Debugging: is a science and art for finding and eliminating bugs in a computer program.

[GDB Installation steps](https://www.javatpoint.com/gdb)

$ man gdb

First we need to compile to program in order to use gdb for debugging.

$ gcc -g ex1.c -o ex1

<-g> it instructs the compiler to keep the debugging symbol in object file.

$ gcc -ggdb ex1.c -o ex1

For gdb particularly

$ gdb

To open gdc prompt

If we want to use shell commands in gdb prompt we need to use band sign ‘:’

(gdb) :clear

(gdb) file ex1

To load the compiled code to gdb

(gdb) run

To run the program

(gdb) q

To come out from gdb

$ gdb ex1

To open gdb prompt with directly loading executable file

$ gdb -d

To load gdb prompt directly without showing any information.

(gdb) attach <Process ID>

To attach which is currently running.

(gdb) info registers

To get it information of registers which are currently in use by that process id.

(gdb) info inferiors

To check how many programs loaded.

(gdb) add-inferior -exec ex2

To load more files.

(gdb) inferior 2

To change the focus or switching to another program.

(gdb) help

(gdb) help break

To get help of a particular command

(gdb) watch num

It will display the modifications done on num variable

(gdb) list

To list the code

$ gdb -d ex3

(gdb) break main

(gdb) break 6

To add break point at that function or line number of code

(gdb) break ex2.c:8

To add break point in different file or multi file program

(gdb) run 5

To run program which needs arguments

‘c’ will continue the program till next break point or till end.

‘n’ execute next line as single instruction. After one ‘n’ we can simply press ENTER to execute next line.

(gdb) info break

To check if there are any break points.

(gdb) delete 1

To remove break points

(gdb) disable 3

To disable break pint temporarily

(gdb) print i

To print the value in variable.

(gdb) print /x i

(gdb) print /o i

(gdb) print /t i

(gdb) whatis i

To check the data type of I variable

(gdb) set variable i = 5

(gdb) set (i=5)

To change the value in variable

Core Dumped: when ever out process abnormally terminates on receiving some signals like segmentation violation, floating point error, zero divisibility error, illegal instruction. Linux creates core file which is the binary image of the program at the time of its death.

If the core file is not present, we might have set ulimit to zero.

$ ulimit -c unlimited

Now we can see core file is generated when the program terminates. But core file size is bit high when ever our code terminates core file keep on generated so its always better to keep ulimit to zero.

$ gdb -q ex5 core

Now we can see at what line there is abnormality is present.

$ gdb -q exe1

(gdb) list

(gdb) break ex1.c:main

(gdb) run

(gdb) disassemble main

To get the assembly of main function. By default flavor of assembly will be AT&T

(gdb) disassemble /m main

To get source as well as assembly at same time

(gdb) info registers

To get the values of integer registers

Note for registers:

rip – is a instruction pointer which contains address of next instruction to be executed. Before a call value of instruction pointer is pushed on the process stack and the control goes to the function and executes once it gets returned the instruction of leave it role back to stack to the previous location and value which is pushed inside the stack is popped inside this register and controller goes back. “rip in 64 bit and eip in 32 bit pc’s”.

rbp – is a base pointer always points to bottom of the current stack frame.

rsp – this register always points to the top of current stack frame.

(gdb) print /x $rax

To get the register value

(gdb) set $rax += 9

To change the value of variable inside register

(gdb) backtrack

To get information of different stack frames. ‘bt’ short cut for backtrack.

(gdb) finish

To execute a function return the value to previous function and stop over there. Ex: for f4() to f3()

(gdb) info all-registers

To get all registers

$ gdb -q -tui stackdemo

(gdb) layout split

To see both high level language and assembly

(gdb) n

To move next line.







