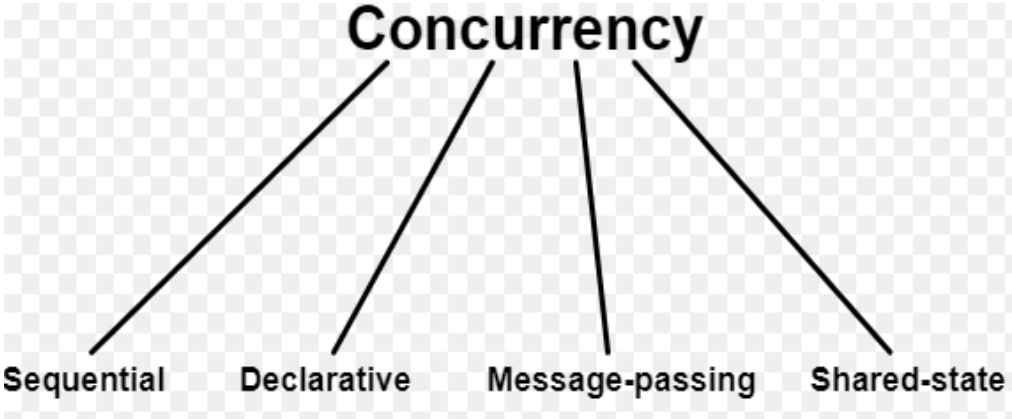
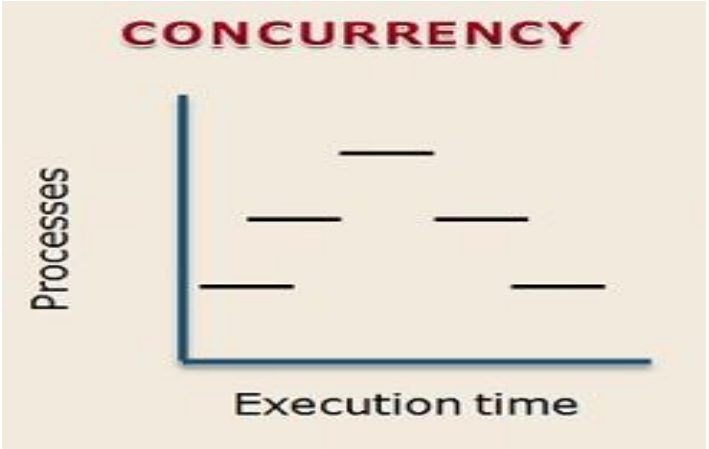
AMIRAJ

