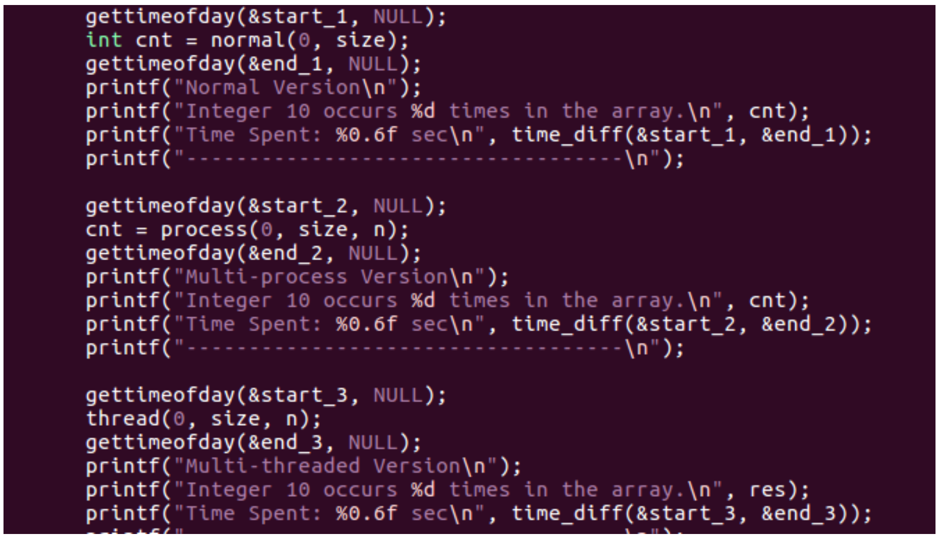
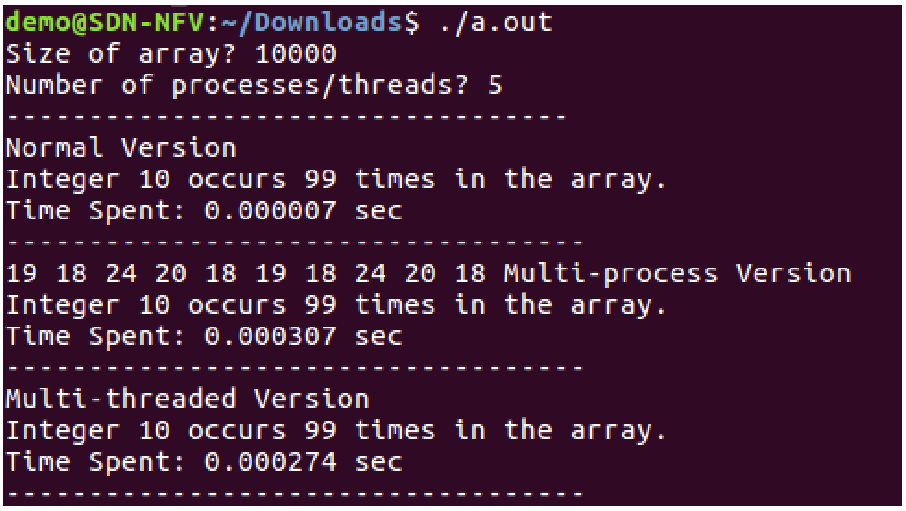
- Platform
    - Virtual machine (VirtualBox)
    - CPU type: AMD Ryzen7 (6 CPU)
    - Kernel version : Linux 4.15.0-142-generic x86_64
    - Memory size : 19.8GB
- Setting
    - Data type: int
    - Target number : 10
    - Data size: 1,000->100,000 (user input)
    - Process/thread: 1->10 (user input)
- Usage
    - gcc normal.c -lpthread
    - ./a.out
    - Input data size
    - Input process/thread
- Measurement result
    - 正確性

