《操作系统》课第 11 次实验报告

学院:	软件学院
姓名:	郁万祥
学号:	2013852
邮箱:	yuwanxiang0114@163.com
时间:	2022.11.25

0. 开篇感言

以往的实验，我们发现，每次在对 linux 内核进行修改的时候（通常是增加新的系统调用），每次都要重新编译内核，进行内核模块的加载，虽然不进行 make clean 这个步骤的时候，会加快这个过程，但是总体所消耗的时间还是比较多的，这还是非常不方便的。那如果我们，既想使用内核模块的方式完成一些自定义的工作，同时又想使这个过程尽量快一些（这里是指就像平常编译 C 文件一样），那么可以使用什么方法呢，本次实验就是实现了这一个过程。

1. 实验题目

Linux kernel Module Development（1）

使用 linux 内核模型进行进程信息的打印

2. 实验目标

1、使用 C 语言，编写程序，实现一个 linux 内核模块，可以完成进程信息的打印。

2、使用 Makefile 文件，进行编译命令的编辑。

3. 原理方法

1、linux 内核模块：模块是在内核空间运行的程序，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，用来实现一种文件系统，一个驱动程序，或其它内核上层的功能。模块机制的完整叫法应该是动态可加载内核模块，一般就简称为模块。

2、进程信息：每个进程在内核中都有一个进程控制块(PCB)来维护进程相关的信息,Linux 内核的进程控制块是 task_struct 结构体。task_struct 是 Linux 内核的一种数据结构，它会被装载到 RAM 中并且包含着进程的信息。

4. 具体步骤

1、使用 c 语言编写文件，实现使用 linux 模块功能完成进程信息的输出。

打开(O) *hellokmodule.c ~/Desktop 保存(S)

Makefile | *hellokmodule.c

```c
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/sched/task.h>
#include <linux/sched/signal.h>
/* init function */Linux kernel Module Development
static int
hellokmodule_init(void)
{
        int sum = 0;
        struct task_struct *p;
        printk("%-10s %-20s %-6s %-6s %-6s\n","ywx2013852","Name","Pid","Father's Pid","Stat");
        for_each_process(p)
        {
                printk("%-10s %-20s %-6d %-6d %-6c\n","ywx2013852",p->comm,p->pid,p->parent->pid,task_state_to_char(p));
                sum = sum + 1;
        }
        printk("%-10s %-20s %-6d \n","ywx2013852","总共进程的数目为：",sum);
        return 0;
}
/* exit function - logs that the module is being removed */
static void
hellokmodule_exit(void)
{
        printk(KERN_ALERT "ALAL:simple module is being unloaded\n");
        printk("ALAL:Jiffies value: %lu\n", jiffies);
}
module_init(hellokmodule_init);
module_exit(hellokmodule_exit);
MODULE_LICENSE ("GPL");
MODULE_AUTHOR ("LKD");
MODULE_DESCRIPTION ("Simple Kernel Module");
MODULE_VERSION("1.01");
```

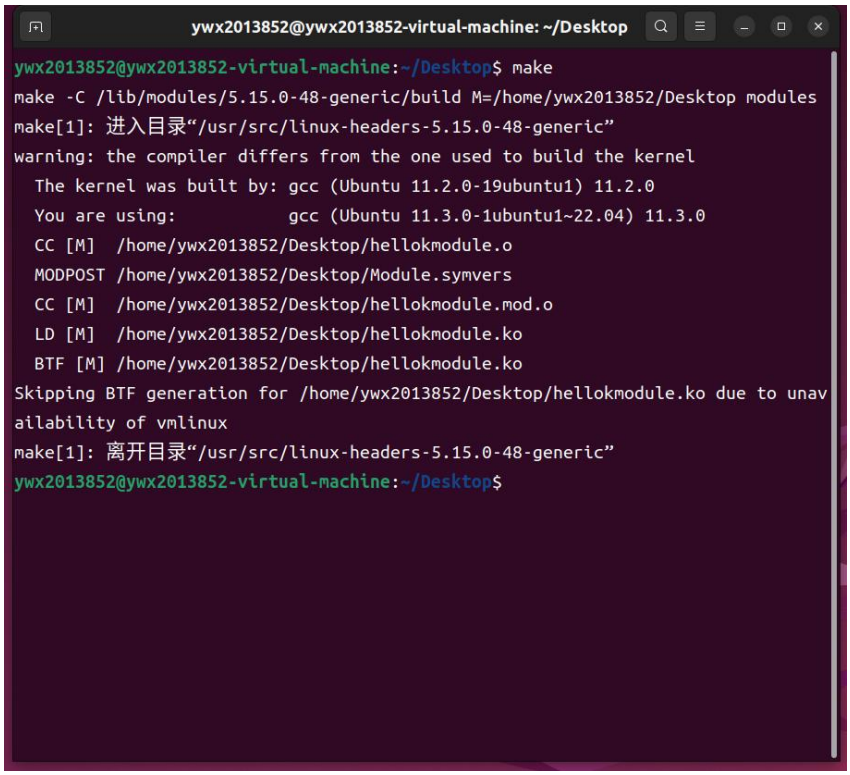
2、编辑 Makefile 文件，构造指令集。

打开(O) Makefile ~/Desktop 保存(S)

