

Stopping Again

q[uit] - Quit. Note: To quit without an 'are you sure?' prompt, use quit *unconditionally* (*q!*)

kill - Really quit. This uses *kill -9*, for situations where quit just isn't fierce enough

Essential Commands

c[ontinue] **<line-number>** - Carry on running until program ends, hits a breakpoint or reaches line *line-number* (if specified)

n[ext] **<number>** - Go to next line, stepping over function calls. If *number* specified, go forward that number of lines

s[tep] **<number>** - Go to next line, stepping into function calls. If *number* is specified, make that many steps

b[ack]t[race] — a.k.a. "w[here]" - Display stack trace

h[elp] **<command-name>** - Help. When passed the name of a command, gives help on using that command

Program Stack

b[ack]t[race] — a.k.a. "w[here]" - Display stack trace

f[rame] **<frame-number>** - Moves to **<frame-number>** (frame numbers are shown by *bt*). With no argument, shows the current frame

up **<number>** - Move up **<number>** frames (or 1, if no number specified)

down **<number>** - Move down **<number>** frames (or 1, if no number specified)

info args - Arguments of the current frame

info locals - Local variables in the current stack frame

info instance_variables - Instance variables in the current stack frame

info global_variables - Current global variables

info variables - Local and instance variables of the current frame

m[method] **<class>****module** - Shows instance methods of the given class or module

Program Stack (cont)

m[method] **i[instance]** **<object>** - Shows methods of **<object>**

m[method] **iv** **<object>** - Shows instance variables of **<object>**

v[ar] **cl[ass]** - Shows class variables of self

v[ar] **co[nst]** **<object>** - Shows constants of **<object>**

v[ar] **g[lobal]** - Shows global variables (same as *info global_variables*)

v[ar] **i[nstance]** **<object>** - Shows instance variables of **<object>** (same as *method iv <object>*)

v[ar] **l[ocal]** - Shows local variables (same as *info locals*)

Display

e[val] — a.k.a. "p" **<expression>** - Evaluate **<expression>** and display result. By default, you can also just type the expression without any command and get the same thing (disabled by using *set noautoeval*)

pp - Evaluate expression and pretty-print the result

putl - Evaluate an expression with an array result and columnize the output

ps - Evaluate an expression with an array result, sort and columnize the output

disp[lay] **<expression>** - Automatically display **<expression>** every time the program halts. With no argument, lists the current display expressions

info display - List all current display expressions

undisp[lay] **<number>** - Remove display expression number **<number>** (as listed by *info display*). With no argument, cancel all current display expressions

disable display **<number>** - Stop displaying expression number **<number>**. The display expression is kept in the list, though, and can be turned back on again using *enable display*

enable display **<number>** - Re-enable previously disabled display expression **<number>**

Breakpoints and Catchpoints

b[reak] - Sets a breakpoint at the current line. These can be conditional: *break if foo != bar*. Keep reading for more ways to set breakpoints!

b[reak] **<filename>**:**<line-number>** - Puts a breakpoint at **line-number** in **filename** (or the current file if **filename** is blank). Again, can be conditional: *b myfile.rb:15 unless my_var.nil?*

b[reak] **<class>****(.|#)****<method>** - Puts a breakpoint at the start of the **method** in **class**. Accepts an optional condition: *b MyClass#my_method if my_boolean*

info breakpoints - List all breakpoints, with status

cond[ition] **<number>** **<expression>** - Add condition **expression** to breakpoint number **<number>**. If no **expression** is given, removes any conditions from that breakpoint

del[ete] **<number>** - Deletes breakpoint **<number>**. With no arguments, deletes all breakpoints

disable breakpoints **<number>** - Disable (but don't delete) breakpoint **<number>**. With no arguments, disables all breakpoints

cat[ch] **<exception>** **off** - Enable or (with *off* argument) disable catchpoint on **<exception>**

cat[ch] - Lists all catchpoints

cat[ch] **off** - Deletes all catchpoints

sk[ip] - Passes a caught exception back to the application, skipping the catchpoint.

Controlling Byebug

hist[ory] **<num-commands>** - view last **<num-commands>** byebug commands (or all, if no argument given).

save **<file>** - saves current byebug session options as a script file in **<file>**

source **<file>** - loads byebug options from a script file at **<file>**

set **<option>** - change value of byebug option **<option>**

show **<option>** - view current value of byebug option **<option>**



Source Files and Code

reload - Reload source code

info file - Information about the current source file

info files - All currently loaded files

info lines - Shows the current line number and filename

l[ist] - Shows source code after the current point. Keep reading for more list options

l[ist] -- - Shows source code before the current point

l[ist] = - Shows source code centred around the current point

l[ist] <first>-<last> - Shows all source code from *<first>* to *<last>* line numbers

edit <file:lineno> - Edit *<file>*. With no arguments, edits the current file

Execution Control

c[ontinue] <line-number> - Carry on running until program ends, hits a breakpoint or reaches line *line-number* (if specified)

n[ext] <number> - Go to next line, stepping over function calls. If *number* specified, go forward that number of lines

s[tep] <number> - Go to next line, stepping into function calls. If *number* specified, make that many steps

fin[ish] <num-frames> - With no argument, run until the current frame returns. Otherwise, run until *<num-frames>* frames have returned

irb - Start an IRB session

restart - Restart the program. This also restarts byebug

Threads

th[read] Show current thread

th[read] l[ist] List all threads

th[read] stop <number> Stop thread number *<number>*

th[read] resume <number> Resume thread number *<number>*

th[read] <number> Switch context to thread *<number>*