COLLEGE OF ENGINEERING & TECHNOLOGY

CHAPTER -3
CONCURRENCY

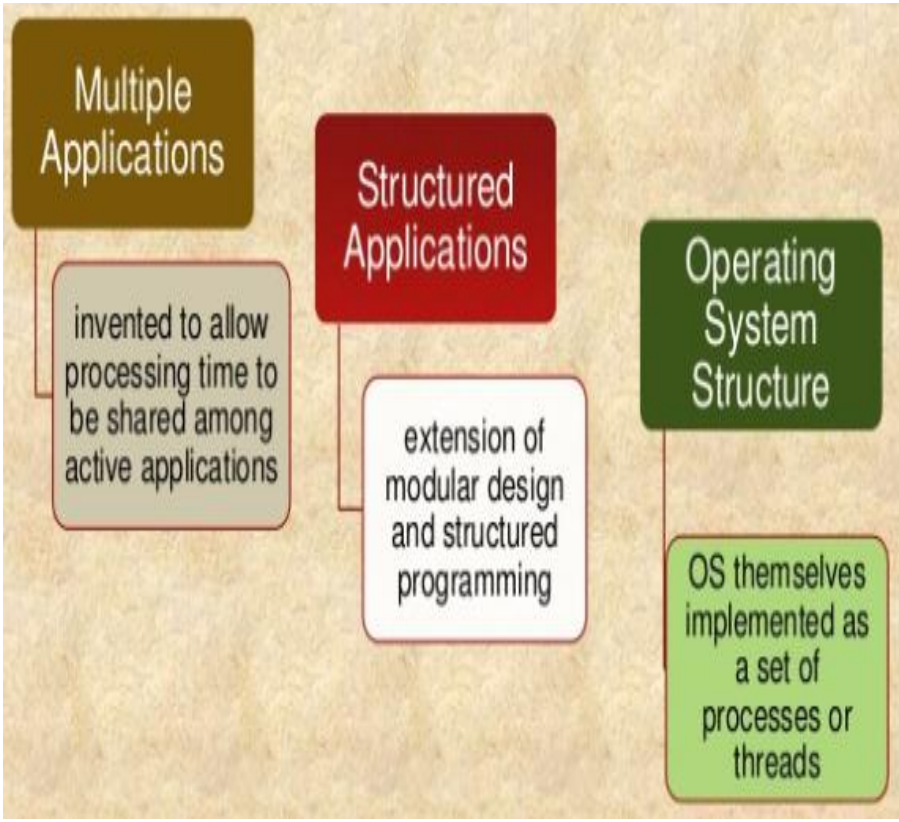




Subject:- OS **Code:-3140702**	**Prepared by:** **Asst.Prof.Foram Patel** **(Computer Department,ACET)**	AMIRAJ COLLEGE OF ENGINEERING & TECHNOLOGY

WHAT IS CONCURRENCY

- **Multiple application:**
- To allow sharing processing time.
- **Structured applications:**
- Extension of the principles of modular design.
- **Operating system structure:**
- Operating system are themselves implemented as a set of processes or threads.

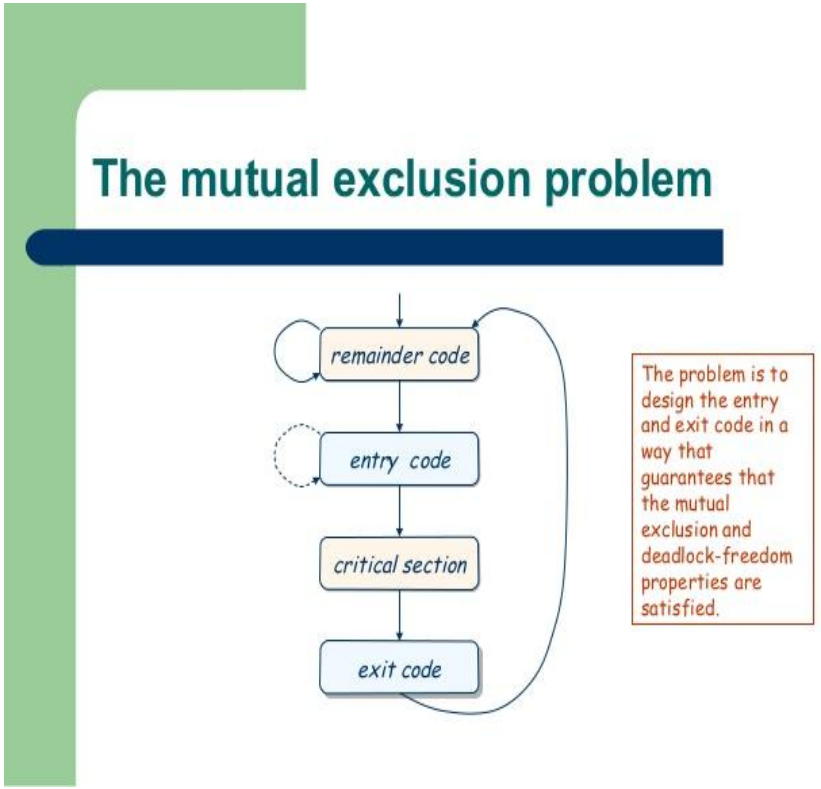


PRICIPLE OF CONCURRENCY

- Concurrent access on shared data and result data inconsistency.
- Concurrency is the computation of process within a time frame to give an impression of simultaneous execution.
- This is not the same thing as actually running simultaneously.
- True parallelism allow simultaneous execution of process.
- **Example of concurrency:**
- Concurrency in multiprogramming.
- Concurrency in multithreading
- Concurrency in multiprocessor.
- Concurrency in multi computer

