Basic Debugging With GDB

By Ben Pearo

THE ESSENTIAL

- You must compile your code with the “-g” flag to use GDB on it
  - This is already done in the provided makefile
- Clone my repository!
  - git clone https://github.com/BenPearo/LearnGDB.git
  - or download the zip from that link (without “.git”)

Basic Debugging

The Commands

The commands we'll be using:

(All can be referenced by the prefix in brackets)

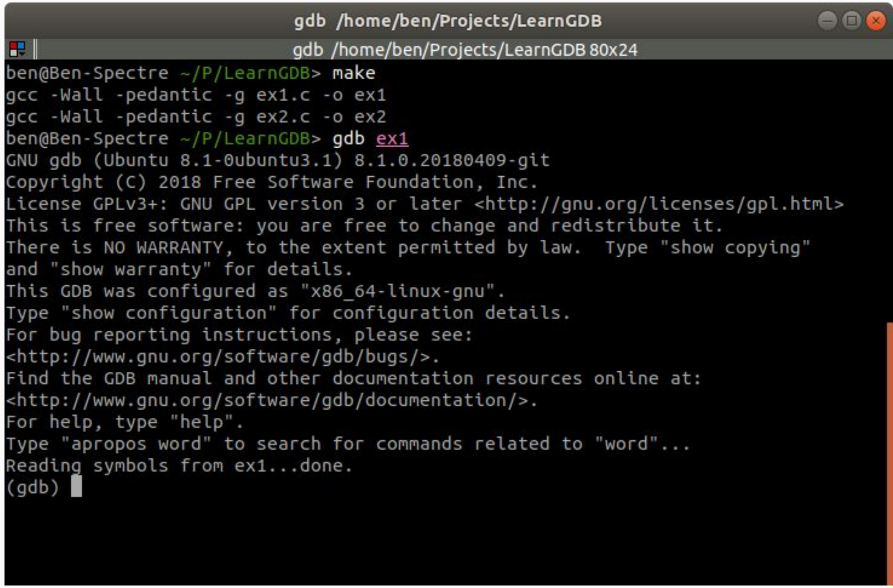
- Break (b)
  - Sets a breakpoint in your code
- Run
  - Starts your program
- Continue (c)
  - Continues to the next breakpoint
- Next (n)
  - Executes the current line and moves you to the next one

The Commands

- Step (s)
  - Allows you to step into a function
- Backtrace (bt)
  - Shows you the stack trace associated with your error
- Frame (f)
  - Allows you to see info associated with a specific frame
- Kill (k)
  - Stops a program after starting it with “run”
- Print (p)
  - Lets you print out any variable in the frame

How to Use GDB for debugging

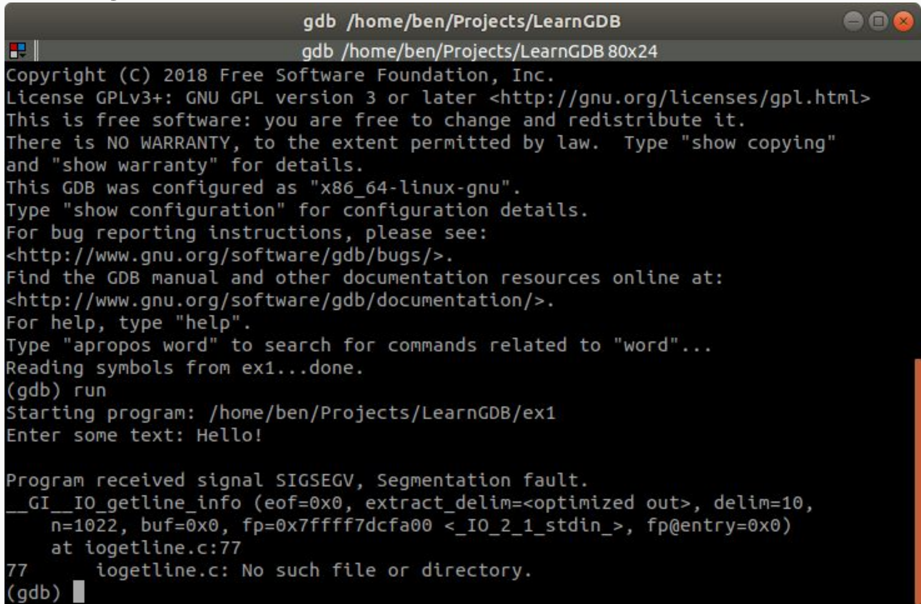
1. After compiling, use the the command “gdb *filename*”
   - gdb ex1


