МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение
высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»
Кафедра 43

КУРСОВОЙ ПРОЕКТ
ЗАЩИЩЕН С ОЦЕНКОЙ
РУКОВОДИТЕЛЬ
ст.преп _____ М.Д.Поляк

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОМУ ПРОЕКТУ

Мониторинг USB

по дисциплине: ОПЕРАЦИОННЫЕ СИСТЕМЫ

РАБОТУ ВЫПОЛНИЛ
СТУДЕНТКА ГР. 4331 _____ Т.Э. Иванова.

Санкт-Петербург, 2016 г.

Содержание

1. Цель работы

Цель работы: реализовать демон для flash накопителей, осуществляющий постоянный мониторинг нахождения нужного USB накопителя и записывающий информацию в лог под ОС Linux

2. Задание

Реализовать демон для flash-накопителя, работающего через интерфейс USB, реализующий мониторинг USB портов на наличие нужного накопителя с правильным серийным номером (прописанном в тексте программы). Демон производит запись в лог с указанием нахождения или ненахождения нужного накопителя в портах и точного времени проверки.

3. Техническая документация

3.1. Установка

Склонировать репозиторий с github при помощи команды:

```
git clone https://github.com/ivanovaTE/OS
```

Для работы демона должен быть установлен пакет gcc.

3.2. Использование

Демон компилируется командой gcc program.c -o program и зупускается командой ./program. После запуска демон выдаст номер процесса. При необходимости его можно будет завершить вручную командой kill "номер процесса".

4. Выводы

В процессе выполнения данной курсовой работы мною были получены знания и навыки, необходимые для работы с USB носителями, демонами в ОС Linux.

5. Приложения

deamon.py:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <string.h>
#include <time.h>

static char* path_log_file = (char*)"/home/tanya/Documents/daemon.log";
static char* serial = (char*)"36UUGDC5VEG79P2Z";
static char* dir_serial1 = (char*)"/media/tanya/EC1667C216678C80/";

char* getSerial( char *str )
{
    ssize_t len;
    char buf[256];
    char *p;
    char buf2[256];
    int i;
    static char comText[256];
    bzero(comText, 256);
    strcpy(comText, "");

    len = readlink(str, buf, 256);
    if (len <= 0) {
        return (char*)"-";
    }
    buf[len] = '\0';
    sprintf(buf2, "%s/%s", "/sys/block/", buf);
    for (i=0; i<6; i++) {
        p = strrchr(buf2, '/');
        *p = 0;
    }
    strcat(buf2, "/serial");

    int f = open(buf2, 0);
    if (f == -1) return (char*)"-";
    len = read(f, buf, 256);
    if (len <= 0) {
        return (char*)"-";
    }
    buf[len-1] = '\0';
    strcat(comText, buf);

    return comText;
}

int isSerial(char* ser)
{
```

```
    if (strcmp(getSerial((char*)"/sys/block/sdb"), ser) == 0
        || strcmp(getSerial((char*)"/sys/block/sdc"), ser) == 0
        || strcmp(getSerial((char*)"/sys/block/sdd"), ser) == 0
        || strcmp(getSerial((char*)"/sys/block/sdf"), ser) == 0
        || strcmp(getSerial((char*)"/sys/block/sdg"), ser) == 0)
    {
        return 1;
    }
    return 0;
}


int main(int argc, char* argv[])
{
FILE *fp= NULL;
pid_t process_id = 0;
pid_t sid = 0;
// Create child process
process_id = fork();
// Indication of fork() failure
if (process_id < 0)
{
printf("fork failed!\n");
// Return failure in exit status
exit(1);
}
// PARENT PROCESS. Need to kill it.
if (process_id > 0)
{
printf("process_id of child process %d \n", process_id);
// return success in exit status
exit(0);
}
//unmask the file mode
umask(0);
//set new session
sid = setsid();
if(sid < 0)
{
// Return failure
exit(1);
}


chdir("/");
// Close stdin. stdout and stderr
close(STDIN_FILENO);
close(STDOUT_FILENO);
close(STDERR_FILENO);
// Open a log file in write mode.
fp = fopen (path_log_file, "w+");
while (1)
```

```
{

sleep(1);
if(isSerial(serial)) {
time_t t=time(NULL);
struct tm tm= *localtime (&t);

fprintf(fp, "Correct USB in d-d-d-d n", tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);

}
else {time_t t=time(NULL);
struct tm tm= *localtime (&t);

fprintf(fp, "No USB in d-d-d-d n", tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);}
fflush(fp);

}
fclose(fp);
return (0);
}
```