

```
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:
```

```
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:
```

```
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: # Dependencies.
33: main.o: main.cpp stringset.h
34: stringset.o: stringset.cpp stringset.h
35:
```

```
1: intern(HEADER.html) returned 0x4c30f68->"HEADER.html"
2: intern(Listing.pdf) returned 0x4c31038->"Listing.pdf"
3: intern(Listing.ps) returned 0x4c31108->"Listing.ps"
4: intern(Makefile) returned 0x4c311d8->"Makefile"
5: intern(RCS) returned 0x4c31298->"RCS"
6: intern(main.cpp) returned 0x4c31368->"main.cpp"
7: intern(main.o) returned 0x4c31428->"main.o"
8: intern(stringset.cpp) returned 0x4c314f8->"stringset.cpp"
9: intern(stringset.h) returned 0x4c315c8->"stringset.h"
10: intern(stringset.o) returned 0x4c31698->"stringset.o"
11: intern(test1.err) returned 0x4c31768->"test1.err"
12: intern(test1.out) returned 0x4c31938->"test1.out"
13: intern(test2.err) returned 0x4c31a08->"test2.err"
14: intern(test2.out) returned 0x4c31ad8->"test2.out"
15: intern(teststring) returned 0x4c31ba8->"teststring"
16: intern(HEADER.html) returned 0x4c30f68->"HEADER.html"
17: intern(Listing.pdf) returned 0x4c31038->"Listing.pdf"
18: intern(Listing.ps) returned 0x4c31108->"Listing.ps"
19: intern(Makefile) returned 0x4c311d8->"Makefile"
20: intern(RCS) returned 0x4c31298->"RCS"
21: intern(main.cpp) returned 0x4c31368->"main.cpp"
22: intern(main.o) returned 0x4c31428->"main.o"
23: intern(stringset.cpp) returned 0x4c314f8->"stringset.cpp"
24: intern(stringset.h) returned 0x4c315c8->"stringset.h"
25: intern(stringset.o) returned 0x4c31698->"stringset.o"
26: intern(test1.err) returned 0x4c31768->"test1.err"
27: intern(test1.out) returned 0x4c31938->"test1.out"
28: intern(test2.err) returned 0x4c31a08->"test2.err"
29: intern(test2.out) returned 0x4c31ad8->"test2.out"
30: intern(teststring) returned 0x4c31ba8->"teststring"
31: stringset[ 0]: 14383043818818809721: 0x4c31ba8->"teststring"
32: 13646535705723827550: 0x4c30f68->"HEADER.html"
33: stringset[ 1]: 2099682443743551108: 0x4c31428->"main.o"
34: stringset[ 3]: 10828860385276898342: 0x4c31a08->"test2.err"
35: stringset[ 4]: 994128771139992428: 0x4c31298->"RCS"
36: stringset[ 6]: 12311697016174216101: 0x4c314f8->"stringset.cpp"
37: stringset[10]: 8902767590177878864: 0x4c311d8->"Makefile"
38: stringset[11]: 12900681736301144686: 0x4c31698->"stringset.o"
39: stringset[13]: 14217879530798290974: 0x4c31768->"test1.err"
40: stringset[15]: 6788790452799346656: 0x4c31038->"Listing.pdf"
41: stringset[16]: 17041606903804112922: 0x4c31368->"main.cpp"
42: stringset[17]: 1553654622159544091: 0x4c31108->"Listing.ps"
43: stringset[20]: 1356443961750385397: 0x4c315c8->"stringset.h"
44: stringset[21]: 7027456624921073501: 0x4c31ad8->"test2.out"
45: 2324604740214821100: 0x4c31938->"test1.out"
46: load_factor = 0.652
47: bucket_count = 23
48: max_bucket_size = 2
```

```
1: ==28074== Memcheck, a memory error detector
2: ==28074== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al
.
3: ==28074== Using Valgrind-3.9.0 and LibVEX; rerun with -h for copyright i
nfo
4: ==28074== Command: teststring HEADER.html Listing.pdf Listing.ps Makefil
e RCS main.cpp main.o stringset.cpp stringset.h stringset.o test1.err test1.out
test2.err test2.out teststring HEADER.html Listing.pdf Listing.ps Makefile RCS
main.cpp main.o stringset.cpp stringset.h stringset.o test1.err test1.out test
2.err test2.out teststring
5: ==28074==
6: ==28074==
7: ==28074== HEAP SUMMARY:
8: ==28074==      in use at exit: 0 bytes in 0 blocks
9: ==28074==    total heap usage: 78 allocs, 78 frees, 2,924 bytes allocated
10: ==28074==
11: ==28074== All heap blocks were freed -- no leaks are possible
12: ==28074==
13: ==28074== For counts of detected and suppressed errors, rerun with: -v
14: ==28074== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 6 from 6)
```

```
1: intern(foo) returned 0x4c30598->"foo"
2: intern(foo) returned 0x4c30598->"foo"
3: intern(foo) returned 0x4c30598->"foo"
4: intern(foo) returned 0x4c30598->"foo"
5: intern(bar) returned 0x4c30778->"bar"
6: intern(bar) returned 0x4c30778->"bar"
7: intern(bar) returned 0x4c30778->"bar"
8: intern(foo) returned 0x4c30598->"foo"
9: intern(qux) returned 0x4c30958->"qux"
10: intern(baz) returned 0x4c30a18->"baz"
11: stringset[ 0]: 9631199822919835226: 0x4c30598->"foo"
12: stringset[ 7]: 8658707281607720454: 0x4c30958->"qux"
13: stringset[ 9]: 11474628671133349555: 0x4c30778->"bar"
14: stringset[10]: 12938591777111562088: 0x4c30a18->"baz"
15: load_factor = 0.364
16: bucket_count = 11
17: max_bucket_size = 1
```

```
1: ==28078== Memcheck, a memory error detector
2: ==28078== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al
.
3: ==28078== Using Valgrind-3.9.0 and LibVEX; rerun with -h for copyright i
nfo
4: ==28078== Command: teststring foo foo foo foo bar bar bar foo qux baz
5: ==28078==
6: ==28078==
7: ==28078== HEAP SUMMARY:
8: ==28078==      in use at exit: 0 bytes in 0 blocks
9: ==28078==    total heap usage: 26 allocs, 26 frees, 824 bytes allocated
10: ==28078==
11: ==28078== All heap blocks were freed -- no leaks are possible
12: ==28078==
13: ==28078== For counts of detected and suppressed errors, rerun with: -v
14: ==28078== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 6 from 6)
```