Минобрнауки России  
Федеральное государственное автономное образовательное  
Учреждение высшего образования  
«Санкт-Петербургский государственный электротехнический  
Университет им. В.И. Ульянова (Ленина)»  
(СПГЭТУ «ЛЭТИ»)  
Факультет компьютерных технологий и информатики

Кафедра вычислительной техники  
  
  
  
**Отчет по лабораторной работе №1**

**на тему:**

**«Управление файловой системой»**

по дисциплине «Операционные системы»

Выполнил студент группы 9308: Дементьев Д.П.

Принял: к.т.н., доцент Тимофеев А.В.

Санкт-Петербург  
2021 г.

Оглавление

[1. Цель работы: 3](#_Toc58297106)

[2. Управление дисками, каталогами и файлами. 3](#_Toc58297107)

[2.1. Указания к выполнению. 3](#_Toc58297108)

[2.2. Результаты работы программы. 4](#_Toc58297109)

[3. Копирование файла с помощью операций перекрывающегося ввода-вывода 11](#_Toc58297110)

[3.1. Указания к выполнению 11](#_Toc58297111)

1. **Цель работы:**

Исследовать управление файловой системой с помощью Win32 API.

1. **Управление дисками, каталогами и файлами.**
   1. **Указания к выполнению.**

Создайте консольное приложение с меню (каждая выполняемая функция и/или операция должна быть доступна по отдельному пункту меню), которое выполняет:

* вывод списка дисков (функции Win32 API – GetLogicalDrives, GetLogicalDriveStrings);
* для одного из выбранных дисков вывод информации о диске и размер свободного пространства (функции Win32 API – GetDriveType, GetVolumeInformation, GetDiskFreeSpace);
* создание и удаление заданных каталогов (функции Win32 API CreateDirectory, RemoveDirectory);
* создание файлов в новых каталогах (функция Win32 API – CreateFile)
* копирование и перемещение файлов между каталогами с возможностью выявления попытки работы с файлами, имеющими совпадающие имена (функции Win32 API – CopyFile, MoveFile, MoveFileEx);
* анализ и изменение атрибутов файлов (функции Win32 API – GetFileAttributes, SetFileAttributes, GetFileInformationByHandle, GetFileTime, SetFileTime).

Ссылка на Github: <https://github.com/CamelNotFemale/5_sem_OS/blob/main/lab1/part1.cpp>

* 1. **Результаты работы программы.**

1. Меню с пунктами, необходимыми в процессе выполнения программы:

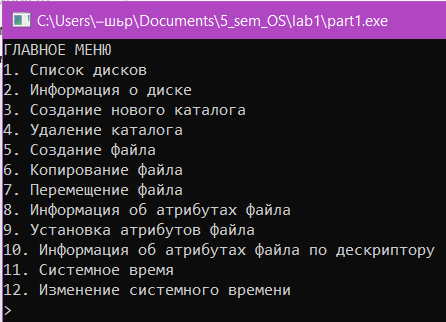


Рисунок . Главное меню

1. Вывод списка дисков (функция GetLogicalDrives()):

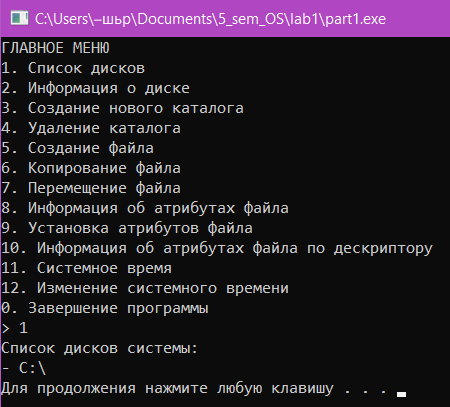


Рисунок 2. Список дисков

1. Вывод информации о дисках и размере свободного пространства (функция GetVolumeInformation()), получаемая переменная lpFileSystemFlags декодируется путём конъюнкции с определёнными атрибутами:

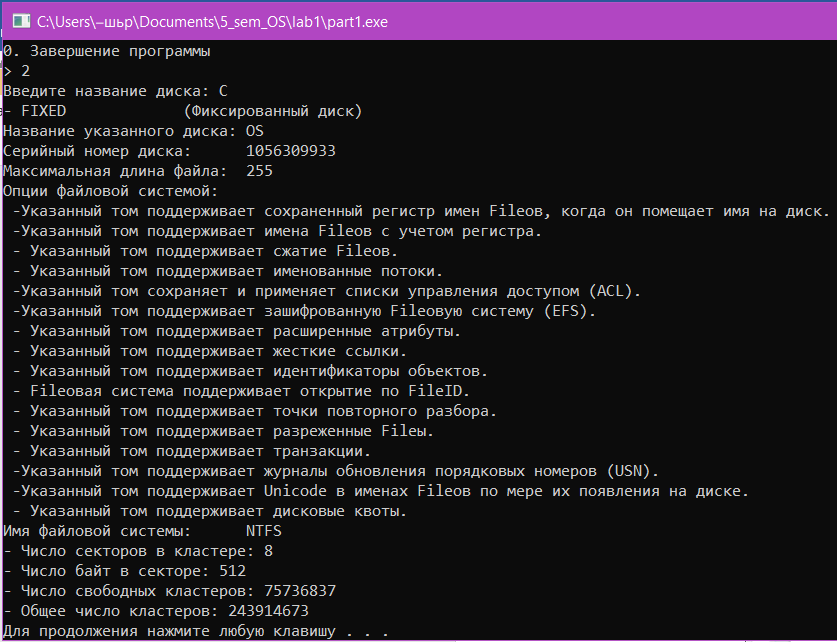


Рисунок 3. Вывод информации о дисках

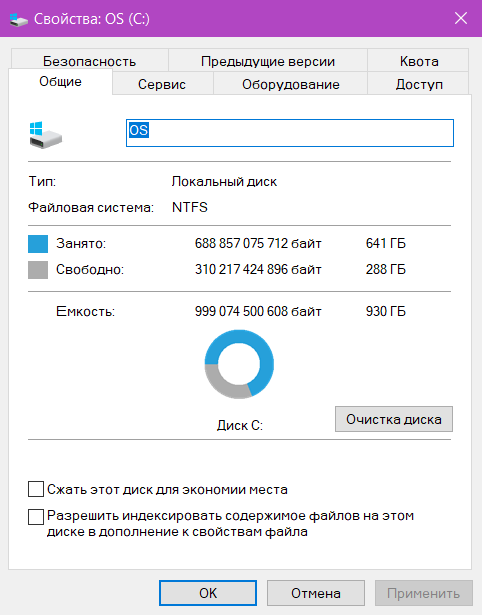


Рисунок 4. Свойства диска

1. Создание каталога (функция CreateDirectory()):

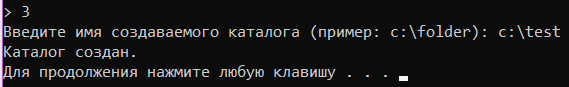


Рисунок 5. Создание каталога

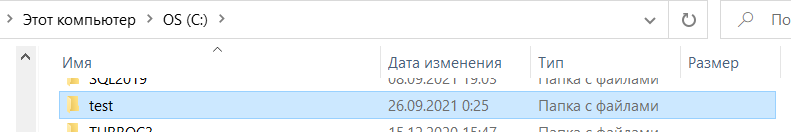


Рисунок 6. Результат создания

1. Удаление каталога (функция RemoveDirectory()):

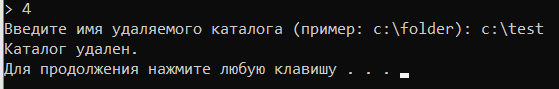


Рисунок 7. Удаление каталога

1. Создание файла (функция CreateFile()):

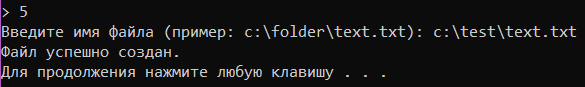


Рисунок 8. Создание файла

1. Копирование файла (функция CopyFile()):

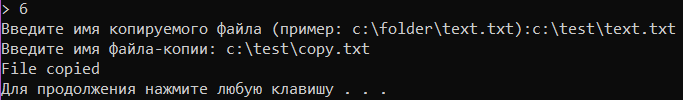


Рисунок 92. Копирование файла.

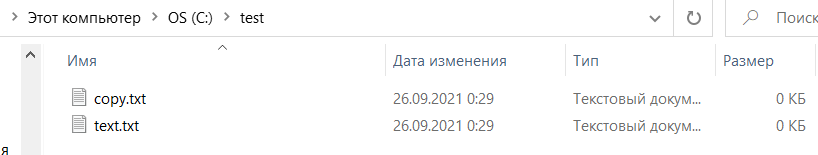


Рисунок 103. Результат копирования файла.

1. Перемещение файла (функция MoveFile()):

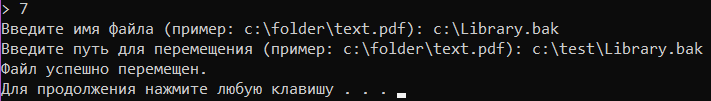


Рисунок 11. Перемещение файла

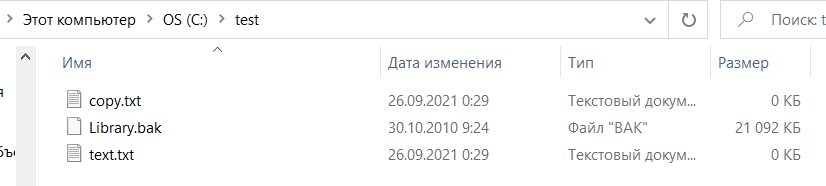


Рисунок 124. Результат перемещения файла.

