**<https://docs.microsoft.com/ru-ru/windows/desktop/api/fileapi/nf-fileapi-createfilea->**

**Создание, открытие, закрытие и удаление файла**

Создание и открытие файла в Win32 производится одной функцией **CreateFile**.

HANDLE CreateFile ( LPCTSTR ipFileName, DWORD dwDesiredAccess.

DWORD dwShareMode. LPSECURITY\_ATTRIBUTES ipSecurityAttributes,

DWORD dwCreationDistribution, DWORD dwFlagsAndAttributes. HANDLE hTemplateFile):

Параметры данной функции имеют размер двойного слова. Их назначение следующее (параметры описаны в порядке, обратном их записи в стек):

* **lpFileName** – указатель на ASCIIZ-строку с именем (путем) открываемого или создаваемого файла;
* **dwDesiredAccess** – тип доступа к файлу:
  + **GENERICREAD = 80000000h** – доступ по чтению;
  + **GENERIC\_WRITE = 40000000h** – доступ по записи;
  + **GENERIC\_READ+GENERIC\_WRITE = 0C0000000h** – доступ по чтению-записи;
* **DwShareMode** – режим разделения файлов между разными процессами, данный параметр может принимать значения:
  + **0** – монополизация доступа к файлу;
  + **FILE\_SHARE\_READ = 00000001h**– этот флаг запрещает модификацию данных на этом устройстве всем  
    другим объектам ядра.
  + **FILESHAREWRITE = 00000002h –** запрещает чтение данных на этом устройстве всем другим объектам ядра.
  + **FILE\_SHARE\_READ+FILE\_SHARE\_WRITE = 00000003h** – другие процессы могут открывать файл по чтению-записи;
* **IpSecurityAttributes** – указатель на структуру **SecurityAttributes** (файл **winbase.h**), определяющую защиту связанного с файлом объекта ядра, при отсутствии защиты заносится NULL;
* **dwCreationDistribution** – определяет действия для случаев, когда файл существует или не существует (аналог этого параметра используется при вызове описанных выше функций MS DOS 6ch и 716ch), данный параметр может принимать значения:
  + **CREATE\_NEW= 1** – создать новый файл, если файл не существует; если файл существует, то функция завершается формированием ошибки;
  + **CREATE\_ALWAYS=2** – создать новый файл, если файл не существует; если он существует, то заместить новым;
  + **OPEN\_EXISTING=3** – открыть файл, если он существует; если файл не существует, то формируется ошибка;
  + **OPEN\_ALWAYS=4** – открыть файл при его существовании и создать его если файла нет;
  + **TRUNCATE\_EXISTING=5** – открыть файл с усечением его до нулевой длины; если файл не существует, то формируется ошибка;
* **DwFlagsAndAttributes** – флаги и атрибуты; этот параметр используется для задания характеристик создаваемого файла:
  + **FILE\_ATTRIBUTE\_READ0NLY=00000001h** – файл только для чтения;
  + **FILE\_ATTRIBUTE\_HIDDEN=00000002h** – скрытый файл;
  + **FILE\_ATTRIBUTE\_SYSTEM=00000004h** – системный файл;
  + **FILE\_ATrRIBUTE\_DIRECTORY=00000010h** – каталог;
  + **FILE ATTRIBUTE ARCHIVE=00000020h** – архивный ф'айл;
  + **FILE\_ATTRIBUTE\_N0RMAL=00000080h** – обычный файл для чтения-записи (этот атрибут нельзя комбинировать с другими);
  + **FILE\_ATTRIBUTE\_TEMPORARY=00000100h** – создается временный файл (преимущество этого файла в том, что система стремится не записывать этот файл на диск, а работать с ним в памяти; этот атрибут выгодно комбинировать с флагом FILE\_FLAG\_DELETE\_ON\_CLOSE, тогда после закрытия файла в программе он будет удален, не оставив следов на диске, иначе, как и bMS DOS, программе придется "подчищать" за собой содержимое диска);
  + **FILE\_FLAG\_WRITE\_THR0UGH=80000000h** – не использовать промежуточное кэширование при записи на диск, а все изменения записывать прямо на диск;
  + **FILE\_FLAG\_NO\_BUFFERING=20000000h** – не использовать средства буферизации операционной системы;
  + **FILE\_FLAG\_RANDOM\_ACCESS=10000000h** – прямой доступ к файлу (установка этого флага или флага FILE\_FLAG\_SEQUENTIAL\_SCAN позволяет оптимизировать системе процесс кэширования);
  + **FILE\_FLAG\_SEQUENTIAL\_SCAN=08000000h** – последовательный доступ к файлу;
  + **FILE\_FLAG\_DELETE\_0N\_CL0SE=04000000h** – удалить файл после его закрытия (см. описание атрибута FILEATTRIBUTETEMPORARY);
  + **FILE\_FLAG\_0VERLAPPED=40000000h** – асинхронный доступ к файлу (синхронность означает то, что программа, вызвавшая функцию для доступа к файлу, приостанавливается до тех пор, пока не закончит работу функция ввода-вывода);
* **hTemplateFile** – параметр используется только при создании нового файла, его значением является дескриптор другого существующего и предварительно открытого файла, а новый файл создается с теми же значениями атрибутов и флагов, что и у файла, дескриптор которого указан в параметре **hTemplateFile**.

