Cheatography

Best GDB Cheat Sheet

by MeLikeyCode via cheatography.com/121889/cs/22457/

Launching

gdb -p <pid>	attach gdb to an existing process
gdb --args <exe_file> [arg1] [arg2] ...	make gdb launch an executable file, passing some arguments to the executable file
gdb --batch –x <command_file> --args <exe_file> [arg1] [arg2] …	make gdb launch an executable file (passing some arguments to it), then have gdb execute gdb commands from a "command file"

Execution Control

continue	continue execution of the process until a breakpoint (or catchpoint) is hit
next	continue execution of the process until the next line of code
step	continue the execution of the process until the first instruction of the next function (aka "step into")
finish	continue execution of the process until the current function returns

"Ctr + c" (while the gdb terminal window is focused) to break the process right where it is at.

Viewing Source

layout next	enable tui mode (top part of terminal will show source location of where you're currently broken)
tui disable	disable tui mode
list	print next 10 lines of code
list -	print previous 10 lines of code
list <file>:<function>	print 10 lines of code around a specific function in a file

Viewing Variables

print <variable_name>	print a variable (must be in scope of course)
backtrace	print callstack
backtrace -n	print top n frames of callstack (n can be any number)
frame n	select frame number n (nth frame from the top). Frame is another word for a particular function in a callstack.
up	select the next frame up
down	select the next frame down
info args	print the arguments passed to the selected frame (function)
info locals	print the local variables defined in the selected frame

By **MeLikeyCode**

cheatography.com/melikeycode/

Not published yet.
Last updated 20th April, 2020.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Breakpoints	
info breakpoints	print information about breakpoints/catchpoints that you've placed
break <function-_name>	place a breakpoint on a particular function. Write the full name of the function, as you would specify it in C++ (e.g. ClassName::function_name). If you have multiple functions with the same name, include argument types to disambiguate.
break <file>:<line-_number>	place a breakpoint on a particular line of a particular file
delete <breakpoint_number>	delete (remove) a particular breakpoint. "info breakpoints" will show you the breakpoint numbers for all the breakpoints that you've placed.
disable <breakpoint_number>	disable a particular breakpoint. A disabled breakpoint still shows up when you do "info breakpoints", but it won't be hit until you enable it.
enable <breakpoint_number>	enable a particular breakpoint
break <function> if <condition>	place a conditional breakpoint (i.e. only break if a certain condition is true). The condition is any valid C++ expression that evaluates to true or false. You can use convenience variables as well as any variables in scope for the condition.
condition <breakpoint_number> [condition]	define a condition for an already placed breakpoint. If you leave out the [condition], then you are saying to make the breakpoint unconditional (i.e. remove any existing conditions, if there are any).
commands <breakpoint_number>	specify gdb commands that should automatically run whenever the breakpoint is hit. After you run this command, typing a bunch of newline seperated commands that should be executed when the breakpoint is hit. After you are done entering your commands, type "end".

By **MeLikeyCode**

cheatography.com/melikeycode/

Not published yet.
Last updated 20th April, 2020.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com