Федеральное государственное бюджетное образовательное учреждение

высшего образования “Сибирский государственный университет телекоммуникаций и информатики”

Факультет ИВТ

Кафедра вычислительных систем

**Курсовая работа**

на тему “Разработка инструментов командой строки OC GNU/LINUX”

Вариант 2.5 “Проверка целостности файлов”

Выполнил:

Студент гр. ИВ-223

Зятчин Д. Ю.

Проверил:

к.т.н., доцент Кафедры ВС

Фульман В. О.

Новосибирск, 2023

Оглавление

[Тема курсовой работы 3](#_Toc148214655)

[Задание на курсовую работу 4](#_Toc148214656)

[Анализ задачи 5](#_Toc148214657)

[Тестовые данные 6](#_Toc148214658)

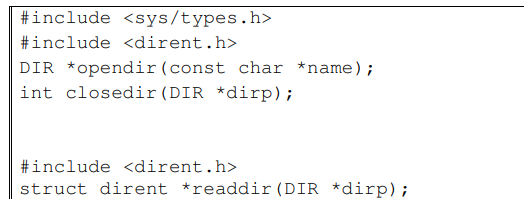
[Листинг программы 7](#_Toc148214659)

# Тема курсовой работы

Тема курсовой работы “Разработка инструментов командной строки OC GNU/LINUX”. В данной теме требуется разработать программные утилиты командной строки в соответствии с вариантом задания. Все входные данные передаются через аргументы командной строки. Использование системного вызова ***system*** запрещено, допустимо использовать только системные вызовы ОС GNU\Linux.

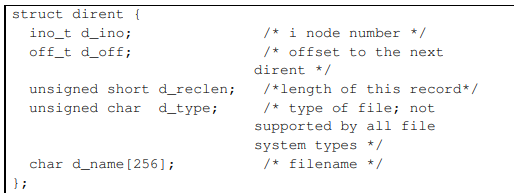
**Взаимодействие с файловой системой**

Для выполнения некоторых заданий данного и последующих разделов потребуются инструменты, позволяющие получать информацию о содержимом директорий. В ОС GNU/Linux для решения этой задачи предусмотрены следующие функции:

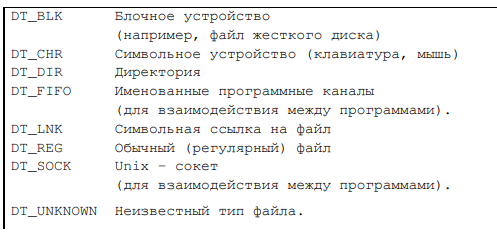


Функция ***opendir*** "открывает" директорию для чтения, на ее вход передается строка, содержащая путь к директории, возвращаемым значением является указатель на структуру DIR, которая используется для дальнейших действий с открытой директорией.

Функция ***readdir*** выполняет последовательный проход по содержимому открытой директории. При каждом ее вызове она возвращает указатель на структуру типа ***struct dirent***, элементы которой приведены ниже:



Для выполнения заданий курсовой работы интерес представляют поля *d\_name* и *d\_type*. Первое содержит имя файла или директории, второе – позволяет определить тип элемента файловой системы, это поле представляет из себя целое число, которое устанавливается функцией ***readdir*** в одно из допустимых значений, которые представлены константами, определенными в файле *dirent.h*:



В рамках курсовой работы предусмотрена обработка только директорий (*DT\_DIR*) и регулярных файлов (*DT\_REG*), другие типы файлов должны игнорироваться.

Функция ***closedir*** закрывает директорию, после ее вызова функция ***readdir*** не должна использоваться с данной структурой типа *DIR*.

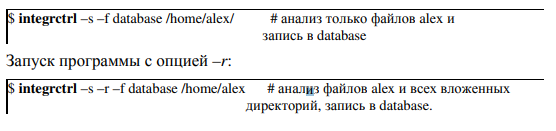
Дополнительную информацию об объектах файловой системы можно получить, используя функцию ***stat*** (самостоятельное изучение).

