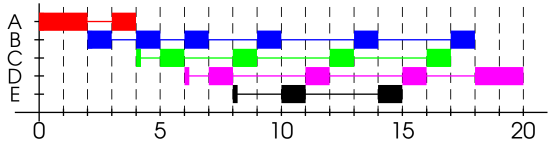
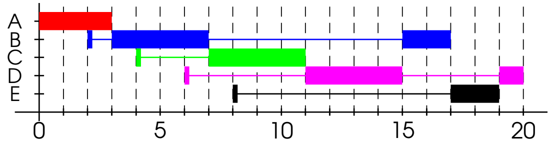
Slide 1

Round-robin

quantum=1:



quantum=4:



- ➜ Scheduled thread is given a time slice
- ➜ Running thread is preempted upon clock interrupt, running thread is returned to *ready* queue

Slide 2

Performance of round-robin scheduling:

- ➜ Average waiting time: not optimal
- ➜ Performance depends heavily on size of time-quantum:
  - too short: overhead for context switch becomes too expensive
  - too large: degenerates to FCFS policy
  - rule of thumb: about 80% of all bursts should be shorter than 1 time quantum
- ➜ no starvation

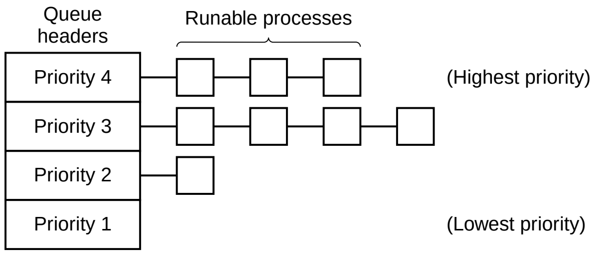
Slide 3

Priorities

- ➜ Each thread is associated with a priority
- ➜ Basic mechanism to influence scheduler decision:
  - Scheduler will always choose a thread of higher priority over one of lower priority
  - Implemented via multiple FCFS ready queues (one per priority)
- ➜ Lower-priority may suffer starvation
  - adapt priority based on thread's age or execution history
- ➜ Priorities can be defined internally or externally
  - internal: e.g., memory requirements, I/O bound vs CPU bound
  - external: e.g., importance of thread, importance of user

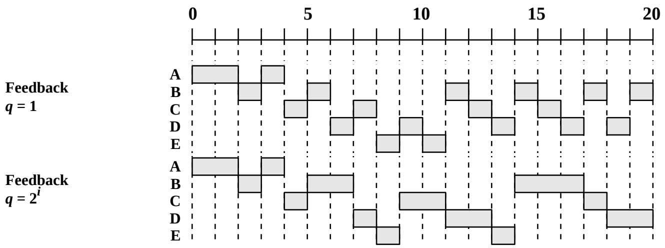
Slide 4

Priority queueing:



Slide 5

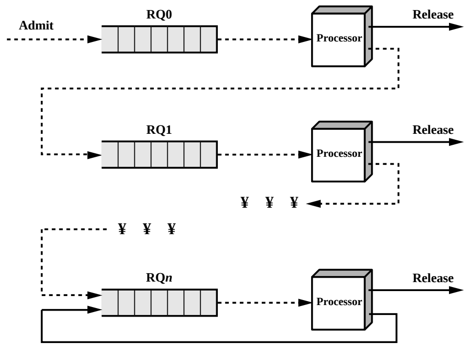
Feedback scheduling:



- ➜ Penalize jobs that have been running longer
- ➜ $q = 2^i$: longer time slice for lower priority (reduce starvation)

Slide 6

Feedback scheduling:



Priorities influence access to resources, but do not guarantee a certain fraction of the resource (CPU etc)!

Slide 7

LOTTERY SCHEDULING

- ➜ process gets "lottery tickets" for various resources
- ➜ more lottery tickets imply better access to resource

Advantages:

- ➜ Simple
- ➜ Highly responsive
- ➜ Allows cooperating processes/threads to implement individual scheduling policy (exchange of tickets)

Slide 8

Example (taken from *Embedded Systems Programming*:

Four processes a running concurrently

- ➜ Process A: 15% of CPU time
- ➜ Process B: 25% of CPU time
- ➜ Process C: 5% of CPU time
- ➜ Process D: 55% of CPU time

How many tickets should each process get to achieve this?

