《软件测试》 大作业报告

警示：本报告两人一组，共同完成。报告以 A4 规格的 PDF 格式文档按规定命名并按时提交，严禁抄袭。

邮件主题命名：《软件测试》期末大作业_学号 1_学号 2

邮件附件命名：《软件测试》期末大作业_学号 1_学号 2.pdf

分组编号	**9**	学号 1	**17343104**	姓名 1	唐婵
		学号 2	**17343105**	姓名 2	田皓
提交邮箱	isscgy@163.com		截止提交时间	2020 年 07 月 31 日 23 时	

【测试材料】

- 一个用于演示操作系统进程互斥与同步的生产者-消费者模型的示范系统。
- 功能说明：
  a) 启动主控程序 syn-pc-con-6，运行参数是一个可以共享的文件名例如 /home/myshm。该文件必须事先建立。
  b) 主控程序输入临时存储产品的缓冲区大小、计划生产的产品总数量、生产者线程数量和消费者线程数量，并激活生产者和消费者进程。
  c) 生产者进程：若干生产线程向上述缓冲区放入产品（忽略具体的生产过程），缓冲区满时需要等待。生产者进程在完成计划产品总数量时结束。
  d) 消费者进程：若干消费线程从上述缓冲区取出产品，缓冲区空时需要等待。消费者进程在发现生产者进程结束时，取空缓冲区的产品后结束。
- C 源代码清单：
  syn-pc-con-6.h
  syn-pc-con-6.c
  syn-pc-producer-6.c
  syn-pc-consumer-6.c
- 某次运行的屏幕快照：
  screensnap.png
- 运行平台：x86-PC / Ubuntu 18.04
- 编译器：gcc version 7.5.0
- 运行库：POSIX pthread （编译选项 -lpthread）

【测试内容】

1. 静态测试：
   a) 对源代码以你的观点进行静态代码检查，给出检查报告。
2. 复杂性分析：
   a) 计算 syn-pc-con-6.c 的 *Hastead* 复杂度；
   b) 计算 syn-pc-consumer-6.c 的 *McCabe* 复杂度。
3. 白盒测试：
   a) 对 syn-pc-consumer-6.c 实现条件覆盖测试。
4. 黑盒测试：
   a) 对主控模块 syn-pc-con-6.c 的输入实现等价类划分测试。
5. 系统测试：
   a) 自行选择两种故障模型进行软件故障静态注入测试。

【测试环境】

操作系统：CentOS 7

编译器：gcc version 7.5.0

运行库：POSIX pthread

【测试用例】

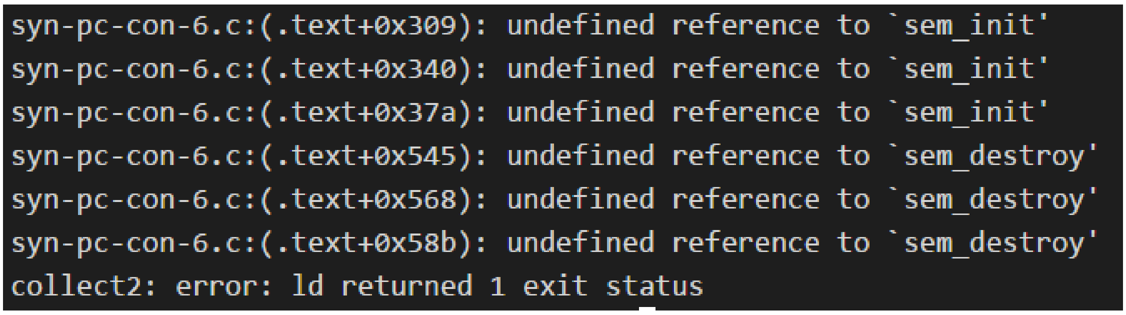
见测试过程的黑盒测试和白盒测试部分

【测试过程】

静态测试

直接尝试编译：gcc syn-pc-con-6.c -o test

报错：找不到 syn-pc-con-5.h ，查看代码发现#include 的文件名错误，改为 syn-pc-con-6.h，再次执行，报错：



```
syn-pc-con-6.c:(.text+0x309): undefined reference to `sem_init'
syn-pc-con-6.c:(.text+0x340): undefined reference to `sem_init'
syn-pc-con-6.c:(.text+0x37a): undefined reference to `sem_init'
syn-pc-con-6.c:(.text+0x545): undefined reference to `sem_destroy'
syn-pc-con-6.c:(.text+0x568): undefined reference to `sem_destroy'
syn-pc-con-6.c:(.text+0x58b): undefined reference to `sem_destroy'
collect2: error: ld returned 1 exit status
```

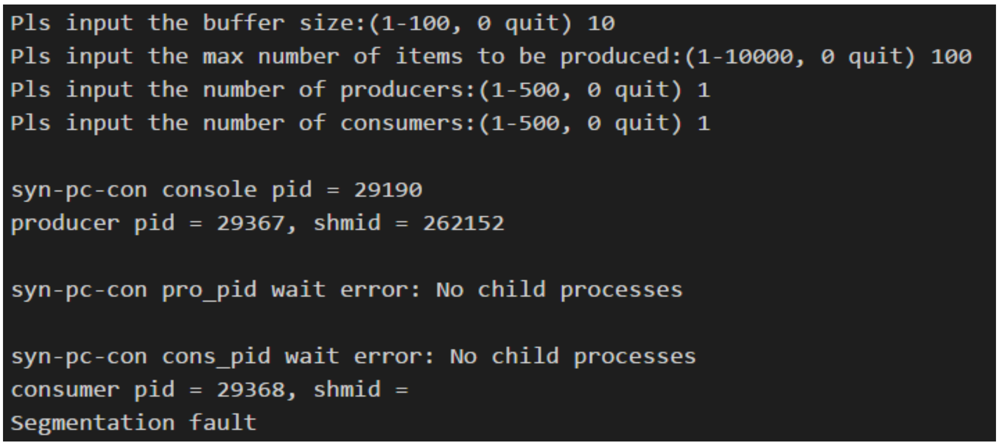
因为 pthread 并非默认 Linux 默认链接库，需要显示链接，即编译命令加上参数 -lpthread ，发现作业要求中【测试材料】部分有提及。

gcc -lpthread syn-pc-con-6.c -o test

编译成功，创建空目录 mytest，运行 ./test mytest 没有反应，查看代码发现文件名写死，再次修改编译命令

gcc -lpthread syn-pc-con-6.c -o syn-pc-con-6.o

编译成功，运行 ./syn-pc-con-6.o mytest：

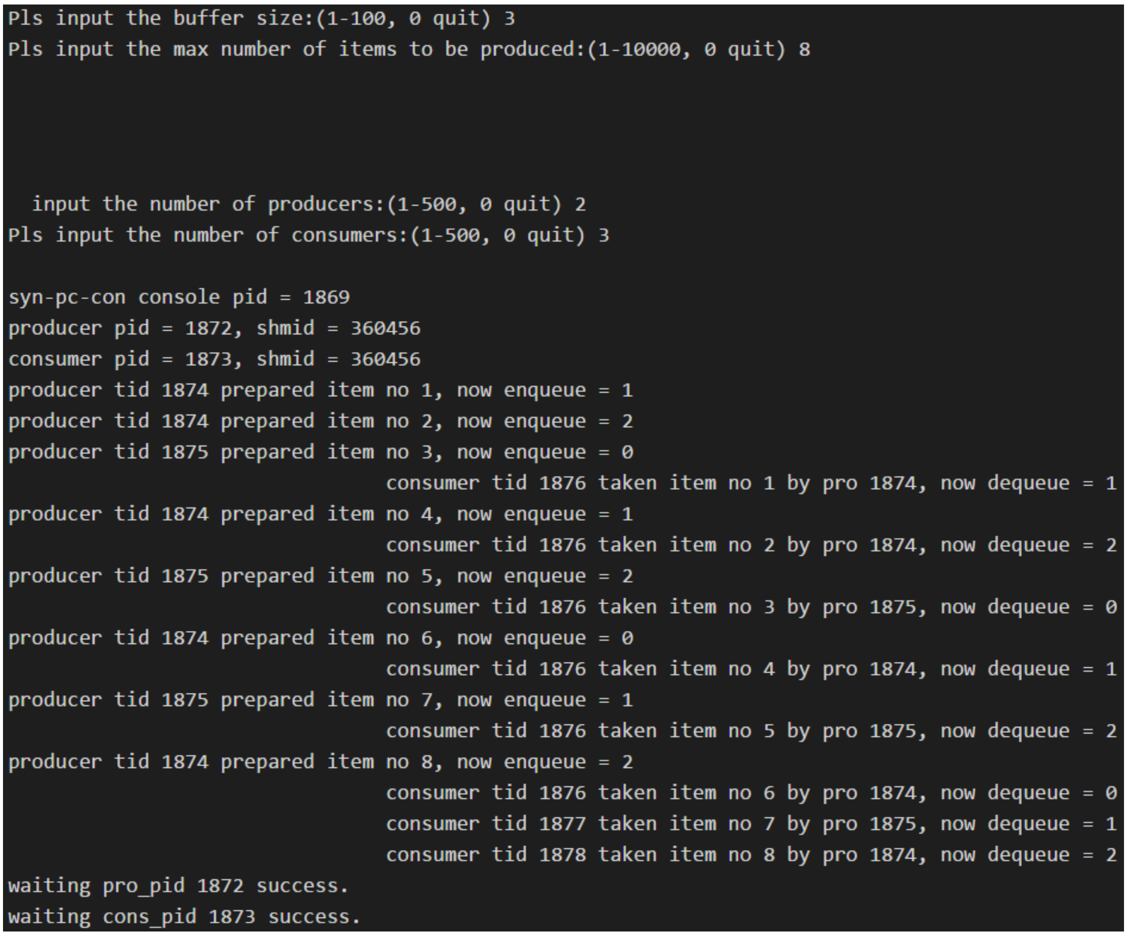


```
Pls input the buffer size:(1-100, 0 quit) 10
Pls input the max number of items to be produced:(1-10000, 0 quit) 100
Pls input the number of producers:(1-500, 0 quit) 1
Pls input the number of consumers:(1-500, 0 quit) 1

syn-pc-con console pid = 29190
producer pid = 29367, shmid = 262152

syn-pc-con pro_pid wait error: No child processes

syn-pc-con cons_pid wait error: No child processes
consumer pid = 29368, shmid =
Segmentation fault
```

