Operating System

Process Management: Synchronization, Semaphores, Deadlock.

Session 7

Tutor: Yeshi Wangchuk, Asstt. Lecturer, IT Dept.

Synchronization -1

- ***Process synchronization*** is when one process waits fro notification of an event that will occur in another process.
- Basic type of ***synchronization*** of processes is ***signaling***.
- Processes needs to signal one another to execute.
- In the ***producer-consumer pattern***, the producer is repeatedly signaling the consumer so that producer doesn't get too far ahead of the consumer.
- In ***client-server pattern***, a single consumer(i.e server) can be signaled by a no. of producers (the clients).
- The client-server ***synchronize so they can communicate***.

Synchronization -2

- **Mutual Exclusion.**
  - The ***mutual exclusion pattern*** is also a version of signaling, but it has some differences.
  - It can be thought as a ***second basic type of synchronization.***
  - In mutual exclusion pattern, a process must wait for another process to leave its ***critical section*** (i.e wait for event to end other process's critical section)
  - ***Signaling*** is the most basic form of ***process cooperation***,
  - Where as ***mutual exclusion*** is the most basic form of ***process competition.***

Synchronization -3

Critical Section(CS)

- The critical section for data item d_s is a section of code that should not be executed concurrently either within itself with other critical section(s) for d_s.
- General structure of a typical process p_i:

```
do{
        entry section
                critical section
        exit section
                remainder section
}while(1);
```

Synchronization -4

- **Properties of a CS implementation:**
  1. ***Mutual Exclusion:*** Only process should enter into CS at a time.
  2. ***Progress:*** If no process is executing in CS, other process which is not in the remainder section can enter its CS next.
  3. ***Bounded Waiting:*** No. of times that other processes are allowed to enter their CS after process has made request to enter its CS and before that request is granted.

Semaphores -1

- The solution to the CS problem are not easy to generalize to more complex problem.
- To overcome this difficulty, we use a synchronization tool called a semaphores.
- Semaphore S is a shared integer variable with non-negative values that can be accessed only through wait and signal operations.

```
wait(S){
        if(S>0)
            S=S-1;
        else
          //block the process on S;
}
```

```
signal(S){
    if(some processes are blocked on S)
            //Activate one blocked process
        else
            S=S+1;
}
```

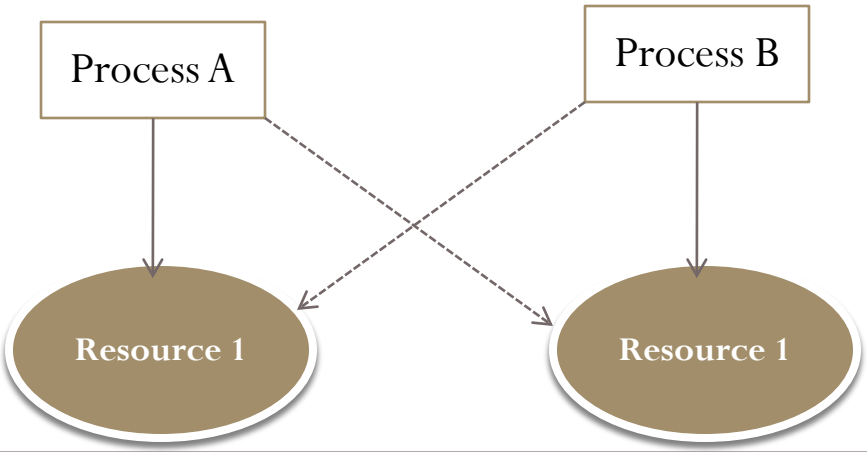
Semaphores -2

- **Binary Semaphores**
  - The semaphores that we have seen is commonly known as a ***counting semaphore***.
  - In counting semaphores, its ***integer value can range*** over an ***unrestricted domain***.
  - Where as a semaphores with its ***integer value*** which range ***only between 0 or 1*** is called ***binary semaphore.***
  - ***Binary semaphore*** is ***simpler to implement*** than ***counting semaphore.***

Deadlock

Deadlock Definition

- Deadlock is a situation in which some processes wait for each other's actions indefinitely.
- Processes involved in a deadlock remain blocked permanently.
- Figure 7.0: deadlock condition



Presentation Topics (Group Wise)

- 5 Group: 4-5 members in each group ***taking 8-10 min***.
- Topics are:
  1. Classic Problems of synchronization
  2. Monitors
  3. Atomic Transaction – Checkpoints, Concurrent Atomic transaction(serialization & Lock)
  4. Deadlock, Deadlock characteristics, Deadlock Prevention
  5. Deadlock Avoidance & Deadlock Detection
  6. Recovery from Deadlock.