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

Резервное копирование

по дисциплине: ОПЕРАЦИОННЫЕ СИСТЕМЫ

РАБОТУ ВЫПОЛНИЛ
СТУДЕНТК ГР. 4331 _____ Князян.Р.А.

Санкт-Петербург, 2016 г.

Содержание

1. Цель работы

Цель работы: реализовать демон для flash накопителей, осуществляющий резервное копирование данных под ОС Linux

2. Задание

Реализовать демон для flash-накопителя, работающего через интерфейс USB, реализующий копирование файлов между двумя flash-накопителями. При подключении двух flash-накопителей с заданными серийными номерами, демон должен запускать копирование файлов с одного накопителя на другой и сохранять текстовый лог успешно завершенных операций копирования на обоих flash-накопителях.

3. Техническая документация

3.1. Установка

Склонировать репозиторий с github при помощи команды:

```
git clone https://github.com/RubenKnyazyan/Kurs_project.git
```

Для работы демона необходима версия ядра не менее 3.19.

3.2. Использование

1) Сборка проекта:
   Необходимо перейти в корневой каталог репозитория и вызвать команду make. Демон будет скомпилирован.

2) Запуск проекта:
   После того, как демон был скомпилирован, он будет и автоматически запущен

3) Использование демона:
   Перед использованием демона необходимо указать серийные номера флеш-накопителей, путь к каталогу флеш-накопителей. Задаются они в файле deamon.c, (пример)

   ```
   //указываем серийный номер накопителей
   serial1 = (char*)"07AB1608151B5B72";
   serial2 = (char*)"c3f41ad10ff131";
   dir_serial1 = (char*)"/media/ruben/STORE/";
   dir_serial2 = (char*)"/media/ruben/B480-C68C/";;
   ```

   После того, как мы подключили накопитель, в log-файле появится запись о том, что демон начал свою работу. Так же, мы увидим, что в log'е отображаются все скопированные на диск файлы.

4) Выключение и удаление проекта:
   Чтобы выключить демона нам понадобиться диспетчер задач, к примеру htop. Запускаем htop, нажимаем F4, выбираем command, далее вводим ./deamon.o, и мы увидим запущенный процесс нашего демона. Нажимая F9 нам будет предложено "убить"процесс, соглашаемся и жмем Enter. Удаление исполняемого файла демона можно произвести вызвав команду make clean. в корневой директории репозитория.

4. Выводы

В процессе выполнения данной курсовой работы мною были получены знания и навыки, необходимые для работы с USB носителями, потоками, файлами и папками, в ОС семейства Linux, а так же знания и навыки в написании демонов.

5. Приложения

deamon.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/time.h>
#include <unistd.h>
#include <errno.h>
#include <fcntl.h>
#include <syslog.h>
#include <fstream>  // to copy files
#include <iostream>
#include <dirent.h> //for searching files
#include <sys/stat.h> // для создания папки
#include <sys/types.h> // для создания папки



static char* serial1 = (char*)"07AB1608151B5B72";
static char* dir_serial1 = (char*)"/media/ruben/STORE/";
static char* serial2 = (char*)"c3f41ad10ff131";
static char* dir_serial2 = (char*)"/media/ruben/B480-C68C/";
//static char* dir_comp = (char*)"/home/ruben/try2/1/";
static char* path_log_file1 = (char*)"/media/ruben/STORE/1.log";
static char* path_log_file2 = (char*)"/media/ruben/B480-C68C/1.log";

int Deamon(void);
int writeLog(char msg[256]);
char* getSerial( char *str );
int isSerial(char* ser);
void copyFile(std::string pathFrom, std::string pathTo, std::string fileName);
void copyDir(std::string pathFrom, std::string pathTo);
int fileSize( std::fstream &f );

//функция получающая серийный номер
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;
}

//функция проверяющая серийный номер на соответствие заданному
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 writeLog(char msg[256]) { //функция записи строки в лог

    FILE * pLog;
    pLog = fopen(path_log_file1, "a");
    if(pLog == NULL) {
        return 1;
    }
    char str[1024];
    bzero(str, 1024);
    strcat(str, msg);
    strcat(str, (char*)"\n");
    fputs(str, pLog);
    fclose(pLog);

FILE * pLog2;
```

```
    pLog2 = fopen(path_log_file2, "a");
    if(pLog2 == NULL) {
        return 1;
    }
    char str2[1024];
    bzero(str2, 1024);
    strcat(str2, msg);
    strcat(str2, (char*)"\n");
    fputs(str2, pLog2);
    fclose(pLog2);
    return 0;
}


void copyFile(std::string pathFrom, std::string pathTo, std::string fileName)
{
    writeLog((char*) "Copping...");

    std::string _pathFrom = pathFrom;
    std::string _pathTo = pathTo;

