嵌入式系統軟體設計
Embedded System Software Design

PA2

指導教授: 陳雅淑 教授

課程學生: M10907305 陳俊億

- **Part1:**

1. Execution result of using mutex and barrier. 20%

```
chen@chen-VirtualBox:~/桌面/pa2$ ./part1.out

========================System Info==========================
Protect Shared Resource: Mutex
Synchronize: Barrier

===========Start Single Thread Matrix Multiplication=========
Program ID : 0  Thread ID : 0   PID : 6785      Core : 6
Single-thread spend time : 121.299

===========Start Multi-Thread Matrix Multiplication===========
Program ID : 0  Thread ID : 1   PID : 6795      Core : 1
Program ID : 0  Thread ID : 0   PID : 6794      Core : 0
Program ID : 0  Thread ID : 3   PID : 6797      Core : 3
Program ID : 0  Thread ID : 2   PID : 6796      Core : 2
Multi-thread spend time : 210.228

===========================Result===========================
Program-0 obtain correct matrix multiplication result.
```

Fig1. Result of using mutex and barrier.

2. Describe how to synchronize thread. 10%

```
sharedSum = new int [PROGRAM_NUM];
/*~~~~~~~~~Your code(PART1&PART3)~~~~~~~~*/
pthread_barrier_init(&System::barr,NULL,PROGRAM_NUM*THREAD_NUM);
pthread_spin_init(&Thread::lock,pshared);
sem_init(&System::sem,1,PROGRAM_NUM*THREAD_NUM);
sem_init(&System::sem1,1,0);
```

Fig2. Code of synchronize thread. (1)

```
Thread::synchronize ()
{
#if SYNCHRONIZE == BARRIER
    /*~~~~~~~~~~~~Your code(PART1)~~~~~~~~~~~*/
    // std::cout << "1" << std::endl;
    pthread_barrier_wait(barr);
    // std::cout << "2" << std::endl;
    /*~~~~~~~~~~~~~~~~~~END~~~~~~~~~~~~~~~~~~*/
#else
    pthread_mutex_lock (ioMutex);
    std::cout << "Synchronize method not supported." << std::endl;
    pthread_mutex_unlock (ioMutex);
#endif
}
```

Fig3. Code of synchronize thread. (2)

```
    obj->synchronize();
    for (int i = obj->startCalculatePoint; i < obj->endCalculatePoint; i++){
        memcpy (obj->matrix [i], obj->multiResult [i], obj->matrixSize * sizeof (int));
    }
    obj->synchronize();
```

Fig4. Code of synchronize thread. (3)

首先先建立 barrier 的初始設定，如圖 2 所示，透過測試需要在 memcpy 上下要進行同步，否則數值會有錯誤，可透過圖 3 的功能來進行同步。

3. Describe how to protect a shared resource. 5%

```
public:
  pthread_t pthreadThread;
  static pthread_spinlock_t lock;
  static pthread_mutex_t count_mutex;
```

Fig5. Code of protect a shared resource. (1)

```
#if (PART != 2)
            pthread_mutex_lock(&count_mutex);
            *obj->sharedSum = 0;
            for (int k = 0 ; k < obj->matrixSize; k++)
                *obj->sharedSum += obj->matrix [i][k] * obj->matrix [k][j];

            obj->multiResult [i][j] = *obj->sharedSum;
            pthread_mutex_unlock(&count_mutex);
#else
```

Fig6. Code of protect a shared resource. (2)

先在 thread.h 宣告 mutex 變數，如圖 6 所示，而主要保護的共用資源為 shareSum，因此它的上下進行 mutex 的共用資源保護。

- **Part2:**

1. Execution result of using reentrant function. 15%

```
chen@chen-VirtualBox:~/桌面/pa2$ ./part2.out

=======================System Info=========================
Protect Shared Resource: Mutex
Synchronize: Barrier

===========Start Single Thread Matrix Multiplication==========
Program ID : 0  Thread ID : 0   PID : 7141      Core : 3
Single-thread spend time : 123.251

===========Start Multi-Thread Matrix Multiplication===========
Program ID : 0  Thread ID : 2   PID : 7156      Core : 2
Program ID : 0  Thread ID : 1   PID : 7155      Core : 1
Program ID : 0  Thread ID : 0   PID : 7154      Core : 0
Program ID : 0  Thread ID : 3   PID : 7157      Core : 3
Multi-thread spend time : 28.2385

============================Result============================
Program-0 obtain correct matrix multiplication result.
```

Fig7. Result of using reentrant function.

