```
 1: #!/usr/bin/perl -w
 2: use strict;
 3: use warnings;
 4: my $RCSID = '$Id: argv.perl,v 1.1 2014-10-03 16:57:20-07 - - $';
 5: #
 6: # NAME
 7: #    $0 - print out command line arguments and other things
 8: #
 9: # SYNOPSIS
10: #    $0 [args]
11: #
12: # DESCRIPTION
13: #    Prints out command line arguments in debug mode, along
14: #    with the RCSID, date and time.
15: #
16:
17: use POSIX qw(strftime);
18:
19: print "\$0 = $0\n";
20: $0 =~ s|^(.*/)?([^/]+)/*$|$2|;
21: print "\$0 = $0\n";
22: print "\$RCSID = $RCSID\n";
23:
24: printf "\$^T = $^T = %s\n",
25:        strftime "%Y-%m-%d %a %H:%M:%S %Z", localtime $^T;
26: map {print "\$ARGV[$_] = $ARGV[$_]\n"} 0..$#ARGV;
27:
```

```
 1: #!/usr/bin/perl -w
 2: use strict;
 3: use warnings;
 4: my $RCSID = '$Id: cat.perl,v 1.1 2014-10-03 16:57:20-07 - - $';
 5:
 6: $0 =~ s|^(.*/)?([^/]+)/*$|$2|;
 7: my $EXITCODE = 0;
 8: END { exit $EXITCODE; }
 9: sub note(@) { print STDERR "$0: @_"; };
10: $SIG{'__WARN__'} = sub { note @_; $EXITCODE = 1; };
11: $SIG{'__DIE__'} = sub { warn @_; exit; };
12:
13: (my $USAGE = <<__END_USAGE__) =~ s/^#[ ]?//gm;
14: #
15: # NAME
16: #    $0 - $0 concatenate and display files
17: #
18: # SYNOPSIS
19: #    $0 [-chnst] [file...]
20: #
21: # DESCRIPTION
22: #    Displays each file in sequence.  The filename `-'
23: #    causes STDIN to be read
24: #
25: # OPTIONS
26: #    -c  comment lines beginning with `#' are ignored
27: #    -h  displays help man page
28: #    -n  each line of output is numbered
29: #    -s  sequences of empty lines are suppressed
30: #    -t  filenames are printed ahead of files
31: #
32: # $RCSID
33: __END_USAGE__
34:
35: use POSIX qw(locale_h);
36: setlocale LC_CTYPE, "iso_8859_1";
37:
38: use Getopt::Std;
39: my %OPTIONS;
40: getopts ("chnstv", \%OPTIONS);
41: print $USAGE and exit if $OPTIONS{'h'};
42:
43: my $eqline = ":" x 64 . "\n";
44: push @ARGV, "-" unless @ARGV;
45: for my $filename (@ARGV) {
46:    open my $infile, "<$filename"
47:            or warn "<$filename: $!\n" and next;
48:    print "\n$eqline$filename\n$eqline\n" if $OPTIONS{'t'};
49:    my $lastempty = 0;
50:    my $thisempty;
51:    while (defined (my $line = <$infile>)) {
52:       chomp $line;
53:       next if $OPTIONS{'c'} and $line =~ m/^\s*#/;
54:       $thisempty = $line =~ m/^\s*$/;
55:       next if $OPTIONS{'s'} and $lastempty and $thisempty;
56:       printf "%6d  ", $. if $OPTIONS{'n'};
57:       printf "%s\n", $line;
58:    }continue {
```

```
59:       $lastempty = $thisempty;
60:    };
61:    close $infile;
62: };
63:
```

```
 1: #!/usr/bin/perl
 2: # $Id: config_sig_name.perl,v 1.1 2014-10-03 16:57:20-07 - - $
 3:
 4: use strict;
 5: use warnings;
 6: use Config;
 7: defined $Config{sig_name} or die "No Config{sig_name}!\n";
 8:
 9: my @sig_names = map {"SIG$_"} split ' ', $Config{sig_name};
10:
11: print "SIG[$_]=$sig_names[$_]\n" for 0..$#sig_names;
```

```
 1: #!/usr/bin/perl
 2: # $Id: dumper.perl,v 1.1 2014-10-03 16:57:20-07 - - $
 3: # Example of using Data::Dumper;
 4:
 5: use Data::Dumper;
 6:
 7: $a = [1,2,3,4,5];
 8: $b = {qw (A B C D)};
 9: @c = (1,2,3);
10: $t = {LEAF=> 'a'};
11: $u = {LEAF=> 'b'};
12: $v = {OPER=> '+', LEFT=> $t, RIGHT=> $u};
13: $w = {OPER=> '*', LEFT=> 'x', RIGHT=> $v};
14: @p = (\$a, \$b, \@c, \$w);
15: print Dumper @p;
16: print "$t, $u, $v\n";
```

```
 1: #!/usr/bin/perl -w
 2: use strict;
 3: use warnings;
 4: # $Id: env.perl,v 1.1 2014-10-03 16:57:20-07 - - $
 5: #
 6: # NAME
 7: #    env.perl - print out process environment variables
 8: #
 9:
10: for my $var (sort keys %ENV) {
11:    print "$var => $ENV{$var}\n";
12: };
```

```
 1: #!/usr/bin/perl -w
 2: # $Id: eratosthenes.perl,v 1.1 2014-10-03 16:57:20-07 - - $
 3:
 4: BEGIN{
 5:    push @INC, "/afs/cats.ucsc.edu/courses/cmps112-wm/Languages/perl/"
 6:             . "Lingua-Romana/Lingua-Romana-Perligata-0.50/lib";
 7: };
 8:
 9: use Lingua::Romana::Perligata
10: # 'converte',
11: # 'discribe'
12: # 'investiga',
13: ;
14:
15: maximum tum val inquementum tum biguttam tum stadium egresso scribe.