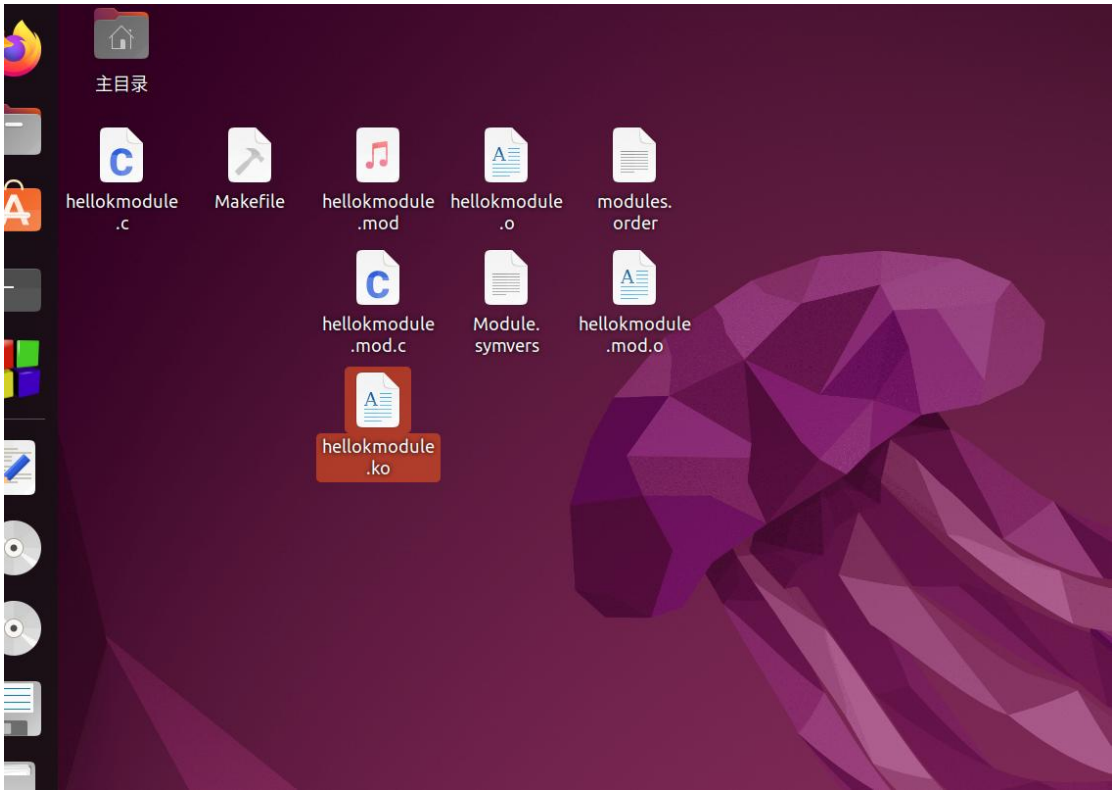
Makefile | *hellokmodule.c

```
ModuleName=hellokmodule
obj-m +=${ModuleName}.o
all:${ModuleName}.ko
${ModuleName}.ko:${ModuleName}.c
        make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
testload:${ModuleName}.ko
        sudo dmesg -C
        sudo insmod ${ModuleName}.ko
        sudo dmesg | grep ywx2013852
testunload:${ModuleName}.ko
        sudo dmesg -C
        sudo rmmod ${ModuleName}.ko
        sudo dmesg | grep ALAL
```

