6.1.2 Basisverzeichnisse

Bevor Sie sich den statischen Eigenschaften Desktop.ConfigDir, Desktop.DataDir, Desktop.RuntimeDir und Desktop.CacheDir der Klasse Desktop (gb.desktop) zuwenden, sollten Sie sich die Informationen auf https://specifications.freedesktop.org/basedir-spec/basedir-spec-latest.html [1] genau durchlesen. Das Kapitel in [1] vermittelt eine gute Einführung in das Konzept der Basis-Verzeichnisse, die durch Umgebungsvariablen wie $XDG_CONFIG_HOME oder $XDG_DATA_HOME definiert werden, um zwei zu nennen.

In einer Konsole wurden für ausgewählte Umgebungsvariablen diese Werte für den Benutzer ´hans´ (das ist der Autor) ausgelesen:

```
hans@mint-183 ~ $ echo $XDG_CONFIG_HOME
(leer oder nicht gesetzt)
hans@mint-183 ~ $ echo $XDG_DATA_HOME
(leer oder nicht gesetzt)
hans@mint-183 ~ $ echo $XDG_RUNTIME_DIR
/run/user/1000
hans@mint-183 ~ $ echo $XDG_CACHE_HOME
(leer oder nicht gesetzt)
```

Mit diesem Quelltext-Ausschnitt können Sie die aktuellen Pfade auf Ihrem System abfragen, wie Gambas sie ausgelesen und für den Fall, dass sie leer oder nicht gesetzt sind, definiert hat:

```
Print "ConfigDir  = "; Desktop.ConfigDir    ' $XDG_CONFIG_HOME
Print "DataDir    = "; Desktop.DataDir      ' $XDG_DATA_HOME
Print "RuntimeDir = "; Desktop.RuntimeDir   ' $XDG_RUNTIME_DIR
Print "CacheDir   = "; Desktop.CacheDir     ' $XDG_CACHE_HOME
```

In der Konsole der Gambas-IDE ergaben sich beim Autor folgende Ausgaben (B):

```
ConfigDir  = /home/hans/.config
DataDir    = /home/hans/.local/share
RuntimeDir = /run/user/1000
CacheDir   = /home/hans/.cache
```

Der Unterschied der beiden Ausgaben (A) und (B) ist schnell erklärt: Wenn eine von den o.a. Umgebungsvariablen leer oder nicht gesetzt ist, dann wird von Gambas nach der Spezifikation in [1] auf Standard-Verzeichnisse zurückgegriffen.

Dabei gilt zusätzlich: Ist die Umgebungsvariable $XDG_RUNTIME_DIR leer oder nicht gesetzt, dann sollten Anwendungen auf ein Ersatzverzeichnis wie /tmp mit ähnlicher Funktionalität zurückgreifen. In der Datei im Gambas-Quelltext gambas_source_code/comp/src/gb.desktop/.src/Desktop.class sind die Eigenschaften Desktop.ConfigDir, Desktop.DataDir, Desktop.RuntimeDir und Desktop.CacheDir der Klasse Desktop (gb.desktop) als Funktionswert so definiert:

```
Static Private Function ConfigDir_Read() As String
  Return GetDir("XDG_CONFIG_HOME", User.Home &/ ".config")
End
Static Private Function DataDir_Read() As String
  Return GetDir("XDG_DATA_HOME", User.Home &/ ".local/share")
End
Static Private Function CacheDir_Read() As String
  Return GetDir("XDG_CACHE_HOME", User.Home &/ ".cache")
End
Static Private Function RuntimeDir_Read() As String
  Dim sPath As String = Env["XDG_RUNTIME_DIR"]
  If Not sPath Then
    sPath = File.Dir(Temp$())
    Error "gb.desktop: warning: XDG_RUNTIME_DIR is not set default to "; sPath
  Endif
  Return sPath
End
```

6.1.2.1 Basis-Verzeichnis 1 – Desktop.ConfigDir

```
Static Property Read ConfigDir As String
```

Gibt das Basis-Verzeichnis zurück, in dem benutzer-spezifische Konfigurationsdateien gespeichert werden sollen.

