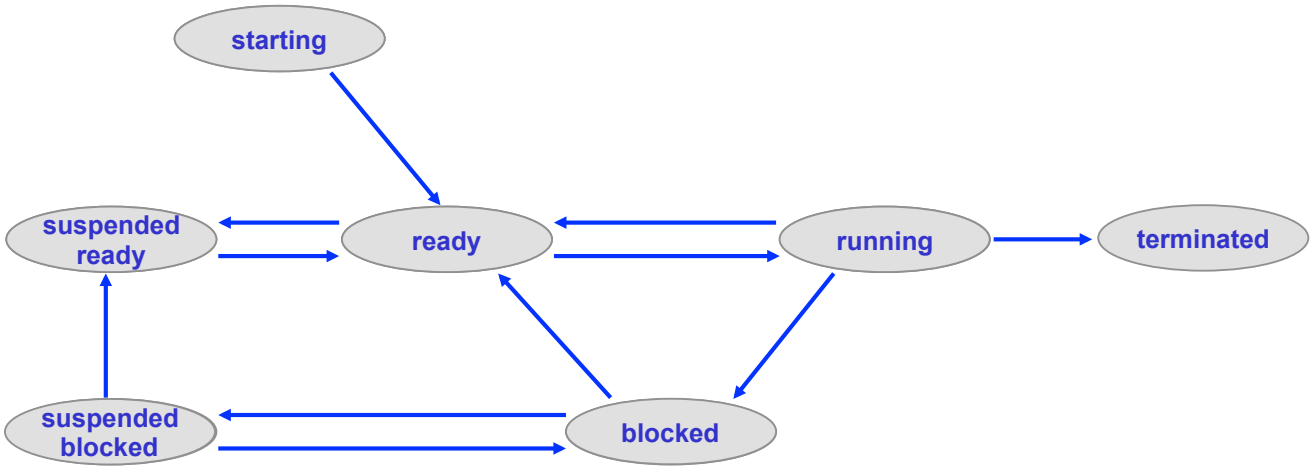
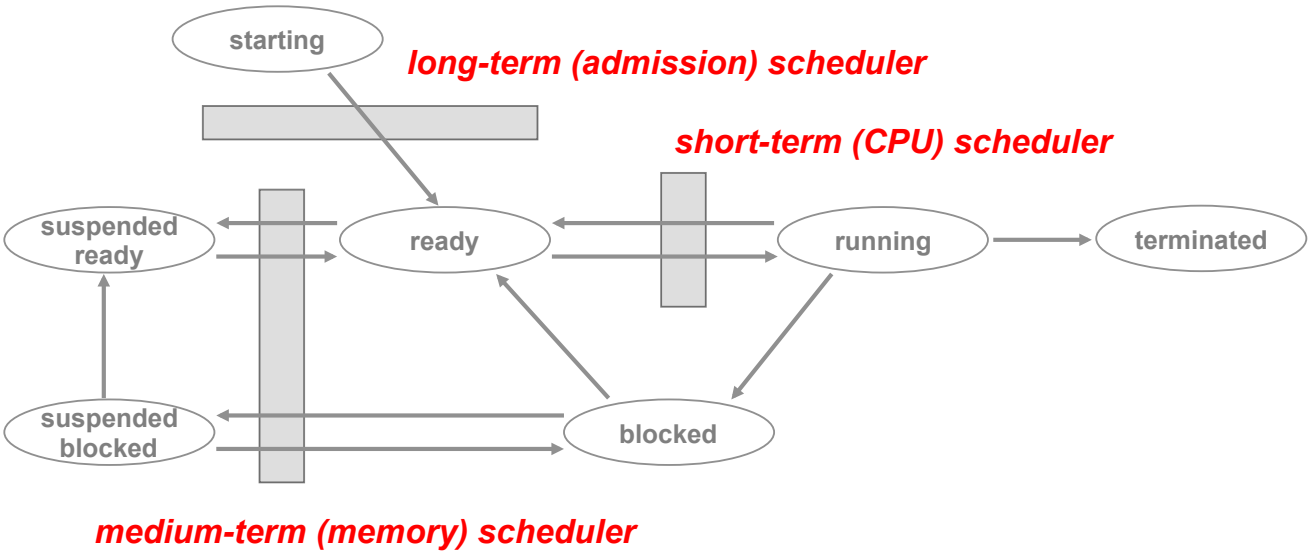
Concurrency and Threading: CPU Scheduling

- Execution State of a Thread or Process
- Schedulers in the OS
- Structure of CPU Schedulers
- Criteria for Scheduling
- Scheduling Algorithms: FIFO, SJF, FP, MLFQ
- Multiprocessor Scheduling

