**Лабораторная работа №6**

**Управление файлами I**

**Цель работы:** Получение практических навыков в работе с файловой системой.

Перед выполнением задания прочитайте его полностью. Схематично представьте себе, что от вас требуется и только после этого выполняйте.

**Постановка задачи для Windows:**

**Приложение Lab-06a:**

Разработайте консольное приложение **Lab-06a** в котором создайте две функции **PrintInfo(LPSTR FileName)** и **PrintText(LPSTR FileName)**.

Функция **PrintInfo** должна выводить следующую информацию о файле:

* Имя файла;
* Размер файла (в Б, КиБ, МиБ);
* Тип файла (не путать с расширением файла);
* Время создания;
* Время последнего доступа;
* Время последнего изменения;
* Бинарный ли файл и если да, то какого типа.

Функция **PrintText** должна выводить содержимое указанного файла на консоль. Если указанный файл не является текстовым, то функция выводит сообщение об этом и завершает своё выполнение.

Путь к файлу должен передаваться через аргументы командной строки.

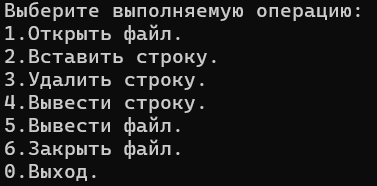
Предусмотреть обработку ошибок и неверных аргументов, передаваемых в функции, а также корректное освобождение используемых ресурсов.

**Приложение Lab-06b:**

Разработайте консольное приложение **Lab-06b** дляработы с текстовым файлом, содержащим список студентов. В приложении создайте следующие функции:

* **OpenFile**(LPSTR filePath);
* **AddRow**(HANDLE hFile, LPSTR row, INT pos);
* **RemRow**(HANDLE hFile, INT pos);
* **PrintRow**(HANDLE hFile, INT pos);
* **PrintRows(**HANDLE hFile);
* **CloseFile**(HANDLE hFile).

Хранение дескриптора файла, с которым работает процесс, а также всех необходимых буферов для данных осуществляется на глобальном уровне (глобальные переменные). Взаимодействие с файлом происходит через упомянутые ранее функции. Вызовы функций приложение обеспечивает через консоль в виде меню подобного вида:



Данные для функций вводятся с консоли. Предусмотреть обработку ошибок ввода данных.

Поведение функций должно быть следующим:

**OpenFile –** должнапроизводить попытку открыть файл и записать дескриптор открытого файла в глобальную переменную, в случае если файл не существует, то возвращает ошибку. Также данная функция должна динамически выделять все необходимые глобальные буферы для последующей работы с файлом.

**AddRow** – должна производить попытку добавления строки в файл, при этом:

* При **pos** больше нуля (>0) строка добавляется в соответствующее место (отсчёт строк в файле по человеческим меркам от 1), при этом строка не может быть добавлена на позицию, превосходящую текущее количество строк + 1;
* При **pos** равном нулю (0) строка добавляется в начало файла;
* При **pos** равном минус одному (-1) строка добавляется в конец файла;
* В остальных случаях строка не может быть вставлена и должна быть возвращена ошибка;
* Также ошибка должна быть возвращена если глобальная переменная не содержит корректного дескриптора или добавляемая строка **row** ни на что не указывает (NULL).

**RemRow** – должна производить попытку удаления строки из файла, при этом:

* При **pos** больше нуля (>0) строка удаляется из соответствующего места (отсчёт строк в файле по человеческим меркам от 1), при этом строка не может быть удалена с позиции, превосходящей текущее количество строк;
* При **pos** равном нулю (0) удаляется первая строка;
* При **pos** равном минус одному (-1) удаляется последняя строка;
* В остальных случаях строка не может быть удалена и должна быть возвращена ошибка;
* Также ошибка должна быть возвращена если глобальная переменная не содержит корректного дескриптора.

**PrintRow** – должна производить попытку вывода строки из файла на консоль, при этом:

* При **pos** больше нуля (>0) выводится строка из соответствующего места (отсчёт строк в файле по человеческим меркам от 1), при этом строка не может быть выведена строка с позиции, превосходящей текущее количество строк;
* При **pos** равном нулю (0) выводится первая строка;
* При **pos** равном минус одному (-1) выводится последняя строка;
* В остальных случаях строка не может быть выведена и должна быть возвращена ошибка;
* Также ошибка должна быть возвращена если глобальная переменная не содержит корректного дескриптора.

**PrintRows –** должна производить попытку вывода содержимого всего файла, при этом, если глобальная переменная не содержит корректного дескриптора, то должна быть возвращена ошибка.

**CloseFile –** должна производить попытку перевода дескриптора из глобальной переменной в недействительное состояние, а в случае если переменная не содержит корректного дескриптора возвращать ошибку. Также данная функция должна освобождать динамически выделенные ресурсы для работы с файлом.

