Управление файлами и каталогами

В этом разделе вводятся основные функции, предназначенные для управления файлами и каталогами.

Управление файлами

Для управления файлами Windows предоставляет целый ряд функций, работа с которыми обычно не представляет сложности. Ниже описаны функции, с помощью которых можно удалять, копировать и переименовывать файлы. Существует также функция, предназначенная для создания имен временных файлов.

При удалении файла достаточно указать его имя. Вспомните, что все полные имена файлов начинаются с буквы диска или имени сервера. Открытый файл, вообще говоря, удалить невозможно (это допускается в Windows 9x и UNIX); попытка выполнения подобной операции закончится неудачей. У такого ограничения есть свои положительные стороны, поскольку оно предотвращает случайное удаление открытых файлов.

BOOL DeleteFile(LPCTSTR lpFileName)

Чтобы скопировать файл целиком, достаточно использовать одну функцию.

BOOL CopyFile(LPCTSTR lpExistingFileName, LPCTSTR lpNewFileName, BOOL fFailIfExists)

Функция CopyFile копирует существующий файл с заданным именем и присваивает копии указанное новое имя. В случае существования файла с таким же именем он будет заменен новым файлом только в том случае, если значением параметра fFailIfExists является FALSE.

Под управлением NT5 можно создать жесткую ссылку (hard link) для двух файлов, аналогичную жестким ссылкам в UNIX, используя для этого функцию CreateHardLink. Жесткие ссылки делают возможным существование файла под двумя различными именами. Заметьте, что в подобных случаях файл как таковой существует в единственном числе, и поэтому можно произвольно использовать любое из его имен, независимо от того, какое из них было использовано для открытия файла.

BOOL CreateHardLink(LPCTSTR lpFileName, LPCTSTR lpExistingFileName, BOOL lpSecurityAttributes)

Два первых аргумента имеют тот же смысл, что и в функции CopyFile, хотя и расположены в обратном порядке. Оба имени файла, новое и существующее, должны относиться к одному и тому же тому файловой системы, но могут соответствовать различным каталогам. Атрибуты защиты файла, если таковые имеются, применимы и к новому имени файла.

Если заглянуть в документацию Microsoft, то можно увидеть, что в структуре BY_HANDLE_FILE_INFO имеется поле "количество ссылок", и именно этот счетчик используется для определения того, может или не может быть удален данный файл. Функция **DeleteFile** удаляет имя из

каталога файловой системы, но сам файл не может быть удален до тех пор, пока значение счетчика "количество ссылок" не станет равным 0.

Гибких ссылок (soft link) в Windows не существует, хотя оболочки Windows (но не сама Windows), руководствующиеся при определении местоположения файла его содержимым, поддерживают ярлыки (shortcuts). Ярлыки предоставляют средства, подобные гибким ссылкам, но воспользоваться ими могут только пользователи оболочки.

Доступны две функции, позволяющие переименовывать, или "перемещать", файл. Эти же функции применимы и к каталогам. (Функции DeleteFile и CopyFile могут применяться только к файлам.)

BOOL MoveFile(LPCTSTR lpExistingFileName, LPCTSTR lpNewFileName)

BOOL MoveFileEx(LPCTSTR lpExistingFileName, LPCTSTR lpNewFileName, DWORD dwFlags)

Если новый файл уже существует, функция MoveFile завершается с ошибкой; в этом случае следует использовать функцию MoveFileEx. Заметьте, что суффикс "Ex" обычно применяется для обозначения усовершенствованных версий функций, обладающих расширенными (extended) функциональными возможностями.

Параметры

lpExistingFileName — указатель на строку, содержащую имя существующего файла *или* каталога.

lpNewFileName — указатель на строку, содержащую имя нового файла *или* каталога, которые, в случае функции MoveFile, до ее вызова существовать не должны. Новый файл может принадлежать другой файловой системе или находиться на другом диске, но новые каталоги обязательно должны находиться на одном и том же диске. Если значение этого параметра положить равным NULL, то существующий файл будет удален.

dwFlags — позволяет задавать следующие опции:

• MOVEFILE_REPLACE_EXISTING — разрешает замену существующего файла.

• MOVEFILE_WRITETHROUGH — используйте этот флаг, если необходимо, чтобы функция выполнила возврат лишь после того, как файл будет фактически перемещен на диске.

• MOVEFILE_COPY_ALLOWED — если новый файл находится на другом томе, перемещение осуществляется путем последовательного выполнения функций CopyFile и DeleteFile.

• MOVEFILE_DELAY_UNTIL_REBOOT — установка этого флага, использование которого является прерогативой администратора системы и который не может применяться совместно с флагом MOVEFILE_COPY_ALLOWED, приводит к тому, что фактическое перемещение файла будет осуществлено только после перезагрузки системы.

С перемещением (переименованием) файлов связаны некоторые ограничения.

• Использование групповых символов в именах файлов или каталогов запрещено. Необходимо указывать фактические имена.

Полные имена файлов в UNIX не включают имен дисков и серверов; корневой системный каталог обозначается обратной косой чертой. В то же время, функции для работы с файлами, входящие в библиотеку Microsoft C, поддерживают имена дисков, как того требуют соглашения Windows относительно именования файлов.

В UNIX отсутствует функция непосредственного копирования файлов. Вместо этого, чтобы выполнить команду cp, вы должны написать небольшую программу или использовать системный вызов system ().

В UNIX эквивалентом функции DeleteFile служит функция unlink, которая, к тому же, может удалять и каталоги.

