4107056019 廖柏丞

前半部分主要是處理值的輸入，在這題中，我是採用一維陣列來處理二維矩陣 matrix1 代表 a matrix2 代表 b matrix3 則代表 c

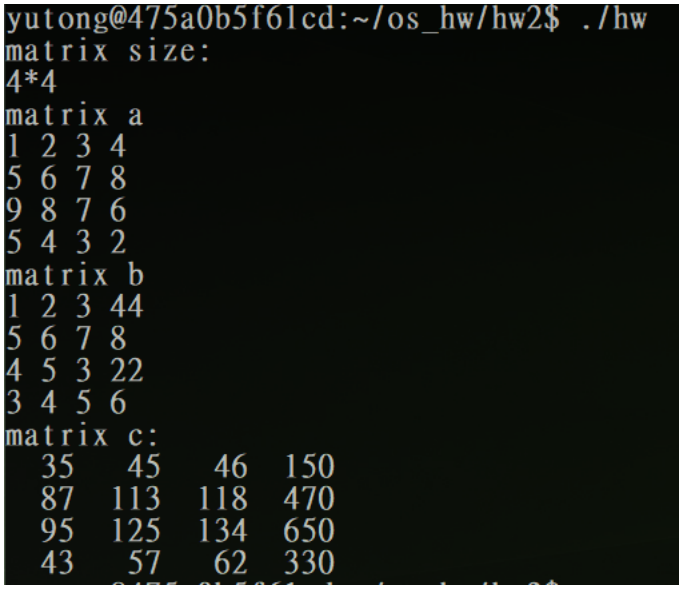
取得資料後，我會開設 n 條執行緒做處理，每一條執行緒會帶入一個 args 參數，來指定 mulArr 要運算第幾個 row 第幾個 col

當運算完畢後，主執行緒會 join n 次，以確保每一條執行緒都已經答案算出

```c
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
int *matrix1, *matrix2, *matrix3;
int n;
void* mulArr(void* r){
        int i, j;
        int row = *((int*)r);
        for(i=0; i<n; i++){
                // row = *(int*)r , col = i
                for(j=0; j<n; j++){
                        matrix3[row * n + i] += matrix1[row * n + j] * matrix2[j*n + i];
                }
        }
}
```

```c
int main(){
        int i;
        printf("matrix size:\n");
        scanf("%d*%d", &n, &n);

        matrix1 = malloc(sizeof(int) * n * n);
        matrix2 = malloc(sizeof(int) * n * n);
        matrix3 = calloc(n*n, sizeof(int));

        printf("matrix a\n");
        for(i=0; i<n*n; i++) scanf("%d", &matrix1[i]);
        printf("matrix b\n");
        for(i=0; i<n*n; i++) scanf("%d", &matrix2[i]);
        printf("matrix c:\n");

        pthread_t* thread_arr = malloc(sizeof(pthread_t) * n);
        int* args = malloc(sizeof(int)*n);
        for(i=0; i<n; i++){
                args[i] = i;
                pthread_create(thread_arr+i, NULL, mulArr, args+i);
        }

        for(i=0; i<n; i++){
                pthread_join(thread_arr[i], NULL);
        }

        for(i=0; i<n*n; ){
                printf("%4d ", matrix3[i]);
                if(!(++i % n)){
                        printf("\n");
                }
        }
}
```

執行結果：


yutong@475a0b5f61cd:~/os_hw/hw2$ ./hw
matrix size:
4*4
matrix a
1 2 3 4
5 6 7 8
9 8 7 6
5 4 3 2
matrix b
1 2 3 44
5 6 7 8
4 5 3 22
3 4 5 6
matrix c:
35 45 46 150
87 113 118 470
95 125 134 650
43 57 62 330