GDB Debugger

CS 211 – Programming Practicum

GDB Debugger

- Part of the GNU Software Tools
- Many Debuggers in IDEs are just Wrappers for GDB
- Huge amount of commands in GBD, many options available to help debug your program
- Even to most basic knowledge can save you lots of time

GDB Debugger

- Step 1 – Compile your program using the –g flag
    - gcc –g mazeflawed.c
- The –g flag creates additional information for the executable that gdb used to convert machine code lines back to the source code line of the original program

GDB Debugger

- Step 2 - Open the GDB debugger
    - gdb a.out
- Give the name of the executable created by the compiler

GDB Debugger

- Step 3 – Run your program with any command line arguments
    - run <command-line-arguments>
- If no command line arguments are needed, just type in run.
- For the mazeflawed.c program, you need the datafile name, so:
    - run mazedata1.txt

GDB Debugger

- Step 4 – Let GDB tell you on which line the Segmentation Fault occurs

Program received signal SIGSEGV, Segmentation fault.

0x0000000004007a9 in main (argc=2, argv=0x7fffffffca78) at mazeflawed.c:52

52 m1.arr[i][j] = '.';

- The above states the Segmentation fault occurred at line 52.
- It also shows the code at line 52

GDB Debugger

- Step 5 – The **list** command will display more lines of code

- Use the **help** command to find out about more commands.
- GDB has so many commands that no one knows them all.
- Most people learn a few key commands.

GDB Debugger

- Step 6 – The **print** command will display values stored in variables

```
52          m1.arr[i][j] = '.';

print i
$1 = 14
```

- The **print i** command shows that the variable i contains the value of 14 at the time of the segmentation fault.

GDB Debugger

- To work with breakpoints, type

  help breakpoints

- To see all of the commands that work with breakpoints.
- Setting a breakpoint causes the program's execution to stop when that line number is reached or when that function is called.
  - break 51 — stops when line 51 is reached
  - break pop — stops when the pop function is called

GDB Debugger

- While at a breakpoint, use **print** to see the values stored in variables

- When at a breakpoint, type continue to resume execution

- Type: **delete breakpoints** to remove all breakpoints

Or

- Type: **delete breakpoints 51** to remove the breakpoint at line 51

GDB Debugger

- Type **quit** to exit gdb

GDB Debugger

- When a segmentation fault occurs in a library function, you can get strange results from gdb.
- In this case, using the **backtrace** command to view the stack is helpful

- The linkedflawed.c program shows how backtrace is useful.

- Running linkedflawed.c in gdb shows a SIGABRT signal
  - The output is not much help

GDB Debugger

- Running linkedflawed.c in gdb shows a SIGABRT signal
- Typing the **backtrace** command, you get:

#0 0x000000309fe2ffc5 in raise () from /lib64/libc.so.6

#1 0x000000309fe31a70 in abort () from /lib64/libc.so.6

#2 0x000000309fe6994b in __libc_message () from /lib64/libc.so.6

#3 0x000000309fe714af in _int_free () from /lib64/libc.so.6

#4 0x000000309fe757ab in free () from /lib64/libc.so.6

#5 0x0000000000400657 in pop (hd=0x601110) at linkedflawed.c:44

#6 0x000000000040074f in main (argc=1, argv=0x7fffffffe1b8) at linkedflawed.c:78

GDB Debugger

#0 0x000000309fe2ffc5 in raise () from /lib64/libc.so.6

#1 0x000000309fe31a70 in abort () from /lib64/libc.so.6

#2 0x000000309fe6994b in __libc_message () from /lib64/libc.so.6

#3 0x000000309fe714af in _int_free () from /lib64/libc.so.6

#4 0x000000309fe757ab in free () from /lib64/libc.so.6

#5 0x0000000000400657 in pop (hd=0x601110) at linkedflawed.c:44

#6 0x000000000040074f in main (argc=1, argv=0x7fffffffe1b8) at linkedflawed.c:78

- What does this mean??????????

GDB Debugger

#0 0x000000309fe2ffc5 in raise () from /lib64/libc.so.6

#1 0x000000309fe31a70 in abort () from /lib64/libc.so.6

#2 0x000000309fe6994b in __libc_message () from /lib64/libc.so.6

#3 0x000000309fe714af in _int_free () from /lib64/libc.so.6

#4 0x000000309fe757ab in free () from /lib64/libc.so.6

#5 0x0000000000400657 in pop (hd=0x601110) at linkedflawed.c:44

#6 0x000000000040074f in main (argc=1, argv=0x7fffffffe1b8) at linkedflawed.c:78

- main( ) called the function pop( ) at line 78

GDB Debugger

#0 0x000000309fe2ffc5 in raise () from /lib64/libc.so.6

#1 0x000000309fe31a70 in abort () from /lib64/libc.so.6

#2 0x000000309fe6994b in __libc_message () from /lib64/libc.so.6

#3 0x000000309fe714af in _int_free () from /lib64/libc.so.6

#4 0x000000309fe757ab in free () from /lib64/libc.so.6

#5 0x0000000000400657 in pop (hd=0x601110) at linkedflawed.c:44

#6 0x000000000040074f in main (argc=1, argv=0x7fffffffe1b8) at linkedflawed.c:78

- pop( ) called the library function free( ) at line 44

GDB Debugger

#0 0x000000309fe2ffc5 in raise () from /lib64/libc.so.6

#1 0x000000309fe31a70 in abort () from /lib64/libc.so.6

#2 0x000000309fe6994b in __libc_message () from /lib64/libc.so.6

#3 0x000000309fe714af in _int_free () from /lib64/libc.so.6

#4 0x000000309fe757ab in free () from /lib64/libc.so.6

#5 0x0000000000400657 in pop (hd=0x601110) at linkedflawed.c:44

#6 0x000000000040074f in main (argc=1, argv=0x7fffffffe1b8) at linkedflawed.c:78

- free( ) gets into trouble ➔ something wrong with the parameter

GDB Debugger

#0 0x000000309fe2ffc5 in raise () from /lib64/libc.so.6

#1 0x000000309fe31a70 in abort () from /lib64/libc.so.6

#2 0x000000309fe6994b in __libc_message () from /lib64/libc.so.6

#3 0x000000309fe714af in _int_free () from /lib64/libc.so.6

#4 0x000000309fe757ab in free () from /lib64/libc.so.6

#5 0x0000000000400657 in pop (hd=0x601110) at linkedflawed.c:44

#6 0x000000000040074f in main (argc=1, argv=0x7fffffffe1b8) at linkedflawed.c:78

- free( ) takes a pointer as a parameter. Question #1 is it a NULL pointer?

GDB Debugger

#0 0x000000309fe2ffc5 in raise () from /lib64/libc.so.6

#1 0x000000309fe31a70 in abort () from /lib64/libc.so.6

#2 0x000000309fe6994b in __libc_message () from /lib64/libc.so.6

#3 0x000000309fe714af in _int_free () from /lib64/libc.so.6

#4 0x000000309fe757ab in free () from /lib64/libc.so.6

#5 0x0000000000400657 in pop (hd=0x601110) at linkedflawed.c:44

#6 0x000000000040074f in main (argc=1, argv=0x7fffffffe1b8) at linkedflawed.c:78

- Question #1: Is it a NULL pointer? No, parameter is not NULL (0x00)