```
 1: // $Id: vectorleak.cpp,v 1.4 2014-04-22 18:58:28-07 - - $
 2:
 3: //
 4: // Illustrate how leaks may happen when a vector of pointers is
 5: // created.  If the vector is not explicitly cleared, when it is
 6: // deleted, the objects it access are not deleted.
 7: //
 8:
 9: #include <iostream>
10: #include <vector>
11:
12: using namespace std;
13:
14: int main (int argc, char **argv) {
15:    vector<string*> vs;
16:    for (int index = 1; index < argc; ++index) {
17:       vs.push_back (new string (argv[index]));
18:    }
19:    auto begin = vs.begin();
20:    for (auto itor = begin; itor != vs.end(); ++itor) {
21:       cout << itor - begin << ": " << *itor << "->" << **itor << endl;
22:    }
23:    return 0;
24: }
25:
26: /*
27: //TEST// alias grind='valgrind --leak-check=full --show-reachable=yes'
28: //TEST// grind vectorleak these are some arguments to check on leak \
29: //TEST//       >vectorleak.out 2>&1
30: //TEST// mkpspdf vectorleak.ps vectorleak.cpp* vectorleak.out
31: */
32:
```

```
1: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting vectorleak.cpp
2: vectorleak.cpp: $Id: vectorleak.cpp,v 1.4 2014-04-22 18:58:28-07 - - $
3: g++ -g -O0 -Wall -Wextra -std=gnu++11 vectorleak.cpp -o vectorleak -lm
4: rm -f vectorleak.o
5: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished vectorleak.cpp
```

```
    1: ==4041== Memcheck, a memory error detector
    2: ==4041== Copyright (C) 2002-2012, and GNU GPL'd, by Julian Seward et al.
    3: ==4041== Using Valgrind-3.8.1 and LibVEX; rerun with -h for copyright in
fo
    4: ==4041== Command: vectorleak these are some arguments to check on leak
    5: ==4041==
    6: 0: 0x4c2b040->these
    7: 1: 0x4c2b140->are
    8: 2: 0x4c2b240->some
    9: 3: 0x4c2b350->arguments
   10: 4: 0x4c2b410->to
   11: 5: 0x4c2b540->check
   12: 6: 0x4c2b5f0->on
   13: 7: 0x4c2b6a0->leak
   14: ==4041==
   15: ==4041== HEAP SUMMARY:
   16: ==4041==     in use at exit: 298 bytes in 16 blocks
   17: ==4041==   total heap usage: 20 allocs, 4 frees, 418 bytes allocated
   18: ==4041==
   19: ==4041== 234 bytes in 8 blocks are indirectly lost in loss record 1 of 2
   20: ==4041==    at 0x4A075FC: operator new(unsigned long) (in /opt/rh/devtoo
lset-2/root/usr/lib64/valgrind/vgpreload_memcheck-amd64-linux.so)
   21: ==4041==    by 0x35DD09C3C8: std::string::_Rep::_S_create(unsigned long,
 unsigned long, std::allocator<char> const&) (in /usr/lib64/libstdc++.so.6.0.13
)
   22: ==4041==    by 0x35DD09CDE4: ??? (in /usr/lib64/libstdc++.so.6.0.13)
   23: ==4041==    by 0x35DD09CF32: std::basic_string<char, std::char_traits<ch
ar>, std::allocator<char> >::basic_string(char const*, std::allocator<char> con
st&) (in /usr/lib64/libstdc++.so.6.0.13)
   24: ==4041==    by 0x400DB8: main (vectorleak.cpp:17)
   25: ==4041==
   26: ==4041== 298 (64 direct, 234 indirect) bytes in 8 blocks are definitely
lost in loss record 2 of 2
   27: ==4041==    at 0x4A075FC: operator new(unsigned long) (in /opt/rh/devtoo
lset-2/root/usr/lib64/valgrind/vgpreload_memcheck-amd64-linux.so)
   28: ==4041==    by 0x400D90: main (vectorleak.cpp:17)
   29: ==4041==
   30: ==4041== LEAK SUMMARY:
   31: ==4041==    definitely lost: 64 bytes in 8 blocks
   32: ==4041==    indirectly lost: 234 bytes in 8 blocks
   33: ==4041==      possibly lost: 0 bytes in 0 blocks
   34: ==4041==    still reachable: 0 bytes in 0 blocks
   35: ==4041==         suppressed: 0 bytes in 0 blocks
   36: ==4041==
   37: ==4041== For counts of detected and suppressed errors, rerun with: -v
   38: ==4041== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 6 from 6)
```