Минобрнауки России  
Федеральное государственное автономное образовательное  
Учреждение высшего образования  
«Санкт-Петербургский государственный электротехнический  
Университет им. В.И. Ульянова (Ленина)»  
(СПбГЭТУ «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: Яловега Н.В.

Принял: к.т.н., доцент Тимофеев А.В.

Санкт-Петербург  
2021 г.

Оглавление

[1. Цель работы: 3](#__RefHeading___Toc348_160529129)

[2. Управление дисками, каталогами и файлами. 3](#__RefHeading___Toc350_160529129)

[2.1. Указания к выполнению. 3](#__RefHeading___Toc352_160529129)

[2.2. Результаты работы программы. 4](#__RefHeading___Toc354_160529129)

[3. Копирование файла с помощью перекрывающегося ввода-вывода 11](#__RefHeading___Toc356_160529129)

[3.1. Указания к выполнению 11](#__RefHeading___Toc358_160529129)

1. **Цель работы:**

Исследовать управление файловой системой с помощью Win32 API.

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

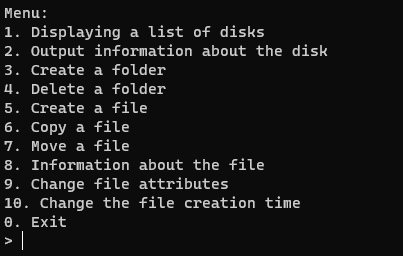
Создайте консольное приложение с меню (каждая выполняемая функция и/или операция должна быть доступна по отдельному пункту меню), которое выполняет:

* 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).

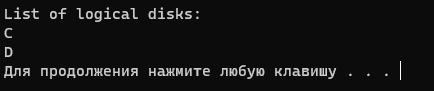
Ссылка на репозиторий с исходным кодом первой части работы: https://github.com/kivyfreakt/eltech/tree/master/os/lab1/task1

* 1. **Результаты работы программы.**

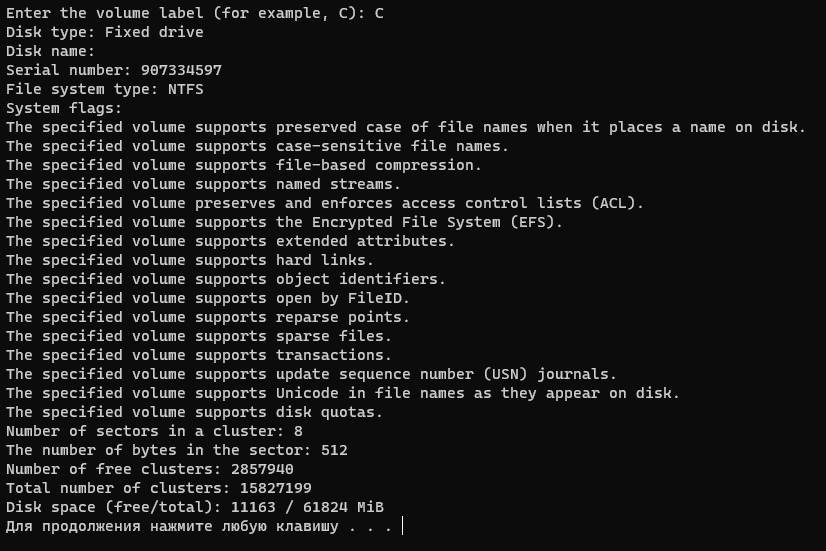
1. Меню с пунктами, необходимыми в процессе выполнения программы:

Рисунок 1: Главное меню

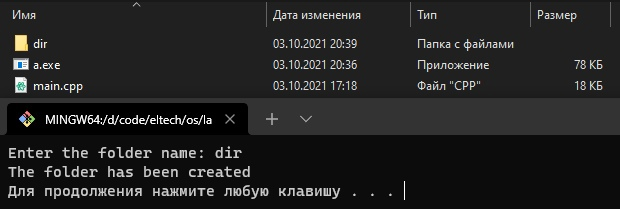
1. Вывод списка дисков (функция GetLogicalDrives()):

