6.1.1 Pfade in Gambas

Pfade in Gambas beziehen sich auf:

- Reguläre Dateien (regular file),
- Verzeichnisse (directory),
- symbolische Links (symbolic link),
- blockorientierte Geräte (block device),
- zeichenorientierte Geräte (char device),
- Datenverbindungen zwischen Prozessen nach dem FIFO-Prinzip (named pipe) und
- Kommunikationsendpunkte (unix-socket).

Sie sollten die Angaben zu den Datei-Pfaden stets so wählen, dass ein Projekt P in den folgenden drei Fällen stets problemlos ausgeführt werden kann:

(F1) Die Anwendung P.gambas wird in der IDE mit F5 oder über die Menüleiste (►) ausgeführt.
(F2) Die Anwendung wird außerhalb der IDE direkt im Projektverzeichnis oder mit `gbr3 path2projectP/P.gambas` oder mit `gbx3 path2projectP_directory` als Kommando in einer Konsole gestartet.
(F3) Die Anwendung wird außerhalb der IDE in einem beliebigen Verzeichnis direkt oder mit `gbr3 ./P.gambas` ausgeführt.

Es gibt in Gambas zwei Arten von Pfaden:

- Absolute Pfade und
- relative Pfade.

6.1.1.1 Absolute Pfade

Absolute Pfade beginnen mit den Zeichen / oder ~. Sie werden wie in einer Shell interpretiert. Wenn ein Pfad mit dem Tilde-Zeichen ~ beginnt – gefolgt von dem Zeichen / – dann wird das Zeichen ~ durch das Home-Verzeichnis des aktuellen Benutzers ersetzt.

Gambas-Pfad	Pfad zeigt auf ...
~/Desktop	/home/hans/Desktop
root/Desktop	/root/Desktop

Tabelle 6.1.1.1.1 : Pfade in Gambas

Verwenden Sie in Ihren Projekten nie absolute Pfade, die auf Ihr Projekt-Verzeichnis zeigen, da diese Pfade nicht mehr existieren, wenn Sie eine ausführbare Datei *.gambas erzeugen! Sie müssen statt dessen relative Pfade verwenden.

6.1.1.2 Relative Pfade

Relative Pfade sind Pfade, die nicht mit dem Zeichen / oder ~ beginnen. Sie beziehen sich auf Dateien oder Verzeichnisse, die sich innerhalb der aktuellen ausführbaren Projektdatei oder der aktuellen Komponente befinden. Relative Pfade beziehen sich *nicht* auf Dateien im aktuellen Arbeitsverzeichnis, da es in Gambas kein Konzept des aktuellen Arbeitsverzeichnisses gibt! Mit der statischen Methode *File.IsRelative(sPath As String)* können Sie prüfen, ob der im Parameter übergebene Pfad *sPath* ein relativer Pfad ist oder nicht.

In Anlehnung an die Gambas-Dokumentation gilt: Dateien, die sich innerhalb des Projektverzeichnisses oder in Unterordnern von diesem befinden und in einer ausführbaren Datei *.gambas archiviert werden, sind lesbar - aber schreibgeschützt. Wenn Sie Ihr Projekt in der IDE ausführen, könnten Projekt-Dateien unter Verwendung absoluter Pfade geändert werden. Aber machen Sie das nicht! Sobald Ihr Projekt als ausführbare Datei ausgeführt wird, existieren diese absoluten Pfade nicht mehr.

Auf eine Nachfrage in der Gambas-Mailing-Liste zum Einsatz relativer Pfade antwortete B. Minisini so:

The current syntax of relative paths is:
'../<path>' ==> access to a file located in the calling component, i.e. the component of the first method

in the stack backtrace that is not in the same component than the current one ; the top-level component being the current project.
'.../<path>' ==> access to a file located in the current project.
'./<component>/<path>' ==> access to a file located in the specified component, provided that, of course, the component has been loaded.
Otherwise, './<path>' or just '<relative path>' ==> access to a file located in the current component.
I hope it's more clear now.
The './<component>/<path>' syntax may change before the 3.12 release.
Maybe for something like '.[<component>]/<path>', or '..<component>/<path>', or something else.

6.1.1.2.1 Syntax relativer Pfade

Diese o.a. Aussagen zur Syntax relativer Pfade wurden in vielen Projekten überprüft. Die Ergebnisse sind in den folgenden Abschnitten und den vorgestellten Beispielen exemplarisch zusammengefasst.

