6.3 Datei- und Verzeichnis-Funktionen

Diese Datei-und Verzeichnisfunktionen sowie ergänzende Funktionen werden Ihnen in den folgenden Abschnitten vorgestellt:

6.3.1 COPY .. TO ..

```
COPY SourcePath TO DestinationPath
```

- Diese Anweisung kopiert eine Datei von der Quelle *SourcePath* zum Ziel *DestinationPath*.
- Der Zielpfad muss nicht den gleichen (Basis-)Namen wie der Quellpfad haben.
- Beachten Sie, dass Sie mit dieser Anweisung keine Verzeichnisse rekursiv kopieren können!
- Es wird ein Fehler (Fehler-Nummer 38) ausgelöst, wenn die Ziel-Datei bereits existiert.
- Diese Anweisung nutzt intern die Funktionen, OPEN, READ, WRITE und CLOSE, weshalb auch alle damit verbundenen Fehlermeldungen auftreten können und die Zeitstempel nicht mit denen der Originaldatei übereinstimmen.

Beispiel 1:

```
Public sScriptPath As String
Public sTempScriptPath As String

Public Sub Form_Open()

  sScriptPath = "./data/test_script.sh"
  sTempScriptPath = Temp(File.BaseName(sScriptPath))

  Copy sScriptPath To sTempScriptPath
  Chmod sTempScriptPath To "r-xr-x---"
  …

End
```

Ein Bash-Skript wird aus dem data-Ordner innerhalb des Projekt-Ordners in eine temporäre Datei /tmp/gambas.1000/5503/test_script.tmp kopiert. Das ist hier notwendig, weil nur dann die Rechte gesetzt werden, so dass das Skript für den Benutzer und die Gruppe ausführbar wird.

Beispiel 2:

```
Public Sub btnCopyFile_Click()

' Based on an idea by Willy Raets
  Dim sSourcePath, sDestinationPath, sFilename As String

  sSourcePath = User.Home &/ "DW"
  sDestinationPath = User.Home &/ "GB3BUCH"
  sFileName = "dokuwiki_syntax.txt"

' Check whether the file to be copied exists.
  If Exist(sSourcePath &/ sFileName) Then
   ' Now check if copyto directory exists
     If Not Exist(sDestinationPath) Then
        Try Mkdir sDestinationPath
```

```
        If Error Then
           Message.Error("Error! " & Error.Text)
           Return
        Endif
     Endif
  ' Now check if copyto file exists
    If Not (Exist(sDestinationPath & sFileName)) Then
       Try Copy sSourcePath &/ sFileName To sDestinationPath &/ sFileName
    Else
     ' If Exists first remove old file then copy new file
       Kill sDestinationPath &/ sFileName
       Try Copy sSourcePath &/ sFileName To sDestinationPath &/ sFileName
       If Error Then
          Message.Error("Error! " & Error.Text)
          Return
       Endif
    Endif
  Else
     Message.Error("File not exist: " & sFileName)
     Return
  Endif

  Message.Info("File '" & sFileName & "' successfully copied!")

End
```

Die Text-Datei *dokuwiki_syntax.txt* aus dem Verzeichnis */home/hans/DW* wird in das Verzeichnis */home/hans/GB3BUCH* kopiert. Diverse Prüfungen und Mitteilungen sichern ein erfolgreiches Kopieren der angegebenen Datei.

6.3.2 DFREE

```
DFree( sPath AS String ) AS Long
```

Gibt die Größe des freien Speicherplatzes (in Bytes) des Massenspeichers zurück, auf dem sich das Verzeichnis *sPath* befindet. Vergleichen Sie das Ergebnis immer mit einem Vergleichswert vom Datentyp Long-Integer, da Speicher-Medien mehr als 2^32 Bytes Kapazität haben können.

Beispiel 1:

```
Dim ilFree As Long

ilFree = DFree(User.Home &/ "DW") ' 346.218.151.936
Print Round(ilFree / 10 ^ 9, -3) & " GB" ' 346.218 GB
```

Das Ergebnis ist so zu interpretieren, dass auf der Festplatte im PC des Autors noch etwa 346 GB freier Speicherplatz zur Verfügung steht.

Beispiel 2:

```
Print DFree("/media/hans/USB-STICK")
```

So ermitteln Sie die Größe des freien Speicherplatzes auf dem angeschlossenen USB-Stick in der Gambas-Konsole.

