|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | | | |
| Институт информационных технологий (ИТ) | |
| Кафедра инструментального и прикладного программного обеспечения (ИиППО) | |

|  |  |
| --- | --- |
| **ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №5** | |
| **по дисциплине** | |
| «Системное программное обеспечение»  по теме: «Файловая система FAT» | |
|  | |
| Выполнил студент группы ИКБО-16-18 | Пономарев А.Р. |
| Принял преподаватель | Волков М.Ю. |

|  |  |  |
| --- | --- | --- |
| Работа выполнена | «\_\_»\_\_\_\_\_\_\_2021 г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_2021 г. | *(подпись руководителя)* |

Москва 2021

**Цель работы**

Ознакомление со структурой и расположением системной информации и файлов на томе FATxx.

**Задание**

На базе архитектуры WIN 32 Application создать многооконное приложение, где первое окно отображает информацию по клавиатуре, второе окно отображает информацию по мыши и третье окно отображает информацию по таймеру.

1. Создать на внешнем носителе (flash) логический диск и отформатировать его под FAT16 или FAT32. Все дальнейшие действия выполняются с этим диском/томом.

2. Загрузить программу WinHex и скопировать с этого диска первый сектор системных данных – BOOT, FAT и ROOT в файлы BOOT.bin, FAT.bin, ROOT.bin.

3. Создать программу, которая:

а) выводит на экран дамп N-го сектора;

б) выводит на экран расшифрованную таблицу BPB;

в) определяет положение и размер системных данных BOOT, FAT, ROOT – и выводит их значения на экран.

4. Сравнить данные, полученные разработанной программой и WinHex. В случае несовпадения откорректировать алгоритм и отладить программу.

5. Создать на томе каталог и скопировать 1-й сектор каталога в файл DIR.bin.

6. Использовать дампы (фрагменты дампов) полученных файлов и фрагменты копий окон программ для подготовки отчета об изучаемых вопросах 1–5.

7. Изучить алгоритм расположения файла на FAT томе:

а) определение номера начального кластера файла;

б) расчет номеров цепочки кластеров файла. Данный пункт выполняется с помощью программы WinHex и не требует написания кода. В отчете привести алгоритм и иллюстрировать ответ фрагментами дампов родительского директория и таблицы FAT.

8. С помощью WinHex получить дампы таблицы FAT и родительского каталога до и после выполнения каждой команды COPY, MOVE, DEL и RENAME любым программным средством. Исследовать изменение элементов FAT и каталога.

**Ход работы**

Итоговая программа представлена в листинге 1. Результат работы представлен на рисунках 1-2.

Листинг 1

#include <Windows.h>

#include <fstream>

#include <string>

#include <iostream>

using namespace std;

#define NUMBER\_OF\_BYTES\_IN\_SECTOR\_OFFSET 0x0B //по этому смещению находится размер сектора

#define NUMBER\_OF\_SECTORS\_IN\_SECTION\_FIRST\_OFFSET 0x13 //по этому смещению находится количество секторов если оно меньше 65535

#define NUMBER\_OF\_SECTORS\_IN\_SECTION\_SECOND\_OFFSET 0x20 //по этому смещению находится количество секторов, если их больше 65535

string drivePath = "\\\\.\\D:";

UINT getNumberOfBytesInSector() {

short int numberOfBytes; //так как размер этого блока 2 байта

ifstream infile(drivePath.c\_str(), ios::in | ios::binary); // полкючаемся к файлу через поток, и указываем, что он бинарный(in - открыть для чтения)

if (!infile) {

cerr << "ERROR! Device input could not be set!\n";

}

infile.ignore(NUMBER\_OF\_BYTES\_IN\_SECTOR\_OFFSET); //не читаем до того момента, пока не начнется блок с размером сектора?

infile.read((char\*)&numberOfBytes, sizeof(short int)); //читаем 2 байта инфы

if (infile.gcount() == 0) {

cerr << "Data read error!" << endl;

}

infile.close(); //stream close

return (UINT)numberOfBytes;

}

UINT getNumberOfSectorsInSection() {

short int numberOfSectors; //так как размер этого блока 2 байта

UINT numberOfSectorsU;

ifstream infile(drivePath.c\_str(), ios::in | ios::binary); // полкючаемся к файлу через поток, и указываем, что он бинарный(in - открыть для чтения)

if (!infile) {

cerr << "ERROR! Device input could not be set!\n";

