CSE 4/521 Introduction to Operating Systems

Lecture 23 – File System Implementation II

(Allocation Methods, Free-Space Management)

Summer 2018

Overview

Objective:

- To discuss how the disk is managed for a file storage.
- To discuss how the disk is managed for free space.

- Allocation Methods
- Free-Space Management

Recap

- File-System Structure
  - Layered File System
- File-System Implementation
  - On-disk data structure : (1.Boot control block, 2. Volume control block, 3. directory structure, 4. per-file File Control Block)
  - In-memory data structure : (1. Mount table, 2. In-memory directory structure, 3. per-process open-file table, 4. system-wide open-file table)
- Directory Implementation
  - Linear List, Hash Table

Questions

1. Among the 2 methods we studied, which is a better method to implement directory structure in File Systems? Why? (Easy)

2. What problems could occur if a system allowed a file system to be mounted simultaneously at more than one location? (Open-ended)

3. Why do systems not use more or larger caches if they are so useful in file systems? (Open-ended)

Overview

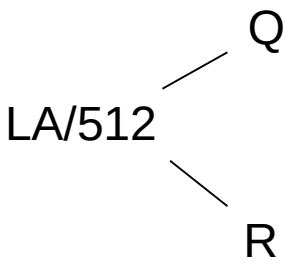
- Allocation Methods
- Free Space Management

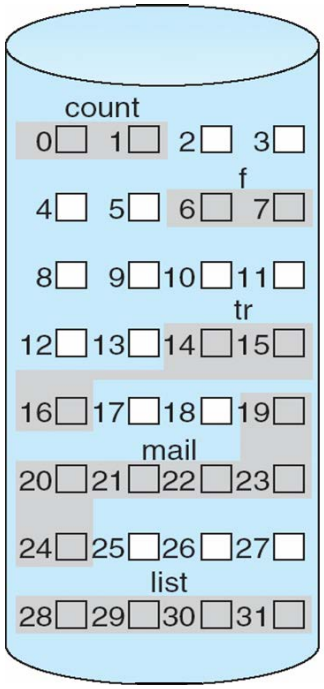
Allocation Methods : Contiguous

- An allocation method refers to how disk blocks are allocated for files:
- Contiguous allocation – each file occupies set of contiguous blocks
  - Best performance in most cases
  - Simple – only starting location (block #) and length (number of blocks) are required
  - Problems include finding space for file, knowing file size, external fragmentation, need for compaction off-line (downtime) or on-line

Allocation Methods : Contiguous Allocation

Mapping from logical to physical





directory

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

Block to be accessed = Q + starting address
Displacement into block = R

Allocation Methods : Contiguous Allocation (Extent-based Systems)

- Many newer file systems (i.e., Veritas File System) use a modified contiguous allocation scheme
- Extent-based file systems allocate disk blocks in extents
- An extent is a contiguous block of disks
  - Extents are allocated for file allocation
  - A file consists of one or more extents

Allocation Methods : Linked

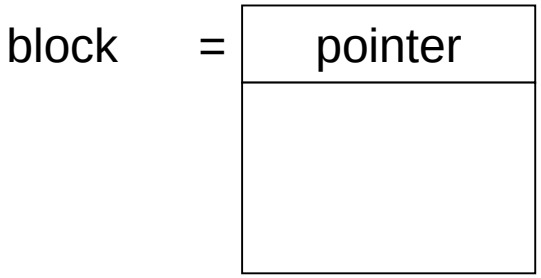
- Linked allocation – each file a linked list of blocks
  - File ends with null pointer
  - No external fragmentation
  - Each block contains pointer to next block
  - No compaction, external fragmentation
  - Free space management system called when new block needed
  - Improve efficiency by clustering blocks into groups but increases internal fragmentation
  - Reliability can be a problem
  - Locating a block can take many I/Os and disk seeks

Allocation Methods : Linked

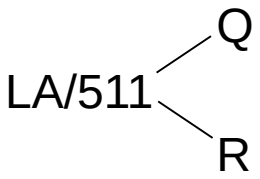
- FAT (File Allocation Table) variation
  - Beginning of volume has table, indexed by block number
  - Much like a linked list, but faster on disk and cacheable
  - New block allocation simple

