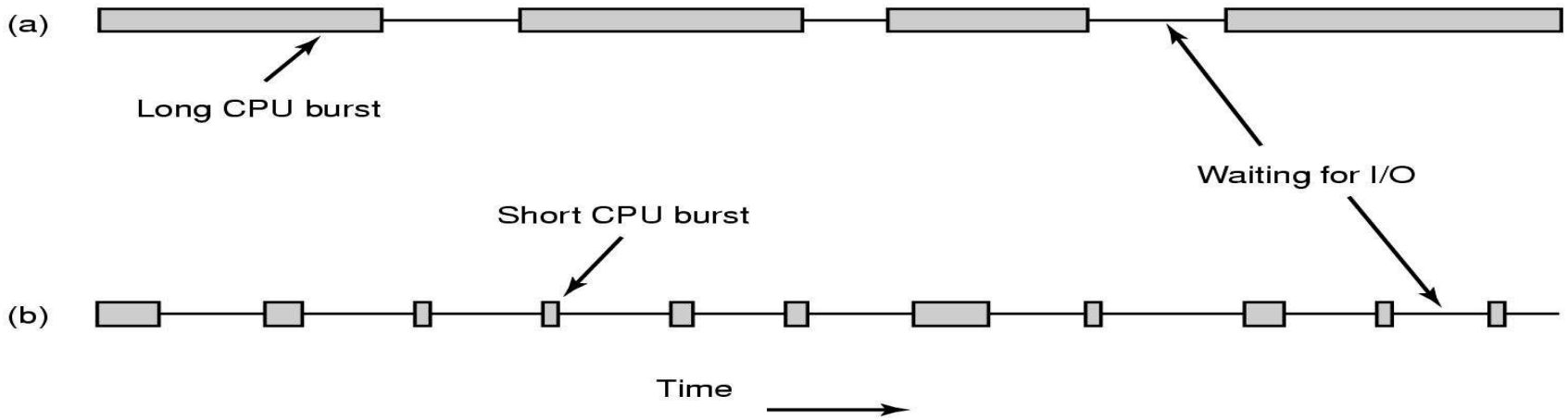
Scheduling

Which process is given control of the CPU and how long?

By switching the processor among the processes, the OS can make the computer more productive.

Scheduling

CPU-I/O Burst



Process execution consists of a cycle of CPU execution and I/O wait.

Process execution begins with a CPU burst that is followed by I/O burst, then another CPU burst, then another I/O burst.....

CPU-bound: Processes that use CPU until the quantum expire.

I/O-bound: Processes that use CPU briefly and generate I/O request.

CPU-bound processes have a long CPU-burst while I/O-bound processes have short CPU burst.

Key idea: when I/O bound process wants to run, it should get a chance quickly.

Scheduling
When to Schedule

1. When a new process is created.
2. When a process terminates.
3. When a process blocks on I/O, on semaphore, waiting for child termination etc.
4. When an I/O interrupt occurs.
5. When quantum expires.

Preemptive vs. Nonpreemptive Scheduling

Nonpreemptive:

- Once a process has been given the CPU, it runs until blocks for I/O or termination.
- Treatment of all processes is fair.
- Response times are more predictable.
- Useful in real-time system.
- Shorts jobs are made to wait by longer jobs - no priority

Preemptive:

- Processes are allowed to run for a maximum of some fixed time.
- Useful in systems in which high-priority processes requires rapid attention.
- In timesharing systems, preemptive scheduling is important in guaranteeing acceptable response times.
- High overhead.

Role of Dispatcher vs. Scheduler

Dispatcher

- Low level mechanism.
- Responsibility: Context switch
  - Save execution state of old process in PCB.
  - Load execution state of new process from PCB to registers.
  - Change the scheduling state of the process (running, ready, blocked)
  - Switch from kernel to user mode.

Scheduler

- Higher-level policy.
- Responsibility: Which process to run next.

Scheduling Criteria

The scheduler is to identify the process whose selection will results the best possible system performance.

