SLIRP-ing FORTRAN into S-Lang

Michael S. Noble
MIT Center For Space Research
mnoble@space.mit.edu
http://space.mit.edu/%7Emnoble/slirp
August 14, 2003
Introduction

- SLIRP is the (SL)ang (I)nte(R)face (P)ackage
- Code generator: greatly simplifies task of using compiled libraries w/in S-Lang / ISIS / CIAO
- In spirit of S.W.I.G. (which lacks S-Lang mode)
- WHY: machines better at boring/mechanical stuff
- WHY: easily hook potentially any C lib to S-Lang
- Created in spare time to ease development of SLgtk
- Now maintained at MIT, released as open source
Auto-generate \( \geq 95\% \) of importable module code

Works by inspecting prototypes, typedefs, ...

... \#defines, and enums from C header (.h) files

Since demonstrated useful for CXC DataModel and forthcoming OpenGL module (3D graphics)

Stress-tested on PVM, XPA, and FITSIO libraries

Multiple requests motivated FORTRAN support
Binding FORTRAN to S-Lang

C header files

C source code

FORTRAN source files

SLIRP

Module init fragment
Function wrappers ("glue")
Intrinsic constants
Intrinsic function table entries

f2c

SLIRP-ing FORTRAN into S-Lang – p.4/10
- specify FORTRAN source instead of C headers
- SLIRP transparently invokes the venerable f2c
  to generate subroutine / function prototypes
- but NOT messy f2c-translated C source code
- SLIRP consumes f2c-generated .h file, as if from C
- to generate more readable/debuggable C code
And Now The Boring Example

```
slirp -d gen/ -init mffjet.f

19 funcs wrapped ...
904 lines of code ...

gcc ... -c gen/mffjet_init.c
f77    -c mffjet.f

gcc ... -o mffjet-module.so \ 
mffjet_init.o mffjet.o ...

NB: zero lines of code written by hand
Great for one-off use of crusty old codes
```
Boring Example, cont.

```plaintext
isis> import ("mffjet");
isis>
isis> ne = 300; ifl = 1;
isis> ebins = <300 4-byte reals>
isis> param = <13 4-byte reals>
isis> photons = <300 4-byte reals>
isis> errs = <300 4-byte reals>
isis>
isis> mfffunc(ebins,ne,param,
              ifl,photons,errs);
```

The `spec` and `dummy` arrays above are output parameters of the subroutine; all others are input parameters.
Case Study: S. Markoff’s Jet codes

- longstanding problem with XSPEC usage
- code runs fine when invoked once at UNIX CL
- spurious results when invoked repeatedly w/in fit
- QUESTION: is problem in XSPEC interface ...
- ... or in the underlying FORTRAN?
- ANSWER: with SLIRP-generated module ...
- ... spurious result duplicated in ISIS w/in 2 hours
Final Thoughts

Conveniences:
- Scalars transparently morphed to FORTRAN references
- S-Lang name same as in FORTRAN source (trailing underscore ellided from func name)

- Renders C/FORTRAN calls indistinguishable (yay!)
- Power of S-Lang/ISIS, plus SLIRP extensibility ...
- ... very attractive alternative to XSPEC and Tcl
Final Thoughts (really)

- Current/Potential Uses of FORTRAN mode:
  - R. Smith et al (CFA)
  - S. Markoff et al (MIT)
  - E. Mandel, A. Vikhlinin et al (CFA)
  - Tell YOUR friends ...

- Complex values not yet supported

- Nor are multidimensional arrays yet remapped (from C row-major to FORTRAN column-major)

- Doesn’t yet generate Makefile for you :(