Allocation Methods : Linked Allocation

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

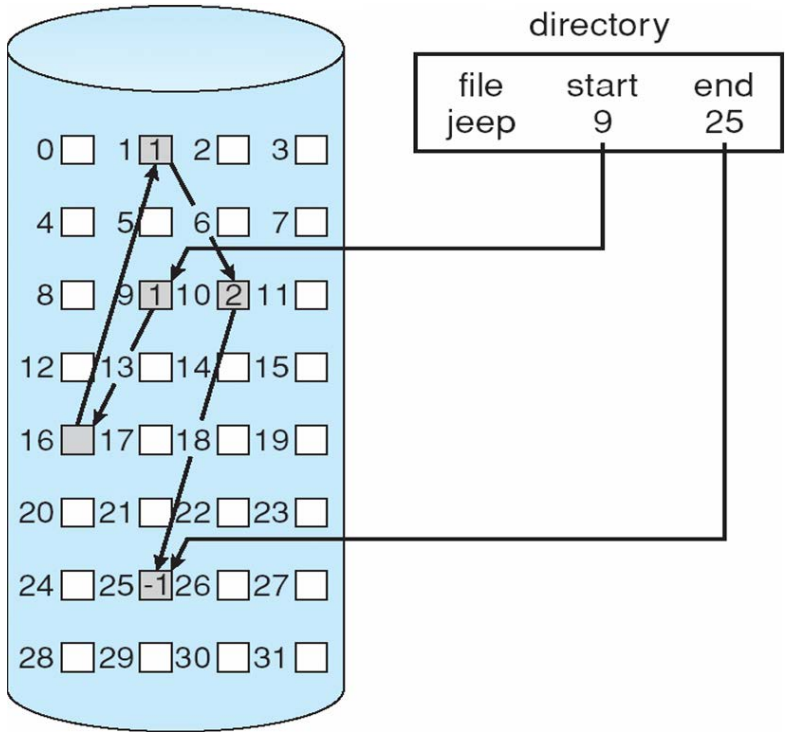


Mapping

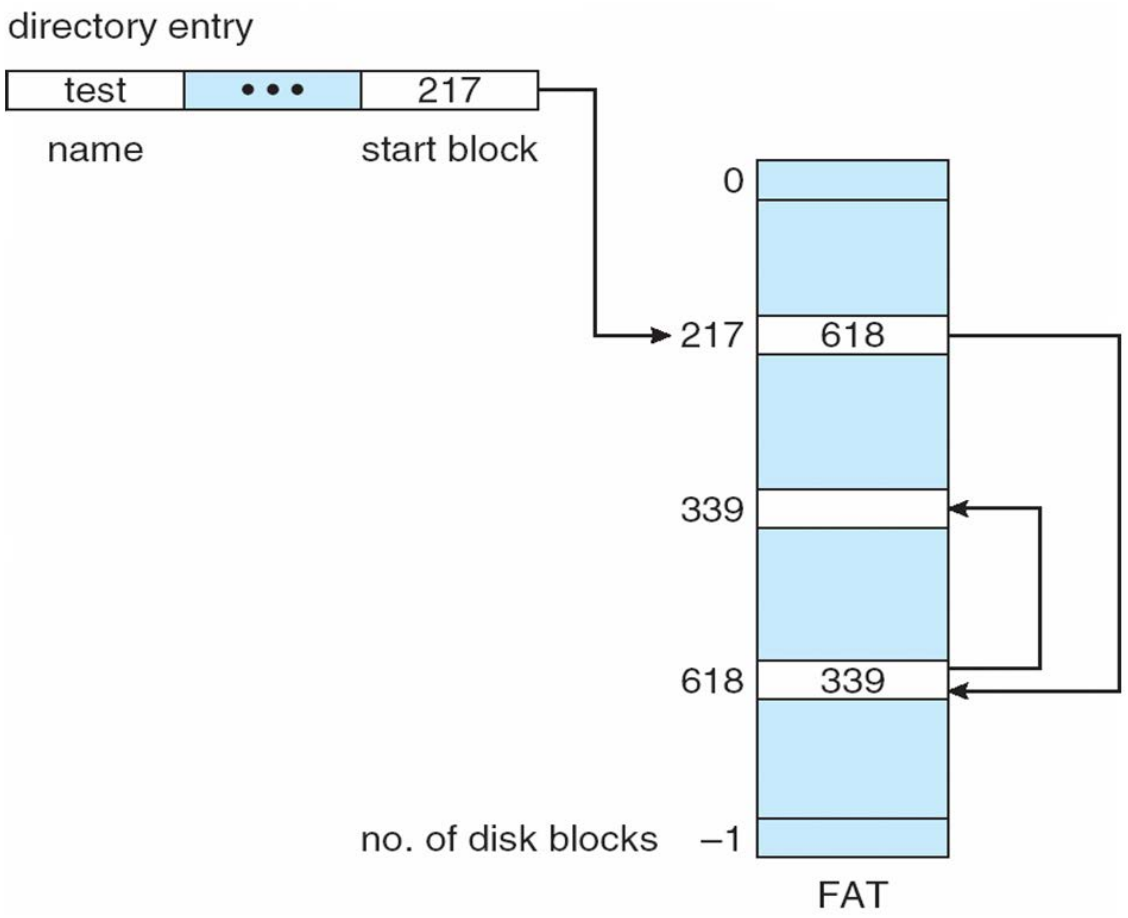


- Block to be accessed is the Qth block in the linked chain of blocks representing the file.
- Displacement into block = R + 1

