SNS COLLEGE OF TECHNOLOGY

Coimbatore-35.
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

COURSE NAME : 23CST202 – OPERATING SYSTEMS

II YEAR/ IV SEMESTER

UNIT – II PROCESS SCHEDULING AND SYNCHRONIZATION

Topic: Process Synchronization, The critical-section problem, Synchronization hardware

Dr.V.Savitha
Associate Professor
Department of Computer Science and Engineering

Need of Synchronization

- Processes can execute concurrently
  - May be interrupted at any time, partially completing execution
- Concurrent access to shared data may result in data inconsistency
- Maintaining data consistency requires mechanisms to ensure the orderly execution of cooperating processes

- Illustration of the problem:
  Suppose that we wanted to provide a solution to the consumer-producer problem that fills ***all*** the buffers. We can do so by having an integer **counter** that keeps track of the number of full buffers. Initially, **counter** is set to 0. It is incremented by the producer after it produces a new buffer and is decremented by the consumer after it consumes a buffer.

Semaphores

- A semaphore is a synchronization tool used in computing to manage access to shared resources.
- It works like a signal that allows multiple processes or threads to coordinate their actions.
- Semaphores use counters to keep track of how many resources are available, ensuring that no two processes can use the same resource at the same time, thus preventing conflicts and ensuring orderly execution.

A **semaphore** S is an integer variable that can be accessed only through two standard operations: wait() and signal().

The **wait()** operation reduces the value of the semaphore by 1 and the **signal()** operation increases its value by 1.

Types of Semaphores

Binary Semaphore – This is similar to mutex lock but not the same thing. It can have only two values – 0 and 1. Its value is initialized to 1. It is used to implement the solution of critical section problems with multiple processes.

Counting Semaphore – Its value can range over an unrestricted domain. It is used to control access to a resource that has multiple instances.

Producer & Consumer

Producer

```
while (true) {
     /* produce an item in next produced */

     while (counter == BUFFER_SIZE) ;
          /* do nothing */
     buffer[in] = next_produced;
     in = (in + 1) % BUFFER_SIZE;
     counter++;
}
```

Producer & Consumer

Consumer

```
while (true) {
     while (counter == 0)
          ; /* do nothing */
     next_consumed = buffer[out];
     out = (out + 1) % BUFFER_SIZE;
        counter--;
     /* consume the item in next consumed
*/
}
```

REFERENCES

TEXT BOOKS:

T1 Silberschatz, Galvin, and Gagne, “Operating System Concepts”, Ninth Edition, Wiley India Pvt Ltd, 2009.)

T2. Andrew S. Tanenbaum, “Modern Operating Systems”, Fourth Edition, Pearson Education, 2010

REFERENCES:

R1 Gary Nutt, “Operating Systems”, Third Edition, Pearson Education, 2004.

R2 Harvey M. Deitel, “Operating Systems”, Third Edition, Pearson Education, 2004.

R3 Abraham Silberschatz, Peter Baer Galvin and Greg Gagne, “Operating System Concepts”, 9th Edition, John Wiley and Sons Inc., 2012.

R4. William Stallings, “Operating Systems – Internals and Design Principles”, 7th Edition, Prentice Hall, 2011