운영체제 실습 4 과제 – 2021320057 컴퓨터학과 김의찬

1) thread_01.c

</1> create

</2> &tid, NULL, magic_box, (void*)10

</3.1> join

</3.2> tid, (void**)&new_number

```
kuc@2021320057:/mnt/shared_folder/assignment 4$ ./th1
Hey magic box, multiply 10 by 6
multiplying 10 by 6...
the new number is 60
kuc@2021320057:/mnt/shared_folder/assignment 4$ _
```

2) thread_02.c

</1.1> exit

</1.2> NULL

</2> create

</3> &tids[i], NULL, worker, &main_static

</4.1> join

</4.2> tids[i], NULL

```
kuc@2021320057:/mnt/shared_folder/assignment 4$ ./th2
global          main            thread          thread-static
0x579fa309f014  0x579fa309c27c  (nil)   (nil)
0x579fa309f014  0x579fa309f01c  0x74e1fa3ffeb4  0x579fa309f018
0x579fa309f014  0x579fa309f01c  0x74e1f8fffeb4  0x579fa309f018
0x579fa309f014  0x579fa309f01c  0x74e1f99ffeb4  0x579fa309f018
```

3) thread_03.c

</1> create

</2> &tids[i], NULL, worker, &progress

</3.1> join

</3.2> tids[i], &progress

</4.1> exit

</4.2> arg

```
kuc@2021320057:/mnt/shared_folder/assignment 4$ ./th3
1509108144
expected: 1000000
result: 997928
kuc@2021320057:/mnt/shared_folder/assignment 4$ _
```

4) thread_04.c

</1> create

</2> &tids[i], NULL, worker, &progress

</3.1> join

</3.2> tids[i], &progress

</4.1> mutex_lock

</4.2> &lock

</5.1> mutex_unlock

</5.2> &lock(코드에는 기존에 &가 있었으나 괄호 안 전체는 &lock)

</7.1> exit

</7.2> NULL

```
kuc@2021320057:/mnt/shared_folder/assignment 4$ ./th4
0
expected: 1000000
result: 1000000
```

5) thread_05.c

thread_05.c > main(int, char * [])

```c
#include <stdio.h>
#include <stdatomic.h>
#include <unistd.h>
#include <pthread.h>
#include <sys/wait.h>

#define NUM_SUBS 3
#define NUM_TASKS 3
#define NUM_TOTAL_TASK (NUM_SUBS * NUM_TASKS)
#define SPREADING 2

static _Atomic int cnt_task = NUM_TOTAL_TASK;

void spread_words(char* sub){\
    sleep(SPREADING);
    printf("[%s] spreading words...\n", sub);
    cnt_task--;
}

void* subordinate(void* arg)
{
    char sub[20];
    sprintf(sub, "%s %d", "subordinate", (int)arg);
    sleep(2);
    printf("[%s] as you wish\n", sub);

    for(int i = 0; i < 3; i++)
    {
        spread_words(sub);
    }
    sleep(1);
    //hint: finish up the threads
    pthread_exit(NULL);
}

void* king(void* arg)
{
    pthread_t tid;
    int status;
    printf("spread the words ");

    //hint: you should start your subordinates here
    //hint: for???
    for(int i = 0; i < NUM_SUBS; i++){
        status = pthread_create(&tid, NULL, subordinate, (void*)i);
        if(status != 0){
            printf("error");
            return -1;
        }
        pthread_detach(tid); //feel free to move this around
    }

    //BUT DON'T move it outside the king function

    printf("that I am king!\n");
    pthread_exit(NULL);
}

int main(int argc, char* argv[])
{
    pthread_t tid;
    int status;

    status = pthread_create(&tid, NULL, king, NULL);

    if (status != 0)
    {
        printf("error");
        return -1;
    }

    pthread_join(tid, NULL);

    //hint: make the main thread wait for the subordinates to finish
    //do NOT use pthread_join
    //hint: possible in 1 line
    while(cnt_task > 0) ;

    printf("The words have been spread...\n");
    return 0;
}
```

```
spread the words that I am king!
[subordinate 1] as you wish
[subordinate 0] as you wish
[subordinate 2] as you wish
[subordinate 0] spreading words...
[subordinate 2] spreading words...
[subordinate 1] spreading words...
[subordinate 0] spreading words...
[subordinate 2] spreading words...
[subordinate 1] spreading words...
[subordinate 0] spreading words...
[subordinate 2] spreading words...
[subordinate 1] spreading words...
The words have been spread...
```

Line 1~10: 헤더파일 및 상수 정의

Line 12: task 의 count 를 정의

Line 14~18: word 를 spreading 하는 함수. sleep 으로 기다렸다가 print 로 spreadig word 를 출력, 전역 카운터 감소

Line 20~34: surbordinate x 별로 대답을 한 뒤 말을 퍼트리는 함수. Sub 에 surbornate 와 인자로 받은 arg 를 저장한 뒤 as you wish 와 함께 출력한다. 그리고 이후 3 회에 걸쳐 spread words 를 호출한 뒤 1 초 sleep 후스레드를 exit 한다.

