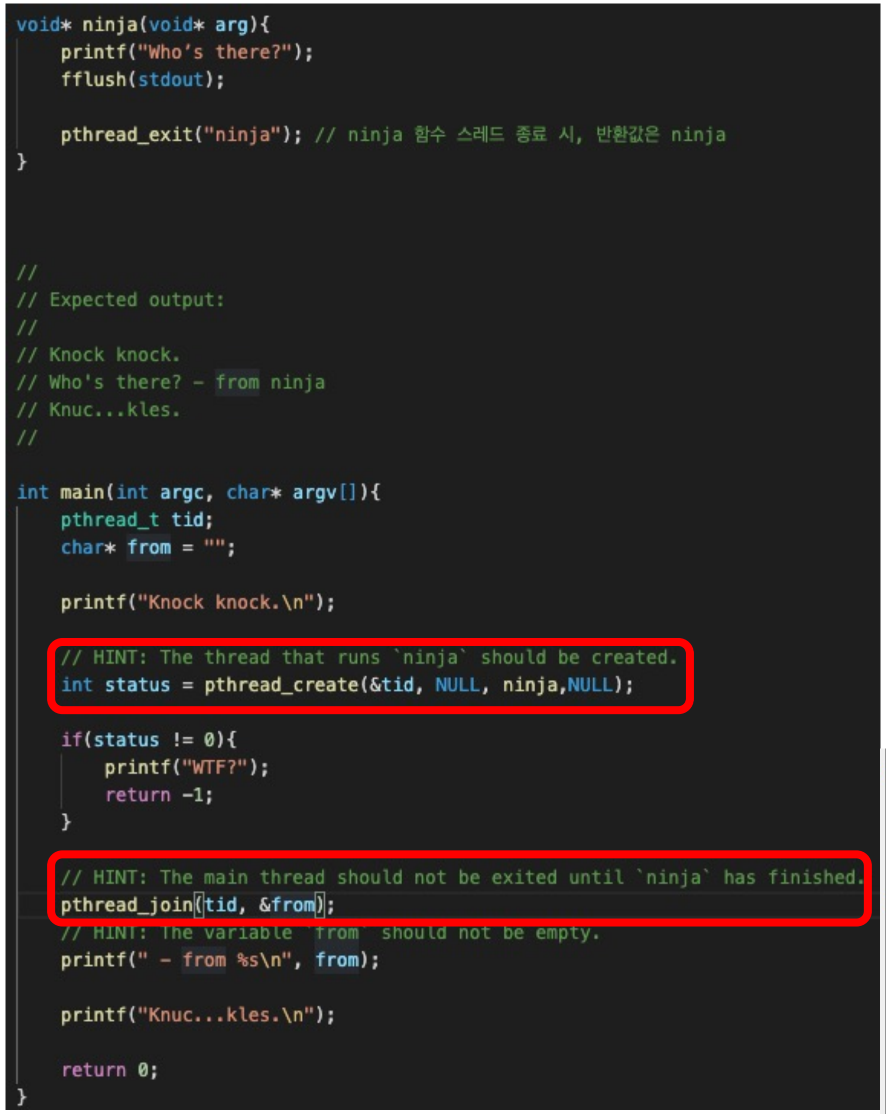
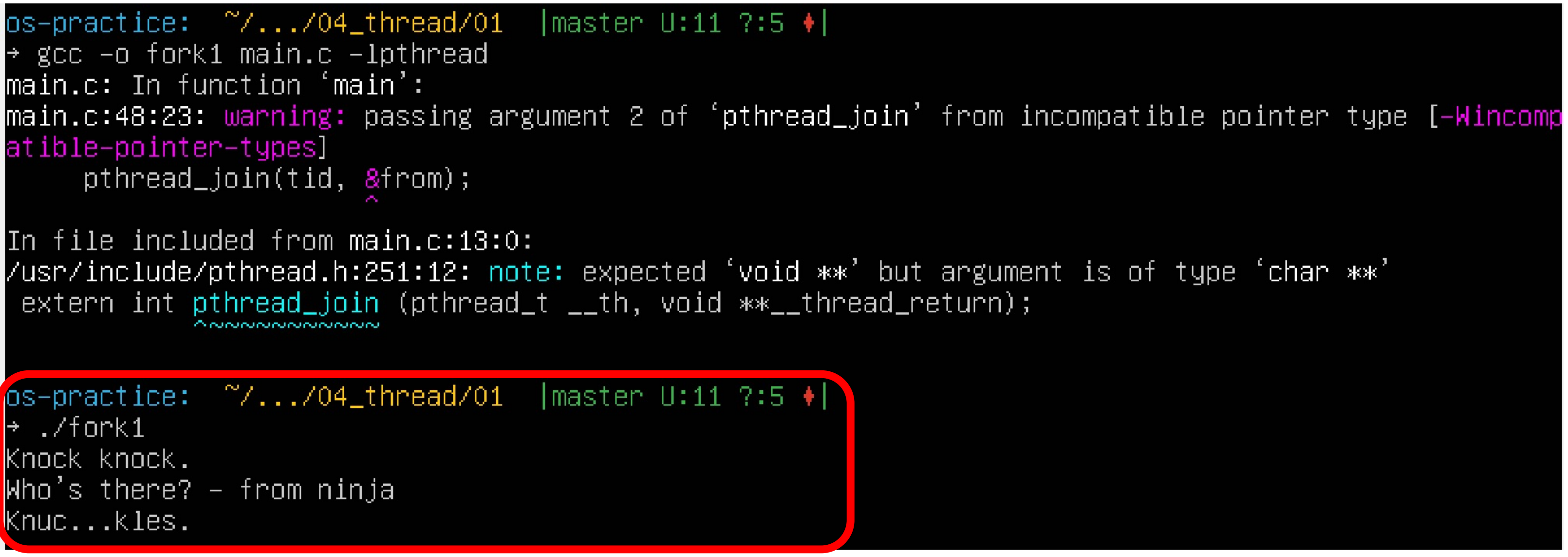
운영체제 과제
스레드 실습

2013130890 영어영문학과 김승태

01번 문제 코드 & 실행 결과

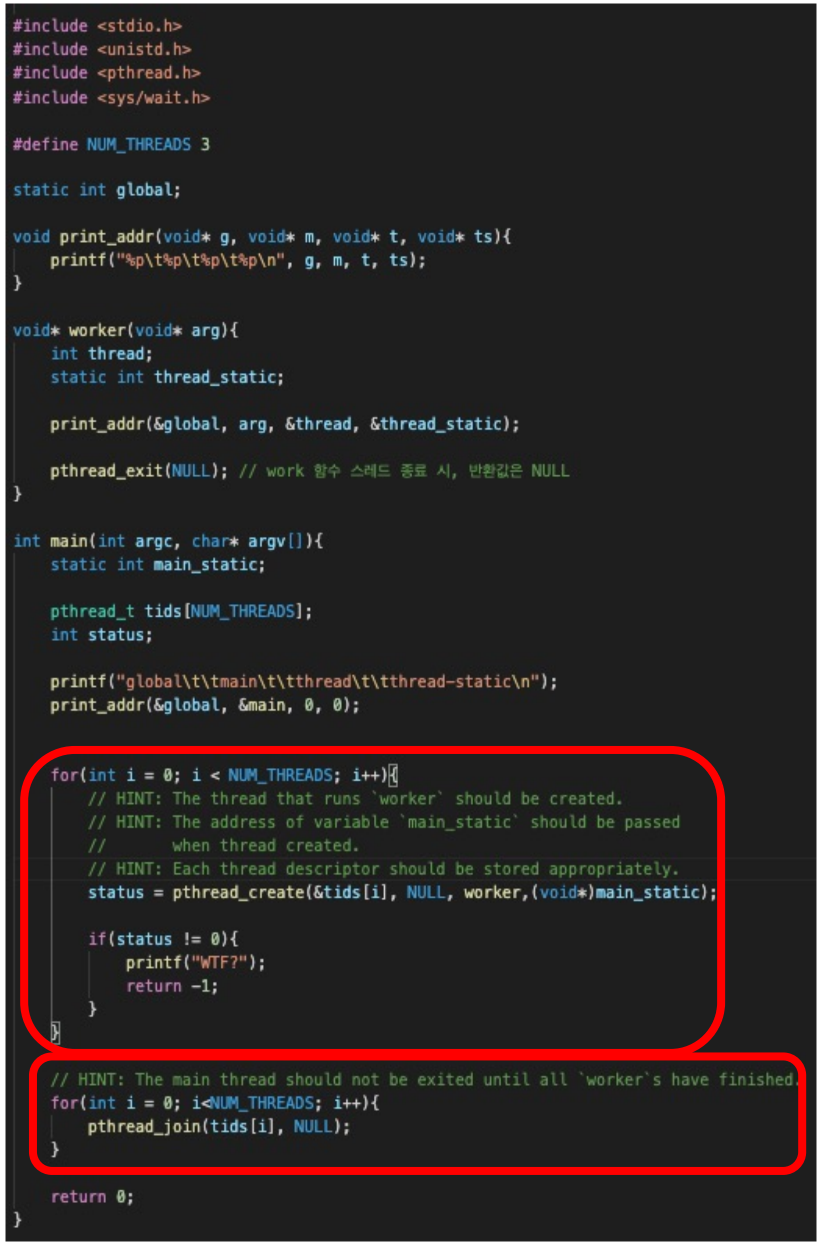


```c
void* ninja(void* arg){
    printf("Who's there?");
    fflush(stdout);

    pthread_exit("ninja"); // ninja 함수 스레드 종료 시, 반환값은 ninja
}



//
// Expected output:
//
// Knock knock.
// Who's there? - from ninja
// Knuc...kles.
//

int main(int argc, char* argv[]){
    pthread_t tid;
    char* from = "";

    printf("Knock knock.\n");

    // HINT: The thread that runs `ninja` should be created.
    int status = pthread_create(&tid, NULL, ninja,NULL);

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

    // HINT: The main thread should not be exited until `ninja` has finished.
    pthread_join(tid, &from);
    // HINT: The variable `from` should not be empty.
    printf(" - from %s\n", from);

    printf("Knuc...kles.\n");

    return 0;
}
```

