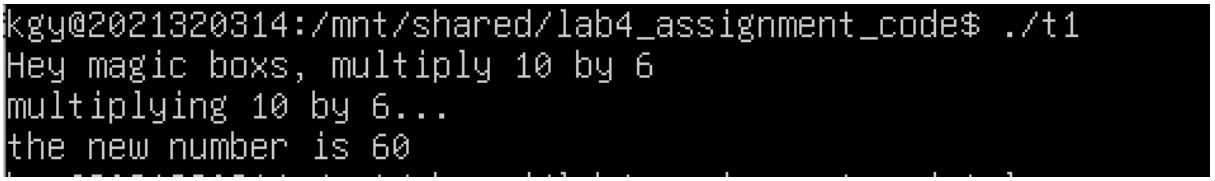
실습4

thread_01.c

</1>	create
</2>	&tid, NULL, magic_box, (void*)10
</3.1>	join
</3.2>	tid, (void**)&new_number

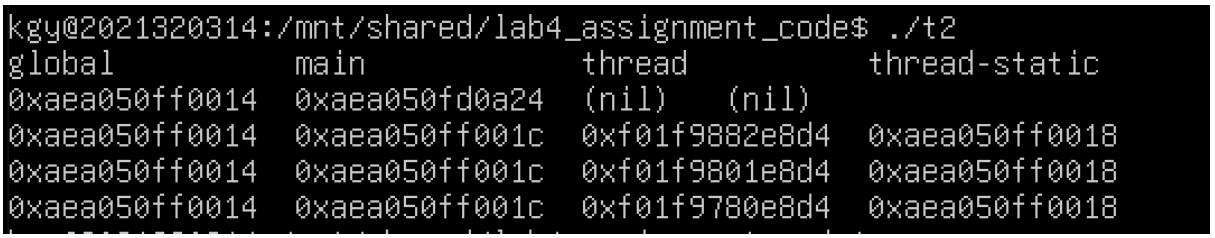
결과 스크린샷:



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

결과 스크린샷:



thread_03.c

</1>	create
</2>	&tids[i], NULL, worker, NULL

</3.1>	join
</3.2>	tids[i], (void**)&progress
</4.1>	exit
</4.2>	(void*)progress

결과 스크린샷:

```
kgy@2021320314:/mnt/shared/lab4_assignment_code$ ./t3
816637
expected: 1000000
result: 818002
```

thread_04.c

</1>	create
</2>	&tids[i], NULL, worker, NULL
</3.1>	join
</3.2>	tids[i], (void**)&progress
</4.1>	mutex_lock
</4.2>	&lock
</5.1>	mutex_unlock
</5.2>	lock
</7.1>	exit
</7.2>	(void*)progress

결과 스크린샷:

```
kgy@2021320314:/mnt/shared/lab4_assignment_code$ ./t4
880547
expected: 1000000
result: 1000000
```

thread_05.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;

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);
    pthread_exit(NULL);
}

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

    for(int i=0; i<NUM_SUBS; i++) {
        pthread_create(&tid, NULL, subordinate, (void*)i);
        pthread_detach(tid);
    }

    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);

    while(cnt_task > 0) usleep(1000);

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

설명:

line 1~5: 필요한 라이브러리 헤더들을 포함

7: `NUM_SUBS` :subordinate의 개수

8: `NUM_TASKS` : 하나의 subordinate thread가 수행할 작업의 개수

9: `NUM_TOTAL_TASK` : 전체 작업 개수 (NUM_SUBS x NUM_TASKS)

10: `SPREADING`: spread_words 함수에서 쓸 sleep시간

12: 전역 원자 변수 `cnt_task`를 전체 작업 수로 초기화. 모든 작업이 끝나면 0

line 14~18: `spread_words` 함수

- 15: 주어진 시간(sleep)만큼 대기 후
- 16: "spreading words..." 메시지 출력
- 17: 작업 완료 시 `cnt_task`를 1 감소

line 20~33: `subordinate` 함수

- 22~23: "subordinate i" 형식으로 스레드 이름 만듦
- 24: 2초 대기
- 25: "as you wish" 메시지 출력
- 27~30: 3번 반복하며(작업 개수만큼) spread_words 호출
- 31: 작업 후 1초 대기
- 32: 스레드 종료(pthread_exit)