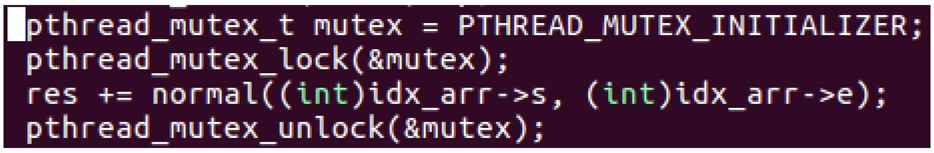


```
gettimeofday(&start_1, NULL);
int cnt = normal(0, size);
gettimeofday(&end_1, NULL);
printf("Normal Version\n");
printf("Integer 10 occurs %d times in the array.\n", cnt);
printf("Time Spent: %0.6f sec\n", time_diff(&start_1, &end_1));
printf("-----------------------------------\n");

gettimeofday(&start_2, NULL);
cnt = process(0, size, n);
gettimeofday(&end_2, NULL);
printf("Multi-process Version\n");
printf("Integer 10 occurs %d times in the array.\n", cnt);
printf("Time Spent: %0.6f sec\n", time_diff(&start_2, &end_2));
printf("-----------------------------------\n");

gettimeofday(&start_3, NULL);
thread(0, size, n);
gettimeofday(&end_3, NULL);
printf("Multi-threaded Version\n");
printf("Integer 10 occurs %d times in the array.\n", res);
printf("Time Spent: %0.6f sec\n", time_diff(&start_3, &end_3));
```

=> 設置三種不同版本的 function，以實作正常、multi-process 以及 multi-threaded，並使用 “gettimeofday” 測量執行時間，執行結果如下圖所示。

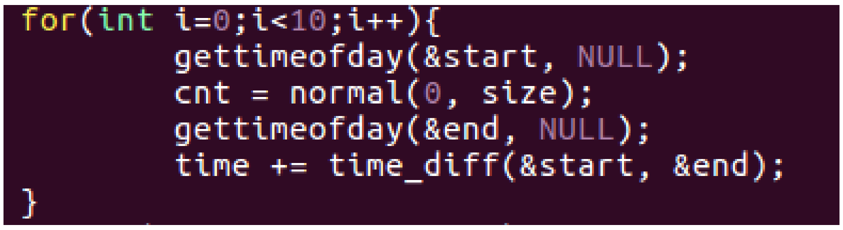


```
demo@SDN-NFV:~/Downloads$ ./a.out
Size of array? 10000
Number of processes/threads? 5
----------------------------------
Normal Version
Integer 10 occurs 99 times in the array.
Time Spent: 0.000007 sec
-----------------------------------
19 18 24 20 18 19 18 24 20 18 Multi-process Version
Integer 10 occurs 99 times in the array.
Time Spent: 0.000307 sec
-----------------------------------
Multi-threaded Version
Integer 10 occurs 99 times in the array.
Time Spent: 0.000274 sec
-----------------------------------
```



```
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_lock(&mutex);
res += normal((int)idx_arr->s, (int)idx_arr->e);
pthread_mutex_unlock(&mutex);
```

