Debuggers

- Step through code one line at a time
- Inspect variables, including structs and classes
- View disassembly
- Check the call stack

Debuggers

- Step through code one line at a time
- Inspect variables, including structs and classes
- View disassembly
- Check the call stack
- `gdb` Command-line debugger
- `kdbg` GUI frontend for `gdb`

Using `gdb`

- `gdb your-program` launches the debugger
- Note: You will want to compile with `g++ -g`
- `run arg1 arg2 ...` runs the command with command line arguments
- `backtrace` or `bt` shows the call stack

Setting breakpoints with `gdb`

- `break filename.cpp:10` will stop execution whenever line 10 in 'filename.cpp' is reached.
- `continue` resumes running as normal.
- `step` runs one more line of code.
- `next` runs until execution is on the next line.
- `finish` runs until the current function returns.
- `delete` removes all breakpoints.
- More on breakpoints

Looking at variables with `gdb`

- `p variable` prints the contents of ‘variable’.
- `p` also works with expressions of just about any sort.

Looking at variables with `gdb`

- `p variable` prints the contents of 'variable'.
- `p` also works with expressions of just about any sort.
- `x address` examines one word memory at a given address.
- `x/2 address` examines two words of memory.
- More on examining memory

Looking at variables with `gdb`

- `p variable` prints the contents of 'variable'.
- `p` also works with expressions of just about any sort.
- `x address` examines one word memory at a given address.
- `x/2 address` examines two words of memory.
- More on examining memory
- `info registers` lists all register values.
- `p $regname` prints the value of a register.

Miscellaneous

Conditional Breakpoints:

`condition breakpoint_number expression`

Editing variables with `gdb`:

- `set var VARIABLE_NAME = value` assigns 'value' to 'VARIABLE_NAME'
- `set {int}0x1234 = 4` writes 4 as an integer to the memory address 0x1234

Miscellaneous

Conditional Breakpoints:

`condition breakpoint_number expression`

Editing variables with `gdb`:

- `set var VARIABLE_NAME = value` assigns 'value' to 'VARIABLE_NAME'
- `set {int}0x1234 = 4` writes 4 as an integer to the memory address 0x1234

Disassembling code: `disassemble function` prints the assembly for a function.