CSE 4/521 Introduction to Operating Systems

Lecture 10 – CPU Scheduling III

(Real-time CPU Scheduling, Operating Systems Examples)

Summer 2018

Overview

Objective:

1. To examine how real-time scheduling works
2. Glimpse of CPU scheduling in real operating systems

- Real-Time CPU Scheduling
- Operating Systems Examples

Recap

- Scheduling Algorithms
  - Priority Scheduling, Round-robin scheduling, Multi-level (Feedback) queues
- Thread Scheduling
  - PCS and SCS, Pthreads
- Real-time CPU Scheduling
  - Interrupt latency, Dispatch latency = (conflicts + dispatch time)

Questions

1. What is soft real-time and hard real-time CPU Scheduling? (Easy)
2. What is round-robin scheduling? In round-robin, what effect does the time quantum have on average throughput time? What's the rule of thumb in deciding time quantum? (Medium)
3. Using round-robin scheduling, compute average throughput for:

process	time
P_1	6
P_2	3
P_3	1
P_4	7

q = 4

Overview

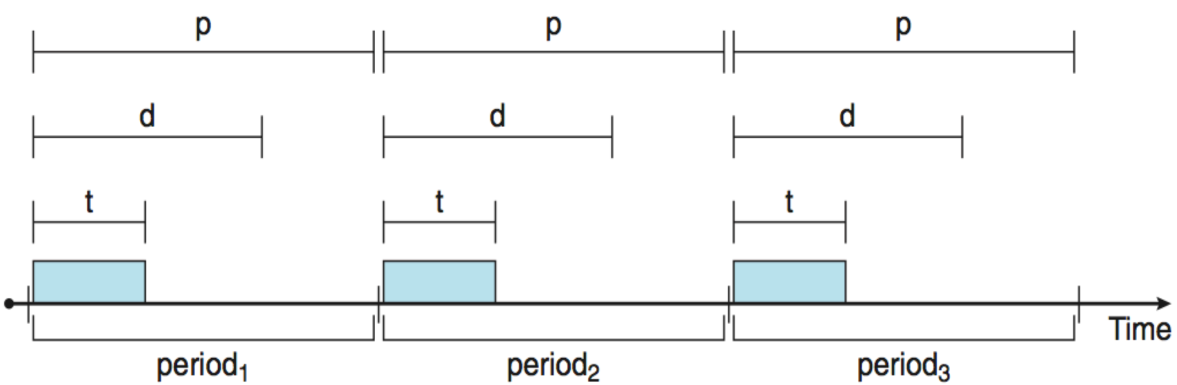
- Real-Time CPU Scheduling
- Operating Systems Examples

Overview

- Real-Time CPU Scheduling
- Operating Systems Examples

Real-Time CPU Scheduling : Priority-based Scheduling

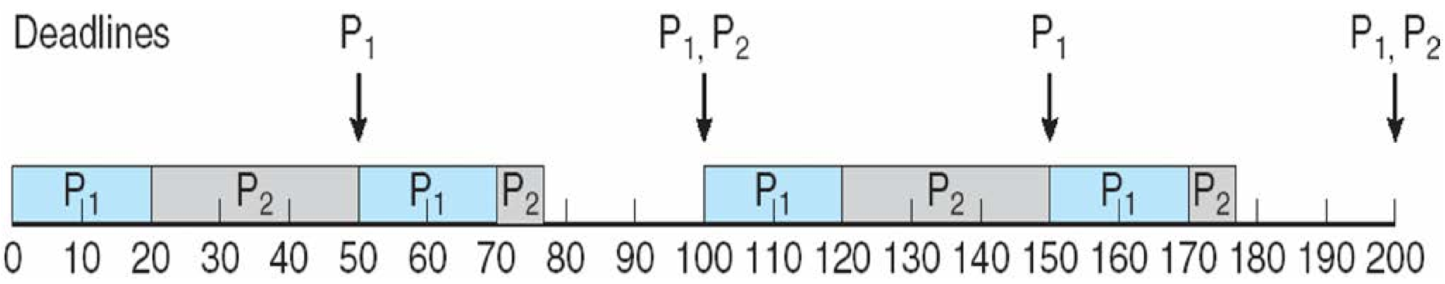
- For real-time scheduling, scheduler must support preemptive, priority-based scheduling
  - But only guarantees soft real-time
- For hard real-time must also provide ability to meet deadlines
- Processes have new characteristics: periodic ones require CPU at constant intervals
  - Has processing time *t*, deadline *d*, period *p*
  - $0 \le t \le d \le p$
  - Rate of periodic task is $1/p$



