Interfacing C/C++ and Python with SWIG

David M. Beazley

Department of Computer Science
University of Chicago
Chicago, Illinois 60615

beazley@cs.uchicago.edu

Prerequisites

C/C++ programming

- You've written a C program.
- You've written a Makefile.
- You know how to use the compiler and linker.

Python programming

- You've heard of Python.
- You've hopefully written a few Python programs.

Optional, but useful

- Some knowledge of the Python C API.
- C++ programming experience.

Intended Audience

- C/C++ application developers interested in making better programs
- Developers who are adding Python to "legacy" C/C++ code.
- Systems integration (Python as a glue language).

Notes

C/C++ Programming

The good

- High performance.
- Low-level systems programming.
- Available everywhere and reasonably well standardized

The bad

- The compile/debug/nap development cycle.
- Difficulty of extending and modifying.
- Non-interactive.

The ugly

- Writing user-interfaces.
- Writing graphical user-interfaces (worse).
- High level programming.
- Systems integration (gluing components together).

Notes

What Python Brings to C/C++

An interpreted high-level programming environment

- Flexibility.
- Interactivity.
- Scripting.
- Debugging.
- Testing
- Rapid prototyping.

Component gluing

- A common interface can be provided to different C/C++ libraries.
- C/C++ libraries become Python modules.
- Dynamic loading (use only what you need when you need it).

The best of both worlds

- Performance of C
- The power of Python.

Notes

Points to Ponder

"Surely the most powerful stroke for software productivity, reliability, and simplicity has been the progressive use of high-level languages for programming. Most observers credit that development with at least a factor of 5 in productivity, and with concomitant gains in reliability, simplicity, and comprehensibility."

--- **Frederick Brooks**

"The best performance improvement is the transition from the nonworking state to the working state."

--- John Ousterhout

"Less than 10% of the code has to do with the ostensible purpose of the system; the rest deals with input-output, data validation, data structure maintenance, and other housekeeping"

--- Mary Shaw

"Don't keep doing what doesn't work"

--- Anonymous

Notes