```
 1: // $Id: stringset.h,v 1.4 2014-10-09 15:42:13-07 - - $
 2:
 3: #ifndef __STRINGSET__
 4: #define __STRINGSET__
 5:
 6: #include <iostream>
 7: #include <string>
 8:
 9: const std::string* intern_stringset (const char*);
10:
11: void dump_stringset (std::ostream&);
12:
13: #endif
14:
```

```cpp
 1: // $Id: stringset.cpp,v 1.11 2014-10-09 15:47:45-07 - - $
 2:
 3: #include <iomanip>
 4: #include <unordered_set>
 5: using namespace std;
 6:
 7: #include "stringset.h"
 8:
 9: using stringset = unordered_set<string>;
10:
11: stringset set;
12:
13: const string* intern_stringset (const char* string) {
14:    pair<stringset::const_iterator,bool> handle = set.insert (string);
15:    return &*handle.first;
16: }
17:
18: void dump_stringset (ostream& out) {
19:    size_t max_bucket_size = 0;
20:    for (size_t bucket = 0; bucket < set.bucket_count(); ++bucket) {
21:       bool need_index = true;
22:       size_t curr_size = set.bucket_size (bucket);
23:       if (max_bucket_size < curr_size) max_bucket_size = curr_size;
24:       for (stringset::const_local_iterator itor = set.cbegin (bucket);
25:            itor != set.cend (bucket); ++itor) {
26:          if (need_index) out << "stringset[" << setw(4) << bucket
27:                               << "]: ";
28:                     else out << setw(17) << "";
29:          need_index = false;
30:          const string* str = &*itor;
31:          out << setw(22) << set.hash_function()(*str) << ": "
32:              << str << "->\"" << *str << "\"" << endl;
33:       }
34:    }
35:    out << "load_factor = " << fixed << setprecision(3)
36:        << set.load_factor() << endl;
37:    out << "bucket_count = " << set.bucket_count() << endl;
38:    out << "max_bucket_size = " << max_bucket_size << endl;
39: }
40:
```

```cpp
 1: // $Id: main.cpp,v 1.6 2014-10-09 15:44:18-07 - - $
 2:
 3: #include <cstdlib>
 4: #include <iostream>
 5: #include <string>
 6: #include <vector>
 7: using namespace std;
 8:
 9:
10: #include "stringset.h"
11:
12: int main (int argc, char** argv) {
13:    vector<string> args (&argv[1], &argv[argc]);
14:    for (const string& arg: args) {
15:       const string* str = intern_stringset (arg.c_str());
16:       cout << "intern(" << arg << ") returned " << str
17:            << "->\"" << *str << "\"" << endl;
18:    }
19:    dump_stringset (cout);
20:    return EXIT_SUCCESS;
21: }
22:
```

```
 1: # $Id: Makefile,v 1.13 2014-10-09 15:49:32-07 - - $
 2:
 3: GPP   = g++ -g -O0 -Wall -Wextra -std=gnu++11
 4: GRIND = valgrind --leak-check=full --show-reachable=yes
 5:
 6: all : teststring
 7:
 8: teststring : main.o stringset.o
 9:         ${GPP} main.o stringset.o -o teststring
10:
11: %.o : %.cpp
12:         ${GPP} -c $<
13:
14: ci :
15:         cid + Makefile stringset.h stringset.cpp main.cpp
16:
17: spotless : clean
18:         - rm teststring Listing.ps Listing.pdf test?.out test?.err
19:
20: clean :
21:         -rm stringset.o main.o
22:
23: test : teststring
24:         ${GRIND} teststring * * >test1.out 2>test1.err
25:         ${GRIND} teststring foo foo foo foo bar bar bar foo qux baz \
26:                 >test2.out 2>test2.err
27:
28: lis : test
29:         mkpspdf Listing.ps stringset.h stringset.cpp main.cpp \
30:                 Makefile test1.out test1.err test2.out test2.err
31:
32: # Depencencies.
