

# GDB TUI Mode - Detailed Notes

## What is GDB TUI Mode?

GDB's **Text User Interface (TUI)** mode provides a **graphical-like debugging interface in the terminal**. It helps visualize:

- **Source code**
  - **Assembly instructions**
  - **Registers and memory**
  - **Breakpoints and variables**
- 

### ♦ 1. Enabling TUI Mode

You can enter TUI mode in **three** different ways:

#### 1 Start GDB in TUI Mode Directly

```
gdb -tui program_name
```

Example:

```
gdb -tui my_program
```

#### 2 Enable TUI Mode Inside GDB

If you're already in GDB:

```
(gdb) layout src
```

or

```
(gdb) tui enable
```

#### 3 Start GDB with Arguments

```
gdb -tui --args ./my_program arg1 arg2
```

---

### ♦ 2. Switching Between Layouts

GDB TUI provides multiple **layouts** that you can switch between.

Command	Description
layout src	Show <b>source code</b> window
layout asm	Show <b>assembly</b> instructions
layout regs	Show <b>registers</b>
layout split	Show <b>source + assembly</b>
layout next	Cycle through different layouts
update	Refresh the display

💡 **Example:**

```
(gdb) layout asm
```

This switches the view to assembly instructions.

---

### ♦ 3. Changing Focus Between Windows

When using TUI mode, you can **change focus** between different windows (source, registers, assembly) using:

Shortcut	Action
Ctrl + X → 0	<b>Cycle through TUI windows</b>
Ctrl + X → A	<b>Toggle TUI mode on/off</b>
Ctrl + X → 1	Focus <b>source code window</b>
Ctrl + X → 2	Focus <b>assembly window</b>
Ctrl + X → 3	Focus <b>registers window</b>

💡 **Example:**

Press Ctrl + X, then 0 to switch between different sections of TUI mode.

---

### ♦ 4. Navigating Inside Windows

Shortcut	Action
Up/Down	Scroll source/assembly code
PgUp/PgDn	Scroll <b>faster</b>
Ctrl + L	Refresh display
tui disable	Disable TUI and return to CLI

💡 **Example:**

Press PgUp or PgDn to scroll through the **source code window**.

---

### ♦ 5. Setting & Managing Breakpoints

A **breakpoint** pauses execution at a specific point in the program.

```
(gdb) break main
```

or

```
(gdb) break 15 # Break at line 15
```

To **remove** a breakpoint:

```
(gdb) delete 1 # Removes breakpoint number 1
```

---

### ♦ 6. Running and Stepping Through Code

Command	Description
run / r	Start execution
continue / c	Resume after a breakpoint
next / n	Step to the next line (without entering functions)
step / s	Step <b>inside functions</b>
finish	Execute until the function returns
until 20	Run until <b>line 20</b>

💡 **Example:**

```
(gdb) run
(gdb) next
```

## ♦ 7. Inspecting Variables

Command	Description
print var	Print <b>value</b> of var
display var	Continuously show var in TUI
info locals	Show all <b>local variables</b>
info registers	Show <b>CPU registers</b>

💡 **Example:**

```
(gdb) print x
$1 = 10
(gdb) display x
```

## ♦ 8. Examining Memory

Command	Description
x/10xw 0xaddr	Examine memory at 0xaddr
x/s var	View a string variable

💡 **Example:**

```
(gdb) x/10xw 0x555555554000
```

This prints **10 words** from memory at address 0x555555554000 .

## ♦ 9. Exiting GDB TUI Mode

Command	Description
quit / q	Exit GDB
Ctrl + D	Exit GDB

Command	Description
tui disable	Exit TUI Mode only

💡 **Example:**

```
(gdb) quit
```

---

## ◆ 10. Full Example Debugging Session

```
gdb -tui ./a.out
(gdb) break main
(gdb) run
(gdb) layout split
(gdb) next
(gdb) print x
(gdb) info registers
(gdb) continue
(gdb) quit
```

---

## ◆ 11. Example Output in TUI Mode

```
[Source Code Window]
10  int main() {
11      int x = 5;
12      x = x + 10;  // Breakpoint here
13      return 0;
14  }

(gdb) print x
$1 = 5
```

---

```
gdb -q -tui ./HelloW -D/S/V/Video11

--Register group: general--
rax      0x0      0      rbx      0x0      0
rcx      0x0      0      rdx      0x0      0
rsi      0x0      0      rdi      0x0      0
rbp      0x0      0x0    rsp      0x7fffffff530  0x7fffffff530
r8        0x0      0      r9        0x0      0
r10       0x0      0      r11       0x0      0
r12       0x0      0      r13       0x0      0
r14       0x0      0      r15       0x0      0
rip      0x401000  0x401000 <_start>
cs        0x33     51      eflags    0x202      [ IF ]
ss        0x2b     43

B> 0x401000 <_start> mov al,0x1
0x401002 <_start+2> mov edi,0x1
0x401007 <_start+7> movabs rsi,0x402000
0x401011 <_start+17> mov edx,0x28
0x401016 <_start+22> syscall
0x401018 <_start+24> mov eax,0x3c
0x40101d <_start+29> mov edi,0x2
0x401022 <_start+34> syscall
0x401024 add BYTE PTR [rax],al
0x401026 add BYTE PTR [rax],al
0x401028 add BYTE PTR [rax],al

native process 25630 (cmd) In: _start L10 PC: 0x401000
Reading symbols from ./HelloWorld...
(gdb) b _start
Breakpoint 1 at 0x401000: file HelloWorld.nasm, line 10.
(gdb) r
Starting program: /home/tyrell/Desktop/SLAE/Videos/Video11/HelloWorld

Breakpoint 1, _start () at HelloWorld.nasm:10
(gdb) layout asm
(gdb) set disassembly-flavor intel
(gdb) layout asm
(gdb) layout reg
(gdb)
```

```
gdb -q -tui ./HelloW -D/S/V/Video11

Registers

[ Register Values Unavailable ]

0x401022 <_start+34> syscall

native No process (cmd) In: L?? PC: ??
breakpoints -- Making program stop at certain points.
data -- Examining data.
files -- Specifying and examining files.
internals -- Maintenance commands.
obscure -- Obscure features.
running -- Running the program.
stack -- Examining the stack.
status -- Status inquiries.
support -- Support facilities.
--Type <RET> for more, q to quit, c to continue without paging--Quit
(gdb) stack
Undefined command: "stack". Try "help".
(gdb)
```

## Pro Tips

- ✓ Use Ctrl + X → A to toggle TUI mode.
- ✓ layout split is useful to see both source and assembly.
- ✓ Use display var to track variable changes during execution.
- ✓ info registers helps in exploit development.