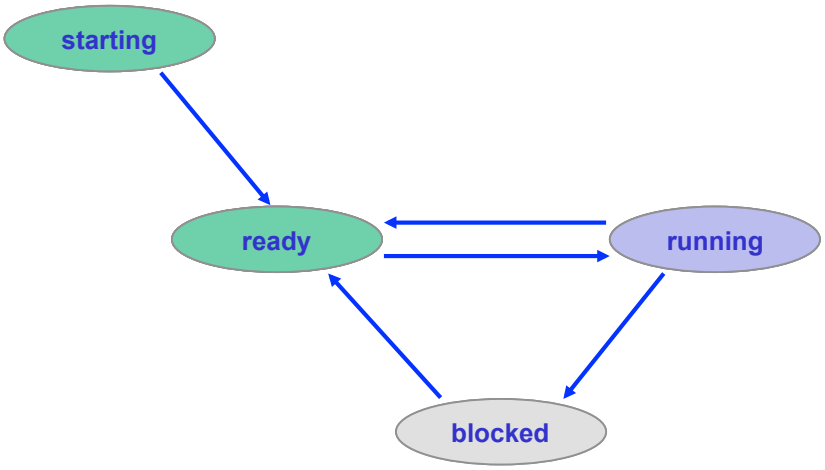
State of a Thread/Process



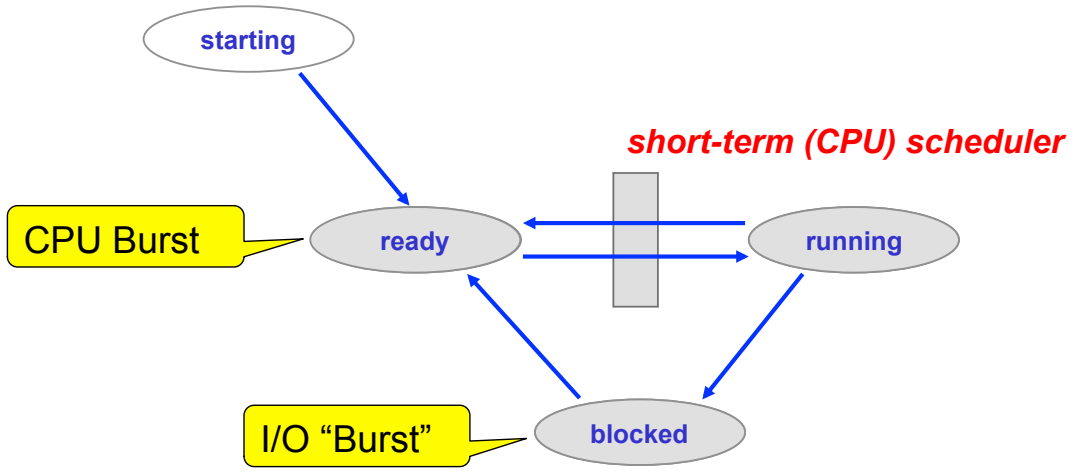
Schedulers



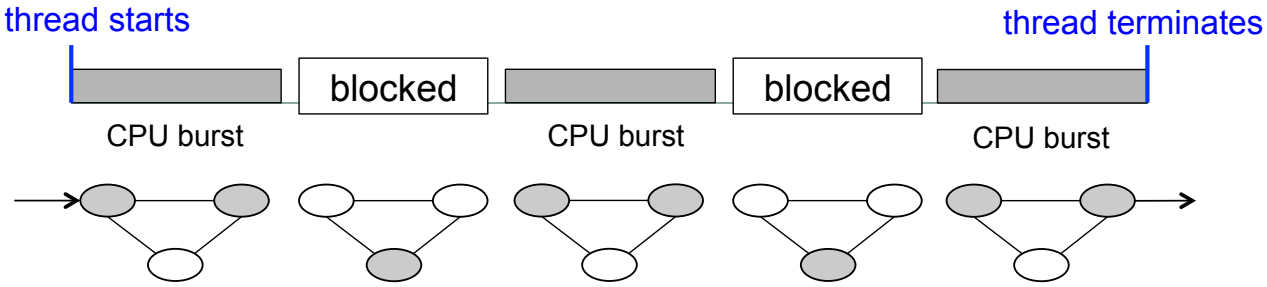
Multiprogramming



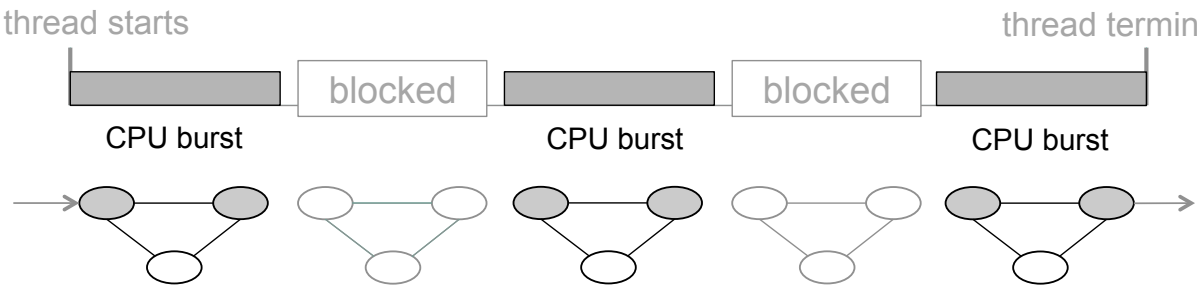
CPU Bursts vs. I/O Bursts



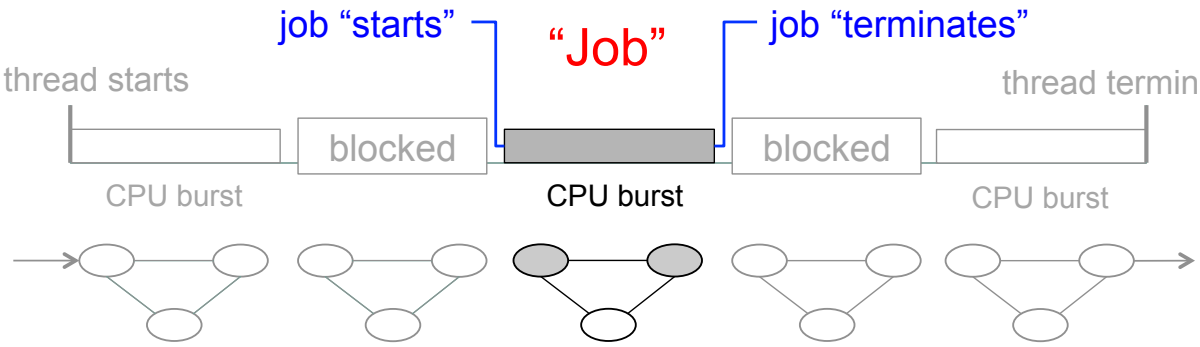
CPU Bursts and Jobs



CPU Bursts and Jobs

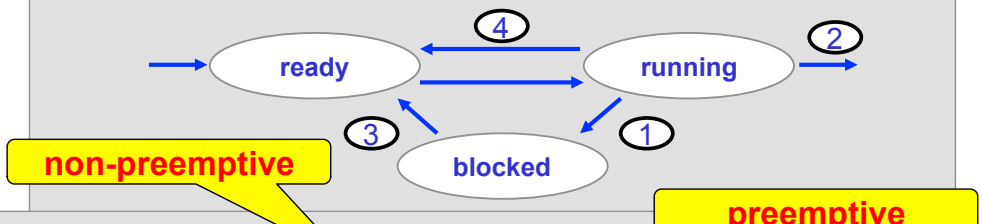


CPU Bursts and Jobs



Scheduling Decisions

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



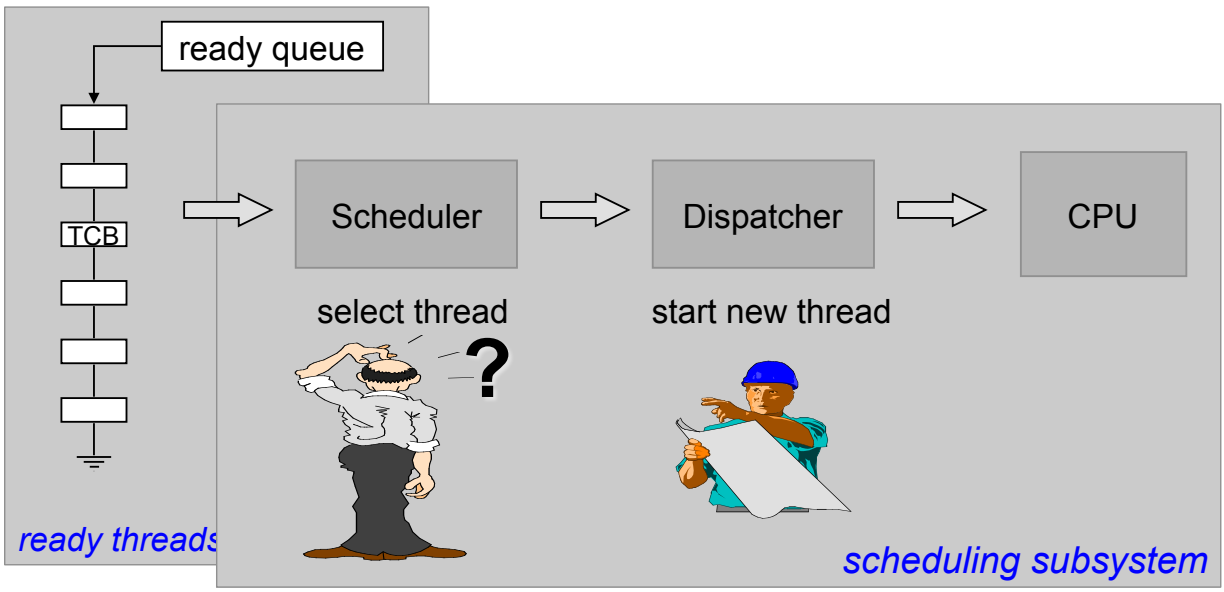
non-preemptive

preemptive

