作業一：使用 gdb 進行偵錯

題目：

請從 github 下載 rdtsc.c

可以從網站上下載單獨的檔案

https://github.com/shiwulo/system-programming/tree/master/ch02

或者使用 git 將所有的範例都下載

git clone https://github.com/shiwulo/system-programming.git

接下來使用 gdb 對 rdtsc.c 除錯

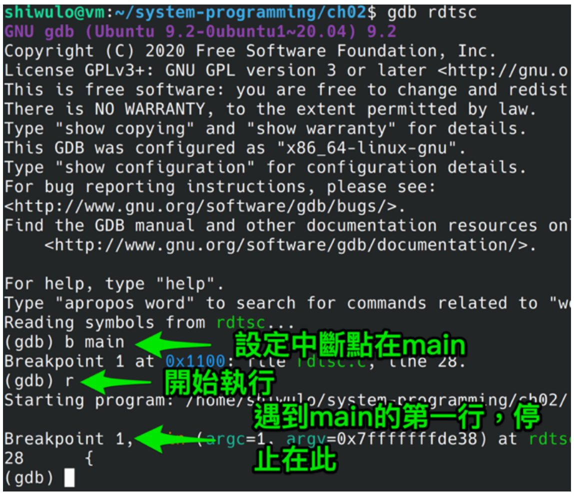
1. 設定中斷點，將中斷點設定在 main 函數
2. 單步執行，遇到函數不會進入
3. 單步執行，遇到函數會進入
4. 列印變數的值
5. 使用 bt、up、down，印出 caller 和 callee 各自的變數
6. 使用 watch 查看變數被修改的情況
7. 修改程式碼，故意存取錯誤的記憶體，看看會發生什麼事

繳交：

1. 繳交報告，報告的格式並須為 pdf。報告前請附上姓名（可隱匿一個字）及學號
2. 繳交到 ecourse2 上
3. 再次提醒，助教會將所有人的作業於 dropbox 上公開
4. 繳交期限：2020/03/16 早上 8:00。不能遲交
5. 如果真的不會寫，記得去請教朋友。在你的報告上寫你請教了誰即可。

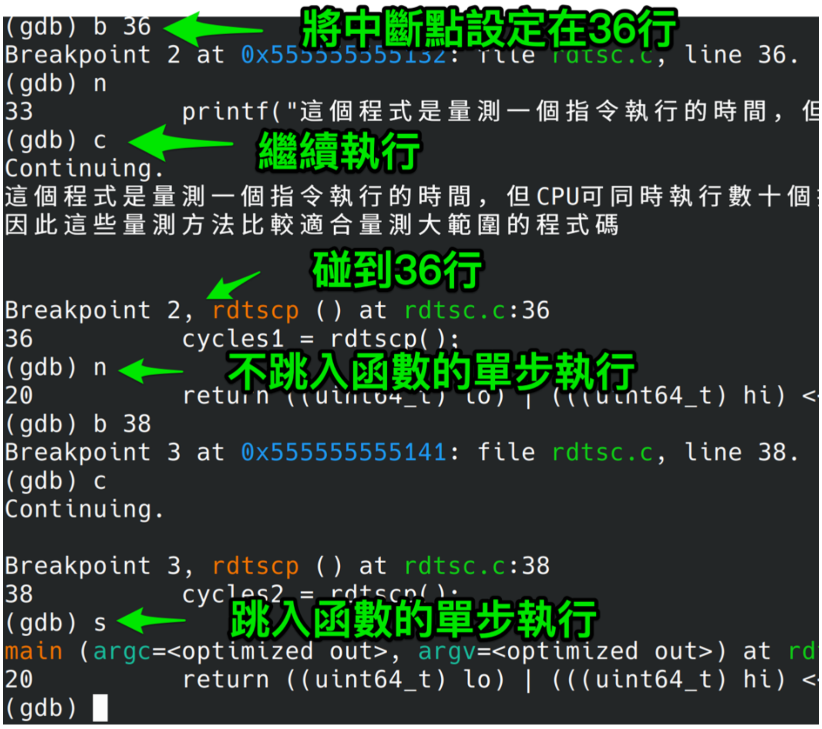
提示

1.



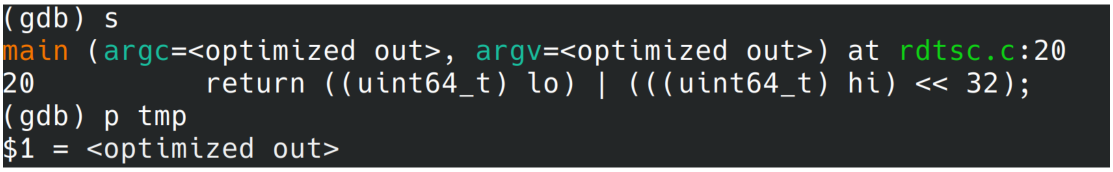
```
shiwulo@vm:~/system-programming/ch02$ gdb rdtsc
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.o
This is free software: you are free to change and redist
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources on
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "w
Reading symbols from rdtsc...
(gdb) b main
Breakpoint 1 at 0x1100: file rdtsc.c, line 28.
(gdb) r
Starting program: /home/shiwulo/system-programming/ch02/

Breakpoint 1, main (argc=1, argv=0x7fffffffde38) at rdts
28      {
(gdb) 
```

2. 3.



```
(gdb) b 36
Breakpoint 2 at 0x555555555132: file rdtsc.c, line 36.
(gdb) n
33          printf("這個程式是量測一個指令執行的時間，但
(gdb) c
Continuing.
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
因此這些量測方法比較適合量測大範圍的程式碼

Breakpoint 2, rdtscp () at rdtsc.c:36
36          cycles1 = rdtscp();
(gdb) n
20          return ((uint64_t) lo) | (((uint64_t) hi) <
(gdb) b 38
Breakpoint 3 at 0x555555555141: file rdtsc.c, line 38.
(gdb) c
Continuing.

Breakpoint 3, rdtscp () at rdtsc.c:38
38          cycles2 = rdtscp();
(gdb) s
main (argc=<optimized out>, argv=<optimized out>) at rd
20          return ((uint64_t) lo) | (((uint64_t) hi) <
(gdb) 
```

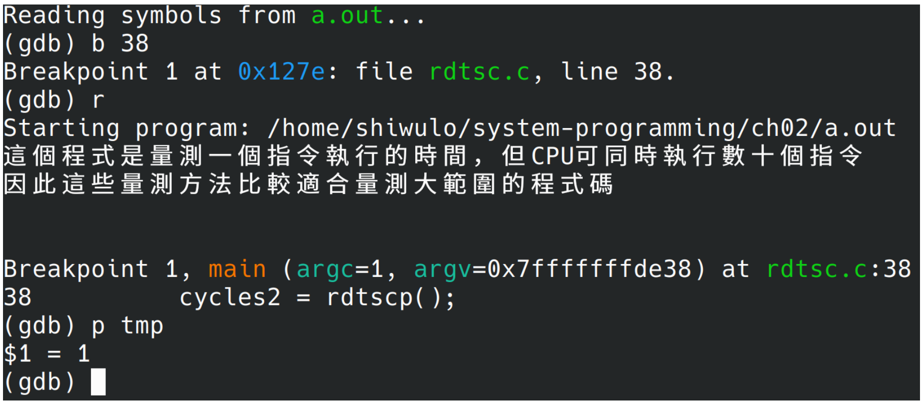
4. 當編譯的指令為 gcc -g -O3，gcc 會做最佳化，tmp 並未配置在記憶體中，因此無法印出，此時顯示 optimized out



```
(gdb) s
main (argc=<optimized out>, argv=<optimized out>) at rdtsc.c:20
20          return ((uint64_t) lo) | (((uint64_t) hi) << 32);
(gdb) p tmp
$1 = <optimized out>
```

