基本操作

執行環境

程式的參數 (Program's Arguments)
set args -- 設定程式的參數
show args -- 顯示 set args 所設定的程式參數
工作目錄 (Program's Working Directory)
cd -- 改變工作目錄，和在 shell 下使用 cd 相同
pwd -- 顯示工作目錄
環境變數 (Environment Variables)
show environment -- 顯示所有環境變數的內容
show environment *varname* -- 顯示所指定環境變數的內容
set environment *varname [=value]* -- 設定環境變數
unset environment *varname* -- 取消環境變數設定
path *DIR* -- 將 DIR 加到 PATH 中
path -- 顯示 PATH 的內容
SHELL
shell *command* --呼叫 shell 執行外部命令。e.g shell ls

程式執行

- run (r) -- 開始執行程式
- continue (c) -- 離開中斷點，繼續執行程式
- next (n) -- 單步執行 (step over)
- step (s) -- 進入函式 (step into)
- until (u) -- 離開 while, for 等迴圈。 執行到程式碼的行數比目前的大，如果目前是迴圈的最後一行，就會離開迴圈
- finish -- 繼續執行程式直到函式返回

- start --在 main 設置暫時中斷點，並開始執行程式
- advance -- 執行程式直到指定的位置
- run *arglist* -- 同 run，並指定程式參數
- start *arglist* -- 同 start，並指定程式參數

- kill (k) --終止程式執行
- quit (q) --離開 GDB

觀看程式碼

- list (l) --列出目前執行程式碼前後各五行的程式碼；或是接著前次 list 的程式碼，列出之後的程式碼
- "list -" -- 上次列出程式碼的前十行，類似向上翻頁
- list *ADDRESS* -- 列出包含指定位址的程式碼，常與 bt 配合使用

```
(gdb) bt
#0  0x08048405 in memory_violation () at test.cpp:9
#1  0x08048427 in main () at test.cpp:15
(gdb) list *0x08048405
```

- directory *DIR* -- 新增一個路徑到程式碼搜尋路徑
- show directories --顯示目前的程式碼搜尋路徑
- disassemble (disas) --反組譯目前執行的程式碼
- disassemble *start_addr end_addr* --反組譯指定範圍的程式碼。e.g. disas 0x32c4 0x32e4

設定中斷點 (breakpoint)

- break 簡寫指令 b
- break (b) *line_number* -- 在指定的行數設定中斷點
- break *function* -- 在指定的 function 設定中斷點。e.g. break main
- break *filename:line_num* -- 在指定檔案的指定行數設定中斷點。e.g. break main.c:10
- break *[LOCATION] [if CONDITION]* -- 條件式中斷點，當 CONDITION 滿足時才中斷。e.g. break main.c:10 if var > 10
- info break -- 列出所有的中斷點及編號
- delete *number* -- 刪除指定編號的中斷點
- disable *number* -- 使指定編號的中斷點失效
- enable *number* -- 取消 disable，使指定編號的中斷點生效

設定 watchpoint

- watch *varname* -- 設定 watch point 監看變數，當變數被寫入時，中斷程式
- rwatch *varname* -- 設定 watch point 監看變數，當變數被讀取時，中斷程式
- awatch *varname* -- 設定 watch point 監看變數，當變數被讀取或寫入時，中斷程式

- watch *(int *)0x12345678 -- 設定 watch point 監看記憶體位址，監看範圍由變數型別決定，當此記憶體位址被寫入時，中斷程式
- info watch -- 列出所有的 watchpoint

觀察變數資料

- print (p) *varname* -- 顯示變數內容
- printf *expression* --使用 C 語言的格式化字串 (printf format string) 功能來顯示。e.g. printf "var=%d\n", var
- display *varname*-- 遇到中斷點時，自動顯示變數的內容
- undisplay *display_num* -- 取消指定編號的自動顯示
- info display -- 列出所有 display 及編號
- whatis *varname* -- 顯示變數型別
- ptype *varname* --顯示 class, struct 的定義內容
- info locals -- 顯示目前的區域變數

觀察記憶體

- x/8xw *ADDRESS* -- 印出 8 個 word (4 bytes) 的記憶體內容
- x/*FMT ADDRESS* -- 詳細的 FMT 說明，請參考 help x
- dump memory *FILE START END* -- 將指定範圍內的記憶體內容 dump 到檔案

顯示函式呼叫堆疊 (Call Stack)

GDB 以函式爲單位分成一個個的 frame，每一個 frame 各代表一個函式。
如 main() 是一個 frame，main() 裡面所呼叫的函式是另外一個 frame。
當呼叫函式時，會把目前的 frame 推到堆疊。這個堆疊就是 frame stack，也就是一般所認知的 call stack。

- backtrace (bt) --顯示目前函式呼叫堆疊 (Call Stack) 內容
- backtrace full -- 一併列出區域變數 (local variable)
- where --顯示目前程式的執行位置，與 backtrace 的作用相同
- frame *frame_num* -- 跳到指定編號的 frame
- up -- 往上一個 frame
- down -- 往下一個 frame
- info args -- 顯示傳給函式的呼叫參數

Multi-thread

- info threads -- 顯示目前所有的 thread
- thread *thread_num* -- 切換 GDB 到指定的 thread_num

改變程式行爲

- return -- 直接從函式目前執行位置返回，放棄未執行的部份
- return *expression* -- 執行 return， 並傳回 expression 的值
- set *varname=xxx* -- 更改變數值

顯示程式相關資訊

- info sharedlibrary -- 顯示被載入的共享函式庫 (shared object library)，及載入的記憶體位置
- info registers -- 顯示基本暫存器的內容
- info all-registers -- 顯示所有暫存器的內容
- print *$eax* -- 顯示 eax 暫存器的內容