

CSC 4103 - Operating Systems
Fall 2009

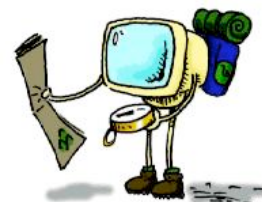
LECTURE - V
CPU SCHEDULING - I

Tevfik Koşar

Louisiana State University
September 10th, 2009

Roadmap

- CPU Scheduling
 - Basic Concepts
 - Scheduling Criteria & Metrics
 - Different Scheduling Algorithms
 - FCFS
 - SJF
 - Priority
 - RR

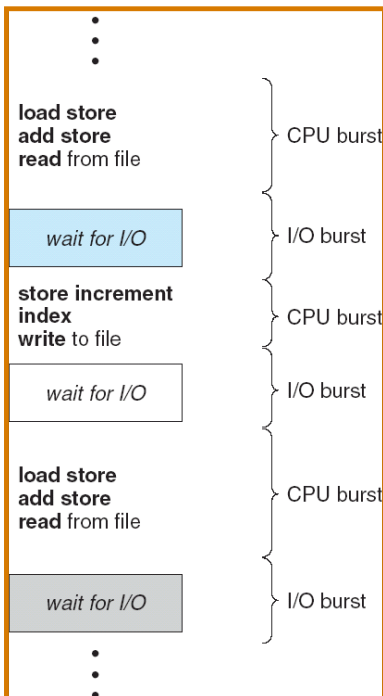


Basic Concepts

- Multiprogramming is needed for efficient CPU utilization
- CPU Scheduling: deciding which processes to execute when
- Process execution begins with a **CPU burst**, followed by an **I/O burst**
- CPU-I/O Burst Cycle - Process execution consists of a cycle of CPU execution and I/O wait

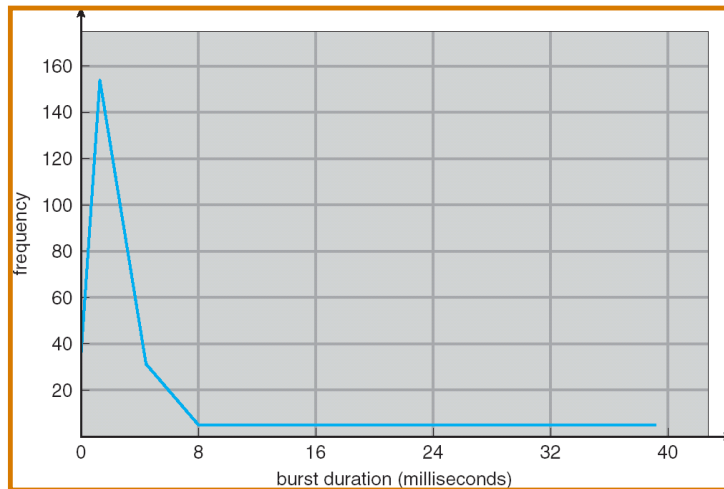
3

Alternating Sequence of CPU And I/O Bursts



4

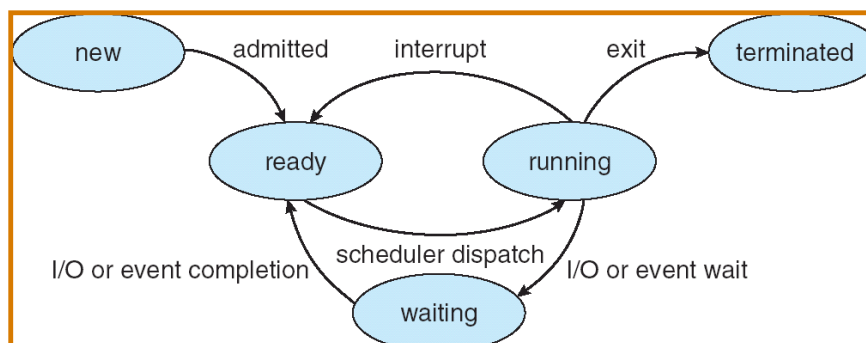
Histogram of CPU-burst Durations



5

Process State

- As a process executes, it changes *state*
 - **new**: The process is being created
 - **ready**: The process is waiting to be assigned to a process
 - **running**: Instructions are being executed
 - **waiting**: The process is waiting for some event to occur
 - **terminated**: The process has finished execution



6

CPU Scheduler

- Selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them
 - short-term scheduler
- 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
- Scheduling under 1 and 4 is *nonpreemptive/cooperative*
 - Once a process gets the CPU, keeps it until termination/switching to waiting state/release of the CPU
- All other scheduling is *preemptive*
 - Most OS use this
 - Cost associated with access to shared data
 - i.e. time quota expires

