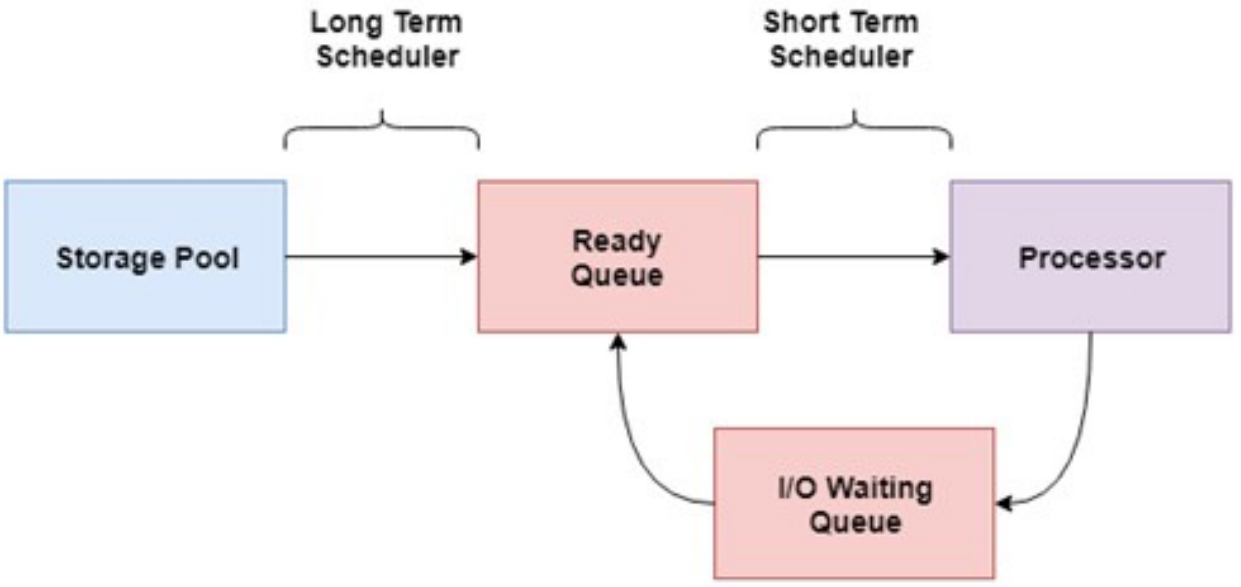
Scheduling Algorithms (Last part)

Scheduler



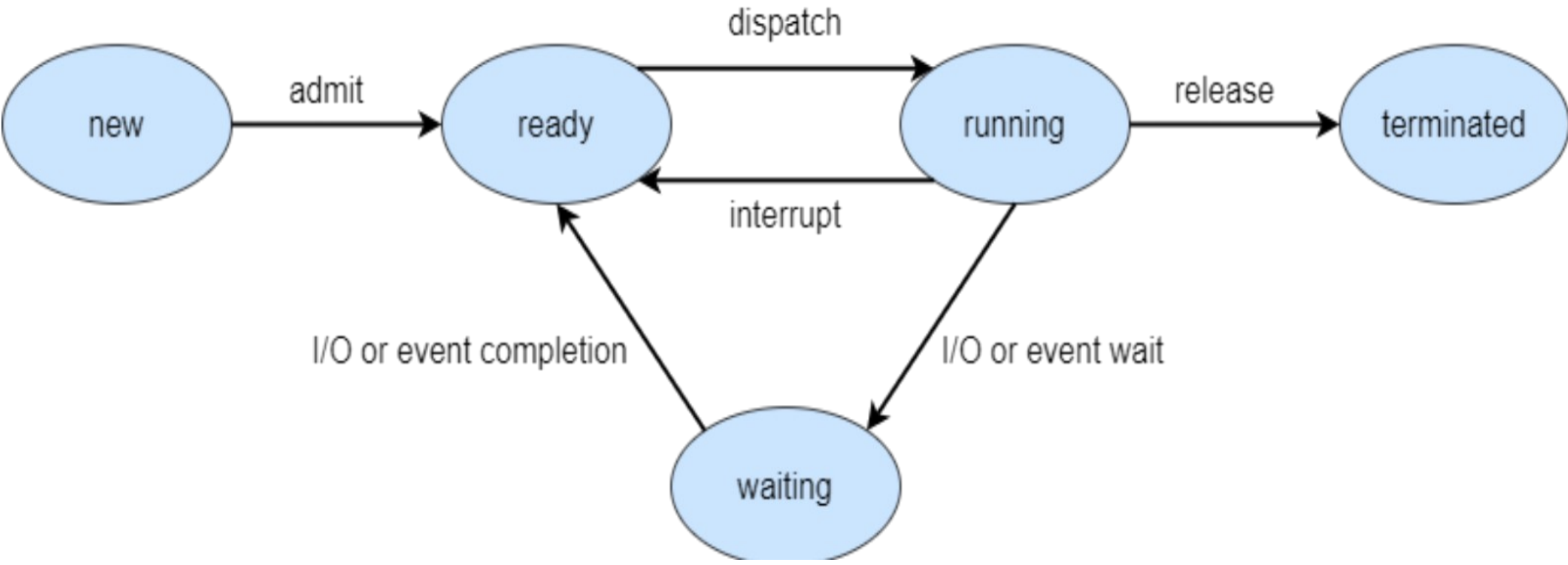
Representation of Short Term and Long Term Scheduler using a Queuing Diagram

