```
import threading
import time
import random

buffer = []
buffer_size = 10
lock = threading.Lock()
empty = threading.Semaphore(buffer_size)
full = threading.Semaphore(0)

def producer(producer_id):
    while True:
        item = random.randint(1, 100)
        empty.acquire()
        with lock:
            buffer.append(item)
            print(f"Producer {producer_id} produced: {item}")
        full.release()
        time.sleep(random.uniform(0.5, 2))

def consumer(consumer_id):
    while True:
        full.acquire()
        with lock:
            item = buffer.pop(0)
            print(f"Consumer {consumer_id} consumed: {item}")
        empty.release()
        time.sleep(random.uniform(0.5, 2))

if __name__ == "__main__":
    producers = [threading.Thread(target=producer, args=(i,)) for i in range(2)]
```

```
    consumers = [threading.Thread(target=consumer, args=(i,)) for i in range(2)]

    for p in producers:
        p.start()
    for c in consumers:
        c.start()

    for p in producers:
        p.join()
    for c in consumers:
        c.join()
```

OUTPUT:

```
Producer 0 produced: 42
Consumer 0 consumed: 42
Producer 1 produced: 17
Consumer 1 consumed: 17
Producer 0 produced: 89
Consumer 0 consumed: 89
...
```