

牧野星辰

据说健身和写代码都会导致秃顶。

[博客园](#) [首页](#) [新随笔](#) [联系](#) [订阅](#) [管理](#)

linux设备驱动程序——将驱动程序编译进内核

linux驱动程序——将驱动程序编译进内核

模块的加载

通常来说, 在驱动模块的开发阶段, 一般是将模块编译成.ko文件, 再使用

```
sudo insmod module.ko
```

或者

```
depmod -a  
modprobe module
```

将模块加载到内核, 相对而言, modprobe要比insmod更加智能, 它会检查并自动处理模块的依赖, 而insmod出现依赖问题时仅仅是告诉你安装失败, 自己想办法吧。

将模块编译进内核

这一章节我们并不关注模块的运行时加载, 我们要讨论的是将模块编译进内核。

在学习内核的Makefile规则的时候就可以知道, 将驱动程序编译成模块时, 只需要使用:

```
obj-m += module.o
```

指定相应的源代码(源代码为module.c)即可, 所以很多朋友就简单地得出结论:如果要 将模块编译进内核, 只要执行下面的的指令就可以了:

```
obj-y += module.o
```

事实上, 这样是行不通的, 要明白怎么将驱动程序编译进内核, 我们还是得先了解linux源码的编译规则。

关于linux源码的编译规则和部分细节可以查看我的另一篇博客[linux内核Makefile概览](#)

本篇博客的所有实验基于arm平台, beagle bone开发板, 内核版本为4.14.79

编译平台

注: 在以下的讨论中, 目标主机和本机指加载运行驱动程序的机器, 是开发的对象。而开发机指只负责编译的机器, 一般指PC机。

本机编译

在对驱动程序进行编译时, 一般会有两种不同的做法:

- 直接 在目标主机上编译
- 在其他平台上构建交叉编译环境, 一般是在PC机上编译出可在目标板上运行的驱动程序

直接在目标主机上编译是比较方便的做法, 本机编译本机运行。

通常, 本机系统中一般不会自带linux内核源码的头文件, 我们需要做的就是 在系统中安装头文件:

```
sudo apt-get install linux-headers-$(uname -r)
```

\$(uname -r)获取当前主机运行的linux版本号。

有了头文件, 那么源代码从哪里来呢? 答案是并不需要源代码, 或者说是并不需要C文件形式的源代码, 而是直接引用当前运行的镜像, 在编译时,

公告

昵称: 牧野星辰
园龄: 3年5个月
粉丝: 112
关注: 3
+加关注

< 2022年2月 >						
日	一	二	三	四	五	六
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	1	2	3	4	5
6	7	8	9	10	11	12

搜索

找找看

谷歌搜索

常用链接

[我的随笔](#)
[我的评论](#)
[我的参与](#)
[最新评论](#)
[我的标签](#)

我的标签

[gcc 编译流程\(2\)](#)
[linux设备驱动程序\(1\)](#)
[linux设备树多级子节点的转换\(1\)](#)
[linux设备驱动程序-i2c框架\(1\)](#)
[linux bus\(1\)](#)
[linux设备驱动程序--在用户空间注册文件接口\(1\)](#)
[linux设备驱动程序--gpio控制\(1\)](#)
[linux设备驱动程序--sysfs\(1\)](#)
[linux 设备驱动程序--hello-world\(1\)](#)
[linux i2c驱动程序源码实现\(1\)](#)
[更多](#)

随笔分类

[C\(6\)](#)
[C++\(5\)](#)
[linux设备驱动程序\(17\)](#)

将/boot/vmlinuz-\$(version)镜像当库文件进行链接即可。

值得注意的是，/boot/vmlinuz-\$(version)是linux启动时读取的镜像，但是在本机中进行驱动程序编译的时候并不会影响到这个镜像，换句话说，即使是指定了obj-y，驱动程序也不会编译到/boot/vmlinuz-\$(version)镜像中，自然达不到将驱动编译进内核的效果。

交叉编译

本机(目标机)编译是比较方便的，但是无法改变生成的镜像，当然也可以将源码下载到本机(目标机)中进行编译，就可以生成相应的linux镜像。

但是一般情况下，在嵌入式开发中，不论是网络、内存还是执行速率，目标主机的性能一般不会太高，如果需要编译完整的源码时，用户会更倾向于在PC端构建编译环境以获取更好的编译性能。

