**Университет ИТМО**

Факультет программной инженерии и компьютерной техники

**Отчет**

По лабораторной работе №2

По дисциплине «Операционные системы»

Вариант procfs: task\_cputime, inode

Выполнил:

Ерехинский Андрей Владимирович  
Студент группы Р33312

Преподаватель:   
Осипов С.В.

Санкт-Петербург  
2022 г.

**Описание задания:**

Разработать комплекс программ на пользовательском уровне и уровне ярда, который собирает информацию на стороне ядра и передает информацию на уровень пользователя, и выводит ее в удобном для чтения человеком виде. Программа на уровне пользователя получает на вход аргумент(ы) командной строки (не адрес!), позволяющие идентифицировать из системных таблиц необходимый путь до целевой структуры, осуществляет передачу на уровень ядра, получает информацию из данной структуры и распечатывает структуру в стандартный вывод. Загружаемый модуль ядра принимает запрос через указанный в задании интерфейс, определяет путь до целевой структуры по переданному запросу и возвращает результат на уровень пользователя.

**Выполнение:**

**Task\_cputime**

static ssize\_t read\_task\_cputime(struct file \*filp, char \_\_user \*buffer, size\_t length, loff\_t \* offset) {

// cat /proc/mymodule/cputime  
printk(KERN\_INFO "read\_task\_cputime called.\n");  
  
if (len) {  
 len=0;  
}  
else {  
 len=1;  
 return 0;  
}

pid\_struct = find\_get\_pid(p\_id);  
cputime = pid\_task(pid\_struct, PIDTYPE\_PID);  
  
clean\_buffer();

clean\_line();  
sprintf(str, "PID: %d\n", p\_id);  
go\_to\_new\_line();

clean\_line();  
sprintf(str, "task\_cputime time spent in user mode: %lld\n", cputime->utime / 1000000);  
go\_to\_new\_line();

clean\_line();  
sprintf(str, "task\_cputime time spent in kernel mode: %lld\n", cputime->stime / 1000000);  
go\_to\_new\_line();

clean\_line();  
sprintf(str, "task\_cputime total time spent on the CPU: %lld\n", cputime->last\_sum\_exec\_runtime / 1000000);  
go\_to\_new\_line();

if(copy\_to\_user(buffer, arr, k)){  
 pr\_err("Data Send : Err!\n");  
}  
return length;

}

static ssize\_t write\_task\_cputime(struct file \*filp, const char \*buff, size\_t len, loff\_t \*off) {

//printf "PID" | sudo tee /proc/mymodule/cputime  
char \*id;  
id = (char \*)kmalloc(1000\*sizeof(char),GFP\_KERNEL);  
if(copy\_from\_user(id,buff,len))  
 return -EFAULT;  
p\_id = simple\_strtoul(id,NULL,0);  
printk(KERN\_INFO "write\_task\_cputime called.\nNew PID: %d\n", p\_id);  
  
return len;

}

**Inode**

static ssize\_t read\_inode(struct file \*filp, char \_\_user \*buffer, size\_t length, loff\_t \* offset) {

// cat /proc/mymodule/inode  
  
printk(KERN\_INFO "read\_inode called.\n");  
  
if (len) {  
 len=0;  
}  
else {  
 len=1;  
 return 0;  
}  
  
kern\_path(path\_name, LOOKUP\_FOLLOW, &path);  
inode = path.dentry->d\_inode;  
  
clean\_buffer();  
clean\_line();  
sprintf(str, "Path name: %s\n", path\_name);  
go\_to\_new\_line();  
  
clean\_line();  
sprintf(str, "Номер индекса: %ld\n", inode->i\_ino);  
go\_to\_new\_line();  
  
clean\_line();  
sprintf(str, "Права доступа: %d\n", inode->i\_mode);  
go\_to\_new\_line();  
  
clean\_line();  
sprintf(str, "Размер: %lld\n", inode->i\_size);  
go\_to\_new\_line();  
  
clean\_line();  
sprintf(str, "Количество жестких ссылок: %d\n", inode->i\_nlink);  
go\_to\_new\_line();  
  
clean\_line();  
sprintf(str, "Номер версии индекса: %d\n", inode->i\_generation);  
go\_to\_new\_line();  
  
if(copy\_to\_user(buffer, arr, k)){  
 pr\_err("Data Send : Err!\n");  
}  
return length;

}

static ssize\_t write\_inode(struct file \*filp, const char \*buff, size\_t len, loff\_t \*off) {

//printf "PATH" | sudo tee /proc/mymodule/inode  
char \*new\_path\_name;  
new\_path\_name = (char \*)kmalloc(1000\*sizeof(char), GFP\_KERNEL);  
  
if(copy\_from\_user(new\_path\_name, buff, len))  
 return -EFAULT;  
  
path\_name = new\_path\_name;  
printk(KERN\_INFO "write\_inode called.\n");  
return len;

}

Сборка Makefile через команду **sudo make -C /usr/src/linux-headers-`uname -r` SUBDIRS=$PWD modules**:

*obj-y := myos.o*

Далее добавление модуля: **insmod myos.ko**

Модуль установлен, в /proc появились:

/proc/mymodule/cputime  
 /proc/mymodule/inode

С которыми можно взаимодействовать(чтение/запись) посредством двух команд:

**cat /proc/mymodule/cputime  
printf "PID" | sudo tee /proc/mymodule/cputime**

**cat /proc/mymodule/inode  
printf "PATH" | sudo tee /proc/mymodule/inode**

**Выводы:** во время выполнения лабораторной работы я углубился в работу ядра linux. Написал собственный модуль ядра, добавил в ядро, а также проверил его на работоспособность.