16:
17: vestibulo perlegementum da meo maximo.
18:
19: maximum tum novumversum egresso scribe.
20:
21: da II tum maximum conscribementa meis listis.
22:
23: dum damentum nexto listis decapitamentum
24:     fac sic
25:
26:         lista sic
27:             hoc tum nextum recidementum
28:         cis
29:         vannementa da listis.
30:
31:         dictum sic
32:             deinde
33:         cis
34:         tum biguttam tum stadium tum cum nextum comementum tum
35:         novumversum scribe egresso.
36:
37:     cis.
38:
```

```
 1: #!/usr/bin/perl -w
 2: use strict;
 3: use warnings;
 4: my $RCSID = '$Id: errno.perl,v 1.1 2014-10-03 16:57:20-07 - - $';
 5: #
 6: # NAME
 7: #    errno.perl - print out system error codes
 8: #
 9: # SYNOPSIS
10: #    errno.perl [errno...]
11: #
12: # DESCRIPTION
13: #    Prints out the system error codes given on the command line.
14: #    Prints all of them if none.
15:
16: if (@ARGV) {
17:    for my $errno (@ARGV) {
18:       if ($errno !~ m/^\d+$/) {
19:          print STDERR "$0: $errno: not a number\n";
20:       }else {
21:          $! = $errno;
22:          print "error($errno) = $!\n";
23:       };
24:    };
25: }else {
26:    for (my $errno = 0; ; ++$errno) {
27:       $! = $errno;
28:       my $strerror = "$!";
29:       last if $strerror eq $errno;
30:       print "error($errno) = $!\n";
31:    };
32: };
```

```
 1: #!/usr/bin/perl -w
 2: # $Id: fixreadme.perl,v 1.1 2014-10-03 16:57:20-07 - - $
 3: #
 4: # SYNOPSIS
 5: #    fixreadme.perl
 6: #
 7: # DESCRIPTION
 8: #    Edits the given file, looking for strings matching /http:lab\d+/
 9: #    and replacing them with actual directory names beginning with
10: #    /lab*/.  In addition, creates a file called `editreadme.ed'
11: #    which can be used to edit the other README file.
12: #
13:
14: $0 =~ s|.*/||;
15: $readfile = "README-all-cmps012m.text";
16: $edfile = "editreadme.ed";
17:
18: open FILE, "<$readfile" or die "$0: $readfile: $!";
19: @readfile = <FILE>;
20: close $readfile;
21: print `cid + $readfile`;
22:
23: @labs = glob "lab[0-9]*";
24: for $lab( @labs ){
25:    @base = $lab =~ m/^(lab\d+)/;
26:    map{ s|http:$base[0]\S*|http:$lab| } @readfile;
27: };
28:
29: open FILE, ">$readfile" or die "$0: $readfile: $!";
30: print FILE @readfile;
31: close $readfile;
32: print `cid + $readfile`;
33:
34: $pattern = "^\..*cmps012m.* --- http:";
35: @greplines = grep{ m|$pattern| } @readfile;
36: ($pwdlast = $ENV{PWD}) =~ s|.*/||;
37: map{ s|http:|$&$pwdlast/| } @greplines;
38:
39: open ED, ">$edfile" or die "$0: $edfile: $!";
40: print ED "g/$pattern/d\n",
41:          "a\n",
42:          @greplines,
43:          ".\n",
44:          "w\n",
45:          "q\n";
46: close ED;
47:
```

```
 1: #!/usr/bin/perl -w
 2: use strict;
 3: use warnings;
 4: my $RCSID = '$Id: getopts.perl,v 1.1 2014-10-03 16:57:20-07 - - $';
 5:
 6: $0 =~ s|^(.*/)?([^/]+)/*$|$2|;
 7: my $EXITCODE = 0;
 8: END{ exit $EXITCODE; }
 9: sub note(@) { print STDERR "$0: @_"; };
10: $SIG{'__WARN__'} = sub { note @_; $EXITCODE = 1; };
11: $SIG{'__DIE__'} = sub { warn @_; exit; };
12:
13: (my $USAGE = <<__END_USAGE__) =~ s/^#[ ]?//gm;
14: #
15: # NAME
16: #    $0 - getopts example
17: #
18: # SYNOPSIS
19: #    $0 [-abcopq] [file...]
20: #
21: # DESCRIPTION
22: #    Illustrates the use of getopts.
