Unit 4

Operating Systems

Syllabus

Operating Systems: Introduction to different types of operating Real Time Operating Systems, System Components, OS services, System structure- Layered Approach.

Process Management: Process Concept- Process states, Process control block, Threads, Process Scheduling: Types of process schedulers, Types of scheduling: Preemptive, Non preemptive. Scheduling algorithms: FCFS, SJF, RR, Priority,

Deadlocks: Methods of handling deadlocks, Deadlock prevention, avoidance and detection, Recovery from deadlocks.

Software

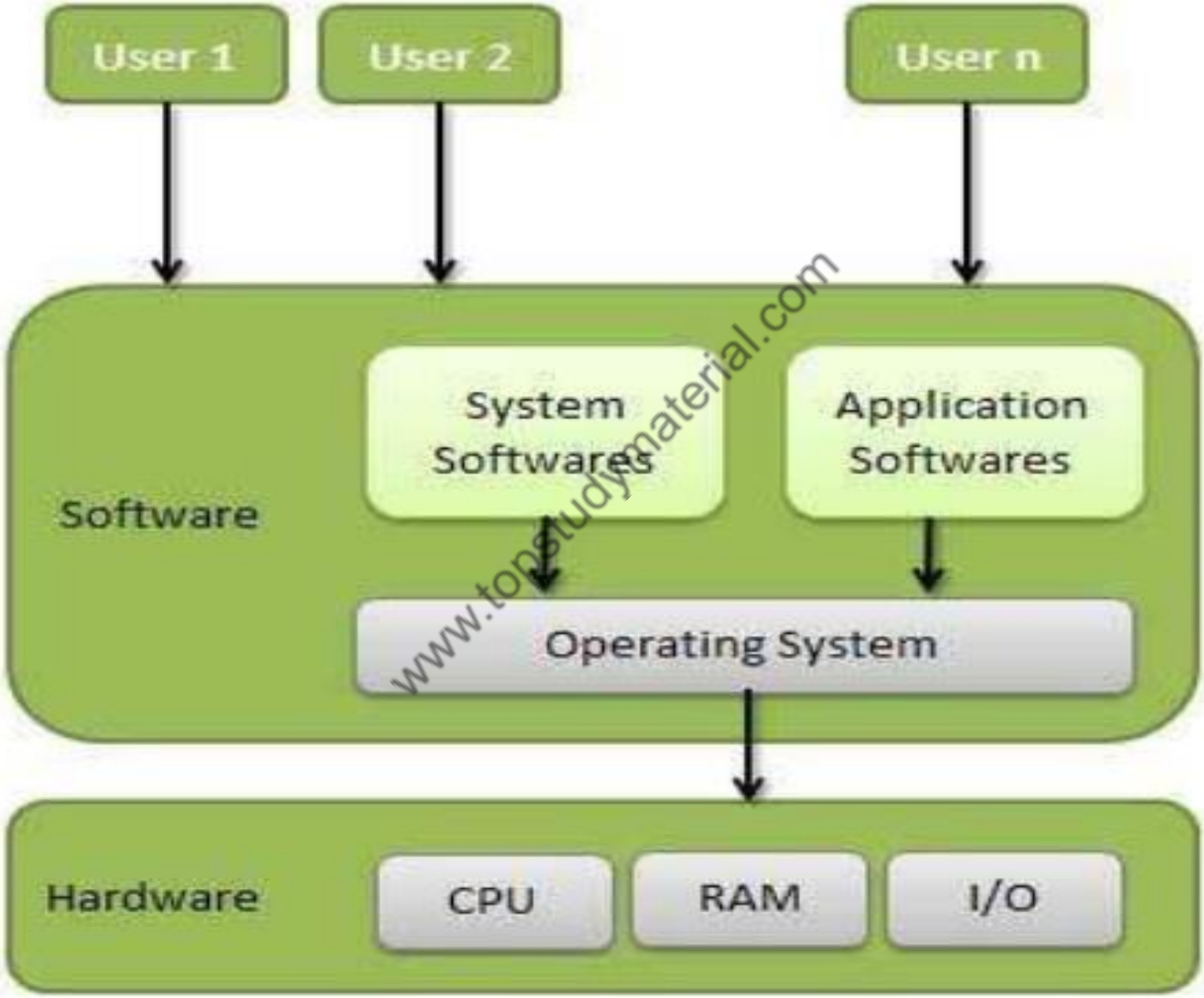
- Types of software:

i) **System Software:** Which manage the operation of computer itself.

Ii) **Application Software:** Which performs the actual work the user wants.

What is OS?

- An **operating system** is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.



Examples of OS

- Windows XP
- Windows 7
- Linux/Unix
- Windows 8

Functions of OS

Functions of OS

- Memory Management
- Processor Management
- Device Management
- File Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

Memory Management

- Memory management refers to management of Primary Memory or Main Memory.
- Main memory is a large array of words or bytes where each word or byte has its own address.
- Main memory provides a fast storage that can be accessed directly by the CPU.
- For a program to be executed, it must in the main memory.

An Operating System does the following activities for memory management:

- Keeps tracks of primary memory, i.e., what part of it are in use by whom, what part are not in use.

- In multiprogramming, the OS decides which process will get memory when and how much.

- Allocates the memory when a process requests it to do so.

- De-allocates the memory when a process no longer needs it or has been terminated.

Processor Management

- In multiprogramming environment, the OS decides which process gets the processor when and for how much time.
- This function is called **process scheduling**.
- An Operating System does the following activities for processor management:
- Keeps tracks of processor and status of process.
- The program responsible for this task is known as **traffic controller**.
- Allocates the processor (CPU) to a process.
- De-allocates processor when a process is no longer required.

Device Management

- An Operating System manages device communication via their respective drivers.
- It does the following activities for device management:
- Keeps tracks of all devices. The program responsible for this task is known as the **I/O controller**.
- Decides which process gets the device when and for how much time.
- Allocates the device in the most efficient way.
- De-allocates devices.

File Management

- A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions.
- An Operating System does the following activities for file management:

- Keeps track of information, location, uses, status etc. The collective facilities are often known as **file system**.

- Decides who gets the resources.

- Allocates the resources.

- De-allocates the resources.

- **Security** -- By means of password and similar other techniques, it prevents unauthorized access to programs and data.

- **Control over system performance** -- Recording delays between request for a service and response from the system.

- **Job accounting** -- Keeping track of time and resources used by various jobs and users.
- **Error detecting aids** -- Production of dumps, traces, error messages, and other debugging and error detecting aids.
- **Coordination between other software and users**
- Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems

Operating System – Services

Operating System – Services

- An Operating System provides services to both the users and to the programs.
- It provides programs an environment to execute.
- It provides users the services to execute the programs in a convenient manner.

- Following are a few common **services** provided by an operating system:
- Program execution
- I/O operations
- File System manipulation
- Communication
- Error Detection
- Resource Allocation
- Protection

Program Execution

- Following are the major activities of an operating system with respect to program management:
- Loads a program into memory
- Executes the program
- Handles program's execution
- Provides a mechanism for process synchronization
- Provides a mechanism for process communication.

