



Lecture – 8



Section-B

Theoretical concept of Unix Operating System

Reference Books: (1) UNIX- Concepts & Applications by
Sumitabha Das

(2) Principles of Operating System- Galvin



Introduction

- CPU Scheduling in UNIX

CPU Scheduling:

- Deals with problem of deciding which of the processes in ready queue is to be allocated to CPU.
- CPU scheduling is the basis of multi-programmed operating systems; by switching among processes.
- The **objective** of multiprogramming is to have some processes running at all times, to maximize the CPU utilization.

Basic Idea of multiprogramming:

Several processes are kept in memory at one time. when one process has to wait; operating system takes the CPU away from that process & gives the CPU to another



CPU scheduling decisions may take place under following four circumstances:

- When a process switches from the running state to the waiting state.
- When a process switches from running state to ready state.
- When a process switches from waiting state to ready state.
- When a process terminates.

CPU Scheduling in UNIX

- CPU scheduling in Unix is designed to benefit interactive processes.
- Processes are given a small CPU time slices by a priority algorithm that reduces to round-robin scheduling for CPU bound jobs.
- The scheduler on UNIX system belongs to the general class of operating system schedulers known as **round-robin with multi-level feedback** which means that the kernel allocates the CPU time to a process for small time – slice, **pre-empts** a process that exceeds its time-slice and feed it back into one of several **priority** queues. A process may need many iterations through the “feedback loop” before it finishes.



Scheduling Algorithms:

- ✓ **Non-Pre-emptive vs. Pre-emptive Scheduling**
- ✓ **Shortest job first**
- ✓ **First come first serve**
- ✓ **Priority Scheduling**
- ✓ **Round Robin Scheduling.**
- ✓ **Multi-level feedback queue scheduling**

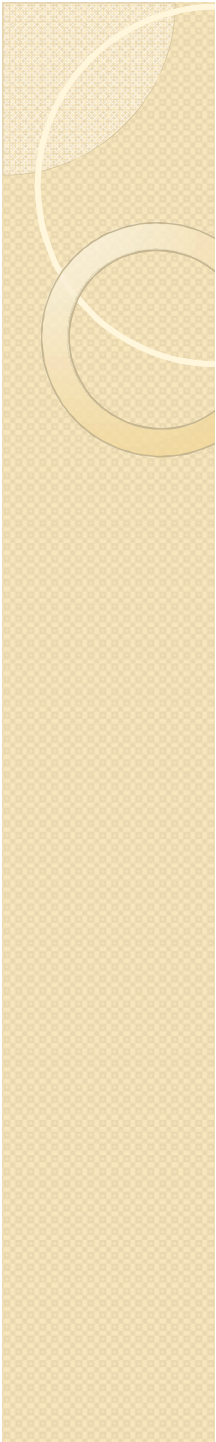
Non-Pre-emptive Vs. Pre-emptive Scheduling

- **Non-Pre-emptive:** Non-pre-emptive algorithms are designed so that once a process enters the running state, it is not removed from the processor until it has completed its service time.
- **Pre-emptive :** the process with the highest priority should always be the one currently using the processor. If a process is currently using the processor and a new process with a higher priority enters, the ready list , the process on the processor should be removed and returned to the ready list until it is once again the highest priority process in the system.

Priority scheduling:

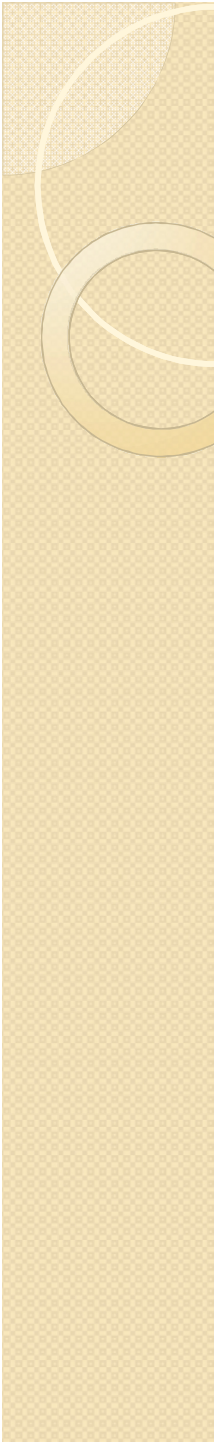
- The SJF is a special case of the general priority scheduling algorithm.
- A priority is associated with each process, and the CPU is allocated to the process with the highest priority. Equal priority processes are scheduled in FCFS order.
- We discuss scheduling in terms of *high* priority and *low* priority.
- Priorities are generally some fixed range of numbers, such as 0 to 7, or 0 to 4095. In this text, we use low numbers to represent high priority.