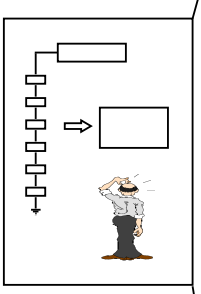
Scheduling **decision points**:

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

Structure of a Scheduling Subsystem



What Is a Good Scheduler? Criteria



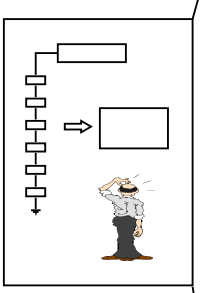
User oriented:

- **Response time** : time interval from start of job to first response
- **Normalized response time**: ratio of response time to execution time
- **Waiting time** : sum of periods spent waiting in ready queue

System oriented:

- **CPU utilization** : percentage of time CPU is busy
- **Throughput** : number of jobs completed per time unit

What Is a Good Scheduler? Criteria



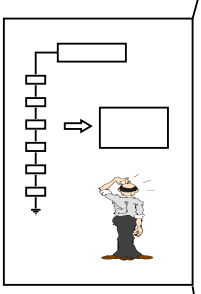
Any good scheduler should:

- *maximize* CPU utilization and throughput
- *minimize* waiting time, response time

Huh?

- maximum/minimum values?
- average values?
- variance?

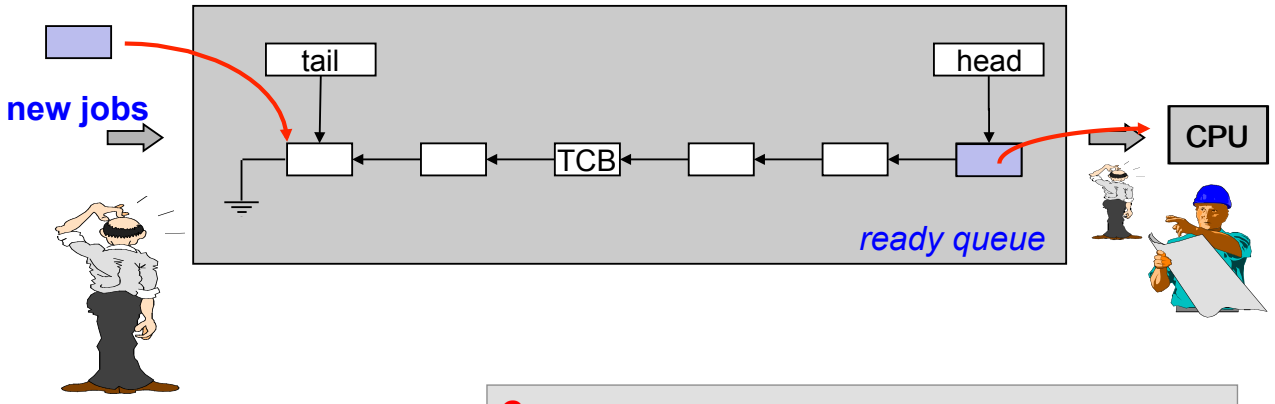
Scheduling Algorithms



- **FCFS** : First-come-first-served
- **SPN**: Shortest Process Next
- **Priority** scheduling
- **RR** : Round-Robin
- **MLFQ**: Multilevel Feedback Queue Scheduling
- **Multiprocessor** scheduling