7

Dispatcher

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

8

Scheduling Criteria

- **CPU utilization** - keep the CPU as busy as possible --> **maximize**
- **Throughput** - # of processes that complete their execution per time unit --> **maximize**
- **Turnaround time** - amount of time passed to finish execution of a particular process --> **minimize**
 - i.e. execution time + waiting time
- **Waiting time** - total amount of time a process has been waiting in the ready queue --> **minimize**
- **Response time** - amount of time it takes from when a request was submitted until the first response is produced, **not** output (for time-sharing environment) --> **minimize**

9

Optimization Criteria

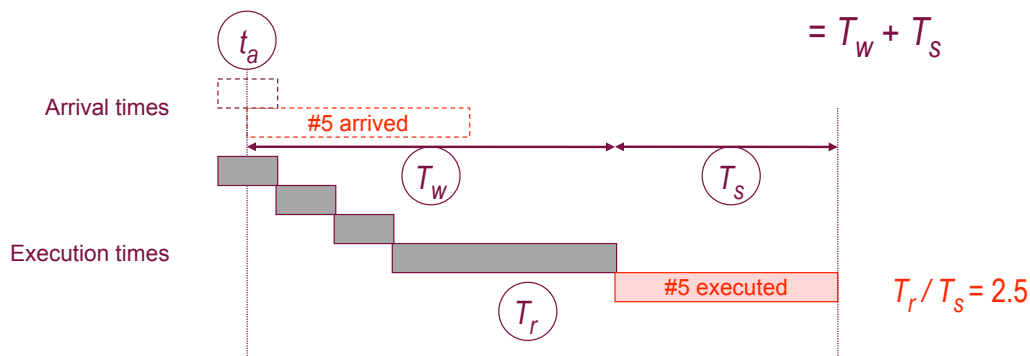
- Maximize CPU utilization
- Maximize throughput
- Minimize turnaround time
- Minimize waiting time
- Minimize response time

10

Scheduling Metrics

➤ Scheduling metrics

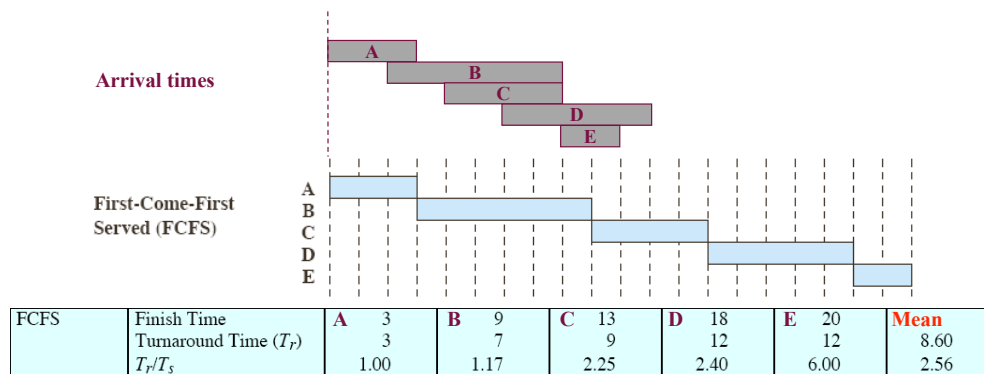
- ✓ arrival time t_a = time the process became “Ready” (again)
- ✓ wait time T_w = time spent waiting for the CPU
- ✓ service time T_s = time spent executing in the CPU
- ✓ turnaround time T_r = total time spent waiting and executing



11

First-Come, First-Served (FCFS) Scheduling

- ✓ processes are assigned the CPU in the order they request it
- ✓ when the running process blocks, the first “Ready” is run next
- ✓ when a process gets “Ready”, it is put at the end of the queue



FCFS scheduling policy

Stallings, W. (2004) Operating Systems: Internals and Design Principles (5th Edition).