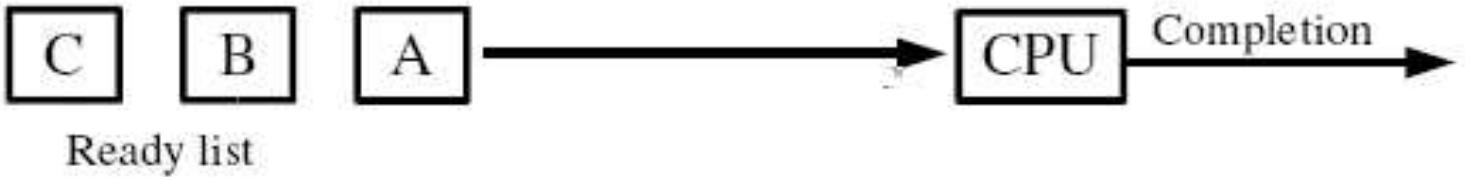
Criteria for comparing scheduling algorithms:

CPU Utilization	*Balance Utilization*
Throughput	*Turnaround Time*
Waiting Time	*Response Time*
Predictability	*Fairness*
Priorities	

The scheduling policy determine the important of each criteria.
The scheduling algorithms should designed to optimizes maximum possible criteria.

Scheduling Algorithms

First-Come First-Serve (FCFS)



Processes are scheduled in the order they are received.

Once the process has the CPU, it runs to completion -Nonpreemptive.

Easily implemented, by managing a simple queue or by storing time the process was received.

Fair to all processes.

Problems:

No guarantee of good response time.

Large average waiting time.

Not applicable for interactive system.

Scheduling Algorithms

Shortest Job First (SJF)

The processing times are known in advanced.

SJF selects the process with shortest expected processing time. In case of the tie FCFS scheduling is used.

The decision policies are based on the CPU burst time.

Advantages:

Reduces the average waiting time over FCFS.

Favors shorts jobs at the cost of long jobs.

Problems:

Estimation of run time to completion. Accuracy?

Not applicable in timesharing system.

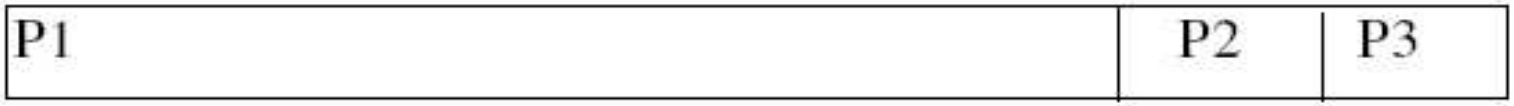
Scheduling Algorithms

SJF -Performance

Scenario: Consider the following set of processes that arrive at time 0, with length of CPU-bust time in milliseconds.

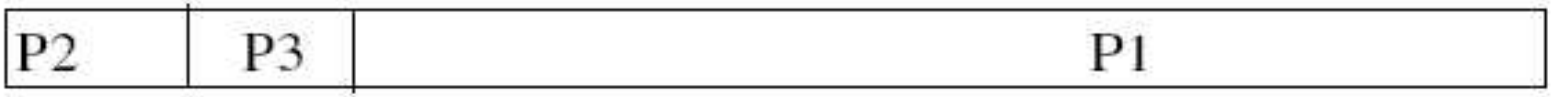
Processes	Burst time
P1	24
P2	3
P3	3

if the processes arrive in the order P1, P2, P3 and are served in FCFS order,



The average waiting time is (0 + 24 + 27)/3 = 17.

if the processes are served in SJF



The average waiting time is (6 + 0 + 3)/3 = 3.

Scheduling Algorithms

Shortest-Remaining-Time-First (SRTF)

Preemptive version of SJF.

Any time a new process enters the pool of processes to be scheduled, the scheduler compares the expected value for its remaining processing time with that of the process currently scheduled. If the new process's time is less, the currently scheduled process is preempted.

