# SAA Interface

Windows Toolkit

BY

JAMES G. HASSLACHER, JR.

## SAA Interface

Microsoft guts as viewed by Win32

Microsoft provides access to its features only to a GUI, and the tool set that was used to build the GUI.  To allow Rexx the ability to manipulate and examine these features, it was necessary to contruct a utility with that functionality.  The utility is hooked into Rexx through the SAA Interface for robustness that a command line interface cannot provide.

## SAA Interface

Features

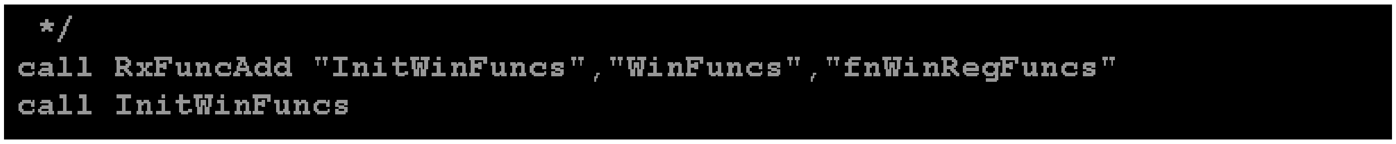
The most important are:

- Access to the Windows' Registry
  - GET, SET, DELete, and FIND
- Extended access to the file system
  - File attributes
  - Win32 executable properties
- Manipulation of the PATH environment variable
- Execution of system commands
  - By-pass "browser lockout" of USER32.dll

# SAA Interface

Invocation

The WinFuncs.dll must be in a directory named by the PATH, and compiled for the Rexx interpreter that will be executing.  Before using any of its function, the following lines must be in the Rexx program:



```
 */
call RxFuncAdd "InitWinFuncs","WinFuncs","fnWinRegFuncs"
call InitWinFuncs
```

## Windows Toolkit

Registry Tutorial

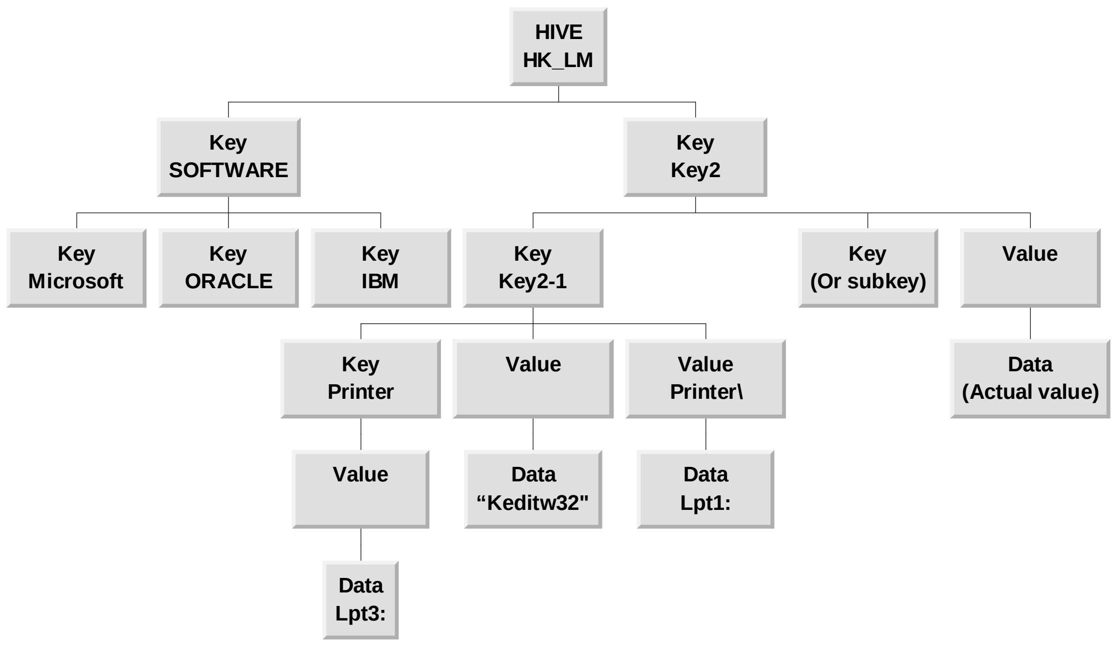
The Registry is a hierarchial tree comprised of four (4) distinct elements:

- Hive
- Key
- Value*
- Not named, but most often refered to as Data

* The name Value is very confusing.  It is not the value of the key, but does contain the value.  The best way to think of it is as the last subkey name with looser naming conventions.

# Windows Toolkit

Registry Tutorial



## Windows Toolkit

Registry ambiguity

What does the Value named by
\HK_LM\Key2\Key2-1\Printer\
Contain: lpt1:, or lpt3?

## Windows Toolkit

Registry Function Syntax

Initial syntax:

o RegGet(fully qualified Value name)
o RegSet(full Value name, data [, suggested data type])
o RegDel(full Value/Key name)
o RegFind(string [, search object type [, starting position]])

Syntax in reaction to the ambiguity:

o RegAPI(action, Hive name, Key Name [,Value Name [,data [,suggested data type])

## Windows Toolkit

Ambiguity resolution

- There are two ways to deal with this:
  - Name it in one string, lettting the ambiguity remain:
    - Data = RegGet("\HK_LM\Key2\Key2-1\Printer\")
  - Force the user to parse it for you:
    - Data = RegAPI("Get","HK_LM","Key2\Key2-1","Printer\")
    - Or
    - Data =RegAPI("Get","HK_LM","Key2\Key2-1\Printer","")

## Windows Toolkit

Registry Demo

- Test.cmd
- Test RegSet.cmd
- Notepad removal demo.CMD

# Windows Toolkit

File System Routines

## ■ Extended File Attributes

Windows, through the FindFirstFile API, will return these attributes:

| | | | |
|---|---|---|---|
| "A" | "Archive" | "O" | "Offline" |
| "C" | "Compressed" | "R" | "ReadOnly" |
| "d" | "Directory" | "P" | "Reparse Point" |
| "E" | "Encrypted" | "$" | "Sparse" |
| "H" | "Hidden" | "S" | "System" |
| "N" | "Normal" | "T" | "Temporary" |

