Operating System
Contact Session 10

Dr. Lucy J. Gudino
WILP & Department of CS & IS

BITS Pilani
Pilani Campus

innovate achieve lead

BITS Pilani

Pilani Campus

Process Scheduling

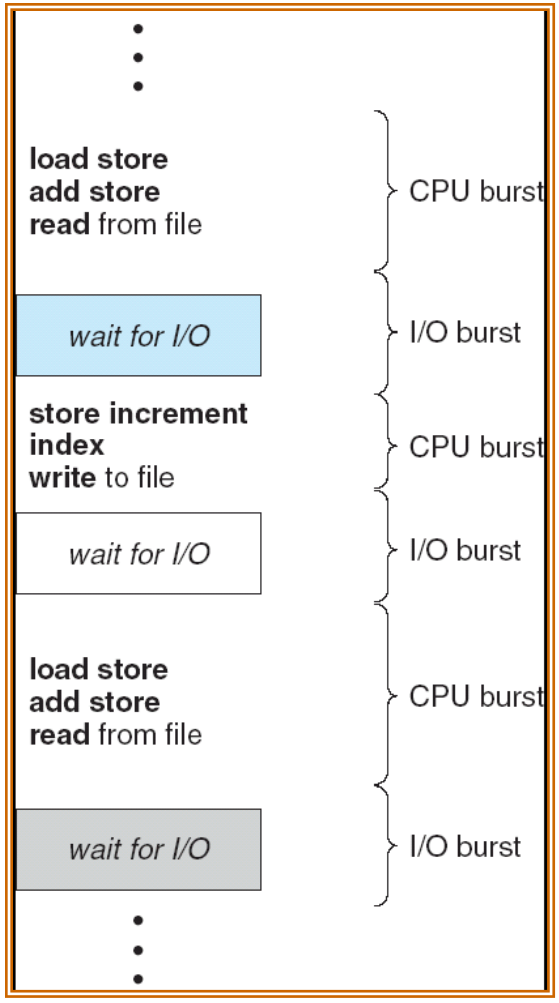
Scheduling: Basic Concepts

innovate achieve lead

- **Scheduling**: The assignment of CPU to a process
- **Preemption**: Removal of CPU control from a process
- **Context Switch**: Switching the CPU to another process
  - Save the state of old process
  - Load the saved state of new process
- **PCB** is used to save the information /context of a process

Scheduling: Basic Concepts…

- Maximum CPU utilization obtained with multiprogramming
- Process execution consists of a *cycle* of CPU execution and I/O wait
  - CPU–I/O Burst Cycle
- CPU burst distribution



Process scheduling

- Job queue – set of all processes in the system
- Ready queue – set of all processes residing in main memory, ready and waiting to execute
- Device queues – set of processes waiting for an I/O device
- Processes migrate among the various queues

Types of Scheduler

- Long-term scheduler
- Medium-term scheduler
- Short-term scheduler
- I/O scheduler

Long Term Schedulers

- Also known as Job Scheduler
- Decides which processes should be brought into the ready queue.
- The long-term scheduler controls the **degree of multiprogramming**
  - Main Goal : to have good mix of CPU bound and I/O bound processes
- Executes infrequently, maybe only when process leaves system
- Does not exist on most modern timesharing systems

