```
 1: // $Id: hashfn.c,v 1.1 2012-11-16 18:04:00-08 - - $
 2:
 3: //
 4: // This program is not part of your project.  It exists just to
 5: // illustrate how to obtain and print hash values.  Each element
 6: // of argv is hashed and printed along with its hashcode.
 7: //
 8:
 9: #include <stdio.h>
10: #include <stdlib.h>
11:
12: #include "../code/strhash.h"
13:
14: int main (int argc, char **argv) {
15:    for (int argi = 0; argi < argc; ++argi) {
16:       char *str = argv[argi];
17:       hashcode_t hashcode = strhash (str);
18:       printf ("%10u = strhash (\"%s\")\n", hashcode, str);
19:    }
20:    printf ("%10u = 0xFFFFFFFFu\n", 0xFFFFFFFFu);
21:    return EXIT_SUCCESS;
22: }
23:
```

```
 1: // $Id: strhash.h,v 1.1 2012-11-16 18:05:22-08 - - $
 2:
 3: //
 4: // NAME
 5: //    strhash - return an unsigned 32-bit hash code for a string
 6: //
 7: // SYNOPSIS
 8: //    hashcode_t strhash (char *string);
 9: //
10: // DESCRIPTION
11: //    Uses Horner's method to compute the hash code of a string
12: //    as is done by java.lang.String.hashCode:
13: //    .  s[0]*31^(n-1) + s[1]*31^(n-2) + ... + s[n-1]
14: //    Using strength reduction, the multiplication is replaced by
15: //    a shift.  However, instead of returning a signed number,
16: //    this function returns an unsigned number.
17: //
18: // REFERENCE
19: //    http://java.sun.com/j2se/1.4.1/docs/api/java/lang/
20: //    String.html#hashCode()
21: //
22: //
23:
24: #ifndef __STRHASH_H__
25: #define __STRHASH_H__
26:
27: #include <inttypes.h>
28:
29: typedef uint32_t hashcode_t;
30:
31: hashcode_t strhash (char *string);
32:
33: #endif
34:
```

```
 1: // $Id: strhash.c,v 1.1 2012-11-16 18:05:22-08 - - $
 2:
 3: #include <assert.h>
 4: #include <stdio.h>
 5: #include <sys/types.h>
 6:
 7: #include "strhash.h"
 8:
 9: hashcode_t strhash (char *string) {
10:    assert (string != NULL);
11:    hashcode_t hashcode = 0;
12:    for (int index = 0; string[index] != '\0'; ++index) {
13:       hashcode = hashcode * 31 + (unsigned char) string[index];
14:    }
15:    return hashcode;
16: }
17:
```

```
 1: # $Id: Makefile,v 1.1 2012-11-16 18:04:00-08 - - $
 2:
 3: GCC       = gcc -g -O0 -Wall -Wextra -std=gnu99
 4: EXECBIN   = hashfn
 5: HASHSRC   = hashfn.c ../code/strhash.c
 6: LISFILES  = hashfn.c ../code/strhash.h ../code/strhash.c \
 7:             Makefile pspell.perl
 8: LISTING   = Listing.misc.ps
 9: HASHOUT   = hashfn.out
10:
11: TESTDATA  = 0 9 A Z a z foo bar baz qux \
12:             quux quuux quuuux quuuuux quuuuuux quuuuuuux quuuuuuuux
13:
14: all : ${EXECBIN}
15:
16: % : %.c
17:         - cid + $<
18:         - checksource $<
19:         ${GCC} -o $@ ${HASHSRC}
20:
21: ci : ${LISFILES}
22:         - checksource ${LISFILES}
23:         - cid + ${LISFILES}
24:
25: lis : ${LISFILES} ${HASHOUT}
26:         mkpspdf ${LISTING} ${LISFILES} ${HASHOUT}
27:
28: ${HASHOUT} : hashfn
29:         hashfn ${TESTDATA} * >${HASHOUT}
30:
31: spotless :
32:         - rm ${EXECBIN} ${HASHOUT}
33:
```

```
 1: #!/usr/bin/perl
 2: # $Id: pspell.perl,v 1.1 2012-11-16 18:04:57-08 - - $
 3: use strict;
 4: use warnings;
 5: use Getopt::Std;
 6:
 7: $0 =~ s|^(.*/)?([^/]+)/*$|$2|;
 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: END {exit $exit_status}
13:
14: my %options;
15: getopts "nd:", \%options;
16:
17: my %dictionary;
18: sub load_dictionary($) {
19:    my ($dictname) = @_;
20:    open my $dict, "<$dictname" or do {warn "$dictname: $!\n"; return};
21:    map {chomp; $dictionary{$_} = 1} <$dict>;
22:    close $dict;
23: }
24: load_dictionary "/usr/share/dict/words" unless $options{'n'};
25: load_dictionary $options{'d'} if defined $options{'d'};
26: die "dictionary is empty\n" unless %dictionary;
27:
28: my $numpat = qr{([[:digit:]]+([-:.][[:digit:]]+)*)};
29: my $wordpat = qr{([[:alnum:]]+([-&'.][[:alnum:]]+)*)};
30: for my $filename (@ARGV ? @ARGV : "-") {
31:    open my $file, "<$filename" or do {warn "$filename: $!\n"; next};
32:    while (defined (my $line = <$file>)) {
33:       while ($line =~ s{^.*?($wordpat)}{}) {
34:          my $word = $1;
35:          next if $word =~ m{$numpat}
36:               || $dictionary{$word} || $dictionary{lc $word};
37:          $exit_status ||= 1;
38:          print "$filename: $.: $word\n";
39:       }
40:    }
41:    close $file;
42: }
43:
```

```
 1: 3070542422 = strhash ("hashfn")
 2:         48 = strhash ("0")
 3:         57 = strhash ("9")
 4:         65 = strhash ("A")
 5:         90 = strhash ("Z")
 6:         97 = strhash ("a")
 7:        122 = strhash ("z")
 8:     101574 = strhash ("foo")
 9:      97299 = strhash ("bar")
10:      97307 = strhash ("baz")
11:     112340 = strhash ("qux")
12:    3482567 = strhash ("quux")
13:  107959604 = strhash ("quuux")
14: 3346747751 = strhash ("quuuux")
15:  669965204 = strhash ("quuuuux")
16: 3589052167 = strhash ("quuuuuux")
17: 3886434804 = strhash ("quuuuuuux")
18:  220394663 = strhash ("quuuuuuuux")
19:  593262906 = strhash ("Listing.misc.pdf")
20: 2374442171 = strhash ("Listing.misc.ps")
21:  105691274 = strhash ("Makefile")
22:      80962 = strhash ("RCS")
23: 3070542422 = strhash ("hashfn")
24:  148736715 = strhash ("hashfn.c")
25: 1202077622 = strhash ("hashfn.out")
26: 3338426469 = strhash ("pspell.perl")
27: 4294967295 = 0xFFFFFFFFu
```