Рисунок 2: Список дисков

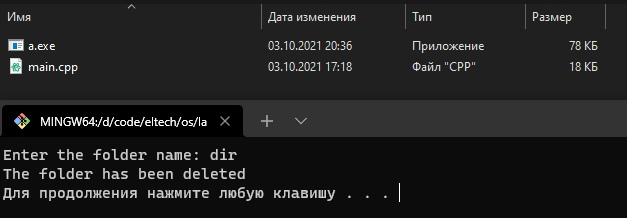
1. Вывод информации о дисках и размере свободного пространства (функция GetVolumeInformation()), получаемая переменная lpFileSystemFlags декодируется путём конъюнкции с определёнными атрибутами:

 Рисунок 3: Свойства диска

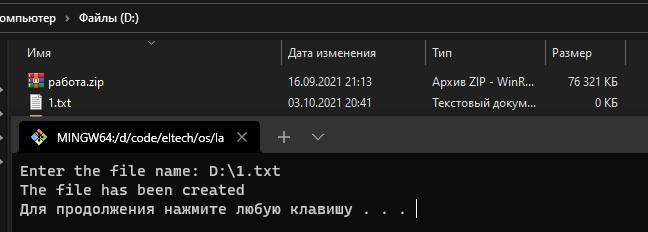
1. Создание каталога (функция CreateDirectory()):

Рисунок 4: Создание каталога

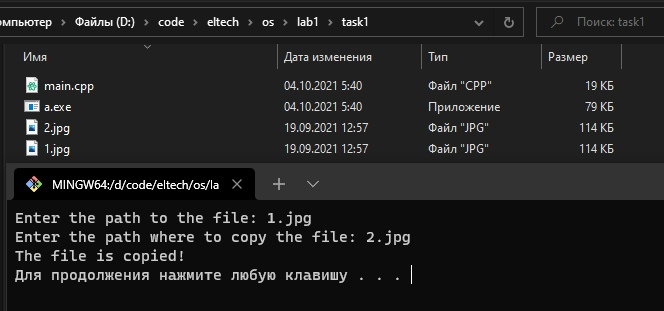
1. Удаление каталога (функция RemoveDirectory()):

Рисунок 5: Удаление каталога

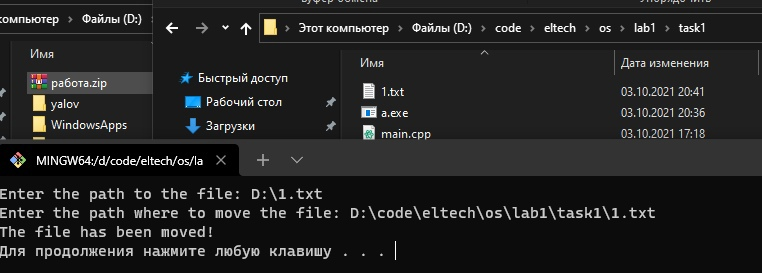
1. Создание файла (функция CreateFile()):

Рисунок 6: Создание файла

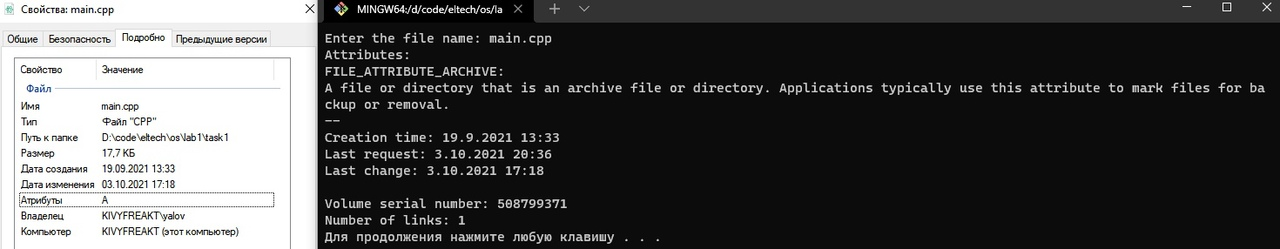
1. Копирование файла (функция CopyFile()):

