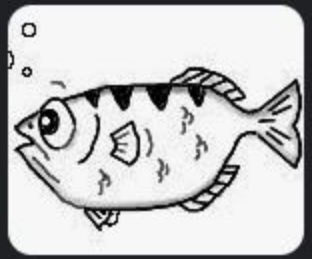
GDB

About 374,000 results (0.38 seconds)

GDB: The GNU Project Debugger

For a fish, the archer fish is known to shoot down bugs from low hanging plants by spitting water at them. Jamie Guinan drew the original Archer Fish Logo. Carlos O'Donell drew the Archer Fish Icon.

Dec 22, 2023



What is GDB?

A debugger for several languages, including C and C++

It allows you to inspect what the program is doing at a certain point during execution

Now why would you want to do that?

This you?

```
printf("===start debug===")
printf("var: %d\n", var)
printf("===end debug===" )
```

(you don’t have to live like this)

Don’t you wish there was a better way?

Compiling your program

Compile with debugging info the -g flag

Gives you debugging info to your program:

```
gcc -g top.c -o top
```

NOTE: this is technically not necessary!

But if you don't do it, I hope you like assembly

How to start it

gdb <program name> --args <param1> <param 2> …

gdb top <- no param

gdb top --args 1 < - 1 param

Set breakpoints

(gdb) break main

Breakpoint 1 at 0x4018bf: file top.c, line 441.

(gdb) break top.c:22

Breakpoint 2 at 0x0019bf: file top.c, line 22.

This makes sure that when you start the program, it stops at the main function and lets you analyze the state and step through your program

Start and Quitting

(gdb) start To start execution

(gdb) run To run the program through

(gdb) quit To exit out of gdb

(gdb) help <command>

To get help on a certain command you are unsure of

Useful GDB Commands summary

step or s:

- executes the next line of code

next or n:

- executes the next line of code (does not enter functions)

print or p:

- prints the value of a variable or register etc.

continue or c:

- runs the program until the next break-point

Useful GDB Commands (Continued)

x:

- examines the data at the given address. (can give it addresses, registers, variables etc.)

info register or i r:

- prints the values of all the registers

Backtrace

backtrace or bt:

- displays a stack trace from your current point of execution

info frame or info f:

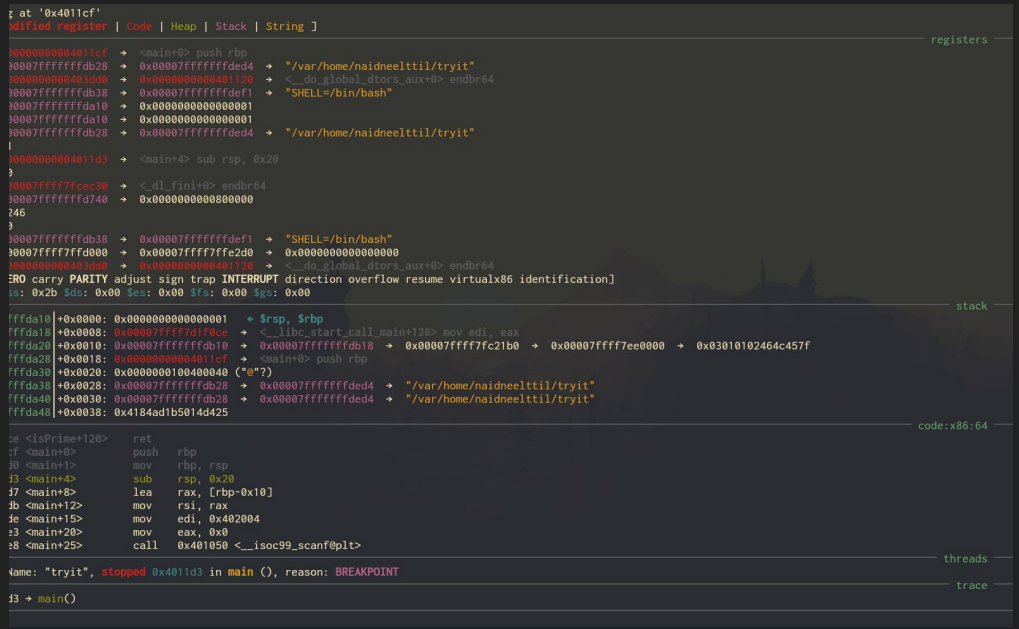
- Prints a verbos description of the selected stackframe. shows
- Address of the frame and surrounding frames.
- Addr of frames' args and local variables, register's saved, etc