# Задание на курсовую работу

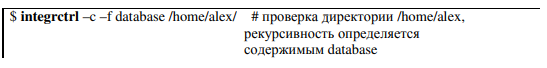
**Задание**

Реализовать программу ***integrctrl*** (Integrity Control) проверки целостности содержимого файловой системы. Использование программы состоит из двух шагов.

1. Запись информации о целостности в файл - базу данных. Для активации данного режима программа должна быть запущена с опцией –s (save integrity info). Если дополнительно указана опция –r, то рекурсивно анализируются все вложенные каталоги. Например:



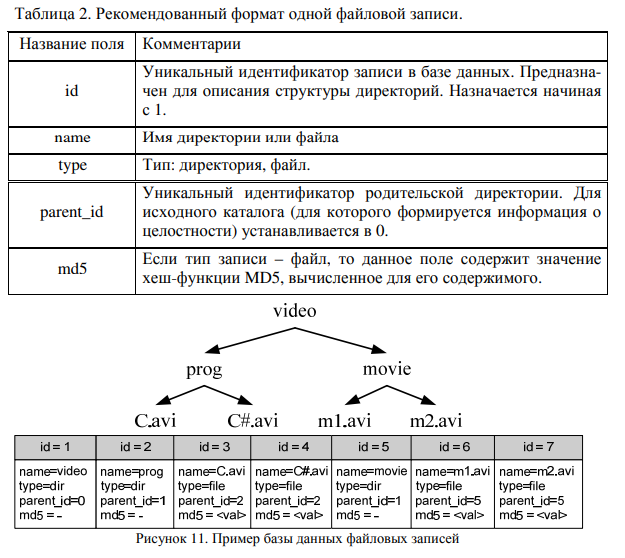
2. Сверка информации о целостности, расположенной в указанном файле. Для активации данного режима необходимо использовать опцию –c (check integrity):



**Указание к выполнению задания**

Проверка целостности файлов должна осуществляться с применением технологии хеширования. Для содержимого каждого обрабатываемого файла вычисляется значение хеш-функции с малой вероятностью коллизии. В рамках данной работы предлагается использование функции MD5. Для генерации хеш 20 кодов MD5 рекомендуется использовать исходные коды, расположенные на сайте Кафедры ВС (http://cpct.sibsutis.ru/~artpol/downloads/prog/s2/MD5.tar.bz2) или реализацию в библиотеке OpenSSL (описание ее использования расположено по адресу: <http://www.firststeps.ru/linux/r.php?18>).

Для хранения базы данных рекомендуется использовать бинарные файлы. Предполагается, что структура каталогов не может изменяться. Перемещение директории рассматривается как ее удаление и создание нового каталога в другом месте. Формат одной файловой записи представлен в виде таблицы 2. Пример записи приведен на рисунке 11.



# Анализ задачи

В данной курсовой работе способ представления данных, над которыми производятся операции, будет выглядеть следующим образом:



Для реализации поставленного задания, нам потребуется использовать пару библиотек: *stdio.h, stdlib.h, string.h, openssl/md5.h, dirent.h, sys/stat.h.*

***stdio.h*** - эта библиотека предоставляет функциональность для ввода/вывода в стандартных потоках, таких как *printf* и *fopen*. В коде использует *printf*, *perror*, *fopen*, *fclose*, *fread*, и *fscanf*, что является частью стандартной библиотеки ввода/вывода Си.

***stdlib.h*** – это заголовочный файл в стандартной библиотеке языка программирования C, который предоставляет функции для управления памятью, конвертации строк в числа, управления выполнением программы и другие общие утилиты.

***string.h*** - содержит функции для работы со строками, такие как strcpy, strcmp, и snprintf. В коде strcmp используется для сравнения строк, а snprintf для безопасного форматирования строк.

