

Re: Can Dir handle different file types?

Source:

<http://www.tech-archive.net/Archive/VB/microsoft.public.vb.general.discussion/2006-08/msg00086.html>

- *From:* "Larry Serflaten" <serflaten@xxxxxxxxxxxxxxxx>
 - *Date:* Mon, 24 Jul 2006 20:29:14 -0500
-

"RB Smissaert" <bartsmissaert@xxxxxxxxxxxxxxxx> wrote

Yes, but how can I fit that in with the Dir based function?
Knocked a bit more time off by using a collection, avoiding a Redim Preserve on an array and might as well post the code here:

I just wanted to suggest that you might want to start creating more modular code. Looking for files should really be separated from looking for folders. You may have need of either individually.

By writting modular code you increase the chance of reuse, and thereby reducing overall code size.

Since many have offered suggested code, here is yet another set of functions you might inspect or use, ready for posting into a new module.... <g>

(They use and return vbCrLf delimited lists, using strings instead of collections or arrays)

example usage:

```
Sub main()  
Dim folderCount As Long, fileCount As Long  
Dim list As String  
  
list = ListFolders("f:", folderCount, True)  
Debug.Print ListFiles(list, "*.txt;*.htm", fileCount)  
Debug.Print fileCount; "files found in"; folderCount; "folders"  
  
End Sub
```

[Module code]

Option Explicit

Re: Can Dir handle different file types?

```
Public Function ListFiles(Paths As String, Patterns As String, Optional ListCount As Long = 0) As String
' Returns a vbCrLf delimited list of files from one or more paths
' Paths = A vbCrLf delimited list of paths to search
' Patterns = A ";" delimited list of patterns to search for (* and ? wildcards accepted)
' ListCount = Optional return value for number of files found
Dim list As String, path As String, pattern As String, fileName As String, filePath As String
Dim pathIndex As Long, patternIndex As Long, listPosition As Long

' Init variables
ListCount = 0
listPosition = 1
' Search by path
pathIndex = 1
path = GetNthItem(Paths, vbCrLf, pathIndex)
' Loop through Paths list
Do While Len(path) > 0
' Search by patterns
patternIndex = 1
pattern = GetNthItem(Patterns, ";", patternIndex)
' Loop through Patterns list
Do While Len(pattern) > 0
' Find file names matching current pattern
fileName = Dir(path & "\" & pattern, vbSystem Or vbHidden Or vbArchive)
Do While Len(fileName) > 0
' Save file name to list
listPosition = AddToList(list, listPosition, LCase$(path & "\" & fileName))
ListCount = ListCount + 1
' Occasional yield to OS
If (ListCount And 31) = 31 Then DoEvents
' Get next filename
fileName = Dir()
Loop
' Increment counter, get next pattern
patternIndex = patternIndex + 1
pattern = GetNthItem(Patterns, ";", patternIndex)
Loop
' Increment counter, get next path
pathIndex = pathIndex + 1
path = GetNthItem(Paths, vbCrLf, pathIndex)
Loop
' Trim trailing spaces and vbCrLf
If ListCount > 0 Then
list = RTrim$(list)
list = Left$(list, Len(list) - 2)
End If
' Return list
ListFiles = list
End Function
```

Re: Can Dir handle different file types?

```
Public Function ListFolders(Paths As String, Optional ListCount As Long = 0, Optional SubFolders As Boolean = False) As String
```

```
' Returns a vbCrLf delimited list of folders from one or more paths ordered by nesting levels
```

```
' Paths = A vbCrLf delimited list of paths to search
```

```
' ListCount = Optional return value for number of folders found
```

```
Dim path As String, folder As String, folderPath As String, list As String
```

```
Dim pathIndex As Long, folderIndex As Long, listPosition As Long
```

```
ListCount = 0
```

```
listPosition = 1
```

```
' Accept lists that start with up to 2 delimiters
```

```
Do While (pathIndex < 3) And (Len(path) = 0)
```

```
pathIndex = pathIndex + 1
```

```
path = GetNthItem(Paths, vbCrLf, pathIndex)
```

```
Loop
```

```
Do While Len(path) > 0
```

```
' Search this path
```

```
folder = Dir(path & "\*.*", vbDirectory)
```

```
Do While Len(folder) > 0
```

```
' Test for . and ..
```

```
If Left$(folder, 1) <> "." Then
```

```
folderPath = LCase$(path & "\" & folder)
```

```
' Test for directory
```

```
If GetAttr(folderPath) And vbDirectory Then
```

```
listPosition = AddToList(list, listPosition, folderPath)
```

```
ListCount = ListCount + 1
```

```
' Occasional yield to OS
```

```
If (ListCount And 7) = 7 Then DoEvents
```

```
End If
```

```
End If
```

```
' Get next folder
```

```
folder = Dir()
```

```
Loop
```

```
' Get next path
```

```
pathIndex = pathIndex + 1
```

```
path = GetNthItem(Paths, vbCrLf, pathIndex)
```

```
Loop
```

```
' Test for success
```

```
If ListCount > 0 Then
```

```
' Trim trailing spaces and vbCrLf
```

```
list = RTrim$(list)
```

```
list = Left$(list, Len(list) - 2)
```

```
' Process sub folders
```

```
If SubFolders Then
```

```
path = ListFolders(list, pathIndex, True)
```

```
' Test for success
```

```
If pathIndex > 0 Then
```

```
list = list & vbCrLf & path
```

