Getopt::Complete

Getopt::Complete

tab-completion for Perl apps

Getopt::Complete

tab-completion for Perl apps

Scott Smith

Getopt::Complete

tab-completion for Perl apps

Scott Smith

Genome Center
Washington University School of Medicine

Getopt::Complete

tab-completion for Perl apps

Scott Smith

Genome Center
Washington University School of Medicine

(see David Dooling's “The Freedom to Cure Cancer” tomorrow at 10:45)

Everyone loves “tab completion”.

This makes it easy to add to any Perl app.

In the Perl program "myprogram":

In the Perl program "myprogram":

```
use Getopt::Complete (
    ’frog’         => [’ribbit’,’urp’,’ugh’],
);
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    ’frog’      => [’ribbit’,’urp’,’ugh’],
    ’fraggle’   => sub { return [’rock’,’roll’] },
);
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    ’frog=s’       => [’ribbit’,’urp’,’ugh’],
    ’fraggle=s’   => sub { return [’rock’,’roll’] },
);
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    'frog=s'       => ['ribbit','urp','ugh'],
    'fraggle=s'   => sub { return ['rock','roll'] },
);

print "the frog says " . $ARGS{frog} . "\n";
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    ’frog=s’       => [’ribbit’,’urp’,’ugh’],
    ’fraggle=s’    => sub { return [’rock’,’roll’] },
);

print "the frog says " . $ARGS{frog} . "\n";
```

In ~/.bashrc or ~/.bash_profile, or directly in bash:

```
$ complete -C myprogram myprogram
```

In the Perl program "myprogram":

```
  use Getopt::Complete (
      ’frog=s’        => [’ribbit’,’urp’,’ugh’],
      ’fraggle=s’   => sub { return [’rock’,’roll’] },
  );

  print "the frog says " . $ARGS{frog} . "\n";
```

In ~/.bashrc or ~/.bash_profile, or directly in bash:

```
 $ complete -C myprogram myprogram
```

Thereafter in the terminal (after next login, or sourcing the updated .bashrc):

```
 $ myprogram  --<TAB>
 --fraggle –frog
```

In the Perl program "myprogram":

```
  use Getopt::Complete (
      ’frog=s’        => [’ribbit’,’urp’,’ugh’],
      ’fraggle=s’   => sub { return [’rock’,’roll’] },
  );

  print "the frog says " . $ARGS{frog} . "\n";
```

In ~/.bashrc or ~/.bash_profile, or directly in bash:

```
 $ complete -C myprogram myprogram
```

Thereafter in the terminal (after next login, or sourcing the updated .bashrc):

```
 $ myprogram  --<TAB>
 --fraggle –frog

 $ myprogram  - -f<TAB>
 $ myprogram  - -fr
```

In the Perl program "myprogram":

```
  use Getopt::Complete (
      ’frog=s’        => [’ribbit’,’urp’,’ugh’],
      ’fraggle=s’   => sub { return [’rock’,’roll’] },
  );

  print "the frog says " . $ARGS{frog} . "\n";
```

In ~/.bashrc or ~/.bash_profile, or directly in bash:

```
 $ complete - C myprogram myprogram
```

Thereafter in the terminal (after next login, or sourcing the updated .bashrc):

```
 $ myprogram  --<TAB>
 --fraggle –frog

 $ myprogram  - - f<TAB>
 $ myprogram  - - fr

 $ myprogram  - - fr<TAB><TAB>
 --fraggle --frog
```

In the Perl program "myprogram":

```
  use Getopt::Complete (
      ’frog=s’        => [’ribbit’,’urp’,’ugh’],
      ’fraggle=s’   => sub { return [’rock’,’roll’] },
  );

  print "the frog says " . $ARGS{frog} . "\n";
```

In ~/.bashrc or ~/.bash_profile, or directly in bash:

```
 $ complete -C myprogram myprogram
```

Thereafter in the terminal (after next login, or sourcing the updated .bashrc):

```
 $ myprogram  --<TAB>
 --fraggle –frog

 $ myprogram  --f<TAB>
 $ myprogram  --fr

 $ myprogram  --fr<TAB><TAB>
 --fraggle --frog

 $ myprogram  --fro<TAB>
```