Allocation Methods : Linked Allocation



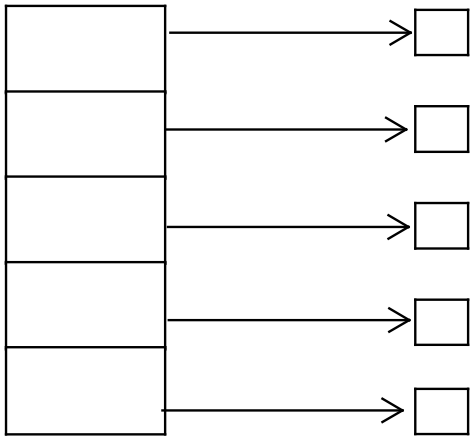
Allocation Methods : Linked Allocation (File-Allocation Table)



Allocation Methods : Indexed Allocation

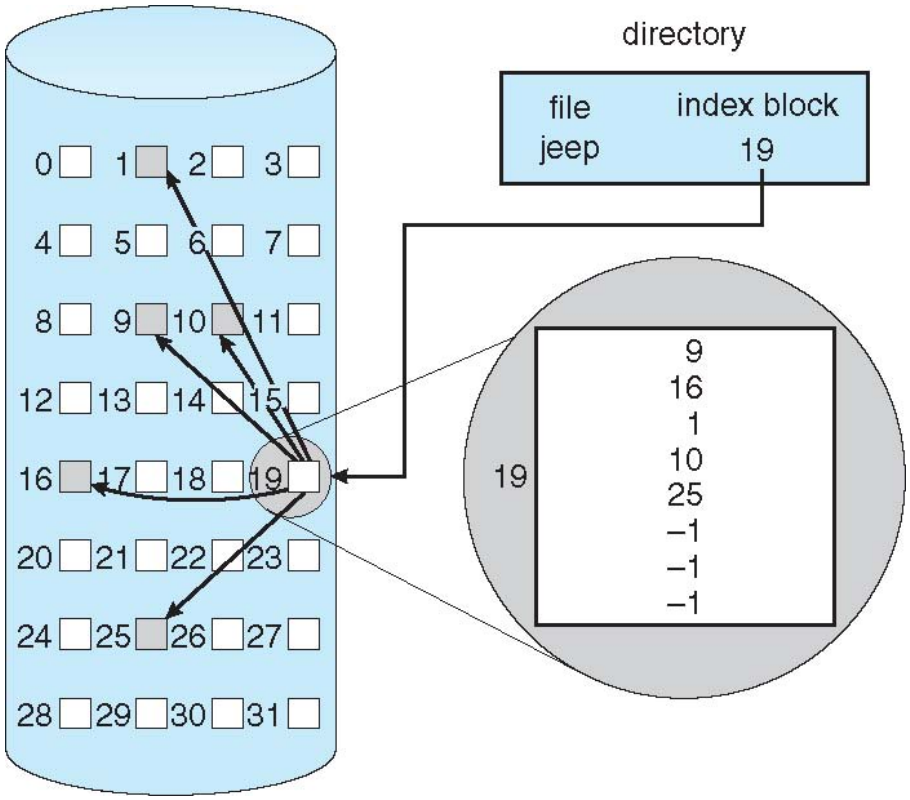
- Indexed allocation
  - Each file has its own index block(s) of pointers to its data blocks

