Intruduction to GDB

Prof. Jin-Soo Kim(jinsoookim@skku.edu)
TA - Kisik Jeong (kisik@csl.skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu
What is “Bug”?

- error, flaw, failure or fault in a computer program or system that causes it to produce an incorrect or unexpected result, or to behave in unintended ways.

(from wikipedia)
What is “Debug”?

- To (1) find and (2) fix bug(s)
How to Debug?

1. How to find bug(s)?
   - Review source codes
   - Use printf
   - Use debugger

2. How to fix bug(s)?
   - Once you find bug(s), fixing them is easy
What is GDB?

- The GNU Project debugger

- https://www.gnu.org/software/gdb/
GDB Debugging Process

0. Compile a program for gdb
1. Start gdb
2. Set breakpoint(s)
3. Run program
4. Stop at the breakpoint
5. Display/watch expression(s)
6. Execute every single line of source code
7. Find bug!
8. Quit gdb
9. Fix it!
GDB Manual

- Compile a program for gdb
  - $ gcc -g SOURCE.c -o OBJECT

- Start gdb
  - $ gdb ./OBJECT
    - OBJECT is the executable file

- Quit gdb
  - (gdb) quit
GDB Manual (Cont’d)

- List parts of the source code
  - (gdb) list
  - (gdb) list function_name
  - (gdb) list line_number

- Examine the assembly code
  - (gdb) disas
  - (gdb) disas function_name
  - (gdb) disas address
Set breakpoint
- (gdb) break function_name
- (gdb) break line_number

Show breakpoint
- (gdb) info break

Delete breakpoint
- (gdb) delete breakpoint_number
GDB Manual (Cont’d)

- **Set watchpoint for an expression**
  - (gdb) `watch` expression
  - (gdb) `watch $rax`

- **Show watchpoint**
  - (gdb) `info watch`
  - (gdb) `info break`

- **Delete watchpoint**
  - (gdb) `delete breakpoint_number`
GDB Manual (Cont’d)

- Run the program
  - (gdb) run
  - (gdb) run arglist < input > output

- Continue the program
  - (gdb) continue

- Run until current function returns
  - (gdb) finish

- Stop the program
  - (gdb) kill
GDB Manual (Cont’d)

- Step into current source line
  - (gdb) step
  - (gdb) step number_of_line

- Step over current source line
  - (gdb) next
  - (gdb) next number_of_line
GDB Manual (Cont’d)

- **Step into current instruction**
  - (gdb) `steipi`
  - (gdb) `steipi number_of_line`

- **Step over current instruction**
  - (gdb) `nexti`
  - (gdb) `nexti number_of_line`
GDB Manual (Cont’d)

- **Display value of expression**
  - Display expression every time the program pauses
  - `(gdb) disp expression`

- **Show display list**
  - `(gdb) info disp`

- **Undisplay expression**
  - `(gdb) undisp display_number`
GDB Manual (Cont’d)

- **Print value of expression**
  - (gdb) `p expression`
  - (gdb) `p $rax`
  - (gdb) `p constant`
  - (gdb) `p /x expression`
    - Print variable in hexadecimal format
  - (gdb) `p /t expression`
    - Print variable in binary format
  - (gdb) `p *(long *) address`
    - Print long integer of long * type pointer
GDB Manual (Cont’d)

- **Examine memory**
  - `(gdb) x address`
  - `(gdb) x $rax`
  - `(gdb) x function_name`
  - `(gdb) x/2g address`
    - Examine two (8-bytes) words starting at the address
  - `(gdb) x/20b address`
    - Examine first 20 bytes at the address
GDB Manual (Cont’d)

- **Useful information**
  - `(gdb) info frame`
    - Information about current stack frame
  - `(gdb) info registers`
    - Values of all the registers
  - `(gdb) info locals`
    - Print all local variables
  - `(gdb) help`
    - Get information about gdb
Debugging Practice

- Let’s try!