What is GEF

a set of commands for x86/64, ARM, MIPS, PowerPC and SPARC to assist exploit developers and reverse-engineers when using old school GDB

-- lifted straight from their github

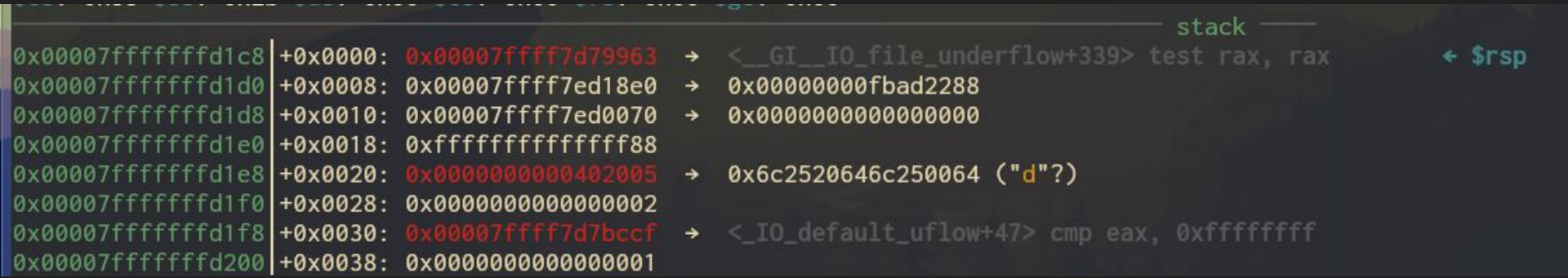
Its prettier and nicer to use



Registers

```
$rax   : 0x00000000004011cf  →  <main+0> push rbp
$rbx   : 0x00007fffffffdb28  →  0x00007fffffffded4  →  "/var/home/naidneelttil/tryit"
$rcx   : 0x0000000000403dd0  →  0x0000000000401120  →  <__do_global_dtors_aux+0> endbr64
$rdx   : 0x00007fffffffdb38  →  0x00007fffffffdef1  →  "SHELL=/bin/bash"
$rsp   : 0x00007fffffffda10  →  0x0000000000000001
$rbp   : 0x00007fffffffda10  →  0x0000000000000001
$rsi   : 0x00007fffffffdb28  →  0x00007fffffffded4  →  "/var/home/naidneelttil/tryit"
$rdi   : 0x1
$rip   : 0x00000000004011d3  →  <main+4> sub rsp, 0x20
$r8    : 0x0
$r9    : 0x00007ffff7fcec30  →  <_dl_fini+0> endbr64
$r10   : 0x00007fffffffd740  →  0x0000000000800000
$r11   : 0x246
$r12   : 0x0
$r13   : 0x00007fffffffdb38  →  0x00007fffffffdef1  →  "SHELL=/bin/bash"
$r14   : 0x00007ffff7ffd000  →  0x00007ffff7ffe2d0  →  0x0000000000000000
$r15   : 0x0000000000403dd0  →  0x0000000000401120  →  <__do_global_dtors_aux+0> endbr64
$eflags: [ZERO carry PARITY adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0x33 $ss: 0x2b $ds: 0x00 $es: 0x00 $fs: 0x00 $gs: 0x00
```

Stack


stack
0x00007fffffffd1c8 +0x0000: 0x00007ffff7d79963 → <__GI__IO_file_underflow+339> test rax, rax ← $rsp
0x00007fffffffd1d0 +0x0008: 0x00007ffff7ed18e0 → 0x00000000fbad2288
0x00007fffffffd1d8 +0x0010: 0x00007ffff7ed0070 → 0x0000000000000000
0x00007fffffffd1e0 +0x0018: 0xffffffffffffff88
0x00007fffffffd1e8 +0x0020: 0x0000000000402005 → 0x6c2520646c250064 ("d"?)
0x00007fffffffd1f0 +0x0028: 0x0000000000000002
0x00007fffffffd1f8 +0x0030: 0x00007ffff7d7bccf → <_IO_default_uflow+47> cmp eax, 0xffffffff
0x00007fffffffd200 +0x0038: 0x0000000000000001

