

```
1  PGMS=Producer_Consumer_1 Producer_Consumer_2 Producer_Consumer_3
2
3  CC=gcc
4  CFLAGS=-Wall -pthread
5
6  all: $(PGMS)
7
8  clean:
9      rm -f $(PGMS)
10     rm -f *.o
end
```

```

1 // Empty/Full with no lock
2
3 // Note: All pthread and sem functions should have their return codes
4 //       checked. The checking has been omitted to clarity in
5 //       this example.
6
7 #include <assert.h>
8 #include <pthread.h>
9 #include <semaphore.h>
10 #include <stdio.h>
11 #include <unistd.h>
12
13 #define MAX 3
14
15 int buffer[MAX];
16 int fill = 0;
17 int use = 0;
18 int loops = 20;
19
20 sem_t empty;
21 sem_t full;
22
23 void put(int value)
24 {
25     buffer[fill] = value;
26     fill = (fill + 1) % MAX;
27 }
28
29 int get()
30 {
31     int tmp = buffer[use];
32     use = (use + 1) % MAX;
33
34     return tmp;
35 }
36
37 void *producer(void *arg)
38 {
39     int val = (long)arg;
40     for (int i = 0; i < loops; i++) {
41         sem_wait(&empty);
42         put(i + val);
43         sem_post(&full);
44     }
45     return NULL;
46 }
47
48 void *consumer(void *arg)
49 {
50
51     //for (int i = 0; i < loops * 2; i++) {
52     for (int i = 0; i < loops; i++) {
53         sem_wait(&full);
54         int tmp = get();
55         sem_post(&empty);
56         printf("%d\n", tmp);
57     }
58     return NULL;
59 }
60
61 int main()
62 {

```

```

63     pthread_t p1;
64     // pthread_t p2;
65     pthread_t c1;
66
67     sem_init(&empty, 0, MAX);
68     sem_init(&full, 0, 0);
69
70     assert(pthread_create(&p1, NULL, producer, (void*)100) == 0);
71     // assert(pthread_create(&p2, NULL, producer, (void*)200) == 0);
72     assert(pthread_create(&c1, NULL, consumer, NULL) == 0);
73
74     assert(pthread_join(p1, NULL) == 0);
75     // assert(pthread_join(p2, NULL) == 0);
76     assert(pthread_join(c1, NULL) == 0);
77
78     return 0;
79 }
end

```

```

1  // Version 2 - Add mutex before empty/full
2
3  // Note: All pthread and sem functions should have their return codes
4  //       checked. The checking has been omitted to clarity in
5  //       this example.
6
7  #include <assert.h>
8  #include <pthread.h>
9  #include <semaphore.h>
10 #include <stdio.h>
11
12 #define MAX 10
13
14 int buffer[MAX];
15 int fill = 0;
16 int use = 0;
17 int loops = 20;
18
19 sem_t empty;
20 sem_t full;
21 sem_t mutex;
22
23 void put(int value)
24 {
25     buffer[fill] = value;
26     fill = (fill + 1) % MAX;
27 }
28
29 int get()
30 {
31     int tmp = buffer[use];
32     use = (use + 1) % MAX;
33     return tmp;
34 }
35
36 void *producer(void *arg)
37 {
38     int val = (long)arg;
39
40     for (int i = 0; i < loops; i++) {
41
42         sem_wait(&mutex);
43         sem_wait(&empty);
44
45         put(i + val);
46
47         sem_post(&full);
48         sem_post(&mutex);
49     }
50     return NULL;
51 }
52
53 void *consumer(void *arg)
54 {
55     for (int i = 0; i < loops; i++) {
56
57         sem_wait(&mutex);
58         sem_wait(&full);
59
60         int tmp = get();
61
62     }

```

```

63         sem_post(&empty);
64         sem_post(&mutex);
65
66         printf("%d\n", tmp);
67     }
68     return NULL;
69 }
70
71 int main()
72 {
73     pthread_t p1;
74     pthread_t p2;
75     pthread_t c1;
76
77     sem_init(&mutex, 0, 1);
78     sem_init(&empty, 0, MAX);
79     sem_init(&full, 0, 0);
80
81     assert(pthread_create(&p1, NULL, producer, (void*)100) == 0);
82     assert(pthread_create(&p2, NULL, producer, (void*)200) == 0);
83     assert(pthread_create(&c1, NULL, consumer, NULL) == 0);
84
85     assert(pthread_join(p1, NULL) == 0);
86     assert(pthread_join(p2, NULL) == 0);
87     assert(pthread_join(c1, NULL) == 0);
88
89     return 0;
90 }
end

```

```

1  // Version 3 - Add empty/full before mutex
2
3  // Note: All pthread and sem functions should have their return codes
4  //       checked. The checking has been omitted to clarity in
5  //       this example.
6
7  #include <assert.h>
8  #include <pthread.h>
9  #include <semaphore.h>
10 #include <stdio.h>
11
12 #define MAX 3
13
14 int buffer[MAX];
15 int fill = 0;
16 int use = 0;
17 int loops = 20;
18
19 sem_t empty;
20 sem_t full;
21 sem_t mutex;
22
23 void put(int value)
24 {
25     buffer[fill] = value;
26     fill = (fill + 1) % MAX;
27 }
28
29 int get()
30 {
31     int tmp = buffer[use];
32     use = (use + 1) % MAX;
33     return tmp;
34 }
35
36 void *producer(void *arg)
37 {
38     int val = (long)arg;
39     for (int i = 0; i < loops; i++) {
40
41         sem_wait(&empty);
42         sem_wait(&mutex);
43
44         put(i + val);
45
46         sem_post(&mutex);
47         sem_post(&full);
48     }
49     return NULL;
50 }
51
52 void *consumer(void *arg)
53 {
54     for (int i = 0; i < (loops*2); i++) {
55
56         sem_wait(&full);
57         sem_wait(&mutex);
58
59         int tmp = get();
60
61         sem_post(&mutex);

```

```

63         sem_post(&empty);
64
65         printf("%d\n", tmp);
66     }
67     return NULL;
68 }
69
70 int main()
71 {
72     pthread_t p1;
73     pthread_t p2;
74     pthread_t c1;
75
76     sem_init(&mutex, 0, 1);
77     sem_init(&empty, 0, MAX);
78     sem_init(&full, 0, 0);
79
80     assert(pthread_create(&p1, NULL, producer, (void*)100) == 0);
81     assert(pthread_create(&p2, NULL, producer, (void*)200) == 0);
82     assert(pthread_create(&c1, NULL, consumer, NULL) == 0);
83
84     assert(pthread_join(p1, NULL) == 0);
85     assert(pthread_join(p2, NULL) == 0);
86     assert(pthread_join(c1, NULL) == 0);
87
88     return 0;
89 }
90
end

```