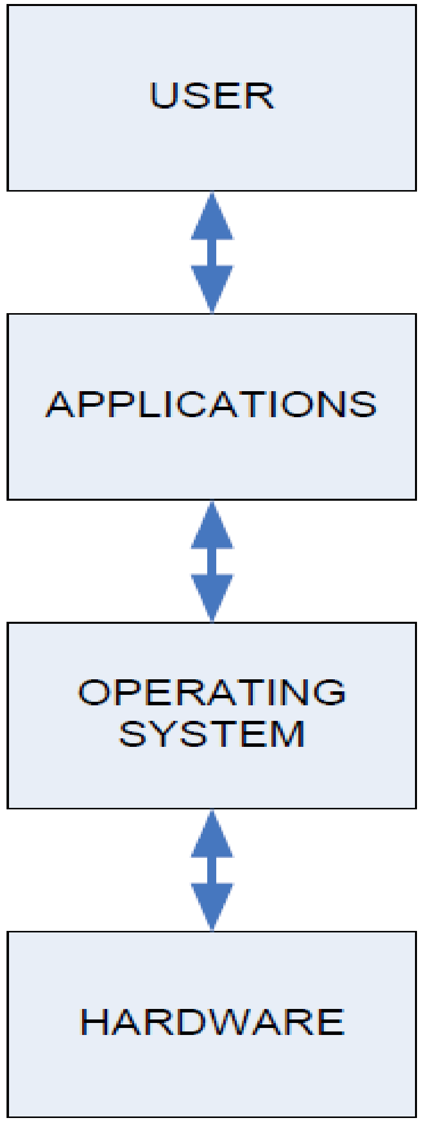
Operating System (OS)

What is an Operating System?

- Computer System = Hardware + Software
- Software = Application Software + System Software(OS)
- An Operating System is a system Software that acts as an intermediary/interface between a user of a computer and the computer hardware.
- Operating system goals:
  - ➢ Execute user programs and make solving user problems easier
  - ➢ Make the computer system convenient to use
  - ➢ Use the computer hardware in an efficient manner

The Structure of Computer Systems

- Accessing computer resources is divided into *layers*.
- Each layer is isolated and only interacts directly with the layer below or above it.
- If we install a new hardware device
  - ✓ No need to change anything about the user/applications.
  - ✓ However, you do need to make changes to the operating system.
  - ✓ You need to install the device drivers that the operating system will use to control the new device.
- If we install a new software application
  - ✓ No need to make any changes to your hardware.
  - ✓ But we need to make sure the application is supported by the operating system
  - ✓ user will need to learn how to use the new application.
- If we change the operating system
  - ✓ Need to make sure that both applications and hardware will compatible with the new operating system.



CPU – Central Processing Unit

- This is the brain of your computer.
- It performs all of the calculations.
- In order to do its job, the CPU needs commands to perform, and data to work with.
- The instructions and data travel to and from the CPU on the system bus.
- The operating system provides rules for how that information gets back and forth, and how it will be used by the CPU.

RAM – Random Access Memory

- This is like a desk, or a workspace, where your computer temporarily stores all of the information (data) and instructions (software or program code) that it is currently using.
- Each RAM chip contains millions of address spaces.
- Each address space is the same size, and has its own unique identifying number (address).
- The operating system provides the rules for using these memory spaces, and controls storage and retrieval of information from RAM.
- Device drivers for RAM chips are included with the operating system.

Problem: *If RAM needs an operating system to work, and an operating system needs RAM in order to work, how does your computer activate its RAM to load the operating system?*

