Unit 7: The Input/Output System

7.3. Win32 File and Directory Management

Win32 I/O
File and Directory Management

- Win32 provides a number of straightforward functions to manage files
  - Delete, copy, rename files
  - Create temporary file names

```
BOOL DeleteFile (LPCTSTR lpszFileName)
```

- Absolute pathnames start with drive letter or server name!
- It is not possible to delete an open file in Windows NT (but it is possible in Windows95)
- UNIX unlink() decrements link count (but does not necessarily delete file)

Moving files

```
BOOL CopyFile( LPCTSTR lpszExistingFile,
               LPCTSTR lpszNewFile,
               BOOL fFailIfExists );
```

- Copies the named existing file and assigns new name
- An existing file will be replaced only if
  fFailIfExists == FALSE
- DeleteFile() and CopyFile do not work for directories
- Win32 does not support any file linking
  (but NTFS and POSIX subsystem do)

Moving Files (contd.)

```
BOOL MoveFile (LPCTSTR lpszExisting,
               LPCTSTR lpszNew);

BOOL MoveFileEx( LPCTSTR lpszExisting,
               LPCTSTR lpszNew, DWORD fdwFlags);
```

- MoveFile() fails if the new file already exists
  (use MoveFileEx() for existing files)
  - Windows 95 does not implement MoveFileEx()
  - New files can be on different drives / directories
  - New directories must be on same drive
  - lpszNew == NULL : existing file is deleted
- fdwFlags:
  - MOVEFILE_REPLACE_EXISTING – replace existing destination file
  - MOVEFILE_COPY_ALLOWED – destination may be on different volume

Directory Management

```
BOOL CreateDirectory( LPCTSTR lpszPath,
                      LPSECURITY_ATTRIBUTES lpsa );

BOOL RemoveDirectory( LPCTSTR lpszPath );
```

- lpszPath points to null-terminated string with the name of the target directory
  - Only an empty directory can be rmoved
  - lpsa == NULL will create a null-ACL for the new directory

Directory Management (contd.)

```
BOOL SetCurrentDirectory( LPCTSTR lpszCurDir );

DWORD GetCurrentDirectory( DWORD cchCurDir,
                           LPTSTR lpszCurDir  );
```

- Each process has current working directory
  - Each individual drive keeps working directory
- GetCurrentDirectory:
  - ccCurDir is size of buffer in characters (!)
  - Buffer to small: GetCurrentDirectory() returns required size (!) or zero on failure
  - Call GetCurrentDirectory twice: first to obtain size of buffer, next to obtain value
    (or use MAX_PATH constant)

Directory Searching

```
HANDLE FindFirstFile( LPCTSTR lpszSearchFile,
LPWIN32_FIND_DATA lpffd );
```

- Search a directory for files that satisfy a specified name pattern
  - Search handles must be obtained via FindFirstFile() and closed via FindClose()
  - FindFirstFile() examines subdirectories and file names
  - Return of INVALID_HANDLE_VALUE indicates failure
- Parameters:
  - lpszSearchFile points to directory/pathname that can contain wildcard characters (? and *; no regular expressions)
  - lpffd points to data structure with access information

WIN32_FIND_DATA structure

FILE_ATTRIBUTE_ARCHIVE, FILE_ATTRIBUTE_COMPRESSED,
FILE_ATTRIBUTE_DIRECTORY, FILE_ATTRIBUTE_ENCRYPTED,
FILE_ATTRIBUTE_HIDDEN,.FILE_ATTRIBUTE_NORMAL,
FILE_ATTRIBUTE_OFFLINE, FILE_ATTRIBUTE_READONLY,
FILE_ATTRIBUTE_REPARSE_POINT, FILE_ATTRIBUTE_SPARSE_FILE,
FILE_ATTRIBUTE_SYSTEM, FILE_ATTRIBUTE_TEMPORARY

```
typedef struct _WIN32_FIND_DATA { // wfd
   DWORD dwFileAttributes;
   FILETIME ftCreationTime;
   FILETIME ftLastAccessTime;
   FILETIME ftLastWriteTime;
   DWORD nFileSizeHigh;
   DWORD nFileSizeLow;
   DWORD dwReserved0;
   DWORD dwReserved1;
   TCHAR cFileName[ MAX_PATH ];
   TCHAR cAlternateFileName[ 14 ];
} WIN32_FIND_DATA;
```

Does not contain path-portion of name

DOS 8.3 name

Directory Searching (contd.)

```
BOOL FindNextFile( HANDLE hFindFile,
                   LPWIN32_FIND_DATA lpffd );

BOOL FindClose( HANDLE hFindFile );
```

- FindNextFile() returns FALSE in case of invalid arguments or if no more matching files are found
  - GetLastError() returns ERROR_NO_MORE_FILES
- Use FindClose() to close search handle
  - CloseHandle() will raise an exception
- GetFileInformationByHandle() obtains same info...
- Programs must do wildcard expansion on their own
  - MS-DOS shell (cmd.exe) does not expand wildcards (sh.exe does)

More File and Directory Attributes

```
BOOL GetFileTime( HANDLE hFiles,
                  LPFILETIME lpftCreation,
                  LPFILETIME lpftLastAccess,
                  LPFILETIME lpftLastWrite );
```

- File times are 64-bit unsigned integers
  (time in 100 ns units (10^7 / s) since January 1, 1601)
- FileTimeToSystemTime()/SystemTimeToFileTime()
  convert into years down to milliseconds (and vice versa)
- CompareFileTime(), SetFileTime()
- NTFS supports all three file times
- FAT is accurate only for LastAccess-time

File Attributes (contd.)

```
DWORD GetFileAttributes( LPCTSTR lpszFileName )
```

- Returns file attribute or 0xFFFFFFFF in case of failure
- Attributes include:
  - FILE_ATTRIBUTE_DIRECTORY
  - FILE_ATTRIBUTE_NORMAL
  - FILE_ATTRIBUTE_READONLY
  - FILE_ATTRIBUTE_TEMPORARY
- SetFileAttribute() changes those attributes for a file