BITS Pilani
K K Birla Goa Campus

Operating Systems

Dr. Lucy J. Gudino
Dept. of CS and IS

innovate achieve lead

BITS Pilani
K K Birla Goa Campus

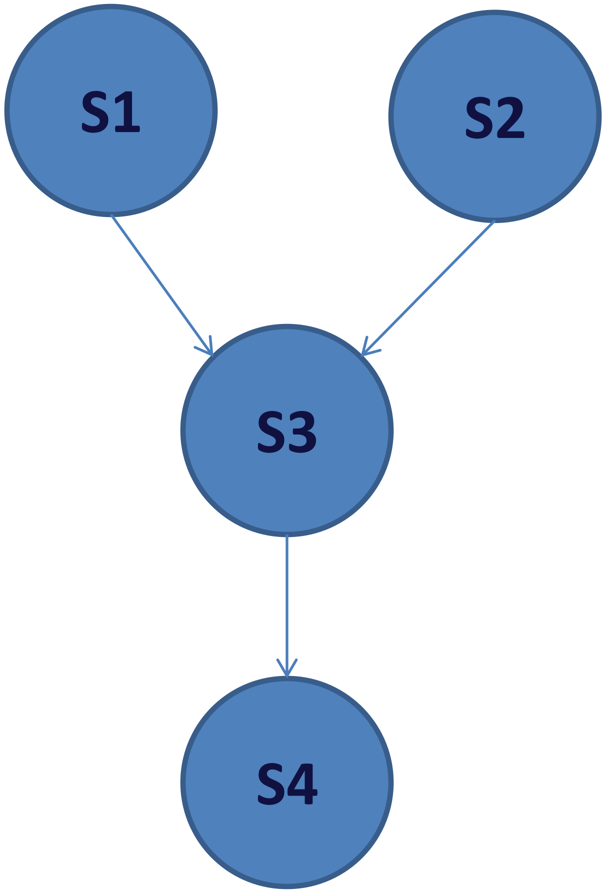
Process Synchronization

Last class

- Scheduling algorithms
  - Non - Preemptive : FCFS, SJF, Priority
  - Preemptive : SRT, RR, Multilevel Queue
- Basic assumption
  - Uni-processor system
- Distributed systems: 3 types
  - Minicomputer model
  - Workstation model
  - Processor pool model
- Processor allocation in multiprocessor system
  - Non-migratory and migratory
- Concurrent Processing

Last Class…

- Example :
  - S1 : a ← w + x
  - S2 : b ← y + z
  - S3: c ← b – a
  - S4: d ← c – 1



- Read Set

R(S1) = {w, x}
R(S2) = {y, z}
R(S3) = {b, a}
R(S4) = {c}

Write Set

W(S1) = {a}
W(S2) = {b}
W(S3) = {c}
W(S4) = {d}

Last Class…

- Concurrency conditions:

$$R(S_i) \cap W(S_j) = \phi$$

$$W(S_i) \cap R(S_j) = \phi$$

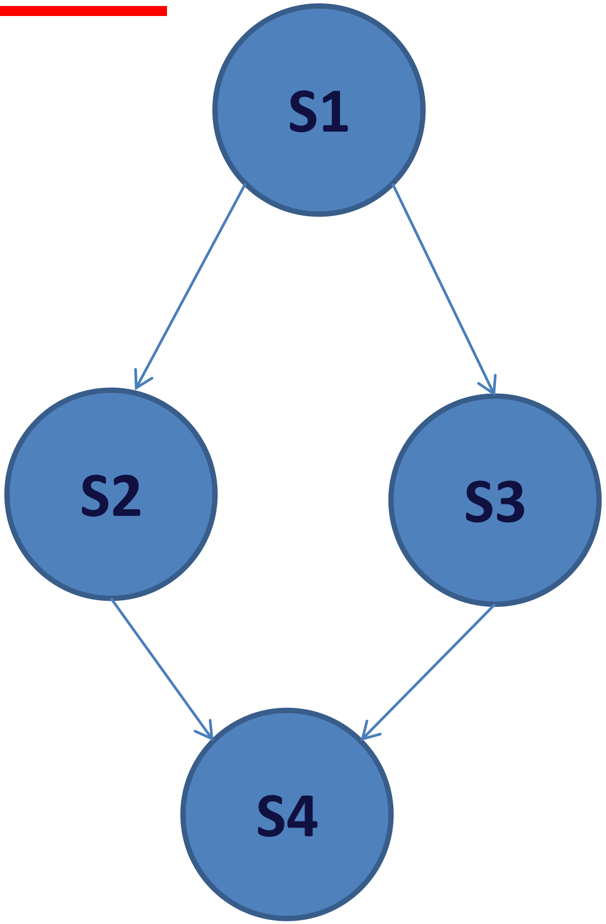
$$W(S_i) \cap W(S_j) = \phi$$

$$R(S_i) \cap R(S_j) \neq \phi$$

•Read Set	Write Set
R(S1) = {w, x}	W(S1) = {a}
R(S2) = {y, z}	W(S2) = {b}
R(S3) = {b, a}	W(S3) = {c}
R(S4) = {c}	W(S4) = {d}

Last Class…

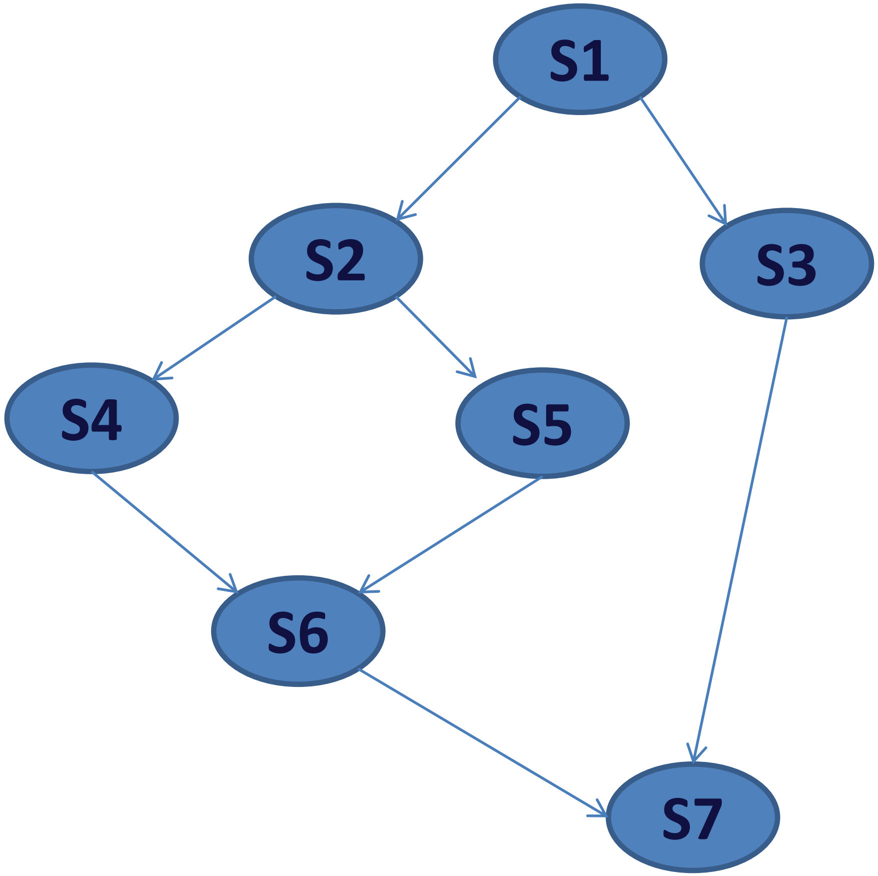
```
begin
        count = 2
        s1
        fork L1
        s2
    L2: join count
        s4
end

L1: s3
    goto L2
```