33: main.o: main.cpp stringset.h
34: stringset.o: stringset.cpp stringset.h
35:
```

```
 1: intern(HEADER.html) returned 0x4c2f9c8->"HEADER.html"
 2: intern(Makefile) returned 0x4c2fa98->"Makefile"
 3: intern(RCS) returned 0x4c2fb58->"RCS"
 4: intern(main.cpp) returned 0x4c2fc28->"main.cpp"
 5: intern(main.o) returned 0x4c2fce8->"main.o"
 6: intern(stringset.cpp) returned 0x4c2fdb8->"stringset.cpp"
 7: intern(stringset.h) returned 0x4c2fe88->"stringset.h"
 8: intern(stringset.o) returned 0x4c2ff58->"stringset.o"
 9: intern(teststring) returned 0x4c30028->"teststring"
10: intern(HEADER.html) returned 0x4c2f9c8->"HEADER.html"
11: intern(Makefile) returned 0x4c2fa98->"Makefile"
12: intern(RCS) returned 0x4c2fb58->"RCS"
13: intern(main.cpp) returned 0x4c2fc28->"main.cpp"
14: intern(main.o) returned 0x4c2fce8->"main.o"
15: intern(stringset.cpp) returned 0x4c2fdb8->"stringset.cpp"
16: intern(stringset.h) returned 0x4c2fe88->"stringset.h"
17: intern(stringset.o) returned 0x4c2ff58->"stringset.o"
18: intern(teststring) returned 0x4c30028->"teststring"
19: stringset[    1]:   12311697016174216101: 0x4c2fdb8->"stringset.cpp"
20: stringset[    2]:    2099682443743551108: 0x4c2fce8->"main.o"
21: stringset[    4]:   14383043818818809721: 0x4c30028->"teststring"
22: stringset[    6]:   17041606903804112922: 0x4c2fc28->"main.cpp"
23:                       994128771139992428: 0x4c2fb58->"RCS"
24:                     13646535705723827550: 0x4c2f9c8->"HEADER.html"
25: stringset[    7]:    8902767590177878864: 0x4c2fa98->"Makefile"
26: stringset[   10]:   12900681736301144686: 0x4c2ff58->"stringset.o"
27:                      1356443961750385397: 0x4c2fe88->"stringset.h"
28: load_factor = 0.818
29: bucket_count = 11
30: max_bucket_size = 3
```

```
 1: ==8881== Memcheck, a memory error detector
 2: ==8881== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
 3: ==8881== Using Valgrind-3.9.0 and LibVEX; rerun with -h for copyright in
fo
 4: ==8881== Command: teststring HEADER.html Makefile RCS main.cpp main.o st
ringset.cpp stringset.h stringset.o teststring HEADER.html Makefile RCS main.cp
p main.o stringset.cpp stringset.h stringset.o teststring
 5: ==8881==
 6: ==8881==
 7: ==8881== HEAP SUMMARY:
 8: ==8881==     in use at exit: 0 bytes in 0 blocks
 9: ==8881==   total heap usage: 47 allocs, 47 frees, 1,672 bytes allocated
10: ==8881==
11: ==8881== All heap blocks were freed -- no leaks are possible
12: ==8881==
13: ==8881== For counts of detected and suppressed errors, rerun with: -v
14: ==8881== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 6 from 6)
```

```
 1: intern(foo) returned 0x4c2f598->"foo"
 2: intern(foo) returned 0x4c2f598->"foo"
 3: intern(foo) returned 0x4c2f598->"foo"
 4: intern(foo) returned 0x4c2f598->"foo"
 5: intern(bar) returned 0x4c2f778->"bar"
 6: intern(bar) returned 0x4c2f778->"bar"
 7: intern(bar) returned 0x4c2f778->"bar"
 8: intern(foo) returned 0x4c2f598->"foo"
 9: intern(qux) returned 0x4c2f958->"qux"
10: intern(baz) returned 0x4c2fa18->"baz"
11: stringset[   0]:    9631199822919835226: 0x4c2f598->"foo"
12: stringset[   7]:    8658707281607720454: 0x4c2f958->"qux"
13: stringset[   9]:   11474628671133349555: 0x4c2f778->"bar"
14: stringset[  10]:   12938591777111562088: 0x4c2fa18->"baz"
15: load_factor = 0.364
16: bucket_count = 11
17: max_bucket_size = 1
```

```
 1: ==8883== Memcheck, a memory error detector
 2: ==8883== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
 3: ==8883== Using Valgrind-3.9.0 and LibVEX; rerun with -h for copyright in
fo
 4: ==8883== Command: teststring foo foo foo foo bar bar bar foo qux baz
 5: ==8883==
 6: ==8883==
 7: ==8883== HEAP SUMMARY:
 8: ==8883==     in use at exit: 0 bytes in 0 blocks
 9: ==8883==   total heap usage: 26 allocs, 26 frees, 824 bytes allocated
10: ==8883==
11: ==8883== All heap blocks were freed -- no leaks are possible
12: ==8883==
13: ==8883== For counts of detected and suppressed errors, rerun with: -v
14: ==8883== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 6 from 6)
```