Seguimiento productor consumidor con semáforos

Juan Felipe Muñoz Fernández

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:    elemento = producir_elemento();
11:    wait(&vacias);
12:    wait(&mutex);
13:    insertar_elemento(elemento);
14:    signal(&mutex);
15:    signal(&llenas);
16: }
17:  }
```

mutex

búfer

vacías

elemento

llenas

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:    elemento = producir_elemento();
11:    wait(&vacias);
12:    wait(&mutex);
13:    insertar_elemento(elemento);
14:    signal(&mutex);
15:    signal(&llenas);
16: }
17:  }
```

mutex

1

vacías

llenas

búfer

elemento

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:    elemento = producir_elemento();
11:    wait(&vacias);
12:    wait(&mutex);
13:    insertar_elemento(elemento);
14:    signal(&mutex);
15:    signal(&llenas);
16: }
17:  }
```

mutex

1

vacías

3

llenas

búfer

elemento

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:   }
```

mutex	búfer
1	
vacías	elemento
3	
llenas	
0	

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:    elemento = producir_elemento();
11:    wait(&vacias);
12:    wait(&mutex);
13:    insertar_elemento(elemento);
14:    signal(&mutex);
15:    signal(&llenas);
16: }
17:  }
```

mutex	búfer
1	
vacías	elemento
3	100
llenas	
0	

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:   }
```

mutex	búfer
1	
vacías	elemento
2	100
llenas	
0	

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:   }
```

mutex	búfer
0	
vacías	elemento
2	100
llenas	
0	

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

mutex	búfer
0	100

vacías	elemento
2	100

llenas
0

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

mutex	búfer
1	100

vacías	elemento
2	100

llenas
0

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

mutex	búfer
1	100
vacías	**elemento**
2	100
llenas	
1	

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:    elemento = producir_elemento();
11:    wait(&vacias);
12:    wait(&mutex);
13:    insertar_elemento(elemento);
14:    signal(&mutex);
15:    signal(&llenas);
16: }
17:  }
```

mutex	búfer
1	100
vacías	elemento
2	101
llenas	
1	

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:   }
```

mutex	búfer
1	100
vacías	elemento
1	101
llenas	
1	

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:   }
```

mutex	búfer
0	100
vacías	elemento
1	101
llenas	
1	

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

mutex	búfer
0	100,101

vacías	elemento
1	101

llenas
1

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

mutex	búfer
1	100,101

vacías	elemento
1	101

llenas
1

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:   }
```

mutex	búfer
1	100,101
vacías	**elemento**
1	101
llenas	
2	

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:    elemento = producir_elemento();
11:    wait(&vacias);
12:    wait(&mutex);
13:    insertar_elemento(elemento);
14:    signal(&mutex);
15:    signal(&llenas);
16: }
17:  }
```

mutex	búfer
1	100,101
vacías	**elemento**
1	102
llenas	
2	

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

mutex	búfer
1	100,101
vacías	elemento
0	102
llenas	
2	

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:   }
```

mutex	búfer
0	100,101
vacías	elemento
0	102
llenas	
2	

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

mutex	búfer
0	100,101,102
vacías	**elemento**
0	102
llenas	
2	

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

mutex	búfer
1	100,101,102
vacías	elemento
0	102
llenas	
2	

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:   }
```

mutex	búfer
1	100,101,102
vacías	**elemento**
0	102
llenas	
3	

Otro ciclo de producción…

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:    elemento = producir_elemento();
11:    wait(&vacias);
12:    wait(&mutex);
13:    insertar_elemento(elemento);
14:    signal(&mutex);
15:    signal(&llenas);
16: }
17:  }
```

mutex	búfer
1	100,101,102
vacías	**elemento**
0	104
llenas	
3	

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

mutex	búfer
1	100,101,102
vacías	elemento
-1	104
llenas	
3	

- Se bloquea productor en línea 11 en el semáforo vacías
- Siguiente línea a ejecutar del productor: línea 12
- Se ejecuta consumidor

```
1:   void consumidor(void)
2:   {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10:     consumir_elemento(elemento);
11: }
12:   }
```

mutex	búfer
1	100,101,102

vacías	elemento
-1	

llenas
3

```
1:   void consumidor(void)
2:   {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10:     consumir_elemento(elemento);
11: }
12:   }
```

mutex	búfer
1	100,101,102

vacías	elemento
-1	

llenas
2

```
1:   void consumidor(void)
2:   {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10:     consumir_elemento(elemento);
11: }
12:   }
```

mutex	búfer
0	100,101,102

vacías	elemento
-1	

llenas
2

