5. **Explain about Inter Process Communication.**

- ✓ Definition
- ✓ Message Passing System
- ✓ Naming
- ✓ Direct Communication
- ✓ Indirect Communication
- ✓ Synchronization
- ✓ Buffering

UNIT – II

TWO MARKS

1. **What is a Thread?**

A thread otherwise called a lightweight process (LWP) is a basic unit of CPU utilization, it comprises of a thread id, a program counter, a register set and a stack. It shares with other threads belonging to the same process its code section, data section, and operating system resources such as open files and signals.

2. **What are the benefits of Multithreaded Programming?**

The benefits of multithreaded programming can be broken down into four major categories:

- ❖ Responsiveness
- ❖ Resource sharing
- ❖ Economy
- ❖ Utilization of multiprocessor architectures

3. **Compare User Threads and Kernel Threads.**

User threads	Kernel threads
User threads are supported above the kernel and are implemented by a thread library at the user level	Kernel threads are supported directly by the operating system
Thread creation & scheduling are done in the user space, without kernel intervention. Therefore they are fast to create and manage	Thread creation, scheduling and management are done by the operating system. Therefore they are slower to create & manage compared to user threads
Blocking system call will cause the entire process to block	If the thread performs a blocking system call, the kernel can schedule another thread in the application for execution

4. **Define Thread Cancellation & Target Thread.**

The thread cancellation is the task of terminating a thread before it has completed. A thread that is to be cancelled is often referred to as the target thread. For example, if multiple threads are concurrently searching through a database and one thread returns the result, the remaining threads might be cancelled.

5. **What are the different ways in which a Thread can be cancelled?**

Cancellation of a target thread may occur in two different scenarios:
Asynchronous cancellation: One thread immediately terminates the target thread is called asynchronous cancellation.
Deferred cancellation: The target thread can periodically check if it should terminate, allowing the target thread an opportunity to terminate itself in an orderly fashion.

6. **Define CPU Scheduling.**

CPU scheduling is the process of switching the CPU among various processes. CPU scheduling is the basis of multiprogrammed operating systems. By switching the CPU among processes, the operating system can make the computer more productive.

7. **What is Preemptive and Non - Preemptive scheduling?**

Under non - preemptive scheduling once the CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either by terminating or switching to the waiting state.
Preemptive scheduling can preempt a process which is utilizing the CPU in between its execution and give the CPU to another process.

8. **What is a Dispatcher?**

The dispatcher is the module that gives control of the CPU to the process selected by the short-term scheduler. This function involves:

- Switching context
- Switching to user mode
- Jumping to the proper location into the user program to restart that program.

9. **What is Dispatch Latency?**

The time taken by the dispatcher to stop one process and start another running is known as dispatch latency.

10. **What are the various scheduling criteria for CPU Scheduling?**

The various scheduling criteria are,

- CPU utilization
- Throughput
- Turnaround time
- Waiting time
- Response time

11. **Define Throughput?**

Throughput in CPU scheduling is the number of processes that are completed per unit time. For long processes, this rate may be one process per hour; for short transactions, throughput might be 10 processes per second.

12. What is Turnaround Time?

Turnaround time is the interval from the time of submission to the time of completion of a process. It is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.

13. Define Race Condition.

When several process access and manipulate same data concurrently, then the outcome of the execution depends on particular order in which the access takes place is called race condition. To avoid race condition, only one process at a time can manipulate the shared variable.

14. What is Critical Section problem?

Consider a system consists of 'n'processes. Each process has segment of code called a critical section, in which the process may be changing common variables, updating a table, writing a file. When one process is executing in its critical section, no other process can allowed executing in its critical section.

15. What are the requirements that a solution to the Critical Section Problem must satisfy?

The three requirements are,

- Mutual exclusion
- Progress
- Bounded waiting

16. Define Entry Section and Exit Section.

The critical section problem is to design a protocol that the processes can use to cooperate. Each process must request permission to enter its critical section. The section of the code implementing this request is the entry section. The critical section is followed by an exit section. The remaining code is the remainder section.

17. Give two hardware instructions and their definitions which can be used for implementing Mutual Exclusion.

Test And Set

```
        boolean TestAndSet (boolean &target)
    {
            boolean rv = target;
            target = true;
            return rv;
    }
```

Swap

```
        void Swap (boolean &a, boolean &b)
    {
            boolean temp = a;
            a = b;
            b = temp;
        }
```

18. What is a Semaphore?

A semaphore 'S' is a synchronization tool which is an integer value that, apart from initialization, is accessed only through two standard atomic operations; wait and signal. Semaphores can be used to deal with the n-process critical section problem. It can be also used to solve various synchronization problems.

The classic definition of 'wait'

```
wait (S)
{
        while (S<=0)
        S--;
}
```

The classic definition of 'signal'

```
signal (S)
{
        S++;
}
```

19. Define Busy Waiting and Spinlock.

When a process is in its critical section, any other process that tries to enter its critical section must loop continuously in the entry code. This is called as busy waiting and this type of semaphore is also called a spinlock, because the process while waiting for the lock.

