C++ debugging

Bruce Merry

IOI Training Dec 2013

Outline

1 The GNU Debugger

- Introduction
- Setup
- Demo

2 Catching Bugs

- Assertions
- Debug Containers
- Address Sanitizer
- Valgrind

Outline

What is GDB?

- Tool that peeks inside your program
- Helps examine what is happening
- Helps trace crashes
- Integrated into Eclipse, some other IDEs

GDB vs debug printing

Debug prints are good for:

- Dumping large amounts of data, when you know what you want to see

A debugger is better for:

- Following the flow of execution
- Determining the cause of a crash
- Testing hypotheses as execution proceeds

Outline

Compiler Options

- Do not compile with `-O2`
- Compile with `-g` to embed debug information
- On Ubuntu 13.10, use `-gdwarf-3` as well

Pretty Printers for STL Containers

For home use (Ubuntu 13.10):

- Install `libstdc++6-4.8-dbg`
- Put the following in .gdbinit:

```
python
sys.path.insert(0, '/usr/share/gcc-4.8/python')
end
```

This allows STL containers to be pretty-printed

Outline

Demo

Outline

1 The GNU Debugger
- Introduction
- Setup
- Demo

2 Catching Bugs
- Assertions
- Debug Containers
- Address Sanitizer
- Valgrind

Assertions

Reminder:

```
assert(condition_that_should_be_true);
```

Use GDB to debug the failure.

Outline

1 The GNU Debugger
- Introduction
- Setup
- Demo

2 Catching Bugs
- Assertions
- Debug Containers
- Address Sanitizer
- Valgrind

Unsafe Containers

STL containers do not check for errors:

```
vector<int> v(4);
v[4] = 123; // ANYTHING can happen!
```

This is good for performance, bad for debugging.

GCC Debug Containers

Compile with `-D_GLIBCXX_DEBUG`.

```
/usr/include/c++/4.8/debug/vector:346:error: attempt to subscript container
    with out-of-bounds index 4, but container only holds 4 elements.

Objects involved in the operation:
sequence "this" @ 0x0x7fffed18fa70 {
  type = NSt7__debug6vectorIiSaIiEEE;
}
Aborted (core dumped)
```

Again, use GDB to investigate.

Outline

Address Sanitizer

- Compiler flag that inserts checks into your code (about 2x slower!)
- Not specific to STL, so can catch array errors
- Also catches other errors like use-after-free
- Available from GCC 4.8

Using Address Sanitizer

Compile with `-fsanitize=address`. Run.

```
==25928== ERROR: AddressSanitizer: heap-buffer-overflow on address 0x60040000e000 at
WRITE of size 4 at 0x60040000e000 thread T0
    #0 0x400dfc (/home/bmerry/compolymp/presentations/cxx-debug/src/debugvector+0x400
    #1 0x7f4a112a4de4 (/lib/x86_64-linux-gnu/libc-2.17.so+0x21de4)
    #2 0x400c78 (/home/bmerry/compolymp/presentations/cxx-debug/src/debugvector+0x400
0x60040000e000 is located 0 bytes to the right of 16-byte region [0x60040000dff0,0x60
allocated by thread T0 here:
    #0 0x7f4a11b7684a (/usr/lib/x86_64-linux-gnu/libasan.so.0.0.0+0x1184a)
    #1 0x40161c (/home/bmerry/compolymp/presentations/cxx-debug/src/debugvector+0x401
    #2 0x401562 (/home/bmerry/compolymp/presentations/cxx-debug/src/debugvector+0x401
    #3 0x4013e2 (/home/bmerry/compolymp/presentations/cxx-debug/src/debugvector+0x401
    #4 0x401140 (/home/bmerry/compolymp/presentations/cxx-debug/src/debugvector+0x401
    #5 0x400fab (/home/bmerry/compolymp/presentations/cxx-debug/src/debugvector+0x400
    #6 0x400da5 (/home/bmerry/compolymp/presentations/cxx-debug/src/debugvector+0x400
    #7 0x7f1564863de4 (/lib/x86_64-linux-gnu/libc-2.17.so+0x21de4)
```

The `asan_symbolize` script can help decide addresses into line numbers (it is not shipped with GCC: you need to download it or install clang).

Outline

1 The GNU Debugger
- Introduction
- Setup
- Demo

2 Catching Bugs
- Assertions
- Debug Containers
- Address Sanitizer
- Valgrind

Valgrind

- Separate program; no recompilation necessary
- More robust and powerful than ASAN
- Also catches uninitialized data
- Slower and more memory-hungry

Using Valgrind


```
valgrind ./myprogram
```

```
==26020== Invalid write of size 4
==26020==    at 0x400A0F: main (debugvector.cpp:7)
==26020==  Address 0x5a1a050 is 0 bytes after a block of size 16 alloc'd
==26020==    at 0x4C2A879: operator new(unsigned long) (in /usr/lib/valgrind/vgpreloa
==26020==    by 0x400FB7: __gnu_cxx::new_allocator<int>::allocate(unsigned long, void
==26020==    by 0x400F02: std::_Vector_base<int, std::allocator<int> >::_M_allocate(u
==26020==    by 0x400DF0: std::_Vector_base<int, std::allocator<int> >::_M_create_sto
==26020==    by 0x400C6C: std::_Vector_base<int, std::allocator<int> >::_Vector_base(
==26020==    by 0x400B43: std::vector<int, std::allocator<int> >::vector(unsigned lon
==26020==    by 0x4009F1: main (debugvector.cpp:6)
```