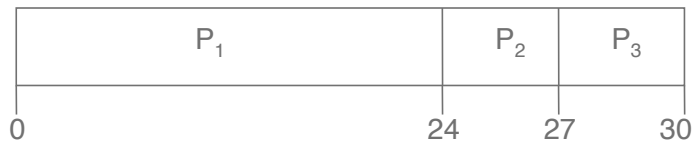
12

FCFS Scheduling - Example

Process	Burst Time
P_1	24
P_2	3
P_3	3

- Suppose that the processes arrive in the order: P_1 , P_2 , P_3

The **Gantt Chart** for the schedule is:



- Waiting time for $P_1 = 0$; $P_2 = 24$; $P_3 = 27$
- Average waiting time: $(0 + 24 + 27)/3 = 17$

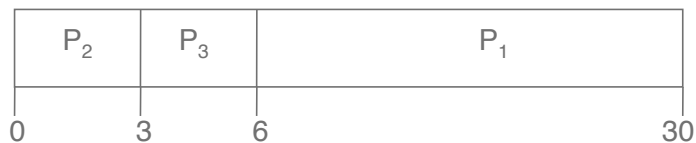
13

FCFS Scheduling - Example

Suppose that the processes arrive in the order

P_2 , P_3 , P_1

- The Gantt chart for the schedule is:



- Waiting time for $P_1 = 6$; $P_2 = 0$; $P_3 = 3$
- Average waiting time: $(6 + 0 + 3)/3 = 3$
- Much better than previous case
- Convoy effect** short process behind long process

14

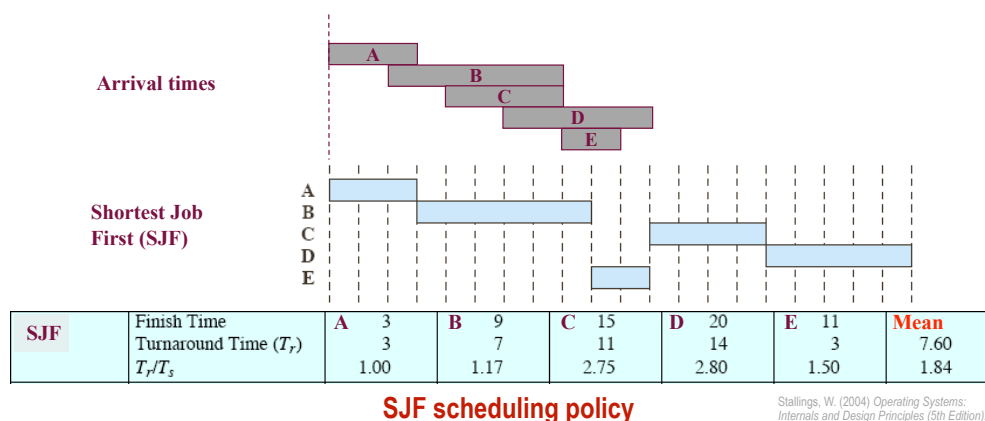
Shortest-Job-First (SJF) Scheduling

- Associate with each process the length of its next CPU burst. Use these lengths to schedule the process with the shortest time
- 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

15

Non-Preemptive SJF

- ✓ nonpreemptive, assumes the run times are known in advance
- ✓ among several equally important "Ready" jobs (or CPU bursts), the scheduler picks the one that will finish the earliest

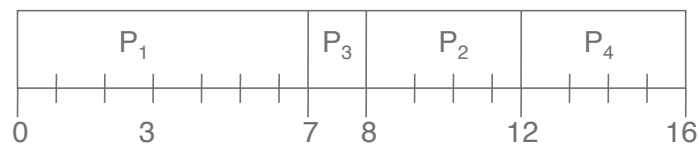


16

Non-Preemptive SJF - Example

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (non-preemptive) **Gantt Chart**



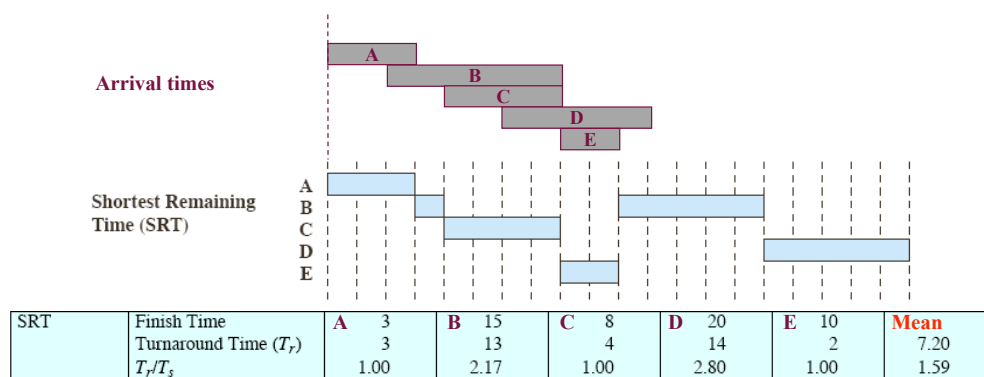
- Average waiting time = $(0 + 6 + 3 + 7)/4 = 4$

