**Лекция 5. Работа с файлами.**

**Работа с файлами через Windows API**

Для работы с файлами нам понадобятся следующие процедуры:

CreateFile(szName, dwAccess, dwShareMode, lpSecurityAttributes,  
           dwCreationDisposition, dwFlags, hTemplateFile);

В Windows для того, чтобы открыть или создать файл, нужно вызвать процедуру, имеющую целых семь аргументов. К счастью, большинство из них приходится использовать крайне редко.

Аргумент

szName задает имя файла, а

dwAccess — желаемый доступ к файлу, обычно это GENERIC\_READ, GENERIC\_WRITE или оба значения, объединенные логическим или.

Параметр dwShareMode определяет, что могут делать с файлом другие процессы, пока мы с ним работаем. Возможные значения — FILE\_SHARE\_READ, FILE\_SHARE\_WRITE, FILE\_SHARE\_DELETE и их комбинации, однако часто этот параметр просто устанавливают в ноль.

Параметр dwCreationDisposition определяет, как именно мы хотим открыть файл, может быть, например, CREATE\_NEW, CREATE\_ALWAYS, OPEN\_EXISTING, OPEN\_ALWAYS. О семантике этого хозяйства нетрудно догадаться самостоятельно.

С помощью dwFlags можно указать дополнительные свойства файла, например, хранить ли его в зашифрованном или сжатом виде, или сказать, что файл является скрытым, временным или системным. Обычно сюда передают FILE\_ATTRIBUTE\_NORMAL.

Наконец, про lpSecurityAttributes и hTemplateFile сейчас знать не нужно, сюда можно смело передавать NULL. В случае успешного создания или открытия файла, процедура [CreateFile](http://msdn.microsoft.com/en-us/library/windows/desktop/aa363858(v=vs.85).aspx) возвращает его HANDLE (HANDLE - тип данных Windows. Дискриптор обьекта\_ 32-битовое целое значение, описывающее местоположение обьекта в памяти). В случае ошибки возвращается специальное значение INVALID\_HANDLE\_VALUE. Узнать подробности об ошибке можно с помощью [GetLastError](http://msdn.microsoft.com/en-us/library/windows/desktop/ms679360(v=vs.85).aspx).

ReadFile(hFile, lpBuff, dwBuffSize, &dwCount, NULL);

Чтение из файла в буфер lpBuff размером dwBuffSize. В переменную dwCount записывается реальное количество прочитанных байт. Последний опциональный аргумент называется lpOverlapped и о нем сейчас знать не нужно.

WriteFile(hFile, lpBuff, dwBuffSize, &dwCount, NULL);

Аргументы и семантика процедуры [WriteFile](http://msdn.microsoft.com/en-us/library/windows/desktop/aa365747(v=vs.85).aspx) полностью аналогичны [ReadFile](http://msdn.microsoft.com/en-us/library/windows/desktop/aa365467(v=vs.85).aspx).

CloseHandle(hFile);

Файловые дескрипторы закрываются с помощью [CloseHandle](http://msdn.microsoft.com/en-us/library/windows/desktop/ms724211(v=vs.85).aspx).

**Класс CFile**

Класс CFile предназначен для работы с файлами, используется как базовый класс для других классов MFC, предназначенных для работы с файлами. Обеспечивает небуфферизованный двоичный ввод/вывод в файлы.

В класс включено несколько конструкторов.