    _pathFrom = _pathFrom+fileName;
    _pathTo = _pathTo+fileName;

    char myA[fileName.size()+1];
    strcpy(myA, fileName.c_str());

    if (strcmp(myA,".") == 0 || strcmp(myA,"..") == 0)
    {
        writeLog((char*) "New Branch");

    }
    else
    {
        const char * c1 = _pathFrom.c_str();
        const char * c2 = _pathTo.c_str();

        char * buffer;
        buffer = new char;

        std::fstream infile(c1, std::ios::in | std::ios::binary);
        if (!infile.is_open()) {// если файл не открыт
           writeLog((char*) "Файл не открывается!"); // сообщить об этом
           //writeLog((char*) c1);
           //
       }
        else
        {
            writeLog((char*) "Файл открылся!");
            infile.read(buffer, sizeof(char)); //читаем первый сивол
            std::string log = "";
```

```
            if (infile.fail() && !infile.eof()) //проверка на папку
            {
                infile.close();
                remove(_pathTo.c_str());
                log = c2;
                log = "dir: " + log;
                mkdir(_pathTo.c_str(), 0755);
                //writeLog((char*) _pathTo);
                writeLog((char*)" -- Это папка.\n");
                copyDir(_pathFrom + "/", _pathTo + "/");
            }
            else
            { //иначе файл
                std::fstream outfile(c2, std::ios::out | std::ios::app | std::ios::binar
                outfile.seekg(0, std::ios::beg);

                if (fileSize(outfile) != fileSize(infile) && fileSize(infile) > 0) //есл
                {
                    outfile.close();
                    outfile.open(c2, std::ios::out | std::ios::binary);
                    log = c2;
                    log = "new file: " + log;
                    while (!infile.eof() && infile.good())
                    {
                        outfile.write(buffer, sizeof(char));
                        infile.read(buffer, sizeof(char));
                    }
                } else
                {
                    log = c2;
                    log = "file exists: " + log;
                }
                infile.close();
                outfile.close();
            }
            writeLog((char *) log.c_str());
        }
        delete buffer;
        std::cout << _pathFrom << "->" << _pathTo << std::endl;
    }
}

void copyDir(std::string pathFrom, std::string pathTo)
{
     writeLog((char*) "Папка тут!");
    struct dirent **namelist;
    int n;

    n = scandir(pathFrom.c_str(), &namelist, 0, alphasort);
    if (n >= 0)
    {
        while (n--)
```

```
        {
            copyFile(pathFrom, pathTo, namelist[n]->d_name);
            free(namelist[n]);
        }
        free(namelist);
    }
}


int fileSize( std::fstream &f )
{
    int size, saveTellg;
    saveTellg = f.tellg();
    f.seekg(0, std::ios::end);
    size = f.tellg();
    f.seekg(saveTellg, std::ios::beg);
    return size;
}

int Deamon(void) { //наш бесконечный цикл демона
    while(1) {
    if(isSerial(serial1) && isSerial(serial2)) // Если это необходимая флешка
        {
            writeLog((char*) dir_serial1);
writeLog((char*) "  to  ");
writeLog((char*) dir_serial2);
            copyDir(dir_serial1, dir_serial2);//запускаем функцию копирования
//copyDir(dir_serial2, dir_serial1);//запускаем функцию копирования
            writeLog((char*)"\n++++++++++++++++++++++++++++++++++++++++\n");
        }
        sleep(3);//ждем до следующей итерации
    }
    return 0;
}

int main(int argc, char* argv[]) {
    writeLog((char*)"USBDeamon Start");

    pid_t parpid, sid;

    parpid = fork(); //создаем дочерний процесс

    if(parpid < 0)
        {
            exit(1);
        }
    else if(parpid != 0)
        {
            exit(0);
        }

    umask(0);//даем права на работу с фс
```

```
    sid = setsid();//генерируем уникальный индекс процесса

    if(sid < 0)
        {
            exit(1);
        }

    if((chdir("/")) < 0)
        {//выходим в корень фс
            exit(1);
        }

    close(STDIN_FILENO);//закрываем доступ к стандартным потокам ввода-вывода
    close(STDOUT_FILENO);
    close(STDERR_FILENO);

    return Deamon();
}
```