

实验 16 驱动模块传参数

16.1 本章导读

加载模块的时候还可以通过 insmod 命令传参数，掌握了这个知识点之后调试起来会方便很多。

16.1.1 工具

16.1.1.1 硬件工具

- 1) iTOP4412 开发板
- 2) U 盘或者 TF 卡
- 3) PC 机
- 4) 串口

16.1.1.2 软件工具

- 1) 虚拟机 Vmware
- 2) Ubuntu12.04.2
- 3) 超级终端 (串口助手)
- 4) 源码文件夹 “module_param”

16.1.2 预备课程

无

16.1.3 视频资源

本节配套视频为“视频 16_驱动模块传参数”

16.2 学习目标

本章需要学习以下内容：

加载模块的时候，向驱动模块传参数

16.3 实验操作

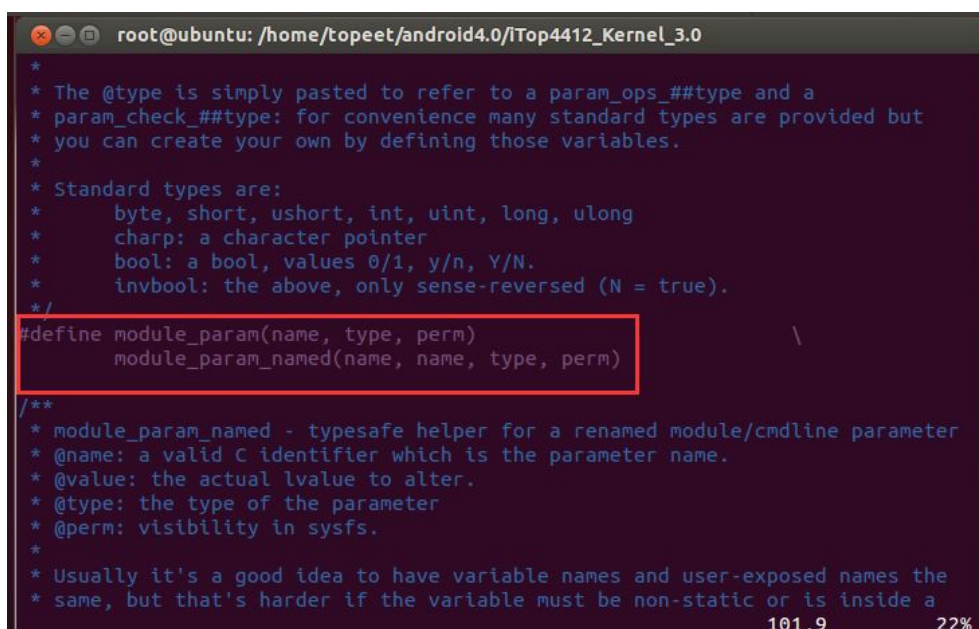
在应用程序中，可以通过 main 函数向其中传参数，这个功能大家用起来已经很熟练了。

实际上在加载模块的时候也是可以向其中传参数的。

在头文件“include/linux/moduleparam.h”中包含了向模块传参数的函数。这个功能是集成的，在任何 linux 系统之中都可以使用。

参数传递有两个函数，分别是函数 module_param 和函数 module_param_array。

函数 module_param 支持单个参数传递，在头文件中，如下图所示。



```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0
*
* The @type is simply pasted to refer to a param_ops_##type and a
* param_check_##type: for convenience many standard types are provided but
* you can create your own by defining those variables.
*
* Standard types are:
*   byte, short, ushort, int, uint, long, ulong
*   charp: a character pointer
*   bool: a bool, values 0/1, y/n, Y/N.
*   invbool: the above, only sense-reversed (N = true).
*/
#define module_param(name, type, perm) \
    module_param_named(name, name, type, perm)
/**
* module_param_named - typesafe helper for a renamed module/cmdline parameter
* @name: a valid C identifier which is the parameter name.
* @value: the actual lvalue to alter.
* @type: the type of the parameter
* @perm: visibility in sysfs.
*
* Usually it's a good idea to have variable names and user-exposed names the
* same, but that's harder if the variable must be non-static or is inside a
101,9 22%
```