В библиотеку C входят функции rename и remove, однако функция remove не позволяет присваивать файлу имя уже существующего файла или присваивать каталогу имя существующего непустого каталога. Новое имя может совпадать с именем существующего каталога только в том случае, если этот каталог пустой.

Управление каталогами

Создание и удаление каталогов осуществляется при помощи двух простых функций.

BOOL CreateDirectory(LPCTSTR lpPathName, LPSECURITY_ATTRIBUTES lpSecurityAttributes)

BOOL RemoveDirectory(LPCTSTR lpPathName)

lpPathName является указателем на завершающуюся нулевым символом строку, которая содержит путь к создаваемому или удаляемому каталогу. Как и в случае других функций, на данном этапе атрибуты защиты файла должны полагаться равными NULL; вопросы безопасности файлов и объектов рассматрим. Удалить можно только пустой каталог.

Как и в UNIX, у каждого процесса имеется текущий, или рабочий, каталог. Кроме того, для каждого диска поддерживается свой рабочий каталог. Программист может как устанавливать рабочий каталог, так и получать информацию о том, какой каталог в данный момент является текущим. Первая функция предназначена для установки каталогов.

BOOL SetCurrentDirectory(LPCTSTR lpPathName)

lpPathName определяет путь к новому текущему каталогу. Это может быть относительный путь или абсолютный полный путь, в начале которого указаны либо буква диска и двоеточие (например, D:), либо имя UNC (например, \\ACCTG_SERVER\PUBLIC).

Если в качестве пути к каталогу указывается только имя диска (например, A: или C:), то рабочим каталогом становится рабочий каталог данного диска. Например, если рабочие каталоги устанавливались в последовательности:

C:\MSDEV
INCLUDE
A:\MEMOS\TODO
C:

то результирующим рабочим каталогом будет:

C:\MSDEV\INCLUDE

Следующая функция возвращает абсолютный полный путь к текущему каталогу, помещая его в буфер, предоставляемый программистом:

DWORD GetCurrentDirectory(DWORD cchCurDir, LPTSTR lpCurDir)

Возвращаемое значение: длина строки, содержащей путь доступа к текущему каталогу, или требуемый размер буфера, если буфер не достаточно велик; в случае ошибки — нуль.

cchCurDir — размер буфера, содержащего имя каталога, который определяется количеством символов (а не байт). Размер буфера должен рассчитываться с учетом завершающего нулевого символа строки.

lpCurDir является указателем на буфер, предназначенный для получения строки, содержащей путь.

Заметьте, что в случае если размер буфера оказался недостаточным для того, чтобы в нем уместилась вся строка пути, функция возвратит значение, указывающее на требуемый размер буфера. Поэтому при тестировании успешности выполнения функции следует проверять два условия: равно ли возвращаемое значение нулю и не превышает ли оно значение, заданное аргументом cchCurDir.

Подобный метод возврата строк и их длины широко распространен в Windows и требует внимательной обработки результатов. Программа 2.6 иллюстрирует типичный фрагмент кода, реализующего эту логику. Аналогичная логика реализуется и в других примерах. Вместе с тем, указанный метод применяется не всегда. Некоторые функции возвращают булевские значения, а параметр размера в них используется дважды: перед вызовом функции его значение устанавливается равным размеру буфера, а затем изменяется функцией. В качестве одного из многих возможных примеров можно привести функцию LookupAccountName, с которой вы встретитесь в главе 15.

Альтернативный подход, демонстрируемый в программе 15.4 функцией GetFileSecurity, заключается в выделении буферной памяти в промежутке между двумя вызовами функций. Первый вызов обеспечивает получение длины строки, на основании чего и выделяется память, тогда как второй — получение самой строки. Самым простым подходом в данном случае является выделение памяти для строки, насчитывающей MAX_PATH символов.

Пример: печать текущего каталога

Программа 2.6 реализует очередную версию команды UNIX pwd. Размер буфера определяется значением параметра MAX_PATH, однако проверка ошибок все равно предусмотрена, чтобы проиллюстрировать работу функции GetCurrentDirectory.

Программа 2.6. pwd: печать текущего каталога

```
/* Глава 2. pwd – вывод на печать содержимого рабочего каталога. */
#include "EvryThng.h"
#define DIRNAME_LEN MAX_PATH + 2

int _tmain(int argc, LPTSTR argv[]) {
 TCHAR pwdBuffer [DIRNAME_LEN];
 DWORD LenCurDir;
 LenCurDir = GetCurrentDirectory(DIRNAME_LEN, pwdBuffer);
 if (LenCurDir == 0) ReportError(_T("Не удается получить путь."), 1, TRUE);
 if (LenCurDir > DIRNAME_LEN) ReportError(_T("Слишком длинный путь."), 2, FALSE);
 PrintMsg(GetStdHandle(STD_OUTPUT_HANDLE), pwdBuffer);
 return 0;
}
```

Резюме

Наряду с функциями, предназначенными для обработки символов, в Windows поддерживается полный набор функций, обеспечивающих управление файлами и каталогами. Кроме того, вы можете создавать переносимые, обобщенные приложения, которые могут быть рассчитаны на работу как с символами ASCII, так и с символами Unicode.

Функции Windows во многом напоминают их аналоги в UNIX и библиотеке C, хотя различия между ними также очевидны. В приложении Б представлена таблица, в которой сведены функции Windows, UNIX и библиотеки C и показано, в чем они соответствуют друг другу, а в чем заметно отличаются.