GDB Cheatsheet

Running

```
# gdb <program> [core dump]
# gdb --args <program> <args...>
# gdb --pid <pid>
(gdb) set args <args...>
(gdb) run
(gdb) kill
(gdb) quit
```

Use script file

```
source [-s] [-v] file
```

Conditions

```
condition BPT_N CONDITION
```

Stepping

Step *into*	Step *over*
`step [N]`	`next [N]`
`stepi`	`nexti`
Step *machine* instruction	
Continue until function return	Continue normal execution
`finish`	`continue`

Information

Info about the program being debugged

```
info sharedlibrary
     args | threads | ...
```

Show register value

```
info registers $eax
```

Info about the debugger

```
show directories
     listsize | width | ...
```

Show limit on chars to print

```
show print elements
```

Disassemble a specified section of memory.

```
disassemble [/m|/r|/s] START [, END]
```

Signals

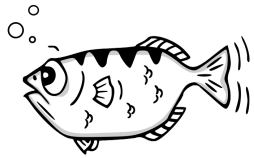
Specify how to handle signals

```
handle SIGNAL [ACTIONS]
```

stop / nostop
print / noprint
pass / nopass

Do not stop on all signals

```
handle all nostop
```



Examining the stack

Print backtrace

backtrace | where | bt

```
backtrace [-full]
```

Print a stack frame

frame | f

```
frame [FRAME_NUM]
```

Breakpoints

break | brea | bre | br | b

```
break [LOCATION] [if CONDITION]
```

hardware br

```
hbreak
```

Break at address when condition is met

```
b *0xCAFEBABE if $eax == 0x42
```

Break at function

```
b func_name
```

one-time br

```
tbreak
```

Subcommand prefixes

```
info  enable  disable  delete  clear
```

Watchpoints

Stop exec when the value of the expression changes

on write

```
watch [-location] EXP
```

on read

```
rwatch
```

Watch the value of variable

```
watch foo_var
```

read/write

```
awatch
```

Watch on mem read at addr

```
rwatch *0xCAFEBABE
```

Commands in breakpoint

Commands to be executed with given breakpoints.
Default - last breakpoint

```
commands [BPT_NUM]
  command-list
end
```

Print variables / memory

Print value of expression

```
print [[OPTION]... --] [/FMT] [EXP]
```

Aliases

print | inspect | p

Print 4 uint from addr in RBX

```
print /d *$rbx@4
```

Print hex value of EAX

```
print /x $eax
```

Print value of expression ***each time*** the program stops

```
display [/FMT] EXP
```

```
undisplay [NUM]
```

Subcommand prefixes

```
info  enable  disable  delete
```

Print current instr-n

```
display /i $pc
```

Print memory contents

```
x /nfu ADDRESS
```

n — Number of units to print (default 1)

f — Format character (like in „print“)

u — Unit
b: byte [1]
h: half [2]
w: word [4]
g: giant [8]

Print memory from addr in register

```
x /32xb $sp
x /64xg $eax
```

Print string from addr

```
x /s 0x05ab81f4
```

Pseudo registers

`$pc`	program counter
`$sp`	stack pointer
`$fp`	stack frame
`$ps`	processor status

[LOCATION]

function_name

```
main
```

line_number

```
28
```

file:line_num

```
test.c:28
```

*address

```
*0xCAFEBABE
*main + 4
```

[EXP]

```
C-like expression
file_name::variable_name
func_name::variable_name
{type} address
start@NumElements
```

FMT (format)

a	Pointer		
c	Integer, print as character	s	Try to treat as C string
d	Integer, signed decimal	t	Integer, print as binary („two“)
f	Floating point number	u	Integer, unsigned decimal
i	Instruction	x	Integer, print as hexadecimal
o	Integer, print as octal	z	Hex, zero padded on the left