Common Structure of a Scheduler

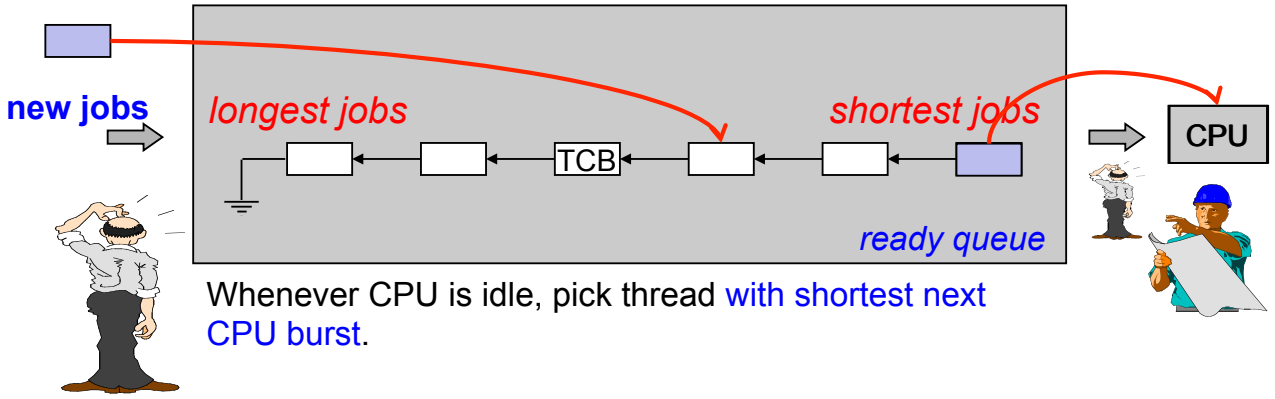
First-Come-First-Served (FCFS/FIFO)



Pros:
- very simple

Cons:
- long average and worst-case **waiting times**
- poor dynamic behavior (**convoy effect**)

Shortest-Job-First



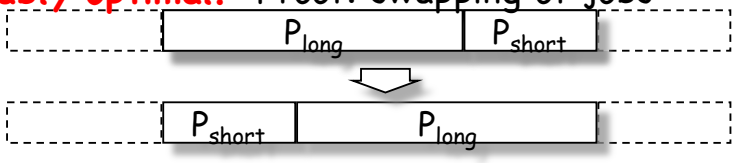
Whenever CPU is idle, pick thread with shortest next CPU burst.

Pros:
- minimizes average waiting times

Cons:
- **Starvation** of threads with long CPU bursts.
- How to determine length of next CPU burst?!

SJF Minimizes Average Waiting Time

Provably optimal! Proof: swapping of jobs

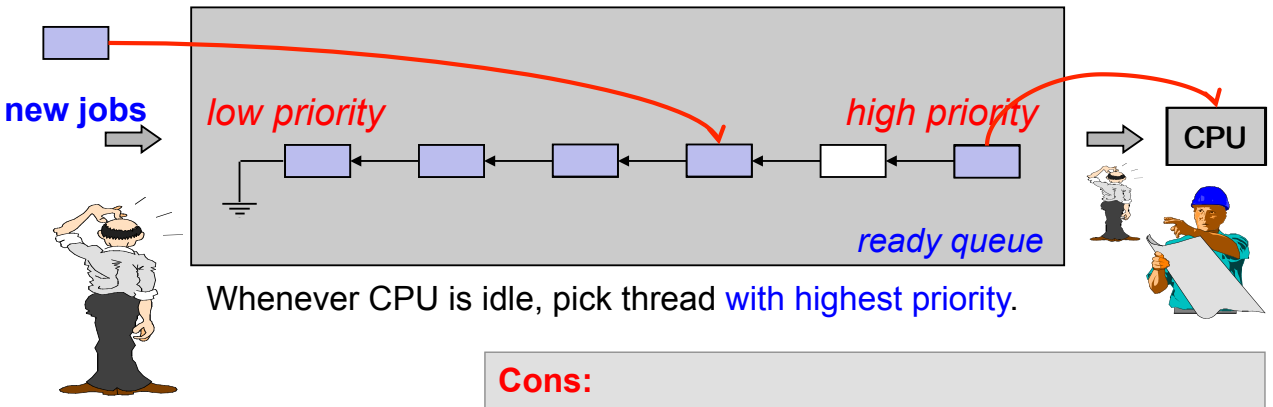


$dW = t_{short} - t_{long} < 0$

Example:

6	12	8	4	W = 6+18+26 = 50
6	8	12	4	W = 6+14+26 = 46
6	8	4	12	W = 6+14+18 = 38
6	4	8	12	W = 6+10+18 = 34
4	6	8	12	W = 4+10+18 = 32