⇨ Multi-threaded 的版本使用 mutex 來保證 global 變數 “res” ，不會因為其他 thread 而變動。

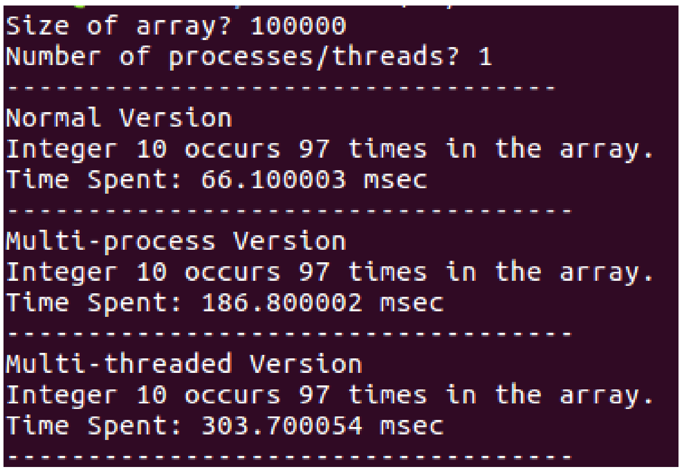
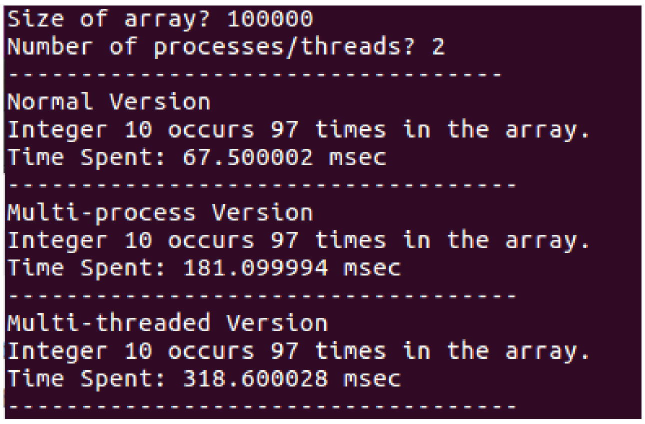
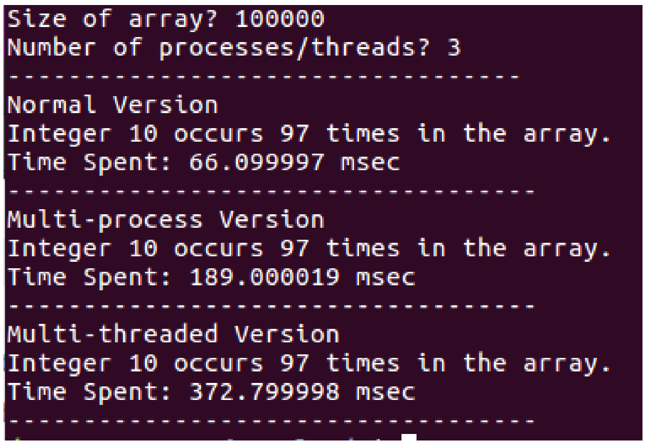
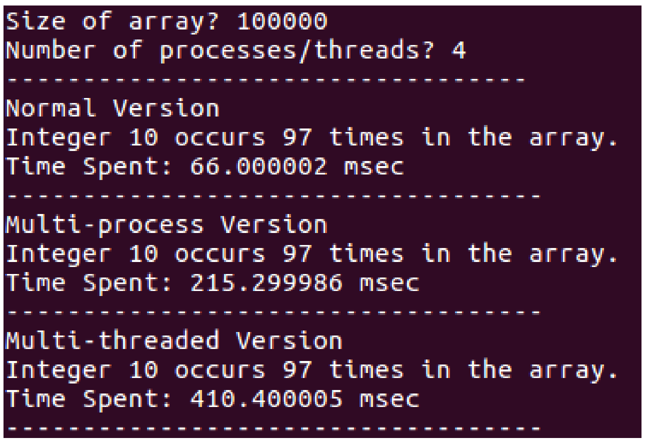


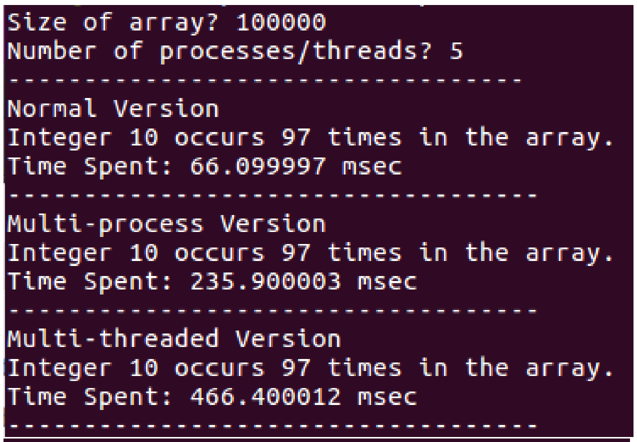
```
for(int i=0;i<10;i++){
        gettimeofday(&start, NULL);
        cnt = normal(0, size);
        gettimeofday(&end, NULL);
        time += time_diff(&start, &end);
}
```

