UF UNIVERSITY of FLORIDA
Herbert Wertheim College of Engineering

COP4600

Ex3: Console Debugging

Overview

In some cases, systems-level code can be hard to debug; pointer errors can result in hours of hair pulling and endless searching to solve simple issues that are obvious in professional development suites. Thankfully, The GNU Debugger (GDB) is here to rescue you! In this exercise, students will learn about the powerful toolset that GBD You will analyze, deconstruct, and debug given programs to help you learn the necessary skills of using a console debugger that will help you with future projects, and any code that you will write. At the end of the exercise, you'll submit several annotated screenshots. This exercise should be completed in Reptilian.

Structure

This exercise will follow this basic structure:

1) Install **GDB 9.1** and its utilities
2) Research on how to utilize GDB and its commands, a helpful link will be provided
3) Create a simple program, and play around with GDB's features
4) Debug ***`process.c`*** and explain the issue encountered / what it was trying to do
5) Analyze ***`password`*** by digging through its code in order to obtain the key

Installation (GDB)

Within Reptilian, use the **`apt-get`** command to update the Linux package list, and install GDB

```
$ sudo apt update
$ sudo apt install gdb
```

Test GDB to make sure that it's the correct version by prompting it:

```
$ gdb -version
```

```
reptilian@localhost:~$ gdb --version
GNU gdb (Ubuntu 9.1-0ubuntu1) 9.1
Copyright (C) 2020 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.
```

Your output should look something like this

Using the GNU Debugger (GDB)

Using the debugger is straightforward. To start up the debugger on a sample program ***myProgram*** we would type:

```
$ gdb myProgram
```

From there, we can learn more about GDB and how to use it by keying:

```
(gdb) help
```

We highly suggest you reference the bottom link to the GDB manual, and the help command if you ever get stuck, or want to learn more about its features. Using the debugging tool follows a simple guideline that will be outlined below:

- Run the program and analyze what it's doing
- Set up necessary breakpoints, and step through the program
- Find the issues utilizing the various tools that GDB offers

Running

To run the code, simply key in:

```
(gdb) run
```

This will run the code as your normal shell would, except it will point out several helpful debugging hints whenever you compile your program with the "**-g**" flag.

```
(gdb) run
Starting program: /mnt/c/Users/sample
```

Breaking

After running the code, you may choose an address, or function name in which you want to place your breakpoint:

```
(gdb) break main
```

This will place a breakpoint inside the program at the address of main, and from there, you will be able to step through the code, or run whatever code the method contains

```
(gdb) break main
Breakpoint 1 at 0x40052e: file sample.c, line 5.
(gdb) run
Starting program: /mnt/c/Users/sample

Breakpoint 1, main () at sample.c:5
5           int a = 1;
```

Stepping

The most useful command next to creating breakpoints, stepping allows the user to step through each line of code to find what the outputs are, and what possible issues the code contains:

```
(gdb) step
```

After stepping through the program, you may delete the breakpoints that are not necessary to you

```
(gdb) delete
```

Useful Commands

- **run**
- **break**
- **step**
- **delete**
- **next**
- **up**
- **down**
- **print**
- **info**
- **quit**

Application

Now that we have learned some basic commands, it's time to apply our newfound knowledge to debug some programs that you will make, and that we provide

Part 1 – Simple Program

Create your own ***"Hello World"*** program and play around with the capabilities of GDB. When you're done, take a screenshot of your program being run in GDB with a breakpoint being set in **main**. **Run the program to completion.** This means that your screenshot will show the return code/message from gdb.

Compiling your program uses these commands:

```
$ gcc -o myProggy.o myProggy.c
$ gcc -g -o myProggy.o myProggy.c
```

Part 2 – Process Table

Download the ***process.c*** code from the source folder of the exercise and compile it with the commands given at the bottom of this page. This program emulates a (very) rudimentary process table, however, there's a catch. Inside the program is a bug, and you must use the GDB to find the issue.

When you're done, you're going to take a screenshot of the issue in GDB, explain why it broke the code (in a canvas submission comment), fix the code (only one line), and submit another screenshot of the working code's output.

To compile ***process.c***:

```
$ gcc -w -g -o process process.c -lpthread -lrt
```

Part 3 – Password

Download the source file ***password*** from the same folder as ***process.c***. As you will notice, this file is an output, and it was compiled **without** debugging symbols. In order to complete this part, you will need to use GDB to find a password that is hidden WITHIN the program itself (you should start by decompiling that code). Look into the ***info*** of the program to see what it contains. See if you can track its motion through compilation. You will take a screenshot of **where** you found the password **before** decoding it, and the message received **after** typing in the correct password.

Hint: Use a HEX translator, like the one provided at the bottom of the page

Submissions

You will submit the following files for this assignment:

Part 1:

- A screenshot of your program being run **to completion** with a breakpoint set in `main` (**`breakpoint.png`**)

Part 2:

- A screenshot of the issue in GDB (**`gdb.png`**)
- A screenshot of the output of the code after it was fixed (**`output1.png`**)
- A comment on the canvas submission explaining what the issue was

Part 3:

- A screenshot of where you found the password (**`password.png`**)
- A screenshot of the output after putting in the password (**`output2.png`**)

Helpful Links

```
(gdb) step
6           int b = 2;
(gdb)
7           int c = a + b;
(gdb)
9           printf("%d\n", c);
```

```
(gdb) delete
Delete all breakpoints? (y or n) y
```

You may also delete a specific breakpoint by specifying the name or address

https://ftp.gnu.org/old-gnu/Manuals/gdb/html_node/gdb_toc.html
https://darkdust.net/files/GDB%20Cheat%20Sheet.pdf
https://www.rapidtables.com/convert/number/hex-to-ascii.html