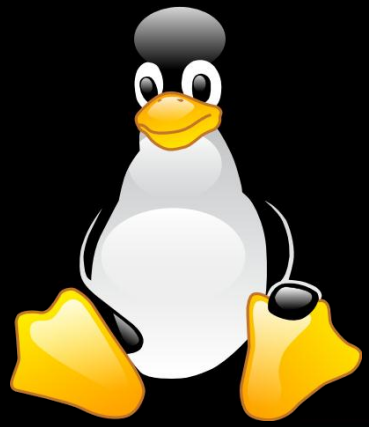
Tushar B Kute,
Assistant Professor,
Sandip Institute of Technology and Research Centre, Nashik (INDIA)
tbkute@gmail.com

University of Pune
T.E. I.T.
Subject code: 314441

OPERATING SYSTEM

Part 12: Scheduling Algorithms



CPU Scheduler

- Selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them
- CPU scheduling decisions may take place when a process:
  - Switches from running to waiting state
  - Switches from running to ready state
  - Switches from waiting to ready
  - Terminates
- Scheduling under 1 and 4 is nonpreemptive
- All other scheduling is preemptive

Dispatcher

- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler; this involves:
  - switching context
  - switching to user mode
  - jumping to the proper location in the user program to restart that program
- Dispatch latency – time it takes for the dispatcher to stop one process and start another running

Scheduling Criteria

- CPU utilization – keep the CPU as busy as possible
- Throughput – # of processes that complete their execution per time unit
- Turnaround time – amount of time to execute a particular process
- Waiting time – amount of time a process has been waiting in the ready queue
- Response time – amount of time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment)

Optimization Criteria

- Max CPU utilization
- Max throughput
- Min turnaround time
- Min waiting time
- Min response time

