操作系统课程设计实验报告

实验题目：Linux 内核模块编程

姓　　名：王翔宇

学　　号：22200215

组　　号：

专　　业：网络空间安全

班　　级：22270311

老师姓名：张桢

日　　期：2024　年　5　月　8　日

目　　录

(大家注意，目录是自动生成的，页码从正文部分开始，当同学们把正文写完后，只需要右击目录，选择更新域，目录会自动更新)

一 题目介绍

本实验通过编写Linux内核模块，加载内核模块，并测试结果，了解Linux内核模块的概念、编译方法、安装和测试方法。

主要掌握内容如下：

- Linux 内核模块的基本概念
- Linux 内核模块的编写方法
- Linux 内核模块的加载和移除
- Linux 内核模块的测试方法

二 实验思路

要编写 Linux 内核模块

1. 要了解 Linux 内核模块的基本结构

（1） 内核模块是一段代码，可以被动的加载在内核中，以扩展内核的功能或添加新的驱动程序

（2） 内核模块通常包含初始化和清理函数，分别在加载和卸载时调用

2. 设置开发环境

   确保 Linux 系统安装了适当的开发工具和内核头文件

3. 编写代码

   创建新的 c 源代码，编写内核模块

4. 编写 Makefile 文件

   创建一个 Makefile 来编译内核模块

5. 编译和加载模块

   使用 make 命令来编译模块

   使用 insmod 命令加载你的模块到内核中

   使用 dmesg 命令查看内核日志，确保模块成功加载

6. 测试内核模块

   使用 insmod 命令来对内核模块进行测试

7. 卸载模块

   使用 rmmod 命令来卸载模块

选题 1，输出某个进程的家族信息实验思路：

1. 将给定的 pid 通过参数传递进去，如果给定的 pid 不存在，则报错退出，如果存在，则打印当前进程信息，进程 pid，进程状态，以及进程进程名字
2. 判断当前进程有无父进程，若无，则打印无父母进程，反之，打印父进程的相关信息
3. 调用 list_for_each 函数打印相关兄弟进程的信息
4. 调用 list_for_each 函数打印相关子进程信息

以上打印信息均输出在系统日志中

选题 10，显示当前系统名称和版本的实验思路

1. 获取当前主机的系统名称和版本，如果获取为空，则输出错误
2. 如果获取成功，则分别打印 sysname 和 version

三 遇到问题及解决方法

问题一：在 make 时找不到目标文件

解决方法：这个问题基本上是一个失误，因为我将 Makefile 文件命名为 MakeFile，从而导致出错

问题二：在编写选题（1）的代码时出错

解决方法：错误理解了哪个指针指向的是自己的兄弟进程，重新对 task_struct 结构进行了学习

四 核心代码及实验结果展示

选题（1）的模块代码编写

```c
#include <linux/kernel.h>
#include <linux/sched.h>
#include <linux/module.h>
#include <linux/init.h>
#include <linux/moduleparam.h>

static pid_t pid = 1;
module_param(pid, int, 0644);

static int module1_init(void)
{
    struct task_struct *p;
    struct list_head *pp;
    struct task_struct *psibling;

    // 根据PID获取进程结构体
    p = pid_task(find_vpid(pid), PIDTYPE_PID);

    if (p == NULL) {
        printk(KERN_ALERT "Process with PID %d not found.\n", pid);
        return -EINVAL;
    }

    // 打印当前进程信息
    printk(KERN_INFO "Current Process: %d %ld %s\n", p->pid, p->state, p->comm);

    // 打印父进程信息
    if (p->parent == NULL) {
        printk(KERN_INFO "No Parent Process\n");
    } else {
        printk(KERN_INFO "Parent Process: %d %ld %s\n", p->parent->pid, p->parent->state, p->parent->comm);
    }

    // 打印兄弟进程信息
    list_for_each(pp, &p->parent->children)
    {
        psibling = list_entry(pp, struct task_struct, sibling);
        printk(KERN_INFO "Sibling Process: %d %ld %s\n", psibling->pid, psibling->state, psibling->comm);
    }

    // 打印子进程信息
    list_for_each(pp, &p->children)
    {
        psibling = list_entry(pp, struct task_struct, sibling);
        printk(KERN_INFO "Child Process: %d %ld %s\n", psibling->pid, psibling->state, psibling->comm);
    }

    return 0;
}

static void module1_exit(void)
{
    printk(KERN_ALERT "Module unloaded.\n");
}

module_init(module1_init);
module_exit(module1_exit);
MODULE_LICENSE("GPL");
```

