Processor scheduling and concurrent programming in windows xp

When a system has an option of processes to execute, it must contain a policy called a processor scheduling policy (or discipline) for make a decision which process to run at a given instance. There are High level, Intermediate level and low level scheduling policies in this manner.

1. High level Scheduling Policy sometimes called job scheduling or long term scheduling and it determines which jobs the system permits to compete dynamically for system resources.

2. Intermediate level Scheduling Policy - After the high-level scheduling policy has acknowledged a job to the system, the intermediate-level scheduling policy determines which processes shall be permitted to compete for a processor.

3. Low level Scheduling Policy - Decides which ready process the system will allocate to a processor when one next becomes accessible. The more important a process, the more likely the scheduling policy is to select it to execute next. In case of Windows XP, operating system supporting the threads at the kernel level must schedule threads (not processes) for execution.

This system schedule threads using preemptive, priority-based scheduling algorithms, including support for real-time threads. The Windows XP scheduler ensures that the highest- priority thread will always run. The portion of the Windows XP kernel that handles scheduling is called the dispatcher. A thread selected to run by the dispatcher will run until it is preempted by a higher-priority thread, until it terminates, until its time

quantum ends, or until it calls a blocking system call, such as for I/O. It has defined six classes of scheduling, which are, in order of priority:

1. Real time.

2. Highest.

3. Above normal.

4. Normal.

5. Below normal.

6. Idle.

A scheduling discipline can be preemptive or nonpreemptive. When scheduling takes place under circumstance that 1. A process switches from the running state to the waiting state or 2. When a process terminates Then the scheduling scheme is nonpreemptive or cooperative. And preemptive when:

1. A process switches from the running state to the ready state. 2. A process switches from the waiting state to ready state. Scheduling Algorithms: 1. First-Come, First-Served Scheduling (FCFS): the implementation of the FCFS policy is easily managed by FIFO queue. With this scheme, the process that requests the CPU first is allocated the CPU first.

2. Shortest-Job-First Scheduling (SJF): In this algorithm when the CPU is available, it is assigned to the process that has the smallest next CPU burst.

3. Priority Scheduling: a priority is associated with each process, and the CPU is allocated to the process with the highest priority. The larger the CPU burst, the lower the priority, and vice versa.

4. Round- Robin Scheduling: In this, processes are dispatched FIFO but are given a limited amount of processor time called a time slice or a quantum.

5. Multilevel Queue Scheduling: It partitions the ready queue into several separate queues. The processes are permanently assigned to one queue, generally based on some property of the process, such as memory size, process priority, or process type.

6. Multilevel Feedback-queue Scheduling: In this scheduling algorithm processes are permanently assigned to a queue when they enter the system. Processes do not move from one queue to the other, since processes do not change their foreground or background nature.

When increasing a scheduling discipline, a system designer must consider a variety of objectives, such as the type of system and the users' needs. These objectives may include maximizing throughput, maximizing the number of interactive users getting " acceptable" response times, maximizing resource consumption, avoiding indefinite delay, imposing priorities, reducing overhead and ensuring predictability of response times. To achieve these goals, a system can use techniques such as aging processes and favoring processes whose requests can be satisfied rapidly. Many scheduling disciplines reveal fairness, predictability and scalability.