Рисунок 7: Копирование файла

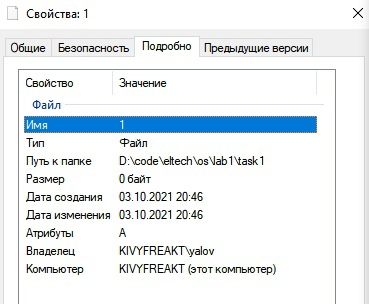
1. Перемещение файла (функция MoveFile()):

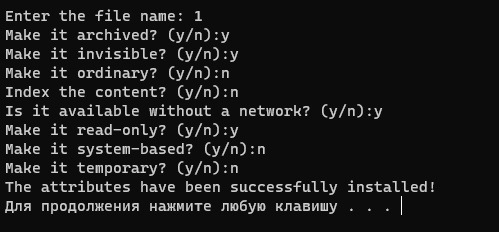
Рисунок 8: Перемещение файла

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

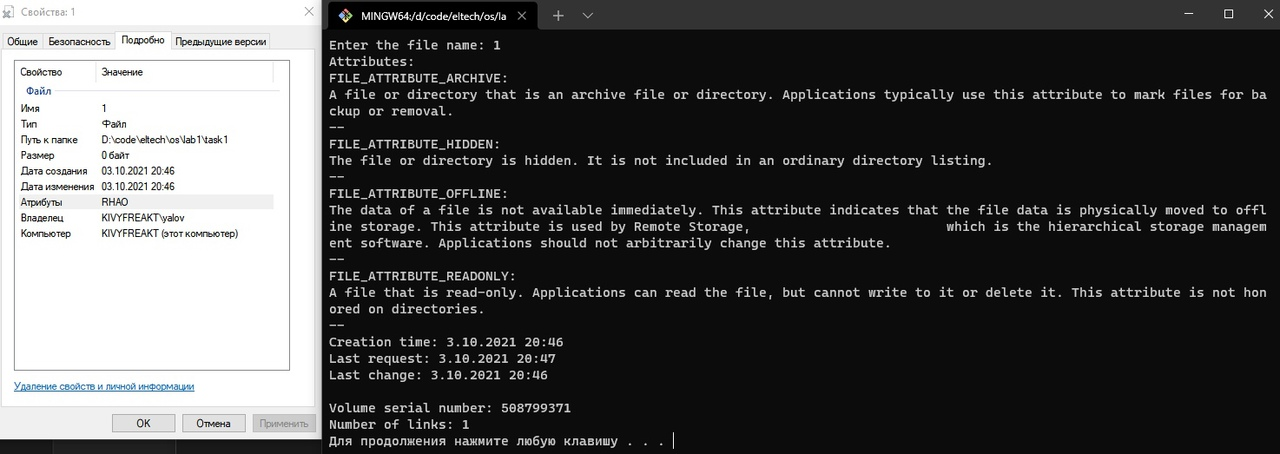
Рисунок 9: Получение аттрибутов файла

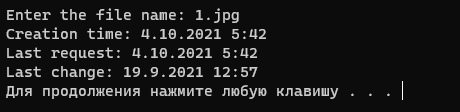
1. Установка атрибутов файла (функция SetFileAttributes()):

Рисунок 10: Атрибуты до

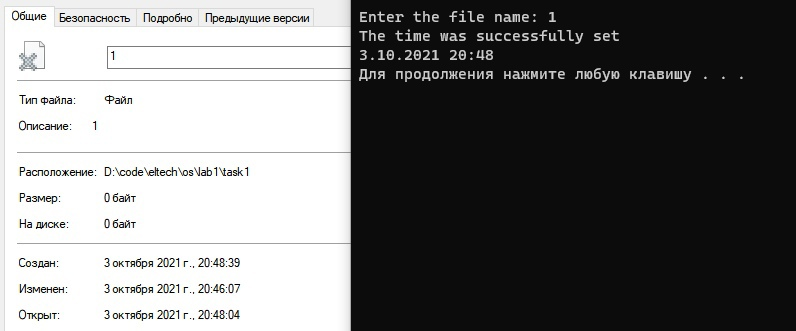
Рисунок 11: Установка атрибутов

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

Рисунок 12: Проверка атрибутов

Рисунок 13: Вывод времени

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

Рисунок 14: Изменение времени

* 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 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 репозиторий с исходным кодом второй части работы: https://github.com/kivyfreakt/eltech/tree/master/os/lab1/task2

