GDB Basics

System Programming 2022 Fall

What is GDB?

- GNU Debugger
- **debugger** for C and C++ programs
  - allows users to control programs, and to examine variables
- helps particularly when there’re bugs regarding memory

How does it work?

1. Run the program up to a certain point (breakpoint)
2. Stop and examine the variables at that time
3. Step forward line by line

Let’s get started!

Preparation

1. Install gdb if you haven’t had one

   ```
   sudo apt install gdb
   ```

2. compile source file (.c or .cpp) into executable

   ```
   gcc -g <src file> [-o <obj file>]
   gcc -g example.c
   ```

3. run gdb

   ```
   gdb <obj file>
   ```

4. quit (from gdb)

   ```
   (gdb) quit
   ```

Useful commands

List source code

```
(gdb) l [<line num>]
```

Adding breakpoints

```
(gdb) b <line num>
(gdb) b <src file>:<line num>
# when you have multiple .c/.cpp files to work with
```

Deleting breakpoints

```
(gdb) d <breakpoint num>
```

List all breakpoints

```
(gdb) info breakpoints
```

Useful commands

Run

```
(gdb) r
```

Show variable values

```
(gdb) p <var name>
(gdb) info locals                    # prints all local variables
```

Modify variable values

```
(gdb) p <var name> = <val>
```

Examining frames in the stack

```
(gdb) bt                                        # backtrace
(gdb) frame <frame num>                         # move to a frame
(gdb) up                                        # move upward
(gdb) down                                      # move downward
```

Go through the program

```
(gdb) s              # step forward; run the next instruction
(gdb) n              # run until next line
(gdb) c              # continue until next breakpoint
(gdb) kill           # end the program
```

Recompile source code (only if there's a Makefile)

```
(gdb) make
```

More Reference...

1. info gdb
2. introduction to GDB a tutorial - Harvard CS50
3. Quick Gdb Guide (depaul.edu)
4. GNU Debugger Tutorial (tutorialspoint.com)

Demo :)