“生产者——消费者”问题是Linux多线程编程中的经典问题，主要是利用信号量处理线程间的同步和互斥问题。

“生产者——消费者”问题描述如下：

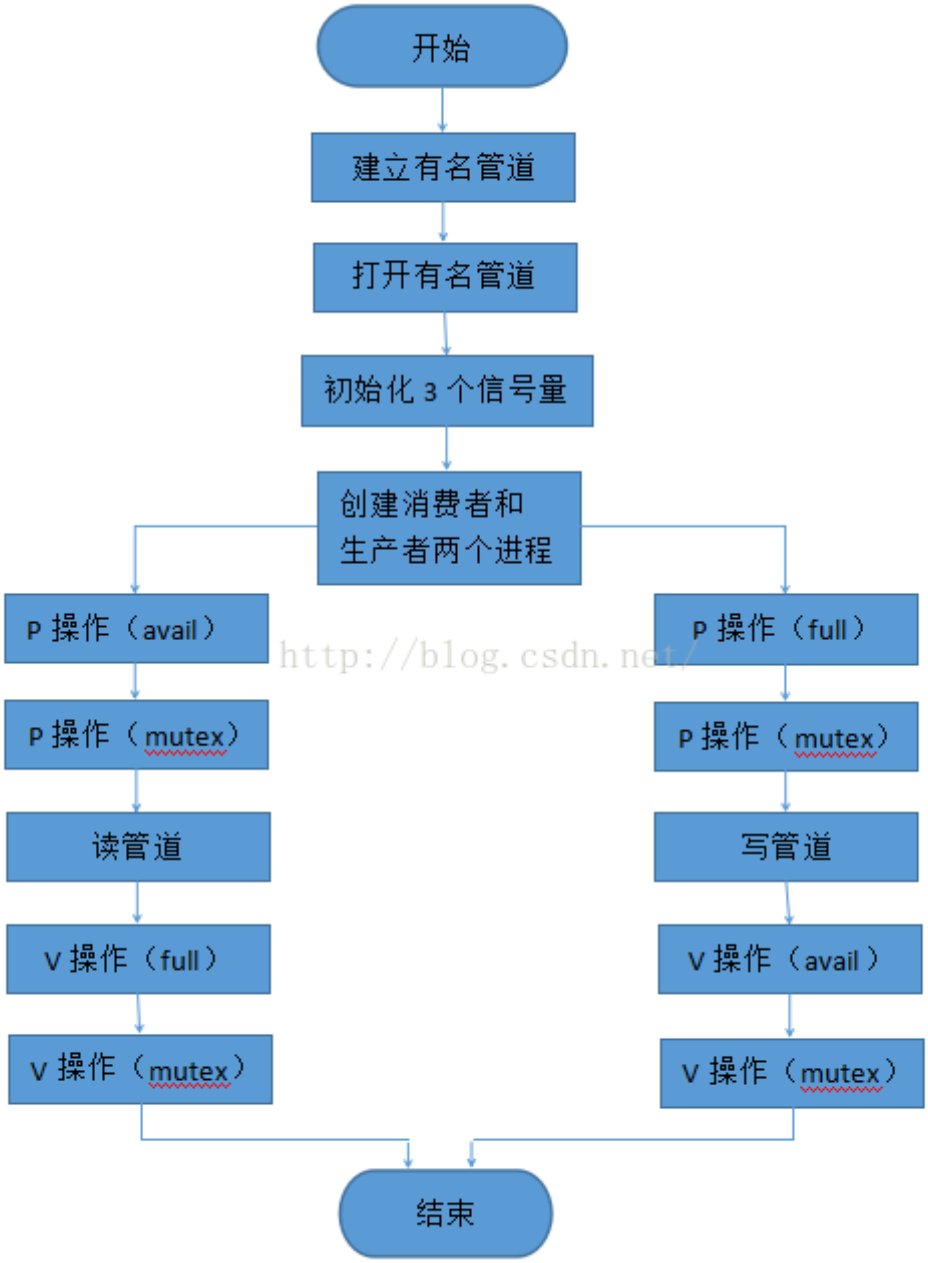
有一个有限缓冲区（这里用有名管道实现 FIFO 式缓冲区）和两个线程：生产者和消费者，它们分别不停地把产品放入缓冲区中拿走产品。一个生产者在缓冲区满的时候必须等待，一个消费者在缓冲区空的时候也不IXUS等待。另外，因为缓冲区是临界资源，所以生产者和消费者之间必须互斥进行。它们之间的关系如下：

