

Notes for learning gdb

Author : Vikas Nagpal (<https://github.com/vikasnagpaliitd>)

Version 1.0

Segments in a running process

- Stack : local variables, arguments (Life: till function returns)
- Heap : dynamically allocated memory // ptr = malloc(200) (Life : till free(ptr) is called)
- Data Segment : Global variables, or static variables (Life == life of process)
- Code Segment : machine code , .text : Read only

It matters because scope and lifetime of variables depend on it

We must understand function call “stack” and “stack frame”

Example code for discussion:

```
int *func1();
main()
{
    int *ptr = func1();
    printf("ptr[1] = ptr[1]);
    free(ptr)
}
int *func1()
{
    // int arr[10] ={3,4,5};
    int *ptr;
    ptr =malloc(sizeof(int) * 10);
    //fill values
    return arr;
}
```

Compiling for gdb

```
$ gcc -g main.c func.c
```

-g adds an enhanced symbol table in the object code

Running gdb

```
$ gdb ./a.out
```

// gdb process is the parent of ./a.out process

you get gdb) prompt

```
gdb) break <fn_name>
gdb) break <line num>
gdb) break <filename>:<linenum>
del <break_num>
print <expr>
```

```
info locals
```

```
info args
```

```
gdb) run
```

```
gdb ) continue
```

Running step by step

- set breakpoint at main()
- run
- use step and next
- use p (print) to print variables and expressions
- info locals
- info args

Can we put the breakpoint a bit deeper at the suspected function?

Taking help in gdb

How to debug a crash (core dump)?

How to intentionally generate core dump (and why)?

`gdb) gcore`

gdb in multi threaded process

Do different threads share or have separate stacks? What about heap?

`gdb) info threads`

`gdb) thread <thread num>`

How to print stack trace of different threads