Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Операционные системы

Студент: Немкович А.В.

ФИТ 3 курс 1 группа

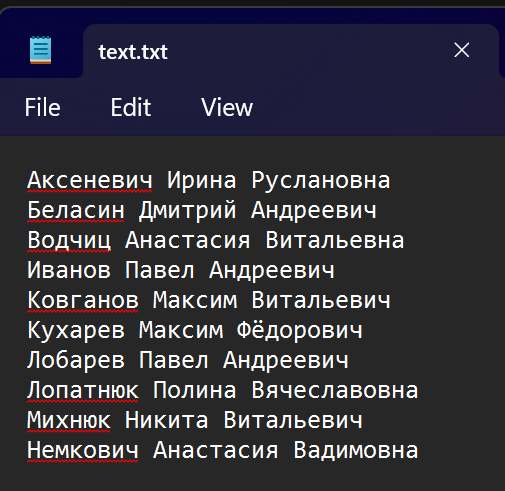
Преподаватель: Бернацкий П.В

Минск 2024

**Лабораторная работа №9**

**Задание 01.Windows**

С помощью Notepad (Notepad+) создайте на дисковом устройстве текстовый файл OS09\_01.txt. Заполните его 10 строками из списка студентов вашей подгруппы. Буквы кириллические.



Разработайте приложение OS09\_01. Приложение OS09\_01 вызывает функцию printFileInfo, имеющую следующий прототип.

Функция printFileInfo выводит в стандартный поток вывода следующую информацию:

- имя файла;

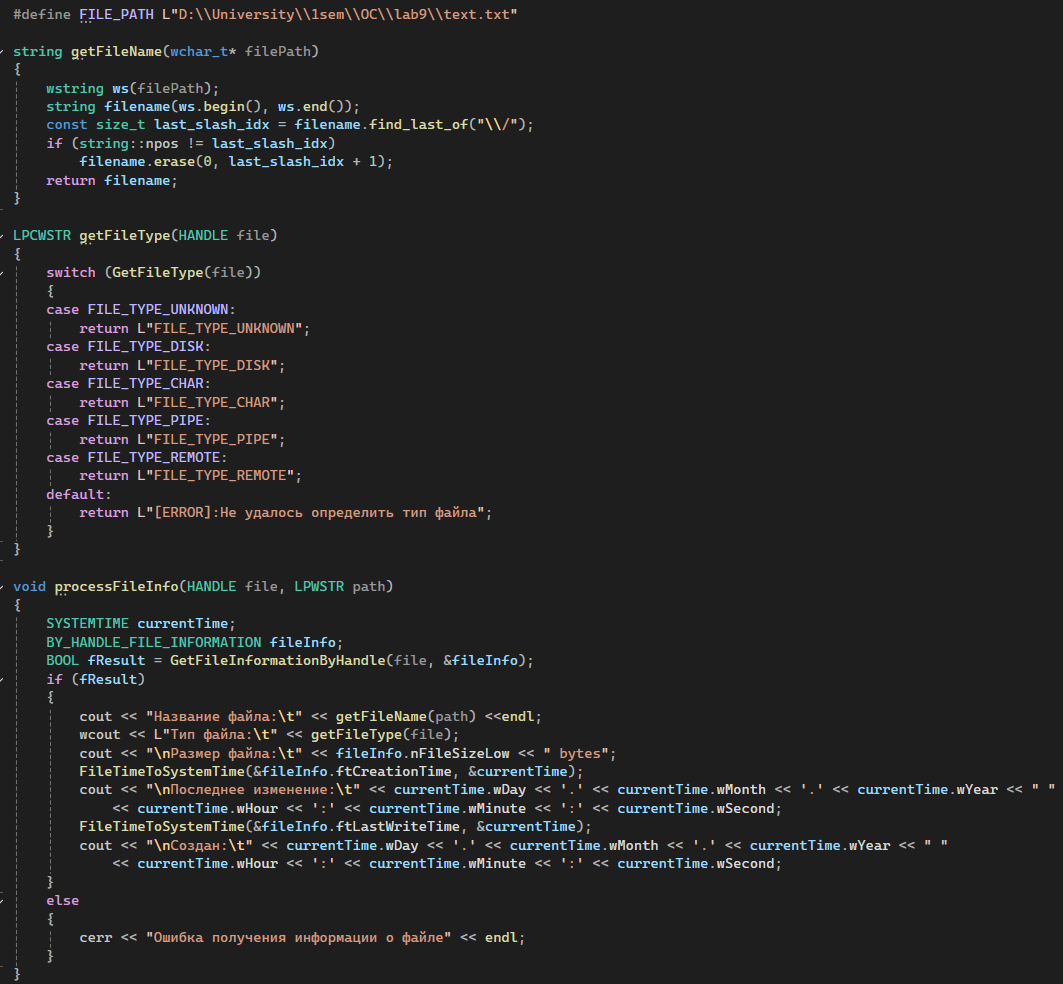
- тип файла;

- размер файла;

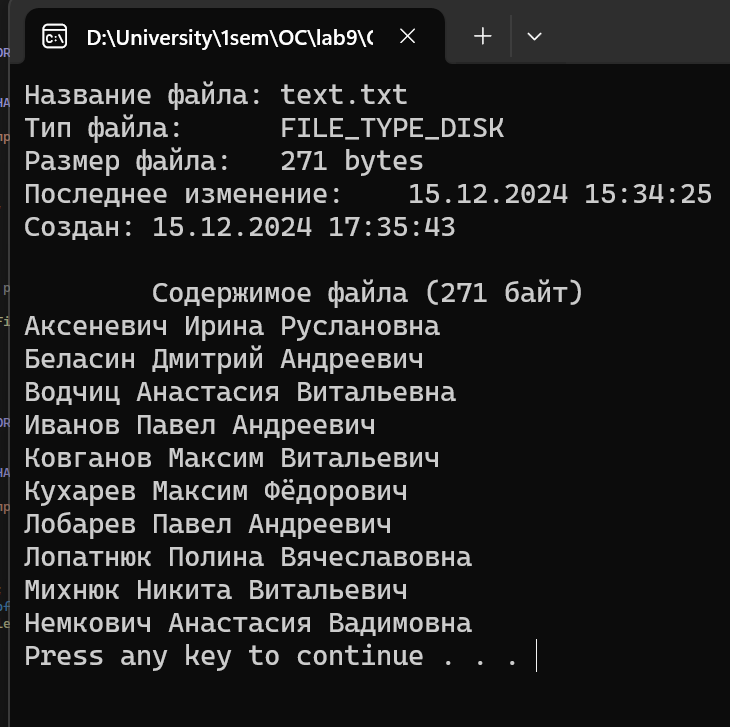
- дата и время создания файла;

- дата и время последнего обновления.

Приложение OS09\_01 вызывает функцию printFileTxt, имеющую следующий прототип. Функция printFileTxt выводит в стандартный поток вывода содержимое файла. При вызове функции укажите в качестве параметра имя текстового файла, созданного в п.2. Продемонстрируйте работоспособность приложения OS09\_01.