- Logical view



index table

Allocation Methods : Example of Indexed Allocation

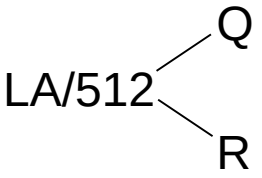


Allocation Methods : Indexed Allocation

- Need index table
- Random access
- Dynamic access without external fragmentation, but have overhead of index block
- Mapping from logical to physical in a file of maximum size of 256K bytes and block size of 512 bytes. We need only 1 block for index table

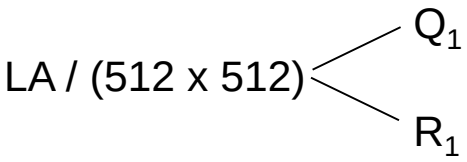
Q = displacement into index table

R = displacement into block

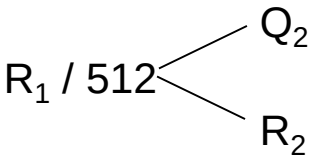


Allocation Methods : Indexed Allocation - Mapping

- Two-level index (4K blocks could store 1,024 four-byte pointers in outer index -> 1,048,567 data blocks and file size of up to 4GB)

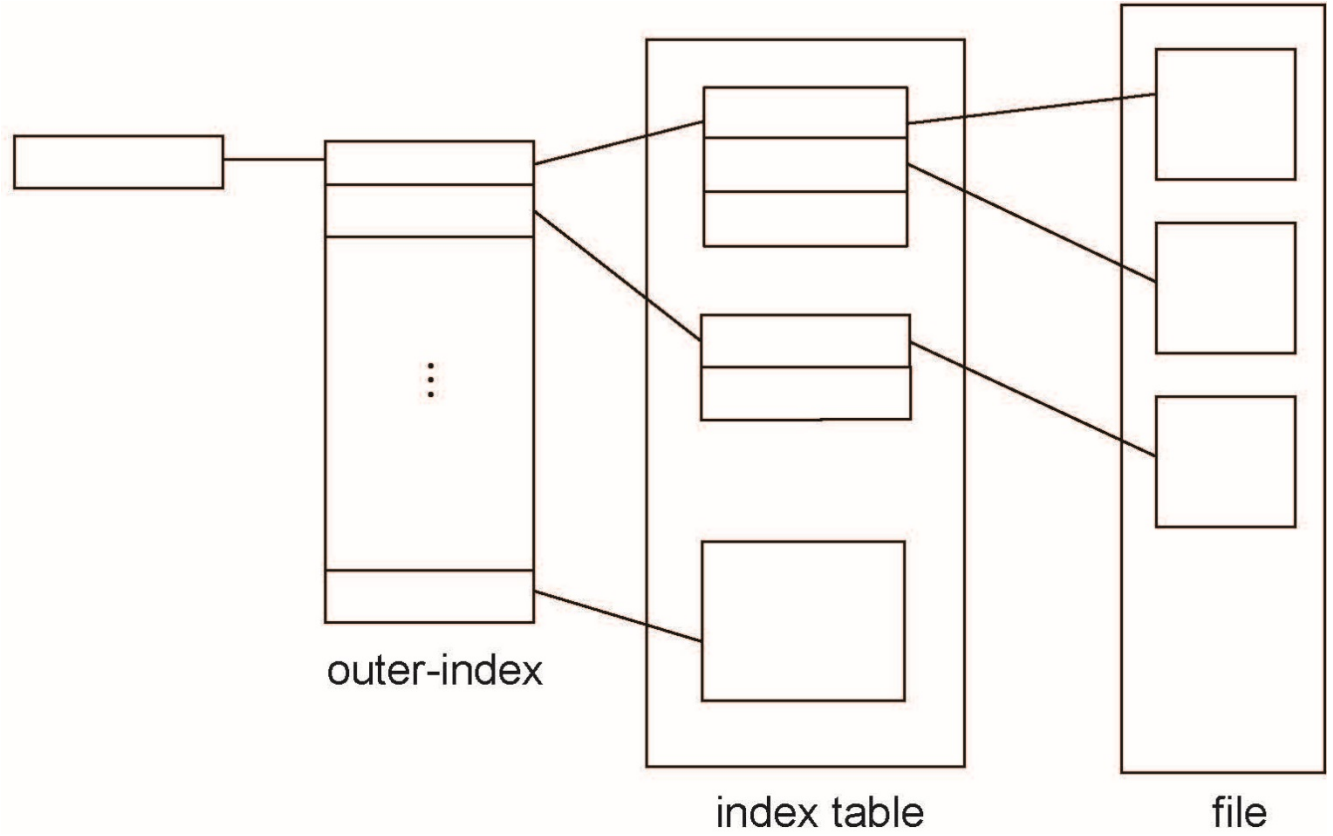


Q_1 = displacement into outer-index
R_1 is used as follows:



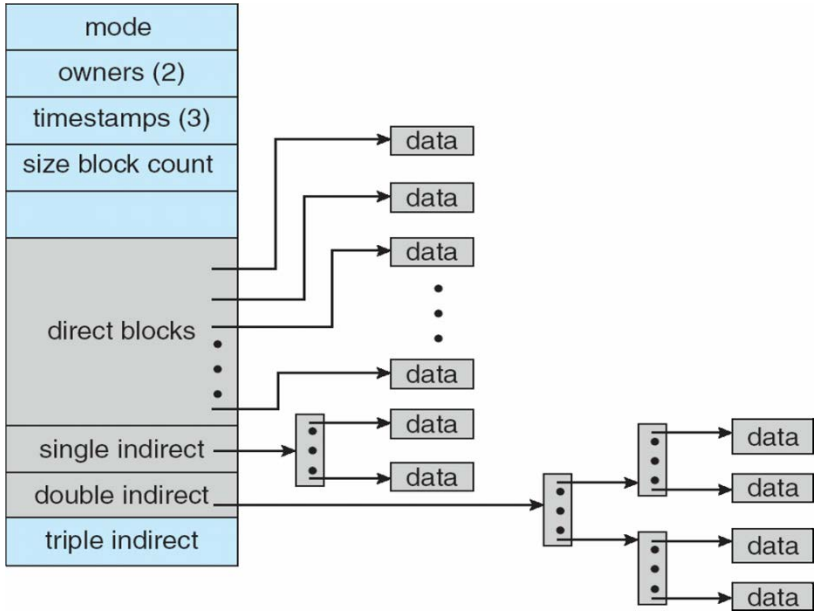
Q_2 = displacement into block of index table
R_2 displacement into block of file:

Allocation Methods : Indexed Allocation – Mapping