段错误，查看源代码，发现需要调用另外两个文件编译生成的程序，于是执行编译

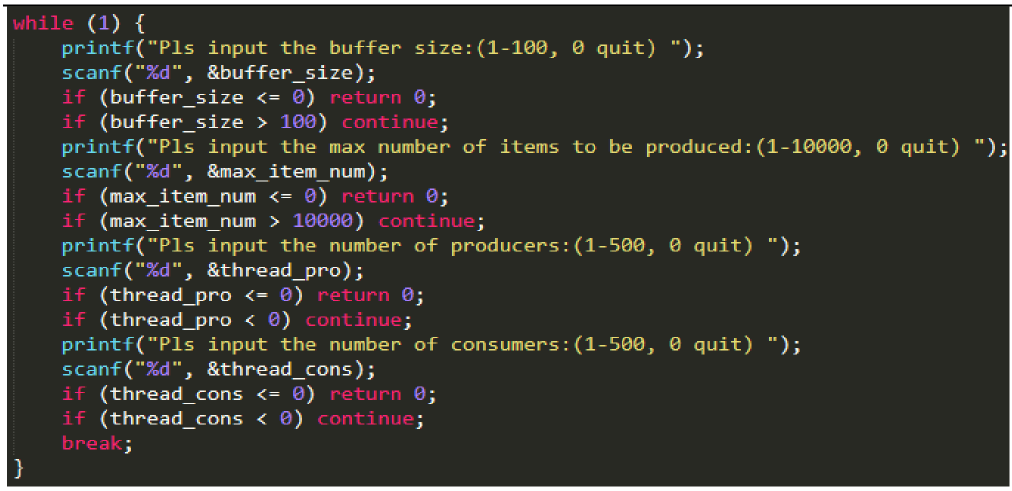
```
gcc -lpthread syn-pc-consumer-6.c -o syn-pc-consumer-6.o
gcc -lpthread syn-pc-producer-6.c -o syn-pc-producer-6.o
```

再次运行 ./syn-pc-con-6.o mytest，得到截图中结果：



```
Pls input the buffer size:(1-100, 0 quit) 3
Pls input the max number of items to be produced:(1-10000, 0 quit) 8




  input the number of producers:(1-500, 0 quit) 2
Pls input the number of consumers:(1-500, 0 quit) 3

syn-pc-con console pid = 1869
producer pid = 1872, shmid = 360456
consumer pid = 1873, shmid = 360456
producer tid 1874 prepared item no 1, now enqueue = 1
producer tid 1874 prepared item no 2, now enqueue = 2
producer tid 1875 prepared item no 3, now enqueue = 0
                                consumer tid 1876 taken item no 1 by pro 1874, now dequeue = 1
producer tid 1874 prepared item no 4, now enqueue = 1
                                consumer tid 1876 taken item no 2 by pro 1874, now dequeue = 2
producer tid 1875 prepared item no 5, now enqueue = 2
                                consumer tid 1876 taken item no 3 by pro 1875, now dequeue = 0
producer tid 1874 prepared item no 6, now enqueue = 0
                                consumer tid 1876 taken item no 4 by pro 1874, now dequeue = 1
producer tid 1875 prepared item no 7, now enqueue = 1
                                consumer tid 1876 taken item no 5 by pro 1875, now dequeue = 2
producer tid 1874 prepared item no 8, now enqueue = 2
                                consumer tid 1876 taken item no 6 by pro 1874, now dequeue = 0
                                consumer tid 1877 taken item no 7 by pro 1875, now dequeue = 1
                                consumer tid 1878 taken item no 8 by pro 1874, now dequeue = 2
waiting pro_pid 1872 success.
waiting cons_pid 1873 success.
```

随后，观察源代码的输入参数处理部分，发现 thread_pro<0 和 thread_cons<0 的判断是冗余的，在 continue 之前就会 return，同时此处的错误会对之后的测试有影响，因为没有判断这两个参数的右边界。



```
while (1) {
    printf("Pls input the buffer size:(1-100, 0 quit) ");
    scanf("%d", &buffer_size);
    if (buffer_size <= 0) return 0;
    if (buffer_size > 100) continue;
    printf("Pls input the max number of items to be produced:(1-10000, 0 quit) ");
    scanf("%d", &max_item_num);
    if (max_item_num <= 0) return 0;
    if (max_item_num > 10000) continue;
    printf("Pls input the number of producers:(1-500, 0 quit) ");
    scanf("%d", &thread_pro);
    if (thread_pro <= 0) return 0;
    if (thread_pro < 0) continue;
    printf("Pls input the number of consumers:(1-500, 0 quit) ");
    scanf("%d", &thread_cons);
    if (thread_cons <= 0) return 0;
    if (thread_cons < 0) continue;
    break;
}
```

复杂性分析

a) 计算 syn-pc-con-6.c 的 *Hastead* 复杂度；

操作符	数量	操作数	数量
if	30	childpid	7
<=	4	pro_pid	8
<	6	cons_pid	8
==	13	statbuf	2
&	11	buffer_size	10
while	1	max*item*num	7
return	14	thread_pro	7
continue	4	thread_cons	7
break	1	argc	2
=	36	argv	11
->	16	EXIT_FAILURE	7
>	2	ctln	20
else	5	data	8
!=	2	key	7
-	13	ret	13
		shmid	13
		0	32
		1	14
		2	3
		10	7
		3	1

		0x28	1
n1 = 15	N1 = 158	n2 = 21	N2 = 185

Program vocabulary：n = n_1 + n_2= 36

Program length: N = N_1 + N_2 = 343

Calculated program length: N^ = $n_1 \ log_2 n_1$ + $n_2 \, log_2 n_2$ = 150.84

Program volume: V = N$log_2 n$ = 1773.28

Program level: L^ = (2/n_1) * (n_2/N_2) = 0.015

Program difficulty：D = 1 / L^ = 66.67

Program Effort：E = V * D = 118218.67

Language level: L’ = L^L * V = 0.399

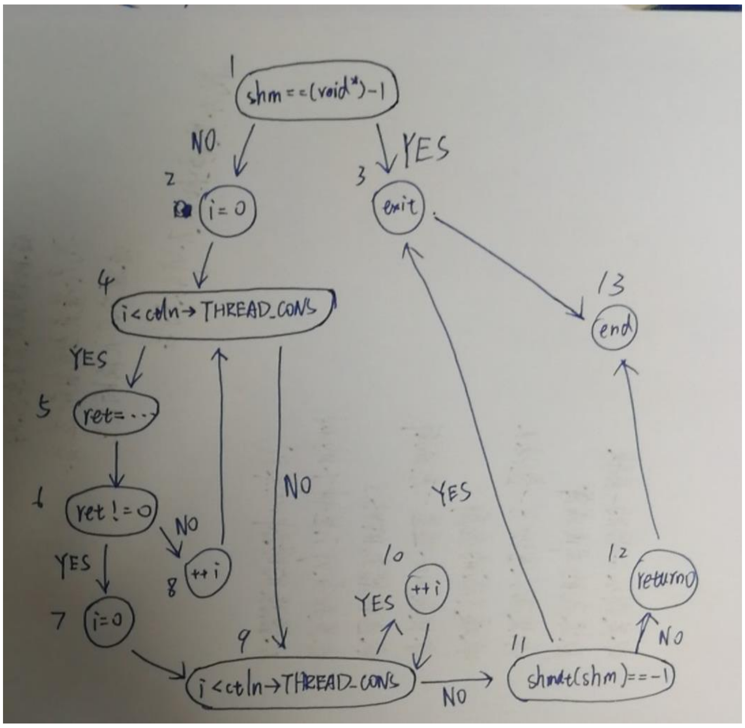
Program Time (hours): T^ = E / (S * f) = 1.82