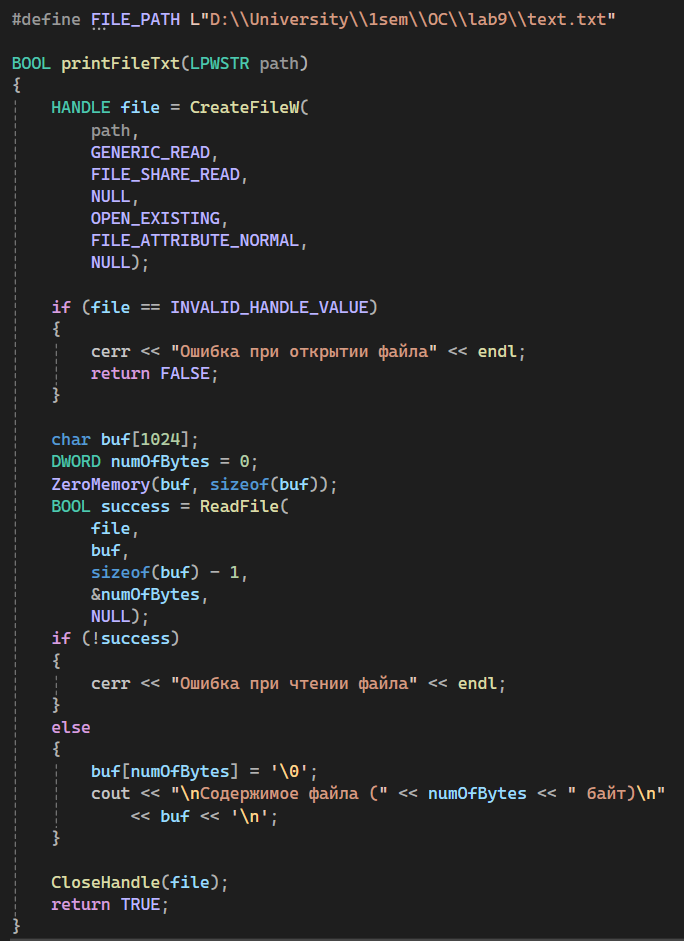


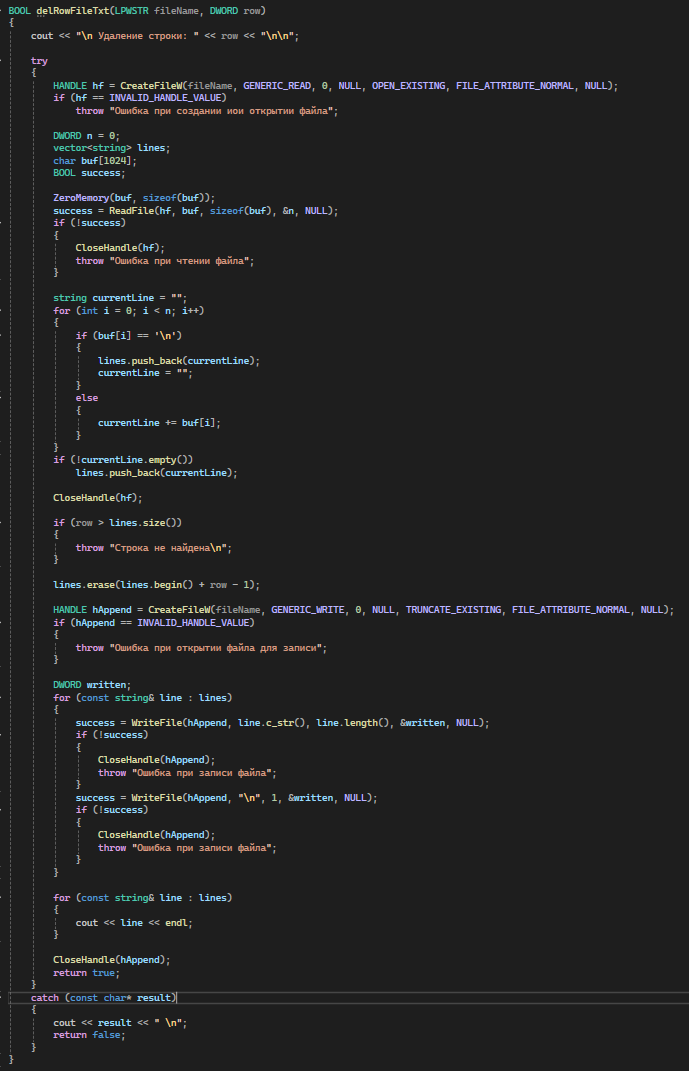


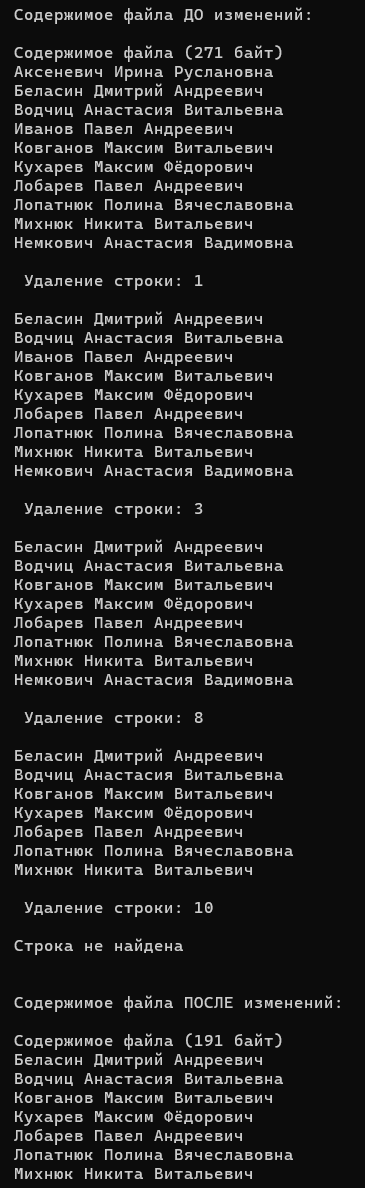
**Задание 02.Windows**

Разработайте приложение OS09\_02. Приложение OS09\_02 вызывает функцию delRowFileTxt, имеющую следующий прототип.

Функция применяется к файлу OS09\_01.txt (п.2) и вызывается последовательно 4 раза, с row = 1,3,8,10. Результат выполнения продемонстрируйте с помощью функции printFileTxt (п.6). Продемонстрируйте работоспособность приложения OS09\_02.







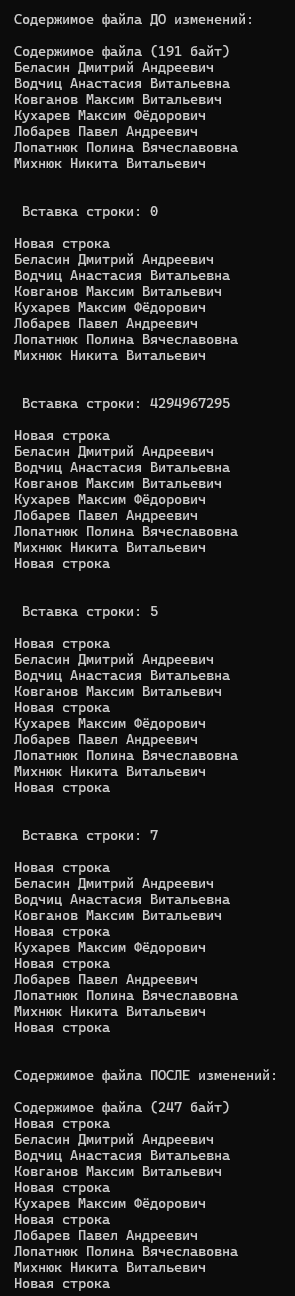
**Задание 03.Windows**

Разработайте приложение OS09\_03. Приложение OS09\_03 вызывает функцию insRowFileTxt, имеющую следующий прототип.



Функция применяется к файлу OS09\_01.txt (п.2) и вызывается последовательно 4 раза, с row = 0,-1,5,7. Результат выполнения продемонстрируйте с помощью функции printFileTxt (п.6). Продемонстрируйте работоспособность приложения OS09\_03.

