UNIT 2

- ## Introduction of Process Management

Program vs Process
A process is a program in execution. For example, when we write a program in C or C++ and compile it, the compiler creates binary code. The original code and binary code are both programs. When we actually run the binary code, it becomes a process.

A single program can create many processes when run multiple times; for example, when we open a .exe or binary file multiple times, multiple instances begin (multiple processes are created).

Attributes or Characteristics of a Process

A process has following attributes.

1. Process Id: A unique identifier assigned by the operating system

2. Process State: Can be ready, running, etc.

3. CPU registers: Like the Program Counter (CPU registers must be saved and restored when a process is swapped in and out of CPU)

5. Accounts information:

6. I/O status information: For example, devices allocated to the process, open files, etc

8. CPU scheduling information: For example, Priority (Different processes may have different priorities, for example a short process may be assigned a low priority in the shortest job first scheduling)

- ## Context Switching

  The process of saving the context of one process and loading the context of another process is known as Context Switching. In simple terms, it is like loading and unloading the process from running state to ready state.

- **When does context switching happen?**

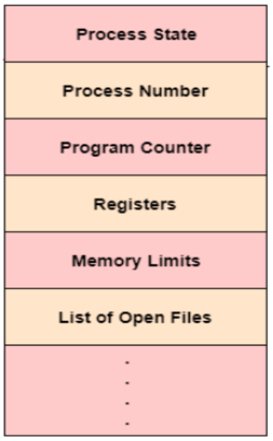
1. When a high-priority process comes to ready state (i.e. with higher priority than the running process)
2. An Interrupt occurs
3. User and kernel mode switch (It is not necessary though)
4. Preemptive CPU scheduling used.

- ## Process Control Block

Process Control Block is a data structure that contains information of the process related to it. The process control block is also known as a task control block, entry of the process table, etc.

Structure of the Process Control Block

The process control stores many data items that are needed for efficient process management. Some of these data items are explained with the help of the given diagram −



Process Control Block (PCB)

The following are the data items −

Process State

This specifies the process state i.e. new, ready, running, waiting or terminated.

Process Number

This shows the number of the particular process.

Program Counter

This contains the address of the next instruction that needs to be executed in the process.

Registers

This specifies the registers that are used by the process. They may include accumulators, index registers, stack pointers, general purpose registers etc.

List of Open Files

These are the different files that are associated with the process

CPU Scheduling Information

The process priority, pointers to scheduling queues etc. is the CPU scheduling information that is contained in the PCB. This may also include any other scheduling parameters.

Memory Management Information

The memory management information includes the page tables or the segment tables depending on the memory system used. It also contains the value of the base registers, limit registers etc.

I/O Status Information

This information includes the list of I/O devices used by the process, the list of files etc.

Accounting information

The time limits, account numbers, amount of CPU used, process numbers etc. are all a part of the PCB accounting information.

Location of the Process Control Block

The process control block is kept in a memory area that is protected from the normal user access. This is done because it contains important process information. Some of the operating systems place the PCB at the beginning of the kernel stack for the process as it is a safe location.

- **Note : For Process states ,Process state diagram, types of schedulers and cpu scheduling algorithms - Refer handwritten notes**

- **Preemptive and Non-Preemptive Scheduling**

1. **Preemptive Scheduling:**

Preemptive scheduling is used when a process switches from running state to ready state or from waiting state to ready state. The resources (mainly CPU cycles) are allocated to the process for the limited amount of time and then is taken away, and the process is again placed back in the ready queue if that process still has CPU burst time remaining. That process stays in ready queue till it gets next chance to execute.

Algorithms based on preemptive scheduling are: Round Robin (RR),Shortest Remaining Time First (SRTF), Priority (preemptive version), etc.

2. **Non-Preemptive Scheduling:**

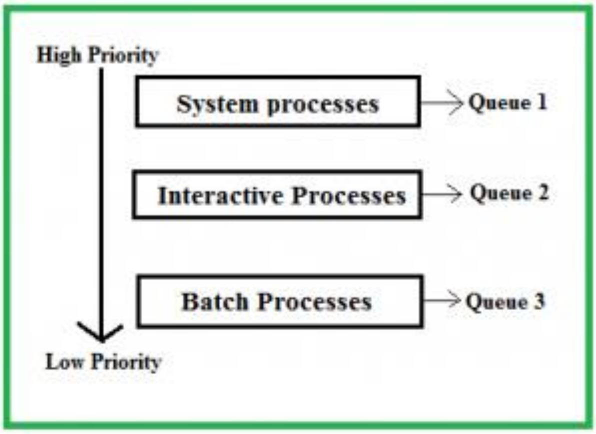
Non-preemptive Scheduling is used when a process terminates, or a process switches from running to waiting state. In this scheduling, once the resources (CPU cycles) is allocated to a process, the process holds the CPU till it gets terminated or it reaches a waiting state. In case of non-preemptive scheduling does not interrupt a process running CPU in middle of the execution. Instead, it waits till the process complete its CPU burst time and then it can allocate the CPU to another process.

