<http://www.cnblogs.com/hoys/archive/2011/04/10/2011470.html>

内核子系统或设备驱动可以直接编译到内核，也可以编译成模块，如果编译到内核，可以使用前一节介绍的方法通过内核启动参数来向它们传递参数，如果编译成模块，则可以通过命令行在插入模块时传递参数，或者在运行时，通过sysfs来设置或读取模块数据。

Sysfs是一个基于内存的文件系统，实际上它基于ramfs，sysfs提供了一种把内核数据结构、它们的属性以及属性与数据结构的联系开放给用户态的方式，它与kobject子系统紧密地结合在一起，因此内核开发者不需要直接使用它，而是内核的各个子系统使用它。用户要想使用 sysfs 读取和设置内核参数，仅需装载 sysfs 就可以通过文件操作应用来读取和设置内核通过 sysfs 开放给用户的各个参数：

# mkdir -p /sysfs

$ mount -t sysfs sysfs /sysfs

注意，不要把 sysfs 和 sysctl 混淆，sysctl 是内核的一些控制参数，其目的是方便用户对内核的行为进行控制，而 sysfs 仅仅是把内核的 kobject 对象的层次关系与属性开放给用户查看，因此 sysfs 的绝大部分是只读的，模块作为一个 kobject 也被出口到 sysfs，模块参数则是作为模块属性出口的，内核实现者为模块的使用提供了更灵活的方式，允许用户设置模块参数在 sysfs 的可见性并允许用户在编写模块时设置这些参数在 sysfs 下的访问权限，然后用户就可以通过sysfs 来查看和设置模块参数，从而使得用户能在模块运行时控制模块行为。

对于模块而言，声明为 static 的变量都可以通过命令行来设置，但要想在 sysfs下可见，必须通过宏 module\_param 来显式声明，该宏有三个参数，第一个为参数名，即已经定义的变量名，第二个参数则为变量类型，可用的类型有 byte, short, ushort, int, uint, long, ulong, charp 和 bool 或 invbool，分别对应于 c 类型 char, short, unsigned short, int, unsigned int, long, unsigned long, char \* 和 int，用户也可以自定义类型 XXX（如果用户自己定义了 param\_get\_XXX，param\_set\_XXX 和 param\_check\_XXX）。该宏的第三个参数用于指定访问权限，如果为 0，该参数将不出现在 sysfs 文件系统中，允许的访问权限为 S\_IRUSR， S\_IWUSR，S\_IRGRP，S\_IWGRP，S\_IROTH 和 S\_IWOTH 的组合，它们分别对应于用户读，用户写，用户组读，用户组写，其他用户读和其他用户写，因此用文件的访问权限设置是一致的。

在源代码中的内核模块 module-param-exam.c 是一个利用模块参数和sysfs来进行用户态与内核态数据交互的例子。该模块有三个参数可以通过命令行设置，下面是作者系统上的运行结果示例：

# insmod ./module-param-exam.ko my\_invisible\_int=10 my\_visible\_int=20 mystring="Hello,World"

my\_invisible\_int = 10

my\_visible\_int = 20

mystring = 'Hello,World'

# ls /sys/module/module\_param\_exam/parameters/

mystring my\_visible\_int

# cat /sys/module/module\_param\_exam/parameters/mystring

Hello,World

# cat /sys/module/module\_param\_exam/parameters/my\_visible\_int

# echo 2000 > /sys/module/module\_param\_exam/parameters/my\_visible\_int

# cat /sys/module/module\_param\_exam/parameters/my\_visible\_int

# echo "abc" > /sys/module/module\_param\_exam/parameters/mystring

# cat /sys/module/module\_param\_exam/parameters/mystring

abc

# rmmod module\_param\_exam

my\_invisible\_int = 10

my\_visible\_int = 2000

mystring = 'abc'

//filename: module-para-exam.c

#include <linux/config.h>

#include <linux/kernel.h>

#include <linux/module.h>

#include <linux/stat.h>

static int my\_invisible\_int = 0;

static int my\_visible\_int = 0;

static char \* mystring = "Hello, World";

module\_param(my\_invisible\_int, int, 0);

MODULE\_PARM\_DESC(my\_invisible\_int, "An invisible int under sysfs");

module\_param(my\_visible\_int, int, S\_IRUSR | S\_IWUSR | S\_IRGRP | S\_IROTH);

MODULE\_PARM\_DESC(my\_visible\_int, "An visible int under sysfs");

module\_param(mystring, charp, S\_IRUSR | S\_IWUSR | S\_IRGRP | S\_IROTH);

MODULE\_PARM\_DESC(mystring, "An visible string under sysfs");

static int \_\_init exam\_module\_init(void)

{

printk("my\_invisible\_int = %d\n", my\_invisible\_int);

printk("my\_visible\_int = %d\n", my\_visible\_int);

printk("mystring = '%s'\n", mystring);

return 0;

}

static void \_\_exit exam\_module\_exit(void)

{

printk("my\_invisible\_int = %d\n", my\_invisible\_int);

printk("my\_visible\_int = %d\n", my\_visible\_int);

printk("mystring = '%s'\n", mystring);

}

module\_init(exam\_module\_init);

module\_exit(exam\_module\_exit);

MODULE\_AUTHOR("Yang Yi");

MODULE\_DESCRIPTION("A module\_param example module");

MODULE\_LICENSE("GPL");