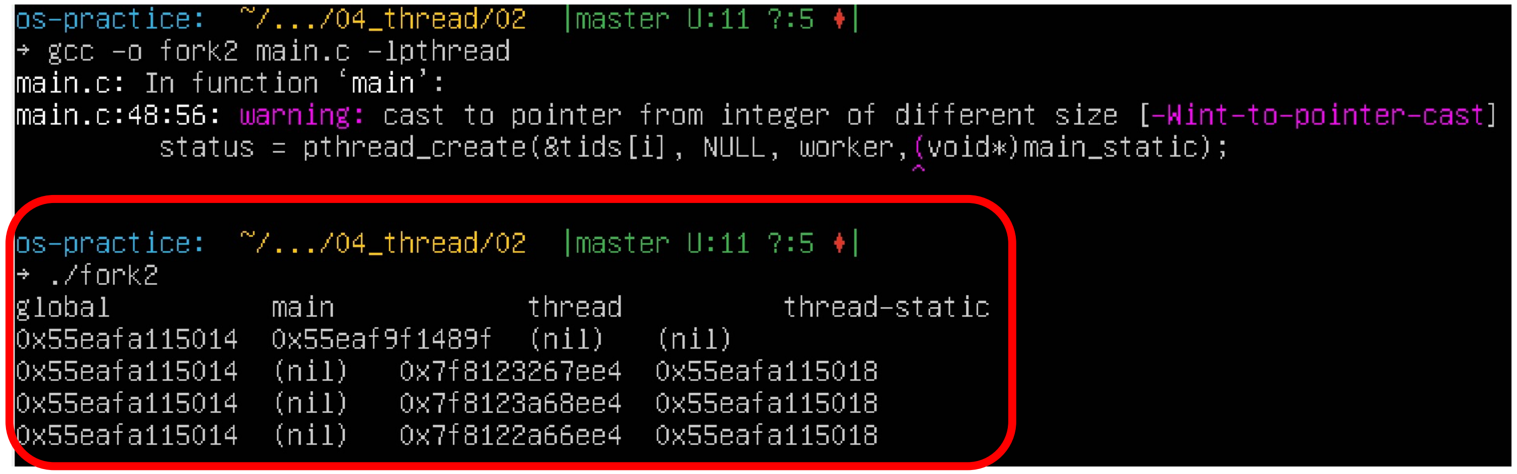


```
os-practice:  ~/.../04_thread/01  |master U:11 ?:5 ♦|
→ gcc -o fork1 main.c -lpthread
main.c: In function ‘main’:
main.c:48:23: warning: passing argument 2 of ‘pthread_join’ from incompatible pointer type [-Wincomp
atible-pointer-types]
     pthread_join(tid, &from);
                       ^
In file included from main.c:13:0:
/usr/include/pthread.h:251:12: note: expected ‘void **’ but argument is of type ‘char **’
 extern int pthread_join (pthread_t __th, void **__thread_return);
            ^~~~~~~~~~~~

os-practice:  ~/.../04_thread/01  |master U:11 ?:5 ♦|
→ ./fork1
Knock knock.
Who's there? - from ninja
Knuc...kles.
```

02번 문제 코드 & 실행 결과

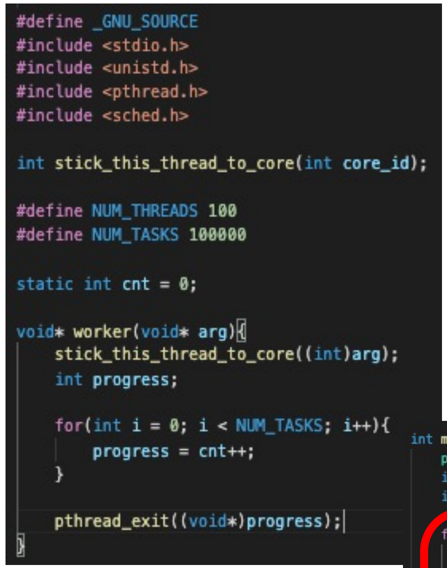


```c
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>
#include <sys/wait.h>

#define NUM_THREADS 3

static int global;

void print_addr(void* g, void* m, void* t, void* ts){
    printf("%p\t%p\t%p\t%p\n", g, m, t, ts);
}

void* worker(void* arg){
    int thread;
    static int thread_static;

    print_addr(&global, arg, &thread, &thread_static);

    pthread_exit(NULL); // work 함수 스레드 종료 시, 반환값은 NULL
}

int main(int argc, char* argv[]){
    static int main_static;

    pthread_t tids[NUM_THREADS];
    int status;

    printf("global\t\tmain\t\tthread\t\tthread-static\n");
    print_addr(&global, &main, 0, 0);


    for(int i = 0; i < NUM_THREADS; i++){
        // HINT: The thread that runs `worker` should be created.
        // HINT: The address of variable `main_static` should be passed
        //       when thread created.
        // HINT: Each thread descriptor should be stored appropriately.
        status = pthread_create(&tids[i], NULL, worker,(void*)main_static);

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

    // HINT: The main thread should not be exited until all `worker`s have finished.
    for(int i = 0; i<NUM_THREADS; i++){
        pthread_join(tids[i], NULL);
    }

    return 0;
}
```

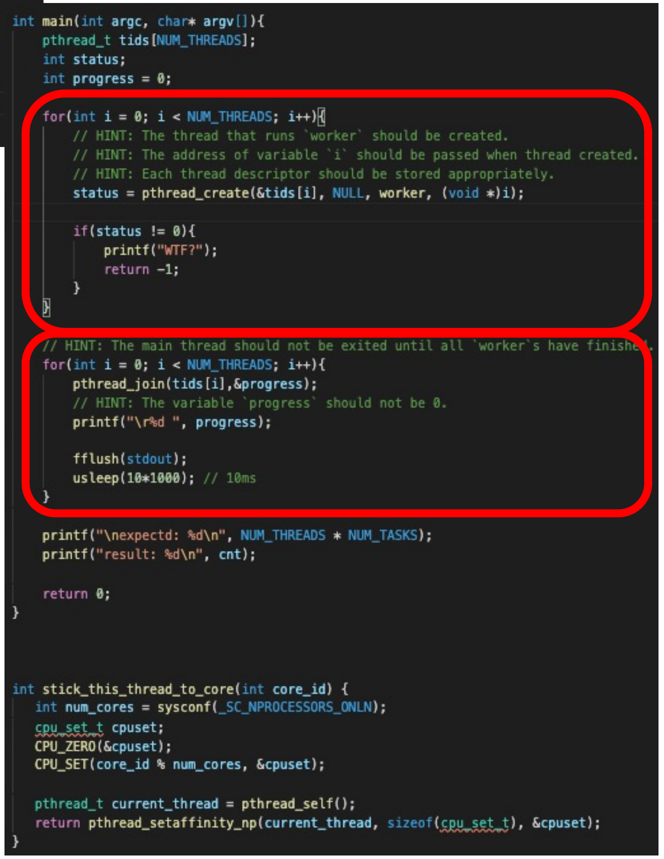


```
os-practice:  ~/.../04_thread/02  |master U:11 ?:5 ♦|
→ gcc -o fork2 main.c -lpthread
main.c: In function ‘main’:
main.c:48:56: warning: cast to pointer from integer of different size [-Wint-to-pointer-cast]
        status = pthread_create(&tids[i], NULL, worker,(void*)main_static);
                                                       ^

os-practice:  ~/.../04_thread/02  |master U:11 ?:5 ♦|
→ ./fork2
global          main            thread          thread-static
0x55eafa115014  0x55eaf9f1489f  (nil)   (nil)
0x55eafa115014  (nil)   0x7f8123267ee4  0x55eafa115018
0x55eafa115014  (nil)   0x7f8123a68ee4  0x55eafa115018
0x55eafa115014  (nil)   0x7f8122a66ee4  0x55eafa115018
```