All of the attributes that follow must be have logic other than the Windows FindFirstFile API to determine their true/falseness.

| | | |
|---|---|---|
| "f" | "Regular" | |
| "r" | "Readable" | Hold over from Unix. Always true, if exists. |
| "s" | "Size > 0" | |
| "w" | "Writeable" | Opposite of 'R', does not check NTFS permission. |

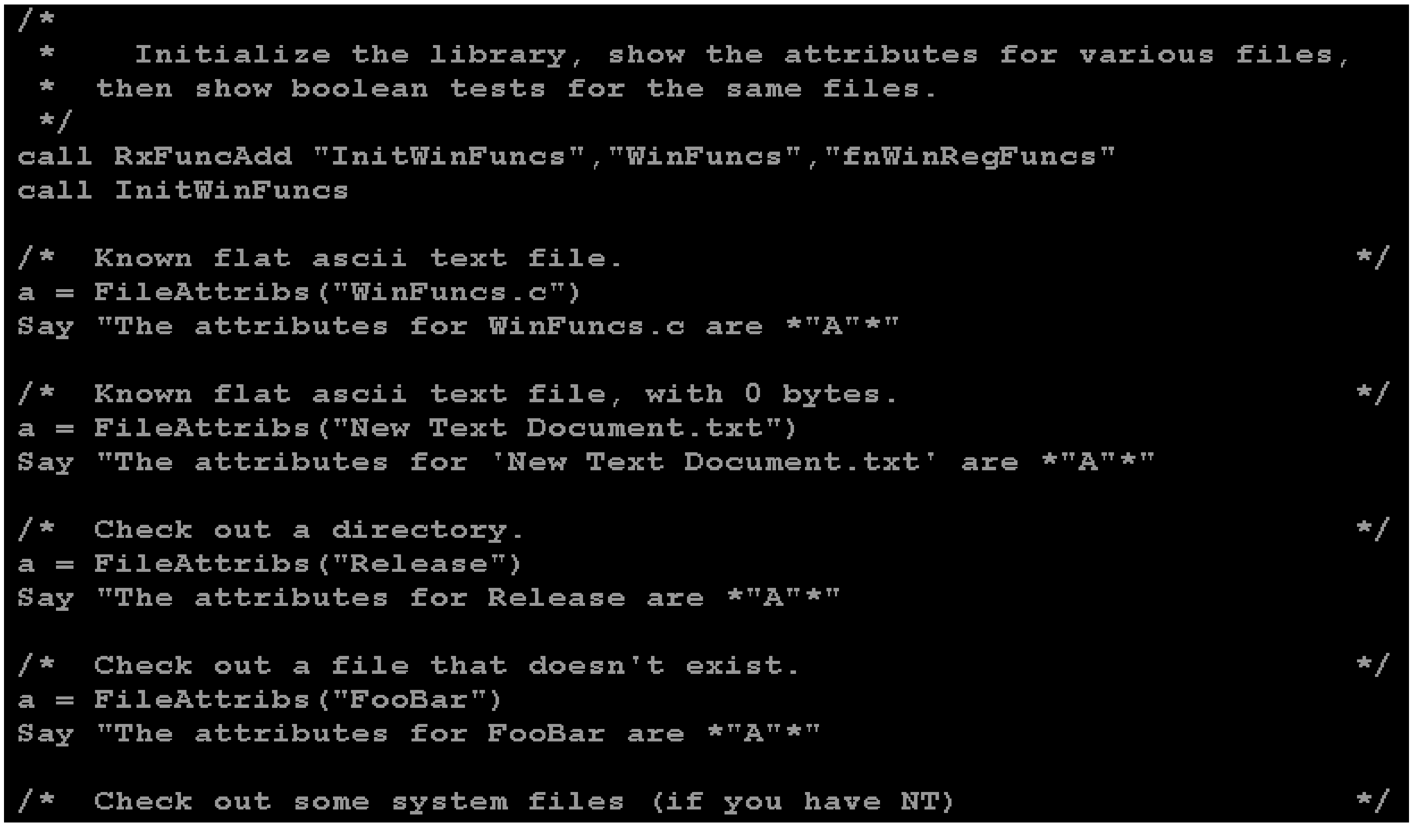
## ■ Executable File Properties

For executables compiled into the Windows PE format that include a resource, the strings of that resource are extracted. Currently, it does NOT do FE format (OS/2 executables, and *.vlm's.).

