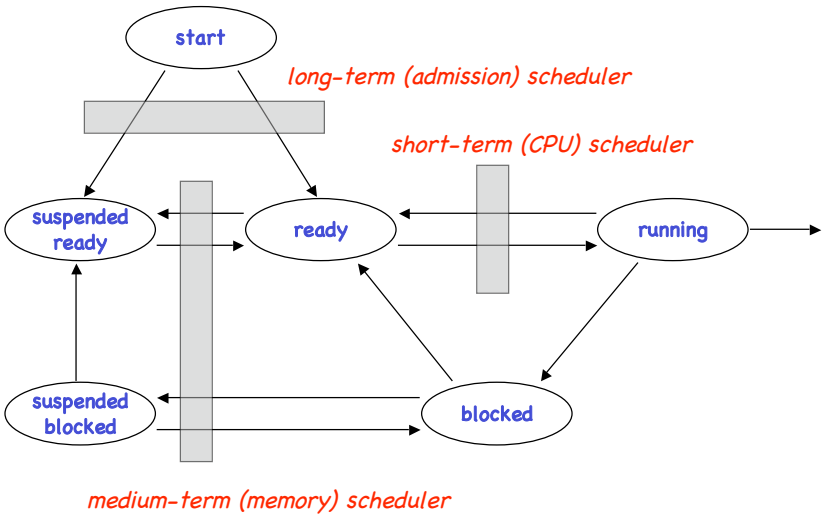
CPU Scheduling

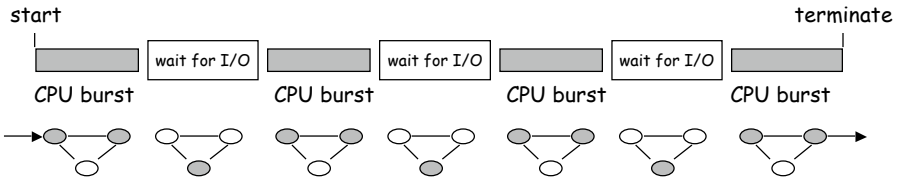
- Schedulers in the OS
- Structure of a CPU Scheduler
  - Scheduling = Selection + Dispatching
- Criteria for scheduling
- Scheduling Algorithms
  - FIFO/FCFS
  - SPF / SRTF
  - Priority - Based

Schedulers



Focus: Short-Term Scheduling

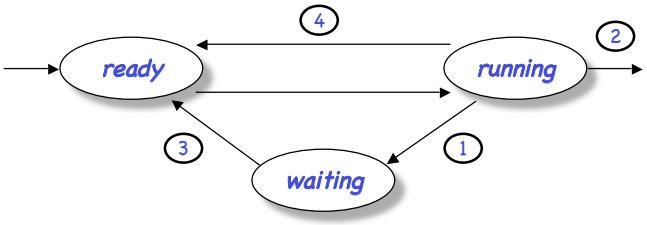
- Recall: Motivation for multiprogramming -- have multiple processes in memory to keep CPU busy.
- Typical execution profile of a process/thread:



- CPU scheduler is managing the execution of CPU bursts, represented by processes in ready or running state.

Scheduling Decisions

"Who is going to use the CPU next?!"



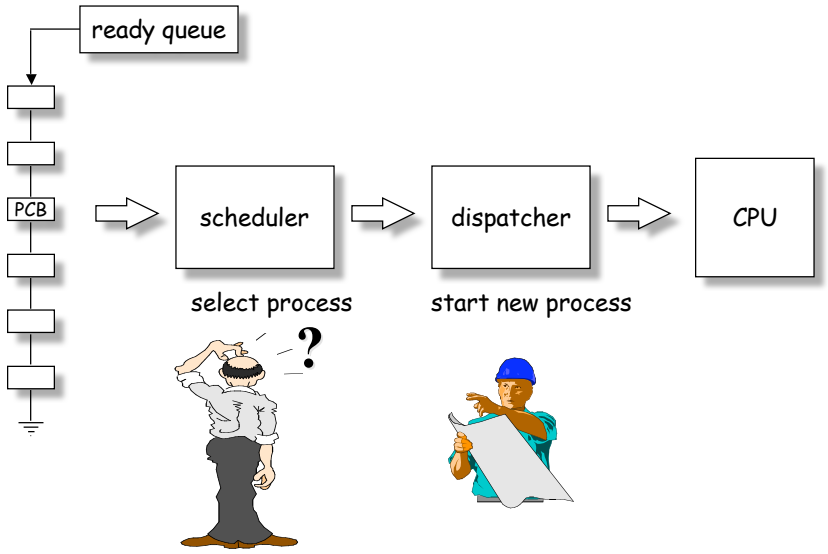
non-preemptive

Scheduling decision points:

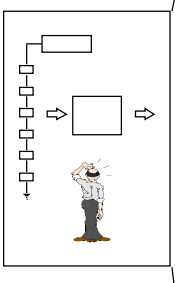
- 1. The running process changes from *running* to *waiting* (current CPU burst of that process is over).
- 2. The running process terminates.
- 3. A waiting process becomes *ready* (new CPU burst of that process begins).
- 4. The current process switches from *running* to *ready* .

preemptive

Structure of a Scheduler

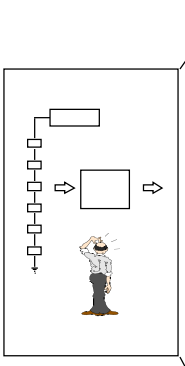


What Is a Good Scheduler? Criteria



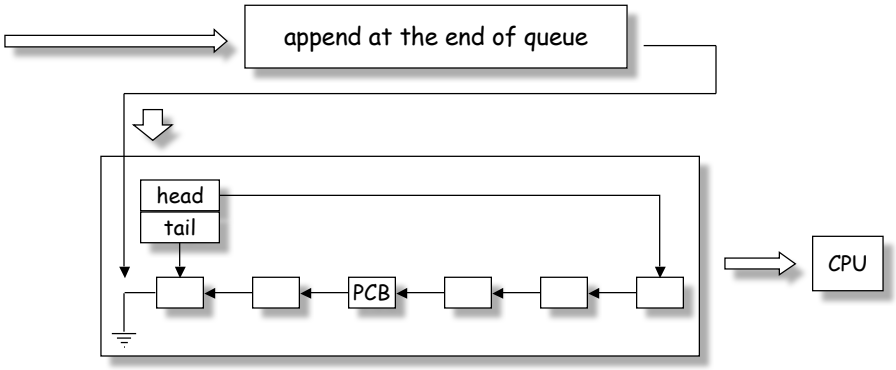
- User oriented:
  - **Turnaround time** : time interval from submission of job until its completion
  - **Waiting time** : sum of periods spent waiting in ready queue
  - **Response time** : time interval from submission of job to first response
  - **Normalized turnaround time**: ratio of turnaround time to service time
- System oriented:
  - **CPU utilization** : percentage of time CPU is busy
  - **Throughput** : number of jobs completed per time unit
- Any good scheduler should:
  - *maximize* CPU utilization and throughput
  - *minimize* turnaround time, waiting time, response time
- Huh?
  - maximum/minimum values *vs.* average values *vs.* variance

Scheduling Algorithms



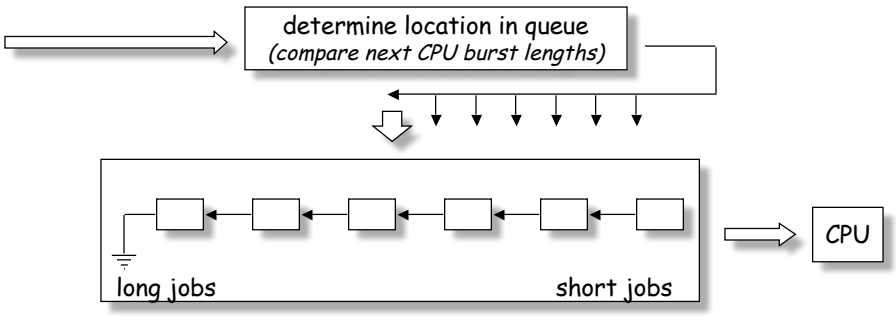
- FCFS : First-come-first-served
- SPN: Shortest Process Next
- SRT: Shortest Remaining Time
- priority scheduling
- RR : Round-robin
- MLFQ: Multilevel feedback queue scheduling
- Multiprocessor scheduling

First-Come-First-Served (FCFS/FIFO)



- Advantages:
  - very simple
- Disadvantages:
  - long average and worst-case waiting times
  - poor dynamic behavior (convoy effect)

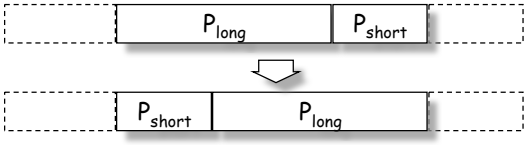
Shortest Process Next



- Whenever CPU is idle, picks process with shortest next *CPU burst*.
- Advantages: minimizes average waiting times.
- Problem: How to determine length of next CPU burst?!
- Problem: Starvation of jobs with long CPU bursts.

