```
# -v, -D, -o ARG, sets $args{v}, $args{D}, $args{o}
getopts("vDo:", \%args);
```

Or the standard Getopt::Long module to permit named arguments:

```
use Getopt::Long;
GetOptions( "verbose"  => \$verbose,        # --verbose
            "Debug"    => \$debug,          # --Debug
            "output=s" => \$output );
        # --output=somestring or --output somestring
```

Another reason for preprocessing arguments is to make an empty argument list default to all files:

```
@ARGV = glob("*") unless @ARGV;
```

You could even filter out all but plain, text files. This is a bit silent, of course, and you might prefer to mention them on the way.

```
@ARGV = grep { -f && -T } @ARGV;
```

If you're using the **-n** or **-p** command-line options, you should put changes to @ARGV in a `BEGIN{}` block.

Remember that a normal `open` has special properties, in that it might call fopen(3S) or it might called popen(3S), depending on what its argument looks like; that's why it's sometimes called "magic open". Here's an example:

```
$pwdinfo = ‘domainname‘ =~ /^(\(none\))?$/
                ? ’< /etc/passwd’
                : ’ypcat passwd |’;

open(PWD, $pwdinfo)
            or die "can’t open $pwdinfo: $!";
```

This sort of thing also comes into play in filter processing. Because <`ARGV`> processing employs the normal, shell-style Perl `open`, it respects all the special things we've already seen:

```
$ myprogram f1 "cmd1|" - f2 "cmd2|" f3 < tmpfile
```

That program will read from the file *f1*, the process *cmd1*, standard input (*tmpfile* in this case), the *f2* file, the *cmd2* command, and finally the *f3* file.

Yes, this also means that if you have files named "-" (and so on) in your directory, they won't be processed as literal files by `open`. You'll need to pass them as "./-", much as you would for the *rm* program, or you could use `sysopen` as described below.

One of the more interesting applications is to change files of a certain name into pipes. For example, to autoprocess gzipped or compressed files by decompressing them with *gzip*:

```
@ARGV = map { /^\.(gz|Z)$/ ? "gzip -dc $_ |" : $_  } @ARGV;
```

Or, if you have the *GET* program installed from LWP, you can fetch URLs before processing them:

```
@ARGV = map { m#^\w+://# ? "GET $_ |" : $_ } @ARGV;
```

It's not for nothing that this is called magic <`ARGV`>. Pretty nifty, eh?