# 一、Linux顶层Makefile基本流程

hdr-arch = arm

# 二、make xxx\_defconfig过程

%config: scripts\_basic outputmakefile FORCE

$(Q)$(MAKE) $(build)=scripts/kconfig $@

scripts\_basic:

$(Q)$(MAKE) $(build)=scripts/basic

include scripts/Kbuild.include

build := -f $(srctree)/scripts/Makefile.build obj

$(Q)$(MAKE) $(build)=scripts/basic

$(Q)$(MAKE) $(build)=scripts/kconfig $@

make -f ./scripts/Makefile.build obj= scripts/basic ->生成fixdep和bin2c

make -f ./scripts/Makefile.build obj=scripts/kconfig xxx\_defconfig ->

scripts/kconfig/conf --defconfig=arch/arm/configs/%\_defconfig Kconfig ->.config

# 三、make过程

通过make或make all来编译Linux内核。默认目标：

\_all:

\_all: all

all: vmlinux

vmlinux: scripts/link-vmlinux.sh $(vmlinux-deps) FORCE

ifdef CONFIG\_HEADERS\_CHECK

$(Q)$(MAKE) -f $(srctree)/Makefile headers\_check

endif

ifdef CONFIG\_SAMPLES

$(Q)$(MAKE) $(build)=samples

endif

ifdef CONFIG\_BUILD\_DOCSRC

$(Q)$(MAKE) $(build)=Documentation

endif

ifdef CONFIG\_GDB\_SCRIPTS

$(Q)ln -fsn `cd $(srctree) && /bin/pwd`/scripts/gdb/vmlinux-gdb.py

endif

+$(call if\_changed,link-vmlinux)

vmlinux-deps := $(KBUILD\_LDS) $(KBUILD\_VMLINUX\_INIT) $(KBUILD\_VMLINUX\_MAIN)

KBUILD\_LDS := arch/$(SRCARCH)/kernel/vmlinux.lds 展开就是：

KBUILD\_LDS := arch/arm/kernel/vmlinux.lds 连接脚本

KBUILD\_VMLINUX\_INIT := $(head-y) $(init-y)

export KBUILD\_VMLINUX\_MAIN := $(core-y) $(libs-y) $(drivers-y) $(net-y)

head-y := arch/arm/kernel/head.o

init-y := init/

init-y := $(patsubst %/, %/built-in.o, $(init-y))

init-y= init/built-in.o

drivers-y := drivers/ sound/ firmware/

net-y := net/

drivers-y := $(patsubst %/, %/built-in.o, $(drivers-y))

net-y := $(patsubst %/, %/built-in.o, $(net-y))

drivers-y= drivers/built-in.o sound/built-in.o firmware/built-in.o

net-y = net/built-in.o

libs-y := lib/

libs-y := arch/arm/lib/ $(libs-y)

libs-y := arch/arm/lib lib/

libs-y1 := $(patsubst %/, %/lib.a, $(libs-y))

libs-y2 := $(patsubst %/, %/built-in.o, $(libs-y))

libs-y := $(libs-y1) $(libs-y2)

libs-y := lib/lib.a arch/arm/lib/lib.a lib/built-in.o arch/arm/lib/built-in.o

所有的.o和.a库文件准备好以后，就可以连接生成vmlinux。命令为：

+$(call if\_changed,link-vmlinux)此命令最终展开为：

cmd\_link-vmlinux = /bin/bash scripts/link-vmlinux.sh arm-linux-gnueabihf-ld -EL -p --noundefined -X --pic-veneer --build-id

# 四、built-in.o文件编译生成过程

vmlinux-deps = arch/arm/kernel/vmlinux.lds arch/arm/kernel/head.o init/built-in.o usr/built-in.o arch/arm/vfp/built-in.o arch/arm/vdso/built-in.o arch/arm/kernel/built-in.o arch/arm/mm/built-in.o arch/arm/common/built-in.o arch/arm/probes/built-in.o arch/arm/net/built-in.o arch/arm/crypto/built-in.o arch/arm/firmware/built-in.o arch/arm/mach-imx/built-in.o kernel/built-in.o mm/built-in.o fs/built-in.o ipc/built-in.o security/built-in.o crypto/built-in.o block/built-in.o arch/arm/lib/lib.a lib/lib.a arch/arm/lib/built-in.o lib/built-in.o drivers/built-in.o sound/built-in.o firmware/built-in.o net/built-in.o

vmlinux-dirs := $(patsubst %/,%,$(filter %/, $(init-y) $(init-m) \

$(core-y) $(core-m) $(drivers-y) $(drivers-m) \

$(net-y) $(net-m) $(libs-y) $(libs-m)))

vmlinux-dirs = init usr arch/arm/vfp arch/arm/vdso arch/arm/kernel arch/arm/mm arch/arm/common arch/arm/probes arch/arm/net arch/arm/crypto arch/arm/firmware arch/arm/mach-imx kernel mm fs ipc security crypto block drivers sound firmware net arch/arm/lib lib

$(vmlinux-dirs): prepare scripts

$(Q)$(MAKE) $(build)=$@

make -f ./scripts/Makefile.build obj=$@ 展开

make -f ./scripts/Makefile.build obj=init

默认目标为\_\_build，规则就是：

\_\_build: $(if $(KBUILD\_BUILTIN),$(builtin-target) $(lib-target) $(extra-y)) \

$(if $(KBUILD\_MODULES),$(obj-m) $(modorder-target)) \

$(subdir-ym) $(always)

@:

简化完成以后就是：

\_\_build: $(builtin-target) $(lib-target) $(extra-y)) $(subdir-ym) $(always)  
@:

cmd\_link\_o\_target = $(if $(strip $(obj-y)),\

$(LD) $(ld\_flags) -r -o $@ $(filter $(obj-y), $^) \

$(cmd\_secanalysis),\

rm -f $@; $(AR) rcs$(KBUILD\_ARFLAGS) $@)

$(builtin-target): $(obj-y) FORCE

$(call if\_changed,link\_o\_target) 相当于调用cmd\_link\_o\_target。

builtin-target := $(obj)/built-in.o

$(obj)/built-in.o: $(obj-y) FORCE

$(call if\_changed,link\_o\_target) 相当于调用cmd\_link\_o\_target。

# 五、make zImage过程

我们需要zImage文件，但是前面一直说的是生成vmlinux，

BOOT\_TARGETS = zImage Image xipImage bootpImage uImage

$(BOOT\_TARGETS): vmlinux

$(Q)$(MAKE) $(build)=$(boot) MACHINE=$(MACHINE) $(boot)/$@

zImage Image xipImage bootpImage uImage: vmlinux

$(Q)$(MAKE) $(build)=$(boot) MACHINE=$(MACHINE) $(boot)/$@ 展开

make -f ./scripts/Makefile.build obj=arch/arm/boot MACHINE=arch/arm/boot/zImage

# 