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

AHELP for CIAO 3.4

apropos

Context: varmm

Jump to: Description Examples Bugs See Also

Synopsis

S−Lang function to find defined symbols (functions and variables)

Syntax

```
apropos( match )
apropos( match, namespace )
```

Description

The S−Lang run−time library provides the _apropos() function which returns a list of defined symbol names (i.e. functions and variables) that contain a given string. The Varmm apropos() function is a wrapper around this function to make it simpler to use from the ChIPS and Sherpa prompts.

When called with only one argument, apropos() searches for all S−Lang functions and variables which match the argument and are defined in the default namespace (which is called "Global"). Therefore

```
apropos("state")
```

will report on any symbol that contains the string "state".

When called with two arguments, the search is restricted to only those symbols defined in the given namespace that match. So, if the namespace "foo" is defined then

```
apropos("state","foo")
```

will report on any symbol that contains the string "state" in the namespace "foo".

Any matches are printed to the standard output, which makes it easy to use in interactive sessions. If you want more control over the output then use the S−Lang intrinsic function _apropos().

Example 1

```
chips> apropos("scale")
Found 6 matches in Global namespace:
chips_set_xscale         chips_get_yscale       chips_get_xscale
chips_get_zscale         chips_set_yscale       chips_set_zscale
```

In this example we find that there are six S−Lang symbols available in ChIPS that contain the name scale. It is not possible to tell from this output whether they are functions or variables. To find this out you can use either of the S−Lang intrinsic functions is_defined() or _apropos():

```
chips> is_defined("chips_get_xscale")
2
```

In this example they all happen to be user−defined functions, as indicated by the return value of 2 (see "ahelp is_defined" for more information).

Example 2

```
chips> apropos("version")
Found 5 matches in Global namespace:
_slang_version           _varmm_version         _slang_version_string
_chips_version_string    _chips_version
```

The set of defined symbols depends on what modules have been loaded. ChIPS, by default, has the ChIPS and Varmm modules loaded and, as shown above, this provides a number of symbols (this time all read−only variables) that contain the string "version" in their name.

However, we can change this by loading in more S−Lang modules into ChIPS, as shown in the following example where we load Sherpa into ChIPS and then see if the number of "version" variables has changed:

```
chips> require("sherpa")
    Abundances set to Anders & Grevesse
chips> apropos("version")
Found 9 matches in Global namespace:
_slxpa_version_string      _slang_version            sherpa_version
_varmm_version             xpa_version               _slxpa_version
_slang_version_string      _chips_version_string     _chips_version
```

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

clearstack, dup_struct, is_struct_defined, print, reverse

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