Re: Can Dir handle different file types?

Re: Can Dir handle different file types?

```
ListCount = ListCount + pathIndex
End If
End If
End If
' Return list
ListFolders = list
End Function

Public Function GroupSubFolders(Paths As String, Optional ListCount As Long) As String
' Returns a vbCrLf delimited list of folders from one or more paths ordered by folders
' Paths = A vbCrLf delimited list of paths to search
' ListCount = Optional return value for number of folders found
Dim list As String, sublist As String, path As String, subPath As String, group As String
Dim pathIndex As Long, subIndex As Long, listPosition As Long, subCount As Long, groupCount As Long

' Init variables
ListCount = 0
listPosition = 1

' Accept lists that start with up to 2 delimiters
Do While (pathIndex < 3) And (Len(path) = 0)
pathIndex = pathIndex + 1
path = GetNthItem(Paths, vbCrLf, pathIndex)
Loop

' Loop though Paths list
Do While Len(path) > 0
' List sub folders in this folder
sublist = ListFolders(path, subIndex)
subCount = 1

' Loop through folders list
Do While subIndex > 0

' Get next folder
subPath = GetNthItem(sublist, vbCrLf, subCount)
Do While Len(subPath) > 0

' Add to list
listPosition = AddToList(list, listPosition, subPath)
ListCount = ListCount + 1

' Find any sub folders
group = GroupSubFolders(subPath, groupCount)
If groupCount > 0 Then
' Add sub folders to list
listPosition = AddToList(list, listPosition, group)
ListCount = ListCount + groupCount
End If

' Get next folder
```

Re: Can Dir handle different file types?

```
subCount = subCount + 1
subPath = GetNthItem(sublist, vbCrLf, subCount)
Loop
```

```
' Continue through folder list
subIndex = subIndex - 1
Loop
```

```
' Get next path
pathIndex = pathIndex + 1
path = GetNthItem(Paths, vbCrLf, pathIndex)
Loop
```

```
' Trim trailing spaces and vbCrLf
If ListCount > 0 Then
list = RTrim$(list)
list = Left$(list, Len(list) - 2)
End If
' Return list
GroupSubFolders = list
End Function
```

```
Public Function GetNthItem(Source As String, Delim As String, ByVal N As Long) As String
```

```
' Returns Nth item from a delimited list
```

```
Dim pos As Long, nxt As Long
```

```
pos = 1
' Get to N
Do While N > 1
pos = InStr(pos, Source, Delim)
If pos = 0 Then Exit Function
pos = pos + Len(Delim)
N = N - 1
Loop
```

```
' Find size of item and extract
If N = 1 Then
nxt = InStr(pos, Source, Delim)
' Test for last item
If nxt = 0 Then nxt = Len(Source) + 1
GetNthItem = Mid$(Source, pos, nxt - pos)
End If
```

```
End Function
```

```
Public Function AddToList(Destination As String, ByVal Position As Long, Source As String) As Long
```

```
' Concatenates strings using Mid instead of &, string size increased when needed
```

```
Dim pos As Long
```

```
' Test for sufficient room
```

Re: Can Dir handle different file types?

Re: Can Dir handle different file types?

```
If Len(Destination) < (Position + Len(Source) + 2) Then
Destination = Destination & Space$(Len(Source) + 4096)
End If
' Add string
Mid(Destination, Position) = Source
pos = Position + Len(Source)
' Add delimiter
Mid(Destination, pos) = vbCrLf
AddToList = pos + 2
End Function
```