gdb /home/ben/Projects/LearnGDB
gdb /home/ben/Projects/LearnGDB 80x24

```
ben@Ben-Spectre ~/P/LearnGDB> make
gcc -Wall -pedantic -g ex1.c -o ex1
gcc -Wall -pedantic -g ex2.c -o ex2
ben@Ben-Spectre ~/P/LearnGDB> gdb ex1
GNU gdb (Ubuntu 8.1-0ubuntu3.1) 8.1.0.20180409-git
Copyright (C) 2018 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 ex1...done.
(gdb) 
```

How to Use GDB for debugging

2. Use "run" to run your program until you hit your error

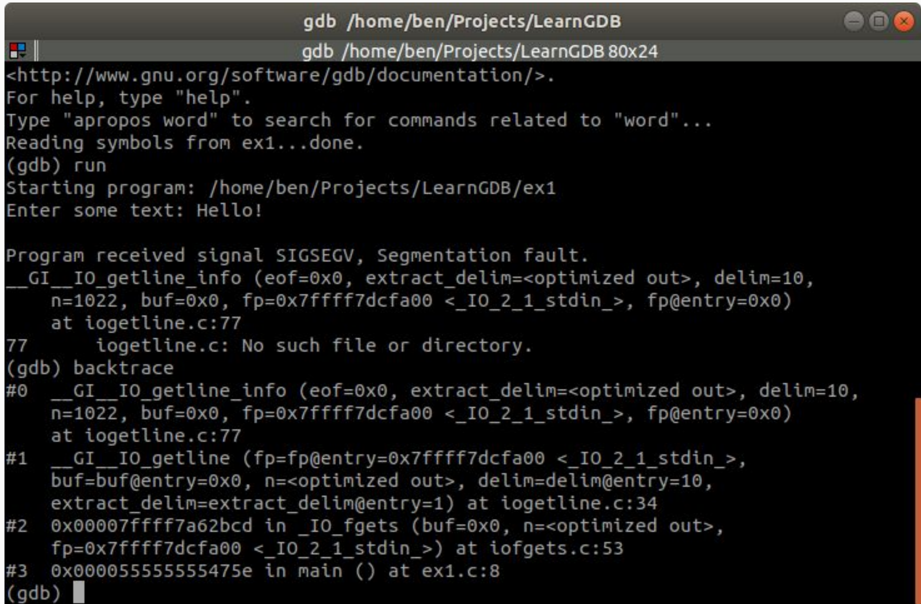

gdb /home/ben/Projects/LearnGDB
gdb /home/ben/Projects/LearnGDB 80x24

```
Copyright (C) 2018 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 ex1...done.
(gdb) run
Starting program: /home/ben/Projects/LearnGDB/ex1
Enter some text: Hello!

Program received signal SIGSEGV, Segmentation fault.
__GI__IO_getline_info (eof=0x0, extract_delim=<optimized out>, delim=10,
    n=1022, buf=0x0, fp=0x7ffff7dcfa00 <_IO_2_1_stdin_>, fp@entry=0x0)
    at iogetline.c:77
77      iogetline.c: No such file or directory.
(gdb) 
```