6.3.3 EXIST

```
Exist( sPath ) As Boolean
```

Zeigt an, ob eine Datei oder ein Verzeichnis mit dem Pfad *sPath* existiert.

Beispiel:

```
If Not Exist(User.Home &/ "Destination") Then
   MkDir(User.Home &/ "Destination")
Endif
```

Das Verzeichnis 'Destination' wird nur dann mit dem angegebenen Pfad angelegt, wenn es nicht existiert.

6.3.4 ISDIR

```
IsDir ( sPath AS String ) AS Boolean
```

Gibt True zurück, wenn die Pfadangabe auf ein Verzeichnis zeigt. Wenn der Pfad *sPath* nicht existiert oder wenn der Pfad auf kein Verzeichnis zeigt, dann gibt diese Funktion False zurück.

Beispiel:

```
Dim sPath As String

sPath = User.Home &/ "test33"
If Not Exist(sPath) Then Mkdir sPath
Print "Is Directory? = "; IsDir(sPath) ' TRUE
sPath = User.Home &/ "test34"
If Not Exist(sPath) Then File.Save(sPath, "CONTENT")
Print "Is Directory? = "; IsDir(sPath) ' FALSE
```

In einem Verzeichnis kann keine Datei den gleichen Namen wie ein Verzeichnis besitzen. Daher die Änderung von test33 auf test34.

6.3.5 KILL

```
KILL FilePath
```

Löscht eine vorhandene Datei, deren (absoluter) Pfad in der Zeichenkette *FilePath* steht.

Beispiel:

```
Dim sSourcePath, sDestinationPath As String

sSourcePath = "./css" &/ fCSSFileName
sDestinationPath = sWorkDirectory &/ "css" &/ fCSSFileName

If Stat(sSourcePath).LastChange > Stat(sDestinationPath).LastChange Then
   Try Kill sDestinationPath
   Try Copy sSourcePath To  sDestinationPath
Endif
```

Wenn eine aktuellere css-Datei im Verzeichnis 'css' im Projekt-Verzeichnis existiert, dann wird die ursprüngliche css-Datei gelöscht und die aktuelle css-Datei in das existierende Arbeitsverzeichnis kopiert.

Wenn Sie dagegen ein *Verzeichnis* löschen möchten, verwenden Sie die folgende Anweisung RMDIR.

6.3.6 RMDIR

```
RMDIR DirectoryPath
```

Löscht das durch *DirectoryPath* angegebene Verzeichnis, das aber leer sein muss!

Beispiel:

```
If RDir(DirectoryPath).Count > 0 Then
   Message.Warning("Das angegebene Verzeichnis ist nicht leer und kann deshalb nicht gelöscht werden.")
   Return
Else
   Try Rmdir DirectoryPath
   Message.Info("Das angegebene Verzeichnis wurde erfolgreich gelöscht!")
Endif
```

Zuerst wird geprüft, ob das angegebene Verzeichnis leer ist. Ist das Verzeichnis leer, dann wird es gelöscht – sonst wird eine Warnung ausgegeben und der Löschvorgang abgebrochen.

Hinweis:
Nutzen Sie als gute Alternative die Funktion Shell.RmDir(DirectoryPath As String) aus der Komponente gb.util. Sie entfernt das angegebene Verzeichnis, indem dessen *kompletter Inhalt rekursiv* entfernt wird!

6.3.7 LINK

```
LINK sOriginalPath TO sSoftLinkPath
```

Erzeugt einen symbolischen Link (Softlink) mit dem Pfad sSoftLinkPath, der auf die Zieldatei oder das Zielverzeichnis sOriginalPath zeigt.

Beispiel:

```
Dim sSoftLinkPath, sOriginalPath As String

sOriginalPath = "/home/hans/DW/0_DW_Convert/librewriter2dokuwiki.gambas"
sSoftLinkPath = "/home/hans/Schreibtisch/Formatting_DokuWiki_Tables"

If Not Exist(sSoftLinkPath) Then
   Try Link sOriginalPath To sSoftLinkPath
   If Error Then Message.Error(Error.Text)
Endif
```

Auf dem Desktop wird eine permanente Verknüpfung mit dem Namen 'Formatting_DokuWiki_Tables' zu einer ausführbaren Gambas-Datei erzeugt.

6.3.8 MKDIR

```
MKDIR DirectoryPath
```

Erzeugt das durch *DirectoryPath* angegebene Verzeichnis. Der MKDIR-Befehl erzeugt standardmäßig ein Verzeichnis mit den Rechten 0755. Benötigen Sie andere Rechte, dann setzen Sie diese mit der CHMOD-Instruktion. Existiert mindestens ein Elternverzeichnis des angegebenen Pfades nicht, so wird ein Fehler ausgelöst.

Beispiel:

```
Dim sDirectoryPath As String
Dim FileInfo As Stat

sDirectoryPath = User.Home &/ "Test18"

If Not Exist(sDirectoryPath) Then
   Mkdir sDirectoryPath
   FileInfo = Stat(sDirectoryPath)
   Print Oct(FileInfo.Mode) ' Ausgabe: 755
Endif
```

Nur wenn das angegebene Verzeichnis nicht existiert, wird es erzeugt. Mit dem Wert der Mode-Eigenschaft des Stat-Objektes FileInfo können Sie sich die Rechte des erzeugten Verzeichnisses in einem geeigneten Format anzeigen.

Hinweis:
Nutzen Sie als Alternative die Funktion Shell.MkDir(DirectoryPath As String) aus der Komponente gb.util, da ein Verzeichnis mit DirectoryPath als Verzeichnispfad und alle seine übergeordneten (Eltern-)Verzeichnisse erzeugt werden.

6.3.9 MOVE .. TO ..

```
MOVE OldName TO NewName
```

Benennt eine Datei oder ein Verzeichnis um oder verschiebt eine Datei oder ein Verzeichnis. Oldname und NewName können sich in verschiedenen Verzeichnissen befinden – müssen sich aber auf demselben Gerät befinden!

Nutzen Sie alternativ auch die Funktion *Move(Source As String, Destination As String)* mit Source als Pfad der Quelldatei und Destination als Zielpfad. Diese Funktion verschiebt eine Datei oder ein Verzeichnis – auch wenn sie sich auf verschiedenen Geräten befinden.

Beispiel 1:

Wenn Sie eine Datei – unabhängig von ihrem Speicherort – verschieben möchten, gehen Sie so vor:

```
Try Move OldFilePath To NewFilePath
If Error Then
   Try Copy OldFilePath To NewFilePath
   If Not Error Then Kill OldFilePath
Endif
```

Eine Datei A mit dem Pfad OldFilePath wird verschoben. Tritt dabei ein Fehler auf, wird die Datei A mit dem Pfad OldFilePath nach NewFilePath kopiert. Tritt dabei kein Fehler auf, wird die Datei A abschließend gelöscht.

Beispiel 2:

```
Public Sub RenameDirectory(OldDirPath As String, NewDirPath As String)

  Try Move OldDirPath To NewDirPath
  If Error Then
     Message.Error("Error when renaming." & gb.Lf & Error.Text & " !")
     Return
  Endif

End

Public Sub btnRenameDirectory_Click()
  RenameDirectory(User.Home &/ "Test18", User.Home &/ "Test19")
End
```

Beim ersten Aufruf der Prozedur wird das Verzeichnis Test18 in Test19 umbenannt. Bei jedem weiteren Aufruf wird ein Fehler ausgelöst und die Fehlermeldung 'Error when renaming. File already exists!' angezeigt.

Beispiel 3:

```
Public Sub MoveDirectory(OldDirPath As String, NewDirPath As String)

  Try Move OldDirPath To NewDirPath
  If Error Then
     Message.Error("Error when moving." & gb.Lf & Error.Text & " !")
     Print Error.Text
     Return
  Endif

End

Public Sub btnMoveDirectory_Click()
  MoveDirectory(User.Home &/ "Test18", User.Home &/ "TWX" &/ "Test18")
End
```

Beim ersten Aufruf der Prozedur wird das Verzeichnis Test18 in das existierende Verzeichnis TWX verschoben. Bei jedem weiteren Aufruf wird ein Fehler ausgelöst und die Fehlermeldung 'Error when renaming. File already exists!' angezeigt.

6.3.10 MOVE .. KILL

```
MOVE OldName KILL NewName
MOVE OldName DOWNTO NewName ' Gilt als veraltet
```

Benennt eine Datei oder ein Verzeichnis um oder verschiebt eine Datei oder ein Verzeichnis, wobei die Zieldatei zuerst gelöscht wird und die beiden Operationen atomar sind. OldName und NewName können sich in verschiedenen Verzeichnissen befinden, müssen sich aber auf demselben Gerät befinden. Normalerweise löst ein Verschieben bei einer bestehenden Datei den Fehler 'Datei bereits vorhanden' aus. In diesem Fall wirkt die Kill-Instruktion wie eine Option, die zuerst die im Ziel vorhandene Datei löscht und erst dann die Datei verschiebt.

Die Funktion ersetzt die Funktionskette MOVE OldName DOWNTO NewName, die als veraltet gilt.

Beispiel:

```
Public Sub btnMoveKill_Click()

  Dim sSourcePath, sDestinationPath, sFilename As String
```

```
  sSourcePath = User.Home &/ "DW"
  sDestinationPath = User.Home &/ "GB3BUCH"
  sFileName = "dokuwiki_syntax.txt"

' Check whether the file to be moved exists.
  If Exist(sSourcePath &/ sFileName) Then
   ' Now check if moveto directory exists
     If Not Exist(sDestinationPath) Then
        Try Mkdir sDestinationPath
        If Error Then
           Message.Error("Error! " & Error.Text)
           Return
        Endif
     Endif
  ' Now the file is moved …
    Move sSourcePath &/ sFileName Kill sDestinationPath &/ sFileName
  Else
     Message.Error("File not exist: " & sFileName)
     Return
  Endif

  Message.Info("File '" & sFileName & "' successfully moved!")

End
```

Variante:

```
Public Sub btnMoveKill_Click()

  Dim sSourcePath, sDestinationPath, sFilename As String

   sSourcePath = User.Home &/ "DW"
   sDestinationPath = User.Home &/ "GB3BUCH"
   sFileName = "dokuwiki_syntax.txt"

  Move sSourcePath &/ sFileName Kill sDestinationPath &/ sFileName
  Message.Info("File '" & sFileName & "' successfully moved!")

  Catch
    Message.Error(Error.Text)

End
```

Die Datei 'dokuwiki_syntax.txt wird nach beiden Varianten verschoben. Wenn eine gleichnamige Datei bereits im Zielverzeichnis existiert, dann wird diese vorher (automatisch) gelöscht – ohne einen Fehler zu erzeugen.

6.3.11 DIR

```
ArrayOfFilenames = Dir ( sDirectory AS String [ , Pattern AS String , Filter AS Integer ] ) AS String[]
```

Die Funktion gibt ein String-Array zurück, das nur die Namen der Dateien und Verzeichnisse im angegebenen Verzeichnis sDirectory enthält, die dem Muster und dem Filter entsprechen. Den Datei-Pfad müssen Sie aus den Datei- oder Verzeichnis-Namen und dem Parameter *sDirectory* zusammenfügen.

- Pattern (optional) kann die gleichen generischen Zeichen enthalten wie der LIKE-Operator. Wenn kein Muster angegeben wird, so wird der Name jeder gefundenen Datei und jedes gefundene Verzeichnisses zurückgegeben.
- Filter (optional) gibt an, ob nur die Namen von Dateien oder nur von Verzeichnissen oder von beiden zurückgegeben werden. Wenn kein Filter angegeben wird, so werden die Namen von allen Dateien und Verzeichnissen im angegebenen Verzeichnis zurückgegeben, die dem Muster entsprechen.

Der Filter kann einer der folgenden Konstanten sein:

- gb.Datei, um nur Dateien zurückzugeben.
- gb.Directory, um nur Verzeichnis-Namen zurückzugeben,
- gb.Datei + gb.Directory für die Rückgabe beider.

Durch die geeignete Festlegung von Muster und Filter ist zum Beispiel eine gezielte Suche nach einer einzelnen Datei oder nach einer bestimmten (Teil-)Menge aller Dateien im angegebenen Basis-Verzeichnis möglich.

