

Operating Systems Concepts

System Calls in xv6

CS 4375, Fall 2025

Instructor: MD Armanuzzaman (*Arman*)

marmanuzzaman@utep.edu

September 24, 2025

Summery

- Assembly programming
- Privileged CPU features
 - Registers and instructions
- xv6 system call
 - How does the whole cycle of system calls work?
 - Go over code snippets

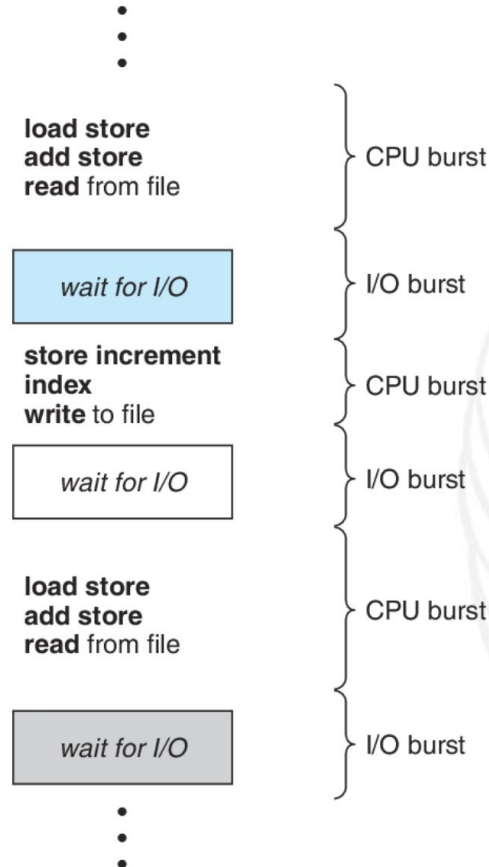
Agenda

- CPU Scheduling:
 - Basic concepts
 - Scheduling Criteria & Metrics
 - Different Scheduling Algorithms
 - FCFS
 - SJF
 - Priority
 - RR
 - Preemptive vs Non-preemptive Scheduling
 - Gantt Charts & Performance Comparison

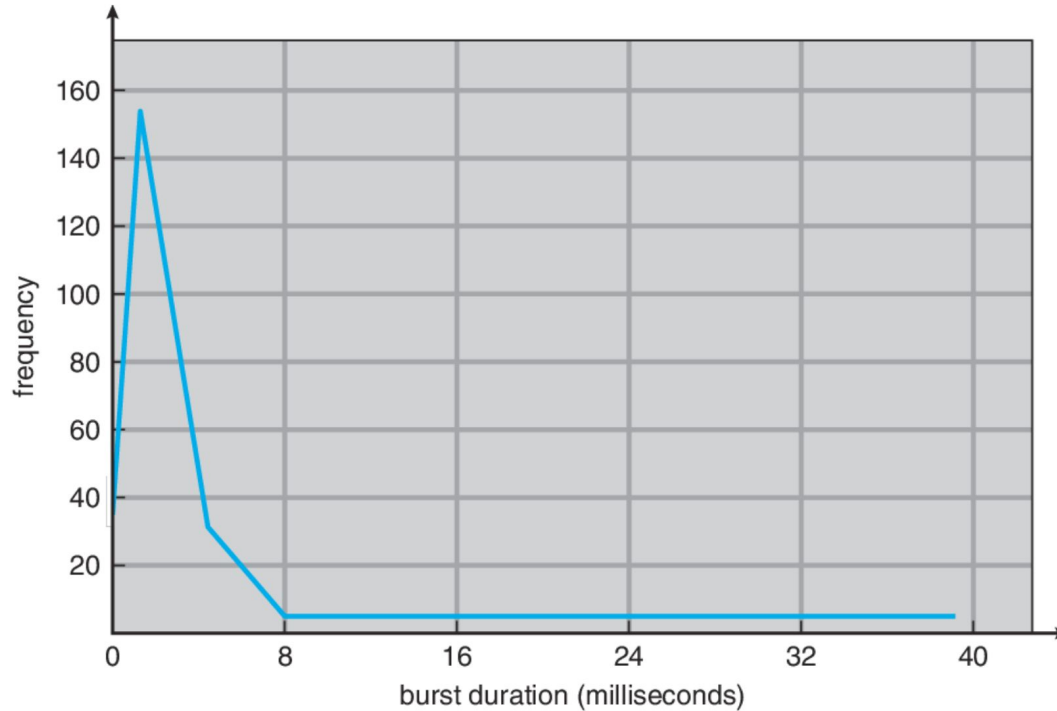
Basic Concepts

- Multiprogramming is needed for efficient CPU utilization
- **CPU Scheduling**: deciding when to execute which processes
- 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**