Scheduler

Medium-Term Scheduling

Medium-term scheduling involves swapping out a process from main memory. The process can be swapped in later from the point it stopped executing. This can also be called as suspending and resuming the process and is done by the medium-term scheduler.

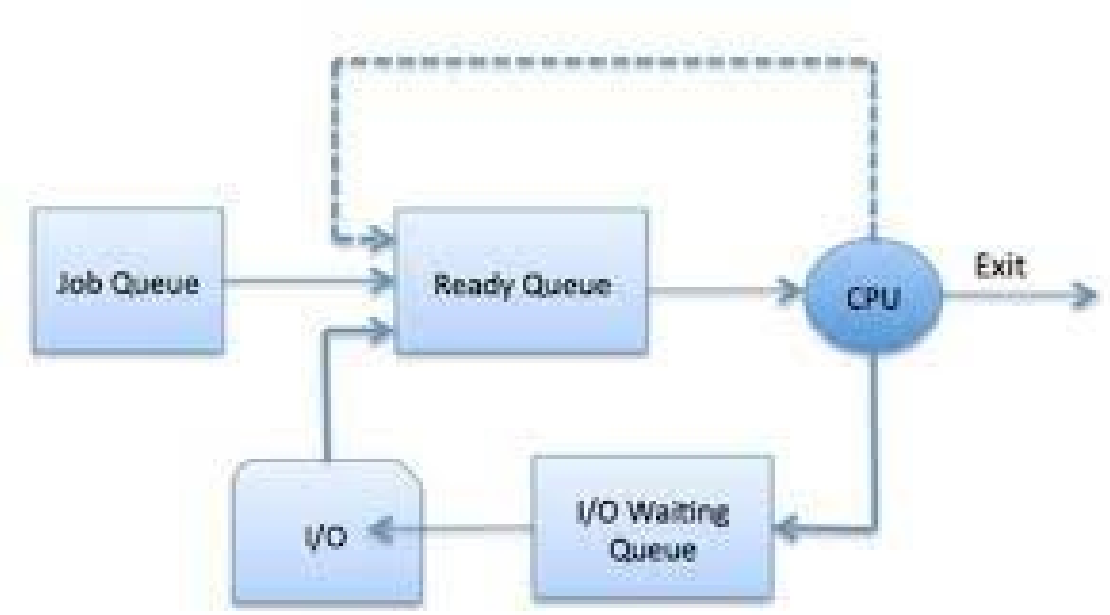
Process states



Process Queues

- Job Queue: In starting, all the processes get stored in the job queue. The long term scheduler (Job scheduler) picks some of the jobs and put them in the primary memory.
- Ready Queue: Ready queue is maintained in primary memory. The short term scheduler picks the job from the ready queue and dispatch to the CPU for the execution.
- Waiting Queue: When the process needs some IO operation in order to complete its execution, OS changes the state of the process from running to waiting. The process is shifted to the waiting queue.

