Chapter 19 - Realtime Systems

1. Definitions of Real-time System

Realtime Systems is Computer System that requires results produced within specified deadline.

There are 3 types of realtime systems:

- **Safety-Critical Systems**: if miss deadline –> CATASTROPHIC. E.g. Weapon, ABS, Flight Control, etc.
- **Hard Real-time Systems**: Guaranteed critical real-time (must completed within deadline)
- **Soft Real-time systems**: Critical real-time tasks are scheduled (but not forced).

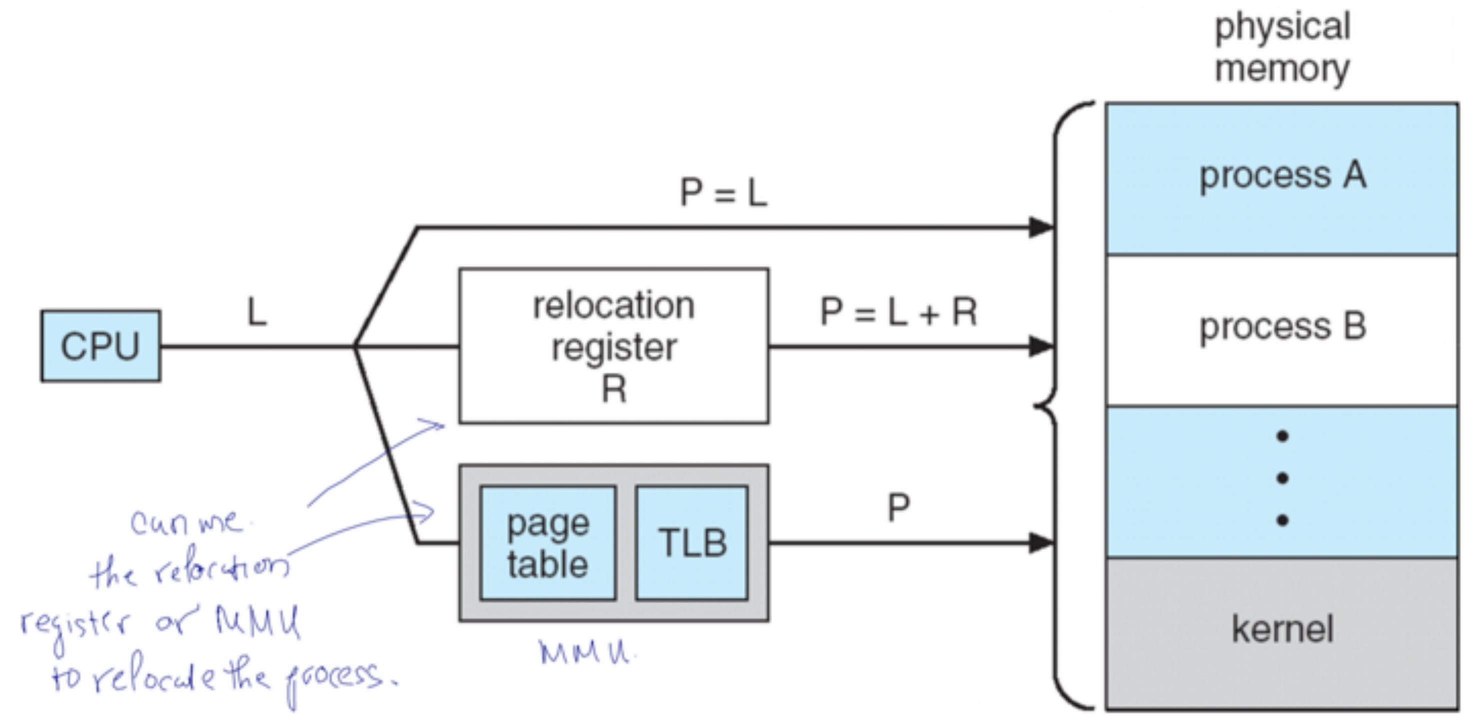
2. Characteristics of Real-time System

Single purpose, **small size**, **mass-produced**, **specific timing requirements**, and does not always provide all features such as standard desktop system.

3. Features of Real-time System

3.1. Address Translation

It uses MMU for virtual memory (sometimes disabled, or use as Address Translation), using Real Addressing Mode (i.e. Logical = Physical address) and Relocation Register to relocate the process.