23: #
24: # OPTIONS
25: #    -h    print help and exit
26: #    -abc  flags not requiring options
27: #    -opq  flags requiring arguments
28: #
29: # $RCSID
30: __END_USAGE__
31:
32: use Getopt::Std;
33: my %OPTS;
34: getopts ("abcho:p:q:", \%OPTS);
35: print $USAGE and exit if $OPTS{'h'};
36:
37: print "$0: -$_ = $OPTS{$_}\n" for sort keys %OPTS;
38: print "$0: ARGV[$_]=$ARGV[$_]\n" for 0 .. $#ARGV;
39:
40:
```

```
 1: #!/usr/bin/perl -w
 2: use strict;
 3: use warnings;
 4: my $RCSID = '$Id: hello.perl,v 1.1 2014-10-03 16:57:20-07 - - $';
 5: #
 6: # NAME
 7: #    hello - hello world program
 8: #
 9: # SYNOPSIS
10: #    hello
11: #
12: # DESCRIPTION
13: #    Prints either message ``Hello, World!''.
14: #
15:
16: print "Hello, world!\n";
17:
```

```
 1: #!/bin/perl -w
 2: use strict;
 3: use warnings;
 4: my $RCSID = '$Id: insult.perl,v 1.1 2014-10-03 16:57:20-07 - - $';
 5: #
 6: # http://www.google.com/search?q=shakespearean+insulter
 7: #
 8:
 9: my @Insults = (
10: [qw(
11:
12: artless bawdy beslubbering bootless churlish cockered clouted
13: craven currish dankish dissembling droning errant fawning
14: fobbing froward frothy gleeking goatish gorbellied impertinent
15: infectious jarring loggerheaded lumpish mammering mangled
16: mewling paunchy pribbling puking puny quailing rank reeky
17: roguish ruttish saucy spleeny spongy surly tottering unmuzzled
18: vain venomed villainous warped wayward weedy yeasty
19:
20: )],
21: [qw(
22:
23: base-court bat-fowling beef-witted beetle-headed boil-brained
24: clapper-clawed clay-brained common-kissing crook-pated
25: dismal-dreaming dizzy-eyed doghearted dread-bolted earth-vexing
26: elf-skinned fat-kidneyed fen-sucked flap-mouthed fly-bitten
27: folly-fallen fool-born full-gorged guts-griping half-faced
28: hasty-witted hedge-born hell-hated idle-headed ill-breeding
29: ill-nurtured knotty-pated milk-livered motley-minded onion-eyed
30: plume-plucked pottle-deep pox-marked reeling-ripe rough-hewn
31: rude-growing rump-fed shard-borne sheep-biting spur-galled
32: swag-bellied tardy-gaited tickle-brained toad-spotted
33: urchin-snouted weather-bitten
34:
35: )],
36: [qw(
37:
38: apple-john baggage barnacle bladder boar-pig bugbear bum-bailey
39: canker-blossom clack-dish clotpole coxcomb codpiece death-token
40: dewberry flap-dragon flax-wench flirt-gill foot-licker
41: fustilarian giglet gudgeon haggard harpy hedge-pig horn-beast
42: hugger-mugger jolthead lewdster loat maggot-pie malt-worm mammet
43: measle minnow miscreant moldwarp mumble-news nut-hook pigeon-egg
44: pignut puttock pumpion ratsbane scut skainsmate strumpet varlet
45: vassal whey-face wagtail
46:
47: )]
48: );
49:
50: print "Thou @{[map { $$_[int rand scalar @$_] } @Insults]}.\n"
51:       for 1 .. (@ARGV && $ARGV[0] =~ m/^\d+$/ ? $ARGV[0] : 1);
```

```
 1: #!/usr/bin/perl
 2: # $Id: isatty.perl,v 1.1 2014-10-03 16:57:20-07 - - $
 3:
 4: use POSIX qw (isatty);
 5:
 6: sub printty ($$) {
 7:    my ($handle, $bool) = @_;
 8:    print "$handle is", ($bool ? "" : " not"), " a tty\n";
 9: }
10:
11: printty "STDIN", -t STDIN;
12: printty "STDOUT", -t STDOUT;
13: printty "STDERR", -t STDERR;
```

```
1: #!/usr/bin/perl -pl
2: # $Id: isprime.perl,v 1.1 2014-10-03 16:57:20-07 - - $
3: s|$| is @{['not 'x(1x$_)=~/^(11+)\1+$/]}prime|
```

```
 1: #!/usr/bin/perl -w
 2: use strict;
 3: use warnings;
 4: my $RCSID = '$Id: loaddict.perl,v 1.1 2014-10-03 16:57:20-07 - - $';
 5: #
 6: # Load a dictionary into a hash and then print the hash.