20. How can we say the First Come First Served scheduling algorithm is Non Preemptive?

Once the CPU has been allocated to the process, that process keeps the CPU until it releases, either by terminating or by requesting I/O. So we can say the First Come First Served scheduling algorithm is non preemptive.

21. What is Waiting Time in CPU scheduling?

Waiting time is the sum of periods spent waiting in the ready queue. CPU scheduling algorithm affects only the amount of time that a process spends waiting in the ready queue.

22. What is Response Time in CPU scheduling?

Response time is the measure of the time from the submission of a request until the first response is produced. Response time is amount of time it takes to start responding, but not the time that it takes to output that response.

23. Differentiate Long Term Scheduler and Short Term Scheduler

The long-term scheduler or job scheduler selects processes from the job pool and loads them into memory for execution.
The short-term scheduler or CPU scheduler selects from among the process that are ready to execute, and allocates the CPU to one of them.

24. Write some classical problems of Synchronization?

- The Bounded-Buffer Problem
- The Readers-Writers Problem
- The Dining Philosophers Problem

25. When the error will occur when we use the Semaphore?

- ❖ When the process interchanges the order in which the wait and signal operations on the semaphore mutex.
- ❖ When a process replaces a signal (mutex) with wait (mutex).
- ❖ When a process omits the wait (mutex), or the signal (mutex), or both.

26. What is Mutual Exclusion?

A way of making sure that if one process is using a shared modifiable data, the other processes will be excluded from doing the same thing. Each process executing the shared data variables excludes all others from doing so simultaneously. This is called mutual exclusion.

27. Define the term Critical Regions?

Critical regions are small and infrequent so that system through put is largely unaffected by their existence. Critical region is a control structure for implementing mutual exclusion over a shared variable.

28. What are the drawbacks of Monitors?

- ❖ Monitor concept is its lack of implementation most commonly used programming languages.
- ❖ There is the possibility of deadlocks in the case of nested monitor's calls.

29. What are the two levels in Threads?

Thread is implemented in two ways.

- ❖ User level and Kernel level

30. What is a Gantt Chart?

A two dimensional chart that plots the activity of a unit on the Y-axis and the time on the X-axis. The chart quickly represents how the activities of the units are serialized.

31. Define Deadlock.

A process requests resources; if the resources are not available at that time, the process enters a wait state. Waiting processes may never again change state, because the resources they have requested are held by other waiting processes. This situation is called a deadlock.

32. What is the sequence in which resources may be utilized?

Under normal mode of operation, a process may utilize a resource in the following sequence:

- ❖ Request: If the request cannot be granted immediately, then the requesting process must wait until it can acquire the resource.
- ❖ Use: The process can operate on the resource.
- ❖ Release: The process releases the resource.

33. What are conditions under which a deadlock situation may arise?

A deadlock situation can arise if the following four conditions hold simultaneously in a system:

- ❖ Mutual exclusion
- ❖ Hold and wait
- ❖ No pre-emption
- ❖ Circular wait

34. What is a Resource-Allocation Graph?

Deadlocks can be described more precisely in terms of a directed graph called a system resource allocation graph. This graph consists of a set of vertices V and a set of edges E. The set of vertices V is partitioned into two different types of nodes; P the set consisting of all active processes in the system and R the set consisting of all resource types in the system.

35. Define Request Edge and Assignment Edge.

A directed edge from process Pi to resource type Rj is denoted by PiàRj; it signifies that process Pi requested an instance of resource type Rj and is currently waiting for that resource. A directed edge from resource type Rj to process Pi is denoted by RjàPi, it signifies that an instance of resource type has been allocated to a process Pi. A directed edge PiàRj is called a request edge. A directed edge RjàPi is called an assignment edge.

36. What are the methods for Handling Deadlocks?

The deadlock problem can be dealt with in one of the three ways:

- ❖ Use a protocol to prevent or avoid deadlocks, ensuring that the system will never enter a deadlock state.
- ❖ Allow the system to enter the deadlock state, detect it and then recover.
- ❖ Ignore the problem all together, and pretend that deadlocks never occur in the system.

37. Define Deadlock Prevention.

Deadlock prevention is a set of methods for ensure that at least any one of the four necessary conditions like mutual exclusion, hold and wait, no pre-emption and circular wait cannot hold. By ensuring that that at least one of these conditions cannot hold, the occurrence of a deadlock can be prevented.

38. Define Deadlock Avoidance.

An alternative method for avoiding deadlocks is to require additional information about how resources are to be requested. Each request requires the system consider the resources currently available, the resources currently allocated to each process, and the future requests and releases of each process, to decide whether the could be satisfied or must wait to avoid a possible future deadlock.

39. What are a Safe State and an Unsafe State?

A state is safe if the system can allocate resources to each process in some order and still avoid a deadlock. A system is in safe state only if there exists a safe sequence. A sequence of processes <P1,P2,….Pn> is a safe sequence for the current allocation state if, for each Pi, the resource that Pi can still request can be satisfied by the current available resource plus the resource held by all the Pj, with j<i. if no such sequence exists, then the system state is said to be unsafe.

