PROBLEM - TOPIC : Enhanced Priority Scheduling Algorithm

METHOD-1

PROBLEM STATEMENT:

One of the most important problems in operating systems designing is CPU Scheduling and challenge in this field is to build a program to achieve proper scheduling. In case of priority scheduling algorithm when similar priority jobs arrive than FCFS is used and the average waiting and turnaround time relatively higher. The process that arrives first is executed first, no matter how long it takes the CPU. So, in this case if long burst time processes execute earlier than other process will remain in waiting queue for a long time. This type of arrangement of processes in the ready queue results in the higher average waiting and turnaround time.

METHOD USED TO SOLVE:

To solve the problem of CPU Scheduling, SJF based Priority Scheduling Algorithm is used. In which each process that have similar priority is executed on the basis of burst time, i.e. the process which have least burst time will execute first. The SJF based priority algorithm results in reduced average waiting time and turnaround time.

Step 1: Assign the process to ready queue.

Step 2: Assign the process to the CPU according to the priority, higher priority process will get the CPU first than lower priority process.

Step 3: If two processes have similar priority then SJF is used to break the tie.

Step 4: Repeat the step 1 to 3 until ready queue is empty.

Step 5: Calculate Waiting time and Turnaround time of individual Process.

Step 6: Calculate Average waiting time and Average Turnaround time.

Pros:

- Average waiting timeis reduced.
- Average turnaround time is reduced
- Simple to implement.
- Minimize average response time.

Cons:

- Difficult to predict the Burst time.
- Possible Starvation (It may possible that many shorts jobs can keep long jobs from making any progress).

METHOD -2

PROBLEM STATEMENT:

In designing a scheduling algorithm, such as type of systems used and user's needs. So, depending on the system, the user might expect the scheduler

- To maximize throughput
- To avoid indefinite blocking or starvation
- To minimize overhead
- Enforcement of priorities
- Achieve balance between response and utilization

METHOD USED TO SOLVE:

To minimize the number of process parameters such as context switches, average waiting time and average turnaround time. The steps that used are as follows:

Step 1: Allocate CPU to every process in round robin fashion, according to the given priority, for given time quantum only for one time.

Step 2: After completion of first step following steps ae performed:

a) Processors are arranged in increasing order or their remaining CPU burst time in the ready queue. New priorities are assigned according to the remaining CPU bursts of processes; the process with shortest remaining CPU burst is assigned with highest priority.
b) The processors are executed according to the new priorities based on the remaining CPU burst, and each process gets the control of the CPU until they finished their execution.

Pros:

- Less Average waiting time
- Less Turnaround times
- Less number of context switches
- Dispatch latency
- Reduce the problem of Starvation

Cons:

- Cannot improve the performance of time sharing systems.
- Not improving the performance of Real time system.

METHOD-3

PROBLEM STATEMENT:

Minimizing the Response time thereby reducing the waiting time for lowest priority processes for existing Priority Scheduling algorithm.

METHOD USED TO SOLVE:

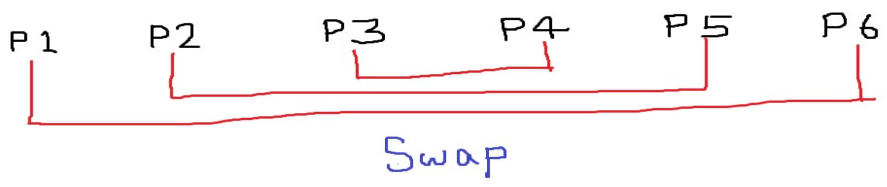
The proposed algorithm in this paper will be executed in the following steps which help to minimize performance parameters such as response time, waiting time for lower priority processes and ensures that each process is given a fair chance to access resources.

Step1*:* The tasks are sorted according to their priorities. Lower numbers are used to represent higher priorities. The process with the highest priority is allocated first and those with the same priorities are scheduled by FCFS policy.

Step 2*:* Allocate all sorted process to CPU for a predetermined time slice. Value of time slice= (Burst time of shortest process + Burst time of largest process) / 2 .

Step3*:* After all the processes have been executed once using the given time slice, swap the process priorities symmetrically with other process as shown below:

Suppose there are 6 processes say P1 ,P2, P3 , P4, P5, P6 where priority of P1 is 1; P2 is 2; P3 is 3 ; P4 is 4; P5 is 5; P6 is 6, so swapping takes place symmetrically as follows:



Pros:

- Response time for individual process is greatly reduced .
- Lowest priority processes is prevented from starvation
- Timely resource allocation is ensured to all processes and hence the problem of indefinite postponement is tackled.
- Fair treatment of all process is ensured due to the introduction of the concept of time slice along with priority .
- Waiting time for the lowest priority process is minimized .

Cons:

- Requires more number of context switches than actual priority scheduling algorithm.
- Difficult to predict the Burst time.

REFERENCES:

1 . Analysis of Priority Scheduling Algorithm on the Basis of FCFS & SJF for Similar Priority Jobs

--By- *Ms. Rukhsar Khan, Mr. Gaurav Kakhani*

2. A Priority based Round Robin CPU Scheduling Algorithm for Real Time Systems

--By *Ishwari Singh Rajput, Deepa Gupta*

3. Enhanced Priority Scheduling Algorithm to minimise Process Starvation

--By -*Siddharth Tyagi , Sudheer Choudhary , Akshant Poonia*