

### Run GDB

<code>gdb &lt;program_path&gt;</code>	Load program into gdb
<code>gdb &lt;program_path&gt; &lt;core_path&gt;</code>	Load program and core dump into gdb

### Breakpoints

<code>break</code>	Set break point at the current location
<code>break if &lt;condition&gt;</code>	Set break point here that triggers if certain condition is met
<code>break &lt;code_location&gt;</code>	Set break point at given code location
<code>break &lt;code_location&gt; if &lt;condition&gt;</code>	Set break point at given code location that triggers if given condition is met
<code>hbreak</code>	works exactly like <code>break</code> but it is hardware assisted breakpoints
<code>info breakpoints</code>	List all breakpoints and their associated numbers
<code>clear</code>	Delete all break points
<code>delete &lt;breakpoint_number&gt;</code>	Delete breakpoint given its number
<code>enable &lt;breakpoint_number&gt;</code>	Enable breakpoint given its number
<code>disable &lt;breakpoint_number&gt;</code>	Disable breakpoint given its number

### code\_location

<code>function_name</code>	self-explanatory
<code>*function_name + offset</code>	move <code>offset</code> bytes from <code>function_name</code>
<code>*math_expr</code>	pointer evaluated from the math expression

### Stepping

<code>run</code>	Run the loaded program
<code>run &lt;arguments&gt;</code>	Run loaded program with given arguments
<code>attach &lt;pid&gt;</code>	Attach debugger to given process
<code>next</code>	Next line of source code
<code>step</code>	Same as <code>next</code> but will dive into calls
<code>nexti</code>	Next assembly instruction
<code>stepi</code>	same as <code>nexti</code> but will dive into calls
<code>finish</code>	Continue till first ret instruction
<code>continue</code>	Continue till next breakpoint

### Examining code

<code>backtrace</code>	Print current backtrace
<code>disassemble &lt;function_name&gt;</code>	Disassemble given function

### Memory

<code>print/&lt;format&gt; &lt;expression&gt;</code>	Evaluate expression and print it in given format
<code>display/&lt;format&gt; &lt;expression&gt;</code>	Same as <code>print</code> however it keeps executing after each <b>step</b> instruction
<code>info display</code>	List all auto-display expressions and their numbers
<code>enable display &lt;number&gt;</code>	Enable display given its number
<code>disable display &lt;number&gt;</code>	Disable display given its number
<code>x/nuf &lt;address&gt;</code>	Examine memory. n: How many units to print (default 1). f: Format character (like "print"). u: Unit. Unit is one of: b: Byte h: Half-word (two bytes) w: Word (four bytes) g: Giant word (eight bytes).

### Format

a	Pointer
c	Character
d	Signed integer
f	Floating point number
i	instruction
o	octal
s	C-type strings
t	Binary
u	Unsigned integer
x	Hexadecimal

### General information

<code>info sharedlibrary</code>	List loaded shared libraries
<code>info proc mappings</code>	list of mapped memory regions.