Файловые системы

Работа с файлами в Windows API

Работа с файлами в Windows API

Работа с логическими дисками и томами

Получение списка томов

- Функция

  DWORD GetLogicalDrives(void)

  Каждый установленный бит возвращаемого значения соответствует существующему в системе логическому устройству. Например, если в системе существуют диски A:, C: и D:, то возвращаемое функцией значение равно 13(десятичное).

- Функция

  DWORD GetLogicalDrivesStrings(DWORD cchBuffer, LPTSTR lpszBuffer)

  заполняет lpszBuffer информацией о корневом каталоге каждого логического диска в системе. В приведенном выше примере буфер будет заполнен символами

  A:\<null>C:\<null>D:\<null><null>

  параметр cchBuffer определяет длину буфера. Функция возвращает реальную длину буфера, необходимую для размещения всей информации.

Определение типа устройства

- Функция

  UINT GetDriveType(LPTSTR lpszRootPathName)

  В качестве параметра ей передается имя корневого каталога (напр. **A:**), а возвращаемое значение имеет следующие значения:

Идентификатор	Описание
0	Тип устройства определить нельзя
1	Корневой каталог не существует
DRIVE_REMOVABLE	Гибкий диск
DRIVE_FIXED	Жесткий диск
DRIVE_REMOTE	Сетевой диск
DRIVE_CDROM	Компакт диск
DRIVE_RAMDISK	RAM диск

Получение подробной информации о томе

- Для получения подробной информации о томе используется функция GetVolumeInformation (). Она заполняет параметры информацией об имени тома, названии файловой структуры, максимальной длине имени файла, дополнительных атрибутах тома, специфических для файловой структуры.
- Функция GetDiskFreeSpace () сообщает информацию о размерах сектора и кластера и о наличии свободных кластеров.

Работа с файлами в Windows API

Работа с каталогами и файлами

Работа с каталогами и файлами

GetCurrentDirectory	Получение текущего каталога
SetCurrentDirectory	Смена текущего каталога
GetSystemDirectory	Получение системного каталога
GetWindowsDirectory	Получение основного каталога системы
CreateDirectory	Создание каталога
RemoveDirectory	Удаление каталога
CopyFile	Копирование файла
MoveFile MoveFileEx	Перемещение или переименование файла
DeleteFile	Удаление файла

Создание и открытие файла

```
HANDLE CreateFile (
    LPCTSTR lpFileName,                 // pointer to name of the file
    DWORD dwDesiredAccess,              // access (read-write) mode
    DWORD dwShareMode,                  // share mode
    LPSECURITY_ATTRIBUTES lpSecurityAttributes,
                                        // pointer to security descriptor
    DWORD dwCreationDistribution,       // how to create
    DWORD dwFlagsAndAttributes,         // file attributes
    HANDLE hTemplateFile                // handle to file with attributes to copy
);
```

- В случае удачи функция *CreateFile ()* возвращает дескриптор открытого файла.
- В случае ошибки функция возвращает не NULL, а значение INVALID_HANDLE_VALUE.

Параметры dwDesiredAccess и dwShareMode

- Параметр **dwDesiredAccess** задает тип доступа к файлу. Можно определить флаги GENERIC_READ и GENERIC_WRITE а так же их комбинацию для разрешения чтения или записи в файл.
- Параметр **dwShareMode** определяет режим совместного использования файла различными процессами. Если этот параметр равен нулю, то никакой другой поток не сможет открыть этот же файл. Флаги FILE_SHARE_READ и FILE_SHARE_WRITE а так же их комбинация разрешают другим потокам осуществлять доступ к файлу для чтения или записи.

Параметр dwCreationDistribution

- Параметр **dwCreationDistribution** определяет действия функции в зависимости от того, существует ли уже файл с указанным именем:
  - CREATE_NEW – создает файл, если файл существует, то ошибка;
  - CREATE_ALWAYS – создает файл , если файл существует, то старый файл удаляется и новый создается;
  - OPEN_EXISTING – открывает существующий файл;
  - OPEN_ALWAYS – создает файл, если файл не существует, то создается новый файл;
  - TRUNCATE_EXISTING – открывает файл и урезает его до нулевой длины.

