# GDB Cheat Sheet

| Basics | |
|---|---|
| `$ gcc -g ...` | create an executable that can be debugged using GDB |
| `$ gdb progName` | start debugging `progName` |
| `$ gdb --args progName args` | start debugging `progName`, using command-line arguments `args` |
| `(gdb) q` | quit GDB |
| `(gdb) help command` | display information about `command`, incl. its syntax |
| `(gbd) run` | start running program |
| `(gbd) kill` | terminate currently running program |

| Examining Data | |
|---|---|
| `print expr` | show current value of expression `expr` |
| `print var->attr`<br>`print *arr@len` | show current value of attribute `attr` of struct `var`<br>show current value of first `len` elements of array `arr` |
| `print/format expr` | show current value of expression `expr` in format `format` |
| `print/x expr`<br>`print/t expr`<br>`print/c expr`<br>`print/f expr`<br>`print/s expr` | show current value of `expr` in hexadecimal<br>show current value of `expr` in binary<br>show current value of `expr` as an integer and its character representation<br>show current value of `expr` in floating point syntax<br>show current value of `expr` as a string, if possible |
| `display expr` | automatically print value of expression `expr` at each halt in execution |
| `undisplay disp#` | stop displaying expression with display number `disp#` |
| `watch expr` | set a watchpoint on expression `expr` (break whenever value of `expr` changes) |
| `info args` | show value of all arguments to current function |
| `info locals` | show current value of all local variables |
| `x addr` | show current word in memory at address `addr`, in hexademical |
| `x/units\|format\|size addr` | show current value of memory of size `units` x `size` at address `addr`, in format `format` |
| `x/3tb addr` | show current value of 3 bytes of memory at address `addr`, in binary |

| Examining the Stack | |
|---|---|
| `backtrace` | display the current call stack (can be used after a runtime error, eg. segfault) |

Gabrielle Singh Cadieux, 2017

| Breakpoints | |
|---|---|
| `break point` | create a breakpoint at `point` |
| `break 5`<br>`break func`<br>`break foo.c:5` | create a breakpoint at line 5 of current source file<br>create a breakpoint at body of function `func`<br>create a breakpoint at line 5 of source file `foo.c` |
| `break point if cond` | create a breakpoint at `point` which triggers if Boolean expression `cond` evaluates to *true* |
| `info breakpoints` | display information about all current breakpoints |
| `delete` | remove all breakpoints |
| `delete breakpoint#` | remove breakpoint with number `breakpoint#` |

| Continuing and Stepping | |
|---|---|
| `continue` | continue executing normally |
| `finish` | continue executing until current function returns |
| `step` | execute next line of source code |
| `next` | execute next line of source code, without descending into functions |

| Altering Execution | |
|---|---|
| `return expr` | return from current function at this point, with return value `expr` |
| `set var var=expr` | store value of expression `expr` into program variable `var` |
| `set var g=4` | store 4 into program variable `g` |
| `set {type}addr = expr` | store value of expression `expr` (represented as type `type`) into memory at address `addr` |
| `set {int}0x83040 = 4` | store 4 as an int at address 0x83040 |
| `signal signal` | continue executing and immediately send signal `signal` to the program |
| `signal SIGINT` | continue executing and immediately send an interrupt signal to the program |

Full GDB documentation: https://sourceware.org/gdb/current/onlinedocs/gdb/index.html

Gabrielle Singh Cadieux, 2017