Preface

Introduction

SWIG is a tool for solving problems.

More specifically, SWIG is a simple tool for building interactive C, C++, or Objective-C programs with common scripting languages such as Tcl, Perl, and Python. Of course, more importantly, SWIG is a tool for making C programming more enjoyable and promoting laziness (an essential feature). SWIG is not part of an overgrown software engineering project, an attempt to build some sort of monolithic programming environment, or an attempt to force everyone to rewrite all of their code (ie. code reuse). In fact, none of these things have ever been a priority.

SWIG was originally developed in the Theoretical Physics Division at Los Alamos National Laboratory for building interfaces to large materials science research simulations being run on the Connection Machine 5 supercomputer. In this environment, we were faced with the problems of working with huge amounts of data, complicated machines, and constantly changing code. As scientists, we needed a mechanism for building interactive programs that was extremely easy to use, could keep pace with code that was constantly changing, and didn't get in the way of the real problems that were being solved. Mainly, we just wanted to "cut the crap" and work on the real problems at hand.

While SWIG was originally developed for scientific applications, it has become a general purpose tool that is being used in an increasing variety of other computing applications--in fact almost anything where C programming is involved. Some of the application areas that I am aware of include scientific applications, visualization, databases, semiconductor CAD, remote sensing and distributed objects. Development has been pragmatic in nature--features have been added to address interesting problems as they arise. Most of the really cool stuff has been contributed or suggested by SWIG's users. There has never really been a "grand" design to SWIG other than the goal of creating a practical programming tool that could be used in other applications.

SWIG resources

The official location of SWIG related material is

```
http://www.cs.utah.edu/~beazley/SWIG
```

This site contains the latest version of the software, users guide, and information regarding bugs, installation problems, and implementation tricks. The latest version of the software and related files are also available via anonymous ftp at

```
ftp://ftp.cs.utah.edu/pub/beazley/SWIG
```

You can also subscribe to the SWIG mailing list by sending a message with the text "subscribe swig" to

```
majordomo@cs.utah.edu
```

The mailing list often discusses some of the more technical aspects of SWIG along with information about beta releases and future work.

About this manual

This manual has been written in parallel with the development of SWIG because I hate black boxes and I don't like using software that is poorly documented. This manual attempts to describe all aspects of SWIG and how it can be used to solve interesting problems. Don't let the size scare you, SWIG can be quite easy to use. However, covering automatic code generation for four different scripting languages takes abit of explanation. SWIG can do quite a few interesting things that might not be so obvious so I hope that the manual can shed some light on many of these issues. The manual also serves as a general reference describing many of SWIG's implementation issues (I use the manual quite regularly myself).

Prerequisites

This manual assumes that you are interested in writing interactive C programs and that you have at least heard of scripting languages such as Tcl, Python, and Perl. A detailed knowledge of these scripting languages is not required although some familiarity certainly won't hurt. No prior experience with building C extensions to these languages is required---after all, this is what SWIG allows you to do automatically. However, I do assume that you are reasonably familiar with the use of compilers, linkers, and makefiles since making scripting language extensions is somewhat more complicated than writing a normal C program (although not significantly more complex).

Organization of this manual

The first few chapters of this manual describe SWIG in general and provide an overview of its capabilities. The remaining chapters are devoted to specific SWIG language modules and are self contained. Thus, if you are using SWIG to build Python interfaces, you can skip right to that chapter and find just about everything you need to know. So, in a sense, you are really getting 3 or 4 manuals in one.

How to avoid reading the manual

If you hate reading manuals, glance at the "Introduction" which contains a few simple examples and the overall philosophy. These examples contain about 95% of everything you need to know to use SWIG. After that, simply use the language-specific chapters for reference. The SWIG distribution also comes with a large directory of examples that illustrate how to do most kinds of things.

Credits

This work would certainly not be possible without the support of many people. I would like to acknowledge Peter Lomdahl, Brad Holian, Shujia Zhou, Niels Jensen, and Tim Germann at Los Alamos National Laboratory for allowing me to pursue this project and for being the first users. Patrick Tullmann at the University of Utah suggested the idea of automatic documentation generation. John Schmidt and Kurtis Bleeker at the University of Utah tested out the early versions.

I'd also like to acknowledge Chris Johnson and the Scientific Computing and Imaging Group at the University of Utah for their continued support. John Buckman, Larry Virden, and Tom Schwaller provided valuable input on the first releases and improving the portability of SWIG. David Fletcher and Gary Holt have provided a great deal of input on improving SWIG's Perl5 implementation. I'd also like to thank Kevin Butler for his valuable input and contribution of a Windows NT port. Finally, I'd like to acknowledge all of the users who have braved the first few releases and have been instrumental in suggesting way to make SWIG more fun to use than I ever imagined possible.

What's new?

The following significant features are new in version 1.1

- Support for typemaps (a mechanism for customizing SWIG).
- Multiple inheritance.
- Default/optional arguments.
- Perl5 shadow classes.
- Tcl8.0 module (uses the native Tcl8 object interface).
- An entirely new documentation system.
- Limited support for nested structures.
- New object oriented Tcl interface.
- User defined exception handling.
- New directives for better library support (%inline, %extern %import)
- Objective-C support.
- Support for Windows-NT and Macintosh.
- Lots of minor bug fixes to almost everything.

This release should be backwards compatible with interface files generated for SWIG 1.0. However, many things have changed in the SWIG C++ API so special purpose SWIG C++ extensions written for 1.0 will need to be modified.

Bug reports

While every attempt has been made to make SWIG bug-free, occasionally bugs will arrise. To report a bug, send mail to the SWIG mailing list at `swig@cs.utah.edu`. In your message, be as specific as possible, including (if applicable), error messages, tracebacks (if a core dump occurred), corresponding portions of the SWIG interface file used, and any important pieces of the SWIG generated wrapper code. I attempt to respond to all bug reports, but I can only fix bugs if I know about them.

SWIG is free

SWIG is a completely free package that you can use in any manner that you wish, including modification, redistribution, and use in commercial products. The only restriction on its use is that redistributions of the SWIG compiler should reproduce the SWIG copyright notice in the supporting documentation. The code generated by SWIG can be redistributed in any manner. On a more personal note, if you've used SWIG to make something cool, I'd like to find out more about it so that I can make SWIG better (and to satisfy my curiosity).