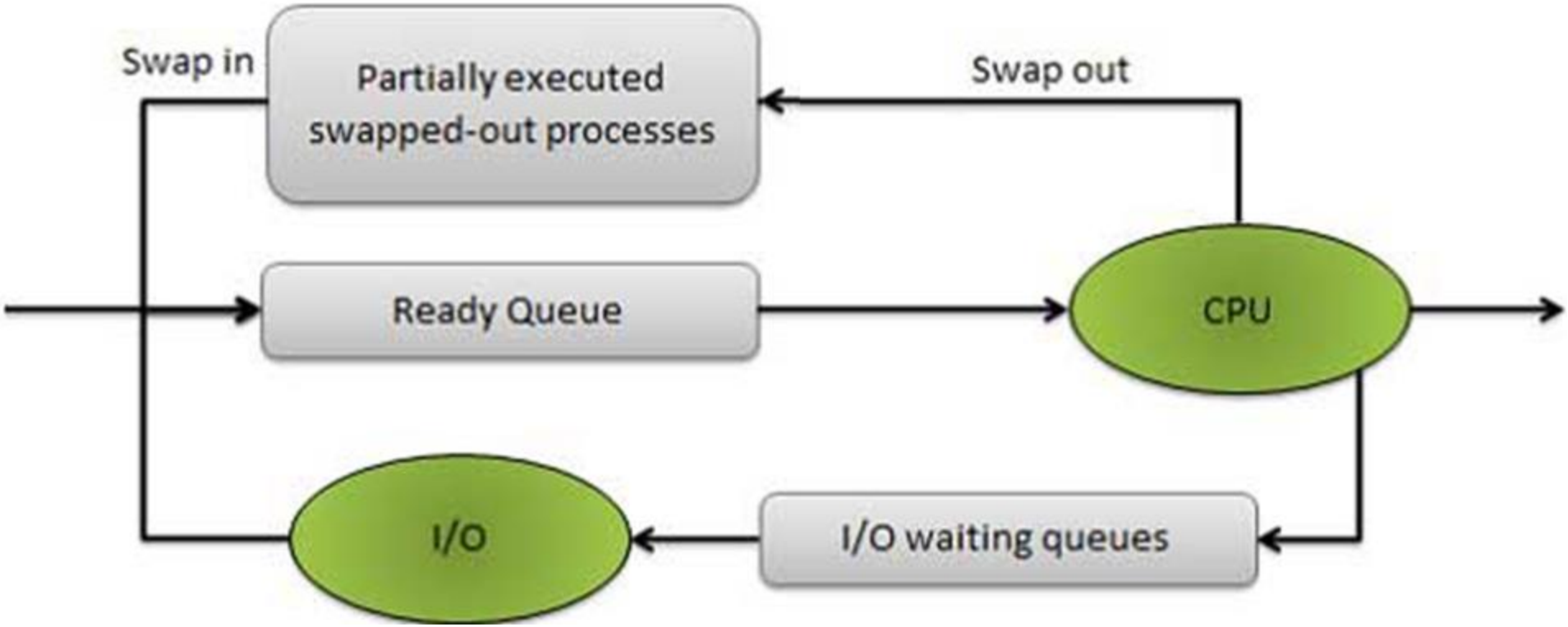
Short Term Scheduler

- Also known as CPU Scheduler or dispatcher
- selects from among the processes that are ready to execute and allocates the CPU to one of them
- Executes frequently
- Runs whenever
  - Process is created and brought in to ready queue
  - Process is terminated
  - Process switches from running to blocked (wait)
  - When interrupt occurs

Short – Term Scheduler…

- Main Goals:
  - Minimize response time
  - Maximize throughput
    - Minimize overhead such as OS overhead, context switching etc.
    - Efficient use of resources
  - Fairness – Share CPU and other resources in an equitable fashion