Alternating Sequence of CPU And I/O Bursts

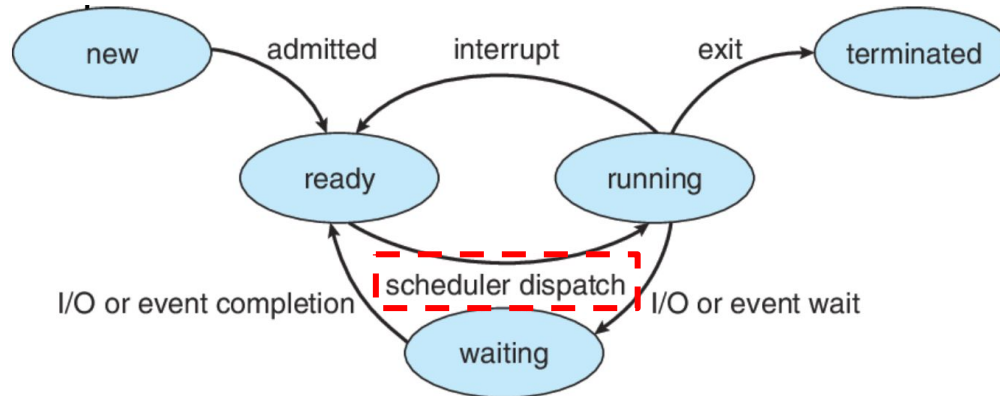


Histogram of CPU-burst Durations



Process State

- A process changes its **state** during execution:
 - **New**: The process is being created
 - **Ready**: The process is waiting to be assigned to a processor
 - **Running**: Instructions are being executed
 - **Waiting**: The process is waiting for some event to occur
 - **Terminated**: The process has finished execution



CPU Scheduler

- Selects from among the processes in memory that are **ready** to execute, and allocates the CPU to one of them \Rightarrow short-term scheduler
- CPU scheduling decisions may take place when a process:
 1. A new process arrives *preemptive*
 2. Switches from running to ready state *preemptive*
 3. Switches from waiting to ready *preemptive*
 4. Switches from running to waiting state *non-preemptive/cooperative*
 5. Terminates *non-preemptive/cooperative*

CPU Scheduler

- Scheduling under 4 (wait) and 5 (termination) is *non-preemptive/cooperative*
 - Once a process gets the CPU, keeps it until termination/switching to waiting state/release of the CPU
- All other schedulings are *preemptive*
 - Most OSs use this
 - e.g. time quota expires
 - Cost associated with access to shared data

Dispatcher

- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler.
- It's function involves:
 - Switching context
 - Switching to user mode
 - Jumping to the proper location in the user program to resume that program
- **Dispatch latency:** Time it takes for the dispatcher to stop one process and start another one.

Scheduling Criteria

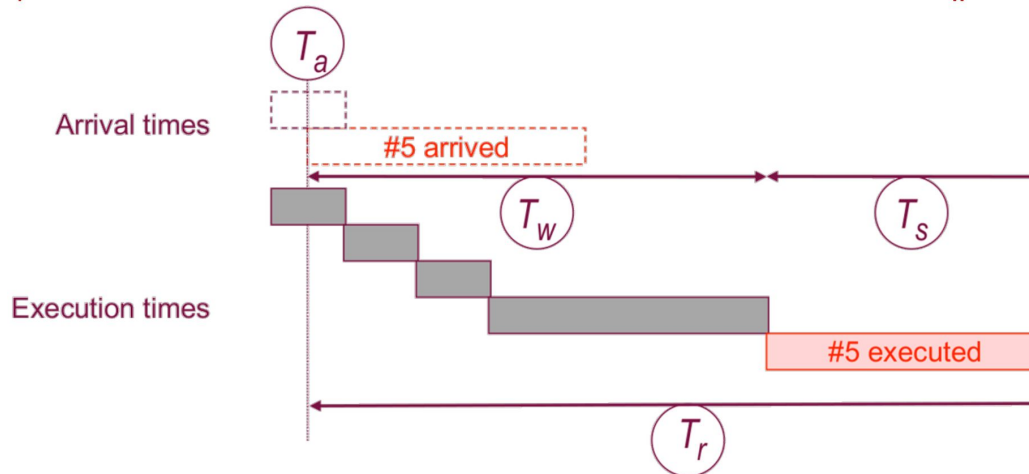
- **CPU Utilization:** Keep the CPU as busy as possible **Maximize**
- **Throughput:** # of processes that complete their execution per time unit **Maximize**
- **Response Time:** Amount of time it takes from when a request was submitted until the first response (not output) is produced (for time-sharing environment) **Minimize**
- **Waiting Time:** Total amount of time a process has been waiting in the ready queue **Minimize**
- **Turnaround Time:** Amount of time passed to finish execution of a particular process i.e. execution(service) time + waiting time **Minimize**