Process Queues



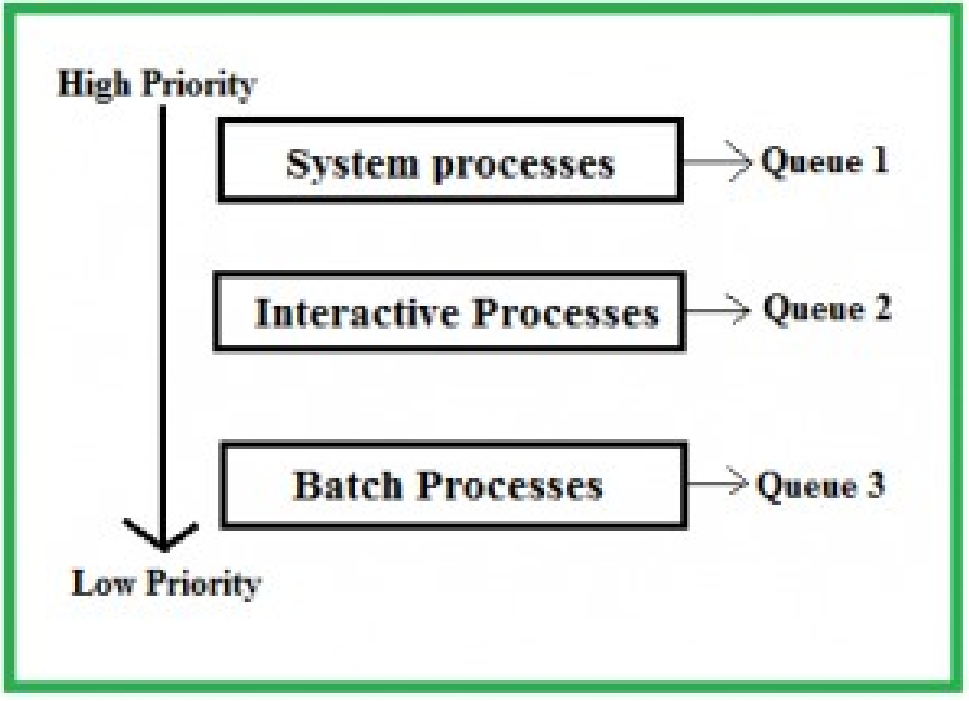
Multilevel Queue Scheduling

- The processes in the ready queue can be divided into different classes where each class has its own scheduling needs.
- For example, a common division is a foreground (interactive) process and a background (batch) process

Multilevel Queue Scheduling

- Ready Queue is divided into separate queues for each class of processes.
- For example, let us take three different types of processes System processes, Interactive processes, and Batch Processes.
- All three processes have their own queue.

Multilevel Queue Scheduling



Multilevel Queue Scheduling

- All three different type of processes can have their own queue.
- Each queue has its own Scheduling algorithm.
- For example, queue 1 and queue 2 uses Round Robin while queue 3 can use FCFS to schedule their processes.

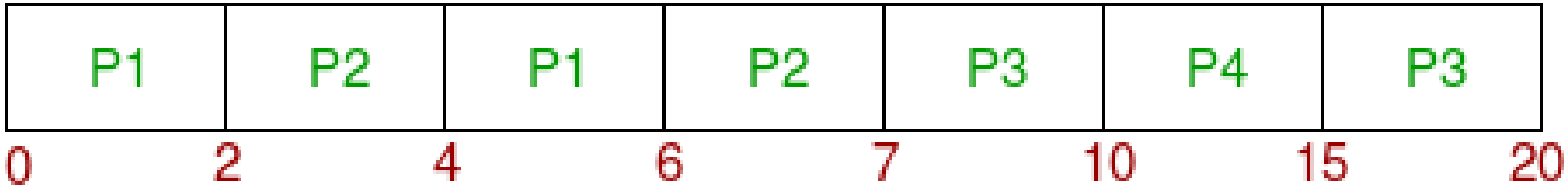
Multilevel Queue Scheduling

- To determine Scheduling among the queues, there are two ways
  - Fixed priority preemptive scheduling method: Each queue has absolute priority over the lower priority queue. According to this algorithm, no process in the batch queue(queue 3) can run unless queues 1 and 2 are empty.
  - Time slicing: In this method, each queue gets a certain portion of CPU time and can use it to schedule its own processes. For instance, queue 1 takes 50 percent of CPU time queue 2 takes 30 percent and queue 3 gets 20 percent of CPU time.

- Consider below table of four processes under Multilevel queue scheduling. Queue number denotes the queue of the process.

Process	Arrival Time	CPU Burst Time	Queue Number
P1	0	4	1
P2	0	3	1
P3	0	8	2
P4	10	5	1

Priority of queue 1 is greater than queue 2. queue 1 uses Round Robin (Time Quantum = 2) and queue 2 uses FCFS.


