All about Cdac Hyderabad - DESD

Friday, 21 October 2016

OS Last Minute Notes - Part 1

Last Minute Notes Part 1

Operating Systems: It is the interface between the user and the computer hardware.

Difference Between Microkernal and Monolythic Kernal

- **Monolithic kernel:**
  - It is a single static binary file.
  - All kernel services exist and execute in the kernel address space.
  - The kernel can invoke functions directly
  - Ex: **Unix, Linux - Monolithic kernel based OSs**
- **Micro kernel:**
  - It is broken down into separate processes, known as servers.
  - Some of the servers run in kernel space and some run in user-space.
  - All servers are kept separate and run in different address spaces and Servers invoke "services" from each other by sending messages via IPC (Interprocess Communication).
  - Ex: **Mac OS X and Windows NT - Micro kernel based OSs**

Type's of OS:

- **Batch OS:** A set of similar jobs are stored in the main memory for execution. A job gets assigned to the CPU, only when theexecution of the previous job completes.
- **Multiprogramming OS:** The main memory consists of jobs waiting for CPU time. The OS selects one of the processes and assigns it the CPU time. Whenever the executing process needs to wait for any other operation (like I/O), the OS selects another process from the job queue and assigns it the CPU. This way, the CPU is never kept idle and the user gets the flavor of getting multiple tasks done atonce.
- **Multitasking OS:** Multitasking OS combines the benefits of Multiprogramming OS and CPU scheduling to perform quick switches between jobs. The switch is so quick that the user can interact with each program as it runs
- **Time Sharing OS:** Time sharing systems require interaction with the user to instruct the OS to perform various tasks. The OS responds with an output. The instructions are usually given through an input device like the keyboard.
- **Real Time OS :** Real Time OS are usually built for dedicated systems to accomplish a specific set of tasks within deadlines.

Threads

A thread is a light weight process and forms a basic unit of CPU utilization. A process can perform more
than one task at the same time by including multiple threads.

- A thread has its own program counter, register set, and stack
- A thread shares with other threads of the same process the code section, the data section, files and signals.

A new thread, or a child process of a given process, can be introduced by using the fork() system call. A process with n fork()system calls generates $2^n - 1$ child processes.
There are two types of threads:

- User threads
- Kernel threads

Example : Java thread, POSIX threads.Example : Window Solaris.

User level thread	Kernel level thread
User thread are implemented by users.	kernel threads are implemented by OS.
OS doesn't recognized user level threads.	Kernel threads are recognized by OS.

Implementation of User threads is easy.	Implementation of Kernel thread is complicated.
Context switch time is less.	Context switch time is more.
Context switch requires no hardware support.	Hardware support is needed.
If one user level thread perform blocking operation then entire process will be blocked.	If one kernel thread perform blocking operation then another thread can continue execution.

Process:
A process is a program under execution. The value of program counter (PC) indicates the address of the current instruction of the process being executed. Each process is represented by a Process Control Block (PCB).

Process Scheduling:
Below are different time with respect to a process.

```
Arrival Time:       Time at which the process arrives in the ready queue.
Completion Time:    Time at which process completes its execution.
Burst Time:         Time required by a process for CPU execution.
Turn Around Time:   Time Difference between completion time and arrival time.
     Turn Around Time = Completion Time - Arrival Time

Waiting Time(W.T): Time Difference between turn around time and burst time.
     Waiting Time = Turn Around Time - Burst Time
```

Why do we need scheduling?
A typical process involves both I/O time and CPU time. In a uniprogramming system like MS-DOS, time spent waiting for I/O is wastedand CPU is free during this time. In multiprogramming systems, one process can use CPU while another is waiting for I/O. This is possible only with process scheduling.

Objectives of Process Scheduling Algorithm

```
Max CPU utilization [Keep CPU as busy as possible]
Fair allocation of CPU.
Max throughput [Number of processes that complete their execution per time unit]
Min turnaround time [Time taken by a process to finish execution]
Min waiting time [Time a process waits in ready queue]
Min response time [Time when a process produces first response]
```

Different Scheduling Algorithms

First Come First Serve (FCFS): Simplest scheduling algorithm that schedules according to arrival times of processes.
Shortest Job First(SJF): Process which have the shortest burst time are scheduled first.
Shortest Remaining Time First(SRTF): It is preemptive mode of SJF algorithm in which jobs are schedule according to shortest remaining time.
Round Robin Scheduling: Each process is assigned a fixed time in cyclic way
Priority Based scheduling (Non Preemptive): In this scheduling, processes are scheduled according to their priorities, i.e., highest priority process is schedule first. If priorities of two processes match, then schedule according to arrivaltime.
Highest Response Ratio Next (HRRN) In this scheduling, processes with highest response ratio is scheduled. This algorithm avoids starvation.

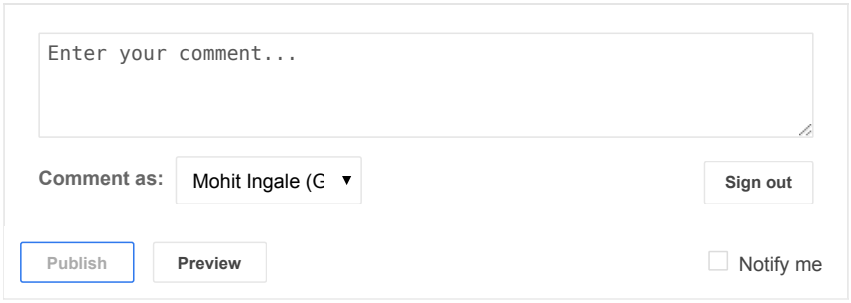
```
Response Ratio = (Waiting Time + Burst time) / Burst time
```

Multilevel Queue Scheduling: According to the priority of process, processes are placed in the different queues. Generally high priority process are placed in the top level queue. Only after completion of processes from top level queue, lower level queuedprocesses are scheduled.
Multi level Feedback Queue Scheduling: It allows the process to move in between queues. The idea is to separate processes according to the characteristics of their CPU bursts. If a process uses too much CPU time, it is moved to a lower-priority queue.

Some useful facts about Scheduling Algorithms:
1) FCFS can cause long waiting times, especially when the first job takes too much CPU time.
2) Both SJF and Shortest Remaining time first algorithms may cause starvation. Consider a situation when long processis there in ready queue and shorter processes keep coming.
3) If time quantum for Round Robin scheduling is very large, then it behaves same as FCFS scheduling.
4) SJF is optimal in terms of average waiting time for a given set of processes. SJF gives minimum average waiting time, but problems with SJF is how to know/predict time of next job.

No comments:

Post a Comment



Newer Post Home Older Post

Subscribe to: Post Comments (Atom)