## Path

|  |  |
| --- | --- |
| **API** | **Description** |
| **DWORD** **GetModuleFileNameA**(  **HMODULE** *inModuleHandle*,  **LPSTR** *outFileName*,  **DWORD** *inSize*  ); | Get the full path (including file name) for the specified inModuleHandle.  If inModuleHandle = NULL, the path of the executable file of the current process is retrieved.  Eg: outFileName = "C:\Module\module.exe" |
| **DWORD** **GetFullPathNameA**(  **LPCSTR** *inFileName*,  **DWORD** *inPathLength*,  **LPSTR** *outFullPath*,  **LPSTR\*** *outFileName*  );  Eg [here](https://docs.microsoft.com/en-us/windows/desktop/api/fileapi/nf-fileapi-getfullpathnamea). | Get the full path (including file name) of the specified inFileName.  Eg:  If the current dir is "C:\Module\" and inFileName = "module.exe", then:   * outFullPath = "C:\Module\module.exe" * outFileName = "module.exe" |
| **LPCSTR** **PathFindFileNameA**(  **LPCSTR** *inPath*  );  Note: include *shlwapi.lib* and *shlwapi.h* | Search a path for a file name.  Eg: if inPath = "C:\Module\module.exe", then retval = "module.exe". |
| **BOOL** **PathRemoveFileSpecA**(  **LPSTR** *ioPath*  );  Note: include *shlwapi.lib* and *shlwapi.h* | Remove the file name and backslash from a path, if they are present.  Eg: if ioPath = "C:\Module\module.exe", then ioPath = "C:\Module" |
| **LPSTR** **PathAddBackslashA**(  **LPSTR** *ioPath*  );  Note: include *shlwapi.lib* and *shlwapi.h* | Add a backslash to the end of a string to create the correct syntax for a path.  If the source path already has a trailing backslash, no backslash will be added.  Eg: if ioPath = "C:\Module", then ioPath = "C:\Module\" |
| COMBINE:   1. **GetModuleFileNameA()** 2. **PathRemoveFileSpecA()** 3. **PathAddBackslashA()** | Get the directory path of the specified file.  Eg:  Input: "C:\Module\module.exe"  Output: "C:\Module\"  Another way:  // Get EXE's full path  char exeFullPath[MAX\_PATH] = {0};  GetModuleFileNameA(NULL, exeFullPath, sizeof(exeFullPath));  // Get EXE's directory path  char\* end = strrchr(exeFullPath, '\\');  char exeDirPath[MAX\_PATH] = {0};  strncpy\_s(exeDirPath, exeFullPath, end-exeFullPath+1); |
| **LPCSTR** **PathFindExtensionA**(  **LPCSTR** *inPath*  );  Note: include *shlwapi.lib* and *shlwapi.h* | Search a path for an extension.  Eg: if inPath = "C:\Module\module.exe", then retval = ".exe". |
| **BOOL** **PathFileExistsA**(  **LPCSTR** *inPath*  );  Note: include *shlwapi.lib* and *shlwapi.h* | Determines whether a path to a file system object (file or folder) is valid. |

## Directory Processing

|  |  |
| --- | --- |
| **API** | **Description** |
| **HANDLE** **CreateDirectoryA**(  **LPCSTR** inPathName,  **LPSECURITY\_ATTRIBUTES** inSecurityAttributes  ); |  |

## File Processing

|  |  |
| --- | --- |
| **API** | **Description** |
| **HANDLE** **CreateFileA**(  **LPCSTR** *inFileName*,  **DWORD** *inDesiredAccess*,  **DWORD** *inShareMode*,  **LPSECURITY\_ATTRIBUTES** *inSecurityAttributes*,  **DWORD** *inCreationDisposition*,  **DWORD** *inFlagsAndAttributes*,  **HANDLE** *inTemplateFile*  ); | Create or open a file or I/O device (directory, physical disk, volume, console buffer, tape drive, communications resource, mailslot, pipe, etc.).  **Notes**:  1. Writing to a text file in appending mode (default) requires setting the file pointer to the ending or beginning (default) of the file. To set this pointer to the ending of the file, use [SetFilePointer()](http://msdn.microsoft.com/en-us/library/windows/desktop/aa365541.aspx).  2. Windows uses a CR/LF combination to signify the end of line, so we need to write \r\n (instead of just \n) to break line to show up correctly in, e.g., Notepad.  3. After open a file or I/O device, must close it using [CloseHandle()](https://docs.microsoft.com/en-us/windows/win32/api/handleapi/nf-handleapi-closehandle). |
| **BOOL WriteFileA**(  **HANDLE** *inFileName,*  **LPCVOID** *inBuffer,*  **DWORD** *inNumberOfBytesToWrite,*  **LPDWORD** *outNumberOfBytesWritten,*  **LPOVERLAPPED** *inOverlapped*  ); | Writes data to the specified file or I/O device.  **Eg:**  DWORD bytesWritten = 0;  if (!WriteFile(m\_LogFileHandler, log, strlen(log), &bytesWritten, NULL)) {  return;  } |
| **DWORD GetFileAttributesA**(  **LPCSTR** *inFileName*  );  **Note**: network share? Volume or mounted folder? | Retrieve file system attributes for a specified file or directory.  Eg: Dir? Hidden? Archive? Encrypted? Readonly? Offline? Etc.  **Other uses:**  Check whether a file or directory exists. Return -1 ifpath doesn’t exist.  Note: Able to work for both local path and network path (\\ or with IP addr), but these paths must be mounted folders, not volumes (or network name). |
| **HANDLE** **FindFirstFileA**(  **LPCSTR** *inFileName*,  **LPWIN32\_FIND\_DATAA** *outFindFileData*  ); | Search a directory for a file or subdirectory with a name that matches a specific name (or partial name if wildcards are used).  To specify additional attributes to use in a search, use the [FindFirstFileEx](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-findfirstfileexa) function.  To perform this operation as a transacted operation, use the [FindFirstFileTransacted](https://docs.microsoft.com/windows/desktop/api/winbase/nf-winbase-findfirstfiletransacteda) function. |
| **BOOL** **FindNextFileA**(  **HANDLE** *inFindFile*,  **LPWIN32\_FIND\_DATAA** *outFindFileData*  ); | Continue a file search from a previous call to the [FindFirstFile](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-findfirstfilea), [FindFirstFileEx](https://docs.microsoft.com/windows/desktop/api/fileapi/nf-fileapi-findfirstfileexa), or [FindFirstFileTransacted](https://docs.microsoft.com/windows/desktop/api/winbase/nf-winbase-findfirstfiletransacteda) functions. |
| **BOOL** **DeleteFileA**(  **LPCSTR** *inFileName*  ); | Deletes an existing file.  To perform this operation as a transacted operation, use the [DeleteFileTransacted](https://docs.microsoft.com/windows/desktop/api/winbase/nf-winbase-deletefiletransacteda) function. |

