



AHELP for CIAO 3.4

print

Context: [varrrm](#)

Jump to: [Description](#) [Examples](#) [CHANGES IN CIAO 3.1](#) [CHANGES IN CIAO 3.0](#) [Bugs](#) [See Also](#)

Synopsis

S-Lang functions to print variables, arrays and structures

Syntax

```
print( variable )  
printarr( array )  
printarr( array, n )
```

Description

The `print()` and `printarr()` functions are used to print the contents of the supplied variable to the standard output. The `writescii()` function can also be used to display arrays.

The `print()` function

The output from `print()` depends on the type of the supplied variable:

- If the variable is a scalar, the value is printed out.
- If the variable is an array or an associative array then the array is printed out as if `printarr()` had been called.
- If the variable is a structure, each field of the structure is printed out with its value.

The `printarr()` function

The `printarr()` function is used to print out the contents of arrays and associative arrays. The optional second argument (`n` in the synopsis above) is ignored if sent an associative array. If sent an array and the second argument is either not supplied or set to 0 then the whole array is printed to standard output. If `n` is greater than 0 then the first `n` elements of the array are printed, and if `n` is less than 0 then the datatype of the array, together with its dimensions, are output. See the examples below for other ways of selecting a range of elements to print.

Note that `print()` and `printarr()` are limited to displaying the output of one and two dimensional arrays. The array type and dimensions will be displayed, along with a warning message, if given an array with more than two dimensions:

```
chips> y = _reshape([1:24],[3,2,4])
chips> print(y)
Integer_Type[3,2,4]
printarr() currently limited to 1D/2D arrays
```

You can use the S-Lang array-indexing capabilities to select a one- or two-dimensional subset of the array:

```
chips> print( y[0,*,*] )
1      2      3      4
5      6      7      8
chips> print( y[2,*,*] )
17     18     19     20
21     22     23     24
chips> print( y[* ,1,2] )
7
15
23
```

Example 1

Displaying scalar variables

Here we use print() to display the contents of various scalar values.

```
chips> a = "How to print a string"
chips> print(a)
How to print a string
chips> b = 23
chips> print(b)
23
chips> print(a + " and a number (" + string(b) + ")")
How to print a string and a number (23)
```

In the last call to print(), we used the string concatenation operator (+) to combine strings, and converted a number to a string. For complicated formats it is probably easier to use the S-Lang intrinsic functions such as vmessage() and printf().

Example 2

Displaying arrays

If you supply print() with an array, it will output the contents of the array (as does printarr).

```
chips> x = [1:3]
chips> y = x + 0.5
chips> print(x)
1
2
3
chips> print(y)
1.5
2.5
3.5
chips> printarr(y,-1)
Double_Type[3]
```

Example 3

Changing the format used to write out arrays

The format used to print out numbers can be changed by using the `set_float_format()` routine from the S–Lang Run–Time Library. In the example below we use this function to change the output from "%g" (the default) to "%13.6e", which means to use exponential notation with 6 numbers after the decimal point and placed within a space 13 characters wide.

```
chips> x = 10^( [0:10:2] / 3.0 )
chips> print( x )
1
4.64159
21.5443
100
464.159
2154.43
chips> set_float_format("%13.6e")
chips> print( x )
 1.000000e+00
 4.641589e+00
 2.154435e+01
 1.000000e+02
 4.641589e+02
 2.154435e+03
chips> set_float_format("%g")
```

Note that we reset the format to the default value ("%g") to avoid changing the layout of other commands, such as `printarr()` and `writescii()`.

Example 4

Displaying structures

If you have a structure – such as returned by the `varmm readfile()` function and its variants – then `print()` can be used to view its contents:

```
chips> arf = readfile("spec.arf")
The inferred file type is ARF.
chips> print(pha)
_filename      = spec.arf
_path          = /data/ciao
_filter        = NULL
_filetype      = 9
_header        = String_Type[162]
_ncols         = 3
_nrows         = 1090
ENERG_LO       = Float_Type[1090]
ENERG_HI       = Float_Type[1090]
SPECRESP       = Float_Type[1090]
```

as well as individual fields of the structure.

```
chips> print(arf.SPECRESP)
0.0232154
0.0509929
0.121241
...
```

Example 5

Displaying associative arrays

The `print()` and `printarr()` commands can also be used to display the contents of associative arrays:

```
chips> xx = Assoc_Type [String_Type]
chips> xx["foo"] = "a string"
chips> xx["bar"] = "another string"
chips> xx["1 3"] = time()
chips> print(xx)
Assoc_Type[1 3] = Wed May 21 10:16:42 2003
Assoc_Type[bar] = another string
Assoc_Type[foo] = a string
```

Note that the order of the keys of the associative array need not match the order that they were defined, as shown above.

Example 6

Limiting the number of elements displayed

When using `printarr()` to display an array there are several methods you can use to restrict the number of elements displayed.

The optional second parameter of `printarr()`, if set to a value greater than 0, limits the display to that many elements:

```
chips> y = [1:10] + 0.5
chips> printarr(y,3)
1.5
2.5
3.5
```

The S-Lang array manipulation capabilities provide much more flexibility, such as picking any range of the array, rather than always starting at the first element

```
chips> printarr(y[[2:3]])
3.5
4.5
```

(remember that S-Lang starts counting from 0 for its array elements, so index 2 is the third element in the array).

CHANGES IN CIAO 3.1

OS-X

In CIAO 3.0, the `print()` and `printarr()` functions would fail on OS-X if given an array with more than 20 elements and the `PAGER` environment variable was not set.

CHANGES IN CIAO 3.0

`print()` now behaves like `printarr()` when given an array

Prior to CIAO 3.0, if `print()` was called with an array then the array type and dimensionality would have been

displayed, as shown in the following example.

```
chips> print([1:3])
Integer_Type[3]
```

As of CIAO 3.0, print() now calls printarr() to display the contents, so the result is now:

```
chips> print([1:3])
1
2
3
```

If you want to find out the array type and dimensions then you can:

- call printarr() with the optional second parameter set to -1
- use the "stack clearing" behaviour of ChIPS and S-Lang and just give the array at the prompt
- call the array_info() and _typeof() S-Lang intrinsic functions

An example of the first two methods is shown below:

```
chips> a = [1:3]
chips> b = sin([1:3])
chips> printarr(a,-1)
Integer_Type[3]
chips> a
Integer_Type[3]
chips> b
Double_Type[3]
```

Associative arrays are now displayed

If given an associative array then print() and printarr() now display the contents of the array, as shown in the examples section.

Bugs

See the [bugs page for the Varrr library](#) on the CIAO website for an up-to-date listing of known bugs.

See Also

modules

[varrr](#)

varrr

[apropos](#), [clearstack](#), [dup_struct](#), [is_struct_defined](#), [reverse](#)