编写 Makefile 文件

```
obj-m := module1.o
KDIR := /lib/modules/$(shell uname -r)/build
PWD := $(shell pwd)
default:
	make -C $(KDIR) M=$(PWD) modules
clean:
	make -C $(KDIR) M=$(PWD) clean
```

文件编写好之后输入 make 命令进行模块编译

编译好后文件类型如下

```
[root@ecs-os test1]# ls
Makefile  module1.c  module1.ko  module1.mod.c  module1.mod.o  module1.o  modules.order  Module.symvers
```

执行命令 insmod module1.ko pid=4 加载模块

执行 dmesg 查看运行结果

```
[90220.708733] Current Process: 4 1026 rcu_par_gp
[90220.709059] Parent Process: 2 1 kthreadd
[90220.709311] Sibling Process: 3 1026 rcu_gp
[90220.709573] Sibling Process: 4 1026 rcu_par_gp
[90220.709878] Sibling Process: 6 1026 kworker/0:0H
[90220.710175] Sibling Process: 8 1026 mm_percpu_wq
[90220.710474] Sibling Process: 9 1 ksoftirqd/0
[90220.710746] Sibling Process: 10 1026 rcu_sched
[90220.711032] Sibling Process: 11 1026 rcu_bh
[90220.711299] Sibling Process: 12 1 migration/0
[90220.711577] Sibling Process: 13 1 cpuhp/0
[90220.711856] Sibling Process: 14 1 cpuhp/1
[90220.712223] Sibling Process: 15 1 migration/1
[90220.712501] Sibling Process: 16 1 ksoftirqd/1
[90220.712872] Sibling Process: 18 1026 kworker/1:0H
[90220.713290] Sibling Process: 19 1 cpuhp/2
[90220.713585] Sibling Process: 20 1 migration/2
[90220.713936] Sibling Process: 21 1 ksoftirqd/2
[90220.714351] Sibling Process: 23 1026 kworker/2:0H
[90220.714655] Sibling Process: 24 1 cpuhp/3
[90220.714910] Sibling Process: 25 1 migration/3
[90220.715189] Sibling Process: 26 1 ksoftirqd/3
[90220.715464] Sibling Process: 28 1026 kworker/3:0H
[90220.715764] Sibling Process: 29 1 kdevtmpfs
[90220.716088] Sibling Process: 30 1026 netns
[90220.716356] Sibling Process: 31 1 kauditd
[90220.716613] Sibling Process: 33 1 khungtaskd
[90220.716885] Sibling Process: 34 1 oom_reaper
[90220.717158] Sibling Process: 35 1026 writeback
[90220.717447] Sibling Process: 36 1 kcompactd0
[90220.717722] Sibling Process: 37 1 ksmd
[90220.718059] Sibling Process: 38 1 khugepaged
[90220.718335] Sibling Process: 40 1026 crypto
[90220.718600] Sibling Process: 41 1026 kintegrityd
[90220.718898] Sibling Process: 42 1026 kblockd
[90220.719172] Sibling Process: 45 1026 md
[90220.719418] Sibling Process: 46 1026 edac-poller
[90220.719711] Sibling Process: 47 1 watchdogd
[90220.720024] Sibling Process: 49 1 kswapd0
[90220.720285] Sibling Process: 102 1026 kthrotld
[90220.720568] Sibling Process: 103 1 irq/42-aerdrv
[90220.720867] Sibling Process: 104 1 irq/42-pciehp
[90220.721163] Sibling Process: 105 1 irq/43-aerdrv
[90220.721458] Sibling Process: 106 1 irq/43-pciehp
[90220.721754] Sibling Process: 107 1 irq/44-aerdrv
[90220.722055] Sibling Process: 108 1 irq/44-pciehp
[90220.722349] Sibling Process: 109 1 irq/45-aerdrv
[90220.722642] Sibling Process: 110 1 irq/45-pciehp
[90220.722939] Sibling Process: 111 1 irq/46-aerdrv
[90220.723235] Sibling Process: 112 1 irq/46-pciehp
[90220.723533] Sibling Process: 113 1 irq/47-aerdrv
[90220.723827] Sibling Process: 114 1 irq/47-pciehp
[90220.724200] Sibling Process: 115 1 irq/48-aerdrv
[90220.724500] Sibling Process: 116 1 irq/48-pciehp
[90220.724795] Sibling Process: 117 1 irq/49-aerdrv
[90220.725095] Sibling Process: 118 1 irq/49-pciehp
[90220.725392] Sibling Process: 119 1 irq/50-aerdrv
```