Allocation Methods : Combining Schemes

4K bytes per block, 32-bit addresses



More index blocks than can be addressed with 64-bit file pointer

Allocation Methods : Performance

- Best method depends on file access type
  - Contiguous great for sequential and random
  - Linked good for sequential, not random
- Declare access type at creation -> select either contiguous or linked
- Indexed more complex
  - Single block access could require 2 index block reads then data block read
  - Clustering can help improve throughput, reduce CPU overhead

Allocation Methods : Performance

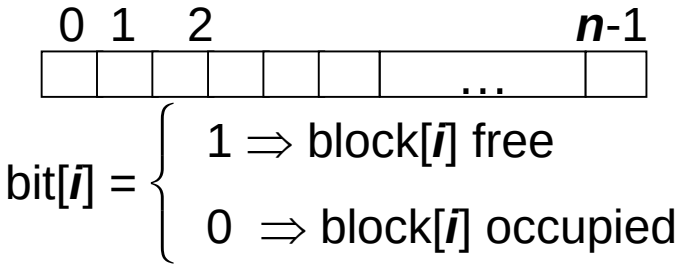
- Adding instructions to the execution path to save one disk I/O is reasonable
  - Intel Core i7 Extreme Edition 990x (2011) at 3.46Ghz = 159,000 MIPS
    - http://en.wikipedia.org/wiki/Instructions_per_second
  - Typical disk drive at 250 I/Os per second
    - 159,000 MIPS / 250 = 630 million instructions during one disk I/O
  - Fast SSD drives provide 60,000 I/Os per second
    - 159,000 MIPS / 60,000 = 2.65 millions instructions during one disk I/O