如上图所示，这个宏定义函数有三个参数分别如下。

参数 name，模块参数的名称；

参数 type，模块参数的数据类型（支持 int long short uint ulong ushort 类型）；

参数 perm，模块参数的访问权限（S_IRUSR 参数表示所有文件所有者可读）。

函数 module_param_array 支持多个参数传递，在头文件中，如下图所示。

```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0
extern int param_get_invbool(char *buffer, const struct kernel_param *kp);
#define param_check_invbool(name, p) __param_check(name, p, bool)

/**
 * module_param_array - a parameter which is an array of some type
 * @name: the name of the array variable
 * @type: the type, as per module_param()
 * @nump: optional pointer filled in with the number written
 * @perm: visibility in sysfs
 *
 * Input and output are as comma-separated values. Commas inside values
 * don't work properly (eg. an array of charp).
 *
 * ARRAY_SIZE(@name) is used to determine the number of elements in the
 * array, so the definition must be visible.
 */
#define module_param_array(name, type, nump, perm) \
    module_param_array_named(name, name, type, nump, perm)

/**
 * module_param_array_named - renamed parameter which is an array of some type
 * @name: a valid C identifier which is the parameter name
 * @array: the name of the array variable
 */
```

如上图所示，如上图所示，这个宏定义函数有四个参数分别如下。

参数 name，模块参数的名称；

参数 type，模块参数的数据类型（支持 int long short uint ulong ushort 类型）；

参数 nump，保存参数的数量；

参数 perm，模块参数的访问权限（S_IRUSR 参数表示所有文件所有者可读）。

这里再介绍一下参数 perm，参数 perm 表示此参数在 sysfs 文件系统中对应的文件节点的属性，其权限在“include/linux/stat.h”中有定义，如下图所示。

```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0
#define S_ISDIR(m) (((m) & S_IFMT) == S_IFDIR)
#define S_ISCHR(m) (((m) & S_IFMT) == S_IFCHR)
#define S_ISBLK(m) (((m) & S_IFMT) == S_IFBLK)
#define S_ISFIFO(m) (((m) & S_IFMT) == S_IFIFO)
#define S_ISSOCK(m) (((m) & S_IFMT) == S_IFSOCK)

#define S_IRWXU 00700
#define S_IRUSR 00400
#define S_IWUSR 00200
#define S_IXUSR 00100

#define S_IRWXG 00070
#define S_IRGRP 00040
#define S_IWGRP 00020
#define S_IXGRP 00010

#define S_IRWXO 00007
#define S_IROTH 00004
#define S_IWOTH 00002
#define S_IXOTH 00001

#endif
```

部分常用参数权限解释如下。

- #defineS_IRUSR 00400 文件所有者可读
- #defineS_IWUSR00200 文件所有者可写
- #defineS_IXUSR 00100 文件所有者可执行
- #defineS_IRGRP00040 与文件所有者同组的用户可读
- #defineS_IWGRP00020
- #defineS_IXGRP 00010
- #defineS_IROTH 00004 与文件所有者不同组的用户可读
- #defineS_IWOTH00002
- #defineS_IXOTH 00001

其它的可以使用下面的方法来判断：

可以将数字最后三位转化为二进制:xxx xxx xxx,高位往低位依次看,第一位为 1 表示文件所有者可读,第二位为 1 表示文件所有者可写,第三位为 1 表示文件所有者可执行;接下来三位表示文件所有者同组成员的权限;再下来三位为不同组用户权限。