***openssl/md5.h*** - используется для вычисления хеш-суммы MD5. Это не стандартная библиотека Си, а библиотека OpenSSL, которая предоставляет широкий спектр криптографических функций.

***dirent.h*** - эта библиотека позволяет работать с директориями и файлами внутри директорий. В коде она используется для открытия и чтения содержимого каталогов.

***sys/stat.h*** - предоставляет функции для получения информации о статусе файла (например, st\_mode). В коде она используется для проверки, является ли элемент каталога файлом или директорией.

Также в коде программы используются макросы #define:

***#define MAX\_FILENAME\_LENGTH 256*** - устанавливает максимальную длину имени файла.

***#define MAX\_HASH\_LENGTH 32*** - устанавливает максимальную длину хеша (MD5 обычно 32 символа в шестнадцатеричном формате).

***#define DATABASE\_FILENAME "integrity\_db.txt"*** - устанавливает имя файла для базы данных целостности.

Для хранения информации о каждом файле и дальнейшей записи этой информации в коде используется структура *IntegrityInfo.* Псевдокод структуры:

struct IntegrityInfo{

    int id - Идентификатор файла или директории.

    char name- Имя файла или директории.

    char type - Тип элемента: 'D' для директории и 'F' для файла. Используется для различения между файлами и директориями

    int parent\_id - Идентификатор родительской директории. Это поле указывает на то, к какой директории принадлежит файл или поддиректория. В контексте данной программы, где сохраняется информация о целостности файловой системы, это может быть полезным для построения структуры дерева файловой системы.

    char md5- Хеш-сумма MD5. Это значение представляет собой уникальный "отпечаток" содержимого файла или директории.

}

Для вычисления MD5-хеша содержимого файла в коде представлена функция *calculateMD5*. Функция открывает файл, создает контекст MD5, читает файл блоками, обновляет контекст MD5 и, наконец, получает окончательный хеш. Освобождение ресурсов и обработка также учтены в коде. Псевдокод функции:

Function calculateMD5(filename, md5):

    file = open(filename, "rb")

    if file is not opened:

        print("Error opening file")

        exit(EXIT\_FAILURE)

    md5Context = createMD5Context()

    buffer\_size = 8192

    buffer = allocateBuffer(buffer\_size)

    while (readBytes = readFromFile(file, buffer, buffer\_size)):

        updateMD5Context(md5Context, buffer, readBytes)

    finalizeMD5(md5Context, md5)

    closeFile(file)

End Function

Также нам понадобится функция *saveIntegrityInfo* для сканирования директории, сохранения информации о целостности файлов в базу данных и, при необходимости, рекурсивного сканирования вложенных директорий. Функция работает так: она открывает директорию, открывает файл базы данных, читает записи из директории, вычисляет MD5-хеш для файлов и сохраняет информацию в базу данных. Если установлен флаг рекурсии и обнаружена поддиректория, функция рекурсивно вызывается для сканирования поддиректории. В конце закрываются файл базы данных и директория. Псевдокод функции:

Function saveIntegrityInfo(directory, recursive, parent\_id):

    dir = openDirectory(directory)

    if dir is not opened:

        print("Error opening directory")

        exit(EXIT\_FAILURE)

    dbFile = openDatabaseFile(DATABASE\_FILENAME, "a")

    if dbFile is not opened:

        print("Error opening integrity database file")

        closeDirectory(dir)

        exit(EXIT\_FAILURE)

    entry = readNextEntry(dir)

    while entry is not null:

        if entry is regular file:

            filepath = createFilePath(directory, entry->d\_name)

            md5 = calculateMD5(filepath)

            fprintf(dbFile, "%d %s F %d %s\n", ++parent\_id, entry->d\_name, parent\_id - 1, md5)

        else if recursive and entry is directory and entry is not "." and entry is not "..":

            subdirectory = createSubdirectoryPath(directory, entry->d\_name)

            saveIntegrityInfo(subdirectory, recursive, parent\_id)

        entry = readNextEntry(dir)

    closeFile(dbFile)

    closeDirectory(dir)