<u>Process</u>	<u>Burst time</u>	<u>priority</u>
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	-	-



Round Robin Scheduling

- This is a form of scheduling where the running jobs are interrupted to give the other jobs a chance to get access to the processor.
- A small unit of time, called a time quantum (or time slice) is defined. A time quantum is generally from 10 to 100 milliseconds.
- The CPU Scheduler goes around the ready queue, allocating the CPU to each process for a time interval up to 1 time quantum.
- To implement RR scheduling, we keep the ready queue as a FIFO queue of processes.
- New processes are added to the tail of the ready queue.
- **The CPU scheduler picks the first process from the ready queue, sets a timer to interrupt after 1 time quantum, and dispatches the process.**

- 
- **One of the two things will then happen:**
 - ✓ The process may have a CPU burst of less than 1 quantum. In this case, the process itself will release the CPU voluntarily. The scheduler will then proceed to the next process in the ready queue.
 - ✓ Otherwise, if the CPU burst of the currently running process is longer than 1 time quantum. The timer will go off and will cause an interrupt to the operating system. And the process will be shifted at the tail of the ready queue.

Process

Burst Time

P1

24

P2

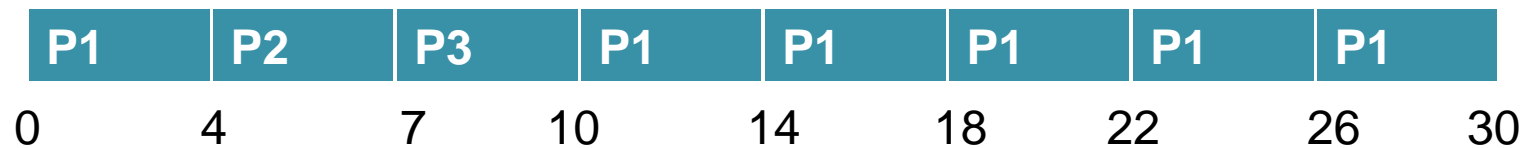
3

P3

3

If we use a time quantum of 4 milliseconds, then process P1 gets the first 4 milliseconds. Since it requires another 20 milliseconds, it is shifted after first time quantum, and the CPU is given to the next process, process P2.

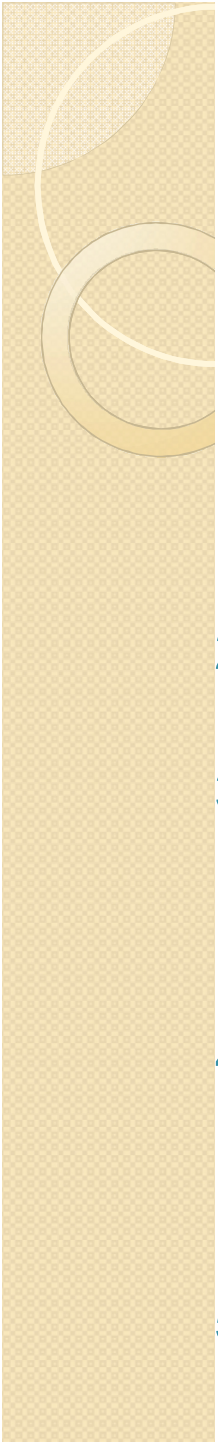
Since P2 does not need 4 milliseconds, it quits before its time quantum expires. The CPU is then given to the next process, P3. Once each process has received 1 time quantum, the CPU is returned to process P1 for an additional time quantum. The resulting RR schedule is :





Multilevel-feedback Queue scheduling

- Normally in **multi-level queue scheduling** algorithm, processes are permanently assigned to a queue on entry to the system. Processes do not move between the queues.
- **Multi-level feedback queue** scheduling however allow the processes to move between the queues. The idea is to separate processes with different CPU burst characteristics.
- If a process uses too much CPU time, it will be moved to a lower priority queue. Similarly, a process that waits too long in a lower priority queues may be moved to a

- 
- **In general, multi-level feedback queue scheduler is defined by the following parameters:**

1. The number of queues.
2. The scheduling algorithm for each queue.
3. The method used to determine when to upgrade a process in a higher priority queue.
4. The method used to determine when to demote a process to a lower priority queue.
5. The method used to determine which queue a process will enter when that process needs



Applications

- In real-time environments, such as embedded systems for automatic control in industry (for example robotics), the scheduler also must ensure that processes can meet deadlines; this is crucial for keeping the system stable. Scheduled tasks are sent to mobile devices and managed through an administrative back end.



Research

- Multimedia applications have unique requirements that must be met by network and operating system components.
- There is extensive research in developing network and operating systems to meet the demands of multimedia computation.
- Certain problem solutions are exclusive to the operating system, some are unique to network research, and some problems cover both domains.
- Research in multimedia spans not only the development of new system solutions, but evaluating existing systems as well.
- Some research papers attempt to prove empirically how useful or impractical a system may be for executing multimedia applications.
- Some research deals not with the evaluation of particular systems, but methods themselves for evaluating them.