CSE 3221.3 Operating System Fundamentals

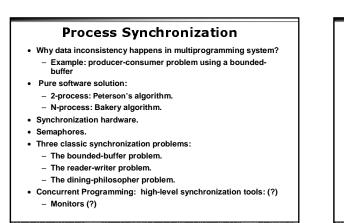
No.5

### **Process Synchronization(1)**

Prof. Hui Jiang Dept of Computer Science and Engineering York University

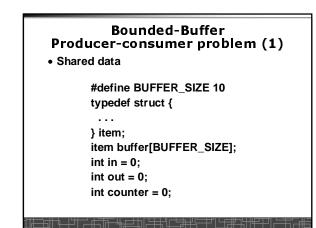
#### Background: cooperating processes with shared memory

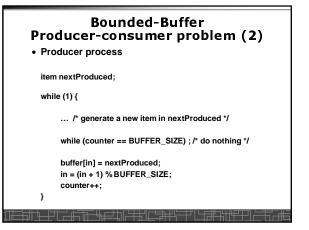
- Many processes or threads are cooperating: – One way is to use shared memory.
- But, concurrent access to shared data may result in data inconsistency.
- To share data among processes (threads), we need some mechanisms to ensure the orderly execution of cooperating processes to maintain data consistency.

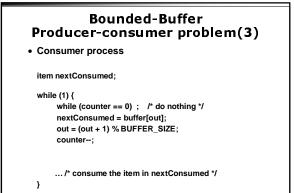


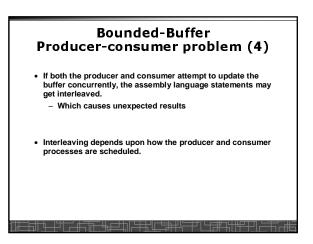
#### Producer-Consumer Problem: using shared memory

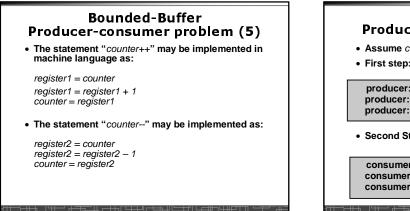
- Producer-Consumer problem:
  - Two parties: producer & consumer processes
  - A producer process produces information that is consumed by a consumer process.
  - Shared memory:
    - Unbounded-buffer: places no practical limit on the size of the buffer (producer never blocks)
    - Bounded buffer: a fixed buffer size (producer blocks)
      when the buffer is full)
  - Example:
    - Printer program → printer driver
    - Compiler  $\rightarrow$  assembler

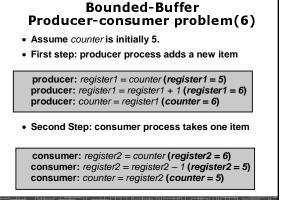


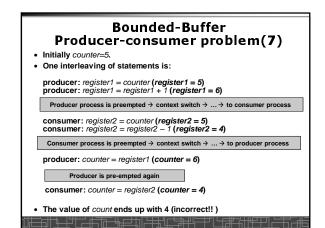


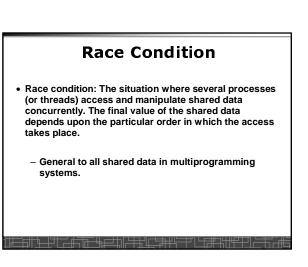












## Prepared by Prof. Hui Jiang (CSE 3221)

### **Race Conditions in OS**

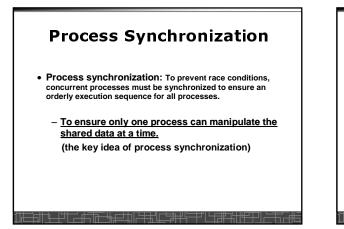
- Non-preemptive kernels – No race condition occurs in kernel.
- Preemptive kernels
  Acce condition could occur in kernel.
  - Protect techniques are needed.

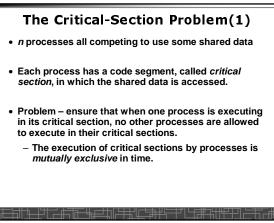


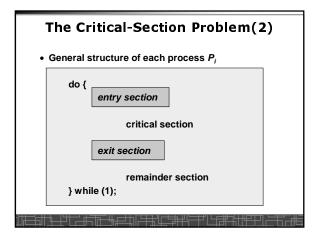
counter --; (in consumer)

must be performed atomically.

