

# CS353 Project 1

姓名：彭嘉淇

学号：519030910229

## 实验一

### 1. 编写模块实现以下要求

接受三个参数：operand1 类型为 int，operand2 类型为 int 数组，operator 类型为 charp（字符串）。

创建 proc 文件 /proc/<你的学号>/calc。

如果 operator 为 add，那么 operand2 的每一个元素都加上 operand1，得到结果数组；如果 operator 为 mul，那么 operand2 的每一个元素都乘上 operand1，得到结果数组。

当读取 proc 文件时，输出结果数组，每个元素用逗号分隔。

当用户向 proc 文件写入一个数字时，这个数字作为新的 operand1 重新进行计算。

### 1、实验过程：

实验环境：ubuntu20.04、linux 5.13.0-37

下载代码模板

```
git clone https://github.com/chengjiagan/CS353-2022-Spring.git
```

查阅ppt，找到实现上述功能的对应函数

```
module_param(test, int, 0644); // 传递参数
proc_create //创建文件
proc_mkdir //创建目录
procfile_read //读取函数
procfile_write //写入函数
copy_from_user //数据从内核区拷贝到用户区
```

函数说明：

函数名	功能
proc_init	创建文件夹并在其文件夹下创建文件
proc_exit	删除文件夹以及文件
proc_read	定义读取函数，对定义的操作及操作数进行计算，将结果存储到缓存区
proc_write	读取用户缓存区，并赋值给操作数1

### 2、实验结果

```

pjqq@ubuntu:~/Desktop/linux_kernel/CS353-2022-Spring/project1$ sudo insmod calc.ko operand1=2 operand2=1,2,3,4,5 operator=mul
pjqq@ubuntu:~/Desktop/linux_kernel/CS353-2022-Spring/project1$ cat /proc/519030910229/calc
2,4,6,8,10
pjqq@ubuntu:~/Desktop/linux_kernel/CS353-2022-Spring/project1$ sudo rmmod calc
pjqq@ubuntu:~/Desktop/linux_kernel/CS353-2022-Spring/project1$ sudo insmod calc.ko operand1=2 operand2=1,2,3,4,5 operator=add
pjqq@ubuntu:~/Desktop/linux_kernel/CS353-2022-Spring/project1$ cat /proc/519030910229/calc
3,4,5,6,7
pjqq@ubuntu:~/Desktop/linux_kernel/CS353-2022-Spring/project1$ echo 3 > /proc/519030910229/calc
pjqq@ubuntu:~/Desktop/linux_kernel/CS353-2022-Spring/project1$ cat /proc/519030910229/calc
4,5,6,7,8
pjqq@ubuntu:~/Desktop/linux_kernel/CS353-2022-Spring/project1$ sudo rmmod calc
pjqq@ubuntu:~/Desktop/linux_kernel/CS353-2022-Spring/project1$ sudo insmod calc.ko operand1=2 operand2=1,2,3,4,5 operator=mul
pjqq@ubuntu:~/Desktop/linux_kernel/CS353-2022-Spring/project1$ cat /proc/519030910229/calc
12,4,6,8,10
pjqq@ubuntu:~/Desktop/linux_kernel/CS353-2022-Spring/project1$ echo 3 > /proc/519030910229/calc
pjqq@ubuntu:~/Desktop/linux_kernel/CS353-2022-Spring/project1$ cat /proc/519030910229/calc
3,6,9,12,15
pjqq@ubuntu:~/Desktop/linux_kernel/CS353-2022-Spring/project1$

```

### 3、实验心得

- 在上手前多阅读相关文档，参考[The Linux Kernel Module Programming Guide](https://github.com/sysprog21/linux_kernel_module_programming_guide) ([sysprog21.github.io](https://github.com/sysprog21/linux_kernel_module_programming_guide))编写。最初在做实验时，在linux5.4的环境，编译时显示"proc\_fs.h"找不到，只好烦请助教同学答疑解惑，经提醒后改到linux5.13版本，终于编译成功。

#### 7.1 The proc\_ops Structure

The `proc_ops` structure is defined in [include/linux/proc\\_fs.h](#) in Linux v5.6+. In older kernels, it used `file_operations` for custom hooks in /proc file system, but it contains some members that are unnecessary in VFS, and every time VFS expands `file_operations` set, /proc code comes bloated. On the other hand, not only the space, but also some operations were saved by this

