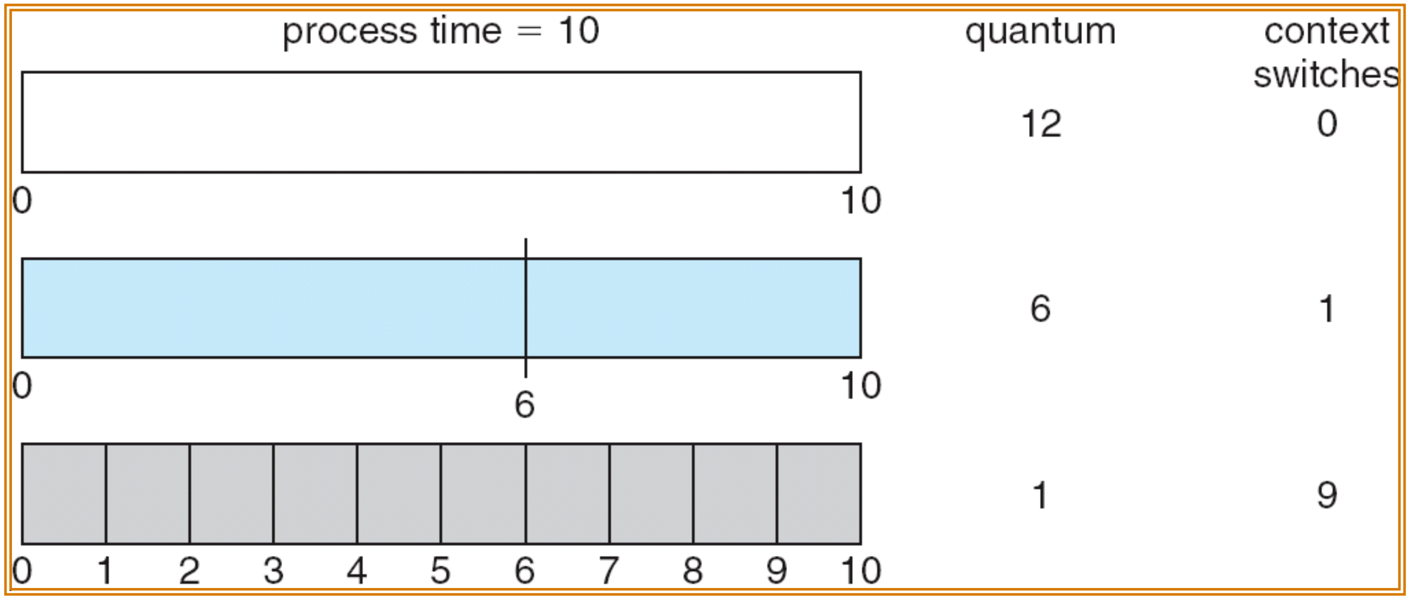
Recap: Scheduling algorithms

- First come, first serve - FCFS
- Shortest Job First
- Priority Scheduling
- Round robin

- Multi-level (different for different classes of processes)

Time Quantum and Context Switch Time

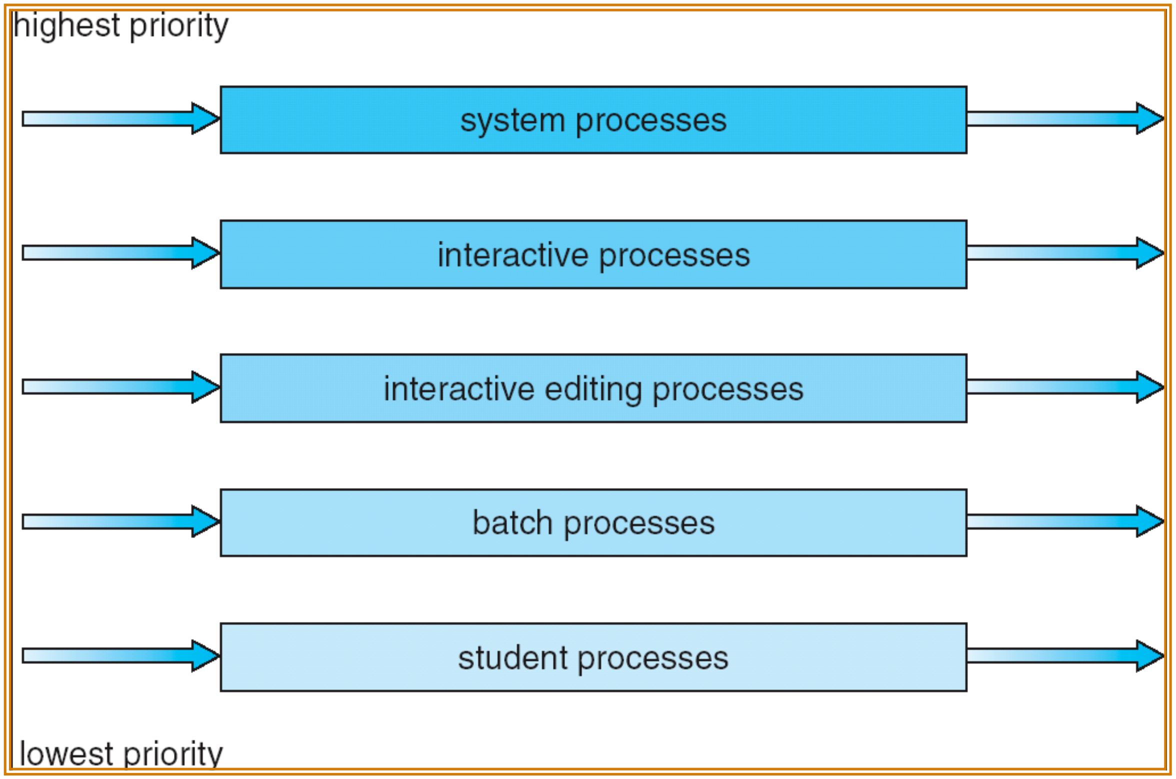


Rule of thumb: 80% of CPU bursts should be shorter than time quantum

Multilevel Queue

- Ready queue is partitioned into separate queues:
  foreground (interactive)
  background (batch)
- Each queue has its own scheduling algorithm
  - foreground – RR
  - background – FCFS
- Scheduling must be done between the queues
  - Fixed priority scheduling; (i.e., serve all from foreground then from background). Possibility of starvation.
  - Time slice – each queue gets a certain amount of CPU time which it can schedule amongst its processes; i.e., 80% to foreground in RR
  - 20% to background in FCFS