Параметр dwFlagsAndAttributes

- Параметр **dwFlagsAndAttributes** определяет атрибуты создания файла:
  - FILE_ATTRIBUTE_ARCHIVE, FILE_ATTRIBUTE_HIDDEN,
  - FILE_ATTRIBUTE_NORMAL, FILE_ATTRIBUTE_READONLY,
  - FILE_ATTRIBUTE_SYSTEM, FILE_ATTRIBUTE_TEMPORARY
- Атрибуты файла могут комбинироваться за исключением FILE_ATTRIBUTE_NORMAL, который всегда используется один.
- Вместе с атрибутами могут комбинироваться и флаги, задающие режим работы с файлом:
  - FILE_FLAG_NO_BUFFERING – не осуществлять кэширование и опережающее чтение;
  - FILE_FLAG_RANDOM_ACCESS – кэшировать как файл произвольного доступа;
  - FILE_FLAG_SEQUENTIAL_SCAN – кэшировать как файл последовательного доступа;
  - FILE_FLAG_WRITE_TROUGH – не буферизовать операцию записи, производить запись на диск немедленно;
  - FILE_FLAG_DELETE_ON_CLOSE – уничтожить файл при закрытии, полезно комбинировать с атрибутом FILE_ATTRIBUTE_TEMPORARY;
  - FILE_FLAG_OVERLAPPED – работа с файлом будет осуществляться асинхронно.

Вопрос

- Какие Вы видите различия между возможностями *CreateFile ()* и аналогичных функций ANSI C?

Получение размера файла

```
DWORD GetFileSize(
    HANDLE hFile, // дескриптор файла
    LPDWORD lpFileSizeHigh     // указатель на старшую часть
                               // 64-разрядного размера файла
);
```

- **Возвращаемое значение:** младшие 32 бита размера файла.

Получение типа файла

```
DWORD GetFileType (
   HANDLE hFile, // дескриптор файла
);
```

- **FILE_TYPE_CHAR** – символьный файл, обычно устройства LPT или консоли;
- **FILE_TYPE_DISK** – файл на диске;
- **FILE_TYPE_PIPE** – файл является именованным или анонимным каналом;
- **FILE_TYPE_UNKNOWN** – тип указанного файла неизвестен, или функция завершилась ошибкой.

Дескрипторы стандартного ввода-вывода

```
HANDLE GetStdHandle (
    DWORD nStdHandle // устройство стандарт. ввода-вывода
);

BOOL SetStdHandle (
    DWORD nStdHandle // устройство стандарт. ввода-вывода
    HANDLE hHandle     // дескриптор файла
);
```

- **STD_INPUT_HANDLE** – дескриптор стандартного устройства ввода данных (исходно – консольный буфер ввода);
- **STD_OUTPUT_HANDLE** – дескриптор устройства стандартного вывода (исходно – активный экранный буфер);
- **STD_ERROR_HANDLE** – дескриптор стандартной ошибки устройства (исходно – активный экранный буфер).

Сброс изменений файла из буфера на диск

- Так как ввод и вывод данных на диск в операционной системе Windows буферизуется, запись данных на диск может быть отложена до тех пор, пока система не освободится от выполнения текущей работы.
- С помощью функции *FlushFileBuffers ()* вы можете принудительно заставить операционную систему записать на диск все изменения для файла, дескриптор которого передается этой функции через единственный параметр:

```
BOOL FlushFileBuffers (HANDLE hFile);
```

- В случае успешного завершения функция возвращает значение TRUE, при ошибке – FALSE.

Сброс изменений всех файлов тома

- Если Вы хотите сбросить на диск изменения не одного файла, а всех файлов тома, то необходимо с помощью функции *CreateFile ()* открыть том как **\\.\<x>:** .
- Для выполнения сброса изменений тома программа должна иметь права доступа администратора.

Работа с атрибутами файла

```
BOOL SetFileAttributes(
    LPCTSTR lpFileName, // имя файла
    DWORD dwFileAttributes // атрибуты
);

DWORD GetFileAttributes(
    LPCTSTR lpFileName // имя файла или каталога
);
```

Ограничения функции SetFileAttributes

- **FILE_ATTRIBUTE_COMPRESSED** – чтобы установить сжатое состояние файла, используйте функцию *DeviceIoControl ()* с операцией **FSCTL_SET_COMPRESSION**;
- **FILE_ATTRIBUTE_DIRECTORY** – файл не может быть преобразован к каталог;
- **FILE_ATTRIBUTE_ENCRYPTED** – чтобы создать зашифрованный файл, используйте функцию *CreateFile ()* этим с атрибутом. Для конвертации существующего файла в зашифрованный, используйте функцию *EncryptFile ()*.
- **FILE_ATTRIBUTE_REPARSE_POINT** – чтобы связать точку монтирования с файлом или каталогом, используйте функцию *DeviceIoControl ()* с операцией **FSCTL_SET_REPARSE_POINT**;
- **FILE_ATTRIBUTE_SPARSE_FILE** – чтобы установить атрибут разреженности файла, используйте функцию *DeviceIoControl ()* с операцией **FSCTL_SET_SPARSE**.

Вопрос

- С помощью какой функции следует закрыть файл, который был открыт/создан с помощью *CreateFile ()* ?