



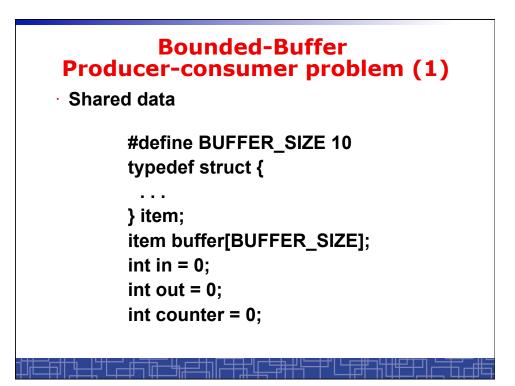
## Background: cooperating processes with shared memory. Many processes or threads are cooperating: One way is to use shared memory. But concurrent access to shared data may result in data inconsistency. To share data among processes (threads), we need some mechanisms to ensure the orderly execution of cooperating processes (threads) to maintain data consistency.

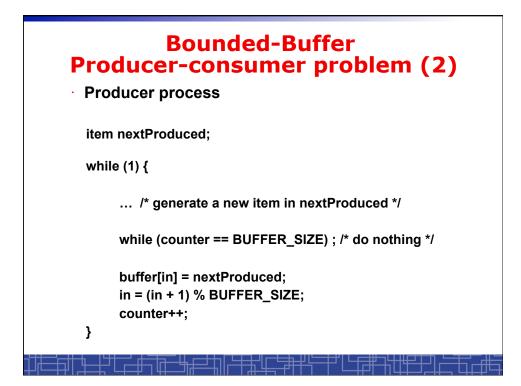


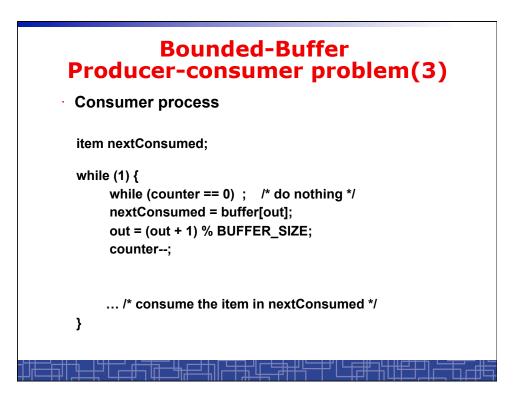
- How data inconsistency happens?
  - Example: producer-consumer problem using a bounded-buffer
- Pure software solution:
  - 2-process: Peterson's algorithm
  - N-process: Bakery algorithm
- Synchronization hardware
- Semaphores
- Three classic synchronization problems:
  - The bounded-buffer problem.
  - The reader-writer problem.
  - The dining-philosopher problem.

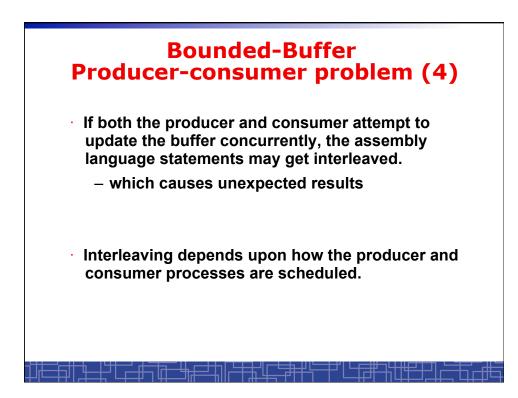


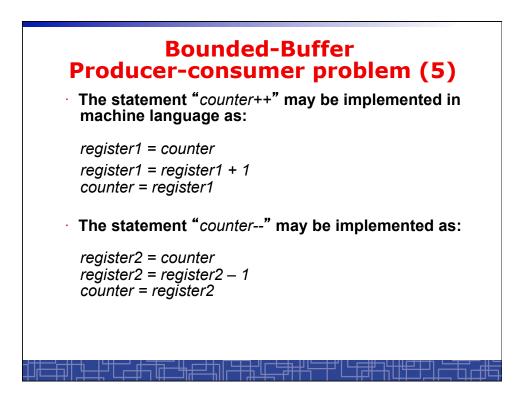
- Producer-Consumer problem:
  - Two parties: producer & consumer processes
  - A producer process produces information that is consumed by a consumer process.
  - Shared memory:
    - Bounded buffer: a fixed buffer size (producer blocks when the buffer is full)
  - Example:
    - Printer program  $\rightarrow$  printer driver
    - Compiler → assembler

