GDB: GNU Debugger

NanoCad Tutorial 12/7/2011
Writing Good Software

- Programming is easy
- Debugging is hard
- Best way to debug is to avoid debugging
  - Use lots of comments in your code
  - Write modular code, and test submodules
  - Utilize good test benches
Methods for debugging

- `fprintf / cout`
  - Useful for dumping program execution details
    - Keep track of the program execution flow
  - Terrible for tracing segmentation faults
- `GDB`
  - Great for tracing segmentation faults
  - Trying to figure out what is wrong with a piece of code
  - Checking the execution of a loop
- `Valgrind`
  - Checking for memory leaks
Methods for debugging

- **fprintf / cout**
  - Useful for dumping program execution details
    - Keep track of the program execution flow
  - Terrible for tracing segmentation faults

- **GDB**
  - Great for tracing segmentation faults
  - Trying to figure out what is wrong with a piece of code
  - Checking the execution of a loop

- **Valgrind**
  - Checking for memory leaks
Things you can do in GDB

- **Breakpoints** *(break)*
  - Stop the program at a line in code
    - Can also stop on conditions (e.g. \texttt{n == 10})
- **Step through program** *(step / next)*
  - Execute the program line by line
- **Print Values** *(print)*
  - Print the values of variables
  - Can also use this to execute subprograms
GDB Example
What about GUI?

- There are GUIs that work on top of GDB
  - Emacs
  - Eclipse