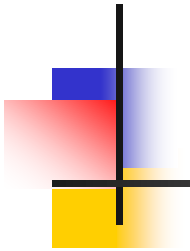
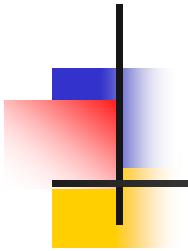


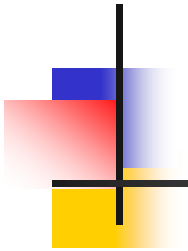
System Programming

Debugging



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

Debugging with gdb

- To debug a program, it has to be compiled with –g option
  - gcc -g hello.c
  - gcc -g -o hello hello.c
- gdb
  - Free, like all GNU software
  - Command line debugger
- Usage
  - gdb a.out
  - gdb hello

Basic gdb Commands

- Help: help
- Breakpoints: break main
- Running: run arg_list
- Step to next line: next
- Step into functions: step
- Continue running: cont
- List source: list
- Quiting: quit

- Running GDB with a core dump
- NetBSD: gdb name name.core
- Solaris: gdb name core

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`
- `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`)