lab05Producer-consumer Problem

软件工程 2018 级 1813075 刘茵

Target

1. Write a c/c++ program
2. To implement the producer-consumer problem
3. Gcc

---

1) Install GCC Software Colletion

```
>> sudo apt-get install build-essential
```

2) How to use GCC

- gcc and make

3) posix thread

```
#include <pthread.h>
pthread_create()
```

4) write a c program to implement the producer-consumer problem, which has 5 producers and 4 consumers, and the size of shared pool is 3, a total of 15 products were produced and consumed, and all producers and consumers will exit. #公共缓存区为 3，product_num=15（total）

The specific requirements are as follows

- 每个产品的数据结构：至少包含产品编号、生产者编号、生产时间、缓冲区中存储编号、消费者编号；
- 生产和消费都需要随机时间(1~5s)；
- 不能产生竞态、不能死锁；
- 对公共缓冲区的操作：操作前、操作后要显示公共缓冲区中各个产品项的状态

- main.cpp 代码:

```
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#include <stdio.h>
#include <semaphore.h> // 信号量 sem_init、sem_wait、sem_post、sem_destroy
#include <sys/time.h>
#include <time.h>

#define INIT_NUM 0
#define TOTAL_NUM 3
// #define PRO_NUM 5
// #define CON_NUM 4
#define PRODUCTS 15

// int gettimeofday(struct timeval *tv,struct timezone *tz); 生产时间
struct product
{
    int id;             //产品编号
    int pro_id;  //生产者编号
    struct timeval tv; //生产时间
    int buf_id;         //缓冲区储存编号
    int con_id;  //消费者编号
};
void printobj(struct product *x)
{
    printf("产品编号:%d,生产者编号:%d,生产时间 %lu:%lu,缓冲区储存编号:%d,消费者编号:%d\n",
            x->id, x->pro_id, x->tv.tv_sec, x->tv.tv_usec, x->buf_id, x->con_id);
}
struct product *list[3] = {NULL, NULL, NULL};  //缓冲池
sem_t p_sem, c_sem, sh_sem;  //信号

int num = INIT_NUM;
int pronum = 0;
int connum = 0;


void *productor(void *args)
{
    int num = (int)(*(int *)args);
    while (true)
    {
```

```
42.         if (connum >= PRODUCTS && pronum >= PRODUCTS)
43.             exit(0);
44.         if (pronum >= PRODUCTS)
45.             break;
46.         pthread_t tid;
47.         // tid = pthread_self();
48.
49.         sem_wait(&p_sem); //可以生产信号减一,即剩余的容量
50.
51.         sleep(rand() % (5 - 1 + 1) + 1); //生产延时
52.
53.         struct timeval tt;
54.         gettimeofday(&tt, 0);
55.
56.         printf("时间:  %lu:%lu , 第 %d 个生产者尝试进入缓冲
    区!\n", tt.tv_sec, tt.tv_usec,num);
57.
58.         sem_wait(&sh_sem); //用来互斥的信号
59.
60.         gettimeofday(&tt, 0);
61.         printf("时间:  %lu:%lu , 第 %d 个生产者进入缓冲区域成
    功!\n", tt.tv_sec, tt.tv_usec, num);
62.
63.         printf("在生产前，各缓冲区产品的数据为：\n");
64.         for (int i = 0; i < 3; i++)
65.         {
66.             if (list[i] == NULL)
67.                 printf("第 %d 个缓冲区为空\n", i);
68.             else
69.             {
70.                 printf("第 %d 个缓冲区\n", i);
71.                 printobj(list[i]);
72.             }
73.         }
74.         struct product x;
75.         pronum++;
76.         x.id = pronum;
77.         x.pro_id = num;
78.         gettimeofday(&x.tv, 0);
79.         for (int j = 0; j < 3; j++)
80.         {
81.             if (list[j] == NULL)
82.             {
83.                 list[j] = &x;
```

```
84.                 x.buf_id = j;
85.                 break;
86.             }
87.         }
88.         printf("时间： %lu:%lu , 第%d 个产品 放入成
    功!\n", x.tv.tv_sec, x.tv.tv_usec, pronum);
89.
90.         printf("在生产后，各缓冲区产品的数据为：\n");
91.         for (int i = 0; i < 3; i++)
92.         {
93.             if (list[i] == NULL)
94.                 printf("第 %d 个缓冲区为空\n", i);
95.             else
96.             {
97.                 printf("第 %d 个缓冲区：\n", i);
98.                 printobj(list[i]);
99.             }
100.         }
101.
102.         sem_post(&sh_sem);
103.
104.         gettimeofday(&tt, 0);
105.         printf("时间： %lu:%lu , 第 %d 个生产者离开出缓冲区成
    功!\n", tt.tv_sec, tt.tv_usec, num);
106.
107.         sem_post(&c_sem); //消费者可以消费的信号量加一
108.     }
109.     return 0;
110. }
111.
112. void *consumer(void *args)
113. {
114.     int num = (int)(*(int *)args);
115.     while (1)
116.     {
117.         if (connum >= PRODUCTS && pronum >= PRODUCTS)
118.             exit(0);
119.         if (connum >= PRODUCTS)
120.             break;
121.         // pthread_t cid;
122.         // cid = pthread_self();
123.         sem_wait(&c_sem); //消费者消费信号量减一
124.
125.         struct timeval tt;
```