Конструктор по умолчанию:

CFile();

Конструктор с параметрами:

CFile(LPCTSTR lpszFileName,UINT nOpenFlags),

где lpszFileName – указатель на строку, содержащую имя файла и путь к нему;

nOpenFlags – целочисленная переменная, определяющая режимы доступа к файлу или его атрибуты и принимающая следующие значения:

* CFile::modeCreate - Создается новый файл. Если указанный файл существует, то его содержимое стирается и длина файла устанавливается равной нулю.
* CFile::modeNoTruncate - Этот файл предназначен для использования совместно с файлом CFile::modeCreate. Если создается уже существующий файл, то его содержимое не будет удалено.
* CFile::modeRead - Файл открывается только для чтения.
* CFile::modeReadWrite - Файл открывается для записи и для чтения.
* CFile::modeWrite - Файл открывается только для записи.
* CFile::typeText - Используется классами, порожденными от класса CFile, например CStdioFile, для работы с файлами в текстовом режиме. Текстовый режим обеспечивает преобразование комбинации символа возврата каретки и символа перевода строки.
* CFile::Binary - Используется классами, порожденными от класса CFile, например CStdioFile, для работы с файлами в двоичном режиме.

**Открытие и создание файлов**

После создания объекта класса CFile можно открыть файл, вызвав метод Open. Методу надо указать путь к открываемому файлу и режим его использования. Прототип метода Open имеет следующий вид:

**virtual BOOL Open(LPCTSTR lpszFileName,**

**UINT nOpenFlags, CFileException\* pError=NULL);**

В качестве параметра lpszFileName надо указать имя открываемого файла. Можно указать только имя файла или полное имя файла, включающее полный путь к нему.

Второй параметр nOpenFlags определяет действие, выполняемое методом Open с файлом, а также атрибуты файла. Ниже представлены некоторые возможеые значения параметра nOpenFlags:

* CFile::modeCreate - Создается новый файл. Если указанный файл существует, то его содержимое стирается и длина файла устанавливается равной нулю.
* CFile::modeNoTruncate - Этот файл предназначен для использования совместно с файлом CFile::modeCreate. Если создается уже существующий файл, то его содержимое не будет удалено.
* CFile::modeRead - Файл открывается только для чтения.
* CFile::modeReadWrite - Файл открывается для записи и для чтения.
* CFile::modeWrite - Файл открывается только для записи.
* CFile::typeText - Используется классами, порожденными от класса CFile, например CStdioFile, для работы с файлами в текстовом режиме. Текстовый режим обеспечивает преобразование комбинации символа возврата каретки и символа перевода строки.
* CFile::Binary - Используется классами, порожденными от класса CFile, например CStdioFile, для работы с файлами в двоичном режиме.

Необязательный параметр pError, который является указателем на объект класса CFileException, используется только в том случае, если выполнение операции с файлом вызовет ошибку. При этом в объект, указываемый pError, будет записана дополнительная информация.

Метод Open возвращает не нулевое значение, если файл открыт и нуль в случае ошибки. Ошибка при открытии файла может случиться, например, если методу Open указан для чтения несуществующий файл.

**Закрытие файлов**

После завершения работы с файлом, его надо закрыть. Класс CFile имеет для этого специальный метод Close. Нужно заметить, что если был создан объект класса CFile и открыт файл, а затем объект удаляется, то связанный с ним файл закрывается автоматически с помощью деструктора.

**Чтение и запись файлов**

Для доступа к файлам предназначено несколько методов класса CFile:

Read, ReadHuge, Write, WriteHuge.

Методы Read и ReadHuge предназначены для чтения данных из предварительно открытого файла. В 32-разрядных операционных системах оба метода могут одновременно считать из файла больше 65535 байт. Спецификация ReadHuge считается устаревшей и оставлена только для совместимости с 16-разрядными операционными системами.

Данные, прочитанные из файла, записываются в буфер lpBuf. Параметр nCount определяет количество байт, которое надо считать из файла. Фактически из файла может быть считано меньше байт, чем запрошено параметром nCount. Это происходит, если во время чтения достигнут конец файла. Методы возвращают количество байт, прочитанных из файла.

Для записи в файл предназначены методы Write и WriteHuge. В 32-разрядных операционных системах оба метода могут одновременно записывать в файл больше 65535 байт. Методы записывает в открытый файл nCount байт из буфера lpBuf. В случае возникновения ошибки записи, например переполнения диска, методы вызывает обработку исключения.

**Операции с файлами**

В состав класса входят методы, позволяющие выполнять над файлами различные операции, например копирование, переименование, удаление, изменение атрибутов.

Для изменения имени файла класс CFile включает статический метод Rename, выполняющий функции этой команды. Метод нельзя использовать для переименования каталогов. В случае возникновения ошибки метод вызывает исключение.

Для удаления файлов в классе CFile включен статический метод Remove, позволяющий удалить указанный файл. Этот метод не позволяет удалять каталоги. Если удалить файл невозможно, то метод вызывает исключение.

Чтобы определить дату и время создания файла, его длину и атрибуты, предназначен статический метод GetStatus. Существует две разновидности метода - первый определен как виртуальный, а второй - как статический метод.

Виртуальная версия метода GetStatus определяет состояние открытого файла, связанного с данным объектом класса CFile. Этот метод вызывается только тогда, когда объект класса CFile создан и файл открыт.

Статическая версия метода GetStatus позволяет определить характеристики файла, не связанного с объектом класса CFile. Чтобы воспользоваться этим методом, необязательно предварительно открывать файл.

**Позиционирование**

Чтобы переместить указатель текущей позиции файла в новое положение, можно воспользоваться одним из следующих методов класса CFile - Seek, SeekToBegin, SeekToEnd. В состав класса CFile также входят методы, позволяющие установить и изменить длину файла, - GetLength, SetLength.

При открытии файла указатель текущей позиции файла находится в самом начале файла. Когда порция данных прочитана или записана, то указатель текущей позиции перемещается в сторону конца файла и указывает на данные, которые будут читаться или записываться очередной операцией чтения или записи в файл.

Чтобы переместить указатель текущей позиции файла в любое место, можно воспользоваться универсальным методом Seek. Он позволяет переместить указатель на определенное число байт относительно начала, конца или текущей позиции указателя.

Чтобы переместить указатель в начало или конец файла, наиболее удобно использовать специальные методы. Метод SeekToBegin перемещает указатель в начало файла, а метод SeekToEnd - в его конец.

Но для определения длины открытого файла совсем необязательно перемещать его указатель. Можно воспользоваться методом GetLength. Этот метод также возвращает длину открытого файла в байтах. Метод SetLength позволяет изменить длину открытого файла. Если при помощи этого метода размер файла увеличивается, то значение последних байт не определено.

Текущую позицию указателя файла можно определить с помощью метода GetPosition. Возвращаемое методом GetPosition 32-разрядное значение определяет смещение указателя от начала файла.

**Класс CStdioFile**

Класс CStdioFile наследован от базового класса CFile. Этот класс позволяет выполнять буферизированный ввод/вывод в текстовом и двоичном режиме. Для объектов класса CStdioFile можно вызывать практически все методы класса CFile.

В класс CStdioFile входит элемент данных m\_pStream, который содержит указатель на открытый файл. Если объект класса CStdioFile создан, но файл еще не открыт, либо закрыт, то m\_pStream содержит константу NULL.

Класс CStdioFile имеет три различных конструктора. Первый конструктор класса CStdioFile не имеет параметров. Этот конструктор только создает объект класса, но не открывает никаких файлов. Чтобы открыть файл, надо вызвать метод Open базового класса CFile.

Второй конструктор класса CStdioFile можно вызвать, если файл уже открыт и нужно создать новый объект класса CStdioFile и связать с ним открытый файл. Этот конструктор можно использовать, если файл был открыт стандартной функцией fopen. Параметр метода должен содержать указатель на файл, полученный вызовом стандартной функции fopen.

Третий конструктор можно использовать, если надо создать объект класса CStdioFile, открыть новый файл и связать его с только что созданным объектом.

Для чтения и записи в текстовый файл класс CStdioFile включает два новых метода: ReadString и WriteString. Первый метод позволяет прочитать из файла строку символов, а второй метод - записать.