目标的集合。
44.   #生成静态库？
45.   $(SUBDIR)$(LIBNAME): $(OBJS)
46.       $(RM) $@
47.       $(AR) rc $@ $^ $(EXTRAOBJS)
48.       $(RANLIB) $@
49.   #安转头文件，根目录的Makefile调用
50.   install-headers: install-lib$(NAME)-headers install-lib$(NAME)-pkgconfig
51.   #install-libs-yes被install-libs（位于根目录Makefile）调用
52.   install-libs-$(CONFIG_STATIC): install-lib$(NAME)-static
53.   install-libs-$(CONFIG_SHARED): install-lib$(NAME)-shared
54.
55.   define RULES
56.   $(EXAMPLES) $(TESTPROGS) $(TOOLS): %$(EXESUF): %.o
57.       $$(LD) $(LDFLAGS) -o $$@ $$^ -l$(FULLNAME) $(FFEXTRALIBS) $$(ELIBS)
58.
59.   $(SUBDIR)$(SLIBNAME): $(SUBDIR)$(SLIBNAME_WITH_MAJOR)
60.       $(Q)cd ./$(SUBDIR) && $(LN_S) $(SLIBNAME_WITH_MAJOR) $(SLIBNAME)
61.
62.   $(SUBDIR)$(SLIBNAME_WITH_MAJOR): $(OBJS) $(SUBDIR)lib$(NAME).ver
63.       $(SLIB_CREATE_DEF_CMD)
64.       $$(LD) $(SHFLAGS) $(LDFLAGS) -o $$@ $$(filter %.o,$$^) $(FFEXTRALIBS) $(EXTRAOBJS)
65.       $(SLIB_EXTRA_CMD)
66.
67.   #SLIBNAME_WITH_MAJOR包含了Major版本号。例如：libavformat-53.dll
68.   ifdef SUBDIR
69.   $(SUBDIR)$(SLIBNAME_WITH_MAJOR): $(DEP_LIBS)
70.   endif
```

```
71.   #清空
72.   clean::
73.       $(RM) $(addprefix $(SUBDIR),*-example$(EXESUF) *-test$(EXESUF) $(CLEANFILES) $(CLEANSUFFIXES) $(LIBSUFFIXES)) \
74.           $(foreach dir,$(DIRS),$(CLEANSUFFIXES:%=$(SUBDIR)$(dir)/%)) \
75.           $(HOSTOBJS) $(HOSTPROGS)
76.
77.   distclean:: clean
78.       $(RM) $(DISTCLEANSUFFIXES:%=$(SUBDIR)%) \
79.           $(foreach dir,$(DIRS),$(DISTCLEANSUFFIXES:%=$(SUBDIR)$(dir)/%))
80.   #安装库文件====================
81.   install-lib$(NAME)-shared: $(SUBDIR)$(SLIBNAME)
82.       $(Q)mkdir -p "$(SHLIBDIR)"
83.       $$(INSTALL) -m 755 $$< "$(SHLIBDIR)/$(SLIB_INSTALL_NAME)"
84.       $$(STRIP) "$(SHLIBDIR)/$(SLIB_INSTALL_NAME)"
85.       $(Q)$(foreach F,$(SLIB_INSTALL_LINKS),cd "$(SHLIBDIR)" && $(LN_S) $(SLIB_INSTALL_NAME) $(F);)
86.       $(if $(SLIB_INSTALL_EXTRA_SHLIB),$$(INSTALL) -m 644 $(SLIB_INSTALL_EXTRA_SHLIB:%=$(SUBDIR)%) "$(SHLIBDIR)")
87.       $(if $(SLIB_INSTALL_EXTRA_LIB),$(Q)mkdir -p "$(LIBDIR)")
88.       $(if $(SLIB_INSTALL_EXTRA_LIB),$$(INSTALL) -m 644 $(SLIB_INSTALL_EXTRA_LIB:%=$(SUBDIR)%) "$(LIBDIR)")
89.
90.   install-lib$(NAME)-static: $(SUBDIR)$(LIBNAME)
91.       $(Q)mkdir -p "$(LIBDIR)"
92.       $$(INSTALL) -m 644 $$< "$(LIBDIR)"
93.       $(LIB_INSTALL_EXTRA_CMD)
94.   #安装头文件====================
95.   #-m
96.   #权限：644,755,777
97.   #644 rw-r--r--
98.   #755 rwxr-xr-x
99.   #777 rwxrwxrwx
100.  #从左至右，1-3位数字代表文件所有者的权限，4-6位数字代表同组用户的权限，7-9数字代表其他用户的权限。
101.  #通过4、2、1的组合，得到以下几种权限：0（没有权限）；4（读取权限）；5（4+1｜读取+执行）；6（4+2｜读取+写入）；7（4+2+1｜读取+写入+执行）
102.  #addprefix()
103.  #$(addprefix src/,foo bar)返回值是"src/foo src/bar"。
104.
