



《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。



《操作系统》课程实验报告

```
打开(O) 文件 保存(S) 窗口 帮助

Makefile  *helloworldmodule.c
~/Desktop  *helloworldmodule.c

1 #include <linux/init.h>
2 #include <linux/kernel.h>
3 #include <linux/module.h>
4 #include <linux/sched/task.h>
5 #include <linux/sched/signal.h>
6 /* init function */Linux kernel Module Development
7 static int
8 helloworldmodule_init(void)
9 {
10     int sum = 0;
11     struct task_struct *p;
12     printk("%-10s %-20s %-6s %-6s\n", "ywx2013852", "Name", "Pid", "Father's Pid", "Stat");
13     for_each_process(p)
14     {
15         printk("%-10s %-20s %-6d %-6d %-6c\n", "ywx2013852", p->comm, p->pid, p->parent->pid, task_state_to_char(p));
16         sum = sum + 1;
17     }
18     printk("%-10s %-20s %-6d\n", "ywx2013852", "总共进程的数目为: ", sum);
19     return 0;
20 }
21 /* exit function - logs that the module is being removed */
22 static void
23 helloworldmodule_exit(void)
24 {
25     printk(KERN_ALERT "ALAL:simple module is being unloaded\n");
26     printk("ALAL:jiffies value: %lu\n", jiffies);
27 }
28 module_init(helloworldmodule_init);
29 module_exit(helloworldmodule_exit);
30 MODULE_LICENSE("GPL");
31 MODULE_AUTHOR("LK0");
32 MODULE_DESCRIPTION("Simple Kernel Module");
33 MODULE_VERSION("1.01");
```

2、编辑 Makefile 文件，构造指令集。

```
打开(O) 文件 保存(S) 窗口 帮助

Makefile  *helloworldmodule.c
~/Desktop  *helloworldmodule.c

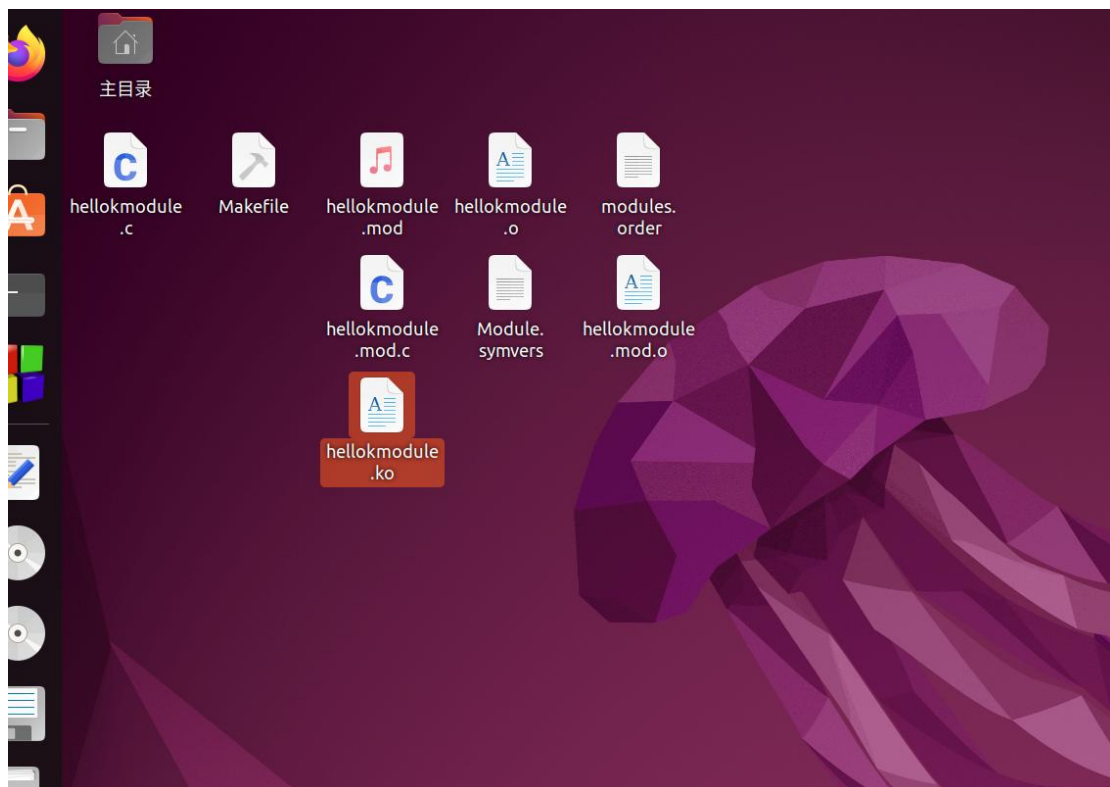
1 ModuleName=helloworldmodule
2 obj-m += $(ModuleName).o
3 all:$(ModuleName).ko
4 $(ModuleName).ko:$(ModuleName).c
5     make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
6 testload:$(ModuleName).ko
7     sudo dmesg -C
8     sudo lsmod $(ModuleName).ko
9     sudo dmesg | grep ywx2013852
10 testunload:$(ModuleName).ko
11     sudo dmesg -C
12     sudo rmmod $(ModuleName).ko
13     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：





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。