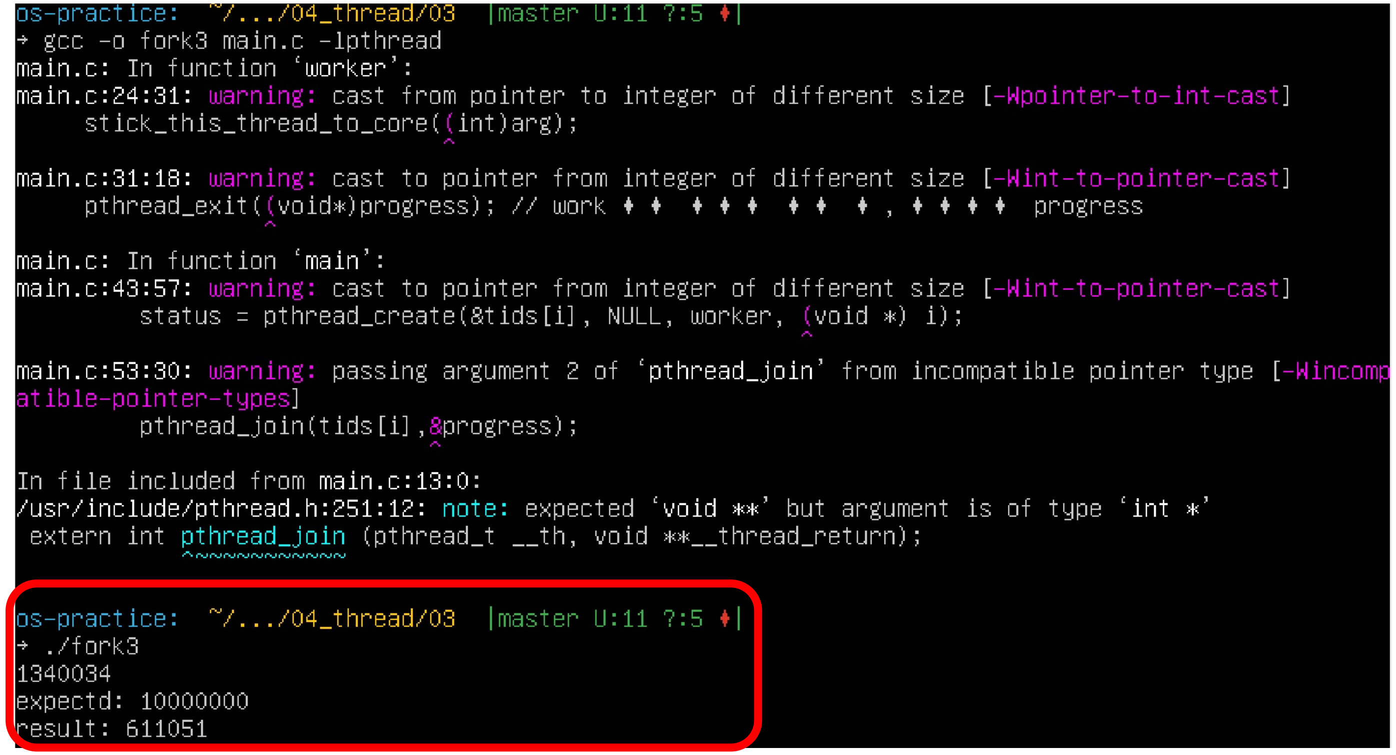
03번 문제 코드 & 실행 결과



```c
#define _GNU_SOURCE
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>
#include <sched.h>

int stick_this_thread_to_core(int core_id);

#define NUM_THREADS 100
#define NUM_TASKS 100000

static int cnt = 0;

void* worker(void* arg){
    stick_this_thread_to_core((int)arg);
    int progress;

    for(int i = 0; i < NUM_TASKS; i++){
        progress = cnt++;
    }

    pthread_exit((void*)progress);
}
```

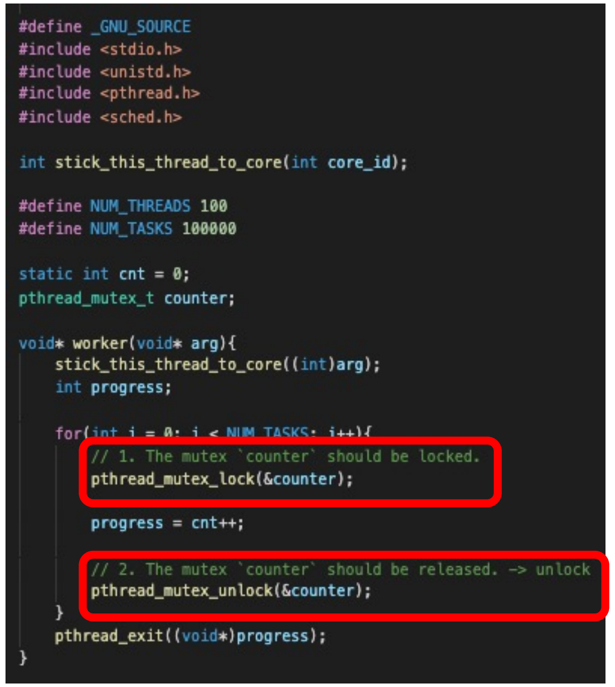


```c
int main(int argc, char* argv[]){
    pthread_t tids[NUM_THREADS];
    int status;
    int progress = 0;

    for(int i = 0; i < NUM_THREADS; i++){
        // HINT: The thread that runs `worker` should be created.
        // HINT: The address of variable `i` should be passed when thread created.
        // HINT: Each thread descriptor should be stored appropriately.
        status = pthread_create(&tids[i], NULL, worker, (void *)i);

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

    // HINT: The main thread should not be exited until all `worker`s have finishe
    for(int i = 0; i < NUM_THREADS; i++){
        pthread_join(tids[i],&progress);
        // HINT: The variable `progress` should not be 0.
        printf("\r%d ", progress);

        fflush(stdout);
        usleep(10*1000); // 10ms
    }

    printf("\nexpectd: %d\n", NUM_THREADS * NUM_TASKS);
    printf("result: %d\n", cnt);

    return 0;
}


int stick_this_thread_to_core(int core_id) {
   int num_cores = sysconf(_SC_NPROCESSORS_ONLN);
   cpu_set_t cpuset;
   CPU_ZERO(&cpuset);
   CPU_SET(core_id % num_cores, &cpuset);

   pthread_t current_thread = pthread_self();
   return pthread_setaffinity_np(current_thread, sizeof(cpu_set_t), &cpuset);
}
```