Real-Time CPU Scheduling : Rate Monotonic Scheduling

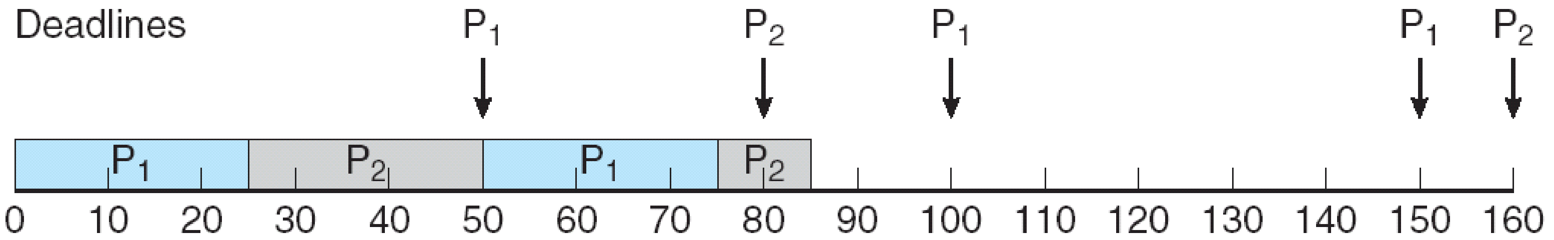
- A priority is assigned based on the inverse of its period
- Shorter periods = higher priority;
- Longer periods = lower priority
- P_1 is assigned a higher priority than P_2.

P1 = 50
P2 = 100
t1 = 20
t2 = 35
Deadline is to complete before next period



Real-Time CPU Scheduling :
Missed Deadlines with Rate Monotonic Scheduling

P1 = 50
P2 = 80
t1 = 20
t2 = 35
Deadline is to complete before next period

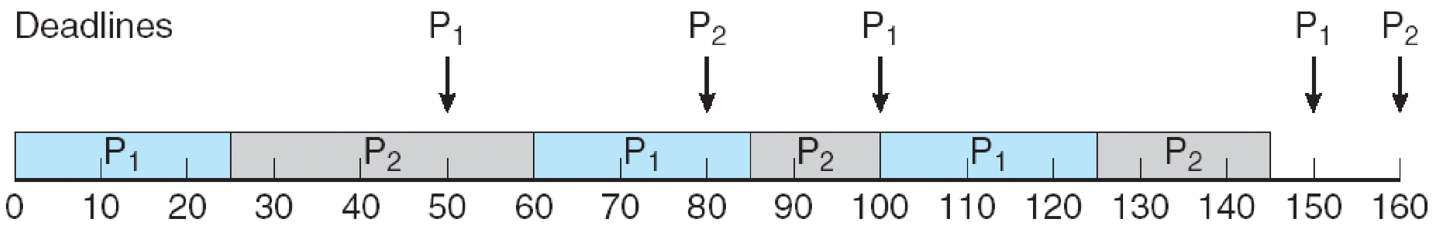


There is no pre-emption in Rate Monotonic Scheduling

Real-Time CPU Scheduling: Earliest Deadline First Scheduling (EDF)

- Priorities are assigned according to deadlines:
  - The earlier the deadline, the higher the priority
  - The later the deadline, the lower the priority

