Exploring GDB – GNU Debugger.

To compile:

```
$ gcc -o hello hello.c
```

./ => execute.

To compile with symbols:

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

To debug:

```
$ gdb -q ./hello
```

GDB support short words for lengthy commands.

For example:

(gdb) list {} — used to list your code from line {}

1) l
2) li
3) lis
4) list

While learning, always write full commands. (X)

Viewing disassembled code:

(gdb) disassemble <func-name/addr>

↳ To disassemble section

(gdb) info registers → while running it will show all available registers.

(gdb) info breakpoint → View bps.

(gdb) break {line}/*{addr} - Set bp

Structure of a compiled binary

1) Text section — The code (ASM)
   - start → Entry point
     ↳ main()
2) Data section } Variables
3) BSS section. }
4) Heap → Dynamic storage
5) Stack → Control flow.

What is a breakpoint:

It is a address or line number where the execution gets paused so that we can look into the current status of the execution.

The run command:

(gdb) run - Runs the code
It gets stopped at the breakpoint.

(gdb) continue - Continues execution until next breakpoint comes.

(gdb) next - Unlike continue, it will only execute the next line. We can use this to execute line by line.

(gdb) nexti - Unlike next, it will only execute next assembly instruction.

The examine command : (x)

x/{N}{U}{F} {exp}

N - Numer of units

U - Unit

- b - byte
- h - 2 bytes
- w - 4 bytes
- g - 8 bytes.

f - format

- x - hexa
- d - signed deim
- u - u.s decimal
- o - octal
- t - binary
- a - addr

cntd...

f - format ① ②

cntd...

- c - char
- f - float
- s - string
- i - Instruct (ASM)

The print command:

print {format} {exp}

i → value
&i → addr

& - addr of

=> It generates in-memory variables in numerical order for every print command call, which can be then used by the examine (x) command.

$1
$2
$3
...

x/f $1