End Function

Функция *checkIntegrity* предназначена для проверки целостности файлов, сравнивая сохраненные в базе данных хеш-суммы с рассчитанными хеш-суммами текущего состояния файлов. Она открывает файл базы данных, читает записи из файла, формирует полные пути к файлам, рассчитывает их текущие хеш-суммы и сравнивает их с сохраненными в базе данных. Если хеш-суммы не совпадают, выводится сообщение об ошибке целостности. В конце закрывается файл базы данных. Псевдокод функции:

Function checkIntegrity():

    dbFile = openDatabaseFile(DATABASE\_FILENAME, "r")

    if dbFile is not opened:

        print("Error opening integrity database file")

        exit(EXIT\_FAILURE)

    while readEntry(dbFile, id, name, type, parent\_id, md5):

        filepath = createFilePath(name, type)

        calculatedMD5 = calculateMD5(filepath)

        if md5 is not equal to calculatedMD5:

            print("Integrity check failed for", filepath, "(id:", id, ")")

        else:

            print("Integrity check passed for", filepath, "(id:", id, ")")

    closeFile(dbFile)

End Function

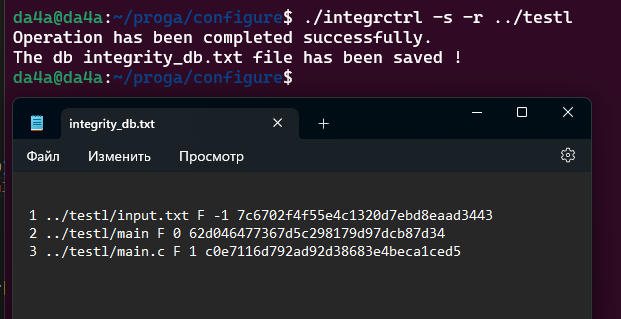
Запуск программы осуществляется следующим образом:

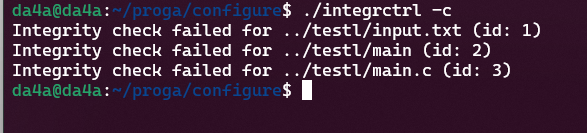
gcc integrctrl.c -o integrctrl -lssl -lcrypto

./integrctrl -s -r [path]

./integrctrl -c

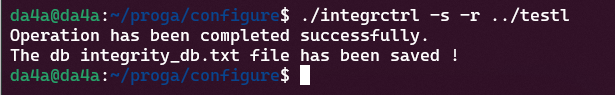
Примерный вывод при запуске программы:

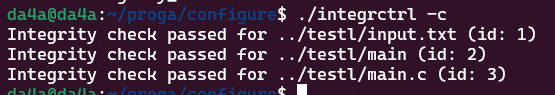




# Тестовые данные

Тест №1 (корректные данные): путь до директории указан



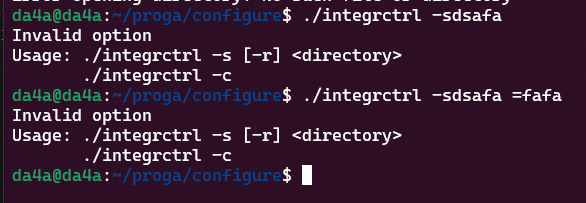


Тест №2 (некорректные данные): если путь до директории не указан => база данных не создалась