Addition of Medium Term Scheduling



Contd…

The mid-term scheduler may decide to swap out

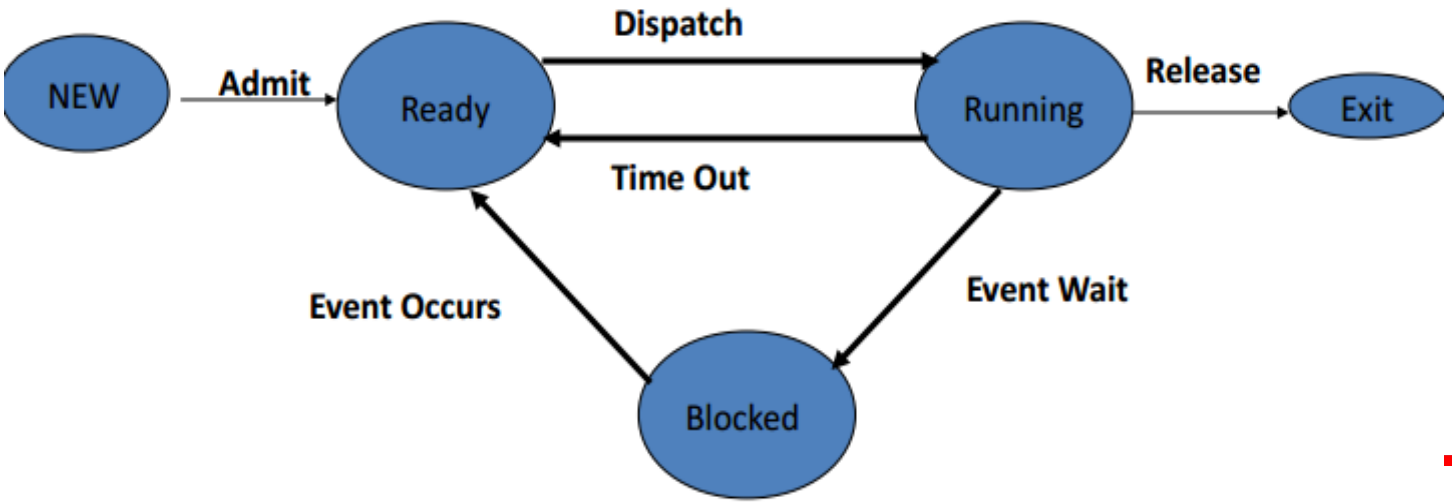
1. a process which has not been active for some time
2. a process which has a low priority
3. a process which is page faulting frequently
4. a process which is taking up a large amount of memory in order to free up main memory for other processes, swapping the process back in later when more memory is available,
5. when the process has been unblocked and is no longer waiting for a resource.

I/O Scheduling

The decision as to which process's pending I/O request shall be handled by an available I/O device

CPU Scheduler

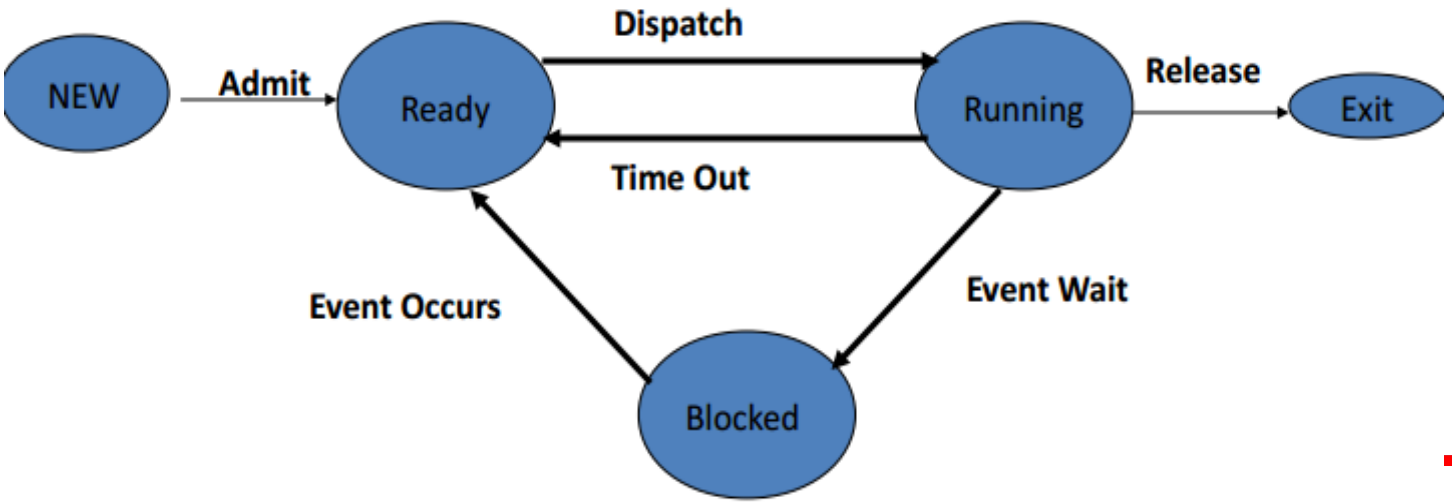
- CPU scheduling decisions may take place when a process:
  - 1.Switches from running to waiting(blocked) state
  - 2.Switches from running to ready state
  - 3.Switches from waiting (blocked) to ready state
  - 4.Switches from running to terminate state



