Lab 2: GDB

CS104

Lab Materials

- Cd into your resources folder
- Git pull to get lab materials (folder called lab2)
- Make copy of lab2 in your hw-username folder
- You will be editing the copy of lab2 in your hw-username folder

1. `answers.txt`
2. `game_of_pointers_student1.cpp`
3. `game_of_pointers_student2.cpp`
4. `input1.txt`
5. `input2.txt`
6. `input3.txt`
7. `Makefile`
8. `output1.check`
9. `output2.check`
10. `output3.check`

What Is GDB?

- Powerful debugging tool!
- Executes your code while also allowing you to see why and when it crashes
- You can set breakpoints within your code to pause the program
    - Execute variables, call functions, examine the stack

Common Terminations

First two will automatically trigger debugger to break

- **Segfault**: when a program tries to read or write outside the memory that is allocated for it, or to write memory that can only be read.
- **Abort**: indicates an error detected by the program itself.

The other issues could be:

- **Infinite loop or recursion**: caused by faulty logic or base case.
- **Logic**: `2 + 2 == 5`? code is correct but the logic is faulty.
- **Translation error**: the logic is correct but it was just coded incorrectly.

Questions to Ask Yourself

- What line is the problem on?
  - When is it terminating/giving you an error when you run gdb?
  - Valgrind can help with segfaults and memory issues
  - Break points, cerr statements (guaranteed flushed to terminal before termination)
- When does the bug occur?
  - Are there specific circumstances? Maybe an edge case? Or only when it goes through one type of if statement?
- Can I reliably produce this bug over and over?
  - Consider making a separate test case to make the problem clearer (such as in future PAs)

Common Commands

run or r -> executes the program from start to end.

break or b -> sets breakpoint on a particular line.

disable -> disable a breakpoint.

enable -> enable a disabled breakpoint.

next or n -> executes next line of code, but don't dive into functions.

step -> go to next instruction, diving into the function.

list or l -> displays the code.

print or p -> used to display the stored value.

quit or q -> exits out of gdb.

clear -> to clear all breakpoints.

continue -> continue normal execution.

Also more in the lab instructions

Game of Pointers

- Lab will guide you through 5 problems
- As you debug, write your answers in answers.txt

Reminder: you have to run GDB in DOCKER!

ch start csci104

ch shell csci104

Checkoffs

- Must complete lab OR show effort throughout whole lab section
- When you think you have finished, ask to go to a breakout room
- Share your answers with one of the CPs
- You with PRIVATELY zoom message that CP your USC email and ID #

Demo