Example: Fast Scan of Directories

Calling IndexedDir for each directory is an O(N^2) problem because to get the nth directory the OS routines underlying IndexedDir need to iterate through directories 0..n-1. This becomes an issue only if you are dealing with hundreds or thousands of directories.

This function illustrates a technique for converting this to an O(N) problem by getting a complete list of paths from IndexedDir in one call and storing them in a text wave. This approach could also be used with IndexedFile.

```
Function ScanDirectories(pathName, printDirNames)
   String pathName           // Name of Igor symbolic path
   Variable printDirNames    // True if you want to print the directory names

   Variable t0 = StopMSTimer(-2)

   String dirList = IndexedDir($pathName, -1, 0)
   Variable numDirs = ItemsInList(dirList)

   // Store directory list in a free text wave.
   // The free wave is automatically killed when the function returns.
   Make/N=(numDirs)/T/FREE dirs = StringFromList(p, dirList)

   String dirName
   Variable i
   for(i=0; i<numDirs; i+=1)
      dirName = dirs[i]
      Print i, dirName
   endfor

   Variable t1 = StopMSTimer(-2)
   Variable elapsed = (t1 - t0) / 1E6
   Printf "Took %g seconds\r", elapsed
End
```

IndexedFile

IndexedFile(*pathName*, *index*, *fileTypeOrExtStr* [, *creatorStr*, *separatorStr*])

If *index* is greater than or equal to zero, IndexedFile returns a string containing the name of the *index*th file in the folder specified by *pathName* which matches the file type or extension specified by *fileTypeOrExtStr*.

If *index* is -1, IndexedFile returns a string list of all matching files separated by a semicolon, or by separatorStr if specified.

IndexedFile returns an empty string ("") if there is no such file.

Parameters

pathName is the *name* of an Igor symbolic path. It is *not* a string.

index normally starts from zero. However, if *index* is -1, IndexedFile returns a string containing a semicolon-separated list of the names of all files in the folder associated with the specified symbolic path which match *fileTypeOrExtStr*.

fileTypeOrExtStr is either:

- A string starting with ".", such as ".txt", ".bwav", or ".c". Only files with a matching file name extension are indexed. Set *fileTypeOrExtStr* to "." to index file names that end with "." such as "myFileNameEndsWithThisDot."
- A string containing exactly four ASCII characters, such as "`TEXT`" or "`IGBW`". Only files of the specified Macintosh file type are indexed. However, if *fileTypeOrExtStr* is "????", files of any type are indexed.
- On Windows, Igor considers files with ".txt" extensions to be of type TEXT. It does similar mappings for other extensions. See **File Types and Extensions** on page III-455 for details.

creatorStr is obsolete and should no longer be used unless you need to specify the separator in which case, use "????" for *creatorStr*.

separatorStr is an optional string argument used when index is -1. If you omit *separatorStr*, file names are separated by ";". This parameter was added in Igor 9.01.

Order of Files Returned By IndexedFile

The order in which files are returned by IndexedFile is determined by the operating system. If the order matters to you, you should explicitly sort the file names.

For example, assume you have files named "File1.txt", "File2.txt" and "File10.txt". An alphabetic order gives you: "File1.txt;File10.txt;File2.txt;". Often what you really want is a combination of alphabetic and numeric sorting returning "File1.txt;File2.txt;File10.txt;". Here is a function that does that:

```
Function DemoIndexedFile()
   String pathName = "Igor"// Refers to "Igor Pro Folder"

   // Get a semicolon-separated list of all files in the folder
   String list = IndexedFile($pathName, -1, ".txt")

   // Sort using combined alpha and numeric sort
   list = SortList(list, ";", 16)

   // Process the list
   Variable numItems = ItemsInList(list)
   Variable i
   for(i=0; i<numItems; i+=1)
      String fileName = StringFromList(i, list)
      Print i, fileName
   endfor
End
```

Treatment of Macintosh Dot-Underscore Files

IndexedFile ignores "dot-underscore" files unless *fileTypeOrExtStr* is "????".

A dot-underscore file is a file created by Macintosh when it writes to a non-HFS volume, for example, when it writes to a Windows volume via SMB file sharing. The dot-underscore file stores Macintosh HFS-specific data such as the file's type and creator codes, and the file's resource fork, if it has one.

For example, if a file named "wave0.ibw" is copied via SMB to a Windows volume, Mac OS X creates two files on the Windows volume: "wave0.ibw" and "._wave0.ibw". Mac OS X makes these two files appear as one to Macintosh applications. However, Windows does not do this. As a consequence, when a Windows program sees "._wave0.ibw", it expects it to be a valid .ibw file, but it is not. This causes problems.

By ignoring dot-underscore files, IndexedFile prevents this type of problem. However, if fileTypeOrExtStr is "????", IndexedFile will return dot-underscore files on Windows.

Examples

```
NewPath/O myPath "MyDisk:MyFolder:"
Print IndexedFile(myPath,-1,"TEXT")         // all text-type files
Print IndexedFile(myPath,0,"TEXT")          // only the first text file
Print IndexedFile(myPath,-1,".dat")         // *.dat
Print IndexedFile(myPath,-1,"TEXT","IGR0")  // all Igor text files
Print IndexedFile(myPath,-1,"????")         // all files, all creators
Print IndexedFile(myPath,-1,"????","????","\r")   // CR separator
```

See **IndexedDir** for another example using IndexedFile and for a method for speeding up scanning of very large numbers of files.

See Also

The **TextFile** and **IndexedDir** functions.