# Windows Toolkit

File Attributes

## Code Sample Part 1

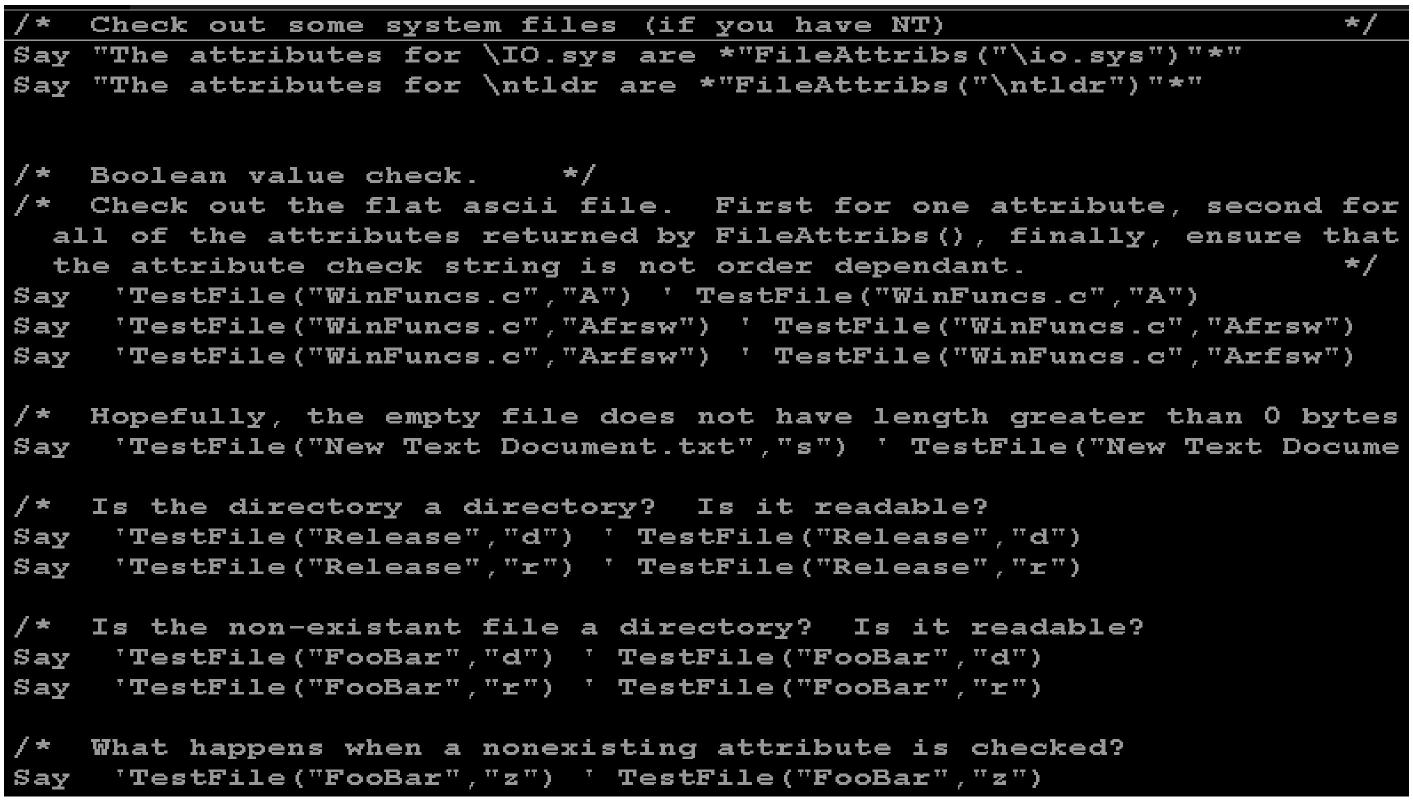


```
/*
 *    Initialize the library, show the attributes for various files,
 *  then show boolean tests for the same files.
 */
call RxFuncAdd "InitWinFuncs","WinFuncs","fnWinRegFuncs"
call InitWinFuncs

/*  Known flat ascii text file.                                    */
a = FileAttribs("WinFuncs.c")
Say "The attributes for WinFuncs.c are *"A"*"

/*  Known flat ascii text file, with 0 bytes.                      */
a = FileAttribs("New Text Document.txt")
Say "The attributes for 'New Text Document.txt' are *"A"*"

/*  Check out a directory.                                         */
a = FileAttribs("Release")
Say "The attributes for Release are *"A"*"

/*  Check out a file that doesn't exist.                           */
a = FileAttribs("FooBar")
Say "The attributes for FooBar are *"A"*"

/*  Check out some system files (if you have NT)                   */
```

# Windows Toolkit

File Attributes

## Code Sample, Part 2

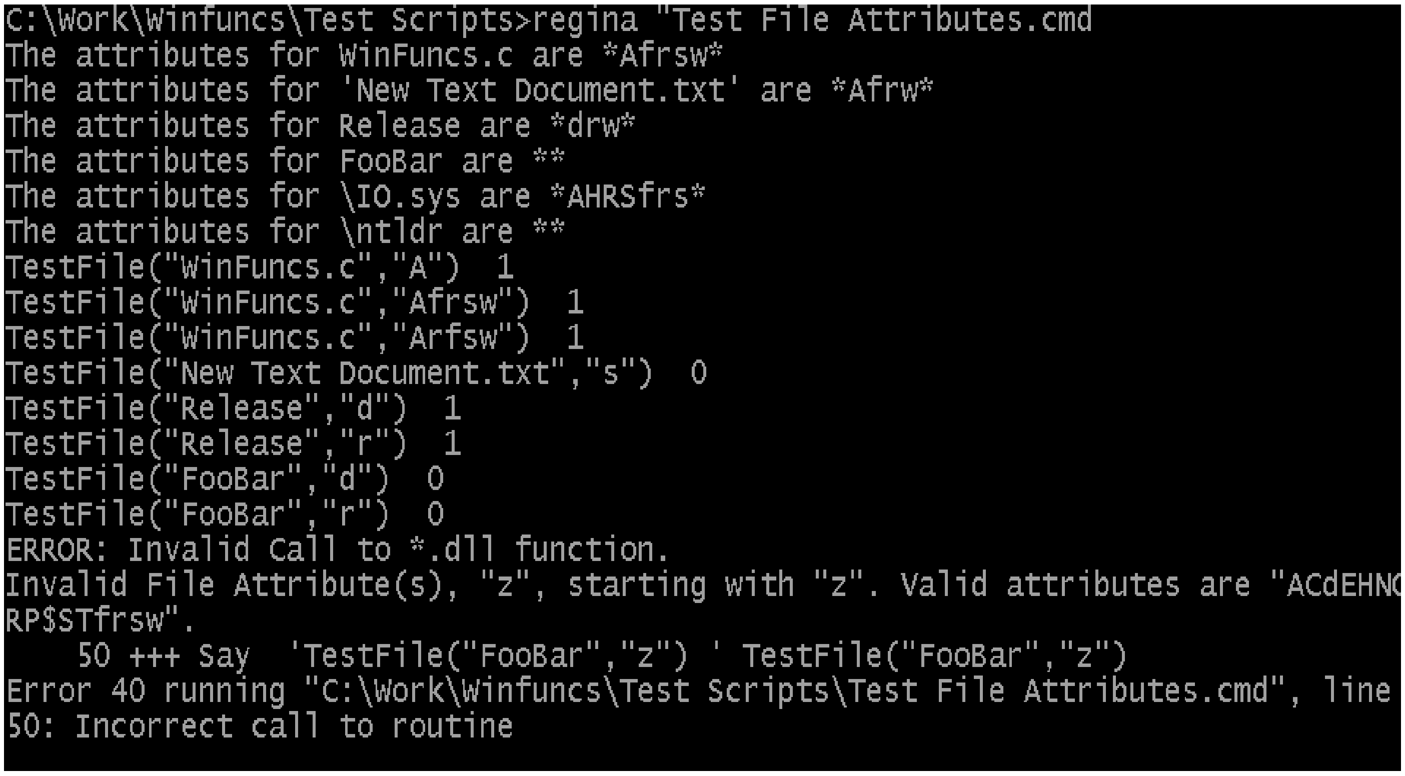


```
/*  Check out some system files (if you have NT)                        */
Say "The attributes for \IO.sys are *"FileAttribs("\io.sys")"*"
Say "The attributes for \ntldr are *"FileAttribs("\ntldr")"*"


/*  Boolean value check.    */
/*  Check out the flat ascii file.  First for one attribute, second for
  all of the attributes returned by FileAttribs(), finally, ensure that
  the attribute check string is not order dependant.                    */
Say  'TestFile("WinFuncs.c","A") ' TestFile("WinFuncs.c","A")
Say  'TestFile("WinFuncs.c","Afrsw") ' TestFile("WinFuncs.c","Afrsw")
Say  'TestFile("WinFuncs.c","Arfsw") ' TestFile("WinFuncs.c","Arfsw")

/*  Hopefully, the empty file does not have length greater than 0 bytes
Say  'TestFile("New Text Document.txt","s") ' TestFile("New Text Docume

/*  Is the directory a directory?  Is it readable?
Say  'TestFile("Release","d") ' TestFile("Release","d")
Say  'TestFile("Release","r") ' TestFile("Release","r")

/*  Is the non-existant file a directory?  Is it readable?
Say  'TestFile("FooBar","d") ' TestFile("FooBar","d")
Say  'TestFile("FooBar","r") ' TestFile("FooBar","r")

/*  What happens when a nonexisting attribute is checked?
Say  'TestFile("FooBar","z") ' TestFile("FooBar","z")
```