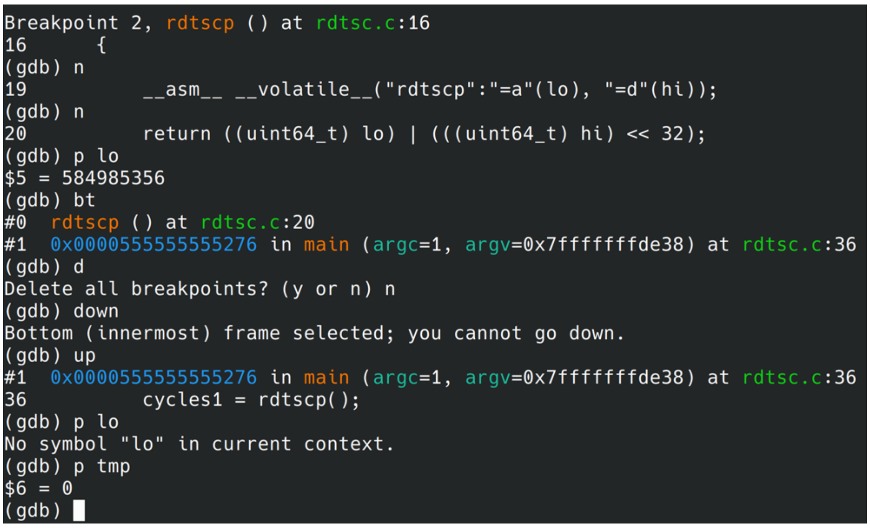
重新編譯一次

gcc -g rdtsc.c



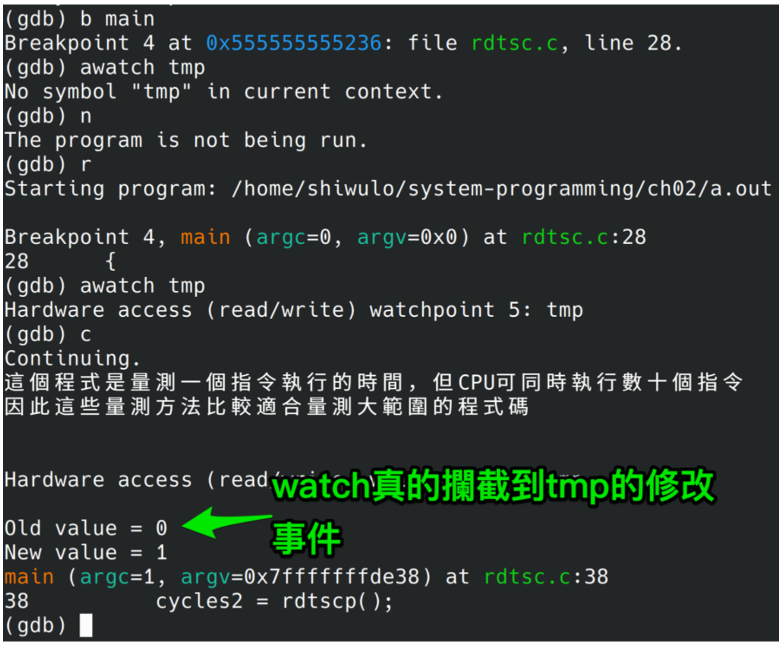
```
Reading symbols from a.out...
(gdb) b 38
Breakpoint 1 at 0x127e: file rdtsc.c, line 38.
(gdb) r
Starting program: /home/shiwulo/system-programming/ch02/a.out
這個程式是量測一個指令執行的時間，但CPU可同時執行數十個指令
因此這些量測方法比較適合量測大範圍的程式碼


Breakpoint 1, main (argc=1, argv=0x7fffffffde38) at rdtsc.c:38
38          cycles2 = rdtscp();
(gdb) p tmp
$1 = 1
(gdb) 
```

5.



```
Breakpoint 2, rdtscp () at rdtsc.c:16
16      {
(gdb) n
19              __asm__ __volatile__("rdtscp":"=a"(lo), "=d"(hi));
(gdb) n
20              return ((uint64_t) lo) | (((uint64_t) hi) << 32);
(gdb) p lo
$5 = 584985356
(gdb) bt
#0  rdtscp () at rdtsc.c:20
#1  0x0000555555555276 in main (argc=1, argv=0x7fffffffde38) at rdtsc.c:36
(gdb) d
Delete all breakpoints? (y or n) n
(gdb) down
Bottom (innermost) frame selected; you cannot go down.
(gdb) up
#1  0x0000555555555276 in main (argc=1, argv=0x7fffffffde38) at rdtsc.c:36
36              cycles1 = rdtscp();
(gdb) p lo
No symbol "lo" in current context.
(gdb) p tmp
$6 = 0
(gdb) 
```

