



# Development: Debugging

---

CS 2204

Class meeting 9



# Overview of software development process

---

- Creation of source files (.c, .h, .cpp)
  - RCS, last week
- Compilation (e.g. \*.c → \*.o) and linking
  - gcc and make, two weeks ago
- Running and testing programs
  - gdb debugger, this week



# Why use a debugger?

---

- No one writes perfect code first time, every time
- Desk checking code can be tedious and error-prone
- Putting print statements in the code requires re-compilation and a guess as to the source of the problem
- Debuggers are powerful and flexible



# Common debugger functions

---

- Run program
- Stop program at breakpoints
- Execute one line at a time
- Display values of variables
- Show sequence of function calls
- Catch signals



# The GNU debugger (gdb)

---

- Free, like all GNU software
- Command line debugger
- Most common ways to invoke:
  - `gdb executable`
  - `gdb executable core`
  - `gdb executable process_id`



# Execution commands

---

- `list` or `l` (list code)
  - `list`
  - `list main`
  - `list 56`
- `run` or `r` (run program from beginning)
  - `run`
  - `run file.txt file2.txt`
- `next` or `n` (execute next line, stepping over function calls)
- `step` or `s` (execute next line, stepping into function calls)



# Breakpoint commands

---

- `break` or `b` (set a breakpoint)
  - `break main`
  - `break 10`
- `delete` or `d` (delete a breakpoint)
  - `delete`
  - `delete 2`
- `condition` (make a breakpoint conditional)
  - `condition 1 x<4`
- `continue` or `c` (continue execution when stopped)



# Program information commands

---

- `print` or `p` (print value)
  - `print x`
  - `print x*y`
  - `print function(x)`
- `display` (continuously display value)
- `undisplay` (remove displayed value)
- `where` (show current function stack)



# Miscellaneous commands

---

- `set` (change a value)
  - `set n=3`
- `help` or `h` (display help text)
  - `help`
  - `help step`
  - `help breakpoints`
- `quit` or `q` (quit gdb)