

## NAME

getopt, getopt - Process single-character switches with switch clustering

## SYNOPSIS

```
use Getopt::Std;

getopt('oDI');    # -o, -D & -I take arg.  Sets $opt_* as a side
effect.
getopt('oDI', \%opts);    # -o, -D & -I take arg.  Values in %opts
getopts('oif:');    # -o & -i are boolean flags, -f takes an argument
    # Sets $opt_* as a side effect.
getopts('oif:', \%opts); # options as above. Values in %opts
```

## DESCRIPTION

The `getopt()` function processes single-character switches with switch clustering. Pass one argument which is a string containing all switches that take an argument. For each switch found, sets `$opt_x` (where `x` is the switch name) to the value of the argument if an argument is expected, or 1 otherwise. Switches which take an argument don't care whether there is a space between the switch and the argument.

The `getopts()` function is similar, but you should pass to it the list of all switches to be recognized. If unspecified switches are found on the command-line, the user will be warned that an unknown option was given. The `getopts()` function returns true unless an invalid option was found.

Note that, if your code is running under the recommended `use strict 'vars'` pragma, you will need to declare these package variables with "our":

```
our($opt_x, $opt_y);
```

For those of you who don't like additional global variables being created, `getopt()` and `getopts()` will also accept a hash reference as an optional second argument. Hash keys will be `x` (where `x` is the switch name) with key values the value of the argument or 1 if no argument is specified.

To allow programs to process arguments that look like switches, but aren't, both functions will stop processing switches when they see the argument `--`. The `--` will be removed from `@ARGV`.

## --help and --version

If `-` is not a recognized switch letter, `getopts()` supports arguments `--help` and `--version`. If `main::HELP_MESSAGE()` and/or `main::VERSION_MESSAGE()` are defined, they are called; the arguments are the output file handle, the name of option-processing package, its version, and the switches string. If the subroutines are not defined, an attempt is made to generate intelligent messages; for best results, define `$main::VERSION`.

If embedded documentation (in pod format, see *perl/pod*) is detected in the script, `--help` will also show how to access the documentation.

Note that due to excessive paranoia, if `$Getopt::Std::STANDARD_HELP_VERSION` isn't true (the default is false), then the messages are printed on `STDERR`, and the processing continues after the messages are printed. This being the opposite of the standard-conforming behaviour, it is strongly recommended to set `$Getopt::Std::STANDARD_HELP_VERSION` to true.

One can change the output file handle of the messages by setting `$Getopt::Std::OUTPUT_HELP_VERSION`. One can print the messages of `--help` (without the `Usage:` line) and `--version` by calling functions `help_mess()` and `version_mess()` with the switches string as an argument.