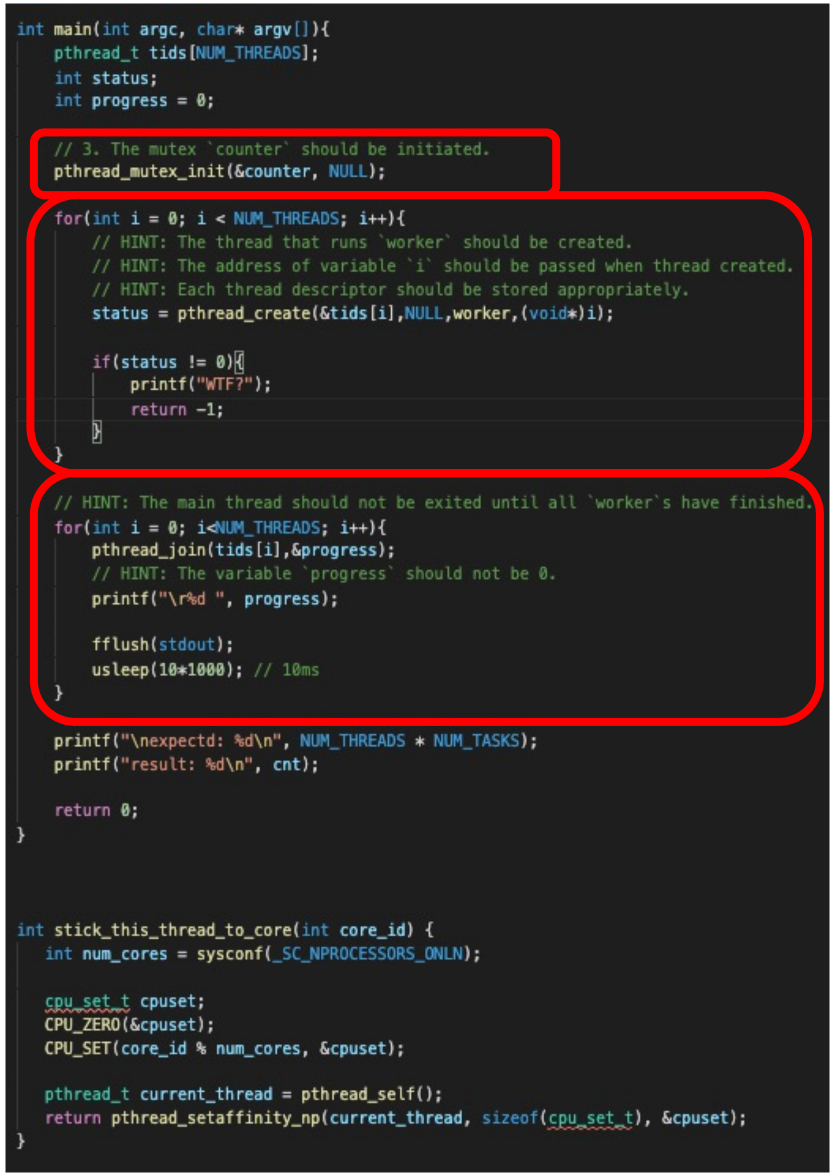


```
os-practice:  ~/.../04_thread/03  |master U:11 ?:5 ♦|
→ gcc -o fork3 main.c -lpthread
main.c: In function ‘worker’:
main.c:24:31: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
     stick_this_thread_to_core((int)arg);
                               ^
main.c:31:18: warning: cast to pointer from integer of different size [-Wint-to-pointer-cast]
     pthread_exit((void*)progress); // work ♦ ♦  ♦ ♦ ♦  ♦ ♦  ♦ , ♦ ♦ ♦ ♦  progress
                  ^
main.c: In function ‘main’:
main.c:43:57: warning: cast to pointer from integer of different size [-Wint-to-pointer-cast]
         status = pthread_create(&tids[i], NULL, worker, (void *) i);
                                                         ^
main.c:53:30: warning: passing argument 2 of ‘pthread_join’ from incompatible pointer type [-Wincomp
atible-pointer-types]
         pthread_join(tids[i],&progress);
                              ^
In file included from main.c:13:0:
/usr/include/pthread.h:251:12: note: expected ‘void **’ but argument is of type ‘int *’
 extern int pthread_join (pthread_t __th, void **__thread_return);
            ^~~~~~~~~~~~

os-practice:  ~/.../04_thread/03  |master U:11 ?:5 ♦|
→ ./fork3
1340034
expectd: 10000000
result: 611051
```

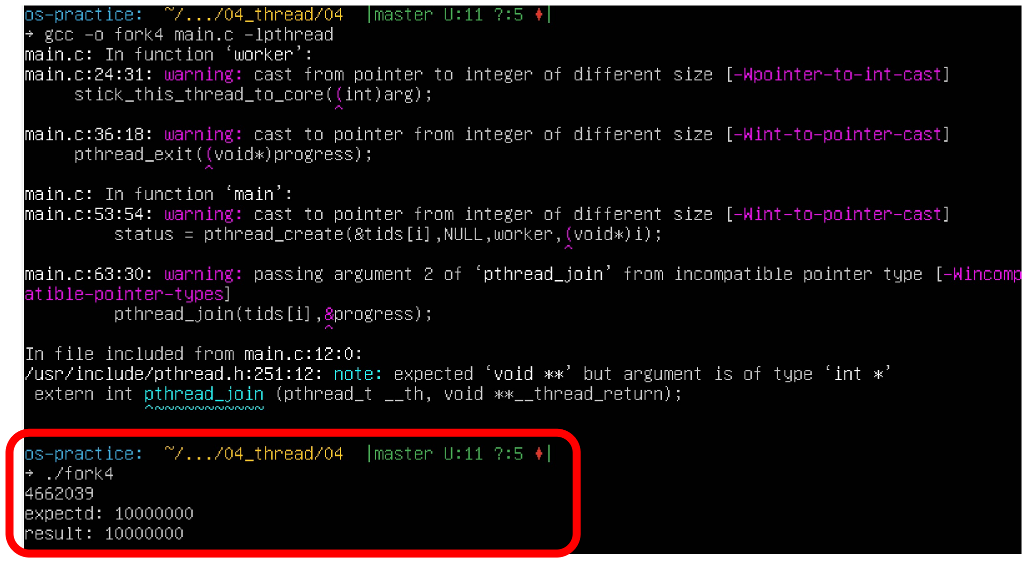
04번 문제 코드 & 실행 결과



```c
#define _GNU_SOURCE
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>
#include <sched.h>

int stick_this_thread_to_core(int core_id);

#define NUM_THREADS 100
#define NUM_TASKS 100000

static int cnt = 0;
pthread_mutex_t counter;

void* worker(void* arg){
    stick_this_thread_to_core((int)arg);
    int progress;

    for(int i = 0; i < NUM_TASKS; i++){
        // 1. The mutex `counter` should be locked.
        pthread_mutex_lock(&counter);

        progress = cnt++;

        // 2. The mutex `counter` should be released. -> unlock
        pthread_mutex_unlock(&counter);
    }
    pthread_exit((void*)progress);
}
```

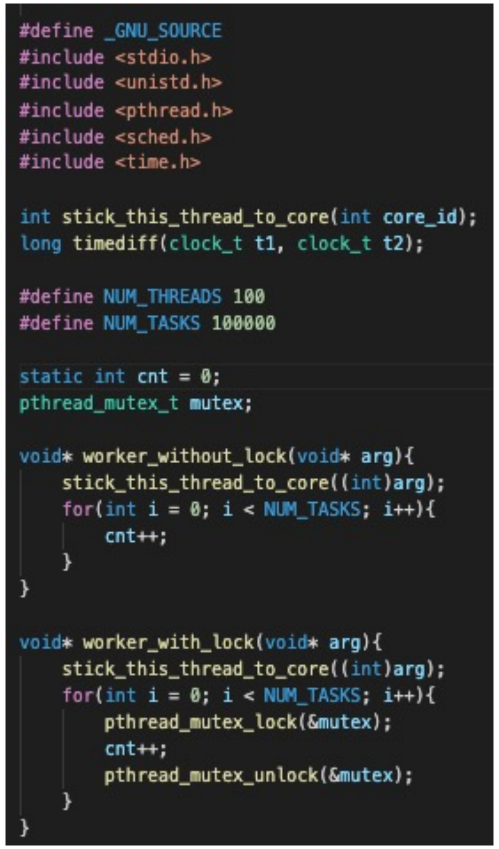


```c
int main(int argc, char* argv[]){
    pthread_t tids[NUM_THREADS];
    int status;
    int progress = 0;

    // 3. The mutex `counter` should be initiated.
    pthread_mutex_init(&counter, NULL);

    for(int i = 0; i < NUM_THREADS; i++){
        // HINT: The thread that runs `worker` should be created.
        // HINT: The address of variable `i` should be passed when thread created.
        // HINT: Each thread descriptor should be stored appropriately.
        status = pthread_create(&tids[i],NULL,worker,(void*)i);

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

    // HINT: The main thread should not be exited until all `worker`s have finished.
    for(int i = 0; i<NUM_THREADS; i++){
        pthread_join(tids[i],&progress);
        // HINT: The variable `progress` should not be 0.
        printf("\r%d ", progress);

        fflush(stdout);
        usleep(10*1000); // 10ms
    }

    printf("\nexpectd: %d\n", NUM_THREADS * NUM_TASKS);
    printf("result: %d\n", cnt);

    return 0;
}


int stick_this_thread_to_core(int core_id) {
   int num_cores = sysconf(_SC_NPROCESSORS_ONLN);

   cpu_set_t cpuset;
   CPU_ZERO(&cpuset);
   CPU_SET(core_id % num_cores, &cpuset);

   pthread_t current_thread = pthread_self();
   return pthread_setaffinity_np(current_thread, sizeof(cpu_set_t), &cpuset);
}
```

