File Paths in Pascal

Here's the translation of the Go code to Pascal, formatted in Markdown suitable for Hugo:

The `SysUtils` unit provides functions to parse and construct file paths in a way that is portable between operating systems; `dir/file` on Linux vs. `dir\file` on Windows, for example.

```
program FilePaths;

uses
  SysUtils;

begin
  // ExtractFilePath should be used to construct paths in a
  // portable way. It takes a path and returns the directory part.
  var p := ExtractFilePath('dir1' + PathDelim + 'dir2' + PathDelim + 'filename');
  WriteLn('p: ', p);

  // You should always use PathDelim instead of concatenating
  // '/'s or '\'s manually. In addition to providing portability,
  // Pascal will also normalize paths by removing superfluous separators.
  WriteLn(ExtractFilePath('dir1' + PathDelim + PathDelim + 'filename'));
  WriteLn(ExtractFilePath('dir1' + PathDelim + '..' + PathDelim + 'dir1' + PathDelim + 'filename'));

  // ExtractFilePath and ExtractFileName can be used to split a path to the
  // directory and the file.
  WriteLn('ExtractFilePath(p): ', ExtractFilePath(p));
  WriteLn('ExtractFileName(p): ', ExtractFileName(p));

  // We can check whether a path is absolute.
  WriteLn(IsPathAbsolute('dir' + PathDelim + 'file'));
  WriteLn(IsPathAbsolute(PathDelim + 'dir' + PathDelim + 'file'));

  var filename := 'config.json';

  // Some file names have extensions following a dot. We
  // can split the extension out of such names with ExtractFileExt.
  var ext := ExtractFileExt(filename);
  WriteLn(ext);

  // To find the file's name with the extension removed,
  // use ChangeFileExt.
  WriteLn(ChangeFileExt(filename, ''));

  // ExtractRelativePath finds a relative path between a base and a
  // target. It returns an empty string if the target cannot
  // be made relative to base.
  var rel := ExtractRelativePath('a' + PathDelim + 'b', 'a' + PathDelim + 'b' + PathDelim + 't' + PathDelim + 'file');
  WriteLn(rel);

  rel := ExtractRelativePath('a' + PathDelim + 'b', 'a' + PathDelim + 'c' + PathDelim + 't' + PathDelim + 'file');
  WriteLn(rel);
end.
```

To run the program, save it as `file_paths.pas` and use the Pascal compiler (for example, Free Pascal):

```
$ fpc file_paths.pas
$ ./file_paths
p: dir1/dir2/
dir1/
dir1/
ExtractFilePath(p): dir1/dir2/
ExtractFileName(p):
False
True
.json
config
t\file
..\c\t\file
```

Note that the output may vary slightly depending on the operating system, as Pascal will use the appropriate path delimiter for the current system.

In this Pascal version:

- We use the `SysUtils` unit, which provides file path manipulation functions.
- `ExtractFilePath` is used instead of `filepath.Dir`.
- `ExtractFileName` is used instead of `filepath.Base`.
- `IsPathAbsolute` checks if a path is absolute.
- `ExtractFileExt` gets the file extension.
- `ChangeFileExt` is used to remove the file extension.
- `ExtractRelativePath` finds a relative path between a base and target.

Pascal doesn't have an exact equivalent to Go's `filepath.Join`, so we manually concatenate paths using the `PathDelim` constant, which provides the correct path delimiter for the current operating system.

Remember that Pascal's file path functions may behave slightly differently from Go's, especially in edge cases, so always test thoroughly when porting code between languages.

Prev
Line Filters in Pascal

Next
Command Line Subcommands in Pascal