S-Lang as the CIAO Scripting Language

CIAO/S-Lang Workshop
January 27, 2003

Michael S. Noble
mnoble@space.mit.edu

Welcome! Thank you for coming.

- We'll discuss the adoption of a scripting language
- Detailing why S-Lang was chosen over alternatives
- As objective/language-agnostically as possible
- Point out how/when/where it can be used today
- Highlight several derived benefits
- References to related discussions:
  - http://cxc.harvard.edu/ciao/why/slang.html
  - http://hea-www.harvard.edu/uinfra/varmm/ScriptLanguageEval
  - http://hea-www.harvard.edu/uinfra/varmm/caseStudy.ps

Why A Scripting Language?

- CIAO effectively lacked extensibility

- In principle one *could* download source and fix/improve

- Unrealistic in practice, given CIAO size/complexity

- Model for user-motivated improvements thus defaults to:
  - Make request (to ascds_help, dsdevel)
  - Wait N months (with $N > 12$ in some cases)
  - For it to be coded/released in new CIAO version
  - If at all (everyone's pretty busy)
  - Meanwhile your proprietary period expires ...

- Over-Centralized, in sharp discord with lessons of:
  evolution/biology, politics, open s/w, etc

- Frequently resort to IDL, MatLab, etc to get job done

- Other general benefits of incorporating scripting language
  - rapid prototyping (no compilation)
  - dynamic/loose typing
  - no memory mgmt (malloc/free) worries
  - expressive power: 1 script line == N compiled lines

CIAO-Specific Requirements/Goals

- wanted powerful, "IDL-like" mathematical capability
- with fast support for ND arrays
- transparency between array/scalar arithmetic
- to serve as basis for rapid scientific development
- concise syntax, natural for casual scientific use
- *embeddability* into *existing* apps (minimize rewrite)
- coexist with existing **sherpa/chips** command language (do not *require* end-user to learn S-Lang)
- as small a footprint as possible (CIAO already large)

CONCLUSION

CIAO clearly needed:

- *some* **general-purpose scripting capability**
- **targeted at small- to medium-size needs**
- *with* **a scientific orientation**
- *without* **huge buy-in for typical CIAO user**
- **which fosters uses** *not envisioned by creators*

To fill this gap, 2+ years ago we surveyed 3 of the most prominent scripting languages in use, and several lesser-known but capable languages.

Highlight of Language Survey

Tcl [(T)ool (C)ommand (L)anguage]

- simplest of Big 3, but still powerful (esp. with Tk)
- String-based; lossy numeric conversions internally
- Poor math support:
  - odd syntax: 'set', {, } , 'expr'
  - but not the equals sign!
  - explicit looping over arrays
- implies poor scalability to large datasets
- as 'FV' on Chandra-sized files will demonstrate
- Tcl/Tk is used in CXC/CIAO
- For GUIs, not number crunching

Python

- "big" platform : OO scales well to larger projects
- yet no procedural support for application-scoped globals
- Per Python FAQ 3.30:

```
... In practice, few applications embed
    Python -- it's much more common to
    have Python extensions ...
```

- too much for our stated needs? (***tail wagging dog***)
- N-D arithmetic unsupported in base language
- again, explicit looping required for array operations
- mindshift: no begin/end braces, scoping by indentation?
- forces clean, readable code, at least in the small, BUT
- do you want language to dicate where TAB/SPACE go?
- problematic w/out editor customizations?
- cut/paste and Win/Mac/Unix interchange problems?
- buy-in and potential snags conflict w/ casual scientific use?

Perl [(P)ractical (E)xtraction and (R)eport (L)anguage]

- like Python, versatile and powerful
- origins in text manipulation, probably best available
- many "other" features grafted on over years
- again, a "big" platform
- "noisy" syntax, reminiscient of regular expressions
- clashes w/ CIAO ; can stymy uninitiated in casual use
- N-D arithmetic unsupported in base language
- again, explicit looping required for array operations

Others (see references for more details)

- Root (CINT/Integral)
- GLish
- CXC-developed language for MTL

Tcl Matrix Multiply

```
proc mmult {m1 m2} {
 set cols [lindex $m2 0]
 foreach row1 $m1 {
  set row [list]
  set i 0
  foreach - $cols {
   set elem 0
   foreach elem1 $row1 row2 $m2 {
    set elem [expr {$elem + $elem1 * [lindex $row2 $i]}]
   }
   lappend row $elem
   incr i
  }
  lappend result $row
 }
 return $result
}
```

Is this the idiom of matrix multiplication?

