Discussion Week 10

TA: Kyle Dewey

Overview

- TA Evaluations
- Project #3
- PE 5.1
- PE 5.3
- PE 11.8 (a,c,d)
- PE 10.1

TA Evaluations

Project #3

PE 5.1

A CPU scheduling algorithm determines an order for the execution of its scheduled processes. Given *n* processes to be scheduled on one processor, how many different schedules are possible? Give a formula in terms of *n*.

PE 5.3

Suppose that the following processes arrive for execution at the times indicated. Each process will run for the amount of time listed. In answering these questions, use nonpreemptive scheduling and base all decisions on the information you have at the time the decision must be made.

Process	Arrival Time	Burst Time
P1	0.0	8
P2	0.4	4
P3	1.0	1

- What is the average turnaround time for these processes with the FCFS algorithm?
- What is the average turnaround time for these processes with the SJF scheduling algorithm?

The SJF algorithm is supposed to improve performance, but notice that we chose to run process P1 at time 0 because we did not know that two shorter processes would arrive soon. Compute what the average turnaround time will be if the CPU is left idle for the first 1 unit and then SJF scheduling is used. Remember that processes P1 and P2 are waiting during this idle time, so their waiting time may increase. This algorithm could be known as future-knowledge scheduling.

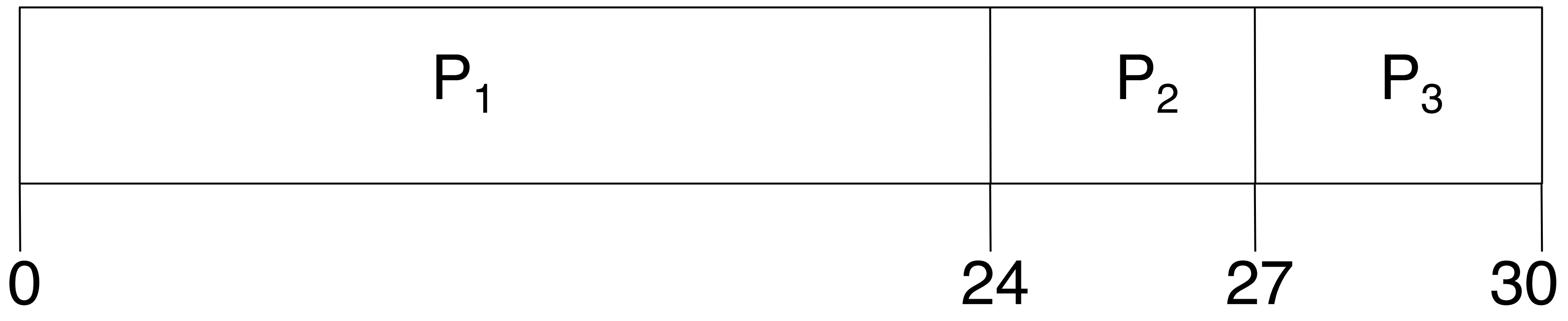
Scheduling Criteria

- **CPU utilization** – keep the CPU as busy as possible
- **Throughput** – # of processes that complete their execution per time unit
- **Turnaround time** – amount of time to execute a particular process
- **Waiting time** – amount of time a process has been waiting in the ready queue
- **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)

First-Come, First-Served (FCFS) Scheduling

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:



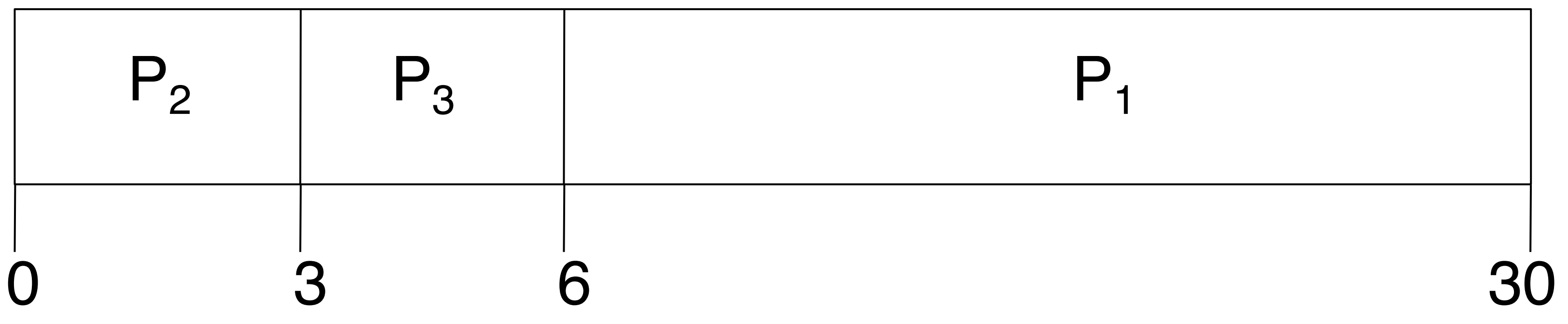
- Waiting time for P_1 = 0; P_2 = 24; P_3 = 27
- Average waiting time: (0 + 24 + 27)/3 = 17

