CS29206 Systems Programming Laboratory
Spring 2024

Introduction to gdb

Abhijit Das
Pralay Mitra

Why debuggers?

- Writing error-free codes (particularly large ones) is every programmer's dream.
  - Compilation errors
  - Logical errors
  - Runtime errors
- The compiler is "*never*" faulty.
- If your program does not run, question your understanding and your code first.
- Debuggers help you out there.
  - You can step through your code line by line.
  - You can keep on watching variables.
  - No need to write diagnostic printf's.
  - . . .
- The GNU debugger (gdb) is a popular choice of this day.
- Several graphic debuggers (like xxgdb) are built on top of gdb.

First example: Area of a triangle

tarea.c

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

int main ()
{
   int x1, y1, x2, y2, x3, y3;
   double area;

   printf("Program to calculate the area of a triangle with integer coordinates\n");

   printf("Enter the coordinates of the first corner: ");
   scanf("%d%d", &x1, &y1);
   printf("Enter the coordinates of the second corner: ");
   scanf("%d%d", &x2, &y2);
   printf("Enter the coordinates of the third corner: ");
   scanf("%d%d", &x3, &y3);

   area = x1 * (y2 - y3);
   area += x2 * (y1 - y3);
   area += x3 * (y1 - y2);
   if (area < 0) area = -area;
   area /= 2;

   printf("Area of the triangle is %lf\n", area);

   exit(0);
}
```

Running codes with gdb

```
$ gcc -Wall -g tarea.c
$ gdb ./a.out
GNU gdb (Ubuntu 9.2-0ubuntu1 20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./a.out...
(gdb) run
Starting program: /home/abhij/IITKGP/course/lab/SPL/Spring22/prog/gdb/a.out
Program to calculate the area of a triangle with integer coordinates
Enter the coordinates of the first corner: 3 8
Enter the coordinates of the second corner: 6 -7
Enter the coordinates of the third corner: -1 5
Area of the triangle is 16.500000
[Inferior 1 (process 6730) exited normally]
(gdb) quit
$
```

Listing your code

- list: Keep on listing your code, ten lines at a time.
- list m,n: List from Line m to Line n of your code.
- list k: List ten lines with Line k at the center.

line k has 6th position

```
(gdb) list
1       #include <stdio.h>
2       #include <stdlib.h>
3
4       int main ()
5       {
6          int x1, y1, x2, y2, x3, y3;
7          double area;
8
9          printf("Program to calculate the area of a triangle with integer coordinates\n");
10
(gdb) list
11         printf("Enter the coordinates of the first corner: ");
12         scanf("%d%d", &x1, &y1);
13         printf("Enter the coordinates of the second corner: ");
14         scanf("%d%d", &x2, &y2);
15         printf("Enter the coordinates of the third corner: ");
16         scanf("%d%d", &x3, &y3);
17
18         area = x1 * (y2 - y3);
19         area += x2 * (y1 - y3);
20         area += x3 * (y1 - y2);
```

Listing your code: Continued

```
(gdb) list
21          if (area < 0) area = -area;
22          area /= 2;
23
24          printf("Area of the triangle is %lf\n", area);
25
26          exit(0);
27      }
(gdb) list
Line number 28 out of range; tarea.c has 27 lines.
(gdb) list 15
10
11          printf("Enter the coordinates of the first corner: ");
12          scanf("%d%d", &x1, &y1);
13          printf("Enter the coordinates of the second corner: ");
14          scanf("%d%d", &x2, &y2);
15          printf("Enter the coordinates of the third corner: ");
16          scanf("%d%d", &x3, &y3);
17
18          area = x1 * (y2 - y3);
19          area += x2 * (y1 - y3);
(gdb) list 18,22
18          area = x1 * (y2 - y3);
19          area += x2 * (y1 - y3);
20          area += x3 * (y1 - y2);
21          if (area < 0) area = -area;
22          area /= 2;
(gdb)
```

Set a breakpoint and step through your code

```
(gdb) break 11
Breakpoint 1 at 0x11d0: file tarea.c, line 11.
(gdb) run
Starting program: /home/abhij/IITKGP/course/lab/SPL/Spring22/prog/gdb/a.out
Program to calculate the area of a triangle with integer coordinates