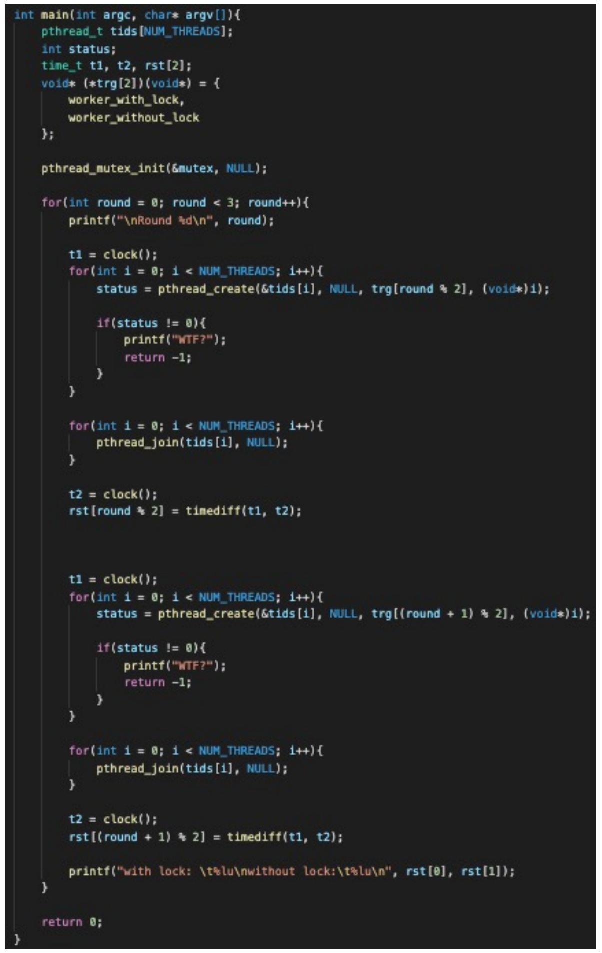


```
os-practice:  ~/.../04_thread/04  |master U:11 ?:5 ♦|
→ gcc -o fork4 main.c -lpthread
main.c: In function ‘worker’:
main.c:24:31: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
     stick_this_thread_to_core((int)arg);
                               ^
main.c:36:18: warning: cast to pointer from integer of different size [-Wint-to-pointer-cast]
     pthread_exit((void*)progress);
                  ^
main.c: In function ‘main’:
main.c:53:54: warning: cast to pointer from integer of different size [-Wint-to-pointer-cast]
         status = pthread_create(&tids[i],NULL,worker,(void*)i);
                                                      ^
main.c:63:30: warning: passing argument 2 of ‘pthread_join’ from incompatible pointer type [-Wincomp
atible-pointer-types]
         pthread_join(tids[i],&progress);
                              ^
In file included from main.c:12:0:
/usr/include/pthread.h:251:12: note: expected ‘void **’ but argument is of type ‘int *’
 extern int pthread_join (pthread_t __th, void **__thread_return);
            ^~~~~~~~~~~~

os-practice:  ~/.../04_thread/04  |master U:11 ?:5 ♦|
→ ./fork4
4662039
expectd: 10000000
result: 10000000
```

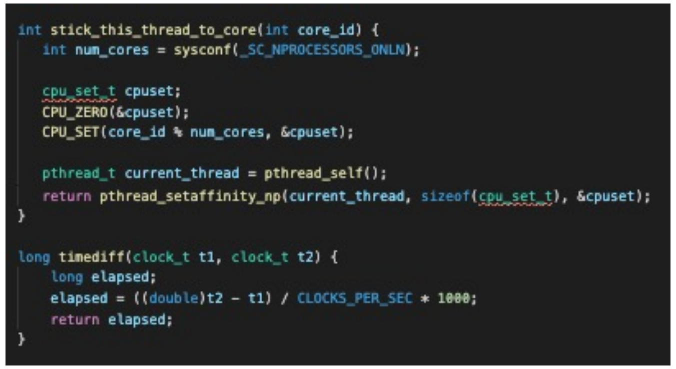
05번 문제 코드 & 실행 결과



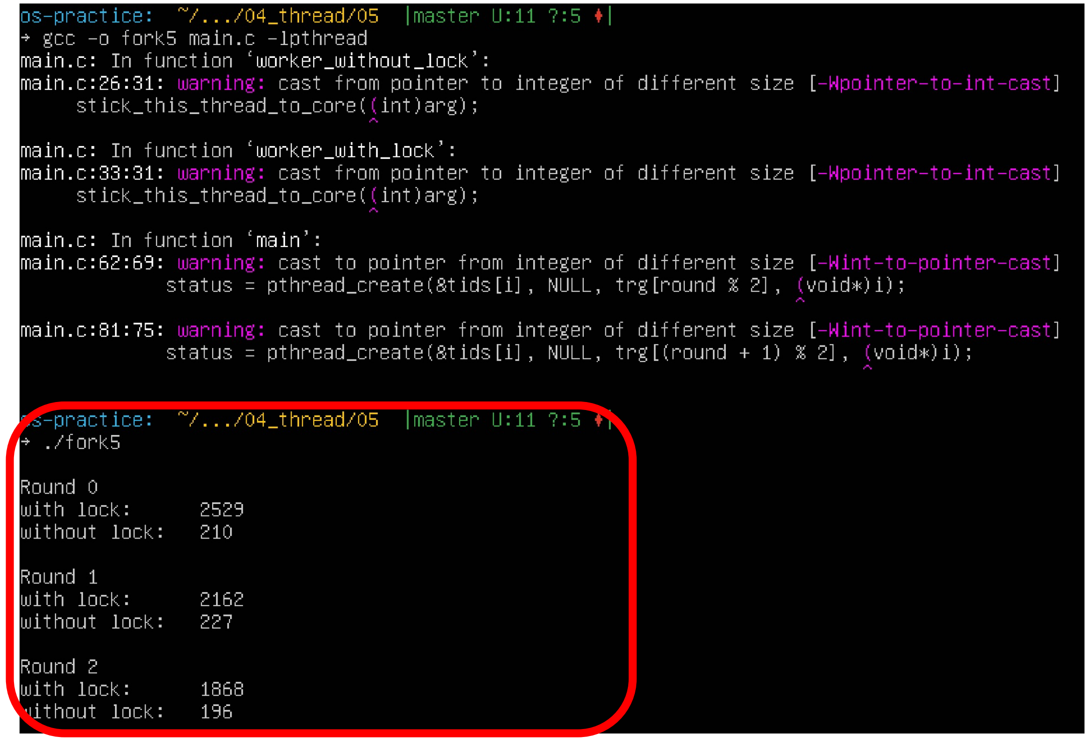
```c
#define _GNU_SOURCE
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>
#include <sched.h>
#include <time.h>

int stick_this_thread_to_core(int core_id);
long timediff(clock_t t1, clock_t t2);

#define NUM_THREADS 100
#define NUM_TASKS 100000

static int cnt = 0;
pthread_mutex_t mutex;

void* worker_without_lock(void* arg){
    stick_this_thread_to_core((int)arg);
    for(int i = 0; i < NUM_TASKS; i++){
        cnt++;
    }
}

void* worker_with_lock(void* arg){
    stick_this_thread_to_core((int)arg);
    for(int i = 0; i < NUM_TASKS; i++){
        pthread_mutex_lock(&mutex);
        cnt++;
        pthread_mutex_unlock(&mutex);
    }
}
```



```c
int main(int argc, char* argv[]){
    pthread_t tids[NUM_THREADS];
    int status;
    time_t t1, t2, rst[2];
    void* (*trg[2])(void*) = {
        worker_with_lock,
        worker_without_lock
    };

    pthread_mutex_init(&mutex, NULL);

    for(int round = 0; round < 3; round++){
        printf("\nRound %d\n", round);

        t1 = clock();
        for(int i = 0; i < NUM_THREADS; i++){
            status = pthread_create(&tids[i], NULL, trg[round % 2], (void*)i);

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

        for(int i = 0; i < NUM_THREADS; i++){
            pthread_join(tids[i], NULL);
        }

        t2 = clock();
        rst[round % 2] = timediff(t1, t2);



        t1 = clock();
        for(int i = 0; i < NUM_THREADS; i++){
            status = pthread_create(&tids[i], NULL, trg[(round + 1) % 2], (void*)i);

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

        for(int i = 0; i < NUM_THREADS; i++){
            pthread_join(tids[i], NULL);
        }

        t2 = clock();
        rst[(round + 1) % 2] = timediff(t1, t2);

        printf("with lock: \t%lu\nwithout lock:\t%lu\n", rst[0], rst[1]);
    }

    return 0;
}
```