Multilevel Queue Scheduling



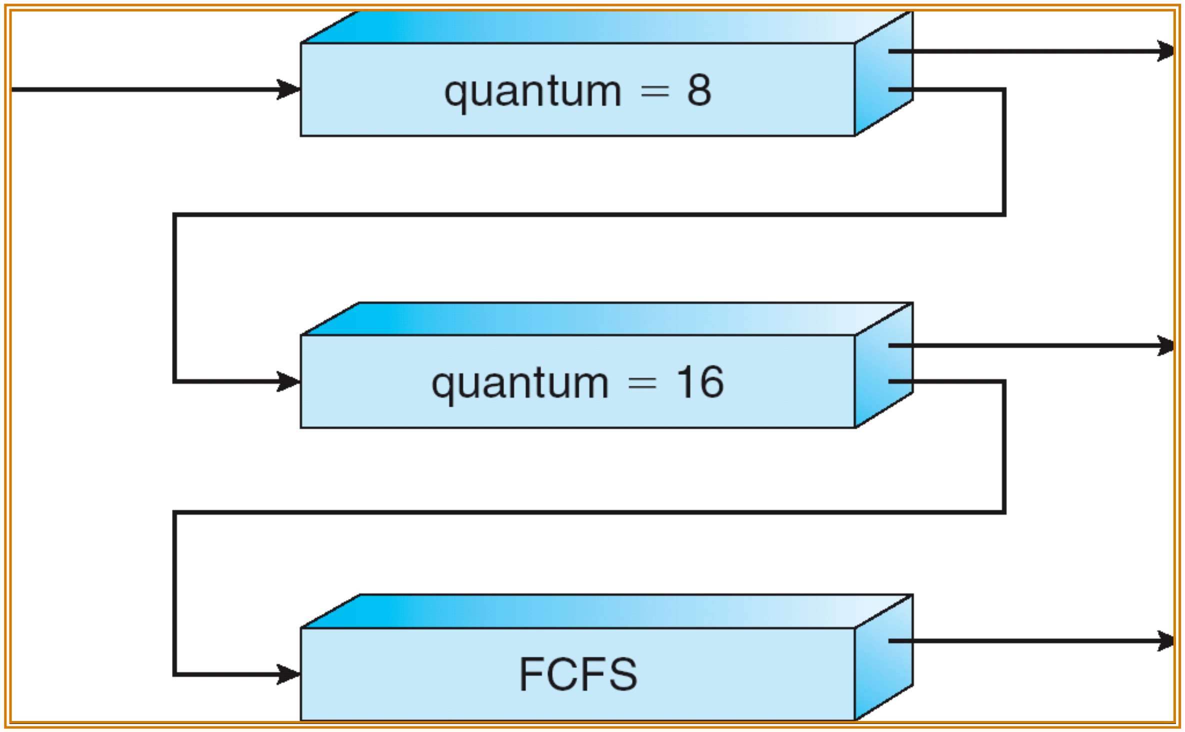
Multilevel Feedback Queue

- A process can move between the various queues; aging can be implemented this way
- Multilevel-feedback-queue scheduler defined by the following parameters:
  - number of queues
  - scheduling algorithms for each queue
  - method used to determine when to upgrade a process
  - method used to determine when to demote a process
  - method used to determine which queue a process will enter when that process needs service

Example of Multilevel Feedback Queue

- Three queues:
  - Q_0 – RR with time quantum 8 milliseconds
  - Q_1 – RR time quantum 16 milliseconds
  - Q_2 – FCFS
- Scheduling
  - A new job enters queue Q_0 which is served FCFS. When it gains CPU, job receives 8 milliseconds. If it does not finish in 8 milliseconds, job is moved to queue Q_1.
  - At Q_1 job is again served FCFS and receives 16 additional milliseconds. If it still does not complete, it is preempted and moved to queue Q_2.

Multilevel Feedback Queues



Multiple-Processor Scheduling

- CPU scheduling more complex when multiple CPUs are available
- We concentrate scenarios with homogeneous processors within a multiprocessor system
- Multiple processor scheduling makes load sharing possible
- Asymmetric multiprocessing – only one processor accesses the operating system data structures, alleviating the need for data sharing
  - Symmetric multiprocessing allows any processor to schedule itself

SMP concerns

- Processor affinity: Processes leave some state with a processor (caches). Processor affinity tries to balance using this state with load balancing
- Gang scheduling: schedule a group of processes/threads on a group of processors all at once (or none at all). These processes may communicate with each other and such scheduling might allow them all to make good progress together.

Real-Time Scheduling

- Hard real-time systems – required to complete a critical task within a guaranteed amount of time
- Soft real-time computing – requires that critical processes receive priority over less fortunate ones

Thread Scheduling

- Local Scheduling – How the threads library decides which thread to put onto an available LWP
- Global Scheduling – How the kernel decides which kernel thread to run next