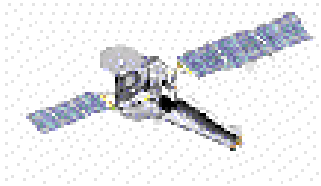


S-Lang for Data Analysis

David Huenemoerder (dph@space.mit.edu)
for the CXC/SDS (12 June 2003)

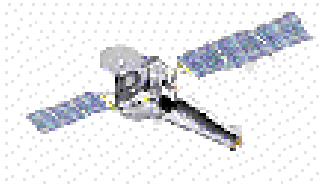
A Brief Recap

Why script?

- to support customized, extendable, automated, and interactive analysis
- to provide a rapid development and prototyping environment, rapid CXC response to ad hoc user requests

Why S-Lang?

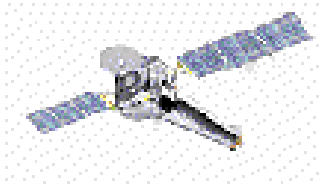
- Fast, efficient, reliable.
- Familiar syntax, strong numerical support.



System-Dependent Modules and Packages

These are all "system-dependent" applications - they depend on CIAO or ISIS:

- **CIAO-3 sherpa/chips: many commands now available as S-Lang functions**
- **CIAO-3 libraries: New modules to access CIAO libraries interactively or from S-Lang scripts (`caldb group paramio pixlib region stackio varmm xpa`)**
- **ISIS: Interactive Spectral Interpretation System; Grating spectral analysis package (see other talk)**
- **Bayesian Blocks: a light-curve utility (unbinned or binned) (`http://space.mit.edu/CXC/analysis/SITAR/index.html`)**
- **Response Library: "Analysis Reference Data" — response functions: for dynamic/interactive/scriptable response generation. (in very early stages of consideration)**



Generic Modules for Advanced Analysis

These are building blocks for S-Lang data analysis packages. As modules, they are have no external dependencies and can be used in any S-Lang environment. We will use them to build CIAO-related analysis utilities, in conjunction with CIAO-libraries which "understand" the Chandra X-Ray Observatory data and responses.

GSL: **Gnu Scientific Library special functions, interpolation, physical constants.** (`http://space.mit.edu/CXC/software/slang/modules/gsl/`) **(released)**

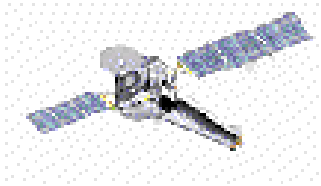
GTK: **GIMP Tool Kit - Graphics/widgets (beta release)**
(`http://space.mit.edu/ mnoble/slgtk/`)

In use for sherpa/chips configuration GUIs. (see other talk)

"`vwhere`": **a visual "where" GUI-let, for those hard-to-express data selections.**

cfitsio : **S-Lang-wrapped cfitsio library; already basis for CIAO/S-Lang, ISIS i/o, but re-packaged for independent use. (currently under test)**

PVM: **Parallel Virtual Machine, for distributed processing. (currently under test)**

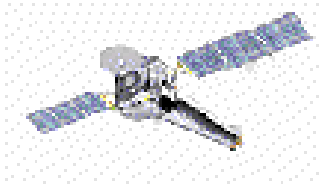


More on S-Lang

For details, see "S-Lang as the CIAO Scripting Language" (`http://space.mit.edu/CXC/docs/SLang/whyslang.ps.gz`).

For a comparison of performances of programming languages, see `http://dada.perl.it/shootout/`, which now includes S-Lang in the benchmarks.

S-Lang is significantly faster than perl, tcl, Java, and some compiled languages at tasks important to scientists.



Quick Examples, Intuitive programming capability

ISIS/S-Lang

```
isis> i=1; product=1;

isis> for (i=1; i<=5; i++) product *= i;
isis> product;
120
```

Xspec/Tcl/Tk

```
XSPEC> set i 1 ; set product 1
1
XSPEC> while {$i <= 5} {
XSPEC> set product [expr $product * $i]
XSPEC> incr i
XSPEC> }
XSPEC> set product
120
```

```
% Get the param by name.
isis> p4 = get_par ("wabs(1).nH");
```

```
# Get the values specified for parameter 4.
XSPEC> tclout param 4
XSPEC> set par4 [string trim $xspec_tclout]

# Turn it into a Tcl list.
XSPEC> regsub -all { +} $par4 { } cpar4
XSPEC> set lpar4 [split $cpar4]
```

```
% open a file and write to it
isis> fp = fopen (filename,"w");
isis> () = fprintf(fp, "nH=%12.4e\n", p4);
```

```
# Print out the result to the file.
XSPEC> puts $fileid "$i [lindex $lpar4 0]"
```