Overview

- Allocation Methods
- Free Space Management

Free-Space Management

- File system maintains free-space list to track available blocks/clusters
  - (Using term "block" for simplicity)
- Bit vector or bit map (***n*** blocks)



Block number calculation

(number of bits per word) *
(number of 0-value words) +
offset of first 1 bit

CPUs have instructions to return offset within word of first "1" bit

Free-Space Management

- Bit map requires extra space
  - Example:

    block size = 4KB = 2^{12} bytes

    disk size = 2^{40} bytes (1 terabyte)

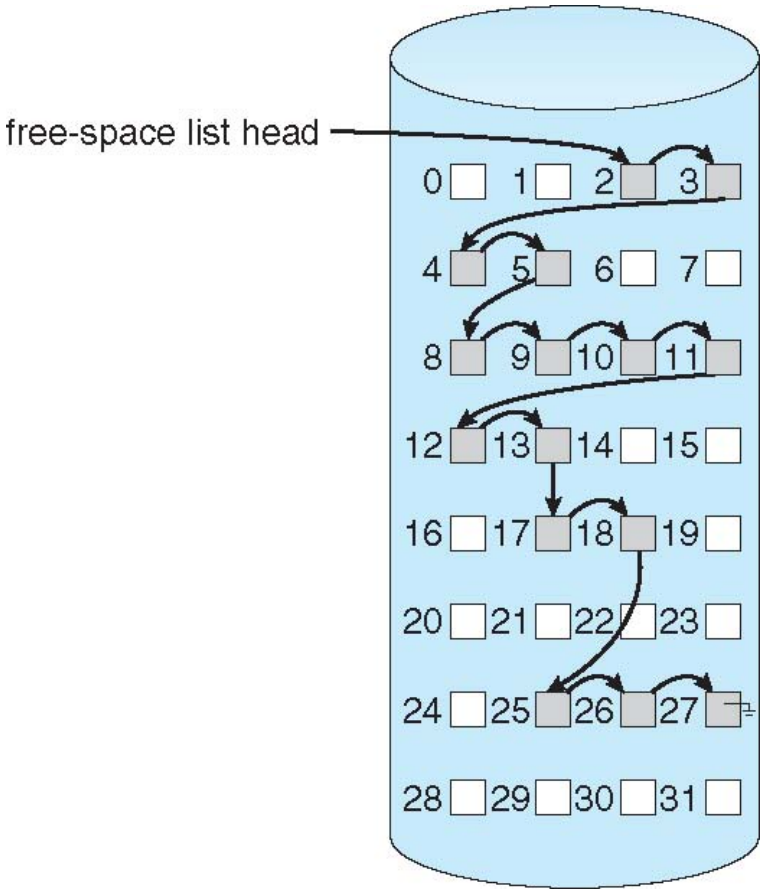
    $n = 2^{40}/2^{12} = 2^{28}$ bits (or 32MB)

    if clusters of 4 blocks -> 8MB of memory

- Easy to get contiguous files

Free-Space Management : Linked Free Space List on Disk

- Linked list (free list)
  - Cannot get contiguous space easily
  - No waste of space
  - No need to traverse the entire list (if # free blocks recorded)



Free-Space Management

- Grouping
  - Modify linked list to store address of next *n-1* free blocks in first free block, plus a pointer to next block that contains free-block-pointers (like this one)
- Counting
  - Because space is frequently contiguously used and freed, with contiguous-allocation allocation, extents, or clustering
    - Keep address of first free block and count of following free blocks
    - Free space list then has entries containing addresses and counts

Free-Space Management

- Space Maps
  - Used in ZFS
  - Consider meta-data I/O on very large file systems
    - Full data structures like bit maps couldn't fit in memory -> thousands of I/Os
  - Divides device space into metaslab units and manages metaslabs
    - Given volume can contain hundreds of metaslabs
  - Each metaslab has associated space map
    - Uses counting algorithm
  - But records to log file rather than file system
    - Log of all block activity, in time order, in counting format
  - Metaslab activity -> load space map into memory in balanced-tree structure, indexed by offset
    - Replay log into that structure
    - Combine contiguous free blocks into single entry

Credits for slides

Silberschatz, Galvin and Gagne