A Tutorial on GDB

Avinash Gautam, CSIS, BITS Pilani

Getting Started

1. Boot to your machine to Linux. Open two terminals, one to work in gdb prompt and the other to work on the shell.

2. Although all commands in gdb are full words, you can type any unambiguous prefix. For instance, p instead of print.

3. Use the `help` command in gdb for a list of topics. Then type `help` *topic* for information on a specific topic. `help` *cmd* gives help on the command cmd e.g. help print will give information about the print command. For comprehensive coverage of gdb, read the Info pages. At the shell prompt, type `info.` When info starts, type mgdb<ENTER>.

Compilation

4. Compile the file *filename.c* and put the output in the executable file called sample

   ```
   $gcc –o sample –g filename.c
   ```

Execution with gdb

5. Before executing the program, put a breakpoint

   ```
   (gdb) b main
   ```

   Run gdb
   ```
   gdb sample
   (gdb) r
   ```

   Breakpoints can be put using either:
   - the function name or
   - the line number or
   - source file name: line number

   Try putting the breakpoint using all the above ways

   Quit gdb
   ```
   (gdb)  q
   ```

6. To have a look at the source code at any point in time

   ```
   (gdb) list
   ```

   By default the number of lines displayed is 10, to change the number of lines to be displayed

   ```
   (gdb) set listsize 20
   (gdb) list
   ```

7. Information of all the set breakpoints can be seen

   ```
   (gdb) info b
   ```

   The first column shows the id of the breakpoint

8. Delete all the breakpoints except the one on function main

```
(gdb) d id1 id2 …
```

9. Now run the program and observe that the execution stops at the first breakpoint that is encountered. Now the control of the program is with gdb and each statement can be executed one at a time using the command:

```
(gdb) n
```

Run the n command till the statement before the call to a function

10. Step inside the function

```
(gdb) s
```

11. Let the program continue on its own

```
(gdb) c
```

12. `p` command can be used to compute an expression or a function

```
(gdb) p strlen("abc")
```

$4 = 3

p command can be used to initialize a variable during the program execution in (debugger) gdb

```
(gdb) p x = 40
(gdb) p str = "str is a string"
```

13. Breakpoints can be disabled and enabled using the following commands:

```
(gdb) dis <id>
(gdb) en <id>
```

Check the status of the breakpoint after disabling and enabling the breakpoint

14. A watchpoint is a special breakpoint that stops your program when the value of an expression changes. You can put a watchpoint on variable x in a function say main() as follows:

```
(gdb) b main
(gdb) r
(gdb) n
```

keep running the n command till the variable x is defined

```
(gdb) watch x
(gdb) c
```

What do you observe?