History of Operating System

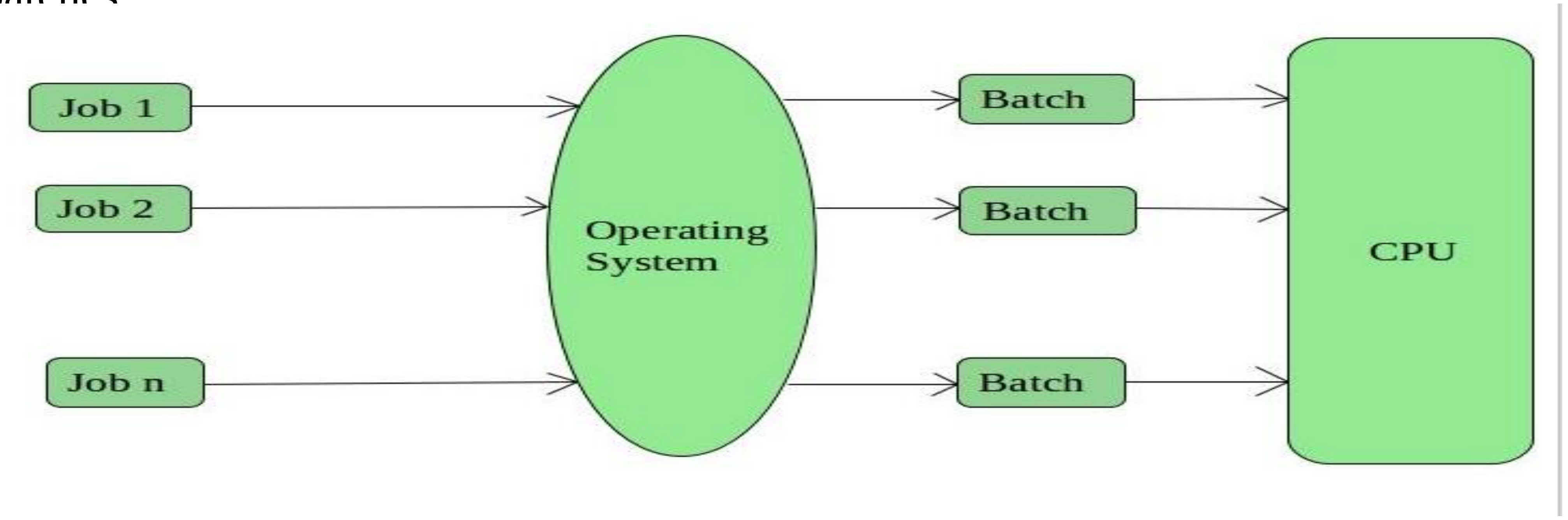
- **The First Generation (1940's to early 1950's)**
  - No Operating System
  - All programming was done in absolute machine language, often by wiring up plug-boards to control the machine’s basic functions.
- **The Second Generation (1955-1965)**
  - First operating system was introduced in the early 1950's.It was called GMOS
  - Created by General Motors for IBM's machine the 701.
  - Single-stream batch processing systems
- **The Third Generation (1965-1980)**
  - Introduction of multiprogramming
  - Development of Minicomputer
- **The Fourth Generation (1980-Present Day)**
  - Development of PCs
  - Birth of Windows/MaC OS

Types of Operating Systems

1. Batch Operating System
2. Multiprogramming Operating System
3. Time-Sharing OS
4. Multiprocessing OS
5. Distributed OS
6. Network OS
7. Real Time OS
8. Embedded OS

1. Batch Operating System

- The users of this type of operating system does not interact with the computer directly.
- Each user prepares his job on an off-line device like punch cards and submits it to the computer operator
- There is an operator which takes similar jobs having the same requirement and group them into batches



1. Batch Operating System cont..

Advantages of Batch Operating System:

- Processors of the batch systems know how long the job would be when it is in queue
- Multiple users can share the batch systems
- The idle time for the batch system is very less
- It is easy to manage large work repeatedly in batch systems

Disadvantages of Batch Operating System:

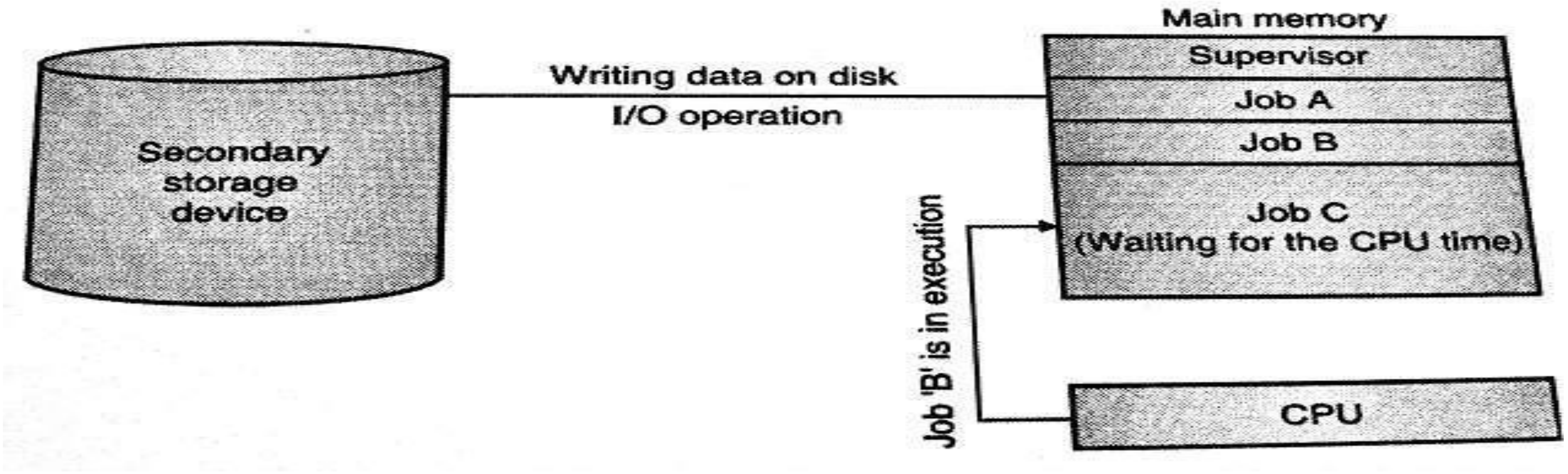
- The computer operators should be well known with batch systems
- Batch systems are hard to debug
- It is sometimes costly
- The other jobs will have to wait for an unknown time if any job fails

Examples of Batch based Operating System:

IBM's MVS

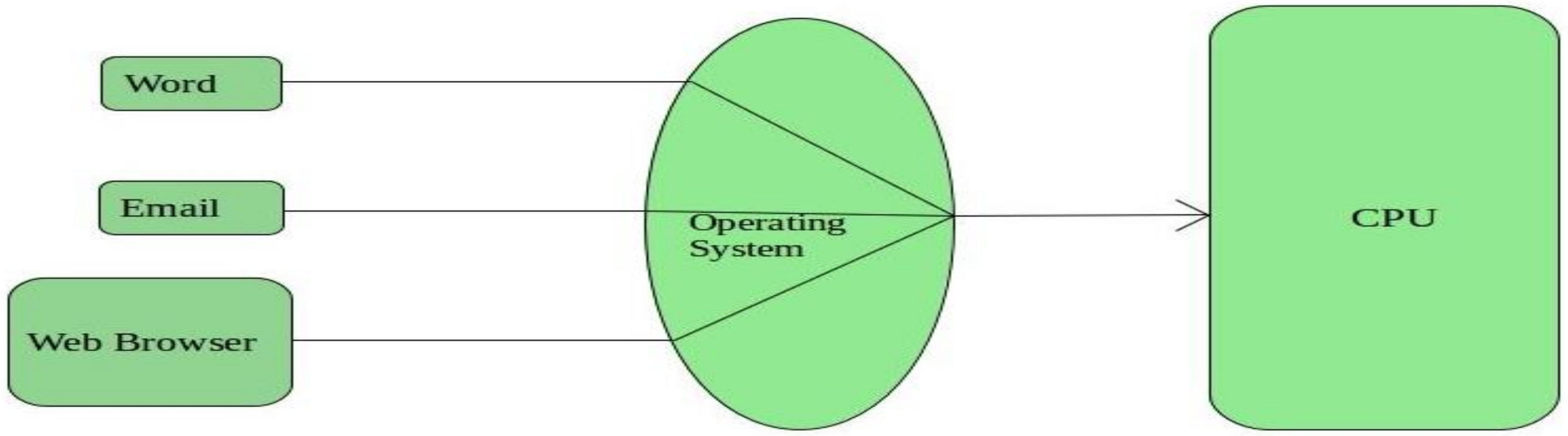
2. Multiprogramming Operating System:

- This type of OS is used to execute more than one jobs simultaneously by a single processor.
- It increases CPU utilization by organizing jobs so that the CPU always has one job to execute.
- Multiprogramming operating systems use the mechanism of job scheduling and CPU scheduling.