Es wird von der Annahme ausgegangen, dass in einem (Haupt-)Projekt P sowohl eine Komponente C als auch eine Bibliothek L eingebunden sind. Die Komponente C besitzt in ihrem Projekt-Verzeichnis die Datei c.ext und die Bibliothek L hat in ihrem Projekt-Verzeichnis die Datei l.ext:

```
              Datei p.ext
                 |
              Projekt P
_____________________________________

  Komponente C            Bibliothek L
       |                       |
    Datei c.ext             Datei l.ext
```

Beispiel A
Zugriff von P auf Dateien im Projektverzeichnis:

```
Public Sub Form_Open()

  Dim sImagePath As String

  FMain.Icon = Picture.Load(".../symbols/form_icon.png") ' <---- Konsequent diese .../-Syntax verwenden!
  sImagePath = ".../images/hgb.png"
  piboxImage.Picture = Picture.Load(sImagePath)
  ...

End
```

Beispiel B
Zugriff von C auf die (versteckte) Datei `.project` im Projektverzeichnis:

```
' Gambas class file

''' Die Klasse C der Komponente C implementiert nur 2 Eigenschaften.
''' Die Klasse dient nur zur Demonstration des Zugriffs aus einer Komponente auf Dateien im Hauptprogramm.

Export

Property Read Version As String
Property Read Vendor As String
'------------------------------
Private $sVersion As String
Private $sVendor As String
Private $sContent As String
Private $aContent As String[]

Public Sub _new()

  Dim sRow As String

  If Exist("../.project") Then
     $sContent = File.Load("../.project") ' <---- ../-Syntax
     $aContent = Split($sContent, gb.NewLine)
     For Each sRow In $aContent
       If sRow Begins "Version" Then $sVersion = Scan(sRow, "*=*")[1]
       If sRow Begins "Vendor" Then $sVendor = Scan(sRow, "*=*")[1]
     Next
  Endif

End
```

```
Private Function Vendor_Read() As String
  Return $sVendor
End

Private Function Version_Read() As String
  Return $sVersion
End
```

Beispiel C
Zugriff von L auf die Datei p.ext in P:

```
sContent = File.Load("../p.ext") ' <---- ../-Syntax
```

Beispiel D - Zugriff von P auf die Datei c.ext in C
Beispiel E - Zugriff von P auf die Datei l.ext in L

Für die letzten beiden Fälle sollten Sie jeweils in C und L in einem Modul eine private Funktion deklarieren, die den Inhalt von c.ext und l.ext in einer nur lesbaren Eigenschaft bereitstellt. Ein Modul ist in diesem Fall gut geeignet, da Sie nur *eine* Instanz der statischen Klasse benötigen.

Im folgenden konkreten Beispiel wird der Inhalt der Hilfe-Datei *help_mqt.txt* in der Komponente C im Modul MHelp.module als nur lesbare Eigenschaft *Help* bereitgestellt:

```
' Gambas module file

''' Das Module MHelp implementiert nur eine Eigenschaft: Help

Export

Property Read Help As String

Private Function Help_Read() As String

  If Exist("./help_mqt.txt") Then
     Return File.Load("./help_mqt.txt") ' <---- ./-Syntax
  Else
     Error.Raise("Help file not found!")
  Endif

End
```

Im Downloadbereich finden Sie die beiden Quelltext-Archive für die Komponente C und das Hauptprogramm P.

6.1.1.3 Pfadangaben in gambas-spezifischem Format

6.1.1.3.1 1. Fall

Viele Steuer-Elemente besitzen die Eigenschaft .Picture, der Sie zum Beispiel mit btnClose.Picture = Picture["icon:/16/close"] ein passendes Icon aus dem Bestand von Gambas zuweisen können. Die Bilder aus dem Bestand von Gambas (in /usr/lib) befinden sich in der Datei gb.form.stock.gambas. Sie liegen nur in diesem Archiv vor! Wenn die Picture-Klasse einen Pfad in gambas-spezifischem Format mit "icon:/" am Anfang erkennt, dann wird in den kompilierten Dateien gesucht und das Bild mit Hilfe der Pfad-Auflösung des Gambas-Interpreters für relative Pfade aus dem Archiv geladen.

Die folgenden alternativen Zuweisungen können Sie für den Einsatz eigener Bilder – gespeichert zum Beispiel im Bild-Verzeichnis `leds` im Projektverzeichnis – nutzen:

```
pboxOnOff.Picture = Picture.Load(".../leds/green16.png")
pboxOnOff.Picture = Picture["leds/green16.png"]
```

Diese Zuweisung löst keinen Fehler aus – es wird aber kein Bild dargestellt:

```
pboxOnOff.Picture = Picture[.../leds/green16.png"]
```

Benötigen Sie passende Icon (farbig/grau) aus dem Bestand von Gambas für Ihre Projekte, dann finden Sie diese im Quelltext von Gambas im Verzeichnis ../comp/src/gb.form.stock/gambas/32. Diese können Sie dann zum Beispiel in einem Unter-Ordner /icons des Projekt-Verzeichnisses abspeichern.