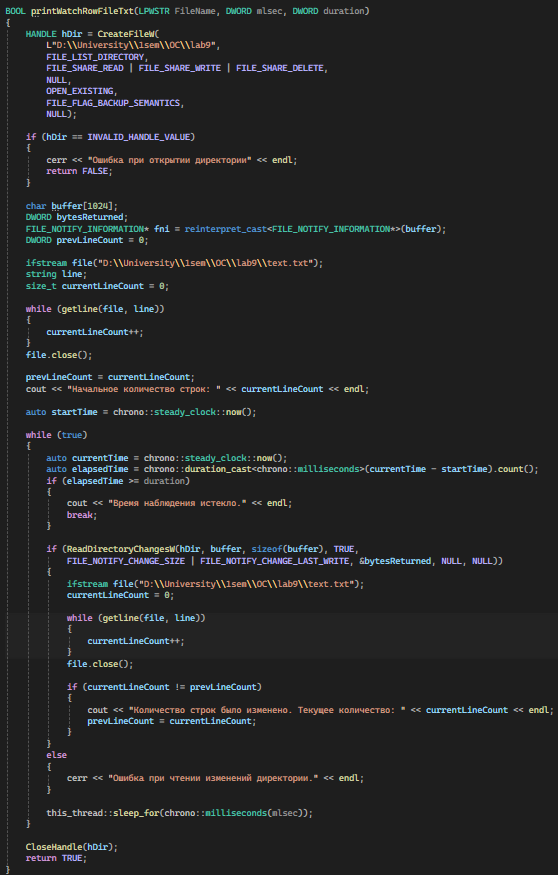


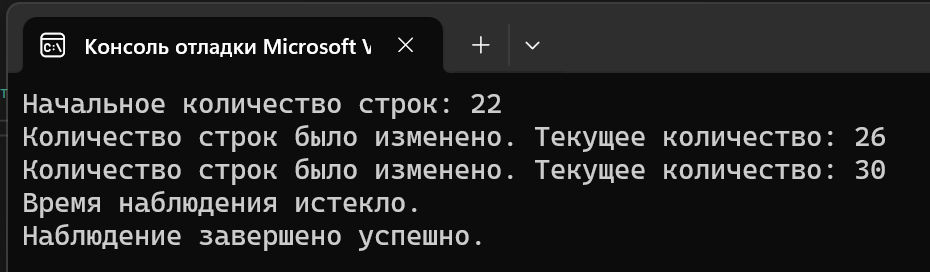


**Задание 04.Windows**

Разработайте приложение OS09\_04.Приложение OS09\_04 вызывает функцию printWathRowFileTxt, имеющую следующий прототип.

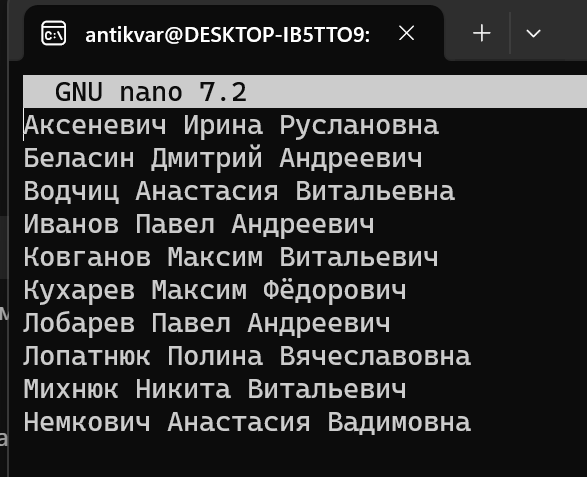
Функция применяется к файлу OS09\_01.txt (п.2), следит (используйте функцию наблюдения за файлами в каталоге) за изменением количества строк в файле в течении mlsec и выводит информацию об изменениях в стандартный поток вывода. Продемонстрируйте работоспособность приложения OS09\_04 совместно с приложениями OS09\_03 и OS09\_04.

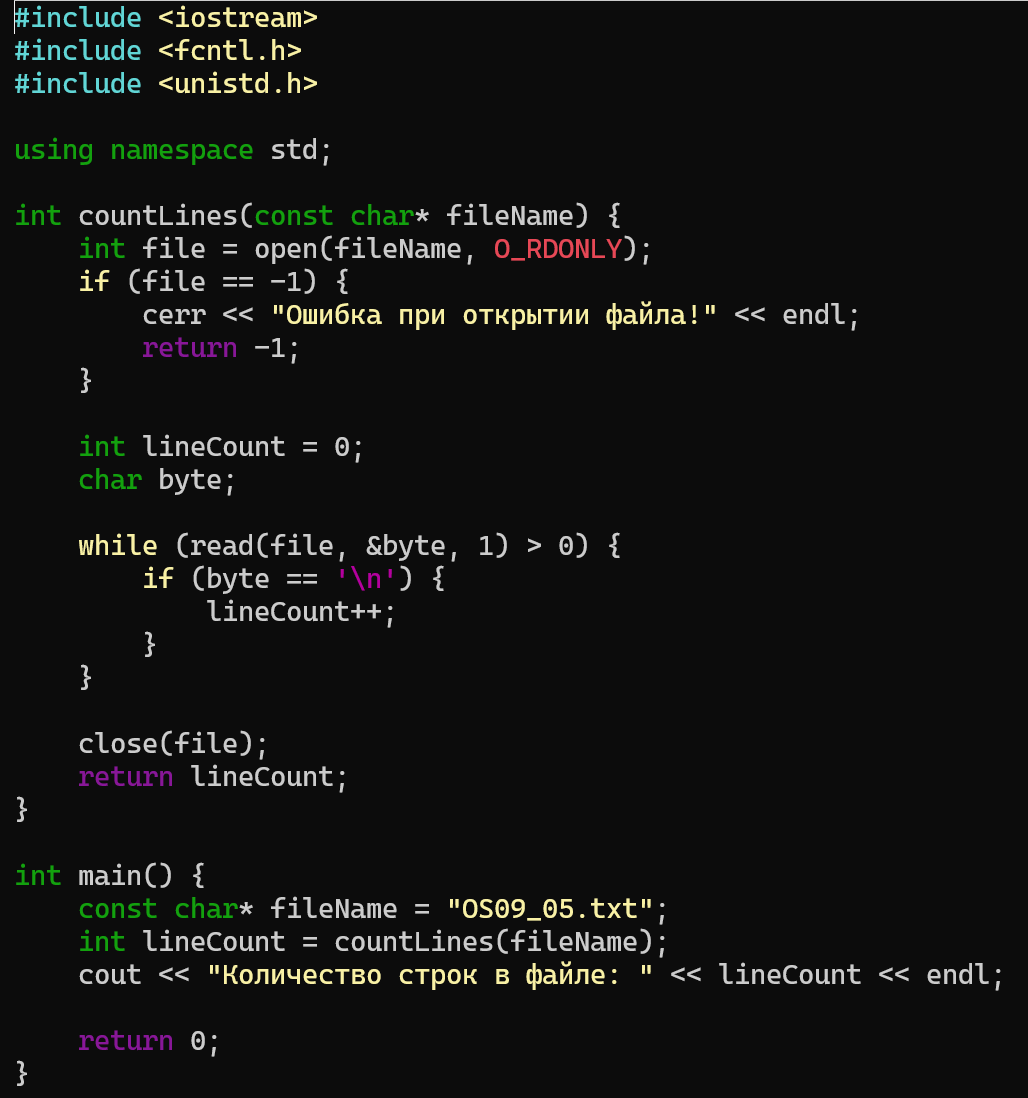


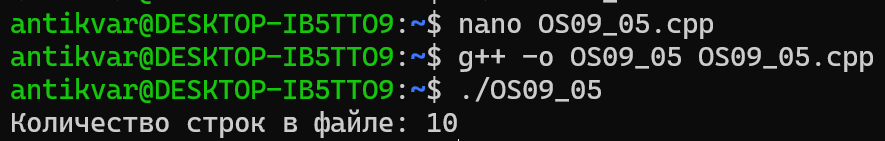


**Задание 05.Linux**

Создайте текстовый файл OS09\_05.txt, аналогичный файлу OS09\_01.txt (п.2). Разработайте приложение OS09\_05, подсчитывающее количество строк и выводящее это значение в стандартный поток. Продемонстрируйте работоспособность приложения OS09\_05.