How to Use GDB for debugging

3. Use “backtrace” to see the stack frames associated with the error



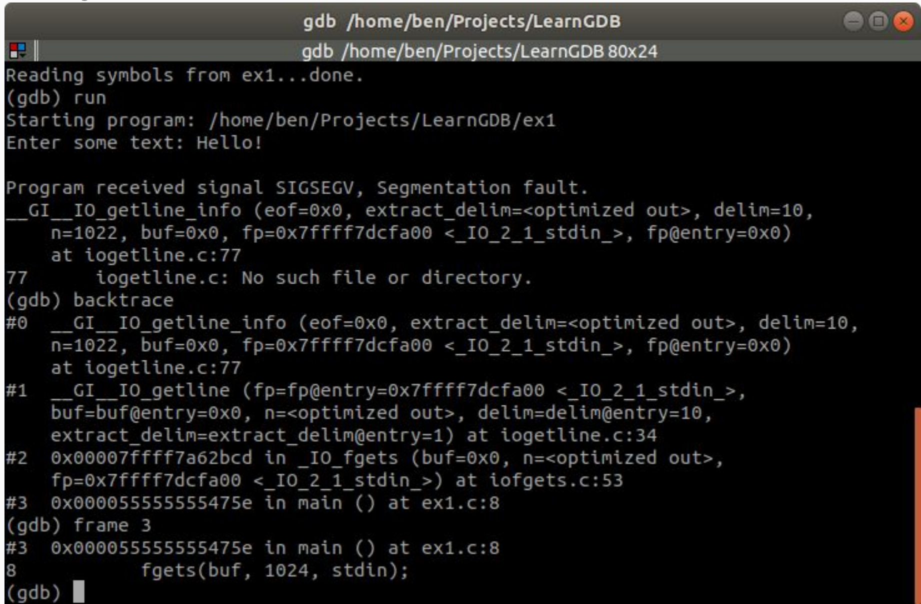
gdb /home/ben/Projects/LearnGDB
gdb /home/ben/Projects/LearnGDB 80x24

```
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ex1...done.
(gdb) run
Starting program: /home/ben/Projects/LearnGDB/ex1
Enter some text: Hello!

Program received signal SIGSEGV, Segmentation fault.
__GI__IO_getline_info (eof=0x0, extract_delim=<optimized out>, delim=10,
    n=1022, buf=0x0, fp=0x7ffff7dcfa00 <_IO_2_1_stdin_>, fp@entry=0x0)
    at iogetline.c:77
77      iogetline.c: No such file or directory.
(gdb) backtrace
#0  __GI__IO_getline_info (eof=0x0, extract_delim=<optimized out>, delim=10,
    n=1022, buf=0x0, fp=0x7ffff7dcfa00 <_IO_2_1_stdin_>, fp@entry=0x0)
    at iogetline.c:77
#1  __GI__IO_getline (fp=fp@entry=0x7ffff7dcfa00 <_IO_2_1_stdin_>,
    buf=buf@entry=0x0, n=<optimized out>, delim=delim@entry=10,
    extract_delim=extract_delim@entry=1) at iogetline.c:34
#2  0x00007ffff7a62bcd in _IO_fgets (buf=0x0, n=<optimized out>,
    fp=0x7ffff7dcfa00 <_IO_2_1_stdin_>) at iofgets.c:53
#3  0x000055555555475e in main () at ex1.c:8
(gdb) 
```

How to Use GDB for debugging

4. Look for an entry associated with one of your .c files, and use “frame *frame_number*”
    - In this case, we see frame #3
    - Choose it because it references “main.c”

