

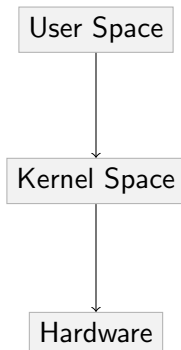
An Introduction to Linux Modules

March 3, 2010

Required Software

- gcc
- make
- kernel headers

User Space versus Kernel Space



Common Module Commands

insmod

Insert a module

modprobe

Intelligently insert a module

rmmod

Remove a module

lsmod

List loaded modules

mknod

Create a special file

Special or device files, for example:

- /dev/dsp
- /dev/lp0
- /dev/sda

Block Devices versus Character Devices

Character Devices

- One character at a time
- Streams of data
- Sequential access
- Mice, keyboards, modems, etc.

Block Devices

- Buffered I/O
- One block at a time
- Random access
- Hard drive, CD-Rom, etc.

Major and Minor Numbers

Major

Distinguishes class of devices

Minor

Distinguishes subdevices

Kernel Ring Buffer



Boilerplate

```
#include <linux/kernel.h> /* printk(), KERN_ALERT, KERN_INFO */
#include <linux/module.h>
#include <linux/fs.h> /* file_operations structure */
#include <linux/init.h> /* __init, __exit macros */
#include <linux/uaccess.h> /* Needed for copy_to_user */

int major;
int alpha = 0;
```

Open and Release

```
int alphabet_open(struct inode *inode, struct file *filp)
{
    return 0;
}

int alphabet_release(struct inode *inode, struct file *filp)
{
    return 0;
}
```

init

```
int __init alphabet_init(void)
{
    major = register_chrdev(0, "alphabet", &alphabet_fops);

    if (major < 0) {
        printk(KERN_ALERT "Alpha: register fail, %d\n", major);
        return major;
    }

    printk(KERN_INFO "Alpha: init, assign major %d\n", major);
    return 0;
}

module_init(alphabet_init);
```

exit

```
void __exit alphabet_exit(void)
{
    unregister_chrdev(major, "alphabet");
    printk(KERN_INFO "Alpha: exit, release major %d\n", major);
}

module_exit(alphabet_exit);
```

```
struct file_operations alphabet_fops = {  
    .open= alphabet_open ,  
    .read= alphabet_read ,  
    .release= alphabet_release  
};
```

read

```
int alphabet_read(struct file *filp, char *ubuf, size_t count,
                 loff_t *f_pos)
{
    char buffer[3];

    if(*f_pos != 0)
        return 0;

    buffer[0] = 65 + alpha;
    buffer[1] = '\n';
    buffer[2] = '\0';
    alpha = (alpha + 1) % 26;
    if(copy_to_user(ubuf, buffer, 3))
        return -EFAULT;
    *f_pos += 3;
    return 3;
}
```