系統程式作業 01

鍾天睿

March 06 2021

1 系統環境

發行版	Arch Linux
Linux Kernel	5.10.16-arch1-1
GCC 版本	10.2.0
GDB 版本	10.1

2 GDB 執行內容

2.1 除錯 rdtsc.c

```
檔案(F) 編輯(E) 檢視(V) 書籤(B) 設定(S) 說明(H)
(gdb) b main
Breakpoint 1 at 0x11de: file rdtsc.c, line 28.
(gdb) r
Starting program: /home/ray1422/tmp_1/SP/system-progra
mming/ch02/rdtsc

Breakpoint 1, main (argc=1, argv=0x7fffffffe498)
    at rdtsc.c:28
28      {
(gdb) n
33          printf("這個程式是量測一個指令執行的時間,
但CPU可同時執行數十個指令\n");
(gdb) n
這個程式是量測一個指令執行的時間, 但CPU可同時執行數十
個指令
34          printf("因此這些量測方法比較適合量測大範圍
的程式碼\n\n");
(gdb) n
因此這些量測方法比較適合量測大範圍的程式碼

36          cycles1 = rdtscp();
(gdb) s
rdtscp () at rdtsc.c:19
19          __asm__ __volatile__("rdtscp":"=a"(lo), "=
d"(hi));
(gdb) bt
#0  rdtscp () at rdtsc.c:19
#1  0x000055555555520a in main (argc=1,
```

圖 1: Debugging with GDB (1)

```
檔案(F) 編輯(E) 檢視(V) 書籤(B) 設定(S) 說明(H)
(gdb) bt
#0  rdtscp () at rdtsc.c:19
#1  0x000055555555520a in main (argc=1,
    argv=0x7fffffffe498) at rdtsc.c:36
(gdb) down
Bottom (innermost) frame selected; you cannot go down.
(gdb) up
#1  0x000055555555520a in main (argc=1,
    argv=0x7fffffffe498) at rdtsc.c:36
36          cycles1 = rdtscp();
(gdb) down
#0  rdtscp () at rdtsc.c:19
19          __asm__ __volatile__("rdtscp":"=a"(lo), "=
d"(hi));
(gdb) n
20          return ((uint64_t) lo) | (((uint64_t) hi)
<< 32);
(gdb) n
21      }
(gdb) watch tmp
No symbol "tmp" in current context.
(gdb) up
#1  0x000055555555520a in main (argc=1,
    argv=0x7fffffffe498) at rdtsc.c:36
36          cycles1 = rdtscp();
(gdb) watch tmp
Hardware watchpoint 2: tmp
(gdb) n
```

圖 2: Debugging with GDB (2)

使用 `gdb ./rdtsc` 開始除錯。圖 1 使用 “n” 單步執行程式（不進入函數），使用 “s” 進入函數，使用 “bt” 檢視程式執行堆疊回朔。圖 4 使用 “up” 及 “down” 移動目前聚焦的堆棧，使用 watch 追蹤變數值變更。

2.2 除錯非法記憶體存取

```c
#include <stdio.h>
int main() {
    int *a;
    *a = 10;
    printf("%d\n", *a);
}
```

圖 3: 非法記憶體存取的程式

```
(gdb) b main
Breakpoint 1 at 0x1141: file mem_test.c, line 5.
(gdb) r
Starting program: /home/ray1422/tmp_1/SP/system-progra

Breakpoint 1, main () at mem_test.c:5
5           *a = 10;
(gdb) n

Program received signal SIGSEGV, Segmentation fault.
0x0000555555555145 in main () at mem_test.c:5
5           *a = 10;
```

圖 4: Debugging with GDB (3)

如圖 3 的程式，顯然 `a` 尚未初始化分配記憶體位址，賦值會造成非法記憶體存取。圖 2 中將斷點設在 main 並逐行執行，在賦值時發生錯誤並結束。