운영체제 및 실습

분 반	00
학 과	컴퓨터공학과
학 번	201502049
이 름	노 효 근

1. 문제 해결 방법

1) 소스코드

```
//producer(void *arg) : 생산자 thread
void *producer(void *arg){
        int i;
        int id;
        int input;

        id = pc++;
        //생산자는 생산 횟수(6)만큼 생산을 하고 버퍼에 넣는다.
        for(i = 0; i<P_COUNT; i++){
                //버퍼에 넣을 임의의 값 생성
                input = random()%100;
                usleep(input);
                printf("producer %d add Q %d\n",id, input);

                /*********************  코드 작성 부분  *************************/
                /*  소비자가 버퍼에 접근하지 못하도록 임계영역으로 설정          */
                pthread_mutex_lock(&buffer_lock);
                /*  버퍼가 가득차면 대기(반드시 while문 사용, if문 사용하지 말 것)*/
                while(CQ_count == 10){
                        pthread_cond_wait(&buffers, &buffer_lock);
                }
                addQ(input);    /* input을 버퍼에 넣는다                        */
                /*              버퍼 안에 item이 있음을 소비자에게 알림         */
                pthread_cond_signal(&items);
                /*  삽입이 끝나면 임계영역을 빠져 나가므로 mutex 락을 해제       */
                pthread_mutex_unlock(&buffer_lock);
                /*****************************************************************/
```

먼저 producer 부분에서 소비자가 버퍼에 접근하지 못하도록 critical section을 설정하는데 mutex를 잠그는 함수인 pthread_mutex_lock(&buffer_lock)를 사용하였다.

pthread_mutex_lock() 함수의 인자로는 mutex 변수를 써야되는데 위에 buffer_lock을 생산 mutex 변수로 초기화 해 놨다. 버퍼가 가득차면 대기해아하므로 버퍼를 나타내는 CQ.count의 값이 10인 경우 꽉 찼다고 생각하여 반복문을 통하여 CQ.count == 10인 동안 pthread_cond_wait(&buffers, &buffer_lock) 함수를 사용하여 조건변수로 signal이 오는것 을 기다리고, wait 상태가 되면서 lock 걸어 놓은 mutex를 잠금 해제, signal을 받아 Thread가 깨어나면 mutex를 다시 잠금하도록 한다. 여기서 buffers는 생산 버퍼 조건 변수를 나타낸다.
버퍼가 가득 차 있지 않은 경우엔 input을 버퍼에 넣고, 버퍼 안에 item이 있음을 소비자에 게 알리기 위해 pthread_cond_signal(&items) 함수를 사용한다. 이는 조건변수에 signal을 보내 기다리고 있는 Thread를 깨우도록 한다. 삽입이 끝나면 임계영역을 빠져나가므로 pthread_mutex_unloack(&buffer_lock)를 통해 mutex 락을 해제하면 된다.

```
//consumer(void *arg) : 소비자 thread
void *consumer(void *arg){
        int i;
        int id;
        int output;
        id = cc++;

        for(i = 0; i<C_COUNT; i++){
                usleep(random()%100);

                /*********************  코드 작성 부분  ************************/
                /*  생산자가 버퍼에 접근하지 못하도록 임계영역으로 설정          */
                pthread_mutex_lock(&buffer_lock);
                /*버퍼가 비어있으면 대기(반드시 while문 사용, if문 사용하지 말 것)*/
                while(CQ_count == 0){
                        pthread_cond_wait(&items, &buffer_lock);
                }
                output=getQ();          /* item을 버퍼로부터 가져온다             */
                /*        버퍼 안에 item을 소비했다고 생산자에게 알림               */
                pthread_cond_signal(&buffers);
                /*     소비가 끝나면 임계영역을 빠져 나가므로 mutex 락을 해제     */
                pthread_mutex_unlock(&buffer_lock);
                /*****************************************************************/

                if(output != -1)
                        printf("consumer %d get Q %d\n",id, output);
        }
}
```

consumer도 producer처럼 소비자가 버퍼에 접근하지 못하도록 critical section을 설정하는 데 mutex를 잠그는 함수인 pthread_mutex_lock(&buffer_lock)를 사용하였다. 그리고 버퍼 가 비어있으면 대기해아하므로 버퍼를 나타내는 CQ.count의 값이 0인 경우 비어있다고 생각하여 반복문을 통하여 CQ.count == 0인 동안 pthread_cond_wait(&items, &buffer_lock) 함수를 사용하여 조건변수로 signal이 오는 것을 기다리고, wait 상태가 되면서 lock 걸어 놓은 mutex를 잠금 해제, signal을 받아 Thread가 깨어나면 mutex를 다시 잠금하도록 한 다. 여기서 items는 소비 조건변수를 나타낸다. 버퍼가 0이 아닌 경우에 input을 버퍼로부터 가져오고, 버퍼 안에 item을 소비했다고 생산 자에게 알리기 위해 pthread_cond_signal(&buffers) 함수를 사용한다. 이는 조건변수에 signal을 보내 기다리고 있는 Thread를 깨우도록 한다. 삽입이 끝나면 임계영역을 빠져나가므로 pthread_mutex_unloack(&buffer_lock)를 통해 mutex 락을 해제하면 된다.

2) 실행화면

```
u201502049@u201502049:~/Desktop/OS$ ./pc
producer 1 add Q 77
consumer 1 get Q 77
producer 2 add Q 15
consumer 2 get Q 15
producer 3 add Q 93
producer 4 add Q 35
producer 5 add Q 86
consumer 2 get Q 93
consumer 1 get Q 35
producer 1 add Q 92
producer 2 add Q 21
producer 3 add Q 27
consumer 2 get Q 86
consumer 1 get Q 92
consumer 2 get Q 21
producer 1 add Q 40
producer 2 add Q 26
producer 3 add Q 72
producer 4 add Q 90
producer 5 add Q 59
consumer 1 get Q 27
consumer 2 get Q 40
producer 1 add Q 67
producer 2 add Q 29
producer 3 add Q 82
producer 4 add Q 30
consumer 1 get Q 26
consumer 2 get Q 72
producer 1 add Q 35
producer 2 add Q 29
producer 3 add Q 2
producer 5 add Q 62
consumer 1 get Q 90
consumer 2 get Q 59
producer 1 add Q 67
producer 4 add Q 22
producer 5 add Q 11
```