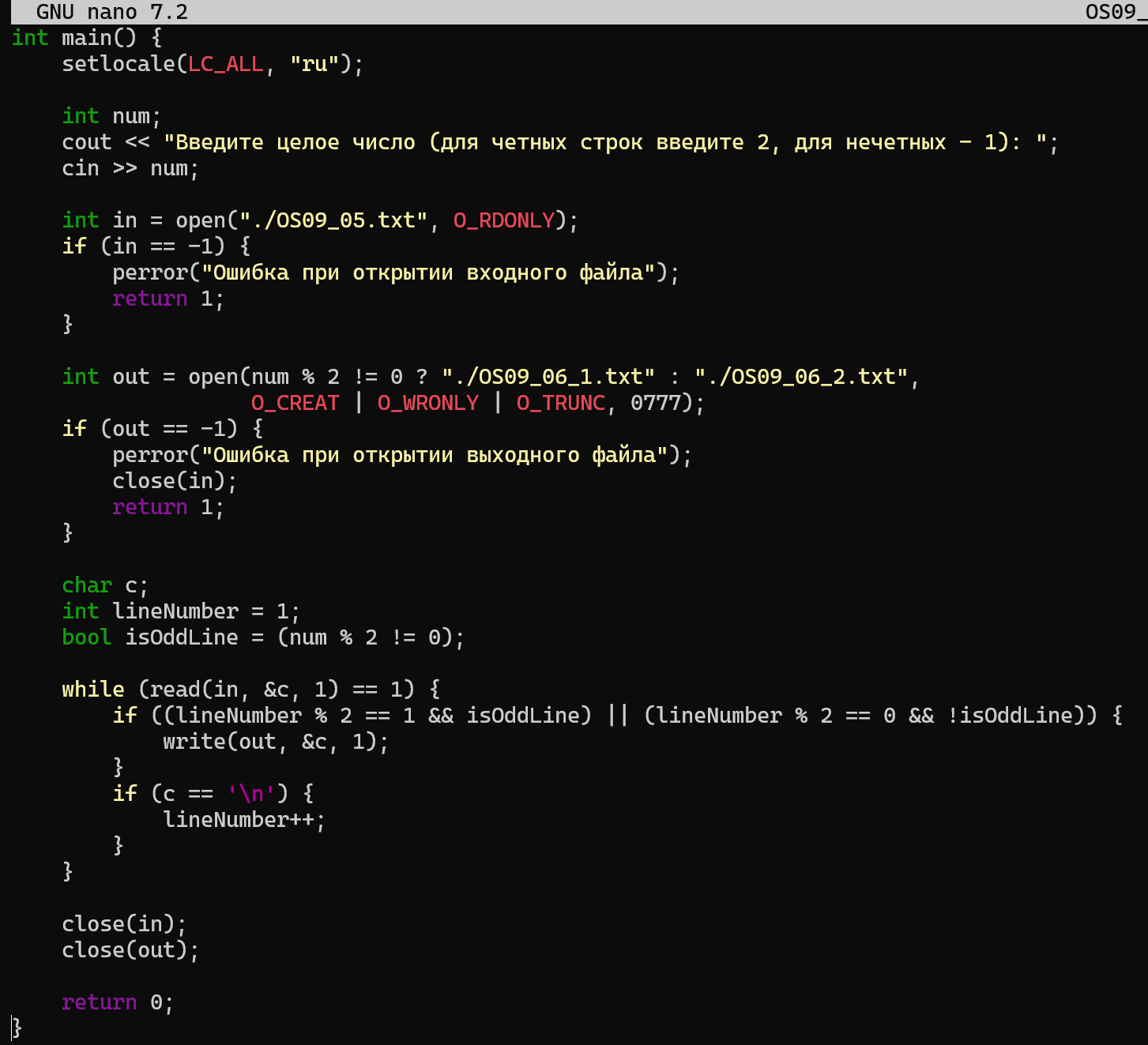


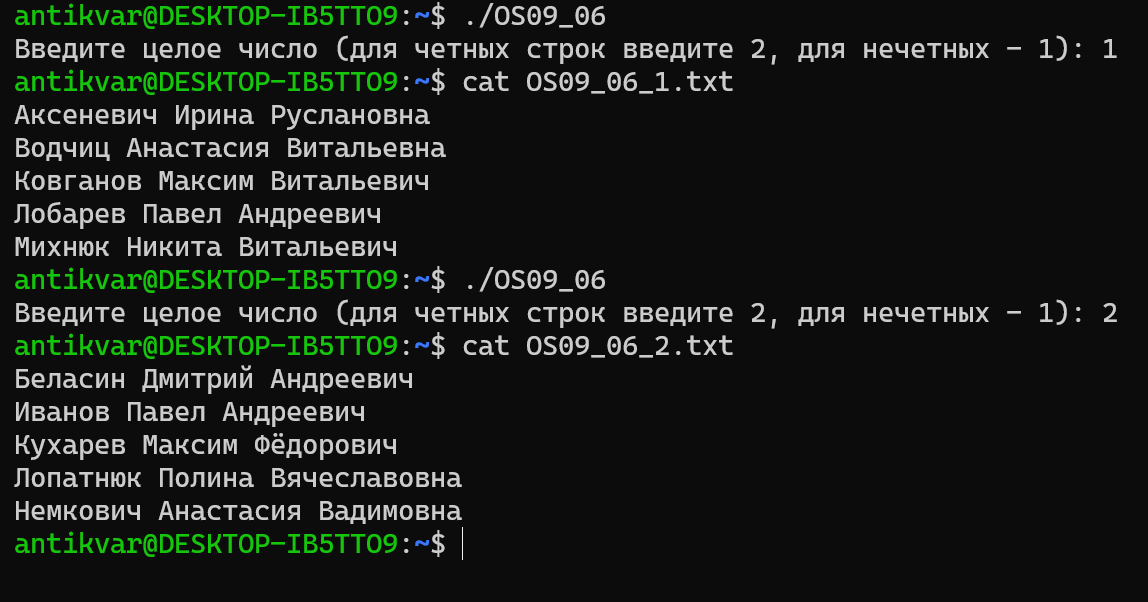




**Задание 06.Linux**

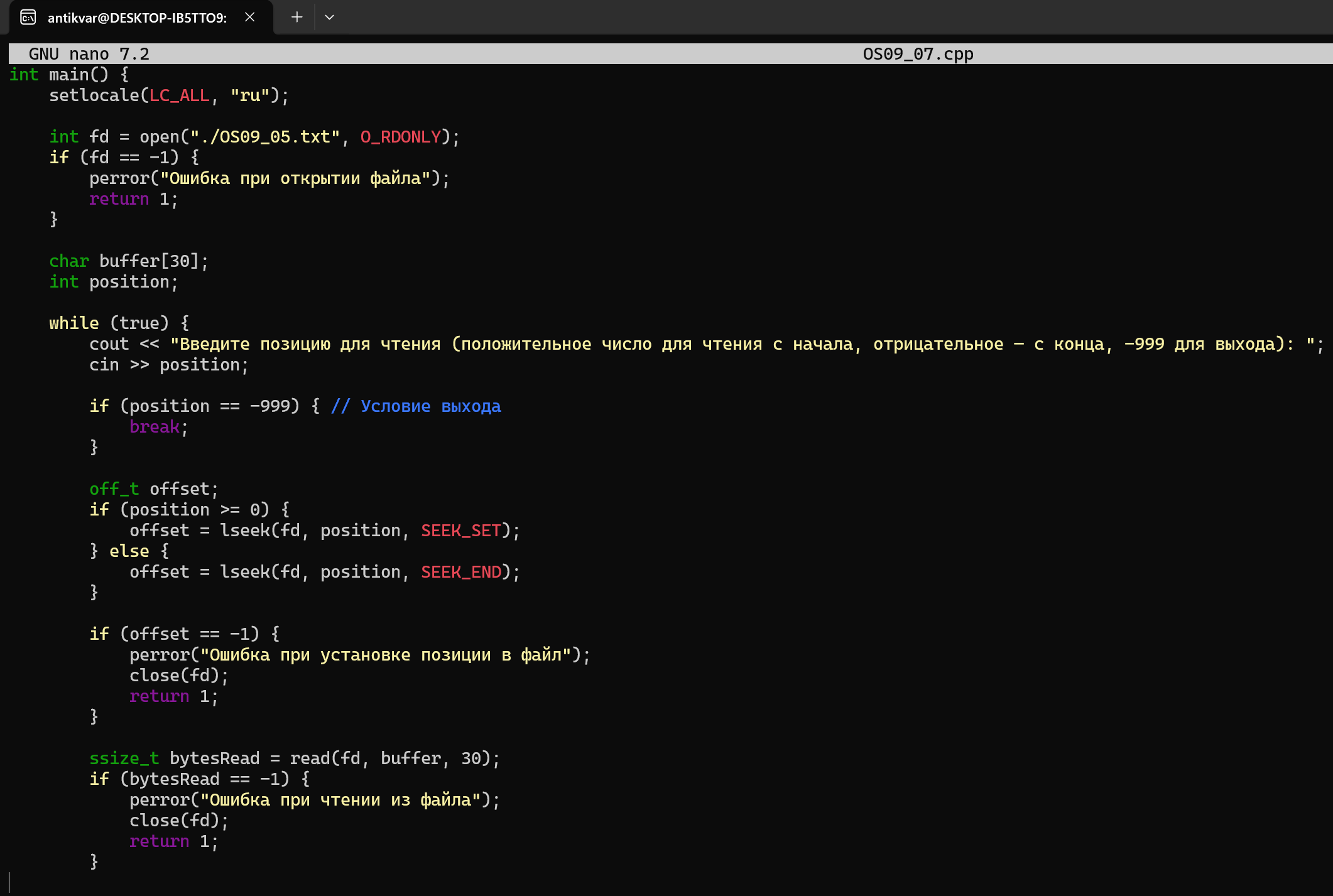
Разработайте приложение OS09\_06, принимающее 1 параметр, принимающее целочисленное числовое значение. Если числовое значение принимает нечетное значение, то приложение создает новый файл OS09\_06\_1.txt, содержащий только нечетные строки из файла OS09\_05.txt. Если числовое значение