WIKIPEDIA

Foreign function interface

A **foreign function interface** (**FFI**) is a mechanism by which a program written in one programming language can call routines or make use of services written in another.

Contents

Naming

The term comes from the specification for Common Lisp, which explicitly refers to the language features for inter-language calls as such;[1] the term is also used officially by the Haskell[2] and Python programming languages.[3] Other languages use other terminology: the Ada programming language talks about "language bindings", while Java refers to its FFI as the JNI (Java Native Interface) or JNA (Java Native Access). Foreign function interface has become generic terminology for mechanisms which provide such services.

Operation

The primary function of a foreign function interface is to mate the semantics and calling conventions of one programming language (the *host* language, or the language which defines the FFI), with the semantics and conventions of another (the *guest* language). This process must also take into consideration the runtime environments and/or application binary interfaces of both. This can be done in several ways:

- Requiring that guest-language functions which are to be host-language callable be specified or implemented in a particular way; often using a compatibility library of some sort.
- Use of a tool to automatically "wrap" guest-language functions with appropriate glue code, which performs any necessary translation.
- Use of wrapper libraries

- Restricting the set of host language capabilities which can be used cross-language. For example, C++ functions called from C may not (in general) include reference parameters or throw exceptions.

FFIs may be complicated by the following considerations:

- If one language supports garbage collection (GC) and the other does not; care must be taken that the non-GC language code does nothing to cause GC in the other to fail. In JNI, for example, C code which "holds on to" object references that it receives from Java must "register" this fact with the Java runtime environment (JRE); otherwise, Java may delete objects before C has finished with them. (The C code must also explicitly release its link to any such object, as soon as there is no further need, by C, of that object.)
- Complicated or non-trivial objects or datatypes may be difficult to map from one environment to another.
- It may not be possible for both languages to maintain references to the same instance of a mutable object, due to the mapping issue above.
- One or both languages may be running on a virtual machine (VM); moreover, if both are, these will probably be different VMs.
- Cross-language inheritance and other differences, such as between type systems or between object-composition models, may be especially difficult.

By language

Examples of FFIs include:

- Ada language bindings, allowing not only to call foreign functions but also to export its functions and methods to be called from non-Ada code.[4]
- C++ has a trivial FFI with C, as the languages share a significant common subset. The primary effect of the `extern "C"` declaration in C++ is to disable C++ name mangling.
- CNI, alternative to JNI used in the GNU compiler environment.
- D does it the same way as C++ does, with extern "C" through extern (C++)
- Dynamic languages, such as Python, Perl, Tcl, and Ruby, all provide easy access to native code written in C/C++ (or any other language obeying C/C++ calling conventions).
- Factor has FFIs for C (http://fun-factor.blogspot.com/2007/10/getting-started-with-factor-easy-ffi.html), Fortran (http://article.gmane.org/gmane.comp.lang.factor.general/2790), Objective-C (http://docs.factorcode.org/content/vocab-cocoa.html), and Windows COM (http://docs.factorcode.org/content/vocab-windows.com.html); all of these enable importing and calling arbitrary shared libraries dynamically.
- The FFIs of Common Lisp and Haskell
- Fortran 2003 has a module ISO_C_BINDING which provides interoperable data types (both intrinsic types and POD structs), interoperable pointers, interoperable global data stores, and mechanisms for calling C from Fortran and for calling Fortran from C.[5]
- Perl 6 can call Ruby, Python, Perl 5, Brainfuck, Lua, C, C++ and Scheme Guile/Gambit [6] [7]
- Go can call C code directly via the `"C"` pseudo-package.[8]
- GWT, in which Java is compiled to JavaScript, has a FFI called JSNI which allows Java source to call arbitrary JavaScript functions, and for JavaScript to call back into Java.
- JNI, which provides an interface between Java and C/C++, the preferred systems languages on most systems where Java is deployed. JNA provides an interface with native libraries without having to write glue code. Another example is JNR (https://github.com /jnr/jnr-ffi)
- Julia has `ccall` keyword to call C (and other languages, e.g. Fortran);[9] while packages, providing similar no-boilerplate support, are available for some languages e.g. for Python[10] (to e.g. provide OO support and GC support), Java (and supports other JDK-

languages, such as Scala) and R. Interactive use with C++ is also possible with Cxx.jl package.