Breakpoint 1, main () at tarea.c:11
11          printf("Enter the coordinates of the first corner: ");
(gdb) next
12          scanf("%d%d", &x1, &y1);
(gdb) next
Enter the coordinates of the first corner: 3 8
13          printf("Enter the coordinates of the second corner: ");
(gdb) next
14          scanf("%d%d", &x2, &y2);
(gdb) next
Enter the coordinates of the second corner: 6 -7
15          printf("Enter the coordinates of the third corner: ");
(gdb) next
16          scanf("%d%d", &x3, &y3);
(gdb) next
Enter the coordinates of the third corner: -1 5
18          area = x1 * (y2 - y3);
(gdb) next
19          area += x2 * (y1 - y3);
(gdb) continue
Continuing.
Area of the triangle is 16.500000
[Inferior 1 (process 7195) exited normally]
(gdb) q
```

Some notes about gdb

- **Command shortcuts:** You do not have to type the entire command (like list or next). Certain abbreviations are allowed. Here is a short list.

b	break
bt	backtrace
c	continue
d	delete
dis	disable
disp	display
do	down
en	enable
f	frame
fin	finish

h	help
i	info
l	list
n	next
p	print
q	quit
ret	return
r	run
s	step

- Command name abbreviations are allowed if unambiguous.
- Hit return to repeat the last command.
- **The *unaltered* source file must be present** for gdb to run correctly.

More about breakpoints

- You can set multiple breakpoints.
- Type

  `info break`

  to list all the breakpoints.
- continue stops until the next breakpoint or the end of the program
- Other things that you can do with breakpoints:
  - Disable a breakpoint
  - Enable a disabled breakpoint
  - Delete a breakpoint

```
(gdb) break main
Breakpoint 1 at 0x11a9:  file tarea.c, line 5.
(gdb) break 11
Breakpoint 2 at 0x11d0:  file tarea.c, line 11.
(gdb) break 18
Breakpoint 3 at 0x1257:  file tarea.c, line 18.
(gdb) break 24
Breakpoint 4 at 0x12e5:  file tarea.c, line 24.
```

Breakpoint examples

```
(gdb) info break
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x00000000000011a9 in main at tarea.c:5
2       breakpoint     keep y   0x00000000000011d0 in main at tarea.c:11
3       breakpoint     keep y   0x0000000000001257 in main at tarea.c:18
4       breakpoint     keep y   0x00000000000012e5 in main at tarea.c:24
(gdb) disable 3
(gdb) info break
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x00000000000011a9 in main at tarea.c:5
2       breakpoint     keep y   0x00000000000011d0 in main at tarea.c:11
3       breakpoint     keep n   0x0000000000001257 in main at tarea.c:18
4       breakpoint     keep y   0x00000000000012e5 in main at tarea.c:24
(gdb) enable 3
(gdb) info break
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x00000000000011a9 in main at tarea.c:5
2       breakpoint     keep y   0x00000000000011d0 in main at tarea.c:11
3       breakpoint     keep y   0x0000000000001257 in main at tarea.c:18
4       breakpoint     keep y   0x00000000000012e5 in main at tarea.c:24
(gdb) delete 3
(gdb) info break
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x00000000000011a9 in main at tarea.c:5
2       breakpoint     keep y   0x00000000000011d0 in main at tarea.c:11
4       breakpoint     keep y   0x00000000000012e5 in main at tarea.c:24
(gdb)
```

Printing variables and expressions

```
(gdb) break 18
Breakpoint 1 at 0x1257: file tarea.c, line 18.
(gdb) run
Starting program: /home/abhij/IITKGP/course/lab/SPL/Spring22/prog/gdb/a.out
Program to calculate the area of a triangle with integer coordinates
Enter the coordinates of the first corner: 3 8
Enter the coordinates of the second corner: 6 -7
Enter the coordinates of the third corner: -1 5

Breakpoint 1, main () at tarea.c:18
18          area = x1 * (y2 - y3);
(gdb) print x1
$1 = 3
(gdb) print x2
$2 = 6
(gdb) print x3
$3 = -1
(gdb) print x1 + x2 + x3
$4 = 8
(gdb) print x1 + x2 * x3
$5 = -3
(gdb) print x1 * (y2 - y3) + x2 * (y1 - y3) + x3 * (y1 - y2)
$6 = -33
(gdb) print $6 / 2
$7 = -16
(gdb)
```

Value history: The printed values are stored as \$1, \$2, . . . , and can be accessed later.

Watching variables after every step

```
(gdb) break 16
Breakpoint 1 at 0x123b: file tarea.c, line 16.
(gdb) run
...
Breakpoint 1, main () at tarea.c:16
16          scanf("%d%d", &x3, &y3);
(gdb) n
Enter the coordinates of the third corner: -1 5
18          area = x1 * (y2 - y3);
(gdb) display area
1: area = 6.953355807476115e-310
(gdb) n
19          area += x2 * (y1 - y3);
1: area = -36
(gdb) n
20          area += x3 * (y1 - y2);
1: area = -18
(gdb) n
21          if (area < 0) area = -area;
1: area = -33
(gdb) n
22          area /= 2;
1: area = 33
(gdb) n
24          printf("Area of the triangle is %lf\n", area);
1: area = 16.5
(gdb) continue
Continuing.
Area of the triangle is 16.500000
[Inferior 1 (process 8608) exited normally]
(gdb)
```

Setting variables: The program

setvar.c

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

int main ()
{
   int a, b;

   a = 5;
   b = 8;
   printf("max(%d,%d) = %d\n", a, b, (a >= b) ? a : b);
   printf("max(%d,%d) = %d\n", a, b, (a >= b) ? a : b);
   exit(0);
}
```

Setting variables: Run gdb

```
(gdb) break main
Breakpoint 1 at 0x1169: file setvar.c, line 5.
(gdb) n
The program is not being run.
(gdb) run
Starting program: /home/abhij/IITKGP/course/lab/SPL/Spring22/prog/gdb/a.out

Breakpoint 1, main () at setvar.c:5
5        {
(gdb) n
8            a = 5;
(gdb) n
9            b = 8;
(gdb) n
10           printf("max(%d,%d) = %d\n", a, b, (a >= b) ? a : b);
(gdb) n
max(5,8) = 8
11           printf("max(%d,%d) = %d\n", a, b, (a >= b) ? a : b);
(gdb) set var b = 2
(gdb) print b
$1 = 2
(gdb) n
max(5,2) = 5
12           exit(0);
(gdb) n
[Inferior 1 (process 12878) exited normally]
(gdb)
```

Conditions and loops

- Nothing special needs to be done.
- Stepping at the end of the loop goes back to the start of the loop.
- When the loop breaks, the line following the loop is executed.

average.c

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

int main ()
{
   int n, x, sum;

   n = sum = 0;
   printf("Keep on entering non-negative integers. Enter a negative integer to end.\n");
   while (1) {
      printf("Next number: "); scanf("%d", &x);
      if (x < 0) break;
      ++n; sum += x;
   }
   printf("Average of the numbers entered is %lf\n", (double)sum / (double)n);
   exit(0);
}
```

Conditions and loops: Example run

```
(gdb) break main
Breakpoint 1 at 0x11a9: file average.c, line 5.
(gdb) run
Starting program: /home/abhij/IITKGP/course/lab/SPL/Spring22/prog/gdb/a.out

Breakpoint 1, main () at average.c:5
5       {
(gdb) n
8           n = sum = 0;
(gdb) n
9           printf("Keep on entering non-negative integers. Enter a negative integer to end.\n");
(gdb) n
Keep on entering non-negative integers. Enter a negative integer to end.
11              printf("Next number: "); scanf("%d", &x);
(gdb) n
Next number: 5
12              if (x < 0) break;
(gdb) n
13              ++n; sum += x;
(gdb) n
11              printf("Next number: "); scanf("%d", &x);
(gdb) n
Next number: 8
12              if (x < 0) break;
(gdb) n
13              ++n; sum += x;
(gdb) n
```

Conditions and loops: Example run (Continued)

```
11              printf("Next number: "); scanf("%d", &x);
(gdb) n
Next number: 4
12              if (x < 0) break;
(gdb) n
13              ++n; sum += x;
(gdb) n
11              printf("Next number: "); scanf("%d", &x);
(gdb) n
Next number: -1
12              if (x < 0) break;
(gdb) n
15          printf("Average of the numbers entered is %lf\n", (double)sum / (double)n);
(gdb) n
Average of the numbers entered is 5.666667
16          exit(0);
(gdb) n
[Inferior 1 (process 10883) exited normally]
(gdb)
```