принимает четное значение, то приложение создает новый файл OS09\_06\_2.txt, содержащий только четные строки из файла OS09\_05.txt. Продемонстрируйте работоспособность приложения OS09\_06.

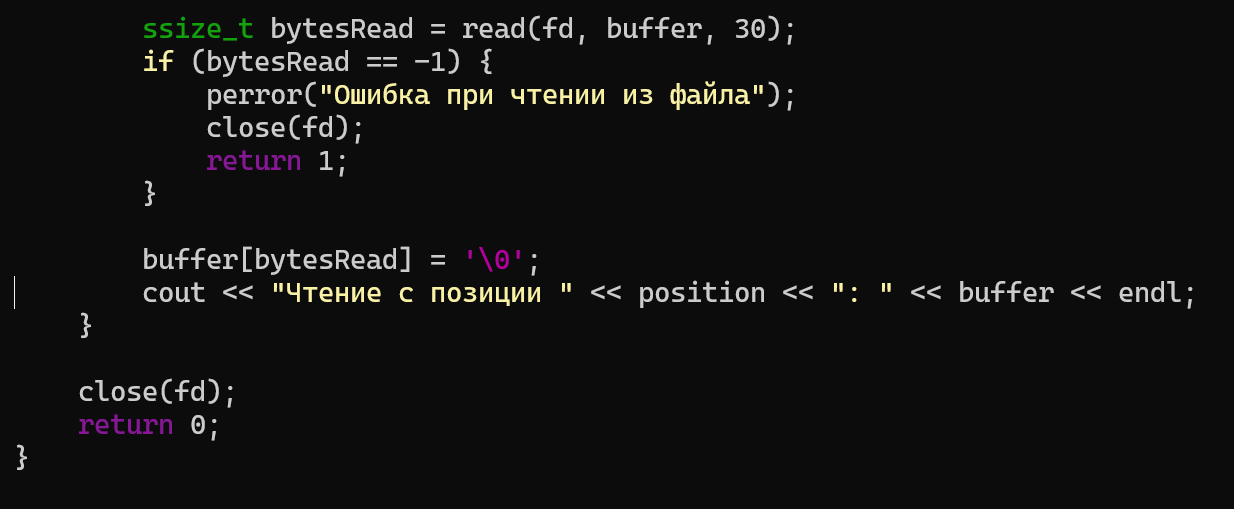


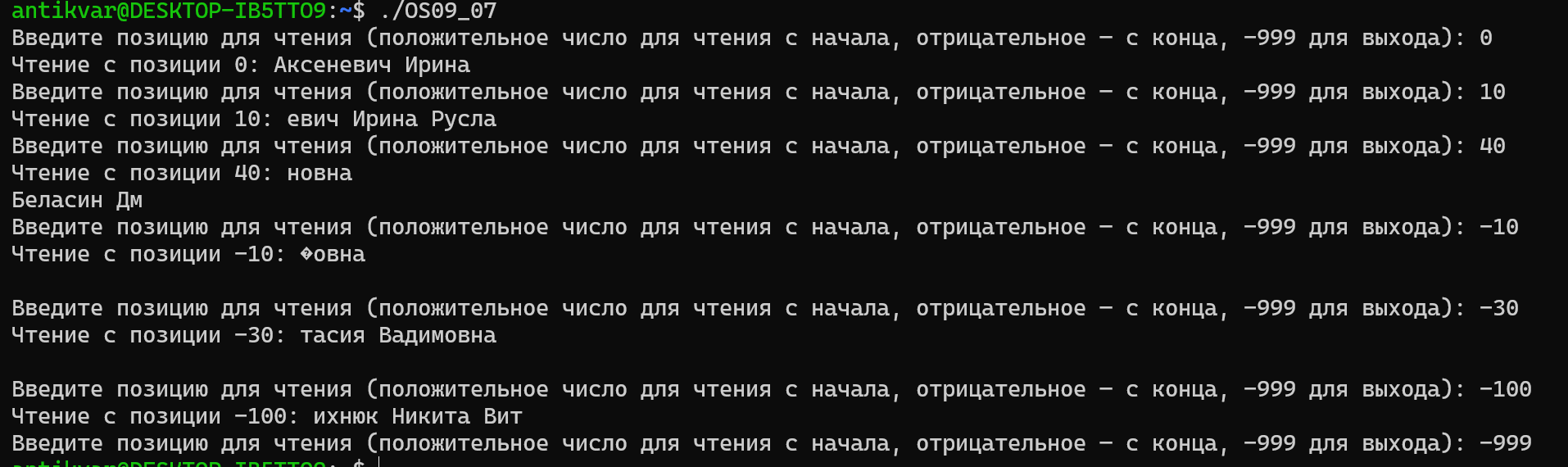


**Задание 07.Linux**

Разработайте приложение OS09\_07, демонстрирующее возможности функции lseek. Поясните назначение и принцип работы функции lseek.







