• Linux Scheduling
  o Introduction
    ▪ Scheduling is based on threads, no processes (all threads are in the kernel)
    ▪ Uses priority-based scheduling with variable quanta and mix of preemption and non-preemption.
    ▪ Quantum is the number of clock ticks (1ms each, called a jiffy)
  o Three classes of threads
    ▪ Real-time FIFO – (not really real-time)
      • Highest priority
      • Cannot be preempted except by higher priority Real-time FIFO
    ▪ Real-time round robin
      • Same as previous except can be preempted by the clock
      • Real-time has priority from 0 – 99 (zero highest)
    ▪ Timesharing – standard threads
      • Priority assigned from 100-139
  o Priority levels assigned different quanta values
    ▪ 100 = 800ms
    ▪ 139 = 5ms
  o runqueue – main structure used to schedule ready threads
    ▪ Associated with each CPU in the system
    ▪ Maintains both an active and expired array for ready threads only
    • Each points to 140 doubly linked lists of processes for each priority
  o waitqueue – Queue associated with each event that a thread could wait on
- I/O, kernel events, etc.
- Linked list of tasks waiting for a particular event and a lock

  0. Scheduling Algorithm
    - Select a thread from the highest-priority active array
    - For all threads with a quantum that expires, move to an expired list
      - Priority is recalculated here
      - Some effort made to reward interactive threads (heuristically) and punish CPU-intensive
    - If a task blocks before the quantum expires, it gets put on the appropriate waitqueue
      - When the wait is over, the thread gets put back on the original *active* array with the execution time already used subtracted from the current quantum.
    - When there are no *active* threads, swap pointers with *expired* and start over

  0. Implications
    - Low priority threads will not starve
    - Scheduling done in O(1) time
    - Runqueue associated with each CPU
      - Affinity scheduling used when possible
      - System calls allow for users to set thread affinities
      - Periodic load-balancing is done
    - Multiple synchronization points within the kernel allow for less contention in multi-processor systems