linux驱动学习（三） helloword 和 驱动 Makefile

先看一个最简单的驱动程序：

//hello.c

1. #include <linux/init.h>
2. #include <linux/module.h>
3. MODULE_LICENSE("Dual BSD/GPL"
4. static hello_init(
5. printk(KERN_ALERT "hello module!\n"
6. return
7. static hello_exit(
8. printk(KERN_ALERT "bye module!\n"
9. module_init(hello_init);
10. module_exit(hello_exit);

```
#include <linux/init.h>
#include <linux/module.h>

MODULE_LICENSE("Dual BSD/GPL");

static int hello_init(void)
{
    printk(KERN_ALERT "hello module!\n");
    return 0;
}

static void hello_exit(void)
{
    printk(KERN_ALERT "bye module!\n");
}

module_init(hello_init);
module_exit(hello_exit);
```

> *一个linux内核模块主要由如下几个部分组成：*

（1）module加载函数。

> *当通过insmod或modprobe命令加载内核module时，module的加载函数会自动被内核运行，完成本module的相关初始化工作。*
>
> *module加载函数通过module_init()函数向内核注册。*

（2）module卸载函数。

> *rmmod命令卸载某个模块时，模块的卸载函数会自动被内核执行，完成本模块初始化的相反功能。*
>
> *module卸载函数通过module_exit()函数向内核注册。*

（3）module许可声明（必须）

> *许可证license声明描述内核模块的许可权限，如果不声明license，模块被加载时，将，收到内核被污染（kernel tainted）的警告。linux中可接受的license包括“GPL”，“GPL v2”，“Dual BSD/GPL”，“Dual MPL/GPL”等。*
>
> *多数情况下，内核模块应遵循GPL兼容许可权，2.6内核模块最常见的是以MODULE_LICENSE("Dual BSD/GPL"）语句声明模块采用BSD/GPL 双LICENSE。*

（4）模块参数（可选）

（5）模块到处符号（可选）

（6）模块作者等信息声明（可选），如MODULE_AUTHOR（），MODULE_DESCRIPTION（），MODULE_ALIAS（）等。

编译得到hello.ko，然后insmod hello.ko加载模块，rmmod hello.ko 卸载模块。

> *linux内核的整体结构已经非常庞大，而其包含的组件也非常多，有两种方法把需要的部分都包含在内核中*

一，把所有功能都编译进内核，但这回导致两个问题，生成的内核会特别打，假如要把现在的内核增加或删除功能，将不得不重新编译整个内核。

二，使用模块module，上述我们写的最简单驱动，就是一个模块module，可以随意的增加或删除。

怎样把hello.c源文件编译成helo.ko内核模块呢，同样使用make，但这里的Makefile与一般的应用程序Makefile有所不同，驱动Makfile要指定内核源代码位置，先看一个简单的驱动Makefile：

```
1. obj-m := hello.o
2. KERNEL_DIR := /lib/modules/$(shell uname -r)/build
3. PWD := $(shell pwd)
4.     make -C $(KERNEL_DIR) SUBDIRS=$(PWD) modules
5. clean:
6.     rm *.o *.ko *.mod.c
```

7. .PHONY:clean

```
obj-m := hello.o
KERNEL_DIR := /lib/modules/$(shell uname -r)/build
PWD := $(shell pwd)
all:
        make -C $(KERNEL_DIR) SUBDIRS=$(PWD) modules
clean:
        rm *.o *.ko *.mod.c

.PHONY:clean
```

KERNEL_DIR为内核源代码build目录，我们知道，内核存放在/usr/src中，/lib/modules其实是连接到这个地方，在shell中执行uname -r会得到正在使用的完整内核版本号，这样就选择了适当的内核源码。
PWD为源文件hello.c所在目录。

make -C （大写C） make会进入KERNEL_DIR目录执行此目录下的Makefile，然后在返回PWD目录执行自己写的Makefile。

在终端中make

1. [root@localhost driver]# make
2. make -C /lib/modules/2.6.9-89.ELsmp/build SUBDIRS=/root/linux/driver modules
3. make[1]: Entering directory `/usr/src/kernels/2.6.9-89.EL-smp-i686'
4. CC [M] /root/linux/driver/hello.o
5. Building modules, stage 2.
6. MODPOST
7. CC /root/linux/driver/hello.mod.o
8. LD [M] /root/linux/driver/hello.ko
9. make[1]: Leaving directory `/usr/src/kernels/2.6.9-89.EL-smp-i686'

```
[root@localhost driver]# make
make -C /lib/modules/2.6.9-89.ELsmp/build SUBDIRS=/root/linux/driver modules
make[1]: Entering directory `/usr/src/kernels/2.6.9-89.EL-smp-i686'
  CC [M]  /root/linux/driver/hello.o
  Building modules, stage 2.
  MODPOST
  CC      /root/linux/driver/hello.mod.o
  LD [M]  /root/linux/driver/hello.ko
make[1]: Leaving directory `/usr/src/kernels/2.6.9-89.EL-smp-i686'
```

这样hello.ko驱动模块就产生好了，insmod加载

1. [root@localhost driver]# insmod hello.ko

```
[root@localhost driver]# insmod hello.ko
```

lsmod一下就会看到hello模块的存在了，并且在系统的日志/var/log/messages中会记录模块的输出，也就

是

1. printk(KERN_ALERT "hello module!\n"

```
printk(KERN_ALERT "hello module!\n");
```

输出的helo module！

1. [root@localhost driver]#tail -1 /var/log/messages
2. Oct 13 11:27:07 localhost kernel: hello module!

```
[root@localhost driver]#tail -1 /var/log/messages
Oct 13 11:27:07 localhost kernel: hello module!
```

rmmod helle 卸载hello.ko
一般驱动Makefile会写得更复杂一点，这个再研究。

Links