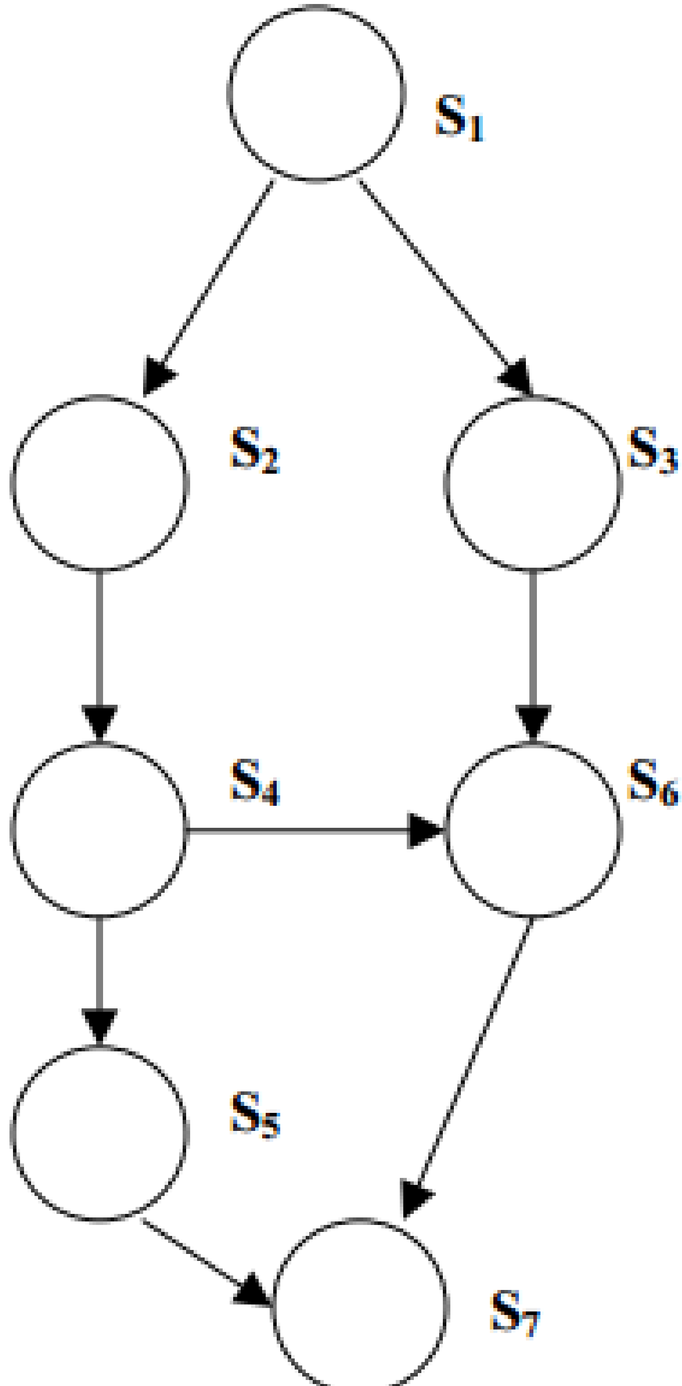
Join count implementation

```
count = count -1;
if count != 0 then
    quit or terminate the process
```

Example



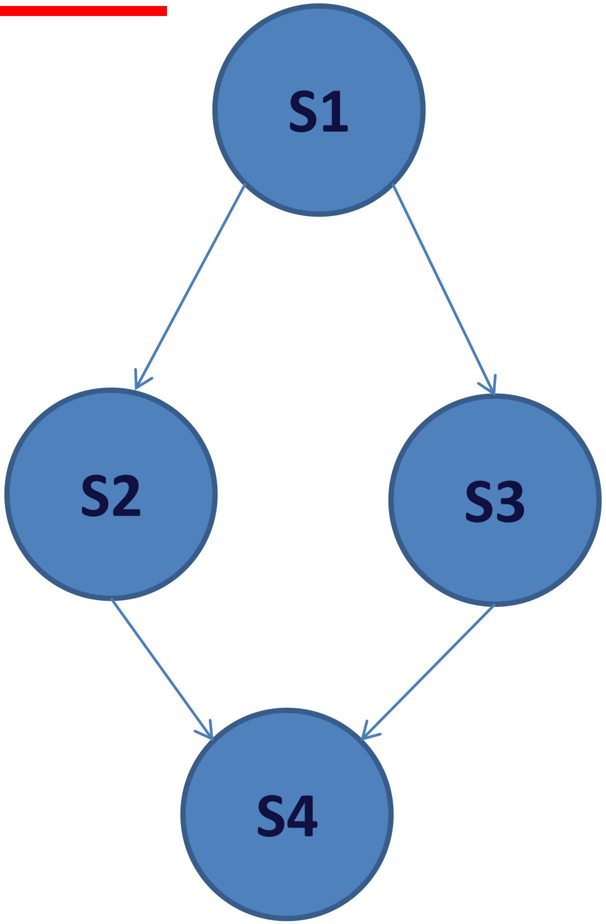
Example 2



S_1;
count1:=2;
fork L1;
S_2;
S_4;
count2:=2;
fork L2;
S_5;
Go to L3;
L1: S_3;
L2: join count1;
S_6;
L3: join count2;
S_7;