P1 = 50
P2 = 80
t1 = 25
t2 = 35
Deadline is to complete before next period



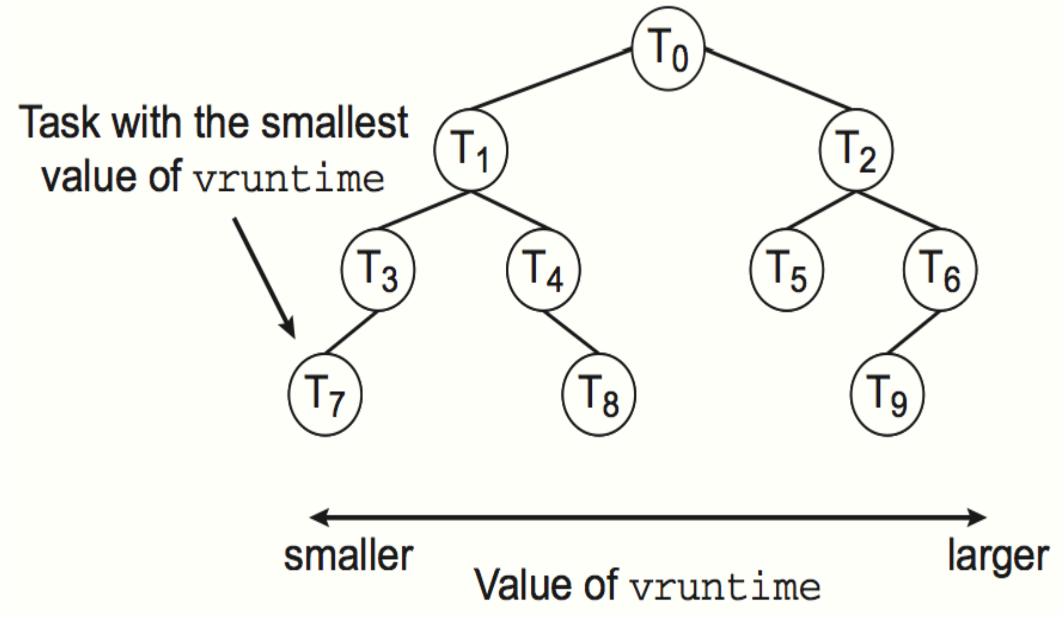
Overview

- Real-Time CPU Scheduling
- Operating Systems Examples
  - Linux Scheduling
  - Windows Scheduling
  - Solaris Scheduling

Operating Systems Examples – Linux Scheduling

- Version 2.5 moved to *O*(1) scheduling time
  - Preemptive, priority based
  - Two priority ranges: time-sharing and real-time
  - Real-time range from 0 to 99 and nice value from 100 to 140
  - Higher priority gets larger q
  - All runable tasks tracked in per-CPU runqueue data structure
    - Two priority arrays (active, expired)
    - Tasks indexed by priority
  - Worked well, but poor response times for interactive processes

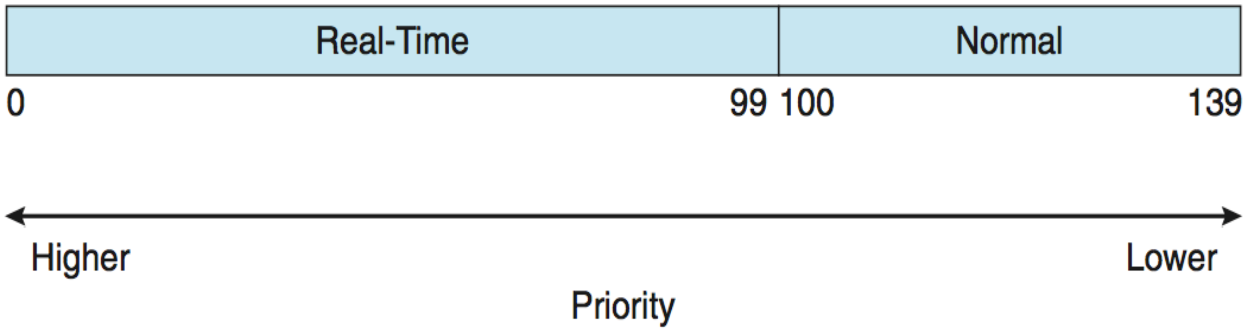
Operating Systems Examples – Linux Scheduling



The **vruntime** is the virtual runtime of a process which keeps track for how much time a process has run

Operating Systems Examples – Linux Scheduling

- Real-time tasks have static priorities
- Real-time plus normal map into global priority scheme
- Nice value of -20 maps to global priority 100
- Nice value of +19 maps to priority 139



Operating Systems Examples – Windows Scheduling

- Windows uses priority-based preemptive scheduling
- Highest-priority thread runs next
- Thread runs until (1) blocks, (2) uses time slice, (3) preempted by higher-priority thread
- Real-time threads can preempt non-real-time
- 32-level priority scheme
- Variable class is 1-15, real-time class is 16-31
- Priority 0 is memory-management thread
- Queue for each priority
- If no run-able thread, runs idle thread

