IndependentModuleList

IndependentModuleList(*listSepStr*)

The IndependentModuleList function returns a string containing a list of independent module names separated by listSepStr.

Use **StringFromList** to access individual names.

Parameters

listSepStr contains the character, usually ";", to be used to to separate the names in the returned list.

Details

In Igor6, only the first byte of *listSepStr* was used. In Igor7 and later, all bytes are used.

ProcGlobal is not in the returned list, and the order of returned names is not defined.

See Also

Independent Modules on page IV-238.

GetIndependentModuleName, **StringFromList**, **FunctionList**.

IndexedDir

IndexedDir(*pathName*, *index*, *flags* [, *separatorStr*])

The IndexedDir function returns a string containing the name of or the full path to the *index*th folder in the folder referenced by *pathName*.

Parameters

pathName is the name of an Igor symbolic path pointing to the parent directory.

index is the index number of the directory (within the parent directory) of interest starting from 0. If *index* is -1, IndexedDir will return the name of *all* of the folders in the parent, separated by semicolons or by *separatorStr* if specified.

flags is a bitwise parameter:

Bit 0:	Set if you want a full path. Cleared if you want just the directory name.

All other bits are reserved and should be cleared.

See **Setting Bit Parameters** on page IV-12 for details about bit settings.

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

Details

You create the symbolic path identifying the parent directory using the **NewPath** operation or the New Path dialog (Misc menu).

Prior to Igor Pro 3.1, IndexedDir was an external function and took a string as the first parameter rather than a name. The *pathName* parameter can now be either a name or a string containing a name. Any of the following will work:

```
String str = "IGOR"
Print IndexedDir(IGOR, 0, 0)         // First parameter is a name.
Print IndexedDir($str, 0, 0)         // First parameter is a name.
Print IndexedDir("IGOR", 0, 0)       // First parameter is a string.
Print IndexedDir(str, 0, 0)          // First parameter is a string.
```

The acceptance of a string is for backward compatibility only. New code should be written using a name.

The returned path uses the native conventions of the OS under which Igor is running.

Examples

Example: Recursively Listing Directories and Files

Here is an example for heavy-duty Igor Pro programmers. It is an Igor Pro user-defined function that prints the paths of all of the files and folders in a given folder with or without recursion. You can rework this to do something with each file instead of just printing its path.

To try the function, copy and paste it into the Procedure window. Then execute the example shown in the comments.

```
// PrintFoldersAndFiles(pathName, extension, recurse, level)
// Shows how to recursively find all files in a folder and subfolders.
// pathName is the name of an Igor symbolic path that you created
// using NewPath or the Misc->New Path menu item.
// extension is a file name extension like ".txt" or "????" for all files.
// recurse is 1 to recurse or 0 to list just the top-level folder.
// level is the recursion level - pass 0 when calling PrintFoldersAndFiles.
// Example: PrintFoldersAndFiles("Igor", ".ihf", 1, 0)
Function PrintFoldersAndFiles(pathName, extension, recurse, level)
   String pathName     // Name of symbolic path in which to look for folders and files.
   String extension    // File name extension (e.g., ".txt") or "????" for all files.
   Variable recurse    // True to recurse (do it for subfolders too).
   Variable level      // Recursion level. Pass 0 for the top level.

   Variable folderIndex, fileIndex
   String prefix

   // Build a prefix (a number of tabs to indicate the folder level by indentation)
   prefix = ""
   folderIndex = 0
   do
      if (folderIndex >= level)
         break
      endif
      prefix += "\t"      // Indent one more tab
      folderIndex += 1
   while(1)

   // Print folder
   String path
   PathInfo $pathName     // Sets S_path
   path = S_path
   Printf "%s%s\r", prefix, path

   // Print files
   fileIndex = 0
   do
      String fileName
      fileName = IndexedFile($pathName, fileIndex, extension)
      if (strlen(fileName) == 0)
         break
      endif
      Printf "%s\t%s%s\r", prefix, path, fileName
      fileIndex += 1
   while(1)

   if (recurse)           // Do we want to go into subfolder?
      folderIndex = 0
      do
         path = IndexedDir($pathName, folderIndex, 1)
         if (strlen(path) == 0)
            break        // No more folders
         endif

         String subFolderPathName = "tempPrintFoldersPath_" + num2istr(level+1)

         // Now we get the path to the new parent folder
         String subFolderPath
         subFolderPath = path

         NewPath/Q/O $subFolderPathName, subFolderPath
         PrintFoldersAndFiles(subFolderPathName, extension, recurse, level+1)
         KillPath/Z $subFolderPathName

         folderIndex += 1
      while(1)
   endif
End
```