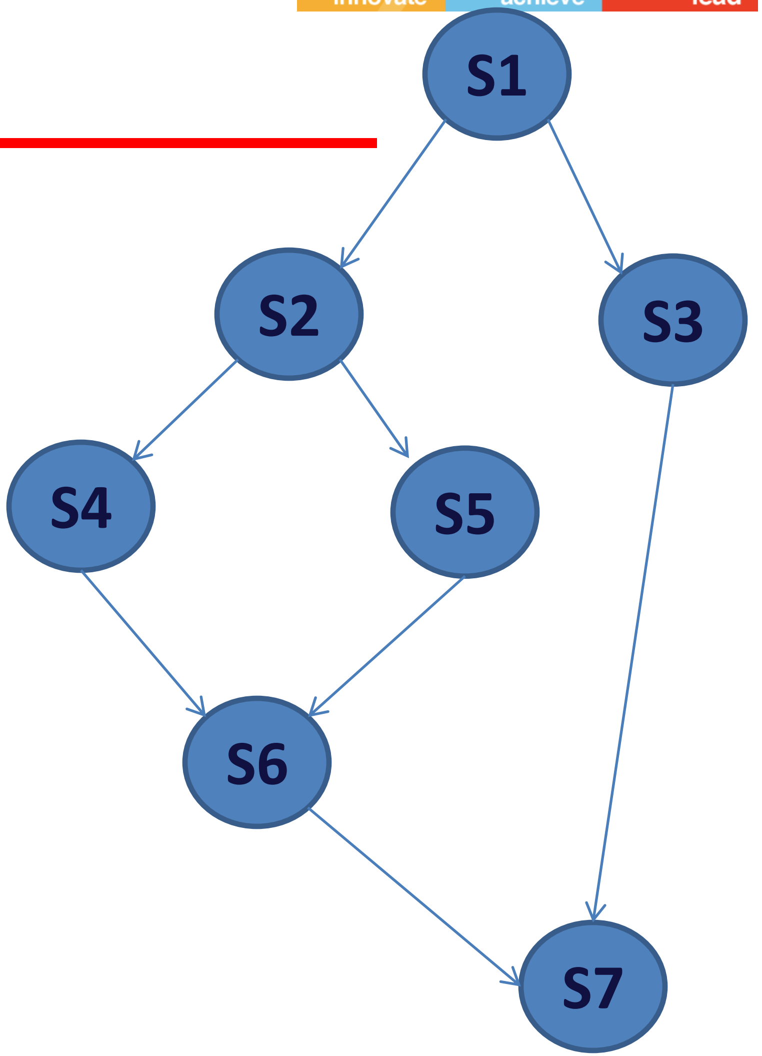
cobegin and coend

```
begin
        s1
        cobegin
                s2
                s3
        coend
        s4
end
```

