LazFileUtils

From Free Pascal wiki

Comparing Filenames

```
function CompareFilenames(const Filename1, Filename2: string): integer; overload;

function CompareFilenamesIgnoreCase(const Filename1, Filename2: string): integer;

function CompareFileExt(const Filename, Ext: string; CaseSensitive: boolean): integer; overload;

function CompareFileExt(const Filename, Ext: string): integer; overload;

function CompareFilenameStarts(const Filename1, Filename2: string): integer;

function CompareFilenames(Filename1: PChar; Len1: integer; Filename2: PChar; Len2: integer): integer; overload;

function CompareFilenamesP(Filename1, Filename2: PChar; IgnoreCase: boolean = false): integer; // false = use default
```

Directories

```
function DirPathExists(DirectoryName: string): boolean;

function DirectoryIsWritable(const DirectoryName: string): boolean;
```

File Names

```
function ExtractFileNameOnly(const AFilename: string): string;

function FilenameIsAbsolute(const TheFilename: string):boolean;
function FilenameIsWinAbsolute(const TheFilename: string):boolean;

function FilenameIsUnixAbsolute(const TheFilename: string):boolean;

function ForceDirectory(DirectoryName: string): boolean;

procedure CheckIfFileIsExecutable(const AFilename: string);

procedure CheckIfFileIsSymlink(const AFilename: string);

function FileIsExecutable(const AFilename: string): boolean;

function FileIsSymlink(const AFilename: string): boolean;

function FileIsHardLink(const AFilename: string): boolean;

function FileIsReadable(const AFilename: string): boolean;

function FileIsWritable(const AFilename: string): boolean;

function FileIsText(const AFilename: string): boolean;

function FileIsText(const AFilename: string; out FileReadable: boolean):
boolean;

function FilenameIsTrimmed(const TheFilename: string): boolean;

function FilenameIsTrimmed(StartPos: PChar; NameLen: integer): boolean;

function TrimFilename(const AFilename: string): string;

function ResolveDots(const AFilename: string): string;

Procedure ForcePathDelims(Var FileName: string);

Function GetForcedPathDelims(Const FileName: string): String;

function CleanAndExpandFilename(const Filename: string): string; // empty
string returns current directory

function CleanAndExpandDirectory(const Filename: string): string; // empty
string returns current directory

function TrimAndExpandFilename(const Filename: string; const BaseDir: string
= ''): string; // empty string returns empty string

function TrimAndExpandDirectory(const Filename: string; const BaseDir: string
= ''): string; // empty string returns empty string

function TryCreateRelativePath(const Dest, Source: String; UsePointDirectory:
boolean; AlwaysRequireSharedBaseFolder: Boolean; out RelPath: String):
Boolean;

function CreateRelativePath(const Filename, BaseDirectory:
string;UsePointDirectory: boolean = false; AlwaysRequireSharedBaseFolder:
Boolean = True): string;

function FileIsInPath(const Filename, Path: string): boolean;

function AppendPathDelim(const Path: string): string;

function ChompPathDelim(const Path: string): string;
```

Search Paths

```
function CreateAbsoluteSearchPath(const SearchPath, BaseDirectory: string):
string;

function CreateRelativeSearchPath(const SearchPath, BaseDirectory: string):
string;

function MinimizeSearchPath(const SearchPath: string): string;

function FindPathInSearchPath(APath: PChar; APathLen: integer; SearchPath:
PChar; SearchPathLen: integer): PChar; overload;

function FindPathInSearchPath(const APath, SearchPath: string): integer;
overload;
```

File Operations

```
function FileExistsUTF8(const Filename: string): boolean;

function FileAgeUTF8(const FileName: string): Longint;

function DirectoryExistsUTF8(const Directory: string): Boolean;

function ExpandFileNameUTF8(const FileName: string; {const} BaseDir: string =
''): string;
```

```
function FindFirstUTF8(const Path: string; Attr: Longint; out Rslt:
TSearchRec): Longint;

function FindNextUTF8(var Rslt: TSearchRec): Longint;

procedure FindCloseUTF8(var F: TSearchrec); inline;
```

File Attributes

```
function FileSetDateUTF8(const FileName: String; Age: Longint): Longint;

function FileGetAttrUTF8(const FileName: String): Longint;

function FileSetAttrUTF8(const Filename: String; Attr: longint): Longint;

function DeleteFileUTF8(const FileName: String): Boolean;

function RenameFileUTF8(const OldName, NewName: String): Boolean;

function FileSearchUTF8(const Name, DirList : String; ImplicitCurrentDir :
Boolean = True): String;

function FileIsReadOnlyUTF8(const FileName: String): Boolean;
```

Get, Set, Create, Remove or Force Directories

```
function GetCurrentDirUTF8: String;

function SetCurrentDirUTF8(const NewDir: String): Boolean;

function CreateDirUTF8(const NewDir: String): Boolean;

function RemoveDirUTF8(const Dir: String): Boolean;

function ForceDirectoriesUTF8(const Dir: string): Boolean;
```

File Open or Created

```
function FileOpenUTF8(Const FileName : string; Mode : Integer) : THandle;

function FileCreateUTF8(Const FileName : string) : THandle; overload;

function FileCreateUTF8(Const FileName : string; Rights: Cardinal) : THandle;
overload;
```

```
Function FileCreateUtf8(Const FileName : String; ShareMode : Integer; Rights
: Cardinal) : THandle; overload;
```

File Attributes and Size

```
function FileSizeUtf8(const Filename: string): int64;

function GetFileDescription(const AFilename: string): string;

function ReadAllLinks(const Filename: string; {%H-}ExceptionOnError:
boolean): string; // if a link is broken returns ''

function TryReadAllLinks(const Filename: string): string; // if a link is
broken returns Filename

function GetShellLinkTarget(const FileName: string): string;
```

For Debugging

```
function DbgSFileAttr(Attr: LongInt): String;

TPhysicalFilenameOnError = (pfeException,pfeEmpty,pfeOriginal); // Get
FileNameOnError
```

Getting File Names

```
// Getting file name except for Unix
function GetPhysicalFilename(const Filename: string; OnError:
TPhysicalFilenameOnError): string;

// for Unix
function GetUnixPhysicalFilename(const Filename: string; ExceptionOnError:
boolean): string; // if a link is broken returns ''
```

Getting paths

```
function GetAppConfigDirUTF8(Global: Boolean; Create: boolean = false):
string;

function GetAppConfigFileUTF8(Global: Boolean; SubDir: boolean = false;
CreateDir: boolean = false): string;

function GetTempFileNameUTF8(const Dir, Prefix: String): String;
```

```
// UNC paths
function IsUNCPath(const {%H-}Path: String): Boolean;

function ExtractUNCVolume(const {%H-}Path: String): String;

function ExtractFileRoot(FileName: String): String;

// Darwin (macOS) Paths
function GetDarwinSystemFilename(Filename: string): string;

function GetDarwinNormalizedFilename(Filename: string; nForm:Integer=2):
string;

// Windows Paths
function SHGetFolderPathUTF8(ID :  Integer) : String;
```

Commandline Parameters

```
procedure SplitCmdLineParams(const Params: string; ParamList: TStrings;
ReadBackslash: boolean = false);

function StrToCmdLineParam(const Param: string): string;

function MergeCmdLineParams(ParamList: TStrings): string;
```

Cache

```
TInvalidateFileStateCacheEvent = procedure(const Filename: string); var
OnInvalidateFileStateCache: TInvalidateFileStateCacheEvent = nil;

procedure InvalidateFileStateCache(const Filename: string = ''); inline;
```

See also

- Reference for LazFileUtils unit.

Retrieved from "http://wiki.freepascal.org/index.php?title=LazFileUtils&oldid=153346"
Categories:

- Lazarus
- LazUtils