GDB

Ananda Gunawardena

Debugging

- The process of finding runtime errors in the code
- Debugging can be done using
  - simple print statements
  - a debugger
- Debugging is done
  - to remove any runtime errors
  - to produce the correct output

Types of errors in C programs

A) dereference of uninitialized or otherwise invalid pointer
B) insufficient (or none) allocated storage for operation
C) storage used after free
D) allocation freed repeatedly
E) free of unallocated or potentially storage
F) free of stack space
G) return, directly or via argument, of pointer to local variable
H) dereference of wrong type
I) assignment of incompatible types
J) program logic confuses pointer and referenced type
K) incorrect use of pointer arithmetic
L) array index out of bounds

What is GDB?

- GDB stands for GNU Debugger
  - Written for many of the standard processors
- It is a powerful text debugger that will let you:
  - You can stop program at specified location.
  - See what has happened when program stopped. Look at the value of the variables
  - Change things in your program, so you can experiment with correcting the effects of one bug and go on to learn about another.

What GDB can do

- GDB can trace and alter program execution
- User can monitor and modify values of program variables
- Can call functions independent of program behavior

Before GDB

> gcc myprogram.c –o myprogram

> ./myprogram input.txt

Segmentation fault

> Why?!

Why?!: command not found

How to use GDB

- Compile your programs using gcc with the -g flag
  - > gcc myprogram.c -ggdb -o myprogram
- Start GDB
  - > gdb ./myprogram

GDB Commands: Control

- r(un) [arglist]
  - Runs your program in GDB with optional argument list
- b(reak) [file:]function/line
  - Puts a breakpoint in that will stop your program when it is reached
- c(ontinue)
  - Resumes execution of your program after it is stopped
- n(ext)
  - When stopped, runs the next line of code, stepping over functions
- s(tep)
  - When stopped, runs the next line of code, stepping into functions
- q(uit)
  - Exits GDB

GDB Commands: Getting info

- print expr
  - Prints out the given expression
- display var
  - Displays the given variable at every step of execution
- l(ist)
  - Lists source code
- help [command]
  - Gives you help with a specified command
- bt
  - Gives a backtrace (Lists the call stack with variables passed in)

Example: Segfault!

- We have a simple program, example1.c that segfaults upon running
- Open it up in GDB, and simply run it
- GDB will tell you where it segfaulted, and let you print out information to help specify why

Example: Arrays

- We have a program randints.c, that fills an array with random values and then prints them out
- Somehow the program seg faults
- Run GDB and step through it one step at a time to determine where the problem might have occured

Example: Strings

- We have written our own string copy function in stringcopy.c, but it doesn’t work.
- Open it up in GDB and see where the original and copied strings differ

Conclusion

- When your C program seg-faults GDB can help
- There are instances where GDB may not be able to help
- Simple code tracing is what we need to do then