Beispiele:

```
Public sConfigDir As String

Public Sub _new()
  ...
  sGlobalConfigDir = Desktop.ConfigDir &/ sVendor &/ Lower(sAppName) ' ~/.config/gambasbook/pathproject
  If Not Exist(sGlobalConfigDir) Then Shell.MkDir(sGlobalConfigDir)
  ...
  hSettings = New Settings(sGlobalConfigDir &/ File.SetExt(Lower(sAppName), "conf"))
' ~/.config/gambasbook/pathproject/pathproject.conf
  ...
End
```

```
Public Sub Form_Open()
  ...
  If hSettings["First/Value", 0] < 2 Then ' Das 'About'-Fenster wird genau 2 mal angezeigt
     i = hSettings["First/Value", 0]
     Inc i
     hSettings["First/Value"] = i
     hSettings.Save()
     FAbout.ShowModal()
  Endif

End
```

Einige Gambas-Programme hinterlegen Skripte – zum Beispiel in Shell, Perl, Python oder Gambas – als `Hooks`, die vom Gambas-Programm bei bestimmten Ereignissen ausführt werden. Damit greift das Skript in die Konfiguration des Programms ein. Diese Skripte beziehungsweise das vereinbarte Verzeichnis, das von Ihrem Programm durchsucht wird, sollte dann im Basis-Verzeichnis Desktop.-ConfigDir liegen.
In der Programmierung bezeichnet `Hook` nach https://de.wikipedia.org/wiki/Hook_(Informatik) eine Schnittstelle, mit der Programm-Quelltext anderer Sprachen in ein Programm integriert werden kann, um dieses zu erweitern, deren Ablauf zu verändern oder um bestimmte Ereignisse abzufangen.

6.1.2.2 Basis-Verzeichnis 2 – Desktop.DataDir

```
Static Property Read DataDir As String
```

Gibt das Basis-Verzeichnis zurück, in dem benutzer-spezifische Daten-Dateien (permanent) gespeichert werden sollen.

6.1.2.2.1 Beispiel 1

In diesem Basis-Verzeichnis werden u.a. die Projekte aus der Gambas-Software-Farm abgespeichert, die man im Start-Fenster der IDE unter ´Software-Farm ...´ im Dialog ausgewählt und installiert hat. Beispiel: /home/hans/.local/share/gambas3/src/ajmsoftware/gbAutoMount oder Desktop.DataDir &/ "gambas3/src/ajmsoftware/gbAutoMount".

Die Angabe `ajmsoftware` steht für den Entwicklernamen (Vendor) und `gbAutoMount` für den Projektnamen.

6.1.2.2.2 Beispiel 2

Skripte (Shellskripte und Skripte in weiteren Sprachen wie Gambas oder Python oder Perl oder PHP), die Sie in das Projekt-Verzeichnis kopiert oder dort neu angelegt haben, können Sie in der IDE problemlos ausführen, wenn die Skript-Datei über die notwendigen Rechte verfügt, die für den Fall einer neuen Skript-Datei mit ´rw-rw-r--´ gesetzt sind. Deshalb müssen Sie diese Rechte so ändern, dass mindestens der Besitzer das Recht zum Ausführen der Skript-Datei besitzt.

1. Fall - Skript-Test in der Gambas-IDE

Während die Anweisung

```
txaResult.Insert(File.Load(sScriptPath))
```

mit dem Pfad sScriptPath = ".../scripts/test_script.sh" den Inhalt der Skript-Datei im Projektverzeichnis in der TextArea txaResult anzeigt, erzeugt die folgende Anweisung einen Fehler:

```
CHMOD sScriptPath To "..x......"
```

Der Grund liegt darin, dass die CHMOD-Instruktion – genauso wie die SHELL-Instruktion – einen absoluten Pfad fordert! Deshalb funktionieren die folgenden beiden Anweisungen ohne Probleme:

```
CHMOD SetFullPath(sScriptPath) To "..x......"
SHELL SetFullPath(sScriptPath) To sStore
```

Das ist der Quelltext für die Funktion SetFullPath(sPath As String):

```
Public Function SetFullPath(sPath As String) As String

  Dim sFullPath As String

  If Left(sPath, 1) = "~" Then
     sFullPath = User.Home & Mid(sPath, 1 + 1)
  Else If Exist(sPath) And Left(sPath, 1) <> "/" Then
     sFullPath = Application.Path &/ sPath
  Else
     sFullPath = sPath
  Endif

  Return sFullPath

End
```

2. Fall - Skript-Test im Projektverzeichnis

Einen völlig anderen Ansatz müssen Sie wählen, wenn Sie eine ausführbare Datei *.gambas erzeugen, diese im Projekt-Verzeichnis öffnen und dort das Programm starten.

```
hans@mint-183 ~/Schreibtisch/RunScriptCHMOD $ gbr3 ./runscript.gambas
```

Der Grund liegt darin, dass Dateien in einem Gambas-Archiv nur gelesen werden können – das Ausführen scheitert! Der folgende Ansatz hat sich für diesen Fall bewährt:

- Zuerst werden die benötigten Verzeichnisse angelegt, wenn sie noch nicht existieren.
- Anschließend wird die Skript-Datei zur Laufzeit in ein geeignetes Basis-Verzeichnis kopiert.
- Dann werden die erforderlichen Rechte der *Kopie der Skript-Datei* gesetzt.
- Danach wird das Shell-Skript ausgeführt.
- Abschließend kann das Skript gelöscht werden.

Quelltext:

```
Public Sub btnRunScript_Click()

  Dim sStore As String

  sRelativeScriptPath = ".../scripts/test_script.sh"
  sAbsoluteScriptPath = Desktop.DataDir &/ File.BaseName(sRelativeScriptPath) & ".sh"

  Copy sRelativeScriptPath To sAbsoluteScriptPath
' Chmod sAbsoluteScriptPath To "r-xr-xr-x" ' Es werden alle Rechte explizit neu gesetzt
  Chmod sAbsoluteScriptPath To "..x......"
' Es wird nur das Recht "Ausführen" für den Besitzer gesetzt - alle anderen bleiben erhalten!

  txaResult.Clear()
  Shell sAbsoluteScriptPath To sStore
  If sStore Then
     txaResult.Text = sStore
     txaResult.Pos = 0
  Else
     Message.Error(Error.Text & "!")
  Endif

  If Exist(sAbsoluteScriptPath) Then Kill sAbsoluteScriptPath

End
```

Oder Sie entscheiden sich für das Kopieren der Skript-Datei in eine temporäre Datei und wählen als Basis-Verzeichnis /tmp. Da das Skript nur temporär gebraucht wird, ändert sich der o.a. Quelltext:

```
Public Sub btnRunScript_Click()
```

```
  Dim sStore As String

  sRelativeScriptPath = "scripts/test_script.sh"
  sTempScriptPath = Temp(File.BaseName(sRelativeScriptPath)) & ".sh"
' Absoluter Pfad: /tmp/gambas.1000/4979/test_script.tmp.sh
  If Not Exist(sTempScriptPath) Then Copy sRelativeScriptPath To sTempScriptPath
' Chmod sTempScriptPath To "r-xr-xr-x"  ' Es werden alle Rechte explizit neu gesetzt
  Chmod sTempScriptPath To "..x....."
' Es wird nur das Recht "Ausführen" für den Besitzer gesetzt - alle anderen bleiben erhalten

  txaResult.Clear()
  Shell sTempScriptPath To sStore
  If sStore Then
     txaResult.Text = sStore
     txaResult.Pos = 0
  Else
     Message.Error(Error.Text & "!")
  Endif
End
```

Gut zu wissen: Die Datei mit dem Pfad sTempScriptPath wird *automatisch* gelöscht, wenn das Gambas-Programm beendet wird.

Bespiel 3

Wenn Sie einen Blick in den Quelltext (app/src/gambas3/.src/Project/Farm/CSoftware.class) in Bezug auf die Software zur Pflege der Datenbank von installierter Software aus der Software-Farm werfen:

```
Public Sub GetInstalledDir() As String
  Return Desktop.DataDir &/ "gambas3/src" &/ LCase(Vendor) &/ Name
End
```

erkennen Sie, dass als Installationsverzeichnis Desktop.DataDir gewählt wird. Der Name des Entwicklers kann über den Eintrag ´Vendor´ in den Projekt-Eigenschaften angegeben werden.

