운영체제 과제 - multithreaded program, synchronization

1. producer-consumer 프로그램은 다음과 같이 동작한다.

- producer와 consumer는 각각 thread로서 동작한다.
- producer와 consumer는 shared bounded buffer를 통해서 메시지를 주고받는다.
- producer는 임의의 시간 지연 후 메시지를 보내는 동작을 반복한다.
- consumer는 임의의 시간 지연 후 메시지를 받는 동작을 반복한다.
- 지연 시간은 난수발생 함수를 사용하여 0과 MAXDELAY 사이의 시간을 얻어서 사용하며 시간 지연은 usleep 함수를 이용한다.

(1) 위와 같이 동작하는 프로그램 아래에(prod-cons.c) 주어졌다. 이 프로그램의 동작을 설명하고 동작을 시켜보시오. (컴파일은 cc -o prod-cons prod-cons.c -lpthread 명령어를 사용한다.)
(2) multi-threaded 프로그래밍에 사용된 주요 pthread API에 대해서 간단히 설명하시오.

2. (synthronization 프로그래밍) 1번 프로그램은 공유자료 접근에 대한 동기화가 적용되지 않았다. 이 프로그램을 다음과 같이 동기화를 적용한 프로그램으로 수정하여 실행시키시오.
(1) Pthread 라이브러리의 semaphore 사용
(2) Pthread 라이브러리의 mutex와 condition variable 사용
제출물:
- 각 라이브러리 사용 방법을 포함한 프로그램 구성에 대한 설명
- 프로그램 소스코드 (코멘트 포함) 및 실행 결과
- 논의: 각 방법의 비교 검토 및 기타 논의 사항

참고자료
- linux programming의 thread에 대해서 제공한 자료

```c
/* prod-cons.c
 *  code without process synchronization
 */

#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define BUFSIZE 2
#define MAXDELAY 3000000    // usec (3 sec)

#define rand2()  ((double)rand() / RAND_MAX)

struct buffer {
        int buf[BUFSIZE];       // buffer
        int in;         // next free position
        int out;        // next full position
        int count;      // number of items in the buffer
};

void initbuf(struct buffer *p);
void *producer(void *arg);
void *consumer(void *arg);
void putmsg(struct buffer *p, int msg);
int getmsg(struct buffer *p);

int main()
{
        struct buffer bbuf;
        pthread_t tid1, tid2;

        initbuf(&bbuf);
        pthread_create(&tid1, NULL, producer, &bbuf);
        pthread_create(&tid2, NULL, consumer, &bbuf);
        pthread_join(tid1, NULL);
        pthread_join(tid2, NULL);
        return 0;
}

void initbuf(struct buffer *p)
{
        p->in = p->out = p->count = 0;
}

void *producer(void *arg)
{
        struct buffer *p = (struct buffer *)arg;
        int msg=0;

        while (1) {
                usleep(rand2()*MAXDELAY);
                msg++;
                printf("Producer inserts message %d\n" , msg);
                putmsg(p,msg);
    }
}
```

```
58
59  void *consumer(void *arg)
60  {
61          struct buffer *p = (struct buffer *)arg;
62          int msg;
63
64          while (1) {
65                  usleep(rand2()*MAXDELAY);
66                  msg = getmsg(p);
67                  printf("Consumer removes message %d\n", msg);
68          }
69  }
70
71  void putmsg(struct buffer *p, int msg)
72  {
73          while (p->count == BUFSIZE)
74                  ;
75          p->count++;
76          p->buf[p->in] = msg;
77          p->in = (p->in + 1) % BUFSIZE;
78
79          printf("putmsg: buffer size = %d\n", p->count);
80  }
81
82  int getmsg(struct buffer *p)
83  {
84          int msg;
85
86          while (p->count == 0)
87                  ;
88          p->count--;
89          msg = p->buf[p->out];
90          p->out = (p->out + 1) % BUFSIZE;
91
92          printf(" > getmsg: buffer size = %d\n", p->count);
93
94          return msg;
95  }
```