⇨ 為了使用平均的時間，於是新增了一項機制，讓每個 function 各跑 10 次，最後 print 出 “time” /10，也就是全部的平均時間。

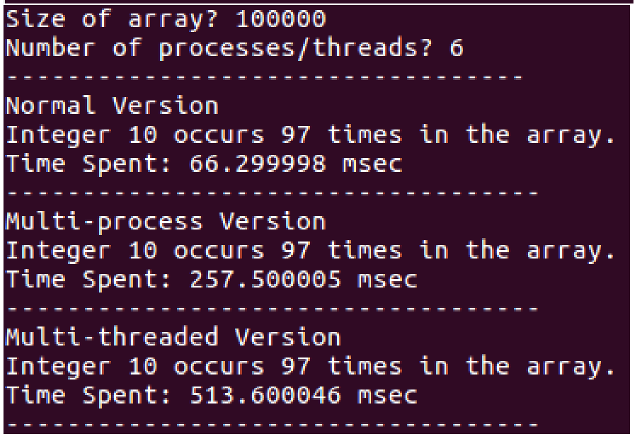
■ 完整性

1. Process / Thread 數目上升，由於 CPU 只有 6 個，因此只測試 1-6 個的結果，分別如下圖所示。


```
Size of array? 100000
Number of processes/threads? 1
-----------------------------------
Normal Version
Integer 10 occurs 97 times in the array.
Time Spent: 66.100003 msec
------------------------------------
Multi-process Version
Integer 10 occurs 97 times in the array.
Time Spent: 186.800002 msec
------------------------------------
Multi-threaded Version
Integer 10 occurs 97 times in the array.
Time Spent: 303.700054 msec
------------------------------------
```

```
Size of array? 100000
Number of processes/threads? 2
-----------------------------------
Normal Version
Integer 10 occurs 97 times in the array.
Time Spent: 67.500002 msec
------------------------------------
Multi-process Version
Integer 10 occurs 97 times in the array.
Time Spent: 181.099994 msec
------------------------------------
Multi-threaded Version
Integer 10 occurs 97 times in the array.
Time Spent: 318.600028 msec
------------------------------------
```

```
Size of array? 100000
Number of processes/threads? 3
-----------------------------------
Normal Version
Integer 10 occurs 97 times in the array.
Time Spent: 66.099997 msec
------------------------------------
Multi-process Version
Integer 10 occurs 97 times in the array.
Time Spent: 189.000019 msec
------------------------------------
Multi-threaded Version
Integer 10 occurs 97 times in the array.
Time Spent: 372.799998 msec
------------------------------------
```

```
Size of array? 100000
Number of processes/threads? 4
-----------------------------------
Normal Version
Integer 10 occurs 97 times in the array.
Time Spent: 66.000002 msec
------------------------------------
Multi-process Version
Integer 10 occurs 97 times in the array.
Time Spent: 215.299986 msec
------------------------------------
Multi-threaded Version
Integer 10 occurs 97 times in the array.
Time Spent: 410.400005 msec
------------------------------------
```



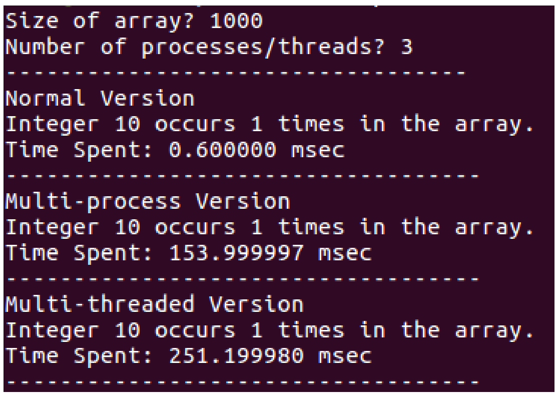
```
Size of array? 100000
Number of processes/threads? 5
----------------------------------
Normal Version
Integer 10 occurs 97 times in the array.
Time Spent: 66.099997 msec
-----------------------------------
Multi-process Version
Integer 10 occurs 97 times in the array.
Time Spent: 235.900003 msec
-----------------------------------
Multi-threaded Version
Integer 10 occurs 97 times in the array.
Time Spent: 466.400012 msec
-----------------------------------
```



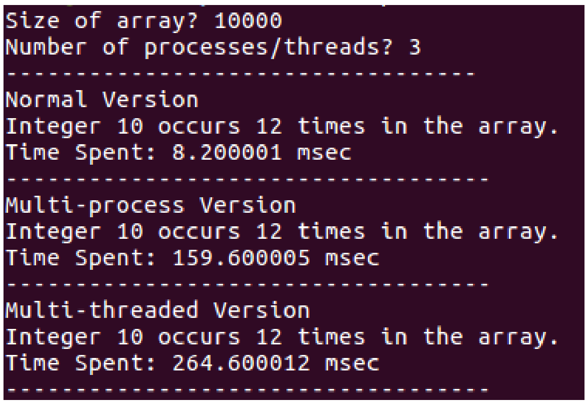
```
Size of array? 100000
Number of processes/threads? 6
----------------------------------
Normal Version
Integer 10 occurs 97 times in the array.
Time Spent: 66.299998 msec
-----------------------------------
Multi-process Version
Integer 10 occurs 97 times in the array.
Time Spent: 257.500005 msec
-----------------------------------
Multi-threaded Version
Integer 10 occurs 97 times in the array.
Time Spent: 513.600046 msec
-----------------------------------
```

⇨ 由這 6 張圖可看出，一般版本下，執行時間幾乎完全一樣，因為沒有多個 processes 或 threads，而當 process/thread 上升後，執行時間都會上升，且不完全呈現線性變化，原因會在後面做解釋。

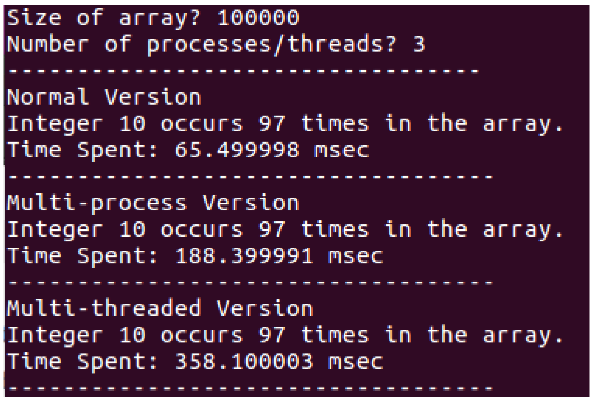
2.Data size 的不同，分別使用 1000、10000 以及 100000 進行測試，process/thread 則固定為 3 個。



```
Size of array? 1000
Number of processes/threads? 3
----------------------------------
Normal Version
Integer 10 occurs 1 times in the array.
Time Spent: 0.600000 msec
-----------------------------------
Multi-process Version
Integer 10 occurs 1 times in the array.
Time Spent: 153.999997 msec
-----------------------------------
Multi-threaded Version
Integer 10 occurs 1 times in the array.
Time Spent: 251.199980 msec
-----------------------------------
```



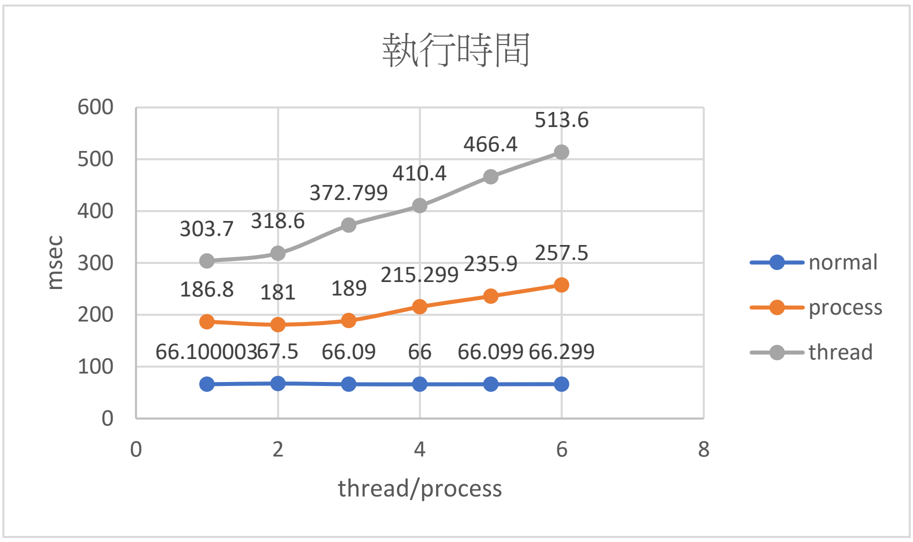
```
Size of array? 10000
Number of processes/threads? 3
----------------------------------
Normal Version
Integer 10 occurs 12 times in the array.
Time Spent: 8.200001 msec
-----------------------------------
Multi-process Version
Integer 10 occurs 12 times in the array.
Time Spent: 159.600005 msec
-----------------------------------
Multi-threaded Version
Integer 10 occurs 12 times in the array.
Time Spent: 264.600012 msec
-----------------------------------
```



```
Size of array? 100000
Number of processes/threads? 3
----------------------------------
Normal Version
Integer 10 occurs 97 times in the array.
Time Spent: 65.499998 msec
-----------------------------------
Multi-process Version
Integer 10 occurs 97 times in the array.
Time Spent: 188.399991 msec
-----------------------------------
Multi-threaded Version
Integer 10 occurs 97 times in the array.
Time Spent: 358.100003 msec
-----------------------------------
```