6.1.2.3 Basis-Verzeichnis 3 – Desktop.CacheDir

```
Static Property Read CacheDir As String
```

Gibt das Basis-Verzeichnis zurück, in dem benutzer-spezifische, nicht unbedingt benötigte Dateien gespeichert werden sollen. Es wird durch die Umgebungsvariable $XDG_CACHE_HOME definiert. Ist $XDG_CACHE_HOME nicht gesetzt oder leer ist, so wird die Voreinstellung ~/.cache verwendet.

6.1.2.4 Basis-Verzeichnis 4 – Desktop.RuntimeDir

```
Static Property Read RuntimeDir As String
```

Gibt das (Basis-)Verzeichnis /run/user/User-ID wie /run/user/1000 zurück, in dem benutzer-spezifische, nur zur Laufzeit benötigte Dateien und andere Dateiobjekte wie zum Beispiel lokale Unix-Sockets oder Named Pipes (temporär) gespeichert werden sollen. Das Verzeichnis muss dem Benutzer gehören. Er muss der Einzige sein, der Lese- und Schreibzugriff darauf hat. Sein Unix-Zugriffsmodus muss 0700 sein. Das Verzeichnis wird durch die Umgebungsvariable $XDG_RUNTIME_DIR definiert:

```
hans@mint-183 ~ $ echo $XDG_RUNTIME_DIR
/run/user/1000
```

Das Verzeichnis /run ein virtuelles, temporäres Dateisystem. Es existiert im Arbeitsspeicher und wird automatisch geleert, wenn der Computer neu gestartet wird.

Beispiel

```
Public Sub _new()
 ...
  sGlobalRuntimeDir = Desktop.RuntimeDir &/ sVendor &/ Lower(sAppName)
' /run/user/1000/gambasbook/pathproject
  If Not Exist(sGlobalRuntimeDir) Then Shell.MkDir(sGlobalRuntimeDir)
  ...
  sGlobalScriptsDir = sGlobalRuntimeDir &/ GetDirPath(sLocalScriptsDir)
' /run/user/1000/gambasbook/pathproject/scripts
  If Not Exist(sGlobalScriptsDir) Then
```

```
    Shell.MkDir(sGlobalScriptsDir)
  Endif
  For Each sFile In Dir(sLocalScriptsDir, "*.{sh,sql,pl,py,gbs}")
    If Not Exist(sGlobalScriptsDir &/ sFile) Then
       Copy sLocalScriptsDir &/ sFile To sGlobalScriptsDir &/ sFile
       Chmod sGlobalScriptsDir &/ sFile To "..x......"
    Endif
  Next
  ...
End
```

- Zuerst werden alle Skripte mit dem passenden Filter für die Extension *temporär* im Basis-Verzeichnis sGlobalScriptsDir gespeichert: /run/user/user_id/gambasbook/pathproject/scripts).
- Für jede kopierte Skript-Datei wird das Recht zum Ausführen (Benutzer) gesetzt.

So erfolgt der Aufruf des Skripts *dump.sh* in einer EXEC-Instruktion:

```
Private Sub GetDBDumpExec()

  Dim aExecCommand As String[]

' The file extension .sql is automatically supplemented with the Dump command!
  aExecCommand = [sTempScriptPath, sGlobalDBHostDir &/ sDBName, sGlobalDBHostDir &/ "dump." & sDBTableName]
  Exec aExecCommand Wait
End
```

Im Verzeichnis ~/.local/share/gambasbook/pathproject/data/databases wird die Datei *contacts.sql* wird gespeichert.

Die Verwendung von Basis-Verzeichnissen für die permanente Speicherung ausgewählter Dateien erfordert für ein Projekt u.U. drei verschiedene Basis-Verzeichnisse einzusetzen. Das fördert sicher den Gedanken, die Basis-Pfade für bestimmte Dateien stets in gleicher Weise zu nutzen.

Es spricht m.E. aber auch Nichts dagegen, *alle relevanten Dateien für ein Projekt* in einem Basis-Verzeichnis wie Desktop.DataDir mit nachgestelltem NameSpace/ProjektName permanent zu speichern.

