Linux

Linux

- Used on desktops, servers, smartphones, embedded devices
- As of Nov 2014, 97% of all HPC system ran Linux

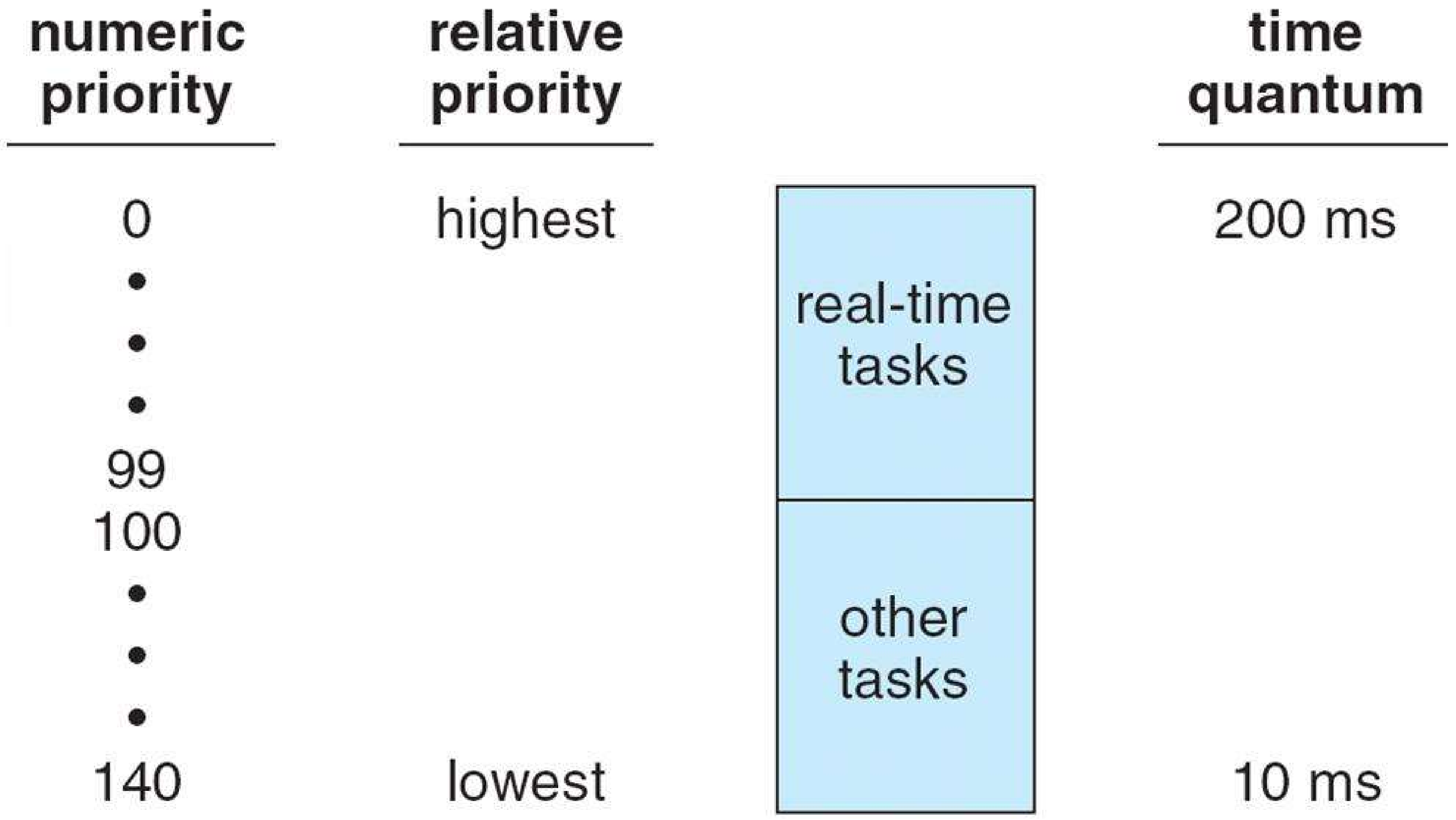
Scheduling through Version 2.5

- Prior to kernel version 2.5, Linux ran avariation of standard UNIX scheduling algorithm
- Version 2.5 moved to constant order O(1)
- Linux uses the term “task” for process

Linux 2.5 Scheduling

- Goals
  - Implement scheduling algorithms in O(1) time.
  - Optimize for the common case of only one or two runnable processes, yet scale well to multiple processors, each with many processes.
- Linux uses the term task to refer to a process/thread
- Linux priority levels are on the next slide

Linux 2.5 - Priority Levels



Priorities and Time-slice length

Linux 2.5 - Priorities

- Priorities 0-99 for real-time processes
- Priorities 100-139 for normal (user) processes