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:
  - 1.Switches from running to waiting (blocked) state
  - 2.Switches from running to ready state
  - 3.Switches from waiting (blocked) to ready
  - 4. Switches from running to terminate sta

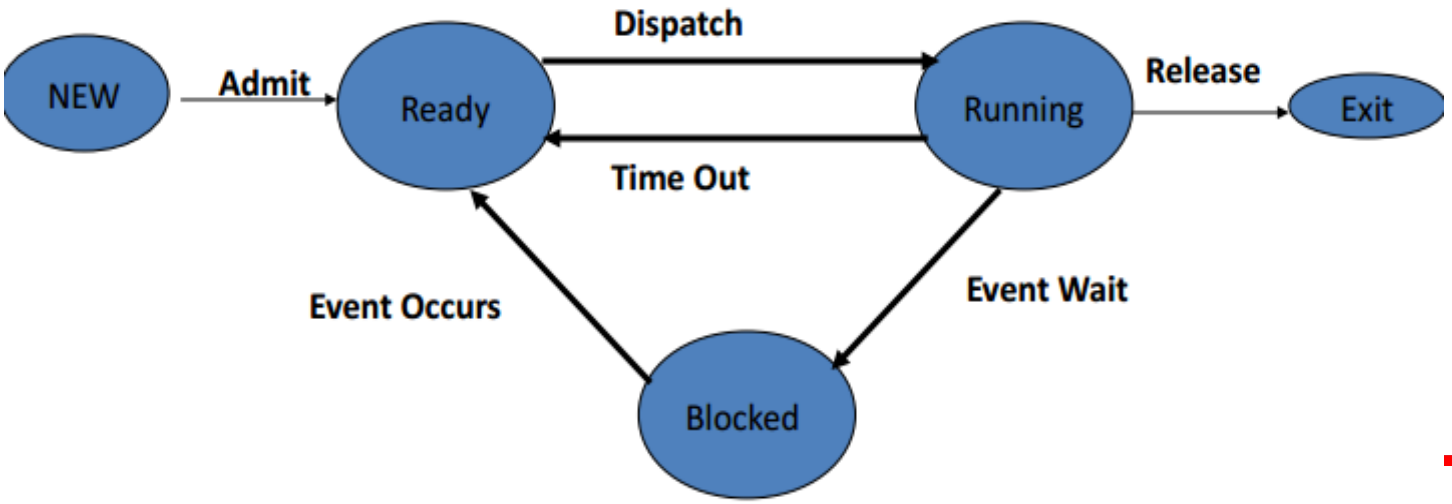
No choice! New process must be selected



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:
  - 1.Switches from running to waiting state
  - 2.Switches from running to ready state
  - 3.Switches from waiting to ready
  - 4.Terminates

There is a choice!



Preemptive and Non-preemptive scheduling

- Non-preemptive scheduling
  - A new process is selected to run either
    - when a process terminates or
    - when an explicit system request causes a wait state
      - (e.g., I/O or wait for child)
- Preemptive scheduling
  - New process selected to run when
    - An interrupt occurs
    - When new processes become ready

Dispatcher

- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler;
- Function of Dispatcher :
  - 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

Good Scheduling Algorithm

- Be fair : don't allow process to starve
- Be efficient : maximize CPU utilization
- Maximize throughput
  - Throughput : number of processes completed in unit time
- Minimize response time
  - Response time : time measured from process creation to the time of first output (response).
- Minimize waiting time
  - Waiting time : the amount of time a process has been waiting in the ready queue

