Practice linux kernal module

Danny Deng

Version 1.0

1 Command in terminal

See the command usage and information

```
$ man {command}
```

Example:

```
$ man insmod
```

Result

Insert a kernal module

```
$ sudo insmod {module name (*.ko)}
```

Show a kernal module info

```
$ sudo modinfo {module name (*.ko)}
```

Print or control the kernel ring buffer

```
$ sudo dmesg
```

Simple program to remove a module from the Linux Kernel

```
$ sudo rmmod {module name}
```

2 Hello world

Example code below

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

MODULE_LICENSE("GPL");
static int  test_hello_init (void) {
    printk(KERN_INFO"%s: In init \n", __func__);
    return 0;
}

static void  test_hello_exit (void) {
    printk(KERN_INFO"%s: In exit \n", __func__);
}

module_init( test_hello_init );
module_exit( test_hello_exit );
```

Include header and define the LICENSE(option)

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

MODULE_LICENSE("GPL");
```

When insert module, it will call the initial function

```
static int  test_hello_init (void) {
    printk(KERN_INFO"%s: In init \n", __func__);
    return 0;
}

module_init( test_hello_init );
```

When rmmod the kernel module, it will call the exit function

```
static void  test_hello_exit (void) {
    printk(KERN_INFO"%s: In exit \n", __func__);
}

module_exit( test_hello_exit );
```

Build steps

1. Write the Makefile

```
obj-m := hello_world.o

all:
    make -C /lib/modules/`uname -r`/build M=${PWD} modules

clean:
    make -C /lib/modules/`uname -r`/build M=${PWD} clean
```

2. Build

```
    $ make all
```

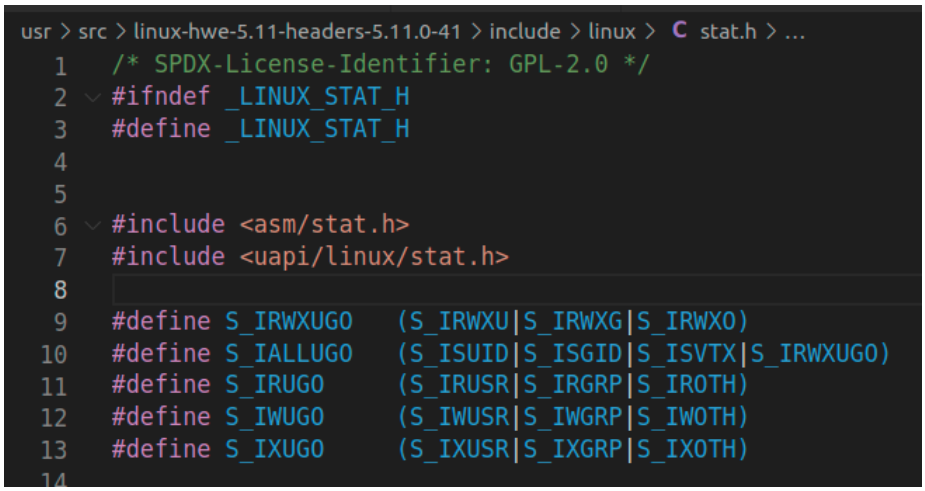
3 Passing parameters to linux kernel module

Using *module_param* function to declare parameters

Example code below

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

MODULE_LICENSE("GPL");

char *name = "Danny";
int count = 0;
module_param(name, charp, S_IRUGO);
module_param(count, int, S_IRUGO);


static int test_arguments_init(void) {
    printk(KERN_INFO"%s: In init\n", __func__);
    printk(KERN_INFO"%s: name: %s\n", __func__, name);
    printk(KERN_INFO"%s: pass count: %d\n", __func__, count);

    return 0;
}


static void test_exit(void) {
    printk(KERN_INFO"%s: In exit: \n", __func__);
}



module_init(test_arguments_init);
module_exit(test_exit);

MODULE_AUTHOR("Danny Deng");
MODULE_DESCRIPTION("Argument parsing example");
```

Each parameters can declare different permission



The macro value define below

```
#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
```

S_I + R/W/X + USR/GRP/OTH

- R: Read
- W: Write
- X: Execute
- USR: User
- GRP: Group
- OTH: Other

Insert the kernel module, and feed the parameter

```
$ sudo insmod passing_simple_count.ko count=10 name=Danny
```

Remove the kernel module

```
$ sudo rmsmod passing_simple_count
```

Show the result

```
$ dmesg
```

```
danny@dannyPC:~/project/kernel/03_passing_parameter$ sudo insmod passing_simple_count.ko count=10 name=Danny
danny@dannyPC:~/project/kernel/03_passing_parameter$ sudo rmmod passing_simple_count
danny@dannyPC:~/project/kernel/03_passing_parameter$ dmesg
[ 3654.716861] test_arguments_init: In init
[ 3654.716864] test_arguments_init: name: Danny
[ 3654.716865] test_arguments_init: pass count: 10
[ 3659.162359] test_exit: In exit:
danny@dannyPC:~/project/kernel/03_passing_parameter$
```