Number of tickets in proportion to CPU time, e.g., if we have 20 tickets overall

- ➜ Process A: 15% of tickets: 3
- ➜ Process B: 25% of tickets: 5
- ➜ Process C: 5% of tickets: 1
- ➜ Process D: 55% of tickets: 11

Realtime Systems

Slide 9

Overview:

- Real time systems
  - Hard and soft real time systems
  - Real time scheduling
  - A closer look at some real time operating systems

Real-Time Systems

Slide 10

What is a real-time system?

A real-time system is a system whose correctness includes its response time as well as its functional correctness.

What is a hard real-time system?

A real-time system with guaranteed worst case response times.

- Hard real-time systems fail if deadlines cannot be met
- Service of soft real-time systems degrades if deadlines cannot be met

Slide 11

Real-time systems:

- no clear separation
- system may meet hard deadline of one application, but not of other
- depending on application, time-scale may vary from microseconds to seconds
- most systems have some real-time requirements

Slide 12

Soft Real-time Applications:

- Many multi-media apps
- e.g., DVD or MP3 player
- Many real-time games, networked games

Hard Real-time Applications:

- Control of laboratory experiments
- Embedded devices
- Process control plants
- Robotics
- Air traffic control
- Telecommunications
- Military command and control systems

Slide 13

Hard real-time systems:

- ➜ often lack full functionality of modern OS
- ➜ secondary memory usually limited or missing
- ➜ data stored in short term or read-only memory
- ➜ no time sharing

Modern operating systems provide support for soft real-time applications

Hard real-time OS either specially tailored OS, modular systems, or customized version of general purpose OS.

Slide 14

CHARACTERISTICS OF REAL-TIME OPERATING SYSTEMS

Deterministic: How long does it take to acknowledge interrupt?

- ➜ Operations are performed at fixed, predetermined times or within predetermined time intervals
- ➜ Depends on
  - response time of system for interrupts
  - capacity of system
- ➜ Cannot be fully deterministic when processes are competing for resources
- ➜ Requires preemptive kernel

Responsive: How long does it take to service the interrupt?

- ➜ Includes amount of time to begin execution of the interrupt
- ➜ Includes the amount of time to perform the interrupt

Slide 15

CHARACTERISTICS OF REAL-TIME OPERATING SYSTEMS

User control: User has much more control compared to ordinary OS's

- ➜ User specifies priority
- ➜ Specify paging
- ➜ Which processes must always reside in main memory
- ➜ Disks algorithms to use
- ➜ Rights of processes

Reliability: Failure, loss, degradation of performance may have catastrophic consequences

- ➜ Attempt either to correct the problem or minimize its effects while continuing to run
- ➜ Most critical, high priority tasks execute

Slide 16

CHARACTERISTICS OF REAL-TIME OPERATING SYSTEMS

General purpose OS objectives like

- ➜ speed
- ➜ fairness
- ➜ maximising throughput
- ➜ minimising average response time

are not priorities in real time OS's!

Slide 17

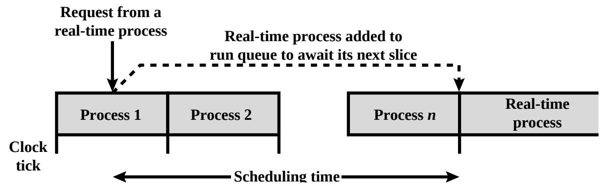
Features of real-time operating systems:

- ➜ Fast context switch
- ➜ Small size
- ➜ Ability to respond to external interrupts quickly
- ➜ Predictability of system performance!
- ➜ Use of special sequential files that can accumulate data at a fast rate
- ➜ Preemptive scheduling based on priority
- ➜ Minimization of intervals during which interrupts are disabled
- ➜ Delay tasks for fixed amount of time

Slide 18

REAL-TIME SCHEDULING

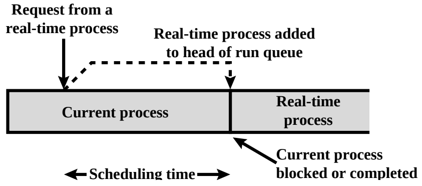
Preemptive round-robin:



Slide 19

REAL-TIME SCHEDULING

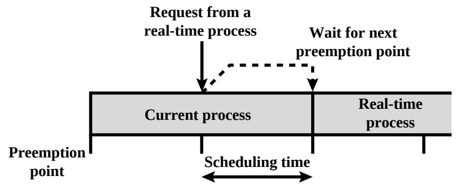
Non-preemptive priority:



Slide 20

REAL-TIME SCHEDULING

Preemption points:



Slide 21

Real-time scheduling

Immediate preemptive: