```
 1: // $Id: stringset.h,v 1.1 2013-09-20 19:45:47-07 - - $
 2:
 3: #ifndef __STRINGSET__
 4: #define __STRINGSET__
 5:
 6: #include <string>
 7: #include <unordered_set>
 8: using namespace std;
 9:
10: #include <stdio.h>
11:
12: const string* intern_stringset (const char*);
13:
14: void dump_stringset (FILE*);
15:
16: #endif
```

```
 1:
 2: #include <string>
 3: #include <unordered_set>
 4: using namespace std;
 5:
 6: #include "stringset.h"
 7:
 8: typedef unordered_set<string> stringset;
 9: typedef stringset::const_iterator stringset_citor;
10: typedef stringset::const_local_iterator stringset_bucket_citor;
11:
12: stringset set;
13:
14: const string* intern_stringset (const char* string) {
15:    pair<stringset_citor,bool> handle = set.insert (string);
16:    return &*handle.first;
17: }
18:
19: void dump_stringset (FILE* out) {
20:    size_t max_bucket_size = 0;
21:    for (size_t bucket = 0; bucket < set.bucket_count(); ++bucket) {
22:       bool need_index = true;
23:       size_t curr_size = set.bucket_size (bucket);
24:       if (max_bucket_size < curr_size) max_bucket_size = curr_size;
25:       for (stringset_bucket_citor itor = set.cbegin (bucket);
26:            itor != set.cend (bucket); ++itor) {
27:          if (need_index) fprintf (out, "stringset[%4lu]: ", bucket);
28:                     else fprintf (out, "          %4s   ", "");
29:          need_index = false;
30:          const string* str = &*itor;
31:          fprintf (out, "%22lu %p->\"%s\"\n", set.hash_function()(*str),
32:                   str, str->c_str());
33:       }
34:    }
35:    fprintf (out, "load_factor = %.3f\n", set.load_factor());
36:    fprintf (out, "bucket_count = %lu\n", set.bucket_count());
37:    fprintf (out, "max_bucket_size = %lu\n", max_bucket_size);
38: }
39:
```

```
 1: // $Id: main.cc,v 1.3 2013-09-23 14:39:10-07 - - $
 2:
 3: #include <string>
 4: using namespace std;
 5:
 6: #include <assert.h>
 7: #include <stdio.h>
 8: #include <stdlib.h>
 9: #include <string.h>
10:
11: #include "stringset.h"
12:
13: int main (int argc, char **argv) {
14:    for (int i = 1; i < argc; ++i) {
15:       const string* str = intern_stringset (argv[i]);
16:       printf ("intern (\"%s\") returned %p->\"%s\"\n",
17:               argv[i], str->c_str(), str->c_str());
18:    }
19:    dump_stringset (stdout);
20:    return EXIT_SUCCESS;
21: }
22:
```

```
 1: # $Id: Makefile,v 1.3 2013-09-23 14:39:10-07 - - $
 2:
 3: GPP   = g++ -g -O0 -Wall -Wextra -std=gnu++0x
 4: GRIND = valgrind --leak-check=full --show-reachable=yes
 5:
 6:
 7: all : teststring
 8:
 9: teststring : main.o stringset.o
10:         ${GPP} main.o stringset.o -o teststring
11:
12: %.o : %.cc
13:         ${GPP} -c $<
14:
15: ci :
16:         cid + Makefile stringset.h stringset.cc main.cc
17:
18: spotless : clean
19:         - rm teststring Listing.ps Listing.pdf \
20:
21: clean :
22:         -rm stringset.o main.o
23:
24: test.out : teststring
25:         ${GRIND} teststring * * * >test.out 2>&1
26:
27: lis : test.out
28:         mkpspdf Listing.ps stringset.h stringset.cc main.cc \
29:                 Makefile test.out
30:
31: # Depencencies.
32: main.o: main.cc stringset.h
33: stringset.o: stringset.cc stringset.h
34:
```

```
    1: ==6948== Memcheck, a memory error detector
    2: ==6948== Copyright (C) 2002-2012, and GNU GPL'd, by Julian Seward et al.