FCFS Scheduling (Cont.)

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

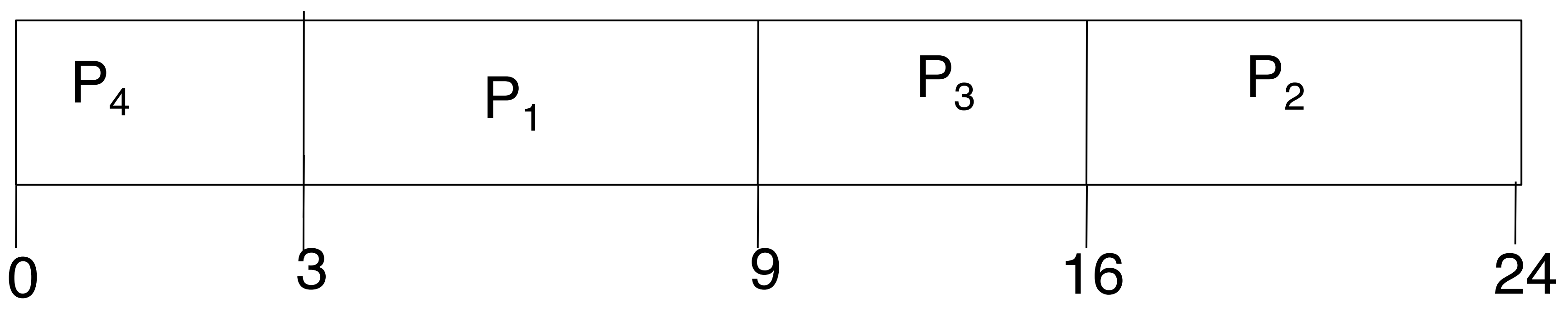
Shortest-Job-First (SJF) Scheduling

- Associate with each process the length of its next CPU burst. Use these lengths to schedule the process with the shortest time.
- SJF is optimal – gives minimum average waiting time for a given set of processes
  - The difficulty is knowing the length of the next CPU request.

Example of SJF

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

- SJF scheduling chart



- Average waiting time = (3 + 16 + 9 + 0) / 4 = 7

PE 11.8 (a,c,d)