P1
P2
P1
P2
P3
P4
P3
0
2
4
6
7
10
15
20

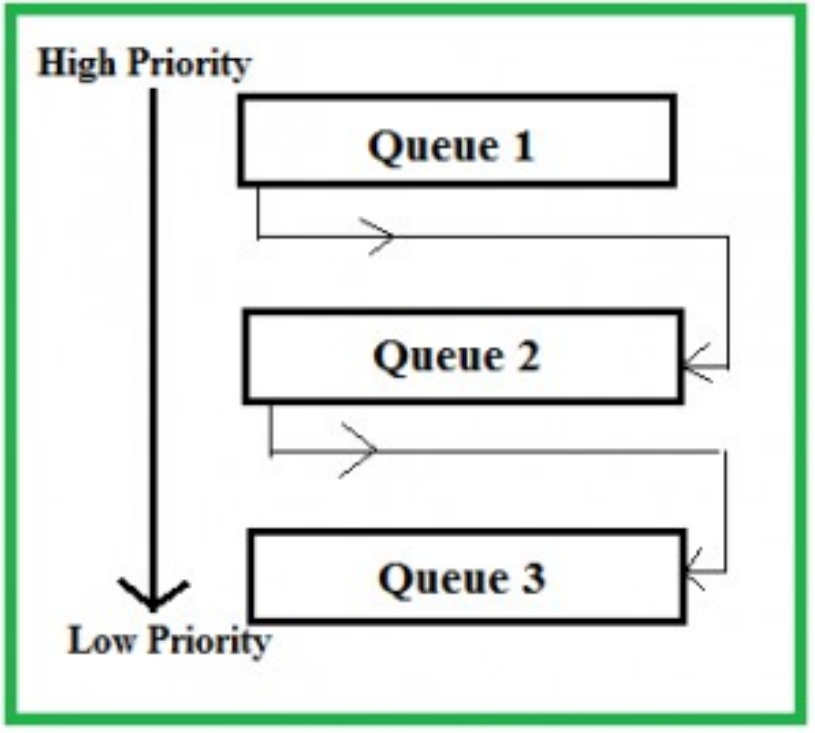
Multilevel Feedback Queue Scheduling

- In a multilevel queue-scheduling algorithm, processes are permanently assigned to a queue on entry to the system and processes are not allowed to move between queues.
- It allows different processes to move between different queues.
- It prevents starvation by moving a process that waits too long for the lower priority queue to the higher priority queue.

Multilevel Feedback Queue Scheduling

- Multilevel feedback queue scheduling allows a process to move between queues.
- Multilevel Feedback Queue Scheduling (MLFQ) keeps analyzing the behavior (time of execution) of processes and according to which it changes its priority.

Multilevel Feedback Queue Scheduling



Multilevel Feedback Queue Scheduling

- When a process starts executing the operating system can insert it into any of the above three queues depending upon its priority. Let's say our current process for consideration is of significant priority so it will be given queue 1.
- In queue 1 process executes for 4 units and if it completes in these 4 units or it gives CPU for I/O operation in these 4 units then the priority of this process does not change and if it again comes in the ready queue then it again starts its execution in Queue 1.
- Above points 2 and 3 are also true for queue 2 processes but the time quantum is 8 units. If a process in queue 1 does not complete in 4 units then its priority gets reduced and it shifted to queue 2.

Multilevel Feedback Queue Scheduling

- If a process does not complete in a time quantum then it is shifted to the lower priority queue.
- In the last queue, processes are scheduled in an FCFS manner.
- A process in a lower priority queue can only execute only when higher priority queues are empty.
- A process running in the lower priority queue is interrupted by a process arriving in the higher priority queue.

Multilevel Feedback Queue Scheduling

- Problems in the implementation: A process in the lower priority queue can suffer from starvation due to some short processes taking all the CPU time.
- Solution: A simple solution can be to boost the priority of all the processes after regular intervals and place them all in the highest priority queue.

Multilevel Feedback Queue Scheduling

Consider a system that has a CPU-bound process, which requires a burst time of 40 seconds. The multilevel Feed Back Queue scheduling algorithm is used and the queue time quantum ‘2’ seconds and in each level it is incremented by ‘5’ seconds. Then how many times the process will be interrupted and in which queue the process will terminate the execution?

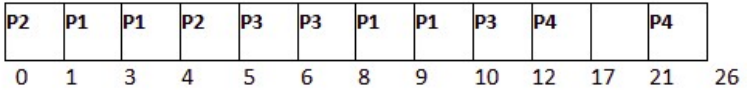
Multilevel Feedback Queue Scheduling

- Process P needs 40 Seconds for total execution.
- At Queue 1 it is executed for 2 seconds and then interrupted and shifted to queue 2.
- At Queue 2 it is executed for 7 seconds and then interrupted and shifted to queue 3.
- At Queue 3 it is executed for 12 seconds and then interrupted and shifted to queue 4.
- At Queue 4 it is executed for 17 seconds and then interrupted and shifted to queue 5.
- At Queue 5 it executes for 2 seconds and then it completes.
- Hence the process is interrupted 4 times and completed on queue 5.

SRTF problem

Process Id	Arrival Time	(Burst Time, IO Burst Time, Burst Time)
1	0	(3,2,2)
2	0	(1,3,1)
3	3	(3,1,2)
4	6	(5,4,5)

SRTF problem



Process Id	Arrival Time	Total CPU Burst Time	Completion Time	Turn Around Time	Waiting Time
1	0	5	10	10	5
2	0	2	5	5	3
3	3	5	12	9	4
4	6	10	26	20	10

Average waiting Time = (5+3+4+10)/4 = 22/4 units