FINDFIRST/FINDNEXT AND THE POWER OF HABITS

BY HEIKO ROMPEL

At work I recently needed a tool for searching the company's server for certain files and showing some information about them.

As soon as I arrived home I opened Lazarus and started programming. For informations about how to program a recursive file search I used a search engine and found this site:
http://www.entwickler-ecke.de/topic_nach+Dateien+suchen_1107,0.html
Here is the content:

```
Procedure FindFiles (aPath, aFindMask: String; aWithSub: Boolean; aResult: tStrings);
Var   FindRec: tSearchRec;
Begin
  // Wenn die Stringliste nil ist oder aPath oder aFind nicht angegeben ist dann raus
  // If the string list is nil or the aPath or aFind is not found then exit
  If (aPath = '') or (aFindMask = '') or Not Assigned (aResult) Then Exit;

  // Wenn am Ende der Pfadangabe noch kein \ steht, dieses hinzufügen
  // If at the end of the path there is no \ add it

  // (Oder die Funktion  IncludeTrailingPathDelimiter aus der Unit SysUtils.pas verwenden)
  // Otherwise use the function IncludeTrailingPathDelimiter from unit SysUtils.pas

  If aPath[Length (aPath)] <> '\' Then aPath := aPath + '\';

  // Im aktuellen Verzeichnis nach der Datei suchen
  // Search for data
  If FindFirst (aPath + aFindMask, faAnyFile, FindRec) = 0 Then
    Repeat
      If (FindRec.Name <> '.') and (FindRec.Name <> '..') Then
        // ...Ergebnis in die Stringlist einfügen
        // ...Fill in result into Stringlist
        aResult.Add (aPath + FindRec.Name);
    Until FindNext (FindRec) <> 0;

  FindClose (FindRec);

  // Wenn nicht in Unterverzeichnissen gesucht werden soll dann raus
  // If not ment to search in other data then exit
  If Not aWithSub Then Exit;

  // In Unterverzeichnissen weiter suchen
  // Search in other Dir's
  If FindFirst (aPath + '*.*', faAnyFile, FindRec) = 0 Then
    Repeat
      If (FindRec.Name <> '.') and (FindRec.Name <> '..') Then
        // Feststellen, ob es sich um ein Verzeichnis handelt
        // Make sure it is a Dir
        If Boolean (FindRec.Attr and faDirectory) Then
          // Funktion erneut aufrufen, um Verzeichnis zu durchsuchen (Rekursion)
          // Call function again, to search the Dir (recursion)
          FindFiles (aPath + FindRec.Name, aFindMask, aWithSub, aResult);
    Until FindNext (FindRec) <> 0;

   FindClose (FindRec);
End;
```

I had already used this code in Delphi before - so I was able to adapt it quickly to my current project.

The first test run took place on a local drive and it was a success. Then came a test run on the **NAS** * at home and that was a success also.
*(*Network-attached storage (NAS) is a file-level computer data storage server connected to a computer network providing data access to a heterogeneous group of clients. NAS is specialized for serving files either by its hardware, software, or configuration. It is often manufactured as a computer appliance – a purpose-built specialized computer.[nb 1] NAS systems are networked appliances which contain one or more hard disk drives, often arranged into logical, redundant storage containers or RAID.*)
So I sent the EXE-File via email to my workplace. Next day at work there was a surprise:
No files found.
Maybe it was because of the UNC path (**`https://de.wikipedia.org/wiki/Uniform_Naming_Convention`**)?
Back at home, I looked at the full paths and on the NAS there were UNC paths but they didn't cause problems. So the problem had to have other origins. Next day - back at work - I went to work with the source code and I ran the program in **DEBUG-mode**.

What on earth could be the problem?
Simple- as ever - special characters like ä,ö,ü !!!
(*In 2015 you still have to bother with them.*)
The almost endless paths had special characters at the start section. I found out I had to use the UTF8-function of Lazarus/FreePascal.
Fortunately changing to the UTF8 wasn't very complicated in this case.

Here are the things you to change in the code:

```
// Is required for the UTF8-function
uses  FileUtil;
```

old:

```
// Search the current directory for the file
 If FindFirst (aPath + aFindMask, faAnyFile,
FindRec) = 0 Then
```

new:

```
// Search the current directory for the file
If FindFirstUTF8(aPath + aFindMask,
faAnyFile, FindRecord) = 0 Then
```

old:

```
Until FindNext (FindRec) <> 0;
FindClose (FindRec);
```

new:

```
Until FindNextUTF8 (FindRecord) <> 0;
FileUtil.FindCloseUTF8(FindRecord);
```

Usually you simply forget that you can't use an old code without problems. But I have to admit that I never encountered the problems described above in Delphi before.

Perhaps this story saves somebody from an irritating job.

A special thanks to my son for the translation.

Heiko Rompel
Germany