6.1.1.3.2 2. Fall

Wenn Sie eine Gambas-Bibliothek zur Laufzeit laden, dann können Sie das zum Beispiel mit Component.Load(":gambasbook/mylibrary:2.1") realisieren. Die Pfadangabe in gambas-spezifischem Format beginnt mit einem Doppelpunkt, dem der Vendor-Name ´gambasbook´, das Zeichen "/" sowie der Name der Bibliothek ´mylibrary´ folgt. Nach einem weiteren Doppelpunkt steht die Angabe der Version ´2.1´. Der Interpreter sucht in diesen Pfaden nach Gambas-Bibliotheken: ~/.local/share/gambas3/lib/ oder in /usr/lib/gambas3/.

Wenn Sie eine externe Bibliothek in Gambas einbinden, dann folgt der Pfad der Syntax "NameOfExternalLibrary:VersionOfExternalLibrary", wie er in den beiden folgenden Beispielen verwendet wird:

```
LIBRARY "libc:6"
EXTERN getgid() AS Integer
```

```
EXTERN getgid() AS Integer IN "libc:6"
```

6.1.1.3.3 3. Fall

Für das Laden einer Gambas-Komponente reicht es aus, wenn Sie als spezifischen Pfad den Namen der Komponente angeben:

```
Public Sub _new()
  Component.Load("gb.desktop")
' Print Component.IsLoaded("gb.desktop")
End
```

6.1.1.3.4 4. Fall

Wenn Sie einen Datei- oder Verzeichnis-Auswahldialog einsetzen, dann sollten Sie genau prüfen, ob der vollständige Pfad oder nur der Name als Resultat zur Verfügung steht. So wird zum Beispiel mit der Funktion IsDir(…) immer der vollständige Pfad geprüft.

6.1.1.4 Dateinamen

Für einen Dateinamen in Gambas sind alle Zeichen außer dem Schrägstrich "/" und dem Null-Zeichen (NUL oder U+2400) erlaubt.

6.1.1.5 Exkurs: Aufbau eines Projekt-Verzeichnisses

Auf einem Datenträger liegt ein Gambas-Projekt in einem Verzeichnis mit einer gambas-spezifischen Verzeichnis-Struktur vor. Das Projekt-Verzeichnis und dessen Basis-Struktur werden beim Erzeugen eines Projektes in der integrierten Entwicklungsumgebung (IDE) *automatisch* erzeugt. Der Verzeichnisname des erzeugten Projekts ist der Projektname! Öffnen Sie ein Projektverzeichnis über Ihren Datei-Browser und schalten Sie in diesem die Anzeige versteckter Dateien/Verzeichnisse ein, so sehen Sie neben einigen versteckten Dateien auch versteckte Verzeichnisse, die Gambas für Steuerdateien verwendet sowie Dateien und Verzeichnisse, die Sie selbst in das Projekt-Verzeichnis kopiert oder dort angelegt haben. Gerade Einsteiger sind überrascht, wie stark sich die Verzeichnisstruktur von der Anzeige im Browser der IDE unterscheidet, wenn Sie ein Projekt in der IDE öffnen. Um den Unterschied zu beschreiben gilt: Ein Verzeichnis wird "physisch" genannt, wenn es auf einem Datenträger liegt und Sie es mit dem Datei-Browser betrachten können. Ein Verzeichnis dagegen ist vom Typ "logisch", wenn die Gambas-IDE einem bestimmten physischen Verzeichnis eine besondere Bedeutung beimisst. Im Datei-Browser sehen Sie nur physische Verzeichnis, in der Gambas-IDE vorwiegend logische (Projekt (.hidden), Quellen (.src) und Daten). Dem logischen Verzeichnis "Projekt" in der IDE entspricht das physische Verzeichnis .hidden. Dateien und Verzeichnisse, die Sie in das physische Projekt-Verzeichnis kopiert oder dort angelegt haben, sehen Sie in der IDE im logischen Verzeichnis "Daten". Nur Verzeichnisse und Dateien aus dem Verzeichnis "Daten" werden beim Kompilieren in die ausführbare Datei *.gambas aufgenommen. Es gibt aber auch Dateien - wie Dateien vom Typ *.class - die Sie nicht im logischen Verzeichnis .src sehen! Sie können diese Dateien jedoch mit F12 oder über den entsprechenden Button im Editor der IDE öffnen. Eine detaillierte Beschreibung finden Sie im Kapitel 2.6.0.7 Verzeichnispfade.

