

*AHELP for CIAO 3.4*

variables

Context: [slang](#)*Jump to: [Description](#) [See Also](#)*

Synopsis

Variables in S-Lang

Description

S-Lang allows you to define variables that hold scalars, arrays, structures, or user-defined data types. Variable names in S-Lang are case sensitive, and the data type of a variable is determined upon assignment:

```
variable foo, bar, baz;
foo = 5;
bar = [0, 5, 10, 15, 20];
baz = "This is a string.";
```

After these assignments, `foo` is an integer (`Integer_Type`), `bar` an array (`Array_Type`) of integers, and `baz` a string (`String_Type`). It is also possible for a variable to be a structure, with fields that store data of different types:

```
variable fileinfo = struct { pathname, filename, nrows };
variable foo = @fileinfo;
foo.pathname = "/data/ciao/";
foo.filename = "evt2.fits";
foo.nrows = 100;
```

The above defines a variable `fileinfo` to be a structure, and then populates the elements of this structure. Structures are used to store data returned by `Varmm` routines, where the data is stored in arrays, and the metadata – such as the number of rows in a table – are stored in fields beginning with a single underscore (ie `'_'`) character. Note that the `"@fileinfo"` command uses the deference operator (`@`) to create an instance of the `fileinfo` structure.

The `Varmm print()` function can be used to view the content of a structure, and S-Lang contains a number of intrinsic functions, such as `typeof()`, for manipulating and querying variable types:

```
chips> print(foo)
pathname      = /data/ciao/
filename      = evt2.fits
nrows         = 100
chips> print(typeof(foo))
Struct_Type
chips> print(typeof(foo.filename))
String_Type
chips> print(typeof(foo.nrows))
Integer_Type
```

Ahelp: variables – CIAO 3.4

Structures are used to store data read in by a Varmm function such as `readfile()`, or if you wish to create a FITS file using `writefits()`. In the following, we read in an ASCII file containing two columns into a structure, and then use the `print()` function to view its contents.

```
sherpa> AGauss = readascii("phas.dat");
sherpa> print(AGauss)
_filename      = phas.dat
_path          = /data/analysis/
_filter        = NULL
_filetype      = 1
_header        = NULL
_ncols         = 2
_nrows         = 128
col1           = Float_Type[128]
col2           = Float_Type[128]
```

Here we use another Varmm function, `readfile()`, to read in selected columns from an event list. Note that the filename can contain DM filters – here we restrict access to the first ten rows and select only the time and status columns:

```
chips> evt = readfile("evt2.fits[#row=1:10][cols time,status]")
chips> print(evt)
_filename      = evt2.fits
_path          = /data/ciao/
_filter        = [#row=1:10][cols time,status]
_filetype      = 4
_ncols         = 2
_nrows         = 10
time           = Double_Type[10]
status         = UChar_Type[10,4]
```

See Also

chips

[chips, chips_eval](#)

modules

[varmm](#)

sherpa

[sherpa_eval](#)

slang

[math, overview, slang, tips](#)

tools

[ascii2fits](#)

The Chandra X-Ray Center (CXC) is operated for NASA by the Smithsonian Astrophysical Observatory.
60 Garden Street, Cambridge, MA 02138 USA.
Smithsonian Institution, Copyright © 1998–2006. All rights reserved.

URL:
<http://cxc.harvard.edu/ciao3.4/slang-variables.html>
Last modified: December 2006