```
ywx2013852@ywx2013852-virtual-machine: ~/Desktop
availability of vmlinux
make[1]: 离开目录“/usr/src/linux-headers-5.15.0-48-generic”
ywx2013852@ywx2013852-virtual-machine:~/Desktop$ make testload
sudo dmesg -C
sudo insmod hellokmodule.ko
sudo dmesg | grep ywx2013852
```

	Pid	Father's	Pid	Stat
[2520.811329] ywx2013852 Name				
[2520.811336] ywx2013852 systemd	1	0		S
[2520.811338] ywx2013852 kthreadd	2	0		S
[2520.811339] ywx2013852 rcu_gp	3	2		I
[2520.811340] ywx2013852 rcu_par_gp	4	2		I
[2520.811341] ywx2013852 netns	5	2		I
[2520.811342] ywx2013852 kworker/0:0H	7	2		I
[2520.811343] ywx2013852 kworker/0:1H	9	2		I
[2520.811344] ywx2013852 mm_percpu_wq	10	2		I
[2520.811345] ywx2013852 rcu_tasks_rude_	11	2		S
[2520.811346] ywx2013852 rcu_tasks_trace	12	2		S
[2520.811347] ywx2013852 ksoftirqd/0	13	2		S
[2520.811348] ywx2013852 rcu_sched	14	2		I
[2520.811362] ywx2013852 migration/0	15	2		S
[2520.811363] ywx2013852 idle_inject/0	16	2		S
[2520.811364] ywx2013852 cpuhp/0	18	2		S
[2520.811365] ywx2013852 cpuhp/1	19	2		S
[2520.811366] ywx2013852 idle_inject/1	20	2		S

```
ywx2013852@ywx2013852-virtual-machine: ~/Desktop
```

[2520.811628] ywx2013852 update-notifier	3664	1885	S
[2520.811628] ywx2013852 gnome-terminal	4149	1738	S
[2520.811629] ywx2013852 gnome-terminal	4150	4149	S
[2520.811630] ywx2013852 gnome-terminal	4155	1738	S
[2520.811631] ywx2013852 bash	4213	4155	S
[2520.811632] ywx2013852 gedit	4457	1738	S
[2520.811633] ywx2013852 snapd	4823	1	S
[2520.811633] ywx2013852 kworker/u256:2	5647	2	I
[2520.811634] ywx2013852 kworker/1:0	17422	2	I
[2520.811635] ywx2013852 kworker/0:3	19148	2	I
[2520.811636] ywx2013852 kworker/1:2	27641	2	I
[2520.811637] ywx2013852 kworker/0:2	28132	2	I
[2520.811638] ywx2013852 kworker/u256:0	28520	2	I
[2520.811638] ywx2013852 kworker/0:0	28537	2	I
[2520.811639] ywx2013852 ibus-engine-sim	28540	2122	S
[2520.811640] ywx2013852 gjs	28569	1919	S
[2520.811641] ywx2013852 kworker/u256:1	28631	2	I
[2520.811642] ywx2013852 tracker-extract	28939	1738	R
[2520.811710] ywx2013852 make	29052	4213	S
[2520.811712] ywx2013852 sudo	29056	29052	S
[2520.811713] ywx2013852 sudo	29057	29056	S
[2520.811714] ywx2013852 insmod	29058	29057	R
[2520.811714] ywx2013852 总共进程的数目为:	297		

```
ywx2013852@ywx2013852-virtual-machine:~/Desktop$
```




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

```
ywx2013852@ywx2013852-virtual-machine: ~/Desktop
[ 2520.811714] ywx2013852 insmod                29058 29057 R
[ 2520.811714] ywx2013852 总共进程的数目为: 297
ywx2013852@ywx2013852-virtual-machine:~/Desktop$ lsmod
Module                  Size  Used by
hellokmodule            16384  0
cpuid                   16384  0
binfmt_misc            24576  1
isofs                   53248  2
rfcomm                  81920  4
bnep                    28672  2
vsock_loopback         16384  0
vmw_vsock_virtio_transport_common 40960  1 vsock_loopback
vmw_vsock_vmci_transport 32768  2
vsock                   49152  7 vmw_vsock_virtio_transport_common,vsock_loopback
,vmw_vsock_vmci_transport
intel_rapl_msr          20480  0
intel_rapl_common       40960  1 intel_rapl_msr
vmw_balloon             24576  0
nls_iso8859_1           16384  1
crct10dif_pclmul       16384  1
ghash_clmulni_intel    16384  0
aesni_intel            376832  0
crypto_simd             16384  1 aesni_intel
cryptd                  24576  2 crypto_simd,ghash_clmulni_intel
```

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。

```
floppy                118784  0
ywx2013852@ywx2013852-virtual-machine:~/Desktop$ make testunload
sudo dmesg -C
sudo rmmod hellokmodule.ko
sudo dmesg | grep ALAL
[ 2704.939363] ALAL:simple module is being unloaded
[ 2704.939371] ALAL:jiffies value: 4295568306
ywx2013852@ywx2013852-virtual-machine:~/Desktop$
```

7、此时我们在使用 lsmod 查看 modules



```
ywx2013852@ywx2013852-virtual-machine: ~/Desktop
ywx2013852@ywx2013852-virtual-machine:~/Desktop$ lsmod
Module                  Size  Used by
cpuid                   16384  0
binfmt_misc            24576  1
isofs                   53248  2
rfcomm                  81920  4
bnep                    28672  2
vsock_loopback         16384  0
vmw_vsock_virtio_transport_common 40960  1 vsock_loopback
vmw_vsock_vmci_transport 32768  2
vsock                   49152  7 vmw_vsock_virtio_transport_common,vsock_loopback
,vmw_vsock_vmci_transport
intel_rapl_msr          20480  0
intel_rapl_common       40960  1 intel_rapl_msr
vmw_balloon             24576  0
nls_iso8859_1           16384  1
crct10dif_pclmul        16384  1
ghash_clmulni_intel     16384  0
aesni_intel             376832 0
crypto_simd             16384  1 aesni_intel
cryptd                  24576  2 crypto_simd,ghash_clmulni_intel
input_leds              16384  0
joydev                  32768  0
serio_raw               20480  0
```

此时的 hello module 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->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");
```