Scheduling Algorithms

- Preemptive
- Non-preemptive

Non-preemptive

- First Come First Serve
- Shortest Job First
- Priority

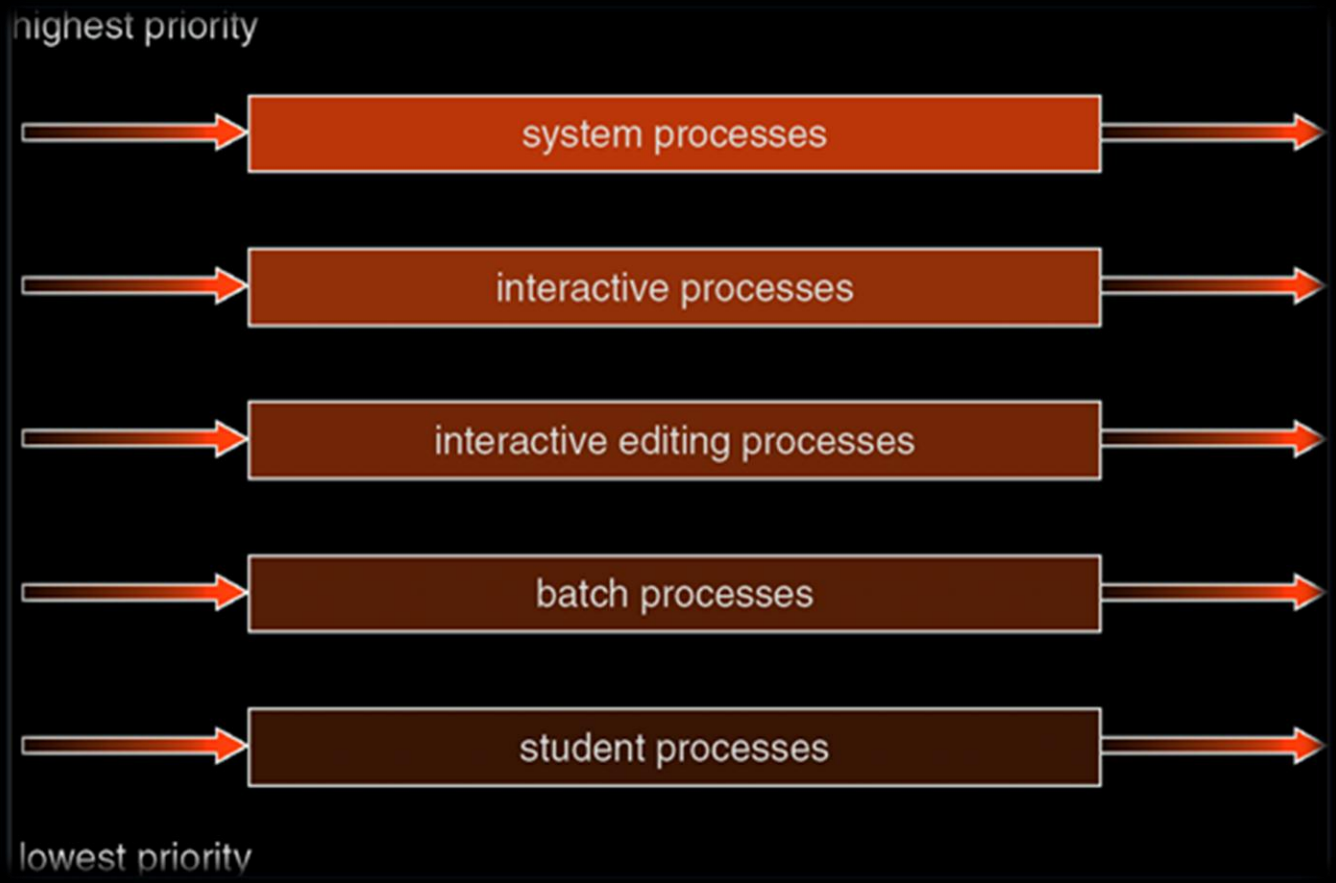
Preemptive

- Shortest Remaining Time First (SRTF)
- Round Robin

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



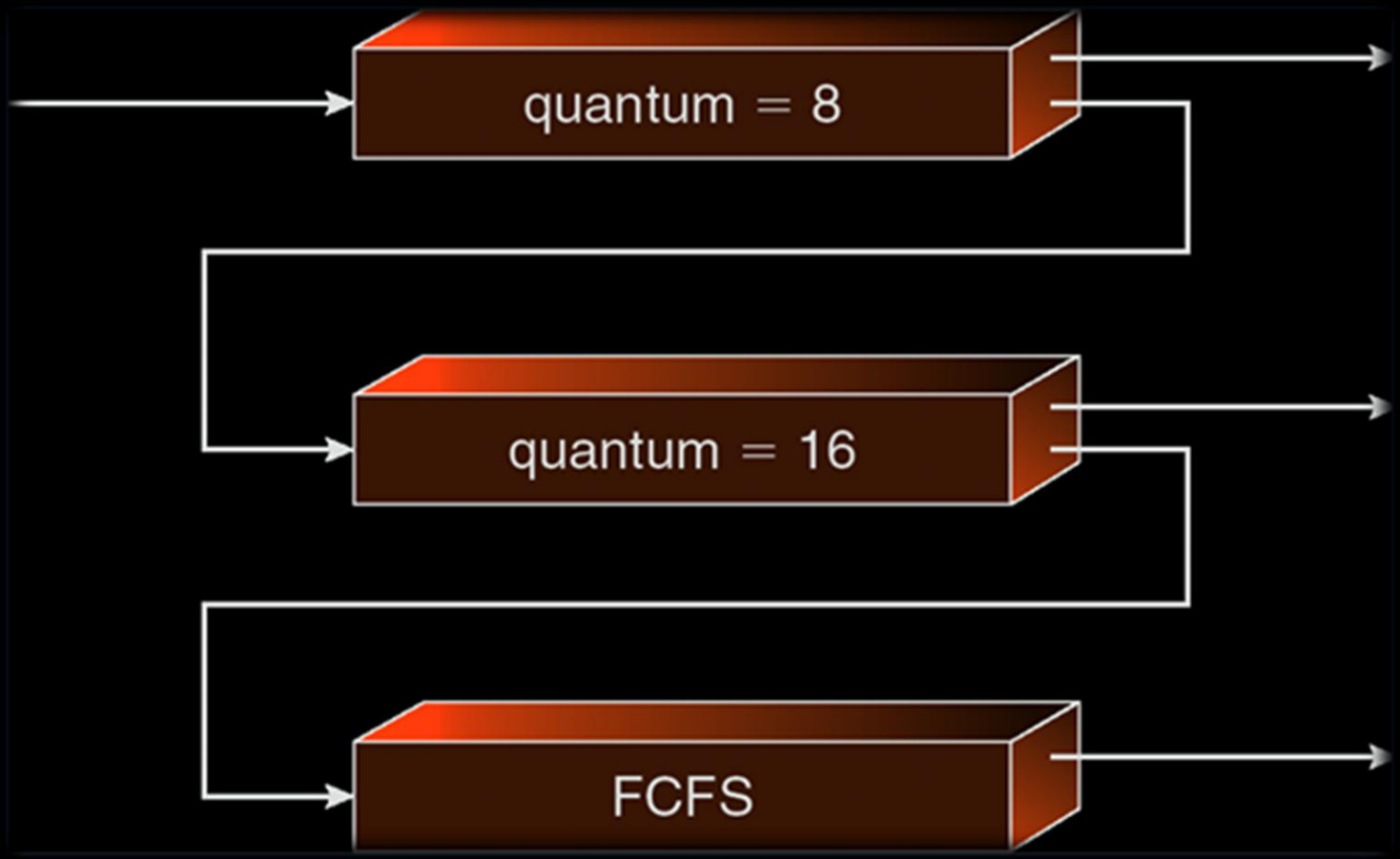
Multilevel feedback queue

- 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: multilevel feedback queue