2. Describe how to modify non-reentrant function into reentrant function. 10%

```
        /*~~~~~~~~~~~~Your code(PART2)~~~~~~~~~~~*/
            int local_parameter = 0;
            for (int k = 0 ; k < obj->matrixSize; k++)
                // *obj->sharedSum += obj->matrix [i][k] * obj->matrix [k][j];
                local_parameter += obj->matrix [i][k] * obj->matrix [k][j];
            obj->multiResult [i][j] = local_parameter;
        /*~~~~~~~~~~~~~~~~~END~~~~~~~~~~~~~~~~~*/
```

Fig8. reentrant function.

要實現 reentrant function，捨棄了共用資源(全局參數)，改用一般參數(局部參數)，如圖 8 所示。

3. Describe the reason why using a non-reentrant function or a reentrant function could obtain better performance. 5%

從圖 1 與圖 7 的結果上看，使用 reentrant function 效果較 non-reentrant function 好，首先 non-reentrant 會使用到 global variables or static variables 因此在不同的 thread 中需要使用到類似於 mutex 去保護共用資源，以免被干擾，但是 mutex 會鎖死當下的 thread 等到做完之後才會讓其他 thread 動作，導致時間會比 reentrant 慢。

- **Part3:**

1. Execution result of using spinlock. 10%

```
chen@chen-VirtualBox:~/桌面/pa2$ ./part3.out

=======================System Info==========================
Protect Shared Resource: Spinlock
Synchronize: Barrier

===========Start Single Thread Matrix Multiplication==========
Program ID : 0  Thread ID : 0   PID : 11643     Core : 5
Single-thread spend time : 125.965

===========Start Multi-Thread Matrix Multiplication===========
Program ID : 0  Thread ID : 0   PID : 11665     Core : 0
Program ID : 0  Thread ID : 1   PID : 11666     Core : 1
Program ID : 0  Thread ID : 2   PID : 11667     Core : 2
Program ID : 0  Thread ID : 3   PID : 11668     Core : 3
Multi-thread spend time : 111.77

===========================Result===========================
Program-0 obtain correct matrix multiplication result.
```

Fig9. Result of using spinlock.

2. Describe which method (mutex and spinlock) could obtain better performance under the benchmark we provided (5%) and why (5%).

```
#if (PART != 2)
                if(PART == 1){
                    pthread_mutex_lock(&count_mutex);
                    *obj->sharedSum = 0;
                    for (int k = 0 ; k < obj->matrixSize; k++)
                        *obj->sharedSum += obj->matrix [i][k] * obj->matrix [k][j];

                    obj->multiResult [i][j] = *obj->sharedSum;
                    pthread_mutex_unlock(&count_mutex);
                }
                else{
                    pthread_spin_lock(&lock);
                    *obj->sharedSum = 0;
                    for (int k = 0 ; k < obj->matrixSize; k++)
                        *obj->sharedSum += obj->matrix [i][k] * obj->matrix [k][j];

                    obj->multiResult [i][j] = *obj->sharedSum;
                    pthread_spin_unlock(&lock);
                }

#else
```

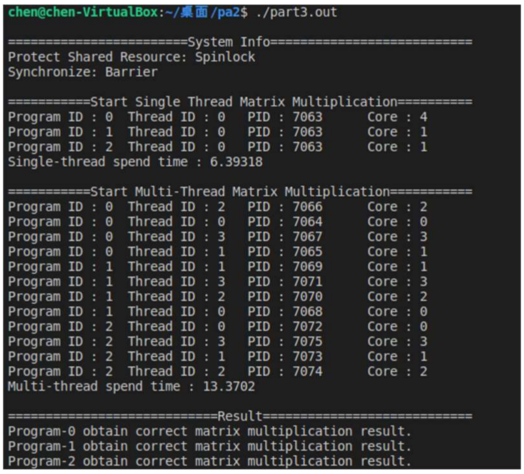
Fig10. Code of using spinlock.