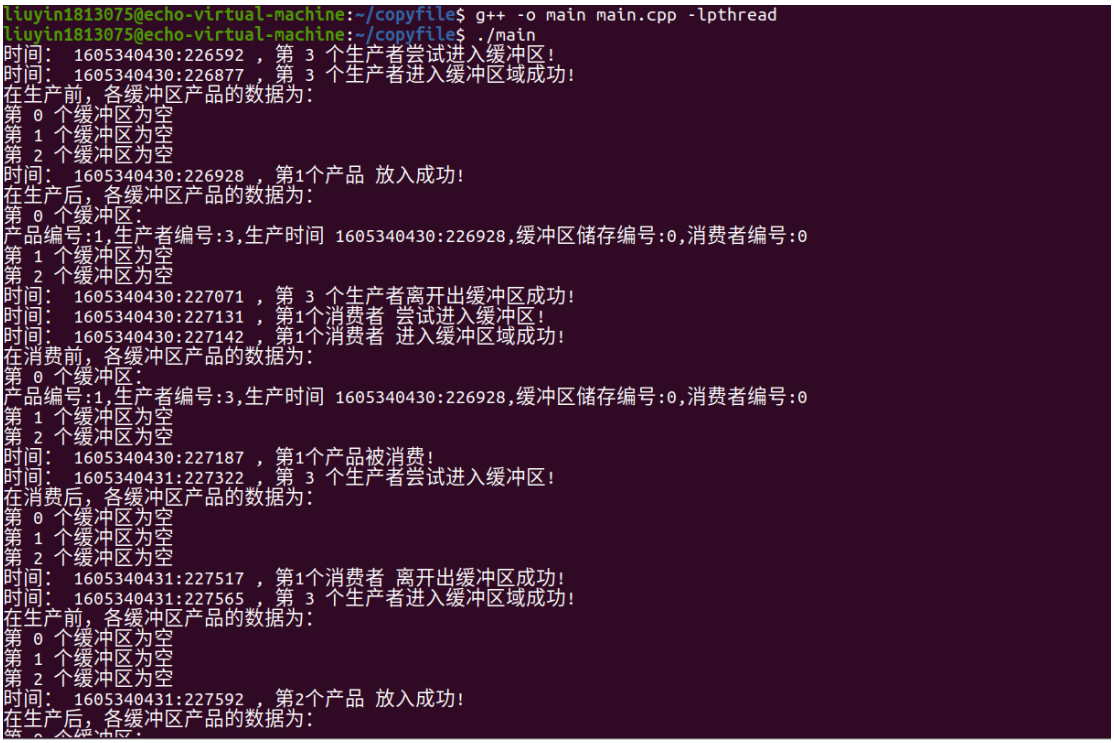
```
126.         gettimeofday(&tt, 0);
127.
128.         printf("时间： %lu:%lu ， 第%d 个消费者 尝试进入缓冲
     区!\n", tt.tv_sec, tt.tv_usec, num);
129.
130.         sem_wait(&sh_sem);
131.
132.         gettimeofday(&tt, 0);
133.         printf("时间： %lu:%lu ， 第%d 个消费者 进入缓冲区域成
     功!\n", tt.tv_sec, tt.tv_usec, num);
134.
135.         printf("在消费前，各缓冲区产品的数据为：\n");
136.         for (int i = 0; i < 3; i++)
137.         {
138.             if (list[i] == NULL)
139.                 printf("第 %d 个缓冲区为空\n", i);
140.             else
141.             {
142.                 printf("第 %d 个缓冲区：\n", i);
143.                 printobj(list[i]);
144.             }
145.         }
146.
147.         for (int i = 0; i < 3; i++)
148.         {
149.             if (list[i] != NULL)
150.             {
151.                 list[i] = NULL;
152.                 break;
153.             }
154.         }
155.
156.         gettimeofday(&tt, 0);
157.         connum++;
158.         printf("时间： %lu:%lu ， 第%d 个产品被消
     费!\n", tt.tv_sec, tt.tv_usec, connum);
159.         sleep(rand() % (5 - 1 + 1) + 1); //生产延时 //消费延时
160.
161.         printf("在消费后，各缓冲区产品的数据为：\n");
162.         for (int i = 0; i < 3; i++)
163.         {
164.             if (list[i] == NULL)
165.                 printf("第 %d 个缓冲区为空\n", i);
166.             else
```