3. Time-Sharing Operating Systems

- Each task is given some time to execute so that all the tasks work smoothly.
- These systems are also known as **Multi-tasking Systems.**
- The task can be from a single user or different users also.
- The time that each task gets to execute is called quantum.
- After this time interval is over OS switches over to the next task.



3. Time-Sharing Operating Systems cont..

- **Advantages of Time-Sharing OS:**
  - ➢ Each task gets an equal opportunity
  - ➢ Fewer chances of duplication of software
  - ➢ CPU idle time can be reduced
- **Disadvantages of Time-Sharing OS:**
  - ➢ Reliability problem
  - ➢ One must have to take care of the security and integrity of user programs and data
  - ➢ Data communication problem
- **Examples of Time-Sharing Oss**

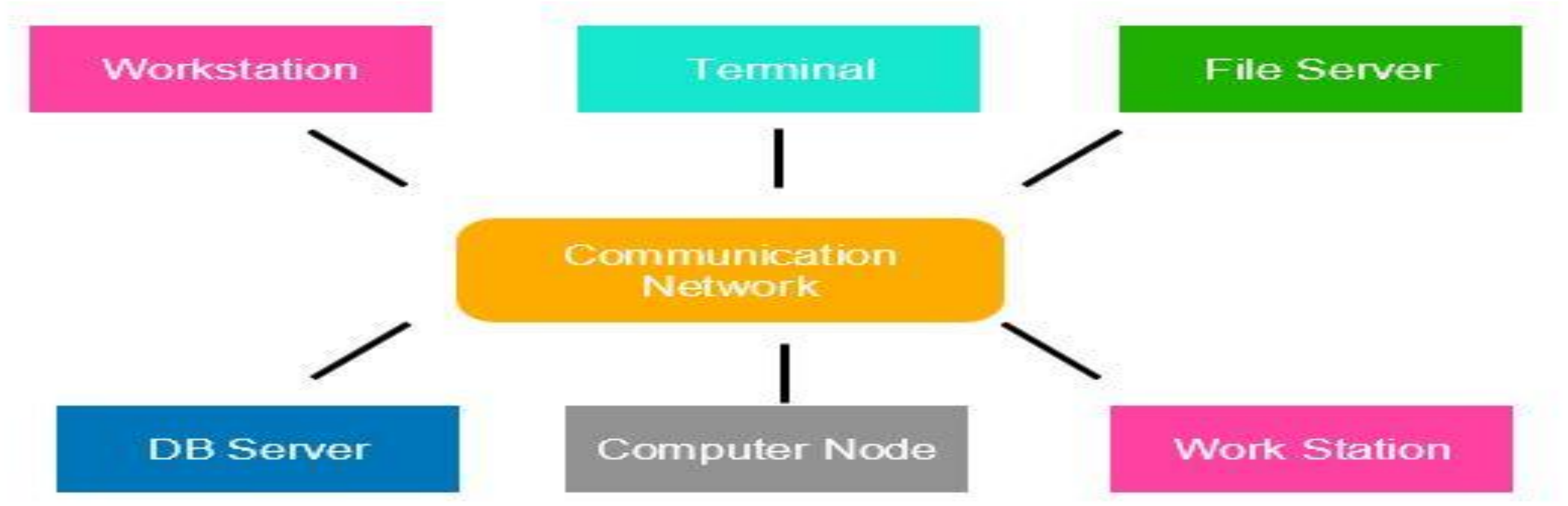
  Multics, Unix, etc.

4. Multiprocessor operating systems

- Multiprocessor operating systems are also known as parallel OS or tightly coupled OS.
- Such operating systems have more than one processor in close communication that sharing the computer bus, the clock and sometimes memory and peripheral devices.
- It executes multiple jobs at the same time and makes the processing faster.
- It supports large physical address space and larger virtual address space.
- If one processor fails then other processor should retrieve the interrupted process state so execution of process can continue.
- Inter-processes communication mechanism is provided and implemented in hardware.

