

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ
ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ
ИНЖЕНЕРИИ

КУРСОВОЙ ПРОЕКТ
ЗАЩИЩЕН С ОЦЕНКОЙ
РУКОВОДИТЕЛЬ

ст.преп.

должность, уч. степень, звание

подпись, дата

Поляк М.Д.

инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОМУ ПРОЕКТУ

ДИСПЕТЧЕР ЗАДАЧ

по дисциплине: ОПЕРАЦИОННЫЕ СИСТЕМЫ И СЕТИ

РАБОТУ ВЫПОЛНИЛА
СТУДЕНТКА 4336
ГР.

подпись, дата

Н.Д. Алексан-
дрова

инициалы, фамилия

Санкт-Петербург
2016

1 Цель работы

Знакомство с устройством ядра ОС Linux. Получение опыта разработки драйвера устройства.

2 Цель работы

Реализовать на уровне модуля отображения устройств (Device Mapper Layer) систему для мониторинга и сбора статистики об использовании ресурсов компьютера (ЦП, память) и запущенных процессах .

3 Техническая документация

1. Сборка проекта:

Скачиваем файлы с репозитория на github при помощи команды:

```
https://github.com/natatalex/device-manager-linux.git
```

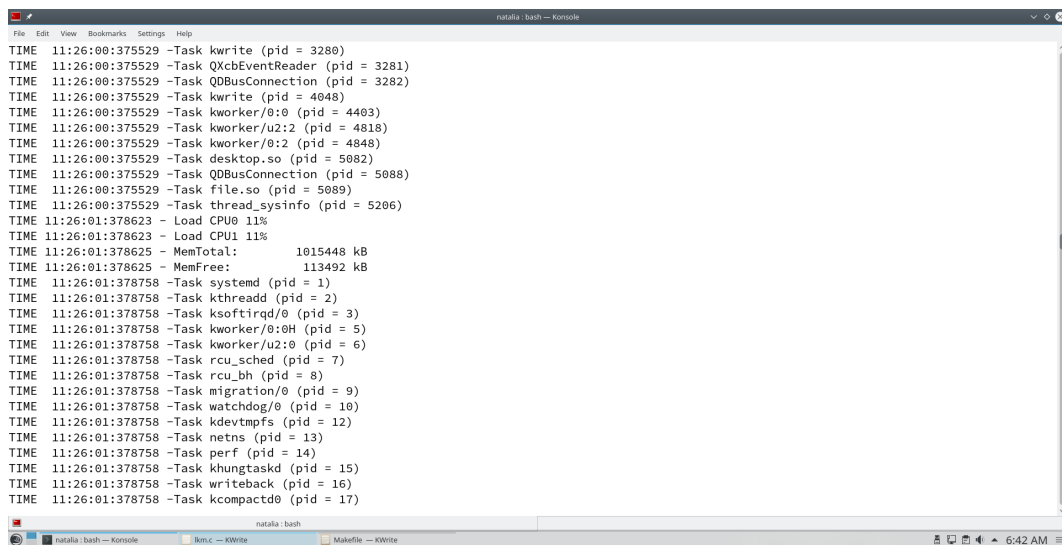
2. Сборка:

Шаг 1: Собираем модуль (lkm.ko) с помощью запуска команды "make". С помощью "dmesg | tail "смотрим что модуль загружен.

Шаг 2: Получаем информацию об процессах и ресурсах утилитой "cat /dev/sysinfo" записываем её в файл.

Шаг 3: Удаляем модуль с помощью "rmmod lkm"и смотрим с помощью "dmesg | tail"что модуль выгружен.

