INHA UNIVERSITY

Operating System

homework#2

과 목 명 : operating system
담당교수 : 송민석 교수님
제 출 일 : 2017년 05월 14일

인하대학교 공과대학
12131597 정민교

목차

1. producer monitor thread

1. producer thread에서 item을 생성
2. 생성된 아이템을 확인하기 위해서 producer monitor thread에서 확인함.
3 .mutex2를 이용하여 producer thread를 lock.
3-1. 만약 item의 값이 50을 넘을 경우 reject produce item message를 출력하고 mutex2를 이용해 producer thread를 unlock함.
3-2.만약 item 값이 50을 넘지 않을 경우, mutex2를 이용해 producer thread를 unlock함.
4. item 값이 50을 넘을 경우, 1번에서부터 다시 수행.
5. item 값이 50을 넘지 않을 경우, buffer에 item을 add 한 후 item 생성 message를 출력함.

```
void *producer(void *param)
{
        srand(time(NULL));
   while(TRUE)
   {
      int rNum = rand() / RAND_DIVISOR;
      sleep(rNum);

      while(TRUE)
      {
        pthread_mutex_lock(&mutex2);
        if(check==0)
        {
          pthread_mutex_unlock(&mutex2);
          break;
        }
        item = (rand()%100)+1;
        check1 =0;
        pthread_mutex_unlock(&mutex2);
        while(check1==0);
      }

      check=1;
      sem_wait(&empty);
      pthread_mutex_lock(&mutex);
      if(insert_item(item))
      {
         printf("report error condition\n");
      }
      else
      {
         printf("producer produced %d\n", item);
      }
      pthread_mutex_unlock(&mutex);
      sem_post(&full);

   }
}
```

①

```
void *producermonitor(void *param)
{

  while(TRUE)
  {
      while(check1);
      pthread_mutex_lock(&mutex2);
      if(item > 50)
      {
        printf("-------reject produce item : %d--------\n",item);
        check1 =1;
        pthread_mutex_unlock(&mutex2);
      }
      else
      {
        check=0;
        check1 =1;
        pthread_mutex_unlock(&mutex2);
      }
  }

}
```

②

②item이 생성될때까지 wait
①item이 생성되면 check1을 0으로 바꾸고, unlock.
monitor thread에 lock을 한 후, item을 확인, 확인 후, check1값을 1로 바꾸는데 producer가 다시 item을 생성하지 못하게 lock.
monitor thread가 끝나면 unlock, item 값을 확인

2. consumer monitor thread

1. consumer thread에서 item을 consume할 때, consumer monitor thread에서 먼저 확인함.
2. mutex3을 이용하여 consumer thread를 lock.

2-1. consume할려는 item이 25를 넘을 경우, reject consume item message를 출력하고, item 값을 2로 나눔. mutex3을 이용하여 unlock함.

2-2. consume할려는 item이 25를 넘지 않을 경우, mutex3을 이용해 unlock함.

3. buffer에서 item을 consume한 후, item consume message를 출력함.

```
void *consumer(void *param)
{
   while(TRUE)
   {
      int rNum = rand() / RAND_DIVISOR;
      sleep(rNum);
      sem_wait(&full);
      pthread_mutex_lock(&mutex);
      while(TRUE)
      {
        if(second==1)
          break;

        first = 0;                                  ②
        while(first==0);
        pthread_mutex_lock(&mutex3);
      }

      second = 0;


      if(remove_item(&item))
      {
         printf("Consumer report error condition\n");
      }
      else
      {
         printf("consumer consumed %d\n", item);
      }

      pthread_mutex_unlock(&mutex);
      sem_post(&empty);
      pthread_mutex_unlock(&mutex3);
    }
}
```

```
void *consumermonitor(void *param)
{
   while(TRUE)
      {                                             ①
        while(first);
        pthread_mutex_lock(&mutex3);
        int con_item = buffer[(counter-1)];
        if(con_item > 25)
        {
          printf("--------reject consume item : %d ---------\n",con_item);
          con_item = con_item/2;
          buffer[(counter-1)] = con_item;
          first=1;
          second =1;
          pthread_mutex_unlock(&mutex3);
        }
        else
        {
          first = 1;
          second =1;
          pthread_mutex_unlock(&mutex3);
        }
      }

}
```

①consumer thread가 consume할려고 할 때까지 wait.

②consume을 할려고 하면, first값을 0으로 바꿔주고 monitor thread를 wait. monitor thread에 lock을 한 후, item을 확인, 확인 후, consume, consume 하는데 monitor가 또 동작하는 것을 방지하기 위해 lock.

3. monitor thread 동작 검증