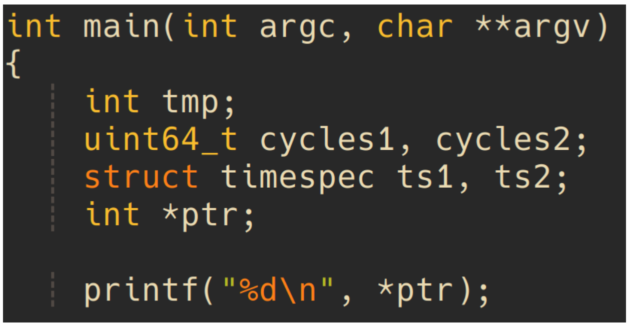
6.



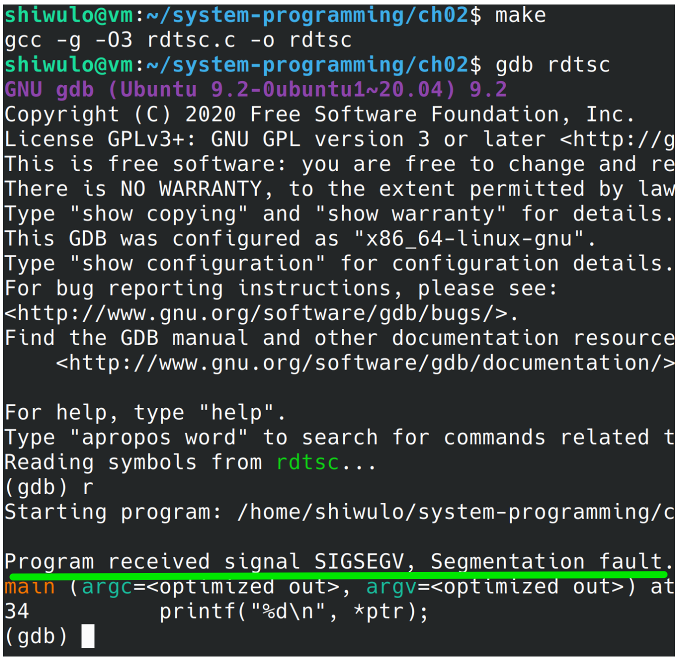
```
(gdb) b main
Breakpoint 4 at 0x555555555236: file rdtsc.c, line 28.
(gdb) awatch tmp
No symbol "tmp" in current context.
(gdb) n
The program is not being run.
(gdb) r
Starting program: /home/shiwulo/system-programming/ch02/a.out

Breakpoint 4, main (argc=0, argv=0x0) at rdtsc.c:28
28      {
(gdb) awatch tmp
Hardware access (read/write) watchpoint 5: tmp
(gdb) c
Continuing.
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
因此這些量測方法比較適合量測大範圍的程式碼


Hardware access (read

Old value = 0
New value = 1
main (argc=1, argv=0x7fffffffde38) at rdtsc.c:38
38              cycles2 = rdtscp();
(gdb) 
```

watch真的攔截到tmp的修改事件

7.



```
int main(int argc, char **argv)
{
    int tmp;
    uint64_t cycles1, cycles2;
    struct timespec ts1, ts2;
    int *ptr;

    printf("%d\n", *ptr);
```