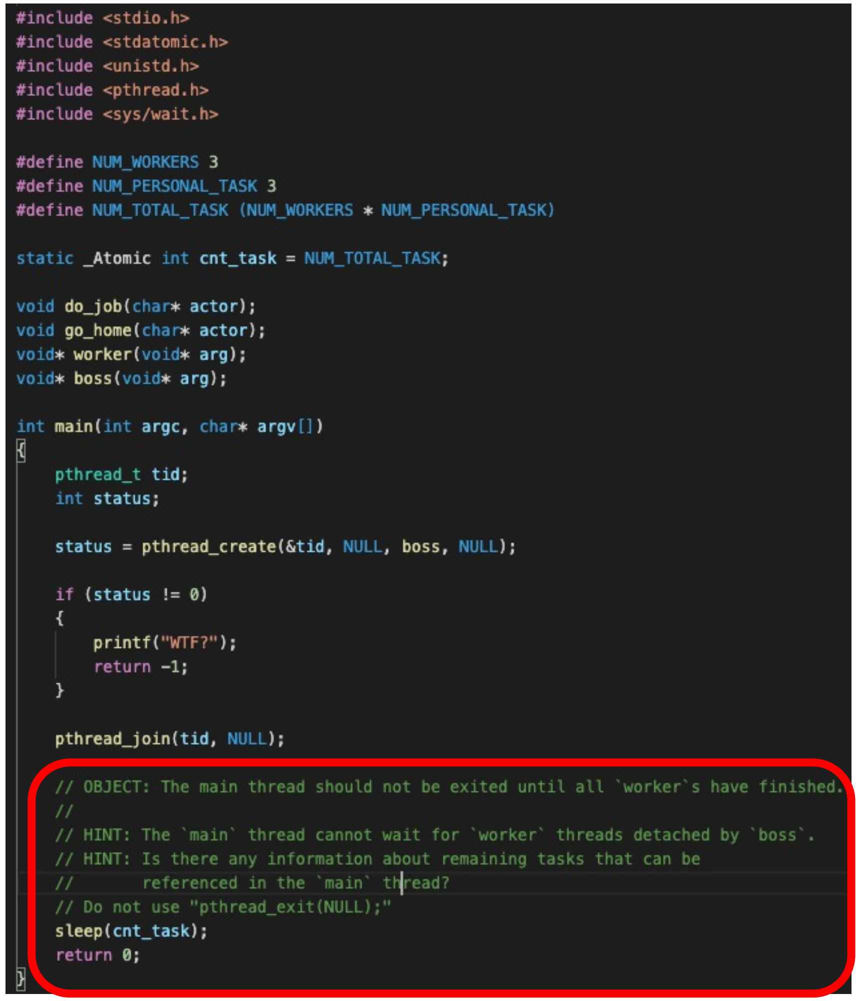


```c
int stick_this_thread_to_core(int core_id) {
   int num_cores = sysconf(_SC_NPROCESSORS_ONLN);

   cpu_set_t cpuset;
   CPU_ZERO(&cpuset);
   CPU_SET(core_id % num_cores, &cpuset);

   pthread_t current_thread = pthread_self();
   return pthread_setaffinity_np(current_thread, sizeof(cpu_set_t), &cpuset);
}

long timediff(clock_t t1, clock_t t2) {
    long elapsed;
    elapsed = ((double)t2 - t1) / CLOCKS_PER_SEC * 1000;
    return elapsed;
}
```

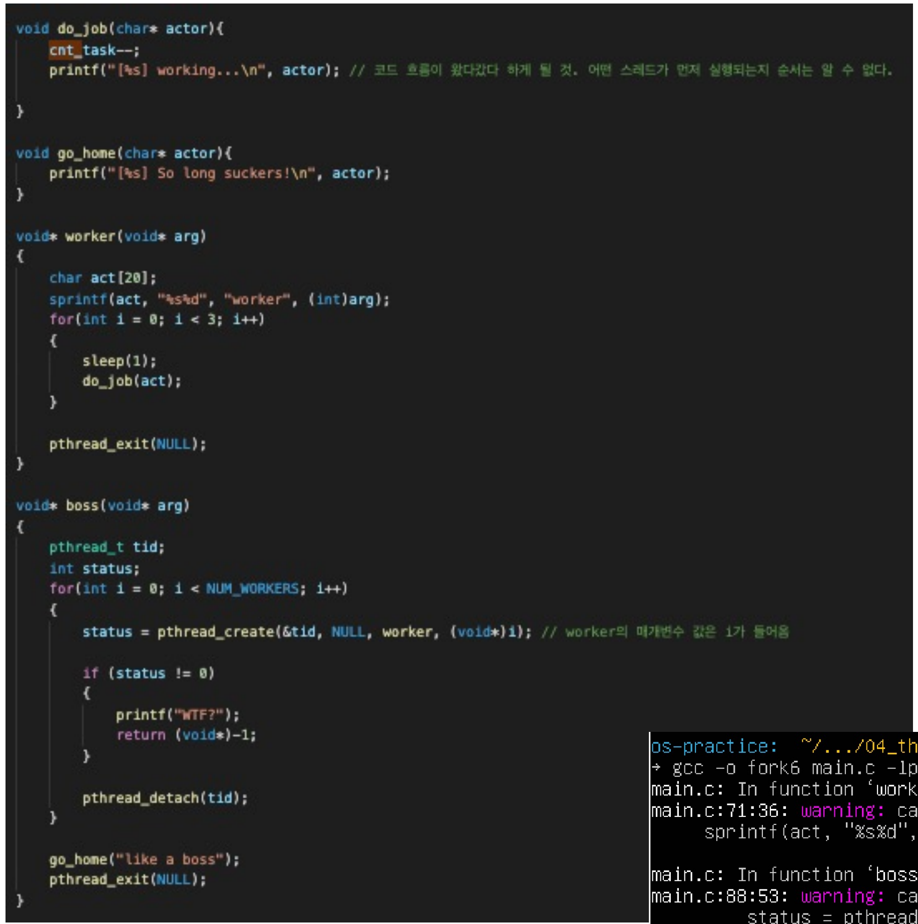


```
os-practice:  ~/.../04_thread/05  |master U:11 ?:5 ✚|
→ gcc -o fork5 main.c -lpthread
main.c: In function ‘worker_without_lock’:
main.c:26:31: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
     stick_this_thread_to_core((int)arg);
                               ^
main.c: In function ‘worker_with_lock’:
main.c:33:31: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
     stick_this_thread_to_core((int)arg);
                               ^
main.c: In function ‘main’:
main.c:62:69: warning: cast to pointer from integer of different size [-Wint-to-pointer-cast]
             status = pthread_create(&tids[i], NULL, trg[round % 2], (void*)i);
                                                                     ^
main.c:81:75: warning: cast to pointer from integer of different size [-Wint-to-pointer-cast]
             status = pthread_create(&tids[i], NULL, trg[(round + 1) % 2], (void*)i);
                                                                           ^

os-practice:  ~/.../04_thread/05  |master U:11 ?:5 ✚|
→ ./fork5

Round 0
with lock:      2529
without lock:   210

Round 1
with lock:      2162
without lock:   227

Round 2
with lock:      1868
without lock:   196
```