1. Перемещение файла с проверкой (функция MoveFileEx()), используется флаг MOVEFILE\_REPLACE\_EXISTING:

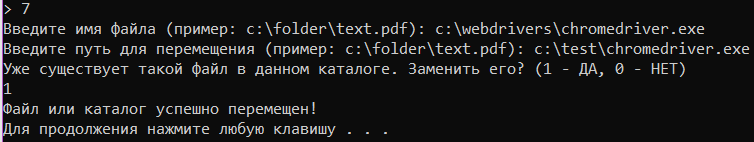


Рисунок 13. Перемещение файла (с проверкой)

1. Получение атрибутов файла (функция GetFileAttributes()):

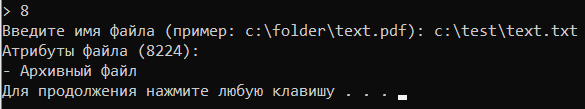


Рисунок 14. Получение атрибутов файла

1. Установка атрибутов файла (функция SetFileAttributes()):

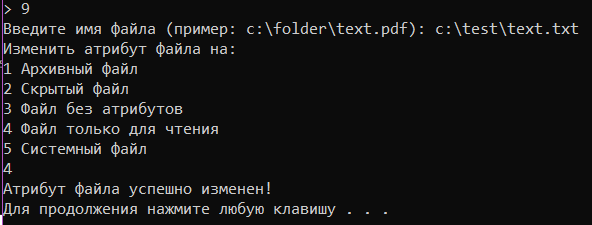


Рисунок 15. Установка атрибутов файла

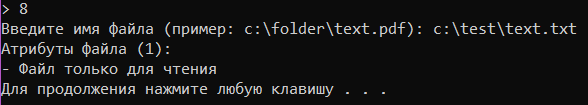


Рисунок 16. Проверка установки атрибутов файла

1. Вывод информации о файле (функция GetFileInformationByHandle()), получаемая переменная декодируется путём конъюнкции с определёнными атрибутами:

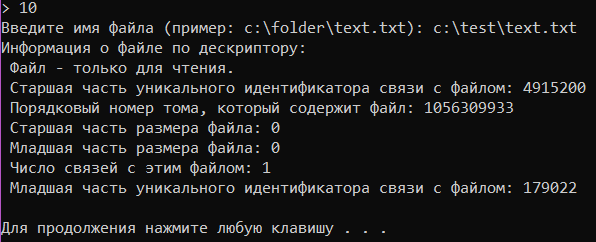


Рисунок 17. Вывод информации о файле

1. Вывод временных характеристик файла (функция GetFileTime()):

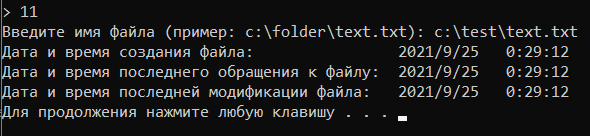


Рисунок 185. Вывод временных характеристик

1. Изменение временных характеристик файла (функция SetFileTime()), устанавливается текущее системное время:

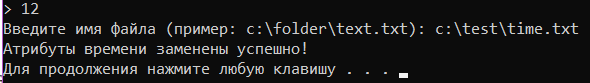


Рисунок 19. Изменение временных характеристик

* 1. **Вывод**

В ходе выполнения первого задания были изучены механизмы работы с файловой системой с помощью Win32 API. Использованные функции позволяют получать информацию о носителях и файлах, манипулировать файлами и директориями, а также получать доступ к атрибутам файловой системы.

1. **Копирование файла с помощью операций перекрывающегося ввода-вывода**
   1. **Указания к выполнению**

Создайте консольное приложение, которое выполняет:

* открытие/создание файлов (функция Win32 API – CreateFile, обязательно использовать флаги FILE\_FLAG\_NO\_BUFFERING и FILE\_FLAG\_OVERLAPPED);
* файловый ввод-вывод (функции Win32 API – ReadFileEx, WriteFileEx) блоками кратными размеру кластера;
* ожидание срабатывания вызова функции завершения (функция Win32 API – SleepEx);
* измерение продолжительности выполнения операции копирования файла (функция Win32 API – TimeGetTime).