 7: #
 8:
 9: $0 =~ s|^(.*/)?([^/]+)/*$|$2|;
10: my $EXITCODE = 0;
11: END{ exit $EXITCODE; }
12: sub note(@){ print STDERR "$0: @_"; };
13: $SIG{'__WARN__'} = sub{ note @_; $EXITCODE = 1; };
14: $SIG{'__DIE__'} = sub{ warn @_; exit; };
15:
16: sub loaddict (\%$) {
17:    my ($hashref, $filename) = @_;
18:    open my $dict, $filename or warn "$filename: $!\n" and return;
19:    for my $word (<$dict>) {
20:       chomp $word;
21:       $hashref->{$word} = 1;
22:    };
23:    close $dict;
24: };
25:
26: push @ARGV, "/usr/dict/words" unless @ARGV;
27:
28: for my $filename (@ARGV) {
29:    my %hash;
30:    loaddict %hash, $filename;
31:    for my $word (sort keys %hash) {
32:       print "$filename: $word\n";
33:    };
34: };
35:
```

```
 1: #!/usr/bin/perl -w
 2: use strict;
 3: use warnings;
 4: my $RCSID = '$Id: merriam.perl,v 1.1 2014-10-03 16:57:20-07 - - $';
 5:
 6: $0 =~ s|^(.*/)?([^/]+)/*$|$2|;
 7: my $EXITCODE = 0;
 8: END{ exit $EXITCODE; }
 9: sub note(@){ print STDERR "$0: @_"; };
10: $SIG{'__WARN__'} = sub{ note @_; $EXITCODE = 1; };
11: $SIG{'__DIE__'} = sub{ warn @_; exit; };
12:
13: (my $USAGE = <<__END_USAGE__) =~ s/^#[ ]?//mg;
14: #
15: # NAME
16: #    $0 - look up words in the Merriam-Webster online dictionary.
17: #
18: # SYNOPSIS
19: #    $0 words...
20: #
21: # DESCRIPTION
22: #    Each word specified in @ARGV is looked up in Merriam-Webster's
23: #    online dictionary, with the results displayed to STDOUT.
24: #
25: # BUGS
26: #    This is a hack into HTML, so if the format of the pages served
27: #    changes, this script will no longer work.
28: #
29: # RCSID
30: #    $RCSID
31: #
32: __END_USAGE__
33:
34: print STDERR $USAGE and exit unless @ARGV;
35:
36: my $MWURL = "http://www.m-w.com/cgi-bin/dictionary?book=Dictionary&va=";
37: my $fmtpipe = "| fmt -s";
38: open FMTPIPE, $fmtpipe or die "$fmtpipe: $!";
39:
40: for my $word( @ARGV ){
41:    my $codeword = $word;
42:    $codeword =~ s/\s+/+/g;
43:    my $cmd = "lynx -source '$MWURL'$codeword\n";
44:    my $page = `$cmd`;
45:    $page =~ s|\r||sig;
46:    $page =~ s|.*<!-- begin content -->(.*)<!-- end content -->.*|$1|si;
47:    $page =~ s|((.*?</table>){2}).*|$1|si;
48:    $page =~ s| | |sig;
49:    $page =~ s|<br>\s*|\n|sig;
50:    $page =~ s|<[^>]*>||sig;
51:    $page =~ s|^\s*|\n|si;
52:    $page =~ s|\s*$|\n|si;
53:    $page =~ s|\s*\n\s*\n+|\n\n|sig;
54:    $page =~ s|&lt;|<|sig;
55:    $page =~ s|&gt;|>|sig;
56:    $page =~ s|&amp;|&|sig;
57:
58:    print FMTPIPE $page;
```

```
59: };
60:
61: close FMTPIPE;
```

```
 1: #!/usr/bin/perl
 2: use strict;
 3: use warnings;
 4: my $RCSID = '$Id: ncat.perl,v 1.1 2014-10-03 16:57:20-07 - - $';
 5:
 6: $0 =~ s|^(.*/)?([^/]+)/*$|$2|;
 7: my $exit_status = 0;
 8: END { exit $exit_status; }
 9: sub note (@) { print STDERR "$0: @_"; };
10: $SIG{'__WARN__'} = sub { note @_; $exit_status = 1; };
11: $SIG{'__DIE__'} = sub { warn @_; exit; };
12:
13:
14: while (<>) {
15:    next if m/^\s*#/;
16:    print "$ARGV:$.:$_";
17: }continue {
18:    close ARGV if eof;
19: };
20:
```

```
 1: #!/usr/bin/perl
 2: # $Id: nvcat.perl,v 1.5 2012-01-13 20:18:20-08 - - $
 3: #
 4: # NAME
 5: #    nvcat - cat files with filenames and line numbers
 6: #
 7: # SYNOPSIS
 8: #    nvcat [filename...]
 9: #
10: # DESCRIPTION
11: #    Display all files given by the list of filenames, or STDIN,
12: #    if none.  Display filenames and line numbers.