40. What is Banker's Algorithm?

Banker's algorithm is a deadlock avoidance algorithm that is applicable to a resource-allocation system with multiple instances of each resource type. The two algorithms used for its implementation are:
***Safety algorithm**:* The algorithm for finding out whether or not a system is in a safe state.
***Resource-request algorithm**:* if the resulting resource-allocation is safe, the transaction is completed and process Pi is allocated its resources. If the new state is unsafe Pi must wait and the old resource-allocation state is restored.

41. Define Logical Address and Physical Address.

An address generated by the CPU is referred as logical address. An address seen by the memory unit that is the one loaded into the memory address register of the memory is commonly referred to as physical address.

42. What are Logical Address Space and Physical Address Space?

The set of all logical addresses generated by a program is called a logical address space; the set of all physical addresses corresponding to these logical addresses is a physical address space.

43. What is the main function of the Memory-Management Unit?

The runtime mapping from virtual to physical addresses is done by a hardware device called a memory management unit (MMU).

44. What are the methods for dealing the Deadlock Problem?

- ❖ Use a protocol to ensure that the system will never enter a deadlock state.
- ❖ Allow the system to enter the deadlock state and then recover.
- ❖ Ignore the problem all together, and pretend that deadlocks never occur in the system.

45. Differentiate Deadlock and Starvation.

A set of processes is in deadlock state when every process in the set is waiting for an event that can be caused only by the other process in the set.
Starvation or indefinite blocking is a situation where processes wait indefinitely within the semaphore.

FIFTEEN MARKS

1. Write about the various CPU Scheduling Algorithms.

- ❖ Optimization Criteria
- ❖ First-Come, First-Served (FCFS) Scheduling
- ❖ Shortest-Job-First (SJF) Scheduling
- ❖ Priority Scheduling
- ❖ Round Robin (RR)
- ❖ Multilevel Queue

- ❖ Multilevel Feedback Queue

2. Explain the classical problem on Synchronization.

Classical Problems are,

- ❖ Bounded-Buffer Problem
- ❖ Readers and Writers Problem
- ❖ Dining-Philosophers Problem

3. Explain about Monitors.

Introduction

High-level synchronization construct allows the safe sharing of an abstract data type among concurrent processes.

```
monitor monitor-name
    {
    shared variable declarations
    procedure body P1 (…) { . . . }
    procedure body P2 (…) {  . . . }
    procedure body Pn (…) { . . . }
    {
    initialization code
    }
    }
```

4. Monitor Implementation Using Semaphores

- ✓ Variables

```
semaphore mutex; // (initially = 1)
semaphore next; // (initially = 0)
int next-count = 0;
```

- ✓ Each external procedure F will be replaced by

```
wait(mutex);
…
body of F;
…
if (next-count > 0)
signal(next)
else
signal(mutex);
```

- ✓ Mutual exclusion within a monitor is ensured.
- ✓ For each condition variable x, we have:

```
        semaphore x-sem; // (initially = 0)
int x-count = 0;
```

- ✓ The operation x.wait can be implemented as:

```
                x-count++;
        if (next-count > 0)
        signal(next);
```

```
else
signal(mutex);
wait(x-sem);
x-count--;
```

- ✓ The operation x.signal can be implemented as:

```
                    if (x-count > 0) {          next-count++;
signal(x-sem);
wait(next);
next-count--; }
```

5. Give a detailed description about Deadlocks and its Characterization

- ✓ Deadlock Characterization
- ✓ Necessary Conditions

- ❖ Mutual exclusion: only one process at a time can use a resource.
- ❖ Hold and wait: a process holding at least one resource is waiting to acquire additional resources held by other processes.
- ❖ No preemption: a resource can be released only voluntarily by the process holding it, after that process has completed its task.
- ❖ Circular wait: there exists a set {P0, P1, …, P0} of waiting processes such that P0 is waiting for a resource that is held by P1, P1 is waiting for a resource that is held by P2, …, Pn–1 is waiting for a resource that is held by Pn, and P0 is waiting for a resource that is held by P0.

6. Explain about the methods used to Prevent Deadlocks

- ✓ Deadlock Prevention

- ❖ Mutual Exclusion – not required for sharable resources; must hold for nonsharable resources.
- ❖ Hold and Wait – must guarantee that whenever a process requests a resource, it does not hold any other resources.
- ❖ No Preemption
- ❖ Circular Wait – impose a total ordering of all resource types, and require that each process requests resources in an increasing order of enumeration.

7. Write in detail about Deadlock Avoidance.

- ❖ Multiple instances.
- ❖ Each process must a priori claim maximum use.
- ❖ When a process requests a resource it may have to wait.
- ❖ When a process gets all its resources it must return them in a finite amount of time.
- ❖ Data Structures for the Banker's Algorithm, Safety Algorithm
- ❖ Resource-Request Algorithm for Process Pi
- ❖ Example of Banker's Algorithm