接着在 02_DriverModule_01 例程中基础上做本实验。

先修改一下 Makefile。

将原来的 “rm -rf *.o” 改为 “rm -rf *.mod.c *.o *.order *.ko *.mod.o *.symvers” 。

将 “mini_linux_module” 改为 “module_param” 。

如下图所示。

```
#!/bin/bash
#通知编译器我们要编译模块的哪些源码
#这里是编译itop4412_hello.c这个文件编译成中间文件mini_linux_module.o
obj-m += module_param.o

#源码目录变量，这里用户需要根据实际情况选择路径
#作者是将Linux的源码拷贝到目录/home/topeet/android4.0下并解压的
KDIR := /home/topeet/android4.0/iTop4412_Kernel_3.0

#当前目录变量
PWD ?= $(shell pwd)

#make命名默认寻找第一个目标
#make -C就是指调用执行的路径
#$(KDIR) Linux源码目录，作者这里指的是/home/topeet/android4.0/iTop4412_Kernel_3.0
#$(PWD) 当前目录变量
#modules要执行的操作
all:
    make -C $(KDIR) M=$(PWD) modules

#make clean执行的操作是删除后缀为o的文件
clean:
    rm -rf *.mod.c *.o *.order *.ko *.mod.o *.symvers
```

接着将 02_DriverModule_01 的例程改为 module_param.c。

如下图所示，添加传参数功能的头文件 “linux/moduleparam.h” 和 “linux/stat.h”。

```
#include <linux/init.h>
/*包含初始化宏定义的头文件,代码中的module_init和module_exit在此文件中*/
#include <linux/module.h>
/*包含初始化加载模块的头文件,代码中的MODULE_LICENSE在此头文件中*/

/*定义module_param module_param_array的头文件*/
#include <linux/moduleparam.h>
/*定义module_param module_param_array中perm的头文件*/
#include <linux/stat.h>
```

如下图所示，然后调用函数 module_param 和 module_param_array 接收参数。


```
MODULE_LICENSE("Dual BSD/GPL");  
/*声明是开源的，没有内核版本限制*/  
MODULE_AUTHOR("iTOPEET_dz");  
/*声明作者*/  
  
static int module_arg1,module_arg2;  
static int int_array[50];  
static int int_num;  
  
module_param(module_arg1,int,S_IRUSR);  
  
module_param(module_arg2,int,S_IRUSR);  
  
module_param_array(int_array,int,&int_num,S_IRUSR);
```

接着添加代码在初始化的时候将数据打印出去，如下图所示。

```
static int hello_init(void)  
{  
    int i;  
  
    printk(KERN_EMERG "module_arg1 is %d!\n",module_arg1);  
    printk(KERN_EMERG "module_arg2 is %d!\n",module_arg2);  
  
    for(i=0;i<int_num;i++){  
        printk(KERN_EMERG "int_array[%d] is %d!\n",i,int_array[i]);  
    }  
  
    printk(KERN_EMERG "Hello World enter!\n");  
    /*打印信息，KERN_EMERG表示紧急信息*/  
    return 0;  
}
```

在 Ubuntu 系统下新建 module_param 文件夹，将写好的 module_param.c、编译脚本以及应用拷贝到 module_param 文件夹下，如下图所示。

```
root@ubuntu: /home/topeet/module_param
root@ubuntu: /home/topeet# mkdir module_param
root@ubuntu: /home/topeet# cd module_param/
root@ubuntu: /home/topeet/module_param# ls
Makefile module_param.c
root@ubuntu: /home/topeet/module_param#
```

使用 Makefile 命令编译驱动命令 “Make” 编译应用，如下图所示。

```
root@ubuntu: /home/topeet/module_param
root@ubuntu: /home/topeet# mkdir module_param
root@ubuntu: /home/topeet# cd module_param/
root@ubuntu: /home/topeet/module_param# ls
Makefile module_param.c
root@ubuntu: /home/topeet/module_param# make
make -C /home/topeet/android4.0/iTop4412_Kernel_3.0 M=/home/topeet/module_param
modules
make[1]: Entering directory `/home/topeet/android4.0/iTop4412_Kernel_3.0'
CC [M] /home/topeet/module_param/module_param.o
Building modules, stage 2.
MODPOST 1 modules
CC /home/topeet/module_param/module_param.mod.o
LD [M] /home/topeet/module_param/module_param.ko
make[1]: Leaving directory `/home/topeet/android4.0/iTop4412_Kernel_3.0'
root@ubuntu: /home/topeet/module_param# ls
Makefile module_param.ko module_param.mod.o modules.order
module_param.c module_param.mod.c module_param.o Module.symvers
root@ubuntu: /home/topeet/module_param#
```