- Three queues:
  - Q0 – RR with time quantum 8 milliseconds
  - Q1 – RR time quantum 16 milliseconds
  - Q2 – FCFS
- Scheduling
  - A new job enters queue Q0 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 Q1.
  - At Q1 job is again served FCFS and receives 16 additional milliseconds. If it still does not complete, it is preempted and moved to queue Q2.

Multilevel feedback queue



Multiple processor scheduling

- CPU scheduling more complex when multiple CPUs are available
- Homogeneous processors within a multiprocessor
- Load sharing
- Asymmetric multiprocessing – only one processor accesses the system data structures, alleviating the need for data sharing

Real-time scheduling

- Control of laboratory experiments.
- Process control in industrial plants.
- Robotics
- Air traffic control
- Telecommunications
- Military command and control systems

Characteristics

- Determinism
- Responsiveness
- User control
- Reliability
- Fail-soft operation

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

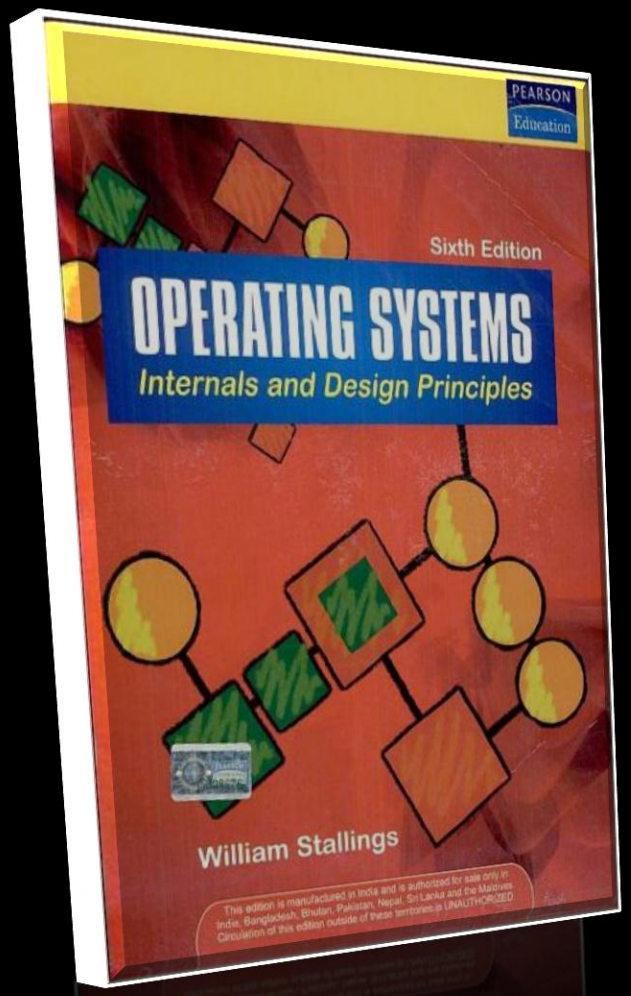
Classes of algorithms

- Static table-driven approaches:
  - These perform a static analysis of feasible schedules of dispatching. The result of the analysis is a schedule that determines, at run time, when a task must begin execution. E.g. earliest deadline first.
- Static priority-driven preemptive approaches:
  - Again, a static analysis is performed, but no schedule is drawn up. Rather, the analysis is used to assign priorities to tasks, so that a traditional priority-driven preemptive scheduler can be used. E.g. time sharing system.

Classes of algorithms

- Dynamic planning-based approaches:
  - Feasibility is determined at run time (dynamically) rather than offline prior to the start of execution (statically). An arriving task is accepted for execution only if it is feasible to meet its time constraints. One of the results of the feasibility analysis is a schedule or plan that is used to decide when to dispatch this task.
- Dynamic best effort approaches:
  - No feasibility analysis is performed. The system tries to meet all deadlines and aborts any started process whose deadline is missed.

Reference Books



- "Operating System: Internals and Design Principles" by *William Stallings*, Pearson Education.