Lab 8 – InLab

CSC 3320 SYSTEM LEVEL PROGRAMMING
FALL 2019

Objective

- To learn how to debug a C program using **gdb**

What is gdb?

- GNU Debugger
- The most popular debugger for UNIX systems to debug C and C++ programs.

Features provided by gdb

- Set up **breakpoints**
- Control the execution : Continue, Stepping over and in
- Printing the variable values
- Miscellaneous

Set up breakpoints

- **break** line_number
  - E.g. Set up a breakpoint at line 10
  - Command: **break 10**

Demo

```
(gdb) break 10
Breakpoint 1 at 0x400605: file debug.c, line 10.
```

Control the execution

- **run** (or **r**) : start a new execution until a breakpoint or to the end if no breakpoints.
- **continue**(or **c**) : continue executing until the next breakpoint
- **next** (or **n**) : execute next line as a single instruction (step over)
- **step** (or **s**) : same as next, but does not treat the function as a single instruction, instead it goes into function and executes it line by line (step in)

Printing the variable values

- **print** variable_name
  - E.g. print i

```
print  j
```

  - Or simply use **p** instead of **print**

```
p    i
p    j
```

Print backtraces of all stack frame

- **where**
  - Shows stack: sequence of function calls executed so far
  - Good for pinpointing location of a program crash
- **up**
  - Goes up one level in the stack

Display the values of expressions at each step

- **display** expression
  - E.g. Show the value of expression of b + 1 at each time your program stops

```
display  b+1
```

Demo

```
(gdb) display  b+1
1: b+1 = 2
(gdb) n
7          scanf("%d",b);
1: b+1 = 2: file debug.c, line 10.
```

Demo

Step 1 : Compile the debug.c with -g debugging option

```
$gcc   -o debug -g  debug.c
```

Step 2: Lauch gdb

```
$gdb  debug
```

These two steps are very important!!!

Demo

Step 3: Set up breakpoint at line 6

```
(gdb) break 6
```

Demo

Step 3: List the source code

```
(gdb) list
1     #include<stdio.h>
2
3     int main(void){
4
5         int b=1,c=2;
6         printf("Please enter the value of
b:");
7         scanf("%d",b);
8         c=b/3;
9         printf("%d/3=%d \n",b,c );
10    }
```

Demo

Step 4 :Run from start until the first breakpoint

```
(gdb) r
Starting program: /home/ylong4/public/debug

Breakpoint 1, main () at debug.c:6
6         printf("Please enter the value of b:");
```

Step 5: Print out value stored in b

```
(gdb) p b
$1 = 1
```

Control the execution

Step 6 : Run the next two instructions

```
(gdb) n
7   scanf("%d",b);
(gdb) n
Please enter the value of b:5

Program received signal SIGSEGV, Segmentation
fault.
0x00007ffff7a72122 in __GI__IO_vfscanf  from
/lib64/libc.so.6
```

Now we find errors.
But how to check
which line generates
this errors ?

Print backtraces of all stack frame

Step 7 : Show stack and go up one level

```
(gdb) where
#0  0x00007ffff7a72122 in __GI__IO_vfscanf () from
/lib64/libc.so.6
#1  0x00007ffff7a80b59 in __isoc99_scanf () from
/lib64/libc.so.6
#2  0x00000000004005d6 in main () at debug.c:7
(gdb) up
#1  0x00007ffff7a80b59 in __isoc99_scanf () from
/lib64/libc.so.6
(gdb) up
#2  0x00000000004005d6 in main
7         scanf("%d",b);
```

This line generated error.
What is the error?

Other Resources

- Online GDB

  https://www.onlinegdb.com
- RMS's gdb debugging tutorial

  http://www.unknownroad.com/rtfm/gdbtut/gdbtoc.html
- GDB Quickstart

  https://www.youtube.com/watch?v=5yZIFmplXsw