06번 문제 코드 & 실행 결과

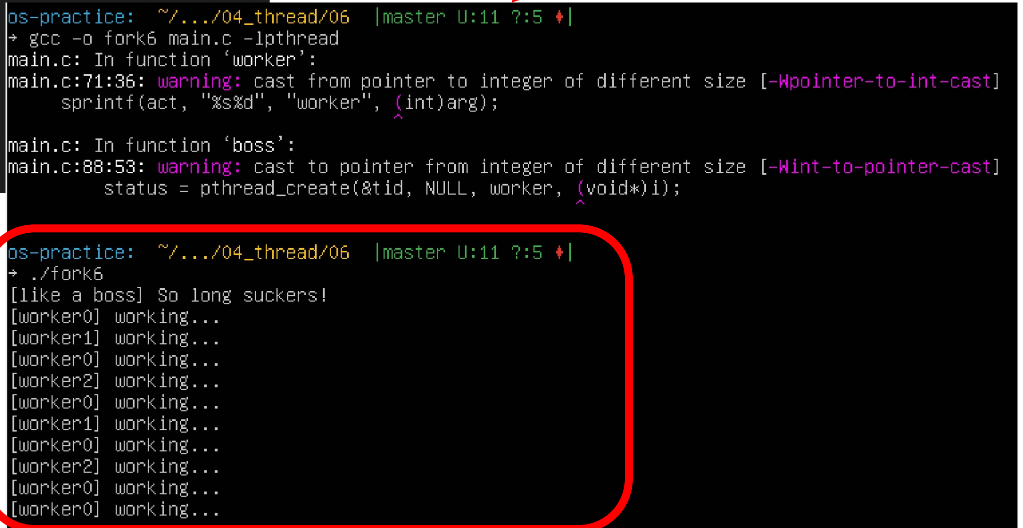


```c
#include <stdio.h>
#include <stdatomic.h>
#include <unistd.h>
#include <pthread.h>
#include <sys/wait.h>

#define NUM_WORKERS 3
#define NUM_PERSONAL_TASK 3
#define NUM_TOTAL_TASK (NUM_WORKERS * NUM_PERSONAL_TASK)

static _Atomic int cnt_task = NUM_TOTAL_TASK;

void do_job(char* actor);
void go_home(char* actor);
void* worker(void* arg);
void* boss(void* arg);

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

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

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

    pthread_join(tid, NULL);

    // OBJECT: The main thread should not be exited until all `worker`s have finished.
    //
    // HINT: The `main` thread cannot wait for `worker` threads detached by `boss`.
    // HINT: Is there any information about remaining tasks that can be
    //       referenced in the `main` thread?
    // Do not use "pthread_exit(NULL);"
    sleep(cnt_task);
    return 0;
}
```



```c
void do_job(char* actor){
    cnt_task--;
    printf("[%s] working...\n", actor); // 코드 흐름이 왔다갔다 하게 될 것. 어떤 스레드가 먼저 실행되는지 순서는 알 수 없다.

}

void go_home(char* actor){
    printf("[%s] So long suckers!\n", actor);
}

void* worker(void* arg)
{
    char act[20];
    sprintf(act, "%s%d", "worker", (int)arg);
    for(int i = 0; i < 3; i++)
    {
        sleep(1);
        do_job(act);
    }

    pthread_exit(NULL);
}

void* boss(void* arg)
{
    pthread_t tid;
    int status;
    for(int i = 0; i < NUM_WORKERS; i++)
    {
        status = pthread_create(&tid, NULL, worker, (void*)i); // worker의 매개변수 값은 i가 들어옴

        if (status != 0)
        {
            printf("WTF?");
            return (void*)-1;
        }

        pthread_detach(tid);
    }

    go_home("like a boss");
    pthread_exit(NULL);
}
```

- 이 문제에서는 worker 스레드가 작업을 다 수행하기 전까지 메인스레드가 종료를 기다려줘야하고, 이에 따라 sleep을 사용해 pthread_join이 수행하는 기능을 유사하게 따라할 수 있다고 볼 수 있습니다.
- 그래서 메인 스레드는 worker 스레드가 종료될 때까지 대기하게 되고, 이후에 모든 스레드 수행 이후 종료되는 것을 확인할 수 있습니다.
- (하단의 실행 결과를 통해 확인 가능합니다.)

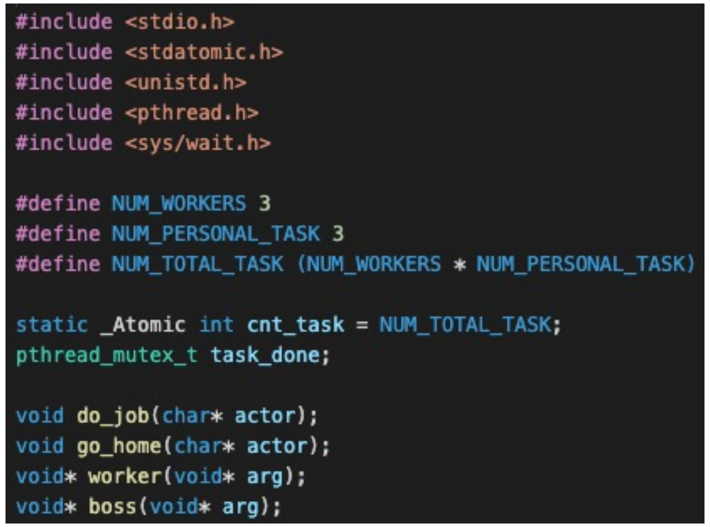


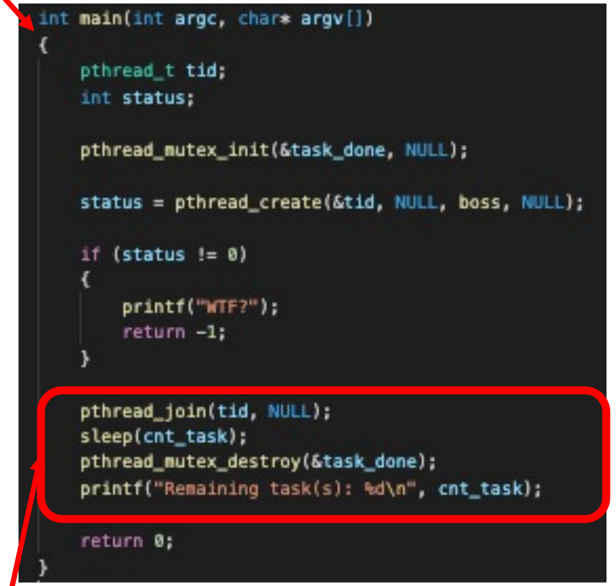
```
os-practice:  ~/.../04_thread/06  |master U:11 ?:5 ✦|
→ gcc -o fork6 main.c -lpthread
main.c: In function ‘worker’:
main.c:71:36: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
     sprintf(act, "%s%d", "worker", (int)arg);
                                    ^
main.c: In function ‘boss’:
main.c:88:53: warning: cast to pointer from integer of different size [-Wint-to-pointer-cast]
         status = pthread_create(&tid, NULL, worker, (void*)i);
                                                     ^

os-practice:  ~/.../04_thread/06  |master U:11 ?:5 ✦|
→ ./fork6
[like a boss] So long suckers!
[worker0] working...
[worker1] working...
[worker0] working...
[worker2] working...
[worker0] working...
[worker1] working...
[worker0] working...
[worker2] working...
[worker0] working...
[worker0] working...
```

- 스레드는 실행이 순차적으로 진행되지 않고, 실행이 겹치기도 하고, 실행의 순서를 예측할 수 없다고 실습 영상에서 설명이 포함되어 있습니다.
- 위 문제에서 “Boss” 스레스 실행에서 “Worker” 스레드는 detach 되어 실행 중이며, 과제 목표는 메인 스레드가 worker 스레드 종료 전에 먼저 종료되면 안되는 상황입니다.

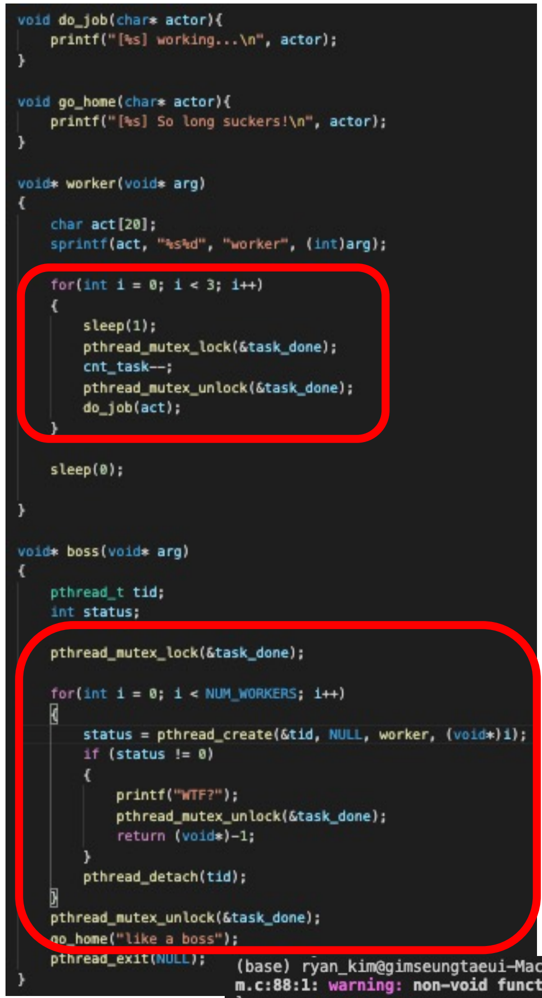
07번 문제 코드 & 실행 결과 (흐름의 시각화 내용은 다음 장에 있습니다)

Mutex 실행을 위해 Mutex_init 실행


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

#define NUM_WORKERS 3
#define NUM_PERSONAL_TASK 3
#define NUM_TOTAL_TASK (NUM_WORKERS * NUM_PERSONAL_TASK)

static _Atomic int cnt_task = NUM_TOTAL_TASK;
pthread_mutex_t task_done;

void do_job(char* actor);
void go_home(char* actor);
void* worker(void* arg);
void* boss(void* arg);
```


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

    pthread_mutex_init(&task_done, NULL);

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

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

    pthread_join(tid, NULL);
    sleep(cnt_task);
    pthread_mutex_destroy(&task_done);
    printf("Remaining task(s): %d\n", cnt_task);

    return 0;
}
```