    3: ==6948== Using Valgrind-3.8.1 and LibVEX; rerun with -h for copyright info
    4: ==6948== Command: teststring Listing.pdf Listing.ps Makefile RCS main.cc mai
n.o stringset.cc stringset.h stringset.o test.out teststring teststring.output List
ing.pdf Listing.ps Makefile RCS main.cc main.o stringset.cc stringset.h stringset.o
 test.out teststring teststring.output Listing.pdf Listing.ps Makefile RCS main.cc
main.o stringset.cc stringset.h stringset.o test.out teststring teststring.output
    5: ==6948==
    6: intern ("Listing.pdf") returned 0x4c280f8->"Listing.pdf"
    7: intern ("Listing.ps") returned 0x4c281b8->"Listing.ps"
    8: intern ("Makefile") returned 0x4c28278->"Makefile"
    9: intern ("RCS") returned 0x4c28338->"RCS"
   10: intern ("main.cc") returned 0x4c283e8->"main.cc"
   11: intern ("main.o") returned 0x4c28498->"main.o"
   12: intern ("stringset.cc") returned 0x4c28548->"stringset.cc"
   13: intern ("stringset.h") returned 0x4c28608->"stringset.h"
   14: intern ("stringset.o") returned 0x4c286c8->"stringset.o"
   15: intern ("test.out") returned 0x4c28788->"test.out"
   16: intern ("teststring") returned 0x4c28848->"teststring"
   17: intern ("teststring.output") returned 0x4c28908->"teststring.output"
   18: intern ("Listing.pdf") returned 0x4c280f8->"Listing.pdf"
   19: intern ("Listing.ps") returned 0x4c281b8->"Listing.ps"
   20: intern ("Makefile") returned 0x4c28278->"Makefile"
   21: intern ("RCS") returned 0x4c28338->"RCS"
   22: intern ("main.cc") returned 0x4c283e8->"main.cc"
   23: intern ("main.o") returned 0x4c28498->"main.o"
   24: intern ("stringset.cc") returned 0x4c28548->"stringset.cc"
   25: intern ("stringset.h") returned 0x4c28608->"stringset.h"
   26: intern ("stringset.o") returned 0x4c286c8->"stringset.o"
   27: intern ("test.out") returned 0x4c28788->"test.out"
   28: intern ("teststring") returned 0x4c28848->"teststring"
   29: intern ("teststring.output") returned 0x4c28908->"teststring.output"
   30: intern ("Listing.pdf") returned 0x4c280f8->"Listing.pdf"
   31: intern ("Listing.ps") returned 0x4c281b8->"Listing.ps"
   32: intern ("Makefile") returned 0x4c28278->"Makefile"
   33: intern ("RCS") returned 0x4c28338->"RCS"
   34: intern ("main.cc") returned 0x4c283e8->"main.cc"
   35: intern ("main.o") returned 0x4c28498->"main.o"
   36: intern ("stringset.cc") returned 0x4c28548->"stringset.cc"
   37: intern ("stringset.h") returned 0x4c28608->"stringset.h"
   38: intern ("stringset.o") returned 0x4c286c8->"stringset.o"
   39: intern ("test.out") returned 0x4c28788->"test.out"
   40: intern ("teststring") returned 0x4c28848->"teststring"
   41: intern ("teststring.output") returned 0x4c28908->"teststring.output"
   42: stringset[   1]:   16535220712839172848 0x4c285a0->"stringset.cc"
   43: stringset[   3]:    2586491021746226264 0x4c288a0->"teststring"
   44:                    12271277041006505511 0x4c284e0->"main.o"
   45: stringset[   4]:   15856372366427549707 0x4c287e0->"test.out"
   46: stringset[   6]:   12165310408658569987 0x4c28150->"Listing.pdf"
   47: stringset[   7]:    3769442993623408023 0x4c28720->"stringset.o"
   48: stringset[  12]:   17940775167660870368 0x4c28430->"main.cc"
   49: stringset[  13]:   18201842504327843073 0x4c282d0->"Makefile"
   50:                      402963791104991022 0x4c28210->"Listing.ps"
   51: stringset[  20]:   11561774995555939875 0x4c28380->"RCS"
   52:                     3769439695088523390 0x4c28660->"stringset.h"
   53: stringset[  21]:   10888049205256263171 0x4c28960->"teststring.output"
   54: load_factor = 0.522
   55: bucket_count = 23
   56: max_bucket_size = 2
   57: ==6948==
```

```
58: ==6948== HEAP SUMMARY:
59: ==6948==     in use at exit: 0 bytes in 0 blocks
60: ==6948==   total heap usage: 50 allocs, 50 frees, 1,722 bytes allocated
61: ==6948==
62: ==6948== All heap blocks were freed -- no leaks are possible
63: ==6948==
64: ==6948== For counts of detected and suppressed errors, rerun with: -v
65: ==6948== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 6 from 6)
```