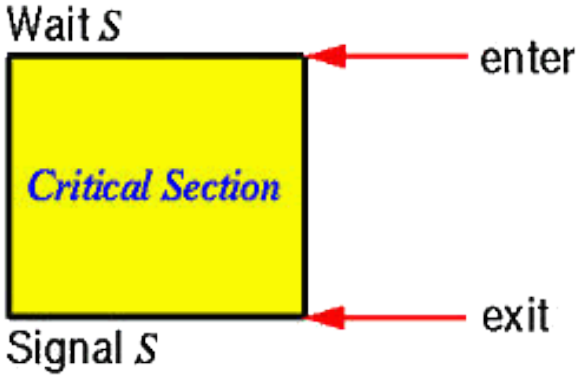
MUTUAL EXCLUSION

- The process is accessing a shared variable so process is in critical section.
- No two threads simultaneously in critical section.
- If process is executing in its critical section ,then no other processes can be executing in their critical sections.



CRITICAL SECTION

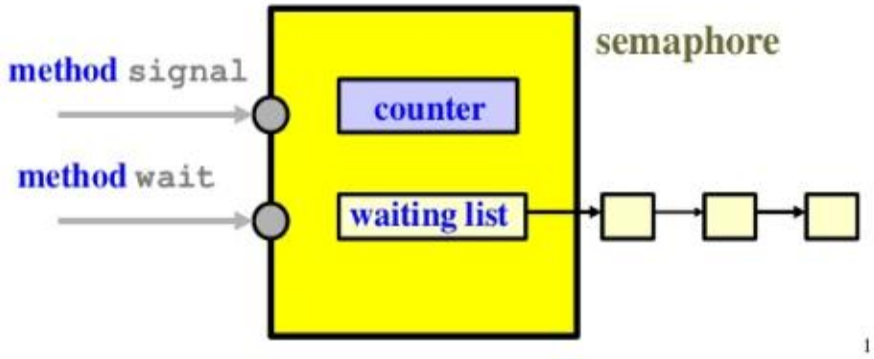
- When one process is in a critical section , all other processes are excluded from their critical section.
- Each process must ask permission to enter critical section in entry section , may follow critical section with exit section , then reminder section.




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

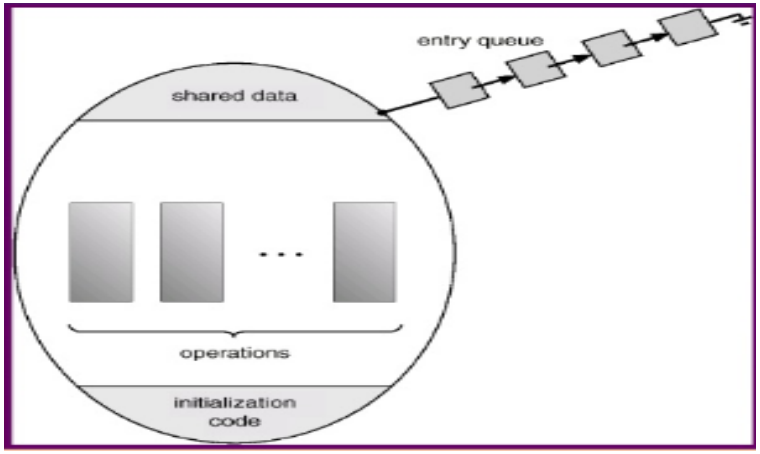
SEMAPHORES

- A semaphore is an object that consists of a counter, a waiting list of processes and two methods : signal & wait.
- Semaphore is a synchronization tool which can be used to deal with the critical section problem.
- It is a protected variable whose value can be accessed and altered only by the operation P & V.



MONITOR

- Monitor is a highly structured programming language construct.
- Only one process may be active within the monitor at a time.
- Private variables and Private procedures - Use within a monitor.
- Monitors have no public data.




```
monitor Monitor-Name
{
    local variable declarations;

    Procedure1(...)
    { // statements };
    Procedure2(...)
    { // statements };
    // other procedures
    {
       // initialization
    }
}
```

MESSAGE PASSING

- Message passing is the basis of the most inter-process communication in distributed system.
- It requires the programmer to know

1) Message
2) Name of source
3) Destination process

- OS send() system call to pass message to kernel. After execution user process waits for result with receive().