В моменты работы приложения просмотреть информацию об открытом файле в программе Process Explorer. Посмотреть информацию до открытия, после открытия и после закрытия файла.

**Приложение Lab-06с:**

Разработайте консольное приложение **Lab-06с** в котором:

* производится вывод содержимого каталога (и файлов, дочерних каталогов), но если заданного каталога не существует программа возвращает ошибку и завершает свою работу;
* после этого запускается бесконечный цикл, в котором происходит отслеживание изменений в заданном каталоге. При получении сигнала о каком-либо событии в каталоге, приложение должно вывести тип этого события;

Каталог задаётся через аргументы командной строки.

**Постановка задачи для Linux:**

**Приложение Lab-06d:**

Разработайте консольное приложение, являющееся полным аналогом **Lab-06а,** за исключением пункта об определении бинарный файл или нет.

**Приложение Lab-06e:**

Разработайте консольное приложение **Lab-06e** являющееся полным аналогом **Lab-06b.**

**Приложение Lab-06f:**

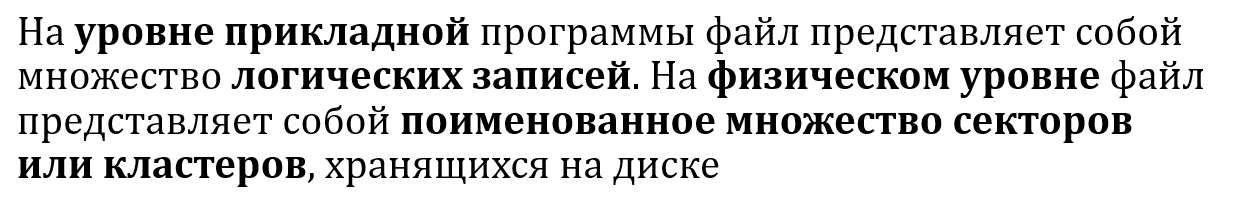
Разработайте консольное приложение **Lab-06f** являющееся полным аналогом **Lab-06c.**

**Требования к выполненной работе:**

* Разработанные приложения должны быть скомпилированы и протестированы как на Windows, так и на ОС из семейства Linux.
* Для генерации и сборки проекта использовать средство CMake.
* Для компиляции файлов при сборке проекта использовать компилятор Clang.

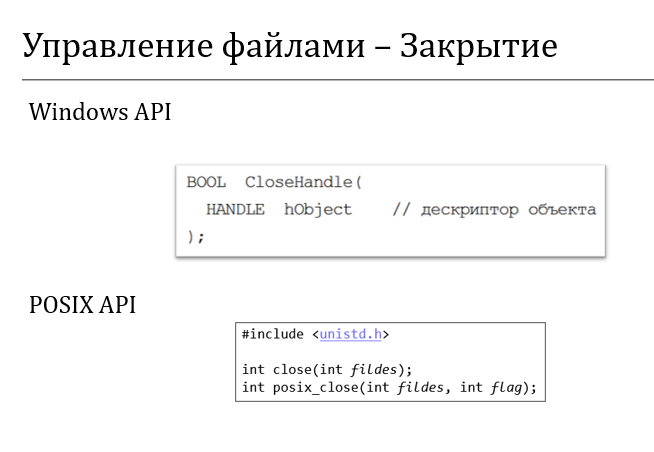
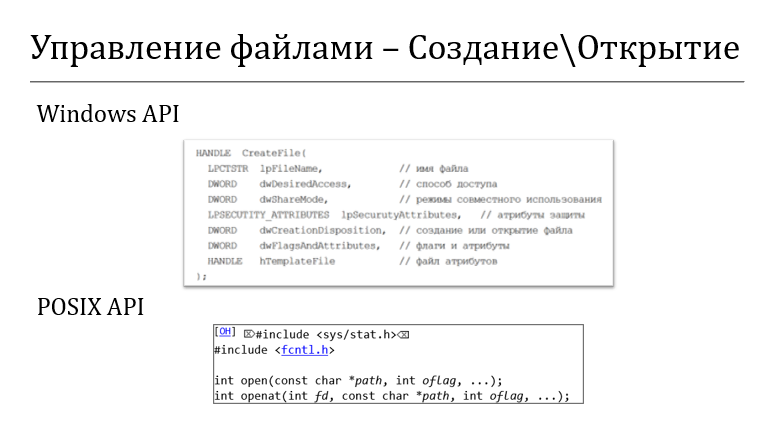
**Вопросы для контроля:**