In the Perl program "myprogram":

```
  use Getopt::Complete (
      ’frog=s’        => [’ribbit’,’urp’,’ugh’],
      ’fraggle=s’   => sub { return [’rock’,’roll’] },
  );

  print "the frog says " . $ARGS{frog} . "\n";
```

In ~/.bashrc or ~/.bash_profile, or directly in bash:

```
 $ complete - C myprogram myprogram
```

Thereafter in the terminal (after next login, or sourcing the updated .bashrc):

```
 $ myprogram  --<TAB>
 --fraggle –frog

 $ myprogram - - f<TAB>
 $ myprogram - - fr

 $ myprogram - - fr<TAB><TAB>
 --fraggle --frog

 $ myprogram - - fro<TAB>
 $ myprogram - - frog
```

In the Perl program "myprogram":

```
  use Getopt::Complete (
      ’frog=s’        => [’ribbit’,’urp’,’ugh’],
      ’fraggle=s’   => sub { return [’rock’,’roll’] },
  );

  print "the frog says " . $ARGS{frog} . "\n";
```

In ~/.bashrc or ~/.bash_profile, or directly in bash:

```
 $ complete -C myprogram myprogram
```

Thereafter in the terminal (after next login, or sourcing the updated .bashrc):

```
 $ myprogram  --<TAB>
 --fraggle –frog

 $ myprogram  - -f<TAB>
 $ myprogram  - -fr

 $ myprogram  - -fr<TAB><TAB>
 --fraggle --frog

 $ myprogram  - -fro<TAB>
 $ myprogram  - -frog

 $ myprogram  - -frog <TAB>
 ribbit urp ugh
```

In the Perl program "myprogram":

```
  use Getopt::Complete (
      ’frog=s’        => [’ribbit’,’urp’,’ugh’],
      ’fraggle=s’   => sub { return [’rock’,’roll’] },
  );

  print "the frog says " . $ARGS{frog} . "\n";
```

In ~/.bashrc or ~/.bash_profile, or directly in bash:

```
 $ complete - C myprogram myprogram
```

Thereafter in the terminal (after next login, or sourcing the updated .bashrc):

```
 $ myprogram  --<TAB>
 --fraggle –frog

 $ myprogram  - - f<TAB>
 $ myprogram  - - fr

 $ myprogram  - - fr<TAB><TAB>
 --fraggle --frog

 $ myprogram  - - fro<TAB>
 $ myprogram  - - frog

 $ myprogram  - - frog <TAB>
 ribbit urp ugh

 $ myprogram  - - frog r<TAB>
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    ’frog=s’       => [’ribbit’,’urp’,’ugh’],
    ’fraggle=s’   => sub { return [’rock’,’roll’] },
);

print "the frog says " . $ARGS{frog} . "\n";
```

In ~/.bashrc or ~/.bash_profile, or directly in bash:

```
$ complete -C myprogram myprogram
```

Thereafter in the terminal (after next login, or sourcing the updated .bashrc):

```
$ myprogram  --<TAB>
--fraggle –frog

$ myprogram  - -f<TAB>
$ myprogram  - -fr

$ myprogram  - -fr<TAB><TAB>
--fraggle --frog

$ myprogram  - -fro<TAB>
$ myprogram  - -frog

$ myprogram  - -frog <TAB>
ribbit urp ugh

$ myprogram  - -frog r<TAB>
$ myprogram  - -frog ribbit
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    ’frog=s’        => [’ribbit’,’urp’,’ugh’],
    ’fraggle=s’   => sub { return [’rock’,’roll’] },
);
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    ’frog=s’        => [’ribbit’,’urp’,’ugh’],
    ’fraggle=s’   => sub { return [’rock’,’roll’] },
);

GetOptions(%myargs, ...);
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    ’frog=s’        => [’ribbit’,’urp’,’ugh’],
    ’fraggle=s’   => sub { return [’rock’,’roll’] },
);

%ARGS # alias for %Getopt::Complete::ARGS
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    ’frog=s’       => [’ribbit’,’urp’,’ugh’],
    ’fraggle=s’   => sub { return [’rock’,’roll’] },
);

%ARGS # alias for %Getopt::Complete::ARGS

$ARGS  # alias for $Getopt::Complete::ARGS object
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    ’frog=s’        => [’ribbit’,’urp’,’ugh’],
    ’fraggle=s’   => sub { return [’rock’,’roll’] },
);

%ARGS # alias for %Getopt::Complete::ARGS

$ARGS  # alias for $Getopt::Complete::ARGS object
```

(See perldoc Getop::Complete::Args.)