这里要求使用有名管道来模拟有限缓冲区，并用信号量来解决“生产者——消费者”问题中的同步和互斥问题。

1、信号量分析

这里使用3个信号量，其中两个信号量 avail 和 full 分别用于解决生产者和消费者线程之间的互斥问题。其中avail 表示缓冲区的空单元数，初始值为N；full 表示缓冲区非空单元数，初始值为 0；mutex 是互斥信号量 ，初始值为 1(当然也可以用互斥锁来实现互斥操作)。

2、画出流程图



3、编写代码

本实验的代码中缓冲区拥有3个单元，每个单元为5个字节。为了尽量体现每个信号量的意义，在程序中生产过程和消费过程是随机（采取0~5s 的随机事件间隔）进行的，而且生产者的速度比消费者的速度平均快两倍左右。生产者一次生产一个单元的产品（放入hello字符串），消费者一次消费一个单元的产品。

```
1. #include
2. #include
3. #include
4. #include
5. #include
6. #include
```

```
7. #include
8. #include
9. #include
10. #include
11. #include
12. #define MYFIFO "myfifo"
13. #define BUFFER_SIZE 3
14. #define UNIT_SIZE 5
15. #define RUN_TIME 30
16. #define DELAY_TIME_LEVELS 5.0
17. void *producer(void *arg);
18. void *customer(void *arg);
19. int fd;
20. time_t end_time;
21. sem_t mutex,full,avail;
22. intmain()
23. {
24. int ret;
25. pthread_t thrd_prd_id,thrd_cst_id;
26. srand(time(NULL));
27. end_time = time(NULL) + RUN_TIME;
28. /*创建有名管道*/
29. if((mkfifo(MYFIFO,0644) < 0) && (errno != EEXIST))
30. {
31. perror("mkfifo error!");
32. exit(-1);
33. }
34. /*打开管道*/
35. fd = open(MYFIFO,O_RDWR);
36. if(fd == -1)
37. {
38. perror("open fifo error");
39. exit(-1);
40. }
41. /*初始化互斥信号量为1*/
42. ret = sem_init(&mutex,0,1);
```

```
43. /*初始化avail信号量为 N */
44. ret += sem_init(&avail,0,BUFFER_SIZE);
45. /*初始化full信号量为0*/
46. ret += sem_init(&full,0,0);
47. if(ret != 0)
48. {
49. perror("sem_init error");
50. exit(-1);
51. }
52. /*创建两个线程*/
53. ret = pthread_create(&thrd_prd_id,NULL,producer,NULL);
54. if(ret != 0)
55. {
56. perror("producer pthread_create error");
57. exit(-1);
58. }
59. ret = pthread_create(&thrd_cst_id,NULL,customer,NULL);
60. if(ret != 0)
61. {
62. perror("customer pthread_create error");
63. exit(-1);
64. }
65. pthread_join(thrd_prd_id,NULL);
66. pthread_join(thrd_cst_id,NULL);
67. close(fd);
68. unlink(MYFIFO);
69. return0;
70. }
71. void *producer(void *arg)//生产者线程
72. {
73. int real_write;
74. int delay_time;
75. while(time(NULL) < end_time)
76. {
77. delay_time = (int)(rand() * DELAY_TIME_LEVELS/RAND_MAX/2.0) + 1;
78. sleep(delay_time);
```

```
79. /*P操作信号量avail和mutex*/
80. sem_wait(&avail);
81. sem_wait(&mutex);
82. printf("\nproducer have delayed %d seconds\n",delay_time);
83. /*生产者写入数据*/
84. if((real_write = write(fd,"hello",UNIT_SIZE)) == -1)
85. {
86. if(errno == EAGAIN)
87. {
88. printf("The buffer is full,please wait for reading!\n");
89. }
90. }
91. else
92. {
93. printf("producer writes %d bytes to the FIFO\n",real_write);
94. printf("Now,the buffer left %d spaces!\n",avail);
95. }
96. /*V操作信号量full 和 mutex*/
97. sem_post(&full);
98. sem_post(&mutex);
99. }
100. pthread_exit(NULL);
101. }
102. void *customer(void *arg)//消费者线程
103. {
104. unsignedchar read_buffer[UNIT_SIZE];
105. int real_read;
106. int delay_time;
107. while(time(NULL) < end_time)
108. {
109. delay_time = (int)(rand() * DELAY_TIME_LEVELS/RAND_MAX/2.0) + 1;
110. sleep(delay_time);
111. sem_wait(&full); //P操作信号量full和mutex
112. sem_wait(&mutex);
113. memset(read_buffer,0,UNIT_SIZE);
114. printf("\nCustomer have delayed %d seconds\n",delay_time);
```

```
115. if((real_read = read(fd,read_buffer,UNIT_SIZE)) == -1)
116. {
117. if(errno == EAGAIN)
118. {
119. printf("The buffer is empty,please wait for writing!\n");
120. }
121. }
122. else
123. {
124. printf("customer reads %d bytes from the FIFO\n",real_read);
125. }
126. sem_post(&avail); //V操作信号量 avail 和 mutex
127. sem_post(&mutex);
128. }
129. pthread_exit(NULL);
130. }
```