17

Preemptive SJF (SRT)

➤ Shortest Remaining Time (SRT)

- ✓ preemptive version of SJF, also assumes known run time
- ✓ choose the process whose remaining run time is shortest
- ✓ allows new short jobs to get good service



SRT scheduling policy

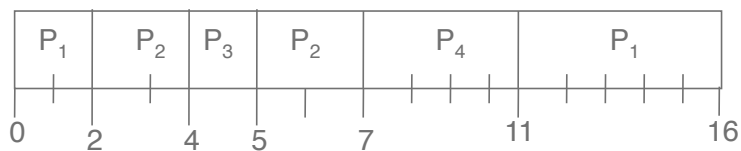
Stallings, W. (2004) Operating Systems: Internals and Design Principles (5th Edition).

18

Example of Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (preemptive) **Gantt Chart**



19

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 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

20

Example of Priority

	<u>Process</u>	<u>Arrival Time</u>	<u>Burst Time</u>	<u>Priority</u>
-	P_1	0.0	7	2
	P_2	2.0	4	1
	P_3	4.0	1	4
	P_4	5.0	4	3

- Priority (non-preemptive)
 - $P_1 \rightarrow P_2 \rightarrow P_4 \rightarrow P_3$
- Priority (preemptive)
 - ??

21

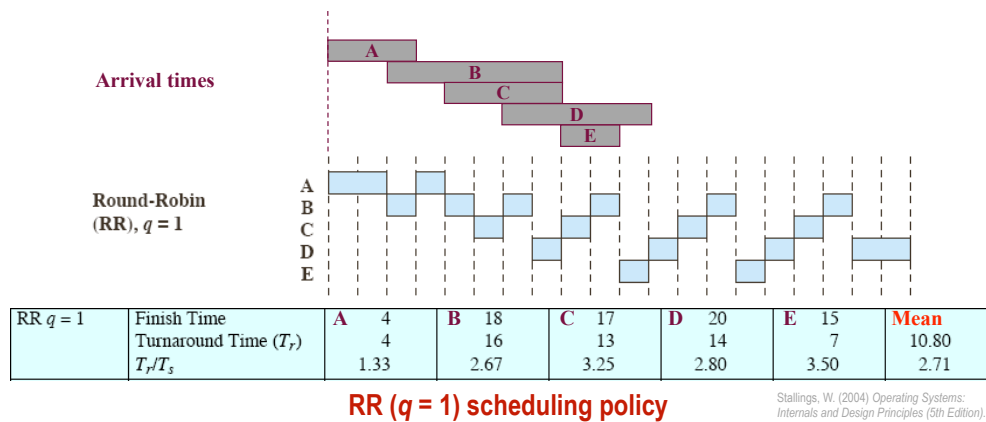
Round Robin (RR)

- 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 FIFO

22

Round Robin (RR)

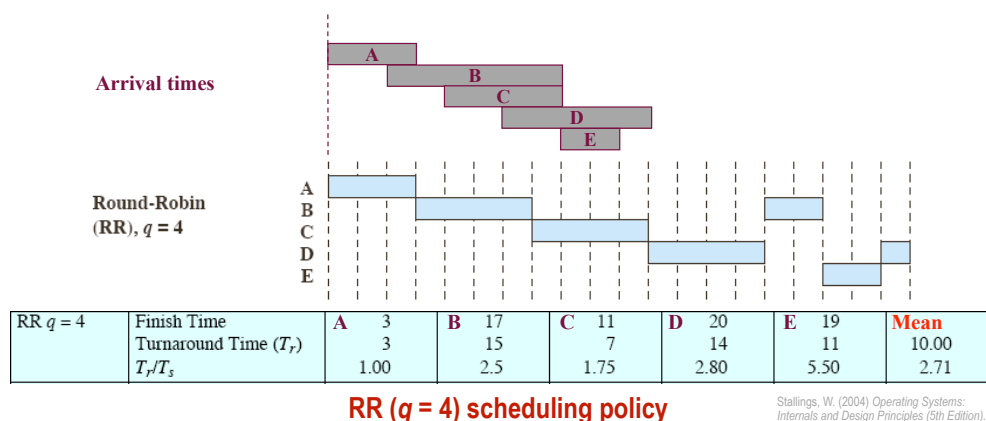
- ✓ preemptive FCFS, based on a timeout interval, the **quantum** q
- ✓ the running process is interrupted by the clock and put last in a FIFO “Ready” queue; then, the first “Ready” process is run instead



