

```
1: // $Id: hashfn.c,v 1.10 2013-08-12 13:30:08-07 - - $
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:         size_t hashcode = strhash (str);
18:         printf ("%20lu = strhash ("%s")\n", hashcode, str);
19:     }
20:     printf ("%20lu = 0xFFFFFFFFLU\n", 0xFFFFFFFFLU);
21:     printf ("%20lu = 0x%016lXLU\n", (size_t)-1L, (size_t)-1L);
22:     return EXIT_SUCCESS;
23: }
24:
```

```
1: // $Id: strhash.h,v 1.3 2014-03-05 19:24:07-08 - - $
2:
3: //
4: // NAME
5: //      strhash - return an unsigned 32-bit hash code for a string
6: //
7: // SYNOPSIS
8: //      size_t strhash (const char *string);
9: //
10:
11: #ifndef __STRHASH_H__
12: #define __STRHASH_H__
13:
14: #include <inttypes.h>
15:
16: size_t strhash (const char *string);
17:
18: #endif
19:
```

```
1: // $Id: strhash.c,v 1.6 2014-03-05 19:24:07-08 - - $
2:
3: #include <assert.h>
4: #include <stdio.h>
5: #include <sys/types.h>
6:
7: #include "strhash.h"
8:
9: size_t strhash (const char *string) {
10:     assert (string != NULL);
11:     size_t hash = 0;
12:     for (; *string != '\0'; ++string) {
13:         hash = *string + (hash << 6) + (hash << 16) - hash;
14:     }
15:     return hash;
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:               quuuuuuuuux quuuuuuuuuux quuuuuuuuuuux quuuuuuuuuuuux \
14:               quuuuuuuuuuuuux quuuuuuuuuuuuuux
15:
16: all : ${EXECBIN}
17:
18: % : %.c
19:     - cid + $<
20:     - checksource $<
21:     ${GCC} -o $@ ${HASHSRC}
22:
23: ci : ${LISFILES}
24:     - checksource ${LISFILES}
25:     - cid + ${LISFILES}
26:
27: lis : ${LISFILES} ${HASHOUT}
28:     mkpspdf ${LISTING} ${LISFILES} ${HASHOUT}
29:
30: ${HASHOUT} : hashfn
31:     hashfn ${TESTDATA} * >${HASHOUT}
32:     cat ${HASHOUT}
33:
34: spotless :
35:     - rm ${EXECBIN} ${HASHOUT}
36:
```

```
1: #!/usr/bin/perl
2: # $Id: pspell.perl,v 1.3 2012-12-07 14:03:09-08 - - $
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: my $defdict = "/afs/cats.ucsc.edu/courses/cmpps012b-wm/usr/dict/words";
19:
20: sub load_dictionary($) {
21:     my ($dictname) = @_;
22:     open my $dict, "<$dictname" or do {warn "$dictname: $!\n"; return};
23:     map {chomp; $dictionary{$_} = 1} <$dict>;
24:     close $dict;
25: }
26: load_dictionary $defdict unless $options{'n'};
27: load_dictionary $options{'d'} if defined $options{'d'};
28: die "dictionary is empty\n" unless %dictionary;
29:
30: my $numpat = qr{([[:digit:]]+([-:.]?[[:digit:]]+)*)};
31: my $wordpat = qr{([[:alnum:]]+([-&'.]?[[:alnum:]]+)*)};
32: for my $filename (@ARGV ? @ARGV : "-") {
33:     open my $file, "<$filename" or do {warn "$filename: $!\n"; next};
34:     while (defined (my $line = <$file>)) {
35:         while ($line =~ s{^.*?($wordpat)}{}) {
36:             my $word = $1;
37:             next if $word =~ m{^$numpat$}
38:                 || $dictionary{$word} || $dictionary{lc $word};
39:             $exit_status ||= 1;
40:             print "$filename: $.: $word\n";
41:         }
42:     }
43:     close $file;
44: }
45:
```

```
1: 7756476997639056566 = 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: 438936619302 = strhash ("foo")
9: 421722785715 = strhash ("bar")
10: 421722785723 = strhash ("baz")
11: 486272529716 = strhash ("qux")
12: 31898991676643207 = strhash ("quux")
13: 8059874666938206708 = strhash ("quuux")
14: 17586379889962775239 = strhash ("quuuux")
15: 8006775946444193460 = strhash ("quuuuux")
16: 2351300060583423495 = strhash ("quuuuuux")
17: 9705473926436590452 = strhash ("quuuuuuux")
18: 16905884376141941063 = strhash ("quuuuuuuux")
19: 9302223190657992756 = strhash ("quuuuuuuuux")
20: 16691869735408698503 = strhash ("quuuuuuuuuux")
21: 8128045823648079092 = strhash ("quuuuuuuuuuux")
22: 6987278989460250567 = strhash ("quuuuuuuuuuuux")
23: 12264430141747745204 = strhash ("quuuuuuuuuuuuux")
24: 16503581815662811911 = strhash ("quuuuuuuuuuuuuux")
25: 1532931250483629228 = strhash ("HEADER.html")
26: 10523919874407150170 = strhash ("Listing.misc.pdf")
27: 8968339241735688475 = strhash ("Listing.misc.ps")
28: 7287865400257976586 = strhash ("Makefile")
29: 352869156898 = strhash ("RCS")
30: 7756476997639056566 = strhash ("hashfn")
31: 235110086206995819 = strhash ("hashfn.c")
32: 7680267118805889189 = strhash ("pspell.perl")
33: 4294967295 = 0xFFFFFFFFLU
34: 18446744073709551615 = 0xFFFFFFFFFFFFFFFFLU
```