In the Perl program "myprogram":

```
use Getopt::Complete (
    ’frog=s’        => [’ribbit’,’urp’,’ugh’],
    ’fraggle=s’   => sub { return [’rock’,’roll’] },
    'name=s'      => undef,
);
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    ’frog=s’        => [’ribbit’,’urp’,’ugh’],
    ’fraggle=s’   => sub { return [’rock’,’roll’] },
    'name=s'     => undef,
    'quiet!'        => undef,
);
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    ’frog=s’       => [’ribbit’,’urp’,’ugh’],
    ’fraggle=s’    => sub { return [’rock’,’roll’] },
    'name=s'       => undef,
    'quiet!'       => undef,
    'go!'          => undef,
);
```

In the Perl program "myprogram":

```
  use Getopt::Complete (
      ’frog=s’        => [’ribbit’,’urp’,’ugh’],
      ’fraggle=s’   => sub { return [’rock’,’roll’] },
      'name=s'     => undef,
      'quiet!'         => undef,
      'go!'              => undef,
  );
```

In the shell:

```
 $ myprogram --<TAB>
--fraggle –frog –go –name –quiet --no-
```

In the Perl program "myprogram":

```
 use Getopt::Complete (
     ’frog=s’       => [’ribbit’,’urp’,’ugh’],
     ’fraggle=s’    => sub { return [’rock’,’roll’] },
     'name=s'       => undef,
     'quiet!'       => undef,
     'go!'          => undef,
 );
```

In the shell:

```
 $ myprogram --no<TAB>
--no-go --no-quiet
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    ’frog=s’       => [’ribbit’,’urp’,’ugh’],
    ’fraggle=s’    => sub { return [’rock’,’roll’] },
    'name=s'       => undef,
    'quiet!'       => undef,
    'go!'          => undef,
);
```

In the shell:

```
$ myprogram --no-q<TAB>
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    ’frog=s’       => [’ribbit’,’urp’,’ugh’],
    ’fraggle=s’    => sub { return [’rock’,’roll’] },
    'name=s'       => undef,
    'quiet!'       => undef,
    'go!'          => undef,
);
```

In the shell:

```
$ myprogram –no-quiet
```

In the Perl program "myprogram":

```
  use Getopt::Complete (
      ’frog=s’       => [’ribbit’,’urp’,’ugh’],
      ’fraggle=s’    => sub { return [’rock’,’roll’] },
      'name=s'       => undef,
      'quiet!'       => undef,
      'go!'          => undef,
  );
```

In the shell:

```
 $ myprogram –name <TAB>
```

In the Perl program "myprogram":

```
  use Getopt::Complete (
      ’frog=s’        => [’ribbit’,’urp’,’ugh’],
      ’fraggle=s’   => sub { return [’rock’,’roll’] },
      'name=s'     => undef,
      'quiet!'        => undef,
      'go!'             => undef,
  );
```

In the shell:

```
 $ myprogram –name <TAB>
(nothing appears)
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    ’frog=s’        => [’ribbit’,’urp’,’ugh’],
    ’fraggle=s’     => sub { return [’rock’,’roll’] },
    'name=s'        => undef,
    'quiet!'        => undef,
    'go!'           => undef,
    'out=s@'        => 'directories'
);
```

In the Perl program "myprogram":

```
  use Getopt::Complete (
      ’frog=s’        => [’ribbit’,’urp’,’ugh’],
      ’fraggle=s’   => sub { return [’rock’,’roll’] },
      'name=s'     => undef,
      'quiet!'        => undef,
      'go!'             => undef,
      'out=s@'     => 'directories'
  );
```

```
$ myprogram --o<TAB>
```

In the Perl program "myprogram":

```
  use Getopt::Complete (
      ’frog=s’        => [’ribbit’,’urp’,’ugh’],
      ’fraggle=s’   => sub { return [’rock’,’roll’] },
      'name=s'     => undef,
      'quiet!'         => undef,
      'go!'              => undef,
      'out=s@'     => 'directories'
  );
```

```
$ myprogram –out
```

