

**NAME**

`my_getopts` – parse command line options

**SYNOPSIS**

```
cp my_getopts /some/where

my_prog B<key[:]=val> ...
    FPATH=/some/where; [ function my_getopts { ...; } ]
    my_getopts B<key[:]=val> ... "$@"

FPATH=/some/where; my_getopts _help=y
FPATH=/some/where; my_getopts _show=y
```

**DESCRIPTION**

**my\_getopts** is a function which is used inside a program (`my_prog` in SYNOPSIS) to process the program's command line options. It can be either referenced by `FPATH` and loaded in from filesystem, or statically included in the program.

The **key[:]=val** can be in any of the following forms:

```
key=val KEY=VAL # no perserve case
key:=VaL KEY=VaL # do perserve case
key[:]="val1 val2 ..."
sep=<char> key=val1<char>val2<char>val3...
sep=<char> key:=val1<char>val2<char>val3...
```

The latter option overwrite the previous one, this feature can be used to initialize or specify the default values for certain keys. The last `sep=<char>` is in effect for later options. The `sep=""` is default.

As “`help=y`”, and “`show=y`” are processed on behalf of calling programs, “`_help=y`”, and “`_show=y`” are reserved option for the function itself.

“`_help=y`” displays the man page of `my_getopts`.

“`_show=y`” shows the striped down version of the `my_getopts` to be statically included in shell programs.

**EXAMPLE**

Suppose you run a program “`my_prog`” with options like:

```
$ my_prog sid:=MySID owner=MySelf email:="a B" \
    sep=: Sql:=MySQL:arg1:arg2 sep=, table=tab1:p1,tab2:p2
```

Inside the program, you can use `my_getopts` to process the command line options:

```
$ cat my_prog

FPATH=/some/where # load the my_getopts from /some/where

# or alternatively, you can include the striped down version
# of the function. Your program will not be affected by the
# availability of the function, or version change.

# function my_getopts {
#   <copy of the function, produced by _show=y>
# }

my_getopts "$@" # process the command line options.
```

```

print "Now I have SID    variable to use: $SID"
print "Now I have OWNER variable to use: $OWNER"
print "Now I have EMAIL variable to use: $EMAIL"
print "Now I have SQL    variable to use: $SQL"
print "Now I have TBALE variable to use: $TABLE"

```

The results:

```

Now I have SID    variable to use: MySID
Now I have OWNER variable to use: MYSELF
Now I have EMAIL variable to use: a B
Now I have SQL    variable to use: MySQL arg1 arg2
Now I have TBALE variable to use: TAB1:P1 TAB2:P2

```

Here is an example of output produced by `_show=y` option:

```

$ FPATH=/some/where my_getopts _show=y

function my_getopts {
#- version 3.141592, 2002-02-02, Michael Wang <xw73@columbia.edu>.
typeset PATH= SEP= i
typeset -u _I
for i; do
  _I=$i
  eval ${_I%%?(*)=*}= unset ${_I%%?(*)=*}
  case $_I in
    *[:]=*) eval $(IFS=$SEP; print ${_I%%=*}=\ "${_I#*=}\") ;;
    *:=*)   eval $(IFS=$SEP; print ${_I%%:=*}=\ "${i#*:=}\") ;;
  esac
done
}

```

#### SEE ALSO

*getopt*(1), *getopts*(1).

#### AUTHORS

Michael Wang <xw73@columbia.edu>.

This is free software. You may copy or redistribute it under the same terms as Perl itself.

If you modify it, you are required to send a copy of the modification to the author via email.