Line 36~57: king 함수는 spread the word 를 출력한 뒤 subourdinate 스레드를 생성하고 바로 detach 를 해 주고 있다. For 을 num_subs 만큼 반복하며 subordinate 스레드를 생성한다.

```c
for(int i = 0; i < NUM_SUBS; i++){
    status = pthread_create(&tid, NULL, subordinate, (void*)i);
    if(status != 0){
        printf("error");
        return -1;
    }
    pthread_detach(tid); //feel free to move this around
}
```

만들자마자 스레드를 분리 상태로 만들어 반환값이 필요 없는 surbordinate 스레드를 분리해주고 있다. 이후 that I am king 을 출력한 뒤 exit 해준다. 위 코드들의 sleep 으로 인해 순서가 꼬이지는 않는다.

Line 59~81: main 함수는 king 스레드를 생성한 뒤 join 으로 king 의 종료를 대기하고 있다.

While(cnt_task > 0) ; 로 busy waiting 을 구현해 스레드들이 작업을 끝낼 때까지 기다린다. 이후 종료문장을 출력한 뒤 프로그램을 종료한다.

6) thread_06.c

```c
#include <stdio.h>
#include <stdatomic.h>
#include <unistd.h>
#include <pthread.h>
#include <sys/wait.h>

#define NUM_SUBS 3
#define NUM_TASKS 3
#define NUM_TOTAL_TASK (NUM_SUBS * NUM_TASKS)
#define SPREADING 2

static _Atomic int cnt_task = NUM_TOTAL_TASK;
pthread_mutex_t lock;

void spread_words(char* sub){\
    sleep(SPREADING);
    printf("[%s] spreading words...\n", sub);
    cnt_task--;
}

void* subordinate(void* arg)
{
    char sub[20];
    sprintf(sub, "%s %d", "subordinate", (int)arg);
    printf("[%s] as you wish\n", sub);

    for(int i = 0; i < 3; i++)
    {
        spread_words(sub);
    }
    printf("[%s] I am done!\n", sub);
    //hint: finish up the threads
    pthread_exit(NULL);
}

void* king(void* arg)
{
    pthread_t tid[NUM_SUBS];
    int status;
    printf("spread the words that I am king!\n");


    //hint: start your subordinates
    //hint: for??????
    for(int j = 0; j < NUM_SUBS; j++){
        //start workers
        status = pthread_create(&tid[j], NULL, subordinate, (void*)j); //feel free to move this around
        if(status != 0){
            printf("error");
            return -1;
        }
    }

    //hint: try using some locks and for
    pthread_mutex_lock(&lock);
    for(int i = 0; i < NUM_SUBS; i++){
        pthread_join(tid[i], NULL); //feel free to move this around
    }
    pthread_mutex_unlock(&lock);
    //BUT DO NOT move it outside the king function

    pthread_exit(NULL);
}

```

```c
int main(int argc, char* argv[])
{
    pthread_t tid;
    int status;
    pthread_mutex_init(&lock, NULL);

    status = pthread_create(&tid, NULL, king, NULL);

    if (status != 0)
    {
        printf("error");
        return -1;
    }

    pthread_detach(tid);

    //added
    sleep(2);

    //hint: look at king function
    //do NOT use pthread_join
    //hint: locks
    pthread_mutex_lock(&lock);
    pthread_mutex_unlock(&lock);

    printf("The words have been spread...\n");\

    return 0;
}
```

```
kuc@2021320057:/mnt/shared_folder/assignment 4$ ./th6
spread the words that I am king!
[subordinate 0] as you wish
[subordinate 1] as you wish
[subordinate 2] as you wish
[subordinate 0] spreading words...
[subordinate 1] spreading words...
[subordinate 2] spreading words...
[subordinate 0] spreading words...
[subordinate 1] spreading words...
[subordinate 2] spreading words...
[subordinate 0] spreading words...
[subordinate 0] I am done!
[subordinate 2] spreading words...
[subordinate 2] I am done!
[subordinate 1] spreading words...
[subordinate 1] I am done!
```

Line 1~10: 헤더 파일 및 상수 정의

Line 12~13: 전역변수 선언 라인

Line 15~19: spread word 함수로 2만큼 sleep 후 sub 인자와 spreading words 출력, 전역 카운터를 감소시킨다.

Line 21~34: subordinate 함수로, arg 인자를 문자열 sub에 "subordinate"와 함께 저장. 이후 sub를 as you wish와 함께 출력한다. 이후 for를 3번 반복하여 spread_words를 호출한다. Spread words가 끝나면 I am done을 출력(신하마다 퍼트리기 종료 시에 done 출력)한다. Exit로 스레드 종료.

Line 36~63: king 함수는 스레드를 생성하고 진행한다. Spread the words that I am king을 출력한 뒤 for 구문으로 surbordinate 스레드를 생성한다. 이후 mutex lock과 unlock 사이에 또 다른 for구문 join을 넣어 king 함수 내부에서만 join을 걸어 서브 스레드가 전부 끝날 때까지 블록한다.

Line 65~93: main함수는 mutex를 init하고 king 스레드를 생성한 뒤 바로 분리한다. Sleep(2)를 통해 king이 서브 스레드를 생성할 시간을 확보하고, mutex lock과 unlock을 이용해 king 내부의 join이 끝나고 unlock이 될 때까지 기다린다. 마지막 문장을 출력한 뒤 종료한다.