201600253 신세정

과제1) 차례 실행해보고 결과 확인

```
main[3392014144]: begin (counter = 0)
A[3392009984]: begin
A[3392009984]: done
main[3392014144]: done (counter = 562599) (ma1.val = 20000000)
u201600253@sejung-VirtualBox:~/Desktop/week06$ gcc -o mutex_20160
ead
u201600253@sejung-VirtualBox:~/Desktop/week06$ ./mutex_201600253
main[3867723584]: begin (counter = 0)
A[3867719424]: begin
A[3867719424]: done
main[3867723584]: done (counter = 5171805) (ma1.val = 20000000)
```

분석 그러한 결과가 나온 이유에 대해서 설명
예상한 counter 0이었다. 하지만 실행을 거듭할수록 counter값은 바뀌었다. 이유는 생성된 스레드들이 counter값에 동시 접근 했기 때문이다.

과제2) 29번 줄 pthread_join을 주석처리하면 어떻게 되는지 결과

```
main[3977660224]: begin (counter = 0)
A[3977656064]: begin
main[3977660224]: done (counter = 2574009) (ma1.val = 20000000)
u201600253@sejung-VirtualBox:~/Desktop/week06$ ./mutex_201600253
main[4175771456]: begin (counter = 0)
A[4175767296]: begin
main[4175771456]: done (counter = 1502378) (ma1.val = 19191027)
u201600253@sejung-VirtualBox:~/Desktop/week06$ ./mutex_201600253
main[3880580928]: begin (counter = 0)
A[3880576768]: begin
main[3880580928]: done (counter = -2029199) (ma1.val = 19446765)
u201600253@sejung-VirtualBox:~/Desktop/week06$ ./mutex_201600253
main[995333952]: begin (counter = 0)
A[995329792]: begin
main[995333952]: done (counter = 224315) (ma1.val = 19338495)
```

확인 및 분석 (수행 후 원상복귀)
Ma1.val의 값이 2000000이 계속 유지가 되어야 하는데, main에서 스레드A를 기다리지 않고 종료를 해버리기 때문에 A에서 val값이 증가되는 것이 완료되기 전 끝난다. 그래서 ma1.val값이 바뀌는 것이다.

과제3) 주어진 소스코드에 critical section이 있다. 어디 있는지 줄 번호를 모두 쓰고

```
 1 #include <stdio.h>
 2 #include <pthread.h>
 3
 4 static volatile int counter = 0;
 5 static const int end = 10000000;
 6
 7 typedef struct { char *id; int val; } myarg;
 8
 9 void *mythread(void *arg)
10 {
11     myarg *ma = (myarg *) arg;
12     printf("%s[%u]: begin\n", ma->id, (unsigned) pthread_self());
13     for(int i = 0; i < end; i++) {
14        --counter;
15        ma->val++;
16     }
17     printf("%s[%u]: done\n", ma->id, (unsigned) pthread_self());
18     return (void *) ma;
19 }
20
21 int main()
22 {
23     printf("main[%u]: begin (counter = %d)\n", (unsigned) pthread_self(), counter);
24     pthread_t t1, t2;
25     myarg ma1 = {"A", end };
26     pthread_create(&t1, NULL, mythread, &ma1);
27     for(int i = 0; i < end; i++)
28         counter++;
29 //    pthread_join(t1, (void **) &ma1);
30     printf("main[%u]: done (counter = %d) (ma1.val = %d)\n",
31             (unsigned) pthread_self(), counter, ma1.val);
32     return 0;
33 }
```

왜 그것들이 critical section인지 설명

13번~16번, 27번~28번. 스레드들이 동시에 접근할 수 있는 변수 counter와 이에 따라 변할 수 있는 변수val이 있는 부분이다. 이 부분은 동시에 접근했을 때 값이 변동되기 때문에 이 부분에 접근하는데 제한을 둬야 한다.

과제4) 뮤텍스(lock) 변수를 이용하여 critical section이 mutual exclusive하게 실행되도록 코드를 작성하고, 실행 결과 확인

```
main[762918720]: begin (counter = 0)
A[762914560]: begin
A[762914560]: done
main[762918720]: done (counter = 0) (ma1.val = 20000000)
```

이때 counter와 ma1.val의 최종 출력은 몇이어야 할까?

counter값은 0, ma1.val의 값은 20000000이 된다.

과제5) 추가된 코드

```c
static const int end = 10000000;

typedef struct { char *id; int val; } myarg;

pthread_mutex_t mutx; // 뮤텍스 선언

void *mythread(void *arg)
{
    myarg *ma = (myarg *) arg;
    printf("%s[%u]: begin\n", ma->id, (unsigned) pthread_self());
    pthread_mutex_lock(&mutx); // 다른 스레드 접근 제한
    for(int i = 0; i < end; i++) {
       --counter;
       ma->val++;
    }
    pthread_mutex_unlock(&mutx); // 다른 스레드 접근 허용
    printf("%s[%u]: done\n", ma->id, (unsigned) pthread_self());
    return (void *) ma;
}

int main()
{
    int state;

    state = pthread_mutex_init(&mutx, NULL); // mutex초기화 함수 호출

    printf("main[%u]: begin (counter = %d)\n", (unsigned) pthread_self(), counter);
    pthread_t t1, t2;
    myarg ma1 = {"A", end };
    myarg ma2 = {"B", end };

    pthread_create(&t1, NULL, mythread, &ma1);
    pthread_create(&t2, NULL, mythread, &ma2);

    pthread_mutex_lock(&mutx); // 다른 스레드 접근 제한
    for(int i = 0; i < end; i++)
        counter++;
    pthread_mutex_unlock(&mutx);

    pthread_join(t1, (void **) &ma1); // 다른 스레드 접근 허용
    pthread_join(t2, (void **) &ma2);

    printf("main[%u]: done (counter = %d) (ma1.val = %d)\n",
           (unsigned) pthread_self(), counter, ma1.val);
    pthread_mutex_destroy(&mutx); // 뮤텍스 삭제
    return 0;
}
```

counter의 최종 출력은 몇인가?

```
main[1976207168]: begin (counter = 0)
B[1967810304]: begin
A[1976203008]: begin
B[1967810304]: done
A[1976203008]: done
main[1976207168]: done (counter = -10000000) (ma1.val = 20000000)
```

느낀점

뮤텍스가 이론에서만 배웠을 때는 왜 그렇게 중요한지 잘 몰랐는데, 실습을 통해서 그 이유를 알 수 있었습니다. 제가 작성한 코드와 결과값이 정답인지는 잘 모르겠지만, 뮤텍스를 공부하는데 예제코드가 너무 적절해서 좋았습니다. 좋은 실습 경험하게 해주셔서 감사합니다.