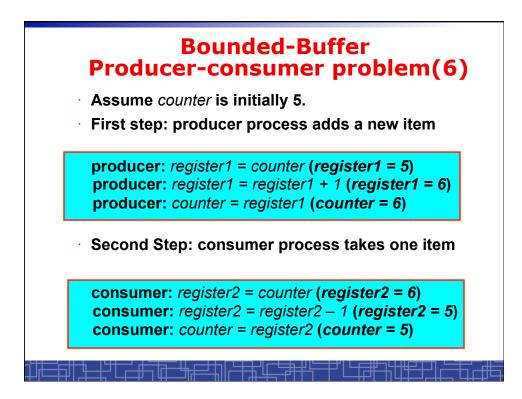


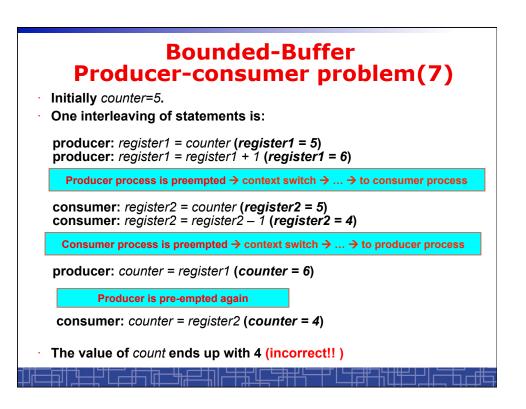


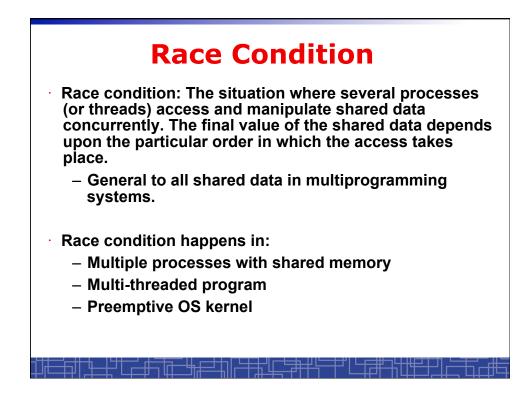


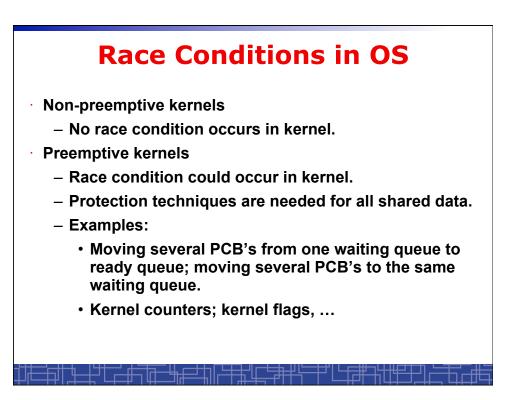


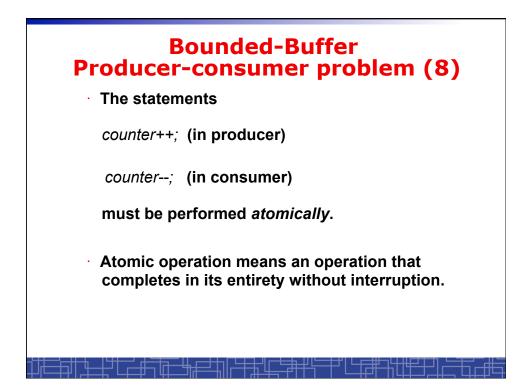


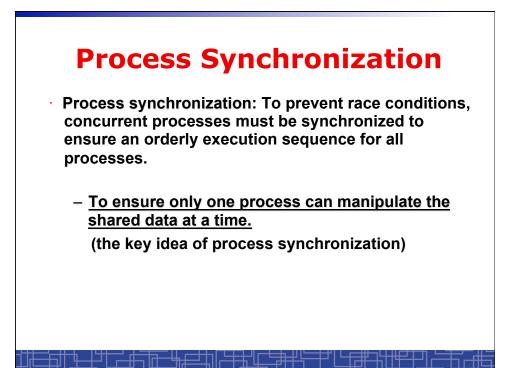


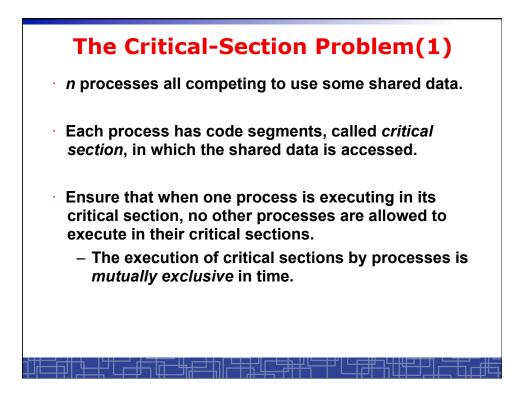












The Critical-Section Problem(2)			
• General structure of each process <i>P<sub>i</sub></i>			
	do {		
		entry section	
	critical section		
		exit section	
	remainder section		
	} whil	le (1);	
┆╫═╪╫┶╶┟╨╘═┙╓═╫╓═╫╖╔╪╫╢┍╪╫╫╚╤╫╵╘╒╫╖╟╫╝╧╢╺╤╦╸╫═			

