

**NAME**

syms – list of known symbols

**SYNOPSIS**

```
syms [-v] [+|-BGLADNSIC] [name ... ]
```

**DESCRIPTION**

The command `syms` lists `spec`'s current variables. Without arguments, all the variables are listed, along with their memory consumption and type. With the `-v` flag, the variables are listed along with their values in a format that can be saved to a file and read back as commands. If arguments are given as `name ...`, only symbols matching the arguments are printed. Such arguments may contain the `*` and `?` metacharacters.

Note, the `-v` flag does not display the values of the new data array types. Rather, a declaration suitable for creating the array is printed. Such a declaration does, in fact, clear the contents of an already existing array.

In addition, the type of symbols listed can be controlled using the flags in the following table where a `-` flag prevents symbols with the given attribute from being listed and a `+` flag includes symbols with the given attribute in the list.

- B – Built-In
- G – Global
- L – Local
- A – Associative array
- D – Data array
- N – Number type
- S – String type
- I – Immutable attribute
- C – Constant attribute

Symbol attributes are as follows:

**Built-In** – Symbols that cannot be removed.

**Global** – Symbols that retain their value outside of statement blocks. All built-in symbols are global.

**Local** – Symbols that have scope only within the statement block in which they are used.

**Immutable** – Built-in symbols that cannot have their value altered.

**Constant** – Global symbols that cannot have their value altered by ordinary assignment. They can only have their value altered using the `constant` command.

Symbols can be of string, number or array type. They may be of both string and number type simultaneously. There are two arrays types: associative and data.

The `print` command always prints the string representation of a symbol. The `printf()` and `sprintf()` functions print the number or string representation, depending on the given format specifications. Arithmetic operators use the number value of a symbol. The relational operators `>` `<` `<=` `>=` use the number value if both operands are of number type, otherwise the string values are lexicographically compared.

If the string value of a symbol cannot be interpreted as a number, its number value is zero.

A nonglobal symbol is automatically brought into existence within a statement or statement block (`{ ... }`) simply by using its name.

A symbol assigned a value at the top level (outside of any curly brackets) is automatically made global.

The `local` command forces the scope of a symbol to be within the current statement block and allows reuse of global names within statement blocks. Names already used as macro names may be used as local symbols within a statement block, although that macro then becomes unavailable within that block.

### BUILT-IN SYMBOLS

`spec`'s built-in symbols include the following:

`FRESH` – is nonzero if `spec` was started with the `-f` (fresh) flag.

`VERSION` – a string containing the `spec` version as in "4.03.12".

`PI` – the numeric constant 3.14159...

`USER` – the user's name, from `/etc/passwd`.

`HOME` – the user's home directory, from the environment.

`TERM` – the user's terminal type, from the environment.

`GTERM` – the user's graphics terminal type, from the environment. May be changed while running.

`DISPLAY` – used by X Window clients to figure out on which host to run. On startup, the value is taken from the environment. If not set in the environment, the value saved in the user's state file is used. The value can be changed while running, and the current value will be exported to new `x11filt` processes as they are spawned.

`COLS` – the number of columns on the screen. Initially taken from the environment, may be changed by assignment, unless the `TIOCGWINSZ ioctl()` is available, in which case the values via that call will be used.

`ROWS` – the number of rows on the screen. Obtained as above.

`CWD` – the current working directory.

`SPEC` – the name of the program.

`SPECD` – the auxiliary file directory.

`DEBUG` – the debugging level.

`MOTORS` – the number of motors from the `config` file.

`COUNTERS` – the number of counters from the `config` file.

`A` – used to communicate motor positions.

`S` – holds counter values.

Other built-in symbols are installed by site-dependent code. In particular, the symbols `G`, `Q`, `Z` and `UB` are used with most of the standard geometry modules.

### EXAMPLE

```
SPEC> syms T*
```

```

80 TEMP_CS      (G.NS.)    96 TITLE        (G..S.)
80 TEMP_SP      (G.NS.)    80 T_AV          (G.NS.)
96 TERAMP_MIN   (G.N..)    80 T_HI_SP       (G.N..)
96 TERM         (B..S.)    80 T_L           (G.NS.)
80 TIME         (G.NS.)    80 T_LO_SP       (G.N..)
96 TIME_END     (G.NS.)
```

```
SPEC> syms -v T*
```

```
TEMP_CS = 0  
TEMP_SP = 0  
TERAMP_MIN = 0.2  
TERM = "xterm"  
TIME = 0  
TIME_END = 0  
TITLE = "fourc"  
T_AV = 0  
T_HI_SP = 100  
T_L = 0  
T_LO_SP = -100
```

**SEE ALSO**

global constant lsdef lscmd whats