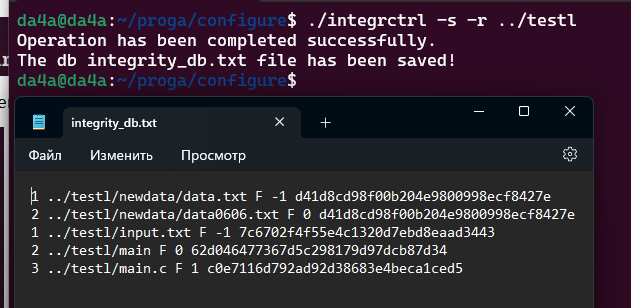


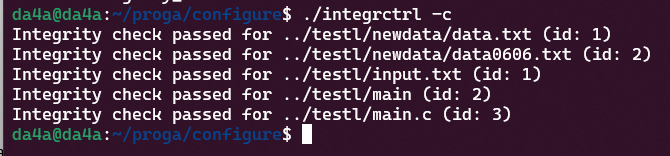


Тест №3 (некорректные данные): если количество больше или меньше положенного

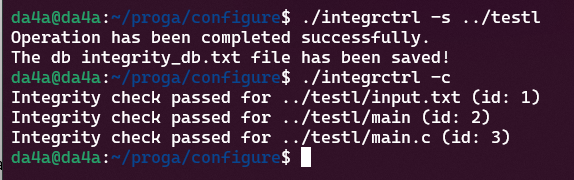


Тест №4 (рекурсивный вызов):

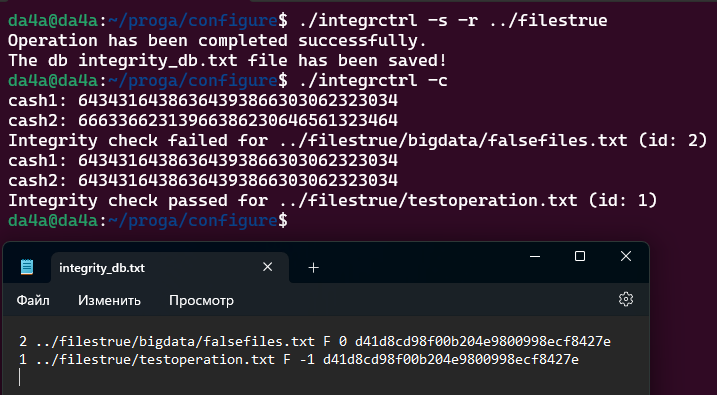




Тест №5 (не рекурсивный вызов):



Тест №6 (проверка целостности файла): проверяем имеющуюся директорию. После записи в дб, изменим один из файлов в директории для изменения его хеша. Для наглядности был добавлен вывод хешей, чтобы удостоверится, что они разные. Стоит заметить, что вывод отличается от данных в бд, все дело в выводе (используем printf("%02x", (unsigned int)md5[i]); и printf("%02x", (unsigned int)calculatedMD5[i])).



# Листинг программы

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <openssl/md5.h>

#include <dirent.h>

#include <sys/stat.h>

#define MAX\_FILENAME\_LENGTH 256

#define MAX\_HASH\_LENGTH 32

#define DATABASE\_FILENAME "integrity\_db.txt"

typedef struct {

    int id;

    char name[MAX\_FILENAME\_LENGTH];

    char type;

    int parent\_id;

    char md5[MAX\_HASH\_LENGTH];

} IntegrityInfo;

void calculateMD5(const char \*filename, char \*md5) {

    FILE \*file = fopen(filename, "rb");

    if (!file) {

        perror("Error opening file");

        exit(EXIT\_FAILURE);

    }

    MD5\_CTX md5Context;

    MD5\_Init(&md5Context);

    const size\_t bufSize = 8192;

    unsigned char buffer[bufSize];

    size\_t bytesRead;

    while ((bytesRead = fread(buffer, 1, bufSize, file))) {

        MD5\_Update(&md5Context, buffer, bytesRead);

    }

    unsigned char hash[MD5\_DIGEST\_LENGTH];

    MD5\_Final(hash, &md5Context);

    for (int i = 0; i < MD5\_DIGEST\_LENGTH; i++) {

        snprintf(md5 + i \* 2, 3, "%02x", hash[i]);

    }

    fclose(file);

}

void saveIntegrityInfo(const char \*directory, int recursive, int parent\_id) {

    DIR \*dir = opendir(directory);

    if (!dir) {

        perror("Error opening directory");

        exit(EXIT\_FAILURE);

