SWIG & The Forth Net: Hands-On

Gerald Wodni[*] M. Anton Ertl [†]

Euroforth 2011

Abstract

We have shown the basic functionality of SWIG and The Forth Net in the past. Now we want to provide two basic examples which explain how to use them and to show how easy it is to create C-interfaces or Forth-libraries and share them.

1 MySQL in Forth

As all mayor Forth-systems provide an interface to call libraries written in the C-programming-language, it should be quite easy to load and use a library like MySQL. But as all of them have their very own interface with their own set of words, this task can be quite cumbersome. The Forth extension for SWIG [1] attempts to provide an easier way, by creating a C-source-file which is platform independent and once it gets compiled, outputs the interface information for the target Forth-system.

First we need to create an interface file for SWIG[2]:

```
1 %module mysql
2 %insert("fsiinclude")
3 %{
4 #include <mysql/mysql.h>
5 %}
6
7 %include <mysql.h>
```

Line 1 tells SWIG the name of our module.

Lines 2-5 put Line 4 into the output-file in the section “fsiinclude”, which happens to be right after the default #include-directives.

Line 7 orders SWIG to parse mysql.h .

Having completed the interface file we invoke SWIG:

```
$ swig -forth -stackcomments -includeall -I/usr/include/mysql -o mysql-fsi.c mysql.i
```

Depending on our system we may add some more include directories (`-I...`). The resulting file (mysql-fsi.c) is platform independent and is exactly what we can upload to The Forth Net (see Section 2), or share in general.

On the target machine we compile the mysql-fsi.c file using the machine’s C-compiler, which includes the C-headers and thereby assigns the correct values for constants, the correct typedefs and cares about all other C-related problems.

We get a binary which we assume to be named mysql.fsx . Executing it will print the selected system’s C-interface definition. Redirecting it into mysql.fs gives us the desired Forth-source-file which we can simply include.

```
$ mysql.fsx -gforth > mysql.fs
```

[*]TU Wien; gerald.wodni@gee.at

[†]TU Wien; anton@mips.complang.tuwien.ac.at

We now have enabled Forth to access MySQL. See Appendix A for a simple example of the library in action. All of the above steps after creating the interface file can be done at once by using the Makefile of Appendix A. If we want to share our newly created library, Section 2 gives an example of how to do just that.

2 Library for The Forth Net

Let us consider we wrote a library for 3D-graphics and want to share it. However, in order to make this library truly system independent and easily accessible, we have to introduce some new words explained below. All words start with "f" which relates to The Forth Net[3].

frequire *(... "file" – ...)* Some Forth-systems use the working directory as the base directory for including files by relatives paths. Others use the file which is currently parsed as the base directory. To circumvent ambiguities for our library we could use absolute paths. Unfortunately this decision would force an absolute path on all users, which is not an option. *frequire* prepends the libraries base path on the current system to *"file"* and loads it.

finclude *(... "name" – ...)* Loads the library by including its main file.

fget *(... "name" – ...)* Downloads the library from The Forth Net[4] to the libraries-root-path.

Using these words we can now extend respectively write our library. Our directory structure is as follows:

```
3d
├──3d.fs
├──models
│  ├──model.fs
│  ├──sphere.fs
│  └──cube.fs
├──data
│  ├──tree.tga
│  └──water.tga
├──doc
│  └──README.txt
└──VERSIONINFO
```

"3d.fs" is the libraries main file which is included by the user who uses "`finclude` 3d". The library allows the user to draw cubes and spheres, so "3d.fs" will provide them by using "`frequire` models/cube.fs" and "`frequire` models/sphere.fs". Cubes and spheres use words which are common amongst all models, and are defined in models/model.fs, so both files use "`frequire` models/models.fs" (we always address files from the libraries base-path).

All other files are optional: the directory "data" provides the library with some textures, "doc" holds the documentation, which explain the user how to use the 3d-library. The content of the file "VERSIONINFO" is displayed on The Forth Net when browsing different versions, and hold some information about recent improvements or bug fixes.

The final step of sharing our library is to upload it to The Forth Net. This is accomplished by creating a new project, clicking manage, zipping the contents of the directory 3d (not including the directory itself), and submitting them as a new version.
From this moment on others can fetch the library by using "`fget` 3d"

A MySQL Demo[5]

```
\ (c) 2011 by Gerald Wodni
\ very small example for interfaceing with the c-api of mysql

\ load binary shared library
s"␣mysqlclient" add-lib

\ include functions and constants
finclude fsi-mysql-client

\ display 0-terminated string
: .cstr ( addr -- )
        begin
                dup c@
        while
                dup c@ emit
                char+
        repeat drop ;

\ -- real program --
\ create connection element
0 mysql_init constant connection

\ connect
connection s\"␣localhost\0" drop s\"␣forth\0" drop s\"␣h4x0r\0"
drop s\"␣forth\0" drop 0 0 0 mysql_real_connect [if]
        ."␣connection␣established" cr
[else]
        ."␣connection␣error" cr bye
[then]

\ launch query
connection s\"␣SELECT␣*␣FROM␣‘systems‘␣\0" drop mysql_query [if]
        ."␣Query-Error:␣" connection mysql_error .cstr cr bye
[then]

\ print result
: tab 9 emit ;
: show-result ( -- )
        \ get result
        connection mysql_use_result
        begin
                dup mysql_fetch_row ?dup
        while
                connection mysql_field_count 0 u+do
                        dup @ tab .cstr
                        cell+
                loop drop
                cr
        repeat
        mysql_free_result ;

."␣Result:" cr
show-result

connection mysql_close
."␣connection␣closed" cr

bye
```

A FSI-Makefile[5]

```
SWIG        = swig
OUTPUT      = mysql
INTERFACE   = mysql.i
OPTIONS     = -forth -no-sectioncomments -stackcomments -includeall\
              -I/usr/include/mysql -I/usr/include\
              -I/usr/lib/gcc/i486-linux-gnu/4.1/include/

$(OUTPUT).fs: $(OUTPUT).fsx
        ./$(OUTPUT).fsx -gforth > $(OUTPUT).fs

$(OUTPUT).fsx: $(OUTPUT).fsi
        $(CC) -o $(OUTPUT).fsx $(OUTPUT)-fsi.c

$(OUTPUT).fsi:
        $(SWIG) $(OPTIONS) -o $(OUTPUT)-fsi.c $(INTERFACE)

.PHONY: clean

clean:
        rm -f $(OUTPUT)-fsi.c
        rm -f $(OUTPUT).fs
        rm -f $(OUTPUT).fsx
```

References

[1] Gerald Wodni and M. Anton Ertl. SWIG-Gforth-Extension. In *Euroforth*, 2009.

[2] David M. Beazley et al. Simplified Wrapper and Interface Generator (SWIG). URL `http://www.swig.org`.

[3] Gerald Wodni and M. Anton Ertl. The Forth Net, 2010.

[4] Gerald Wodni. The Forth Net. URL `http://theforth.net`.

[5] Gerald Wodni and M. Anton Ertl. SWIG Erweiterung für Forth Neuigkeiten, 2011.