In the Perl program "myprogram":

```
  use Getopt::Complete (
      ’frog=s’        => [’ribbit’,’urp’,’ugh’],
      ’fraggle=s’   => sub { return [’rock’,’roll’] },
      'name=s'     => undef,
      'quiet!'         => undef,
      'go!'              => undef,
      'out=s@'     => 'directories'
  );
```

```
$ myprogram –out <TAB>
```

In the Perl program "myprogram":

```
 use Getopt::Complete (
     ’frog=s’        => [’ribbit’,’urp’,’ugh’],
     ’fraggle=s’     => sub { return [’rock’,’roll’] },
     'name=s'        => undef,
     'quiet!'        => undef,
     'go!'           => undef,
     'out=s@'        => 'directories'
 );
```

```
$ myprogram –out dir
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    ’frog=s’      => [’ribbit’,’urp’,’ugh’],
    ’fraggle=s’   => sub { return [’rock’,’roll’] },
    'name=s'      => undef,
    'quiet!'      => undef,
    'go!'         => undef,
    'out=s@'      => 'directories'
);
```

```
$ myprogram –out dir<TAB>
```

In the Perl program "myprogram":

```
  use Getopt::Complete (
      ’frog=s’        => [’ribbit’,’urp’,’ugh’],
      ’fraggle=s’   => sub { return [’rock’,’roll’] },
      'name=s'     => undef,
      'quiet!'         => undef,
      'go!'              => undef,
      'out=s@'     => 'directories'
  );
```

```
$ myprogram –out dir
dir1/ dir2/ dir2/
```

In the Perl program "myprogram":

```
  use Getopt::Complete (
      ’frog=s’        => [’ribbit’,’urp’,’ugh’],
      ’fraggle=s’   => sub { return [’rock’,’roll’] },
      'name=s'     => undef,
      'quiet!'        => undef,
      'go!'              => undef,
      'out=s@'     => 'directories'
  );
```

```
$ myprogram –out dir1
```

In the Perl program "myprogram":

```
  use Getopt::Complete (
      ’frog=s’        => [’ribbit’,’urp’,’ugh’],
      ’fraggle=s’   => sub { return [’rock’,’roll’] },
      'name=s'     => undef,
      'quiet!'         => undef,
      'go!'              => undef,
      'out=s@'     => 'directories'
  );
```

```
$ myprogram –out dir1 –o<TAB>
```

In the Perl program "myprogram":

```
  use Getopt::Complete (
      ’frog=s’        => [’ribbit’,’urp’,’ugh’],
      ’fraggle=s’   => sub { return [’rock’,’roll’] },
      'name=s'     => undef,
      'quiet!'         => undef,
      'go!'              => undef,
      'out=s@'     => 'directories'
  );
```

```
$ myprogram –out dir1 –out
```

In the Perl program "myprogram":

```
  use Getopt::Complete (
      ’frog=s’        => [’ribbit’,’urp’,’ugh’],
      ’fraggle=s’   => sub { return [’rock’,’roll’] },
      'name=s'     => undef,
      'quiet!'         => undef,
      'go!'              => undef,
      'out=s@'     => 'directories'
  );
```

```
$ myprogram –out dir1 –out <TAB>
```

