URL: http://cxc.harvard.edu/ciao3.4/readascii.html
Last modified: December 2006

AHELP for CIAO 3.4

readascii

Context: varmm

Jump to: Description Examples CHANGES IN CIAO 3.1 CHANGES IN CIAO 3.0.2 CHANGES IN CIAO 3.0 Bugs See Also

Synopsis

S−Lang function to read in an ASCII data file.

Syntax

```
Struct_Type readascii( filename )
Struct_Type readascii( filename, cols )
Struct_Type readascii( filename, cols, nskip )

Error Return Value: NULL

Arguments:
filename is a String_Type variable
cols is either a String_Type variable or an Int_Type array
nskip is an Int_Type variable
```

Description

The readascii() function allows you to read in ASCII data files into S−Lang. It can be called either directly or indirectly (i.e. when using the readfile() function). The ahelp page for readfile describes the features of this routine that are common to all the "read" functions provided by the Varmm module. This page describes those features that are unique to the readascii() command.

The filename argument should be a string that contains the name of the file to be read in. Unlike the other "read" functions, it can not include any Data Model syntax. The optional parameter "cols" is a list of column names or numbers to be read, and "nskip" is the number of rows to be skipped. As examples:

```
  chips> d1 = readascii( "table.dat" )
  chips> d2 = readascii( "table.dat","1,3,5", 10 )
```

where the second example only reads in the first, third, and fifth columns and skips the first ten lines of the files.

File format

The format is the same as recognised by the ascii2fits tool. An optional header − with lines beginning with the "#" character − is ignored. The first line that does not begin with a "#" character is used to define the format of the file (i.e. the number of columns). The following lines should contain the same number of columns as the first line, otherwise the data may not be read in correctly. Any lines beginning with a "#" in this section will be read as a set of 0's. Only numeric columns are currently read in; character columns will generally result in a column of zeroes.

What does the function return?

The function returns a structure whose fields contain the data read in from the file. If an error occurred − such as the file not being found, or it is not an ASCII file − then NULL is returned instead. The returned structure follows the format of the other "read" functions: metadata − i.e. information about the file − is stored in fields beginning with an underscore character followed by fields containing the input data. The initial fields are discussed in "ahelp readfile"; here we concentrate on those fields specific to ASCII files.

Fields specific to tables:

Field name:	Description:
_ncols	The number of columns read in.
_nrows	The number of rows read in.
col1	The first column.
col<n>	The nth column.

For example:

```
  chips> !cat test.dat
  1 2 3
  5 9 -3.1
  chips> dat = readascii( "test.dat" )
  chips> print( evt )
  _filename        =  test.dat
  _path            =  /data/analysis/
  _filter          =  NULL
  _filetype        =  1
  _header          =  NULL
  _ncols           =  3
  _nrows           =  2
  col1             =  Float_Type[2]
  col2             =  Float_Type[2]
  col3             =  Float_Type[2]
```

Example 1

Here we read an ASCII file called "phas.dat", which contains two columns, into a S−lang variable called dat:

```
  sherpa> dat = readfile("phas.dat");
```

We then use the print() function to view the contents of this variable:

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

Example 2

Since readfile() calls readascii() when given an ASCII file, the results of the following are the same as in the previous example.

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

Example 3

Here we use the optional arguments to only read in the second column and skip the first 64 lines of the file.

```
sherpa> dat = readascii("phas.dat","2",64);
sherpa> print(dat);
_filename        =  phas.dat
_path            =  /data/analysis/
_filter          =  NULL
_filetype        =  1
_header          =  NULL
_ncols           =  1
_nrows           =  64
col1             =  Float_Type[64]
```

This could also have been achieved by using an array of integers to list the required columns:

```
sherpa> dat = readascii("phas.dat",[2],64);
```

CHANGES IN CIAO 3.1

Files are no longer cached

Prior to CIAO 3.0, if a table or image was read in then that data would be cached so that any attempt to re−read the file would lead to the original data being returned, even if the actual file on disk had changed. In CIAO 3.0 this cacheing was removed for tables, and with the CIAO 3.1 release it has also been removed for images.

Reading a file in a directory containing the string '::'

The routines no longer crash when reading a file within a directory whose name contains the string "::".

Enhanced documentation

The readascii function is now documented separately from readfile.

CHANGES IN CIAO 3.0.2

Stack Underflow errors

It is now possible to use readfile() − or any of the other read functions described here − in an if statement. Prior to CIAO 3.0.2 you could not write something like

```
  if ( NULL == readfile("evt2.fits") ) error("Failed to read file.");
```

since it would result in a "Stack Underflow" error message. This means that many routines that use readfile() − such as Sherpa's load_dataset() and related functions − can also now be used in an if statement such as:

```
  if ( 1 != load_image(imgname) )
    verror( "Unable to load %s as an image.", imgname );
```

CHANGES IN CIAO 3.0

New field "_filetype"

A new field called "_filetype" has been added to the data structure which describes the type of the file read in. The contents of the field are described in the "Format of data structure" section in "ahelp readfile".

Bugs

See the bugs page for the Varmm library on the CIAO website for an up−to−date listing of known bugs.

See Also

modules
: varmm

varmm
: fits_bitpix, readarf, readbintab, readfile, readimage, readpha, readrdb, readrmf, writeascii, writefits

URL: http://cxc.harvard.edu/ciao3.4/readascii.html
Last modified: December 2006