C++ debugging

Bruce Merry

IOI Training Feb 2020

Outline

Outline

What is GDB?

- Tool that peeks inside your program
- Helps examine what is happening
- Helps trace crashes
- Integrated into a number of 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
- Install C++ library symbols (`libstdc++6-5-dbg`)

Outline

Demo

Outline

Configuration

Behaviour can be adjusted via `~/.gdbinit`

```
set history filename ~/.gdb_history
set history save on
set env ASAN_OPTIONS=detect_leaks=0
set print array on
set print pretty on
```

Outline

Assertions


```
assert(condition_that_should_be_true);
```

Use GDB to debug the failure.

Outline

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`.

- Out-of-bounds accesses
- Pop from an empty container
- Incrementing/decrementing terminal iterators
- Undefined iterator comparisons
- And more...

Outline

Address Sanitizer

Compile with `-fsanitize=address`

- 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

Undefined Behaviour Sanitizer

Compile with `-fsanitize=undefined`

- Signed integer overflow
- Invalid bit shifts
- Falling off end of function without returning
- Some out-of-bounds array accesses (not all)
- Misc other checks

Outline

Valgrind

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