5. Distributed Operating System

- Various autonomous interconnected computers communicate with each other using a shared communication network.
- Independent systems possess their own memory unit and CPU.
- These are referred to as **loosely coupled systems.**



6. Network Operating System

- These systems run on a server and provide the capability to manage data, users, groups, security, applications, and other networking functions.
- These types of operating systems allow shared access of files, printers, security, applications, and other networking functions over a small private network.
- The “ other" computers arc called client computers, and each computer that connects to a network server must be running client software designed to request a specific service.
- popularly known as **tightly coupled systems**.

6. Network Operating System

Advantages of Network Operating System:

- ➢ Highly stable centralized servers
- ➢ Security concerns are handled through servers
- ➢ New technologies and hardware up-gradation are easily integrated into the system
- ➢ Server access is possible remotely from different locations and types of systems

Disadvantages of Network Operating System:

- ➢ Servers are costly
- ➢ User has to depend on a central location for most operations
- ➢ Maintenance and updates are required regularly

Examples of Network Operating System are:

Microsoft Windows Server 2003/2008/2012, UNIX, Linux, Mac OS X, Novell NetWare, and BSD, etc.

7. Real-Time Operating System

- These types of OSs serve real-time systems.
- The time interval required to process and respond to inputs is very small.
- This time interval is called **response time**.
- **Real-time systems** are used when there are time requirements that are very strict like
  - missile systems,
  - air traffic control systems,
  - robots, etc.

Popular types of OS

- Desktop Class
  - ❖ Windows
  - ❖ OS X
  - ❖ Unix/Linux
  - ❖ Chrome OS
- Server Class
  - ❖ Windows Server
  - ❖ Mac OS X Server
  - ❖ Unix/Linux
- Mobile Class
  - ❖ Android
  - ❖ iOS
  - ❖ Windows Phone

Operating System a Resource Manager

- Now-a-days all modern computers consist of processors, memories, timers, network interfaces, printers, and so many other devices.
- The operating system provides for an orderly and controlled allocation of the processors, memories, and I/O devices among the various programs in the bottom-up view.
- Operating system allows multiple programs to be in memory and run at the same time.
- Resource management includes multiplexing or sharing resources in two different ways: in time and in space.
- In time multiplexed, different programs take a chance of using CPU. First one tries to use the resource, then the next one that is ready in the queue and so on. For example: Sharing the printer one after another.

Threads

A thread is a single sequence stream within a process. Threads are also called lightweight processes as they possess some of the properties of processes. Each thread belongs to exactly one process

- The primary difference is that threads within the same process run in a shared memory space, while processes run in separate memory spaces. Threads are not independent of one another like processes are, and as a result, threads share with other threads their code section, data section, and OS resources (like open files and signals). But, like a process, a thread has its own program counter (PC), register set, and stack space.

Advantages

- **Responsiveness:** If the process is divided into multiple threads, if one thread completes its execution, then its output can be immediately returned.
- **Faster context switch:** Context switch time between threads is lower compared to the process context switch. Process context switching requires more overhead from the CPU.
- **Effective utilization of multiprocessor system:** If we have multiple threads in a single process, then we can schedule multiple threads on multiple processors. This will make process execution faster.

- **Resource sharing:** Resources like code, data, and files can be shared among all threads within a process. Note: Stacks and registers can’t be shared among the threads. Each thread has its own stack and registers.
- **Communication:** Communication between multiple threads is easier, as the threads share a common address space. while in the process we have to follow some specific communication techniques for communication between the two processes.

- **Enhanced throughput of the system:** If a process is divided into multiple threads, and each thread function is considered as one job, then the number of jobs completed per unit of time is increased, thus increasing the throughput of the system.

