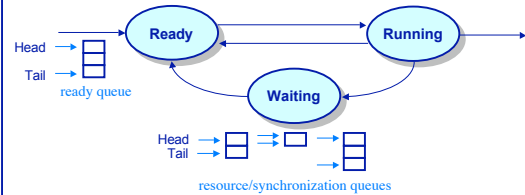


## Process Scheduling

### Processes and State Transitions



- ◆ Three states: Ready, Running, and Waiting

When a process makes a transition:

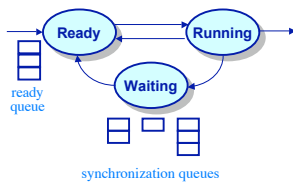
1. from *running* to *waiting*
2. from *running* to *ready*
3. from *waiting* to *ready*  
(3a. a process is created)
4. from *running* to *terminated*

Why a process makes a transition:

1. an action of the *process*  
*non-preemptive scheduling*
2. occurrence of an *external event*  
*preemptive scheduling*

### Process Scheduling

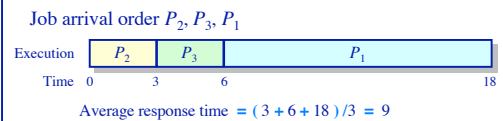
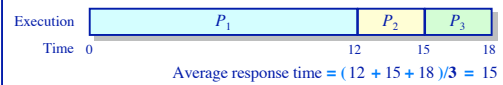
- ◆ Process scheduling
  - Select a process from ready queue for execution
- ◆ Evaluation metrics
  - CPU/device utilization
  - System throughput
  - Waiting time
  - Response time



### Scheduling Policies

#### First-Come-First-Served (FCFS)

- ◆ The discipline corresponding to FIFO queuing
- ◆ Example — 3 processes w/ compute times 12, 3, and 3
  - Job arrival order  $P_1, P_2, P_3$



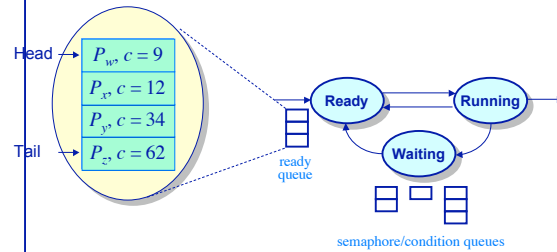
## FCFS Scheduling (Cont'd.)

- ◆ Advantage:
  - Simple
- ◆ Disadvantages:
  - Average waiting time is highly variable
    - ◆ Short jobs may wait behind long ones !!
  - May lead to poor overlap between I/O and CPU processing
    - ◆ CPU bound processes will make I/O bound processes to wait ⇒ I/O devices remain idle

5

## Scheduling Policies Shortest-Job-First (SJF)

- ◆ Select the shortest job first
  - Enqueue jobs in order of estimated completion time

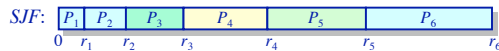


6

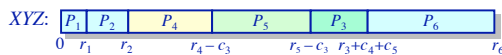
## Shortest-Job-First Scheduling An optimal policy for minimizing response times

- ◆ Intuition: Consider an SJF execution of a set of processes

Average response time =  $(r_1 + r_2 + r_3 + r_4 + r_5 + r_6)/6$



Can switching the execution order reduce response time?



$$\begin{aligned} \text{Average response time} &= (r_1 + r_2 + r_4 - c_3 + r_5 - c_3 + r_4 + c_4 + c_5 + r_6)/6 \\ &= (r_1 + r_2 + r_3 + r_4 + r_5 + r_6 + (c_4 + c_5 - 2c_3))/6 \end{aligned}$$

7

## SJF Scheduling --- The Catch

- ◆ It's unfair !!
  - Continuous stream of short jobs will starve long jobs
- ◆ Needs clairvoyance
  - Need to know the execution time of a process
  - Simple solution: ask the user !
  - Yeah, right !!
- ◆ So, what if you don't subscribe to the Psychic Network ??

8

### Short-Job-First Scheduling

Estimating execution time

- Jobs are enqueued in order of estimated completion time
  - "Recent history is a good indicator of the near future"

```

process P
begin
loop
  <read input from user>
  <process input>
end loop
end P
    
```

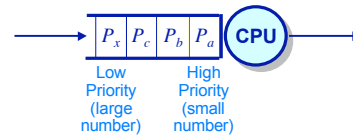
$t_n$  — duration of the  $n^{\text{th}}$  CPU burst  
 $\tau_{n+1}$  — predicted duration of the  $n+1^{\text{st}}$  CPU burst  
 $\tau_{n+1} = \alpha t_n + (1-\alpha)\tau_n$  for  $0 \leq \alpha \leq 1$

9

### Scheduling Policies

Priority Scheduling (PS)

- Assign a priority (a number) to each job and schedule jobs in order of priority
  - Typically low priority values = "high priority"
  - E.g., if priority =  $\tau_n$ , then a priority scheduler becomes a SJF scheduler.

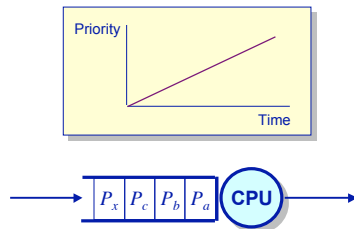


10

### Priority Scheduling

Avoiding starvation

- Aging
  - Gradually increase a process's priority (decrease its priority value) over time



11

### Non Pre-emptive vs. Pre-emptive Scheduling

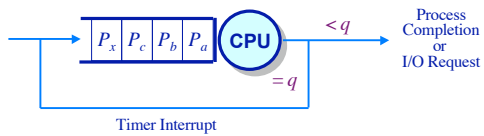
- Non Pre-emptive Scheduling:
  - Once a process begins execution, it occupies CPU until it finishes or it blocks
  - Advantage: simplicity, but ...
  - Creates problems ... (like what?)
  - Examples: FCFS, SJF, PS, ...
- Pre-emptive Scheduling:
  - A process is switched back and forth between running and ready states
  - Advantage: more efficient, better capabilities, but ...
  - More complex and needs hardware support (e.g., timer interrupts)
  - Examples: Round Robin, Shortest Remaining Time First (SRTF), Multi-level Feedback Queue (MLF)

12

### Scheduling Policies

#### Round-Robin Scheduling (RR)

- ◆ Allocate the processor in discrete unit called *quanta* (or *time-slices*)
- ◆ Switch to the next ready process at the end of each quantum
  - Processes execute every  $(n - 1)q$  time units



13

### RR Scheduling: Selecting a Time Quantum

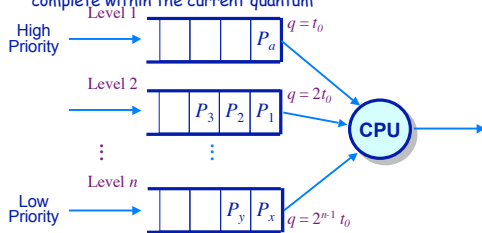
- ◆ Too large
  - Long waiting time
  - Degenerates to FCFS in the limit
- ◆ Too small
  - Responsive, but ...
  - Throughput suffers due to large context switch overhead
- ◆ Goal:
  - Select a time quantum that balances this tradeoff
  - Rule of thumb: maintain context switch overhead to less than 1%

14

### Scheduling Policies

#### Multi-level feedback queues (MLF)

- ◆  $n$  priority levels — priority scheduling between levels, round-robin within a level
- ◆ Quantum size decreases with priority level
- ◆ Jobs are demoted to lower priority levels if they don't complete within the current quantum



15