Operating Systems Examples – Windows Scheduling

- Win32 API has several priority classes to which a process can belong
  - REALTIME_PRIORITY_CLASS, HIGH_PRIORITY_CLASS, ABOVE_NORMAL_PRIORITY_CLASS,NORMAL_PRIORITY_CLASS, BELOW_NORMAL_PRIORITY_CLASS, IDLE_PRIORITY_CLASS
  - All are variable except REALTIME
- A thread within a given priority class has a relative priority
  - TIME_CRITICAL, HIGHEST, ABOVE_NORMAL, NORMAL, BELOW_NORMAL, LOWEST, IDLE
- Priority class and relative priority combine to give numeric priority
- Base priority is NORMAL within the class
- If quantum expires, priority lowered, but never below base

Operating Systems Examples – Windows Scheduling

- If wait occurs, priority boosted depending on what was waited for
- Foreground window given 3x priority boost
- Windows 7 added user-mode scheduling (UMS)
  - Applications create and manage threads independent of kernel
  - For large number of threads, much more efficient
  - UMS schedulers come from programming language libraries like C++ Concurrent Runtime (ConcRT) framework

Operating Systems Examples – Windows Scheduling

	real-time	high	above normal	normal	below normal	idle priority
time-critical	31	15	15	15	15	15
highest	26	15	12	10	8	6
above normal	25	14	11	9	7	5
normal	24	13	10	8	6	4
below normal	23	12	9	7	5	3
lowest	22	11	8	6	4	2
idle	16	1	1	1	1	1

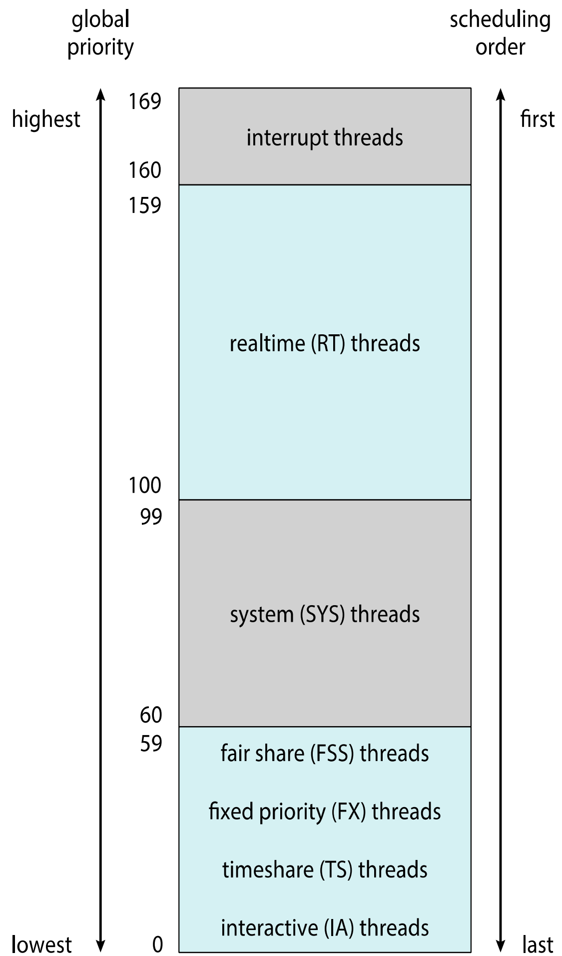
Operating Systems Examples – Solaris Scheduling

- Priority-based scheduling
- Six classes available
  - Time sharing (default) (TS)
  - Interactive (IA)
  - Real time (RT)
  - System (SYS)
  - Fair Share (FSS)
  - Fixed priority (FP)
- Given thread can be in one class at a time
- Each class has its own scheduling algorithm
- Time sharing is multi-level feedback queue

Operating Systems Examples – Solaris Scheduling

priority	time quantum	time quantum expired	return from sleep
0	200	0	50
5	200	0	50
10	160	0	51
15	160	5	51
20	120	10	52
25	120	15	52
30	80	20	53
35	80	25	54
40	40	30	55
45	40	35	56
50	40	40	58
55	40	45	58
59	20	49	59

Operating Systems Examples – Solaris Scheduling



Operating Systems Examples – Solaris Scheduling

- Scheduler converts class-specific priorities into a per-thread global priority
  - Thread with highest priority runs next
  - Runs until (1) blocks, (2) uses time slice, (3) preempted by higher-priority thread
  - Multiple threads at same priority selected via RR

Credits for slides

Silberschatz, Galvin and Gagne