In the Perl program "myprogram":

```
  use Getopt::Complete (
      ’frog=s’        => [’ribbit’,’urp’,’ugh’],
      ’fraggle=s’   => sub { return [’rock’,’roll’] },
      'name=s'     => undef,
      'quiet!'        => undef,
      'go!'             => undef,
      'out=s@'     => 'directories'
  );
```

```
$ myprogram –out dir1 –out <TAB>
dir1/ dir2/ dir2/
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    ’myfile’       => ’files’,        # or ’f’
    ’mydir’        => ’directories’,  # or ’d’
    ’mycommand’    => ’commands’,     # or ’c’
    ’myuser’       => ’users’,        # or ’u’
    ’mygroup’      => ’groups’,       # or ’d’
    ’myenv’        => ’environment’,  # or ’e’
    ’myservice’    => ’services’,     # or ’s’
    ’myalias’      => ’aliases’,      # or ’a’
    ’mybuiltin’    => ’builtins’      # or ’b’
);
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    ’myfile’         => ’files’,           # or ’f’
    ’mydir’          => ’directories’,   # or ’d’
    ’mycommand’ => ’commands’,   # or ’c’
    ’myuser’         => ’users’,          # or ’u’
    ’mygroup’       => ’groups’,        # or ’d’
    ’myenv’          => ’environment’, # or ’e’
    ’myservice’    => ’services’,       # or ’s’
    ’myalias’        => ’aliases’,         # or ’a’
    ’mybuiltin’      => ’builtins’          # or ’b’
);
```

```
$ man bash
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    ’myfile’       => ’files’,          # or ’f’
    ’mydir’        => ’directories’,    # or ’d’
    ’mycommand’    => ’commands’,       # or ’c’
    ’myuser’       => ’users’,          # or ’u’
    ’mygroup’      => ’groups’,         # or ’d’
    ’myenv’        => ’environment’,    # or ’e’
    ’myservice’    => ’services’,       # or ’s’
    ’myalias’      => ’aliases’,        # or ’a’
    ’mybuiltin’    => ’builtins’        # or ’b’
);
```

```
$ compgen -h
$ man bash
```

In the Perl program "myprogram":

```
  use Getopt::Complete (
      ’frog=s’        => [’ribbit’,’urp’,’ugh’],
      ’fraggle=s’   => sub { return [’rock’,’roll’] },
      'name=s'     => undef,
      'quiet!'         => undef,
      'go!'              => undef,
      'out=s@'     => 'directories',
      '<>'              => 'files'
  );
```

```
$ myprogram <TAB>
dir1/ dir2/ dir3/ file1 file2 file3
```

In the Perl program "myprogram":

```
  use Getopt::Complete (
      ’frog=s’        => [’ribbit’,’urp’,’ugh’],
      ’fraggle=s’   => sub { return [’rock’,’roll’] },
      'name=s'     => undef,
      'quiet!'         => undef,
      'go!'              => undef,
      'out=s@'     => 'directories',
      '<>'              => 'files'
  );
```

```
$ myprogram dir1/<TAB>
dir1/dirX dir1/fileA dir1/fileB dir1/fileC
```

In the Perl program "myprogram":

```
  use Getopt::Complete (
      ’frog=s’        => [’ribbit’,’urp’,’ugh’],
      ’fraggle=s’     => sub { return [’rock’,’roll’] },
      'name=s'        => undef,
      'quiet!'        => undef,
      'go!'           => undef,
      'out=s@'        => 'directories',
      '<>'            => 'files'
  );
```

```
$ myprogram dir1/dirX/<TAB>
dir1/dirX/file1 dir1/dirX/dir1
```

In the Perl program "myprogram":

```
  use Getopt::Complete (
      ’frog=s’       => [’ribbit’,’urp’,’ugh’],
      ’fraggle=s’    => sub { return [’rock’,’roll’] },
      'name=s'       => undef,
      'quiet!'       => undef,
      'go!'          => undef,
      'out=s@'       => 'directories',
      '<>'           => 'files'
  );
```

```
$ myprogram dir1/dirX/f<TAB>
```

In the Perl program "myprogram":

```
  use Getopt::Complete (
      ’frog=s’        => [’ribbit’,’urp’,’ugh’],
      ’fraggle=s’   => sub { return [’rock’,’roll’] },
      'name=s'     => undef,
      'quiet!'         => undef,
      'go!'              => undef,
      'out=s@'     => 'directories',
      '<>'              => 'files'
  );
```

```
$ myprogram dir1/dirX/file
```

In the Perl program "myprogram":

```
use Getopt::Complete (
  type => [’names’,’places’,’things’],

  instance => sub {
    my ($command, $value, $option, $other_opts) = @_;

    if ($other_opts{type} eq ’names’) {
      return [qw/larry moe curly/],
    }
    elsif ($other_opts{type} eq ’places’) {
      return [qw/here there everywhere/],
    }
    elsif ($other_opts{type} eq ’things’) {
      return [ query_database_matching("${value}%") ]
    }
    else {
      # invalid type: no matches
      return []
    }
  },

);
```

