5.4.1 Gambas-Skripte (lokal)

In den Gambas-Skripten als Textdateien können alle Sprachelemente von Gambas verwendet werden. Der Aufbau eines Gambas-Skriptes ist bis auf den Header nicht formalisiert. Die Datei-Extension können Sie frei vergeben. Sie sollten dem Vorschlag folgen als Extension *gbs3* zu verwenden. Das folgende Skript zeigt ein erprobtes Format:

```
#!/usr/bin/env gbs3

PUBLIC SUB Main()
  PRINT
  PRINT " Aktuelles Datum: " & SetDateToGerman(Now) & "."
  PRINT " Es war genau " & Format$(Now, "hh:nn:ss") & " Uhr!"
  PRINT
END ' Main()

PRIVATE FUNCTION SetDateToGerman(dDatum AS Date) AS String

  DIM aMonatMatrix, aWochenTagMatrix AS NEW String[]
  DIM sWochenTag, sTag, sMonat, sJahr AS String

  aMonatMatrix.Clear
  aMonatMatrix = "Januar","Februar","März","April","Mai","Juni","Juli","August",
                 "September","Oktober","November","Dezember"]
  aWochenTagMatrix.Clear
  aWochenTagMatrix = Split("Sonntag Montag Dienstag Mittwoch Donnerstag Freitag Samstag", " ")

  sWochenTag = aWochenTagMatrix[WeekDay(dDatum)]
  sTag = Str(Day(dDatum))
  sMonat = aMonatMatrix[Month(dDatum) - 1]
  sJahr = Str(Year(dDatum))

  RETURN sWochenTag & " - " & sTag & ". " & sMonat & " " & sJahr ' Formatierter String

END
```

Kommentare:

- Die erste Zeile ist der sogenannte *Magic Header*: #!/usr/bin/env gbs3
- In der Prozedur Main() werden alle Anweisungen sowie die deklarierten Funktionen und Prozeduren zusammengefasst und ausgeführt.
- Für alle Ausgaben müssen Sie den Print-Befehl verwenden.
- Jeder Print-Befehl schließt mit einem Zeilenwechsel ab. Das können Sie verhindern, indem Sie den Print-Befehl mit einem Semikolon abschließen.
- Kommentare dürfen im Quelltext innerhalb von Funktionen oder Prozeduren stehen.
- Wird nach dem *letzten* 'End' – zum Beispiel am Ende Funktion *SetDateToGerman(...)* – ein Kommentar eingefügt wird, dann gibt es bis Revision 5491 eine Fehlermeldung!

Der o.a. Inhalt wird in der Datei *date_time2.gbs3* im Home-Verzeichnis gespeichert. Die Datei wird anschließend ausführbar gemacht:

```
hans@linux:~$ chmod u+x ./date_time2.gbs3

hans@linux:~$ ls -l # Kontrolle
-rwxr-xr-x   1 hans hans    1041 Dez 30 15:24 date_time2.gbs3
```

So erfolgt der Aufruf der Gambas-Skript-Datei *date_time2.gbs3* in einer Konsole. Es werden das aktuelle Datum und die (System-)Zeit angezeigt:

```
hans@linux:~$ gbs3 ./date_time2.gbs3

 Aktuelles Datum: Dienstag - 1. Januar 2013.
 Es war genau 15:26:25 Uhr!

hans@linux:~$
```

Die Funktion *SetDateToGerman(..)* hätte man auch durch diese Zeile ersetzen können:

```
PRINT " Aktuelles Datum: " & Format(Now, "dddd - dd. mmmm yyyy" ) & "."
```

Durch den Einsatz der Funktion *SetDateToGerman(..)* sollte nur gezeigt werden, wie Sie unterschiedliche Sprachelemente von Gambas im Skript-Quelltext einsetzen können. Ein Vorteil gegenüber herkömmlichen Skript-Sprachen, den Sie erst bei umfangreichen Skripten erkennen werden – eine Sprache für unterschiedliche Aufgabenbereiche!