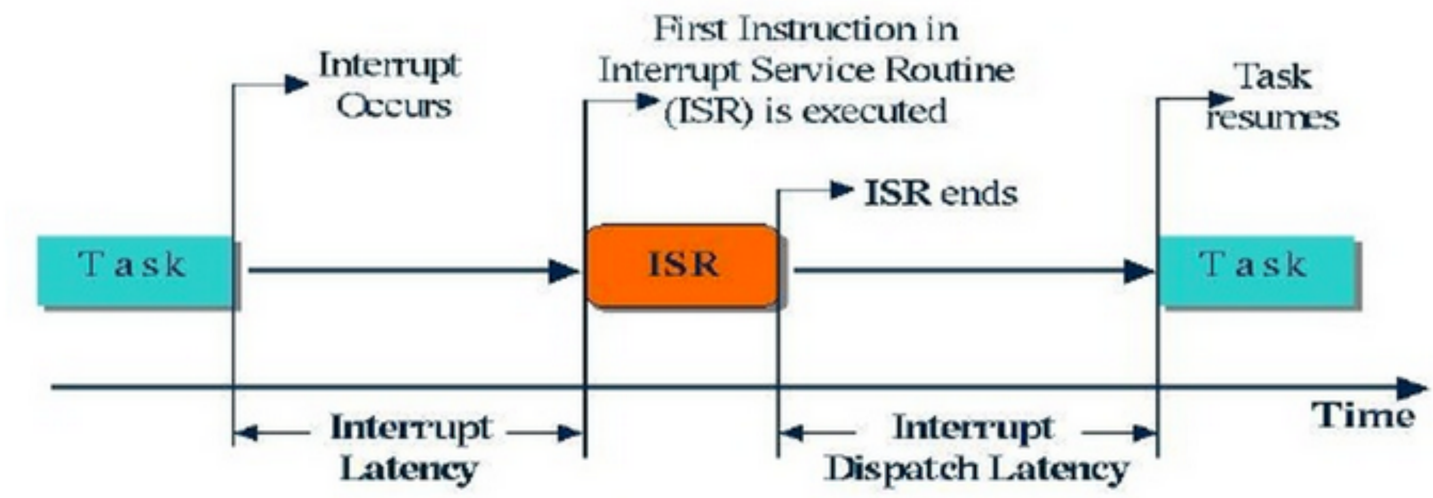
3.2. Preemptive, Priority Based Scheduling

- Is a MUST for Real-Time Systems to be preemptive and real-time process should be assigned highest scheduling priority.
- Preemptive **Soft** Real-Time Systems: assign highest scheduling priority, for e.g. Solaris, Windows, Linux.
- Preemptive **Hard** Real-Time Systems: guaranteed service within deadline requirements.

3.3. Interrupt Latency & Dispatch Latency

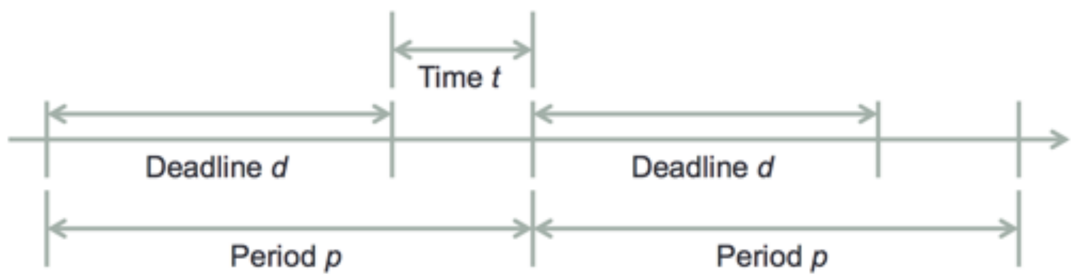
- **Interrupt Latency**: time from arrival of interrupt to start of routine that handles interrupt:
  - Save state of current process, determine interrupt type, context switching, then hand over to ISR to handle the interrupt.
  - Increased when kernel disables interrupt handler.
- **Disptach Latency**: time for scheduler to take current process off CPU and switch to another.
  - Preemptive Kernel keeps dispatch latency low. During the conflict phase (i.e. preemption), preempt any process running in Kernel, and release resource of low-priority processes. Some times Priority

Inversion is used to resolve Dispatch Latency issue.



3.4. CPU Scheduling

- Each process requires a block timing with constant interval called **period p**;
- However, in each period, it only uses a t time *(t < p)* to execute the job, **t is execution time or burst time**;
- So the difference of `d = p - t` is defined as the **deadline**, i.e. if the process is not given the CPU before or by this d time it will not be able to finish its task on time.

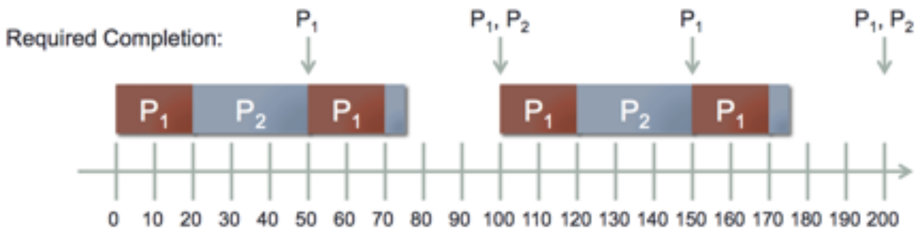


There are 3 scheduling algorithms.