Thread States

When a thread moves through the system, it is always in one of the five states:

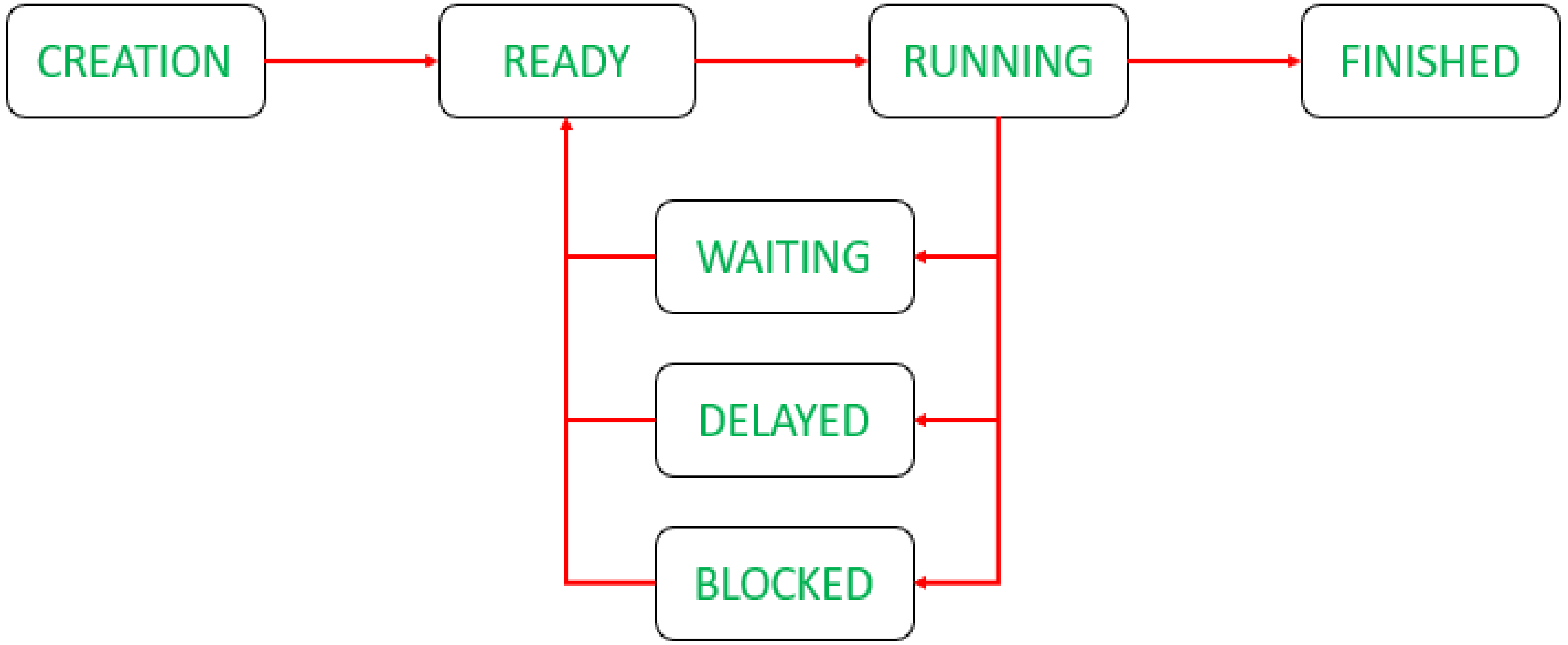
(1) Ready

(2) Running

(3) Waiting

(4) Delayed

(5) Blocked


CREATION
READY
RUNNING
FINISHED
WAITING
DELAYED
BLOCKED

- When an application is to be processed, then it creates a thread.
- It is then allocated the required resources(such as a network) and it comes in the **READY** queue.
- When the thread scheduler (like a process scheduler) assign the thread with processor, it comes in **RUNNING** queue.

- When the process needs some other event to be triggered, which is outsides it’s control (like another process to be completed), it transitions from **RUNNING** to **WAITING** queue.
- When the application has the capability to delay the processing of the thread, it when needed can delay the thread and put it to sleep for a specific amount of time. The thread then transitions from **RUNNING** to **DELAYED** queue.
- An example of delaying of thread is snoozing of an alarm. After it rings for the first time and is not switched off by the user, it rings again after a specific amount of time. During that time, the thread is put to sleep.

- When thread generates an I/O request and cannot move further till it’s done, it transitions from **RUNNING** to **BLOCKED** queue.
- After the process is completed, the thread transitions from **RUNNING** to **FINISHED**.

- The difference between the **WAITING** and **BLOCKED** transition is that in WAITING the thread waits for the signal from another thread or waits for another process to be completed, meaning the burst time is specific. While, in BLOCKED state, there is no specified time (it depends on the user when to give an input).