}

infile.ignore(NUMBER\_OF\_SECTORS\_IN\_SECTION\_FIRST\_OFFSET); // доходим до первого блока с количеством секторов

infile.read((char\*)&numberOfSectors, sizeof(short int)); //считываем 2 байта информации

if (infile.gcount() == 0) {

cerr << "Data read error!" << endl;

}

infile.close(); //stream close

if (numberOfSectors != 0) { // если в первом блоке было пусто, то читаем вторйо блок

return (UINT)numberOfSectors;

} else {

ifstream infile(drivePath.c\_str(), ios::in | ios::binary); // полкючаемся к файлу через поток, и указываем, что он бинарный(in - открыть для чтения)

if (!infile) {

cerr << "ERROR! Device input could not be set!\n";

}

infile.ignore(NUMBER\_OF\_SECTORS\_IN\_SECTION\_SECOND\_OFFSET);// доходим до второго блока с количеством секторов

infile.read((char\*)&numberOfSectorsU, sizeof(UINT)); // считываем 4 байта информации

if (infile.gcount() == 0) {

cerr << "Data read error!" << endl;

}

infile.close(); //stream close

return numberOfSectorsU;

}

}

void getSectorDump(UINT numberOfBytes, UINT sectorNumber) {

BYTE\* dump = new BYTE[numberOfBytes];

ifstream infile(drivePath.c\_str(), ios::in | ios::binary); // полкючаемся к файлу через поток, и указываем, что он бинарный(in - открыть для чтения)

if (!infile) {

cerr << "ERROR! Device input could not be set!\n";

}

infile.ignore(static\_cast<\_\_int64>(numberOfBytes) \* sectorNumber);

infile.read((char\*)dump, numberOfBytes);

if (infile.gcount() == 0) {

cerr << "Data read error!" << endl;

}

infile.close();

for (int i = 0; i < numberOfBytes / 16; i++) {

printf("%09X| ", numberOfBytes \* sectorNumber + i \* 16);

for (int j = 0; j < 16; j++) {

if (j == 8) {

printf(" ");

}

printf("%02X ", dump[i \* 16 + j]);

}

printf(" | ");

for (int j = 0; j < 16; j++) {

if (dump[i \* 16 + j] != 0x0A && dump[i \* 16 + j] != 0x0D && dump[i \* 16 + j] != 0x09) {// проверка на невидимые символы по таблице ASCII

printf("%c", dump[i \* 16 + j]);

}

}

printf("\n");

}

}

int main() {

UINT numberOfBytes;

UINT numberOfSectors;

numberOfBytes = getNumberOfBytesInSector();

numberOfSectors = getNumberOfSectorsInSection();

cout << "Number of bytes in sector: " << numberOfBytes << endl;

cout << "Number of sectors in section: " << numberOfSectors << endl;

getSectorDump(numberOfBytes, 0);

}

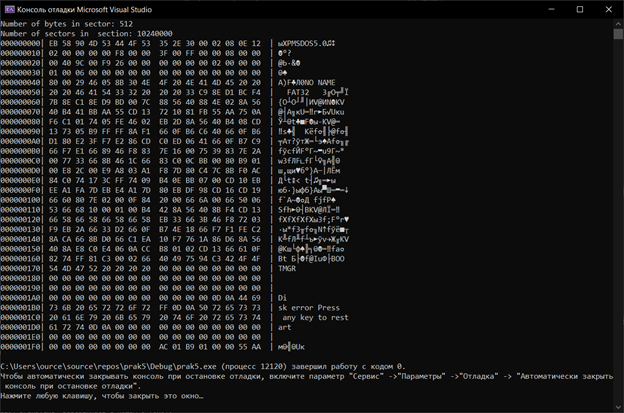


Рис. 1 – Вывод программы.