- 做实验时经常混淆用户态和内存态的缓存区，不知道在读时还是写时做相应操作，后来把老师给的相关示例看懂，并上网查找资料，终于明白了。

## 实验二

### 2. 编写一个程序实现以下要求：

从 /proc 文件系统中得到系统中的所有进程 PID 以及相关信息。  
 输出这些进程的 PID，进程状态，进程的命令行参数三列信息。  
 PID 5字符宽度，右对齐，空格填补空缺；每列信息之间用一个空格分隔。  
 输出效果可以参考命令 `ps -e -ww -o pid:5,state,cmd` 的输出效果。

### 1、实验过程

实验环境：ubuntu20.04、linux 5.13.0-37

下载代码模板

```
git clone https://github.com/chengjiagan/CS353-2022-Spring.git
```

在ps.c文件中编写代码。

查阅ppt，在proc文件系统中找到包含对应信息的文件，参考[proc\(5\) - Linux manual page \(man7.org\)](#)了解 proc 中各个文件的作用。

在/proc文件夹下查找数字文件名的进程文件，需要的 proc 文件有 /proc/cmdline 和 /proc/PID/stat。对于部分进程，其cmdline 文件为空，此时可输出 /proc/comm 文件中的内容。

## 2、实验结果

本实验程序结果

```
pjq@ubuntu:~/Desktop/linux_kernel/CS353-2022-Spring/project1$ gcc ps.c -o ps
pjq@ubuntu:~/Desktop/linux_kernel/CS353-2022-Spring/project1$ ./ps
PID S CMD
1 S /sbin/init
2 S [kthreadd]
3 I [rcu_gp]
4 I [rcu_par_gp]
6 I [kworker/0:0H-events_highpri]
9 I [mm_percpu_wq]
10 S [rcu_tasks_rude_]
11 S [rcu_tasks_trace]
12 S [ksoftirqd/0]
13 I [rcu_sched]
14 S [migration/0]
15 S [idle_inject/0]
16 S [cpuhp/0]
17 S [cpuhp/1]
18 S [idle_inject/1]
19 S [migration/1]
20 S [ksoftirqd/1]
22 I [kworker/1:0H-events_highpri]
23 S [cpuhp/2]
24 S [idle_inject/2]
25 S [migration/2]
26 S [ksoftirqd/2]
28 I [kworker/2:0H-kblockd]
29 S [cpuhp/3]
30 S [idle_inject/3]
31 S [migration/3]
```

运行一下命令后的结果，可见与上图几乎一致

```
ps -e -ww -o pid:5,state,cmd
```

```
8038 R ./ps
pjq@ubuntu:~/Desktop/linux_kernel/CS353-2022-Spring/project1$ ps -e -ww -o pid:5,state,cmd
PID S CMD
1 S /sbin/init auto noprompt
2 S [kthreadd]
3 I [rcu_gp]
4 I [rcu_par_gp]
6 I [kworker/0:0H-events_highpri]
9 I [mm_percpu_wq]
10 S [rcu_tasks_rude_]
11 S [rcu_tasks_trace]
12 S [ksoftirqd/0]
13 I [rcu_sched]
14 S [migration/0]
15 S [idle_inject/0]
16 S [cpuhp/0]
17 S [cpuhp/1]
18 S [idle_inject/1]
19 S [migration/1]
20 S [ksoftirqd/1]
22 I [kworker/1:0H-events_highpri]
23 S [cpuhp/2]
24 S [idle_inject/2]
25 S [migration/2]
26 S [ksoftirqd/2]
28 I [kworker/2:0H-kblockd]
29 S [cpuhp/3]
30 S [idle_inject/3]
31 S [migration/3]
```

## 3、实验心得

- 用C语言编写程序经常会遇到指针使用的问题，例如在编写程序时想对字符串进行拼接，最初采用 strcat 的方法，经常报错，但是采用 sprintf 的方式可以很方便地进行赋值，甚至还可以很好地将数字等其他类型的变量转成字符串。

```
sprintf(stat_path, "/proc/%s/stat", entry->d_name);
```

- 通过从底层实现ps命令，使我对/proc文件系统的理解更深了一层，以后查看系统状态时可以方便地从proc文件目录中获取相关信息。
- 成功运行一遍程序后，时过一日再运行自己的程序时发现相关文件找不到，最后发现原来是实验一的模块没有卸载，proc的进程文件夹下没有对应的stat等文件。这是实验中值得注意的地方，在运行实验二时记得卸载实验一的模块。