Quick Facts about gdb

Ref : https://www.tutorialspoint.com/gnu_debugger/gnu_debugger_tutorial.pdf

Definition :

A debugger is a program that runs other programs, allowing the user to exercise control over these programs, and to examine variables when problems arise. GNU Debugger, which is also called gdb, is the most popular debugger for UNIX systems to debug C and C++ programs.

How does it help ?

GNU Debugger helps you in getting information about the following:

1. If a core dump happened, then what statement or expression did the program crash on?

2. If an error occurs while executing a function, what line of the program contains the call to that function, and what are the parameters?

3. What are the values of program variables at a particular point during execution of the program?

4. What is the result of a particular expression in a program?

How GDB Debugs?

GDB allows you to run the program up to a certain point, then stop and print out the values of certain variables at that point, or step through the program one line at a time and print out the values of each variable after executing each line. GDB uses a simple command line interface.

Points to Note

- Even though GDB can help you in finding out memory leakage related bugs, but it is not a tool to detect memory leakages.
- GDB cannot be used for programs that compile with errors and it does not help in fixing those errors.

Getting Started:

Starting and Stopping

· gcc -g myprogram.c -o myprog

Compiles myprogram.c with the debugging option (-g). You still get an a.out, but it contains debugging information that lets you use variables and function names inside GDB, rather than raw memory locations (not fun).

· gdb myprog

Opens GDB with file myprog, but does not run the program. You'll see a prompt

(gdb)

- all examples are from this prompt.

Basic gdb commands :

GDB offers a big list of commands, however the following commands are the ones used most frequently:

· b main - Puts a breakpoint at the beginning of the program

· b - Puts a breakpoint at the current line

· b N - Puts a breakpoint at line N of current file

· b +N - Puts a breakpoint N lines down from the current line of current file

· b fn - Puts a breakpoint at the beginning of function "fn"

· d N - Deletes breakpoint number N

· info break - list breakpoints

· l - list the lines of your source-code, in order to navigate through your program.

· r - Runs the program until a breakpoint or error or end

· c - Continues running the program until the next breakpoint or error or end

· f - Runs until the current function is finished

· s - Runs the next line of the program · s N - Runs the next N lines of the program

· n - Like s, but it does not step into functions

· u N - Runs until you get N lines in front of the current line

· p var - Prints the current value of the variable "var"

· bt - Prints a stack trace

· u - Goes up a level in the stack

· d - Goes down a level in the stack

· q - Quits gdb

You can find it handy : http://users.ece.utexas.edu/~adnan/gdb-refcard.pdf

Some more Ref :

http://www.cabrillo.edu/~shodges/cs19/progs/guide_to_gdb_1.1.pdf

https://cs.brown.edu/courses/cs033/docs/guides/gdb.pdf

https://beej.us/guide/bggdb/