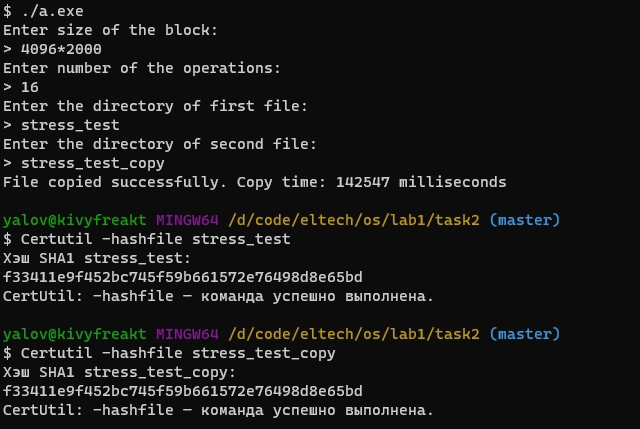
* 1. **Результаты выполнения программы**

1. **Проверка работоспособности программы**

Для проверки:

* был скопирован файл объёмом 5.8 ГБ;
* был выбран размер блока – 2000;
* были использованы 16 перекрывающих операций.

Результаты проверки приведены ниже:

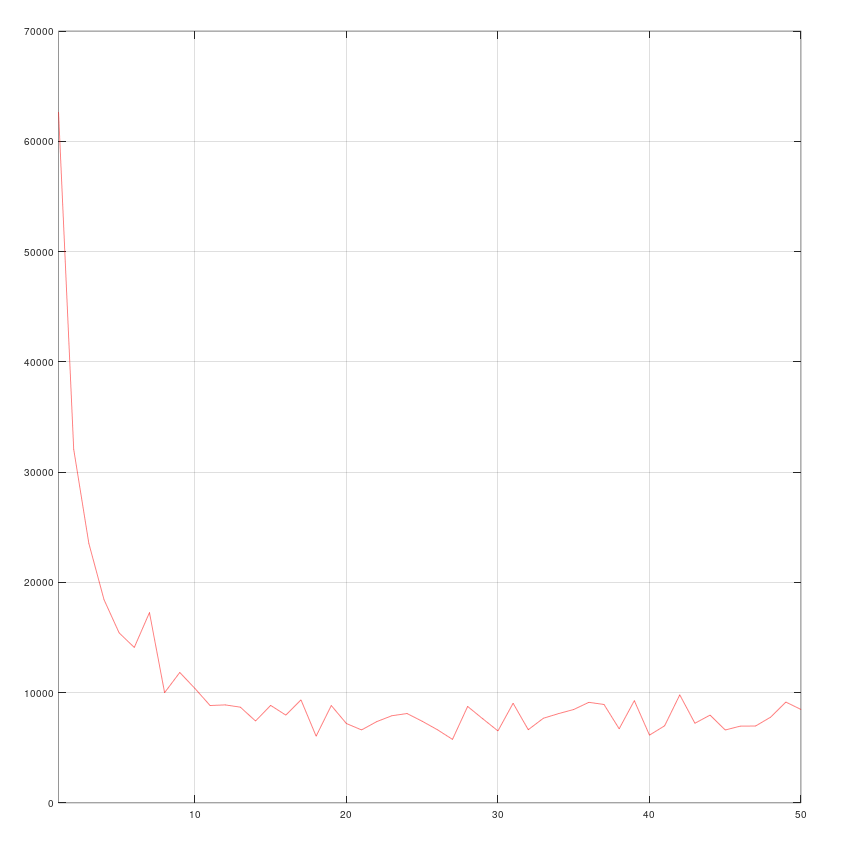
Рисунок 15: Проверка работоспособности

Для проверки идентичности файлов сравнили их хеши. Отличий между хешами исходного и скопированного файла нет, что означает, что программа работает корректно.

