Project 2-2: Linux Kernel Module for Task Information

白骐硕 518030910102

任务概述

In this project, you will write a Linux kernel module that uses the `/proc` file system for displaying a task's information based on its process identifier value pid. This project will involve writing a process identifier to the file `/proc/pid`. Once a pid has been written to the `/proc` file, subsequent reads from `/proc/pid` will report (1) the command the task is running, (2) the value of the task's pid , and (3) the current state of the task.

具体实现

对于内核模块的加载、卸载以及 /proc 文件的创建、删除等操作与 Project1 中类似，在此不再赘述。下面，主要介绍对于 `/proc/pid` 文件读取和写入函数的编写。

由于该内核模块涉及到对 /proc 文件的写入操作，file_operations 结构体的定义增加 .write内容，具体如下：

```
static ssize_t proc_read(struct file *file, char *buf, size_t count, loff_t *pos);
static ssize_t proc_write(struct file *file, const char __user *usr_buf, size_t count, loff_t *pos);

static struct file_operations proc_ops = {
    .owner = THIS_MODULE,
    .read = proc_read,
    .write = proc_write,
};
```

下面详细介绍两个函数，`proc_read(...)` 以及 `proc_write(...)`。

`proc_write(...)` 的具体操作为：

- 分配 kernel memory
- copy_from_user
- 将写入的字符串转换为int，记录到全局变量current_pid中
- 释放kernel memory

代码如下：

```
static ssize_t proc_write(struct file *file, const char __user *usr_buf, size_t count, loff_t *pos) {
    char *k_mem;

    // allocate kernel memory
    k_mem = kmalloc(count, GFP_KERNEL);
    if(raw_copy_from_user(k_mem, usr_buf, count)){
        printk(KERN_INFO "Error copying from user!\n");
        return -1;
    }
    k_mem[count] = '\0';
    kstrtoint(k_mem, 10, &current_pid);
    printk(KERN_INFO "Set current PID to: %d\n", current_pid);
    kfree(k_mem);
    return count;
}
```

`proc_read(...)` 的具体操作为：

- 根据pid获取pid_task，如果pid_task不为空，将command,pid,state的信息格式化写入buffer；如果为空，则将("Invalid PID %d" , current_pid) 的错误提示信息写入buffer
- 将buffer通过函数 copy_to_user()传入到用户空间

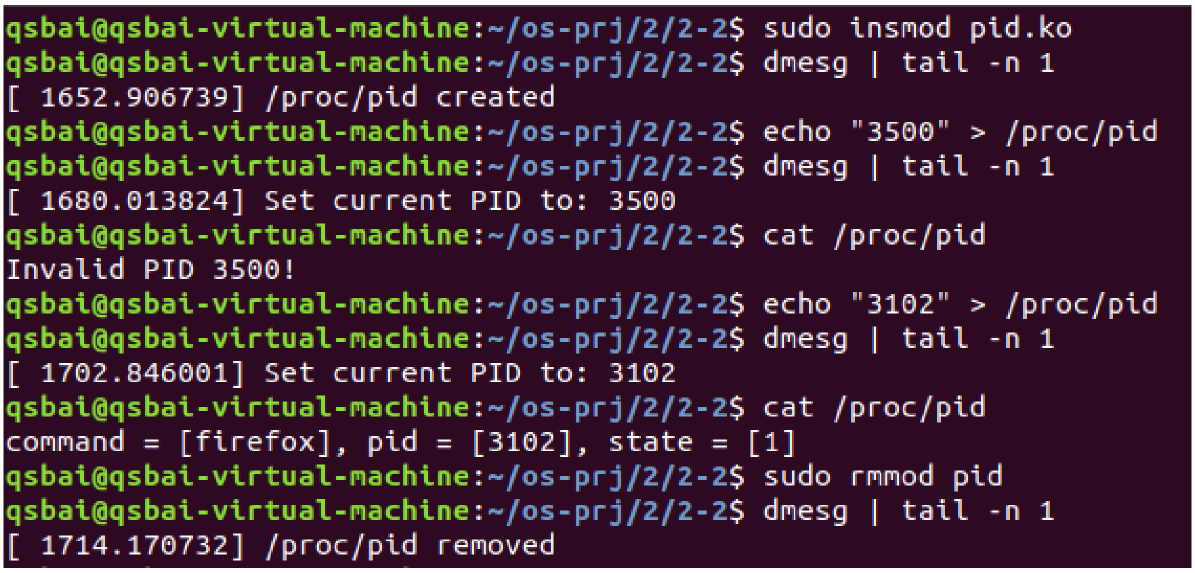
代码如下：

```
static ssize_t proc_read(struct file *file, char __user *usr_buf, size_t count,
loff_t *pos) {
    int rv = 0;
    char buffer[BUFFER_SIZE];
    static int completed = 0;

    struct task_struct *tsk = NULL;
    if (completed) {
        completed = 0;
        return 0;
    }
    tsk = pid_task(find_vpid(current_pid), PIDTYPE_PID);

    if(tsk) {
        rv = snprintf(buffer, BUFFER_SIZE,
                      "command = [%s], pid = [%d], state = [%ld]\n",
                      tsk->comm, current_pid, tsk->state);
    } else {
        printk(KERN_INFO "Invalid PID %d!\n", current_pid);
        rv = snprintf(buffer, BUFFER_SIZE, "Invalid PID %d!\n", current_pid);
    }

    completed = 1;
    if(raw_copy_to_user(usr_buf, buffer, rv)){
        rv = -1;
    }

    return rv;
}
```

实验结果

实验结果的截图如下，其中测试了两个不同的 pid值：

- pid = 3500，为 Invalid pid
- pid = 3102，为 firefox浏览器的进程号



```
qsbai@qsbai-virtual-machine:~/os-prj/2/2-2$ sudo insmod pid.ko
qsbai@qsbai-virtual-machine:~/os-prj/2/2-2$ dmesg | tail -n 1
[ 1652.906739] /proc/pid created
qsbai@qsbai-virtual-machine:~/os-prj/2/2-2$ echo "3500" > /proc/pid
qsbai@qsbai-virtual-machine:~/os-prj/2/2-2$ dmesg | tail -n 1
[ 1680.013824] Set current PID to: 3500
qsbai@qsbai-virtual-machine:~/os-prj/2/2-2$ cat /proc/pid
Invalid PID 3500!
qsbai@qsbai-virtual-machine:~/os-prj/2/2-2$ echo "3102" > /proc/pid
qsbai@qsbai-virtual-machine:~/os-prj/2/2-2$ dmesg | tail -n 1
[ 1702.846001] Set current PID to: 3102
qsbai@qsbai-virtual-machine:~/os-prj/2/2-2$ cat /proc/pid
command = [firefox], pid = [3102], state = [1]
qsbai@qsbai-virtual-machine:~/os-prj/2/2-2$ sudo rmmod pid
qsbai@qsbai-virtual-machine:~/os-prj/2/2-2$ dmesg | tail -n 1
[ 1714.170732] /proc/pid removed
```