Linux 2.5 Scheduling

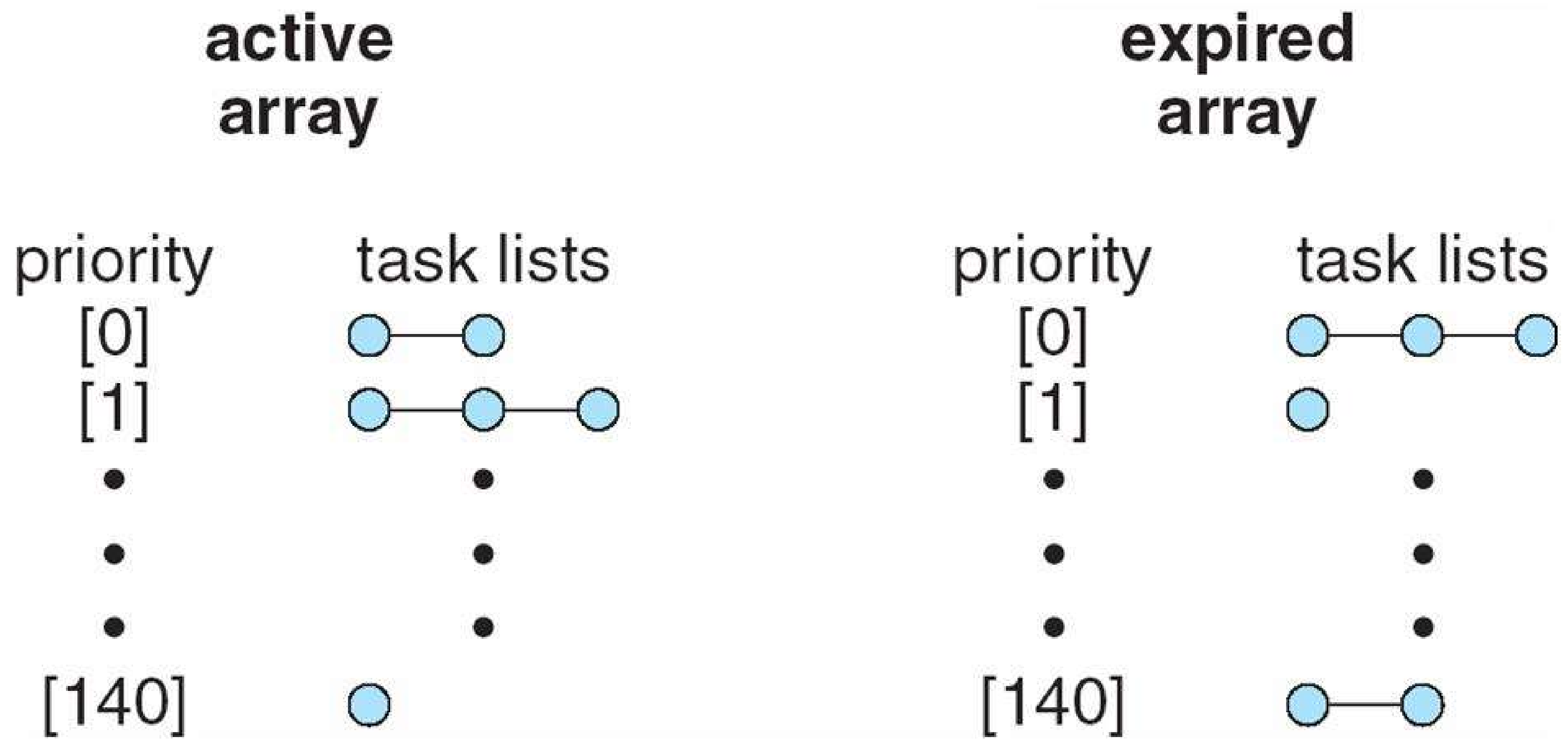
- A process is considered for execution on the CPU as long as it has time remaining in its time slice (quantum)

- When a task has exhausted its time slice, it is considered expired and is not eligible for execution again until all other tasks have all exhausted their time slice

Linux 2.5 Data Structures

- The kernel maintains a list of all runnable tasks in a runqueue data structure
- A runqueue consists of two priority arrays:
  - Active: Contains all tasks with time remaining in their time slices
  - Expired: Contains all expired tasks
- The active, expired queues are indexed according to priority

Linux 2.5 Data Structures



List of Tasks Indexed According to Priorities

Linux 2.5 Scheduling

- How is a process selected for execution?
  - The scheduler finds the highest-priority queue with a runnable process
  - Finds the first process on the queue

- What happens to a running process that does not complete its time quantum?
  - When that process returns to the ready state it goes into the active array

Linux 2.5 Scheduling

- When there are no more processes in the active array, the expired array becomes the active array and the active array becomes the expired array.

- A process's priority (queue) is recalculated when the process has exhausted its time quantum and is to be moved to the expired array
  - Calculation considers time spent waiting for I/O