fileutil

From Lazarus wiki

Jump to navigationJump to search

| **English (en)** | français (fr) | русский (ru) | 中文（中国大陆） (zh_CN) |

The Lazarus **FileUtil** unit contains functions and procedures to maintain compatibility with Delphi's FileUtil unit. File routines that specifically deal with UTF8 file names should go into the LazFileUtils unit.

List of functions

```
   // file attributes and states
 function CompareFilenames(const Filename1, Filename2: string): integer;
inline;
 function CompareFilenamesIgnoreCase(const Filename1, Filename2: string):
integer; inline;
 function CompareFilenames(const Filename1, Filename2: string; ResolveLinks:
boolean): integer;
 function CompareFilenames(Filename1: PChar; Len1: integer; Filename2: PChar;
Len2: integer; ResolveLinks: boolean): integer;
 function FilenameIsAbsolute(const TheFilename: string):boolean; inline;
 function FilenameIsWinAbsolute(const TheFilename: string):boolean; inline;
 function FilenameIsUnixAbsolute(const TheFilename: string):boolean; inline;
 procedure CheckIfFileIsExecutable(const AFilename: string); inline;
 procedure CheckIfFileIsSymlink(const AFilename: string); inline;
 function FileIsReadable(const AFilename: string): boolean; inline;
 function FileIsWritable(const AFilename: string): boolean; inline;
 function FileIsText(const AFilename: string): boolean; inline;
 function FileIsText(const AFilename: string; out FileReadable: boolean):
boolean; inline;
 function FileIsExecutable(const AFilename: string): boolean; inline;
 function FileIsSymlink(const AFilename: string): boolean; inline;
 function FileIsHardLink(const AFilename: string): boolean; inline;
 function FileSize(const Filename: string): int64; overload; inline;
 function GetFileDescription(const AFilename: string): string; inline;
 function ReadAllLinks(const Filename: string; ExceptionOnError: boolean):
string; // if a link is broken returns ''
 function TryReadAllLinks(const Filename: string): string; // if a link is
broken returns Filename
```

```
    // directories
  function DirPathExists(const FileName: String): Boolean; inline;
  function ForceDirectory(DirectoryName: string): boolean; inline;
  function DeleteDirectory(const DirectoryName: string; OnlyChildren:
boolean): boolean;
  function ProgramDirectory: string;
  function DirectoryIsWritable(const DirectoryName: string): boolean; inline;

    // filename parts
  function ExtractFileNameOnly(const AFilename: string): string; inline;
  function ExtractFileNameWithoutExt(const AFilename: string): string;
  function CompareFileExt(const Filename, Ext: string; CaseSensitive:
boolean): integer; overload; inline;
  function CompareFileExt(const Filename, Ext: string): integer; overload;
inline;
  function FilenameIsPascalUnit(const Filename: string): boolean;
  function AppendPathDelim(const Path: string): string; inline;
  function ChompPathDelim(const Path: string): string; inline;
  function TrimFilename(const AFilename: string): string; inline;
  function CleanAndExpandFilename(const Filename: string): string; inline;
  function CleanAndExpandDirectory(const Filename: string): string; inline;
  function CreateAbsoluteSearchPath(const SearchPath, BaseDirectory: string):
string;
  function CreateRelativePath(const Filename, BaseDirectory: string;
UsePointDirectory: boolean = false;  AlwaysRequireSharedBaseFolder: Boolean =
True): string; inline;
  function CreateAbsolutePath(const Filename, BaseDirectory: string): string;
  function FileIsInPath(const Filename, Path: string): boolean;
  function FileIsInDirectory(const Filename, Directory: string): boolean;

  // file search
  type
    TSearchFileInPathFlag = (
      sffDontSearchInBasePath,
      sffSearchLoUpCase
      );
    TSearchFileInPathFlags = set of TSearchFileInPathFlag;
  const
    AllDirectoryEntriesMask = '*';

  function GetAllFilesMask: string; inline;
  function GetExeExt: string; inline;
  function SearchFileInPath(const Filename, BasePath, SearchPath, Delimiter:
```

```
string; Flags: TSearchFileInPathFlags): string;
 function SearchAllFilesInPath(const Filename, BasePath, SearchPath,
Delimiter: string; Flags: TSearchFileInPathFlags): TStrings;
 function FindDiskFilename(const Filename: string): string;
 function FindDiskFileCaseInsensitive(const Filename: string): string;
 function FindDefaultExecutablePath(const Executable: string; const BaseDir:
string = ''): string;

 type
  TFileIterator = class
  private
    FPath: String;
    FLevel: Integer;
    FFileInfo: TSearchRec;
    FSearching: Boolean;
    function GetFileName: String;
  public
    procedure Stop;
    function IsDirectory: Boolean;
  public
    property FileName: String read GetFileName;
    property FileInfo: TSearchRec read FFileInfo;
    property Level: Integer read FLevel;
    property Path: String read FPath;
    property Searching: Boolean read FSearching;
  end;

  TFileFoundEvent = procedure (FileIterator: TFileIterator) of object;
  TDirectoryFoundEvent = procedure (FileIterator: TFileIterator) of object;
  TDirectoryEnterEvent = procedure (FileIterator: TFileIterator) of object;

  TFileSearcher = class(TFileIterator)
  private
    FMaskSeparator: char;
    FFollowSymLink: Boolean;
    FOnFileFound: TFileFoundEvent;
    FOnDirectoryFound: TDirectoryFoundEvent;
    FOnDirectoryEnter: TDirectoryEnterEvent;
    FFileAttribute: Word;
    FDirectoryAttribute: Word;
    procedure RaiseSearchingError;
  protected
    procedure DoDirectoryEnter; virtual;
    procedure DoDirectoryFound; virtual;
```

```
    procedure DoFileFound; virtual;
  public
    constructor Create;
    procedure Search(const ASearchPath: String; ASearchMask: String = '';
      ASearchSubDirs: Boolean = True; CaseSensitive: Boolean = False);
  public
    property MaskSeparator: char read FMaskSeparator write FMaskSeparator;
    property FollowSymLink: Boolean read FFollowSymLink write FFollowSymLink;
    property FileAttribute: Word read FFileAttribute write FFileAttribute
default faAnyfile;
    property DirectoryAttribute: Word read FDirectoryAttribute write
FDirectoryAttribute default faDirectory;
    property OnDirectoryFound: TDirectoryFoundEvent read FOnDirectoryFound
write FOnDirectoryFound;
    property OnFileFound: TFileFoundEvent read FOnFileFound write
FOnFileFound;
    property OnDirectoryEnter: TDirectoryEnterEvent read FOnDirectoryEnter
write FOnDirectoryEnter;
  end;

 function [[FindAllFiles]]( const SearchPath: String; SearchMask: String =
''; SearchSubDirs: Boolean = True): TStringList;
 function FindAllDirectories(const SearchPath: string;  SearchSubDirs:
Boolean = True): TStringList;

  // Copy a file or a whole directory tree
 function [[CopyFile]](const SrcFilename, DestFilename: string; Flags:
TCopyFileFlags=[cffOverwriteFile]): boolean;
 function [[CopyFile]](const SrcFilename, DestFilename: string; PreserveTime:
boolean): boolean;
 function CopyDirTree(const SourceDir, TargetDir: string; Flags:
TCopyFileFlags=[]): Boolean;

   // file actions
 function ReadFileToString(const Filename: string): string;
 function GetTempFilename(const Directory, Prefix: string): string; inline;

   // basic functions similar to the RTL but working with UTF-8 instead of
the system encoding
   // AnsiToUTF8 and UTF8ToAnsi need a widestring manager under Linux, BSD,
MacOSX
   // but normally these OS use UTF-8 as system encoding so the
widestringmanager is not needed.
 function NeedRTLAnsi: boolean; inline;// true if system encoding is not UTF-
```

```
8
 procedure SetNeedRTLAnsi(NewValue: boolean); inline;
 function UTF8ToSys(const s: string): string; inline;// as UTF8ToAnsi but
more independent of widestringmanager
 function SysToUTF8(const s: string): string; inline;// as AnsiToUTF8 but
more independent of widestringmanager
 function ConsoleToUTF8(const s: string): string; inline;// converts OEM
encoded string to UTF8 (used with some Windows specific  functions )
 function UTF8ToConsole(const s: string): string; inline;// converts UTF8
string to console encoding (used by Write, WriteLn)

   // file operations
 function FileExistsUTF8(const Filename: string): boolean; inline;
 function FileAgeUTF8(const FileName: string): Longint; inline;
 function DirectoryExistsUTF8(const Directory: string): Boolean; inline;
 function ExpandFileNameUTF8(const FileName: string): string; inline;
 function ExpandUNCFileNameUTF8(const FileName: string): string;
 function ExtractShortPathNameUTF8(Const FileName : String) : String;
 function FindFirstUTF8(const Path: string; Attr: Longint; out Rslt:
TSearchRec): Longint; inline;
 function FindNextUTF8(var Rslt: TSearchRec): Longint; inline;
 procedure FindCloseUTF8(var F: TSearchrec); inline;
 function FileSetDateUTF8(const FileName: String; Age: Longint): Longint;
inline;
 function FileGetAttrUTF8(const FileName: String): Longint; inline;
 function FileSetAttrUTF8(const Filename: String; Attr: longint): Longint;
inline;
 function DeleteFileUTF8(const FileName: String): Boolean; inline;
 function RenameFileUTF8(const OldName, NewName: String): Boolean; inline;
 function FileSearchUTF8(const Name, DirList : String; ImplicitCurrentDir :
Boolean = True): String; inline;
 function FileIsReadOnlyUTF8(const FileName: String): Boolean; inline;
 function GetCurrentDirUTF8: String; inline;
 function SetCurrentDirUTF8(const NewDir: String): Boolean; inline;
 function CreateDirUTF8(const NewDir: String): Boolean; inline;
 function RemoveDirUTF8(const Dir: String): Boolean; inline;
 function ForceDirectoriesUTF8(const Dir: string): Boolean; inline;
 function FileOpenUTF8(Const FileName : string; Mode : Integer) : THandle;
inline;
 function FileCreateUTF8(Const FileName : string) : THandle; overload;
inline;
 function FileCreateUTF8(Const FileName : string; Rights: Cardinal) :
THandle; overload; inline;
```

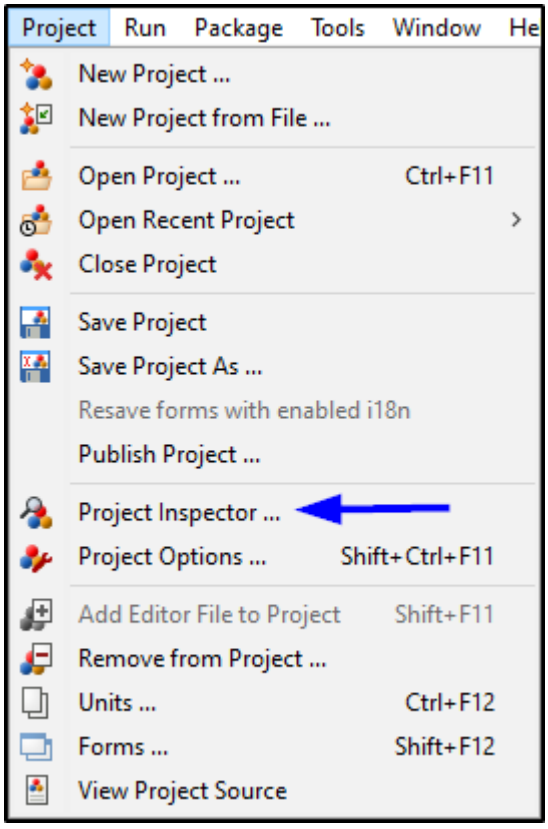
```
    // environment
  function ParamStrUTF8(Param: Integer): string; inline;
  function GetEnvironmentStringUTF8(Index: Integer): string; inline;
  function GetEnvironmentVariableUTF8(const EnvVar: string): String; inline;
  function GetAppConfigDirUTF8(Global: Boolean; Create: boolean = false):
string; inline;
  function GetAppConfigFileUTF8(Global: Boolean; SubDir: boolean = false;
CreateDir: boolean = false): string; inline;

    // other
  function SysErrorMessageUTF8(ErrorCode: Integer): String; inline;
```