Beispiel 1:

```
[1]  Public Sub ScanDirectory()
[2]
[3]    Dim sDirectoryPath, sPattern, sDirFileName As String
[4]    Dim sFilter As Integer
[5]    Dim bMode, bSorted As Boolean
[6]
[7]    sDirectoryPath = User.Home &/ "BildTon"
[8]    sPattern = "[^0-9P]*.{png,jpg,gif}"
[9]    sFilter = gb.File  ' gb.Directory | gb.File + gb.Directory
[10]   bSorted = True
[11]   bMode = gb.Ascent  ' gb.Descent | gb.Ascent
[12]
[13]   If bSorted Then
[14]      If Dir(sDirectoryPath, sPattern, sFilter).Sort(bMode).Count = 0 Then
[15]         Message.Info("The searched set is empty ...")
[16]         Return
[17]      Else
[18]         For Each sDirFileName In Dir(sDirectoryPath, sPattern, sFilter).Sort(bMode)
[19]           Print sDirFileName
[20]         Next
[21]      Endif
[22]   Else
[23]      If Dir(sDirectoryPath, sPattern, sFilter).Count = 0 Then
[24]         Message.Info("The searched set is empty ...")
[25]         Return
[26]      Else
[27]         For Each sDirFileName In Dir(sDirectoryPath, sPattern, sFilter)
[28]           Print sDirFileName
[29]         Next
[30]      Endif
[31]   Endif
[32]
[33] End
[34]
[35] Public Sub btnScanDirectory_Click()
[36]   ScanDirectory()
[37] End
```

Mit Hilfe der Prozedur ScanDirectory() können Sie sich nach der Festlegung des Verzeichnisses, des Musters, des Filters, der Sortierung und der Sortierreihenfolge (Zeilen 7 bis 11) die Menge der Suchergebnisse in der Konsole ausgeben lassen.

Das Muster im Beispiel 1 ist so gewählt, dass nur Dateien mit der Extension png, jpg und gif in der Ergebnismenge sind – deren Namen aber weder mit einer Ziffer aus der Menge [0-7] noch mit einem P beginnen. Sie können aber auch in den Zeilen 19 und 28 die Ergebnisse weiter verarbeiten. Denkbar wäre etwa die Speicherung aller Pfade der (Bild-)Dateien der Ergebnismenge in einem String-Array.

In den folgenden Beispielen werden das (Basis-)Verzeichnis, das Muster, der Filter, die Sortierung und die Sortierreihenfolge geändert und die Ergebnisse kommentiert.

Beispiel 2:

```
sDirectoryPath = User.Home
sPattern = "[^.]*.txt"
sFilter = gb.File
bSorted = False
bMode = gb.Ascent
```

Es werden nur nicht-versteckte Dateien mit der Extension .txt im Home-Verzeichnis angezeigt. Die Ergebnismenge wird nicht sortiert.

Beispiel 3:

```
sDirectoryPath = User.Home
sPattern = ".[^A-Z]*"
sFilter = gb.Directory
bSorted = True
bMode = gb.Ascent
```

Es werden nur versteckte Verzeichnisse im Home-Verzeichnis aufgelistet, die nicht mit einem großen Buchstaben beginnen. Die Ergebnismenge wird sortiert.

6.3.12 RDIR

```
ArrayOfFilenames = RDir ( BaseDirectory AS String [ , Pattern AS String , Filter AS Integer , FollowLink AS
Boolean ] ) AS String[]
```

Gibt ein String-Array zurück, das die Namen von den Dateien und Verzeichnissen im angegebenen Basis-Verzeichnis und dessen Unterverzeichnissen enthält, die dem Muster und dem Filter entsprechen. Es gelten die Hinweise zum Muster und zum Filter zur Funktion DIR. Wenn der optionale Parameter *FollowLink* TRUE ist, wird symbolischen Links auf Verzeichnisse gefolgt. Ansonsten werden sie wie normale Dateien verarbeitet.

Beispiel:

```
[1]  Public Sub ScanDirectoryR()
[2]
[3]    Dim sDirectoryPath, sPattern, sDirFileName As String
[4]    Dim sFilter As Integer
[5]    Dim bMode, bSorted As Boolean
[6]
[7]    sDirectoryPath = User.Home &/ "BildTon"
[8]    sPattern = "*fractal_0.png"
[9]    sFilter = gb.File
[10]   bSorted = False
[11]   bMode = gb.Ascent
[12]
[13]   If bSorted Then
[14]      If RDir(sDirectoryPath, sPattern, sFilter).Sort(bMode).Count = 0 Then
[15]         Message.Info("The searched set is empty ...")
[16]         Return
[17]      Else
[18]         For Each sDirFileName In RDir(sDirectoryPath, sPattern, sFilter).Sort(bMode)
[19]           Print sDirFileName
[20]         Next
[21]      Endif
[22]   Else
[23]      If RDir(sDirectoryPath, sPattern, sFilter).Count = 0 Then
[24]         Message.Info("The searched set is empty ...")
[25]         Return
[26]      Else
[27]         For Each sDirFileName In RDir(sDirectoryPath, sPattern, sFilter)
[28]           Print sDirFileName
[29]         Next
[30]      Endif
[31]   Endif
[32]
[33] End
[34]
[35] Public Sub btnScanDirectoryR_Click()
[36]
[37]   ScanDirectoryR()
[38]
[39] End
```