以圖 1 以圖 9 的結果來比較，使用 spinlock 較優於 mutex，其原因 spinlock 會一直 busy waiting 等待解鎖，而 mutex 只限定於當下的 thread 可以動作，而從特性中得知，spinlock 有利於保護單一變數(global variables)，而 mutex 比較適合保護一段程式，在圖 10 中，基本上都算是保護單一參數，因此 spinlock 較 mutex 佳。

3. Show the benchmark your used (5%), explain the properties of such benchmark (5%) and the execution results (5%).

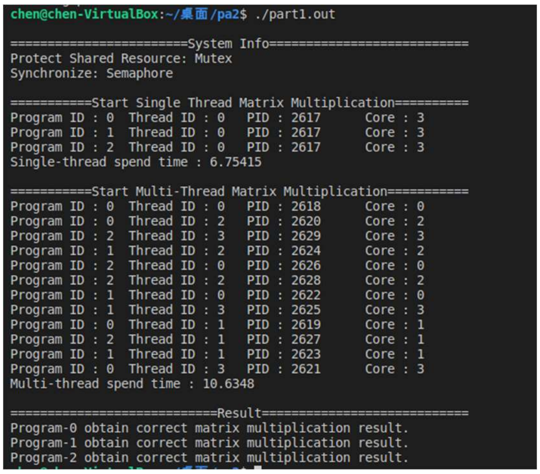
表 1. 參數測試表:

測試組/參數名稱	PROGRAM_NUM	MATRIX_SIZE	MULTI_TIME
1	3	500	2
2	2	500	2



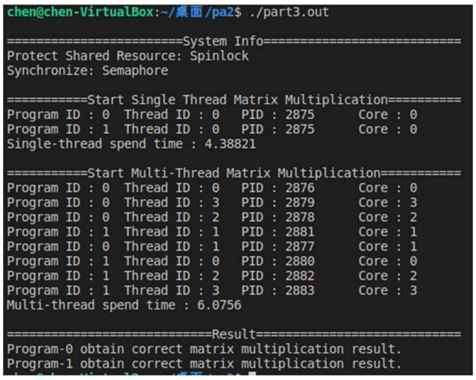
```
chen@chen-VirtualBox:~/桌面/pa2$ ./part3.out

======================System Info=========================
Protect Shared Resource: Spinlock
Synchronize: Barrier

==========Start Single Thread Matrix Multiplication=========
Program ID : 0  Thread ID : 0   PID : 7063      Core : 4
Program ID : 1  Thread ID : 0   PID : 7063      Core : 1
Program ID : 2  Thread ID : 0   PID : 7063      Core : 1
Single-thread spend time : 6.39318

==========Start Multi-Thread Matrix Multiplication==========
Program ID : 0  Thread ID : 2   PID : 7066      Core : 2
Program ID : 0  Thread ID : 0   PID : 7064      Core : 0
Program ID : 0  Thread ID : 3   PID : 7067      Core : 3
Program ID : 0  Thread ID : 1   PID : 7065      Core : 1
Program ID : 1  Thread ID : 1   PID : 7069      Core : 1
Program ID : 1  Thread ID : 3   PID : 7071      Core : 3
Program ID : 1  Thread ID : 2   PID : 7070      Core : 2
Program ID : 1  Thread ID : 0   PID : 7068      Core : 0
Program ID : 2  Thread ID : 0   PID : 7072      Core : 0
Program ID : 2  Thread ID : 3   PID : 7075      Core : 3
Program ID : 2  Thread ID : 1   PID : 7073      Core : 1
Program ID : 2  Thread ID : 2   PID : 7074      Core : 2
Multi-thread spend time : 13.3702

==========================Result==========================
Program-0 obtain correct matrix multiplication result.
Program-1 obtain correct matrix multiplication result.
Program-2 obtain correct matrix multiplication result.
```

Fig12. execution results of part3.out.(測試組 1)