Optimization Criteria

- Maximize CPU Utilization
- Maximize Throughput
- Minimize Response Time
- Minimize Waiting Time
- Minimize Turnaround Time

Scheduling Metrics

- T_a - **Arrival time**: Time the process became “READY” [again]
- T_w - **Waiting time**: Time spent waiting for the CPU
- T_s - **Service time**: Time spent executing in the CPU
- T_r - **Turnaround time**: Time spent waiting and executing = $T_w + T_s$

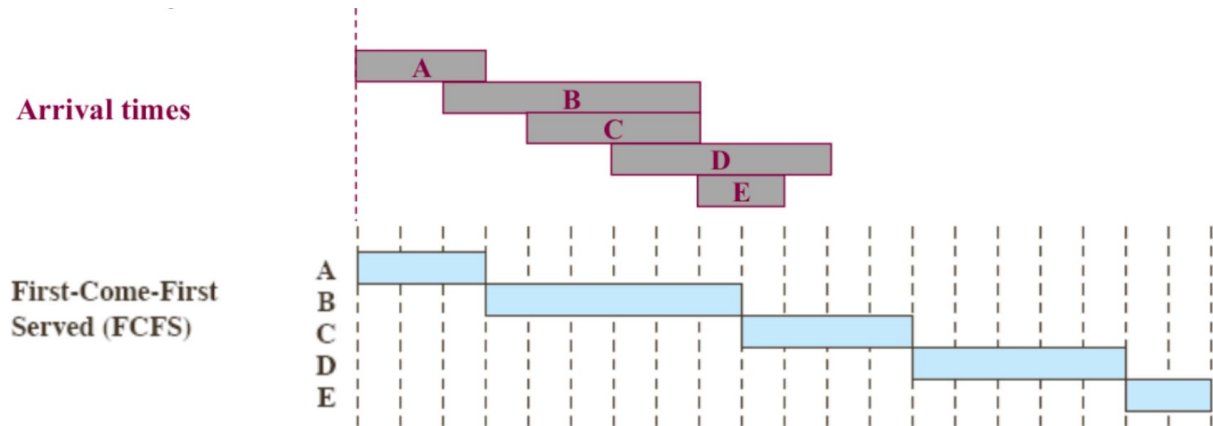


$$T_r / T_s = 2.5$$

Scheduling: First-Come, First-Served (FCFS)

- Processes are assigned the CPU in the order they request it
- When the running process blocks, the first “READY” process is selected
- When a process gets “READY”, it is put at the end of the queue

FCFS
Scheduling
Policy



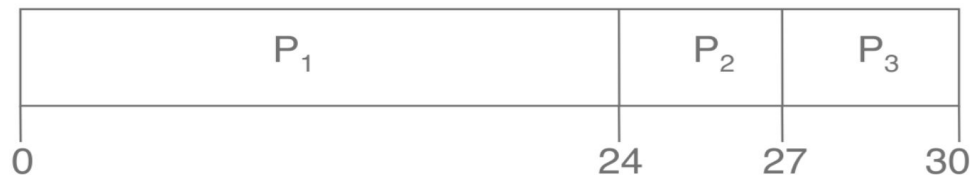
FCFS	Finish Time	A	3	B	9	C	13	D	18	E	20	Mean
	Turnaround Time (T_T)		3		7		9		12		12	8.60
	T_T/T_S		1.00		1.17		2.25		2.40		6.00	2.56

Scheduling: FCFS Example

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

Process	Burst Time
P_1	24
P_2	3
P_3	3

- The Gantt chart for the schedule is:



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

Scheduling: FCFS Example

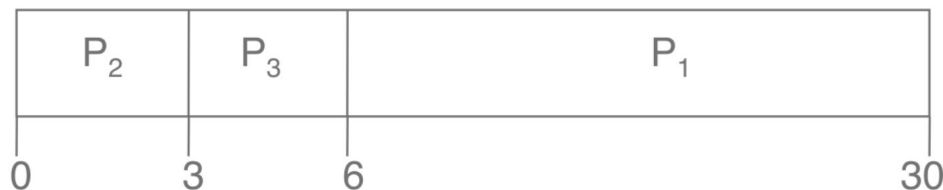
- Suppose that the processes arrive in the order: P_2, P_3, P_1
- The **Gantt chart** for the schedule is:



- 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

Scheduling: FCFS 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

Scheduling: FCFS Example



Convoy effect: short process behind long process

Scheduling: Shortest-Job-First (SJF)

- Associate with each process the length of its next CPU burst. Use these lengths to schedule the process with the shortest time
- Two schemes:
 - **Non-preemptive** – 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)**
- **Assumes the run times are known in advance!**
- SJF(SRTF) is optimal – gives minimum average waiting time for a given set of processes

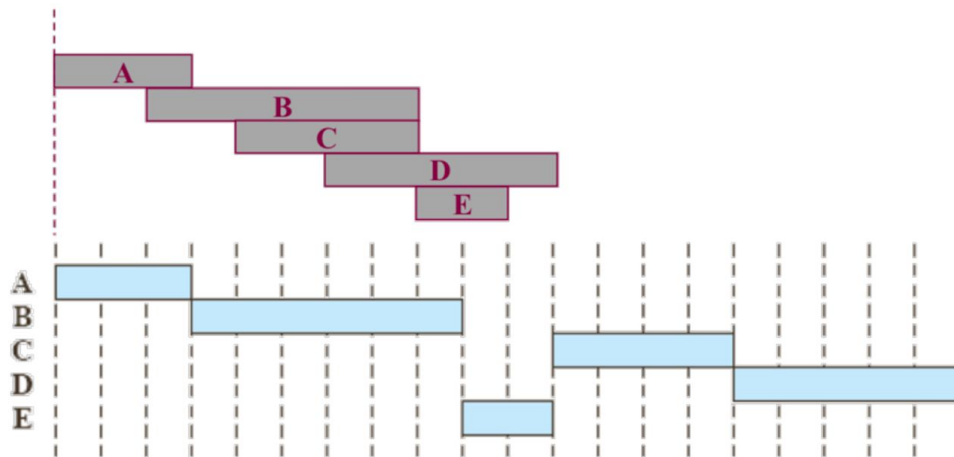
Scheduling: Non-Preemptive SJF

- Among several equally important “**READY**” jobs (or CPU bursts), the scheduler picks the one that will finish the earliest

SJF
Scheduling
Policy

Arrival times

Shortest Job
First (SJF)

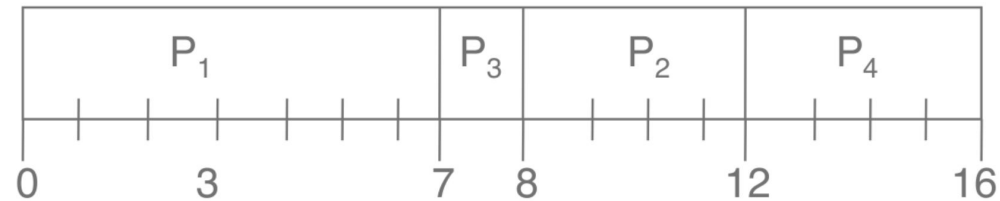


SJF	Finish Time	A	3	B	9	C	15	D	20	E	11	Mean
	Turnaround Time (T_T)		3		7		11		14		3	7.60
	T_T/T_S		1.00		1.17		2.75		2.80		1.50	1.84

Scheduling: Non-Preemptive SJF Example

Process	Arrival Time	Burst Time
P ₁	0	7
P ₂	2	4
P ₃	4	1
P ₄	5	4

- The Gantt chart for the schedule is:



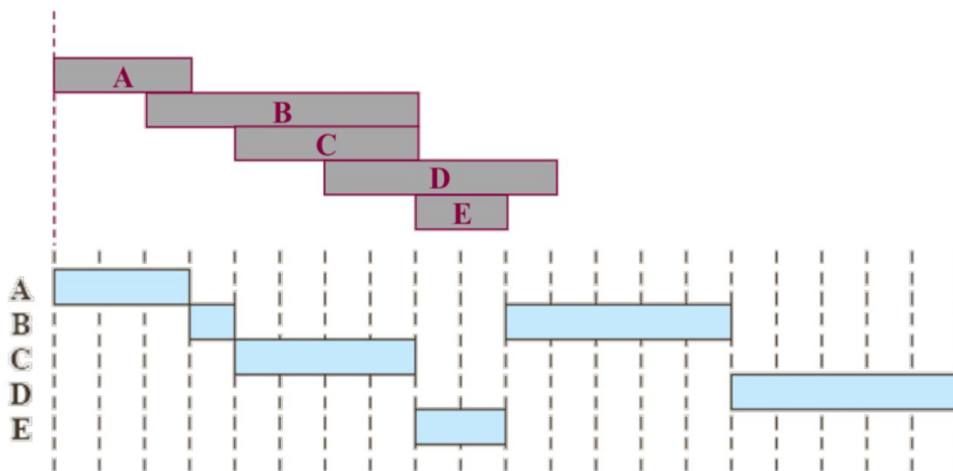
- Average waiting time: $(0 + 6 + 3 + 7) / 4 = 4$