平均语句大小：N / 语句数 = 2.98

程序中的错误数预测值：B = V / 3000 = 39.4

b) 计算 syn-pc-consumer-6.c 的 *McCabe* 复杂度。

程序控制流图如下：



McCabe 复杂度：V(G) = d +1 = 5 + 1 = 6，　d 为单条件判断节点个数

白盒测试

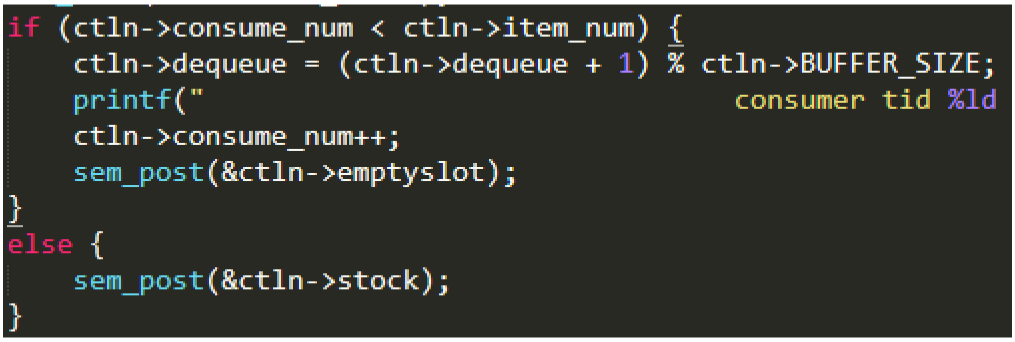
a) 对 syn-pc-consumer-6.c 实现条件覆盖测试。

条件覆盖测试即对源代码中每个条件表达式的 True 和 False 各取值一次。

阅读 syn-pc-consumer-6.c 源代码，判断条件如下：

1. if (ctln->consume_num < ctln->item_num)
2. if (shm == (void *)-1)
3. if (ret != 0)
4. if (shmdt(shm) == -1)

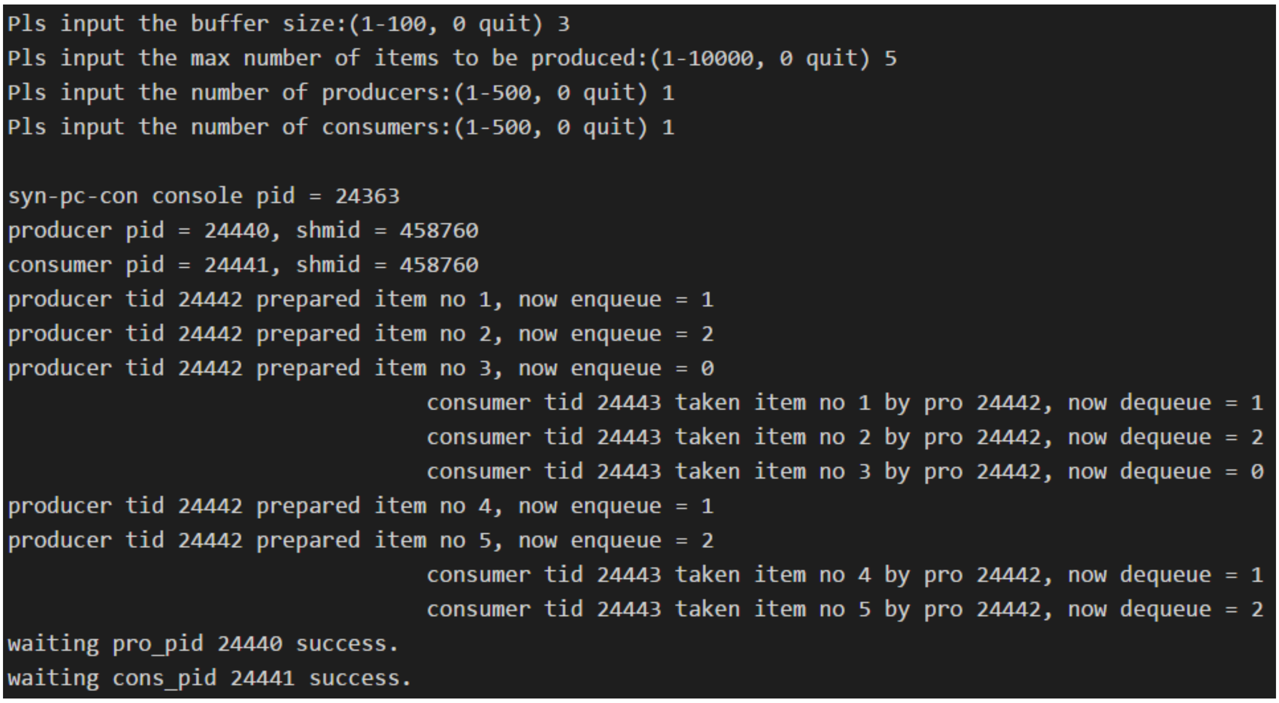
首先考虑覆盖条件 1，相关参数有两个，已消费产品总数量和已生产产品总数量



```
if (ctln->consume_num < ctln->item_num) {
    ctln->dequeue = (ctln->dequeue + 1) % ctln->BUFFER_SIZE;
    printf("                    consumer tid %ld
    ctln->consume_num++;
    sem_post(&ctln->emptyslot);
}
else {
    sem_post(&ctln->stock);
}
```

若判断为 True 则 deuqeue 移动且消费数加 1，False 则什么都不做，设计测试用例如下：

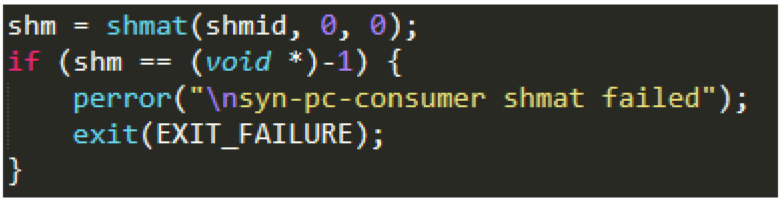
BUFFER_SIZE	MAX_ITEM_NUM	THREAD_PRO	THREAD_CONS
3	5	1	1



```
Pls input the buffer size:(1-100, 0 quit) 3
Pls input the max number of items to be produced:(1-10000, 0 quit) 5
Pls input the number of producers:(1-500, 0 quit) 1
Pls input the number of consumers:(1-500, 0 quit) 1

syn-pc-con console pid = 24363
producer pid = 24440, shmid = 458760
consumer pid = 24441, shmid = 458760
producer tid 24442 prepared item no 1, now enqueue = 1
producer tid 24442 prepared item no 2, now enqueue = 2
producer tid 24442 prepared item no 3, now enqueue = 0
                                consumer tid 24443 taken item no 1 by pro 24442, now dequeue = 1
                                consumer tid 24443 taken item no 2 by pro 24442, now dequeue = 2
                                consumer tid 24443 taken item no 3 by pro 24442, now dequeue = 0
producer tid 24442 prepared item no 4, now enqueue = 1
producer tid 24442 prepared item no 5, now enqueue = 2
                                consumer tid 24443 taken item no 4 by pro 24442, now dequeue = 1
                                consumer tid 24443 taken item no 5 by pro 24442, now dequeue = 2
waiting pro_pid 24440 success.
waiting cons_pid 24441 success.
```