Merits:

Low average waiting time than SJF.

Useful in timesharing.

Demerits:

Very high overhead than SJF.

Requires additional computation.

Favors short jobs, longs jobs can be victims of starvation.

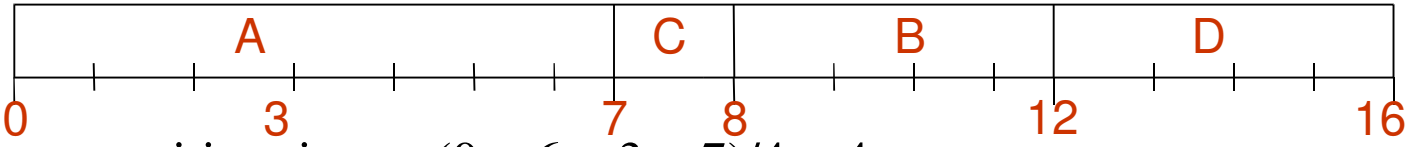
Scheduling Algorithms

SRTF-Performance

Scenario: Consider the following four processes with the length of CPU-burst time given in milliseconds:

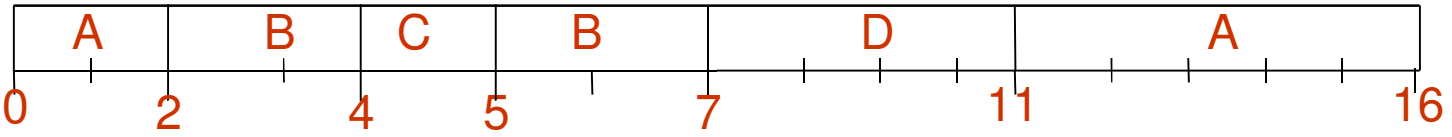
Processes	Arrival Time	Burst Time
A	0.0	7
B	2.0	4
C	4.0	1
D	5.0	4

SJF:



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

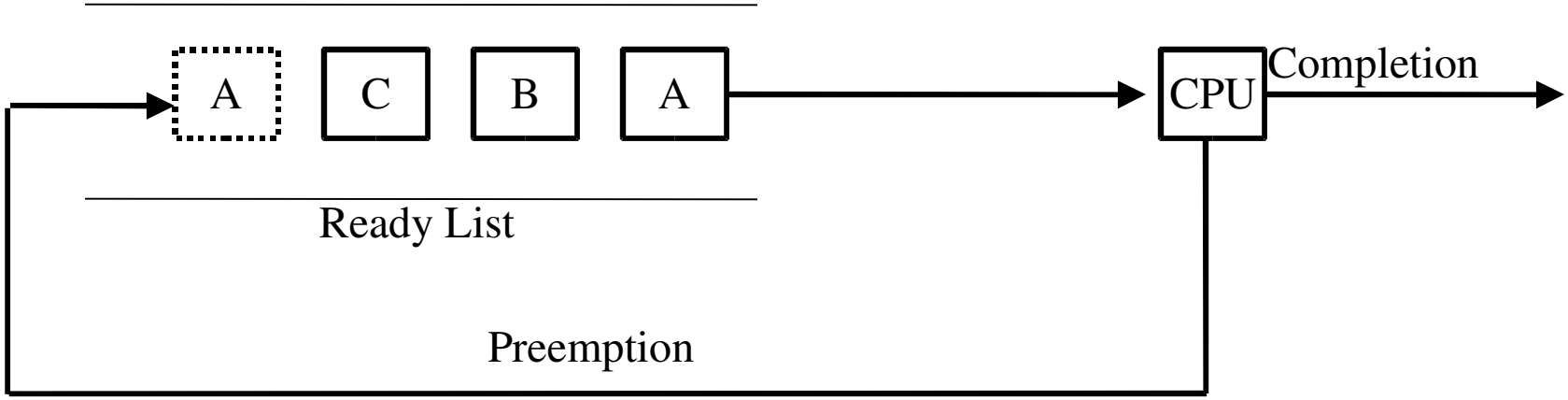
SRTF:



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

Scheduling Algorithms

Round-Robin (RR)



Preemptive FCFS.