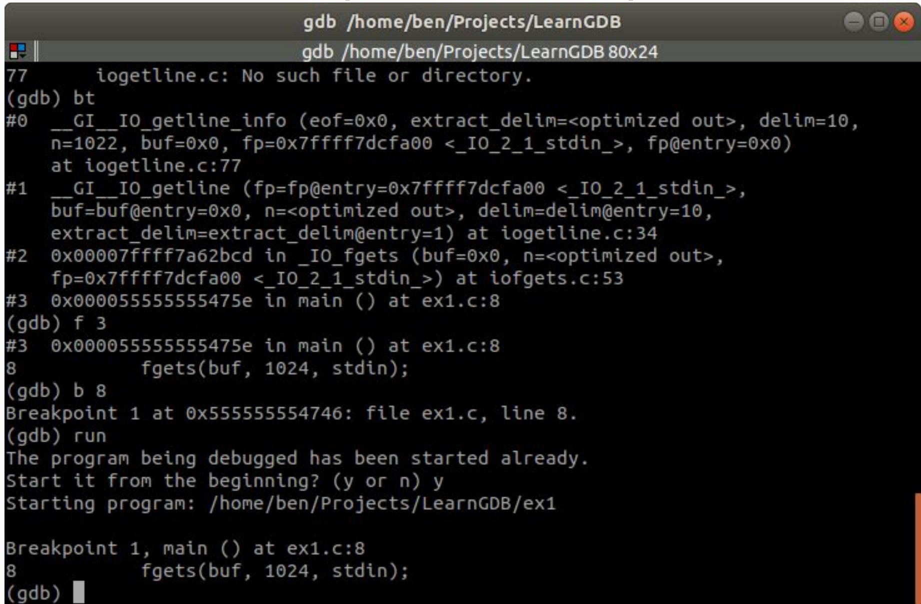

gdb /home/ben/Projects/LearnGDB
gdb /home/ben/Projects/LearnGDB 80x24

```
Reading symbols from ex1...done.
(gdb) run
Starting program: /home/ben/Projects/LearnGDB/ex1
Enter some text: Hello!

Program received signal SIGSEGV, Segmentation fault.
__GI__IO_getline_info (eof=0x0, extract_delim=<optimized out>, delim=10,
    n=1022, buf=0x0, fp=0x7ffff7dcfa00 <_IO_2_1_stdin_>, fp@entry=0x0)
    at iogetline.c:77
77      iogetline.c: No such file or directory.
(gdb) backtrace
#0  __GI__IO_getline_info (eof=0x0, extract_delim=<optimized out>, delim=10,
    n=1022, buf=0x0, fp=0x7ffff7dcfa00 <_IO_2_1_stdin_>, fp@entry=0x0)
    at iogetline.c:77
#1  __GI__IO_getline (fp=fp@entry=0x7ffff7dcfa00 <_IO_2_1_stdin_>,
    buf=buf@entry=0x0, n=<optimized out>, delim=delim@entry=10,
    extract_delim=extract_delim@entry=1) at iogetline.c:34
#2  0x00007ffff7a62bcd in _IO_fgets (buf=0x0, n=<optimized out>,
    fp=0x7ffff7dcfa00 <_IO_2_1_stdin_>) at iofgets.c:53
#3  0x000055555555475e in main () at ex1.c:8
(gdb) frame 3
#3  0x000055555555475e in main () at ex1.c:8
8           fgets(buf, 1024, stdin);
(gdb) 
```

How to Use GDB for debugging

5. Now that you know the line where the error occurred, place a breakpoint and rerun the program
    - b 8
    - run

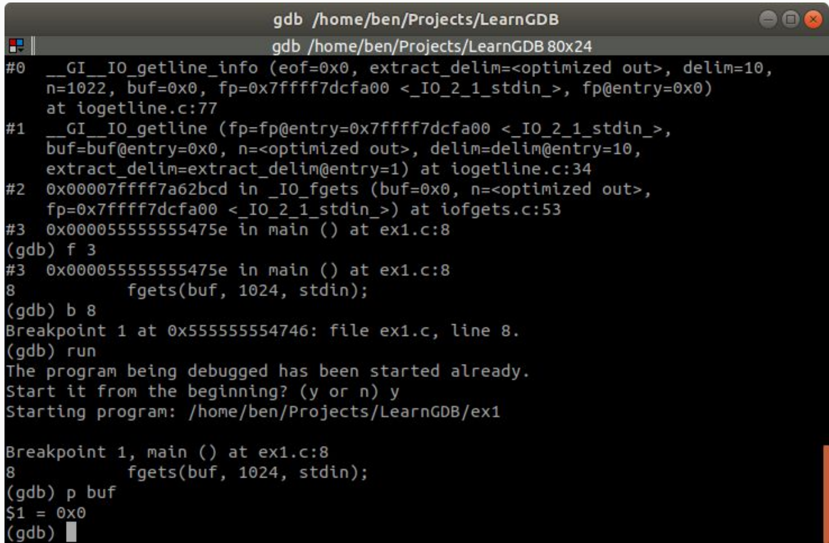

gdb /home/ben/Projects/LearnGDB
gdb /home/ben/Projects/LearnGDB 80x24

```
77      iogetline.c: No such file or directory.
(gdb) bt
#0  __GI__IO_getline_info (eof=0x0, extract_delim=<optimized out>, delim=10,
    n=1022, buf=0x0, fp=0x7ffff7dcfa00 <_IO_2_1_stdin_>, fp@entry=0x0)
    at iogetline.c:77
#1  __GI__IO_getline (fp=fp@entry=0x7ffff7dcfa00 <_IO_2_1_stdin_>,
    buf=buf@entry=0x0, n=<optimized out>, delim=delim@entry=10,
    extract_delim=extract_delim@entry=1) at iogetline.c:34
#2  0x00007ffff7a62bcd in _IO_fgets (buf=0x0, n=<optimized out>,
    fp=0x7ffff7dcfa00 <_IO_2_1_stdin_>) at iofgets.c:53
#3  0x000055555555475e in main () at ex1.c:8
(gdb) f 3
#3  0x000055555555475e in main () at ex1.c:8
8           fgets(buf, 1024, stdin);
(gdb) b 8
Breakpoint 1 at 0x555555554746: file ex1.c, line 8.
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/ben/Projects/LearnGDB/ex1

Breakpoint 1, main () at ex1.c:8
8           fgets(buf, 1024, stdin);
(gdb) 
```

How to Use GDB for debugging

6. Now you can check the values of your variables to detect the issue!
   - print buf
   - We can see that it's null
   - This could be the cause the seg fault


gdb /home/ben/Projects/LearnGDB
gdb /home/ben/Projects/LearnGDB 80x24

```
#0  __GI__IO_getline_info (eof=0x0, extract_delim=<optimized out>, delim=10,
    n=1022, buf=0x0, fp=0x7ffff7dcfa00 <_IO_2_1_stdin_>, fp@entry=0x0)
    at iogetline.c:77
#1  __GI__IO_getline (fp=fp@entry=0x7ffff7dcfa00 <_IO_2_1_stdin_>,
    buf=buf@entry=0x0, n=<optimized out>, delim=delim@entry=10,
    extract_delim=extract_delim@entry=1) at iogetline.c:34
#2  0x00007ffff7a62bcd in _IO_fgets (buf=0x0, n=<optimized out>,
    fp=0x7ffff7dcfa00 <_IO_2_1_stdin_>) at iofgets.c:53
#3  0x000055555555475e in main () at ex1.c:8
(gdb) f 3
#3  0x000055555555475e in main () at ex1.c:8
8           fgets(buf, 1024, stdin);
(gdb) b 8
Breakpoint 1 at 0x555555554746: file ex1.c, line 8.
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/ben/Projects/LearnGDB/ex1

Breakpoint 1, main () at ex1.c:8
8           fgets(buf, 1024, stdin);
(gdb) p buf
$1 = 0x0
(gdb) 
```

How to Use GDB for debugging

So now we see that the buffer somehow wasn’t set. If you check the code, you’ll see that the malloc is very wrong and potentially the cause of the error. If you change it to a value like 1024, the program will work fine.

ADVANCED PRINTING

- Type “make” in the terminal if you haven’t already
- Open example 2
  - gdb ex2
- Place a breakpoint at the end of main
  - b 33
- Run the program
  - run
- Let’s get printing

ADVANCED PRINTING

- Check out the value of any variable
    - “p node1” will show you the address stored in the pointer named node1
- You can dereference pointers in a familiar way
    - “p *node1” will show you the contents of the node1
- You can even use arrows!!
    - “p *node1->next” will show you the contents of node2

ADVANCED PRINTING

- Finally, typecasting
  - void_data holds the address of a string, but gdb doesn’t know that, so printing it normally won’t work
  - Try “p node1->void_data”
  - Aw, just an address?
  - Just typecast it to char*!
  - “p (char*)node1->void_data” will show you the string

Additional Notes

- To use command line arguments, use *gcc --args ex1 arg1 arg2 arg…*
- The editor CLion has both GDB and Valgrind built in. A very powerful tool available for all operating systems.
- Go to your labs!!!