# Windows Toolkit

File Attributes

## Results of execution

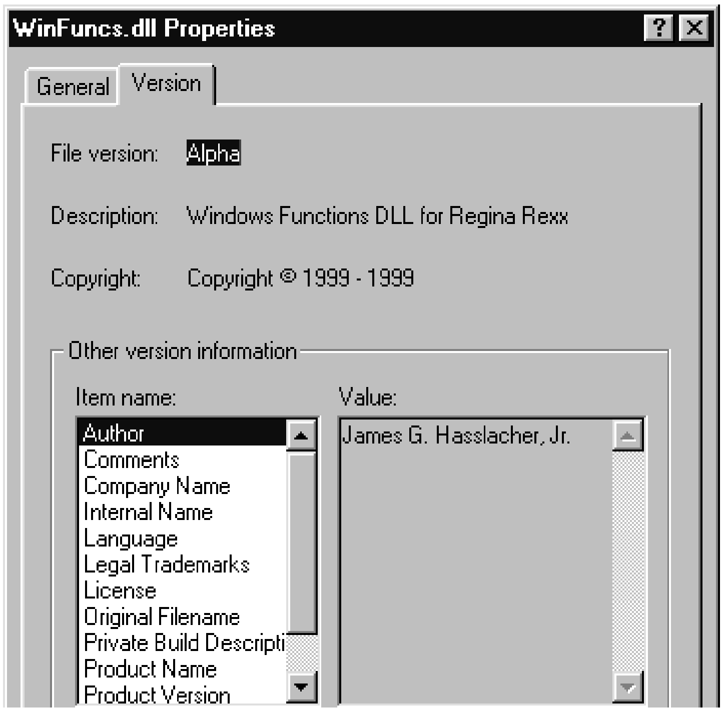


```
C:\Work\Winfuncs\Test Scripts>regina "Test File Attributes.cmd
The attributes for WinFuncs.c are *Afrsw*
The attributes for 'New Text Document.txt' are *Afrw*
The attributes for Release are *drw*
The attributes for FooBar are **
The attributes for \IO.sys are *AHRSfrs*
The attributes for \ntldr are **
TestFile("WinFuncs.c","A")  1
TestFile("WinFuncs.c","Afrsw")  1
TestFile("WinFuncs.c","Arfsw")  1
TestFile("New Text Document.txt","s")  0
TestFile("Release","d")  1
TestFile("Release","r")  1
TestFile("FooBar","d")  0
TestFile("FooBar","r")  0
ERROR: Invalid Call to *.dll function.
Invalid File Attribute(s), "z", starting with "z". Valid attributes are "ACdEHNO
RP$STfrsw".
    50 +++ Say  'TestFile("FooBar","z") ' TestFile("FooBar","z")
Error 40 running "C:\Work\Winfuncs\Test Scripts\Test File Attributes.cmd", line
50: Incorrect call to routine
```

## Windows Toolkit

File Properties

- The Version section of the Resource file
- The same information that is obtained by right-clicking the file, and choosing “Properties” from the list


WinFuncs.dll Properties

General | Version

File version: Alpha

Description: Windows Functions DLL for Regina Rexx

Copyright: Copyright © 1999 - 1999

Other version information

| Item name: | Value: |
|---|---|
| Author | James G. Hasslacher, Jr. |
| Comments | |
| Company Name | |
| Internal Name | |
| Language | |
| Legal Trademarks | |
| License | |
| Original Filename | |
| Private Build Descripti | |
| Product Name | |
| Product Version | |