3.4.1. Rate-Monotonic Scheduling Algorithm

- Task's priority is inversely assigned with their period: The **shorter** the period, the **higher** the priority, and vice versa.
- Higher priority preempts the lower one.
- Process must execute on specific period p, and complete burst t during each period.
- Rate-Monotonic Scheduling can only shedule **n** processes with no more CPU Utilization than $n(2^{1/n} -1)$, in **general is 0.69**.

- Example: two real-time processes
  - P_1 has a period of 50 clocks, CPU burst of 20 clocks
  - P_2 has a period of 100 clocks, CPU burst of 35 clocks
- P_1 and P_2 can begin executing at the same time…
  - P_1 has the higher priority, so it takes the CPU first
  - P_1 completes its processing, and then P_2 starts…
- Part way through P_2's CPU burst, P_1 must execute again
  - Preempts P_2, and completes
  - P_2 regains the CPU and completes its processing



3.4.2. Earliest Deadline First (EDF)

- Priority sticks to the deadline, i.e. the earlier the deadline the higher priority.
- EDF is theoretically optimal.

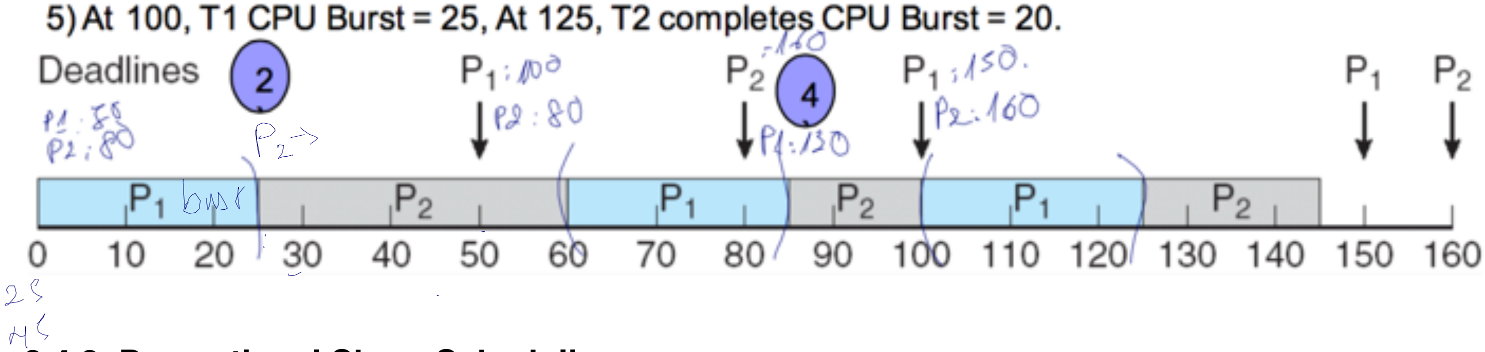
☐ **Priorities adjusted** to reflect Deadline of schedulable process.

P1 = 50 P2 = 80

T1 = 25 T2 = 35

75

1) At 0, P1 is the earliest deadline, CPU Burst T1 = 25.
2) At 25, T2 CPU Burst = 35. P2 (Deadline = 80) has higher priority than P1 (Deadline = 100)
3) At 60, T1 CPU Burst = 25.
4) At 85, T2 CPU Burst = 15, preempted by P1 (Deadline=150) over P2 (Deadline=160)
5) At 100, T1 CPU Burst = 25, At 125, T2 completes CPU Burst = 20.



3.4.3. Proportional Share Scheduling

- CPU shares are divided proportionaly to the the deadline, if deadline is short, more shares.

3.4.4. Pthread API

- Pthread API is used to manage real-time threads.

- SCHED_FIFO - for FCFS with a FIO queue, no time slicing.
- SCHEDRR - SCHEDFIFO with time-slicing for equal share of CPU time with same priority threads.
- SCHED_OTHER: not sure.

3.4.5. VxWorks

- VxWorks 5 does not distinguish between User mode and Kernel mode (only in v6).
- Events are handled in kernel.

4. Questions

4.1. Which systems below considered Hard Real Time scheduling?

Ans: Anti Lock Brake System.

4.2. Why Virtual Memory is not good for Hard Real Time system?

Ans: Translation Time introduces latency, which is the important factor that RealTime System always wants to reduce.

4.3. In Real-time system, which is NOT related to Interrupt Latency?

Ans: Use the scheduler to schedule the highest priority ISR (this belongs to distpatch latency)

4.4. Which statement is FALSE with Rate-Monotonic Scheduling?

Ans: The lower the rate, the lower the priority.

4.5. Which one is NOT the param for Realtime Scheduling?

Ans: SCHED_NORMAL (not mentioned in the slides).