- Python provides the ctypes (https://docs.python.org/3/library/ctypes.html) and cffi (https://cffi.readthedocs.io/en/latest/) modules. For example, the ctypes module can load C functions from shared libraries/DLLs on-the-fly and translate simple data types automatically between Python and C semantics as follows:

```
import ctypes
libc = ctypes.CDLL( '/lib/libc.so.6' )   # under Linux/Unix
t = libc.time(None)                      # equivalent C code: t = time(NULL)
print t
```

- P/Invoke, which provides an interface between the Microsoft Common Language Runtime and native code.
- Racket has a native FFI based heavily on macros that enables importing arbitrary shared libraries dynamically.[11][12]
- Rust also defines a foreign function interface.[13]
- Visual Basic has a declarative syntax that allows it to call non-Unicode C functions.
- One of the bases of the Component Object Model is a common interface format, which natively uses the same types as Visual Basic for strings and arrays.
- LuaJIT, a just-in-time implementation of Lua, has an FFI that allows "calling external C functions and using C data structures from pure Lua code".[14]
- PhoneGap (was called by the name Apache Callback, but now Apache Cordova) is a platform for building native mobile applications using HTML, CSS and JavaScript. Additionally has FFIs via JavaScript callback functions for access to methods and properties of mobile phone's native features including Accelerometer, Camera (also PhotoLibrary and SavedPhotoAlbum), Compass, Storage (SQL database and localStorage), Notification, Media and Capture (playing and recording or audio and video), File, Contacts (address book), Events, Device and Connection information.[1] (http://phonegap.com/),[2] (http://cordova.apache.org/).
- Wolfram Language provides a technology called WSTP (Wolfram Symbolic Transfer Protocol) which enables bidirectional calling of code between other languages with bindings for C++, Java, .NET and other languages.

In addition, many FFIs can be generated automatically: for example, SWIG. However, in the case of an extension language a semantic inversion of the relationship of guest and host can occur, when a smaller body of extension language is the guest invoking services in the larger body of host language, such as writing a small plugin [15] for GIMP.[16]

Some FFIs are restricted to free standing functions, while others also allow calls of functions embedded in an object or class (often called method calls); some even permit migration of complex datatypes and/or objects across the language boundary.

In most cases, a FFI is defined by a "higher-level" language, so that it may employ services defined and implemented in a lower level language, typically a systems language like C or C++. This is typically done to either access OS services in the language in which the OS' API is defined, or for performance considerations.

Many FFIs also provide the means for the called language to invoke services in the host language as well.

The term foreign function interface is generally not used to describe multi-lingual runtimes such as the Microsoft Common Language Runtime, where a common "substrate" is provided which enables any CLR-compliant language to use services defined in any other. (However, in this case the CLR does include an FFI, P/Invoke, to call outside the runtime.) In addition, many distributed computing architectures such as the Java remote method invocation (RMI), RPC, CORBA, SOAP and D-Bus permit different services to be written in different languages; such architectures are generally not considered FFIs.

Special cases

There are some special cases, in which the languages compile into the same bytecode VM, like Clojure and Java, as well as Elixir and Erlang. Since there is no interface, it is not a FFI, strictly speaking, while it offers the same functionality to the user.