I/O Operation

- An Operating System manages the communication between user and device drivers.
- I/O operation means read or write operation with any file or any specific I/O device.
- Operating system provides the access to the required I/O device when required.

File System Manipulation

- Following are the major activities of an operating system with respect to file management:
- Program needs to read a file or write a file.
- The operating system gives the **permission to the program for operation on file.**
- Permission varies from **read-only, read-write, denied,** and so on.
- Operating System provides an interface to the user to **create/delete files.**
- Operating System provides an interface to the user to **create/delete directories.**
- Operating System provides an interface to create the **backup of file system.**

Communication

- Following are the major activities of an operating system with respect to communication:
- Two processes often require **data to be transferred** between them.
- Both the processes can be on one computer or on different computers, but are connected through a computer network.
- Communication may be implemented by two methods, either by Shared Memory or by Message Passing.

Error Handling

- Errors can occur anytime and anywhere. An error may occur in CPU, in I/O devices or in the memory hardware.
- Following are the major activities of an operating system with respect to error handling:
- The OS constantly checks for possible errors.
- The OS takes an appropriate action to ensure correct and consistent computing.

Resource Management

- In case of multi-user or multi-tasking environment, resources such as main memory, CPU cycles and files storage are to be allocated to each user or job.
- Following are the major activities of an operating system with respect to resource management:
- The OS manages all kinds of resources using schedulers.
- CPU scheduling algorithms are used for better utilization of CPU.

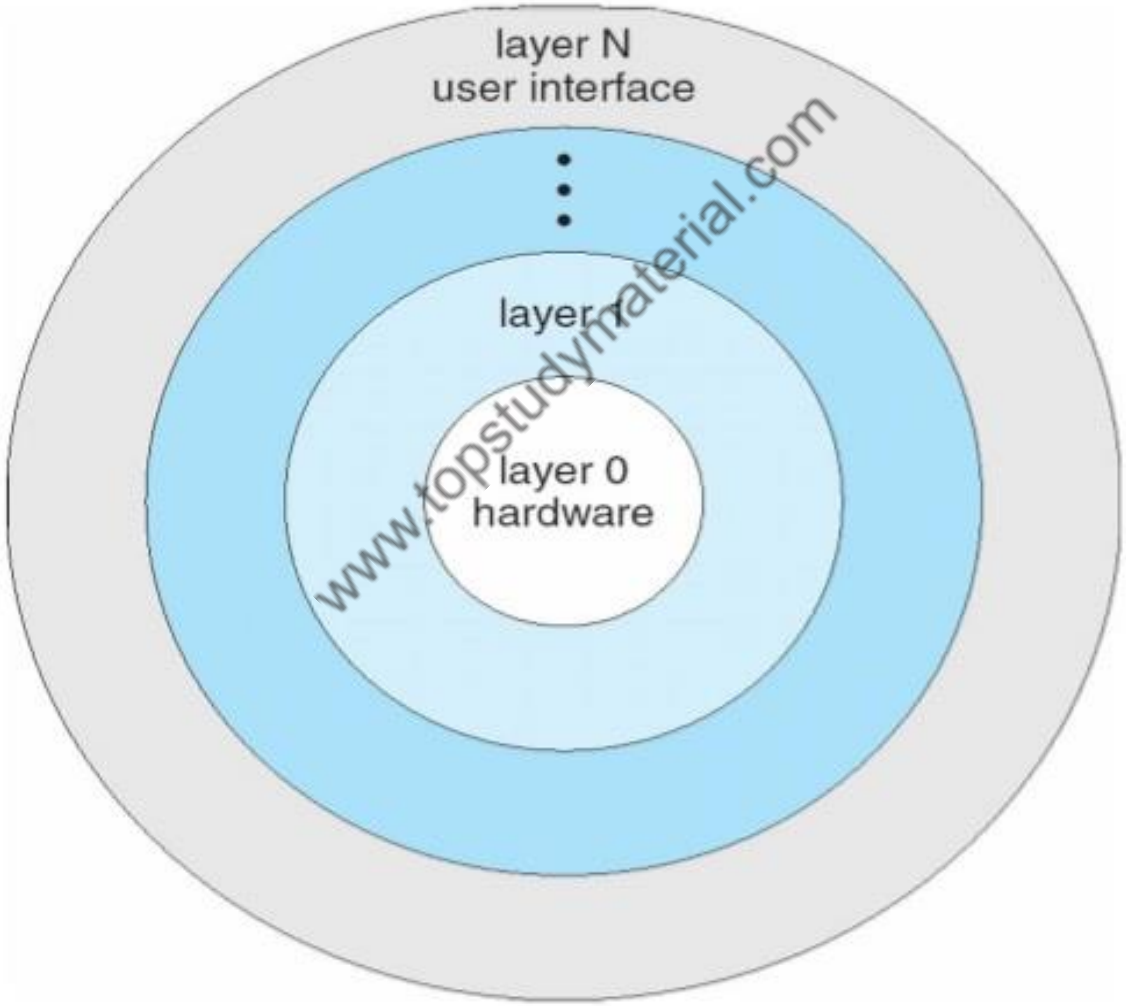
Protection

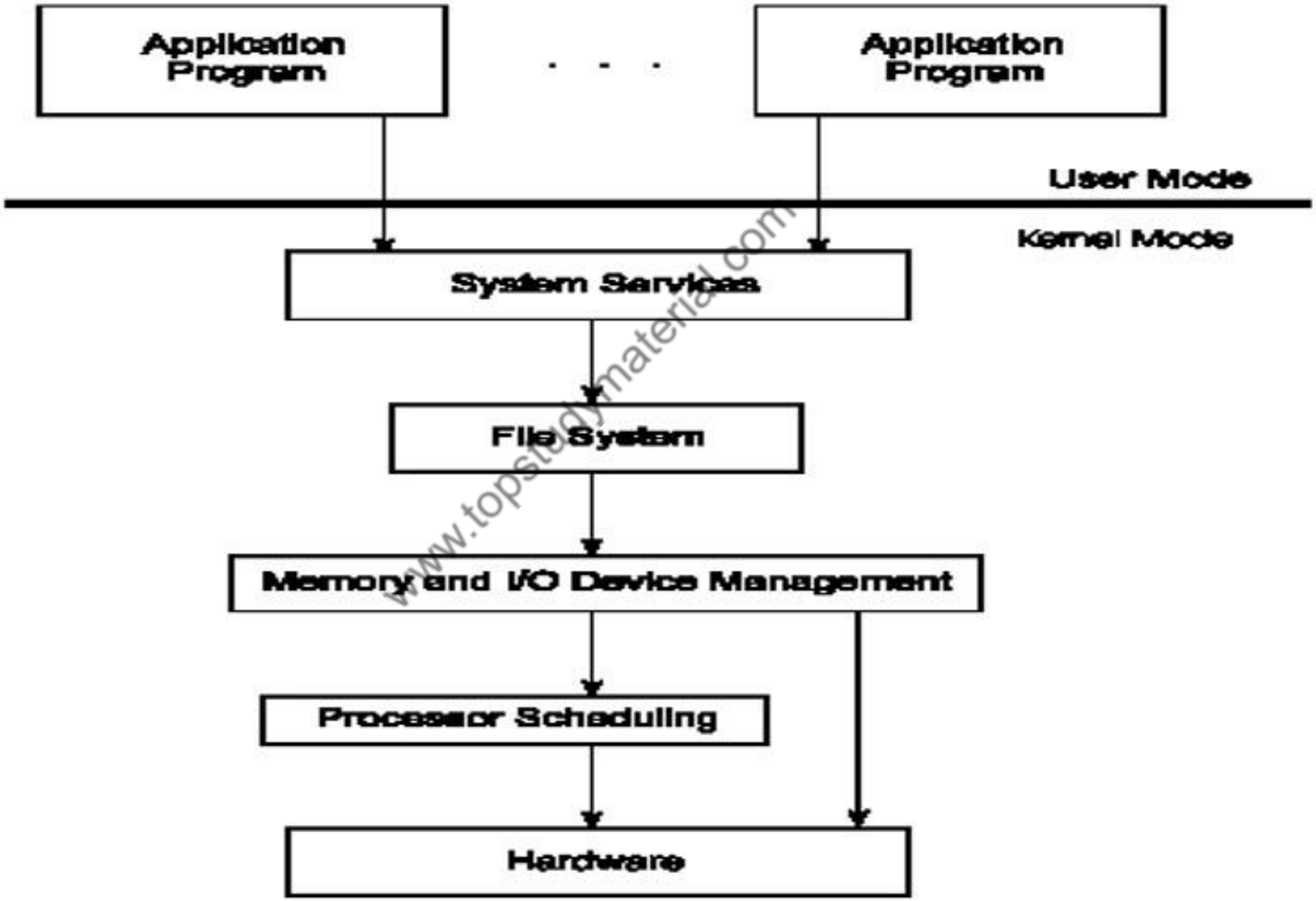
- Following are the **major activities** of an operating system with respect to protection:
- The OS ensures that all access to system resources is controlled.
- The OS ensures that external I/O devices are protected from invalid access attempts.
- The OS provides authentication features for each user by means of passwords.

