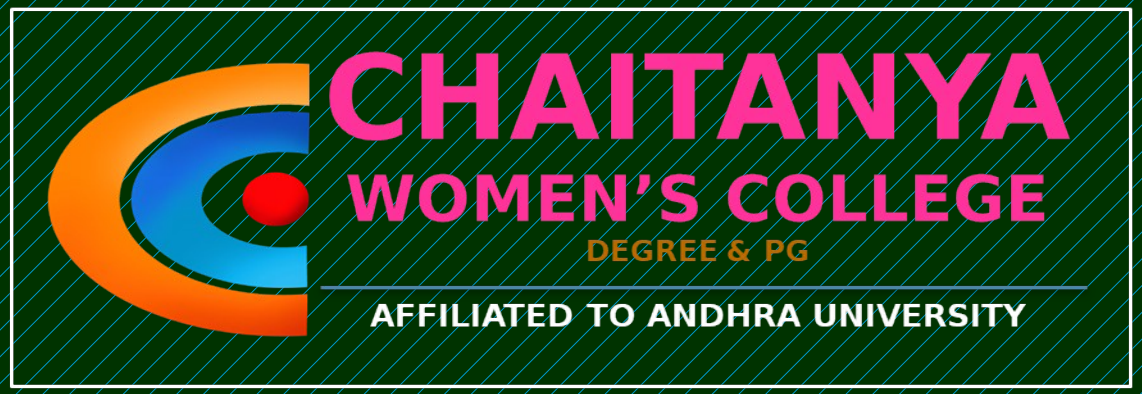

REAL TIME OPERATING SYSTEMS

REAL TIME OPERATING SYSTEMS

- A system is said to be **Real Time** if it is required to complete it's work & deliver it's services on time.
- Example – Flight Control System
  - All tasks in that system must execute on time.
- Non Example – PC system

HARD AND SOFT REAL TIME SYSTEMS

Hard Real Time System

Validation by provably correct procedures or extensive simulation that the system always meets the timings constraints

- Failure to meet deadlines is fatal
- example : Flight Control System

SOFT REAL TIME SYSTEM

- Demonstration of jobs meeting some statistical constraints suffices.
- Late completion of jobs is undesirable but not fatal.
- System performance degrades as more & more jobs miss deadlines
- Online Databases

ROLE OF AN OS IN REAL TIME SYSTEMS

Standalone Applications

- Often no OS involved
- Micro controller based Embedded Systems

Some Real Time Applications are huge & complex

- Multiple threads
- Complicated Synchronization Requirements
- Filesystem / Network / Windowing support
- OS primitives reduce the software design time

FEATURES OF RTOS'S

- **Scheduling.**
- **Resource Allocation.**
- **Interrupt Handling.**
- **Other issues like kernel size.**

SCHEDULING IN RTOS

More information about the tasks are known

- No of tasks
- Resource Requirements
- Release Time
- Execution time
- Deadlines

Being a more deterministic system better scheduling algorithms can be devised.

SCHEDULING ALGORITHMS IN RTOS

- Clock Driven Scheduling
- Weighted Round Robin Scheduling
- Priority Scheduling
  (Greedy / List / Event Driven)

CLOCK DRIVEN

- All parameters about jobs (release time/ execution time/deadline) known in advance.
- Schedule can be computed offline or at some regular time instances.
- Minimal runtime overhead.
- Not suitable for many applications.

WEIGHTED ROUND ROBIN

- Jobs scheduled in FIFO manner
- Time quantum given to jobs is proportional to it’s weight
- Example use : High speed switching network QOS guarantee.
- Not suitable for precedence constrained jobs.
  Job A can run only after Job B. No point in giving time quantum to Job B before Job A.

PRIORITY SCHEDULING

(Greedy/List/Event Driven)

- Processor never left idle when there are ready tasks
- Processor allocated to processes according to priorities
- Priorities
  - static - at design time
  - Dynamic - at runtime

PRIORITY SCHEDULING

Earliest Deadline First (EDF)

- Process with earliest deadline given highest priority

Least Slack Time First (LSF)

- slack = relative deadline – execution left

Rate Monotonic Scheduling (RMS)

- For periodic tasks
- Tasks priority inversely proportional to it’s period

RESOURCE ALLOCATION IN RTOS

Resource Allocation

- The issues with scheduling applicable here.
- Resources can be allocated in
  - Weighted Round Robin
  - Priority Based

Some resources are non preemptible

- Example : semaphores

Priority Inversion if priority scheduling is used

INTERRUPT HANDLING IN LINUX

Interrupts are disabled in ISR/critical sections of the kernel

No worst case bound on interrupt latency avaliable

- eg: Disk Drivers may disable interrupt for few hundred milliseconds

Not suitable for Real Time Applications

- Interrupts may be missed

TWO LEVEL INTERRUPT HANDLING

Two Level Interrupt Handling

- Top Half Interrupt Handler
  - Called Immediately – Kernel never disables interrupts
  - Cannot invoke thread library functions - Race Conditions
- Bottom Half Interrupt Handler
  - Invoked when kernel not in Critical Section
  - Can invoke thread library functions

Very Low Response time (as compared to Linux)

OTHER FEATURES

Footprint

- Small footprint (~50kb)

Oskit’s Device Driver Framework

- Allows direct porting of existing drivers from Linux.
- Example – Ethernet Driver of Linux