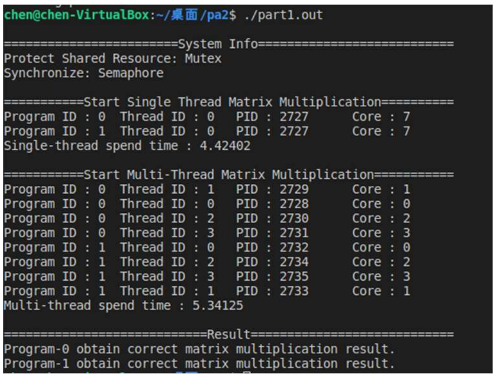
```
chen@chen-VirtualBox:~/桌面/pa2$ ./part1.out

======================System Info=========================
Protect Shared Resource: Mutex
Synchronize: Semaphore

==========Start Single Thread Matrix Multiplication=========
Program ID : 0  Thread ID : 0   PID : 2617      Core : 3
Program ID : 1  Thread ID : 0   PID : 2617      Core : 3
Program ID : 2  Thread ID : 0   PID : 2617      Core : 3
Single-thread spend time : 6.75415

==========Start Multi-Thread Matrix Multiplication==========
Program ID : 0  Thread ID : 0   PID : 2618      Core : 0
Program ID : 0  Thread ID : 2   PID : 2620      Core : 2
Program ID : 2  Thread ID : 3   PID : 2629      Core : 3
Program ID : 1  Thread ID : 2   PID : 2624      Core : 2
Program ID : 2  Thread ID : 0   PID : 2626      Core : 0
Program ID : 2  Thread ID : 2   PID : 2628      Core : 2
Program ID : 1  Thread ID : 0   PID : 2622      Core : 0
Program ID : 1  Thread ID : 3   PID : 2625      Core : 3
Program ID : 0  Thread ID : 1   PID : 2619      Core : 1
Program ID : 2  Thread ID : 1   PID : 2627      Core : 1
Program ID : 1  Thread ID : 1   PID : 2623      Core : 1
Program ID : 0  Thread ID : 3   PID : 2621      Core : 3
Multi-thread spend time : 10.6348

==========================Result==========================
Program-0 obtain correct matrix multiplication result.
Program-1 obtain correct matrix multiplication result.
Program-2 obtain correct matrix multiplication result.
```

Fig13. execution results of part1.out. (測試組 1)



```
chen@chen-VirtualBox:~/桌面/pa2$ ./part3.out

======================System Info=========================
Protect Shared Resource: Spinlock
Synchronize: Semaphore

==========Start Single Thread Matrix Multiplication=========
Program ID : 0  Thread ID : 0   PID : 2875      Core : 0
Program ID : 1  Thread ID : 0   PID : 2875      Core : 0
Single-thread spend time : 4.38821

==========Start Multi-Thread Matrix Multiplication==========
Program ID : 0  Thread ID : 0   PID : 2876      Core : 0
Program ID : 0  Thread ID : 3   PID : 2879      Core : 3
Program ID : 0  Thread ID : 2   PID : 2878      Core : 2
Program ID : 1  Thread ID : 1   PID : 2881      Core : 1
Program ID : 0  Thread ID : 1   PID : 2877      Core : 1
Program ID : 1  Thread ID : 0   PID : 2880      Core : 0
Program ID : 1  Thread ID : 2   PID : 2882      Core : 2
Program ID : 1  Thread ID : 3   PID : 2883      Core : 3
Multi-thread spend time : 6.0756

==========================Result==========================
Program-0 obtain correct matrix multiplication result.
Program-1 obtain correct matrix multiplication result.
```

