Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Ульяновский государственный технический университет» Кафедра «Вычислительная техника»

Дисциплина «Автоматизация проектирования микропроцессорных систем»

Лабораторная работа №1

Вариант № 5

Выполнил студент группы ИВТАПбд-41:

Галацков И.А.

Проверил:

Игонин А.Г.

Ульяновск, 2024

**Содержание**

[Постановка задачи 3](#_Toc185514970)

[Краткие теоретические сведения 3](#_Toc185514971)

[Ход выполнения работы 4](#_Toc185514972)

[Тестирование 5](#_Toc185514973)

[Вывод 6](#_Toc185514974)

[Список используемой литературы 7](#_Toc185514975)

[Приложение 8](#_Toc185514976)

# **Постановка задачи**

Требуется написать программу на языке С, которая реализует подключение к устройству и считывает с него данные. Полученные данные необходимо представить в формате удобном для восприятия.

Устройство: жесткий диск.

# **Краткие теоретические сведения**

Master Boot Record (MBR) — это первый сектор (512 байт) жесткого диска или другого носителя, который содержит важную информацию для начальной загрузки системы и организации разделов диска. В MBR располагаются три ключевые части: код загрузчика (первые 446 байт), таблица разделов (64 байта), и подпись MBR (последние 2 байта, содержащие значение 0x55AA). Код загрузчика отвечает за начальную загрузку операционной системы, таблица разделов хранит информацию о разделах диска, а подпись служит маркером, удостоверяющим, что это действительно MBR.

Таблица разделов состоит из четырех записей по 16 байт каждая, что позволяет MBR описывать до четырех разделов. Каждая запись содержит такие данные, как статус раздела (активный или неактивный), тип раздела, начальный и конечный адреса (в устаревшем формате CHS и современном формате LBA), а также размер раздела в секторах. CHS (Cylinder-Head-Sector) — это устаревший формат адресации, поддерживающий диски размером до 8 ГБ, тогда как LBA (Logical Block Addressing) — это более современный метод, который используется для работы с большими объемами данных.

# **Ход выполнения работы**

Для начала необходимо подключится к диску на устройстве. Подключение к диску осуществляется с использованием функции Windows API CreateFile. Она открывает дескриптор для доступа к физическому устройству, представляющему диск.

Функция CreateFile используется для открытия физического устройства, в данном случае диска (листинг 1).

Листинг 1. Подключение к устройству

|  |
| --- |
| HANDLE hDisk = CreateFile(  disk\_path, // Имя устройства (например, "\\\\.\\PhysicalDrive0")  GENERIC\_READ, // Доступ для чтения  FILE\_SHARE\_READ | FILE\_SHARE\_WRITE, // Разрешить совместное использование  NULL, // Защита по умолчанию  OPEN\_EXISTING, // Открыть существующее устройство  0, // Флаги  NULL // Нет шаблона  ); |

Для доступа к первому физическому диску в системе Windows используется строка "\\\\.\\PhysicalDrive0, такой путь позволяет обращаться к устройству напрямую. GENERIC\_READ указывает, что открываемый дескриптор предоставляет только доступ для чтения. Это нужно, чтобы предотвратить случайное изменение данных на диске. FILE\_SHARE\_READ | FILE\_SHARE\_WRITE необходимо, чтобы другие процессы могут читать и записывать на устройство, даже если оно открыто текущей программой.

Далее после вызова CreateFile проверяется, успешно ли открылось устройство. Если дескриптор равен INVALID\_HANDLE\_VALUE, значит, подключение к устройству не удалось (листинг 2).

Листинг 2. Проверка подключения

|  |
| --- |
| if (hDisk == INVALID\_HANDLE\_VALUE) {  printf("Ошибка открытия устройства: %lu\n", GetLastError());  return;  } |

После успешного открытия устройства с помощью CreateFile, программа может использовать дескриптор (hDisk) для выполнения операций чтения. Это делается с помощью функции ReadFile, которая считывает данные с устройства (листинг 3).

Листинг 3. Доступ к данным

|  |
| --- |
| ReadFile(hDisk, buffer, BUFFER\_SIZE, &bytesRead, NULL) |

После завершения работы с устройством дескриптор закрывается с помощью функции CloseHandle, чтобы освободить ресурсы.

Таком образом, выполняется подключение к диску, а также чтение данных. Вышеописанный подход обеспечивает прямой доступ к устройству, что полезно для низкоуровневых операций, таких как чтение MBR. Однако такие операции требуют административных привилегий и осторожности, чтобы избежать повреждения данных.