**Задание 08. Ответьте на следующие вопросы**

1. **Что такое файл?**

**Файл** — это единица хранения данных на компьютере, которая используется для записи, хранения и управления информацией. Он представляет собой последовательность байтов, организованных и сохранённых в памяти компьютера или на внешнем носителе (жёстком диске, флешке, CD и т. д.).

1. **Перечислите основные характеристики (атрибуты) файла.**

1.Имя файла – это название, присвоенное файлу. Обычно это строка символов.

2.Идентификатор: Уникальный номер, присвоенный файлу. Он используется для идентификации файлов в файловой системе. В отличие от имен файлов, идентификатор не читаем для человека.

3.Тип: Определяет тип файла, например, архивный файл (.zip), файл исходного кода (.c, .java), текстовый файл (.docx, .txt) и другие.

4.Расположение: Указывает местоположение файла на устройстве (путь к каталогу). Этот атрибут представляет собой указатель на устройство.

5.Размер: Указывает текущий размер файла (в КБ, МБ, ГБ и т.д.) и, возможно, максимальный допустимый размер.

6.Защита: Определяет информацию о контроле доступа (разрешения на чтение, редактирование, запись и выполнение файла). Обеспечивает безопасность конфиденциальной информации.

7.Время, дата и идентификация пользователя: Эта информация включает дату и время создания файла, дату последнего изменения, а также данные о пользователе, который создал или изменил файл.

1. **Что такое файловая система?**

File System, файловая система: система управления файлами, часть операционной системы обеспечивающая доступ к файлам. Устанавливает связь между логическим представлением и физическим расположением данных (абстракция над данными).

Некоторые распространенные типы файловых систем:

* FAT (File Allocation Table): Старая файловая система, используемая в ранних версиях Windows и других ОС.
* NTFS (New Technology File System): Современная файловая система, используемая в Windows. Поддерживает такие функции, как разрешения на файлы и папки, сжатие и шифрование.
* ext (Extended File System): Файловая система, часто используемая в Linux и Unix-подобных ОС.
* HFS (Hierarchical File System): Файловая система, используемая в macOS.
* APFS (Apple File System): Новая файловая система, представленная Apple для Mac и устройств iOS.

1. **Перечислите основные функции файловой системы.**

* Организация данных: файлы хранятся в виде структур (например, директорий и поддиректорий).
* Именование: предоставляет способ присваивания уникальных имен файлам и каталогам.
* Доступ к данным: обеспечивает механизмы чтения, записи, удаления и обновления данных.
* Управление пространством: контролирует распределение свободного и занятого места на носителе.
* Защита данных: ограничивает доступ к файлам и папкам, поддерживая безопасность данных.
* Устранение фрагментации: поддерживает целостность данных и производительность, оптимизируя расположение файлов.

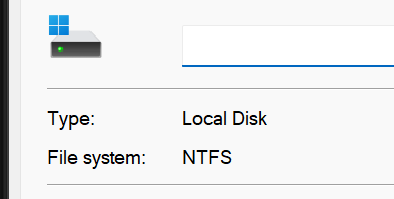
1. **Перечислите 3 названия файловой системы.**

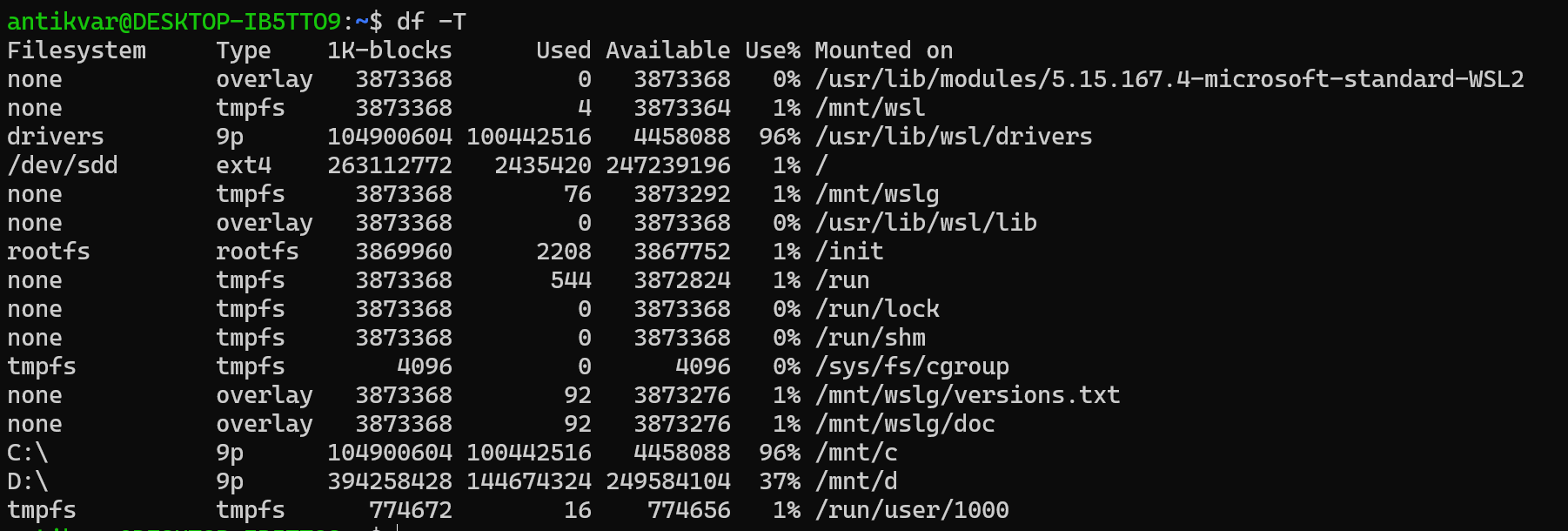
NTFS (New Technology File System). Используется в операционных системах Windows. Поддерживает большие файлы, права доступа, шифрование и восстановление данных.

ext (Fourth Extended File System). Применяется в Linux. Предоставляет высокую производительность, устойчивость к сбоям и поддержку больших объёмов данных.

APFS (Apple File System). Разработана для macOS и iOS. Оптимизирована для работы с SSD и поддерживает шифрование и мгновенные снимки данных.

1. **Какая файловая система установлена на вашем компьютере под Windows? под Linux?**





Раздел / (корневая файловая система) использует файловую систему **ext4**

tmpfs и overlay типичны для виртуализированных систем.

1. **Что такое каталог файловой системы? перечислите наименования специальных каталогов.**

**Каталог файловой системы** — это логическая структура, которая используется для организации файлов. Он представляет собой "папку", содержащую ссылки на файлы или другие каталоги (подкаталоги). Каталоги помогают упорядочить данные и упрощают навигацию в файловой системе.

Каталоги образуют иерархическую структуру:

Корневой каталог (/ в Linux или C:\ в Windows) — это вершина дерева каталогов. Подкаталоги находятся на более низких уровнях, организуя файлы в группы.

**В Linux**

/ — корневой каталог: Вершина файловой системы. Содержит все остальные каталоги и файлы.

/home — домашние каталоги пользователей:Хранит данные и настройки для каждого пользователя (например, /home/user).

/root — домашний каталог суперпользователя (root): Используется для работы администратора системы.

/bin — основные исполняемые файлы: Программы, необходимые для базового функционирования системы (например, ls, cp).

/sbin — системные исполняемые файлы: Команды для администрирования (например, reboot, fdisk).

/etc — конфигурационные файлы: Настройки системы и программ.

/dev — устройства: Файлы, представляющие подключённые устройства (например, жесткие диски, USB).

