CSCI-2270
Recitation 2
Instructor: Boese

Debugging Memory

Objectives

- Use gdb to find and understand memory issues
- Practice hunting for bugs in code instead of using endless cout statements ☺
- Understand one of the common causes of segmentation faults – array indices out of bounds

GDB (the gnu debugger) is a debugger used for C and C++ code. It can run your program interactively, so you can see how it branches or recurs, and check for certain kinds of errors. Understanding gdb's basics will generalize well to other debugging frameworks.

Files

Download the files required for this lab from the website Labs section.

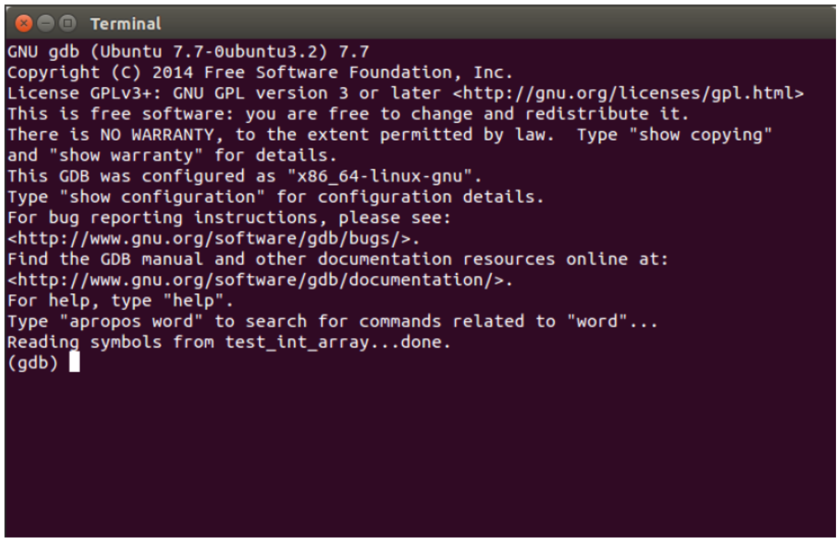
GDB

To use gdb, we have to compile our code with a special flag, `–g`

For instance, here are our compiler commands for the code for this lab:
make

```
g++ -std=c++11 -Wall -g -c test_int_array.cpp -o test_int_array.o
g++ -std=c++11 -Wall -g -c terrible_dynamic_size_array_unsorted.cpp -o
terrible_dynamic_size_array_unsorted.o
g++ -std=c++11 -Wall -g test_int_array.o terrible_dynamic_size_array_unsorted.o -o test_int_array
make[1]: `terrible_dynamic_size_array_unsorted.o' is up to date.
make[1]: Leaving directory `/home/user/Dropbox/csci2270-Fall-2014/labs/lab-7/Solved'
Compilation finished successfully.
```

Starting gdb by typing
gdb



```
GNU gdb (Ubuntu 7.7-0ubuntu3.2) 7.7
Copyright (C) 2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from test_int_array...done.
(gdb)
```

To run the code, you can type

```
run
```

or just

```
r
```

Add a number and see what the code does.

Recall that at this point, only the init and add methods have run. One of them must be causing this problem. Or both of them.

Cancel out of that run of gdb (with control+c), close the window, and we'll step through the code more slowly.

Run gdb.
This time, we'll begin by telling gdb to set a breakpoint, which pauses the code so we can look at it, in the main function:

```
break main
```

Now, tell it to run and it stops at the first line of main. This is the call to `init()`:

Ideally, we want to see what `init` is really up to. We can do that by typing

```
step
```

or

```
s
```

to tell the debugger we want to step into (follow the code through) this function:

Notice that now we're in the first line of `init`. The line `arr.count` is about to execute.
Let's take a look at some of these variables in the code. We can use `disp` to show them at each step:

```
disp arr.count
disp arr.capacity
disp arr
```

They'll display now with each command as long as we're in `init()`.
To advance to the next line of code, type

```
next
```

or

```
n
```

Notice that you've now set the array's count and are about to set its capacity:

Type another `next` and see where you are:

Now we're about to execute the closing bracket of `init()` (!!). When we type the next `n`, we'll be back in the test_int_array.cpp code, and these variables will be out of scope, so we won't see them.

Back in the test code, we have a loop to add numbers to our buggy `int_array`. Use `n` to advance to the next line of code, and type a number in at the prompt. When you come to the line that is about to execute the `add()` method, type `s` to step into that code.

From here, you can step through your first add to the array again. (You will need to re-display the important variables with `disp` to see them.)

For this week, just practice finding bugs with break, disp, next, and step. This code contains a bunch of errors that get past the compiler but cause failures. You don't have to find all of them; I tried to add all of the errors I have ever seen students make on this assignment, and that's pretty diabolical. If you can find 5 of the errors in the array code, describe them in your own words in comments, make the needed changes, and describe those in your own words in comments.

To get credit for this lab exercise, show the TA and sign the lab's completion log.

More info on GDB on the internet or this tutorial
http://www.cs.umd.edu/~srhuang/teaching/cmsc212/gdb-tutorial-handout.pdf