执行结果如下：

```
1. fs@ubuntu:~/qiang/pthread$ ./cust_prod
2. producer have delayed 2 seconds
3. producer writes 5 bytes to the FIFO
4. Now,the buffer left 2 spaces!
5. Customer have delayed 2 seconds
6. customer reads 5 bytes from the FIFO
7. producer have delayed 2 seconds
8. producer writes 5 bytes to the FIFO
9. Now,the buffer left 2 spaces!
10. Customer have delayed 2 seconds
11. customer reads 5 bytes from the FIFO
12. producer have delayed 2 seconds
13. producer writes 5 bytes to the FIFO
14. Now,the buffer left 2 spaces!
15. Customer have delayed 2 seconds
16. customer reads 5 bytes from the FIFO
17. producer have delayed 1 seconds
18. producer writes 5 bytes to the FIFO
```

```
19. Now,the buffer left 2 spaces!
20. Customer have delayed 2 seconds
21. customer reads 5 bytes from the FIFO
22. producer have delayed 1 seconds
23. producer writes 5 bytes to the FIFO
24. Now,the buffer left 2 spaces!
25. Customer have delayed 3 seconds
26. customer reads 5 bytes from the FIFO
27. producer have delayed 3 seconds
28. producer writes 5 bytes to the FIFO
29. Now,the buffer left 2 spaces!
30. producer have delayed 1 seconds
31. producer writes 5 bytes to the FIFO
32. Now,the buffer left 1 spaces!
33. Customer have delayed 2 seconds
34. customer reads 5 bytes from the FIFO
35. Customer have delayed 1 seconds
36. customer reads 5 bytes from the FIFO
37. producer have delayed 3 seconds
38. producer writes 5 bytes to the FIFO
39. Now,the buffer left 2 spaces!
40. Customer have delayed 1 seconds
41. customer reads 5 bytes from the FIFO
42. producer have delayed 2 seconds
43. producer writes 5 bytes to the FIFO
44. Now,the buffer left 2 spaces!
45. Customer have delayed 2 seconds
46. customer reads 5 bytes from the FIFO
47. producer have delayed 1 seconds
48. producer writes 5 bytes to the FIFO
49. Now,the buffer left 2 spaces!
50. Customer have delayed 2 seconds
51. customer reads 5 bytes from the FIFO
52. producer have delayed 1 seconds
53. producer writes 5 bytes to the FIFO
54. Now,the buffer left 2 spaces!
```

```
producer have delayed 1 seconds
producer writes 5 bytes to the FIFO
Now,the buffer left 1 spaces!
Customer have delayed 2 seconds
customer reads 5 bytes from the FIFO
producer have delayed 2 seconds
producer writes 5 bytes to the FIFO
Now,the buffer left 1 spaces!
Customer have delayed 3 seconds
customer reads 5 bytes from the FIFO
Customer have delayed 1 seconds
customer reads 5 bytes from the FIFO
producer have delayed 3 seconds
producer writes 5 bytes to the FIFO
Now,the buffer left 2 spaces!
producer have delayed 1 seconds
producer writes 5 bytes to the FIFO
Now,the buffer left 1 spaces!
Customer have delayed 2 seconds
customer reads 5 bytes from the FIFO
Customer have delayed 1 seconds
customer reads 5 bytes from the FIFO
producer have delayed 3 seconds
producer writes 5 bytes to the FIFO
Now,the buffer left 2 spaces!
Customer have delayed 2 seconds
customer reads 5 bytes from the FIFO
producer have delayed 2 seconds
producer writes 5 bytes to the FIFO
Now,the buffer left 2 spaces!
fs@ubuntu:~/qiang/pthread$
```