1. **Анализ оптимального размера блока**

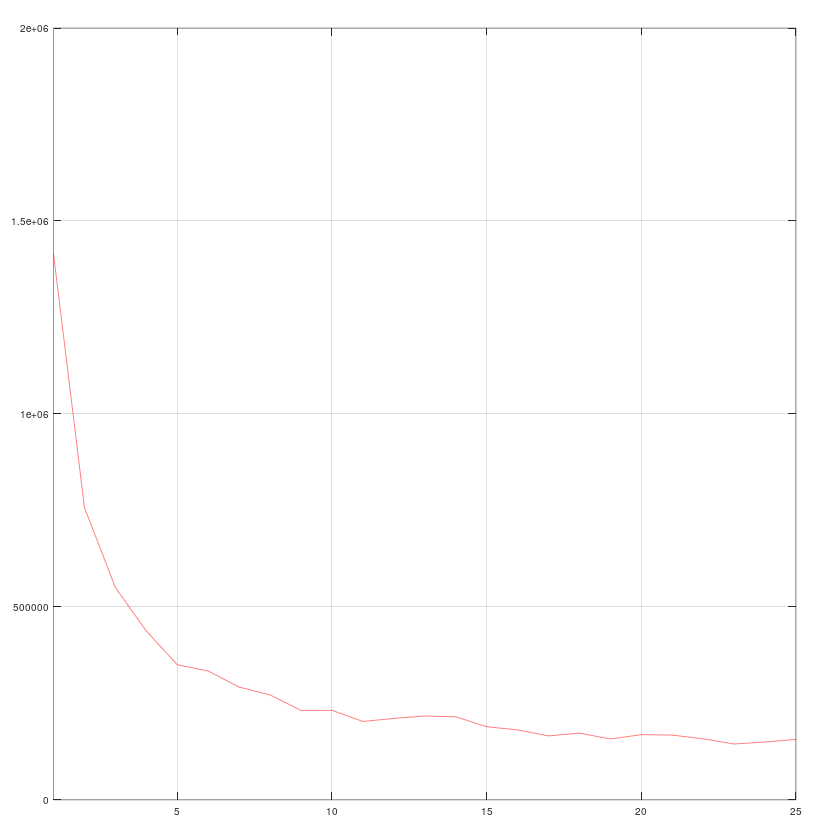
Для того, чтобы найти оптимальный размер блока, было использовано 1 перекрывающаяся операция, размер блока увеличивался с каждым запуском программы.

Для проверки было выбрано два файла размером 220МБ и 2.9 ГБ.

 Рисунок 16: График для файла размером 220МБ.

Из графика видно, что размер блока обратно пропорционален времени копирования: чем больше размер блока, тем меньше время копирования. Зависимость логарифмическая 1/log(x). Минимальное время было достигнуто при значении множителя 27.

Из графика видно, что размер блока обратно пропорционален времени копирования: чем больше размер блока, тем меньше время копирования. Зависимость логарифмическая 1/log(x). Минимальное время было достигнуто при значении множителя 23.

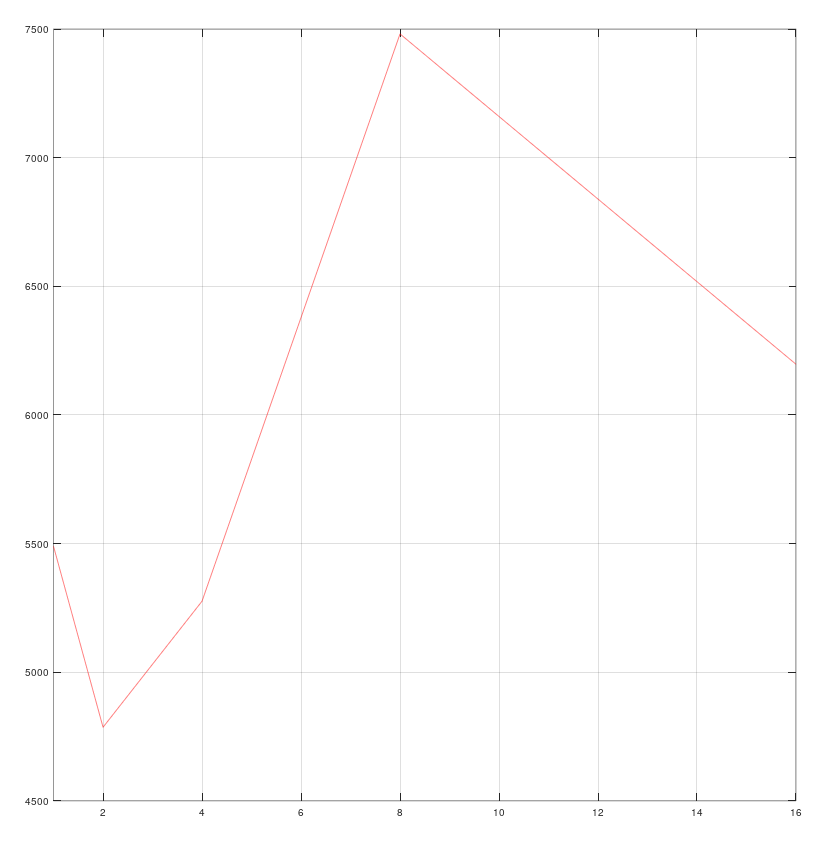
Рисунок 17: График для файла размером 2.9 ГБ

На обоих графиках можно заметить, что в какой-то момент времени время операции копирования уже будет слабо зависеть от зависеть от блока копируемых данных. В зависимости от ПК и используемого накопителя оптимальное количество операций будет разным. Также необходимо учитывать, что ПК во время замеров мог быть нагружен в неопределённый момент времени, так как полностью избавиться от посторонних процессов невозможно, и поэтому на графиках можно заметить "горки".

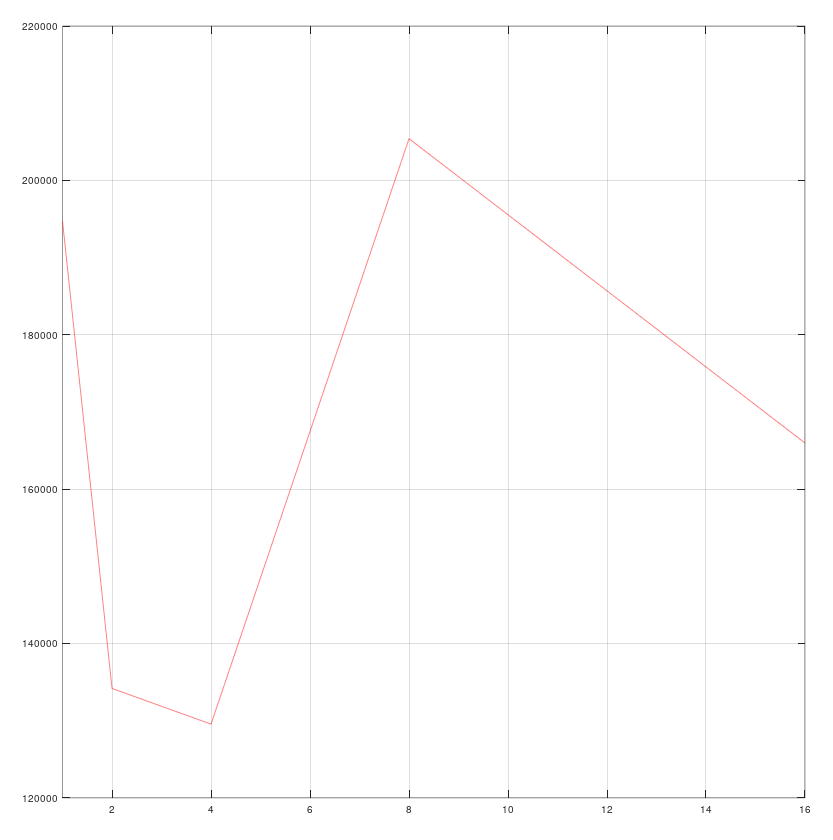
1. **Анализ оптимального количества перекрывающих операций**

Для поиска оптимального количества перекрывающих операций возьмём размер блока, равный 25.

Минимальное значение времени при величине количества операции равной 2.

Рисунок 18: График для файла размером 220МБ.

Минимальное значение времени при величине количества операции равной 4.

Рисунок 19: График для файла размером 2.9ГБ

В зависимости от ПК оптимальное количество операций в-в будет разным. Если задать их слишком много при относительно небольшом размере файла, то копирование будет производиться только медленнее. Также необходимо учесть, что ПК во время замеров мог быть нагружен в неопределённый момент времени, и поэтому на графике можно заметить "горки".

* 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.