微算機系統實習

LAB 07

組別：19

109590014　　沈煒翔

109590015　　楊挺煜

109590023　　廖堃霖

日期：111/06/10

2. 實驗

(1)lab7_semaphore.cpp

```cpp
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
#include <semaphore.h>
#include "gpio.h"

#define LED1 396
#define LED2 397
#define LED3 429
#define LED4 393

sem_t sem;
int gpioPin[4] = {LED1, LED2, LED3, LED4};

//子執行緒
void* child(void* data) {
    sem_wait(&sem);
    int* input = (int*) data;
    GPIO::Set_dir(input[0], "out");
    GPIO::Set_val(input[0], input[1]);
    printf("GPIO: %d status: %d\n", input[0], input[1]);
    sleep(1);
    GPIO::Set_val(input[0], 0);
    pthread_exit(NULL);
}
```

```
int main(int argc, char** argv) {
    sem_init(&sem, 0, 0);
    pthread_t t1, t2, t3, t4;
    for(int i = 0; i < 4; i++) {
        GPIO::Export(gpioPin[i]);
    }


    int s0[2] = {LED1, argv[1][0] - '0'};
    int s1[2] = {LED2, argv[1][1] - '0'};
    int s2[2] = {LED3, argv[1][2] - '0'};
    int s3[2] = {LED4, argv[1][3] - '0'};

    printf("status: %s\n", argv[1]);
```

```
    for(int i = 0; i < argv[2][0] - '0'; i++) {
        pthread_create(&t1, NULL, child, s0);
        sleep(1);
        sem_post(&sem);
        pthread_create(&t2, NULL, child, s1);
        sleep(1);
        sem_post(&sem);
        pthread_create(&t3, NULL, child, s2);
        sleep(1);
        sem_post(&sem);
        pthread_create(&t4, NULL, child, s3);
        sleep(1);
        sem_post(&sem);
        pthread_join(t1, NULL);
        pthread_join(t2, NULL);
        pthread_join(t3, NULL);
        pthread_join(t4, NULL);
    }
    return 0;
}
```

(2)lab7_mutex.cpp

```cpp
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#include "gpio.h"

#define LED1 396
#define LED2 397
#define LED3 429
#define LED4 393

pthread_mutex_t mutex;
int gpioPin[4] = {LED1, LED2, LED3, LED4};

//子執行緒
void* child(void * data) {
    pthread_mutex_lock(&mutex);
    int* input = (int*) data;
    GPIO::Set_dir(input[0], "out");
    GPIO::Set_val(input[0], input[1]);
    printf("GPIO: %d status: %d\n", input[0], input[1]);
    pthread_mutex_unlock(&mutex);
    sleep(1);
    GPIO::Set_val(input[0], 0);
    pthread_exit(NULL);
}
```

```
int main(int argc, char** argv) {
    pthread_t t1, t2, t3, t4;
    pthread_mutex_init(&mutex, 0);
    for(int i = 0; i < 4; i++) {
        GPIO::Export(gpioPin[i]);
    }


    int s0[2] = {LED1, argv[1][0] - '0'};
    int s1[2] = {LED2, argv[1][1] - '0'};
    int s2[2] = {LED3, argv[1][2] - '0'};
    int s3[2] = {LED4, argv[1][3] - '0'};

    printf("status: %s\n", argv[1]);
```

```
    for(int i = 0; i < argv[2][0] - '0'; i++) {
        pthread_create(&t1, NULL, child, s0);
        sleep(1);
        pthread_create(&t2, NULL, child, s1);
        sleep(1);
        pthread_create(&t3, NULL, child, s2);
        sleep(1);
        pthread_create(&t4, NULL, child, s3);
        sleep(1);
        pthread_join(t1, NULL);
        pthread_join(t2, NULL);
        pthread_join(t3, NULL);
        pthread_join(t4, NULL);
    }


    return 0;
}
```

3. 實驗影片

https://youtu.be/ZnAdCQr4-wg

https://youtu.be/K1gbveCN1Qs

4. 組員貢獻

沈煒翔：34%

楊挺煜：33%

廖堃霖：33%

5. 心得

沈煒翔：

這次lab7，是要讓我們使用多執行續去執行多執行續去控制4顆LED，這次的實習我覺得比上次的驅動程式簡單一點，只要穩定控制他的4個子執行續，還有互斥鎖，以及信號鎖，就可以讓他穩定運行，然後其中的BUG我跟隊友修的時間也非常的快速，所以我們不到1小時就做完了，希望以後能更多用子執行續，去優化程式，讓他執行時間能夠縮短。

楊挺煜：

這次的實驗是用thread分別控制LED燈，做完我還是對thread這個東西沒什麼概念，我覺得Mutex也比Semaphore好懂一點，Semaphore上網找了一下還是沒很懂它的意思。期末忙完還需要再多研究一下了。

廖堃霖：

在這次微算機實習中，我們使用了多線程去操作LED燈，一開始在閱讀講義時，對於許多地方感到困惑，尤其是分析它如何運作時也花了大把的時間，但多虧了班上的好同學充滿著耐心的指導著我們，讓我了解到多線程的撰寫及運作，雖然寫的途中也有遇到困難，像是如何讓他順利輸出，但我與我的組員不畏艱難地克服它，並完成了Lab7。

這次實驗也讓我們聯想到了另一堂課，物件導向程式設計，應該也是運用了類似的技術構成。