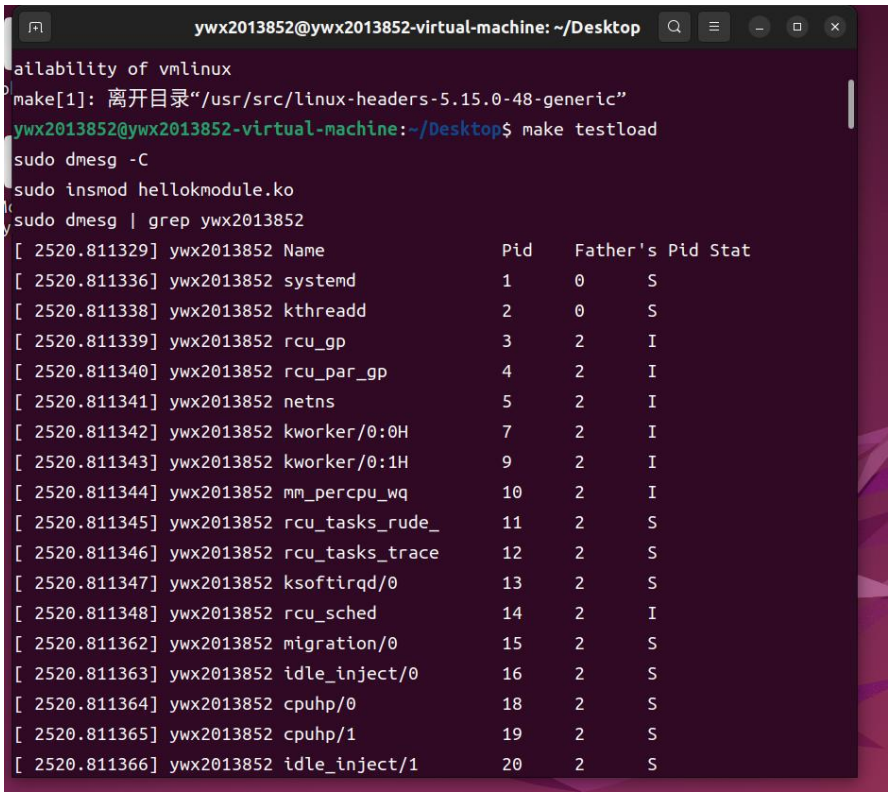
3、使用 make 命令对 linux module 进行编译。

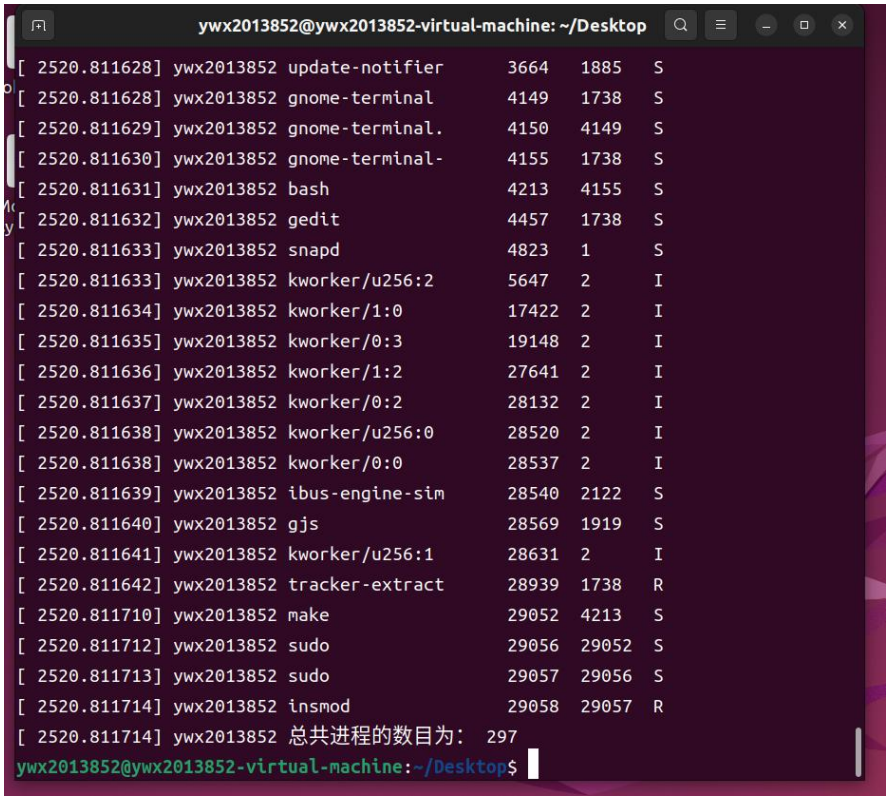

ywx2013852@ywx2013852-virtual-machine: ~/Desktop
ywx2013852@ywx2013852-virtual-machine:~/Desktop$ make
make -C /lib/modules/5.15.0-48-generic/build M=/home/ywx2013852/Desktop modules
make[1]: 进入目录“/usr/src/linux-headers-5.15.0-48-generic”
warning: the compiler differs from the one used to build the kernel
The kernel was built by: gcc (Ubuntu 11.2.0-19ubuntu1) 11.2.0
You are using: gcc (Ubuntu 11.3.0-1ubuntu1~22.04) 11.3.0
CC [M] /home/ywx2013852/Desktop/hellokmodule.o