Good Scheduling Algorithm…

- Be predictable : a given job should take about the same amount of time to run when run multiple times
- Minimize overhead :Keep scheduling time and context switch time at a minimum
- Maximize resource use : favor processes that will use underutilized resources
- Avoid indefinite postponement : every process should get a chance to run eventually
- Enforce priorities
- Degrade gracefully : as the system becomes more heavily loaded, performance should deteriorate gradually, not abruptly

Performance Metrics

- CPU utilization – keep the CPU as busy as possible. CPU utilization vary from 0 to 100. It varies from 40 (lightly loaded) to 90 (heavily loaded)
- Throughput – Number of processes that complete their execution per time unit.
- Turnaround time – Amount of time to execute a particular process (interval from time of submission to time of completion of process).
- Response time
- Waiting time

Scheduling Algorithms

1. First Come First Served (FCFS)
2. Shortest Job First (SJF)
3. Shortest Remaining Time First (SRTF)
4. Priority Scheduling
5. Round Robin Scheduling
6. Multi level Feedback Queue

Problems

BITS Pilani
Pilani Campus

Scheduling Algorithm - FCFS

- Simplest scheduling algorithm.
- Non-preemptive type of scheduling.
- Process which requests the CPU first is allocated the CPU first.
- The implementation of FCFS is managed with a FIFO queue.

FCFS: Example 1

- Draw Gantt chart
- Compute the average wait time, TAT and RT for processes

Process	AT	BT	FT	TAT	WT	RT
P1	0	7				
P2	0	3				
P3	0	4				
P4	0	6				

FCFS: Example 2

- Draw Gantt chart
- Compute the average wait time, TAT and RT for processes

Process	AT	BT	FT	TAT	WT	RT
P1	0	7				
P2	8	3				
P3	3	4				
P4	5	6				

FCFS: Example 3

- Draw Gantt chart
- Compute the average wait time, TAT and RT for processes

Process	AT	BT	FT	TAT	WT	RT
P1	0	2				
P2	3	1				
P3	5	5				

FCFS Drawbacks

- A process that does not perform any I/O will monopolize the processor (Convoy Effect).
- Favors CPU-bound processes:
  - I/O-bound processes have to wait until CPU-bound process completes.
  - They may have to wait even when their I/O are completed (poor device utilization).

SJF: Shortest Job First

- The job with the shortest next CPU burst time is selected
- Associates with each process the length of its **next CPU burst**.
- Use these lengths to schedule the process with the shortest time.
- If two processes have the same **next CPU burst**, **FCFS** is used to break the tie.
- also known as "**shortest next CPU burst algorithm**"

SJF: Shortest Job First ➔ Types

- Two schemes:
  - nonpreemptive – once CPU given to the process, it cannot be preempted until completes its CPU burst
  - preemptive – if a new process arrives with CPU burst length less than remaining time of current executing process, preempt. This scheme is known as the Shortest-Remaining-Time-First (SRTF)
- SJF is optimal – gives minimum average waiting time for a given set of processes

SJF (non-preemptive): Example

- Draw Gantt chart
- Compute the average wait time, TAT and RT for processes

Process	AT	BT	FT	TAT	WT	RT
P1	0	7				
P2	0	3				
P3	0	4				
P4	0	6				

SJF (non-preemptive): Example 5

innovate achieve lead

- Draw Gantt chart
- Compute the average wait time, TAT and RT for processes

Process	AT	BT	FT	TAT	WT	RT
P1	0	7				
P2	8	3				
P3	3	4				
P4	5	6				

SJF (Preemptive) / SRTF: Example 6

- Draw Gantt chart
- Compute the average wait time, TAT and RT for processes

Process	AT	BT	FT	TAT	WT	RT
P1	0	7				
P2	8	3				
P3	3	2				
P4	5	6				

SJF…

- Optimal: Shortest average wait time
- It's unfair !!
  - Continuous stream of short jobs will starve long job
- SJF scheduling is used frequently in long-term scheduling.
  - user needs to estimate the process time
- Short term Scheduling: Need to know the execution time of a process
  - May have an estimate, to predict next CPU burst
- Jobs are organized in order of estimated completion time

SJF Drawbacks

- Possibility of starvation for longer processes as long as there is a steady supply of shorter processes.
- CPU bound process gets lower priority but a process doing no I/O could still monopolize the CPU if it is the first one to enter the system.

