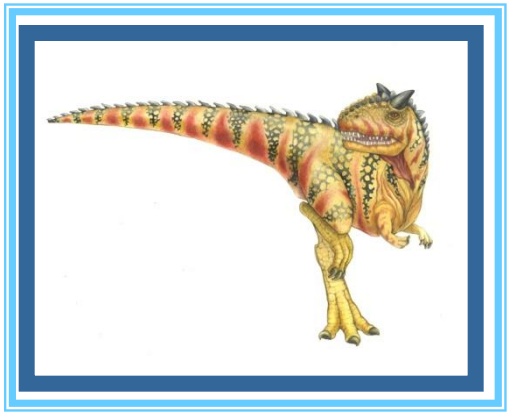
内核模块



内核模块Loadable Kernel Module

本章内容：

- 什么是内核模块

  内核模块机制

  内核模块与应用程序的区别

- 内核模块的使用

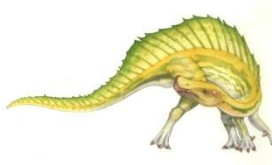
  举例，helloworld.c

  insmod

  lsmod

  rmmod

  ksyms

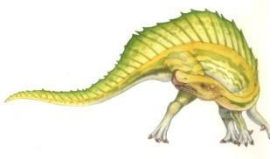


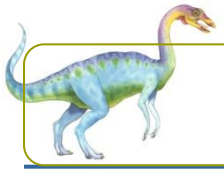
什么是内核模块

- Linux操作系统的内核是单一体系结构(monolithic kernel)的。
- Linux操作系统使用了一种全新的内核模块机制。用户可以根据需要，在不需要对内核重新编译的情况下，模块能动态地装入内核或从内核移出。

什么是内核模块

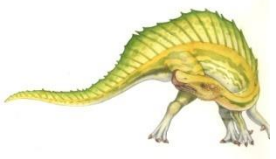
- 模块在内核空间运行，实际上是一种目标对象文件，没有链接，不能独立运行，但是其代码可以在运行时链接到系统中作为内核的一部分运行或从内核中取下，从而可以动态扩充内核的功能
- 这种目标代码通常由一组函数和数据结构组成,如用来实现一种文件系统、一个驱动程序或其他内核上层功能。
- 模块完整叫法：动态可加载内核模块（Loadable Kernel Module LKM）

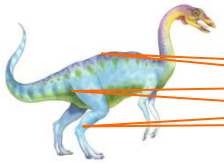




内核模块的优点

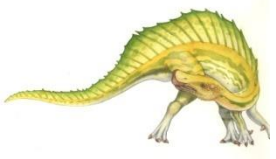
- 使得内核更加紧凑和灵活。
- 修改内核时，不必全部重新编译整个内核。系统如果需要使用新模块，只要编译相应的模块，然后使用insmod将模块插入即可。
- 模块不依赖于某个固定的硬件平台。
- 模块的目标代码一旦被链接到内核，它的作用域和静态链接的内核目标代码完全等价。





内核模块的缺点

- 由于内核所占用的内存是不会被换出的，所以链接进内核的模块会给整个系统带来一定的性能和内存利用方面的损失。
- 装入内核的模块就成为内核的一部分，可以修改内核中的其他部分，因此，模块的使用不当会导致系统崩溃。
- 为了让内核模块能访问所有内核资源，内核必须维护符号表，并在装入和卸载模块时修改符号表。
- 模块会要求利用其它模块的功能，所以，内核要维护模块之间的依赖性。



helloworld.c

- helloworld.c

```
#define MODULE
#include <linux/module.h>
int init_module(void)  {
    printk(“ Hello World!\n”);
    return 0;
}
void cleanup_module(void) {
    printk(“Goodbye!\n”);
}
MODULE_LICENSE(“GPL”);
```

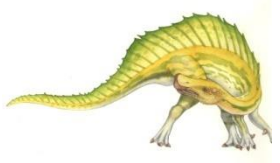
helloworld.c

- 在编译内核模块前，先准备一个Makefile文件：

```
TARGET = helloworld
KDIR = /lib/modules/$(shell uname -r)/build
PWD = $(shell pwd)
   obj-m += $(TARGET).o
default:
   make -C $(KDIR) M=$(PWD) modules
```

- 然后输入命令make：

```
#make
```



helloworld.c

- insmod命令加载模块

  # insmod helloworld.ko

  Hello World!