Algorithms based on non-preemptive scheduling are: Shortest Job First (SJF basically non preemptive) and Priority (non preemptive version), etc.

- **Note : shortest remaining time next scheduling algorithm(means preemptive version of shortest job first) for this refer handwritten notes**

➢ Multilevel Queue (MLQ) CPU Scheduling (also refer handwritten notes)

It may happen that processes in the ready queue can be divided into different classes where each class has its own scheduling needs. For example, a common division is a **foreground (interactive)** process and **background (batch)** processes.These two classes have different scheduling needs. For this kind of situation Multilevel Queue Scheduling is used.Now, let us see how it works.
Ready Queue is divided into separate queues for each class of processes. For example, let us take three different types of process System processes, Interactive processes and Batch Processes. All three process have there own queue.



All three different type of processes have there own queue. Each queue have its own Scheduling algorithm. For example, queue 1 and queue 2 uses **Round Robin** while queue 3 can use **FCFS** to schedule there processes.

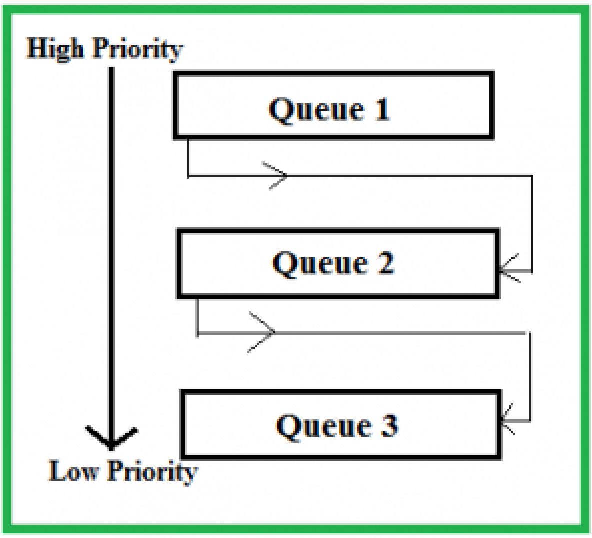
Scheduling among the queues : What will happen if all the queues have some processes? Which process should get the cpu? To determine this Scheduling among the queues is necessary. There are two ways to do so –

1. **Fixed priority preemptive scheduling method –** Each queue has absolute priority over lower priority queue. Let us consider following priority order **queue 1 > queue 2 > queue 3**.According to this algorithm no process in the batch queue(queue 3) can run unless queue 1 and 2 are empty. If any batch process (queue 3) is running and any system (queue 1) or Interactive process(queue 2) entered the ready queue the batch process is preempted.

2. **Time slicing** – In this method each queue gets certain portion of CPU time and can use it to schedule its own processes.For instance, queue 1 takes 50 percent of CPU time queue 2 takes 30 p

- **<u>Multilevel Feedback Queue Scheduling (MLFQ) CPU Scheduling Scheduling (also refer handwritten notes)</u>**

This Scheduling is like Multilevel Queue(MLQ) Scheduling but in this process can move between the queues. **Multilevel Feedback Queue Scheduling (MLFQ)** keep analyzing the behavior (time of execution) of processes and according to which it changes its priority.



Now let us suppose that queue 1 and 2 follow round robin with time quantum 4 and 8 respectively and queue 3 follow FCFS.One implementation of MFQS is given below –

1. When a process starts executing then it first enters queue 1.

2. In queue 1 process executes for 4 unit and if it completes in this 4 unit or it gives CPU for I/O operation in this 4 unit than the priority of this process does not change and if it again comes in the ready queue than it again starts its execution in Queue 1.

3. If a process in queue 1 does not complete in 4 unit then its priority gets reduced and it shifted to queue 2.

4. Above points 2 and 3 are also true for queue 2 processes but the time quantum is 8 unit.In a general case if a process does not complete in a time quantum than it is shifted to the lower priority queue.

5. In the last queue, processes are scheduled in FCFS manner.

6. A process in lower priority queue can only execute only when higher priority queues are empty.

7. A process running in the lower priority queue is interrupted by a process arriving in the higher priority queue.

Problems in the above implementation – A process in the lower priority queue can suffer from starvation due to some short processes taking all the CPU time.

Solution – A simple solution can be to boost the priority of all the process after regular intervals and place them all in the highest priority queue.

Advanatages of using Multilevel feedback queue scheduling

- Firstly, it is more flexible than the multilevel queue scheduling.
- To optimize turnaround time algorithms like SJF is needed which require the running time of processes to schedule them. But the running time of the process is not known in advance. MFQS runs a process for a time quantum and then it can change its priority(if it is a long process). Thus it learns from past behavior of the process and then predicts its future behavior.This way it tries to run shorter process first thus optimizing turnaround time.
- MFQS also reduces the response time.