Beachten Sie:
Nur Dateien, die Sie im versteckten Verzeichnis .hidden ablegen, werden in der IDE im Verzeichnis "Projekt" angezeigt und nur Dateien aus diesem Verzeichnis können Sie in ein Verzeichnis Ihrer Wahl

auf dem Ziel-Rechner kopieren, wenn Sie in der IDE ein *Installationspaket* schnüren. Eine Beschreibung finden Sie auf https://www.gambas-buch.de/doku.php?id=k11:k11.10:start.

6.1.1.6 Vorgaben für Pfade in Gambas

6.1.1.6.1 Ausführbare Datei *.gambas

Wenn Sie in der IDE eine ausführbare Datei erzeugen, dann wird die Archiv-Datei *.gambas automatisch im Projekt-Verzeichnis abgespeichert. Der Dateiname des Archivs ist gleich dem Projektnamen, den Sie im Dialog bei der Erzeugung eines neuen Projektes angegeben hatten. Den Namen der Datei und den Speicherort können Sie aber auch im Dialog in der IDE frei wählen.

Alternativ können Sie die Archiv-Datei aber auch so in zwei Schritten erzeugen:

```
hans@mint-183 ~ $ gbc3 -ag $HOME/GB3BUCH/6K_Stream/6.1_Pfade/BuchProgramm/PathProject
OK
hans@mint-183 ~ $ gba3 $HOME/GB3BUCH/6K_Stream/6.1_Pfade/BuchProgramm/PathProject
```

Es wird empfohlen, die Archiv-Datei später nicht umzubenennen, weil das zu unerwarteten Seiten-Effekten führen kann, wenn Sie die Eigenschaft *Application.Name* als Implementationsdetail verwenden.

6.1.1.6.2 Gambas-Bibliothek

Wenn Sie aus einem Projekt vom Typ 'Bibliothek' eine ausführbare Datei library_name.gambas erzeugen, dann sorgt die IDE dafür, dass die Bibliothek automatisch im Projekt-Verzeichnis abgespeichert und zusätzlich auch im Bibliothek-Basis-Verzeichnis mit dem Pfad

```
~/.local/share/gambas3/lib/<vendorname>/<projektname:version1.version2.gambas>.
```

Die Bibliothek wie zum Beispiel: ~/.local/share/gambas3/lib/gambasbook/libmath:2.3.gambas und deren exportierte(!) Klassen und Methoden kann so in anderen Projekten verwendet werden.

Verwendet Ihr Projekt eine selbst entwickelte Gambas-Bibliothek, so kennt der Gambas-Interpreter die Pfade zu den Bibliothek-Basis-Verzeichnissen, in denen er suchen muss. Sie müssen in den Projekt-Eigenschaften unter ´Bibliotheken´ die passende Bibliothek jeweils explizit hinzufügen und bestätigen! Dieser benutzer-basierte Pfad ist eigentlich recht praktisch, da Sie als Entwickler und/oder Tester stets die neueste Version der Bibliothek verwenden können. Das bedeutet, dass Sie Ihre Test-Version nutzen, ohne eine u.U. existierende systemweite, stabile Version überschreiben zu müssen. Verwenden Sie eine externe Gambas-Bibliothek, dann sollte der Pfad entweder so:

(Fall 1) ~/.local/share/gambas3/lib/$vendor oder mit
(Fall 2) /usr/lib/gambas3/$vendor

beginnen.

- Der erste Fall liegt vor, wenn Sie eine Bibliothek library_name.gambas als einfacher Benutzer installieren oder Sie eine Bibliothek aus der Gambas-Farm installieren.
- Der zweite, wenn Sie die Bibliothek als Root installieren. In diesem Fall gilt der Pfad systemweit. Hinweis: Wurde der Projekttyp in den Projekt-Eigenschaften auf ´Bibliothek´ gesetzt, ist der Vendorname zwingend erforderlich. Den Wert von Vendor in den Projekt-Eigenschaften sollten Sie mit einem passenden Entwicklernamen (Vendorname) belegen. Wenn Sie dort keinen Entwicklernamen eingeben, dann fügt Gambas als Wert für Vendor automatisch ´(unknown)´ ein. Ein problematisches Vorgehen, wenn Sie den Pfad zu einer Bibliothek zum Beispiel in einer Konsole einsetzen wollen, denn es erscheint diese Fehlermeldung: *-bash: Syntaxfehler beim unerwarteten Wort ´(´* !