5.4.1.1 Beispiel 1

Mit dem folgenden Skript werden die Umgebungsvariablen ausgelesen und angezeigt:

```
#!/usr/bin/env gbs3

PUBLIC SUB Main()
  Print
  Print "Anzeige Umgebungsvariablen"
  Print "--------------------------------"
  Print
  GetEnviroment()
END ' Main()

Public Sub GetEnviroment()
  DIM sElement as String

  FOR EACH sElement IN Application.Env
      Print sElement & " ---> " & Application.Env[sElement]
  NEXT ' sElement

End
```

Der Aufruf in der Konsole erfolgt mit

```
hans@linux:~$ gbs3 ./enviroment.gbs3
```

und liefert die Umgebungsvariablen als Name-Wert-Paare, die jeweils durch das Symbol '---->' getrennt sind:

```
LC_PAPER ---> de_DE.UTF-8
LC_ADDRESS ---> de_DE.UTF-8
SSH_AGENT_PID ---> 1856
LC_MONETARY ---> de_DE.UTF-8
GPG_AGENT_INFO ---> /tmp/keyring-XpXASO/gpg:0:1
TERM ---> xterm
SHELL ---> /bin/bash
..
XAUTHORITY ---> /home/hans/.Xauthority
LC_NAME ---> de_DE.UTF-8
_ ---> /usr/local/bin/gbs3
```

5.4.1.2 Beispiel 2

Mit diesem Gambas-Skript wird der aktuell benutzte Speicher ausgegeben, wobei einige Speichersegmente nicht beachtet werden:

```
#!/usr/bin/env gbs3

Public Sub Main()
  Print "--> Der aktuell benutzte Speicher beträgt " & CStr(GetUsedMemory()) & " Byte."
End ' Main()

Private Function GetUsedMemory() As Integer
  Dim sExecResult, sValue As String
  Dim aResult As String[]
  Dim cValue As New Collection

  Exec ["cat", "/proc/meminfo"] To sExecResult
```

```
  For Each sValue In Split(sExecResult, "\n", "", True)
      aResult = Split(sValue, " ", "", True)
      cValue[Left$(aResult[0], -1)] = aResult[1]
  Next ' sValue

  Return cValue!MemTotal - cValue!MemFree - cValue!Buffers - cValue!Cached +
         cValue!SwapTotal - cValue!SwapFree - cValue!SwapCached
End
```

Das Skript kann dann in einem Terminal aufgerufen werden und gibt den benutzten Speicher aus:

```
hans@linux:~$ gbs3 ./speicher.gbs3
--> Der aktuell benutzte Speicher beträgt 649116 Byte.
hans@linux:~$
```

Das Anzeigen der über 40 Speichersegmente ist auch direkt in einem Terminal möglich – jedoch ohne die Manipulationen in der Funktion *GetUsedMemory()*, die nur in einem Skript möglich sind und verwendet werden:

```
hans@linux:~$ cat /proc/meminfo
MemTotal:        8093648 kB
MemFree:         6519468 kB
Buffers:          145764 kB
..
DirectMap4k:       63104 kB
DirectMap2M:     8241152 kB
hans@linux:~$
```

5.4.1.3 Beispiel 3

Auch die Verbindung zu einer Datenbank und das Auslesen sowie Anzeigen ausgewählter Datenbank-Daten sind keine Hürde für ein Gambas-Skript. Hier der Quelltext für das Skript *tb_sqlite3.gbs3*:

```
#!/usr/bin/env gbs3

USE "gb.db"
USE "gb.db.sqlite3"

Public cDBVerbindung As New Connection

Public Sub Main()
  Dim iDatensatzNummer, iSpaltenNummer As Integer
  Dim sFehler, sFeldName, sSQL_Anweisung As String
  Dim rDBResult As Result

' Syntax:   goDBServer(  TYP,     HOST,               USER,PASS, DATENBANK,    PORT,TABELLE)
  sFehler = goDBServer("sqlite3", User.Home &/ "Liste", "", "", "liste.sqlite", "", "liste")
  If sFehler = "DBFehler" Then
     Print " Eine DB-Verbindung zum DB-Server konnte nicht hergestellt werden!"
     Return
  Endif ' sFehler = "DBFehler" ?

  sSQL_Anweisung = "SELECT * FROM liste"
  rDBResult = cDBVerbindung.Exec(sSQL_Anweisung)

  If rDBResult.Count = 0 Then
     Print "Die Anzahl der selektierten Datensätze ist Null!"
     Return
  Endif

  Print
  Print "Report vom " & Format(Now, "dd. mmmm yyyy") & " - " & Format$(Now, "hh:nn") & " Uhr"
Print "-----------------------------------------------"
  Print

  IF rDBResult.Available THEN

  For iDatensatzNummer = 0 To rDBResult.Count - 1
   rDBResult.MoveTo(iDatensatzNummer)
   For iSpaltenNummer = 0 To rDBResult.Fields.Count - 1
```

```
    Print Upper(rDBResult.Fields[iSpaltenNummer].Name) & " ---> " & rDBResult[iSpaltenNummer]
   Next ' Spalte
    Print
  Next ' Datensatz

  Endif ' MDataBase.rDBResult.Available

  Print "===== DB-Report-Ende ============================"
  Print

End ' Main()

Private Function goDBServer(DBType As String, DBHost As String, DBUserName As String,
                 DBUserPassword As String, DBName As String, DBPort As String,
                 DBTabellenName As String) As String

  cDBVerbindung.Type = Lower(DBType)      ' Der Typ muss klein geschrieben werden!
  cDBVerbindung.Host = DBHost
  cDBVerbindung.User = DBUserName         ' ---> Nur bei MySQL und PostgreSQL
  cDBVerbindung.Password = DBUserPassword ' ---> Nur bei MySQL und PostgreSQL
  cDBVerbindung.Name = DBName
  cDBVerbindung.Port = DBPort             ' ---> Nur bei MySQL und PostgreSQL

  Try cDBVerbindung.Open()
  If Error Then Return "DBFehler" ' ERROR

End
```

Kommentar:

- Die beiden Anweisungen *USE "gb.db"* und *USE "gb.db.sqlite3"* geben an, welche Komponenten zusätzlich verwendet werden sollen.

Die Anzeige der Datensätze erfolgt mit dem o.a. Gambas-Skript nur block- und zeilenweise:

```
hans@linux:~$ gbs3 ./tb_sqlite3.gbs3

Datenbank-Report vom 30. Dezember 2012 - 16:22 Uhr
-------------------------------------------------

ID ---> 1
NACHNAME ---> Adler
VORNAME ---> Anne
WOHNORT ---> Osterburg
PLZ ---> 39606
STRASSE ---> Werbener Strasse 20
TELEFONFESTNETZ ---> 03937864322
TELEFONMOBIL ---> 0171232323456
EMAILADRESSE ---> adler.anne@web.de
WEBADRESSE --->
GEBDATUM ---> 06/22/2005
HINWEISE ---> "Jugend forscht" - 2013
```

Auch Fehlermeldungen werden in der Konsole ausgegeben. Im ersten Fall wird der Fehler abgefangen und durch eine eigene Fehlermeldung angezeigt, während im 2. Fall nur die interne Fehler-Routine anspringt:

```
hans@linux:~$ gbs3 ./tb_sqlite3.gbs3
 Eine DB-Verbindung zum DB-Server konnte nicht hergestellt werden!
hans@linux:~$
```

```
hans@linux:~$ gbs3 ./tb_sqlite3.gbs3
MMain.Main.26: Query failed: SQL error or missing database
1: MMain.Main.26
hans@linux:~$
```