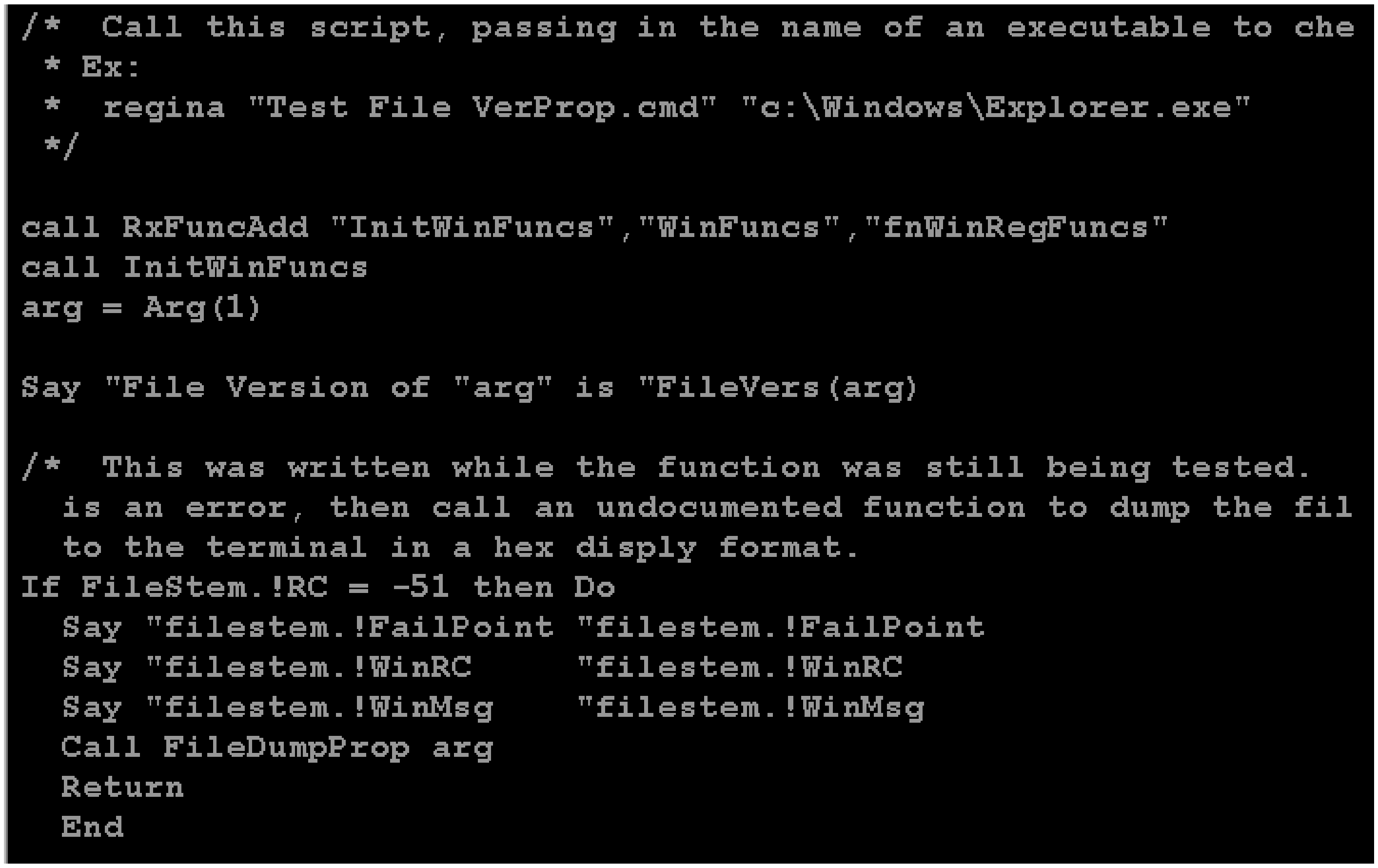
## Windows Toolkit

File Properties

- FileVers(filename)
  - Extracts only the version of the executable
- FileProp(filename)
  - Extracts all of the properties into two stems.
  - FileStem contains the names of the property strings.
    - The first index to this stem is 0.  That contains the number of number string names.
  - FileVal contains all of the values for the property strings.
    - The index to this stem is the property name; ie, the values in FileStem
- FileDumpProp(filename)
  - Used for debugging.  Prints the properties section in a hex dump format.

## Windows Toolkit

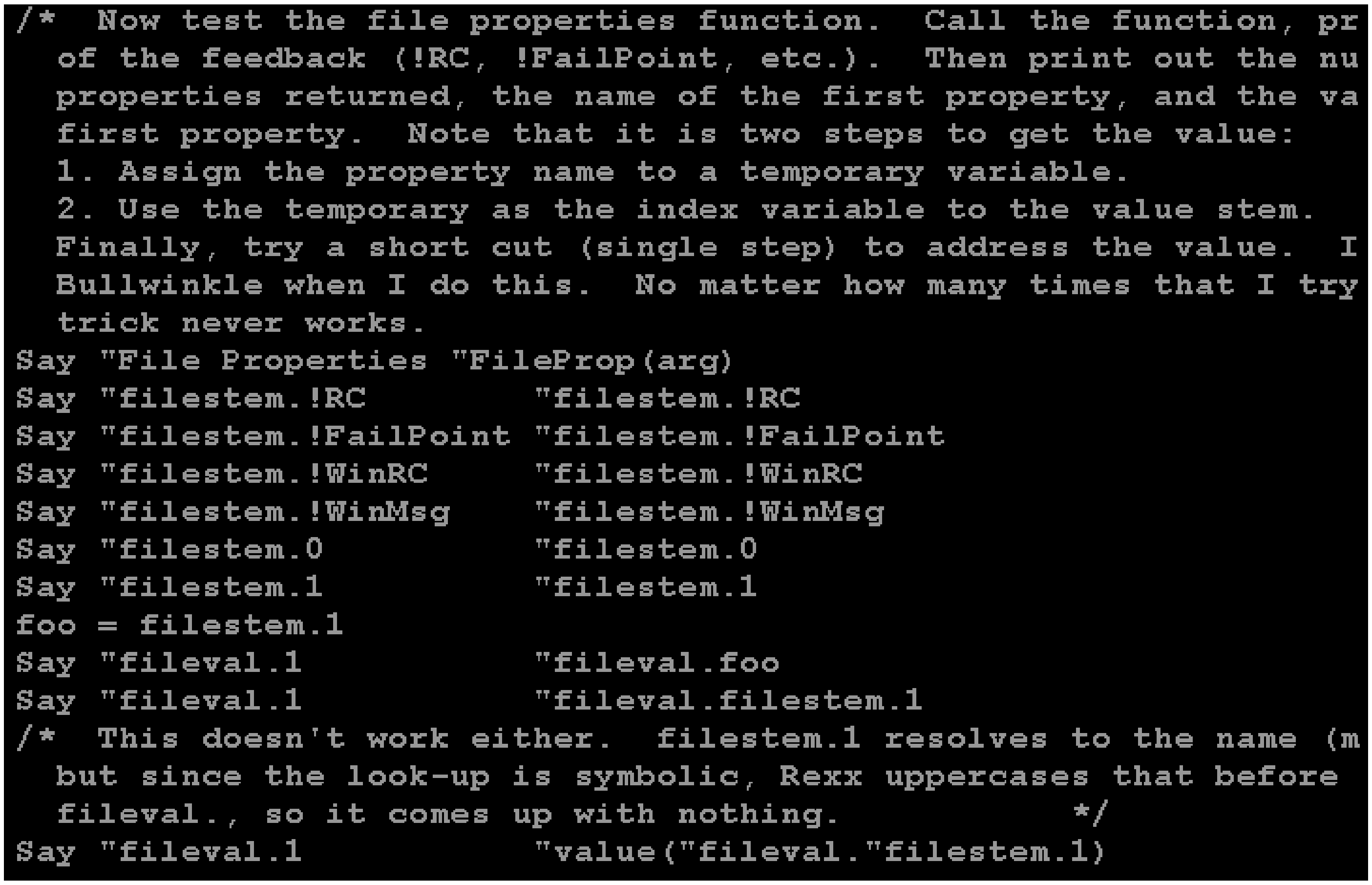
File Properties Code 1 of 3



```
/*  Call this script, passing in the name of an executable to che
 * Ex:
 *  regina "Test File VerProp.cmd" "c:\Windows\Explorer.exe"
 */

call RxFuncAdd "InitWinFuncs","WinFuncs","fnWinRegFuncs"
call InitWinFuncs
arg = Arg(1)

Say "File Version of "arg" is "FileVers(arg)

/*  This was written while the function was still being tested.
  is an error, then call an undocumented function to dump the fil
  to the terminal in a hex disply format.
If FileStem.!RC = -51 then Do
  Say "filestem.!FailPoint "filestem.!FailPoint
  Say "filestem.!WinRC     "filestem.!WinRC
  Say "filestem.!WinMsg    "filestem.!WinMsg
  Call FileDumpProp arg
  Return
  End
```

# Windows Toolkit

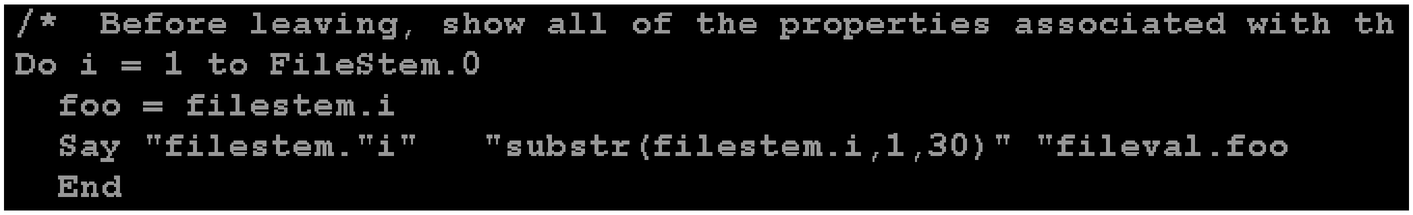
File Properties Code 2 of 3



```
/*  Now test the file properties function.  Call the function, pr
  of the feedback (!RC, !FailPoint, etc.).  Then print out the nu
  properties returned, the name of the first property, and the va
  first property.  Note that it is two steps to get the value:
  1. Assign the property name to a temporary variable.
  2. Use the temporary as the index variable to the value stem.
  Finally, try a short cut (single step) to address the value.  I
  Bullwinkle when I do this.  No matter how many times that I try
  trick never works.
Say "File Properties "FileProp(arg)
Say "filestem.!RC         "filestem.!RC
Say "filestem.!FailPoint "filestem.!FailPoint
Say "filestem.!WinRC      "filestem.!WinRC
Say "filestem.!WinMsg     "filestem.!WinMsg
Say "filestem.0           "filestem.0
Say "filestem.1           "filestem.1
foo = filestem.1
Say "fileval.1            "fileval.foo
Say "fileval.1            "fileval.filestem.1
/*  This doesn't work either.  filestem.1 resolves to the name (m
  but since the look-up is symbolic, Rexx uppercases that before
  fileval., so it comes up with nothing.           */
Say "fileval.1            "value("fileval."filestem.1)
```

# Windows Toolkit

File Properties Code 3 of 3

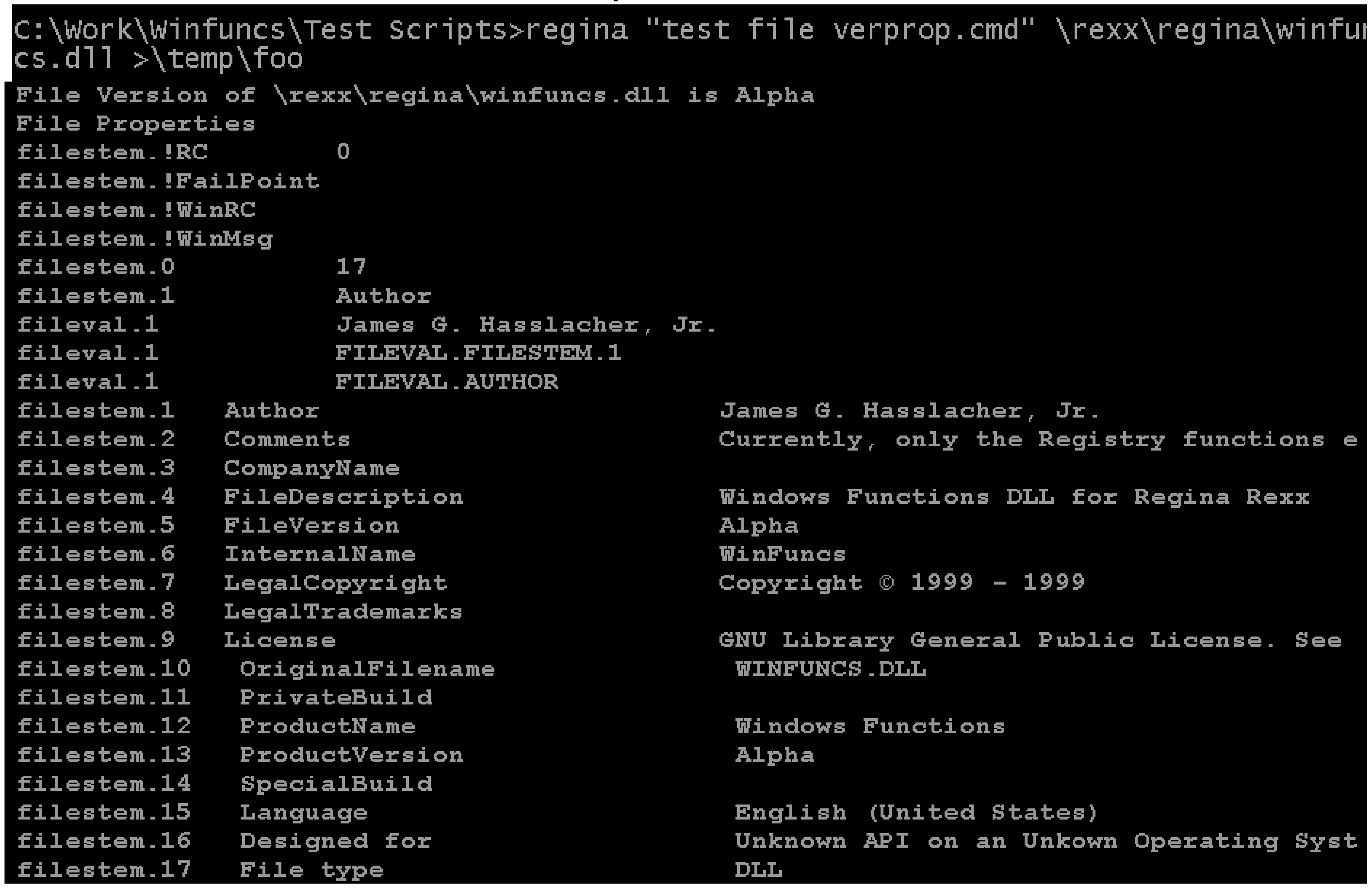


```
/*  Before leaving, show all of the properties associated with th
Do i = 1 to FileStem.0
  foo = filestem.i
  Say "filestem."i"   "substr(filestem.i,1,30)" "fileval.foo
  End
```

Note that this is a two stage process.  Extract the stem index from FileStem., use that to look up the actual value in FileVal..

## Windows Toolkit

File Properties Results



```
C:\Work\Winfuncs\Test Scripts>regina "test file verprop.cmd" \rexx\regina\winfun
cs.dll >\temp\foo
File Version of \rexx\regina\winfuncs.dll is Alpha
File Properties
filestem.!RC        0
filestem.!FailPoint
filestem.!WinRC
filestem.!WinMsg
filestem.0          17
filestem.1          Author
fileval.1           James G. Hasslacher, Jr.
fileval.1           FILEVAL.FILESTEM.1
fileval.1           FILEVAL.AUTHOR
filestem.1   Author                        James G. Hasslacher, Jr.
filestem.2   Comments                      Currently, only the Registry functions e
filestem.3   CompanyName
filestem.4   FileDescription               Windows Functions DLL for Regina Rexx
filestem.5   FileVersion                   Alpha
filestem.6   InternalName                  WinFuncs
filestem.7   LegalCopyright                Copyright © 1999 - 1999
filestem.8   LegalTrademarks
filestem.9   License                       GNU Library General Public License. See
filestem.10   OriginalFilename              WINFUNCS.DLL
filestem.11   PrivateBuild
filestem.12   ProductName                   Windows Functions
filestem.13   ProductVersion                Alpha
filestem.14   SpecialBuild
filestem.15   Language                      English (United States)
filestem.16   Designed for                  Unknown API on an Unkown Operating Syst
filestem.17   File type                     DLL
```

