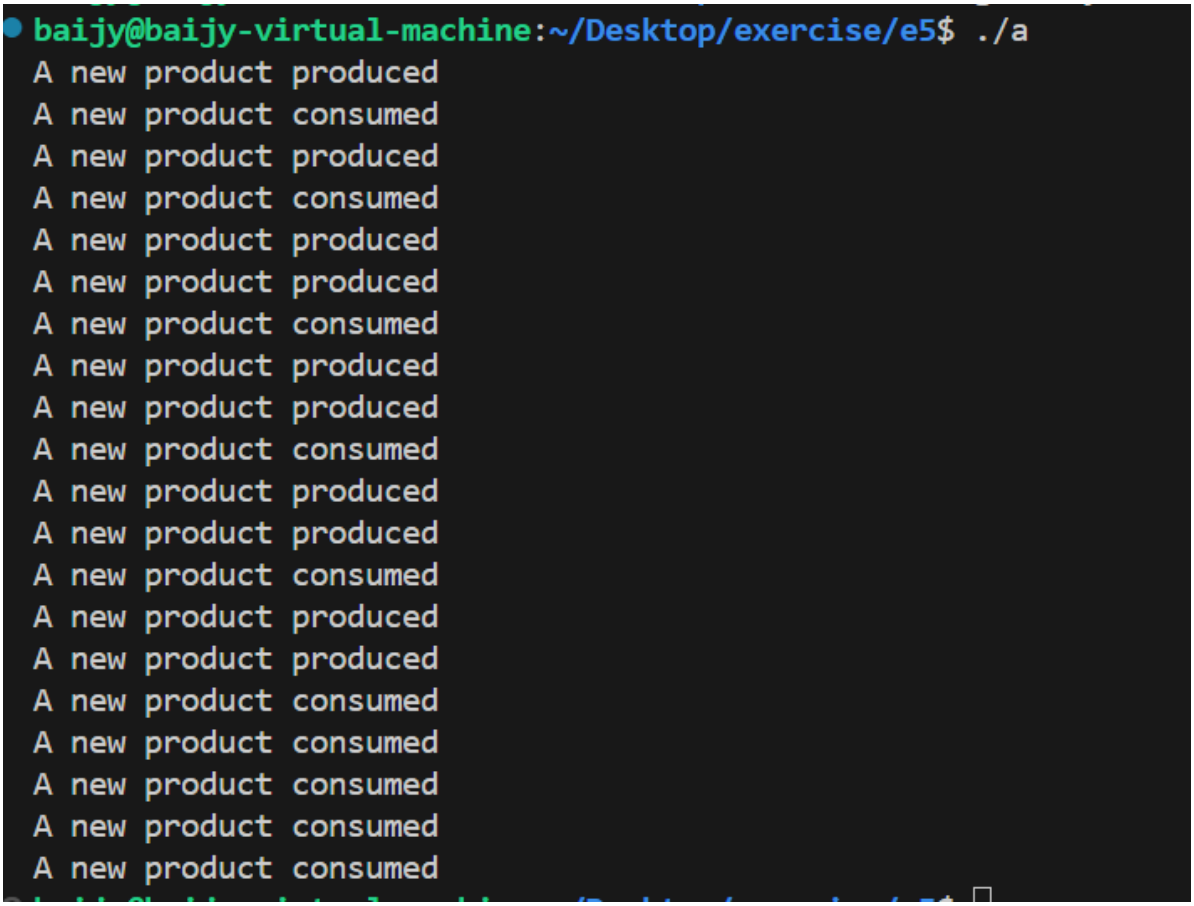
实验报告

为了实现简单的单个消费者和单生产者问题，我们定义一个全局的信号量，来代表目前的产品数量。

随后分别创建消费者函数和生产者函数，并建立线程，在线程结束后回收，再摧毁信号量，结束程序。

一个简单的运行结果：

为了突出差别，生产者线程每个循环会sleep 1s，而消费者是2s。



代码：

```
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
#include<semaphore.h>
#include<unistd.h>
#define NUM 4
sem_t product;

void* producer(void *arg){
    for(int i=0;i<10;i++){
        sem_post(&product);
        printf("A new product produced\n");
        sleep(1);
    }
    return NULL;
}
void * consumer(void *arg){
    for(int i=0;i<10;i++){
```

```
        sem_wait(&product);
        printf("A new product consumed\n");
        sleep(2);
    }
    return NULL;
}
int main(){
    pthread_t pro_thread, cos_thread;
    if(sem_init(&product,0,0)){
        printf("fail to init\n");
    }

 if(pthread_create(&pro_thread,NULL,producer,NULL)||pthread_create(&cos_thread,N
ULL,consumer,NULL)){
        printf("fail to create threads\n");
    }
    if(pthread_join(pro_thread,NULL)||pthread_join(cos_thread,NULL)){
        printf("fail to destroy threads");
    }
    if(sem_destroy(&product)){
        printf("fail to destroy the semaphore");
    }
    return 0;
}
```

经过简单修改，可以测试2个消费者的情况：

```
baijy@baijy-virtual-machine:~/Desktop/exercise/e5$ ./a
A new product produced 124231962588736
A new product consumed 124231954196032
A new product produced 124231962588736
A new product consumed 124231811585600
A new product produced 124231962588736
A new product consumed 124231954196032
A new product produced 124231962588736
A new product consumed 124231811585600
A new product produced 124231962588736
A new product consumed 124231954196032
A new product produced 124231962588736
A new product consumed 124231811585600
A new product produced 124231962588736
A new product consumed 124231954196032
A new product produced 124231962588736
A new product consumed 124231811585600
A new product produced 124231962588736
A new product consumed 124231954196032
A new product produced 124231962588736
A new product consumed 124231811585600
```

可以看到，生产出来的产品被两个线程依次消费，符合预期。