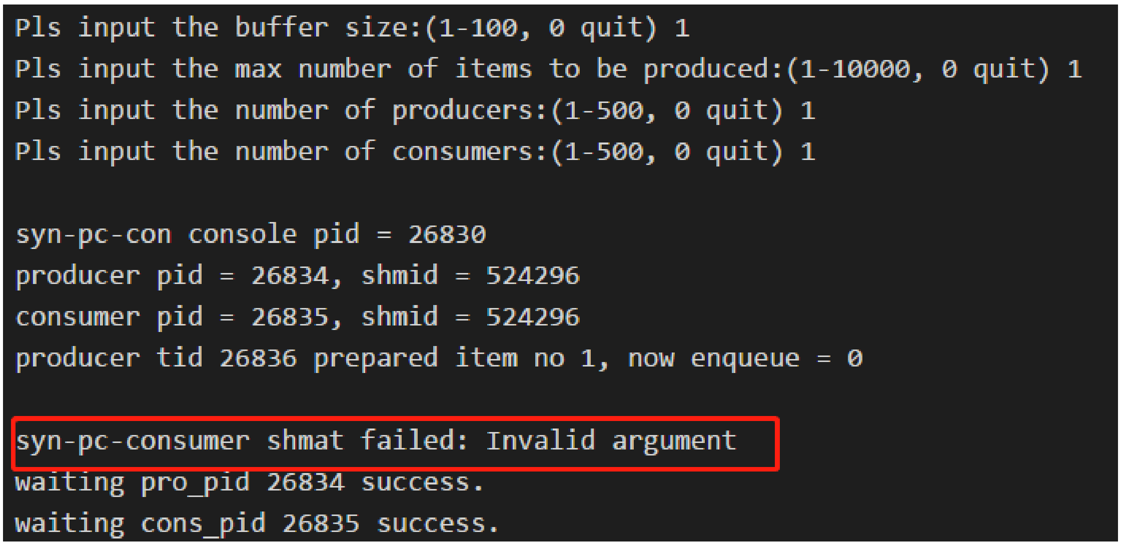
生产者先生产 3 个产品，此时缓冲区已满，消费者开始消费，条件 1 一直为真，直到消费了三个产品，此时条件 1 为假，逻辑覆盖完成。

条件 2：if (shm == (void *)-1)



```
shm = shmat(shmid, 0, 0);
if (shm == (void *)-1) {
    perror("\nsyn-pc-consumer shmat failed");
    exit(EXIT_FAILURE);
}
```

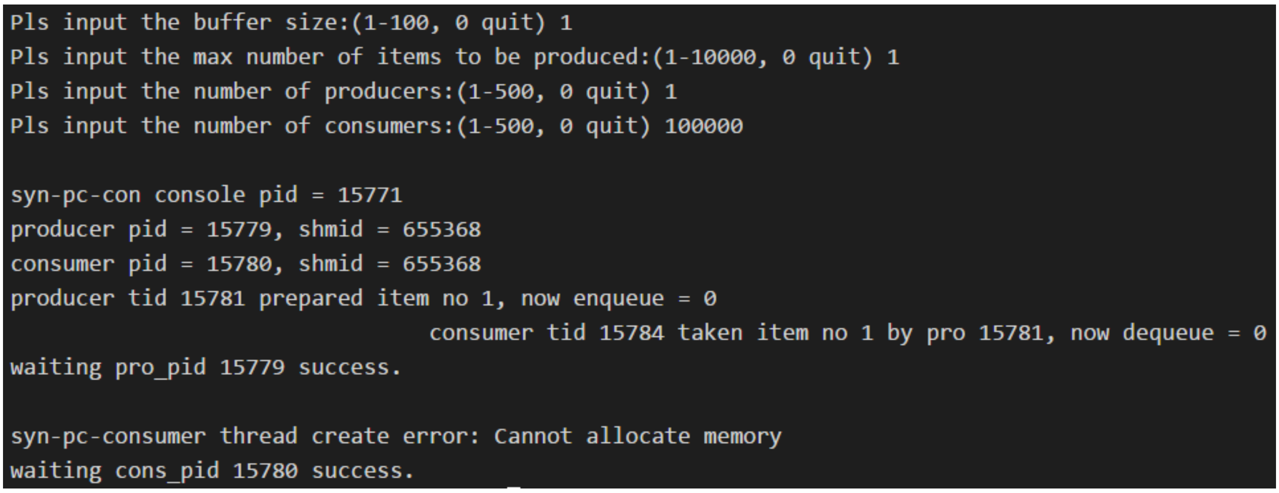
shmat 为开启共享内存对该进程的访问，需要让该函数调用失败使条件 2 为 True，修改传入的参数即可



```
Pls input the buffer size:(1-100, 0 quit) 1
Pls input the max number of items to be produced:(1-10000, 0 quit) 1
Pls input the number of producers:(1-500, 0 quit) 1
Pls input the number of consumers:(1-500, 0 quit) 1

syn-pc-con console pid = 26830
producer pid = 26834, shmid = 524296
consumer pid = 26835, shmid = 524296
producer tid 26836 prepared item no 1, now enqueue = 0

syn-pc-consumer shmat failed: Invalid argument
waiting pro_pid 26834 success.
waiting cons_pid 26835 success.
```

条件 3 需要让子线程创建失败，因为在静态检查中提到的对输入参数处理的漏洞，所以可以实现这一点，测试用例如下：

BUFFER_SIZE	MAX_ITEM_NUM	THREAD_PRO	THREAD_CONS
1	1	1	10000



```
Pls input the buffer size:(1-100, 0 quit) 1
Pls input the max number of items to be produced:(1-10000, 0 quit) 1
Pls input the number of producers:(1-500, 0 quit) 1
Pls input the number of consumers:(1-500, 0 quit) 100000

syn-pc-con console pid = 15771
producer pid = 15779, shmid = 655368
consumer pid = 15780, shmid = 655368
producer tid 15781 prepared item no 1, now enqueue = 0
                                consumer tid 15784 taken item no 1 by pro 15781, now dequeue = 0
waiting pro_pid 15779 success.

syn-pc-consumer thread create error: Cannot allocate memory
waiting cons_pid 15780 success.
```

可以看到打印了创建子线程错误的信息，说明条件 3 进入 True 分支，因为数量太多而导致内存不够，条件 3 已覆盖

条件 4 与条件 2 相同。

黑盒测试

a) 对主控模块 syn-pc-con-6.c 的输入实现等价类划分测试。

参数一共有四个，根据他们的取值范围设计测试用例

buffer_size：1-100，0 退出

max_*item*_num:：1-10000，0 退出

producer：1-500，0 退出

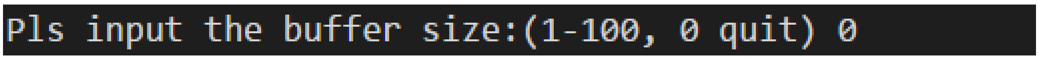
consumer：1-500，0 退出

	buffer_size	max_*item*_num	thread_pro	thread_cons	预期结果
W1	0	1	1	1	

W2	1	0	1	1	输入 0 之后退出程序
W3	1	1	0	1	
W4	1	1	1	0	
W5	101	1	1	1	输入超过范围的数据会提示重新输入
W6	1	10001	1	1	
W7	1	1	501	1	
W8	1	1	1	501	
W9	-1	1	1	1	
W10	1	-1	1	1	
W11	1	1	-1	1	
W12	1	1	1	-1	
W13	5	3	1	1	
W14	3	5	1	1	
W15	5	3	3	3	
W16	3	5	3	3	

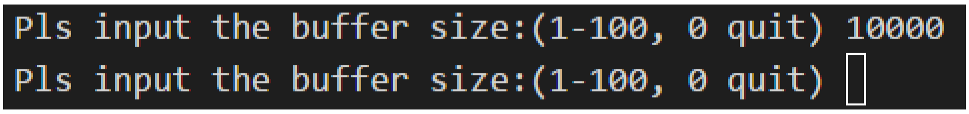
W1234:

输入 0 之后退出程序，符合预期



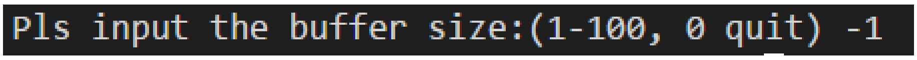
```
Pls input the buffer size:(1-100, 0 quit) 0
```

W5678:



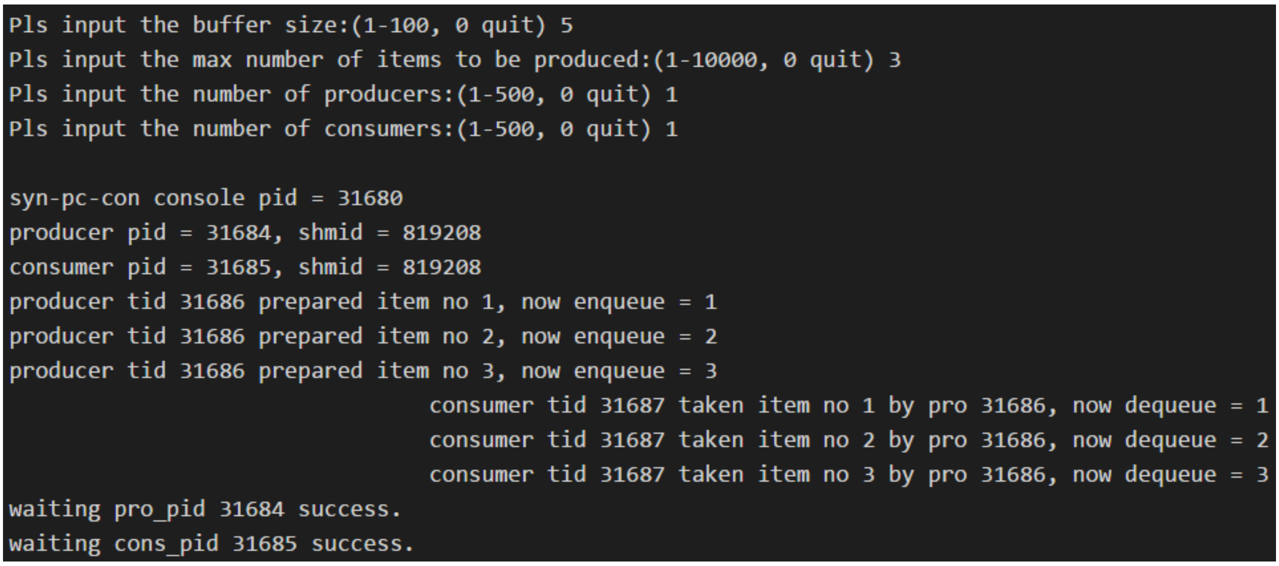
```
Pls input the buffer size:(1-100, 0 quit) 10000
Pls input the buffer size:(1-100, 0 quit) 
```

W9,10,11,12:



```
Pls input the buffer size:(1-100, 0 quit) -1
```

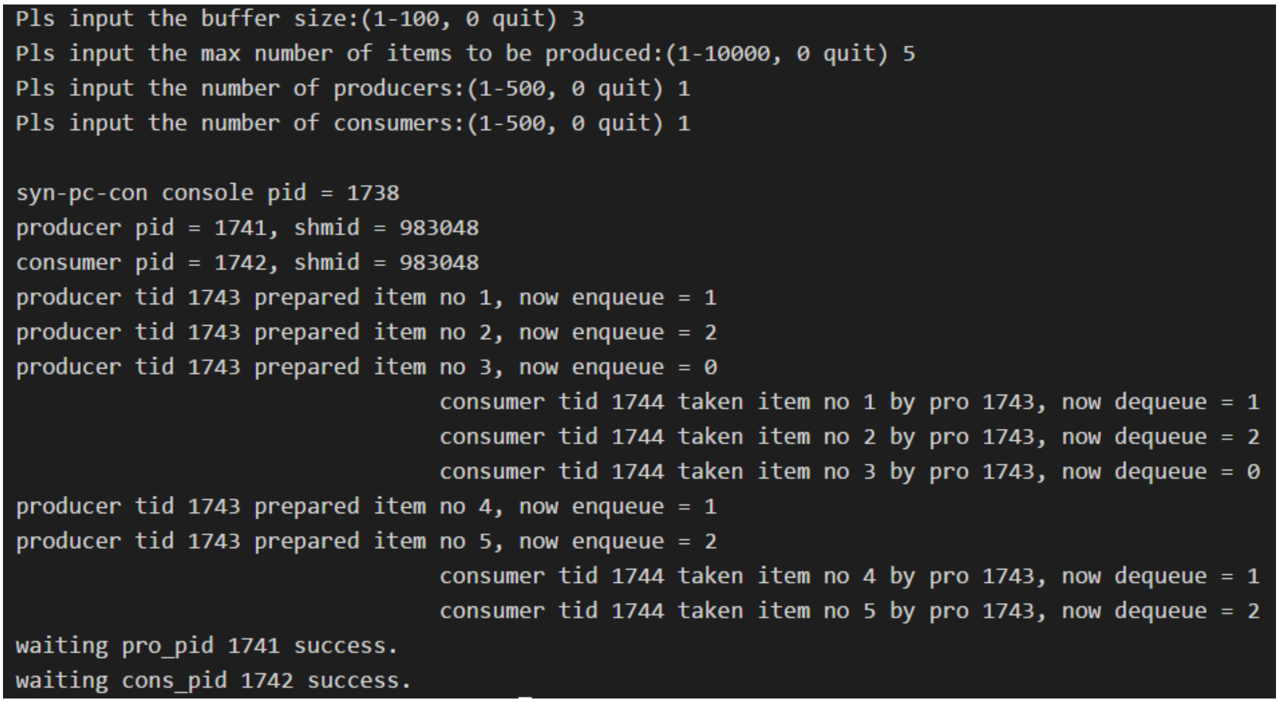
W13:



```
Pls input the buffer size:(1-100, 0 quit) 5
Pls input the max number of items to be produced:(1-10000, 0 quit) 3
Pls input the number of producers:(1-500, 0 quit) 1
Pls input the number of consumers:(1-500, 0 quit) 1

syn-pc-con console pid = 31680
producer pid = 31684, shmid = 819208
consumer pid = 31685, shmid = 819208
producer tid 31686 prepared item no 1, now enqueue = 1
producer tid 31686 prepared item no 2, now enqueue = 2
producer tid 31686 prepared item no 3, now enqueue = 3
                                consumer tid 31687 taken item no 1 by pro 31686, now dequeue = 1
                                consumer tid 31687 taken item no 2 by pro 31686, now dequeue = 2
                                consumer tid 31687 taken item no 3 by pro 31686, now dequeue = 3
waiting pro_pid 31684 success.
waiting cons_pid 31685 success.
```

缓冲区比最大产品数量大，线程数都为 1 时，生产者线程会占用锁，，一次生产完所有产品，消费者再开始消费。

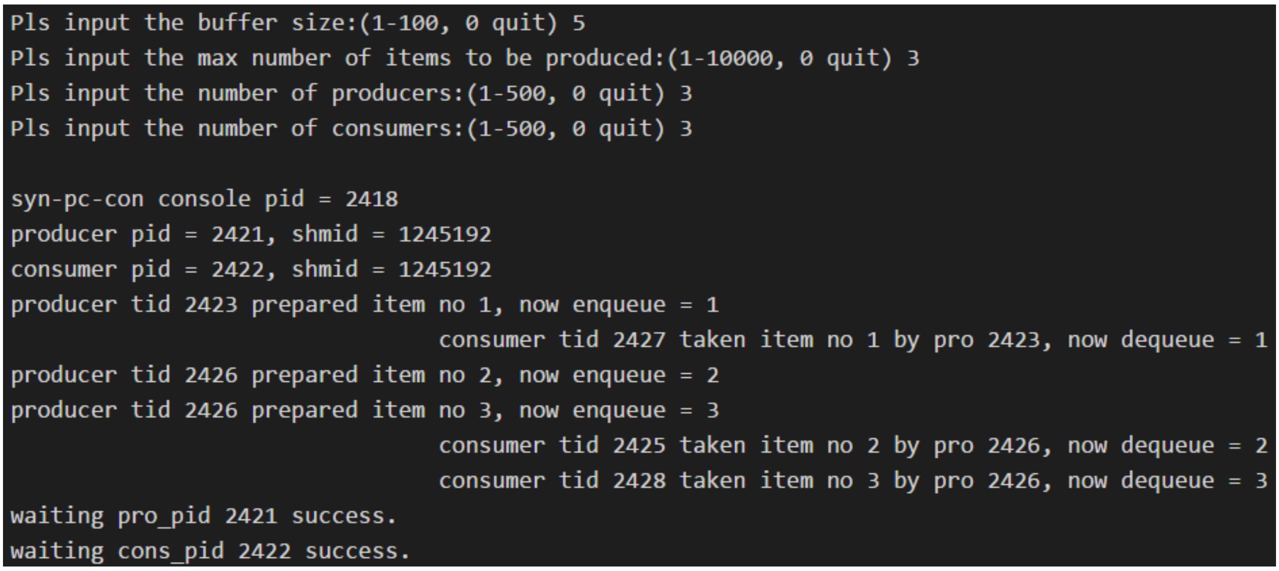
W14：



```
Pls input the buffer size:(1-100, 0 quit) 3
Pls input the max number of items to be produced:(1-10000, 0 quit) 5
Pls input the number of producers:(1-500, 0 quit) 1
Pls input the number of consumers:(1-500, 0 quit) 1

syn-pc-con console pid = 1738
producer pid = 1741, shmid = 983048
consumer pid = 1742, shmid = 983048
producer tid 1743 prepared item no 1, now enqueue = 1
producer tid 1743 prepared item no 2, now enqueue = 2
producer tid 1743 prepared item no 3, now enqueue = 0
                                consumer tid 1744 taken item no 1 by pro 1743, now dequeue = 1
                                consumer tid 1744 taken item no 2 by pro 1743, now dequeue = 2
                                consumer tid 1744 taken item no 3 by pro 1743, now dequeue = 0
producer tid 1743 prepared item no 4, now enqueue = 1
producer tid 1743 prepared item no 5, now enqueue = 2
                                consumer tid 1744 taken item no 4 by pro 1743, now dequeue = 1
                                consumer tid 1744 taken item no 5 by pro 1743, now dequeue = 2
waiting pro_pid 1741 success.
waiting cons_pid 1742 success.
```