При удачном завершении функция возвращает в регистре ЕАХ дескриптор нового файла. В случае неудачи функция возвращает в регистре ЕАХ значение **NULL**.

## Файловый ввод-вывод в Win32

В этом разделе будут приведены минимальные сведения, необходимые для выполнения простых операций с файлами. В отличие от MS DOS среда Win32 способна поддерживать несколько файловых систем. Главные требования к этим системам – иерархичность и соблюдение определенных правил присвоения имен каталогам и файлам.

Перечислим функции API Win32, имеющие отношение к работе с файловой системой. Полное их описание можно получить в MSDN.

| **Функция** | **Назначение** |
| --- | --- |
| **AreFileApisANSI** | Определение набора символов файла – ANSI или OEM |
| **CancelIo** | Отменить все ждущие обработки операции (I/O) ввода и вывода |
| **CloseHandle** | Закрыть открытый дескриптор файла |
| **CopyFile**, **CopyFileEx CopyProgressRoutine** | Копирование существующего файла в новый |
| **CreateDirectory**,**CreateDirectoryEx** | Создать каталог |
| **CreateFile** | Создать файл или объект специального типа |
| **DefineDosDevice** | Определить, переопределить или удалить имена устройства MS DOS |
| **DeleteFile** | Удалить файл |
| **FindClose** | Закрыть указанный поисковый дескриптор (см. функции **FindFirstFile** и**FindNextFile**) |
| **FindCloseChangeNotification** | Закрыть объект-уведомление об изменении файла |
| **FindFirstChangeNotification** | Создать объект-уведомление об изменении файла |
| **FindFirstFile**,**FindFirstFileEx**, **FindNextFile** | Поиск файлов |
| **FindNextChangeNotification** | Сброс объекта-уведомления в занятое состояние |
| **FlushFileBuffers** | Очистка буфера для указанного файла и запись всех буферизированных данных в файл |
| **GetBinaryType** | Определить, является ли файл исполняемым, и если это так, то для какой подсистемы – Win32, MS DOS, OS/2, POSIX и т. д. |
| **GetCurrentDirectory** | Получить текущий каталог |
| **GetDiskFreeSpace**,**GetDiskFreeSpaceEx** | Информация относительно указанного диска, включая количество свободного пространства на нем |
| **GetDriveType** | Определить тип диска – съемный, фиксированный, CD-ROM, электронный или сетевой |
| **GetFileAttributes**,**GetFileAttributesEx** | Получить атрибуты файла или каталога |
| **GetFilelnformationByHandle** | Найти информацию относительно указанного файла |
| **GetFileSize**, **GetFileSizeEx** | Получить размер указанного файла |
| **GetFileType** | Получить тип указанного файла |
| **GetFullPathName** | Получить полный путь и имя для указанного файла |
| **GetLogicalDrives**,**GetLogicalDriveStrings** | Определить доступные в настоящее время дисководы |
| **GetLongPathName** | Преобразовать указанный путь к его длинной форме |
| **GetShortPathName** | Получить псевдоним файла |
| **GetTempFileName** | Создать имя для временного файла |
| **GetTempPath** | Получить путь каталога для временных файлов |
| **LockFile**, **LockFileEx** | Блокировка файла |

Далее на примерах конкретных программ разберемся с тем, как использовать в программах на ассемблере наиболее интересные и часто применяемые функции из перечисленных выше для работы с файлами API Win32. В целях экономии места все примеры реализованы в виде консольных приложений.

Основное внимание уделено не полноте описания параметров для вызова той или иной функции и результатов ее работы (эту информацию можно найти в справочниках по функциям API), а деталям практической реализации файловых операций в программах на языке ассемблера. Для изучения подробностей работы функций API Win32 необходимо использовать какой-либо отладчик для Windows, например **TD32.EXE**.

### Обработка ошибок

Прежде чем рассматривать функции API Win32, относящиеся к файловому вводу-выводу, отметим, как можно выяснить причину их ошибочного завершения. Для этого Windows предоставляет функцию **GetLastError**.

DWORD GetLastError(void):

Для вызова функции **GetLastError** не нужно передавать никаких параметров. Эту функцию необходимо вызывать сразу после функции API Win32, успешность работы которой мы проверяем.

;………

push offset info

push hFile

call GetFilelnformationByHandle

call GetLastError;в регистре ЕАХ возвращается код ошибки

В регистре ЕАХ возвращается код ошибки. Расшифровать его можно с помощью файла**Winerror.h**, где вместе с кодами ошибок приведены короткие сообщения о причине их возникновения.