基础3-U-boot根目录下的config.mk分析

该config.mk位于uboot源码的根目录下， 其包含了子目录下许多同名的config.mk。所以千万注意这些同名文件的主次区别
该文件内容主要结构为：

1. 设置各种路径
2. 设置主机环境的编译选项
3. 确定各交叉编译工具
4. 确定各种级别的编译选项
5. 指定链接脚本
6. 获得起始链接地址
7. 设置头文件搜寻路径
8. 使用起始链接地址
9. 设置自动推导规则

需要注意的是，结构顺序并不一定代表代码执行顺序，关于代码的执行顺序以及推荐阅读顺序请移步 [U-boot配置及编译阶段流程宏观分析]

1. 设置各种路径

```
ifneq ($(OBJTREE),$(SRCTREE))
ifeq ($(CURDIR),$(SRCTREE))
dir :=
else
dir := $(subst $(SRCTREE)/,,$(CURDIR))
endif

obj := $(if $(dir),$(OBJTREE)/$(dir)/,$(OBJTREE)/)
src := $(if $(dir),$(SRCTREE)/$(dir)/,$(SRCTREE)/)

$(shell mkdir -p $(obj))
else
obj :=
src :=
endif
```

- 本段代码还是在进行原地编译和外部输出编译的一些路径设置
- 说到底，其实是把主Makefile中和路径有关的变量导入进来

```
PLATFORM_RELFLAGS =
PLATFORM_CPPFLAGS =
PLATFORM_LDFLAGS =

ifeq ($(ARCH),ppc)
ifeq ($(CROSS_COMPILE),powerpc-netbsd-)
PLATFORM_CPPFLAGS+= -D__PPC__
endif
ifeq ($(CROSS_COMPILE),powerpc-openbsd-)
PLATFORM_CPPFLAGS+= -D__PPC__
endif
endif

ifeq ($(ARCH),arm)
ifeq ($(CROSS_COMPILE),powerpc-netbsd-)
PLATFORM_CPPFLAGS+= -D__ARM__
endif
ifeq ($(CROSS_COMPILE),powerpc-openbsd-)
PLATFORM_CPPFLAGS+= -D__ARM__
endif
endif
```

- 本段都是和powerpc有关的交叉编译工具链的设置

2.设置主机环境的编译选项（72至85行）

```
CONFIG_SHELL    := $(shell if [ -x "$$BASH" ]; then echo $$BASH; \
            else if [ -x /bin/bash ]; then echo /bin/bash; \
            else echo sh; fi ; fi)

ifeq ($(HOSTOS)-$(HOSTARCH),darwin-ppc)
HOSTCC      = cc
else
HOSTCC      = gcc
endif
HOSTCFLAGS  = -Wall -Wstrict-prototypes -O2 -fomit-frame-pointer
HOSTSTRIP   = strip
```

- shell的一些配置过程，和主机编译器的一些配置

3. 确定各交叉编译工具

```
cc-option = $(shell if $(CC) $(CFLAGS) $(1) -S -o /dev/null -xc /dev/null \
        > /dev/null 2>&1; then echo "$(1)"; else echo "$(2)"; fi ;)
```

- 本段是编译链接的一些过程选项

```
AS  = $(CROSS_COMPILE)as
LD  = $(CROSS_COMPILE)ld
CC  = $(CROSS_COMPILE)gcc
CPP = $(CC) -E
AR  = $(CROSS_COMPILE)ar
NM  = $(CROSS_COMPILE)nm
LDR = $(CROSS_COMPILE)ldr
STRIP   = $(CROSS_COMPILE)strip
OBJCOPY = $(CROSS_COMPILE)objcopy
OBJDUMP = $(CROSS_COMPILE)objdump
RANLIB  = $(CROSS_COMPILE)RANLIB
```

- 确定了完整的各交叉编译工具，通过顶层makefile中得到的CROSS_COMPILE变量（即工具链的前缀），由此定义各工具的名称

4. 确定各种级别的编译选项

```
sinclude $(OBJTREE)/include/autoconf.mk

ifdef   ARCH
sinclude $(TOPDIR)/$(ARCH)_config.mk # include architecture dependend rules
endif
ifdef   CPU
sinclude $(TOPDIR)/cpu/$(CPU)/config.mk  # include  CPU  specific rules
endif
ifdef   SOC
sinclude $(TOPDIR)/cpu/$(CPU)/$(SOC)/config.mk  # include  SoC  specific rules
endif
ifdef   VENDOR
BOARDDIR = $(VENDOR)/$(BOARD)
else
BOARDDIR = $(BOARD)
endif
ifdef   BOARD
sinclude $(TOPDIR)/board/$(BOARDDIR)/config.mk # include board specific rules
endif
```