```
1:   void consumidor(void)
2:   {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10:     consumir_elemento(elemento);
11: }
12:   }
```

mutex	búfer
0	101,102

vacías	elemento
-1	100

llenas
2

```
1:   void consumidor(void)
2:   {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10:     consumir_elemento(elemento);
11: }
12:   }
```

mutex	búfer
1	101,102

vacías	elemento
-1	100

llenas
2

```
1:   void consumidor(void)
2:   {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10:     consumir_elemento(elemento);
11: }
12:   }
```

mutex	búfer
1	101,102

vacías	elemento
0	100

llenas
2

- Se desbloquea proceso productor porque se bloqueó en el semáforo vacías
- ¿Se ejecuta?

```
1:   void consumidor(void)
2:   {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10:     consumir_elemento(elemento);
11: }
12:   }
```

mutex	búfer
1	101,102

vacías	elemento
0	100

llenas
1

```
1:   void consumidor(void)
2:   {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10:     consumir_elemento(elemento);
11: }
12:   }
```

mutex	búfer
0	101,102

vacías	elemento
0	100

llenas
1

```
1:   void consumidor(void)
2:   {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10:     consumir_elemento(elemento);
11: }
12:   }
```

mutex	búfer
0	102

vacías	elemento
0	101

llenas
1

```
1:   void consumidor(void)
2:   {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10:     consumir_elemento(elemento);
11: }
12:   }
```

mutex	búfer
1	102

vacías	elemento
0	101

llenas
1

```
1:   void consumidor(void)
2:   {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10:     consumir_elemento(elemento);
11: }
12:   }
```

mutex	búfer
1	102

vacías	elemento
1	101

llenas
1

```
1:   void consumidor(void)
2:   {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10:     consumir_elemento(elemento);
11: }
12:   }
```

mutex	búfer
1	102

vacías	elemento
1	101

llenas
0

```
1:   void consumidor(void)
2:   {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10:     consumir_elemento(elemento);
11: }
12:   }
```

mutex	búfer
0	102

vacías	elemento
1	101

llenas
0

```
1:   void consumidor(void)
2:   {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10:     consumir_elemento(elemento);
11: }
12:   }
```

mutex	búfer
0	

vacías	elemento
1	102

llenas
0

```
1:   void consumidor(void)
2:   {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10:     consumir_elemento(elemento);
11: }
12:   }
```

mutex	búfer
1	

vacías	elemento
1	102

llenas
0

```
1:   void consumidor(void)
2:   {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10:     consumir_elemento(elemento);
11: }
12:   }
```

mutex	búfer
1	
vacías	elemento
2	102
llenas	
0	

- Otro ciclo de consumo…(for fun and profit ☺️)

```
1:   void consumidor(void)
2:   {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10:     consumir_elemento(elemento);
11: }
12:   }
```

mutex	búfer
1	
vacías	elemento
2	102
llenas	
-1	

- Se bloquea consumidor en línea 5, en el semáforo llenas
- Siguiente línea a ejecutarse en consumidor: línea 6
- Se ejecuta productor que había quedado bloqueado en línea 11

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:   }
```

mutex	búfer
1	
vacías	elemento
2	104
llenas	
-1	

- Se ejecuta productor
- Siguiente línea a ejecutarse: 12
- Se restaura su estado: variable local elemento que al momento de bloquearse tenía el valor 104

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

mutex	búfer
0	
vacías	elemento
2	104
llenas	
-1	

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

mutex	búfer
0	104

vacías	elemento
2	104

llenas
-1

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

mutex	búfer
1	104

vacías	elemento
2	104

llenas
-1

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

mutex	búfer
1	104
vacías	**elemento**
2	104
llenas	
0	

- Se desbloquea proceso consumidor en semáforo llenas
- ¿Se ejecuta?