MODPOST /home/ywx2013852/Desktop/Module.symvers
CC [M] /home/ywx2013852/Desktop/hellokmodule.mod.o
LD [M] /home/ywx2013852/Desktop/hellokmodule.ko
BTF [M] /home/ywx2013852/Desktop/hellokmodule.ko
Skipping BTF generation for /home/ywx2013852/Desktop/hellokmodule.ko due to unavailability of vmlinux
make[1]: 离开目录“/usr/src/linux-headers-5.15.0-48-generic”
ywx2013852@ywx2013852-virtual-machine:~/Desktop$

生成的文件:

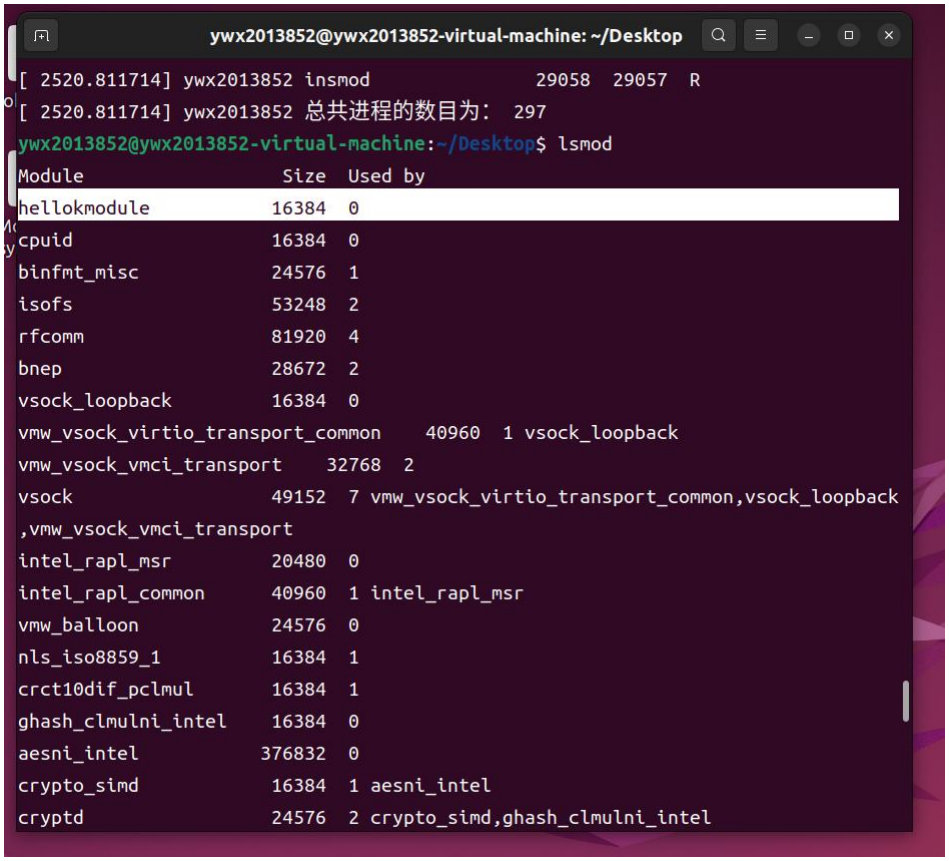

主目录
hellokmodule.c
Makefile
hellokmodule.mod
hellokmodule.o
modules.order
hellokmodule.mod.c
Module.symvers
hellokmodule.mod.o
hellokmodule.ko

