Intelligence Of Things®

GLOBAL EDGE

•ALTRAN GROUP

GDB – (The GNU Debugger)

D Karthik

Contents

Contents contd...

Introduction

- What is GDB?

  GNU Debugger(GDB) is a tool used for debugging the program.

- Why GDB?

  The purpose of GDB is to allow us to see what is going on at the backend while the program executes.

- GDB can do four main functions:
  - Start the program
  - Make the program to stop on specified conditions
  - Examine what has happened
  - Change things in the program

Introduction contd...

- Command:

```
gcc -g -o (executable filename) (filename.c)
```

g -->This flag is used at the time of compilation to add debugging symbols into the executable file.

Basic commands

- Breakpoint(b) - set a place where GDB stops executing your program.
- Continue(c) – continue program execution.
- List(l) - list the source code.
- Next(n) - execute the next line of code, then stop.
- Print(P) - print the value of a variable or expression.
- Quit(q) - quit GDB.
- Run(r) - run the program from the beginning.
- Step(s) - step to the next line of code.
- Examine memory(x) - examine memory.

Breakpoint and watchpoint

- break arg (or) b arg - The break command's arguments may be a line number, a function name, or the address of an instruction. With no arguments, the break command sets a breakpoint at the next instruction to be executed.

- We can get information about all the breakpoints we have set using the info command: info breakpoints.

- Watchpoint - To stop execution whenever the value of an expression changes.

- watch arg - The argument may be a variable or expression. GDB will stop execution when the value of a variable or expression changes.

Continuing and stepping

- continue(c) – To resume the program's execution from a certain breakpoint to a next breakpoint.
- step(s) - Executes one line of source code (or) one machine instruction for si command. Steps inside any functions called within the line.
- next(n) - Executes one line of source code (or) one machine instruction for ni command. If instruction is a function call, proceed until the function returns.

Examining the stack

- frame arg - allows to move from one stack frame to another, and to print the stack frame.
- Select-frame - allows to move from one stack frame to another without printing the frame.
- backtrace(bt) – prints the backtrace of entire stack.
- bt full - print the values of the local variables also.

Examining the source code

- list(l) arg – prints the source code. The argument may be a line number, address, function.
- list - - print the lines just before the lines last printed.
- By default 10 lines will be printed. We can change the number of lines that has to be printed with command: set listsize:
- list linenum - Print lines centered around line number.

Examining the data

- print(p) /f arg – The arg may be expression or variable. Here, 'f' is format specifier.
- x -- print the integer in hexadecimal.
- d – print as integer in signed decimal.
- u – print as integer in unsigned decimal.
- o – print as integer in octal.
- t – print as integer in binary.
- display /f arg – To display the value of a variable or expression each time the program stops. Display does not repeat if we press RET after using it.

Examining the memory

- x/ nfu addr – 'n' is the repeat count. 'f' is the format specifier. 'u' is the unit size. This may be byte(b), halfword(h), word(w), giant words(g) . Display the contents from a particular address.

- For example, `x/3uh 0x54320' is a request to display three halfwords (h) of memory, formatted as unsigned decimal integers (`u'), starting at address 0x54320.

Change the value of variable

- print x=4 – To change the value of x and print.
- setvar – To change the value of x and do not print the value.

Examining the symbol table

- info functions – prints a list of all function names.
- whatis [arg] – prints the datatype of the argument.
- ptype [arg] - prints a detailed description of the type, instead of just the name of the type.

Saving contents from memory to a file

- dump [format] memory filename start_addr end_addr - Dump the contents of memory from start_addr to end_addr to filename in the given format.
- The format parameter may be any one of:
- binary

  Raw binary form.
- ihex

  Intel hex format.
- srec

  Motorola S-record format.
- tekhex

  Tektronix Hex format

Corefiles

- When program terminates abnormally (e.g. a segmentation fault), Linux creates a core file which is a binary image of our program at the time of execution.
- The name of core files is core.pid, where pid is the numeric process-id of the program at the time it was executing.
- We can use GDB to investigate where our program terminated by starting GDB with the name of the core file (e.g. gdb core.2367).

Snapshots

- checkpoint - Save a snapshot of the debugged program's current execution state. Each checkpoint is assigned a small integer id, similar to a breakpoint id.

- info checkpoints – print information about all checkpoints.

- delete checkpoint checkpointid - Delete the previously saved checkpoint identified by

  checkpoint-id.

Data Display Debugger(DDD)

- Data Display Debugger(DDD) is essentially just a graphical interface for GDB. We can point and click the commands with mouse, but the commands have the same names and do the same thing as the commands in GDB.

Conclusion

- We have now seen enough to try GBD out on your own.

- Remember that GDB comes built in with an excellent help system. Just type help in the (gdb) prompt and you will be presented with options of what you could need help with. For details about a specific command, use the syntax

  ```
  help <command>
  ```

- Another important point to note is the use of shortcuts (like 'q' for 'quit'). GDB lets you use shortcuts for commands when it is not ambiguous.

Large enough to Deliver, Small enough to Care

Global Village
IT SEZ
Bangalore

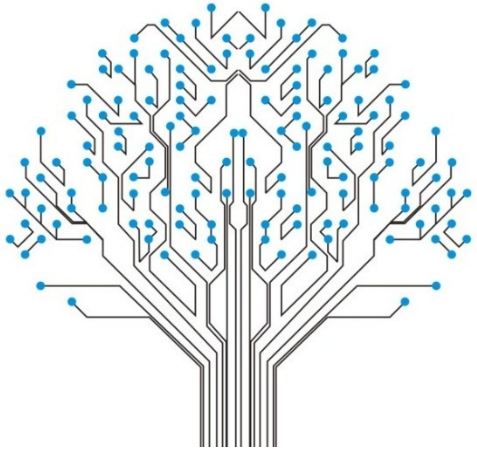
South Main Street
Milpitas
California

Raheja Mindspace
IT Park
Hyderabad

www.globaledgesoft.com

GLOBAL EDGE
Intelligence Of Things®

Thank you



Fairness

Learning

Responsibility

Innovation

Respect