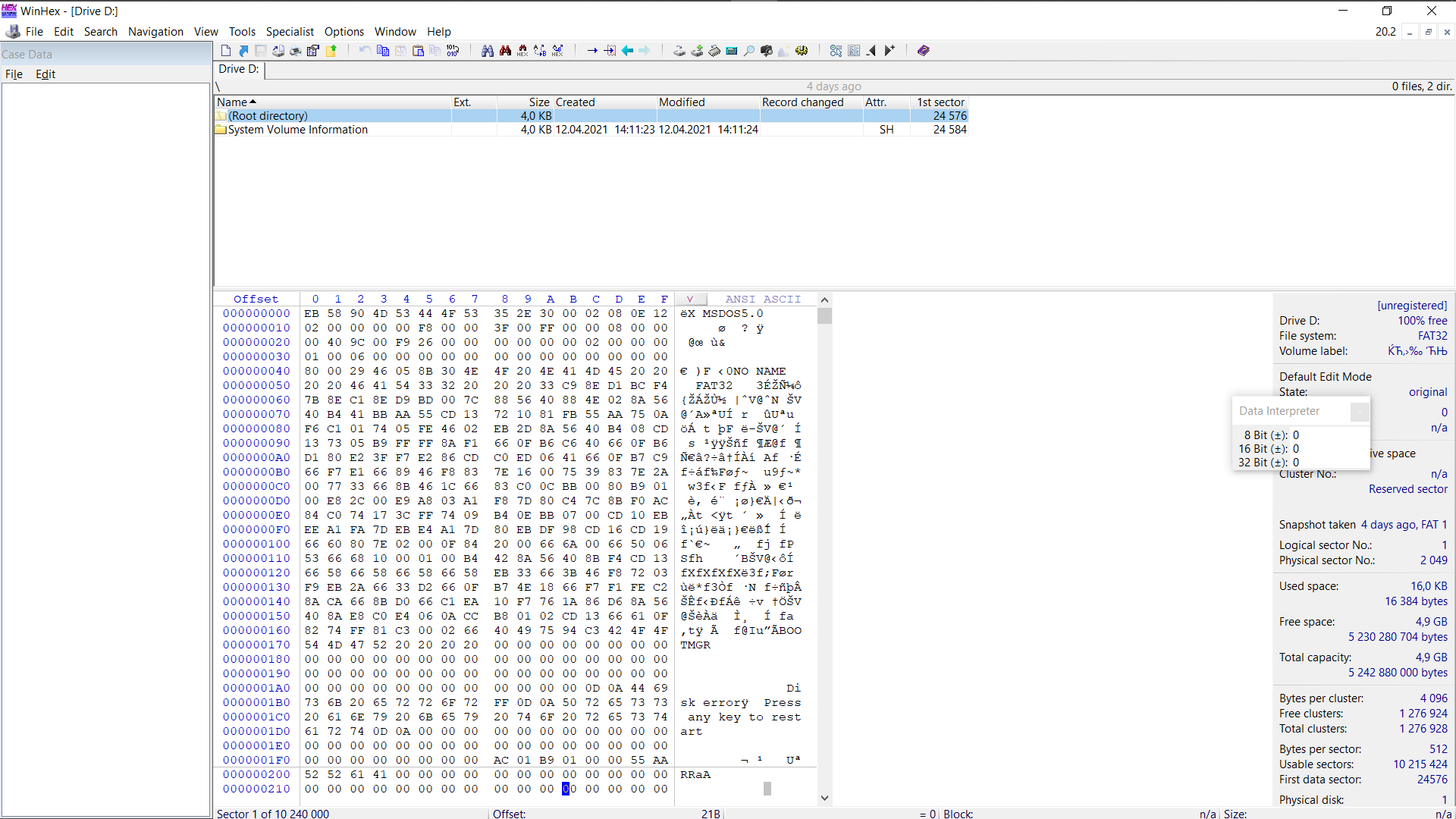


Рис.2 – Сектор в программе WinHex.

**Вывод**

В ходе практической работы мы познакомились с файловой структурой FAT и научились с ней работать.

**Список использованных источников**

1. Пошаговое руководство. Создание классического оконного приложение [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/cpp/windows/walkthrough-creating-windows-desktop-applications-cpp?view=msvc-160
2. Администрирование в Windows 10. [Электронный ресурс] : [https://docs.microsoft.com/ru-ru/windows/client-management/administrative-tools-in-windows-10 /](https://docs.microsoft.com/ru-ru/windows/client-management/administrative-tools-in-windows-10%20/) Ричард Ворд 2020 — 100 с.
3. «Тайм-менеджмент для системных администраторов». / Лимончелли Т.А. М.: Символ-плюс 2007 — 247 с. ISBN:978-5-93286-090-8