/var — переменные данные: Логи, временные файлы и данные служб (например, почта, база данных).

/tmp — временные файлы: Каталог для временных данных, удаляемых при перезагрузке.

/usr — пользовательские приложения: Программы и библиотеки, установленные пользователем.

**В Windows** (основные каталоги):

C:\ — корневой каталог системного диска: Вершина файловой системы для основного диска.

C:\Windows — системные файлы Windows: Содержит ядро ОС, драйверы и конфигурационные файлы.

C:\Program Files — программы: Каталог для установленных приложений.

C:\Users — профили пользователей: Хранит данные и настройки для каждого пользователя.

C:\Temp или %TEMP% — временные файлы: Используется для хранения временных данных.

C:\System32 — системные утилиты и драйверы: Важный каталог, содержащий программы и библиотеки для работы ОС.

1. **Поясните понятие «текущий каталог приложения».**

**Текущий каталог приложения** — это каталог (папка) в файловой системе, в котором в данный момент выполняется программа или с которым она активно работает. Он используется по умолчанию для операций с файлами, если в команде или коде программы не указаны другие пути.

1. **Что такое специальные имена файлов? перечислите их, для чего они нужны.**

**Специальные имена файлов** — это зарезервированные операционной системой имена, которые представляют собой устройства, потоки данных или специальные функции, а не обычные файлы. Они используются для взаимодействия с оборудованием, ввода/вывода данных и выполнения системных операций.

Специальные имена, такие как con, lpt1, prn, aux, com, и другие, не могут быть использованы как имена файлов в операционных системах Windows. Это связано с тем, что эти имена зарезервированы для системных устройств, и использование их в качестве имен файлов или папок может вызвать проблемы с функциональностью системы.

1. **Для чего используются буферы ввода-вывода?**

Буфер ввода-вывода — это область памяти, которая используется для временного хранения данных, которые были считаны или должны быть записаны на физический носитель (например, жесткий диск, SSD, или сетевое устройство). Буфер является своего рода промежуточным звеном между медленным устройством хранения и более быстрыми компонентами системы, такими как процессор и оперативная память.

Назначение буферов ввода-вывода:

* Без буферизации каждое обращение к устройству ввода/вывода требует отдельной операции. Буферизация позволяет собирать данные в памяти и выполнять операции с устройством пакетами, что уменьшает нагрузку.
* При записи или чтении данных без буфера процесс может быть ограничен медленной скоростью устройства. Буферизация позволяет процессору работать с данными в оперативной памяти, а затем передавать их на устройство целыми блоками.
* Процессор работает значительно быстрее, чем устройства ввода/вывода. Буфер помогает выравнивать скорость обработки данных, обеспечивая эффективное взаимодействие.
* Буферизация позволяет системе продолжать работу, пока данные поступают или записываются, без необходимости ожидать завершения операции ввода/вывода.

1. **Поясните понятие «кэширование».**

**Кэширование** — это процесс хранения копий данных в более быстрой памяти (обычно в кэше), чтобы ускорить доступ к этим данным в будущем. Вместо того чтобы каждый раз обращаться к более медленным источникам данных (например, жесткому диску или сети), система сохраняет наиболее часто используемые или недавно использованные данные в быстродействующем хранилище (например, оперативной памяти или специализированном кэш-памяти).

1. **Поясните понятие «указатель позиции файла».**

Указатель файла — это специальный объект, который в файловой системе используется для отслеживания текущего положения в файле. Указатель помогает операционной системе или приложению понимать, где в файле находятся данные, которые нужно прочитать или записать.

Этот указатель позиции в файле - значение смещения, определяющее следующий байт, который читается или место, которое получит следующий записываемый байт. Каждый раз, когда файл открывается, система размещает указатель позиции в файле в его начале, который является нулевым смещением. Каждая операция чтения и записи продвигают указатель позиции в файле на число байтов, прочитанных и записанных.

Указатель позиции в файле может быть возвращен обратно при помощи вызова функции SetFilePointer.

Когда указатель позиции в файле достигает его конца, а приложение пытается читать из файла, ошибка не происходит, но никакие байты не читаются. Поэтому, чтение нулевых байтов без ошибки означает, что приложение достигло конца файла. Запись нулевых байтов ничего не делает.

Приложение может обрезать или продлить файл, используя функцию SetEndOfFile. Эта функция устанавливает конец файла в текущей позиции указателя позиции в файле.

1. **Поясните понятие «маркер конца файла».**

**Маркер конца файла** (EOF, End of File) — это специальный символ или код, который указывает, что файл был полностью прочитан, и дальнейшее чтение невозможно. Он сигнализирует программе, что достигнут конец данных в файле.

* Уведомляет программу о завершении чтения данных из файла.
* Позволяет корректно завершить процесс обработки файла.

В языке C при чтении из файла с помощью функции fgetc() возвращается значение EOF, если достигнут конец файла.

Чтобы проверить конец файла, в C++ используют метод eof().

1. **Поясните понятие «блокировка файла».**

Блокировка файла (или lockfile) — это механизм управления доступом к файлам, используемый для предотвращения одновременного доступа нескольких процессов к одному файлу, что может привести к конфликтам и ошибкам.

Когда файл заблокирован, другие процессы или потоки, пытающиеся получить доступ к этому файлу, вынуждены ждать, пока блокировка не будет снята.

LockFile(), UnlockFile().

1. **Windows. Функция OS API для создания файла.**

|  |
| --- |
| HANDLE CreateFile(  LPCSTR lpFileName,  DWORD dwDesiredAccess,  DWORD dwShareMode,  LPSECURITY\_ATTRIBUTES lpSecurityAttributes,  DWORD dwCreationDisposition,  DWORD dwFlagsAndAttributes,  HANDLE hTemplateFile  ); |

lpFileName: Имя файла.

dwDesiredAccess: Запрашиваемые права доступа (например, чтение или запись).

dwShareMode: Режим совместного использования файла.

dwCreationDisposition: Указывает, как должна быть обработана ситуация, если файл уже существует (например, создать новый или открыть существующий).

dwFlagsAndAttributes: Атрибуты файла (например, скрытый, системный).

hTemplateFile: Указатель на шаблон файла (обычно NULL).

1. **Windows. Функция OS API для открытия файла.**

|  |
| --- |
| HANDLE CreateFile(  LPCSTR lpFileName,  DWORD dwDesiredAccess,  DWORD dwShareMode,  LPSECURITY\_ATTRIBUTES lpSecurityAttributes,  DWORD dwCreationDisposition,  DWORD dwFlagsAndAttributes,  HANDLE hTemplateFile  ); |

dwDesiredAccess: Запрашиваемые права доступа (чтение, запись).

dwShareMode: Режим совместного использования файла.

dwCreationDisposition: Указывает, что делать, если файл уже существует (например, открыть существующий).

dwFlagsAndAttributes: Атрибуты файла (например, архивный).

1. **Windows. Функция OS API для удаления файла.**

|  |
| --- |
| BOOL DeleteFile(  LPCSTR lpFileName  ); |

lpFileName: Имя файла, который необходимо удалить.

1. **Windows. Функция OS API для записи в файл.**

|  |
| --- |
| BOOL WriteFile(  HANDLE hFile,  LPCVOID lpBuffer,  DWORD nNumberOfBytesToWrite,  LPDWORD lpNumberOfBytesWritten,  LPOVERLAPPED lpOverlapped  ); |

hFile: Дескриптор файла.

lpBuffer: Указатель на данные для записи.

nNumberOfBytesToWrite: Количество байт для записи.

lpNumberOfBytesWritten: Указатель на переменную для записи фактически записанных байт.

lpOverlapped: Указатель на структуру для асинхронных операций (обычно NULL для синхронных).

