Project 4: 生产者消费者问题

516030910259 刘欣鹏

1. 实验目的

通过实现对有限缓冲的生产者消费者问题的简单解决方案，加深对线程、线程间资源共享的理解。

2. 实验原理

2.1.缓冲区

从内部来说，缓冲区是一个元数据类型为buffer_item的固定大小的数组。而从实现上来说，这个数组可以按环形队列处理。缓冲区通过insert_item()和remove_item()为生产者线程和消费者线程分别使用。

主函数main()接受三个命令行参数：睡眠多长之后才终止；生产者线程数量；消费者线程数量。它将缓冲初始化，创建生产者与消费者线程，睡眠一段时间，并在被唤醒时终止应用程序。

2.2.生产者与消费者线程

生产者线程不断交替执行如下两个阶段：睡眠一段随机时间，向缓冲插入一个随机数。消费者也睡眠随机时间，在醒后，从缓冲内取出一项。

2.3.Pthread互斥量与信号锁

互斥锁采用pthread_mutex_t数据类型。pthread_mutex_init(&mutex,NULL)创建互斥锁。pthread_mutex_lock()和pthread_mutex_unlock()用来获取与释放锁。

本项目使用无名称信号量sem_t数据类型。函数sem_init()用于创建并初始化一个信号量，有三个参数：信号量指针，表示共享级别的标记，信号量的初始值。经典信号量操作分别为sem_wait()和sem_post()。

3. 实验步骤

3.1.源代码procon.c

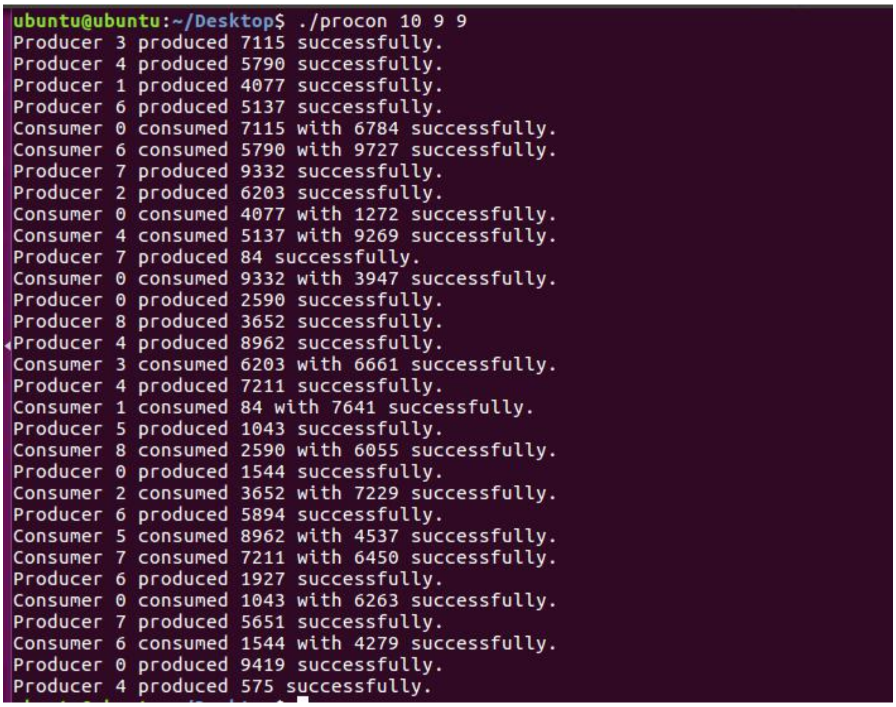
```
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>
#include <string.h>
#define BUFFER_SIZE 5
typedef int buffer_item;
buffer_item buffer[BUFFER_SIZE];
pthread_mutex_t mutex;
sem_t empty,full;
pthread_t pro_id[10],con_id[10];
pthread_attr_t pro_attr[10],con_attr[10];
int l,r;
int insert_item(buffer_item item){
sem_wait(&empty);
pthread_mutex_lock(&mutex);
r++;
if (r==5) r=0;
buffer[r]=item;
pthread_mutex_unlock(&mutex);
```

```
sem_post(&full);
return 0;
}
int remove_item(buffer_item item){
int x=0;
sem_wait(&full);
pthread_mutex_lock(&mutex);
x=buffer[l];
buffer[l]=item;
l++;
if (l==5) l=0;
pthread_mutex_unlock(&mutex);
sem_post(&empty);
return x;
}
void *producer(int *param){
buffer_item r;
while (1){
    r=rand()%10;
    sleep(r);
    r=rand()%10000;
    if (!insert_item(r))
        printf(″Producer %d produced %d successfully.\n″,*param,r);
}
}

void *consumer(int *param){
buffer_item r,x=0;
while (1){
    r=rand()%10;
    sleep(r);
    r=rand()%10000;
    if (x=remove_item(r))
        printf(″Consumer %d consumed %d with %d.\n″,*param,x,r);
}
}
int main(int argc, char *argv[]){
if (argc!=4) {
    printf(″Invalid arguments!\n″);
    return 0;
}
int x,y,z,i;
x=atoi(argv[1]);
y=atoi(argv[2]);
```

```
z=atoi(argv[3]);
if (x<=0 || y>10 || z>10 || y<=0 || z<=0) {
    printf(”Too many threads!\n”);
    return 0;
}
void *data=NULL;
pthread_mutex_init(&mutex,NULL);
sem_init(&empty,0,5);
sem_init(&full,0,0);
srand((unsigned)time(NULL));
l=0;r=-1;
for (i=0;i<y;i++) pthread_attr_init(&pro_attr[i]);
for (i=0;i<z;i++) pthread_attr_init(&con_attr[i]);
for (i=0;i<y;i++){
    int *data=(int *) malloc(sizeof(int));
    *data=i;
    pthread_create(&pro_id[i],&pro_attr[i],producer,data);
}
for (i=0;i<z;i++){
    int *data=(int *) malloc(sizeof(int));
    *data=i;
    pthread_create(&con_id[i],&con_attr[i],consumer,data);
}
sleep(x);
return 0;
}
```