4 Скриншоты

A screenshot of a terminal window titled 'natalia: bash — Konsole'. The terminal displays a list of system processes and their resource usage. The output is as follows:

```
TIME 11:26:00:375529 -Task kwrite (pid = 3280)
TIME 11:26:00:375529 -Task QXcbEventReader (pid = 3281)
TIME 11:26:00:375529 -Task QDBusConnection (pid = 3282)
TIME 11:26:00:375529 -Task kwrite (pid = 4048)
TIME 11:26:00:375529 -Task kworker/0:0 (pid = 4403)
TIME 11:26:00:375529 -Task kworker/u2:2 (pid = 4818)
TIME 11:26:00:375529 -Task kworker/0:2 (pid = 4848)
TIME 11:26:00:375529 -Task desktop.so (pid = 5082)
TIME 11:26:00:375529 -Task QDBusConnection (pid = 5088)
TIME 11:26:00:375529 -Task file.so (pid = 5089)
TIME 11:26:00:375529 -Task thread_sysinfo (pid = 5206)
TIME 11:26:01:378623 - Load CPU0 11%
TIME 11:26:01:378623 - Load CPU1 11%
TIME 11:26:01:378625 - MemTotal:      1015448 kB
TIME 11:26:01:378625 - MemFree:      113492 kB
TIME 11:26:01:378758 -Task systemd (pid = 1)
TIME 11:26:01:378758 -Task kthreadd (pid = 2)
TIME 11:26:01:378758 -Task ksoftirqd/0 (pid = 3)
TIME 11:26:01:378758 -Task kworker/0:0H (pid = 5)
TIME 11:26:01:378758 -Task kworker/u2:0 (pid = 6)
TIME 11:26:01:378758 -Task rcu_sched (pid = 7)
TIME 11:26:01:378758 -Task rcu_bh (pid = 8)
TIME 11:26:01:378758 -Task migration/0 (pid = 9)
TIME 11:26:01:378758 -Task watchdog/0 (pid = 10)
TIME 11:26:01:378758 -Task kdevtmpfs (pid = 12)
TIME 11:26:01:378758 -Task netns (pid = 13)
TIME 11:26:01:378758 -Task perf (pid = 14)
TIME 11:26:01:378758 -Task khungtaskd (pid = 15)
TIME 11:26:01:378758 -Task writeback (pid = 16)
TIME 11:26:01:378758 -Task kcompactd0 (pid = 17)
```

The terminal window has a menu bar with 'File', 'Edit', 'View', 'Bookmarks', 'Settings', and 'Help'. The bottom status bar shows the time as 6:42 AM.

Рис. 1: Перечень процессов и затрачиваемых ресурсов

5 Заключение

В процессе выполнения данной курсовой работы мною были получены знания и навыки, необходимые для работы с ядром ОС Linux, а так же знания и навыки в разработке драйверов устройств.

6 Приложение

lkm.c

```
#include <linux/kernel.h>    //Для printk()
#include <linux/module.h>    //Модуль __init, __exit ....
#include <linux/init.h>      //Определения макросов
#include <linux/fs.h>
#include <asm/uaccess.h>      //Для put_user()
#include <linux/kthread.h>    //thread
#include <linux/delay.h>      //msleep()
#include <linux/slab.h>       //kmalloc()
#include <linux/sched.h>      //task_struct, put_task_struct
#include <linux/time.h>       //current time

//Информация о модуле, которую можно будет увидеть с помощью Modinfo
MODULE_LICENSE("LGPL");
MODULE_AUTHOR("Developer by natatalex");
MODULE_DESCRIPTION("System info module");
MODULE_SUPPORTED_DEVICE("sysinfo");           // /dev/sysinfo

#define DEVICE_NAME "sysinfo"    //Имя устройства
#define SIZE_BUFFER_DEVICE 65536 //размер буфера

//Прототипы функций поддерживаемые нашим устройством (открытие, закрытие,
static int device_open(struct inode *, struct file *);
static int device_release(struct inode *, struct file *);
static ssize_t device_read(struct file *, char *, size_t, loff_t *);
static ssize_t device_write(struct file *, const char *, size_t, loff_t *)

//Прототипы функций работы с файлом (открытие, чтение)
struct file *file_open(const char *path, int flags, int rights);
int file_read(struct file *file, unsigned long long offset, unsigned char

//Прототипы функций, вспомогательных операций (построчное чтение из файла,
int fgets(unsigned char *name, struct file *fp, int iRead_offset); //Чтение
void putBufferDevice(char *message, int size);           //Запись в буфер новы

//Глобальные переменные
static int superior_number;           //Старший номер устройства, нашего уст
static int isDeviceOpen=0;            //Используется устройства 0 - нет
static char *textUser;               //Данные отображаемые пользователю
static char *p_textUser;             //указатель на начало буфера
static int pointText=0;              //Последний символ в буфере
static int bRang=1;                 //переменная количества расчетов загрузки
```

```

struct task_struct *taskThread;

//структура операций над устройством
static struct file_operations fops={
    .read=device_read,
    .write=device_write,
    .open=device_open,
    .release=device_release
};

////////////////////////////////////

//загрузка процессора и каждого из его ядра в отдельности 0 - общая по про
typedef struct{
    unsigned long usage[16];
} proc_info_t;

//сбор информации по загрузке процессора, итерация
long rangCPUInfo(proc_info_t *info){
    struct file *fp;
    unsigned char tempRead[1024];
    int iRead_offset=0, iStrLen, index=0, iCPU;

    static unsigned long pre_total[16]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    unsigned long cpu[16], nice[16], system[16], idle[16];
    unsigned long used[16], total[16];

    //открываем файл с данными по загрузке процессора
    fp=file_open("/proc/stat", O_RDONLY, 0);

    //читаем по строчно
    do{
        //читаес строку из файла
        iStrLen=fgets(tempRead, fp, iRead_offset);
        if(iStrLen==0) //если достигнут конец файла
            break;
        iRead_offset+=iStrLen; //увеличиваем смещение от начала файла

        //если первые символы в строке 'cpu'
        if(tempRead[0]=='c' && tempRead[1]=='p' && tempRead[2]=='u'){
            sscanf(tempRead, "%s %lu %lu %lu %lu", &cpu[index], &nice[ind
            index++;
        }
    }

