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.		Поляк М.Д.
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ

ДИСПЕТЧЕР ЗАДАЧ

по дисциплине: ОПЕРАЦИОННЫЕ СИСТЕМЫ И СЕТИ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. 4331		
	подпись, дата	инициалы, фамилия

Санкт-Петербург
2016

1 Цель работы

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

2 Задание(5 вариант)

Диспетчер задач. Реализовать демон для мониторинга использования блочных устройств. Необходимо в реальном времени отслеживать число и размер операций чтения и записи для блочных устройств.

3 Техническая документация

1. Сборка проекта:
   Скачиваем файлы с репозитория на github при помощи команды:

   ```
   git clone https://github.com/gurianoff/IOstatistic/IOstatistic.git
   ```

2. Запуск проекта:
   Шаг 1: Необходимо перейти в корневой каталог репозитория и вызвать команду:

   ```
   gcc daemon.c -o daemon.exe
   ```

   Демон будет скомпилирован.
   Шаг 2: После того, как демон скомпилирован, его необходимо загрузить, для этого надо воспользоваться командой:

   ```
   ./daemon.exe
   ```

   Шаг 3: Демон каждые 5 секунд обновляет информацию на экране. Информация состоит из имени устройства, данных о количестве операций чтения и записи, и данных о количестве считанных секторов. Также приведено уточнение, что сектор состоит из 512 байт.
   Шаг 4: Выключение демона осуществляется командой

   ```
   ^C
   ```

   , либо уничтожением процесса командой

   ```
   kill pid
   ```

4 Выводы

В ходе курсового проекта была изучена реализация демонов и способы получения информации об использовании блочных устройств. В последствии демон был реализован на языке C. Полученный результат удовлетворяет всем поставленным целям данного проекта.

5 Приложение

daemon.c:

```
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>
#include <syslog.h>
#include <string.h>

#define DISKSTATS_FILE "/proc/diskstats"

void clrscr(void);

int main(void) {

  /* Наши ID процесса и сессии */
  pid_t pid, sid;

  /* Ответвляемся от родительского процесса */
  pid = fork();
  if (pid < 0) {
    exit(EXIT_FAILURE);
  }
  /* Если с PID'ом все получилось, то родительский процесс можно завершить
  if (pid > 0) {
      clrscr();
    while(1) {
printf("%8s","NameDev");
printf("%8s", "NumRe");/*kol-vo*/
printf("%8s", "SecRe");/*razmer*/
printf("%8s", "NumWr");/*kol-vo*/
printf("%8s \n", "SecWr");/*razmer*/
       struct device {
int majorNumber;
int minorNumber;
char deviceName[6];
int readsCompletedSuccessfully;
int readsMerged;
int sectorsRead;
int timeSpentReading;
```

```
int writesCompleted;
int writesMerged;
int sectorsWritten;
int timeSpentWriting;
int IOsCurrentlyInProgress;
int timeSpentDoingIOs;
int weightedTimeSpentDoingIOs;
};
FILE * file = fopen(DISKSTATS_FILE,"r");
FILE * fileCopy = fopen(DISKSTATS_FILE,"r");
int lines_count = 0;
//s4itaem kol-vo strok
if (file==NULL)
printf("Trouble of reading file!");
else {
while(!feof(file))
{
if(fgetc(file)=='\n')
lines_count++;
}
lines_count++;
        fclose(file);
//kajdaia stroka iz faila eto structura soderjawaia polia
struct device block[lines_count];
char i=0;
while(fscanf (fileCopy, "%d%d%s%d%d%d%d%d%d%d%d%d%d", &(block[i].majorNu

//if(block[i].majorNumber==11){//comment this line you need all devices
printf("%6s %8d %8d %6d %6d \n", block[i].deviceName, block[i].readsComple
//}//comment this line you need all devices
i++;
}
      fclose(fileCopy);
printf("\n");
printf("Note: Program shows you the number of hits and the number of used
printf("One sector is 512 bytes\n");
      sleep(5);
      clrscr();
     }
    }
    exit(1);
  }

  /* Изменяем файловую маску */
```

```
  umask(0);

  /* Здесь можно открывать любые журналы */
  /* Создание нового SID для дочернего процесса */
  sid = setsid();
  if (sid < 0) {
    /* Журналируем любой сбой */
    exit(EXIT_FAILURE);
  }

  /* Изменяем текущий рабочий каталог */
  if ((chdir("/")) < 0) {
    /* Журналируем любой сбой */
    exit(EXIT_FAILURE);
  }

  /* Закрываем стандартные файловые дескрипторы */
  close(STDIN_FILENO);
  close(STDOUT_FILENO);
  close(STDERR_FILENO);

  exit(EXIT_SUCCESS);
}

void clrscr(void){
printf("\033[2J"); /*clear the entire screen*/
printf("\033[0;0f"); /*Move cursor to the top*/
}
```