选择在开发机上编译而不是本机编译时，需要注意的一点就是：通常嵌入式开发都是基于arm、mips等嵌入式架构，而PC常用X86架构，在编译时就不能使用开发机上的gcc编译器，因为开发机上编译器是针对开发平台(X86)，而非运行平台(arm、mips)，所以需要使用交叉编译工具链，同时在编译时指定运行的主机平台。

指令是这样的：

```
make arch=arm CROSS_COMPILE=$COMPILE_PATH/$COMPILE_TOOL
```

也可以在makefile中给相应的arch和CROSS_COMPILE变量赋值，直接执行make指令即可。

显然，这种交叉编译方式是对linux内核源码的完整编译，主要生成这一些目标文件：

- 生成linux的可启动镜像，通常是zImage或者vmlinuz，这是一个可boot执行的压缩文件
- 伴随着的还有镜像对应的map文件，这个文件对应镜像中的编译符号以及符号的地址信息
- 未编译进内核的模块,也就是在配置时被选为M的选项
- linux内核头文件等等

在上文中有提到，目标主机中，linux的启动镜像放置在/boot目录下，所以如果我们需要替换linux的镜像，需要替换/boot目录下的以下两个文件：

- linux的可启动镜像,也就是生成的zImage或者vmlinuz
- .map文件

在主机中，模块一般被放置在/lib/modules目录中，如果交叉编译出的版本与本机中模块版本不一致，将无法识别，所以编译出的模块也需要替换。

驱动程序编译进内核

根据上文，可以得出的结论是：在试图将驱动程序编译进内核时，我们需要编译完整的linux内核源码以生成相应的镜像文件，然后将其替换到目标主机的/boot目录下即可。

那么，怎样将驱动的源码C文件编译进内核呢？

这个问题得从makefile的执行流程说起：

make的执行

首先，如果你有基本的linux内核编译经验，就知道在编译linux源码前，需要进行config(配置)，来决定哪些部分编译进内核、哪些部分编译成模块。

通常使用make menuconfig，不同的config方式通常只是选择界面的不同，其中稍微特殊的make oldconfig则是沿用之前的配置。

在配置完成之后将生成一个.config文件，makefile根据.config文件选择性地进入子目录中执行编译工作。

流程基本如上所说，但是我们需要知道更多的细节：

- make menuconfig执行的原理是什么？
- 顶层makefile是怎样执行子目录中的编译工作的？

答案是这样的：

- make menuconfig肯定是读取某个配置文件来陈列出所有的配置选项，递归地进入到子目录中，发现几乎每个子目录中都有一个名为Kconfig的文件，这个文件负责配置驱动在配置菜单中的显示以及配置行为，且Kconfig遵循某种语法，make menuconfig就是读取这些文件来显示配置项。
- 递归地进入到每个子目录中，发现其中都有一个makefile中，可以想到，makefile递归地进入子目录，然后通过调用子目录中的makefile来执行各级子目录的编译，最后统一链接。

整个linux内核的编译都是采用一种分布式的思想，需要添加一个驱动到模块中，我们需要做的事情就是：

1. 将驱动源文件放在内核对应目录中，一般的驱动文件放在drivers目录下，字符设备放在drivers/char中，块设备就放在drivers/blok中，文件的位置遵循这个规律，如果是单个的字符设备源文件，就直接放在drivers/char目录下，如果内容较多足以构成一个模块，则新建一个文件夹。
2. 如果是新建文件夹，因为是分布式编译，需要在文件夹下添加一个Makefile文件和Kconfig文件并修改成指定格式，如果是单个文件直接添加，则直接修改当前目录下的Makefile和Kconfig文件将其添加进去即可。
3. 如果是新建文件夹，需要修改上级目录的Makefile和Kconfig，以将文件夹添加到整个源码编译树中。

Python(5)

操作系统(8)

数学(1)

随笔档案

- 2019年3月(40)
- 2018年9月(5)
- 2018年8月(1)

阅读排行榜

1. linux中apt-get使用(78688)
2. linux开机自启动(69897)
3. python函数调用时参数传递方式(38317)
4. C++ STL hash表用法(28979)
5. linux内核模块编译makefile(22704)

评论排行榜

1. linux中apt-get使用(8)
2. linux的initcall机制(5)
3. C++中string的实现原理(3)
4. 不同平台下int类型、指针类型的数据大小(3)
5. linux设备驱动程序-设备树(2)-device_node转换成platform_device(2)