6.1.2.5 Application.Name, Application.Dir und Application.Path

Die Verwendung der Eigenschaften Application.Name, Application.Path und Application.Dir der Klasse Application (gb) birgt einige Überraschungen. Wenn Sie diese Eigenschaften in Ihren Projekten verwenden wollen, dann sollten Sie sich die folgenden Abschnitte genau durchlesen. Die vorgestellten Überlegungen gelten auch für den Projekt-Typ 'Bibliothek' und 'Komponente'.

6.1.2.5.1 Application.Name

In der Dokumentation steht: 'Return the application name, as defined in the IDE project properties dialog.' Dieser Text ist ungenau und müsste so geändert werden:

'Gibt den Namen der auszuführenden Anwendung zurück.'

- Wenn die Anwendung in der IDE ausgeführt wird, ist es der Projekt-Name – so wie er im Dialog "Neues Projekt" definiert wurde.
- Der Projekt-Name ist auch Name des automatisch angelegten Projekt-Verzeichnisses.
- Wenn die Anwendung außerhalb der IDE direkt oder mit `gbr3 path2project/archive_name.gambas` ausgeführt wird, ist es der Name der ausführbaren Datei archive_name.gambas.
- Wenn die Anwendung außerhalb der IDE mit `gbx3 path2project_directory` ausgeführt wird, ist es der Name des Projekt-Verzeichnisses.'

Die folgenden 4 Fälle demonstrieren den Einsatz der Eigenschaft *Application.Name* in einem Projekt.

1. Fall: Ausführung der Anwendung in der IDE - Projekt-Name 'GetAppName'

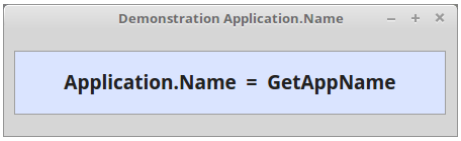

Abbildung 6.1.2.5.1: Start in der IDE

2. Fall: Ausführung im Projektverzeichnis GetAppName

```
$ gbx3 ~/GB3BUCH/6K_Stream/6.1_Pfade/BuchProgramm/GetAppName
```

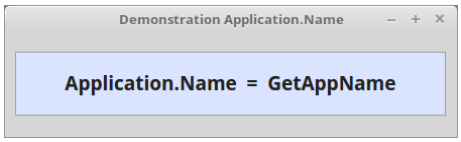

Abbildung 6.1.2.5.2: Start im Projektverzeichnis

Fall 3: Ausführung im Projektverzeichnis GetAppName

Der Name der ausführbaren Datei wurde vom (Standard-)Namen GetAppName.gambas in get.app.name.gambas umbenannt

```
$ gbr3 ~/GB3BUCH/6K_Stream/6.1_Pfade/BuchProgramm/GetAppName/get.app.name.gambas
```

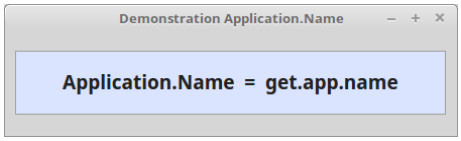

Abbildung 6.1.2.5.3: Start im Projektverzeichnis

4. Fall - Ausführung einer Kopie des Projektverzeichnisses

Das originale Projekt-Verzeichnis ../GetAppName wurde nach ../GetAppNameCopy kopiert und dort das Programm gestartet.

```
hans@mint-183 ~ $ gbx3 ~/GB3BUCH/6K_Stream/6.1_Pfade/BuchProgramm/GetAppNameCopy
```

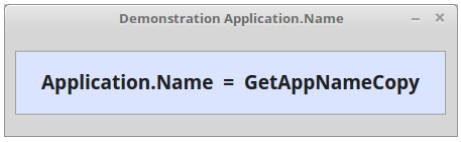

Abbildung 6.1.2.5.4: Start in einer Kopie des Projektverzeichnisses

Fazit: Sie sollten davon absehen, ein Projekt-Verzeichnis oder das existierende Gambas-Archiv *.gambas – ohne Notwendigkeit – umzubenennen, wenn Sie die Eigenschaft *Application.Name* als Implementationsdetail in Ihrem Projekt sicher verwenden wollen.

