Producer/Consumer Problem

- One or more producers are generating data and placing these in a buffer
- A single consumer is taking items out of the buffer one at time
- Only one producer or consumer may access the buffer at any one time

```
Producer

producer:
while (true) {
    /* produce item v */
    b[in] = v;
    in++;
}
```
Consumer

consumer:
while (true) {
    while (in <= out)
        /* do nothing */;
    w = b[out];
    out++;
    /* consume item w */
}

Producer/Consumer Problem

![Buffer Diagram](image)

Note: shaded area indicates portion of buffer that is occupied

Figure 5.8 Infinite Buffer for the Producer/Consumer Problem
Producer with Circular Buffer

producer:
while (true) {
    /* produce item v */
    while ( ((in + 1) % n == out) )
        /* do nothing */;
    b[in] = v;
    in = (in + 1) % n
}

Consumer with Circular Buffer

consumer:
while (true) {
    while (in == out)  
        /* do nothing */;
    w = b[out];
    out = (out + 1) % n;
    /* consume item w */
}

/* program producerconsumer */
int n;
Binary_semaphore s = 1;
Binary_semaphore delay = 0;
void producer()
{
    while (true)
    {
        produce();
        semWaitB(s);
        append();
        n++;
        if (n==1)
            semSignalB(delay);
        semSignalB(s);
    }
}
void consumer()
{
    semWaitB(delay);
    while (true)
    {
        semWaitB(s);
        take();
        o--;
        semSignalB(s);
        consume();
        if (n==0)
            semWaitB(delay);
    }
}
void main()
{
    n = 0;
    parbegin (producer, consumer);
}

Figure 5.9 An Incorrect Solution to the Infinite-Buffer Producer/Consumer Problem Using Binary Semaphores

Figure 5.12 Finite Circular Buffer for the Producer/Consumer Problem
Figure 5.10 A Correct Solution to the Infinite-Buffer Producer/Consumer Problem Using Binary Semaphores

```c
/* program producerconsumer */
int n;
binary_semaphore x = 1;
binary_semaphore delay = 0;
void producer()
{
    while (true)
    {
        produce();
        semWait(x);
        append();
        if (n==1) semSignal(delay);
        semSignal(x);
    }
}
void consumer()
{
    int n; /* a local variable */
    semWait(delay);
    while (true)
    {
        semWait(x);
        take();
        n--;
        if (n>0) semWait(x);
        consume();
        if (n==0) semWait(delay);
    }
}
void main()
{
    n = 0;
    parbegin (producer, consumer);
}
```

Figure 5.11 A Solution to the Infinite-Buffer Producer/Consumer Problem Using Semaphores

```c
/* program producerconsumer */
semaphore n = 0;
semaphore x = 1;
void producer()
{
    while (true)
    {
        produce();
        semWait(x);
        append();
        semSignal(x);
        semSignal(n);
    }
}
void consumer()
{
    while (true)
    {
        semWait(n);
        semWait(x);
        take();
        semSignal(x);
        consume();
    }
}
void main()
{
    parbegin (producer, consumer);
}
```
/* program boundedbuffer */
const int sizeofbuffer = /* buffer size */;
semaphore e = 1;
semaphore n = 0;
semaphore m = sizeofbuffer;
void producer()
{
    while (true)
    {
        produce();
        semWait(e);
        semWait(e);
        append();
        semSignal[n];
        semSignal[n]
    }
}
void consumer()
{
    while (true)
    {
        semWait(n);
        semWait(e);
        take();
        semSignal[s];
        semSignal[e];
        consume();
    }
}
void main()
{
    parbegin (producer, consumer);
}