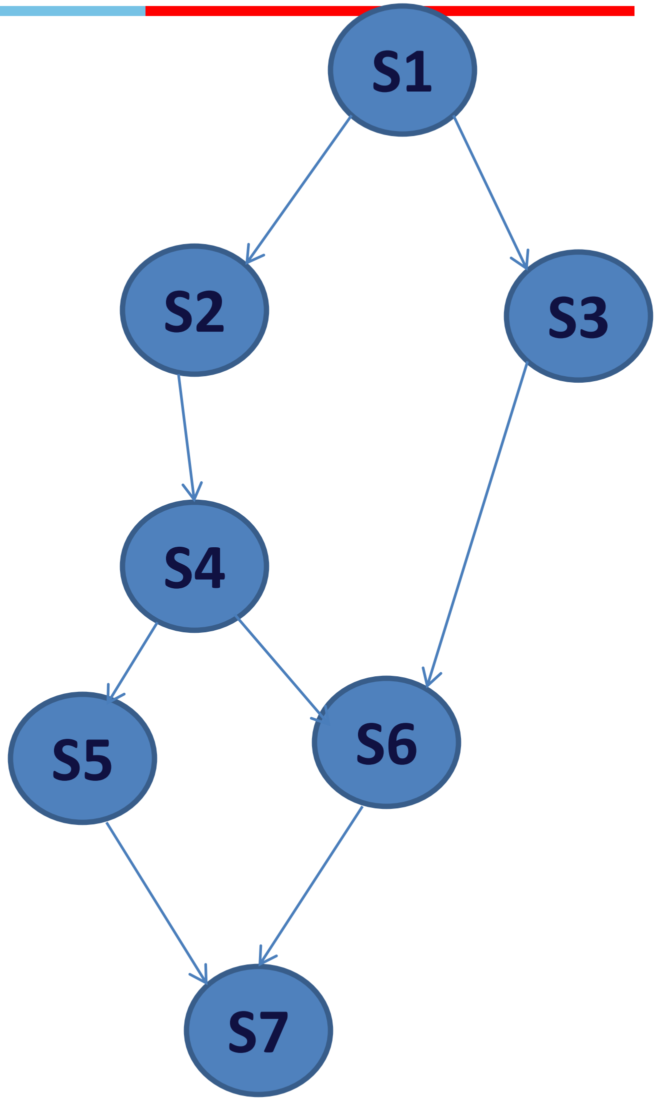


contd…

```
begin
        s1
        cobegin
                s3
                begin
                        s2
                        cobegin
                                s4
                                s5
                        coend
                        s6
                end
        coend
        s7
end
```

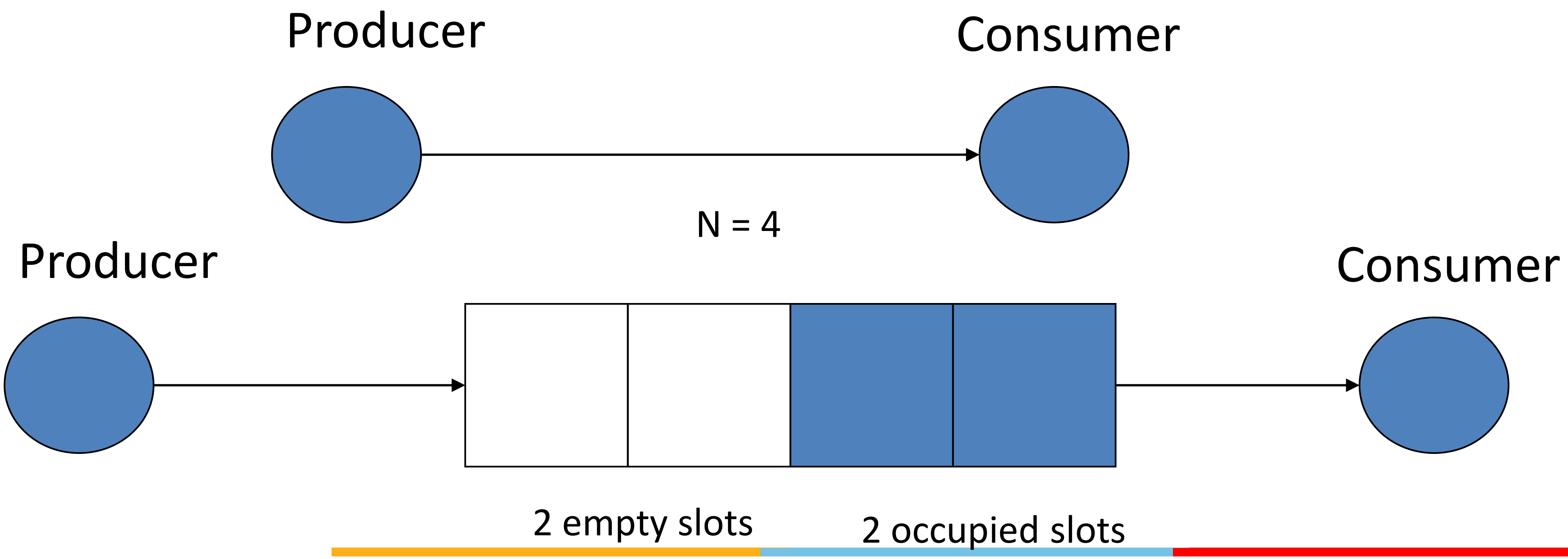


Example



contd…

- Concurrent access to shared data may result in data inconsistency
- Maintaining data consistency requires mechanisms to ensure the orderly execution of cooperating processes
- Producer / Consumer Problem



Producer / Consumer Problem…

- Two types of buffers:
  - unbounded buffer:
    - No limit on the size of the buffer
    - always space for producer to store data items
    - if producer is slower consumer needs to wait
  - bounded buffer: 2 cases
    - if the producer is faster than consumer
      - some point in time buffer will be full, producer has to wait
    - if the consumer is faster than producer
      - some point in time buffer will be empty, consumer has to wait
- use shared memory