Introduction to Debugging C++ from the terminal using GDB

November 24, 2019

Paul DeTrempe ([detrempe@stanford.edu](mailto:detrempe@stanford.edu))

Hello and welcome to a crash course in debugging C++ code from the terminal for AA236A. This is meant to offer a quick introduction to debugging so you can write code that works (yay), and get a peak behind the curtain of your compiled executables. This tutorial assumes you are using a Linux system like Ubuntu (as pretty much everything happens from the terminal). With that being said, let’s get started. This tutorial is largely based on [this one](https://www.geeksforgeeks.org/gdb-step-by-step-introduction/) from Geeks for Geeks, but combined with some insight gained from Googling and StackOverflow.

1. First, open the terminal (Ctrl+Alt+T on PC).
2. Check if you’ve got GDB (GNU debugger) installed on your machine
   1. Type the following at the terminal
      1. $ gdb -help
   2. If the terminal outputs options for GDB, you’re good. Continue to 3.
   3. If output options did not display, you should update your Advanced Package Tool and install GDB using the following commands
      1. $ sudo apt-get update
      2. $ sudo apt-get install gdb
3. Congrats! At this point you either have GDB or have some serious googling to do, as your APT is weird (good luck). Typing gdb at the terminal should give info about the version, license, etc. Type quit to exit back to terminal after entering the gdb environment.
4. This next part assumes you will have written some kind of program in C++ that you want to debug. If not, go to the GeeksForGeeks tutorial linked in the first paragraph of this page and write up their “test.c” program (For us, this will be a program called “test.cpp”); the syntax for this simple program is the same in C and C++. Write it in whatever text editor you prefer (Notepad++, Sublime, Atom, CLion, etc.).
5. With the program written, we’re now going to compile it into a binary (that’s pretty neat).
   1. Navigate to the folder containing your program.
      1. If you don’t know how to navigate in the terminal, I highly recommend [Ubuntu’s “Command Line for Beginners” tutorial.](https://tutorials.ubuntu.com/tutorial/command-line-for-beginners#0)
   2. To compile the C++ program “your\_program.cpp” to an executable named “your\_executable.exe” type the following at the command prompt:
      1. $ g++ your\_program.cpp -o your\_executable.exe -ggdb
   3. It should take a few seconds to a few minutes to compile your program, so in the meantime, what did we just do? Let’s look at the commands individually:
      1. g++ Invokes a GNU C++ compiler (you can check your version with g++ --version
      2. -o generates an executable file. The file name of the executable is whatever follows -o
      3. -ggdb tells the compiler to put a bunch of debugging information (line numbers, variable names, etc.) for GDB into the executable.
      4. More insight on the g++ compiler can be found [here.](https://www.geeksforgeeks.org/compiling-with-g-plus-plus/)
   4. Now that we’ve compiled our program, let’s debug it!
      1. Start debugging in the GBD environment by typing the following command into the terminal
         1. $ gdb your\_executable.exe
      2. A bunch of version information should show up, and you’re now in the debugger. I *highly* recommend having your text editor open in parallel with your terminal to be able to look at lines of particular interest for debugging. Check you are in the debugger by typing the following in the GDB environment
         1. (gdb) list
            1. Side note: All GDB commands can also be abbreviated to their first letter (i.e. the above command can also be executed by typing the letter l)
         2. This should show the first 10 lines of your program.
      3. You can run the program using the following commands:
         1. (gdb) run
         2. (gdb) r
      4. You can set break points using the following commands:
         1. (gdb) break <line number>
         2. (gdb) b <line number>
      5. You can continue to the next break point using the following commands:
         1. (gdb) continue
         2. (gdb) c
      6. You can view set breakpoints and their status (enabled/disables) using the following commands:
         1. (gdb) info breakpoints
         2. (gdb) info b
         3. (gdb) i b
      7. You can disable breakpoints using the following command:
         1. (gdb) disable <breakpoint number>
   5. This should be plenty to get you started debugging, but if you want more shortcuts, cheatsheets can be found at the following links:
      1. <http://www.yolinux.com/TUTORIALS/GDB-Commands.html>
      2. <https://condor.depaul.edu/glancast/373class/docs/gdb.html>
   6. You can exit the GDB environment at any time by typing the following commands:
      * 1. (gdb) quit
        2. (gdb) q