6.1.2.5.2 Application.Dir

Zur Eigenschaft Application.Dir steht in der Dokumentation:

'Returns the application directory. It is the current working directory defined at application startup.'
'Gibt das Programmverzeichnis zurück. Es ist das aktuelle, beim Programmstart definierte Arbeitsverzeichnis.'

Hier ein Vorschlag für eine erweiterte Definition:

A) Returns the application directory as default working directory if in Gambas execution mode.
B) Returns the user home directory as default working directory if in development mode (inside the IDE).

Die Eigenschaft *Application.Dir* gibt das CWD (current working directory) zurück - so wie es zum Zeitpunkt des Programmstarts gesetzt ist.

- Der Gambas-Prozess erbt sein CWD von der IDE. Wenn Sie die IDE von Ihrem "Programme"-Menü aus starten, ist das CWD das Home-Verzeichnis.
- In allen anderen Fällen ist es das Verzeichnis, aus dem heraus Sie das Programm starteten. Machen Sie dieses Experiment: Öffnen Sie ein Terminal. Geben Sie 'cd /tmp' ein. Starten Sie

die Gambas-IDE mit 'gambas3'. Jetzt sind Sie in der IDE. In einem Projekt liefert 'Print Application.Dir' die Ausgabe /tmp!

Gambas hat kein Konzept von relativen Datei-System-Pfaden. Jeder relative Pfad, den Sie zum Beispiel an OPEN, File.Load(...) o.ä. übergeben, wird als relativ zu den Dateien im Projekt-Verzeichnis interpretiert. Um in das Datei-System des Rechners zu kommen, müssen Sie absolute Pfade verwenden. In diesem Fall kann der Einsatz der Eigenschaft Application.Dir nützlich werden.

Beispiel:

Es wird angenommen, dass Sie ein Programm myprogram.gambas erzeugt haben, dem eine Datei 'myfile' zur Verarbeitung mit einem Kommandozeilen-Argument übergeben muss. Damit zum Beispiel die Methoden File.Load(absolutepath2myfile) oder Picture.Load(absolutepath2myfile) die Datei laden können, muss der Datei-Pfad *absolut* angegeben werden, was nutzerunfreundlich ist. Als Nutzer würden Sie sicher gern statt

```
$ gbr3 ./myprogram.gambas absolutepath2myfile
```

```
$ gbr3 ./myprogram.gambas myfile
```

schreiben, wie Sie es von Programmen in anderen Programmiersprachen mit relativen Pfaden gewohnt sind. Genau für diesen Fall können Sie *Application.Dir* verwenden:

Quelltext-Ausschnitt für das Projekt ShowImageD

```
' Gambas class file

Public Sub Form_Open()

  Dim sImagePath As String
' Annahme: Der Bild-Pfad ist relativ zum CWD - wie in allen anderen Sprachen auch.
' Das Bild liegt im Projektverzeichnis.

  sImagePath = Application.Args[1]
  If sImagePath Not Begins "/" Then sImagePath = Application.Dir &/ sImagePath

  piboxImage.Picture = Picture.Load(sImagePath)

End
```

Programm-Start in zwei Varianten:

```
hans@mint-183 ~/Schreibtisch/ShowImageD $ gbr3 ./ShowImageD.gambas hgb.png
hans@mint-183 ~/Schreibtisch/ShowImageD $ gbx3 /home/hans/Schreibtisch/ShowImageD -- hgb.png
```

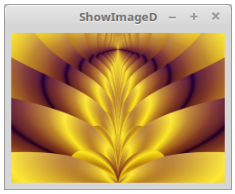

Abbildung 6.1.2.5.5: Start mit dem Bild hgb.png

Fazit: Sie können für Application.Dir auch die Eigenschaft Application.Env["PWD"] verwenden. Es nutzt die entsprechende Umgebungsvariable, die das CWD anzeigen soll, wenn diese richtig gesetzt ist. Greift man diesen Gedanken auf, dann könnte die direkte Verwendung der Umgebungsvariable $PWD in der Bash oder in einer anderen Shell ein vermittelnder Ansatz sein:

```
$ gbr3 ./myprogram.gambas $PWD/myfile
```