System Structure-Layered Approach

System Structure-Layered Approach

- The operating system is divided into a number of layers (levels), each built on top of lower layers.
- The bottom layer (layer 0) is the hardware; the highest (layer N) is the user interface.
- With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers.





Types Of OS

Types Of OS(H.W.)

- Batch Operating System
- Time Sharing Operating System
- Distributed Operating System
- Network Operating System
- **Real Time Operating System**

Syllabus

Operating Systems: Introduction to different types of operating Real Time Operating Systems, System Components, OS services, System structure- Layered Approach.

Process Management: Process Concept- Process states, Process control block, Threads, Process Scheduling: Types of process schedulers, Types of scheduling: Preemptive, Non preemptive. Scheduling algorithms: FCFS, SJF, RR, Priority,

Deadlocks: Methods of handling deadlocks, Deadlock prevention, avoidance and detection, Recovery from deadlocks.

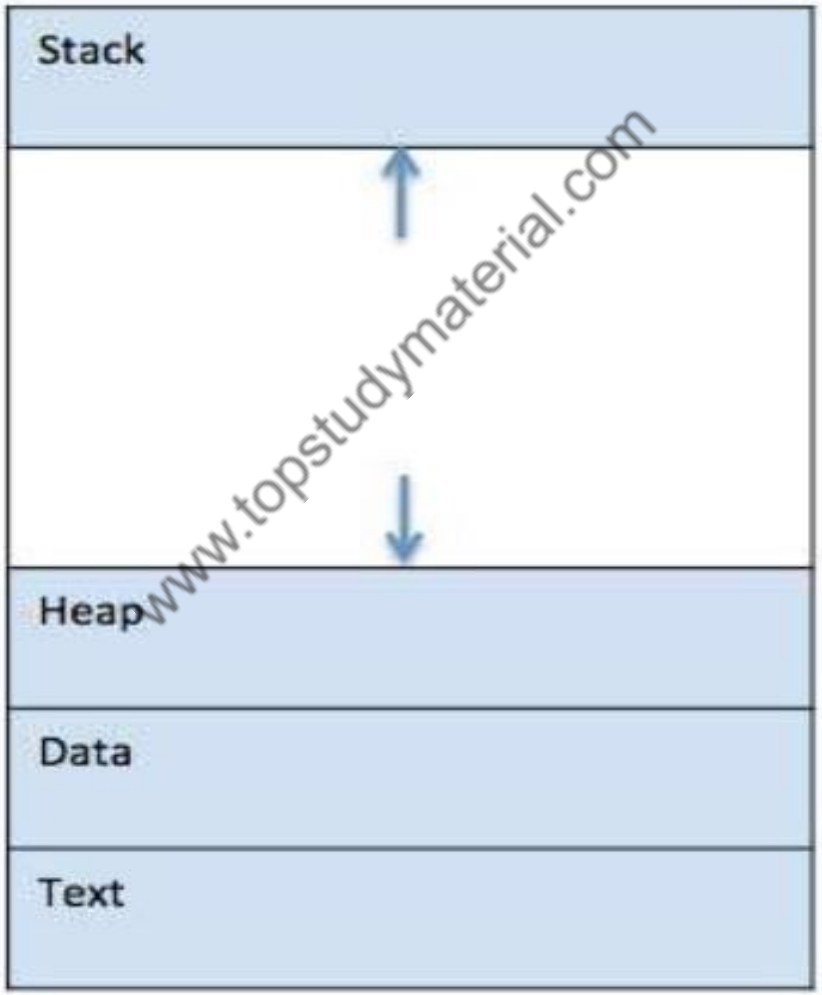
Process Management

Process

- A process is basically a program in execution.
- The execution of a process must progress in a sequential fashion.
- We write our computer programs in a text file and when we execute this program, it becomes a process which performs all the tasks mentioned in the program.

- When a program is loaded into the memory and it becomes a process, it can be divided into four sections – stack, heap, text and data.

Process in memory



Process

S.N.	Component & Description
1	**Stack** The process Stack contains the temporary data such as method/function parameters, return address, and local variables.
2	**Heap** This is a dynamically allocated memory to a process during its runtime.
3	**Text** This includes the current activity represented by the value of Program Counter and the contents of the processor's registers.
4	**Data** This section contains the global and static variables.

