UNIVERSITY OF ILLINOIS

AT URBANA-CHAMPAIGN

GDB Tutorial

~CS 225

illinois.edu

What is GDB?

- GNU Debugger

- Your new best friend

Compiling for GDB

- GDB uses special debugging symbols

- Compile with “-g” option in g++
  - Put it in your Makefile compiler options!
  - “-ggdb” option will give more specific information to GDB

- Compile with “-O0” to turn off optimizations
  - These confuse GDB

Running GDB

- Use command “gdb [file]”
  - Ex: “gdb ./foo”
  - Do not put arguments here!
- GDB will start with lots of information about itself (Google how to turn these off)
- You will see a prompt like this:

  (gdb)
- If you didn't specify the file, use “file [filename]”
  - Ex: “(gdb) file ./foo”

HELP

- The first command you need to know:
  - (gdb) help [command]

Running programs

```
(gdb) run [arguments]
```

- This will execute the program, with the arguments specified
- All outputs should be the same as executing from the command line, plus some gdb outputs

- GDB will give lots of information on errors

Quitting GDB

```
(gdb) quit
```

Backtrace / Where

```
(gdb) backtrace
(gdb) where
```

- Outputs stack frames
  - Which functions called which
  - What parameters were passed at each point
- Really helpful with segfaults!
- (Up/down/frame [n] – movement within frames)

List

```
(gdb) list [line/function]
```

- Lists source code!
- 10 lines at a time
  - Centered at [line/function]
  - Next 10 lines if no argument

Breakpoints!

```
(gdb) break [line/function]
```

- Stops execution before a specific line
- Returns control to GDB user
- Each breakpoint has a number
- Other commands:
  - List of breakpoints:
    - *Info breakpoints*

Breakpoints!

(gdb) break [line/function]

- Other commands:
  - Removal:
    - *Delete [number]*
    - *Clear [line]*

Print/Display

(gdb) print <expr>   (gdb) display <expr>

- Used to see values of variables!
- Print:
  - Shows the value of the <expr>
    - *Useful to check variable values*
- Display:
  - Shows the value of the <expr> at every next/step call
    - *Useful to track variable values through loops*

Continue/Next/Step

```
(gdb) continue      (gdb) next      (gdb) step
```

- Instruction movement commands in GDB
- Continue
  - Resume execution until next break/fault/end
- Next
  - Execute the current instruction
    - *Don't enter function calls*
- Step
  - Execute the current instruction
    - *Go into all function calls*

Conditional Breakpoints

```
(gdb) condition <id> <expression>
(gdb) break [line/func] if <expression>
```

- Will only break on breakpoint <id> if the <expression> evaluates true
- Useful in loops!
- Ex: “condition 2 i==35”
  - Stops at breakpoint 2 if variable 'i' has value 35

Watchpoints

```
(gdb) watch <var>        (gdb) rwatch <var>
          (gdb) awatch <var>
```

- Used to monitor variables
- watch:
  - Breaks when variable is written
- rwatch:
  - Breaks when variable is read
- awatch:
  - Breaks on both!

References

http://www.yolinux.com/TUTORIALS/GDB-Commands.html

http://www.unknownroad.com/rtfm/gdbtut/gdbtoc.html