编写一个简单的内核模块

实验目的

Linux 操作系统的内核是单一体系结构(monolithic kernel)的，也就是说，整个内核是一个单独的非常大的程序。这样，系统的速度和性能都很好，但是可扩展性和维护性就相对比较差。为了弥补单一体系结构的这种缺陷，Linux 操作系统使用了一种全新的机制－—模块机制，用户可以根据需要，在不需要对内核重新编译的情况下，模块能动态地载入内核或从内核移出。

本实验通过分析代码，学习 Linux 是如何实现模块机制的；通过一个实例，掌握如何编写模块程序并进一步掌握内核模块的机理。

实验内容

编写一个内核模块 helloworld。当用 insmod 命令加载模块后，会显示

Hello World ！

此模块与 Linux 内核模块机制兼容，可以用 lsmod 命令显示模块信息，可以用 rmmod 命令删除该模块。

实验提示

helloworld.c

```
#define MODULE
#include <linux/module.h>

int init_module(void)
{
   printk("<1> Hello World!\n");
   return 0;
}

void cleanup_module(void)
{
   printk(" <1>Goodbye!\n");
}
```

说明：

1. 代码的第一行#define MODULE 首先明确这是一个模块。任何模块程序的编写都需要包含 linux/module.h 这个头文件，这个文件包含了对模块的结构定义以及模块的版本控制。文件里的主要数据结构我们会在后面详细介绍。
2. 函数 init_module( )和函数 cleanup_module( )是模块编程中最基本的也是必须的两个函数。init_module 向内核注册模块所提供的新功能；cleanup_module 负责注销所有由模块注册的功能。
3. 注意我们在这儿使用的是 printk 函数(不要习惯性地写成 printf)，printk 函数是由 linux 内核定义的，功能与 printf 相似。字符串<1>表示消息的优先级，printk 的一个特点就是它对于不同优先级的消息进行不同的处理，之所以要在这儿使用高优先级是因为默认优先级的消息可能不能显示在控制台上。这个问题我们就不详细讲了，你可以用 man 命令寻求帮助。

接下来，我们就要编译和加载这个模块了。在前面的章节里我们已经学习了如何使用 gcc，现在还要注意的一点就是：确定你现在是超级用户，因为只有超级用户才能加载和卸载模块。

```
root# gcc –c helloworld.c
root# insmod helloworld.o
Hello World!
```

这个时候，helloworld 模块就已经加载到内核中了。我们可以使用 lsmod 命令查看。lsmod 命令的作用是告诉我们所有在内核中运行的模块的信息，包括模块的名称，占用空间的大小，使用计数以及当前状态和依赖性。

```
root# lsmod
Module      Size      Used   by
helloworld   464      0     (unused)
…
```

最后，我们要卸载这个模块。

```
root# rmmod helloworld
Goodbye!
```

如果这时候我们再使用 lsmod 查看，会发现 helloworld 模块已经不在了。

关于 insmod 和 rmmod 这两个命令，我现在只能简单的告诉你他们是两个用于把模块插入内核和从内核移走模块的实用程序。前面用到的 insmod, rmmod 和 lsmod 都属于 modutils 模块实用程序。