Fig14. execution results of part3.out.(測試組 2)



```
chen@chen-VirtualBox:~/桌面/pa2$ ./part1.out

======================System Info=========================
Protect Shared Resource: Mutex
Synchronize: Semaphore

==========Start Single Thread Matrix Multiplication=========
Program ID : 0  Thread ID : 0   PID : 2727      Core : 7
Program ID : 1  Thread ID : 0   PID : 2727      Core : 7
Single-thread spend time : 4.42402

==========Start Multi-Thread Matrix Multiplication==========
Program ID : 0  Thread ID : 1   PID : 2729      Core : 1
Program ID : 0  Thread ID : 0   PID : 2728      Core : 0
Program ID : 0  Thread ID : 2   PID : 2730      Core : 2
Program ID : 0  Thread ID : 3   PID : 2731      Core : 3
Program ID : 1  Thread ID : 0   PID : 2732      Core : 0
Program ID : 1  Thread ID : 2   PID : 2734      Core : 2
Program ID : 1  Thread ID : 3   PID : 2735      Core : 3
Program ID : 1  Thread ID : 1   PID : 2733      Core : 1
Multi-thread spend time : 5.34125

==========================Result==========================
Program-0 obtain correct matrix multiplication result.
Program-1 obtain correct matrix multiplication result.
```

Fig15. execution results of part1.out. (測試組 2)

PROGRAM_NUM 在我理解為開設每顆 core 各開幾個 thread，假設 program_num 等於 2 就相當於在每個 core 上開兩個 thread，如圖 12 與圖 13 所示，program_num=3 時每顆 core 各開 3 個 thread，而 Matrix_size 為運算的矩陣大小，最後一個 Multi_time 重複計算次數，但因為每次一個迴圈算完之後會更新 matrix 的數值，所以每次迴圈完的結果會不一樣。從上述的圖 12 與圖 13 得知 mutex 反而比 spinlock 效果好，主要原因是 mutex 多了 2 個 thread 如果有一個被擋住還可以切到其他同一顆 core 的 thread 繼續做，但是 spinlock 還是一直在 busy waiting，而圖 14 與圖 15 也是相同例子只是修改 PROGRAM_NUM，結果與剛剛一樣。

- Bonus: using semaphore.

1. Describe how to use semaphore.

```
    sem_init(&System::sem,1,PROGRAM_NUM*THREAD_NUM);
    sem_init(&System::sem1,1,0);
```

Fig16. Define the Sme_init.

```
    then \
        sed -i "/#define PROTECT_SHARED_RESOURCE/c\#define PROTECT_SHARED_RESOURCE SPINLOCK" ./src/config.h; \
        sed -i "/#define SYNCHRONIZE/c\#define SYNCHRONIZE SEMAPHORE" ./src/config.h; \
    else \
        sed -i "/#define PROTECT_SHARED_RESOURCE/c\#define PROTECT_SHARED_RESOURCE MUTEX" ./src/config.h; \
        sed -i "/#define SYNCHRONIZE/c\#define SYNCHRONIZE SEMAPHORE" ./src/config.h; \
    fi
```

Fig17. Modify the makefile.

```
#elif SYNCHRONIZE == SEMAPHORE

    sem_wait(sem);
    pthread_mutex_lock (ioMutex);
    ++sem1_key ;
    pthread_mutex_unlock (ioMutex);
    if( sem1_key == THREAD_NUM*PROGRAM_NUM){
        pthread_mutex_lock (ioMutex);
        for(int i = 0 ; i<THREAD_NUM*PROGRAM_NUM ; ++i){
            sem_post(sem1);
        }
        pthread_mutex_unlock (ioMutex);
    }
    sem_post(sem);
    sem_wait(sem1);
    sem2_key = 0;
    -- sem1_key;
    while(1){
        if(sem1_key==0 or sem2_key ==1){
            break;
        }
    }
    sem2_key = 1 ;
```