105.  # 【HEADERS来自于每个类库的Makefile】
106.  #例如libavformat中HEADERS = avformat.h avio.h version.h
107.  install-lib$(NAME)-headers: $(addprefix $(SUBDIR),$(HEADERS) $(BUILT_HEADERS))
108.      $(Q)mkdir -p "$(INCINSTDIR)"
109.      $$(INSTALL) -m 644 $$^ "$(INCINSTDIR)"
110.
111.  install-lib$(NAME)-pkgconfig: $(SUBDIR)lib$(NAME).pc
112.      $(Q)mkdir -p "$(LIBDIR)/pkgconfig"
113.      $$(INSTALL) -m 644 $$^ "$(LIBDIR)/pkgconfig"
114.
115.  #卸载
116.  uninstall-libs::
117.      -$(RM) "$(SHLIBDIR)/$(SLIBNAME_WITH_MAJOR)" \
118.              "$(SHLIBDIR)/$(SLIBNAME)"               \
119.              "$(SHLIBDIR)/$(SLIBNAME_WITH_VERSION)"
120.      -$(RM) $(SLIB_INSTALL_EXTRA_SHLIB:%="$(SHLIBDIR)"%)
121.      -$(RM) $(SLIB_INSTALL_EXTRA_LIB:%="$(LIBDIR)"%)
122.      -$(RM) "$(LIBDIR)/$(LIBNAME)"
123.
124.  uninstall-headers::
125.      $(RM) $(addprefix "$(INCINSTDIR)/",$(HEADERS)) $(addprefix "$(INCINSTDIR)/",$(BUILT_HEADERS))
126.      $(RM) "$(LIBDIR)/pkgconfig/lib$(NAME).pc"
127.      -rmdir "$(INCINSTDIR)"
128.  endef
129.
130.  $(eval $(RULES))
131.
132.  $(EXAMPLES) $(TESTPROGS) $(TOOLS): $(THIS_LIB) $(DEP_LIBS)
133.  $(TESTPROGS): $(SUBDIR)$(LIBNAME)
134.
135.  examples: $(EXAMPLES)
136.  testprogs: $(TESTPROGS)
```

library.mak代码中首先包含了common.mak文件。这个文件定义了通用的一些编译规则。然后定义了类库的依赖关系。

此外library.mak中也定义了install-headers，install-lib$(NAME)-shared，install-lib$(NAME)-static，install-lib$(NAME)-headers，clean等等一系列的伪目标（NAME取值avformat、avcodec等）。这些目标主要配合根目录的Makefile使用。

common.mak

common.mak文件定义了通用的一些编译规则。代码如下所示。

```python
1.    # FFmpeg common.mak
2.    #
3.    # 注释：雷霄骅
4.    # leixiaohua1020@126.com
5.    # http://blog.csdn.net/leixiaohua1020
6.    #
7.    # 通用的Makefile，其中包含了通用的编译规则。
8.    #
9.    # common bits used by all libraries
10.   #
11.
```

```makefile
12.    # first so "all" becomes default target
13.    all: all-yes
14.
15.    ifndef SUBDIR
16.    #在控制台打印信息
17.    ifndef V
18.    Q      = @
19.    #输出
20.    ECHO   = printf "$(1)\t%s\n" $(2)
21.    BRIEF  = CC CXX AS YASM AR LD HOSTCC STRIP CP
22.    SILENT = DEPCC YASMDEP RM RANLIB
23.    MSG    = $@
24.    M      = @$(call ECHO,$(TAG),$@);
25.    $(foreach VAR,$(BRIEF), \
26.        $(eval override $(VAR) = @$$(call ECHO,$(VAR),$$(MSG)); $($(VAR))))
27.    $(foreach VAR,$(SILENT),$(eval override $(VAR) = @$($(VAR))))
28.    $(eval INSTALL = @$(call ECHO,INSTALL,$$(^:$(SRC_DIR)/%=%)); $(INSTALL))
29.    endif
30.    #所有的lib
31.    ALLFFLIBS = avcodec avdevice avfilter avformat avutil postproc swscale swresample
32.
33.    # NASM requires -I path terminated with /
34.    #各种Flag
35.    #SRC_PATH=.
