

Introduction to GDB:

- GDB stands for GNU Debugger and is a debugger for several languages, including C and C++.
- To use GDB, you need to add the -g flag when you run gcc to compile your code.
- A **breakpoint** is like a stop sign in your code. Whenever gdb gets to a breakpoint it halts execution of your program and allows you to examine it.

Commands	Description
Help	
help	List the gdb command topics.
help <i>class</i>	List the gdb commands within the specified class.
help <i>command</i>	Give a description of the specified command.
Running	
run/r run/r <i>command-line arguments</i> run/r <i>file filename</i>	Start program execution from the beginning of the program.
q/quit	Exit GDB.
kill	Stop the program execution.
BreakPoints	
break <i>line-number/function-name</i>	Suspend the program at specified function of line number.
delete <i>breakpoint</i>	Deletes the specified breakpoint.
delete/d	Delete all breakpoints.
clear <i>function/line</i>	Delete all breakpoints in the given function or line.
Stepping	
continue/c	Continue executing until the next breakpoint.
next/n	Execute the next line of code. Will not enter functions.
step/s	Step to the next line of code. Will enter functions.
until <i>line-number</i>	Continue processing until you reach a specified line number. until is like next, except that if you are at the end of a loop, until will continue execution until the loop is exited, whereas next will just take you back up to the beginning of the loop. This is convenient if you want to see what happens after the loop, but don't want to step through every iteration.
finish	Continue until the current function returns.
where	Shows current line number and which function you are in.
print <i>variable-name</i>	Prints the value stored in the specified variable.
Stack	
up	Move up a single frame (element in the call stack)
down	Move down a single frame
up/down <i>number</i>	Move up/down by the specified number of frames in the stack.