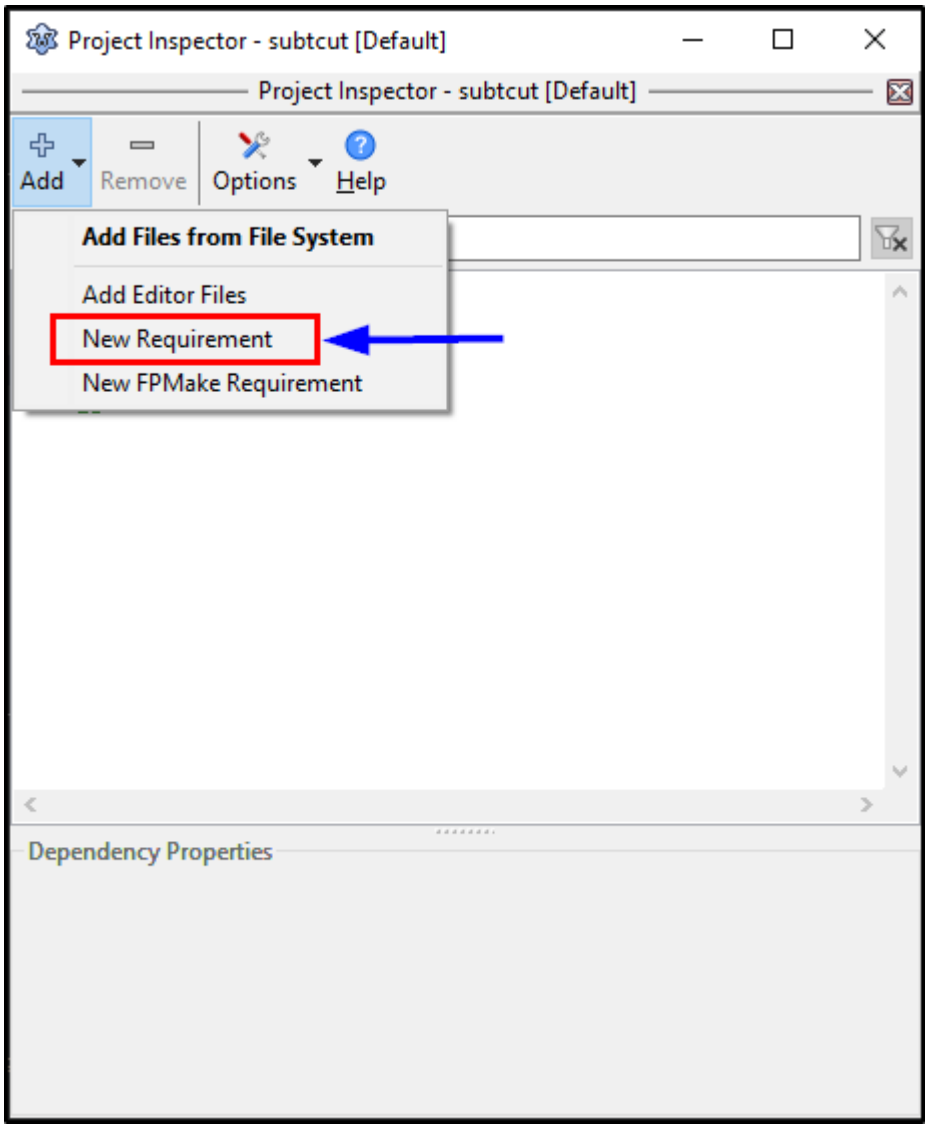
How to use the unit

To enable FileUtil in a project, just add LazUtils into required package. Follow this steps:

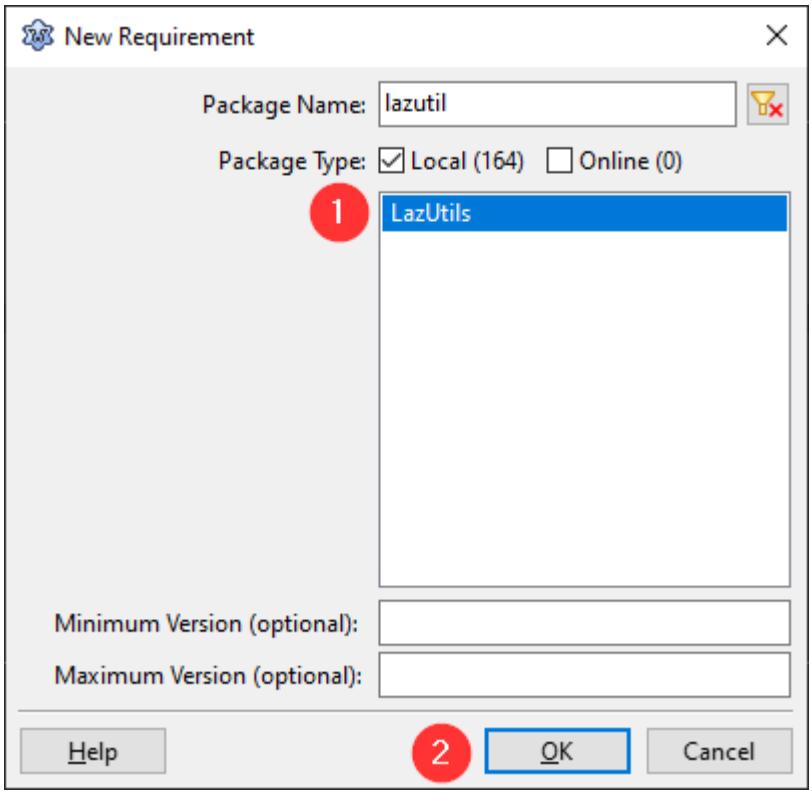
1- Go to Lazarus IDE Menu "Project > Project Inspector".



2- In the "Project Inspector" dialog window, click "Add > New Requirement".



3- In the "New Requirement" dialog window, find LazUtils package then click OK.



4- You can now add *FileUtil* in a *Uses* clause.

Categories:

- Lazarus
- fileutil