```

```

}while(true);

//закрываем файл
filp_close(fp, NULL);

//расчет по процессору и каждому ядру в отдельности
for(iCPU=0; iCPU<index; iCPU++){
    used[iCPU]=cpu[iCPU]+nice[iCPU]+system[iCPU];
    total[iCPU]=cpu[iCPU]+nice[iCPU]+system[iCPU]+idle[iCPU];

    if(pre_total[iCPU]==0 || pre_used[iCPU]==0) //если первый расчет
        info->usage[iCPU]=0;
    else //если второй и послед
        info->usage[iCPU]=100*(used[iCPU]-pre_used[iCPU])/(total[iCPU]-pre_used[iCPU]);

    //данные текущего расчета будут использоваться для следующего
    pre_used[iCPU]=used[iCPU];
    pre_total[iCPU]=total[iCPU];
}

return index;
}

//информация по загрузке процессора
void infoCPU(void){
    proc_info_t info;
    char sBuffer[256];
    int index, lenString, iCPU;
    struct timespec curr_tm;

    //выполняем чтение и расчет
    index=rangCPUInfo(&info);

    if(bRang==2){ //если есть данные предвещущего расчета
        getnstimeofday(&curr_tm);
        for(iCPU=0; iCPU<index; iCPU++){
            lenString=sprintf(sBuffer, "TIME %.2lu:%.2lu:%.2lu:%.6lu - Load: %.2f%%\n",
                               curr_tm.tv_sec/3600, curr_tm.tv_sec/60, curr_tm.tv_sec,
                               curr_tm.tv_nsec/1000, info.usage[iCPU]);
            putBufferDevice(sBuffer, lenString);
        }
    }
    else //если еще был один расчет
        bRang=2;
}

```

```
}
```

```
void infoRAM(void){
    char tempRead[256], sBuffer[256];
    int lenString, iRead_offset=0, iStrLen;
    struct file *fp;
    struct timespec curr_tm;

    //получаем текущее значения времени
    getnstimeofday(&curr_tm);

    //открываем файл инф о состоянии памяти
    fp=file_open("/proc/meminfo", O_RDONLY, 0);

    do{
        //читаем по строчно
        iStrLen=fgets(tempRead, fp, iRead_offset);
        if(iStrLen==0) //если достигнут конец файла
            break;
        iRead_offset+=iStrLen; //увеличиваем смещение в файле

        //если строка начинается с 'MemTotal'
        if(tempRead[0]=='M' && tempRead[1]=='e' && tempRead[2]=='m' && tempRead[6]=='a' && tempRead[7]=='l'){
            lenString=sprintf(sBuffer, "TIME %.2lu:%.2lu:%.2lu:%.6lu -
                                   (curr_tm.tv_sec/3600)%24, (curr_tm.tv_
                                   tempRead);
            putBufferDevice(sBuffer, lenString);    //сохраняем в буфере
        }
        //если строка начинается с 'MemFree'
        if(tempRead[0]=='M' && tempRead[1]=='e' && tempRead[2]=='m' && tempRead[6]=='e'){
            lenString=sprintf(sBuffer, "TIME %.2lu:%.2lu:%.2lu:%.6lu -
                                   (curr_tm.tv_sec/3600)%24, (curr_tm.tv_
                                   tempRead);
            putBufferDevice(sBuffer, lenString);    //сохраняем в буфере
        }
    }while(true);

    //закрываем файл
    filp_close(fp, NULL);
}
```

```
void infoPS(void){
```

```

struct task_struct *g, *p;
char sBuffer[256];
int iBuf;
struct timespec curr_tm;

//получаем текущее значения времени
getnstimeofday(&curr_tm);

//получаем структура с инф по процессу
do_each_thread(g, p){
    iBuf=sprintf(sBuffer, "TIME  %.2lu:%.2lu:%.2lu:%.6lu -Task %s (pid
                        (curr_tm.tv_sec/3600)%24, (curr_tm.tv_sec/60)%60,
                        p->comm, task_pid_nr(p));
    putBufferDevice(sBuffer, iBuf); //сохраняем в буфер
}while_each_thread(g, p);
}

int fgets(unsigned char *name, struct file *fp, int iRead_offset){
    int readByte=0;
    char tempChar;

    while(true){
        if(file_read(fp, iRead_offset, &tempChar, 1)!=0 && tempChar!='\n')
            name[readByte]=tempChar;
        else
            break;
        ++iRead_offset;
        ++readByte;
    }

    if(readByte==0) //если нет прочитанных символов
        return 0;

    //последний символ 0
    ++readByte;
    name[readByte]=0;

    return readByte;
}

void putBufferDevice(char *message, int lenString){
    int iBuf=0;

    //