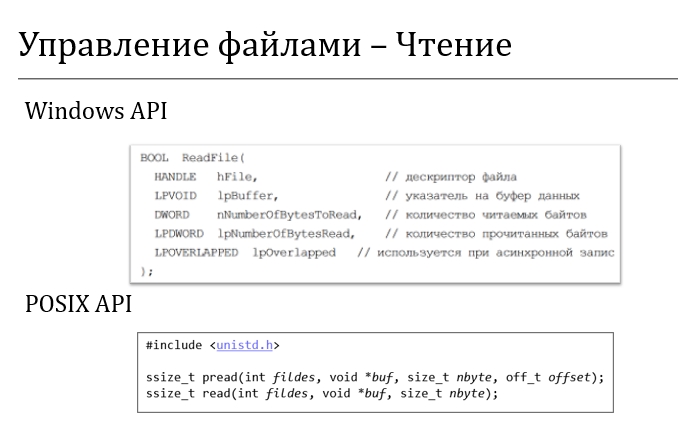
1. Что такое файл?
2. Что такое каталог?
3. Какие функции для работы с файлами из WinAPI вы знаете? Из POSIX?
4. Какие функции для работы каталогами из WinAPI вы знаете? Из POSIX?
5. Поясните механизм отслеживания изменений в каталогах?
6. Поясните функцию **SetFilePointer** и **lseek**.
7. **Что такое файл?**

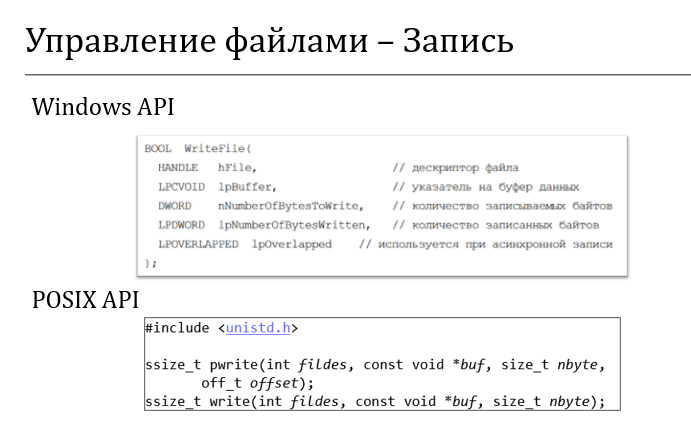
****

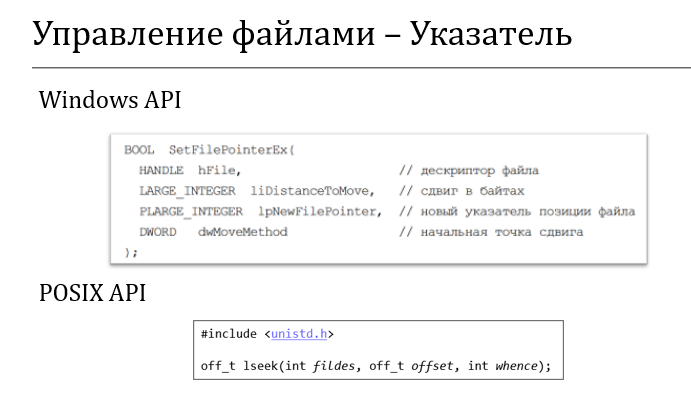
**Файл** — это именованная область на носителе данных, которая используется для хранения информации. Файл содержит данные в текстовом или бинарном формате и может быть организован по различным структурам: последовательности байт, записей, блоков и т.д.

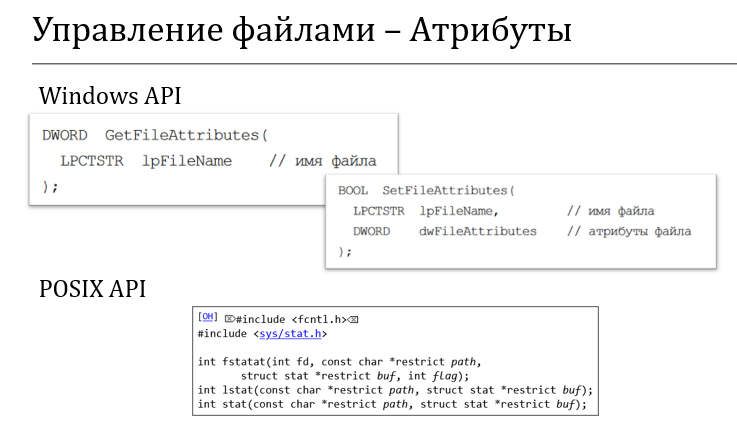


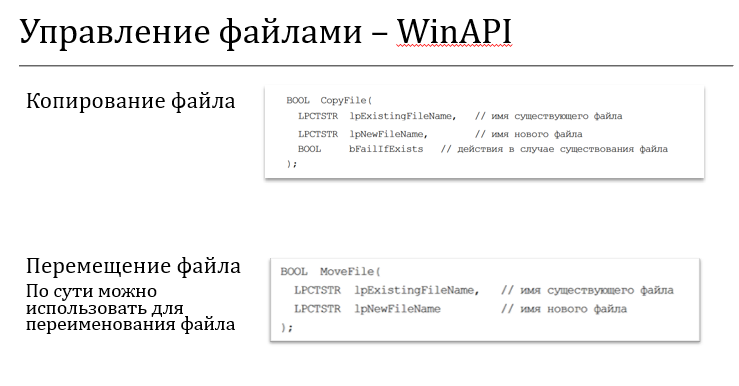


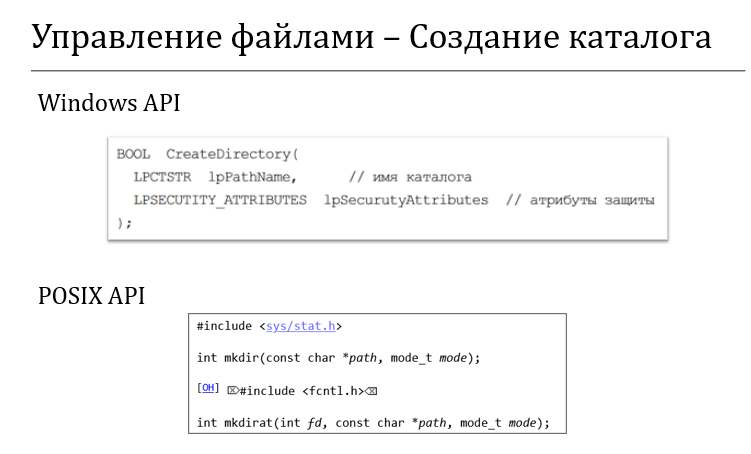


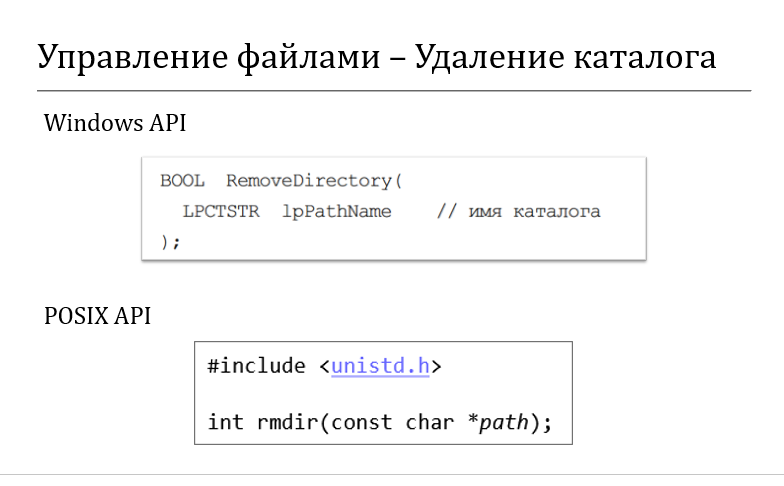


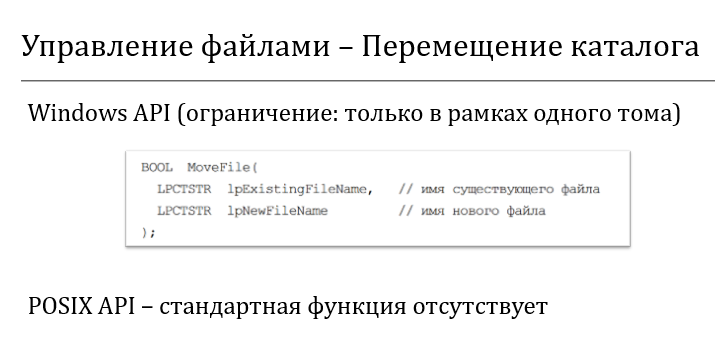












### 5. Механизм отслеживания изменений в каталогах

Отслеживание изменений в каталогах позволяет приложению реагировать на создание, изменение или удаление файлов и подкаталогов.

#### **WinAPI:**

Используется функция ReadDirectoryChangesW:

* Мониторит изменения в каталоге (создание, удаление, изменение файлов).
* Работает асинхронно, отправляя уведомления в буфер.

#### **POSIX (Linux):**

Используется механизм inotify:

* Системный вызов inotify\_add\_watch добавляет наблюдение за каталогом.
* Системный вызов read на дескрипторе inotify возвращает информацию об изменениях.

### 6. Функции SetFilePointer и lseek

#### **SetFilePointer (WinAPI):**

Изменяет текущую позицию указателя в открытом файле.

#### **lseek (POSIX):**

Изменяет текущую позицию указателя в открытом файле.