gdb cheat-sheet for reverse-engineering

Nota bene: character ‘ is a backquote (AltGr+7) !

Starting GDB

`gdb`	start GDB, with no debugging files
`gdb` *program*	begin debugging *program*
`gdb --args` *prg args*	begin debugging *prg args*

Stopping GDB

`quit`	exit GDB; also `q` or `EOF` (eg `C-d`)
`INTERRUPT`	(eg `C-c`) terminate current command, or send to running process

Getting Help

`help`	list classes of commands
`help` *class*	one-line descriptions for commands in *class*
`help` *command*	describe *command*

Executing your Program

`r[un]` *arglist*	start your program with *arglist*
`r[un] $(`*cmd*`)`	start your program with the output of command *cmd* as an argument
`r[un] ‘`*cmd*`‘`	
`r[un] ... <`*inf* `>`*outf*	start your program with I/O redirected
`r[un] ... <<<` *str*	start your program with *str* as standard input content
`r[un] ... < <(`*cmd*`)`	start your program with the output of command *cmd* as standard input content
`r[un] ... <<< $(`*cmd*`)`	
`r[un] ... <<< ‘`*cmd*`‘`	
`kill`	kill running program
`set args` *arglist*	specify *arglist* for next `run`
`set args`	specify empty argument list
`show args`	display argument list
`set disable-randomization [on|off]`	disable ASLR

Breakpoints and Watchpoints

`break` [*file*`:`]*line*	set breakpoint at *line* number [in *file*]
`b` [*file*`:`]*line*	eg: `break main.c:37`
`break` [*file*`:`]*func*	set breakpoint at *func* [in *file*]
`break [+|-]`*offset*	set break at *offset* lines from current stop
`break *`*addr*	set breakpoint at address *addr*
`break`	set breakpoint at next instruction
`catch` *event*	break at *event*, which may be `catch`, `throw`, `exec`, `fork`, `vfork`, `load`, or `unload`.
`info break`	show defined breakpoints
`delete` [*n*]	delete breakpoints

[ ] surround optional arguments ... show one or more arguments

Program Stack

`backtrace` [*n*]	print trace of all frames in stack; or of *n* frames—innermost if *n*>0, outermost if *n*<0
`bt` [*n*]	
`frame` [*n*]	select frame number *n* or frame at address *n*; if no *n*, display current frame
`up` *n*	select frame *n* frames up
`down` *n*	select frame *n* frames down
`info frame` [*addr*]	describe selected frame, or frame at *addr*
`info args`	arguments of selected frame
`info locals`	local variables of selected frame
`info reg` [*rn*]...	register values [for regs *rn*] in selected frame; `all-reg` includes floating point
`info all-reg` [*rn*]	

Execution Control

`continue` [*count*]	continue running; if *count* specified, ignore this breakpoint next *count* times
`c` [*count*]	
`step` [*count*]	execute until another line reached; repeat *count* times if specified
`s` [*count*]	
`s[tep]i` [*count*]	step by machine instructions
`next` [*count*]	execute next line, including any function calls
`n` [*count*]	
`n[ext]i` [*count*]	next machine instruction
`until` [*location*]	run until next instruction (or *location*) or the current stack frame returns
`finish`	run until selected stack frame returns
`return` [*expr*]	pop selected stack frame without executing [setting return value]
`jump` *line*	resume execution at specified *line* number or *address*
`jump *`*address*	
`set var=`*expr*	evaluate *expr* without displaying it;

Working Files

`file` [*file*]	use *file* for both symbols and executable; with no arg, discard both
`core` [*file*]	read *file* as coredump; or discard
`exec` [*file*]	use *file* as executable only; or discard
`symbol` [*file*]	use symbol table from *file*; or discard
`load` *file*	dynamically link *file* and add its symbols
`add-sym` *file addr*	read additional symbols from *file*, dynamically loaded at *addr*
`info files`	display working files and targets in use
`path` *dirs*	add *dirs* to front of path searched for executable and symbol files
`show path`	display executable and symbol file path
`info share`	list names of shared libraries currently loaded

Display

`print` [`/`*f*] [*expr*]	show value of *expr* [or last value `$`] according to format *f*:
`p` [`/`*f*] [*expr*]	
`x`	hexadecimal
`d`	signed decimal
`u`	unsigned decimal
`o`	octal
`t`	binary
`a`	address, absolute and relative
`c`	character
`f`	floating point
`call` [`/`*f*] *expr*	like `print` but does not display `void`
`x` [`/`*Nuf*] *expr*	examine memory at address *expr*; optional format spec follows slash
N	count of how many units to display
u	unit size; one of: `b` individual bytes, `h` halfwords (two bytes), `w` words (four bytes), `g` giant words (eight bytes)
f	printing format. Any `print` format, or `s` null-terminated string, `i` machine instructions
`disassem` [*addr*]	display memory as machine instructions

Debugging Targets

`target` *type param*	connect to machine, process, or file; e.g. `target remote | sshpass -p`*pw* `ssh -T` [`-p` *port*] [user`@`]*host* `gdbserver -` *prog* [*args*]
`attach` *param*	connect to another process
`detach`	release target from GDB control

Source Files

`dir` *names*	add directory *names* to front of source path
`dir`	clear source path
`show dir`	show current source path
`list`	show next ten lines of source
`list -`	show previous ten lines
`list` *lines*	display source surrounding *lines*, specified as:
[*file*`:`]*num*	line number [in named file]
[*file*`:`]*function*	beginning of function [in named file]
`+`*off*	*off* lines after last printed
`-`*off*	*off* lines previous to last printed
`*`*address*	line containing *address*
`list` *f*`,`*l*	from line *f* to line *l*