4、使用 make testload 命令进行模块的插入，并且将输出到缓冲区的进程信息战术出来。



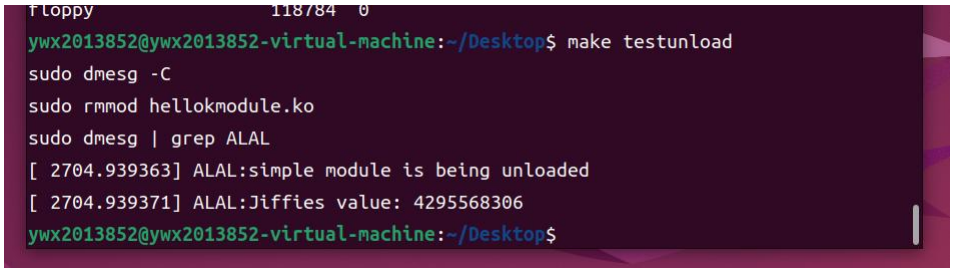


5、我们可以使用 lsmod 命令查看此时的 modules 里面是否含有刚刚添加的 hellokmodule

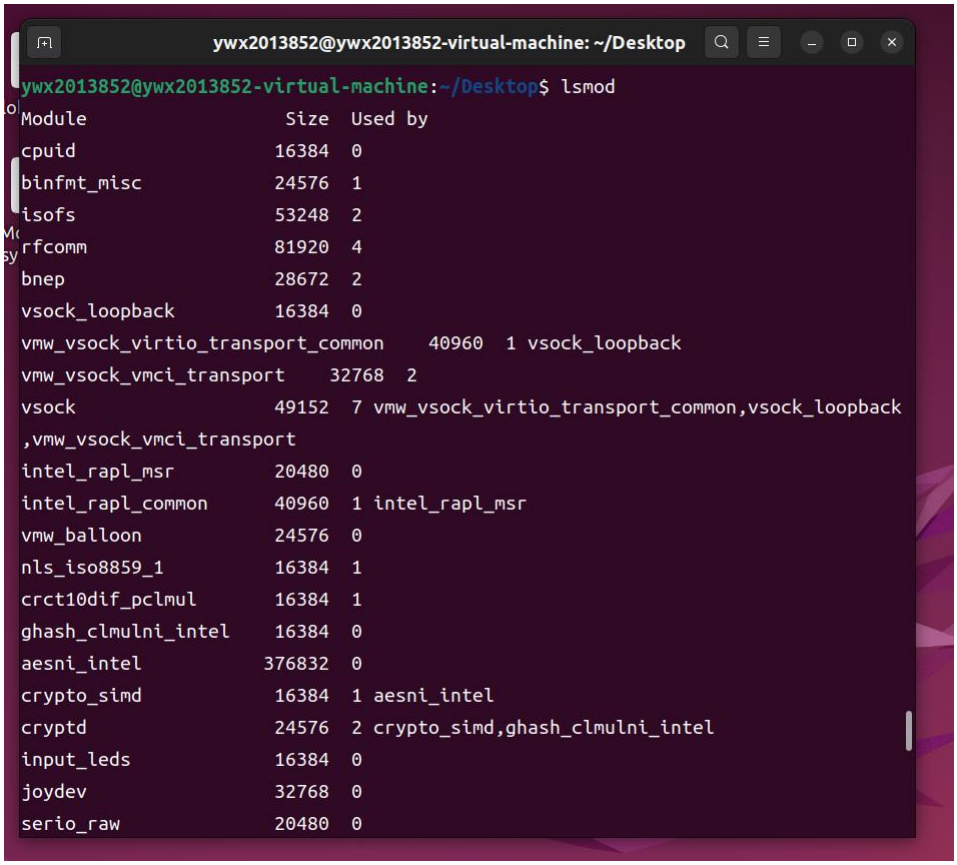


可以看到第一个 module 就是我们刚刚添加的自定义的 linux module

6、使用 make testunload 命令将插入的 linux 模块进行卸载。



7、此时我们在使用 lsmod 查看 modules



此时的 hellocmodule 已经被卸载。

5. 总结心得

之前的实验过程中，总是抱怨编译内核的时间过于长，每次都在 waiting，其实，问题总比困难多，出现了问题，自然就会产生相应的解决方式，无论是操作系统，还是其他的技术，都是在这个过程中不断发展进步的。

6. 参考资料

源码：

```
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/sched/task.h>
#include <linux/sched/signal.h>
```

```
/* init function */
static int
hellokmodule_init(void)
{
	int sum = 0;
	struct task_struct *p;
	printk("%-10s %-20s %-6s %-6s %-6s\n","ywx2013852","Name","Pid","Father's Pid","Stat");
	for_each_process(p)
	{

	printk("%-10s %-20s %-6d %-6d %-6c\n","ywx2013852",p->comm,p->pid,p->parent->pi
d,task_state_to_char(p));
		sum = sum + 1;
	}
	printk("%-10s %-20s %-6d \n","ywx2013852","总共进程的数目为：  ",sum);
	return 0;
}
/* exit function - logs that the module is being removed */
static void
hellokmodule_exit(void)
{
	printk(KERN_ALERT "ALAL:simple module is being unloaded\n");
	printk("ALAL:Jiffies value: %lu\n", jiffies);
}
module_init(hellokmodule_init);
module_exit(hellokmodule_exit);
MODULE_LICENSE ("GPL");
MODULE_AUTHOR ("LKD");
MODULE_DESCRIPTION ("Simple Kernel Module");
MODULE_VERSION("1.01");
```