• Atomic operation means an operation that completes in its entirety without interruption.







### **Solution to Critical-Section Problem**

- 1. Mutual Exclusion
  - If a process is executing in its critical section, then no other processes can be executing in their critical sections.

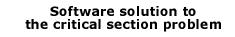
2. Progress

 If no process in its critical section and some processes wish to enter their critical sections, then only these processes wishing to enter the critical section can participate in the decision on which will enter the critical section next, and the decision selection of the processes cannot be postponed indefinitely.

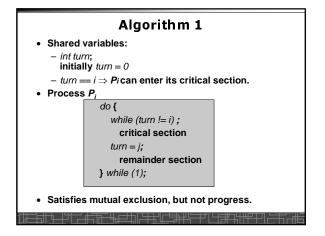
3. Bounded Waiting

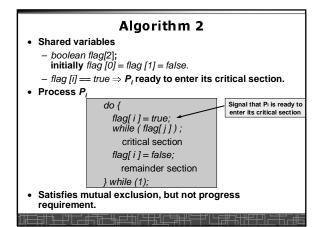
 After a process has made a request to enter its critical section, there much be a bound on the number of times that other processes are allowed to enter their critical sections before that request is granted.

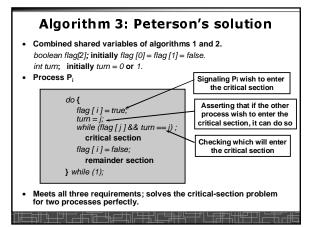
## Prepared by Prof. Hui Jiang (CSE 3221)

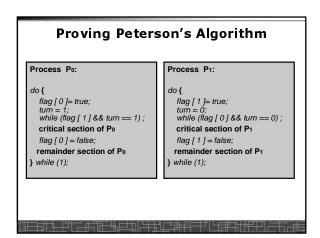


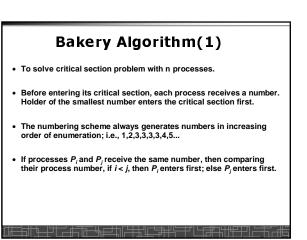
- Assume each process is executing at a non-zero speed. No assumption on their relative speed.
- No assumption on special hardware instructions except each instruction is executed atomically.
- No assumption on the number of CPU's in the system.
- Starting from the case with only two processes
  - Process P<sub>0</sub> and P<sub>1</sub>
  - When presenting Pi, use Pj to indicate another (j=1-i)

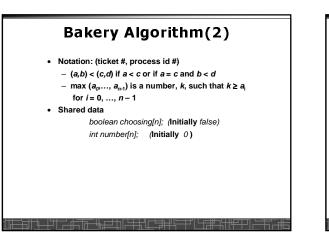


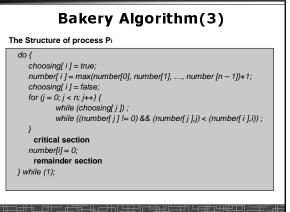


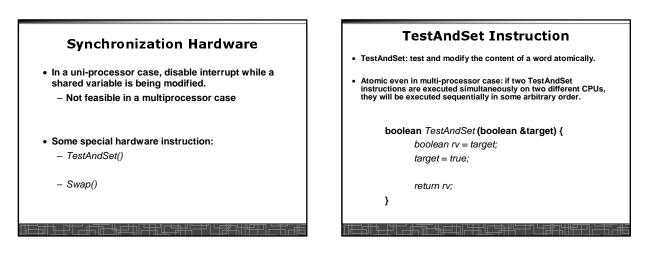


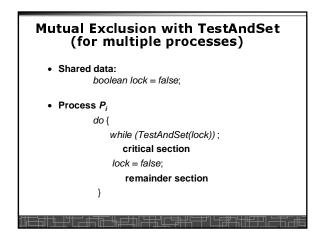


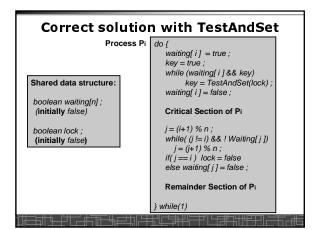












# Prepared by Prof. Hui Jiang (CSE 3221)

