Linux内核Makefile浅析 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/645/2021_2022_Linux_E5_86_85_E6_A0_c103_645656.htm 1． 配置系统的基本结构 Linux内核的配置系统由三个部分组成，分别是： Makefile：分布在 Linux 内核源代码中的 Makefile，定义 Linux 内核的编译规则；配置文件（config.in）：给用户提供配置选择的功能；配置工具：包括配置命令解释器（对配置脚本中使用的配置命令进行解释）和配置用户界面（提供基于字符界面、基于 Ncurses 图形界面以及基于 Xwindows 图形界面的用户配置界面，各自对应于 Make config、Make menuconfig 和 make xconfig）。 这些配置工具都是使用脚本语言，如 Tcl/TK、Perl 编写的（也包含一些用 C 编写的代码）。本文并不是对配置系统本身进行分析，而是介绍如何使用配置系统。所以，除非是配置系统的维护者，一般的内核开发者无须了解它们的原理，只需要知道如何编写 Makefile 和配置文件就可以。所以，在本文中，我们只对 Makefile 和配置文件进行讨论。另外，凡是涉及到与具体 CPU 体系结构相关的内容，我们都以 arm 为例，这样不仅可以将讨论的问题明确化，而且对内容本身不产生影响。 2． Makefile 2.1 Makefile 概述 Makefile 的作用是根据配置的情况，构造出需要编译的源文件列表，然后分别编译，并把目标代码链接到一起，最终形成 Linux 内核二进制文件。 由于 Linux 内核源代码是按照树形结构组织的，所以 Makefile 也被分布在目录树中。Linux 内核中的 Makefile 以及与 Makefile 直接相关的文件有： Makefile：顶层 Makefile，是整个内核配置、编译的总体控制文件。 .config

：内核配置文件，包含由用户选择的配置选项，用来存放内核配置后的结果（如 make config）。 arch/*/Makefile：位于各种 CPU 体系目录下的 Makefile，如 arch/arm/Makefile，是针对特定平台的 Makefile。 各个子目录下的 Makefile：比如 drivers/Makefile，负责所在子目录下源代码的管理。 Rules.make：规则文件，被所有的 Makefile 使用。 用户通过 make config 配置后，产生了 .config。顶层 Makefile 读入 .config 中的配置选择。顶层 Makefile 有两个主要的任务：产生 vmlinux 文件和内核模块（module）。为了达到此目的，顶层 Makefile 递归的进入到内核的各个子目录中，分别调用位于这些子目录中的 Makefile。至于到底进入哪些子目录，取决于内核的配置。在顶层 Makefile 中，有一句：include arch/$(ARCH)/Makefile，包含了特定 CPU 体系结构下的 Makefile，这个 Makefile 中包含了平台相关的信息。 位于各个子目录下的 Makefile 同样也根据 .config 给出的配置信息，构造出当前配置下需要的源文件列表，并在文件的最后有 include $(TOPDIR)/Rules.make。 Rules.make 文件起着非常重要的作用，它定义了所有 Makefile 共用的编译规则。比如，如果需要将本目录下所有的 c 程序编译成汇编代码，需要在 Makefile 中有以下的编译规则： %.s: %.c $(CC) $(CFLAGS) -S $ 