Fixed-Priority Scheduler

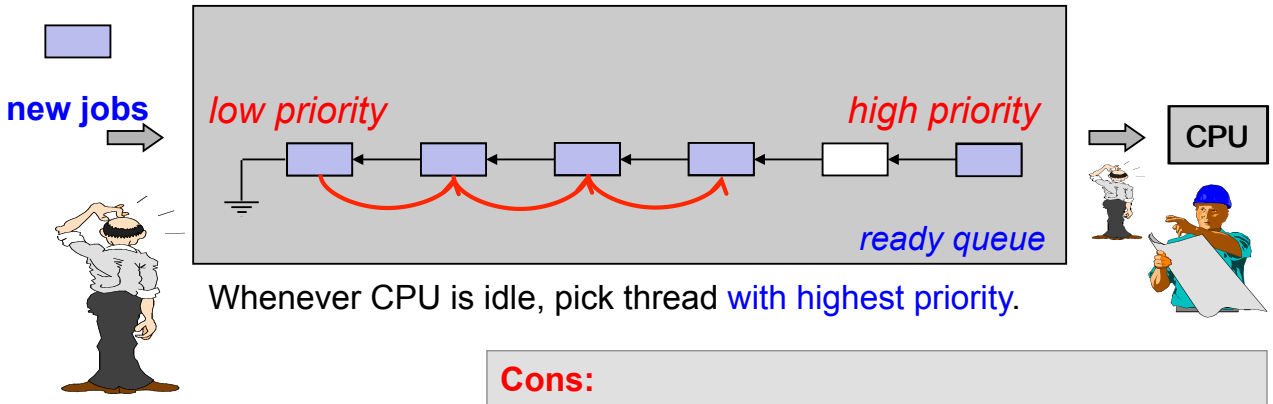


Whenever CPU is idle, pick thread with highest priority.

Priority:
- thread/process class
- urgency

Cons:
- **Starvation** of low-priority threads

Fixed-Priority Scheduler



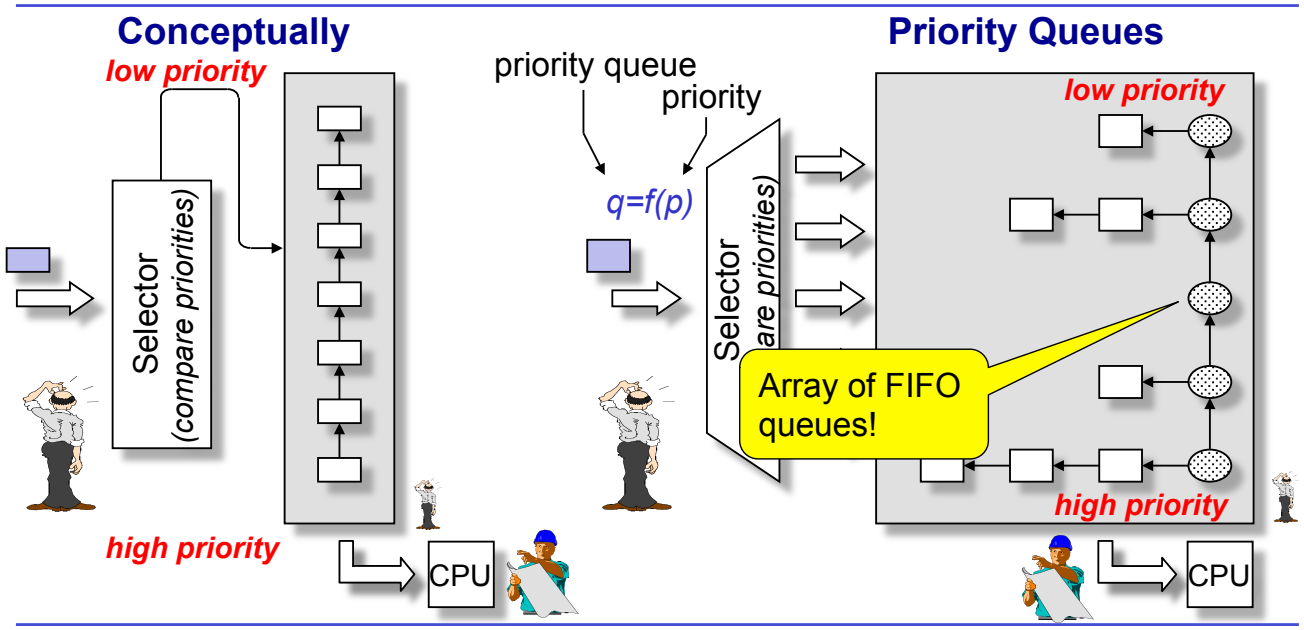
Whenever CPU is idle, pick thread with highest priority.