内核模块测试

```
insmod: ERROR: could not insert module module1.ko: File exists
[root@ecs-os test1]# lsmod
Module                  Size  Used by
module1                262144  0
rfkill                 262144  1
aes_ce_blk             262144  0
crypto_simd            262144  1 aes_ce_blk
cryptd                 262144  1 crypto_simd
aes_ce_cipher          262144  1 aes_ce_blk
ghash_ce               262144  0
sha2_ce                262144  0
sha256_arm64           262144  1 sha2_ce
sha1_ce                262144  0
virtio_balloon         262144  0
vfat                   262144  1
fat                    262144  1 vfat
sch_fq_codel           262144  3
ip_tables              262144  0
ext4                   851968  1
mbcache                262144  1 ext4
jbd2                   327680  1 ext4
virtio_gpu             262144  1
virtio_net             262144  0
net_failover           262144  1 virtio_net
virtio_blk             262144  3
failover               262144  1 net_failover
virtio_pci             262144  0
virtio_mmio            262144  0
virtio_ring            262144  6 virtio_mmio,virtio_balloon,virtio_gpu,virtio_pci,virtio_blk,virtio_net
virtio                 262144  6 virtio_mmio,virtio_balloon,virtio_gpu,virtio_pci,virtio_blk,virtio_net
```

执行完毕之后运行命令 rmmod 卸载模块

```
[90220.737141] Sibling Process: 158 1 irq/69-pciehp
[90220.737434] Sibling Process: 159 1 irq/70-aerdrv
[90220.737729] Sibling Process: 160 1 irq/70-pciehp
[90220.738036] Sibling Process: 161 1 irq/41-ACPI:Ged
[90220.738341] Sibling Process: 162 1026 acpi_thermal_pm
[90220.738666] Sibling Process: 163 1026 kmpath_rdacd
[90220.738972] Sibling Process: 164 1026 kaluad
[90220.739244] Sibling Process: 165 1026 ipv6_addrconf
[90220.739556] Sibling Process: 166 1026 kstrp
[90220.739822] Sibling Process: 359 0 kworker/1:1H
[90220.740189] Sibling Process: 364 1026 ttm_swap
[90220.740479] Sibling Process: 378 1026 kworker/3:1H
[90220.740782] Sibling Process: 379 2 jbd2/vda2-8
[90220.741066] Sibling Process: 380 1026 ext4-rsv-conver
[90220.741393] Sibling Process: 448 1026 kworker/2:1H
[90220.741700] Sibling Process: 450 1026 kworker/0:1H
[90220.742012] Sibling Process: 5124 1026 kworker/0:3
[90220.742320] Sibling Process: 5622 1026 kworker/u8:2
[90220.742630] Sibling Process: 5824 1026 kworker/2:1
[90220.742935] Sibling Process: 5874 1026 kworker/1:1
[90220.743241] Sibling Process: 6281 1026 kworker/3:2
[90220.743546] Sibling Process: 6457 1026 kworker/2:0
[90220.743869] Sibling Process: 6475 1026 kworker/0:0
[90220.744177] Sibling Process: 6515 0 kworker/1:0
[90220.744465] Sibling Process: 6526 1026 kworker/3:3
[90220.744766] Sibling Process: 6528 1026 kworker/u8:0
[90220.745076] Sibling Process: 6970 1026 kworker/u8:1
[90386.799087] Module unloaded.
```