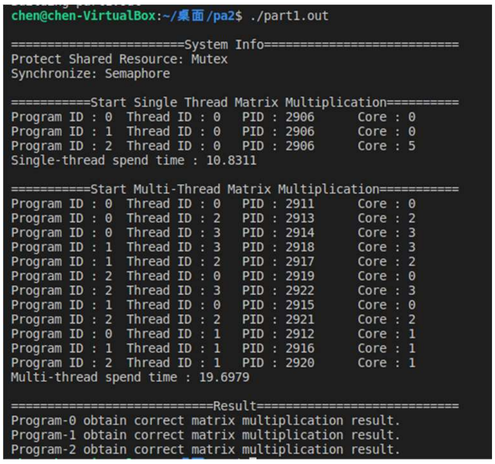
Fig18. Semaphore function

透過 Semaphore 實現同步功能，首先跟 barrier 一樣先做初始化設定，為了實現同步功能，我使用到兩個 semaphore 去實現，如圖 14 所示，第一個 semaphore 也是要決定當下有幾個人可以拿 key，在此我設定為 program_num*thread_num 的數量，也是同時我有幾個 thread 的可以執行，但是每個 thread 的執行速度不一，不能確保說他們速度都一樣，有時候像是 thread 1 跑很快，有機會 thread 8 還沒拿到 key，thread1 就搶先拿到 key 執行第二次計算，會導致在 memcpy 那裡出現錯誤，所以透過另一個 smeaphore 去擋住其他已做好的 thread，現在期可做到同步功能，且在 makefile 需要設定為 semaphore，如圖 15 所示。

2. Modify benchmark to show execution result.

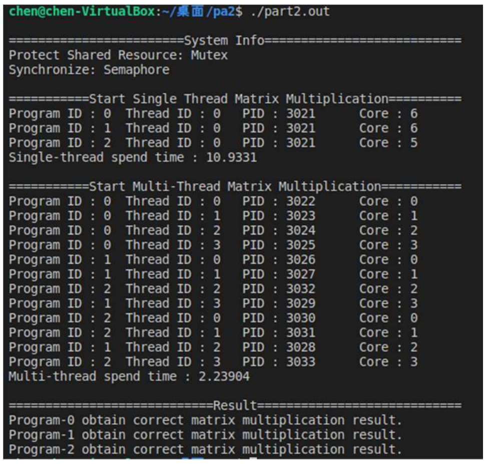
表 2. 參數測試表:

測試組/參數名稱	PROGRAM_NUM	MATRIX_SIZE	MULTI_TIME
1	3	500	2



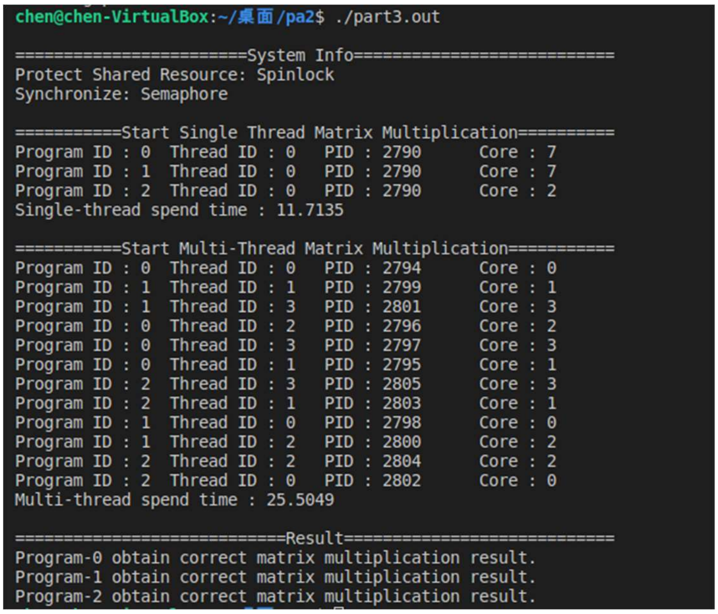
```
chen@chen-VirtualBox:~/桌面/pa2$ ./part1.out

========================System Info===========================
Protect Shared Resource: Mutex
Synchronize: Semaphore

==========Start Single Thread Matrix Multiplication==========
Program ID : 0  Thread ID : 0   PID : 2906      Core : 0
Program ID : 1  Thread ID : 0   PID : 2906      Core : 0
Program ID : 2  Thread ID : 0   PID : 2906      Core : 5
Single-thread spend time : 10.8311

==========Start Multi-Thread Matrix Multiplication===========
Program ID : 0  Thread ID : 0   PID : 2911      Core : 0
Program ID : 0  Thread ID : 2   PID : 2913      Core : 2
Program ID : 0  Thread ID : 3   PID : 2914      Core : 3
Program ID : 1  Thread ID : 3   PID : 2918      Core : 3
Program ID : 1  Thread ID : 2   PID : 2917      Core : 2
Program ID : 2  Thread ID : 0   PID : 2919      Core : 0
Program ID : 2  Thread ID : 3   PID : 2922      Core : 3
Program ID : 1  Thread ID : 0   PID : 2915      Core : 0
Program ID : 2  Thread ID : 2   PID : 2921      Core : 2
Program ID : 0  Thread ID : 1   PID : 2912      Core : 1
Program ID : 1  Thread ID : 1   PID : 2916      Core : 1
Program ID : 2  Thread ID : 1   PID : 2920      Core : 1
Multi-thread spend time : 19.6979

===========================Result============================
Program-0 obtain correct matrix multiplication result.
Program-1 obtain correct matrix multiplication result.
Program-2 obtain correct matrix multiplication result.
```