Code (without debugging info)

```
code:x86:64
   0x7ffff7df7e0b <read+11>       je     0x7ffff7df7e20 <read+32>
   0x7ffff7df7e0d <read+13>       xor    eax, eax
   0x7ffff7df7e0f <read+15>       syscall
 → 0x7ffff7df7e11 <read+17>       cmp    rax, 0xfffffffffffff000
   0x7ffff7df7e17 <read+23>       ja     0x7ffff7df7e70 <read+112>
   0x7ffff7df7e19 <read+25>       ret
   0x7ffff7df7e1a <read+26>       nop    WORD PTR [rax+rax*1+0x0]
   0x7ffff7df7e20 <read+32>       sub    rsp, 0x28
   0x7ffff7df7e24 <read+36>       mov    QWORD PTR [rsp+0x18], rdx
```

Code + Source with debugging info

code:x86:64

```
   0x401126 <main+0>        push   rbp
   0x401127 <main+1>        mov    rbp, rsp
   0x40112a <main+4>        sub    rsp, 0x30
→  0x40112e <main+8>        mov    DWORD PTR [rbp-0x4], 0x0
   0x401135 <main+15>       jmp    0x40114f <main+41>
   0x401137 <main+17>       mov    eax, DWORD PTR [rbp-0x4]
   0x40113a <main+20>       cdqe
   0x40113c <main+22>       movzx  eax, BYTE PTR [rbp+rax*1-0x30]
   0x401141 <main+27>       movsx  eax, al
```

source:tryit.c+41

```
36
37      // Null-terminate the string at index 0x24 hex (36 decimal)
38  //    param_1[0x24] = '\0';
39
40      // Print the string
            // i=0x7fff
→ 41        for (int i = 0; i < 0x24; i++) {
42          printf("%c", param_1[i]);
43      }
44      printf("\n");
45      // Free the allocated memory
46  //    free(param_1);
```

threads

Threads and trace info

```
threads
[#0] Id 1, Name: "tryit", stopped 0x4011d3 in main (), reason: BREAKPOINT
trace
[#0] 0x4011d3 → main()
```

Demo

Examples: does this Print 4?

```
┌[naidneelttil@Athena]-(~)
└[15:40]-(^_^)-(78%)-[$] gcc -g tryit.c -o four

┌[naidneelttil@Athena]-(~)
└[15:40]-(^_^)-(78%)-[$] ./four
32719
```

No, let's explore why

```c
#include <stdio.h>
#include <stdbool.h>
#include <math.h>
#include <stdlib.h>

int main() {

    int x;

    for (int i = 0; i < 4; i++) {
        x++;
    }

    // print 4?
    printf("%d\n", x);
    return 0;
}
```

Ex:2 crash.c

```
┌[naidneelttil@Athena]-(~)
└[16:01]-(^_^)-(73%)-[$] gcc -g crash.c -o crash

┌[naidneelttil@Athena]-(~)
└[16:01]-(^_^)-(73%)-[$] ./crash
enter a number:
4
Segmentation fault (core dumped)
```

Let's explore why it crashes

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

char * buf;

int sum_to_n(int num)
{
    int i,sum=0;
    for(i=1;i<=num;i++)
        sum+=i;
    return sum;
}

void printSum()
{
    char line[10];
    printf("enter a number:\n");
    fgets(line, 10, stdin);
    if(line != NULL)
        strtok(line, "\n");
    sprintf(buf,"sum=%d",sum_to_n(atoi(line)));
    printf("%s\n",buf);
}

int main(void)
{
    printSum();
    return 0;
}
```

Why would I need GDB?
Im not a C newbie, I passed
230 I know exactly what these
Programs are doing. You know K & R?
Thats my first and middle name. Das me.

I could never make these
errors in the first place.

Except what if this is your code : UnNetHack