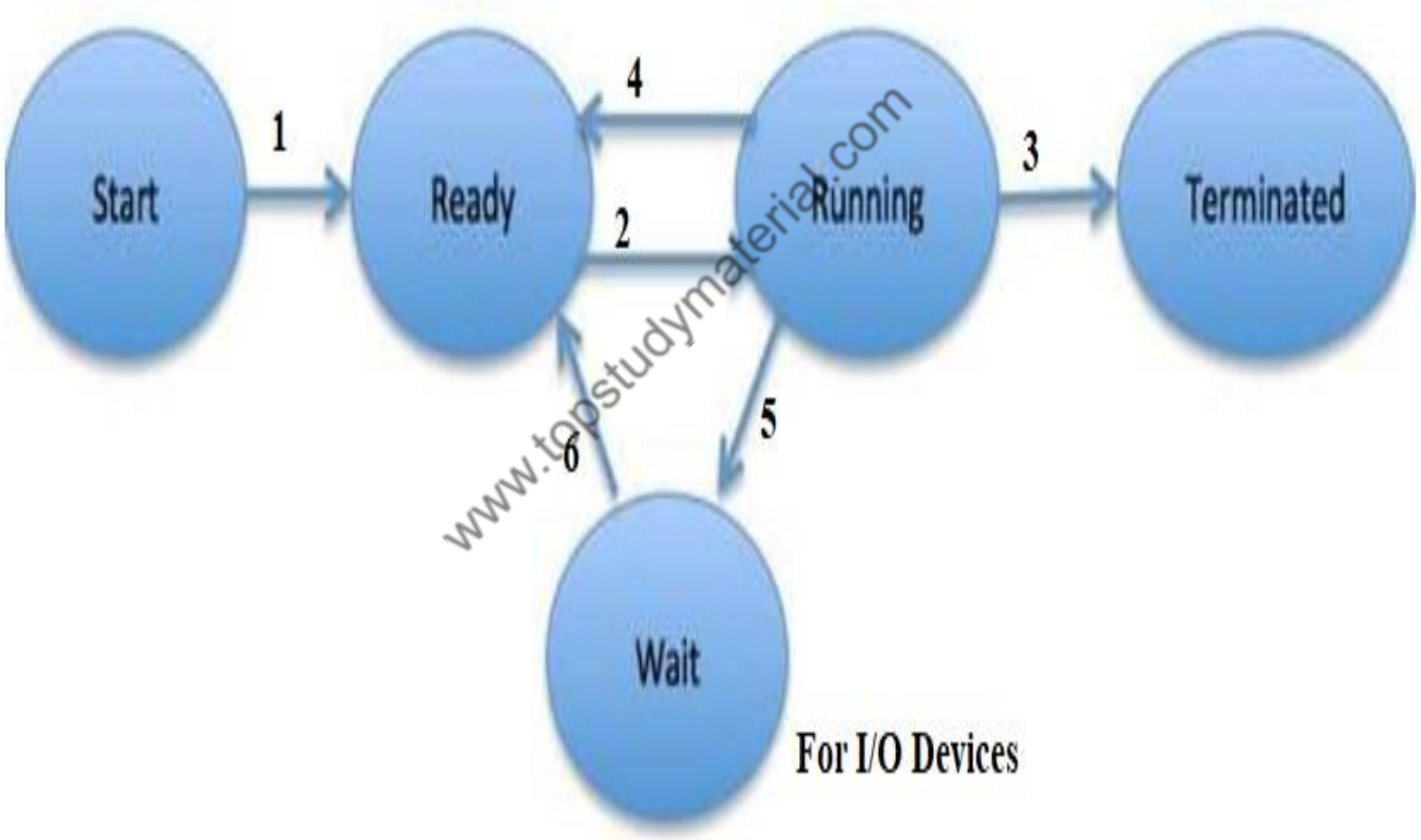
Process State Diagram

- When a process executes, it passes through different states.

- These stages may differ in different operating systems, and the names of these states are also not standardized.

- In general, a process can have one of the following five states at a time.

Process State Diagram



- **Start:** This is the initial state when a process is first started/created.
- **Ready:** The process is waiting to be assigned to a processor.
- Ready processes are waiting to have the processor allocated to them by the operating system so that they can run.
- Process may come into this state after **Start** state or while running it by but interrupted by the scheduler to assign CPU to some other process.

- **Running:** Once the process has been assigned to a processor by the OS scheduler, the process state is set to running and the processor executes its instructions.

- **Waiting:** Process moves into the waiting state if it needs to wait for a resource, such as waiting for user input, or waiting for a file to become available.

- **Terminated or Exit:** Once the process finishes its execution, or it is terminated by the operating system, it is moved to the terminated state where it waits to be removed from main memory.

Process Control Block

- A **Process Control Block** is a data structure maintained by the Operating System for every process.
- The PCB is identified by an integer process ID (PID).

Diagram of Process Control Block

S.N.	Information & Description
1	**Process State** The current state of the process i.e., whether it is ready, running, waiting, or whatever.
2	**Process privileges** This is required to allow/disallow access to system resources.
3	**Process ID** Unique identification for each of the process in the operating system.
4	**Pointer** A pointer to parent process.

5	**Program Counter** Program Counter is a pointer to the address of the next instruction to be executed for this process.
6	**CPU registers** Various CPU registers where process need to be stored for execution for running state.
7	**CPU Scheduling Information** Process priority and other scheduling information which is required to schedule the process.
8	**Memory management information** This includes the information of page table, memory limits, Segment table depending on memory used by the operating system.
9	**Accounting information** This includes the amount of CPU used for process execution, time limits, execution ID etc.
10	**IO status information** This includes a list of I/O devices allocated to the process.

Process Scheduling Queues

Process Scheduling Queues

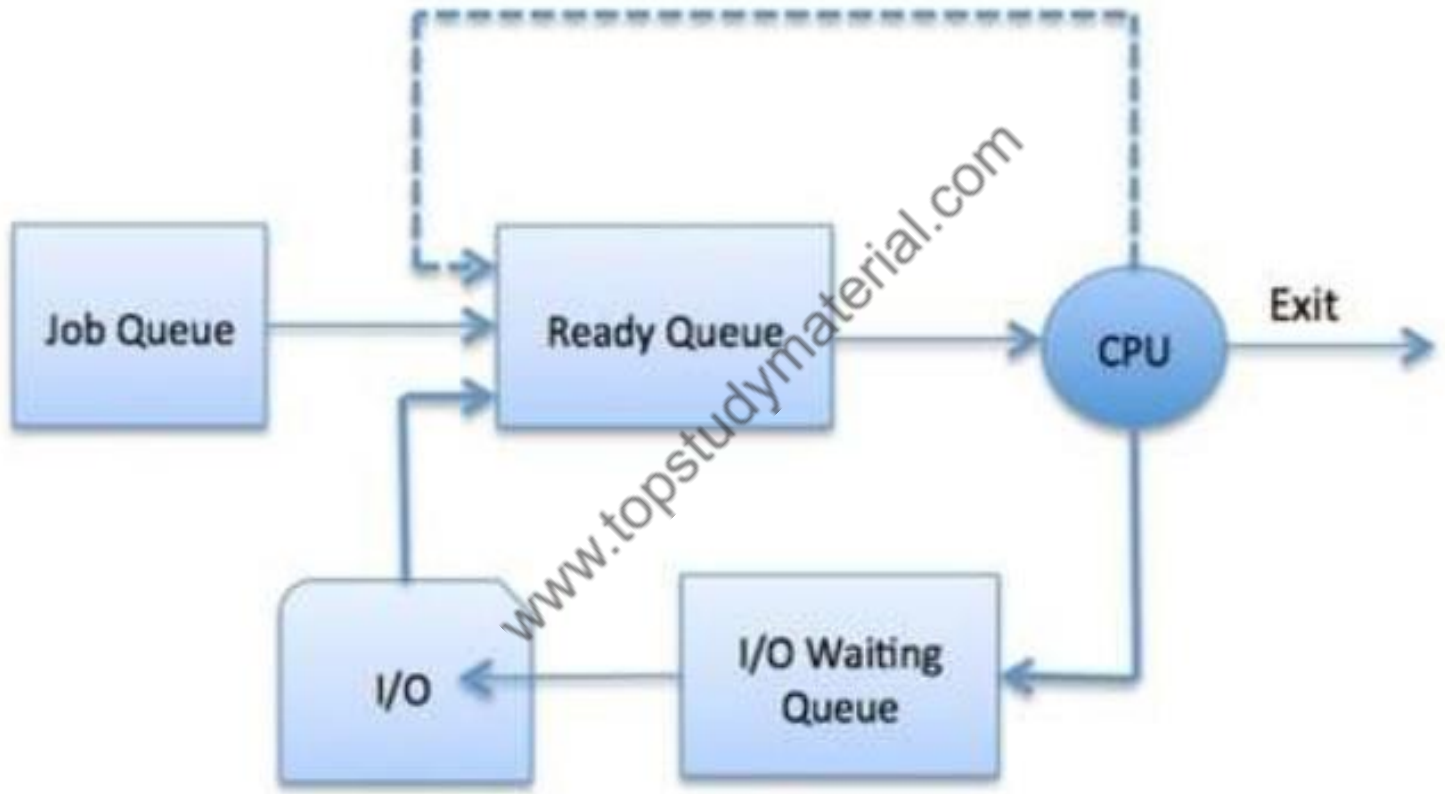
- The Operating System maintains the following important process scheduling queues:

- **Job queue -** This queue keeps all the processes in the system.

- **Ready queue -** This queue keeps a set of all processes residing in main memory, ready and waiting to execute. A new process is always put in this queue.

- **Device queues -** The processes which are blocked due to unavailability of an I/O device constitute this queue.

Process Scheduling Queues



The OS can use different policies to manage each queue (FIFO, Round Robin, Priority, etc.).

Process Schedulers

- **Schedulers:** Schedulers are special system software which handle process scheduling in various ways.
- Their main task is to select the jobs to be submitted into the system and to decide which process to run. **Schedulers are of three types:**
- Long-Term Scheduler
- Short-Term Scheduler
- Medium-Term Scheduler

S.N.	Long-Term Scheduler	Short-Term Scheduler	Medium-Term Scheduler
1	It is a job scheduler	It is a CPU scheduler	It is a process swapping scheduler.
2	Speed is lesser than short term scheduler	Speed is fastest among other two	Speed is in between both short and long term scheduler.
3	It controls the degree of multiprogramming	It provides lesser control over degree of multiprogramming	It reduces the degree of multiprogramming.
4	It is almost absent or minimal in time sharing system	It is also minimal in time sharing system	It is a part of Time sharing systems.
5	It selects processes from pool and loads them into memory for execution	It selects those processes which are ready to execute	It can re-introduce the process into memory and execution can be continued.

Types Of Process Scheduling Algorithms

- A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms.
- There are some popular process scheduling algorithms:
- **First-Come, First-Served (FCFS) Scheduling**
- **Shortest-Job-Next (SJN) Scheduling**
- **Priority Scheduling**
- **Shortest Remaining Time**
- **Round Robin(RR) Scheduling**

- **Non-preemptive algorithms** are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time.
- **Preemptive scheduling** is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

First Come, First Served (FCFS)

- Jobs are executed on first come, first served basis.
- It is a non-preemptive scheduling algorithm.
- Easy to understand and implement.
- Its implementation is based on FIFO queue.
- Poor in performance, as average wait time is high.

Shortest Job First (SJF)

- This is also known as **shortest job Next**, or SJN.
- This is a non-preemptive scheduling algorithm.
- Best approach to minimize waiting time.
- Easy to implement in Batch systems where required CPU time is known in advance.
- Impossible to implement in interactive systems where the required CPU time is not known.
- The processer should know in advance how much time a process will take.

Priority scheduling

- Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
- Each process is assigned a priority. Process with highest priority is to be executed first and so on.
- Processes with same priority are executed on first come first served basis.
- Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Shortest Remaining Time

- Shortest remaining time (SRT) is the preemptive version of the SJN algorithm.
- The processor is allocated to the job closest to completion but it can be preempted by a newer ready job with shorter time to completion.
- Impossible to implement in interactive systems where required CPU time is not known.
- It is often used in batch environments where short jobs need to be given preference.

Round Robin Scheduling

- Round Robin is a preemptive process scheduling algorithm.
- Each process is provided a fix time to execute; it is called a **quantum**.
- Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
- Context switching is used to save states of preempted processes.

Threads

- A thread is called a **lightweight process**.
- Threads provide a way to improve application performance through parallelism.
- Each thread belongs to exactly one process and no thread can exist outside a process.
- Each thread represents a separate flow of control.
- Threads have been successfully used in implementing network servers and web server.

Difference between Process and Thread

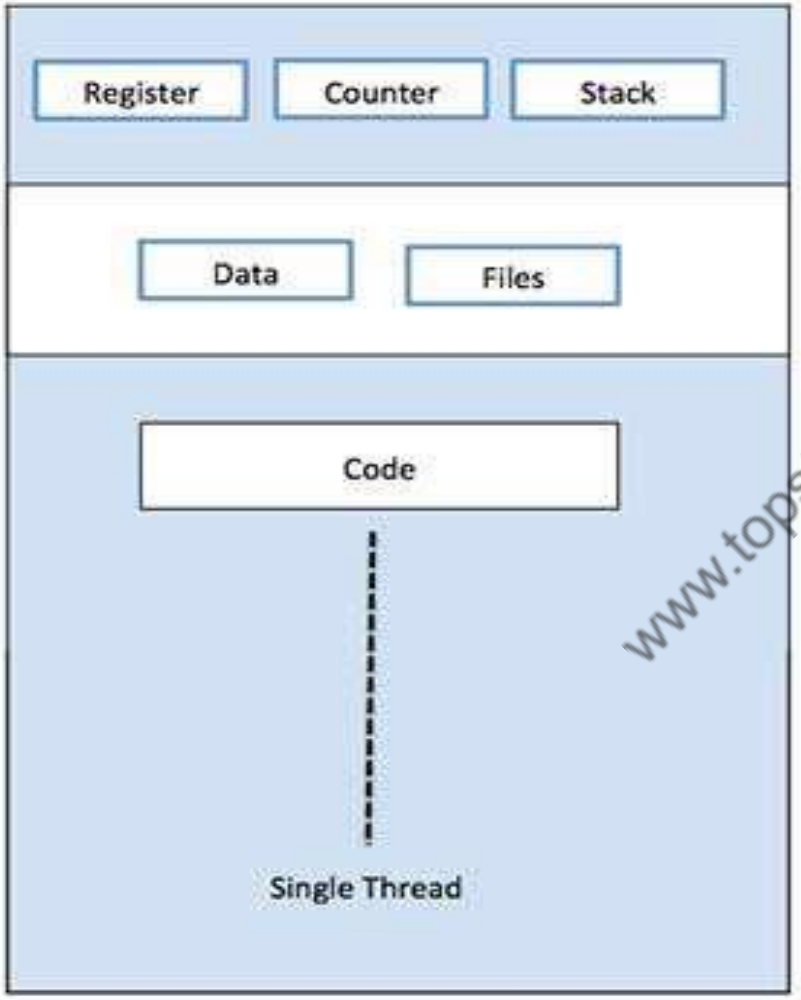
S.N.	Process	Thread
1	Process is heavy weight or resource intensive.	Thread is lightweight, taking lesser resources than a process.
2	Process switching needs interaction with operating system.	Thread switching does not need to interact with operating system.
3	In multiple processing environments, each process executes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes.
4	If one process is blocked, then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, a second thread in the same task can run.
5	Multiple processes without using threads use more resources.	Multiple threaded processes use fewer resources.
6	In multiple processes each process operates independently of the others.	One thread can read, write or change another thread's data.

Advantages of Thread

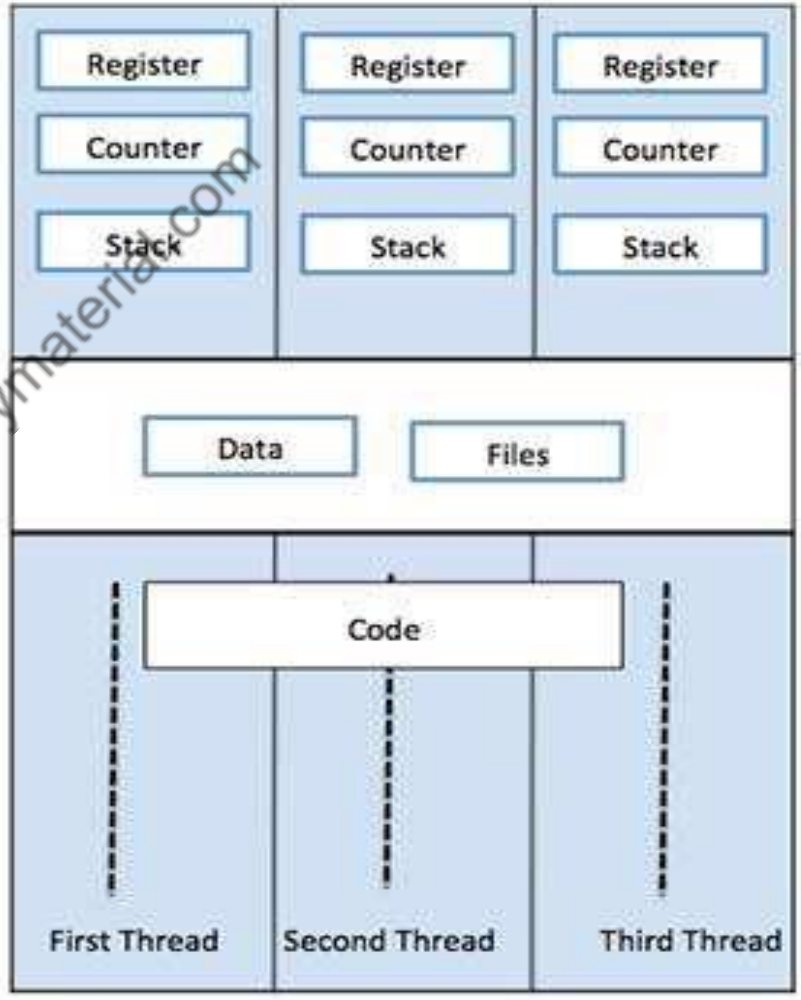
- Threads minimize the context switching time.
- Use of threads provides concurrency within a process.
- Efficient communication.
- It is more economical to create and context switch threads.
- Threads allow utilization of multiprocessor architectures to a greater scale and efficiency.

Types of Thread

- Threads are implemented in following two ways:
- **User Level Threads** -- User managed threads
- **Kernel Level Threads** -- Operating System managed threads acting on kernel, an operating system core.



Single Process P with single thread



Single Process P with three threads

Multithreading Models

- Multithreading models are three types:

- **Many-to-many relationship**
- **Many-to-one relationship**
- **One-to-one relationship**

Many-to-Many Model

- The many-to-many model multiplexes any number of user threads onto an equal or smaller number of kernel threads.

Many-to-One Model

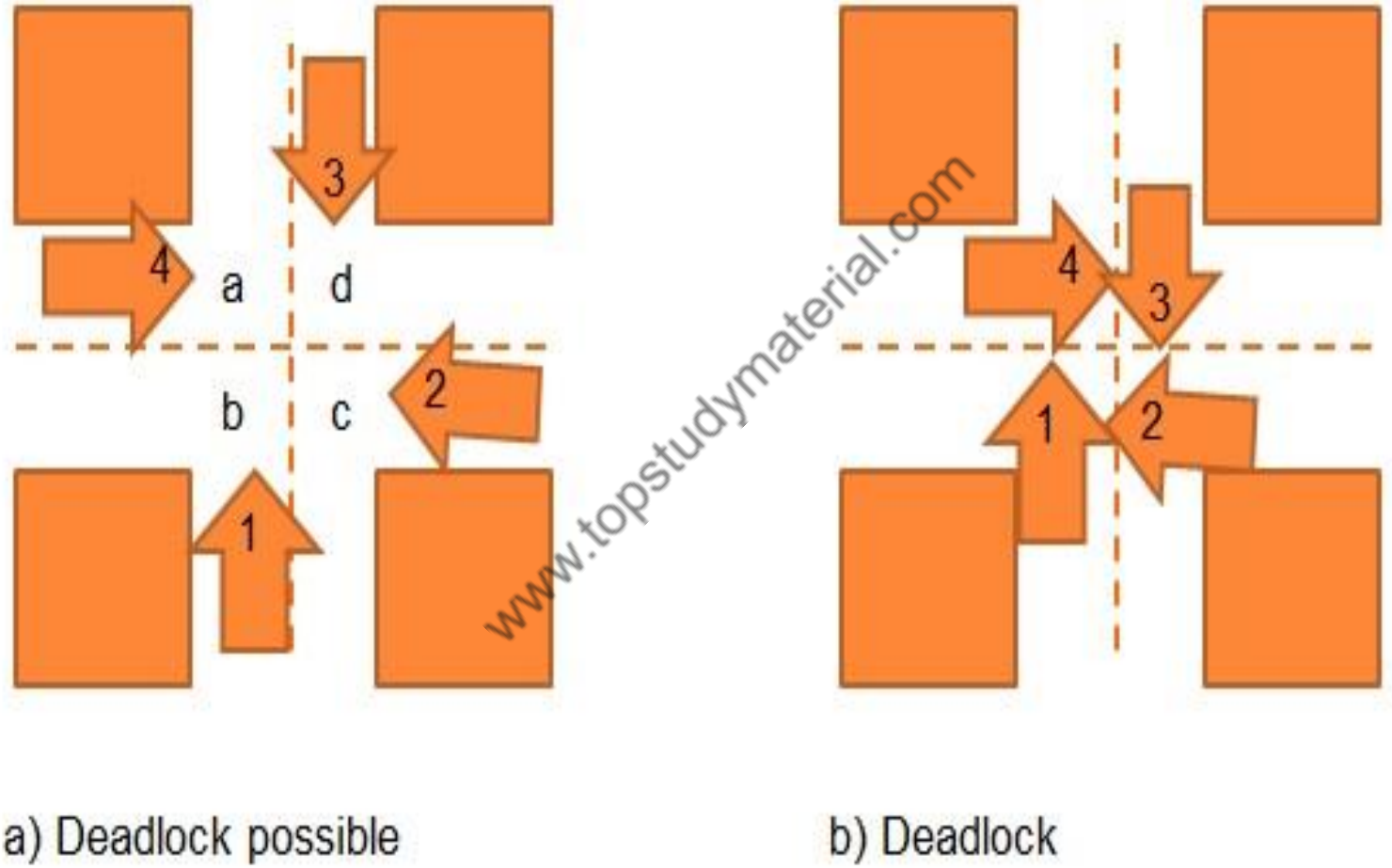
- Many-to-one model maps many user level threads to one Kernel-level thread.
- Thread management is done in user space by the thread library.

One-to-One Model

- There is one-to-one relationship of user-level thread to the kernel-level thread.
- This model provides more concurrency than the many-to-one model.

Deadlock

Deadlock

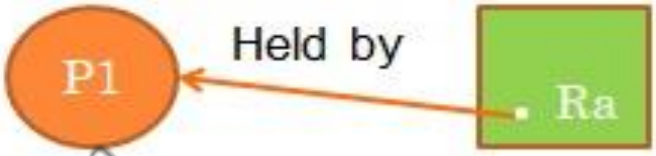


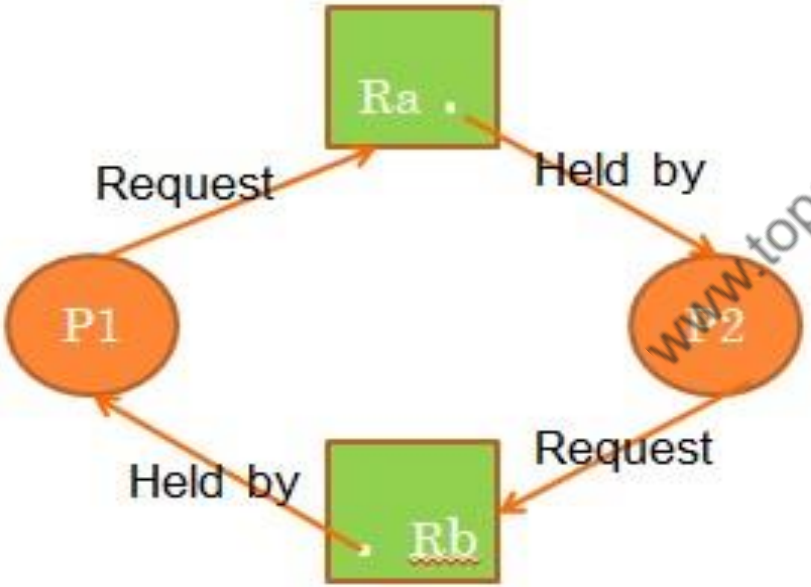
a) Deadlock possible

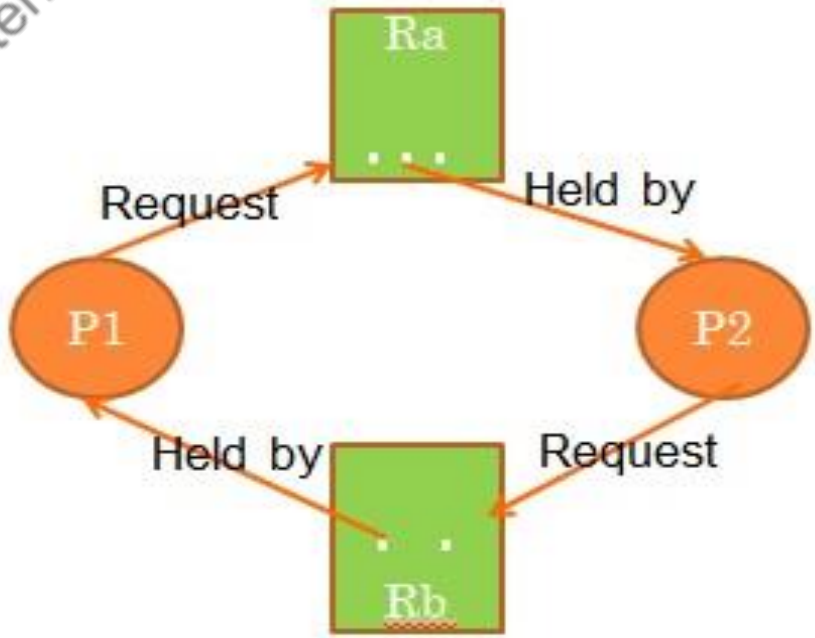
b) Deadlock

Resource Allocation Graphs

- A useful tool in characterizing the allocation of resources to processes is the resource allocation graph.
- It is a directed graph that depicts a state of the system of resources & processes, with each process & each resource represented by a node.
- A graph edge directed from a process to a resource indicates a resource that has been requested by the process but not yet granted.
- Within a resource node a dot is shown for each instance of that resource.
- A graph edge directed from reusable resource node dot to a process indicates a request that has been granted.







The Conditions for Deadlock

- 3 conditions of policy must be present for a deadlock to be possible:
- **Mutual Exclusion:** Only 1 process may use a resource at a time. No process may access a resource unit that has been allocated to another process.
- **Hold & Wait:** A process may hold allocated resources while awaiting assignment of other resources.
- **No preemption:** No resource can be forcibly removed from a process holding it.
- **Circular wait:** A closed chain of processes exists, such that process holds at least one resource needed by the next process in the chain.

Deadlock Prevention

- **Indirect method** of deadlock prevention is to prevent the occurrence of one of the three necessary conditions.

- **Direct method** is to prevent the occurrence of circular wait

- Mutual Exclusion
- Hold & Wait
- No Preemption
- Circular Wait

Deadlock Avoidance

- Do not start a process if its demands might lead to deadlock.

- Do not grant an incremental resource request to a process if this allocation might lead to deadlock.

Process initiation denial(Bankers Algo)

Consider the system of n processes & m different types of resources.

Resource = R = (R1, R2,.....,Rm)

Available = V = (V1, V2,.....,Vm)

Claim = C =

C11	C12	.....	C1m
C21	C22	.....	C2m
:			
Cn1	Cn2	.....	Cnm

Allocation = A =

A11	A12	....	A1m
A21	A22	....	A2m
:			
An1	An2	.....	Anm

- Following relationships hold:

1) $R_j = V_j + \sum A_{ij}$ $i = 1$ to n for all j
2) $C_{ij} <= R_j$ for all i,j
3) $A_{ij} <= C_{ij}$ for all i,j

Start a new process only if

$R_j >= C_{(n+1)j} + \sum C_{ij}$ if $i=1$ to n for all j

That is a process is only started if the maximum claim of all current processes plus those of the new process can be met

Bankers Algorithm

- This strategy is call banker's algorithm.