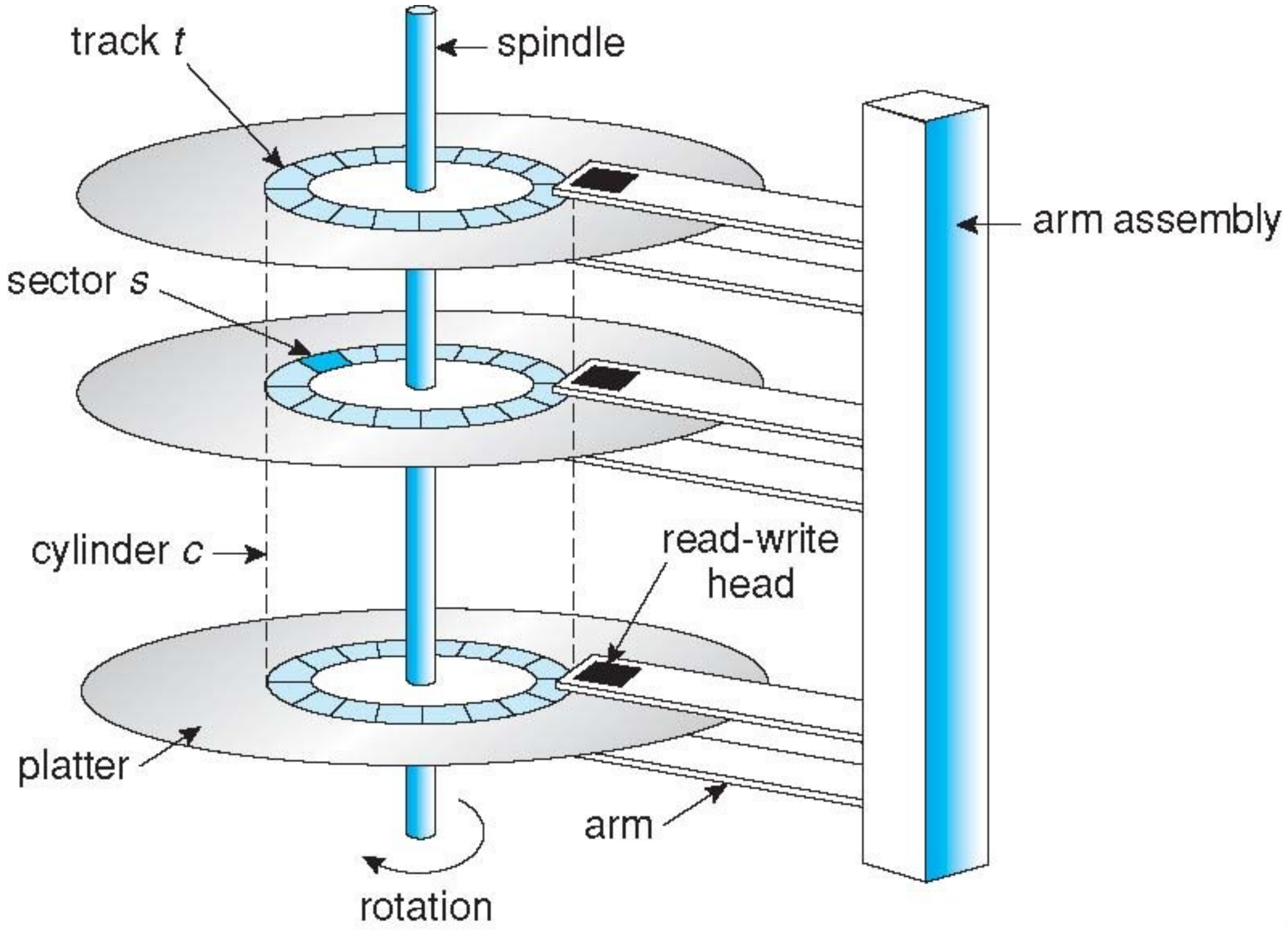
The term "Fast Wide SCSI-II" denotes a SCSI bus that operates at a data rate of 20 megabytes per second when it moves a packet of bytes between the host and a device. Suppose that a Fast Wide SCSI-II disk drive spins at 7,200 RPM, has a sector size of 512 bytes, and holds 160 sectors per track.

a.) Estimate the sustained transfer rate of this drive in megabytes per second.

c.) Suppose that the average seek time for the drive is 8 milliseconds. Estimate the I/O operations per second and the effective transfer rate for a random-access workload that reads individual sectors that are scattered across the disk.

d.) Calculate the random access I/O operations per second and transfer rate for I/O size of 4, 8, 64 Kb.

Moving-head Disk Mechanism



Estimate sustained average transferring

Estimate sustained average transferring

- Suppose that a disk drive spins at 7200 RPM (revolutions per minute), has a sector size of 512 bytes, and holds 160 sectors per track.

Estimate sustained average transferring

- Suppose that a disk drive spins at 7200 RPM (revolutions per minute), has a sector size of 512 bytes, and holds 160 sectors per track.
- What is sustained average transfer rate of this drive in megabytes per second?

Estimate sustained average transferring

- Suppose that a disk drive spins at 7200 RPM (revolutions per minute), has a sector size of 512 bytes, and holds 160 sectors per track.
- What is sustained average transfer rate of this drive in megabytes per second?
    - .Disk spins 120 times per second (7200 RPM/60)

Estimate sustained average transferring

- Suppose that a disk drive spins at 7200 RPM (revolutions per minute), has a sector size of 512 bytes, and holds 160 sectors per track.
- What is sustained average transfer rate of this drive in megabytes per second?

  - .Disk spins 120 times per second (7200 RPM/60)
  - Each spin transfers a track of 80 KB (160 sectors x0.5K)

Estimate sustained average transferring

- Suppose that a disk drive spins at 7200 RPM (revolutions per minute), has a sector size of 512 bytes, and holds 160 sectors per track.
- What is sustained average transfer rate of this drive in megabytes per second?