In the Perl program "myprogram":

```
use Getopt::Complete (
  type => [’names’,’places’,’things’],

  instance => sub {
    my ($command, $value, $option, $other_opts) = @_;

    if ($other_opts{type} eq ’names’) {
      return [qw/larry moe curly/],
    }
    elsif ($other_opts{type} eq ’places’) {
      return [qw/here there everywhere/],
    }
    elsif ($other_opts{type} eq ’things’) {
      return [ query_database_matching("${value}%") ]
    }
    else {
      # invalid type: no matches
      return []
    }
  },

);
```

In the shell:

```
$ myprogram –type <TAB>
names people places
```

In the Perl program "myprogram":

```
use Getopt::Complete (
  type => [’names’,’places’,’things’],

  instance => sub {
    my ($command, $value, $option, $other_opts) = @_;

    if ($other_opts{type} eq ’names’) {
      return [qw/larry moe curly/],
    }
    elsif ($other_opts{type} eq ’places’) {
      return [qw/here there everywhere/],
    }
    elsif ($other_opts{type} eq ’things’) {
      return [ query_database_matching("${value}%") ]
    }
    else {
      # invalid type: no matches
      return []
    }
  },

);
```

In the shell:

```
$ myprogram –type places
```

In the Perl program "myprogram":

```
use Getopt::Complete (
  type => [’names’,’places’,’things’],

  instance => sub {
    my ($command, $value, $option, $other_opts) = @_;

    if ($other_opts{type} eq ’names’) {
      return [qw/larry moe curly/],
    }
    elsif ($other_opts{type} eq ’places’) {
      return [qw/here there everywhere/],
    }
    elsif ($other_opts{type} eq ’things’) {
      return [ query_database_matching("${value}%") ]
    }
    else {
      # invalid type: no matches
      return []
    }
  },

);
```

In the shell:

```
$ myprogram –type places –instance <TAB>
everywhere here there
```

In the Perl program "myprogram":

```
use Getopt::Complete (
  type => [’names’,’places’,’things’],

  instance => sub {
    my ($command, $value, $option, $other_opts) = @_;

    if ($other_opts{type} eq ’names’) {
      return [qw/larry moe curly/],
    }
    elsif ($other_opts{type} eq ’places’) {
      return [qw/here there everywhere/],
    }
    elsif ($other_opts{type} eq ’things’) {
      return [ query_database_matching("${value}%") ]
    }
    else {
      # invalid type: no matches
      return []
    }
  },

);
```

In the shell:

```
$ myprogram –type people
```

In the Perl program "myprogram":

```
use Getopt::Complete (
  type => [’names’,’places’,’things’],

  instance => sub {
    my ($command, $value, $option, $other_opts) = @_;

    if ($other_opts{type} eq ’names’) {
      return [qw/larry moe curly/],
    }
    elsif ($other_opts{type} eq ’places’) {
      return [qw/here there everywhere/],
    }
    elsif ($other_opts{type} eq ’things’) {
      return [ query_database_matching("${value}%") ]
    }
    else {
      # invalid type: no matches
      return []
    }
  },

);
```

In the shell:

```
$ myprogram –type people –instance <TAB>
curly larry moe
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    '>dog' => [
        '>bark' => [
            'ferocity'  => ['yip','wail','ruf','grrrr'],
            'count'  => ['1','2','one too many'],
        ],
        '>drool' => [
            'buckets=n' => undef,
            'lick'      => 'users',
        ],
        'list!' => undef,
    ],
    '>cat' => [
        '>purr' => [],
        '>meow' => [
            'volume=n' => undef,
            'bass' => ['low','medium','high'],
        ]
    ],
);
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    '>dog' => [
        '>bark' => [
            'ferocity'  => ['yip','wail','ruf','grrrr'],
            'count'  => ['1','2','one too many'],
        ],
        '>drool' => [
            'buckets=n' => undef,
            'lick'      => 'users',
        ],
        'list!' => undef,
    ],
    '>cat' => [
        '>purr' => [],
        '>meow' => [
            'volume=n' => undef,
            'bass' => ['low','medium','high'],
        ]
    ],
);
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    '>dog' => [
        '>bark' => [
            'ferocity'  => ['yip','wail','ruf','grrrr'],
            'count'  => ['1','2','one too many'],
        ],
        '>drool' => [
            'buckets=n' => undef,
            'lick'      => 'users',
        ],
        'list!' => undef,
    ],
    '>cat' => [
        '>purr' => [],
        '>meow' => [
            'volume=n' => undef,
            'bass' => ['low','medium','high'],
        ]
    ],
);
```