Note: SJF implicitly incorporates priorities: shortest jobs are given preferences.

Priority Scheduling

- A priority number (integer) is associated with each process
- The CPU is allocated to the process with the highest priority (smallest integer ≡ highest priority)
  - Preemptive
  - nonpreemptive
- SJF is a special case of priority scheduling where priority is the predicted next CPU burst time
- Problem ≡ Starvation – low priority processes may never execute
- Solution ≡ Aging – as time progresses increase the priority of the process

Note: Some use larger integer ≡ highest priority)

Priority (non-preemptive): Example 7

- Draw Gantt chart (Lower Number Higher priority,)
- Compute the average wait time, TAT and RT for processes

Process	AT	BT	Pri	FT	TAT	WT	RT
P1	0	3	5				
P2	2	2	3				
P3	3	5	2				
P4	4	4	4				
P5	6	1	1				

Priority (preemptive): Example 8

- Draw Gantt chart
- Compute the average wait time, TAT and RT for processes

Process	AT	BT	Pri	FT	TAT	WT	RT
P1	0	3	5				
P2	2	2	3				
P3	3	5	2				
P4	4	4	4				
P5	6	1	1				

Round Robin (RR)

- Each process gets a small unit of CPU time (*time quantum*), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.
- If there are *n* processes in the ready queue and the time quantum is *q*, then each process gets 1/*n* of the CPU time in chunks of at most *q* time units at once. No process waits more than (*n*-1)*q* time units.
- Performance
  - *q* large $\Rightarrow$ FCFS
  - *q* small $\Rightarrow$ *q* must be large with respect to context switch, otherwise overhead is too high

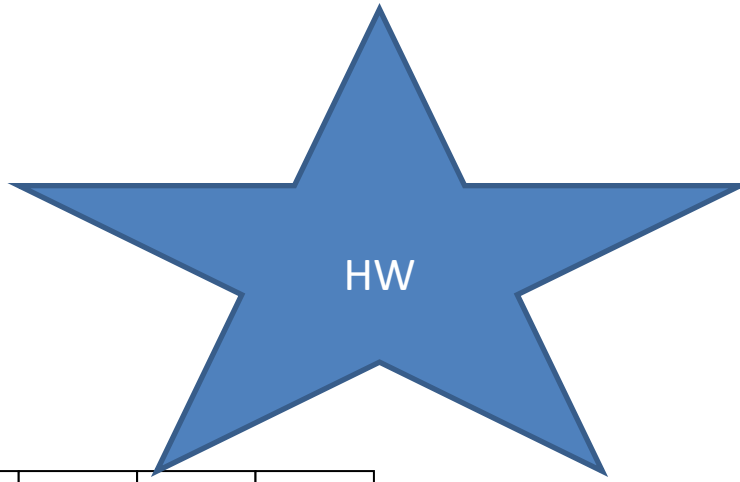
Round Robin: Example 9

- Draw Gantt chart , Time Quantum = 4
- Compute the average wait time, TAT and RT for processes

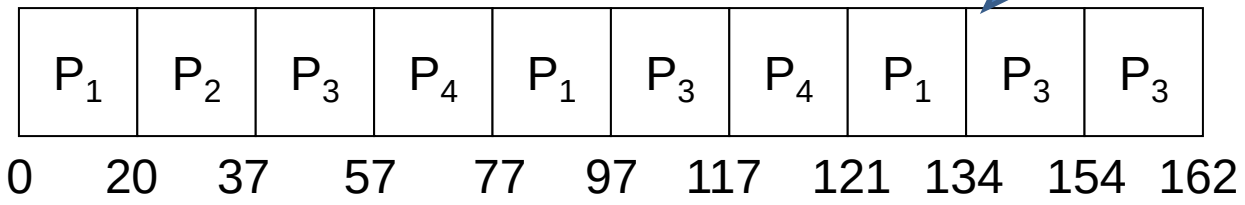
Process	AT	BT	FT	TAT	WT	RT
P1	5	5				
P2	4	6				
P3	3	7				
P4	1	9				
P5	2	2				
P6	6	3				

Example 10 : RR with Time Quantum = 20

Process	Burst Time
P_1	53
P_2	17
P_3	68
P_4	24



The Gantt chart is:

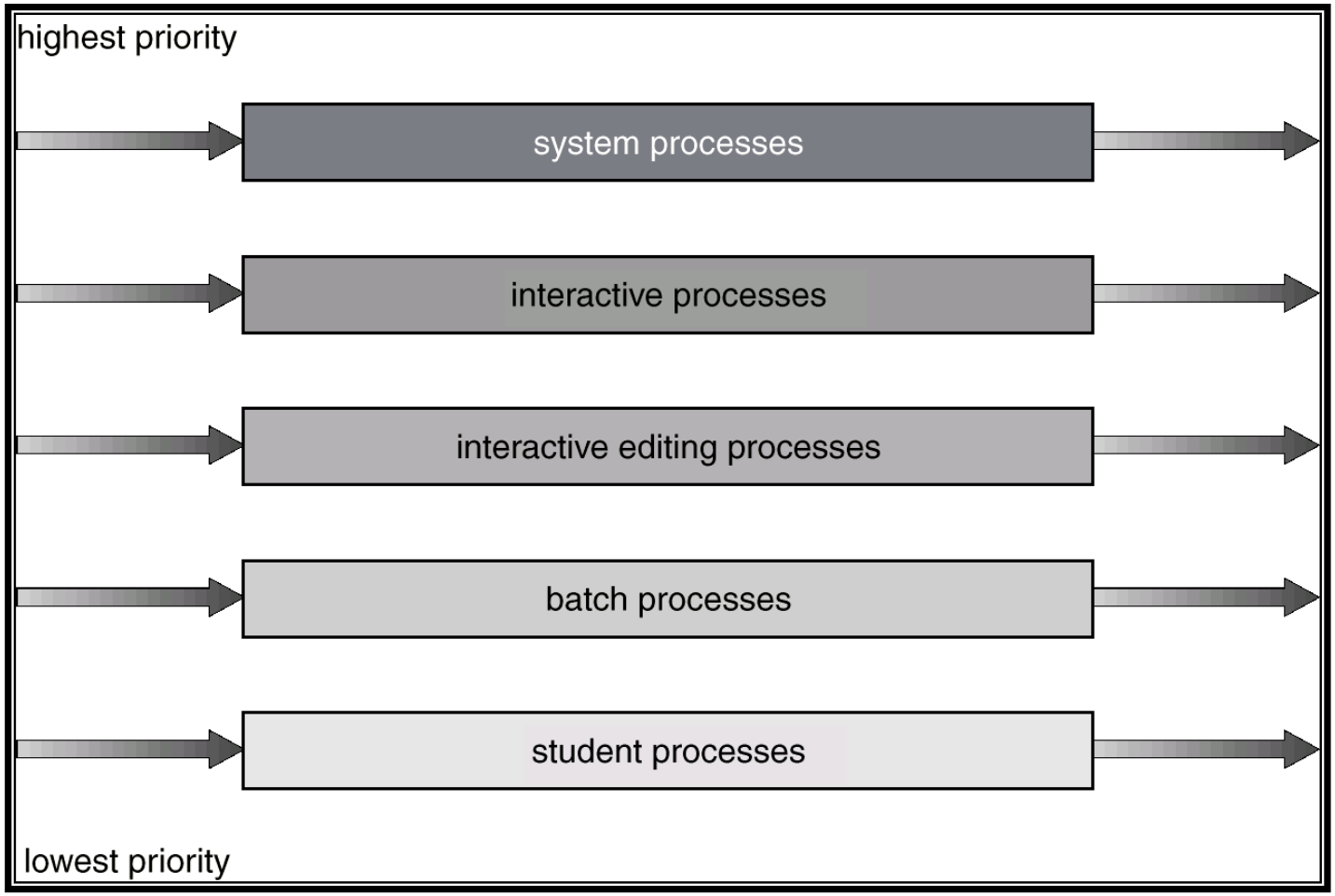


Typically, higher average turnaround than SJF, but better *response*

Multilevel Queuing

- Process classification based on response-time requirements and scheduling needs
  - Foreground processes
  - background processes
- Foreground processes may have priority (externally defined) over background processes
- Multilevel queue scheduling algorithm
  - Partition the ready queue into several separate queues
  - Each queue has its own scheduling algorithm
  - Example: foreground and background processes.
    - The foreground queue scheduled by an RR algorithm
    - background queue is scheduled by an FCFS algorithm.
- Process assignment to queue based on
  - Memory size, priority of process or process type

Multilevel Queue Scheduling

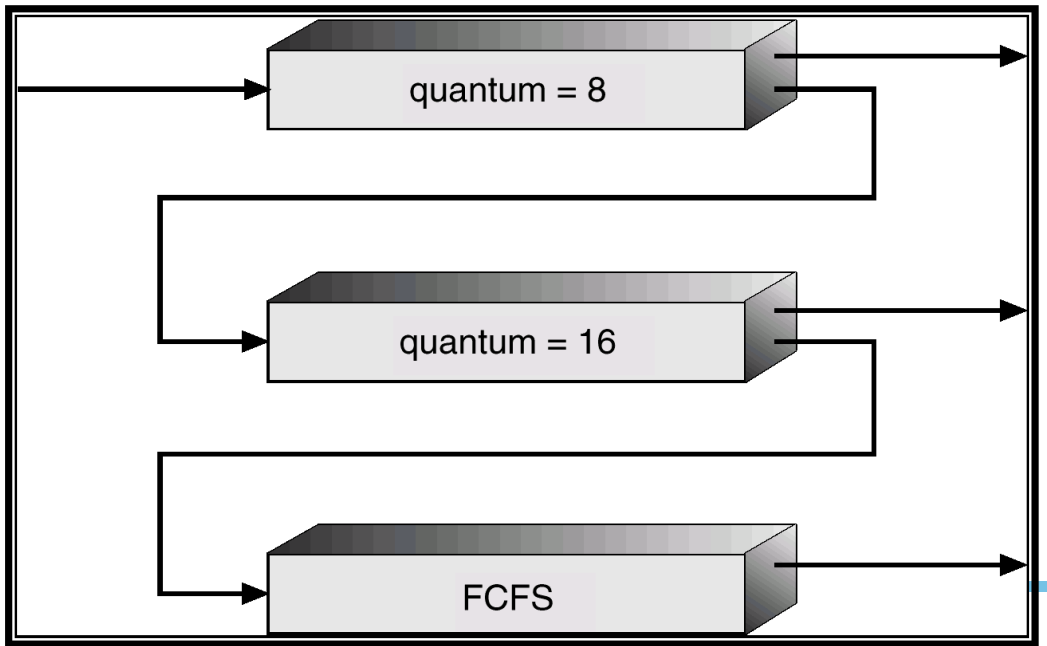


Multilevel Feedback Queue

- Disadvantage of Multilevel queue : Inflexible
- 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 - time quantum 8 ms
  - Q_1 – RR- time quantum 16ms
  - Q_2 – FCFS



- Scheduling
  - A new process enters queue Q_0. When it gains CPU, process receives 8 milliseconds. If it does not finish in 8 milliseconds, process is moved to queue Q_1.
  - At Q_1 process receives 16 additional milliseconds. If it still does not complete, it is preempted and moved to queue Q_2.