- 本段最开始包含了一个autoconf.mk文件，此文件也不是源码自带的，其内容全部都是CONFIG_开头的变量，makefile利用这些变量来指导编译过程的走向（.c文件条件编译）
- autoconf.mk其实是由顶层Makefile利用根目录下include/configs/x210_sd.h生成的，其内容与x210_sd.h没什么区别，其生成方式在顶层makefile的470多行左右
- uboot的可移植性很大程度来源于x210_sd.h文件，它也是移植工作的关键所在
- 后面几行分别inlude了和ARCH、CPU、SOC、VENDOR、BOARD相关的子config.mk，也就是包含了各种级别的编译属性选项

5. 指定链接脚本

```
ifneq (,$(findstring s,$(MAKEFLAGS)))
ARFLAGS = cr
else
ARFLAGS = crv
endif
RELFLAGS= $(PLATFORM_RELFLAGS)
DBGFLAGS= -g # -DDEBUG
OPTFLAGS= -Os #-fomit-frame-pointer
ifndef LDSCRIPT
#LDSCRIPT := $(TOPDIR)/board/$(BOARDDIR)/u-boot.lds.debug
ifeq ($(CONFIG_NAND_U_BOOT),y)
LDSCRIPT := $(TOPDIR)/board/$(BOARDDIR)/u-boot-nand.lds
else
LDSCRIPT := $(TOPDIR)/board/$(BOARDDIR)/u-boot.lds
endif
endif
OBJCFLAGS += --gap-fill=0xff
```

- 本段最开始先配置了一些编译选项
- 随后检测是否定义过LDSCRIPT这个和链接脚本有关的变量，如果没有定义，则判断CONFIG_NAND_U_BOOT（autoconf.mk中定

义）这个值是否为y
- 若未使用nandflash，则链接脚本为指定路径下的u-boot.lds。若使用了nandflash，则链接脚本为指定路径下的u-boot-nand.lds
-

6. 获得起始链接地址

```
gccincdir := $(shell $(CC) -print-file-name=include)

CPPFLAGS := $(DBGFLAGS) $(OPTFLAGS) $(RELFLAGS)       \
    -D__KERNEL__
ifneq ($(TEXT_BASE),)
CPPFLAGS += -DTEXT_BASE=$(TEXT_BASE)
endif
```

- 本段最开始先配置了一些编译选项，随后做了一些和TEXT_BASE相关的工作
- 在前面120多行左右include了一个board级别的config.mk文件，这个.mk文件源码中不存在，是在顶层makefile中由x210_sd_config创建的，并在ARCH、SOC等变量后面添加TEXT_BASE，展开后就获得了一个TEXT_BASE变量
- TEXT_BASE这个变量的含义是uboot将来被链接时的起始地址，是规定好的，但由于uboot使用虚拟地址映射，所以这个地址并不是真正的物理地址
- 最后进行判断，如果TEXT_BASE不为空，就将其设置到CPPFLAGS里面去
-

7. 设置头文件搜寻路径

```
ifneq ($(OBJTREE),$(SRCTREE))
CPPFLAGS += -I$(OBJTREE)/include2 -I$(OBJTREE)/include
endif

CPPFLAGS += -I$(TOPDIR)/include
CPPFLAGS += -fno-builtin -ffreestanding -nostdinc   \
    -isystem $(gccincdir) -pipe $(PLATFORM_CPPFLAGS)
```

- 本段设置了头文件的搜寻路径，添加了顶层目录下的include文件夹作为搜索路径。然后进行了一些其他设置（比如禁止标准的include路径）
- 这样，编译器在编译的时候就能正确的读取到include文件夹下的头文件（其实是配置阶段创建的符号链接）了

8.使用起始链接地址

```
LDFLAGS += -Bstatic -T $(LDSCRIPT) $(PLATFORM_LDFLAGS)
ifneq ($(TEXT_BASE),)
LDFLAGS += -Ttext $(TEXT_BASE)
endif
```

- 本段是把TEXT_BASE用-Ttext传给了链接脚本，故链接脚本将从TEXT_BASE开始链接

9. 设置自动推导规则

```
ifndef REMOTE_BUILD

%.s:    %.S
    $(CPP) $(AFLAGS) -o $@ $<
%.o:    %.S
    $(CC) $(AFLAGS) -c -o $@ $<
%.o:    %.c
    $(CC) $(CFLAGS) -c -o $@ $<

else

$(obj)%.s: %.S
    $(CPP) $(AFLAGS) -o $@ $<
$(obj)%.o: %.S
    $(CC) $(AFLAGS) -c -o $@ $<
$(obj)%.o: %.c
    $(CC) $(CFLAGS) -c -o $@ $<
endif
```

- 本段是makefile的自动推导规则，和顶层Makefile内的规则配合使用，就能实现众多文件的编译