Ссылка на Github: <https://github.com/CamelNotFemale/5_sem_OS/blob/main/lab1/part2.cpp>

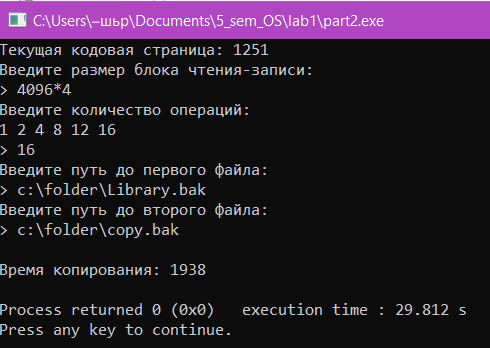
* 1. **Результаты выполнения программы**

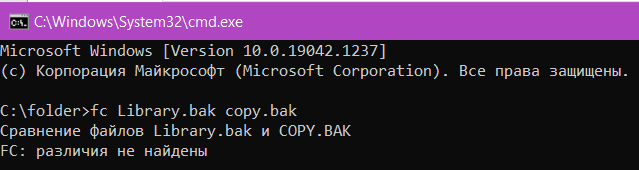
1. **Проверка работоспособности программы**

Для проверки:

* был скопирован файл объёмом ;
* был выбран размер блока – ;
* были использованы 16 перекрывающих операций.

Результаты проверки приведены ниже:



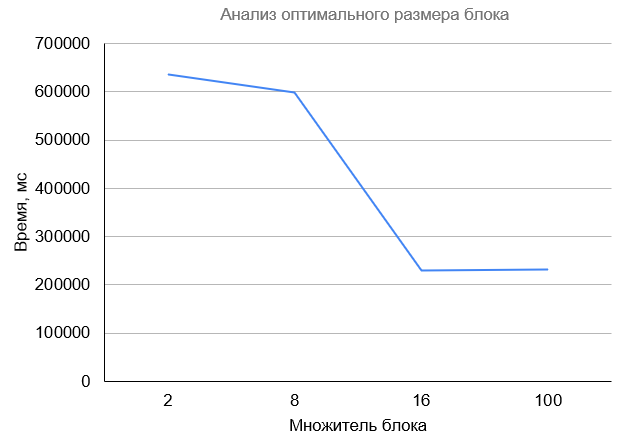


Консольная команда FC не нашла различий между исходным и скопированным файлом, что означает, что программа работает корректно. Каждая последующая итерация запуска программы будет проверена таким же образом.

1. **Анализ оптимального размера блока**

Для того, чтобы найти оптимальный размер блока, было использовано 8 перекрывающихся операций, размер блока увеличивался с каждым запуском программы.

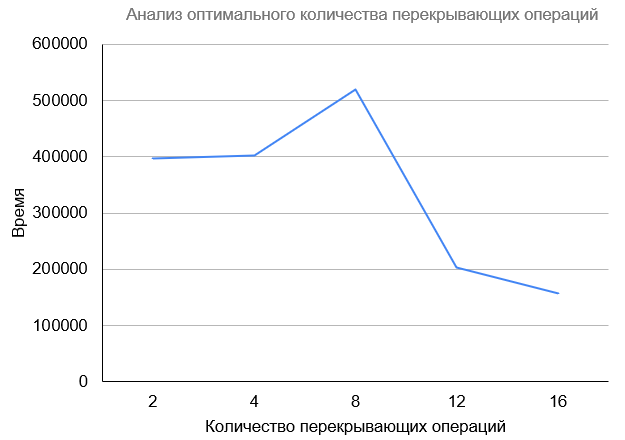
Для проверки был выбран файл Office.iso размером :



Из графика видно, что размер блока обратно пропорционален времени копирования: чем больше размер блока, тем меньше время копирования. Зависимость близка к 1/log(x). Вероятно, оптимальным будет размер блока равный размеру файла, разделенному на количество перекрывающих операций.

1. **Анализ оптимального количества перекрывающих операций**

Для поиска оптимального количества перекрывающих операций возьмём размер блока, равный :



Судя по данному графику, можно сказать, что 16 перекрывающих операций позволяет достичь большей скорости, с дальнейшим увеличением их количества время только увеличивается. Вероятно, оптимальное количество зависит от типа и модели носителя, а также от количества ядер процессора.

* 1. **Вывод**

В ходе выполнения данного задания был изучен механизм асинхронного копирования файла. При этом использовалось переменное количество блоков, участвующих в копировании, и переменное количество перекрывающих операций ввода/вывода.

В процессе выполнения получены результаты, исходя из которых можно сделать выводы:

* изменение параметров заметно сказывается на скорости копирования;
* при асинхронном копировании при увеличении размера копируемого блока увеличивается скорость копирования файла;
* оптимальное количество операций перекрывающего ввода/вывода – 16.