Using GDB

Prof. Jinkyu Jeong (jinkyu@skku.edu)
TA – Donghyun Kim (wadong100@csi.skku.edu)
TA – Jiwon Woo (jiwon.woo@csi.skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu
GCC

- GNU C Compiler
- `sudo apt-get update`
- `sudo apt-get install build-essential`
- `gcc -o test test.c`

- [https://modoocode.com/14](https://modoocode.com/14)
GNU Debugger (GDB)

• What is GDB?
  – GDB, the GNU Project debugger, allows you to see
    • what is going on `inside' another program while it executes.
    • what another program was doing at the moment it crashed.

  – https://www.gnu.org/software/gdb/
GNU Debugger (GDB)

• GDB can do four main kinds of things
  – Start your program, specifying anything that might affect its behavior.
    • run, continue, finish, kill
  – Make your program stop on specified conditions.
    • breakpoint
  – Examine what has happened, when your program has stopped.
    • examine(x), print(p), disp, info, disas, list
  – Change things in your program, so you can experiment with correcting the effects of one bug and go on to learn about another.
    • set reg
GDB Debugging Process

1. Compile program with –g option
2. Launch gdb
3. Set breakpoints
4. Execute the program in gdb
5. Examine & Set things
6. Execute code line by line
7. Find bugs
Start & Quit GDB

• Compile a program for gdb

```
ziwoo@cs1:~/ta/lab2$ gcc -O1 -g ex.c exercises.c -o ex1
```

• Start gdb

```
ziwoo@cs1:~/ta/lab2$ gdb ./ex1
GNU gdb (Ubuntu 8.1-0ubuntu3) 8.1.0.20180409-git
Copyright (C) 2018 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:
Find the GDB manual and other documentation resources online at:
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./ex1...done.
(gdb) quit
ziwoo@cs1:~/ta/lab2$
```
GDB commands

• (gdb) help
  – aliases
  – breakpoints
  – data
  – files
  – internals
  – obscure
  – running
  – stack
  – status
  – support
  – tracepoints
  – user-defined

(gdb) help
List of classes of commands:

aliases -- Aliases of other commands
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
tracepoints -- Tracing of program execution without stopping the program
user-defined -- User-defined commands

Type "help" followed by a class name for a list of commands in that class.
Type "help all" for the list of all commands.
Type "help" followed by command name for full documentation.
Type "apropos word" to search for commands related to "word".
Command name abbreviations are allowed if unambiguous.
Disassemble

- `disas <function_name>`
Executing

• run
  – execute code from beginning

• continue
  – execute code from current point

• si / ni
  – execute current instruction

• kill
  – kill current program

• return <value>
  – without executing current function
Breakpoint

- `break <point>`
- `info breakpoint (i b)`
- `del <number>`
- `watch / rwatch / awatch $eax`

```
(gdb) break exercise_1
Breakpoint 1 at 0x85d: file exercises.c, line 12.
(gdb) break exercise_2
Breakpoint 2 at 0x87e: file exercises.c, line 22.
(gdb) run
Starting program: /home/z1woo/ta/lab2/ex1
Exercise #1 | Enter a number
1212

Breakpoint 1, exercise_1 (a=1212) at exercises.c:12
12     if ( a != 0x2030 ) {
(gdb) watch $rax
Watchpoint 3: $rax
(gdb) i b
Num Type Disp Enb Address What
1  breakpoint keep y 0x0000555555555485d in exercise_1 at exercises.c:12
2  breakpoint keep y 0x0000555555555487e in exercise_2 at exercises.c:22
3  watchpoint keep y $rax
```
Registers

- info registers (i r)
- set $rax=0x10
- print $rax
objdump

• `objdump -t <file>`
  – print out symbol table of the file.

• `objdump -d <file>`
  – disassemble target file.
Exercise

• Find out what those functions do and find Answers for each.

```c
#include <stdio.h>
#include "exercises.h"

int main (void)
{
    int ret=0;
    int input1, input2;

    printf("Exercise #1 | Enter a number\n");
    ret = scanf("%d", &input1);
    ret = exercise_1( input1 );

    printf("Exercise #2 | Enter two numbers\n");
    ret = scanf("%d %d", &input1, &input2);
    ret = exercise_2( input1, input2 );

    printf("Exercise #3 | Enter two numbers\n");
    ret = scanf("%d %d", &input1, &input2 );
    ret = exercise_3(input1, input2);

    printf("Exercise Finished!!\n");
    return 0;
}
```
Project #2

• Announcement