**浅谈内核的Makefile、Kconfig和.config文件**

Linux内核源码文件繁多，搞不清Makefile、Kconfig、.config间的关系，不了解内核编译体系，编译修改内核有问题无从下手，自己写的驱动不知道怎么编进内核，不知道怎么配置内核，这些问题都和Makefile、Kconfig、.config有关，下面简单谈谈Makefile、Kconfig和.config。希望对你有启发。

**三者的作用：**

简单来说就是去饭店点菜：Kconfig是菜单，Makefile是做法，.config就是你点的菜。

Makefile：一个文本形式的文件，编译源文件的方法。

Kconfig：一个文本形式的文件，内核的配置菜单。

.config：编译内核所依据的配置。

**三者的语法**

**Makefile**

参考：linux-3.4.2/drivers/Makefile

作用：用来定义哪些内容作为模块编译，哪些条件编译等。子目录Makefile被顶层Makefile包含。

（1）直接编译

obj-y      += xxx.o

表示由xxx.c或xxx.s编译得到xxx.o并直接编进内核。

（2）条件编译

obj -$(CONFIG\_HELLO)  += xxx.o

根据.config文件的CONFIG\_XXX来决定文件是否编进内核。

（3）模块编译

obj-m     +=xxx.o

表示xxx作为模块编译，即执行make modules时才会被编译。

**2、Kconfig**

每个config菜单项都有类型定义: bool布尔类型、 tristate三态(内建、模块、移除)、string字符串、 hex十六进制、integer整型。

作用：决定make menuconfig时展示的菜单项，

参考：linux-3.4.2/drivers/leds/ kconfig：

config LEDS\_S3C24XX

       tristate "LED Support for Samsung S3C24XX GPIO LEDs"

       depends on LEDS\_CLASS

       depends on ARCH\_S3C24XX

       help

         This option enables support for LEDs connected to GPIO lines

         on Samsung S3C24XX series CPUs, such as the S3C2410 and S3C2440.

LEDS\_S3C24XX：配置选项的名称，省略了前缀"CONFIG\_"

Tristate：

 表示该项是否编进内核、编成模块。显示为< > , 假如选择编译成内核模块，则会在.config中生成一个 CONFIG\_HELLO\_MODULE=m的配置，选择Y就是直接编进内核，会在.config中生成一个 CONFIG\_HELLO\_MODULE=y的配置项。Tristate后的字符串是make menuconfig时显示的配置项名称。

bool：

此类型只能选中或不选中，make menuconfig时显示为[ ]，即无法配置成模块。

dependon:

该选项依赖于另一个选项，只有当依赖项被选中时，当前配置项的提示信息才会出现，才能设置当前配置项。

select:

反向依赖关系，该选项选中时，同时选中select后面定义的那一项。

help：

帮助信息。

目录层次迭代 ：

Kconfig中有类似语句：source "drivers/usb/Kconfig" ，用来包含（或嵌套）新的Kconfig文件，使得各个目录管理各自的配置内容，不必把那些配置都写在同一个文件里，方便修改和管理。

**3、.config**

参考：linux-3.4.2/.config

通过前俩个文件的分析，.config的含义已经很清晰：内核编译参考文件，查看里面内容可以知道哪些驱动被编译进内核。

配置内核方式有3种(任选其一)：

（1）make menuconfig

（2）make xxx\_defconfig

（3）直接修改.config

注意: 如果直接修改.config，不一定会生效，因为有些配置可能存在依赖关系，make时会根据依赖关系，进行规则的检查，直接修改.config有时无效，所以不推荐直接修改。

以上可能有点抽象，下面举例说明：

写一个简单的入口函数输出hello world的驱动并编译进内核。

**步骤**：

（1）在drivers目录下新建hello文件夹，里面实现hello.c、Makefile、Kconfig。

**hello.c：**

#include <linux/module.h>

#include <linux/kernel.h>

#include <linux/init.h>

static int first\_drv\_init(void)

{

   printk("------------------hello world !--------------------");

   return 0;

}

static void first\_drv\_exit(void)

{

    printk("------------------exit hello world !--------------------");

}

module\_init(first\_drv\_init);

module\_exit(first\_drv\_exit);

MODULE\_LICENSE("GPL");

**Makefile**：

obj-$(CONFIG\_HELLO)         += hello.o

**Kconfig**：

config HELLO

   tristate "Hello World for fengyuwuzu"

   help

     Hello  for fengyuwuzu

config HELLO决定名字：CONFIG\_HELLO。

Hello World for fengyuwuzu：决定了在make menuconfig时显示的名字

（2）修改上一级（Linux-3.4.2/drivers下）的Makefile、Kconfig。

**Makefile：**

obj-y  += hello/

**Kconfig：**

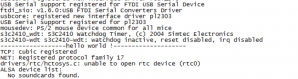
source  "drivers/hello/Kconfig"

（3）make menuconfig



（4）make uImage再烧写到开发板。

查看内核启动Log，伟大的helloworld 出来了！说明hello.c成功编进内核



欢迎参考本文实验，相信你对内核的组织结构会有更清晰的认识。