Fig19. Result of part1.out.



```
chen@chen-VirtualBox:~/桌面/pa2$ ./part2.out

========================System Info===========================
Protect Shared Resource: Mutex
Synchronize: Semaphore

==========Start Single Thread Matrix Multiplication==========
Program ID : 0  Thread ID : 0   PID : 3021      Core : 6
Program ID : 1  Thread ID : 0   PID : 3021      Core : 6
Program ID : 2  Thread ID : 0   PID : 3021      Core : 5
Single-thread spend time : 10.9331

==========Start Multi-Thread Matrix Multiplication===========
Program ID : 0  Thread ID : 0   PID : 3022      Core : 0
Program ID : 0  Thread ID : 1   PID : 3023      Core : 1
Program ID : 0  Thread ID : 2   PID : 3024      Core : 2
Program ID : 0  Thread ID : 3   PID : 3025      Core : 3
Program ID : 1  Thread ID : 0   PID : 3026      Core : 0
Program ID : 1  Thread ID : 1   PID : 3027      Core : 1
Program ID : 2  Thread ID : 2   PID : 3032      Core : 2
Program ID : 1  Thread ID : 3   PID : 3029      Core : 3
Program ID : 2  Thread ID : 0   PID : 3030      Core : 0
Program ID : 2  Thread ID : 1   PID : 3031      Core : 1
Program ID : 1  Thread ID : 2   PID : 3028      Core : 2
Program ID : 2  Thread ID : 3   PID : 3033      Core : 3
Multi-thread spend time : 2.23904

===========================Result============================
Program-0 obtain correct matrix multiplication result.
Program-1 obtain correct matrix multiplication result.
Program-2 obtain correct matrix multiplication result.
```

Fig20. Result of part2.out.



```
chen@chen-VirtualBox:~/桌面/pa2$ ./part3.out

========================System Info===========================
Protect Shared Resource: Spinlock
Synchronize: Semaphore

==========Start Single Thread Matrix Multiplication==========
Program ID : 0  Thread ID : 0   PID : 2790      Core : 7
Program ID : 1  Thread ID : 0   PID : 2790      Core : 7
Program ID : 2  Thread ID : 0   PID : 2790      Core : 2
Single-thread spend time : 11.7135

==========Start Multi-Thread Matrix Multiplication===========
Program ID : 0  Thread ID : 0   PID : 2794      Core : 0
Program ID : 1  Thread ID : 1   PID : 2799      Core : 1
Program ID : 1  Thread ID : 3   PID : 2801      Core : 3
Program ID : 0  Thread ID : 2   PID : 2796      Core : 2
Program ID : 0  Thread ID : 3   PID : 2797      Core : 3
Program ID : 0  Thread ID : 1   PID : 2795      Core : 1
Program ID : 2  Thread ID : 3   PID : 2805      Core : 3
Program ID : 2  Thread ID : 1   PID : 2803      Core : 1
Program ID : 1  Thread ID : 0   PID : 2798      Core : 0
Program ID : 1  Thread ID : 2   PID : 2800      Core : 2
Program ID : 2  Thread ID : 2   PID : 2804      Core : 2
Program ID : 2  Thread ID : 0   PID : 2802      Core : 0
Multi-thread spend time : 25.5049

===========================Result============================
Program-0 obtain correct matrix multiplication result.
Program-1 obtain correct matrix multiplication result.
Program-2 obtain correct matrix multiplication result.
```

Fig21. Result of part3.out