缓冲区比最大产品数小，线程数都为 1，缓冲区满之后停止生产，消费者开始消费，由于都是单线程， 所以会一直消费完所有产品再由生产者生产完剩余的产品，然后再次消费。

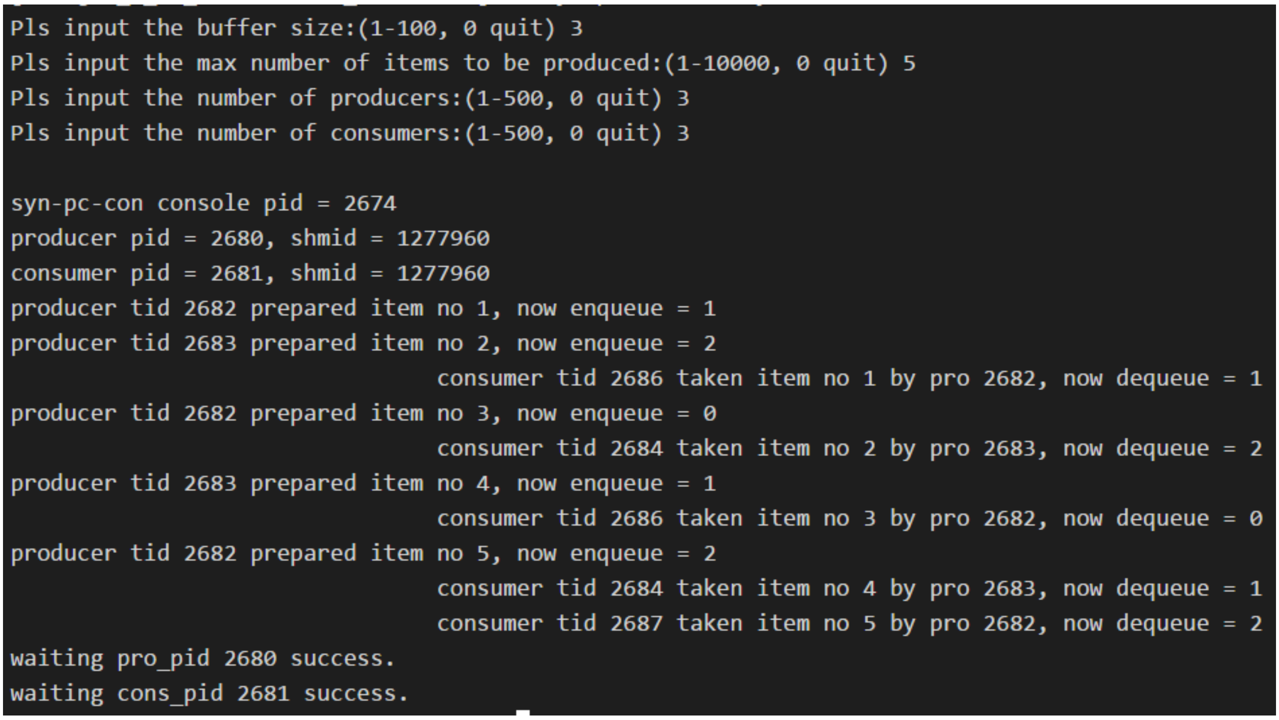
W15：



```
Pls input the buffer size:(1-100, 0 quit) 5
Pls input the max number of items to be produced:(1-10000, 0 quit) 3
Pls input the number of producers:(1-500, 0 quit) 3
Pls input the number of consumers:(1-500, 0 quit) 3

syn-pc-con console pid = 2418
producer pid = 2421, shmid = 1245192
consumer pid = 2422, shmid = 1245192
producer tid 2423 prepared item no 1, now enqueue = 1
                                consumer tid 2427 taken item no 1 by pro 2423, now dequeue = 1
producer tid 2426 prepared item no 2, now enqueue = 2
producer tid 2426 prepared item no 3, now enqueue = 3
                                consumer tid 2425 taken item no 2 by pro 2426, now dequeue = 2
                                consumer tid 2428 taken item no 3 by pro 2426, now dequeue = 3
waiting pro_pid 2421 success.
waiting cons_pid 2422 success.
```

缓冲区比最大产品数量大，线程数都为 3， 线程的调度结果不可预测。

W16：



```
Pls input the buffer size:(1-100, 0 quit) 3
Pls input the max number of items to be produced:(1-10000, 0 quit) 5
Pls input the number of producers:(1-500, 0 quit) 3
Pls input the number of consumers:(1-500, 0 quit) 3

syn-pc-con console pid = 2674
producer pid = 2680, shmid = 1277960
consumer pid = 2681, shmid = 1277960
producer tid 2682 prepared item no 1, now enqueue = 1
producer tid 2683 prepared item no 2, now enqueue = 2
                                consumer tid 2686 taken item no 1 by pro 2682, now dequeue = 1
producer tid 2682 prepared item no 3, now enqueue = 0
                                consumer tid 2684 taken item no 2 by pro 2683, now dequeue = 2
producer tid 2683 prepared item no 4, now enqueue = 1
                                consumer tid 2686 taken item no 3 by pro 2682, now dequeue = 0
producer tid 2682 prepared item no 5, now enqueue = 2
                                consumer tid 2684 taken item no 4 by pro 2683, now dequeue = 1
                                consumer tid 2687 taken item no 5 by pro 2682, now dequeue = 2
waiting pro_pid 2680 success.
waiting cons_pid 2681 success.
```

