



A.P. Lawrence

Information and Resources for Unix and Linux Systems

[Printer Friendly Version](#)

Perl Getopt and GetOptions

[More Articles](#)

Two Perl modules (`Getopt` and `GetOptions::Long`) work to extract program flags and arguments much like [Getopt and Getopts](#) do for shell programming. The Perl modules, especially `GetOptions::Long`, are much more powerful and flexible.

Simple scripts show the power of these:

```
#!/usr/bin/perl
# script is "./g"
```

```

use Getopt::Std;
%options=();
getopts("od:fF",\%options);
# like the shell getopt, "d:" means d takes
an argument
print "-o $options{o}\n" if defined $options
{o};
print "-d $options{d}\n" if defined $options
{d};
print "-f $options{f}\n" if defined $options
{f};
print "-F $options{F}\n" if defined $options
{F};
print "Unprocessed by Getopt::Std:\n" if
$ARGV[0];
foreach (@ARGV) {
    print "$_\n";
}

```

Trying it out:

```

bash-2.05a$ ./g -f -d F -o -F
-o 1
-d F
-f 1
-F 1
bash-2.05a$ ./g -o -d foo
-o 1
-d foo
bash-2.05a$ ./g -o rough -d foo
-o 1
Unprocessed by Getopt::Std:
rough

```

```
-d
foo
```

Processing of arguments stops when it saw "rough".

If you leave off a required argument, it just gets swallowed:

```
bash-2.05a$ ./g -d
bash-2.05a$ ./g -d foo
-d foo
```

But it's easily confused:

```
bash-2.05a$ ./g -d -o -f
-d -o
-f 1
```

It thinks that -o is the argument of -d.

```
bash-2.05a$ ./g -k
Unknown option: k
```

Like the simple shell "getopt", this complains when it gets an option that it doesn't know about. Unlike the shell "getopt", prefacing the option string with a ":" doesn't help. Instead, use "getopt":

```
#!/usr/bin/perl
# we'll call this one ./gg
use Getopt::Std;
%options=();
getopt("odfF", \%options);
print "-o $options{o}\n" if defined $options{o};
print "-d $options{d}\n" if defined $options{d};
print "-f $options{f}\n" if defined $options
```

```
{f};
print "-F $options{F}\n" if defined $options
{F};
```

Note the lack of any ":". This module doesn't care which flags take values and which don't: it assumes ALL of them take arguments.

The "getopt" isn't very bright:

```
bash-2.05a$ ./gg -f -o -d foo
-d foo
-f -o
bash-2.05a$ ./g -f -o -d foo
-o 1
-d foo
-f 1
```

But it doesn't complain:

```
bash-2.05a$ ./gg -l
bash-2.05a$ ./g -l
Unknown option: l
```

Unlike their shell cousins, neither of these have any issues with arguments containing spaces:

```
bash-2.05a$ ./g -o -d "foo bar"
-o 1
-d foo bar
bash-2.05a$ ./gg -o "foo" -d "foo bar"
-o foo
-d foo bar
```

Far better than either of these is the Getopt::Long module. Here's a script to play with it:

```

#!/usr/bin/perl
use Getopt::Long;
GetOptions("o"=>\$oflag,
           "verbose!"=>\
$verboseornoverbose,
           "string=s"=>\$stringmandatory,
           "optional:s",\$optionalstring,
           "int=i"=> \$mandatoryinteger,
           "optint:i"=> \$optionalinteger,
           "float=f"=> \$mandatoryfloat,
           "optfloat:f"=> \$optionalfloat);
print "oflag $oflag\n" if $oflag;
print "verboseornoverbose
$verboseornoverbose\n" if
$verboseornoverbose;
print "stringmandatory $stringmandatory\n"
if $stringmandatory;
print "optionalstring $optionalstring\n" if
$optionalstring;
print "mandatoryinteger $mandatoryinteger
\n" if $mandatoryinteger;
print "optionalinteger $optionalinteger\n"
if $optionalinteger;
print "mandatoryfloat $mandatoryfloat\n" if
$mandatoryfloat;
print "optionalfloat $optionalfloat\n" if
$optionalfloat;

print "Unprocessed by Getopt::Long\n" if
$ARGV[0];
foreach (@ARGV) {
    print "$_\n";
}

```

The hash array that this uses holds the argument name and the type of argument; what that points to is where it will store values for options processed.

Playing with it:

```
#
# doesn't care if it's -o or --o
bash-2.05a$ ./ggg -o
oflag 1
bash-2.05a$ ./ggg --o
oflag 1
#
# abbreviating is ok too
bash-2.05a$ ./ggg -verbose
verboseornoverbose 1
bash-2.05a$ ./ggg -verb
verboseornoverbose 1
#
# but not this
bash-2.05a$ ./ggg -verbosity
Unknown option: verbosity
#
# $verboseornoverbose will be 0 here
bash-2.05a$ ./ggg -noverb
#
# strings
bash-2.05a$ ./gggg -s
Option string requires an argument
bash-2.05a$ ./ggg -s=foo
stringmandatory foo
bash-2.05a$ ./ggg -optional
bash-2.05a$ ./ggg -optional=foo
optionalstring foo
```

```

#
# ambiguity
bash-2.05a$ ./ggg --opt
Option opt is ambiguous (optfloat, optint,
optional)
#
# floats and integers
bash-2.05a$ ./ggg --optfloat=75.6
optionalfloat 75.6
bash-2.05a$ ./ggg --optint=75.6
Value "75.6" invalid for option optint
(number expected)
bash-2.05a$ ./ggg --optint=75
optionalinteger 75
bash-2.05a$ ./ggg -f --optfloat=75.6 -o
Value "--optfloat=75.6" invalid for option
float (real number expected)
oflag 1
# once it runs out of options, it leaves
@ARGV alone:
bash-2.05a$ ./ggg -o foo bar
oflag 1

```

```

Unprocessed by Getopt::Long
foo
bar

```

GetOpt::Long is obviously much more flexible. The hash you pass is a little clumsy, but if you think about it, there's no better way to do it. In some places, you might use something like this:

```

#!/usr/bin/perl
use Getopt::Long;

```

```

my %moo=();
GetOptions("o"=>\$moo{$oflag},
           "verbose!"=>\$moo{verbose},
           "string=s"=>\$moo
{stringmandatory},
           "optional:s"=>\$moo
{optionalstring},
           "int=i"=> \$moo
{mandatoryinteger},
           "optint:i"=> \$moo
{optionalinteger},
           "float=f"=> \$moo
{mandatoryfloat},
           "optfloat:f"=> \$moo
{optionalfloat});

foreach (keys %moo) {
  print "$_ = \$moo{$_}\n";
}

print "Unprocessed by Getopt::Long\n" if
$ARGV[0];
foreach (@ARGV) {
  print "$_\n";
}

```

but if you have so many flags that you are thinking that is helpful, your program is surely trying much too hard to be all things to all people.

Comments /Unix/perlgetopts.html

| (older comments)

PerlGetOpts :

" but if you have so many flags that you are thinking that is helpful, your program is surely trying much too hard to be all things to all people."

Isn't that the philosophy behind Windows development? <Grin>

After reading through all this, I said to myself, "One of these days I have to become an expert in Perl." <Smile>

I have to admit that I'm much more likely to turn to a shell script or C to do a lot of the things that Perl would probably be better suited to doing. In many cases, a Perl program would certainly involve less work than writing and compiling a C program, and would undoubtedly be more elegant and efficient than a typical shell script. However, as with using greenbar paper, old habits are hard to shake. I grok C and the shell, whereas I'm a rank amateur when it comes to Perl. One of these days...

--BigDumbDinosaur

The nice thing about Perl is that you don't have to be anything close to an expert to have it be helpful. If you know awk and sed and a little sh or ksh, you already understand a lot of Perl without even

knowing it yet.

See <http://aplawrence.com/Unixart/loveperl.html>

--TonyLawrence

---September 18, 2004

I recently resumed perl programming after a ten year hiatus;
yup, just like riding a bike - I was really suprised how much perl lore lay dormant in the deep recesses of my aging brain. It has been a boon in refining a nascent automated java build framework I unexpectedly inherited. Combing ant, cruisecontrol, java, ksh, and perl, I was able to produce a flexible production grade build process in a matter of days... Getopts::Long was instrumental in enabling the central build.pl program to flexibly support any flavour of build one can imagine.

--Fleh

---February 10, 2005

SO how would I define the vars when I want to use-strict ? When I put it in I get:
Global symbol "\$oflag" requires explicit package name at ./tester.pl line 5.

```
Global symbol "$verboseornoverbose" requires
explicit package name at ./tester.pl line 6.
...etc. etc
```

Which is easy to fix with normal vars but I don't know how with references in this context! Help appreciated.

```
--Scary B.
```

```
---February 10, 2005
```

Just like usual:

```
my $oflag;
my $verboseornoverbose;
```

```
etc.
```

```
--TonyLawrence
```

Wed May 4 00:00:32 2005 Anyway to get rid of this warning? [Dave \(key aV2QbVUqr\)](#)

```
test_opts2.pl -optional test
```

```
Use of uninitialized value in hash element at /home/dmckeon/scripts/test_opts2.pl
line 11.
```

```
optionalstring = test
```

Program

```
*****
```

```
#!/usr/bin/perl -w
```

```
# Option test dm

use strict;
use Getopt::Long;
my moo=();
my $oflag;

GetOptions("o"=>\$moo{$oflag},
"verbose!"=>\$moo{verbose},
"string=s"=>\$moo{stringmandatory},
"optional:s"=>\$moo{optionalstring},
"int=i"=> \$moo{mandatoryinteger},
"optint:i"=> \$moo{optionalinteger},
"float=f"=> \$moo{mandatoryfloat},
"optfloat:f"=> \$moo{optionalfloat});

foreach (keys moo) {
if ( defined $moo{$_} && exists $moo{$_}) {
print "$_ = $moo{$_}\n";
}
}

print "Unprocessed by Getopt::Long\n" if $ARGV[0];
foreach (@ARGV) {
print "$_\n";
}
}
```

Wed May 4 10:28:54 2005 [TonyLawrence \(key /jwW0kwdn\)](#)

Don't use -w or define your variables. Simple as that.

Mon Sep 12 18:51:10 2005 [anonymous \(key EXEDABwU8\)](#)

-optional is the same as -o -p -t -i -o -n -a -l. You should use --optional.

[Add your comments](#)

[Printer Friendly Version](#)

Views for this page

Today This Week This Month This Year Overall

77 160 184 19,552 50,069

/Unix/perlgetopts.html copyright September 2003 Tony Lawrence All Rights Reserved

[Site map](#) | [Disclaimer](#)

[/Unix/perlgetopts.html copyright and reprint notice](#)



Except where otherwise noted, this content is licensed under a [Creative Commons License](#).

[Publish your articles, comments, book reviews or opinions here!](#)

[Related Articles](#)

Have you tried [Searching this site?](#)

[Please read this disclaimer](#)

Unix/Linux/Mac OS X support by phone, email or on-site: [Support Rates](#)

This is a Unix/Linux resource website. It contains technical articles about Unix, Linux and general computing related subjects, opinion, news, help files, how-to's, tutorials and more. We appreciate comments and [article submissions](#).

[Printer Friendly Version](#)