    }

    FILE \*dbFile = fopen(DATABASE\_FILENAME, "a");

    if (!dbFile) {

        perror("Error opening integrity database file");

        closedir(dir);

        exit(EXIT\_FAILURE);

    }

    struct dirent \*entry;

    while ((entry = readdir(dir)) != NULL) {

        if (entry->d\_type == DT\_REG) {

            char filepath[MAX\_FILENAME\_LENGTH];

            snprintf(filepath, sizeof(filepath), "%s/%s", directory, entry->d\_name);

            char path[MAX\_FILENAME\_LENGTH];

            strncpy(path, filepath, sizeof(path) - 1);

            path[sizeof(path) - 1] = '\0';

            char md5[MAX\_HASH\_LENGTH];

            calculateMD5(filepath, md5);

            fprintf(dbFile, "%d %s F %d %s\n", ++parent\_id, path, parent\_id - 1, md5);

        } else if (recursive && entry->d\_type == DT\_DIR && strcmp(entry->d\_name, ".") != 0 && strcmp(entry->d\_name, "..") != 0) {

            char subdirectory[MAX\_FILENAME\_LENGTH];

            snprintf(subdirectory, sizeof(subdirectory), "%s/%s", directory, entry->d\_name);

            saveIntegrityInfo(subdirectory, recursive, parent\_id);

        }

    }

    fclose(dbFile);

    closedir(dir);

}

void checkIntegrity() {

    FILE \*dbFile = fopen(DATABASE\_FILENAME, "r");

    if (!dbFile) {

        perror("Error opening integrity database file");

        exit(EXIT\_FAILURE);

    }

    char currentDir[MAX\_FILENAME\_LENGTH];

    getcwd(currentDir, sizeof(currentDir));

    int id, parent\_id;

    char name[MAX\_FILENAME\_LENGTH], type, md5[MAX\_HASH\_LENGTH];

    while (fscanf(dbFile, "%d %s %c %d %s\n", &id, name, &type, &parent\_id, md5) != EOF) {

        char filepath[MAX\_FILENAME\_LENGTH];

        snprintf(filepath, sizeof(filepath), "%s", name);

        if (type == 'D') {

            DIR \*dir = opendir(filepath);

            if (dir) {

                closedir(dir);

            } else {

                perror("Error opening directory");

                exit(EXIT\_FAILURE);

            }

        }

        char calculatedMD5[MAX\_HASH\_LENGTH];

        calculateMD5(filepath, calculatedMD5);

        printf("cash1: ");

        for (int i = 0; i < MD5\_DIGEST\_LENGTH; i++) {

            printf("%02x", (unsigned int)md5[i]);

        }

        printf("\n");

        printf("cash2: ");

        for (int i = 0; i < MD5\_DIGEST\_LENGTH; i++) {

            printf("%02x", (unsigned int)calculatedMD5[i]);

        }

        printf("\n");

        if (memcmp(md5, calculatedMD5, MD5\_DIGEST\_LENGTH) != 0) {

            printf("Integrity check failed for %s (id: %d)\n", filepath, id);

        } else {

            printf("Integrity check passed for %s (id: %d)\n", filepath, id);

        }

    }

    fclose(dbFile);

}

int main(int argc, char \*argv[]) {

    if (strcmp(argv[1], "-s") == 0) {

        int recursive = 0;

        if (argc == 4 && strcmp(argv[2], "-r") == 0) {

            recursive = 1;

        }

        saveIntegrityInfo(argv[argc - 1], recursive, 0);

        printf("Operation has been completed successfully.\nThe db %s file has been saved!\n", DATABASE\_FILENAME);

    } else if (strcmp(argv[1], "-c") == 0) {

        checkIntegrity();

    } else {

        printf("Invalid option\n");

        printf("Usage: %s -s [-r] <directory>\n", argv[0]);

        printf("       %s -c\n", argv[0]);

        return 1;

    }

    return 0;

}