13: #
14:
15: use strict;
16: use warnings;
17:
18: $0 =~ s|^(.*/)?([^/]+)/*$|$2|;
19: my $EXIT_STATUS = 0;
20: END {exit $EXIT_STATUS}
21: sub note(@) {print STDERR "$0: @_"}
22: $SIG{'__WARN__'} = sub {note @_; $EXIT_STATUS = 1};
23: $SIG{'__DIE__'} = sub {warn @_; exit};
24:
25: my $eqline = ":" x 32 . "\n";
26: push @ARGV, "-" unless @ARGV;
27: for my $filename (@ARGV) {
28:    open my $infile, "<$filename" or warn "<$filename: $!\n" and next;
29:    print "\n$eqline$filename\n$eqline";
30:    while (defined (my $line = <$infile>)) {
31:       chomp $line;
32:       printf "%6d  %s\n", $., $line;
33:    }
34:    close $infile;
35: }
36:
```

```
 1: #!/usr/bin/perl
 2: # $Id: pgrep.perl,v 1.1 2014-10-03 16:57:20-07 - - $
 3: use strict;
 4: use warnings;
 5: use Getopt::Std;
 6:
 7: $0 =~ s|^.*?/?([^/]+)/*$|$1|;
 8: my $exit_status = 0;
 9: sub note(@) {print STDERR "$0: @_"}
10: $SIG{'__WARN__'} = sub {note @_; $exit_status = 2};
11: $SIG{'__DIE__'} = sub {warn @_; exit};
12:
13: sub usage_exit {
14:    print STDERR "Usage: $0 [-ilnv] regex [filename...]\n";
15:    $exit_status = 2;
16:    exit;
17: }
18:
19: my %opts;
20: getopts ("ilnv", \%opts);
21: my $regex = shift or usage_exit;
22: my $anymatch = 0;
23: $regex = eval ('$opts{"i"} ? qr($regex)i : qr($regex)');
24: die $@ if $@;
25: print "regex=$regex;\n";
26: push @ARGV, "-" unless @ARGV;
27:
28: for my $filename (@ARGV) {
29:    open my $file, "<$filename" or warn "$filename: $!\n" and next;
30:    my $filematch = 0;
31:    while (defined (my $line = <$file>)) {
32:       if (my $match = $opts{"v"} ? $line !~ $regex : $line =~ $regex) {
33:          $anymatch = $filematch = 1;
34:          unless ($opts{"l"}) {
35:             printf "%s:", $filename if @ARGV > 1;
36:             printf "%s:", $. if $opts{"n"};
37:             print $line;
38:          }
39:       }
40:    }
41:    print "$filename\n" if $filematch and $opts{"l"};
42:    close $file;
43: }
44:
45: $exit_status ||= ! $anymatch;
46: exit $exit_status;
47:
```

```
 1: #!/usr/bin/perl
 2: # $Id: printstatus.perl,v 1.1 2014-10-03 16:57:20-07 - - $
 3: #
 4: # Copy code from this file in order to print an exit status.
 5: #
 6: use strict;
 7: use warnings;
 8:
 9: my %status_strings = (
10:     1=> "Hangup",
11:     2=> "Interrupt",
12:     3=> "Quit",
13:     4=> "Illegal Instruction",
14:     5=> "Trace/Breakpoint Trap",
15:     6=> "Abort",
16:     7=> "Emulation Trap",
17:     8=> "Arithmetic Exception",
18:     9=> "Killed",
19:    10=> "Bus Error",
20:    11=> "Segmentation Fault",
21:    12=> "Bad System Call",
22:    13=> "Broken Pipe",
23:    14=> "Alarm Clock",
24:    15=> "Terminated",
25:    16=> "User Signal 1",
26:    17=> "User Signal 2",
27:    18=> "Child Status Changed",
28:    19=> "Power-Fail/Restart",
29:    20=> "Window Size Change",
30:    21=> "Urgent Socket Condition",
31:    22=> "Pollable Event",
32:    23=> "Stopped (signal)",
33:    24=> "Stopped (user)",
34:    25=> "Continued",
35:    26=> "Stopped (tty input)",
36:    27=> "Stopped (tty output)",
37:    28=> "Virtual Timer Expired",
38:    29=> "Profiling Timer Expired",
39:    30=> "Cpu Limit Exceeded",
40:    31=> "File Size Limit Exceeded",
41:    32=> "No runnable lwp",
42:    33=> "Inter-lwp signal",
43:    34=> "Checkpoint Freeze",
44:    35=> "Checkpoint Thaw",
45:    36=> "Thread Cancellation",
46:    37=> "Resource Lost",
47:    38=> "First Realtime Signal",
48:    39=> "Second Realtime Signal",
49:    40=> "Third Realtime Signal",
50:    41=> "Fourth Realtime Signal",
51:    42=> "Fourth Last Realtime Signal",
52:    43=> "Third Last Realtime Signal",
53:    44=> "Second Last Realtime Signal",
54:    45=> "Last Realtime Signal",
55: );
56:
57: #
58: # See man -s 2 wait for an explanation.
```

```
59: #
60: sub status_string ($) {
61:    my ($status) = @_;
62:    return undef unless $status;
63:    printf "0x%08X\n", $status;
64:    print $status & 0xFF, "\n";
65:    return sprintf "Error %d", $status >> 8 if ($status & 0xFF) == 0;
66:    my $message = $status_strings{$status & 0x7F}
67:                  || "Invalid Signal Number";
68:    $message .= " (core dumped)" if $status & 0x80;
69:    return $message;
70: }
71:
72: #
73: # What you need is the hash and the function.
74: # The following is just a dummy main function for testing.
