An example: bounded buffer

Suppose one process is creating information that is going to be used by another process, e.g., suppose one process reads information from the disk, and another compiles that information from source to machine code.

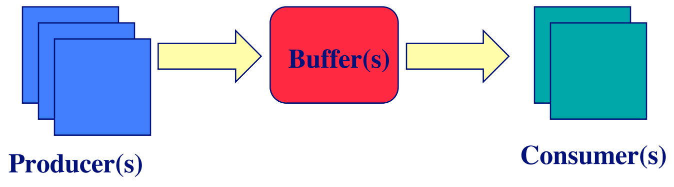
Processes should *not* have to operate in strict alteration: producer should be able to get ahead of consumer.

Why?

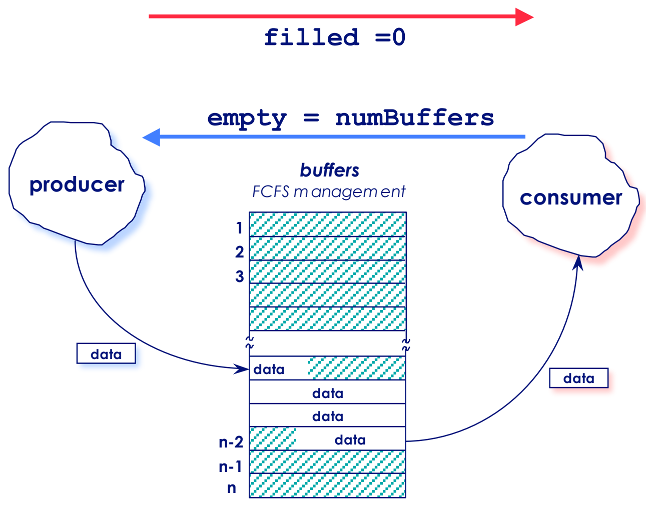
(Chapter 2.3.4 and 2.3.5, Tanenbaum)

Bounded buffer—details

- *Producer*: creates copies of a resource.
- *Consumer*: uses up copies of a resource.
- *Buffers*: used to hold information after producer has created it but before consumer has used it.
- *Signaling*: keeping control of producer and consumer (e.g., preventing overrun of the producer).
- *Constraints* (definition of what is “correct”).



Bounded buffer—scenario



Also known as a “circular buffer”

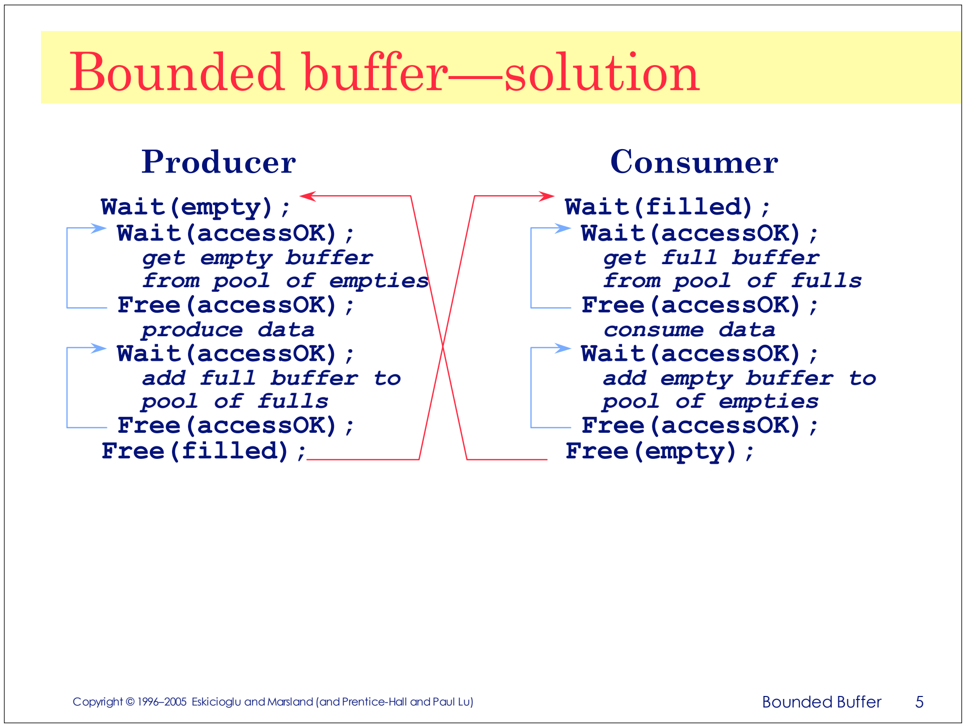
Bounded buffer—constraints

The constraints must be defined *before* coding:

- Consumer *must* wait for a producer to fill buffers. (*signaling*)
- Producer *must* wait for consumer to empty buffers, when all buffer space is in use. (*signaling*)
- Only one process *must* manipulate buffer pool at once. (*mutual exclusion*)

A separate semaphore is used for each constraint. They are initialized as follows:

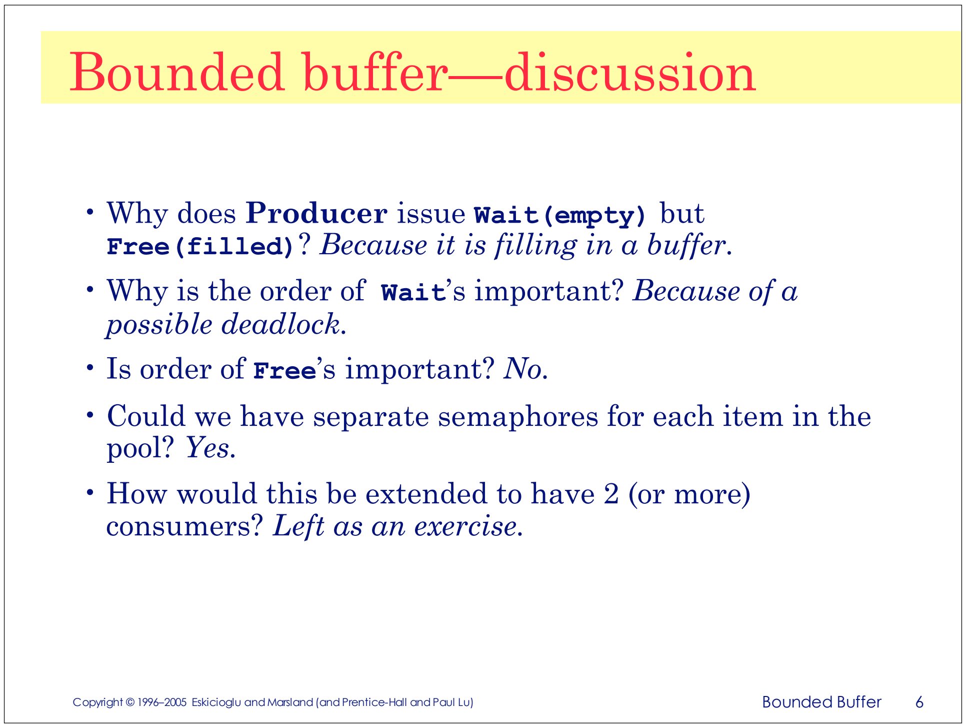
```
empty = numBuffers;    filled = 0;    accessOK = 1;
```



Bounded buffer—solution

```
Producer                          Consumer

Wait(empty);                      Wait(filled);
  Wait(accessOK);                   Wait(accessOK);
    get empty buffer                  get full buffer
    from pool of empties              from pool of fulls
  Free(accessOK);                   Free(accessOK);
    produce data                      consume data
  Wait(accessOK);                   Wait(accessOK);
    add full buffer to                add empty buffer to
    pool of fulls                     pool of empties
  Free(accessOK);                   Free(accessOK);
Free(filled);                     Free(empty);
```



Bounded buffer—discussion

- Why does **Producer** issue **`Wait(empty)`** but **`Free(filled)`**? *Because it is filling in a buffer.*
- Why is the order of **`Wait`**'s important? *Because of a possible deadlock.*
- Is order of **`Free`**'s important? *No.*
- Could we have separate semaphores for each item in the pool? *Yes.*
- How would this be extended to have 2 (or more) consumers? *Left as an exercise.*