第九课　文件系统

一、文件系统基本概念

Windows操作系统管理磁盘数据的方式：FAT、FAT32、NTFS

磁道：由若干扇区组成
扇区：512字节

文件系统的最小管理单位：簇——连续的若干扇区
    FAT32 : 1簇 = 32扇区 = 16K字节
    NTFS : 1簇 = 8扇区 = 4K字节
    文件存储时，以簇为单位占用磁盘空间，
    即使只有1个字节，也要占用1簇空间。

二、目录

1. 获取磁盘驱动器信息

```
DWORD WINAPI GetLogicalDrives (void);
```

返回当前可用磁盘驱动器掩码位。

```
高位 <------------------------ 低位
XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX
              ... PONMLKJI HGFEDCBA
```

0：无此磁盘驱动器
1：有此磁盘驱动器

2. 获取系统驱动器

```
DWORD WINAPI GetLogicalDriveStrings (
    DWORD  nBufferLength, // 缓冲区大小(以字符为单位)
    LPTSTR lpBuffer       // 缓冲区指针
);
```

成功返回字符串长度，失败返回0。

3. 获取当前目录

```
DWORD GetCurrentDirectory (
    DWORD  nBufferLength, // 缓冲区大小(以字符为单位)
    LPTSTR lpBuffer       // 缓冲区指针
);
```

成功返回字符串长度，失败返回0。

4. 设置当前目录

~~~~~~~~~~~~~~~

```
BOOL SetCurrrentDirectory (
    LPCTSTR lpPathName // 当前目录路径
);
```

成功返回TRUE，失败返回FALSE。

5. 获取WINDOWS目录

```
UINT GetWindowsDirectory (
    LPSTR lpBuffer, // 缓冲区指针
    UINT uSize      // 缓冲区大小(以字符为单位)
);
```

成功返回字符串长度，失败返回0。

6. 获取系统(system32)目录

```
UINT GetSystemDirectory (
    LPTSTR lpBuffer, // 缓冲区指针
    UINT   uSize     // 缓冲区大小(以字符为单位)
);
```

成功返回字符串长度，失败返回0。

7. 获取临时目录

```
DWORD GetTempPath (
    DWORD  nBufferLength, // 缓冲区大小(以字符为单位)
    LPTSTR lpBuffer       // 缓冲区指针
);
```

成功返回字符串长度，失败返回0。

8. 创建目录

```
BOOL CreateDirectory (
    LPCTSTR lpPathName, // 目录路径
    LPSECURITY_ATTRIBUTES lpSecurityAttributes
                        // 安全属性，置NULL
);
```

成功返回TRUE，失败返回FALSE。

9. 删除空目录

```
BOOL RemoveDirectory (
    LPCTSTR lpPathName // 目录路径
);
```
~~~~~~~~~~~~~~~

成功返回TRUE，失败返回FALSE。

10. 目录/文件修改

```
BOOL MoveFile (
    LPCTSTR lpExistingFileName, // 当前路径
    LPCTSTR lpNewFileName       // 目标路径
);
```

成功返回TRUE，失败返回FALSE。

注意：不能跨驱动器移动目录，但是可以跨驱动器移动文件。

范例：WinDir

三、文件

1. 创建/打开文件

```
HANDLE CreateFile (
    LPCTSTR lpFileName,             // 文件路径
    DWORD   dwDesiredAccess,        // 访问方式
    DWORD   dwShareMode,            // 共享方式
                                    // 其它进程以何种方式打开此文件
    LPSECURITY_ATTRIBUTES lpSecurityAttributes,
                                    // 安全属性(NULL)
    DWORD   dwCreationDisposition, // 创建/打开方式
    DWORD   dwFlagsAndAttributes,  // 文件属性
    HANDLE  hTemplateFile           // 文件句柄模板
                                    // 磁盘文件设NULL同步传输，
                                    // 打印机同步或异步传输
);
```