75: #
76:
77: for my $code (@ARGV) {
78:    my $string = status_string $code;
79:    next unless $string;
80:    printf "status 0x%04X = %s\n", $code, $string;
81: }
```

```
1: #!/usr/bin/perl -p
2: # $Id: rot13.perl,v 1.1 2014-10-03 16:57:20-07 - - $
3: tr/a-zA-Z/n-za-mN-ZA-M/
```

```
 1: #!/usr/bin/perl -w
 2: # $Id: run.perl,v 1.1 2014-10-03 16:57:20-07 - - $
 3: #
 4: # NAME
 5: #    run.perl - run a Unix command and print exit status and signal
 6: #
 7: # SYNOPSIS
 8: #    run.perl command operands...
 9: #
10: # DESCRIPTION
11: #    Runs a command and prints the resulting exit status, etc.
12: #
13: # EXAMPLE
14: #    run.perl echo foo bar
15: #
16:
17: $0 =~ s{.*/}{};
18: $EXITCODE = 0;
19: sub setexit($){
20:    my( $newexit ) = @_;
21:    $newexit &= 0xFF;
22:    $EXITCODE = $newexit if $EXITCODE < $newexit;
23: };
24: $SIG{'__WARN__'} = sub{ print STDERR "$0: @_"; setexit 1; };
25: $SIG{'__DIE__'} = sub{ warn @_; exit; };
26: END{ exit $EXITCODE; }
27:
28: # Source: signal(3HEAD)
29: $SIG[ 1]="SIGHUP: Hangup (see termio(7I))";
30: $SIG[ 2]="SIGINT: Interrupt (see termio(7I))";
31: $SIG[ 3]="SIGQUIT: Quit (see termio(7I))";
32: $SIG[ 4]="SIGILL: Illegal Instruction";
33: $SIG[ 5]="SIGTRAP: Trace or Breakpoint Trap";
34: $SIG[ 6]="SIGABRT: Abort";
35: $SIG[ 7]="SIGEMT: Emulation Trap";
36: $SIG[ 8]="SIGFPE: Arithmetic Exception";
37: $SIG[ 9]="SIGKILL: Killed";
38: $SIG[10]="SIGBUS: Bus Error";
39: $SIG[11]="SIGSEGV: Segmentation Fault";
40: $SIG[12]="SIGSYS: Bad System Call";
41: $SIG[13]="SIGPIPE: Broken Pipe";
42: $SIG[14]="SIGALRM: Alarm Clock";
43: $SIG[15]="SIGTERM: Terminated";
44: $SIG[16]="SIGUSR1: User Signal 1";
45: $SIG[17]="SIGUSR2: User Signal 2";
46: $SIG[18]="SIGCHLD: Child Status Changed";
47: $SIG[19]="SIGPWR: Power Fail or Restart";
48: $SIG[20]="SIGWINCH: Window Size Change";
49: $SIG[21]="SIGURG: Urgent Socket Condition";
50: $SIG[22]="SIGPOLL: Pollable Event (see streamio(7I))";
51: $SIG[23]="SIGSTOP: Stopped (signal)";
52: $SIG[24]="SIGTSTP: Stopped (user) (see termio(7I))";
53: $SIG[25]="SIGCONT: Continued";
54: $SIG[26]="SIGTTIN: Stopped (tty input) (see termio(7I))";
55: $SIG[27]="SIGTTOU: Stopped (tty output) (see termio(7I))";
56: $SIG[28]="SIGVTALRM: Virtual Timer Expired";
57: $SIG[29]="SIGPROF: Profiling Timer Expired";
58: $SIG[30]="SIGXCPU: CPU time limit exceeded (see getrlimit(2))";
```

```
59: $SIG[31]="SIGXFSZ: File size limit exceeded (see getrlimit(2))";
60: $SIG[32]="SIGWAITING: Concurrency signal reserved by threads library";
61: $SIG[33]="SIGLWP: Inter-LWP signal reserved by threads library";
62: $SIG[34]="SIGFREEZE: Check point Freeze";
63: $SIG[35]="SIGTHAW: Check point Thaw";
64: $SIG[36]="SIGCANCEL: Cancellation signal reserved by threads library";
65:
66: push @ARGV, '2>&1';
67: print `@ARGV`;
68:
69: $exit_value  = ($? >> 8) & 0xFF;
70: $signal_num  = $? & 0x7F;
71: $dumped_core = $? & 0x80;
72: $signal_msg  = $SIG[$signal_num];
73:
74: $message = "=> exit $exit_value, status $signal_num";
75: $message .= ", $signal_msg" if $signal_msg;
76: $message .= " (core dumped)" if $dumped_core;
77:
78: print "\n$0: @ARGV\n$message\n";
79:
80: $EXITCODE = $signal_num if $signal_num;
81: $EXITCODE = $exit_value if $exit_value;
82:
```

```
 1: #!/usr/bin/perl -00p
 2: # $Id: squeeze.perl,v 1.1 2014-10-03 16:57:20-07 - - $
 3: #
 4: # NAME
 5: #    squeeze - squeeze out multiple empty lines
 6: #
 7: # SYNOPSIS
 8: #    squeeze [filename...]
 9: #
10: # DESCRIPTION
11: #    Read either from the list of filenames given, or stdin.
12: #    Copy input to output, and eliminate blank lines.  Note
13: #    that this program consists entirely of comments and all
14: #    of the program logic is in the #! line.
15: #
16: #    -00     put perl into paragraph mode for input.