1. 显示系统名称和版本模块编写

   实现代码编写

```c
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/utsname.h>
#include <linux/module.h>

static int hello_init(void)
{
    struct new_utsname *uname = utsname();
    if (uname == NULL) {
        printk(KERN_ALERT "Failed to retrieve system information.\n");
        return -EINVAL;
    }

    printk(KERN_INFO "System Name: %s\n", uname->sysname);
    printk(KERN_INFO "Version: %s\n", uname->version);

    return 0;
}

static void hello_exit(void)
{
    printk(KERN_INFO "Module unloaded.\n");
}

module_init(hello_init);
module_exit(hello_exit);
MODULE_LICENSE("GPL");
```

Makefile 文件编写

```
obj-m := module2.o
KDIR := /lib/modules/$(shell uname -r)/build
PWD := $(shell pwd)
default:
	make -C $(KDIR) M=$(PWD) modules
clean:
	make -C $(KDIR) M=$(PWD) clean
```

make 命令编译模块

编译好后相关文件如下

```
[root@ecs-os test2]# ls
Makefile  module2.c  module2.ko  module2.mod.c  module2.mod.o  module2.o  modules.order  Module.symvers
```

用 insmod module2.ko 加载模块

用 dmesg 显示运行结果

```
[90659.225674] System Name: Linux
[90659.225911] Version: #2 SMP Thu Apr 18 11:48:31 CST 2024
```

测试模块

```
[root@ecs-os test2]# lsmod
Module                  Size  Used by
module2                262144  0
rfkill                 262144  1
aes_ce_blk             262144  0
crypto_simd            262144  1 aes_ce_blk
cryptd                 262144  1 crypto_simd
aes_ce_cipher          262144  1 aes_ce_blk
ghash_ce               262144  0
sha2_ce                262144  0
sha256_arm64           262144  1 sha2_ce
sha1_ce                262144  0
virtio_balloon         262144  0
vfat                   262144  1
fat                    262144  1 vfat
sch_fq_codel           262144  3
ip_tables              262144  0
ext4                   851968  1
mbcache                262144  1 ext4
jbd2                   327680  1 ext4
virtio_gpu             262144  1
virtio_net             262144  0
net_failover           262144  1 virtio_net
virtio_blk             262144  3
failover               262144  1 net_failover
virtio_pci             262144  0
virtio_mmio            262144  0
virtio_ring            262144  6 virtio_mmio,virtio_balloon,virtio_gpu,virtio_pci,virtio_blk,virtio_net
virtio                 262144  6 virtio_mmio,virtio_balloon,virtio_gpu,virtio_pci,virtio_blk,virtio_net
[root@ecs-os test2]#
```

用 rmmod module2 卸载模块

```
[90659.225674] System Name: Linux
[90659.225911] Version: #2 SMP Thu Apr 18 11:48:31 CST 2024
[90712.864687] Module unloaded.
```

五 个人实验改进与总结

5.1 个人实验改进

本次实验改进重点在于功能扩展、错误处理、参数验证、性能优化和代码注释。通过添加更多系统状态信息、加强错误处理机制、严格验证参数合法性、优化代码性能，以及增加代码注释和文档，我旨在提高模块的功能性、稳定性和可维护性。这些改进将有助于提高我的编程技能，加深对 Linux 内核模块编程的理解，并培养良好的工程实践习惯。

5.2 个人实验总结

完成上述实验后，我对 Linux 内核模块编程有了更深入的理解。通过编写简单的模块，我学会了如何初始化模块、处理模块参数、以及在模块中调用内核函数。我还了解了如何在模块中获取进程信息，并通过打印信息来实现简单的功能。在实验过程中，我学会了模块编译、加载和卸载的基本操作，以及如何使用内核日志进行调试。此外，我还学习了错误处理的重要性，以及如何添加注释和文档来提高代码的可读性和可维护性。总的来说，这些实验让我对 Linux 内核模块编程有了更深入的认识，并为我今后进一步探索内核开发打下了良好的基础。

六 参考文献

Linux 内核模块编程指南(一) linux 内核编程指南-CSDN 博客

如何编写一个 Linux 内核模块，这次手把手教你 - 知乎 (zhihu.com)