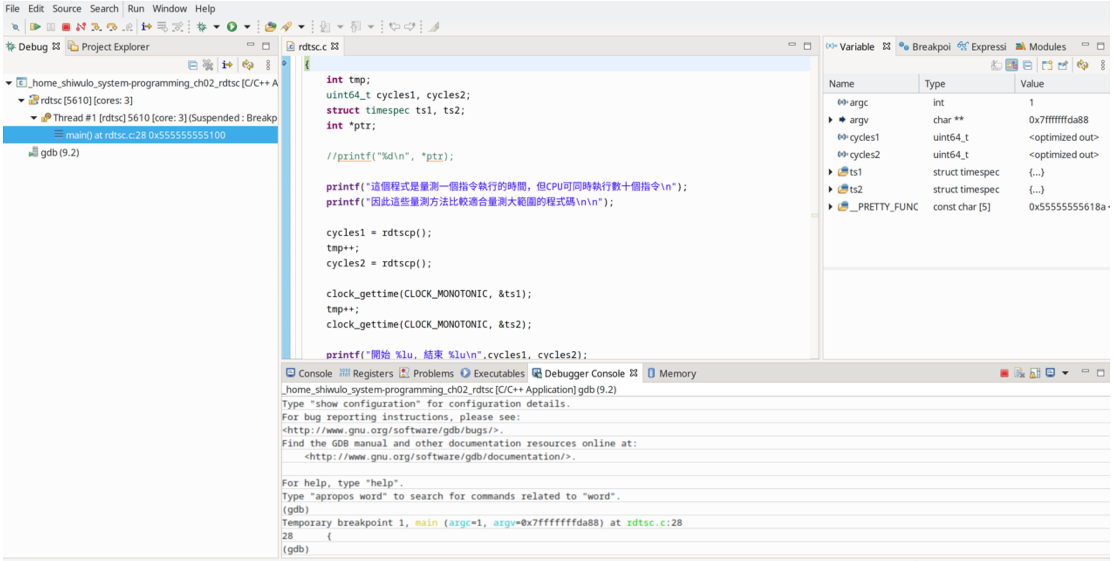


```
shiwulo@vm:~/system-programming/ch02$ make
gcc -g -O3 rdtsc.c -o rdtsc
shiwulo@vm:~/system-programming/ch02$ gdb rdtsc
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://g
This is free software: you are free to change and re
There is NO WARRANTY, to the extent permitted by law
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resource
    <http://www.gnu.org/software/gdb/documentation/>

For help, type "help".
Type "apropos word" to search for commands related t
Reading symbols from rdtsc...
(gdb) r
Starting program: /home/shiwulo/system-programming/c

Program received signal SIGSEGV, Segmentation fault.
main (argc=<optimized out>, argv=<optimized out>) at
34          printf("%d\n", *ptr);
(gdb) 
```

也可以用 cdtdebug 除錯

cdtdebug 背後依然是 gdb，換句話說 cdtdebug 就是 gdb 的圖形化前端

$ cdtdebug ./rdstsc



```
_home_shiwulo_system-programming_ch02_rdtsc [C/C++ A
  rdtsc [5610] [cores: 3]
    Thread #1 [rdtsc] 5610 [core: 3] (Suspended : Breakp
      main() at rdtsc.c:28 0x555555555100
  gdb (9.2)
```

```c
{
    int tmp;
    uint64_t cycles1, cycles2;
    struct timespec ts1, ts2;
    int *ptr;

    //printf("%d\n", *ptr);

    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");
    printf("因此這些量測方法比較適合量測大範圍的程式碼\n\n");

    cycles1 = rdtscp();
    tmp++;
    cycles2 = rdtscp();

    clock_gettime(CLOCK_MONOTONIC, &ts1);
    tmp++;
    clock_gettime(CLOCK_MONOTONIC, &ts2);

    printf("開始 %lu, 結束 %lu\n",cycles1, cycles2);
```

Name	Type	Value
argc	int	1
argv	char **	0x7fffffffda88
cycles1	uint64_t	<optimized out>
cycles2	uint64_t	<optimized out>
ts1	struct timespec	{...}
ts2	struct timespec	{...}
__PRETTY_FUNC	const char [5]	0x55555555618a

```
_home_shiwulo_system-programming_ch02_rdtsc [C/C++ Application] gdb (9.2)
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word".
(gdb)
Temporary breakpoint 1, main (argc=1, argv=0x7fffffffda88) at rdtsc.c:28
28      {
(gdb)
```