Die zurückgegebenen Datei-Pfade sind relativ zum gesuchten Verzeichnis. Das Muster entspricht aber stets dem *vollständigen relativen Pfad*, nicht nur dem Dateinamen! Das zeigt sich nicht nur im Muster sPattern = "*fractal_0.png", sondern auch auch im Ergebnis des Scans: Fractals/fractal_0.png mit der Angabe des zutreffenden Verzeichnisses. Auch für die rekursive Suche im Verzeichnisbaum gilt: Wählen Sie mit Sorgfalt das Muster und den Filter.

6.3.13 Exkurs 1 – Bestimmung der Größe eines Verzeichnisses

1. Variante – Shell-Instruktion

```
Dim ilSourceSize As Long
Dim sSource, sResult, sProjectFolder As String

sProjectFolder = "GambasBookProjects"
sDestination = "/media/hans/DW_4GB" &/ sProjectFolder
sSource = User.Home &/ sProjectFolder

Exec ["du", "-sb", sSource] To sResult

ilSourceSize = CLong(Split(sResult, "\t")[0])

Print ilSourceSize
```

2. Variante

```
Public Function GetDirSize(sDir As String) As Long

' Coding and adaptation of Tony Morehen and Fabien Bodard

   Dim sFile As String
   Dim sPath As String
   Dim hStat As Stat
   Dim iSize As Long

   For Each sFile In Dir(sDir)
     sPath = sDir &/ sFile
     hStat = Stat(sPath)
     With hStat
       If .Type = gb.Directory Then
         iSize += GetDirSize(sPath) ' Recursion!
       Endif
       iSize += .Size
     End With
   Next

   Return iSize

End

Public Sub btnGetDirSize_Click()

  Dim ilDirSize As Long

  Try ilDirSize = GetDirSize(User.Home &/ "GambasBookProjects")
  If Error Then Message.Error(Error.Text)
  Print ilDirSize

End
```

6.3.14 Exkurs 2 – Datei erzeugen

```
File.Save(sFilePath, "")
File.Save(sFilePath, sContent)
```

Die Anweisungen erzeugen eine leere Datei mit dem angegebenen (absoluten) Datei-Pfad *sFilePath* oder eine Datei mit dem Inhalt sContent. Es wird kein Fehler ausgelöst, wenn die Datei bereits existiert, da dann ihr Inhalt überschrieben wird!

```
Private Sub CreateTextFiles(sPathDir As String, iNumber As String)

  Dim sFileName As String
  Dim i As Integer

  For i = 1 To iNumber
    sFileName = Hex$(Rand(0, 2 ^ 32 - 1)) ' Random file name
    If Not Exist(sPathDir &/ sFileName & ".txt") Then
       File.Save(sPathDir &/ sFileName & ".txt", "Text line 1.\nText line 2.")
    Endif
  Next
End
```

Die Prozedur CreateTextFiles(...) erzeugt eine vorgegebene Anzahl (iNumber) von Text-Dateien – mit Inhalt – mit den angegebenen (absoluten) Datei-Pfaden. Der (Basis)-Datei-Name ist ein Zufallsstring.

6.3.15 Exkurs 3 – Verzeichnis kopieren

Die Aufgabe besteht darin, ein Verzeichnis – mit Inhalt – zu kopieren:

```
Public Sub CopyDir(Source As String, Destination As String)

  Dim sFile As String

  If IsDir(Source) Then
     If Not Exist(Destination) Then Mkdir Destination
     For Each sFile In Dir(Source)
       If Not Exist(Destination &/ sFile) Then
          CopyDir(Source &/ sFile, Destination &/ sFile) ' Recursion
          Wait
       Endif
     Next
  Else
```

```
    If Not Exist(Destination) Then
       Try Copy Source To Destination
       If Error Then Message.Error(Error.Text)
       Return
       Wait
    Endif
  Endif

End

Public Sub btnBackup_Click()

  CopyDir(User.Home &/ "GambasBookProjects", "/media/hans/DW_4GB" &/ "GambasBookProjects")
  Print "DONE!"

End
```