- There are four basic thread management operations:
- Thread creation: The creating thread is the ***parent thread***, and the created thread is a ***child thread***.
- Thread termination: threads are not created and run forever. After finishing their work, threads terminate. **Moreover, if the parent thread terminates, all of its child threads terminate as well.** Thus, if the parent thread runs faster and terminates earlier than its child threads do, we have a problem!

- Thread join:**In general, thread join is for a parent to join with one of its child threads.** Thread join has the following activities, assuming that a parent thread ***P*** wants to join with one of its child threads ***C***.
- When ***P*** executes a thread join in order to join with ***C***, which is still running, ***P*** is suspended until ***C*** terminates. Once ***C*** terminates, ***P*** resumes.
- When ***P*** executes a thread join and ***C*** has already terminated, ***P*** continues as if no such thread join has ever executed (*i.e.*, join has no effect).

- A parent thread may join with many child threads created by the parent. Or, a parent only join with some of its child threads, and ignore other child threads.
- In this case, those child threads that are ignored by the parent will be terminated when the parent terminates.
-

- Thread yield:When a thread executes a thread yield, the executing thread is suspended and the CPU is given to some other runnable thread. This thread will wait until the CPU becomes available again. Technically, in process scheduler's terminology, the executing thread is put back into the ***ready queue*** of the processor and waits for its next turn.

Types of Threads

User Level Threads

- User Level Thread is a type of thread that is not created using system calls. The kernel has no work in the management of user-level threads. User-level threads can be easily implemented by the user. In case when user-level threads are single-handed processes, kernel-level thread manages them.

Kernel Level Threads

- A kernel Level Thread is a type of thread that can recognize the Operating system easily. Kernel Level Threads has its own thread table where it keeps track of the system. The operating System Kernel helps in managing threads. Kernel Threads have somehow longer context switching time. Kernel helps in the management of threads.

Threading Models

There are four basic thread models :

1. User Level Single Thread Model :

- Each process contains a single thread.
- Single process is itself a single thread.
- process table contains an entry for every process by maintaining its PCB.

2. User Level Multi Thread Model :

- Each process contains multiple threads.
- All threads of the process are scheduled by a thread library at user level.
- Thread switching can be done faster than process switching.
- Thread switching is independent of operating system which can be done within a process.
- Blocking one thread makes blocking of entire process.
- Thread table maintains Thread Control Block of each thread of a process.
- Thread scheduling happens within a process and not known to Kernel.

3. Kernel Level Single Thread Model :

- Each process contains a single thread.
- Thread used here is kernel level thread.
- Process table works as thread table.

4. Kernel Level Multi Thread Model :

- Thread scheduling is done at kernel level.
- Fine grain scheduling is done on a thread basis.
- If a thread blocks, another thread can be scheduled without blocking the whole process.
- Thread scheduling at Kernel process is slower compared to user level thread scheduling.
- Thread switching involves switch.

Processor Scheduling

- The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.
- Process scheduling is an essential part of a Multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU

Scheduling levels

Long Term or Job Scheduler

- It brings the new process to the ‘Ready State’. It controls the ***Degree of Multi-programming***, i.e., the number of processes present in a ready state at any point in time. It is important that the long-term scheduler make a careful selection of both I/O and CPU-bound processes. I/O-bound tasks are which use much of their time in input and output operations while CPU-bound processes are which spend their time on the CPU. The job scheduler increases efficiency by maintaining a balance between the two. They operate at a high level and are typically used in batch-processing systems.

Short-Term or CPU Scheduler

- It is responsible for selecting one process from the ready state for scheduling it on the running state. Note: Short-term scheduler only selects the process to schedule it doesn't load the process on running. Here is when all the scheduling algorithms are used. The CPU scheduler is responsible for ensuring no starvation due to high burst time processes.***The dispatcher*** is responsible for loading the process selected by the Short-term scheduler on the CPU (Ready to Running State) Context switching is done by the dispatcher only.

A dispatcher does the following:

- Switching context.
- Switching to user mode.
- Jumping to the proper location in the newly loaded program.