36.    IFLAGS     := -I. -I$(SRC_PATH)/
37.    CPPFLAGS   := $(IFLAGS) $(CPPFLAGS)
38.    CFLAGS     += $(ECFLAGS)
39.    CCFLAGS     = $(CFLAGS)
40.    CXXFLAGS   := $(CFLAGS) $(CXXFLAGS)
41.    YASMFLAGS  += $(IFLAGS) -I$(SRC_PATH)/libavutil/x86/ -Pconfig.asm
42.    HOSTCFLAGS += $(IFLAGS)
43.    #avcodec处理后成为-Llibavcodec
44.    #config.mak文件中：
45.    #LDFLAGS= -Wl,--as-needed -Wl,--warn-common -Wl,
46.    #-rpath-link=libpostproc:libswresample:libswscale:libavfilter:libavdevice:libavformat:libavcodec:libavutil
47.    LDFLAGS    := $(ALLFFLIBS:%=-Llib%) $(LDFLAGS)
48.
49.    #命令包
50.    #具体编译命令
51.    #
52.    #$(1)可以取CC、CXX等
53.    #例如取$(1)取CC
54.    #config.mak文件中：
55.    #SRC_PATH=.
56.    #CC=gcc
57.    #
58.    #CCFLAGS=$(CFLAGS)
59.    #CFLAGS    -std=c99 -fno-common -fomit-frame-pointer -I/include/SDL -D_GNU_SOURCE=1 -Dmain=SDL_main
60.    # -g -Wdeclaration-after-statement -Wall -Wno-parentheses -Wno-switch -Wno-format-zero-length
61.    # -Wdisabled-optimization -Wpointer-arith -Wredundant-decls -Wno-pointer-sign -Wcast-qual -Wwrite-strings
62.    # -Wtype-limits -Wundef -Wmissing-prototypes -Wno-pointer-to-int-cast -Wstrict-prototypes
63.    # -O3 -fno-math-errno -fno-signed-zeros -fno-tree-vectorize -Werror=implicit-function-declaration -Werror=missing-prototypes
64.    #
65.    #CPPFLAGS= -D_ISOC99_SOURCE -D_FILE_OFFSET_BITS=64 -D_LARGEFILE_SOURCE -U_STRICT_ANSI_
66.    #CC_O=-o $@
67.    #CC_DEPFLAGS=-MMD -MF $(@:.o=.d) -MT $@
68.    #举例：
69.    #gcc -I. -Itest/ -c -o $@ $<
70.    #再例如$(1)取CXX
71.    #CXXFLAGS=  -D__STDC_CONSTANT_MACROS
72.
73.    define COMPILE
74.        $($(1)DEP)
75.        $($(1)) $(CPPFLAGS) $($(1)FLAGS) $($(1)_DEPFLAGS) -c $($(1)_O) $<
76.    endef
77.
78.    #编译命令
79.    #$(call <expression>,<parm1>,<parm2>,<parm3>...)
80.    #当make执行这个函数时，<expression>参数中的变量，如$(1), $(2), $(3)等，会被参数
81.    #<parm1>, <parm2>, <parm3>依次取代。而<expression>的返回值就是call函数的返回值。
82.    COMPILE_C = $(call COMPILE,CC)
83.    COMPILE_CXX = $(call COMPILE,CXX)
84.    COMPILE_S = $(call COMPILE,AS)
85.
86.    #COMPILE_C为：
87.    #$(CC DEP)
88.    #$($(CC) $(CPPFLAGS) $($(1)FLAGS) $($(1)_DEPFLAGS) -c $($(1)_O) $<
89.
90.    #依赖关系
91.    #C语言
92.    %.o: %.c
93.    #编译
94.        $(COMPILE_C)
95.
96.    #C++
97.    %.o: %.cpp
98.        $(COMPILE_CXX)
99.
100.   %.s: %.c
101.       $(CC) $(CPPFLAGS) $(CFLAGS) -S -o $@ $<
102.
```

```
103.    %.o: %.S
104.        $(COMPILE_S)
105.
106.    %.ho: %.h
107.        $(CC) $(CPPFLAGS) $(CFLAGS) -Wno-unused -c -o $@ -x c $<
108.
109.    %.ver: %.v
110.        $(Q)sed 's/$$MAJOR/$($(basename $(@F))_VERSION_MAJOR)/' $^ > $@
111.
112.    %.c %.h: TAG = GEN
113.
114.    # Dummy rule to stop make trying to rebuild removed or renamed headers
115.    %.h:
116.        @:
117.