In the shell:

In the Perl program "myprogram":

```
use Getopt::Complete (
    '>dog' => [
        '>bark' => [
            'ferocity'  => ['yip','wail','ruf','grrrr'],
            'count'  => ['1','2','one too many'],
        ],
        '>drool' => [
            'buckets=n' => undef,
            'lick'      => 'users',
        ],
        'list!' => undef,
    ],
    '>cat' => [
        '>purr' => [],
        '>meow' => [
            'volume=n' => undef,
            'bass' => ['low','medium','high'],
        ]
    ],
);
```

In the shell:

```
$ myprogram <TAB>
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    '>dog' => [
        '>bark' => [
            'ferocity'  => ['yip','wail','ruf','grrrr'],
            'count'  => ['1','2','one too many'],
        ],
        '>drool' => [
            'buckets=n' => undef,
            'lick'      => 'users',
        ],
        'list!' => undef,
    ],
    '>cat' => [
        '>purr' => [],
        '>meow' => [
            'volume=n' => undef,
            'bass' => ['low','medium','high'],
        ]
    ],
);
```

In the shell:

```
$ myprogram
cat dog
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    '>dog' => [
        '>bark' => [
            'ferocity'  => ['yip','wail','ruf','grrrr'],
            'count'  => ['1','2','one too many'],
        ],
        '>drool' => [
            'buckets=n' => undef,
            'lick'      => 'users',
        ],
        'list!' => undef,
    ],
    '>cat' => [
        '>purr' => [],
        '>meow' => [
            'volume=n' => undef,
            'bass' => ['low','medium','high'],
        ]
    ],
);
```

In the shell:

```
$ myprogram dog
```

In the Perl program "myprogram":

```perl
use Getopt::Complete (
    '>dog' => [
        '>bark' => [
            'ferocity'  => ['yip','wail','ruf','grrrr'],
            'count'  => ['1','2','one too many'],
        ],
        '>drool' => [
            'buckets=n' => undef,
            'lick'      => 'users',
        ],
        'list!' => undef,
    ],
    '>cat' => [
        '>purr' => [],
        '>meow' => [
            'volume=n' => undef,
            'bass' => ['low','medium','high'],
        ]
    ],
);
```

In the shell:

```
 $ myprogram dog <TAB>
```

In the Perl program "myprogram":

```
   use Getopt::Complete (
       '>dog' => [
           '>bark' => [
               'ferocity'  => ['yip','wail','ruf','grrrr'],
               'count'  => ['1','2','one too many'],
           ],
           '>drool' => [
               'buckets=n' => undef,
               'lick'      => 'users',
           ],
           'list!' => undef,
       ],
       '>cat' => [
           '>purr' => [],
           '>meow' => [
               'volume=n' => undef,
               'bass' => ['low','medium','high'],
           ]
       ],
   );
```

In the shell:

```
 $ myprogram dog
 bark drool
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    '>dog' => [
        '>bark' => [
            'ferocity'  => ['yip','wail','ruf','grrrr'],
            'count'  => ['1','2','one too many'],
        ],
        '>drool' => [
            'buckets=n' => undef,
            'lick'      => 'users',
        ],
        'list!' => undef,
    ],
    '>cat' => [
        '>purr' => [],
        '>meow' => [
            'volume=n' => undef,
            'bass' => ['low','medium','high'],
        ]
    ],
);
```

In the shell:

```
 $ myprogram dog b<TAB>
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    '>dog' => [
        '>bark' => [
            'ferocity'  => ['yip','wail','ruf','grrrr'],
            'count'  => ['1','2','one too many'],
        ],
        '>drool' => [
            'buckets=n' => undef,
            'lick'      => 'users',
        ],
        'list!' => undef,
    ],
    '>cat' => [
        '>purr' => [],
        '>meow' => [
            'volume=n' => undef,
            'bass' => ['low','medium','high'],
        ]
    ],
);
```

In the shell:

```
$ myprogram dog bark
```

In the Perl program "myprogram":

```perl
use Getopt::Complete (
    '>dog' => [
        '>bark' => [
            'ferocity'  => ['yip','wail','ruf','grrrr'],
            'count'  => ['1','2','one too many'],
        ],
        '>drool' => [
            'buckets=n' => undef,
            'lick'      => 'users',
        ],
        'list!' => undef,
    ],
    '>cat' => [
        '>purr' => [],
        '>meow' => [
            'volume=n' => undef,
            'bass' => ['low','medium','high'],
        ]
    ],
);
```

In the shell:

```
 $ myprogram dog bark --<TAB>
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    '>dog' => [
        '>bark' => [
            'ferocity'  => ['yip','wail','ruf','grrrr'],
            'count'  => ['1','2','one too many'],
        ],
        '>drool' => [
            'buckets=n' => undef,
            'lick'      => 'users',
        ],
        'list!' => undef,
    ],
    '>cat' => [
        '>purr' => [],
        '>meow' => [
            'volume=n' => undef,
            'bass' => ['low','medium','high'],
        ]
    ],
);
```

In the shell:

```
 $ myprogram dog bark –
 --count --ferocity
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    '>dog' => [
        '>bark' => [
            'ferocity'  => ['yip','wail','ruf','grrrr'],
            'count'  => ['1','2','one too many'],
        ],
        '>drool' => [
            'buckets=n' => undef,
            'lick'      => 'users',
        ],
        'list!' => undef,
    ],
    '>cat' => [
        '>purr' => [],
        '>meow' => [
            'volume=n' => undef,
            'bass' => ['low','medium','high'],
        ]
    ],
);
```

In the shell:

```
 $ myprogram dog bark –count 1 --f<TAB>
```

In the Perl program "myprogram":

```
   use Getopt::Complete (
       '>dog' => [
           '>bark' => [
               'ferocity'  => ['yip','wail','ruf','grrrr'],
               'count'  => ['1','2','one too many'],
           ],
           '>drool' => [
               'buckets=n' => undef,
               'lick'      => 'users',
           ],
           'list!' => undef,
       ],
       '>cat' => [
           '>purr' => [],
           '>meow' => [
               'volume=n' => undef,
               'bass' => ['low','medium','high'],
           ]
       ],
   );
```

In the shell:

```
 $ myprogram dog bark –count 1 –ferocity <TAB>
```

In the Perl program "myprogram":

```
   use Getopt::Complete (
       '>dog' => [
           '>bark' => [
               'ferocity'  => ['yip','wail','ruf','grrrr'],
               'count'  => ['1','2','one too many'],
           ],
           '>drool' => [
               'buckets=n' => undef,
               'lick'      => 'users',
           ],
           'list!' => undef,
       ],
       '>cat' => [
           '>purr' => [],
           '>meow' => [
               'volume=n' => undef,
               'bass' => ['low','medium','high'],
           ]
       ],
   );
```

In the shell:

```
 $ myprogram dog bark –count 1 –ferocity
 grrr ruf wail yip
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    '>dog' => [
        '>bark' => [
            'ferocity'  => ['yip','wail','ruf','grrrr'],
            'count'  => ['1','2','one too many'],
        ],
        '>drool' => [
            'buckets=n' => undef,
            'lick'      => 'users',
        ],
        'list!' => undef,
    ],
    '>cat' => [
        '>purr' => [],
        '>meow' => [
            'volume=n' => undef,
            'bass' => ['low','medium','high'],
        ]
    ],
);
```

In the shell:

```
$ myprogram dog bark –count 1 –ferocity ruf
```

In the Perl program "myprogram":

```
use Getopt::Complete (
    '>dog' => [
        '>bark' => [
            'ferocity'  => ['yip','wail','ruf','grrrr'],
            'count'  => ['1','2','one too many'],
        ],
        '>drool' => [
            'buckets=n' => undef,
            'lick'      => 'users',
        ],
        'list!' => undef,
    ],
    '>cat' => [
        '>purr' => [],
        '>meow' => [
            'volume=n' => undef,
            'bass' => ['low','medium','high'],
        ]
    ],
);

$ARGS{'>'};
# [ 'dog', 'bark' ]

$ARGS{count}
# 1

$ARGS{ferocity}
# ruf
```

Getopt::Complete

tab-completion for Perl apps

Scott Smith

Genome Center
Washington University School of Medicine

(see David Dooling's "The Freedom to Cure Cancer" tomorrow at *10:45*)