Common gdb Commands

Command	**Description**
help	List gdb command topics.
help *topic-classes*	List gdb command within class.
help *command*	Command description.
apropos *search-word*	Search for commands and command topics containing *search-word*.
info args i args	List program command line arguments
info breakpoints	List breakpoints
info break	List breakpoint numbers.
info break *breakpoint-number*	List info about specific breakpoint.
info watchpoints	List breakpoints
Break and Watch	

break *funtion-name*
b *funtion-name*
break *line-number*
`break ClassName::functionName`
	Suspend program at specified function of line number.

break +*offset*
break -*offset*
	Set a breakpoint specified number of lines forward or back from the position at which execution stopped.

break *filename:function*	Don't specify path, just the file name and function name.
break *filename:line-number*	Don't specify path, just the file name and line number. `break Directory/Path/filename.cpp:62`
break *line-number* if *condition*	Where condition is an expression. i.e. `x > 5` Suspend when boolean expression is true.
tbreak	Temporary break. Break once only. Break is then removed. See "break" above for options.
watch *condition*	Suspend processing when condition is met. i.e. `x > 5`
clear clear *function* clear *line-number*	Delete breakpoints as identified by command option.
delete d	Delete all breakpoints, watchpoints, or catchpoints.
delete *breakpoint-number* delete *range*	Delete the breakpoints, watchpoints, or catchpoints of the breakpoint ranges specified as arguments.
disable *breakpoint-number-or-range* enable *breakpoint-number-or-range*	Does not delete breakpoints. Just enables/disables them. Example: Show breakpoints: `info break` Disable: `disable 2-9`
enable *breakpoint-number* once	Enables once
continue c	Continue executing until next break point/watchpoint.
continue *number*	Continue but ignore current breakpoint *number* times. Usefull for breakpoints within a loop.
finish	Continue to end of function.

Line Execution	
step s step *number-of-steps-to-perform*	Step to next line of code. Will step into a function.
next n next *number*	Execute next line of code. Will not enter functions.

until
until *line-number*
	Continue processing until you reacha aspecified line number. Also: function name, address, filename:function or filename:line-number.

where	Shows current line number and which function you are in.
Source Code	
list l list *line-number* list *function* list - list *start#,end#* list *filename:function*	List source code.
set listsize *count* show listsize	Number of lines listed when `list command given.`
Examine Variables	
print *variable-name* p *variable-name* p *file-name::variable-name* p '*file-name*'::*variable-name*	Print value stored in variable.
Start and Stop	
run r run *command-line-arguments* run < *infile* > *outfile*	Start program execution from the beginning of the program. The command `break main` will get you started. Also allows basic I/O redirection.
continue c	Continue execution to next break point.
kill	Stop program execution.
quit q	Exit GDB debugger.