```



```

for(; iBuf<lenString; iBuf++){
    if(pointText==SIZE_BUFFER_DEVICE)    //если достигнут конец буфера,
        pointText=0;                    //устанавливаем указатель в на

    textUser[pointText]=message[iBuf];
    ++pointText;
}

if(pointText==SIZE_BUFFER_DEVICE)    //если достигнут конец буфера, нач
    textUser[0]=0;                    //обнуляем буфер
else                                  //устанавливаем последний символ 0
    textUser[pointText]=0;
}

struct file *file_open(const char *path, int flags, int rights){
    struct file *fp=NULL;
    mm_segment_t oldfs;
    int err=0;

    //выделяем область в памяти для отображения файла
    oldfs=get_fs();
    set_fs(get_ds());
    //открываем файл
    fp=filp_open(path, flags, rights);
    set_fs(oldfs);
    if(IS_ERR(fp)){ //если ошибка открытия
        printk("*** sysinfo - error open file - %s\n", path);
        err=PTR_ERR(fp);
        return NULL;
    }

    return fp;
}

int file_read(struct file *file, unsigned long long offset, unsigned char
mm_segment_t oldfs;
int ret;

//файл не открыт
if(file==NULL)
    return 0;

oldfs=get_fs();
set_fs(get_ds());

```

```

    //чтения данных из файла
    ret=vfs_read(file, data, size, &offset);
    set_fs(oldfs);

    return ret;
}

static int thread_sysinfo(void *data){
    while(!kthread_should_stop()){
        //cpu
        infoCPU();

        //ram
        infoRAM();

        //proccess
        infoPS();

        //задержка перед след. расчетом
        msleep(1000);
    }

    return 0;
}

// Функция загрузки модуля
static int __init sysinfo_init(void){
    printk(KERN_ALERT "*** sysinfo module loaded...");

    //Регистрируем устройство и получаем старший номер устройства
    superior_number=register_chrdev(0, DEVICE_NAME, &fops);
    if(superior_number<0){
        printk("FAILED\n*** sysinfo - Registering the character device fai
        return superior_number;
    }

    //Выделяем буффер
    textUser=kmalloc(SIZE_BUFFER_DEVICE, GFP_KERNEL);

    printk("OK\n");
    //Сообщаем присвоенный старший номер устройства
    printk("*** sysinfo - Please, create a dev file with 'mknod /dev/sysin

```

```

//Запускаем поток, для отслеживания состояния системы
taskThread=kthread_run(thread_sysinfo, NULL, "thread_sysinfo");

return 0;
}

// Функция выгрузки модуля
static void __exit sysinfo_exit(void){
    printk(KERN_ALERT "sysinfo module is unloaded...");

    //Уничтожение буфера
    printk(" *** sysinfo - Buffer free\n");
    kfree(textUser);

    //Останавливаем поток
    printk("*** sysinfo - Thread stop\n");
    kthread_stop(taskThread);

    //Освобождаем устройство
    printk("*** sysinfo - destroy device\n");
    unregister_chrdev(superior_number, DEVICE_NAME);
}

//Указываем функции загрузки и выгрузки
module_init(sysinfo_init);
module_exit(sysinfo_exit);

static int device_open(struct inode *inode, struct file *file){
    if(isDeviceOpen)
        return -EBUSY;
    isDeviceOpen++;

    //указатель на буфер в начало
    p_textUser=textUser;

    return 0;
}

static int device_release(struct inode *inode, struct file *file){
    isDeviceOpen--;
    return 0;
}

static ssize_t device_write(struct file *filp, const char *buff, size_t le

```

```

    printk("Sorry, this operation isn't supported.\n");
    return -EINVAL;
}

static ssize_t device_read(struct file *filp, char *buffer, size_t length,
    int byte_read=0;

    //если текста нет
    if(pointText==0)
        return 0;

    //выводим пока есть буфер и не закончили текст
    while(length && *p_textUser){
        //выводим
        put_user(*(p_textUser++), buffer++);

        length--;
        byte_read++;
    }

    return byte_read;
}

```