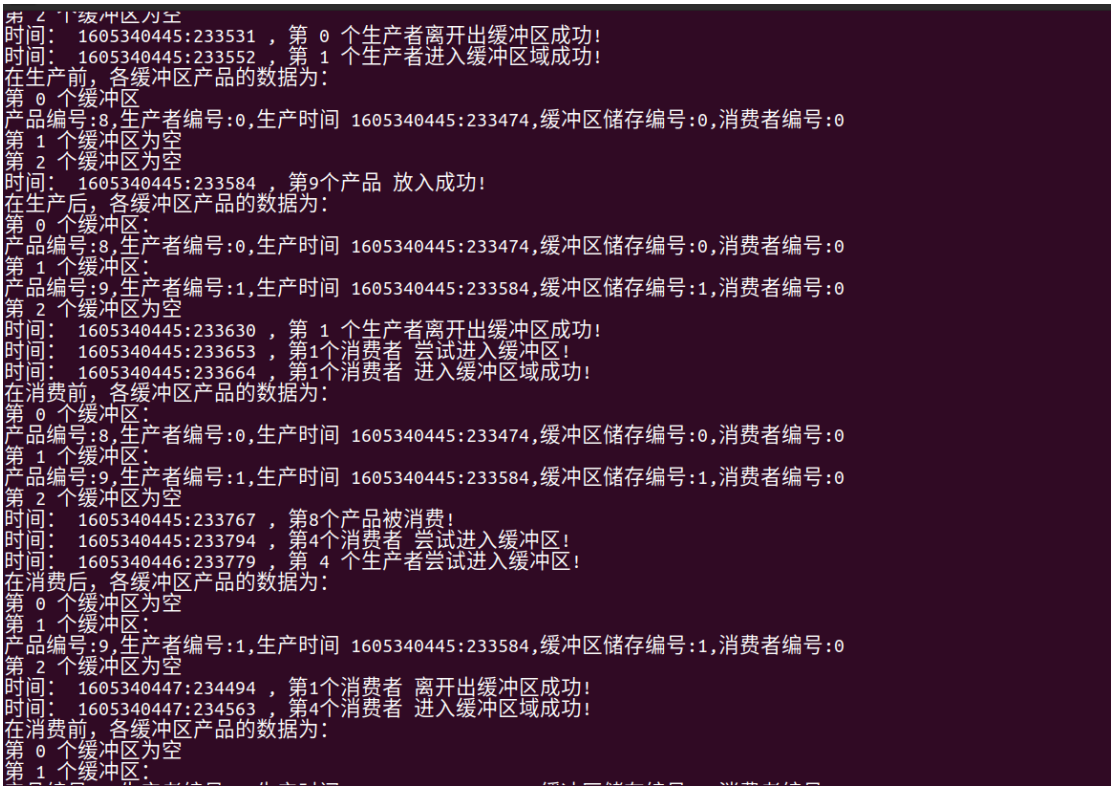
```
167.             {
168.                 printf("第 %d 个缓冲区：\n", i);
169.                 printobj(list[i]);
170.             }
171.         }
172.
173.         sem_post(&sh_sem);
174.
175.         gettimeofday(&tt, 0);
176.         printf("时间： %lu:%lu ，第%d 个消费者 离开出缓冲区成
    功!\n", tt.tv_sec, tt.tv_usec, num);
177.
178.         sem_post(&p_sem); //生产者生产信号量加一
179.     }
180.     return 0;
181. }
182.
183. int main(int argc, char *argv[])
184. {
185.     // pthread_t pid1, pid2, pid3, pid4, pid5;
186.     pthread_t pid[5];
187.     pthread_t cid[4];
188.
189.     sem_init(&p_sem, 0, TOTAL_NUM - INIT_NUM);    //设计为 3 个
190.     sem_init(&c_sem, 0, INIT_NUM);
191.     sem_init(&sh_sem, 0, 1);
192.     for(int i=0;i<5;i++){
193.         pthread_create(&pid[i], NULL, productor, &i);
194.     }
195.     for(int i=0;i<4;i++){
196.         pthread_create(&cid[i], NULL, consumer, &i);
197.     }
198.     // 指向线程标识符的指针；设置线程属性；线程运行函数的起始地址；运行函数的参
    数
199.     for(int i=0;i<5;i++){
200.         pthread_join(pid[i],NULL);
201.     }
202.     for(int i=0;i<4;i++){
203.         pthread_join(cid[i],NULL);
204.     }
205.     return 0;
206. }
```

- 运行结果

>g++ -o main main.cpp -lpthread

部分运行截图：





```
第 1 个缓冲区为空
第 2 个缓冲区为空
时间:  1605340463:240783 , 第14个产品 放入成功!
在生产后, 各缓冲区产品的数据为:
第 0 个缓冲区:
产品编号:14,生产者编号:1,生产时间 1605340463:240783,缓冲区储存编号:0,消费者编号:0
第 1 个缓冲区为空
第 2 个缓冲区为空
时间:  1605340463:241094 , 第 1 个生产者离开出缓冲区成功!
时间:  1605340463:241141 , 第 4 个生产者进入缓冲区域成功!
在生产前, 各缓冲区产品的数据为:
第 0 个缓冲区
产品编号:14,生产者编号:1,生产时间 1605340463:240783,缓冲区储存编号:0,消费者编号:0
第 1 个缓冲区为空
第 2 个缓冲区为空
时间:  1605340463:241357 , 第15个产品 放入成功!
在生产后, 各缓冲区产品的数据为:
第 0 个缓冲区:
产品编号:14,生产者编号:1,生产时间 1605340463:240783,缓冲区储存编号:0,消费者编号:0
第 1 个缓冲区:
产品编号:15,生产者编号:4,生产时间 1605340463:241357,缓冲区储存编号:1,消费者编号:0
第 2 个缓冲区为空
时间:  1605340463:241523 , 第 4 个生产者离开出缓冲区成功!
时间:  1605340463:241151 , 第3个消费者 尝试进入缓冲区!
时间:  1605340463:241588 , 第2个消费者 尝试进入缓冲区!
时间:  1605340463:241598 , 第2个消费者 进入缓冲区域成功!
在消费前, 各缓冲区产品的数据为:
第 0 个缓冲区:
产品编号:14,生产者编号:1,生产时间 1605340463:240783,缓冲区储存编号:0,消费者编号:0
第 1 个缓冲区:
产品编号:15,生产者编号:4,生产时间 1605340463:241357,缓冲区储存编号:1,消费者编号:0
第 2 个缓冲区为空
时间:  1605340463:241705 , 第14个产品被消费!
产品生产数已达15, 退出此次生产在消费后, 各缓冲区产品的数据为:
第 0 个缓冲区为空
第 1 个缓冲区:
```