1. **Windows. Функция OS API для чтения файла.**

|  |
| --- |
| BOOL ReadFile(  HANDLE hFile,  LPVOID lpBuffer,  DWORD nNumberOfBytesToRead,  LPDWORD lpNumberOfBytesRead,  LPOVERLAPPED lpOverlapped  ); |

hFile: Дескриптор файла.

lpBuffer: Указатель на буфер для хранения прочитанных данных.

nNumberOfBytesToRead: Количество байт для чтения.

lpNumberOfBytesRead: Указатель на переменную для записи фактически прочитанных байт.

lpOverlapped: Указатель на структуру для асинхронных операций (обычно NULL для синхронных).

1. **Windows. Назначение и отличие функций OS API: CopyFile, MoveFile, ReplaseFile.**

Функция CopyFile используется для копирования одного файла в другое место (или под другим именем). Она копирует файл из одного расположения в другое, сохраняя его данные.

Функция MoveFile используется для перемещения файла или каталога из одного места в другое. В отличие от CopyFile, при использовании MoveFile исходный файл удаляется из старого местоположения и появляется в новом.

Функция ReplaceFile используется для замены одного файла другим. Она позволяет заменить файл в одном месте другим файлом (с возможностью создания резервной копии), при этом замещаемый файл удаляется после замены. ReplaceFile заменяет содержимое одного файла другим файлом, сохраняя метаданные (атрибуты безопасности, время создания и т.д.) файла назначения, с возможностью создания резервной копии.

1. **Windows. Перечислите функции OS API, которые изменяют текущее значение указателя позиции файла.**

SetFilePointer и SetFilePointerEx — функции для изменения текущего положения указателя в файле.

SetFilePointer работает с 32-битными смещениями.

SetFilePointerEx поддерживает 64-битные смещения, что позволяет работать с файлами размером более 4 ГБ.

1. **Windows. Перечислите функции OS API для блокировки и разблокировки файлов.**

LockFile — блокирует часть файла синхронно.

LockFileEx — блокирует часть файла с поддержкой асинхронных операций и дополнительных флагов.

UnlockFile — разблокирует часть файла синхронно.

UnlockFileEx — разблокирует часть файла с поддержкой асинхронных операций.

1. **Windows. Поясните механизм «наблюдение за каталогом», перечислите набор функций OS API, позволяющий реализовать этот механизм.**

Механизм «наблюдения за каталогом» в Windows позволяет приложению отслеживать изменения в файловой системе, такие как создание, удаление, изменение или переименование файлов и каталогов в указанной директории.

Для реализации механизма наблюдения за каталогом в Windows используются следующие основные функции:

FindFirstChangeNotification — для начала наблюдения за изменениями в каталоге.

FindNextChangeNotification — для продолжения наблюдения.

FindCloseChangeNotification — для завершения наблюдения.

ReadDirectoryChangesW — для получения подробной информации о произошедших изменениях в каталоге.

1. **Windows. Перечислите функции OS API для работы с каталогами, поясните их назанчения.**

* CreateDirectory – Создаёт новый каталог.
* RemoveDirectory – Удаляет каталог (только если он пуст).
* GetCurrentDirectory – Получает текущий рабочий каталог.
* SetCurrentDirectory – Устанавливает новый текущий рабочий каталог.
* FindFirstFile – Открывает каталог для поиска файлов с указанным шаблоном.
* FindNextFile – Получает информацию о следующем файле или каталоге в текущем каталоге.
* GetFileAttributes – Получает атрибуты файла или каталога.
* SetFileAttributes – Устанавливает атрибуты для файла или каталога.
* MoveFile – Перемещает файл или каталог в новое место.
* MoveFileEx – Перемещает файл или каталог с дополнительными параметрами (например, заменяет существующий).
* GetVolumeInformation – Получает информацию о томе (диске), его типе и файловой системе.

1. **Linux. Что такое FHS?**

FHS (Filesystem Hierarchy Standard) — это стандарт файловой иерархии в операционных системах Linux, который определяет структуру каталогов и назначения файлов в системе. FHS описывает, как должна быть организована файловая система, чтобы она была удобной для использования и совместимой с различными дистрибутивами Linux.

Ключевые каталоги, описанные в FHS:

* / (root) — корневой каталог, из которого начинается файловая система.
* /bin — содержит основные системные бинарные файлы (например, команды, которые могут быть использованы пользователями и системными процессами).
* /boot — файлы, необходимые для загрузки системы, включая ядро.
* /dev — устройства, которые представлены как файлы (например, диски, терминалы).
* /etc — конфигурационные файлы системы.
* /home — домашние каталоги пользователей.
* /lib — библиотеки, необходимые для работы программ в /bin и /sbin.
* /media – для монтирования съемных носителей и других внешних устройств.
* /mnt — временные точки монтирования файловых систем.
* /opt — дополнительные пакеты программ, не входящие в стандартный набор.
* /root — домашний каталог пользователя root (суперпользователя).
* /sbin — системные бинарные файлы, которые обычно используются только администраторами системы.
* /srv — данные, которые предоставляются сервисами, запущенными на системе.
* /tmp — временные файлы, используемые системой и приложениями.
* /usr — содержит программы, библиотеки, документацию и другие файлы, доступные для использования пользователями.
* /var — переменные файлы, такие как логи, базы данных, временные файлы.

1. **Linux. Перечислите типы файловых систем.**

ext4 — наиболее популярная файловая система, поддерживает журналирование и хороша для большинства пользователей.

ext3 — предшественник ext4, также поддерживает журналирование, но менее эффективен.

ext2 — старый тип файловой системы без журналирования.

Btrfs — современная файловая система с поддержкой снимков и сжатия данных.

XFS — высокая производительность для больших файлов и серверных приложений.

F2FS — оптимизирована для работы с флеш-памятью (SSD).

ReiserFS — старая файловая система, хорошо работает с маленькими файлами.

JFS — файловая система от IBM с журналированием, используется на серверах.

NTFS — файловая система Windows, поддерживается в Linux для совместимости.

FAT32 — устаревшая, но совместимая с множеством ОС, ограничена размером файлов до 4 ГБ.

exFAT — улучшенная версия FAT32, поддерживает большие файлы.

ZFS — продвинутая файловая система с поддержкой снимков и проверки целостности данных.

1. **Linux. Что такое inode?**

Inode (индексный дескриптор) — это структура данных, которая хранит информацию о файле в файловой системе Linux (и других Unix-подобных системах). В отличие от имени файла, inode содержит метаданные, которые описывают файл, такие как:

* Тип файла (например, обычный файл, каталог, символическая ссылка).
* Права доступа (разрешения на чтение, запись и выполнение).
* Владелец файла (пользователь и группа).
* Размер файла.
* Временные метки (время создания, последнего изменения и последнего доступа).
* Ссылки на блоки данных, где физически хранится содержимое файла.

Имя файла не хранится в inode. Оно хранится в директории, которая сопоставляет имя файла с его inode. Каждый файл в файловой системе имеет уникальный inode. Если файл имеет несколько имен (жесткие ссылки), все они будут ссылаться на один и тот же inode.

ls -i

1. **Linux. Поясните назначение функций open, read, write, close, ioctl, stat, flush, lseek, lstat, fstat.**

open — Открывает файл или устройство для чтения/записи.

read — Читает данные из файла в буфер.

write — Записывает данные из буфера в файл.

close — Закрывает файл, освобождая ресурсы.

ioctl — Отправляет управляющие команды устройствам или файлам.

stat — Получает информацию о файле по его пути.

flush — Сбрасывает буферизованные данные в файл (для потоков).

lseek — Изменяет позицию указателя в файле.

lstat — Получает информацию о файле, не следуя за символической ссылкой.

fstat — Получает информацию о файле по файловому дескриптору.