118.    # Disable suffix rules.  Most of the builtin rules are suffix rules,
119.    # so this saves some time on slow systems.
120.    .SUFFIXES:
121.
122.    # Do not delete intermediate files from chains of implicit rules
123.    $(OBJS):
124.    endif
125.
126.    OBJS-$(HAVE_MMX) +=  $(MMX-OBJS-yes)
127.
128.    #源自Makefile
129.    #OBJS：该类库必须的目标文件
130.    #OBJS-yes：该类库可配置的目标文件
131.    OBJS      += $(OBJS-yes)
132.    #FFLIBS：必须的类库
133.    #FFLIBS-yes：可选的类库
134.    #FFLIBS = avcodec avutil ....
135.    FFLIBS    := $(FFLIBS-yes) $(FFLIBS)
136.    TESTPROGS += $(TESTPROGS-yes)
137.
138.    FFEXTRALIBS := $(FFLIBS:%=-l%$(BUILDSUF)) $(EXTRALIBS)
139.
140.    EXAMPLES  := $(EXAMPLES:%=$(SUBDIR)%-example$(EXESUF))
141.    #排序？
142.    OBJS      := $(sort $(OBJS:%=$(SUBDIR)%))
143.    TESTOBJS  := $(TESTOBJS:%=$(SUBDIR)%) $(TESTPROGS:%=$(SUBDIR)%-test.o)
144.    TESTPROGS := $(TESTPROGS:%=$(SUBDIR)%-test$(EXESUF))
145.    HOSTOBJS  := $(HOSTPROGS:%=$(SUBDIR)%.o)
146.    HOSTPROGS := $(HOSTPROGS:%=$(SUBDIR)%$(HOSTEXESUF))
147.    TOOLS     += $(TOOLS-yes)
148.    TOOLOBJS  := $(TOOLS:%=tools/%.o)
149.    TOOLS     := $(TOOLS:%=tools/%$(EXESUF))
150.
151.    #DEP_LIBS= libavcodec/libavcodec.a libavutil/libavutil.a ....
152.    DEP_LIBS := $(foreach NAME,$(FFLIBS),lib$(NAME)/$($(CONFIG_SHARED:yes=S)LIBNAME))
153.
154.    ALLHEADERS := $(subst $(SRC_DIR)/,$(SUBDIR),$(wildcard $(SRC_DIR)/*.h $(SRC_DIR)/$(ARCH)/*.h))
155.    SKIPHEADERS += $(ARCH_HEADERS:%=$(ARCH)/%) $(SKIPHEADERS-)
156.    SKIPHEADERS := $(SKIPHEADERS:%=$(SUBDIR)%)
157.    checkheaders: $(filter-out $(SKIPHEADERS:.h=.ho),$(ALLHEADERS:.h=.ho))
158.
159.    alltools: $(TOOLS)
160.
161.    $(HOSTOBJS): %.o: %.c
162.        $(HOSTCC) $(HOSTCFLAGS) -c -o $@ $<
163.
164.    $(HOSTPROGS): %$(HOSTEXESUF): %.o
165.        $(HOSTCC) $(HOSTLDFLAGS) -o $@ $< $(HOSTLIBS)
166.
167.    $(OBJS):     | $(sort $(dir $(OBJS)))
168.    $(HOSTOBJS): | $(sort $(dir $(HOSTOBJS)))
169.    $(TESTOBJS): | $(sort $(dir $(TESTOBJS)))
170.    $(TOOLOBJS): | tools
171.
172.    OBJDIRS := $(OBJDIRS) $(dir $(OBJS) $(HOSTOBJS) $(TESTOBJS))
173.
174.    CLEANSUFFIXES     = *.d *.o *~ *.ho *.map *.ver *.gcno *.gcda
175.    DISTCLEANSUFFIXES = *.pc
176.    LIBSUFFIXES       = *.a *.lib *.so *.so.* *.dylib *.dll *.def *.dll.a *.exp
177.
178.    #依赖文件.d (dependence)
179.    -include $(wildcard $(OBJS:.o=.d) $(TESTOBJS:.o=.d))
```

从代码中可以看出，common.mak定义了一些通用的编译规则，例如编译时候的控制台输出格式，编译命令COMPILE_C、COMPILE_CXX、COMPILE_S，以及.c、.o等文件之间的依赖关系等。

雷霄骅
leixiaohua1020@126.com

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/44556525

文章标签： ffmpeg makefile make configure 编译

个人分类： FFMPEG

所属专栏： FFmpeg