17: #    -p      echo input to output
18: #
19: # To make this into a csh alias, put the following line in
20: # (without the hash on the front) into your .cshrc:
21: # alias squeeze perl -00pe0
22: #
23: # Or run the following from the command line:
24: # % perl -00pe0 foo bar baz
25: #
26: # This latter will cat the three files specified.  Note that
27: # from the command line you need the option -e0 which is an inline
28: # perl program.  0 is just a place keeper which does nothing.
```

```
 1: #!/usr/bin/perl -w
 2: use strict;
 3: use warnings;
 4: my $RCSID = '$Id: stat.perl,v 1.1 2014-10-03 16:57:20-07 - - $';
 5:
 6: $0 =~ s|^(.*/)?([^/]+)/*$|$2|;
 7: my $EXITCODE = 0;
 8: END { exit $EXITCODE; }
 9: sub note(@) { print STDERR "$0: @_"; };
10: $SIG{'__WARN__'} = sub { note @_; $EXITCODE = 1; };
11: $SIG{'__DIE__'} = sub { warn @_; exit; };
12:
13: (my $USAGE = <<__END_USAGE__) =~ s/^#[ ]?//gm;
14: #
15: # NAME
16: #    $0 - $0 display file status
17: #
18: # SYNOPSIS
19: #    $0 [-l] filename...
20: #
21: # DESCRIPTION
22: #    Displays the status of each file specified in the arg list.
23: #
24: # OPTIONS
25: #    -l  Use lstat(2) instead of stat(2) to display information
26: #        about the symbolic link instead of the file itself.
27: #
28: # SEE ALSO
29: #    stat(2), lstat(2)
30: #
31: # $RCSID
32: __END_USAGE__
33:
34: use POSIX qw(locale_h strftime);
35: setlocale LC_CTYPE, "iso_8859_1";
36:
37: use Getopt::Std;
38: my %OPTIONS;
39: getopts ("l", \%OPTIONS);
40: print $USAGE and exit unless @ARGV;
41: my $statlink = $OPTIONS{'l'} ? " -l" : "";
42:
43: sub print_dev($$) {
44:    my ($dev, $label) = @_;
45:    printf "%6d,%4d = %s\n", $dev >> 8, $dev & 0xFF, $label;
46: };
47:
48: sub print_time($$) {
49:    my ($time, $label) = @_;
50:    printf "%11d = %s = %s\n", $time, $label,
51:           strftime "%C", localtime $time;
52: };
53:
54: for my $filename (@ARGV) {
55:    my @status = $statlink ? lstat $filename : stat $filename;
56:    warn "stat $filename: $!" and next unless @status;
57:    my ($dev, $ino, $mode, $nlink, $uid, $gid, $rdev, $size,
58:        $atime, $mtime, $ctime, $blksize, $blocks) = @status;
```

```
   print "$0 $statlink \"$filename\":\n";
   print_dev $dev, "dev: device inode resides on";
   printf "%11d = ino: file's inode number\n", $ino;
   print "###$mode\n";
   printf "%11d = nlink: number of hard links to file\n", $nlink;
   printf "%11d = uid: file's user id = %s\n", $uid, getpwuid $uid;
   printf "%11d = gid: file's group id = %s\n", $gid, getgrgid $gid;
   print_dev $rdev, "rdev: device if a special file";
   printf "%11d = size: file size in bytes\n", $size;
   print_time $atime, "atime: file last access";
   print_time $mtime, "mtime: file last modify";
   print_time $ctime, "ctime: file last change";
   printf "%11d = blksize: preferred I/O block size\n", $blksize;
   printf "%11d = blocks: number 512 byte blocks allocated\n", $blocks;
};

```

```
 1: #!/usr/bin/perl -w
 2: use strict;
 3: use warnings;
 4: my $RCSID = '$Id: text2html.perl,v 1.1 2014-10-03 16:57:20-07 - - $';
 5:
 6: $0 =~ s|^(.*/)?([^/]+)/*$|$2|;
 7: my $EXITCODE = 0;
 8: END{ exit $EXITCODE; }
 9: sub note(@){ print STDERR "$0: @_"; };
10: $SIG{'__WARN__'} = sub{ note @_; $EXITCODE = 1; };
11: $SIG{'__DIE__'} = sub{ warn @_; exit; };
12:
13: (my $USAGE = <<__END_USAGE__) =~ s/^#[ ]?//gm;
14: #
15: # NAME
16: #    $0 - convert text to html <PRE>
17: #
18: # SYNOPSIS
19: #    $0 [-h] [infile...]
20: #
21: # DESCRIPTION
22: #    Reads <> and writes STDOUT, converting the input from
23: #    text to <PRE> HTML.  Links are added for any sequence
24: #    matching /http:\\S+/.