3.2.测试结果



```
ubuntu@ubuntu:~/Desktop$ ./procon 10 9 9
Producer 3 produced 7115 successfully.
Producer 4 produced 5790 successfully.
Producer 1 produced 4077 successfully.
Producer 6 produced 5137 successfully.
Consumer 0 consumed 7115 with 6784 successfully.
Consumer 6 consumed 5790 with 9727 successfully.
Producer 7 produced 9332 successfully.
Producer 2 produced 6203 successfully.
Consumer 0 consumed 4077 with 1272 successfully.
Consumer 4 consumed 5137 with 9269 successfully.
Producer 7 produced 84 successfully.
Consumer 0 consumed 9332 with 3947 successfully.
Producer 0 produced 2590 successfully.
Producer 8 produced 3652 successfully.
Producer 4 produced 8962 successfully.
Consumer 3 consumed 6203 with 6661 successfully.
Producer 4 produced 7211 successfully.
Consumer 1 consumed 84 with 7641 successfully.
Producer 5 produced 1043 successfully.
Consumer 8 consumed 2590 with 6055 successfully.
Producer 0 produced 1544 successfully.
Consumer 2 consumed 3652 with 7229 successfully.
Producer 6 produced 5894 successfully.
Consumer 5 consumed 8962 with 4537 successfully.
Consumer 7 consumed 7211 with 6450 successfully.
Producer 6 produced 1927 successfully.
Consumer 0 consumed 1043 with 6263 successfully.
Producer 7 produced 5651 successfully.
Consumer 6 consumed 1544 with 4279 successfully.
Producer 0 produced 9419 successfully.
Producer 4 produced 575 successfully.
```

4. 心得体会

本实验加深了对线程及线程间资源共享的理解。

由于此前已接触过pthread库，本实验在实现上相对更加简单。