推荐排行榜

1. linux中apt-get使用(21)
2. linux的initcall机制(8)
3. linux内核makefile概览(5)
4. python函数调用时参数传递方式(5)
5. linux内核模块编译makefile(4)

最新评论

1. Re:C语言中指针和数组
好人一生平安，感谢博主带善人
--cyy489
2. Re:python调用C语言接口
真的爱了，楼主讲的真好
--别爱太满
3. Re:C++ 构造函数的理解
当我们创建一个const对象时，直到构造函数完成初始化过程，对象才能真正取得其常量属性。这句话写的太好了，为博主点赞
--leihao-lester
4. Re:C++中string的实现原理
有没有完整实现示例？
--明明1109
5. Re:C/C++函数调用时传参过程与可变参数实现原理
11
--hell0_word

4. 执行make menuconfig, 执行make

5. 将生成的新镜像以及相应boot文件拷贝到目标主机中, 测试。

beagle bone的启动文件包括:vmlinuz、System.map, 编译出的模块文件为modules

Kconfig的语法简述

上文中提到, 在添加源码时, 一般需要一个Kconfig文件, 这样就可以在make menuconfig时对其进行配置选择, 在对一个源文件进行描述时, 遵循相应的语法。

在这里介绍一些常用的语法选项:

source

source:相当于C语言中的include, 表示包含并引用其他Kconfig文件

config

新建一个条目, 用法:

```
source drivers/xxx/Kconfig
config TEST
    bool "item name"
    depends on NET
    select NET
    help
        just for test
```

config是最常用的关键词了, 它负责新建一个条目, 对应linux中的编译模块, 条目前带有选项。

config TEST:

config后面跟的标识会被当成名称写入到.config文件中, 比如:当此项被选择为[y], 即编译进内核时, 最后会在.config文件中添加这样一个条目:

```
CONFIG_TEST=y
```

CONFIG_TEST变量被传递给makefile进行编译工作。

bool "item name":

其中bool表示选项支持的种类, bool表示两种, 编译进内核或者是忽略, 还有另一种选项就是tristate, 它更常用, 表示支持三种配置选项:编译进内核、编译成可加载模块、忽略。而item name就是显示在menu中的名称。

depends on:

表示当前模块需要依赖另一个选项, 如果另一个选项没有选择编译, 当前条目选项不会出现在窗口中

select:

同样是依赖相关的选项, 表示当前选项需要另外其他选项的支持, 如果选择了当前选项, 那么需要支持的那些选项就会被强制选择编译。

help:

允许添加一些提示信息

menu/menuend

用法:

```
menu "Test option"
...
endmenu
```

这一对关键词创建一个选项目录,该选项目录不能被配置, 选项目录中可以包含多个选项

menuconfig

相当于menu+config, 此选项创建一个选项目录, 而且当前选项目录可配置。

编译示例

梳理了整个添加的流程, 接下来就以一个具体的示例来进行详细的说明。

背景如下:

- 目标开发板为开源平台beagle bone, 基于4.14.79内核版本, arm平台
- 需要添加的源代码为字符设备驱动, 名为cdev_test.c,新建一个目录cdev_test
- 字符设备驱动实现的功能:在/dev目录下生成一个basic_demo文件, 用来检测是否成功将源代码编译进内核

将驱动编译进内核

放置目录

鉴于字符设备, 所以将源文件目录放置在\$KERNEL_ROOT/drivers/char/下面。

如果是块设备, 就会被放置在block下面, 但是这并不是绝对的, 类似USB为字符设备, 但是独立了一个文件出来。

放置后目标文件位置为:\$KERNEL_ROOT/drivers/char/cdev_test

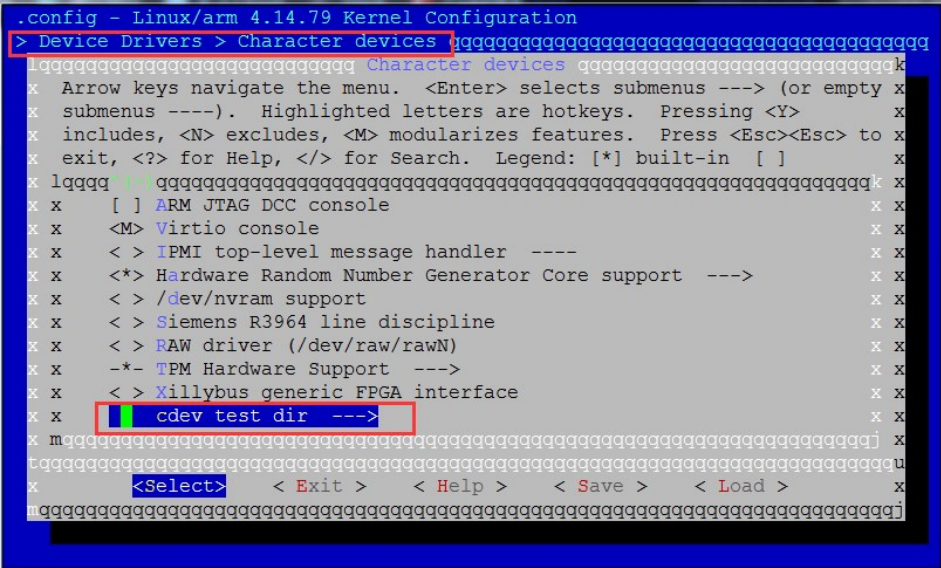
Kconfig文件

在\$KERNEL_ROOT/drivers/char/cdev_test目录下创建一个Kconfig文件, 并修改文件如下:

```
menu "cdev test dir"
config CDEV_TEST
    bool "cdev test support"
    default n
    help
    just for test ,hehe
endmenu
```

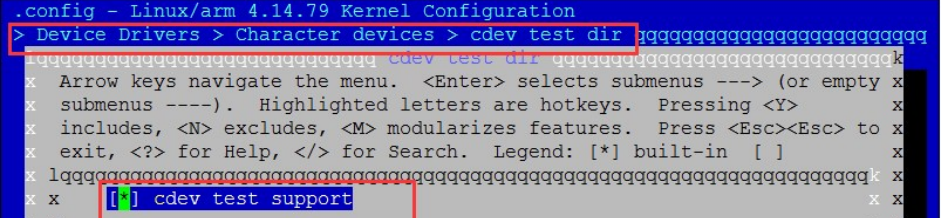
根据上文中对Kconfig文件的语法描述, 可以看出, 这个Kconfig文件的作用就是:

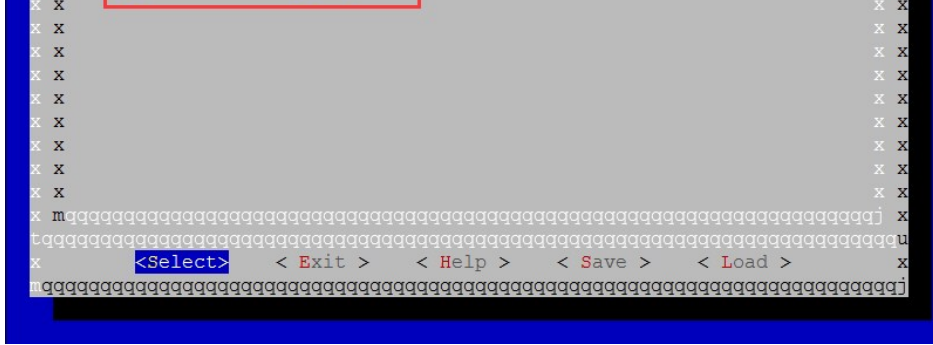
1. 在menuconfig的菜单中, 在Device Driver(对应drivers目录) ---> Character devices(对应char目录)菜单下创建一个名为"cdev test dir"的菜单选项, 执行效果是这样的



2. 在"cdev test dir"的菜单选项下创建一个"cdev test support"的条目, 这个条目的选项只有两个, [*]表示编译进内核和[]表示不进行编译, 默认选择, 不进行编译。

执行效果是这样的





3. 选择help可以查看相应的提示信息。

在上文中还提到，Kconfig分布式地存在于子目录下，同时需要注意的是，在编译时，配置工具并非无差别地进入每个子目录，收集所有的Kconfig信息，而是遵循一定的规则递归进入。

那么，既然是新建的目录，怎么让编译器知道要进入到这个子目录下呢？答案是，在上级目录的Kconfig中包含当前路径下的Kconfig文件。

打开char目录下的Kconfig文件，并且在文件的靠后位置添加：

```
source "drivers/char/xillybus/Kconfig"
```

就把新的Kconfig文件包含到系统中检索目录中了，那么drivers/char/又是怎么被检索到的呢？

就是在drivers的Kconfig中添加drivers/char/目录下的Kconfig索引，以此类推。

Makefile文件

在\$KERNEL_ROOT/drivers/char/cdev_test目录下创建一个Makefile文件，并且编译Makefile文件如下：

```
obj-$(CONFIG_CDEV_TEST) += cdev_test.o
```

表示当前子目录下Makefile的作用就是将cdev_test.c源文件编译成cdev_test.o目标文件。

值得注意的是, 这里的编译选项中使用的是:

```
obj-$ (CONFIG_CDEV_TEST)
```

而非

obj-y

如果确定要将驱动程序编译进内核永远不变,那么可以直接写死,使用obj-y,如果需要进行灵活的定制,还是需要选择第一种做法。

CONFIG_CDEV_TEST是怎么被配置的呢？在上文提到的Kconfig文件编写时，有这么一行：

```
config CDEV_TEST
...

```

在Kconfig被添加到配置菜单中,且被选中编译进内核时,就会在\$KERNEL_ROOT/.config文件中添加一个变量:

```
CONFIG_CDEV_TEST=y
```

自动添加CONFIG_前缀，而名称CDEV_TEST则是由Kconfig指定，看到这里，我想你应该明白了这是怎么回事了。

是不是这样就已经将当前子目录添加到内核编译树中了呢？其实并没有。就像Kconfig一样，Makefile也是分布式存在于整个源码树中，顶层makefile根据配置递归地进入到子目录中，调用子目录中的Makefile进行编译。

同样地，需要修改drivers/char/目录下的Makefile文件，添加一行：

```
obj-$(CONFIG_CDEV_TEST) += cdev_test/
```

在编译时, 如果CONFIG_CDEV_TEST变量为y, cdev_test/Makefile就会被调用。

在make menuconfig中选中

生成配置的部分完成, 就需要在menuconfig菜单中进行配置, 执行:

```
make menuconfig
```

进入目录选项Device Driver --> Character devices--->cdev test dir.

然后按'y'选中模块cdev test support

保存退出, 然后执行编译:

```
make
```

拷贝到目标主机

将vmlinuz(zImage)、System.map拷贝到目标主机的/boot目录下。

在编译生成的modules拷贝到目标主机的/lib/modules目录下。

需要注意的是, 启动文件也好, 模块也好, 在目标板上很可能文件名为诸如vmlinuz-\$version, 会包含版本信息, 需要将文件名修改成一致, 不然无法启动。对于模块而言, 就是相应模块无法加载。

验证

最后一步就是验证自己的驱动程序是否被编译进内核, 如果被编译进内核, 驱动程序中的module_init()程序将被系统调用, 完成一些开发者指定的操作。

这一部分的验证操作就是各显身手了。

好了, 关于linux将驱动程序编译进内核的讨论就到此为止啦, 如果朋友们对于这个有什么疑问或者发现有文章中有什么错误, 欢迎留言

原创博客, 转载请注明出处!


祝各位早日实现项目从想过, bug不沾身.

分类: [linux设备驱动程序](#)

标签: [linux设备驱动程序](#)

好文要顶 关注我 收藏该文



 牧野星辰

关注 - 3

粉丝 - 112

±加关注


1 0

 推荐  反对

« 上一篇: [linux设备驱动程序-设备树\(3\)-设备树多级子节点的转换](#)

posted @ 2019-03-26 15:09 牧野星辰 阅读(15111) 评论(0) 编辑 收藏 举报

[刷新评论](#) [刷新页面](#) [返回顶部](#)

 登录后才能查看或发表评论. 立即 [登录](#) 或者 [逛逛](#) [博客园首页](#)

编辑推荐:

- [记一次最近生产环境项目中发生的两个事故及处理方法](#)
- [.NET 20周年软件趋势随想](#)
- [Three.js 实现2022冬奥主题3D趣味页面](#)
- [技术部如何做复盘——“年终盘点一对一”之创业失败的工程师](#)
- [三探循环依赖 → 记一次线上偶现的循环依赖问题](#)

 百度智能云 企业级云服务器305元 [立即购买](#)

最新新闻:

- [专访阿里云全局高可用技术团队:2022 年了, 怎样才能做到真正的“永不宕机”?](#)
- [一墩难求, 冰墩墩是怎么火的](#)
- [网友爆料B站员工过年加班猝死](#)
- [终于轮到“洋品牌”焦虑了](#)
- [九成订单没付款, 贾跃亭又双叒被拆穿](#)
- » [更多新闻...](#)