Each process is assigned a time interval (quantum), after the specified quantum, the running process is preempted and a new process is allowed to run.

Preempted process is placed at the back of the ready list.

Advantages:

Fair allocation of CPU across the process.

Used in timesharing system.

Low average waiting time when process lengths very widely.

Scheduling Algorithms

RR-Performance

- Poor average waiting time when process lengths are identical.

Imagine 10 processes each requiring 10 msec burst time and 1msec quantum is assigned.

RR: All complete after about 100 times.

FCFS is better! (About 20% time wastages in context-switching).

- Performance depends on quantum size.

Quantum size:

If the quantum is very large, each process is given as much time as needs for completion; RR degenerate to FCFS policy.

If quantum is very small, system busy at just switching from one process to another process, the overhead of context-switching causes the system efficiency degrading.

Optimal quantum size?

Key idea: 80% of the CPU bursts should be shorter than the quantum.

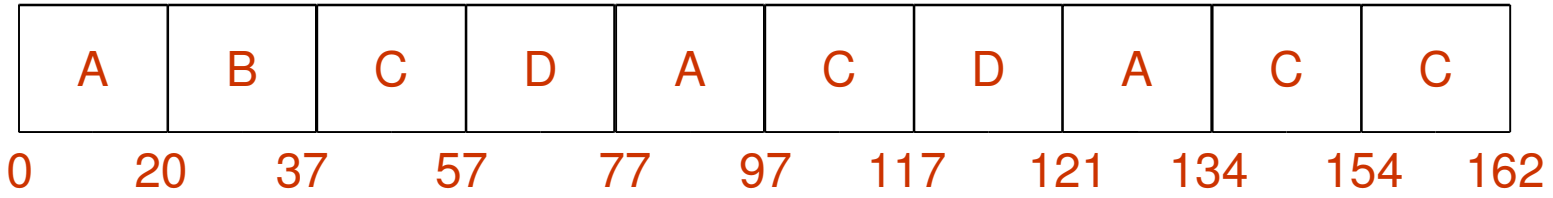
20-50 msec reasonable for many general processes.

Scheduling Algorithms

Example of RR with Quantum = 20

Process	Burst Time
A	53
B	17
C	68
D	24

- The Gantt chart is:



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

Scheduling Algorithms

Priority

Each process is assigned a priority value, and runnable process with the highest priority is allowed to run.

FCFS or RR can be used in case of tie.

To prevent high-priority process from running indefinitely, the scheduler may decrease the priority of the currently running process at each clock tick.

Assigning Priority

Static:

Some processes have higher priority than others.

Problem: Starvation.

Dynamic:

Priority chosen by system.

Decrease priority of CPU-bound processes.

