<where>			
line_num	Line number in current src file.	file:line_num	Line number in the named src file.
func_name	Named function.	file:func_name	Function defined in named file.
<what>			
expression	Almost any C expr.	<file>::<var>	var value defined in the file (static)
<func>::<var>	var val defined in the func (on stack).	{type}address	Content at addr, as C type.
$register	Content of named register.		
run			
gdb binary [core]	Start GDB (with optional core dump).	(gdb)file binry	Start GDB, and then load binary.
gdb --pid <pid> or (gdb) attach PID	Start GDB and attach to process. Whenever GDB attaches to a running process, the process is paused, so you can get a handle on what the call stack look like and change variable values. When detached from GDB, the process will continue along its merry way.		
set args <args…>	Set args to pass to program. If no paras, then run stop automatically passing arguments.		
run [args]	Run will always use the arguments u just used, until u tell it to use different arguments.		
kill	Kill the running program		
sources			
directory <dir>	Add dir to the list of dirs for search	set listsize n	Set how many lines to show in "list"
list	Next 10 lines (1st:centered on main())	list -	Previous 10 lines
list <file>:<line>	List 10 lines centered on <line>	list <f>:<fun>	List 10 lines centered on <fun>
list <first>,<last>	List a range of lines. [5,]:start line 5; [,28]:end line 28		
stack			
backtrace/bt [full]		Where [full]	Show call stack, full: local vars
frame <#>	Select the stack frame to operate on.		
breakpoints/watchpoints			
break <where>	Set a new break point.	watch <whre>	Set a new watchpoint.
delete <#>	Remove a point.	clear <where>	Clear points at <where> (no wh, all)
enable <#>	Enable a disabled point.	disable <#>	Disable a point.
stepping			
step/s [count]	Go to next source line, diving into function at that line.		
next/n [count]	Go to next source line, but don't dive into functions.		
finish	Continue until the current func rtns.	continue	Continue normal execution.
format			
a	pointer	c	Read as integer, print as char
d/u	Integer, signed/unsigned decimal	f	Floating point number.
o/x/t	Integer, print as octal / hex / binary.	s	Try to treat as C string.
variables and memory			
print/fmt <what>	Print content of var/addr/register. p (double)var; p myIntArr; p myIntArr[3]; p myIntArr[3]@5; p myStruct; p myStruct.name.		
display/fmt <what>	Frequently print variables can be added to automatic display list so that GDB prints their values each time the program stops.		
undisplay <#>	Remove "display" with the given #	ptype var	Print variable type.
enable display <#>	Enable display.	disable display <#>	Disable display.
x/nfu <addr>	Print memory. n: how many units to print (default 1); f: format character; u: unit. Unit is one of: b-Byte, h-halfword (two bytes), w-word(four bytes), g-giant word(8 bytes)		
manipulating the program			
set var name=val	Change variable to the given value.		
return expr	You can cancel execution of a function call with the return command. If you give an expression argument, its value is used as the function's return value.		
informations			
disassemble [<where>]	Disassemble the current function or given location.	info args	Print the arguments to the function of the current stack frame.
info breakpoints	Print info of break- and watchpoints	info display	Print info of "displays"
info locals	Print local variables in current frame	info sharedLibrary	List loaded shared libraries.
info signals	List all signals and how are handled	info threads	List all threads
show directories	Print all dirs where GDB searches	show listsize	Print how many are shown in "list"
whatis var_name	Print type of named variable		
other			
shell cmd / !cmd	Execute shell cmds in GDB	make args	Execute make (i.e. shell make args)
<RET>	Repeat the previous cmd ('run' not)		