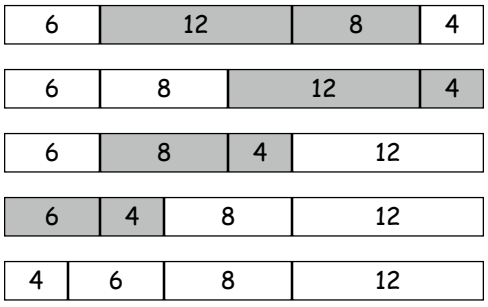
SJF Minimizes Average Waiting Time

- Provably optimal: Proof: swapping of jobs



$dW = t_{short} - t_{long} < 0$

- Example:



W = 6+18+26 = 50

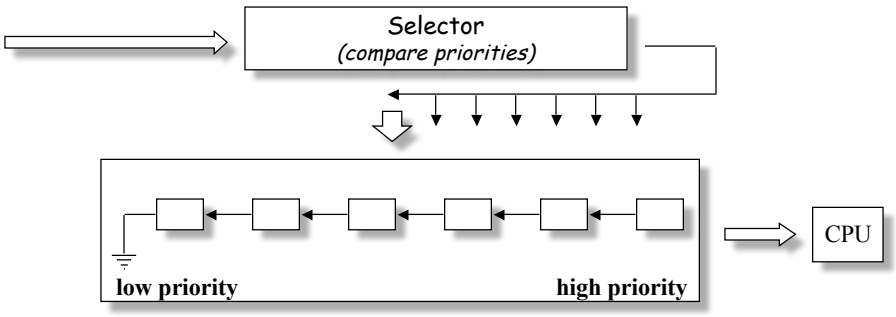
W = 6+14+26 = 46

W = 6+14+18 = 38

W = 6+10+18 = 34

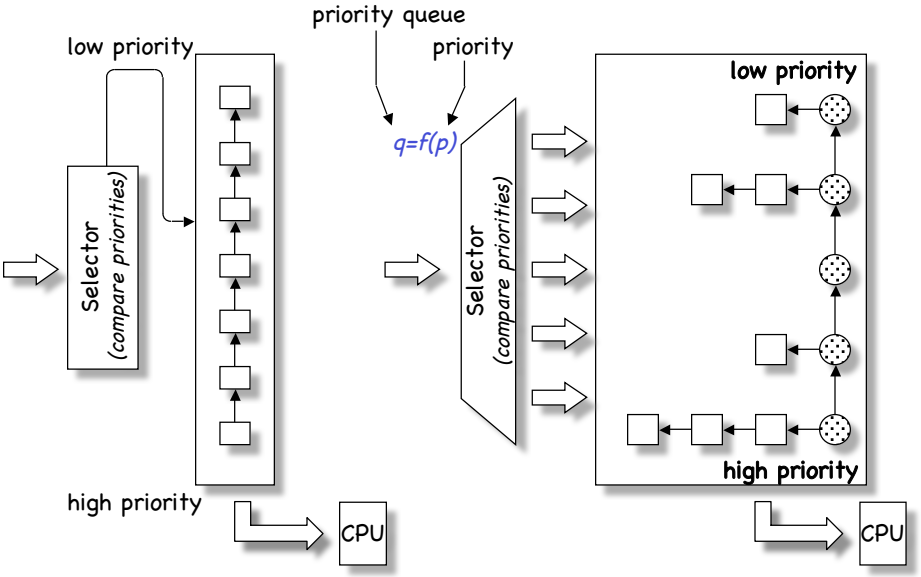
W = 4+10+18 = 32

(Fixed) Priority Scheduling

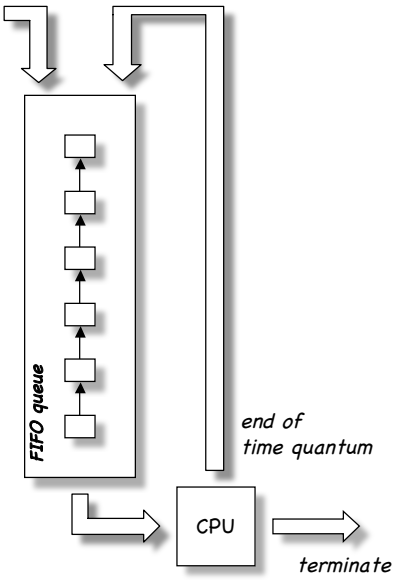


- Whenever CPU is idle, picks process with highest priority.
- Priority:
  - process class, urgency, pocket depth.
- Unbounded blocking: Starvation
  - Increase priority over time: aging

- **Conceptually**
- **Priority Queues**



Round-Robin



- FIFO with preemption after time quantum
- Method for time sharing
- Choice of time quantum:
  - large: FCFS
  - small: Processor sharing
- Time quantum also defines context-switching overhead

Multilevel Feedback Queue Scheduling

(conceptually)