⇨ 無論是由哪種版本的，都可看出執行時間會隨著 data size 上升而增加。

■ 獨特/創新性

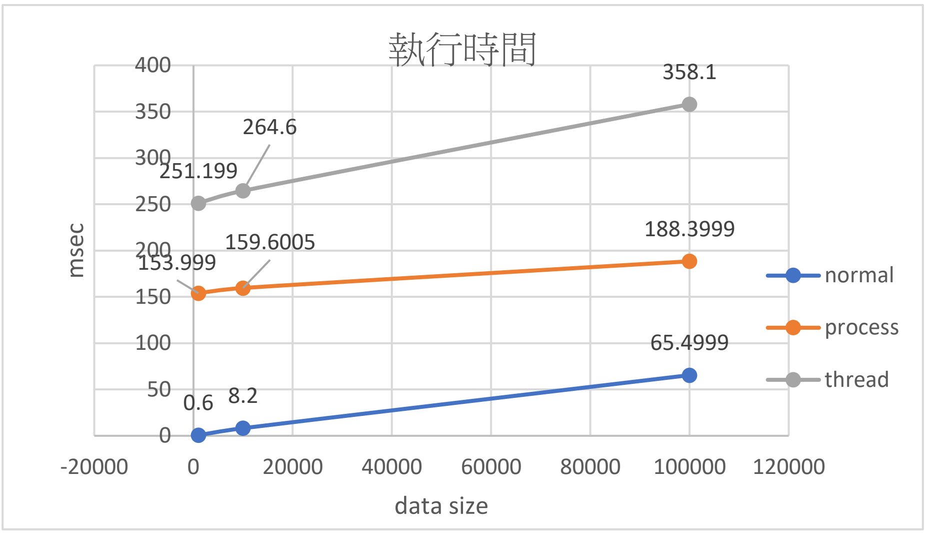


1. 圖表如上圖所示，照理來說執行時間應該要隨著 process/thread 增加而下降，但此程式卻上升，我認為主因可能有兩個:

(1) 程式架構的問題，因為我為了維持每個版本使用相同資料，所以都寫在同個檔案底下，且使用許多函式，所以能在此情況下，過多的 context switch 導致 overhead 過高。

(2) Thread 的部分由於有使用 mutex，因此其他 thread 必須等待，才能做自己的計算。

2. 另一個圖表如下圖所示，基本上無論是哪個版本，執行時間都會隨著 data size 增加而增加，原因前面已經有做說明。



3. Mutex 對效能的影響

(1) 能得到較好的準確度，因為一次只能有一個 thread 接觸 global variable，所以不會受到影響。

(2) 執行時間上，可能會受到影響，因為程式必須平行處理運算該 global 變數，因此多少會減慢速度。

4. 執行環境對效能的影響

(1) 雖然我是在虛擬機上跑，但有配置較大的記憶體空間，以及性能不差的 CPU，因此跑的時間都算蠻快。

(2) 原先以為有多個 CPU，可能再多程緒的部分會有不錯性能，但因為程式架構問題，導致無法展現出來，但還是能推論出有一定的幫助，尤其是 1 和 2 process/thread 時，其實成效不會差太多。