Quelltext-Ausschnitt ShowImageP

```
' Gambas class file

Public Sub Form_Open()

  Dim sImagePath As String
```

```
  sImagePath = Application.Args[1]
  piboxImage.Picture = Picture.Load(sImagePath)

End
```

Programm-Start:

```
hans@mint-183 ~/Schreibtisch/ShowImageP $ gbr3 ./ShowImageP.gambas $PWD/hgb.png
hans@mint-183 ~/Schreibtisch/ShowImageP $ gbx3 /home/hans/Schreibtisch/ShowImageP -- $PWD/hgb.png
```

Fazit: Das aktuelle Arbeitsverzeichnis [CWD] ist nur für Kommandozeilenprogramme interessant und in Gambas schreibt man wesentlich häufiger GUI-Programme. Letztere greifen zu Dialogen zur Dateiauswahl und diese Dialoge liefern absolute Pfade. Die Eigenschaft Application.Dir wird also recht wenig gebraucht.

6.1.2.5.3 Application.Path

Die Eigenschaft *Application.Path* liefert immer das Projekt-Verzeichnis, in dem der Quelltext des Projektes liegt.

Damit ergeben sich folgende Überlegungen:

(1) Die Eigenschaft Application.Path ist höchstens zum Debuggen oder für Phasen der Programm-Entwicklung in der IDE zu gebrauchen, um zum Beispiel in einem speziellen Entwicklermodus vom Programms erzeugte Dateien im Projektverzeichnis abzuspeichern.
(2) Die Eigenschaft Application.Path sollte nicht als Implementationsdetail eines Projektes verwendet werden, weil sonst das Programm nicht funktionieren wird, wenn sich ein Nutzer dazu entscheidet, aus dem Projekt eine ausführbare Archiv-Datei *.gambas zu erzeugen und diese zu öffnen. In diesem Fall sind alle Projektdateien in einem Archiv gespeichert und Application.Path ergibt keinen Sinn!

Beispiel Projekt 'Intro':

Beim Programmstart wird ein Bild als Intro angezeigt. Das Bild hgb.png liegt im Projektordner im (Unter-)Ordner 'images'. Die ausführbare Archiv-Datei erhält den Namen show.intro.gambas. Der Quelltext ist kurz:

```
' Gambas class file

Public Sub Form_Open()

  Dim sImagePath As String

  FMain.Icon = Picture.Load(".../symbols/form_icon.png")
  sImagePath = Application.Path &/ "images/hgb.png"

  piboxImage.Picture = Picture.Load(sImagePath)

End
```

(a) Start in der IDE mit F5 oder über den Button in der Symbolleiste mit dem Symbol ► → ok
(b) Das Programm wird direkt im Projekt-Verzeichnis gestartet → ok
(c) Start Konsole: hans@mint-183 ~/Schreibtisch $ gbr3 Intro/show.intro.gambas → ok
(d) Start Konsole: hans@mint-183 ~/Schreibtisch $ gbx3 Intro → ok

Abschließend wird nur die ausführbare Archiv-Datei *show.intro.gambas* in das Home-Verzeichnis kopiert und dort direkt gestartet:

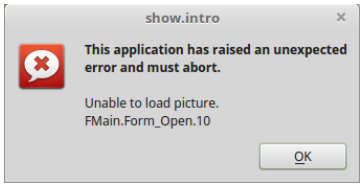

Abbildung 6.1.2.5.6: Start im Home-Verzeichnis

Der Fehler wird verständlich, weil die Eigenschaft Application.Path nicht mehr auf das Verzeichnis /home/hans/Schreibtisch/Intro sondern auf /home/hans zeigt und die Bild-Datei daher nicht geladen werden kann. Was nun - was tun? Hier ein Vorschlag:

Ersetzen Sie die Zeile

```
sImagePath = Application.Path &/ "images/hgb.png"
```

durch

```
sImagePath = ".../images/hgb.png"
```

und der Fehler ist beseitigt, weil die Auflösung des relativen Pfades jetzt stets korrekt erfolgt!