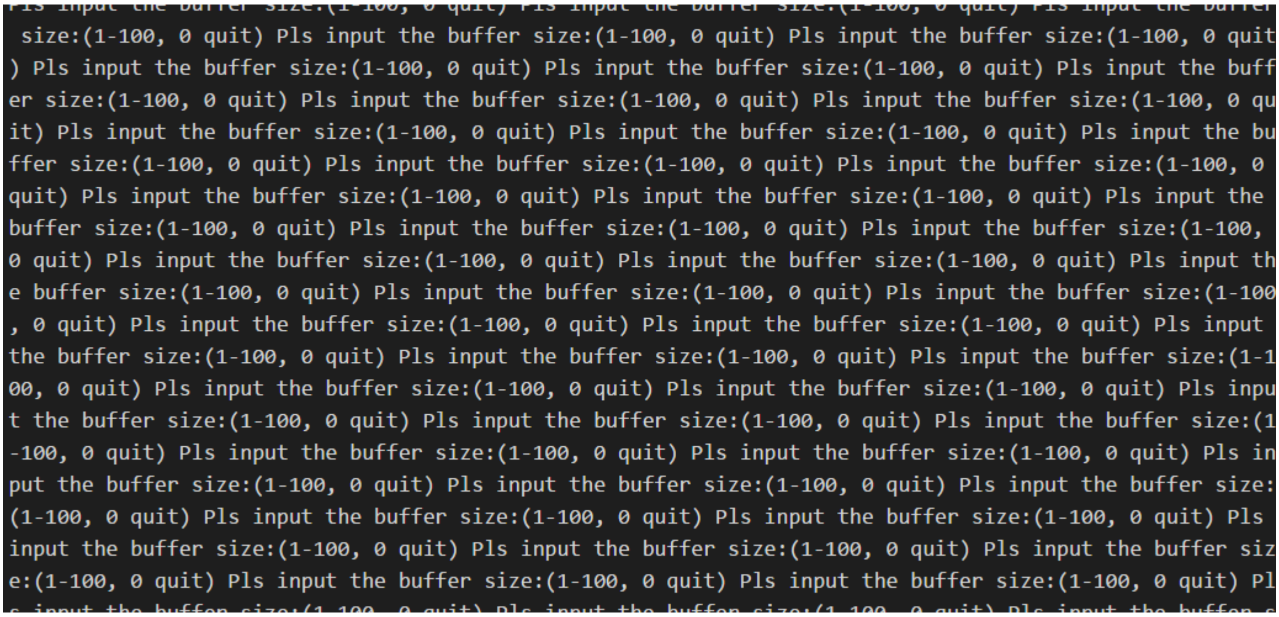
缓冲区比最大产品数量小，线程数都为 3， 线程的调度结果不可预测。

系统测试

a) 自行选择两种故障模型进行软件故障静态注入测试。

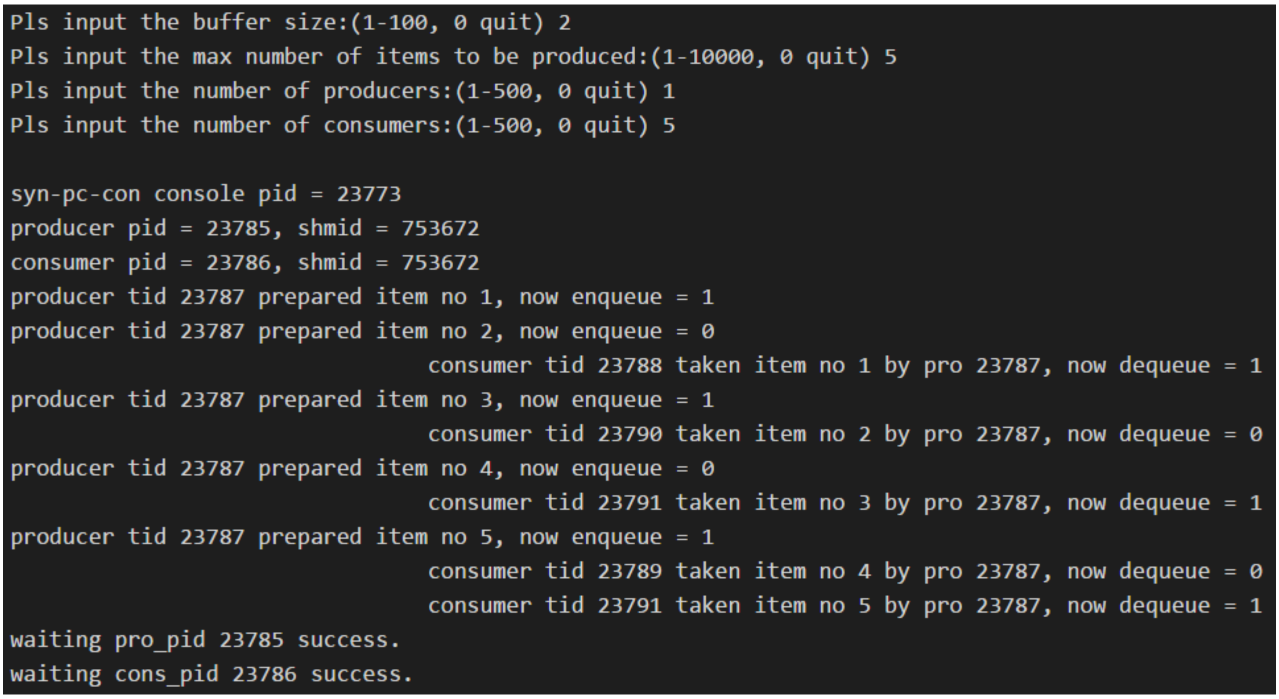
1. 输入非法数据

输入字符串，会一直死循环，并未对字符串做异常处理。

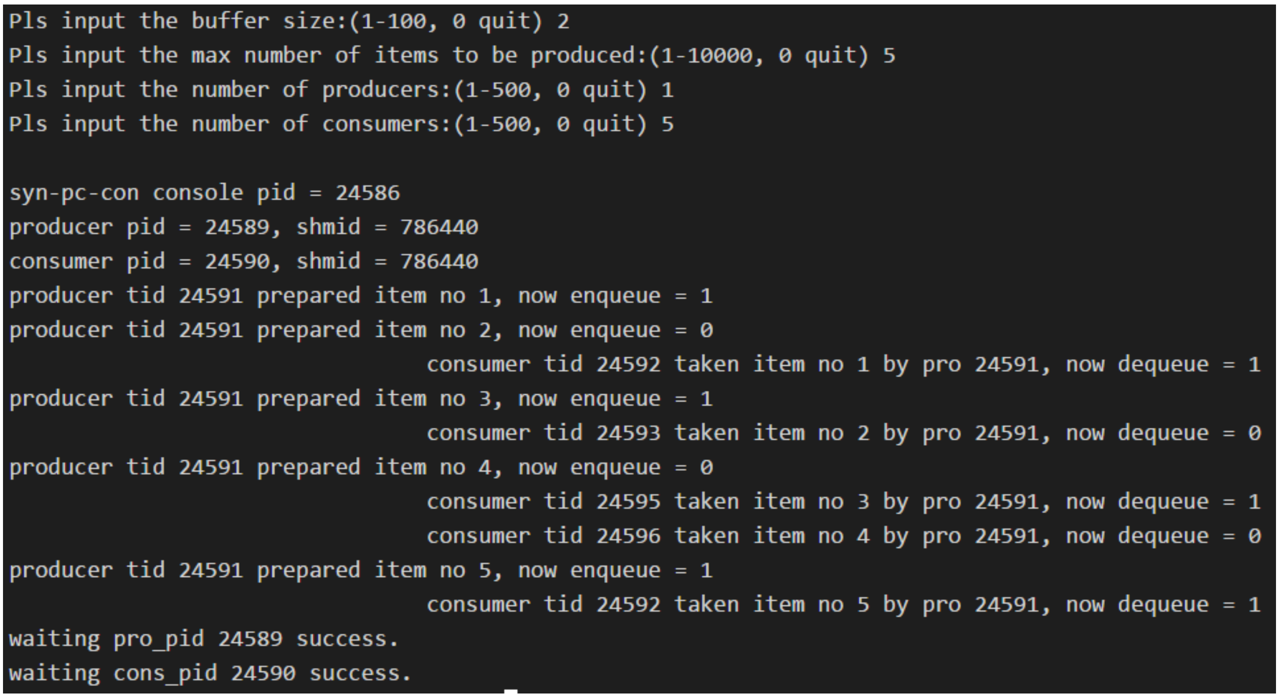


```
Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer
 size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit
) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buff
er size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 qu
it) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the bu
ffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0
quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the
buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100,
0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input th
e buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100
, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input
the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-1
00, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls inpu
t the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1
-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls in
put the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:
(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls
input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer siz
e:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pl
s input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer s
```

2. 同样输入产生不同结果



```
Pls input the buffer size:(1-100, 0 quit) 2
Pls input the max number of items to be produced:(1-10000, 0 quit) 5
Pls input the number of producers:(1-500, 0 quit) 1
Pls input the number of consumers:(1-500, 0 quit) 5

syn-pc-con console pid = 23773
producer pid = 23785, shmid = 753672
consumer pid = 23786, shmid = 753672
producer tid 23787 prepared item no 1, now enqueue = 1
producer tid 23787 prepared item no 2, now enqueue = 0
                                consumer tid 23788 taken item no 1 by pro 23787, now dequeue = 1
producer tid 23787 prepared item no 3, now enqueue = 1
                                consumer tid 23790 taken item no 2 by pro 23787, now dequeue = 0
producer tid 23787 prepared item no 4, now enqueue = 0
                                consumer tid 23791 taken item no 3 by pro 23787, now dequeue = 1
producer tid 23787 prepared item no 5, now enqueue = 1
                                consumer tid 23789 taken item no 4 by pro 23787, now dequeue = 0
                                consumer tid 23791 taken item no 5 by pro 23787, now dequeue = 1
waiting pro_pid 23785 success.
waiting cons_pid 23786 success.
```



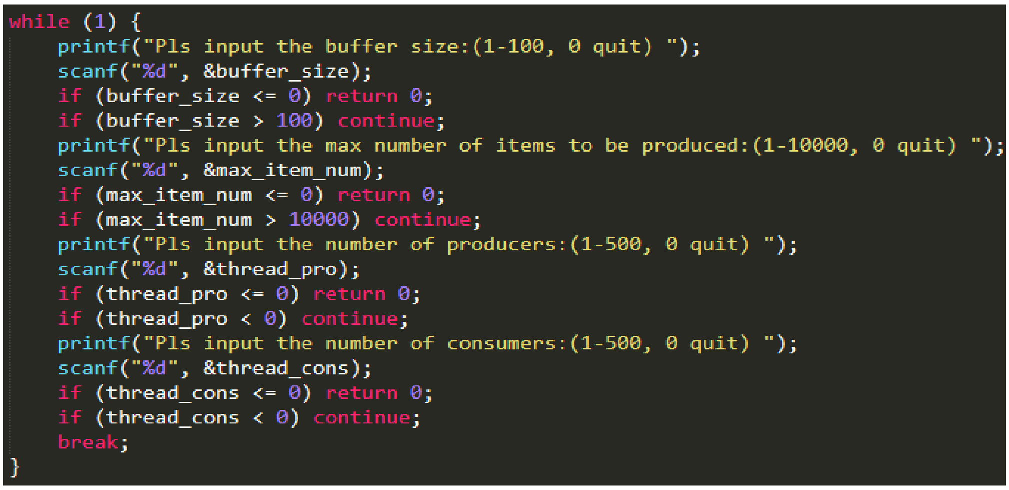
```
Pls input the buffer size:(1-100, 0 quit) 2
Pls input the max number of items to be produced:(1-10000, 0 quit) 5
Pls input the number of producers:(1-500, 0 quit) 1
Pls input the number of consumers:(1-500, 0 quit) 5

syn-pc-con console pid = 24586
producer pid = 24589, shmid = 786440
consumer pid = 24590, shmid = 786440
producer tid 24591 prepared item no 1, now enqueue = 1
producer tid 24591 prepared item no 2, now enqueue = 0
                                consumer tid 24592 taken item no 1 by pro 24591, now dequeue = 1
producer tid 24591 prepared item no 3, now enqueue = 1
                                consumer tid 24593 taken item no 2 by pro 24591, now dequeue = 0
producer tid 24591 prepared item no 4, now enqueue = 0
                                consumer tid 24595 taken item no 3 by pro 24591, now dequeue = 1
                                consumer tid 24596 taken item no 4 by pro 24591, now dequeue = 0
producer tid 24591 prepared item no 5, now enqueue = 1
                                consumer tid 24592 taken item no 5 by pro 24591, now dequeue = 1
waiting pro_pid 24589 success.
waiting cons_pid 24590 success.
```