Perl Matrix Multiply

```
sub mmult {
    my ($rows, $cols, $m1, $m2) = @_;
    my @m3 = ();
    --$rows; --$cols;
    for my $i (0 .. $rows) {
       my @row = ();
       my $m1i = $m1->[$i];
       for my $j (0 .. $cols) {
          my $val = 0;
          for my $k (0 .. $cols) {
              $val += $m1i->[$k] * $m2->[$k]->[$j];
           }
          push(@row, $val);
       }
       push(@m3, \@row);
    }
    return(\@m3);
}
```

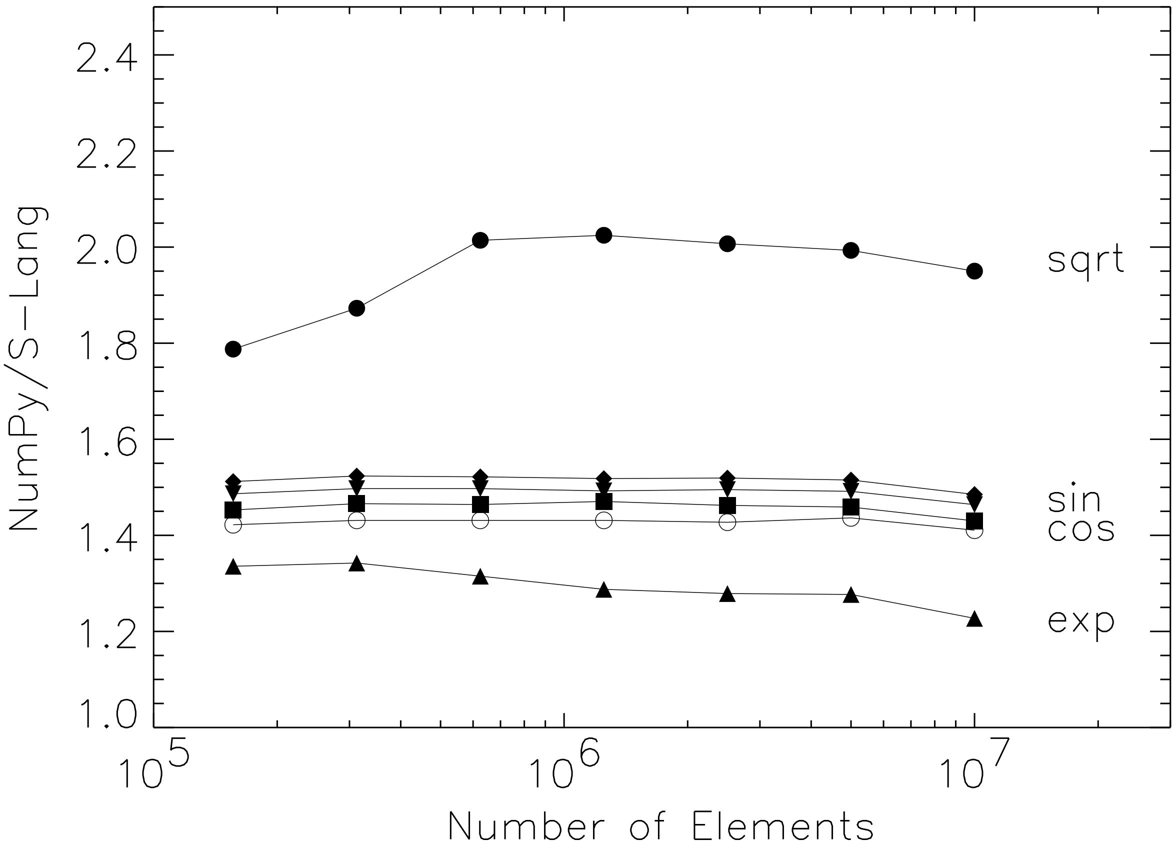
Code samples taken from

The Great Computer Language Shootout
http://www.bagley.org/%7Edoug/shootout

Python Matrix Multiply

```
def mmult(rows, cols, m1, m2):
    m3 = [ None ] * rows
    for i in range( rows ):
        m3[i] = [0] * cols
        for j in range( cols ):
            val = 0
            for k in range( cols ):
                val += m1[i][k] * m2[k][j]
            m3[i][j] = val
    return m3
```

What none of these snippets show is that in each language an additional 12-15 lines of code are needed simply to *define* the 30x30 matrices.