- 使用lsmod命令查看 模块

  # lsmod

  Module Size Used by

  helloworld 464 0 (unused)

  …

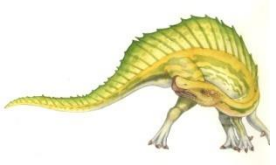
- rmmod 命令卸载模块

  # rmmod helloworld

  Goodbye!

•Linux用户模式，重定向printk输出。查看信息：cat /var/log/syslog 或dmesg | tail -5

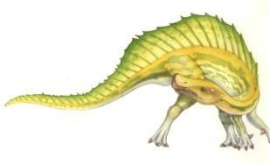
•在GNOME或KDE下可以用ALT-CTRL-F1~6控制台下查看，ALT-CTRL-F7返回。 //修改虚拟机下hotkey



insmod 命令

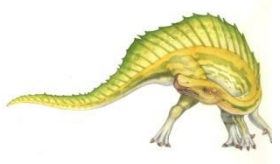
- 调用insmod程序将把需要插入的模块以目标代码的形式插入到内核中。
- 在插入的时候，insmod会自动运行在init_module()函数中定义的过程。注意，只有超级用户才能使用这个命令
- 格式：

  insmod [path]modulename.ko



insmod 命令

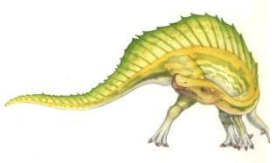
- insmod程序完成下面一系列工作：
  1. 从命令行中读入要链接的模块名，通常是扩展名为“.ko”，elf格式的目标文件。
  2. 确定模块对象代码所在文件的位置。通常这个文件都是在lib/modules的某个子目录中。
  3. 计算存放模块代码、模块名和module对象所需要的内存大小。
  4. 在用户空间中分配一个内存区，把module对象、模块名以及为正在运行的内核所重定位的模块代码拷贝到这个内存里。其中，module对象中的init域指向这个模块的入口函数重新分配到的地址；exit域指向出口函数所重新分配的地址。
  5. 调用init_module()，向它传递上面所创建的用户态的内存区的地址，其实现过程我们已经详细分析过了。
  6. 释放用户态内存，整个过程结束。



lsmod命令

- lsmod 显示当前系统中正在使用的模块信息。
- 实际上这个程序的功能就是读取/proc文件系统中的文件/proc/modules中的信息。所以这个命令和cat /proc/modules等价。
- 格式：

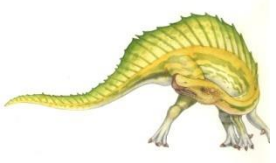
  lsmod



rmmod命令

- rmmod将已经插入内核的模块从内核中移出
- rmmod会自动运行在cleanup_module()函数中定义的过程
- 格式：

  rmmod [path]modulename



ksyms

- 显示内核符号和模块符号表的信息，可以读取/proc/kallsyms文件。

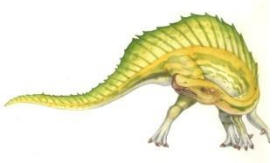
  cat /proc/kallsyms

modprobe 命令

- modprobe是自动根据模块之间的依赖性插入模块的程序
- 按需装入的模块加载方法会调用这个程序来实现按需装入的功能。举例来讲，如果模块A依赖模块B，而模块B并没有加载到内核里，当系统请求加载模块A时，modprobe程序会自动将模块B加载到内核。

实验

- 2.编写一个Linux的内核模块，其功能是遍历进程，要求输出系统中：每个进程的名字、进程pid、进程的状态、父进程的名字；统计系统中进程个数，统计系统中TASK_RUNNING、TASK_INTERRUPTIBLE、TASK_UNINTERRUPTIBLE、TASK_ZOMBIE、TASK_STOPPED等状态进程的个数。
- 实验指导：
  - 如何编写内核模块程序及编译、安装内核模块，可以参考“边干边学—Linux内核指导”教材第13章，及**课件**。
  - 每个进程的进程名字、pid、进程状态、父进程的指针等在task-struct结构的字段中。在内核中使用printk函数打印有关变量的值。遍历进程可以使用next_task宏，init_task进程为0号进程。遍历task-struct结构参阅“边干边学—Linux内核指导”教材11.2节;进程方法可以参阅“边干边学—Linux内核指导”教材11.6节。



练习

- 练习 P.352