

```

1: /*
2:    Windows 10
3:    Program Dev c ++
4:    The C Programming Language
5: */
6: #include <stdio.h>
7: #include <stdlib.h>
8: #include <unistd.h>
9: #include <pthread.h>
10: #include <time.h>
11: void *add_item();
12: void *remove_item();
13: void *append_buffer();
14: void *remove_buffer();
15: int i;
16: pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
17: int PRODUCERS, CONSUMERS, BUFFER_SIZE, REQUEST;
18: int buffer[100000]; //Buffer Size
19: int tail = 0, head = 0, request = 0, success = 0;
20: clock_t timer, timer1;
21:
22: //Define the main function
23: int main (int argc, char*argv)
24: {
25:     printf("PRODUCERS:\n") ;
26:     scanf("%d", &PRODUCERS);
27:     printf("Consumer:\n");
28:     scanf("%d", &CONSUMERS);
29:     printf("Buffer:\n");
30:     scanf("%d", &BUFFER_SIZE);
31:     printf("Request:\n");
32:     scanf("%d", &REQUEST);
33:
34:     timer = clock(); //time of cpu run
35:
36:     pthread_t thread_producer[PRODUCERS];
37:     pthread_t thread_consumer[CONSUMERS];
38:
39:     for( i=0; i<PRODUCERS; i++){
40:
41:         pthread_create(&thread_producer[i], NULL, append_buffer, NULL);
42:     }
43:     for(i=0; i<CONSUMERS; i++){
44:         pthread_create(&thread_consumer[i], NULL, remove_buffer, NULL);
45:     }
46:     for(i=0; i<CONSUMERS; i++){
47:         pthread_join(thread_consumer[i], NULL);
48:     }
49:     for(i=0; i<PRODUCERS; i++){
50:         pthread_join(thread_producer[i], NULL);
51:     }
52:     timer1 = clock(); //
53:     float elapsed = ((float)(timer1 - timer) / CLOCKS_PER_SEC); //Declare a translator to
    hold the elapsed numeric value.
54:

```

```

55:     printf("\n");
56:     printf("# buff %d %d %d %d\n", PRODUCERS , CONSUMERS , BUFFER_SIZE , REQUEST);
57:     printf("Producers %d, Consumers %d\n", PRODUCERS, CONSUMERS);
58:     printf("Buffer size %d\n", BUFFER_SIZE);
59:     printf("Requests %d\n\n", request);
60:     printf("Successfully consumed %d requests (%.1f%%)\n", success, (float)success * 100
/ request);
61:     printf("Elapsed Time %.2f s\n", elapsed);
62:     printf("Throughput %.2f successful requests/s\n", (float)(success) / elapsed);
63:
64:     exit(EXIT_SUCCESS);
65: }
66:
67: // function Add item
68: void *add_item()
69: {
70:     buffer[head++] = 1;
71:     head = head % BUFFER_SIZE;
72:     printf("Append Head %d Tail %d Buff %d\n", head , tail , buffer[head]);
73: }
74: // function Remove item
75: void *remove_item()
76: {
77:     buffer[tail++] = 0;
78:     tail = tail % BUFFER_SIZE;
79:     printf("Remove Head %d Tail %d Buff %d\n", head , tail , buffer[tail]);
80: }
81: // function Append in buffer item
82: void *append_buffer()
83: {
84:     printf("Append thread number %ld\n", pthread_self());
85:
86:     while(request<REQUEST) {
87:         if(!pthread_mutex_trylock(&mutex) && request<REQUEST) {
88:             if(buffer[head] == 0) {
89:                 add_item();
90:                 request++;
91:                 printf(" + thread %ld append success\n", pthread_self());
92:             }
93:             else {
94:                 printf("Buffer overflow\n");
95:             }
96:             pthread_mutex_unlock(&mutex);
97:         }
98:     }
99:     pthread_exit(NULL);
100: }
101:
102:
103: // function Remove in buffer item
104: void *remove_buffer()
105: {
106:     printf("Remove thread number %ld\n", pthread_self());
107:
108:     while(success<REQUEST) {

```

```
109:         if(!pthread_mutex_trylock(&mutex) && success<REQUEST) {
110:             if(buffer[tail] == 1) {
111:                 remove_item();
112:                 success++;
113:                 printf(" - thread %ld remove success\n", pthread_self());
114:             }
115:             else {
116:                 printf("Buffer underflow\n");
117:             }
118:             pthread_mutex_unlock(&mutex);
119:         }
120:     }
121:     pthread_exit(NULL);
122: }
123:
```