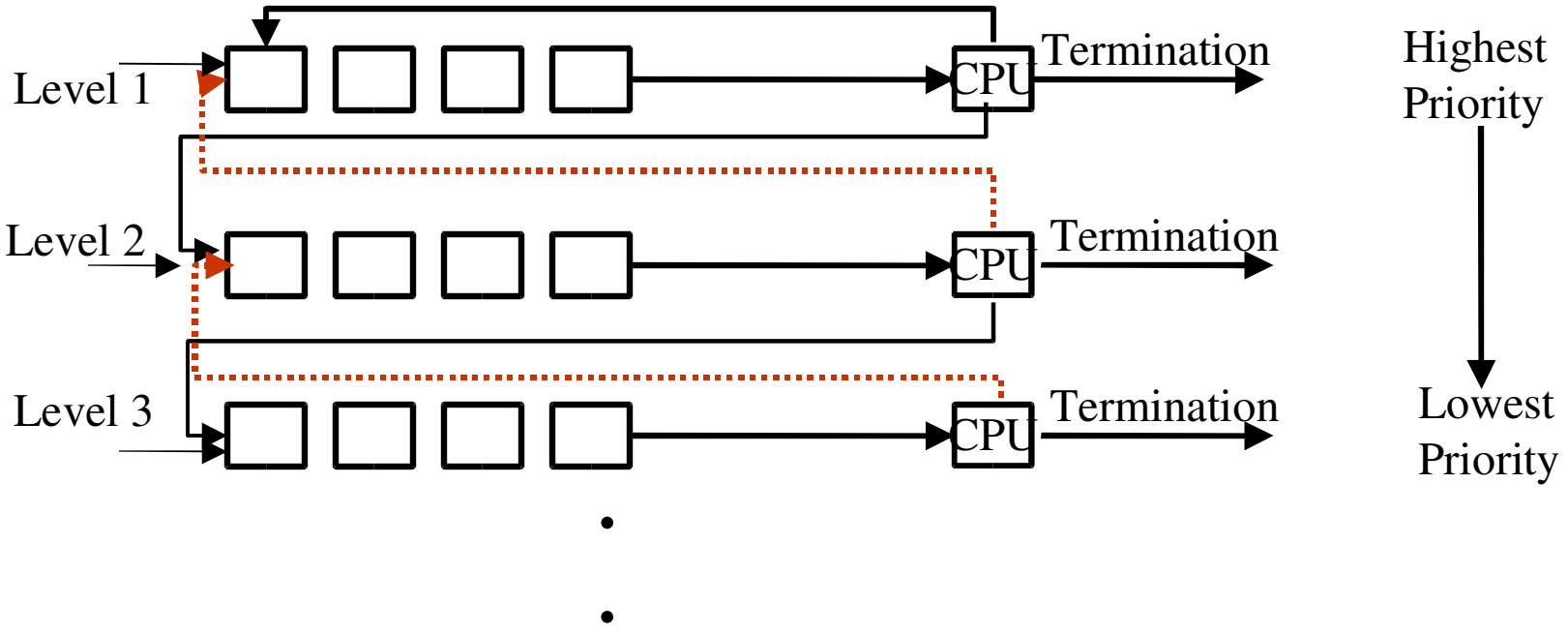
Increase priority of I/O-bound processes.

Many different policies possible…….

E. g.: priority = (time waiting + processing time)/processing time.

Scheduling Algorithms

Multilevel-Feedback-Queues (MFQ)



MFQ implements multilevel queues having different priority to each level (here lower level higher priority), and allows a process to move between the queues. If the process use too much CPU time, it will be moved to a lower-priority queue.

Each lower-priority queue larger the quantum size.

This leaves the I/O-bound and interactive processes in the high priority queue.

Scheduling Algorithms

MFQ-Example

Consider a MFQ scheduler with three queues numbered 1 to 3, with quantum size 8, 16 and 32 msec respectively.

The scheduler execute all process in queue 1, only when queue 1 is empty it execute process in queue 2 and process in queue 3 will execute only if queue 1 and queue 2 are empty.

A process first enters in queue 1 and execute for 8 msec. If it does not finish, it moves to the tail of queue 2.

If queue 1 is empty the processes of queue 2 start to execute in FCFS manner with 16 msec quantum. If it still does not complete, it is preempted and move to the queue 3.

If the process blocks before using its entire quantum, it is moved to the next higher level queue.

Home Works

HW#5

1. Textbook (Tanenbaum) 35, 36, 37, 38, 39 & 40.
2. For the processes listed in following table, draw a Gantt chart illustrating their execution using:
   (a) First-Come-First-Serve.
   (b) Short-Job-First.
   (c) Shortest-Remaining-Time-Next.
   (d) Round-Robin (quantum = 2).
   (e) Round-Robin (quantum = 1).

Processes	Arrival Time	Burst Time
A	0.00	4
B	2.01	7
C	3.01	2
D	3.02	2

   - i) What is the turnaround time?
   - ii) What is average waiting?