<http://www.cnblogs.com/hoys/archive/2011/04/10/2011459.html>

Linux 提供了一种通过 bootloader 向其传输启动参数的功能，内核开发者可以通过这种方式来向内核传输数据，从而控制内核启动行为。

通常的使用方式是，定义一个分析参数的函数，而后使用内核提供的宏 \_\_setup把它注册到内核中，该宏定义在 linux/init.h 中，因此要使用它必须包含该头文件：

\_\_setup("para\_name=", parse\_func)

para\_name 为参数名，parse\_func 为分析参数值的函数，它负责把该参数的值转换成相应的内核变量的值并设置那个内核变量。内核为整数参数值的分析提供了函数 get\_option 和 get\_options，前者用于分析参数值为一个整数的情况，而后者用于分析参数值为逗号分割的一系列整数的情况，对于参数值为字符串的情况，需要开发者自定义相应的分析函数。在源代码包中的内核程序kern-boot-params.c 说明了三种情况的使用。该程序列举了参数为一个整数、逗号分割的整数串以及字符串三种情况，读者要想测试该程序，需要把该程序拷贝到要使用的内核的源码目录树的一个目录下，为了避免与内核其他部分混淆，作者建议在内核源码树的根目录下创建一个新目录，如 examples，然后把该程序拷贝到 examples 目录下并重新命名为 setup\_example.c，并且为该目录创建一个 Makefile 文件：

obj-y = setup\_example.o

Makefile 仅许这一行就足够了，然后需要修改源码树的根目录下的 Makefile文件的一行，把下面行

core-y := usr/

修改为

core-y := usr/ examples/

注意：如果读者创建的新目录和重新命名的文件名与上面不同，需要修改上面所说 Makefile 文件相应的位置。 做完以上工作就可以按照内核构建步骤去构建新的内核，在构建好内核并设置好lilo或grub为该内核的启动条目后，就可以启动该内核，然后使用lilo或grub的编辑功能为该内核的启动参数行增加如下参数串：

setup\_example\_int=1234 setup\_example\_int\_array=100,200,300,400 setup\_example\_string=Thisisatest

当然，该参数串也可以直接写入到lilo或grub的配置文件中对应于该新内核的内核命令行参数串中。读者可以使用其它参数值来测试该功能。

下面是作者系统上使用上面参数行的输出：

setup\_example\_int=1234

setup\_example\_int\_array=100,200,300,400

setup\_example\_int\_array includes 4 intergers

setup\_example\_string=Thisisatest

读者可以使用$dmesg | grep setup 来查看该程序的输出。

//filename: kern-boot-params.c

#include <linux/kernel.h>

#include <linux/init.h>

#include <linux/string.h>

#define MAX\_SIZE 5

static int setup\_example\_int;

static int setup\_example\_int\_array[MAX\_SIZE];

static char setup\_example\_string[16];

static int \_\_init parse\_int(char \* s)

{

int ret;

ret = get\_option(&s, &setup\_example\_int);

if (ret == 1) {

printk("setup\_example\_int=%d\n", setup\_example\_int);

}

return 1;

}

static int \_\_init parse\_int\_string(char \*s)

{

char \* ret\_str;

int i;

ret\_str = get\_options(s, MAX\_SIZE, setup\_example\_int\_array);

if (\*ret\_str != '\0') {

printk("incorrect setup\_example\_int\_array paramters: %s\n", ret\_str);

}

else {

printk("setup\_example\_int\_array=");

for (i=1; i<MAX\_SIZE; i++) {

printk("%d", setup\_example\_int\_array[i]);

if (i < (MAX\_SIZE -1)) {

printk(",");

}

}

printk("\n");

printk("setup\_example\_int\_array includes %d intergers\n", setup\_example\_int\_array[0]);

}

return 1;

}

static int \_\_init parse\_string(char \*s)

{

if (strlen(s) > 15) {

printk("Too long setup\_example\_string parameter, \n");

printk("maximum length is less than or equal to 15\n");

}

else {

memcpy(setup\_example\_string, s, strlen(s) + 1);

printk("setup\_example\_string=%s\n", setup\_example\_string);

}

return 1;

}

/\*宏\_\_setup()将分析参数的函数注册到内核中\*/

\_\_setup("setup\_example\_int=", parse\_int);

\_\_setup("setup\_example\_int\_array=", parse\_int\_string);

\_\_setup("setup\_example\_string=", parse\_string);