成功返回文件句柄，失败返回INVALID_HANDLE_VALUE(-1)。

dwDesiredAccess为以下值的位或：
```
    0             - 质询(判断文件是否存在)
    GENERIC_READ  - 读取
    GENERIC_WRITE - 写入
```

dwShareMode为以下值的位或：
```
    FILE_SHARE_DELETE - 允许其它进程删除
    FILE_SHARE_READ   - 允许其它进程读取
    FILE_SHARE_WRITE  - 允许其它进程写入
```

dwCreationDisposition取值：
```
    CREATE_NEW        - 不存在就创建，已存在就失败
    CREATE_ALWAYS     - 不存在就创建，已存在就删除原文件再创建
    OPEN_EXISTING     - 已存在就打开，不存在就失败
    OPEN_ALWAYS       - 已存在就打开，不存在就创建新文件再打开
    TRUNCATE_EXISTING - 已存在就先清空再打开，不存在就失败
```

2. 写文件

~~~~~~~~~

```
BOOL WriteFile (
    HANDLE       hFile,                    // 文件句柄
    LPCVOID      lpBuffer,                 // 数据缓冲区
    DWORD        nNumberOfBytesToWrite,    // 期望写入的字节数
    LPDWORD      lpNumberOfBytesWritten,   // 实际写入的字节数,
                                           // 可为NULL
    LPOVERLAPPED lpOverlapped              // NULL(同步传输)
);
```

成功返回TRUE，失败返回FALSE。

3. 获取文件大小
~~~~~~~~~~~~~~

```
DWORD GetFileSize (
    HANDLE  hFile,          // 文件句柄
    LPDWORD lpFileSizeHigh // 文件字节数的高32位，可为NULL
);
```

返回文件字节数的低32位。

32位 - 4G
64位 - 16E，实际2T

小知识：数量单位的中英文表示法
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

英文表示法：

K, Kilo   - 10^3 - 2^10
M, Mega   - 10^6 - 2^20
G, Giga   - 10^9 - 2^30
T, Tera   - 10^12 - 2^40
P, Peta   - 10^15 - 2^50
E, Exa   - 10^18 - 2^60
B, Bronto - 10^21 - 2^70

中文表示法：

个   - 10^0
十   - 10^1
百   - 10^2
千   - 10^3
万   - 10^4
亿   - 10^8
兆   - 10^12
京   - 10^16
垓   - 10^20
秭   - 10^24
穰   - 10^28
沟   - 10^32
涧   - 10^36
正   - 10^40
载   - 10^44

```
极       - 10^48
恒河沙   - 10^52
阿僧只   - 10^56
那由他   - 10^60
不可思议 - 10^64
无量     - 10^68
大数     - 10^72
```

4. 读文件

```
BOOL ReadFile (
    HANDLE       hFile,                // 文件句柄
    LPVOID       lpBuffer,             // 数据缓冲区
    DWORD        nNumberOfBytesToRead, // 期望读取的字节数
    LPDWORD      lpNumberOfBytesRead,  // 实际读取的字节数
    LPOVERLAPPED lpOverlapped          // NULL(同步传输)
);
```

成功返回TRUE，失败返回FALSE。

5. 设置文件指针

```
DWORD SetFilePointer (
    HANDLE hFile,                 // 文件句柄
    LONG   lDistanceToMove,       // 偏移量低32位
    PLONG  lpDistanceToMoveHigh, // 偏移量高32位(输入输出),
                                  // 可为NULL
    DWORD  dwMoveMethod           // 偏移量相对位置
);
```

成功返回新位置的低32位(高32位通过lpDistanceToMoveHigh输出)，
失败返回-1。

```
dwMoveMethod取值：
    FILE_BEGIN   - 从文件头
    FILE_CURRENT - 从当前位置
    FILE_END     - 从文件尾
```

四、拷贝、移动和删除
---

1. 拷贝文件

```
BOOL CopyFile (
    LPCTSTR lpExistingFileName, // 源路径
    LPCTSTR lpNewFileName,      // 目标路径
    BOOL    bFailIfExists       // TRUE存在就失败,
                                // FALSE存在就覆盖
);
```