Priority:
- thread/process class
- urgency

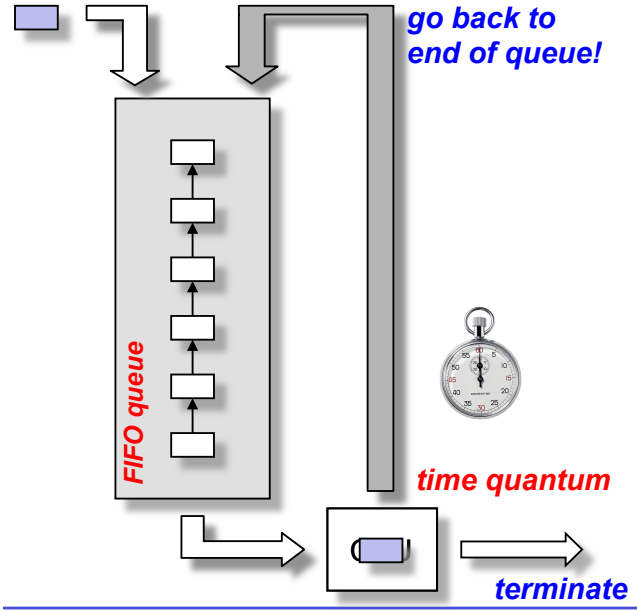
Cons:
- **Starstarvation** of low-priority threads

Solution:
- **Aging**: increase priority over time

Fixed Priority Scheduling (implementation)



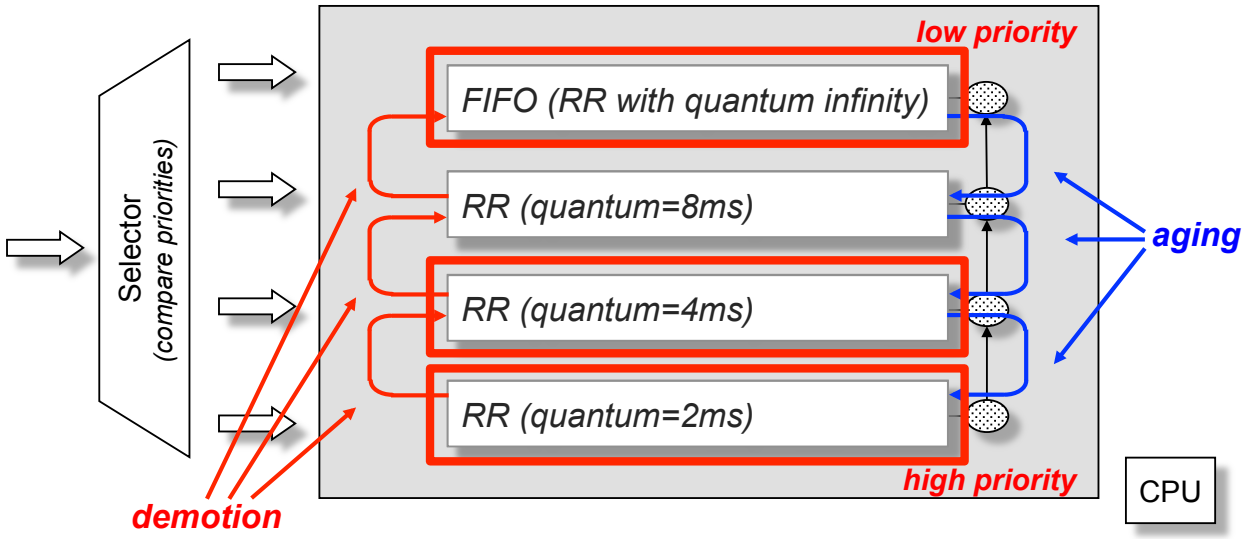
Round-Robin



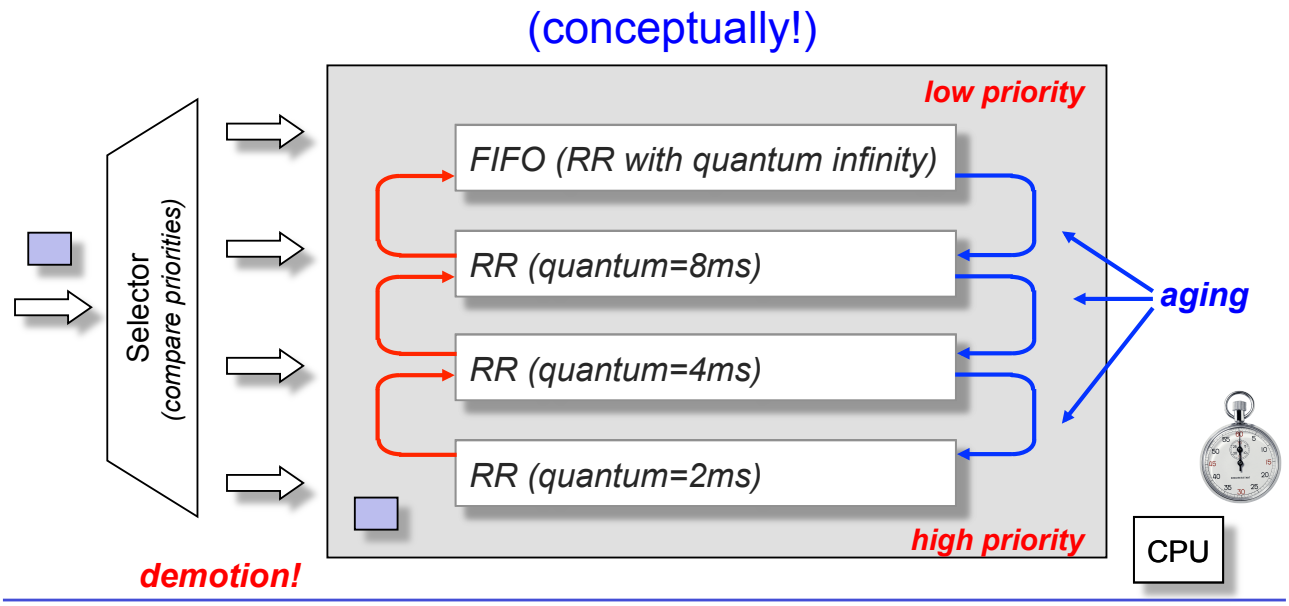
- FIFO with preemption after time quantum
- Popular method for time sharing
- Choice of time quantum:
  - large: FCFS
  - small: "Processor sharing"
- Time quantum also defines context-switching overhead