```
┌[naidneelttil@Hestia]-(~/UnNetHack/src)(master U:1 ?:2 ✗)
└[16:44]-(^_^)-(61%)-[$] ls
allmain.c   dlb.c      explode.c   mhitm.c    muse.c      read.c       sp_lev.c    vision.c
alloc.c     do.c       extralev.c  mhitu.c    music.c     rect.c       steal.c     weapon.c
apply.c     dog.c      files.c     minion.c   objects.c   region.c     steed.c     were.c
artifact.c  dogmove.c  fountain.c  mklev.c    objnam.c    restore.c    sys.c       wield.c
attrib.c    dokick.c   hack.c      mkmap.c    o_init.c    rip.c        teleport.c  windows.c
ball.c      do_name.c  hacklib.c   mkmaze.c   options.c   rnd.c        timeout.c   wizard.c
base32.c    dothrow.c  insight.c   mkobj.c    pager.c     rnd_isaac.c  topten.c    worm.c
bones.c     do_wear.c  invent.c    mkroom.c   pickup.c    role.c       track.c     worn.c
botl.c      drawing.c  light.c     mksheol.c  pline.c     rumors.c     trap.c      write.c
cmd.c       dump.c     livelog.c   mon.c      polyself.c  save.c       tutorial.c  zap.c
dbridge.c   dungeon.c  lock.c      mondata.c  potion.c    shk.c        uhitm.c
decl.c      eat.c      mail.c      monmove.c  pray.c      shknam.c     u_init.c
detect.c    end.c      makemon.c   monst.c    priest.c    sit.c        unicode.c
dig.c       engrave.c  mapglyph.c  mplayer.c  quest.c     sounds.c     vault.c
display.c   exper.c    mcastu.c    mthrowu.c  questpgr.c  spell.c      version.c

┌[naidneelttil@Hestia]-(~/UnNetHack/src)(master U:1 ?:2 ✗)
└[16:44]-(^_^)-(61%)-[$]
```

And this is your error:

```
(gdb) bt
#0  __pthread_kill_implementation (threadid=<optimized out>, signo=signo@entry=6, no_tid=no_tid@entry=0) at pthr
#1  0x00007f734250c0e3 in __pthread_kill_internal (signo=6, threadid=<optimized out>) at pthread_kill.c:78
#2  0x00007f73424bce06 in __GI_raise (sig=sig@entry=6) at ../sysdeps/posix/raise.c:26
#3  0x00007f73424a58f5 in __GI_abort () at abort.c:79
#4  0x00007f73424a67a1 in __libc_message (fmt=fmt@entry=0x7f734261e2f8 "*** %s ***: terminated\n") at ../sysdeps
#5  0x00007f734259b1d9 in __GI___fortify_fail (msg=msg@entry=0x7f734261e2df "buffer overflow detected") at fort
#6  0x00007f734259ab94 in __GI___chk_fail () at chk_fail.c:28
#7  0x00000000005b2ac5 in strcpy (__src=0x7ffe68838b00 "Shall I pick a character's race, role, gender and alignm
    __dest=0x7ffe68838990 "\001") at /nix/store/B0S2LKF593R3585038WS4JD3LYLF2WDX-glibc-2.38-44-dev/include/bits/
#8  curses_break_str (str=str@entry=0x7ffe68838b00 "Shall I pick a character's race, role, gender and alignment
    line_num=line_num@entry=1) at ../win/curses/cursmisc.c:275
#9  0x00000000005b3f51 in curses_character_input_dialog (prompt=prompt@entry=0x7ffe68838cf0 "Shall I pick a cha
    choices=choices@entry=0x7ffe68838d70 "YNTQ", def=def@entry=121) at ../win/curses/cursdial.c:211
#10 0x00000000005b9ca0 in curses_choose_character () at ../win/curses/cursinit.c:556
#11 0x0000000000404eb1 in main (argc=<optimized out>, argv=<optimized out>) at ./../sys/unix/unixmain.c:309
```

Frame 8 reveals the issue

This corresponds to this `gcc` warning:

```
../win/curses/cursmisc.c: In function 'curses_break_str':
../win/curses/cursmisc.c:275:5: warning: '__builtin___strcpy_chk' writing one too many bytes into a regio
  275 |     strcpy(substr, str);
      |     ^
```

End