23

Round Robin (RR)

- ✓ a crucial parameter is the quantum q (generally ~10–100ms)
 - q should be big compared to context switch latency (~10 μ s)
 - q should be less than the longest CPU bursts, otherwise RR degenerates to FCFS



24

Example of RR with Time Quantum = 20

Process	Burst Time
P_1	53
P_2	17
P_3	68
P_4	24

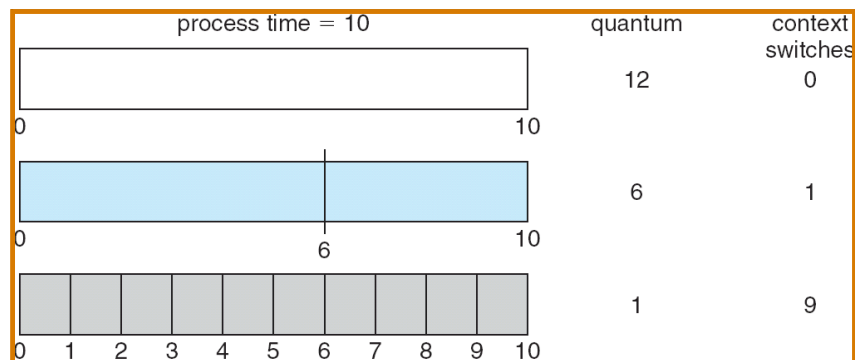
- For $q=20$, the **Gantt chart** is:

P ₁	P ₂	P ₃	P ₄	P ₁	P ₃	P ₄	P ₁	P ₃	P ₃	
0	20	37	57	77	97	117	121	134	154	162

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

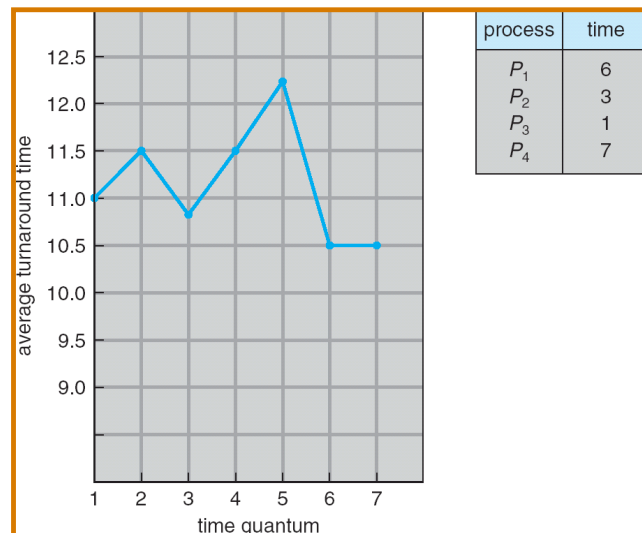
25

Time Quantum and Context Switch Time



26

Turnaround Time Varies With The Time Quantum



27

Exercise

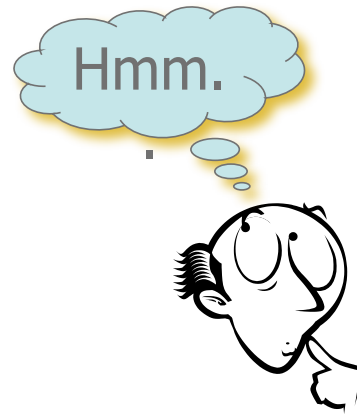
Process ID	Arrival Time	Priority	Burst Time
A	0	3	20
B	5	1	15
C	10	2	10
D	15	4	5

- Draw gantt charts, find average turnaround and waiting times for above processes, considering:
- 1) First Come First Served Scheduling
- 2) Shortest Job First Scheduling (non-preemptive)
- 3) Shortest Job First Scheduling (preemptive)
- 4) Round-Robin Scheduling
- 5) Priority Scheduling (non-preemptive)
- 6) Priority Scheduling (preemptive)

28

Summary

- CPU Scheduling
 - Basic Concepts
 - Scheduling Criteria & Metrics
 - Different Scheduling Algorithms
 - FCFS
 - SJF
 - Priority
 - RR



- Next Lecture: Project Overview
- Reading Assignment: Chapter 5 from Silberschatz.

29

Acknowledgements

- “Operating Systems Concepts” book and supplementary material by A. Silberschatz, P. Galvin and G. Gagne
- “Operating Systems: Internals and Design Principles” book and supplementary material by W. Stallings
- “Modern Operating Systems” book and supplementary material by A. Tanenbaum
- R. Doursat and M. Yuksel from UNR

30