	R1	R2	R3
P1	3	2	2
P2	6	1	3
P3	3	1	4
P4	4	2	2

Claim Matrix C

	R1	R2	R3
P1	1	0	0
P2	6	1	2
P3	2	1	1
P4	0	0	2

Allocation Matrix A

	R1	R2	R3
P1	2	2	2
P2	0	0	1
P3	1	0	3
P4	4	2	0

C-A

R1	R2	R3
9	3	6

Resource Vector R

R1	R2	R3
0	1	1

Available Vector V

(a) Initial State

	R1	R2	R3
P1	3	2	2
P2	6	1	3
P3	3	1	4
P4	4	2	2

Claim Matrix C

	R1	R2	R3
P1	1	0	0
P2	0	0	0
P3	2	1	1
P4	0	0	2

Allocation Matrix A

	R1	R2	R3
P1	2	2	2
P2	0	0	0
P3	1	0	3
P4	4	2	0

C-A

R1	R2	R3
9	3	6

Resource Vector R

R1	R2	R3
6	2	3

Available Vector V

	R1	R2	R3
P1	3	2	2
P2	6	1	3
P3	3	1	4
P4	4	2	2

Claim Matrix C

	R1	R2	R3
P1	0	0	0
P2	0	0	0
P3	2	1	1
P4	0	0	2

Allocation Matrix A

	R1	R2	R3
P1	0	0	0
P2	0	0	0
P3	1	0	3
P4	4	2	0

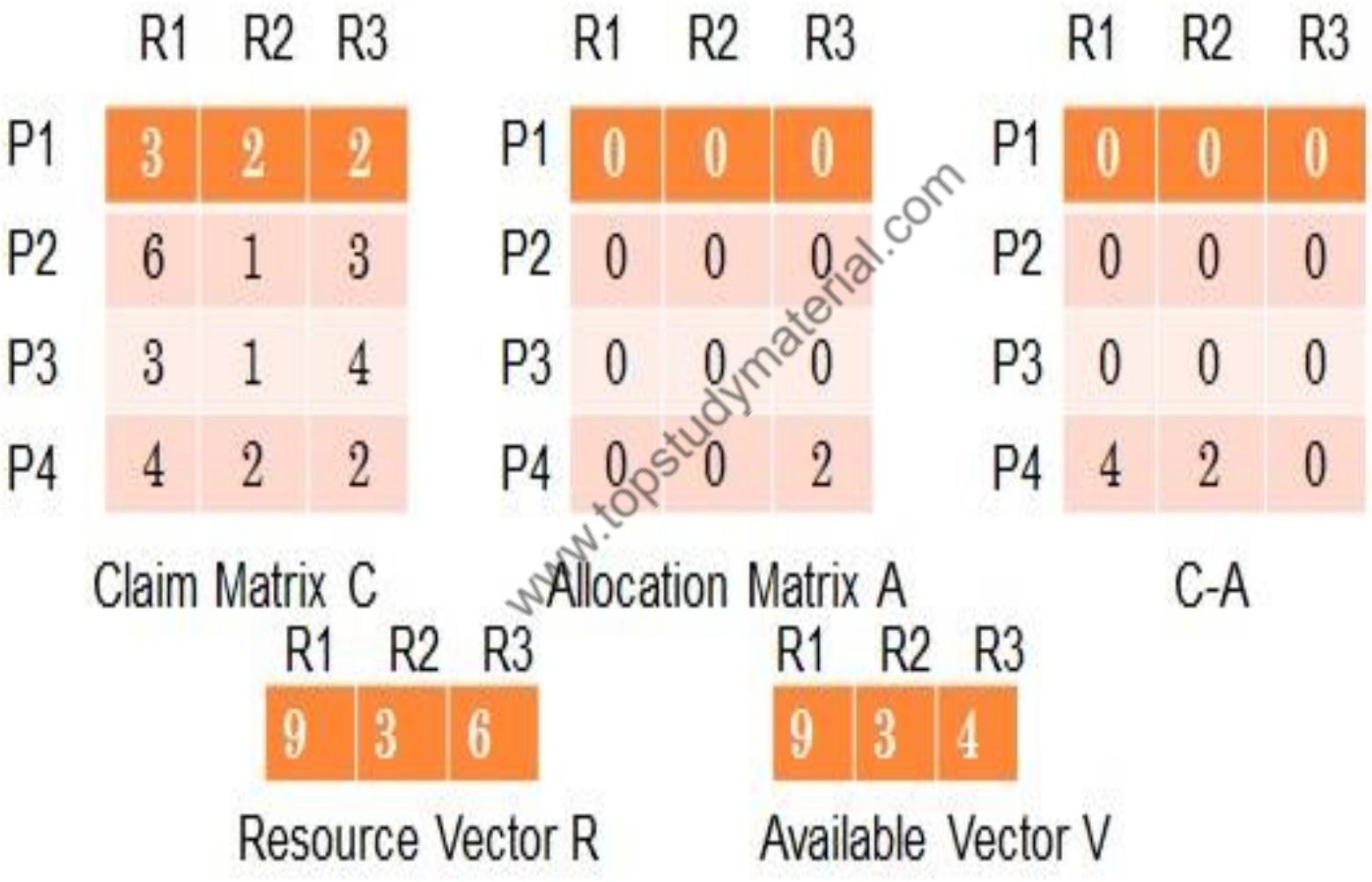
C-A

R1	R2	R3
9	3	6

Resource Vector R

R1	R2	R3
7	2	3

Available Vector V



Deadlock Detection

	R1	R2	R3	R4	R5
P1	0	1	0	0	1
P2	0	0	1	0	1
P3	0	0	0	0	1
P4	1	0	1	0	1

Request Matrix Q

	R1	R2	R3	R4	R5
P1	1	0	1	1	0
P2	1	1	0	0	0
P3	0	0	0	1	0
P4	0	0	0	0	0

Allocation Matrix A

R1	R2	R3	R4	R5
2	1	1	2	1

Resource Vector

R1	R2	R3	R4	R5
0	0	0	0	1

Allocation Vector

1. Mark P4, because P4 has no allocated resources
2. Set W = (0 0 0 0 1).
3. The request of process P3 is less than or equal to W, so mark P3 & set W = W+(0 0 0 1 0) = (0 0 0 1 1).
4. No other unmarked process has a row in Q that is less than or equal to W. Therefore terminate the algorithm.

The algorithm concludes with P1 & P2 unmarked, indicating that these processes are deadlocked.

Deadlock Recovery

- There are various ways of recovery from deadlock.

1. Recovery through preemption.
2. Recovery through rollback
3. Recovery through killing process.

Deadlock Recovery

- Following are possible approaches for recovery once deadlock is detected.
  - Abort all deadlocked processes.
  - Back up each deadlocked process to some previously defined checkpoint & restart all processes. This requires that rollback & restart mechanisms be built in to the system.
  - Successively abort deadlocked process until deadlock no longer exists. Detection algorithm needs to be reinvoked.
  - Successively preempt resources until deadlock no longer exist. A process that has a resource preempted from it must be rolled back to a point prior to its acquisition of that resource.