  - .Disk spins 120 times per second (7200 RPM/60)
  - Each spin transfers a track of 80 KB (160 sectors x0.5K)
  - Sustained average transfer rate is 120x80 = 9.6MB/s.

Average performance of random access

- 7200 RPM, sector size of 512 bytes, and 160 sectors per track.
- Average seek time for the drive is 8 milliseconds
- Estimate # of random sector I/Os per second that can be done and the effective average transfer rate for random-access of a sector?

Average performance of random access

- 7200 RPM, sector size of 512 bytes, and 160 sectors per track.
- Average seek time for the drive is 8 milliseconds
- Estimate # of random sector I/Os per second that can be done and the effective average transfer rate for random-access of a sector?

•Average rotational cost is time to travel half track: 1/120 * 50%=4.167ms

Average performance of random access

- 7200 RPM, sector size of 512 bytes, and 160 sectors per track.
- Average seek time for the drive is 8 milliseconds
- Estimate # of random sector I/Os per second that can be done and the effective average transfer rate for random-access of a sector?

•Average rotational cost is time to travel half track: 1/120 * 50%=4.167ms

•Transfer time is 8ms to seek

Average performance of random access

- 7200 RPM, sector size of 512 bytes, and 160 sectors per track.
- Average seek time for the drive is 8 milliseconds
- Estimate # of random sector I/Os per second that can be done and the effective average transfer rate for random-access of a sector?

•Average rotational cost is time to travel half track: 1/120 * 50%=4.167ms

•Transfer time is 8ms to seek
+ 4.167 ms rotational latency

Average performance of random access

- 7200 RPM, sector size of 512 bytes, and 160 sectors per track.
- Average seek time for the drive is 8 milliseconds
- Estimate # of random sector I/Os per second that can be done and the effective average transfer rate for random-access of a sector?

•Average rotational cost is time to travel half track: 1/120 * 50%=4.167ms

•Transfer time is 8ms to seek
+ 4.167 ms rotational latency
+ 0.052 ms (reading one sector takes 0.5MB/9.6MB).

Average performance of random access

- 7200 RPM, sector size of 512 bytes, and 160 sectors per track.
- Average seek time for the drive is 8 milliseconds
- Estimate # of random sector I/Os per second that can be done and the effective average transfer rate for random-access of a sector?

•Average rotational cost is time to travel half track: 1/120 * 50%=4.167ms

•Transfer time is 8ms to seek
 + 4.167 ms rotational latency
 + 0.052 ms (reading one sector takes 0.5MB/9.6MB).
 =12.219ms

Average performance of random access

- 7200 RPM, sector size of 512 bytes, and 160 sectors per track.
- Average seek time for the drive is 8 milliseconds
- Estimate # of random sector I/Os per second that can be done and the effective average transfer rate for random-access of a sector?

•Average rotational cost is time to travel half track: 1/120 * 50%=4.167ms

•Transfer time is 8ms to seek
 + 4.167 ms rotational latency
 + 0.052 ms (reading one sector takes 0.5MB/9.6MB).
 =12.219ms

•# of random sector access/second= 1/0.012219=81.8

Average performance of random access

- 7200 RPM, sector size of 512 bytes, and 160 sectors per track.
- Average seek time for the drive is 8 milliseconds
- Estimate # of random sector I/Os per second that can be done and the effective average transfer rate for random-access of a sector?

•Average rotational cost is time to travel half track: 1/120 * 50%=4.167ms

•Transfer time is 8ms to seek
+ 4.167 ms rotational latency
+ 0.052 ms (reading one sector takes 0.5MB/9.6MB).
=12.219ms

•# of random sector access/second= 1/0.012219=81.8
•Effective transferring rate: 0.5 KB/0.012.219s=0.0409MB/s.