Medium-Term Scheduler

- It is responsible for suspending and resuming the process. It mainly does swapping (moving processes from main memory to disk and vice versa). Swapping may be necessary to improve the process mix or because a change in memory requirements has overcommitted available memory, requiring memory to be freed up. It is helpful in maintaining a perfect balance between the I/O bound and the CPU bound.

Some Other Schedulers

- **I/O schedulers:** I/O schedulers are in charge of managing the execution of I/O operations such as reading and writing to discs or networks. They can use various algorithms to determine the order in which I/O operations are executed, such as FCFS (First-Come, First-Served) or RR (Round Robin).
- **Real-time schedulers:** In real-time systems, real-time schedulers ensure that critical tasks are completed within a specified time frame. They can prioritize and schedule tasks using various algorithms such as EDF (Earliest Deadline First) or RM (Rate Monotonic).

Categories

- **Non-preemptive:** Here the resource can’t be taken from a process until the process completes execution. The switching of resources occurs when the running process terminates and moves to a waiting state.

- **Preemptive:** Here the OS allocates the resources to a process for a fixed amount of time. During resource allocation, the process switches from running state to ready state or from waiting state to ready state. This switching occurs as the CPU may give priority to other processes and replace the process with higher priority with the running process.

Scheduling Objectives

- Be Fair.
- Maximize throughput.
- Maximize number of users receiving acceptible response times.
- Be predictable.
- Balance resource use.
- Avoid indefinite postponement.
- Enforce Priorities.
- Give preference to processes holding key resources.

Scheduling criteria

Scheduling is a process of allowing one process to use the CPU resources, keeping on hold the execution of another process due to the unavailability of resources *CPU*.

- **First-Come First-Serve Scheduling, FCFS.**
- **Shortest-Job-First Scheduling, SJF.**
- **Priority Scheduling.**
- **Round Robin Scheduling.**

Real time scheduling

A real-time system comprises real tasks or applications which need to get processed without any delay. In this system, a time-bound approach is followed for fixed time constraints and the tasks have to be processed within the time constraints. This timing constraint is termed the deadline for real-time tasks. The requirements specified for real-time systems are given by timelines and predictability, where timelines denote how near/close the task to the deadline given is and predictability defines the amount of deviation that occurs in the timelines that are delivered.

- In real-time systems, the scheduler is considered as the most important component which is typically a short-term task scheduler. The main focus of this scheduler is to reduce the response time associated with each of the associated processes instead of handling the deadline.

If a preemptive scheduler is used, the real-time task needs to wait until its corresponding tasks time slice completes. In the case of a non-preemptive scheduler, even if the highest priority is allocated to the task, it needs to wait until the completion of the current task. This task can be slow (or) of the lower priority and can lead to a longer wait.

A better approach is designed by combining both preemptive and non-preemptive scheduling. This can be done by introducing time-based interrupts in priority based systems which means the currently running process is interrupted on a time-based interval and if a higher priority process is present in a ready queue, it is executed by preempting the current process.

- **Static table-driven approaches:**
  These algorithms usually perform a static analysis associated with scheduling and capture the schedules that are advantageous. This helps in providing a schedule that can point out a task with which the execution must be started at run time.

Static priority-driven preemptive approaches:
Similar to the first approach, these type of algorithms also uses static analysis of scheduling. The difference is that instead of selecting a particular schedule, it provides a useful way of assigning priorities among various tasks in preemptive scheduling.

- **Dynamic planning-based approaches:**
  Here, the feasible schedules are identified dynamically (at run time). It carries a certain fixed time interval and a process is executed if and only if satisfies the time constraint.

- **Dynamic best effort approaches:**
  These types of approaches consider deadlines instead of feasible schedules. Therefore the task is aborted if its deadline is reached. This approach is used widely is most of the real-time systems.

Assignment 1

1. What do you mean by Real time System? Discuss its both types.
2. Difference between Long term scheduler and short term scheduler.
3. What do you mean by process and thread? Differentiate both.
4. Explain Race condition with suitable example.

5. Consider the processes P1 to P5with the following CPU burst time. Find the average turnaround time and average waiting time for round robin scheduling with time quantum of 2 units.

process	CPU Burst time	Arrival Time
P1	3	1
P2	6	2
P3	4	4
P4	5	6
P5	2	8

Thank you!