Performance comparison of Python ***numerical extension*** with ***native*** S-Lang, on common mathematical operations.

(Courtesy: Ming Feng Gu, MIT Center for Space Research)

Why S-Lang?

In 3 words : Numbers, Numbers, Numbers!

- 3 lines to define matrices (could be done in 2):

```
% define 1D array with 30*30 elements,
% with values initialized to 1..900

sherpa>   m1 = [1:30*30]
sherpa>   reshape(m1,[30,30])  % morph into 2D
sherpa>   m2 = @m1             % clone
```

- and just one line of code to multiply them

```
sherpa>   m3 = m1 # m2
```

Significant cognitive leverage for our niche:

- VERY good at math (written by professional physicist)
- ***Nearly analytic*** expressions
- Efficient as compiled C for whole-array arithmetic
- familiar "C-like" procedural syntax
- familiar "IDL-like" whole-array operations

Other significant factors:

- S-Lang library designed from outset to be embeddable
- 800K download; 5-10 min compile on average system
- generates only 500K shared object lib (e.g. Solaris)
- international following: bundled w/ all major Linux
- ported to just about everywhere else
- strong reputation for economy, quality, speed
- used in a number of popular/visible open-source packages:
  - JED (faster, smaller Emacs; Stallman doesn't like!)
  - SLRN (possibly most popular open-source newsreader)
  - Lynx (popular character-based web browser)
  - Midnight Commander (directory/file manager)
  - Mutt (email client, successor to Elm)

Using S-Lang with CIAO

Interactive/CL: sherpa, chips, isis, PRISM (tangentially)

Batch: **slsh**, sherpa, chips, isis

Importable Modules (compiled C/C++/Fortran libs)

varmm	abstraction between C/C++ and S-Lang
	maps disk files to/from S-Lang structs (DM, FITS, ASCII, RDB)
	maps S-Lang vars to/from C/C++ scope
	misc utility funcs, VI keybindings, etc
XPA	SAO communications lib
isis, chips, sherpa	module-ized versions of CL apps
paramio	CIAO IRAF-compat param interface
tools libs	CALDB, regions, grouping, pixlib, stacklib
SLgtk	bindings to Gtk graphical widget set
Perl	forthcoming (see Doug Burke)

- Fundamentally **different** approach to analysis
- Per experience, demonstrably more flexible/powerful than:
  - Simple command-oriented interactive applications
  - Tool-oriented **file in, file out** at UNIX prompt

Useful in Ways Not Envisioned ...

- Borne out many times already, per DSDEVEL/HELPDESK:
  - vtpdetect: float/int image conversion
    http://icxc.harvard.edu/pipe/ascds_help/2001/2582.html
  - merge ascii table column into fits table
    http://icxc.harvard.edu/pipe/ascds_help/2001/2406.html
  - modeling/fitting functions of 2 independent variables
    http://icxc.harvard.edu/pipe/dsdevel/2001/2893.html
  - Clearing EVENT status bits
    http://space.mit.edu/CXC/ISIS/examples/clear_bits
  - and more ...
- WebGUIDE (pure S-Lang GUIDE, w/in chips backend)
- ASCII/RDB input to PRISM
- And to entire DM toolset, through ToolAgent
- 250 S-Lang SLOCS fills 5+ year gap in DM I/O
- 8-10X faster than IDL ascii I/O
  (icxc.harvard.edu/pipe/dsdevel/2001/2791.html)
- None of these were *goals*, per se, of CIAO development
- Trend of spontaneous, creative solutions should increase
- As CIAO capability & user familiarity increases

Case Study: add_grating_orders

Nutshell Scripting Comparison

Metric	Bourne Shell	S-Lang	% Savings
Script Length (lines)	494	340	32%
Non-blank Comments	23	23	na
#Lines Invoking processes	50	4	92%
#Lines Creating UNIX Pipes	18	0	100%

- Qualitatively different: far fewer temp files/subprocesses
- KISS: Simpler == GOOD
- S-Lang avoids 16 UNIX/DM tools needed in Bourne shell

Performance (CIAOX, 12/12/01)

Task	Bourne Shell	S-Lang	xDiff
Coadding +1 -1 order spectra and pre-made ARFs for HRC-S-LETG	144.5 sec	5.55	26x
Coadding +2 -2 order spectra and ARFs for ACIS-HETG[HEG]	89.2	6.15	14.5x
Coadding +3 -3 order spectra and ARFs for ACIS HETG (MEG)	85.7	6.2	13.8x