Scheduling: Preemptive SJF

- Choose the process whose **remaining** run time is shortest.
- Allows new short jobs to get good service

Arrival times

Shortest Remaining
Time (SRT)

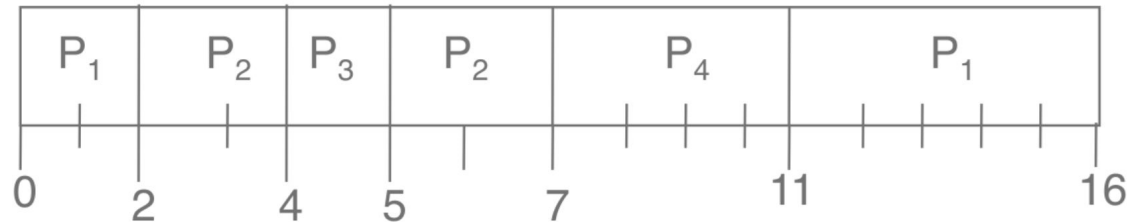


SRT	Finish Time	A	3	B	15	C	8	D	20	E	10	Mean
	Turnaround Time (T_r)	3	13	4	14	2	7.20					
	T_r/T_s	1.00	2.17	1.00	2.80	1.00						

Scheduling: Preemptive SJF Example

Process	Arrival Time	Burst Time
P ₁	0	7
P ₂	2	4
P ₃	4	1
P ₄	5	4

- The Gantt chart for the schedule is:



- Average waiting time: $(9 + 1 + 0 + 2) / 4 = 3$

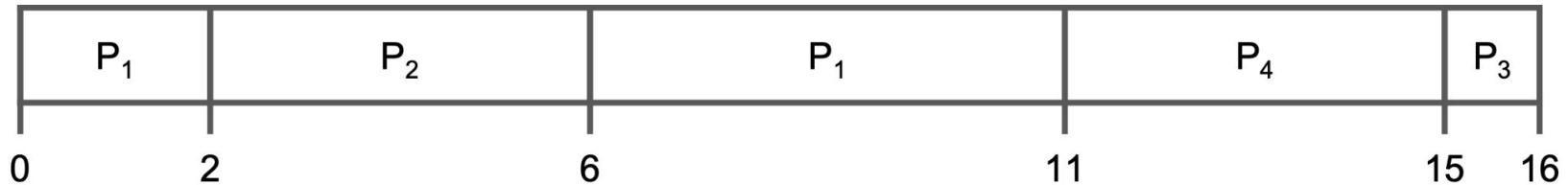
Scheduling: Priority

- A priority number (integer) is associated with each process
- The CPU is allocated to the process with the highest priority (**smallest/largest** number)
- Two schemes:
 - **Non-preemptive** – Once CPU given to the process it cannot be preempted until completes its CPU burst
 - **Preemptive** – If a new process arrives with higher priority, preempt
- 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

Scheduling: Priority Example

Process	Arrival Time	Burst Time	Priority
P ₁	0	7	2
P ₂	2	4	1
P ₃	4	1	4
P ₄	5	4	3

- Non-preemptive:
 - $P_1 \rightarrow P_2 \rightarrow P_4 \rightarrow P_3$
- Preemptive:



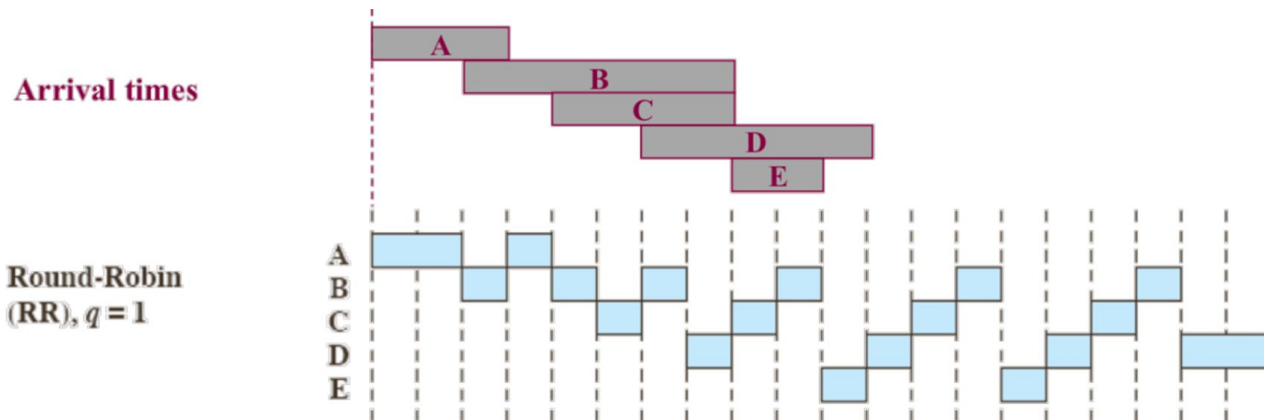
Scheduling: Round-Robin (RR)

- Each process gets a small unit of CPU time (i.e. *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:
 - Large $q \rightarrow$ FCFS
 - Small $q \rightarrow q$ must be large enough with respect to context switch time, otherwise overhead would be too high

Scheduling: Round-Robin (RR)

- Preemptive FCFS, based on a timeout interval, the **time quantum (TIME_SLICE)**
- The running process is interrupted by the clock and put last in a FIFO **“READY”** queue

RR ($q = 1$)
Scheduling
Policy

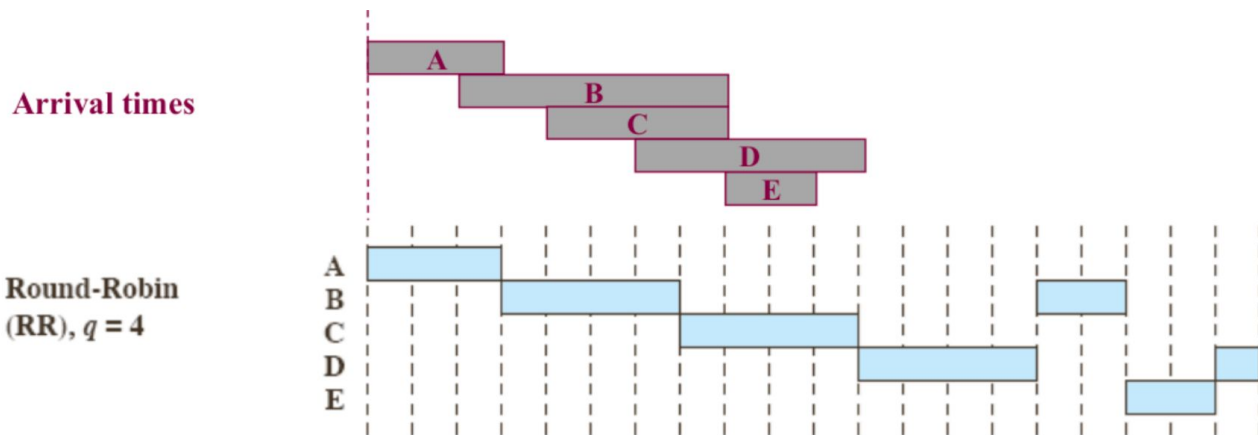


RR $q = 1$	Finish Time	A	4	B	18	C	17	D	20	E	15	Mean	
	Turnaround Time (T_r)		4		16		13		14		7		10.80
	T_r/T_s		1.33		2.67		3.25		2.80		3.50		2.71

Scheduling: Round-Robin (RR)

- A crucial parameter is the quantum q ($\sim 10\text{-}100\text{ms}$)
 - q should be large compared to context switch latency ($\sim 10\mu\text{s}$)
 - q should be less than the longest CPU burst, or RR **degenerates** to FCFS