Multilevel Feedback Queue Scheduling

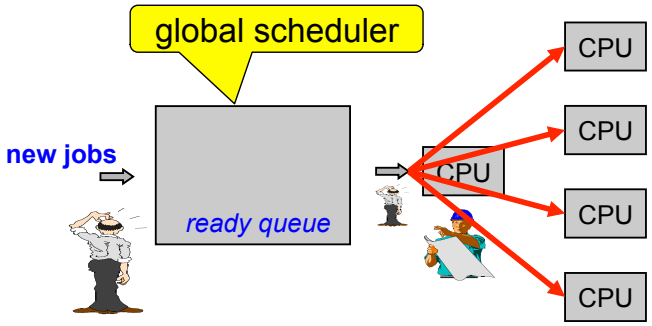
(conceptually!)



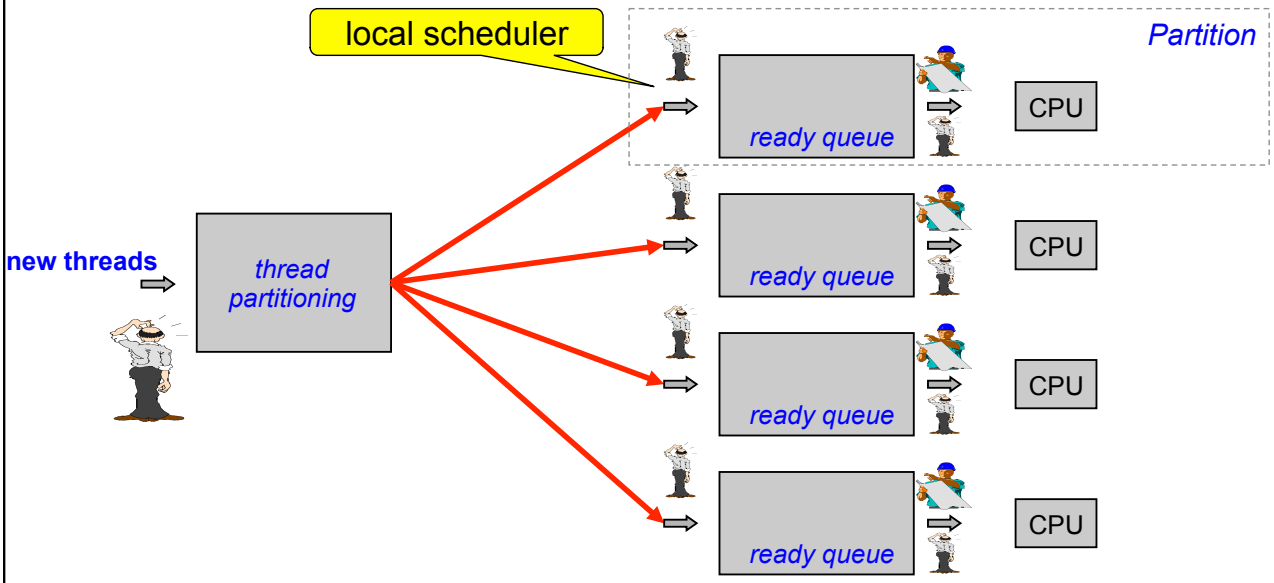
Multilevel Feedback Queue Scheduling



Dynamic Multiprocessor Scheduling



Static Multiprocessor Scheduling



CPU Scheduling: Summary

- Execution State of a Thread or Process
- Schedulers in the OS
- Structure of Schedulers
- Criteria for Scheduling
- Scheduling Algorithms: FIFO, SJF, FP, MLFQ
- Multiprocessor Scheduling