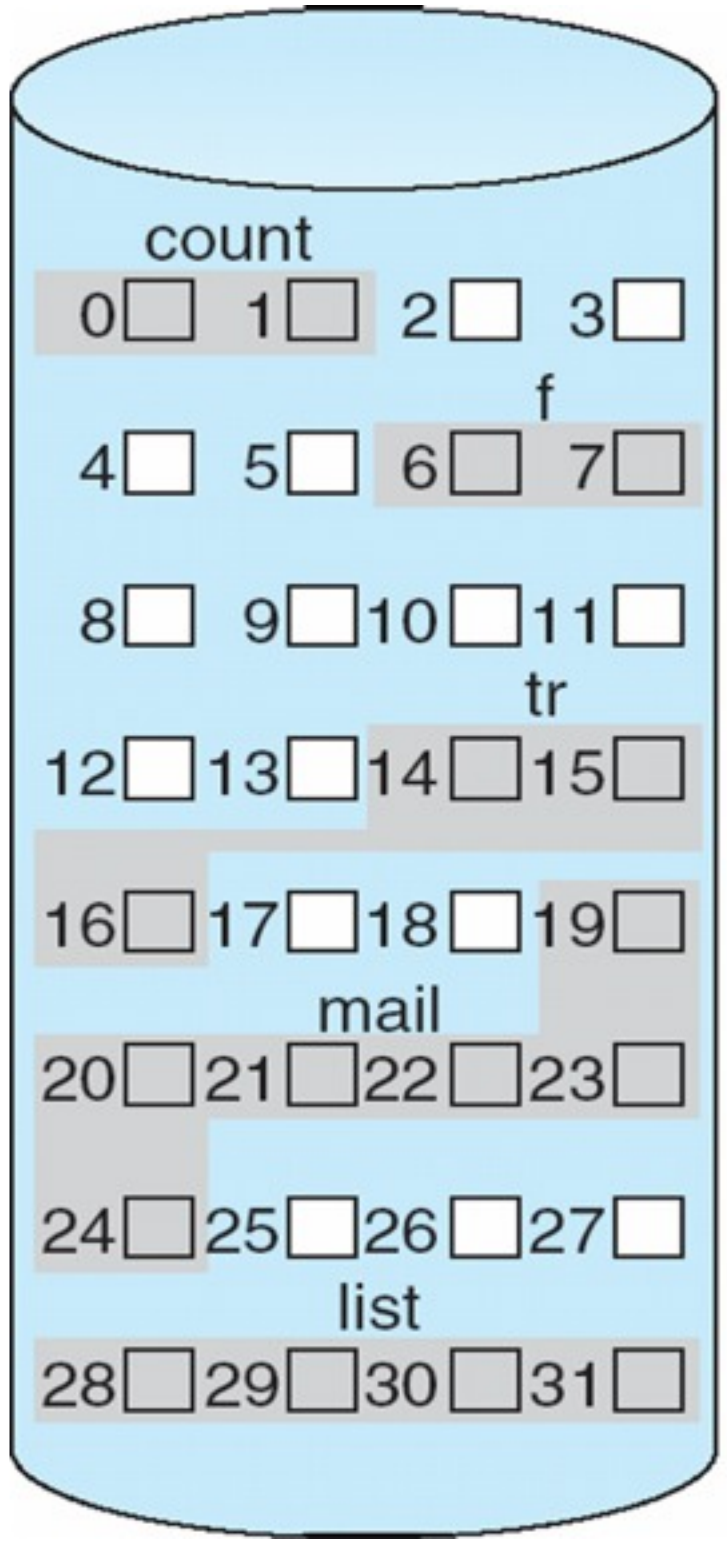
PE 10.1

Consider a file currently consisting of 100 blocks. Assume that the file-control block (and the index block, in the case of indexed allocation) is already in memory. Calculate the number of disk I/O operations are required for contiguous, linked, and indexed (single-level) allocation strategies, if, for one block,the following conditions hold. In the contiguous allocation-case, assume that there is no room to grow at the beginning but there is room to grow at the end. Also assume that the block information to be added is stored in memory.

- The block is added at the beginning
- The block is added in the middle
- The block is added at the end
- The block is removed from the beginning
- The block is removed from the middle
- The block is removed from the end

Possible strategy: Develop a Formula

Contiguous Allocation of Disk Space



directory

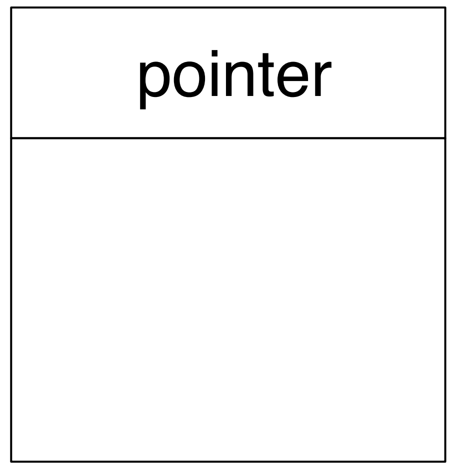
file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Linked Allocation

- In pure linked design, there isn't a FAT
- Pointers to blocks stored directly on disk

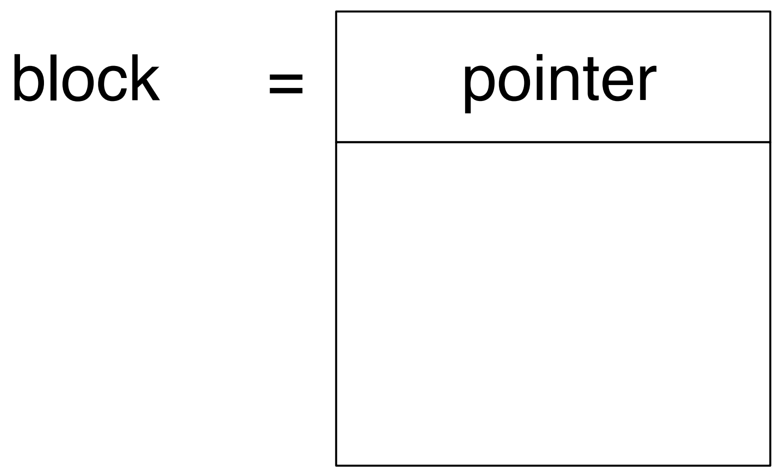
Linked Allocation

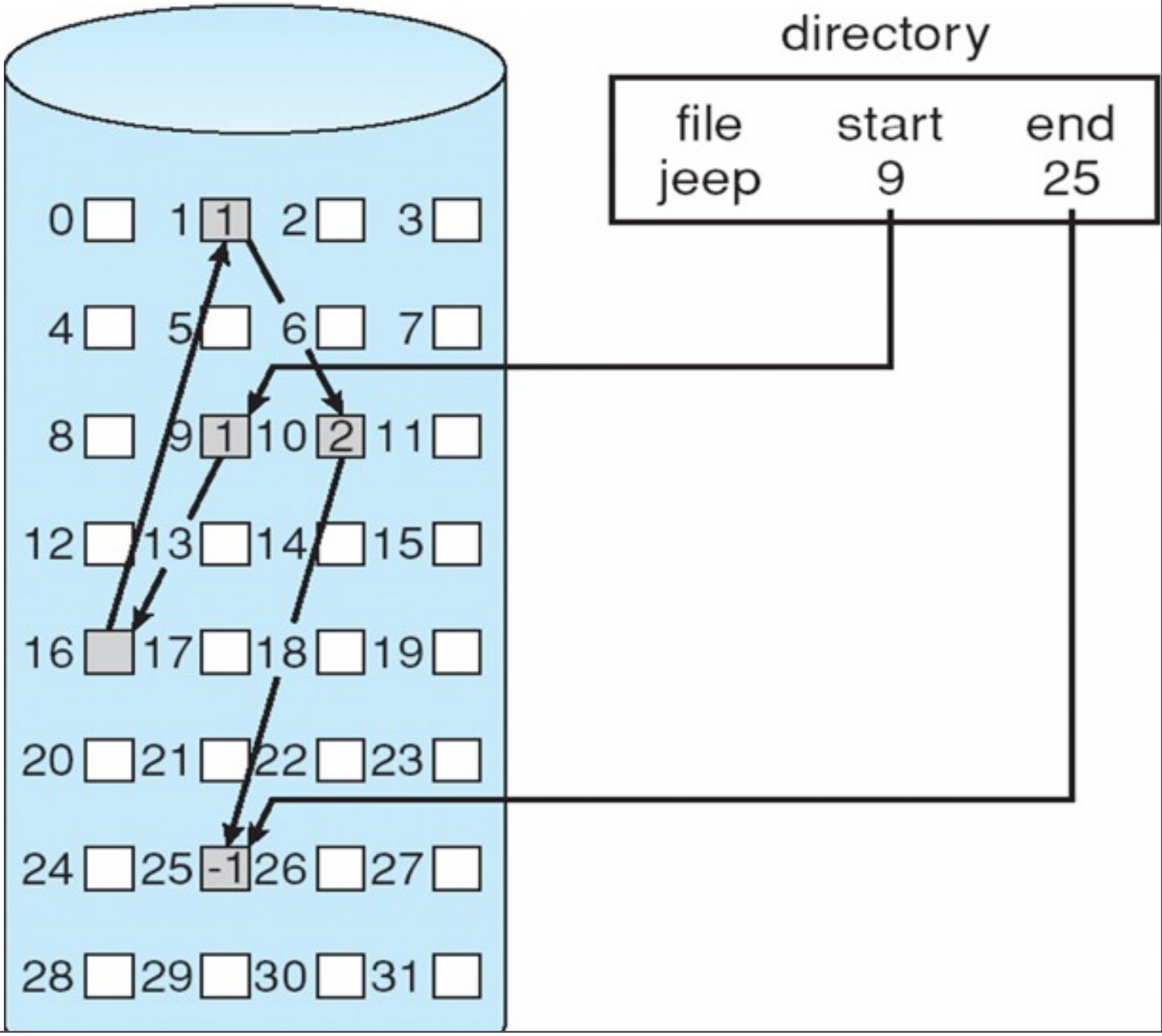
- Each file is a linked list of disk blocks: blocks may be scattered anywhere on the disk.

block = 

Linked Allocation

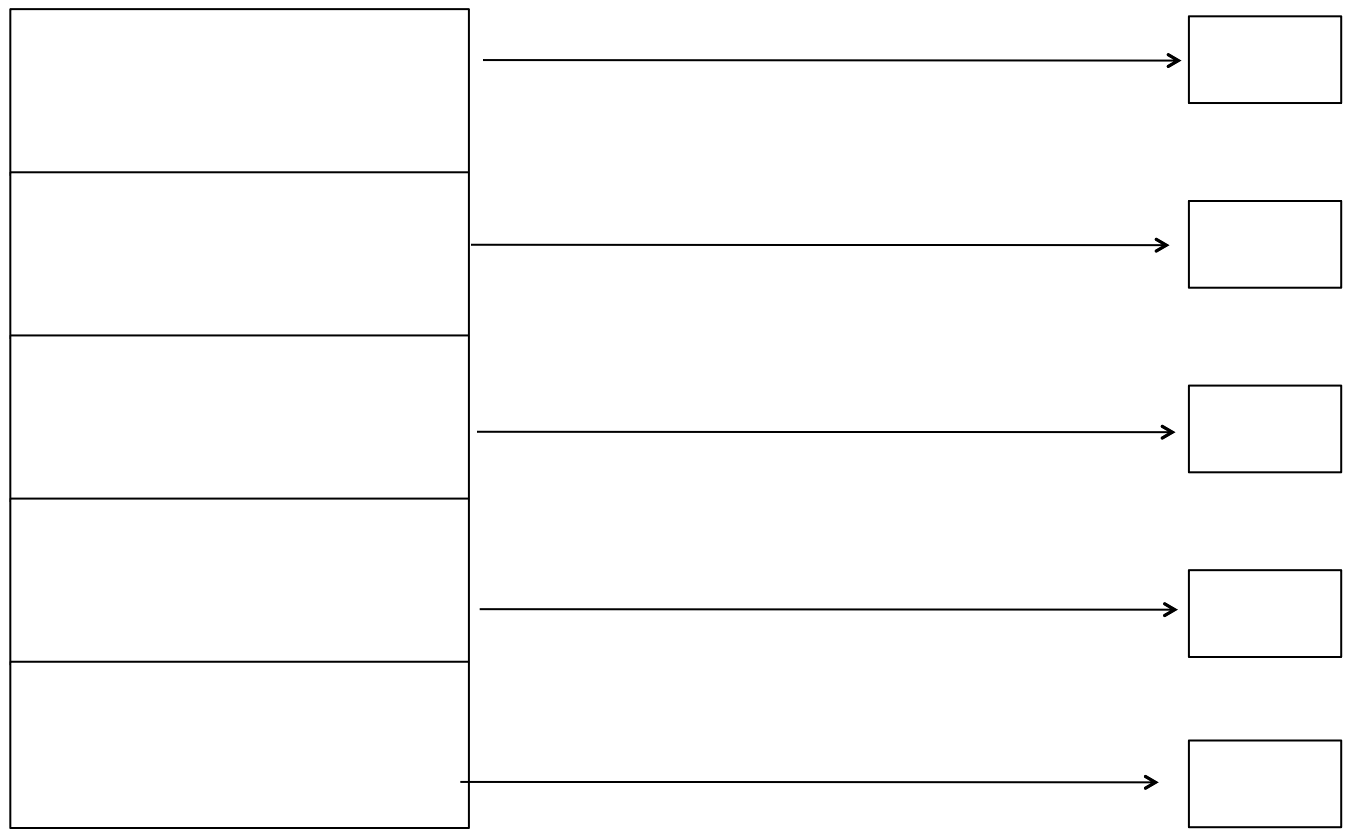
- Each file is a linked list of disk blocks: blocks may be scattered anywhere on the disk.