将上图中的文件 module_param.ko 拷贝到 U 盘。

启动开发板，将 U 盘插入开发板，使用命令 “mount /dev/sda1 /mnt/udisk/” 加载 U 盘。

使用命令 “insmod /mnt/udisk/module_param.ko module_arg1=10

module_arg2=20 int_array=11,12,13,14,15,16,17,18” 加载驱动 module_param.ko , 如下图所示。

```
COM1
[root@iTOP-4412]#
[root@iTOP-4412]# mount /dev/sda1 /mnt/udisk/
[root@iTOP-4412]# [ 44.401868] CPU3: shutdown

[root@iTOP-4412]# insmod /mnt/udisk/module_param.ko module_arg1=10 module_arg2=20
0 int_array=11,12,13,14,15,16,17,18
[ 63.510332] module_arg1 is 10!
[ 63.511906] module_arg2 is 20!
[ 63.514944] int_array[0] is 11!
[ 63.519408] int_array[1] is 12!
[ 63.521238] int_array[2] is 13!
[ 63.524318] int_array[3] is 14!
[ 63.527482] int_array[4] is 15!
[ 63.530604] int_array[5] is 16!
[ 63.533693] int_array[6] is 17!
[ 63.536830] int_array[7] is 18!
[ 63.539941] Hello World enter!
[root@iTOP-4412]#
```

另外还可以使用 cat 命令在系统的 module 下查参数。

使用命令 “cat /sys/module/module_param/parameters/xxx” , 如下图所示。

```
[root@iTOP-4412]# cat /sys/module/module_param/parameters
cat: read error: Is a directory
[root@iTOP-4412]# cat /sys/module/module_param/parameters/
int_array module_arg1 module_arg2
[root@iTOP-4412]# cat /sys/module/module_param/parameters/int_array
11,12,13,14,15,16,17,18
[root@iTOP-4412]# cat /sys/module/module_param/parameters/module_arg1
10
[root@iTOP-4412]# cat /sys/module/module_param/parameters/module_arg2
20
[root@iTOP-4412]#
```

另外在 Ubuntu 的 module_param 文件夹下可以使用以下清除命令 “make clean” , 可以看到除了 Makefile 文件和*.c 文件其它都被清除了。如下图所示。

```
root@ubuntu: /home/topeet/module_param
root@ubuntu:/home/topeet# mkdir module_param
root@ubuntu:/home/topeet# cd module_param/
root@ubuntu:/home/topeet/module_param# ls
Makefile  module_param.c
root@ubuntu:/home/topeet/module_param# make
make -C /home/topeet/android4.0/iTop4412_Kernel_3.0 M=/home/topeet/module_param
modules
make[1]: Entering directory `/home/topeet/android4.0/iTop4412_Kernel_3.0'
  CC [M]  /home/topeet/module_param/module_param.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/topeet/module_param/module_param.mod.o
  LD [M]  /home/topeet/module_param/module_param.ko
make[1]: Leaving directory `/home/topeet/android4.0/iTop4412_Kernel_3.0'
root@ubuntu:/home/topeet/module_param# ls
Makefile      module_param.ko      module_param.mod.o    modules.order
module_param.c  module_param.mod.c    module_param.o         Module.symvers
root@ubuntu:/home/topeet/module_param# make clean
rm -rf *.mod.c *.o *.order *.ko *.mod.o *.symvers
root@ubuntu:/home/topeet/module_param# ls
Makefile  module_param.c
root@ubuntu:/home/topeet/module_param#
```