See also

- Language interoperability
- Transcompiler
- Calling convention
- Name mangling
- Application programming interface
- Application binary interface
- Comparison of application virtual machines
- SWIG
- Remote procedure call
- libffi

References

1. "CFFI User Manual" (https://common-lisp.net/project/cffi/manual/cffi-manual.html). common-lisp.org. Retrieved 2015-06-18.
2. "FFI Introduction" (https://wiki.haskell.org/FFI_Introduction). *HaskellWiki*. Retrieved 19 June 2015. "Haskell's FFI is used to call functions from other languages (basically C at this point), and for C to call Haskell functions."
3. "CFFI documentation" (https://cffi.readthedocs.org/). Retrieved 19 June 2015. "C Foreign Function Interface for Python. The goal is to provide a convenient and reliable way to call compiled C code from Python using interface declarations written in C."
4. "Interface to Other Languages" (http://www.adaic.org/standards/05aarm/html/AA-B.html). Adaic.org. Retrieved 2013-09-29.
5. https://stackoverflow.com/tags/fortran-iso-c-binding/info
6. "Inline implementations" (https://modules.perl6.org/s/Inline%3A%3A). Retrieved 2017-08-15.
7. "Native Call" (https://docs.perl6.org/language/nativecall). Retrieved 2017-08-15.
8. "cgo — The Go Programming Language" (https://golang.org/cmd/cgo/). Retrieved 2015-08-23.
9. "Calling C and Fortran Code · The Julia Language" (https://docs.julialang.org/en/release-0.6/manual/calling-c-and-fortran-code/). *docs.julialang.org*. Retrieved 2018-02-11.
10. *PyCall.jl: Package to call Python functions from the Julia language* (https://github.com/JuliaPy/PyCall.jl), JuliaPy, 2018-02-08, retrieved 2018-02-11

11. Eli Barzilay. "The Racket Foreign Interface" (http://docs.racket-lang.org/foreign/). Docs.racket-lang.org. Retrieved 2013-09-29.
12. "TR600.pdf" (http://repository.readscheme.org/ftp/papers/sw2004/barzilay.pdf) (PDF). Retrieved 2013-09-29.
13. https://doc.rust-lang.org/1.5.0/book/ffi.html (https://doc.rust-lang.org/1.5.0/book/ffi.html). Retrieved 2017-08-07. Missing or empty `|title=` (help)
14. Mike Pall. "FFI Library" (http://luajit.org/ext_ffi.html). Luajit.org. Retrieved 2013-09-29.
15. "4. A sample script" (http://www.gimp.org/docs/scheme_plugin/scheme-sample.html). Gimp.org. 2001-02-04. Retrieved 2013-09-29.
16. "Script-Fu and plug-ins for The GIMP" (http://www.gimp.org/docs/scheme_plugin/). Gimp.org. Retrieved 2013-09-29.

External links

- c2.com: Foreign function interface (http://www.c2.com/cgi/wiki?ForeignFunctionInterface)
- Haskell 98 Foreign Function Interface (http://www.cse.unsw.edu.au/~chak/haskell/ffi/)
- Allegro Common Lisp FFI (http://www.franz.com/support/documentation/6.2/doc/foreign-functions.htm)
- A Foreign Function Interface generator for occam-pi (http://www.cs.kent.ac.uk/pubs/2005/2254/)
- UFFI: Lisp Universal Foreign Function Interface (http://uffi.b9.com/)
- CFFI: Common Foreign Function Interface, for Common Lisp (http://common-lisp.net/project/cffi/)
- Java Native Interface: Programmer's Guide and Specification (https://web.archive.org/web/20120728074805/http://java.sun.com/docs/books/jni/)
- The JNI Specification (https://docs.oracle.com/javase/8/docs/technotes/guides/jni/spec/jniTOC.html)
- JSNI (JavaScript Native Interface) (https://code.google.com/webtoolkit/documentation/com.google.gwt.doc.DeveloperGuide.JavaScriptNativeInterface.html)
- dyncall library using assembly call kernels for a variety of processors,OS and calling conventions (http://dyncall.org/)
- FFCALL (https://www.gnu.org/software/libffcall/)
- C/Invoke (http://www.nongnu.org/cinvoke/)
- libffi (http://sourceware.org/libffi/)