# **Тестирование**

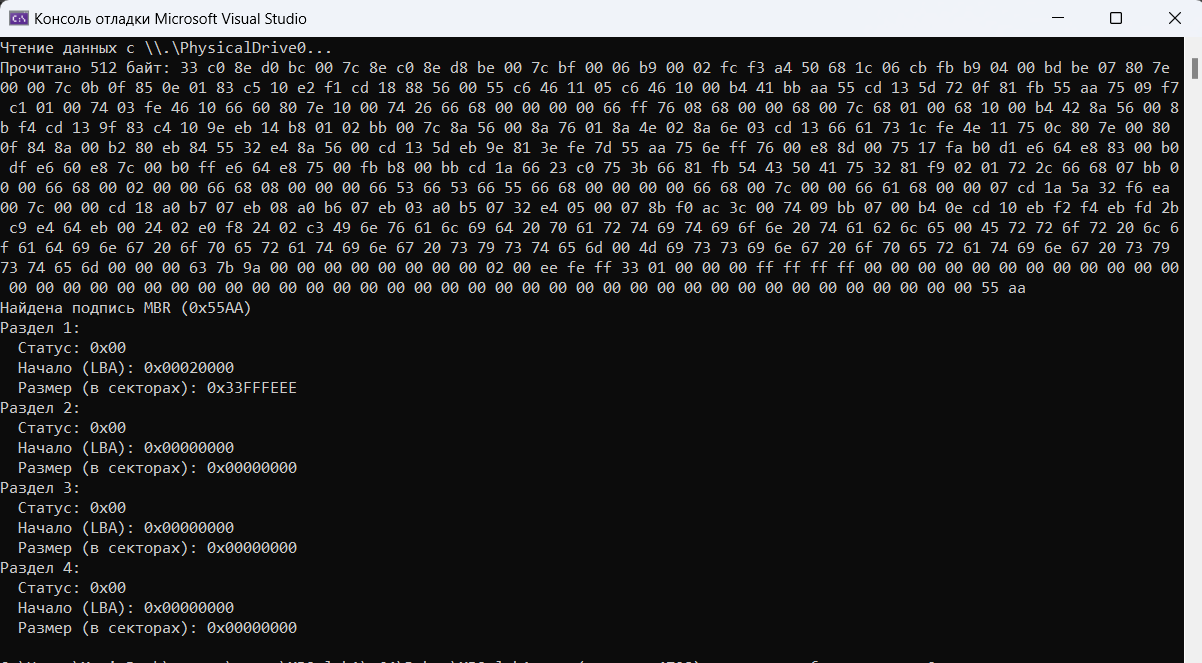
Программа считывает первый сектор диска MRB, и выводит прочитанные биты в шестнадцатеричной системе счисления, а также выводит информацию о таблице разделов: статус, начальный сектор и размер.

Рис. 1. Вывод программы

# **Вывод**

В ходе выполнения данной лабораторной работы была написана программа, реализующая подключение к жесткому диску и считывающая с него данные, а именно сектор MBR.

# **Список используемой литературы**

1. Электронный ресурс – URL: <https://docs.altlinux.org/ru-RU/archive/4.1/html-single/school-server/hd_structure/index.html>
2. Электронный ресерс – URL: <https://habr.com/ru/articles/347002/>

# **Приложение**

#include <windows.h>

#include <stdio.h>

#include <stdint.h>

#include <locale.h>

#define BUFFER\_SIZE 512 // Размер блока для чтения

wchar\_t\* char\_to\_wchar(const char\* str) {

size\_t len = strlen(str) + 1;

wchar\_t\* wstr = (wchar\_t\*)malloc(len \* sizeof(wchar\_t));

size\_t conventered = 0;

errno\_t err = mbstowcs\_s(&conventered, wstr, len, str, \_TRUNCATE);

if (err != 0) {

perror("Ошибка преобразования строки");

free(wstr);

return NULL;

}

return wstr;

}

void read\_from\_disk(wchar\_t\* disk\_path, unsigned char\* data) {

HANDLE hDisk = CreateFile(

disk\_path, // Имя устройства (например, "\\\\.\\PhysicalDrive0")

GENERIC\_READ, // Доступ для чтения

FILE\_SHARE\_READ | FILE\_SHARE\_WRITE, // Разрешить совместное использование

NULL, // Защита по умолчанию

OPEN\_EXISTING, // Открыть существующее устройство

0, // Флаги

NULL // Нет шаблона

);

if (hDisk == INVALID\_HANDLE\_VALUE) {

printf("Ошибка открытия устройства: %lu\n", GetLastError());

return;

}

unsigned char buffer[BUFFER\_SIZE];

DWORD bytesRead;

size\_t byte\_r = 0;

if (ReadFile(hDisk, buffer, BUFFER\_SIZE, &bytesRead, NULL)) {

printf("Прочитано %lu байт: ", bytesRead);

for (DWORD i = 0; i < bytesRead; ++i) {

data[byte\_r++] = buffer[i];

printf("%02x ", buffer[i]);

}

printf("\n");

}

else {

printf("Ошибка чтения данных: %lu\n", GetLastError());

}

CloseHandle(hDisk);

}

void parse\_mbr(unsigned char\* data) {

// Проверка на подпись MBR (0x55AA)

if (data[510] == 0x55 && data[511] == 0xAA) {

printf("Найдена подпись MBR (0x55AA)\n");

// Извлечение таблицы разделов (64 байта)

for (int i = 0; i < 4; i++) {

unsigned char\* partition = &data[446 + i \* 16];

printf("Раздел %d:\n", i + 1);

printf(" Статус: 0x%02X\n", partition[0]);

printf(" Начало (LBA): 0x%02X%02X%02X%02X\n",

partition[3], partition[2], partition[1], partition[0]);

printf(" Размер (в секторах): 0x%02X%02X%02X%02X\n",

partition[7], partition[6], partition[5], partition[4]);

}

}

else {

printf("Подпись MBR не найдена.\n");

}

}

int main() {

unsigned char data[512];

setlocale(LC\_ALL, "Russian");

const char\* disk\_path = "\\\\.\\PhysicalDrive0"; // Диск 0

wchar\_t\* wpath = char\_to\_wchar(disk\_path);

printf("Чтение данных с %s...\n", disk\_path);

read\_from\_disk(wpath, data);

parse\_mbr(data);

return 0;

}