成功返回TRUE，失败返回FALSE。

2. 移动(改名)目录/文件

~~~~~~~~~~~~~~~~~~~~~~

```
BOOL MoveFile (
    LPCTSTR lpExistingFileName, // 当前路径
    LPCTSTR lpNewFileName       // 目标路径
);
```

成功返回TRUE，失败返回FALSE。

注意：不能跨驱动器移动目录，但是可以跨驱动器移动文件。

3. 删除文件

```
BOOL DeleteFile (
    LPCTSTR lpFileName // 路径
);
```

成功返回TRUE，失败返回FALSE。

即使打开的文件也可以删除。

五、属性
----

1. 设置目录/文件属性

```
BOOL SetFileAttributes  (
    LPCTSTR lpFileName,      // 目录/文件路径
    DWORD   dwFileAttributes // 目录/文件属性
);
```

成功返回TRUE，失败返回FALSE。

dwFileAttributes为以下值的位或：

```
    FILE_ATTRIBUTE_ARCHIVE    - 归档
    FILE_ATTRIBUTE_COMPRESSED - 压缩
    FILE_ATTRIBUTE_OFFLINE    - 离线
    FILE_ATTRIBUTE_DIRECTORY  - 目录
    FILE_ATTRIBUTE_ENCRYPTED  - 加密
    FILE_ATTRIBUTE_HIDDEN     - 隐藏
    FILE_ATTRIBUTE_NORMAL     - 普通
    FILE_ATTRIBUTE_READONLY   - 只读
    FILE_ATTRIBUTE_SYSTEM     - 系统
    FILE_ATTRIBUTE_TEMPORARY  - 临时
```

注意：增加属性应该先获取属性，位或上新属性，再一并设置。

2. 获取目录/文件属性

```
DWORD GetFileAttributes (
    LPCTSTR lpFileName // 目录/文件路径
);
```
~~~~~~~~~~~~~~~~~~~~~~

成功返回文件属性，失败返回-1。

3. 获取目录/文件扩展属性(创建时间、最后访问时间、最后修改时间)

```
BOOL GetFileAttributesEx (
    LPCTSTR                 lpFileName,
    GET_FILEEX_INFO_LEVELS fInfoLevelId,
    LPVOID                  lpFileInformation
);
```

成功返回TRUE，失败返回FALSE。

六、遍历
----

1. 查找第一个目录/文件

```
HANDLE FindFirstFile (
    LPCTSTR           lpFileName,    // 查找路径
    LPWIN32_FIND_DATA lpFindFileData // 查找信息
);

typedef struct _WIN32_FIND_DATA {
    DWORD    dwFileAttributes;       // 属性
    FILETIME ftCreationTime;         // 创建时间
    FILETIME ftLastAccessTime;       // 最后访问时间
    FILETIME ftLastWriteTime;        // 最后修改实现
    DWORD    nFileSizeHigh;          // 文件字节数高32位
    DWORD    nFileSizeLow;           // 文件字节数低32为
    DWORD    dwReserved0;            // 保留
    DWORD    dwReserved1;            // 保留
    TCHAR    cFileName[MAX_PATH];    // 目录/文件名
    TCHAR    cAlternateFileName[14]; // 8.3格式的目录/文件名
}   WIN32_FIND_DATA, *PWIN32_FIND_DATA, *LPWIN32_FIND_DATA;
```

成功返回查找句柄，用于后续函数调用的参数，
失败返回INVALID_HANDLE_VALUE(-1)

2. 查找下一个目录/文件

```
BOOL WINAPI FindNextFile (
    HANDLE hFindFile, // 查找句柄(FindFirstFile函数的返回值)
    LPWIN32_FIND_DATA lpFindFileData // 查找信息
);
```

成功返回TRUE，失败返回FALSE。
GetLastError函数返回ERROR_NO_MORE_FILES表示没有下一个目录/文件了。

范例：WinFile