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:    elemento = producir_elemento();
11:    wait(&vacias);
12:    wait(&mutex);
13:    insertar_elemento(elemento);
14:    signal(&mutex);
15:    signal(&llenas);
16: }
17:  }
```

mutex	búfer
1	104
vacías	elemento
2	105
llenas	
0	

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

mutex	búfer
1	104
vacías	elemento
1	105
llenas	
0	

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:   }
```

mutex	búfer
0	104

vacías	elemento
1	105

llenas
0

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

mutex	búfer
0	104,105

vacías	elemento
1	105

llenas
0

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

mutex	búfer
1	104,105

vacías	elemento
1	105

llenas
0

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:   }
```

mutex	búfer
1	104,105
vacías	**elemento**
1	105
llenas	
1	

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:    elemento = producir_elemento();
11:    wait(&vacias);
12:    wait(&mutex);
13:    insertar_elemento(elemento);
14:    signal(&mutex);
15:    signal(&llenas);
16: }
17:  }
```

mutex	búfer
1	104,105
vacías	**elemento**
1	106
llenas	
1	

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

mutex	búfer
1	104,105
vacías	elemento
0	106
llenas	
1	

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

mutex	búfer
0	104,105
vacías	elemento
0	106
llenas	
1	

```
1:   #define N 3
2:   typedef int semaforo;
3:   semaforo mutex = 1;
4:   semaforo vacias = N;
5:   semaforo llenas = 0;
6:   void productor(void)
7:   {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:   }
```

mutex	búfer
0	104,105,106

vacías	elemento
0	106

llenas
1

- Supongamos que este proceso/hilo sale del procesador en este punto.
- ¿Se ejecuta consumidor?
- Podría pero su siguiente instrucción es **wait(&mutex)** lo que hace que se bloquee en ese semáforo
- Continuemos con la ejecución del productor

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

mutex	búfer
1	104,105,106

vacías	elemento
0	106

llenas
1

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

mutex	búfer
1	104,105,106
vacías	elemento
0	106
llenas	
2	

- Otro ciclo de producción…

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8: int elemento;
9: while(TRUE){
10:    elemento = producir_elemento();
11:    wait(&vacias);
12:    wait(&mutex);
13:    insertar_elemento(elemento);
14:    signal(&mutex);
15:    signal(&llenas);
16: }
17:  }
```

mutex	búfer
1	104,105,106
vacías	elemento
0	107
llenas	
2	

```
1:   #define N 3
2:   typedef int semaforo;
3:   semaforo mutex = 1;
4:   semaforo vacias = N;
5:   semaforo llenas = 0;
6:   void productor(void)
7:   {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:   }
```

mutex	búfer
1	104,105,106
vacías	elemento
-1	107
llenas	
2	

- Se bloquea productor en línea 11 en el semáforo vacías.
- Siguiente línea a ejecutar del productor: línea 12

Seguimiento paso a paso

En ejecución paralela: atentos(a)s

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

elemento

```
1:  void consumidor(void)
2:  {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10: }
11:  }
```

elemento

- Mutex, vacías y llenas con datos ya que hacen parte del programa principal y son globales a los dos procesos/hilos
- Cada proceso/hilo tiene su variable local **elemento**.

mutex

1

vacías

3

llenas

0

búfer

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

elemento

101

```
1:  void consumidor(void)
2:  {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10: }
11:  }
```

elemento

mutex

1

vacías

3

llenas

-1

búfer

- Se bloquea consumidor en línea 5 en el semáforo llenas.
- Siguiente línea a ejecutar en consumodor: línea 6

Consumidor

Productor

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

elemento

101

```
1:  void consumidor(void)
2:  {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10: }
11:  }
```

elemento

mutex

1

búfer

vacías

2

llenas

-1

Consumidor está bloqueado: no avanza

Consumidor

Productor

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

elemento

101

```
1:  void consumidor(void)
2:  {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10: }
11:  }
```

elemento

mutex

0

vacías

2

llenas

-1

búfer

- Consumidor está bloqueado: no avanza

Consumidor

Productor

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

elemento

101

```
1:  void consumidor(void)
2:  {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10: }
11:  }
```

elemento

mutex

0

búfer

101

vacías

2

llenas

-1

- Consumidor está bloqueado: no avanza

Consumidor

Productor

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

elemento

101

```
1:  void consumidor(void)
2:  {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10: }
11:  }
```

elemento

mutex

1

búfer

101

vacías

2

llenas

-1

- Consumidor está bloqueado: no avanza

Consumidor

Productor

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

elemento

101

```
1:  void consumidor(void)
2:  {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10: }
11:  }
```

elemento

mutex

0

búfer

101

vacías

2

llenas

0

- Se desbloquea consumidor y sigue su ejecución inmediatamente.
- Consumidor estaba bloqueado en el semáforo llenas.
- Siguiente línea a ejecutar en consumidor: línea 6

Consumidor

Productor

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

elemento

102

```
1:  void consumidor(void)
2:  {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10: }
11:  }
```

elemento

101

mutex

0

vacías

2

llenas

0

búfer

101

Consumidor

Productor

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

elemento

102

```
1:  void consumidor(void)
2:  {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10: }
11:  }
```

elemento

101

mutex

1

vacías

0

llenas

0

búfer

Consumidor

Productor

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

elemento

102

```
1:  void consumidor(void)
2:  {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10: }
11:  }
```

elemento

101

mutex

0

vacías

2

llenas

0

búfer

Consumidor

Productor

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

elemento

102

```
1:  void consumidor(void)
2:  {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10: }
11:  }
```

elemento

101

mutex

0

búfer

102

vacías

2

llenas

-1

- Se bloquea consumidor en semáforo llenas

Consumidor

Productor

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

elemento

102

```
1:  void consumidor(void)
2:  {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10: }
11:  }
```

elemento

101

mutex

1

búfer

102

vacías

2

llenas

-1

- Consumidor sigue bloqueado

Consumidor

Productor

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

elemento

102

```
1:  void consumidor(void)
2:  {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10: }
11:  }
```

elemento

101

mutex

0

búfer

102

vacías

2

llenas

0

- Se desbloquea consumidor y reanuda ejecución

Consumidor

Productor

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

elemento

103

```
1:  void consumidor(void)
2:  {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10: }
11:  }
```

elemento

102

mutex

0

vacías

2

llenas

0

búfer

Consumidor

Productor

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

elemento

103

```
1:  void consumidor(void)
2:  {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10: }
11:  }
```

elemento

102

mutex

1

vacías

1

llenas

0

búfer

Consumidor

Productor

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

elemento

103

```
1:  void consumidor(void)
2:  {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10: }
11:  }
```

elemento

102

mutex

0

vacías

2

llenas

0

búfer

Consumidor

Productor

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

elemento

103

```
1:  void consumidor(void)
2:  {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10: }
11:  }
```

elemento

102

mutex

0

búfer

103

vacías

2

llenas

-1

Consumidor

Productor

- Se bloquea consumidor

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

elemento

103

```
1:  void consumidor(void)
2:  {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10: }
11:  }
```

elemento

102

mutex

1

búfer

103

vacías

2

llenas

-1

- Consumidor sigue bloqueado

Consumidor

Productor

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

elemento

103

```
1:  void consumidor(void)
2:  {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10: }
11:  }
```

elemento

102

mutex

0

búfer

103

vacías

2

llenas

0

- Se desbloquea consumidor

Consumidor

Productor

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

elemento

104

```
1:  void consumidor(void)
2:  {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10: }
11:  }
```

elemento

103

mutex

0

vacías

2

llenas

0

búfer

Consumidor

Productor

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

elemento

104

```
1:  void consumidor(void)
2:  {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10: }
11:  }
```

elemento

103

mutex

1

vacías

1

llenas

0

búfer

Consumidor

Productor

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

elemento

104

```
1:  void consumidor(void)
2:  {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10: }
11:  }
```

elemento

103

mutex

0

vacías

2

llenas

0

búfer

Consumidor

Productor

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

elemento

104

```
1:  void consumidor(void)
2:  {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10: }
11:  }
```

elemento

103

mutex

0

búfer

104

vacías

2

llenas

-1

- Se bloquea consumidor

Consumidor

Productor

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

elemento

104

```
1:  void consumidor(void)
2:  {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10: }
11:  }
```

elemento

103

mutex

1

búfer

104

vacías

2

llenas

-1

- Consumidor sigue bloqueado

Consumidor

Productor

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

elemento

104

```
1:  void consumidor(void)
2:  {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10: }
11:  }
```

elemento

103

mutex

0

búfer

104

vacías

2

llenas

0

- Se desbloquea consumidor

Consumidor

Productor

```
1:  #define N 3
2:  typedef int semaforo;
3:  semaforo mutex = 1;
4:  semaforo vacias = N;
5:  semaforo llenas = 0;
6:  void productor(void)
7:  {
8:  int elemento;
9:  while(TRUE){
10:     elemento = producir_elemento();
11:     wait(&vacias);
12:     wait(&mutex);
13:     insertar_elemento(elemento);
14:     signal(&mutex);
15:     signal(&llenas);
16: }
17:  }
```

elemento

105

```
1:  void consumidor(void)
2:  {
3:      int elemento;
4:  while(TRUE){
5:      wait(&llenas);
6:      wait(&mutex);
7:      elemento = quitar_elemento();
8:      signal(&mutex);
9:      signal(&vacias);
10: }
11:  }
```

elemento

104

mutex

0

vacías

2

llenas

0

búfer

Consumidor

Productor

- Y así sucesivamente por los siglos de los siglos…