GDB Debugger

- **Multiple Platform**
  - x86
  - ARM
  - Opensource and closed source binaries.

- **Uses of GDB :**
  - Runtime Analysis
  - Manipulating Program Flow
  - Disassembly
  - Reverse Engg.

- **Debuggers Symbol :**
  - Information about Variables ,functions etc. About he binary which can be read by a debugger
  - Debugger now Understands the binary better.
  - It can be included in the binary , if not then it can be parsed seperatly . Symbol file types are **Dwarf 2,COFF,XCOFF,Stabs**
  - GCC uses `-g` option
  - GCC `-ggbd` for **GDB Specific Symbols**

NM -(List Symbols from Object Files)

`nm 'File_name'`

- Output :

```
08048474 T AddNumbers
0804a028 B IamAGlobalVariable
08048491 T SubtractNumbers
08049f28 d _DYNAMIC
08049ff4 d _GLOBAL_OFFSET_TABLE_
0804863c R _IO_stdin_used
         w _Jv_RegisterClasses
08049f18 d __CTOR_END__
08049f14 d __CTOR_LIST__
```

- Symbol Types :

Symbol TYPE	Meaning
A	Absolute Symbol
B	In the Uninitialized Data Section (BSS)
D	In the Initialized Data Section
N	Debugging Symbol
T	In the Text Section
U	Symbol Undefined right now

- **Lower case is Local Symbol**
- **Upper case is External**

- Commands:
  - `nm 'File_name' |grep 'function_name'` (will find the symbole in the binary).
  - `nm ./* |grep 'function_name'` (will find the symbol in every File in the given Direct..).
  - `nm -n 'File_name'`(Sort the output)

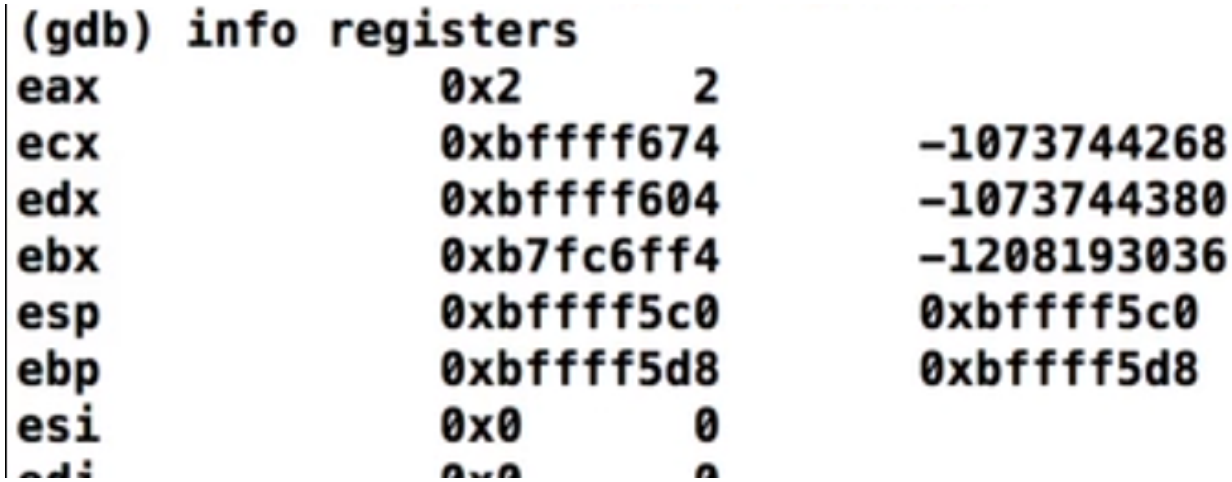
Strace :

- Helper tooll to understand how program interacts with the os.
- Traces all system calls made by the program
- `strace 'File_name'`

Modifying Registers and Memory :

- Commmands :
  - `list` (list the src file)
  - `list 1`(List from the first line)
  - `info functions` (Will give you all the functions)
  - `info sources` (will give you info about sorc file)
  - `info variables`(will give info about variables but by default it does'nt print local variables **{only global variables}**)
  - `info scope`(Will list all the available scopes)
  - `info scope "scope name"` (list Scope of the scopename)

- `objcopy --only-keep-debug 'File-name' 'Output _File'` (Separete Command from GDB Will Copy Only Symbols from binary and copy to a separate file)
- `strip --strip-debug --strip-unneeded 'File_name'`(Will remove debug symbols from a binary)
- `symbol-file 'Debug_File'`
- `run 'Arguments'` (Run the program in GDB With command line Args if any).
- `break main`(Set a Breakpoint at main Function)
- `info registers` (Information of CPU Registers)



- `info breakpoints` (List all break Points)
- `disable 'breakpoint number'` (Disable Selected breakpoint)
- `enable 'breakpoint number'`(enable Breakpoint)
- `delete 'breakpoint number'`(delete breakpoint)
- `x/s argv[1]`(Examine the second argument passed)
- `x/s argv[0]`(Examine the first argument passed)
- `x/i 'Address'`(Examine a Instruction at a address Location)
- `x/3i,x/10i` (can also specify how many instructionn you want 3 or 10).
- `x/10xw $esp` (x means hex , w for word , $esp is address of esp register which points at stack).
- `set $eax = 1` (Set value to registers)
- `set variable p = ' '` (Set any value to any variable)
- `stepi` (to come 1 step at a time after a brekpoint)
- `disassemble main` (show assembly code)

- GDB TUI Mode :
  - `gdb -q ./hello -tui`(start gdb)
  - `layout asm` (Add Assembly to the layout)

- `layout regs`(add registers to the layout)
- `stepi`

(Gdb tui is best to use and easy)