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
4. Install GCC Software Colletion

>> sudo apt-get install build-essential

1. How to use GCC

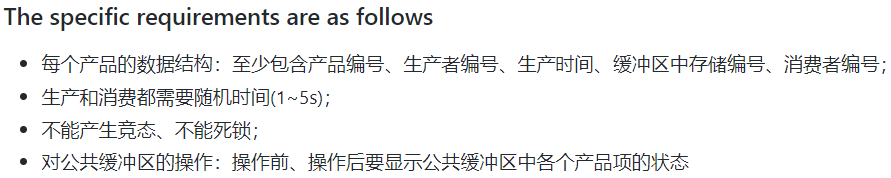
* [gcc and make](https://www3.ntu.edu.sg/home/ehchua/programming/cpp/gcc_make.html)

1. posix thread

#include <pthread.h>

pthread\_create()

1. 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）



* main.cpp代码：

1. #include <stdlib.h>
2. #include <pthread.h>
3. #include <unistd.h>
4. #include <stdio.h>
5. #include <semaphore.h> // 信号量 sem\_init、sem\_wait、sem\_post、sem\_destroy
6. #include <sys/time.h>
7. #include <time.h>
9. #define INIT\_NUM 0
10. #define TOTAL\_NUM 3
11. // #define PRO\_NUM 5
12. // #define CON\_NUM 4
13. #define PRODUCTS 15
15. // int gettimeofday(struct timeval \*tv,struct timezone \*tz); 生产时间
16. **struct** product
17. {
18. **int** id;            //产品编号
19. **int** pro\_id;  //生产者编号
20. **struct** timeval tv; //生产时间
21. **int** buf\_id;        //缓冲区储存编号
22. **int** con\_id;  //消费者编号
23. };
24. **void** printobj(**struct** product \*x)
25. {
26. 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",
27. x->id, x->pro\_id, x->tv.tv\_sec, x->tv.tv\_usec, x->buf\_id, x->con\_id);
28. }
29. **struct** product \*list[3] = {NULL, NULL, NULL};  //缓冲池
30. sem\_t p\_sem, c\_sem, sh\_sem;  //信号
32. **int** num = INIT\_NUM;
33. **int** pronum = 0;
34. **int** connum = 0;

37. **void** \*productor(**void** \*args)
38. {
39. **int** num = (**int**)(\*(**int** \*)args);
40. **while** (**true**)
41. {
42. **if** (connum >= PRODUCTS && pronum >= PRODUCTS)
43. exit(0);
44. **if** (pronum >= PRODUCTS)
45. **break**;
46. pthread\_t tid;
47. // tid = pthread\_self();
49. sem\_wait(&p\_sem); //可以生产信号减一,即剩余的容量
51. sleep(rand() % (5 - 1 + 1) + 1); //生产延时
53. **struct** timeval tt;
54. gettimeofday(&tt, 0);
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);
58. sem\_wait(&sh\_sem); //用来互斥的信号
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);
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);
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. }
102. sem\_post(&sh\_sem);
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);
107. sem\_post(&c\_sem); //消费者可以消费的信号量加一
108. }
109. **return** 0;
110. }
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
125. **struct** timeval tt;
126. gettimeofday(&tt, 0);
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);
130. sem\_wait(&sh\_sem);
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);
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. }
147. **for** (**int** i = 0; i < 3; i++)
148. {
149. **if** (list[i] != NULL)
150. {
151. list[i] = NULL;
152. **break**;
153. }
154. }
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
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. }
173. sem\_post(&sh\_sem);
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);
178. sem\_post(&p\_sem); //生产者生产信号量加一
179. }
180. **return** 0;
181. }
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];
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：  
  