Scripting Ciao with S-Lang

John C. Houck
<houck@space.mit.edu>
Outline

- S-Lang features
- Debugging
- Scripts that run programs
- Creating specialized tools
- Modules
- Example: spectral mapping
S-Lang

- Written by John E. Davis <davis@space.mit.edu>; under development for > 10 years.
- Designed as an embedded interpreter
- http://www.s-lang.org/
S-Lang Features

- variable’s data type is determined by usage
- variety of simple data types (char, int, float, double, complex, string)
- aggregate data types (array, struct, list, associative array)
- variety of looping and control structures
  (for, _for, foreach, loop, while, do, if, switch)
- extensive subroutine library
- fast, efficient array-based math
Its a handy calculator:

sherpa> (4*PI/3) * (3.086e18)^3
1.23105e+56
sherpa> sqrt ((4249.3-3918.1)^2 + (3102.6-3347)^2)
411.612
sherpa> r =sqrt ((4249.3-3918.1)^2 + (3102.6-3347)^2)
sherpa> a = PI*r^2;
sherpa> a;
532264
sherpa>
Working with Arrays

Traditional Method:

```plaintext
x = Double_Type[20];
for (i=0; i<20; i++)
{
    x[i] = sin (2*PI*i/20.0);
}
```

In S-Lang, this is faster:

```plaintext
x = sin ((2*PI/20.0)*[0:19]);
```
Working with Arrays

Traditional Method:

```c
for (i=0; i<20; i++)
{
    if (x[i] < 0)
        x[i] = 0;
}
```

In S-Lang, this is faster:

```c
x[where(x < 0)] = 0;
```
How $x[\text{where}(x < 0)] = 0$ works:

1. $x<0$ tests each element of $x$ to produce an array of 0s and 1s.

   $$\text{test} = x < 0;$$

2. The $\text{where}$ function returns a list of indices that indicates where its argument has non-zero elements.

   $$i = \text{where } (\text{test});$$

3. The value of $x$ at each of the indices is set to 0.

   $$x[i] = 0;$$
Debugging

- `vmessage, fprintf`
- `_print_stack;`

  (2) [Array_Type]: Double_Type[3]
  (1) [String_Type]: a
  (0) [Integer_Type]: 2

- `_traceback=n;`
  have interrupt print trace information.
  Options are `n=-1, 0, 1`

- `_debug_info=1;`
  have trace include line number information.
Running External Programs

Generate the command line:

```python
    cmd = sprintf("pgm %s %s", arg1, arg2);
```

Run the command in a subshell:

```python
    status = system(cmd);
```
Example: Running \texttt{mkarf}

```python
public define generate_arf (x, y)
{
    variable cmd_format, cmd, outfile, def;

    outfile = sprintf("arf_%g_%g.fits", x, y);
    cmd_format = "mkarf outfile=%s sourcepixelx=%f "
                 + "sourcepixely=%f %s";
    def = ["mirror=HRMA", "grating=NONE"];

    cmd = sprintf(cmd_format, outfile, x, y, strjoin(def, " "));

    return system (cmd);
}
```
Automation

Explicit Loop:

```c
for (i = 0; i < n; i++)
{
    status[i] = generate_arf (x[i], y[i]);
}
```

Alternative:

```c
status = array_map (Integer_Type, &generate_arf, x, y);
```
Example: Bit Manipulations

```plaintext
define status_bits_histogram (evt_file) {
    variable status = fits_read_col (evt_file, "status");
    status = status [where (status)];

    variable i, hist = Int_Type[32];

    for (i = 0; i < 32; i++)
    {
        hist[i] = length(where(status&(1 shl i)));
    }

    return hist;
}
```
A Status Bits Tool

```bash
#!/usr/bin/env slsh
if (__argc != 2) {
    vmmessage("Usage: %s: evt-file\n", __argv[0]);
    exit (1);
}
require ("fits");
variable file = __argv[1];
define status_bits_histogram (evt_file) {...}
variable i, hist = status_bits_histogram (file);

for (i = 0; i < 32; i++) {
    if (hist[i])
        vmmessage ("Bit %02d: %d", i, hist[i]);
} exit (0);
```
Tool demo:

/tmp> ./statustool
Usage: ./statustool: evt-file

/tmp> ./statustool acisf03828_000N001_evt1.fits.gz
Bit 04: 459182
Bit 05: 112481
Bit 06: 13093
Bit 16: 47692
Bit 17: 16896
Bit 18: 2627
Bit 19: 207

bits 4-6 = bad pixels
bits 16-19 = afterglow
Modules

- cfitsio
- histogram
- SLgsl (GNU Scientific Library)
- pgplot
- SLgtk (graphical interface toolkit)
- pvvm (Parallel Virtual Machine)
- paramio
Example: spectral mapping

- use adaptively sized spectral extraction regions (may overlap)
- 64x64 pixel map $\implies$ 4096 spectrum fits
- read event file once
- extract and fit using S-Lang variables (no FITS files)