line 35~48: `king` 함수

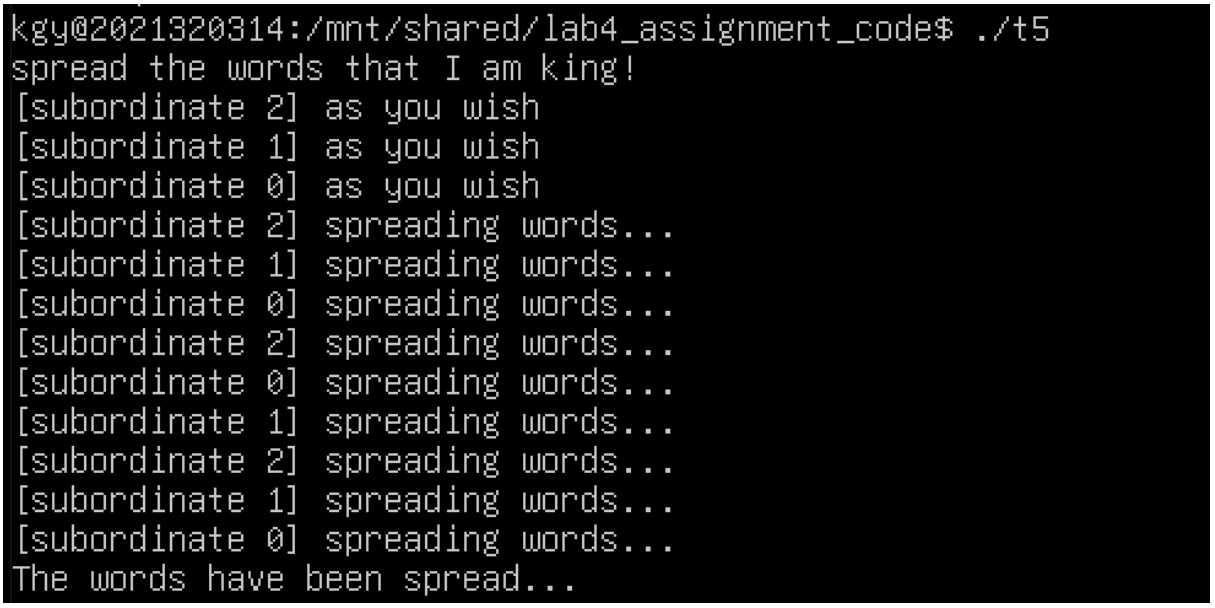
- 37: subordinate 스레드 생성을 위한 id 변수 선언
- 39: "spread the words " 메시지 출력.
- 41~44: NUM_SUBS만큼 subordinate 스레드 생성
  - 42: 각 스레드를 생성, arg로 i를 넘김
  - 43: 생성된 스레드를 바로 detach
- 46: "that I am king!" 출력
- 47: 스레드 종료

line 50~69: `main` 함수

- 52~53: king 스레드 생성을 위한 변수 선언
- 55: king 스레드 생성(pthread_create)
- 57~61: 생성 실패 시 에러 처리
- 63: king 스레드가 종료될 때까지 기다림(pthread_join)
- 65: 남은 작업 수(cnt_task)가 0이 될 때까지 반복 대기
- 67: "The words have been spread..." 메시지 출력

- 68: 프로그램 정상 종료
- 모든 작업이 끝나면 "The words have been spread..." 메시지를 출력하고 main 함수 종료

결과 스크린샷:



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);
    pthread_exit(NULL);
}

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

    for(int i = 0; i < NUM_SUBS; i++) {
        pthread_create(&tid[i], NULL, subordinate, (void*)i);
    }

    pthread_mutex_lock(&lock);
    for(int i=0; i < NUM_SUBS; i++){
        pthread_join(tid[i], NULL);
    }
    pthread_mutex_unlock(&lock);

    pthread_exit(NULL);
}

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);

    sleep(2);

    pthread_mutex_lock(&lock);
    pthread_mutex_unlock(&lock);
    pthread_mutex_destroy(&lock);

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

    return 0;
}
```

설명:

line 1~5: 필요한 라이브러리 헤더들을 포함

7: `NUM_SUBS` :subordinate의 개수

8: `NUM_TASKS` : 하나의 subordinate thread가 수행할 작업의 개수

9: `NUM_TOTAL_TASK` : 전체 작업 개수 (NUM_SUBS x NUM_TASKS)

10: `SPREADING` : spread_words 함수에서 쓸 sleep시간

12: 전역 원자 변수 `cnt_task` 를 전체 작업 수로 초기화. 모든 작업이 끝나면 0

13: mutex 선언. 여러 스레드가 동시에 자원에 접근하지 않게 하기 위한 용도

line 15~19: `spread_words` 함수

- 16: SPREADING 초 동안 대기(sleep)
- 17: "[subordinate i] spreading words..." 메시지 출력
- 18: `cnt_task` 를 1 감소

line 21~32: `subordinate` 함수

- 23~24: 스레드 이름("subordinate i")을 sub 배열에 저장
- 25: "[subordinate i] as you wish" 메시지 출력
- 27~30: 3번 반복하며 spread_words 호출
- 31: "[subordinate i] I am done!" 메시지 출력
- 32: 스레드 종료(pthread_exit)

line 35~51: `king` 함수

- 37: subordinate 스레드의 id를 저장할 배열 선언
- 39: "spread the words that I am king!" 메시지 출력
- 41~43: for문을 돌며 NUM_SUBS만큼 subordinate 스레드 생성
  - 42: subordinate 스레드 생성, i 값을 인자로 넘김
- 45: mutex 락을 획득
- 46~48: 모든 subordinate가 끝날 때까지 join으로 대기
- 49: 락 해제(main이 아래에서 lock을 얻을 수 있게 됨)
- 51: king 스레드 종료(pthread_exit)

53~79: `main` 함수

- 56~57: mutex 초기화
- 60: king 스레드 생성(pthread_create)
- 62~66: 생성 실패 시 에러 메시지 출력 후 종료
- 68: king 스레드를 detach
- 70: king이 lock을 잡을 시간을 벌기 위해 잠시 대기
- 71~72: king이 lock을 해제할 때까지 main이 lock을 얻으려고 시도(블로킹)
- 74: mutex 리소스 해제
- 76: "The words have been spread..." 메시지 출력
- 79: 프로그램 종료

결과 스크린샷: