操作系统实验 3　进程同步和通信

计算机 1202　张艺瀚
学号:20123852

June 23, 2015

1　实验题目

生产者和消费者问题模拟

2　实验目的

这是一个验证型实验。通过对给出的程序进行验证、修改，进一步加深理解进程的概念，了解同步和通信的过程，掌握进程通信和同步的机制，特别是利用缓冲区进行同步和通信的过程。通过补充新功能，加强对知识的灵活运用，培养创新能力。

3　实验要求

1. 调试、运行给出的程序，从操作系统原理的角度验证程序的正确性。
2. 发现并修改程序中的原理性错误或不完善的地方。
3. 鼓励在程序中增加新的功能。完成基本。
4. 在程序中适当地加入注释。
5. 认真进行预习，阅读原程序，发现其中的原理性错误，完成预习报告。
6. 实验完成后，要认真总结，完成实验报告。

4　程序流程图

见图 1-3

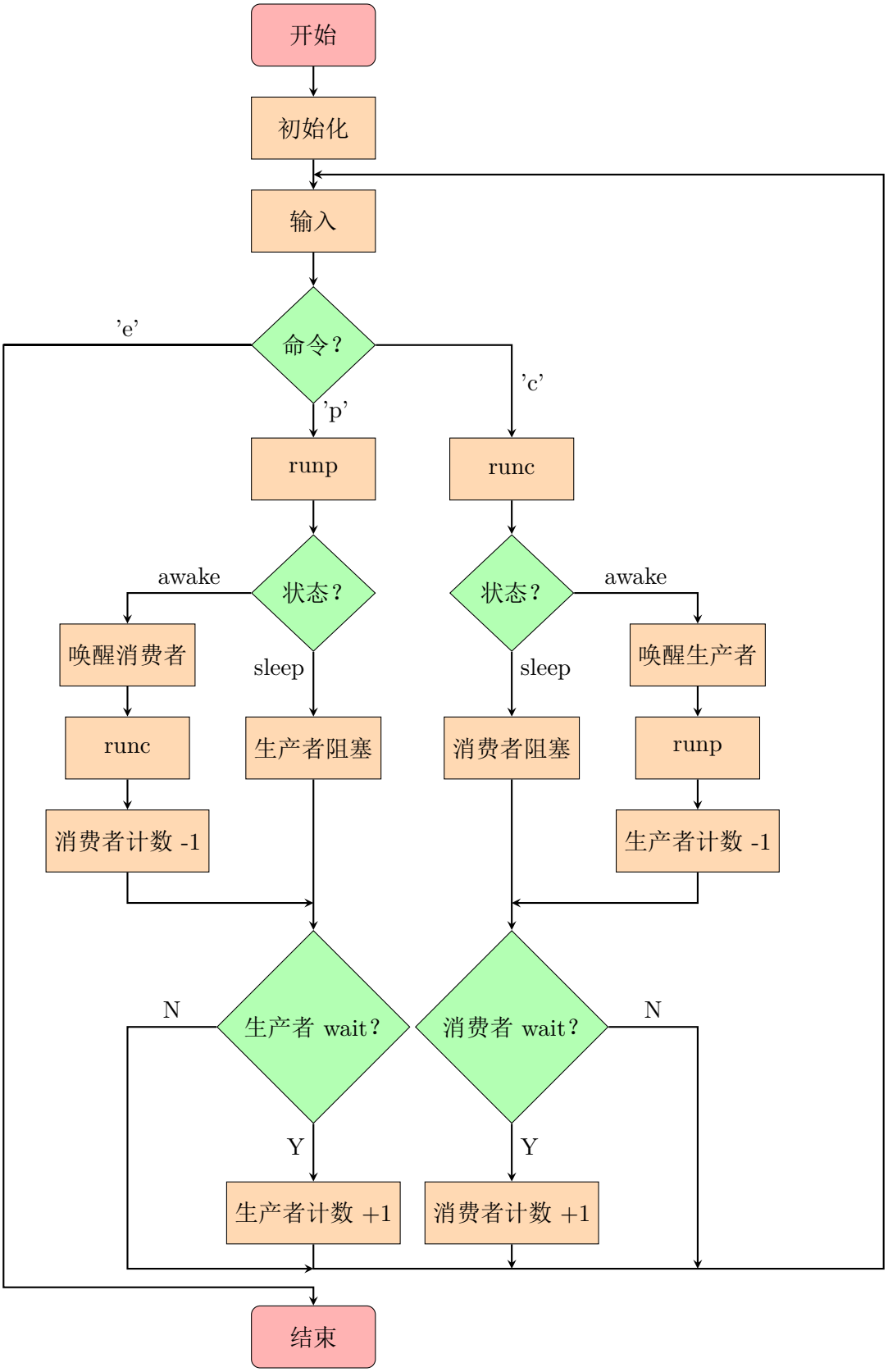

开始
初始化
输入
命令？
'e'
'c'
'p'
runp
runc
状态？
awake
状态？
awake
唤醒消费者
sleep
sleep
唤醒生产者
runc
生产者阻塞
消费者阻塞
runp
消费者计数 -1
生产者计数 -1
生产者 wait？
消费者 wait？
N
N
Y
Y
生产者计数 +1
消费者计数 +1
结束

Figure 1: 主过程

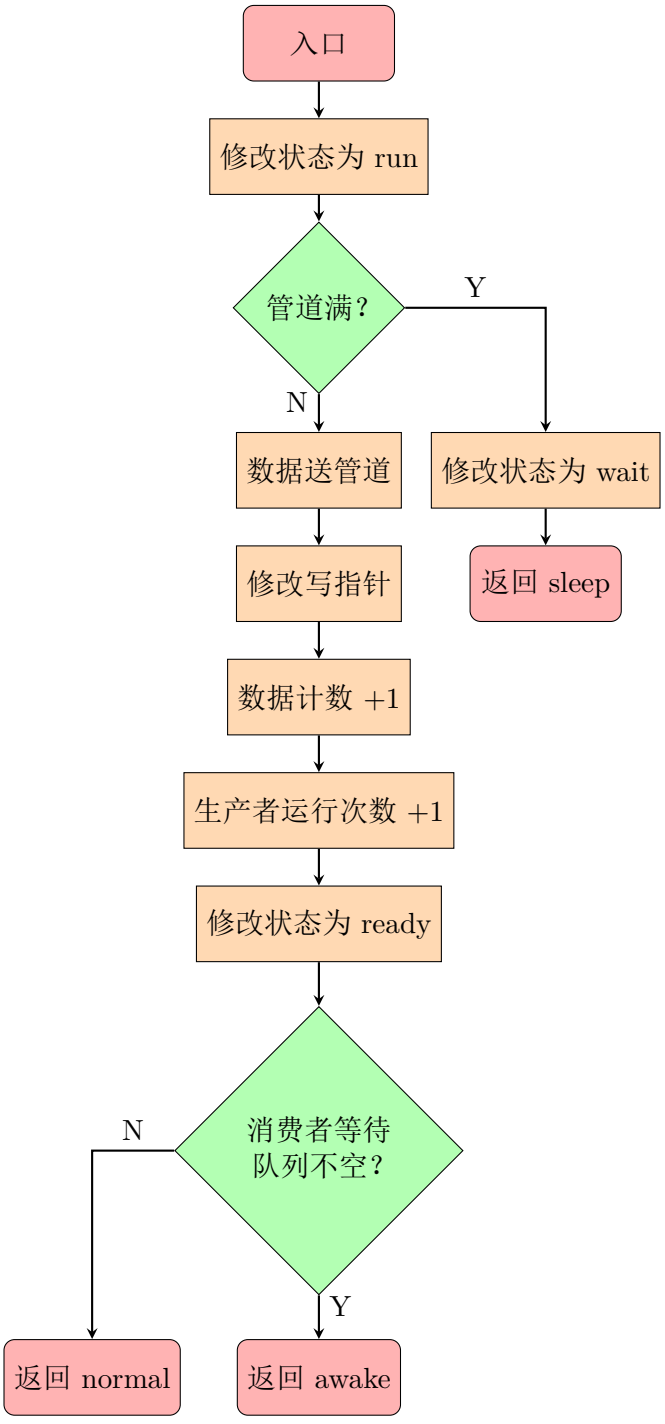


Figure 2: runp

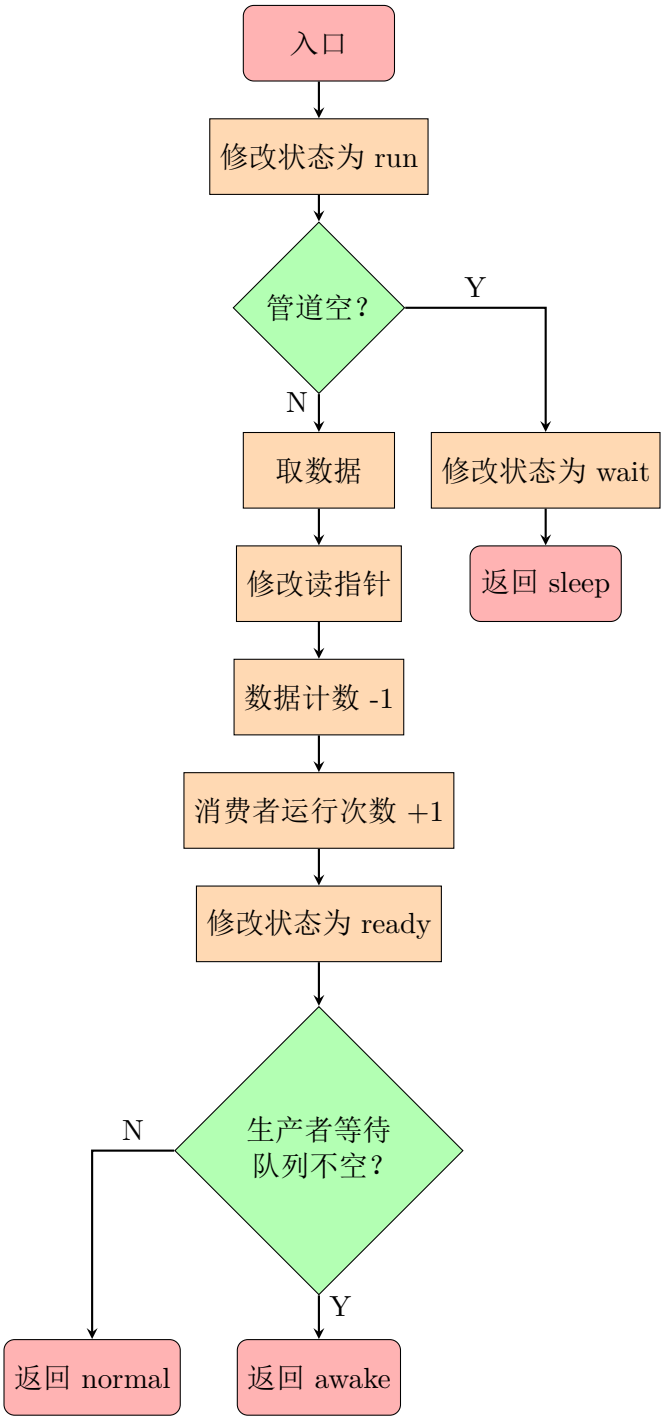


Figure 3: runc

5 源程序

增加生产者计数和消费者计数，作用相当与生产者等待队列和消费者等待队列。若有等待消费者，则生产的数据不会出现在队列中，知道满足所有等待消费者，讲述写入上次指针位置。

```
/*************************************************************/
/*         PROGRAM NAME:         PRODUCER_CONSUMER           */
/*    This program simulates two processes, producer which   */
/* continues  to produce message and  put it into a buffer   */
/* [implemented by PIPE], and consumer which continues to get*/
/* message from the buffer and use it.                       */
/*    The program also demonstrates the synchronism between  */
/* processes and uses of PIPE.                               */
/*************************************************************/

#define PIPESIZE 8

#define PRODUCER 0
#define CONSUMER 1

#define RUN      0    /* statu of process */
#define WAIT     1    /* statu of process */
#define READY    2    /* statu of process */

#define NORMAL   0
#define SLEEP    1
#define AWAKE    2

#include <stdio.h>
struct pcb
{
  char *name;
  int  statu;
  int  time;  /* times of execution */
};

struct pipetype
{
  char type;
  int  writeptr;
```

```
36    int  readptr;
37    struct pcb *pointp; /* write wait point */ // producer queue
38    struct pcb *pointc; /* read wait point  */ // consumer queue
39  };
40
41  int pipe[PIPESIZE];
42  struct pipetype pipetb;
43  struct pcb process[2];
44  int count=0;  // resource count
45  int producerCount=0, consumerCount=0; // semaphore
46
47  int runp(int out, struct pcb p[], int pipe[], struct pipetype *tb
        , int t)
48  {
49    p[t].statu=RUN;
50    printf("run PRODUCER. product %d     ",out);
51    if(count >= 8)  // full
52    {
53      p[t].statu=WAIT;
54      return(SLEEP);
55    }
56    pipe[tb->writeptr]=out;
57    tb->writeptr=++tb->writeptr%8;  // revised
58    // tb->writeptr++;
59    printf("writeptr%d\n",tb->writeptr);  // revised
60    count++;  // revised
61    printf("count = %d\n",count); // revised
62    p[t].time++;
63    printf("time = %d\n",p[t].time);  // revised
64    p[t].statu=READY;
65    if((tb->pointc) != NULL)
66    {
67      printf("return AWAKE");
68      return(AWAKE);
69    }
70    return(NORMAL);
71  }
72
73  int runc(struct pcb p[], int pipe[], struct pipetype *tb, int t)
74  {
75    int c;
```

```
76    p[t].statu = RUN;
77    printf("run CONSUMER. ");
78    if(count <= 0)  //
79    {
80      p[t].statu=WAIT;
81      return(SLEEP);
82    }
83    c=pipe[tb->readptr];
84    pipe[tb->readptr]=0; // revised
85    // tb->readptr++;
86    tb->readptr=++tb->readptr%8;  // revised
87    printf("readptr=%d\n",tb->readptr); // revised
88    printf(" use %d      ",c);
89    // delete some code
90    // if(tb->readptr >= tb->writeptr)
91    // {
92    //  tb->readptr=tb->writeptr=0;
93    // }
94    count--;// revised
95    printf("count = %d\n",count); // revised
96    p[t].time++;
97    printf("time = %d\n",p[t].time);  // revised
98    p[t].statu=READY;
99    if(tb->pointp != NULL)
100   {
101     printf("return AWAKE\n"); // revised
102     return(AWAKE);
103   }
104   return(NORMAL);
105 }
106
107 void prn(struct pcb p[], int pipe[], struct pipetype tb)
108 {
109   int i;
110   printf("\n          ");
111   for(i=0;i<PIPESIZE;i++)
112   {
113     printf("------ ");  // print upper border
114   }
115   printf("\n         |");  // left dash
116   for(i=0;i<PIPESIZE;i++) // second line and vertical bar
```

```
117   {
118     if(pipe[i] != 0)
119     {
120       printf("  %2d  |",pipe[i]);
121     }
122     else
123     {
124       printf("      |");
125     }
126   }
127   printf("\n          ");
128   for(i=0;i<PIPESIZE;i++) // print lower border
129   {
130     printf("------ ");
131   }
132   printf("\nwriteptr = %d, readptr = %d,  ",tb.writeptr,tb.
       readptr);
133   if(p[PRODUCER].statu == WAIT)
134   {
135     printf("PRODUCER wait ");
136   }
137   else
138   {
139     printf("PRODUCER ready ");
140   }
141   if(p[CONSUMER].statu == WAIT)
142   {
143     printf("CONSUMER wait ");
144   }
145   else
146   {
147     printf("CONSUMER ready ");
148   }
149   printf("\n");
150 }
151
152 int main(int argc, char** argv)
153 {
154   int output,ret,i;
155   char in[2];
156
```

```
157   pipetb.type = 'c';
158   pipetb.writeptr = 0;
159   pipetb.readptr = 0;
160   pipetb.pointp = pipetb.pointc = NULL;
161   process[PRODUCER].name = "Producer\0";
162   process[CONSUMER].name = "Consumer\0";
163   process[PRODUCER].statu = process[CONSUMER].statu = READY;
164   process[PRODUCER].time = process[CONSUMER].time = 0;
165   output = 0;
166   printf("Now starting the program!\n");
167   printf("Press 'p' to run PRODUCER, press 'c' to run CONSUMER.\n
       ");
168   printf("Press 'e' to exit from the program.\n");
169   for(i=0;i<1000;i++)
170   {
171     in[0]='N';
172     while(in[0] == 'N')
173     {
174       scanf("%s",in);
175       if(in[0] != 'e'&&in[0] != 'p'&&in[0] != 'c')  // if input
       is not e/p/c, input again
176       {
177         in[0]='N';
178       }
179     }
180     if(in[0] == 'e')
181     {
182       return 1; // exit(1);
183     }
184     if(in[0] == 'p'&&process[PRODUCER].statu == READY)
185     {
186       if(count<8) // revised
187       {
188         output = (output+1)%100;
189       }
190       if((ret=runp(output,process,pipe,&pipetb,PRODUCER)) ==
       SLEEP)
191       {
192         pipetb.pointp = &process[PRODUCER];
193       }
194       if(ret == AWAKE)
```

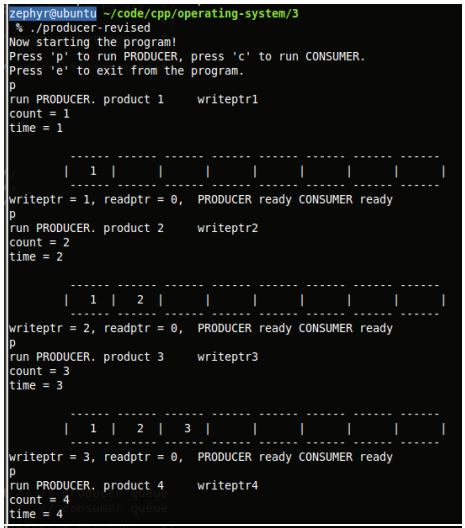
```
195         {
196           (pipetb.pointc)->statu=READY;
197           runc(process,pipe,&pipetb,CONSUMER);  // revised
198           consumerCount--;  // revised
199           printf("consumerCount=%d\n",consumerCount); // revised
200           if(consumerCount == 0)  // revised
201           { // revised
202             pipetb.pointc=NULL;// revised
203           } // revised
204         }
205       }
206       if(in[0] == 'c'&&process[CONSUMER].statu == READY)
207       {
208         if((ret=runc(process,pipe,&pipetb,CONSUMER)) == SLEEP)
209         {
210           pipetb.pointc = &process[CONSUMER];
211         }
212         if(ret == AWAKE)
213         {
214           (pipetb.pointp)->statu=READY;
215           output=(output+1)%100;  // revised
216           runp(output,process,pipe,&pipetb,PRODUCER);
217           producerCount--;  // revised
218           printf("producerCount=%d\n",producerCount); // revised
219           if(producerCount == 0)  // revised
220           { // revised
221             pipetb.pointp=NULL;// revised
222           } // revised
223         }
224       }
225       if(in[0] == 'p'&&process[PRODUCER].statu == WAIT)
226       {
227         producerCount++;  // revised
228         printf("producerCount=%d\n",producerCount); // revised
229         printf("PRODUCER is waiting, can't be scheduled.\n");
230       }
231       if(in[0] == 'c'&&process[CONSUMER].statu == WAIT)
232       {
233         consumerCount++;  // revised
234         printf("consumerCount=%d\n",consumerCount); // revised
235         printf("CONSUMER is waiting, can't be scheduled.\n");
```

```
236     }
237     prn(process,pipe,pipetb);
238     in[0]='N';
239   }
240   return 0;
241 }
```

Listing 1: 代码清单

6 运行结果及其说明

队列满后继续生产，生产者阻塞，且修改生产者计数；队列空后继续消费，消费者阻塞，且修改消费者计数。生产者生产数据后，若有等待的消费者，则唤醒，否则数据送管道。

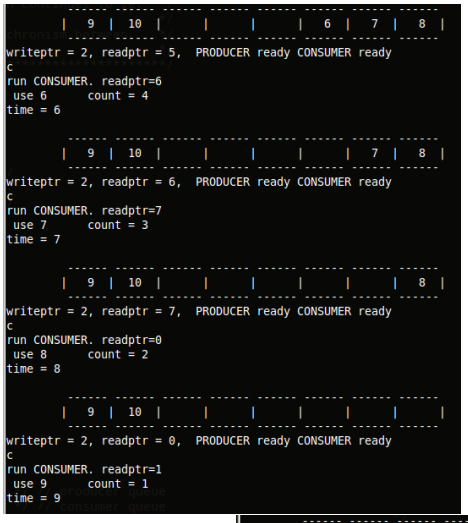


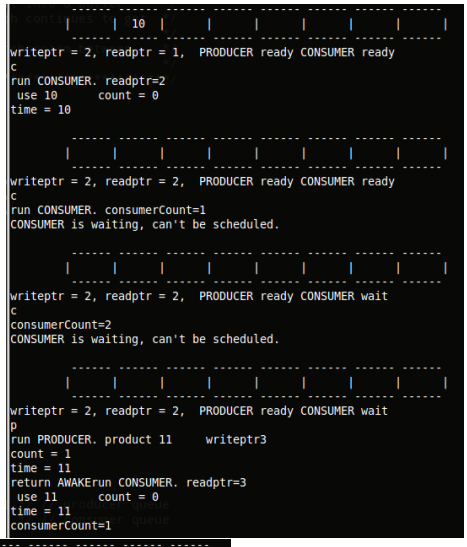
```
zephyr@ubuntu ~/code/cpp/operating-system/3
 % ./producer-revised
Now starting the program!
Press 'p' to run PRODUCER, press 'c' to run CONSUMER.
Press 'e' to exit from the program.
p
run PRODUCER. product 1     writeptr1
count = 1
time = 1

        ------ ------ ------ ------ ------ ------ ------ ------
       |   1  |      |      |      |      |      |      |      |
        ------ ------ ------ ------ ------ ------ ------ ------
writeptr = 1, readptr = 0,  PRODUCER ready CONSUMER ready
p
run PRODUCER. product 2     writeptr2
count = 2
time = 2

        ------ ------ ------ ------ ------ ------ ------ ------
       |   1  |   2  |      |      |      |      |      |      |
        ------ ------ ------ ------ ------ ------ ------ ------
writeptr = 2, readptr = 0,  PRODUCER ready CONSUMER ready
p
run PRODUCER. product 3     writeptr3
count = 3
time = 3

        ------ ------ ------ ------ ------ ------ ------ ------
       |   1  |   2  |   3  |      |      |      |      |      |
        ------ ------ ------ ------ ------ ------ ------ ------
writeptr = 3, readptr = 0,  PRODUCER ready CONSUMER ready
p
run PRODUCER. product 4     writeptr4
count = 4
time = 4
```

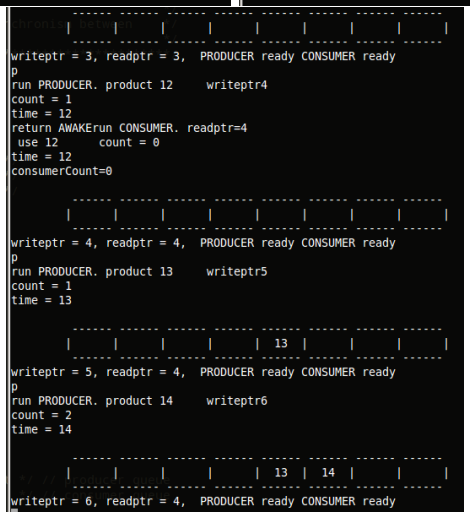
```
        ------ ------ ------ ------ ------ ------ ------ ------
       |   1  |   2  |   3  |   4  |      |      |      |      |
        ------ ------ ------ ------ ------ ------ ------ ------
writeptr = 4, readptr = 0,  PRODUCER ready CONSUMER ready
p
run PRODUCER. product 5     writeptr5
count = 5
time = 5

        ------ ------ ------ ------ ------ ------ ------ ------
       |   1  |   2  |   3  |   4  |   5  |      |      |      |
        ------ ------ ------ ------ ------ ------ ------ ------
writeptr = 5, readptr = 0,  PRODUCER ready CONSUMER ready
p
run PRODUCER. product 6     writeptr6
count = 6
time = 6

        ------ ------ ------ ------ ------ ------ ------ ------
       |   1  |   2  |   3  |   4  |   5  |   6  |      |      |
        ------ ------ ------ ------ ------ ------ ------ ------
writeptr = 6, readptr = 0,  PRODUCER ready CONSUMER ready
p
run PRODUCER. product 7     writeptr7
count = 7
time = 7

        ------ ------ ------ ------ ------ ------ ------ ------
       |   1  |   2  |   3  |   4  |   5  |   6  |   7  |      |
        ------ ------ ------ ------ ------ ------ ------ ------
writeptr = 7, readptr = 0,  PRODUCER ready CONSUMER ready
p
run PRODUCER. product 8     writeptr0
count = 8
time = 8
```

```
        ------ ------ ------ ------ ------ ------ ------ ------
       |   1  |   2  |   3  |   4  |   5  |   6  |   7  |   8  |
        ------ ------ ------ ------ ------ ------ ------ ------
writeptr = 0, readptr = 0,  PRODUCER ready CONSUMER ready
p
run PRODUCER. product 8     producerCount=1
PRODUCER is waiting, can't be scheduled.

        ------ ------ ------ ------ ------ ------ ------ ------
       |   1  |   2  |   3  |   4  |   5  |   6  |   7  |   8  |
        ------ ------ ------ ------ ------ ------ ------ ------
writeptr = 0, readptr = 0,  PRODUCER wait CONSUMER ready
p
producerCount=2
PRODUCER is waiting, can't be scheduled.

        ------ ------ ------ ------ ------ ------ ------ ------
       |   1  |   2  |   3  |   4  |   5  |   6  |   7  |   8  |
        ------ ------ ------ ------ ------ ------ ------ ------
writeptr = 0, readptr = 0,  PRODUCER wait CONSUMER ready
c
run CONSUMER. readptr=1
 use 1      count = 7
time = 1
return AWAKE
run PRODUCER. product 9     writeptr1
count = 8
time = 9
producerCount=1

        ------ ------ ------ ------ ------ ------ ------ ------
       |   9  |   2  |   3  |   4  |   5  |   6  |   7  |   8  |
        ------ ------ ------ ------ ------ ------ ------ ------
writeptr = 1, readptr = 1,  PRODUCER ready CONSUMER ready
c
```

```
run CONSUMER. readptr=2
 use 2      count = 7
time = 2
return AWAKE
run PRODUCER. product 10     writeptr2
count = 8
time = 10
producerCount=0

        ------ ------ ------ ------ ------ ------ ------ ------
       |   9  |  10  |   3  |   4  |   5  |   6  |   7  |   8  |
        ------ ------ ------ ------ ------ ------ ------ ------
writeptr = 2, readptr = 2,  PRODUCER ready CONSUMER ready
c
run CONSUMER. readptr=3
 use 3      count = 7
time = 3

        ------ ------ ------ ------ ------ ------ ------ ------
       |   9  |  10  |      |   4  |   5  |   6  |   7  |   8  |
        ------ ------ ------ ------ ------ ------ ------ ------
writeptr = 2, readptr = 3,  PRODUCER ready CONSUMER ready
c
run CONSUMER. readptr=4
 use 4      count = 6
time = 4

        ------ ------ ------ ------ ------ ------ ------ ------
       |   9  |  10  |      |      |   5  |   6  |   7  |   8  |
        ------ ------ ------ ------ ------ ------ ------ ------
writeptr = 2, readptr = 4,  PRODUCER ready CONSUMER ready
c
run CONSUMER. readptr=5
 use 5      count = 5
time = 5
```


writeptr = 2, readptr = 5, PRODUCER ready CONSUMER ready
c
run CONSUMER. readptr=6
use 6 count = 4
time = 6
writeptr = 2, readptr = 6, PRODUCER ready CONSUMER ready
c
run CONSUMER. readptr=7
use 7 count = 3
time = 7
writeptr = 2, readptr = 7, PRODUCER ready CONSUMER ready
c
run CONSUMER. readptr=0
use 8 count = 2
time = 8
writeptr = 2, readptr = 0, PRODUCER ready CONSUMER ready
c
run CONSUMER. readptr=1
use 9 count = 1
time = 9


writeptr = 2, readptr = 1, PRODUCER ready CONSUMER ready
c
run CONSUMER. readptr=2
use 10 count = 0
time = 10
writeptr = 2, readptr = 2, PRODUCER ready CONSUMER ready
c
run CONSUMER. consumerCount=1
CONSUMER is waiting, can't be scheduled.
writeptr = 2, readptr = 2, PRODUCER ready CONSUMER wait
c
consumerCount=2
CONSUMER is waiting, can't be scheduled.
writeptr = 2, readptr = 2, PRODUCER ready CONSUMER wait
p
run PRODUCER. product 11 writeptr3
count = 1
time = 11
return AWAKErun CONSUMER. readptr=3
use 11 count = 0
time = 11
consumerCount=1


writeptr = 3, readptr = 3, PRODUCER ready CONSUMER ready
p
run PRODUCER. product 12 writeptr4
count = 1
time = 12
return AWAKErun CONSUMER. readptr=4
use 12 count = 0
time = 12
consumerCount=0
writeptr = 4, readptr = 4, PRODUCER ready CONSUMER ready
p
run PRODUCER. product 13 writeptr5
count = 1
time = 13
writeptr = 5, readptr = 4, PRODUCER ready CONSUMER ready
p
run PRODUCER. product 14 writeptr6
count = 2
time = 14
writeptr = 6, readptr = 4, PRODUCER ready CONSUMER ready