25: #
26: # $RCSID
27: __END_USAGE__
28:
29: print $USAGE and exit if @ARGV and $ARGV[0] eq "-h";
30:
31: my %htmlchars = (
32:    '&'=> '&amp;',
33:    '<'=> '&lt;',
34:    '>'=> '&gt;',
35: );
36:
37: print "<PRE>\n";
38:
39: while (defined (my $line = <>)) {
40:    $line =~ s![&<>]!$htmlchars{$&}!g;
41:    $line =~ s!(^|\W)(http:\S+)!$1<A HREF=$2>$2</A>!g;
42:    print $line;
43: };
44:
```

```
 1: @a=(Lbzjoftt,Inqbujfodf,
 2: Hvcsjt); $b="Lbssz Wbmm"
 3: ;$b =~ y/b-z/a-z/ ; $c =
 4: " Tif ". @a ." hsfbu wj"
 5: ."suvft pg b qsphsbnnfs"
 6: . ":\n";$c =~y/b-y/a-z/;
 7: print"\n\n$c ";for($i=0;
 8: $i<@a; $i++) { $a[$i] =~
 9: y/b-y/a-z/;if($a[$i]eq$a
10: [-1]){print"and $a[$i]."
11: ;}else{ print"$a[$i], ";
12: }}print"\n\t\t--$b\n\n";
```

```
 1: #!/usr/bin/perl -w
 2: use strict;
 3: use warnings;
 4: my $RCSID = '$Id: wc.perl,v 1.1 2014-10-03 16:57:20-07 - - $';
 5:
 6: $0 =~ s|^(.*/)?([^/]+)/*$|$2|;
 7: my $EXITCODE = 0;
 8: END{ exit $EXITCODE; }
 9: sub note(@){ print STDERR "$0: @_"; };
10: $SIG{'__WARN__'} = sub{ note @_;; $EXITCODE = 1; };
11: $SIG{'__DIE__'} = sub{ warn @_; exit; };
12:
13: use Getopt::Std;
14: my %OPTIONS;
15: getopts ("cwl", \%OPTIONS);
16: %OPTIONS = qw(c 1 w 1 l 1) unless %OPTIONS;
17:
18: push @ARGV, "-" unless @ARGV;
19:
20: for my $filename (@ARGV) {
21:    open my $file, $filename or warn "$filename: $!\n" and next;
22:    my $linect = 0;
23:    my $wordct = 0;
24:    my $charct = 0;
25:    while (defined (my $line = <$file>)) {
26:       $linect += 1;
27:       my @words = $line =~ m/(\S+)/;
28:       $wordct += @words;
29:       $charct += length $line;
30:    };
31:    printf " %7d", $linect if $OPTIONS{'l'};
32:    printf " %7d", $wordct if $OPTIONS{'w'};
33:    printf " %7d", $charct if $OPTIONS{'c'};
34:    printf " %s\n", $filename;
35:    close $file;
36: };
37:
```

```
 1: #!/usr/bin/perl -w
 2: use strict;
 3: use warnings;
 4: my $RCSID = '$Id: wordfreq.perl,v 1.1 2014-10-03 16:57:20-07 - - $';
 5:
 6: $0 =~ s|^(.*/)?([^/]+)/*$|$2|;
 7: my $EXITCODE = 0;
 8: END{ exit $EXITCODE; }
 9: sub note(@){ print STDERR "$0: @_"; };
10: $SIG{'__WARN__'} = sub{ note @_; $EXITCODE = 1; };
11: $SIG{'__DIE__'} = sub{ warn @_; exit; };
12:
13: my $MAN_PAGE = <<__END_MAN_PAGE__;
14: #
15: # NAME
16: #    $0 - count the frequencies of words in a file
17: #
18: # SYNOPSIS
19: #    $0 [-Dh] [file...]
20: #
21: # DESCRIPTION
22: #    Reads words from the files given as arguments and prints them
23: #    out along with their frequencies.  If no arguments are given
24: #    reads STDIN.
25: #
26: # OPTIONS
27: #    -D  produces a debug dump (not implemented in Perl)
28: #    -h  prints this help message.
29: #
30: # $RCSID
31: __END_MAN_PAGE__
32:
33: use Getopt::Std;
34: my %OPTIONS;
35: getopts ("Dh", \%OPTIONS);
36: print $MAN_PAGE and exit if $OPTIONS{'h'};
37:
38: push @ARGV, "-" unless @ARGV;
39:
40: my %WORDFREQ;
41: while (defined (my $line = <>)) {
42:    map{ ++$WORDFREQ{$_} } $line =~ m/([[:alnum:]]+)/g;
43: };
44:
45: map { printf "%7d %s\n", $WORDFREQ{$_}, $_ } sort keys %WORDFREQ;
46:
```

```
1: #!/usr/bin/perl
2: # $Id: xref.perl,v 1.1 2014-10-03 16:57:20-07 - - $
3:
4: map { $hash{lc $_} .= " $." } m/(\w+)/g while <>;
5: map { print "$_$hash{$_}\n" } sort keys %hash;
6:
```

```
 1: #!/usr/bin/perl
 2: # $Id: yes.perl,v 1.1 2014-10-03 16:57:20-07 - - $
 3: #
 4: # NAME
 5: #    yes - be repetitively affirmative
 6: #
 7: # SYNOPSIS
 8: #    yes [expletive]
 9: #
10: # DESCRIPTION
11: #    yes repeatedly outputs y, or if expletive is given, that is
12: #    output repeatedly.  Termination is by typing an interrupt
13: #    character or breaking the pipe.
14: #
15:
16: my $expletive = "@ARGV" || "y";
17: print "$expletive\n" while 1;
```