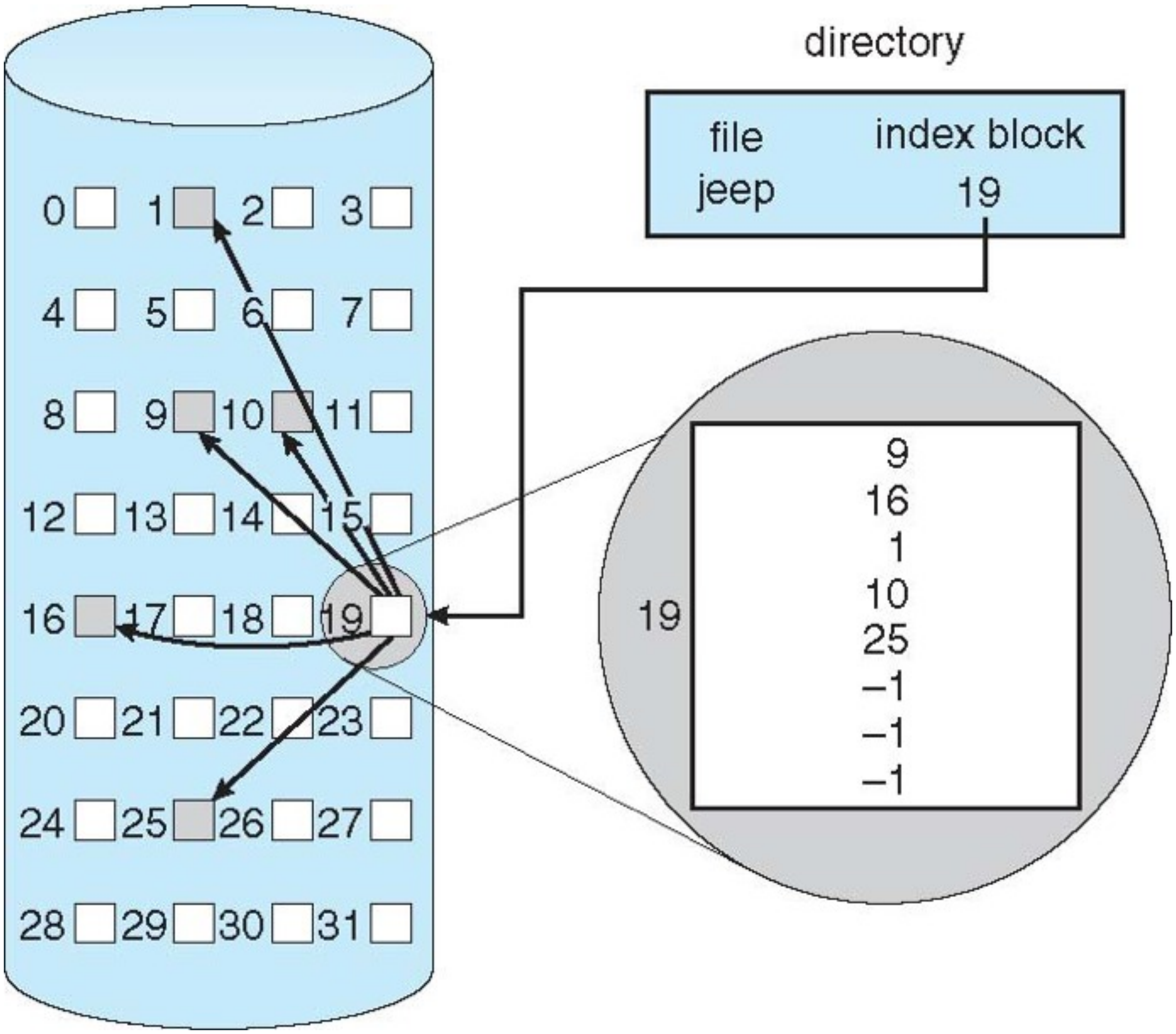
Indexed Allocation

- Brings all pointers together into the **index block**
- Logical view



index table

Example of Indexed Allocation



directory
file
index block
jeep
19
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
19
9
16
1
10
25
–1
–1
–1