Useful gdb Commands

Commands that are especially useful in this week’s exercises

break – Set software breakpoint

For setting an INT 3 breakpoint
Usage: **break *0x<hex_addr>**

or: **break <location_name>**

(only if symbols are present)

or: **break <filename>:<line_num>**

(only if debugging information is present)

enable and disable commands with:

enable <command_number>

and

disable <command_number>

hbreak – Set hardware breakpoint

For setting hardware breakpoints
Usage: similar to **break**

Important note: Because of hardware limitations only a limited number of HW breakpoints can be set.

gdb won’t alert you about exceeding the limit when setting the breakpoint, but only after resuming execution.

If setting a HW breakpoint before the program is loaded, gdb may warn you about “No hardware breakpoint support in the target.” running the program (using e.g. starti) should fix this

watch – Set hardware watchpoint

For setting hardware watchpoints (memory write breakpoints)

Uses the same HW mechanism as **hbreak** so its use is also limited

Program pauses execution when the location is written to

As with **break** and **hbreak**, addresses should be preceded by “*****”

rwatch – Set hardware read watchpoint

Similar to **watch** only it pauses when the memory is read

commands – Execute gdb commands on breakpoint

A very useful command, it lets you specify which commands to automatically execute when encountering a breakpoint of any kind.

Usage:

commands <breakpoint_number>

[enter gdb commands in multiple lines]

end

The last command could be **c** (**continue**) such that things are done automatically and no user interaction is needed

set – Set a C expression to a certain value

As before you can use an address preceded by an asterisk but you have to specify what type of variable it references (in C-style notation)

For example: **set *(char *)(0x555555554100) = 0xCC**

Alternatively: **set {char}0x555555554100 = 0xCC**

You can also set register values this way (names of resiters are preceded by "**$**", e.g:

set $rax=0

dump binary memory – Dump a section of memory to file

Dumps a section of memory as it is in the current stage of execution to a binary file. This file can later be used to patch an executable, for example.