Using the gdb

What is gdb

- GDB is a debugger that helps you debug your program.
- The time you spend now learning gdb will save you days of debugging time.
- A debugger will make a good programmer a better programmer.

Compiling a program for gdb

- You need to compile with the "-g" option to be able to debug a program with gdb.
- The "-g" option adds debugging information to your program

```
g++ -g -o hello hello.cpp
```

Running a Program with gdb

- To run a program with gdb

  gdb hello

- Then set a breakpoint in the main function.

  (gdb) break main

  – A breakpoint is a marker in your program that will make the program stop and return control back to gdb.

- Now run your program.

  (gdb) run

Stepping Through your Program

- Your program will start running and when it reaches "main()" it will stop.

  gdb>

- Now you have the following commands to run your program step by step:
  - **(gdb) step:** It will run the next line of code and stop. (If it is a function call, it will enter into it)
  - **(gdb) next:** It will run the next line of code and stop. If it is a function call, it will not enter the function and it will go through it.
- Example:

  (gdb) step

  (gdb) next

Setting breakpoints

- You can set breakpoints in a program in multiple ways:

(gdb) break function

Set a breakpoint in a function E.g.

(gdb) break main

(gdb) break line

Set a break point at a line in the current file. E.g.

(gdb) break 66

It will set a break point in line 66 of the current file.

(gdb) break file:line

It will set a break point at a line in a specific file. E.g.

(gdb) break hello.c:78

Regaining the Control

- When you type

  (gdb) run

  the program will start running and it will stop at a break point.

- If the program is running without stopping, you can regain control again typing ctrl-c.

Where is your Program

- The command

  (gdb) where

  Will print the current function being executed and the chain of functions that are calling that fuction.

  This is also called the backtrace.

  Example:

```
(gdb) where
#0  main () at test_mystring.c:22
(gdb)
```

Printing the Value of a Variable

- The command

  (gdb) print var

  Prints the value of a variable.

E.g.

```
(gdb) print i
$1 = 5
(gdb) print s1
$1 = 0x10740 "Hello"
(gdb) print stack[2]
$1 = 56
(gdb) print stack
$2 = {0, 0, 56, 0, 0, 0, 0, 0, 0, 0}
(gdb)
```

Exiting gdb

- The command "quit" exits gdb.

```
(gdb) quit
The program is running.  Exit anyway? (y or n) y
```