RR ($q = 4$)
Scheduling
Policy



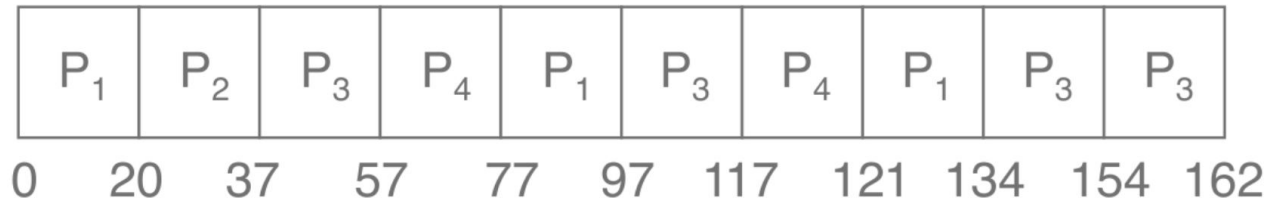
RR $q = 4$	Finish Time	A 3	B 17	C 11	D 20	E 19	Mean
	Turnaround Time (T_T)	3	15	7	14	11	10.00
	T_T/T_S	1.00	2.5	1.75	2.80	5.50	2.71

Scheduling: RR Example

- Consider $q = 20$

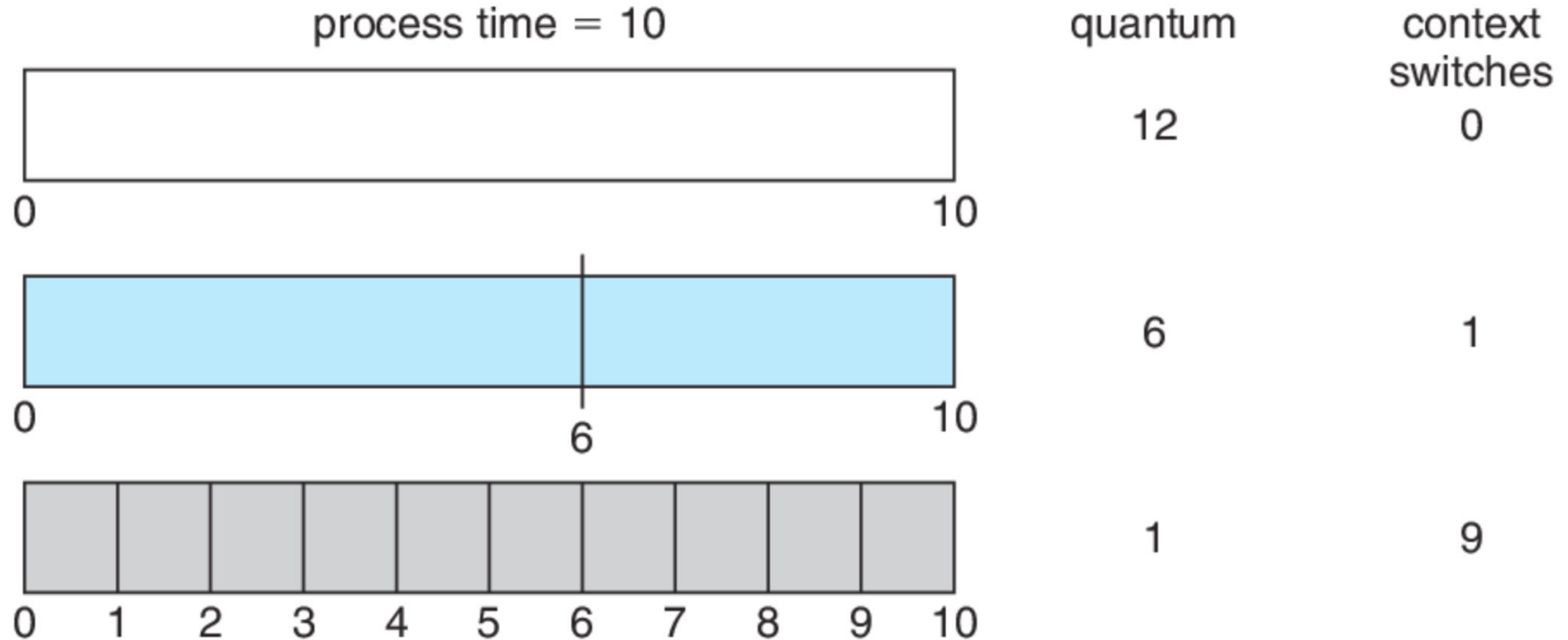
Process	Burst Time
P_1	53
P_2	17
P_3	68
P_4	24

- The Gantt chart for the schedule is:

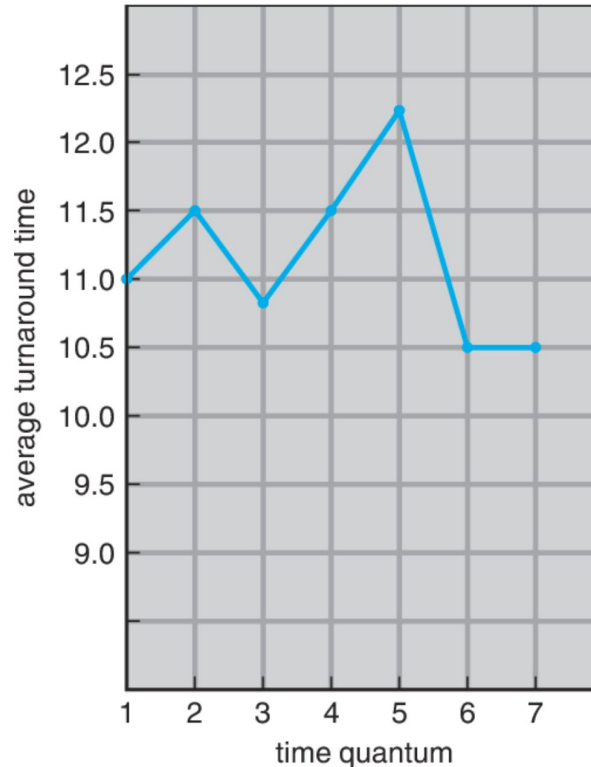


- Typically, higher average turnaround than SJF, but better *response*

Time Quantum and Context Switch Time



Turnaround Time Varies With The Time Quantum



process	time
P_1	6
P_2	3
P_3	1
P_4	7

Exercise

- Consider a system using round-robin scheduling with a fixed quantum q . Every context switch takes s milliseconds. Any given process runs for an average of t milliseconds before it blocks or terminates (burst time)
- Determine the **fraction of CPU time** that will be wasted because of context switches for each of the following cases (Your answer should be in terms of q , s , and t)
 - $t \leq q$
 - $t \gg q$ (t is much greater than q)
 - $q \rightarrow 0$

Context switching time

Context switching time + useful CPU burst

Exercise

- Consider a system using round-robin scheduling with a fixed quantum q . Every context switch takes s milliseconds. Any given process runs for an average of t milliseconds before it blocks or terminates (burst time)
- Determine the **fraction of CPU time** that will be wasted because of context switches for each of the following cases (Your answer should be in terms of q , s , and t)
 - $t \leq q$
$$s / (t + s)$$
 - $t \gg q$ (t is much greater than q)
$$s / (q + s)$$
 - $q \rightarrow 0$
$$100\%$$

Comparison: FCFS

- PROS:

- It is a fair algorithm
 - Schedule in the order that they arrive

- CONS:

- Average response time can be lousy
 - Small requests wait behind big ones (convoy effect)
- May lead to poor utilization of other resources
 - FCFS may result in poor overlap of CPU and I/O activity
 - E.g., a CPU-intensive job prevents an I/O intensive job from doing a small bit of computation, thus preventing it from going back and keep the I/O subsystem busy

Comparison: SJF

- PROS:

- Provably optimal with respect to average waiting time
 - Prevents convoy effect

- CONS:

- Can cause starvation of long jobs
- Requires advanced knowledge of CPU burst times
 - This is not easy to predict!

Comparison: Priority

- PROS:

- Guarantees early completion of high priority jobs

- CONS:

- Can cause starvation of low priority jobs
- How to decide/assign priority value?

Comparison: Round-Robin

- PROS:

- Great for timesharing
 - No starvation
- Does not require prior knowledge of CPU burst times
- Generally reduces average response time

- CONS:

- What if all jobs are almost the same length
 - Increases the turnaround time
- How to set the “best” time quantum?
 - If too small, it increases context-switch overhead
 - If too large, response time degrades

Exercise

Process	Arrival Time	Burst Time	Priority
P ₁	9	3	5
P ₂	6	4	4
P ₃	8	5	3
P ₄	7	9	2
P ₅	5	7	1

- Consider Shortest-Remaining-Time-First (SRTF) scheduling policy is used.
 - a. Draw the Gantt chart
 - b. Find the response time, waiting time, and turnaround time for each process.

Exercise

Process	Arrival Time	Burst Time	Priority
P_1	0	20	3
P_2	5	15	1
P_3	10	10	2
P_4	15	5	4

- Draw Gantt charts, find average turnaround time, waiting time, response time for each processor, for each of FCFS, SJF (non-preemptive), SRTF (preemptive), RR ($q=4$), Priority (preemptive and non-preemptive)

Announcement

- Quiz 2 (Released on Blackboard)
 - Due at 11.59 PM tomorrow
 - 30 minutes, single attempt, 15 MCQs
- Homework 2
 - Due on Wednesday october 1st, 11.59 PM