## ini File

|  |  |
| --- | --- |
| **API** | **Description** |
| **DWORD** **GetPrivateProfileSection**(  **LPCWSTR** *inAppName*,  **LPWSTR** *outReturnedString*,  **DWORD** *inSize*,  **LPCWSTR** *inpFileName*  ); | Retrieves all the keys and values for the specified section of an initialization file. |
| **DWORD** **GetPrivateProfileString**(  **LPCTSTR** *inAppName*,  **LPCTSTR** *inKeyName*,  **LPCTSTR** *inDefault*,  **LPTSTR** *outReturnedString*,  **DWORD** *inSize*,  **LPCTSTR** *inFileName*  ); | Retrieves a string from the specified section in an initialization file. |
| **UINT GetPrivateProfileInt(**  **LPCTSTR** *inAppName*,  **LPCTSTR** *inKeyName*,  **INT** *inDefaul*t,  **LPCTSTR** *inFileName*  **);** | Retrieves an integer associated with a key in the specified section of an initialization file. |
| **WritePrivateProfile\*\*\*** |  |

## Error Handling

|  |  |
| --- | --- |
| **API** | **Description** |
| **DWORD** **GetLastError**();  Notes:   * Convert error code to error message string [here](https://stackoverflow.com/a/17387176). * Better alternative to this API [here](https://www.codeproject.com/Articles/1025/Do-Not-Call-GetLastError). | Retrieves the calling thread's last-error code value.  The last-error code is maintained on a per-thread basis. Multiple threads do not overwrite each other's last-error code. |

## Thread

|  |  |
| --- | --- |
| **API** | **Description** |
| Critical Section | <https://docs.microsoft.com/en-us/windows/win32/sync/critical-section-objects>  <https://docs.microsoft.com/en-us/windows/win32/sync/using-critical-section-objects> |
| Mutex | <https://docs.microsoft.com/en-us/windows/win32/sync/mutex-objects> |
| Semaphore | <https://docs.microsoft.com/en-us/windows/win32/sync/semaphore-objects> |

## Time

|  |  |
| --- | --- |
| **API** | **Description** |
| **void** **GetLocalTime**(  **LPSYSTEMTIME** *outSystemTime*  ); | Retrieves the current local date and time.  Note: To retrieve the current date and time in Coordinated Universal Time (UTC) format, use the [GetSystemTime](https://docs.microsoft.com/windows/desktop/api/sysinfoapi/nf-sysinfoapi-getsystemtime) function. |
| **LONG** **CompareFileTime**(  **const FILETIME** \**inFileTime1*,  **const FILETIME** \**inFileTime2*  ); | Compares two file times. |

## Timer

|  |  |
| --- | --- |
| **API** | **Description** |
| **UINT\_PTR** **SetTimer**(  **HWND** *inWnd*,  **UINT\_PTR** *inIDEvent*,  **UINT** *inElapse*,  **TIMERPROC** *inTimerFunc*  ); | Creates a timer with the specified time-out value.  <https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-settimer> |
|  |  |
|  |  |

## Type Converting

|  |  |
| --- | --- |
| **API** | **Description** |
| **int** **MultiByteToWideChar**(  **UINT** *inCodePage*,  **DWORD** *inFlags*,  **\_In\_NLS\_string\_(cbMultiByte)LPCCH** *inMultiByteStr*,  **int** *inMultiByte*,  **LPWSTR** *outWideCharStr*,  **int** *outWideCharCount*  ); | Maps a character string to a UTF-16 (wide character) string. The character string is not necessarily from a multibyte character set.  <https://docs.microsoft.com/en-us/windows/win32/api/stringapiset/nf-stringapiset-multibytetowidechar> |
|  |  |
|  |  |