## Windows Toolkit

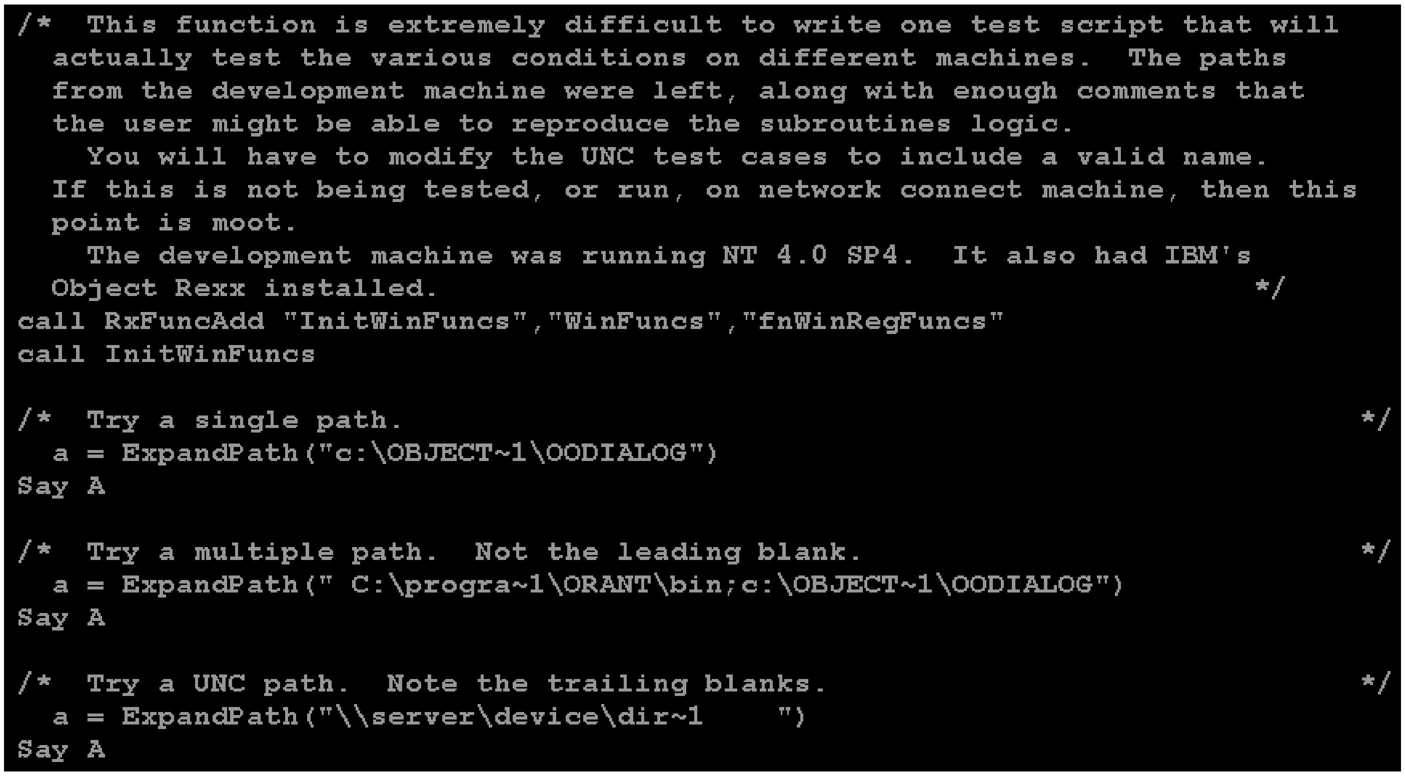
PATH manipulation

One function, ExpandPath(argument), performs all of the necessary manipulation.

- The argument can be any valid PATH syntax
  - One, or more, path's separated by semi-colon's(;).
  - Paths may be:
    - absolute
    - relative
    - UNC
- The return string is the argument with all of the short path names expanded to long names, and any duplicates removed.
- If a path does not exist, then it and its associated semicolon are removed from the return string.

# Windows Toolkit

PATH manipulation Code 1 of 2



```
/*  This function is extremely difficult to write one test script that will
  actually test the various conditions on different machines.  The paths
  from the development machine were left, along with enough comments that
  the user might be able to reproduce the subroutines logic.
    You will have to modify the UNC test cases to include a valid name.
  If this is not being tested, or run, on network connect machine, then this
  point is moot.
    The development machine was running NT 4.0 SP4.  It also had IBM's
  Object Rexx installed.                                               */
call RxFuncAdd "InitWinFuncs","WinFuncs","fnWinRegFuncs"
call InitWinFuncs

/*  Try a single path.                                                      */
  a = ExpandPath("c:\OBJECT~1\OODIALOG")
Say A

/*  Try a multiple path.  Not the leading blank.                            */
  a = ExpandPath(" C:\progra~1\ORANT\bin;c:\OBJECT~1\OODIALOG")
Say A

/*  Try a UNC path.  Note the trailing blanks.                              */
  a = ExpandPath("\\server\device\dir~1    ")
Say A
```

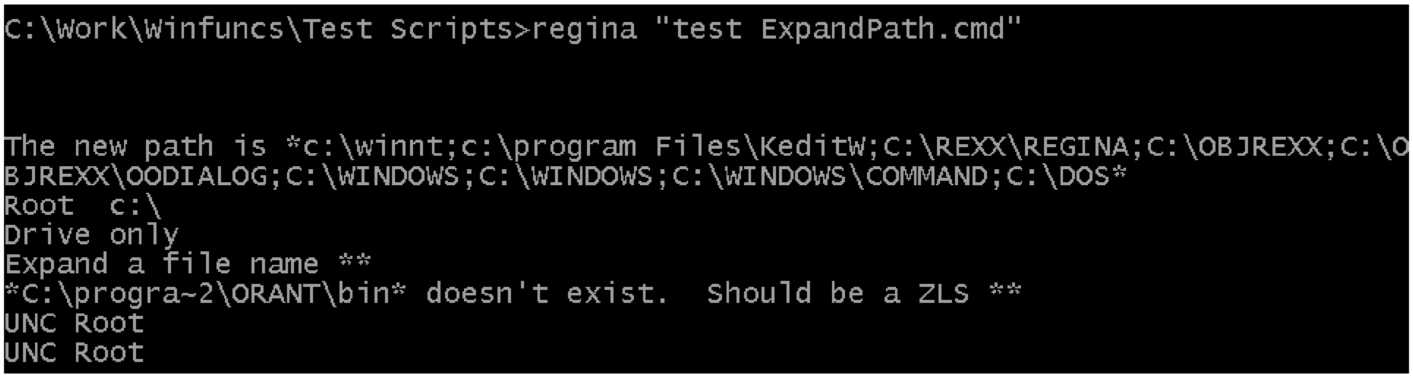
# Windows Toolkit

PATH manipulation Code 2 of 2

```
/*  Try a more of a real life test.  Retrieve the PATH environment variable.
  Expand it.  See if the path that we want is in the PATH.  If not, prepend it. */
  P = ExpandPath(Value("PATH",,"ENVIRONMENT"))
  D = "c:\program Files\KeditW"
  If Pos(';'Translate(D)';',';'Translate(P)';') = 0 Then P = D';'P
  D = "c:\winnt"
  If Pos(';'Translate(D)';',';'Translate(P)';') = 0 Then P = D';'P
  Q = Value("PATH",P,"ENVIRONMENT")
  R = Value("PATH",,"ENVIRONMENT")
Say "The new path is *"R"*"

/*  Now for all of the wierdo cases.                                        */
/*  See what the root returns.                                              */
  Say "Root " ExpandPath("c:\")
  Say "Drive only " ExpandPath("c:")

/*  See what a file returns.                                                */
  a = ExpandPath("c:\Autoexec.bat")
Say "Expand a file name *"A"*"

/*  What about a directory that doesn't exist?                              */
  a = ExpandPath("C:\progra~2\ORANT\bin")
Say "*C:\progra~2\ORANT\bin* doesn't exist.  Should be a ZLS *"A"*"

/*  Try a UNC root.  try with and without ending \.                         */
  Say "UNC Root " ExpandPath("\\Server\device\")
  Say "UNC Root " ExpandPath("\\Server\device")
```

## Windows Toolkit

PATH manipulation Results



```
C:\Work\Winfuncs\Test Scripts>regina "test ExpandPath.cmd"



The new path is *c:\winnt;c:\program Files\KeditW;C:\REXX\REGINA;C:\OBJREXX;C:\O
BJREXX\OODIALOG;C:\WINDOWS;C:\WINDOWS;C:\WINDOWS\COMMAND;C:\DOS*
Root  c:\
Drive only
Expand a file name **
*C:\progra~2\ORANT\bin* doesn't exist.  Should be a ZLS **
UNC Root
UNC Root
```