因为线程调度是随机的，所以可能会出现同一个用例线程调度的顺序不一样的情况。

【测试结果】

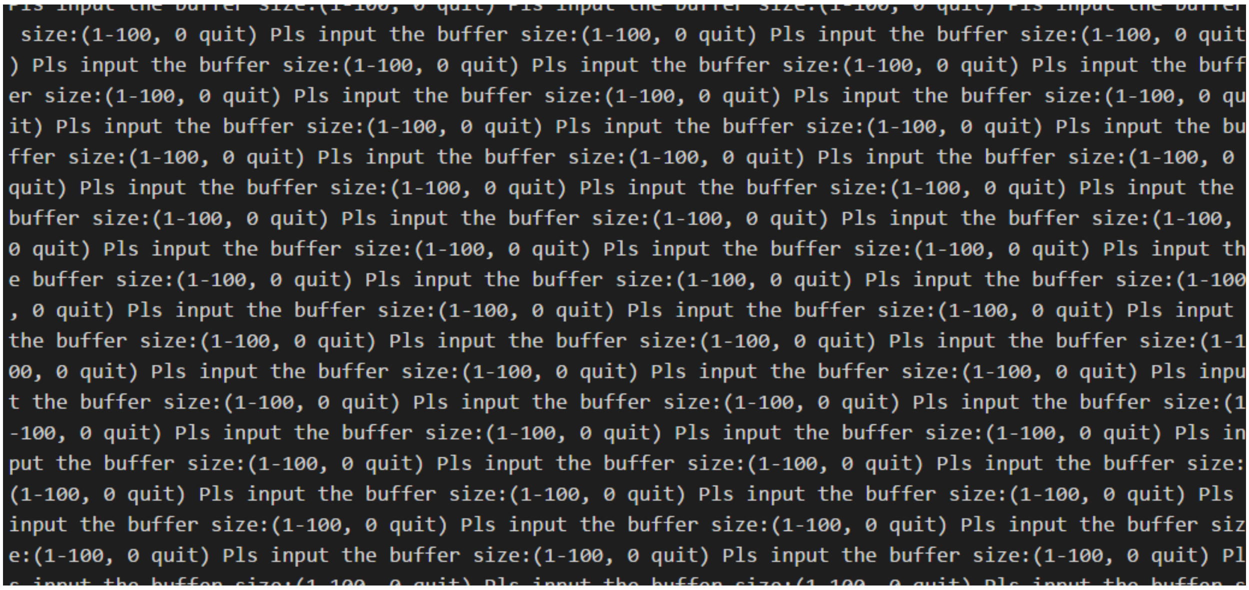
一共发现缺陷：5处

1. 头文件包含错误
2. 无法进入的分支路径



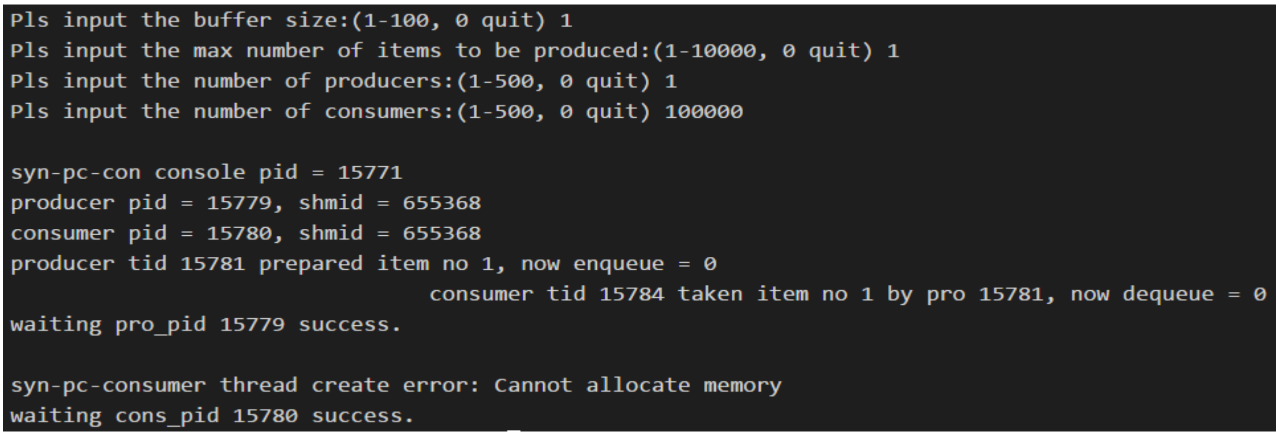
```
while (1) {
    printf("Pls input the buffer size:(1-100, 0 quit) ");
    scanf("%d", &buffer_size);
    if (buffer_size <= 0) return 0;
    if (buffer_size > 100) continue;
    printf("Pls input the max number of items to be produced:(1-10000, 0 quit) ");
    scanf("%d", &max_item_num);
    if (max_item_num <= 0) return 0;
    if (max_item_num > 10000) continue;
    printf("Pls input the number of producers:(1-500, 0 quit) ");
    scanf("%d", &thread_pro);
    if (thread_pro <= 0) return 0;
    if (thread_pro < 0) continue;
    printf("Pls input the number of consumers:(1-500, 0 quit) ");
    scanf("%d", &thread_cons);
    if (thread_cons <= 0) return 0;
    if (thread_cons < 0) continue;
    break;
}
```

3. 任何一个参数输入字符串死循环



```
Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer
 size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit
) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buff
er size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 qu
it) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the bu
ffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 
quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the 
buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 
0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input th
e buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100
, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input 
the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-1
00, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls inpu
t the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1
-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls in
put the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:
(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls 
input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer siz
e:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pl
s input the buffer size:(1-100, 0 quit) Pls input the buffer size:(1-100, 0 quit) Pls input the buffer s
```

4. 线程数输入10000没有判断超过边界，并且会出现内存错误



```
Pls input the buffer size:(1-100, 0 quit) 1
Pls input the max number of items to be produced:(1-10000, 0 quit) 1
Pls input the number of producers:(1-500, 0 quit) 1
Pls input the number of consumers:(1-500, 0 quit) 100000

syn-pc-con console pid = 15771
producer pid = 15779, shmid = 655368
consumer pid = 15780, shmid = 655368
producer tid 15781 prepared item no 1, now enqueue = 0
                                consumer tid 15784 taken item no 1 by pro 15781, now dequeue = 0
waiting pro_pid 15779 success.

syn-pc-consumer thread create error: Cannot allocate memory
waiting cons_pid 15780 success.
```

5. 功能说明中只有运行方法，并没有具体编译方法，可能出现用户编译时文件名写错导致无法运行程序的情况。

【技术日志】

奇怪的 (void *)-1

```
shm = shmat(shmid, 0, 0);
if (shm == (void *)-1) {
    perror("\nsyn-pc-consumer shmat failed");
    exit(EXIT_FAILURE);
}
```

(void *)-1 表示把 -1 转换为无类型指针 0xFFFFFFF，以前没有见过这样的条件判断，所以了解了一下用途。

shmat 函数的原型为:

```
void *shmat(int shm_id, const void *shm_addr, int shcmflg);
```

调用成功时返回一个指向共享内存第一个字节的指针，如果调用失败返回-1，类型为 void *，所以用 shm == (void *)-1 来判断 shmat 是否调用成功。