6번 코드와 동일한 부분
(메인스레드는 worker 종료 전에 먼저 종료되면 안됨)
pthread_mutex_destroy(&task_done) 추가

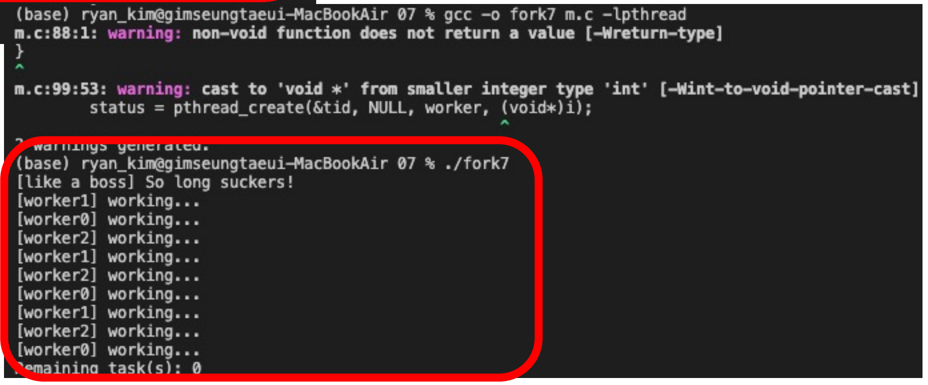

```c
void do_job(char* actor){
    printf("[%s] working...\n", actor);
}

void go_home(char* actor){
    printf("[%s] So long suckers!\n", actor);
}

void* worker(void* arg)
{
    char act[20];
    sprintf(act, "%s%d", "worker", (int)arg);

    for(int i = 0; i < 3; i++)
    {
        sleep(1);
        pthread_mutex_lock(&task_done);
        cnt_task--;
        pthread_mutex_unlock(&task_done);
        do_job(act);
    }

    sleep(0);

}

void* boss(void* arg)
{
    pthread_t tid;
    int status;

    pthread_mutex_lock(&task_done);

    for(int i = 0; i < NUM_WORKERS; i++)
    {
        status = pthread_create(&tid, NULL, worker, (void*)i);
        if (status != 0)
        {
            printf("WTF?");
            pthread_mutex_unlock(&task_done);
            return (void*)-1;
        }
        pthread_detach(tid);
    }
    pthread_mutex_unlock(&task_done);
    go_home("like a boss");
    pthread_exit(NULL);
}
```

6번 내용과 차이가 있다면, worker & boss 스레드에서mutex관련 메소드를 사용하는 부분인데요.

해당 부분들에서 CS로 진입할 cnt_task 값을 각 스레드가 실행되고 있는 기간 동안 서로 접근하는 것을 막고, 공유 자원을 업데이트 하기 위해 각 스레드에 lock & unlock을 사용해 CS를 보장하고 있는 것을 알 수 있습니다.

결과는 아래와 같이 실행된 화면을 볼 수 있는데요.

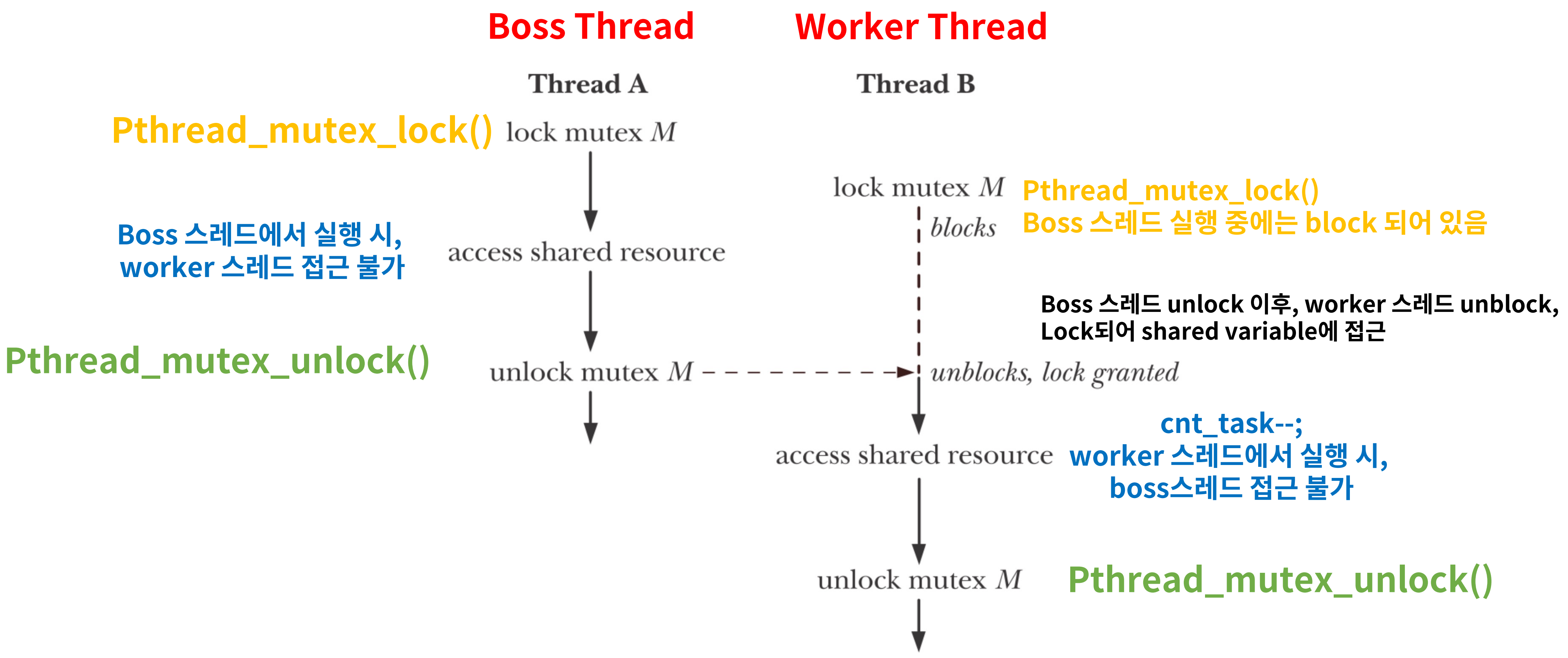
0,1,2 스레드가 서로 작업이 번갈아 가면서 다음 작업이 올라가는 형태로 변화가 있고, 최종적으로 남은 태스크가 0개로 만들어지는 것을 볼 수 있습니다.


```
(base) ryan_kim@gimseungtaeui-MacBookAir 07 % gcc -o fork7 m.c -lpthread
m.c:88:1: warning: non-void function does not return a value [-Wreturn-type]
}
^
m.c:99:53: warning: cast to 'void *' from smaller integer type 'int' [-Wint-to-void-pointer-cast]
        status = pthread_create(&tid, NULL, worker, (void*)i);
                                                     ^
  warnings generated.
(base) ryan_kim@gimseungtaeui-MacBookAir 07 % ./fork7
[like a boss] So long suckers!
[worker1] working...
[worker0] working...
[worker2] working...
[worker1] working...
[worker2] working...
[worker0] working...
[worker1] working...
[worker2] working...
[worker0] working...
emaining task(s): 0
```

- 6번 문제의 코드와 겹치는 부분이 많은 7번 코드입니다.
- 다만, 6번과는 다르게 문제 설명에도 나와 있지만, 적절하게 lock & unlock을 이용해서 하나의 스레드가 실행되는 동안 다른 스레드가 공유 자원에 접근하지 못하게 해야하며, 적절히 동기화 작업이 필요한 상황입니다.
- 조교님이 pthread_create & join을 사용하는 실습이 아니라고 언급했고, lock & unlock을 반드시 사용해야하는 상황인만큼, mutex_lock & unlock을 boss, woker 스레드에 적용했습니다.
- 다만, Critical section을 통제할 때는 lock & unlock을 명확하게 사용했지만, 메인스레드에서 최종적으로 worker 스레드 실행의 대기를 위해 sleep을 사용했습니다.
- 조교님 실습 영상을 리포트 제작 기간동안 8회 넘게 돌려봤는데, conditional variable을 사용안하고, pthread_create & join 없이 woker 스레드 대기를 하는 방법이 도저히 안떠올라 sleep을 사용했습니다.

07번 문제 해설 (코드 흐름도)

답안지 8장의 코드 실행의 흐름을 의사 코드로 대략적인 구조를 정리하면 아래와 같이 형성할 수 있습니다.



감사합니다

2013130890 영어영문학과 김승태