Tamkang Universty Software Engineering Group
淡 江 軟 體 工 程 實 驗 室
http://www.tkse.tku.edu.tw/

Operating Systems (2)

教材： **Operating System Concepts Abraham Silberschatz** 8th Edition
開發圖書代理

授課教師：王英宏

Presented by：Ying-Hong Wang
E-mail：inhon@mail.tku.edu.tw http://mail.tku.edu.tw/inhon
Date：2011/5/22

上課用書、參考用書暨相關規定

- 成績評定
  - **出席: 15% 實習課: 20% 作業+程式: 30% 期中考: 15% 期末考: 20%**
  - 點名成績係用以區隔同學每週出席與否的成績，故不接受各種形式未出席的原因，但每人可以有三次的緩衝機會，以因應不可抗拒的情況。缺席超過三次者，才開始扣分，反之，缺席少於三次者，可以獲得加分。
  - 每週點名一次，點名超過一次者，即為加分點名，不在前述100%內累計。正規點名可以接受補點，加分點名則不接受補點
  - 期末考比校訂時間提前一週
  - 指派聆聽演講之出席，每場加一分(採外加計分)
  - 除隨堂作業與上機驗收作業外，其餘作業可接受補交，自繳交截止時間點起，每24小時內補交者扣10分，扣至0分為止
  - 請助教作業每次發還後隨即公佈登錄資料以供核對，有疑義者須於一週內完成補正，逾期不再受理
- 上課方式
  - 板書為主、投影片相輔
  - 隨堂作業與小考，隨堂作業為主
  - 隨時自備A4紙
- 上課規定
  - 手機請改設震動或關機、不要私下講話
  - 鼓勵提問

Contents

- Recall Process, Memory, Virtual Memory Management
- Storage Management
  - File Systems
  - Implementing File Systems
  - Secondary Storage Structure
  - I/O Systems
- Distributed Systems
  - Distributed System Structures
- Parallel System
- Real-Time System

實習課部份

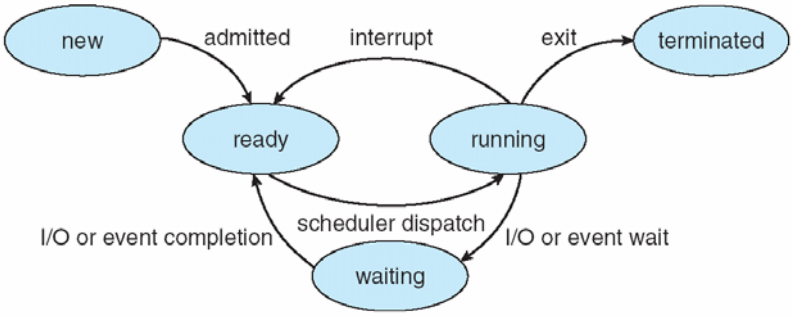
•Protection and Security

•Linux System

•Android

Recall Process Management

- Process States and Transformation



Recall Process Management

- Process Scheduling
  - Long Term Scheduling (From Disk to Memory)
  - Medial Term Scheduling (From Memory to Disk)
  - Short Term Scheduling (CPU Scheduling)
    - Preemptive vs. Non-Preemptive
    - Scheduling Algorithms
      - First Come First Service (FCFS)
      - Shortest Job First (SJF)
      - Priority
      - Round Robin (RR)
      - Multilevel Queue
      - Multilevel Feedback Queue

Recall Process Management

- Process Synchronization
  - Critical Section Problem
    - Peterson's Solution (Pure Software solution)
    - Hardware Solution
    - Semaphores (System Call)
    - Monitors (Central Control)

Recall Process Management

- Deadlock Problem
  - Necessary Conditions
    - Mutual exclusion
    - Hold and Wait
    - No Preemptive
    - Circular Wait
  - Methods and Handling
    - Deadlock Prevention
    - Deadlock Avoidance
    - Deadlock Detection
    - Recovery from Deadlock

Recall Memory Management

- Memory Allocation
  - First-Fit
  - Best-Fit
  - Worst Fit
- Concept of Fragmentation
  - Internal Fragmentation
  - External Fragmentation

Recall Memory Management

- Paging
- Segmentation
- Hybrid of Paging and Segmentation
  - One Segmentation has some Pages

Recall Virtual Memory Management

- Focus on Paging Technique
- Page Replacement
- Methods and Handling
  - FIFO
  - Optimal
  - LRU (Least Recently Used)
  - LRU Approximation
    - Additional-Reference-Bit
    - Second-Chance
    - Enhanced Second-Chance
  - Counting
    - LFU (Least-Frequently-Used)
    - MFU (Most Frequently-Used)

Recall Virtual Memory Management

- Trashing
- Working Set Model
- Buddy System
- Slab Allocation

Storage Management

File System – Chapter 10

- File Concept
- Access Methods
- Directory Structure
- File-System Mounting
- File Sharing
- Protection

Objectives

- To explain the function of file systems
- To describe the interfaces to file systems
- To discuss file-system design tradeoffs, including access methods, file sharing, file locking, and directory structures
- To explore file-system protection

File Concept

- Contiguous logical address space
- Types:
  - Data
    - numeric
    - character
    - binary
  - Program

File Structure

- None - sequence of words, bytes
- Simple record structure
  - Lines
  - Fixed length
  - Variable length
- Complex Structures
  - Formatted document
  - Re-locatable load file
- Can simulate last two with first method by inserting appropriate control characters
- Who decides:
  - Operating system
  - Program

File Attributes

- **Name** – only information kept in human-readable form
- **Identifier** – unique tag (number) identifies file within file system
- **Type** – needed for systems that support different types
- **Location** – pointer to file location on device
- **Size** – current file size
- **Protection** – controls who can do reading, writing, executing
- **Time, date, and user identification** – data for protection, security, and usage monitoring
- Information about files are kept in the directory structure, which is maintained on the disk

File Operations

- File is an **abstract data type**
- **Create**
- **Write**
- **Read**
- **Reposition within file**
- **Delete**
- **Truncate**
- *Open*(F_i) – search the directory structure on disk for entry F_i, and move the content of entry to memory
- *Close* (F_i) – move the content of entry F_i in memory to directory structure on disk

Open Files

- Several pieces of data are needed to manage open files:
  - **File pointer**: pointer to last read/write location, per process that has the file open
  - **File-open count**: counter of number of times a file is open – to allow removal of data from open-file table when last processes closes it
  - **Disk location of the file**: cache of data access information
  - **Access rights**: per-process access mode information

Open File Locking

- Provided by some operating systems and file systems
- Mediates access to a file
- Mandatory or advisory:
  - **Mandatory** – access is denied depending on locks held and requested
  - **Advisory** – processes can find status of locks and decide what to do

File Locking Example – Java API

```
import java.io.*;
import java.nio.channels.*;
public class LockingExample {
    public static final boolean EXCLUSIVE = false;
    public static final boolean SHARED = true;
    public static void main(String arsg[]) throws IOException {
        FileLock sharedLock = null;
        FileLock exclusiveLock = null;
        try {
            RandomAccessFile raf = new RandomAccessFile("file.txt", "rw");
            // get the channel for the file
            FileChannel ch = raf.getChannel();
            // this locks the first half of the file - exclusive
            exclusiveLock = ch.lock(0, raf.length()/2, EXCLUSIVE);
            /** Now modify the data . . . */
            // release the lock
            exclusiveLock.release();
```


File Locking Example – Java API (cont)

```
            // this locks the second half of the file - shared
            sharedLock = ch.lock(raf.length()/2+1, raf.length(),
                    SHARED);
            /** Now read the data . . . */
            // release the lock
            sharedLock.release();
        } catch (java.io.IOException ioe) {
            System.err.println(ioe);
        }finally {
            if (exclusiveLock != null)
            exclusiveLock.release();
            if (sharedLock != null)
            sharedLock.release();
        }
    }
}
```

File Types – Name, Extension

file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

Access Methods

- **Sequential Access**

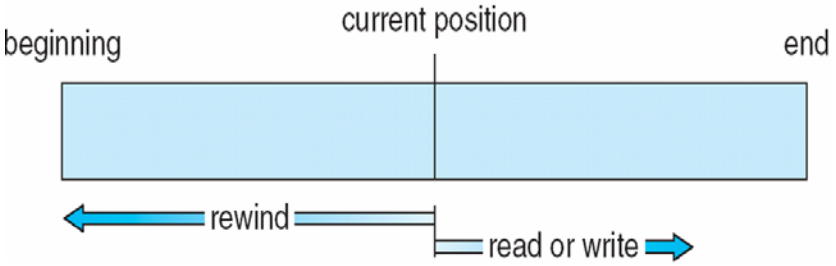
  read next
  write next
  reset
  no read after last write
  (rewrite)

- **Direct Access**

  read *n*
  write *n*
  position to *n*
  read next
  write next
  rewrite *n*

  n = relative block number

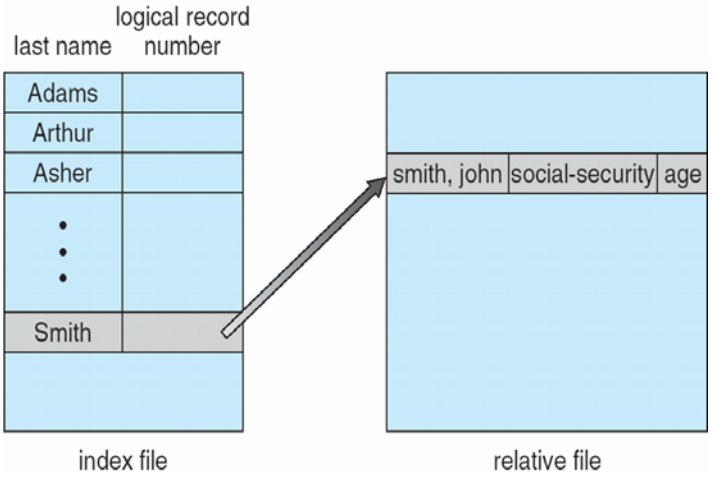
Sequential-access File



Simulation of Sequential Access on Direct-access File

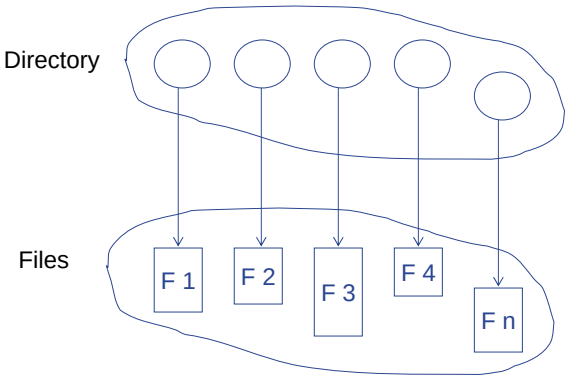
sequential access	implementation for direct access
reset	$cp = 0$;
read next	*read cp*; $cp = cp + 1$;
write next	*write cp*; $cp = cp + 1$;

Example of Index and Relative Files



Directory Structure

- A collection of nodes containing information about all files

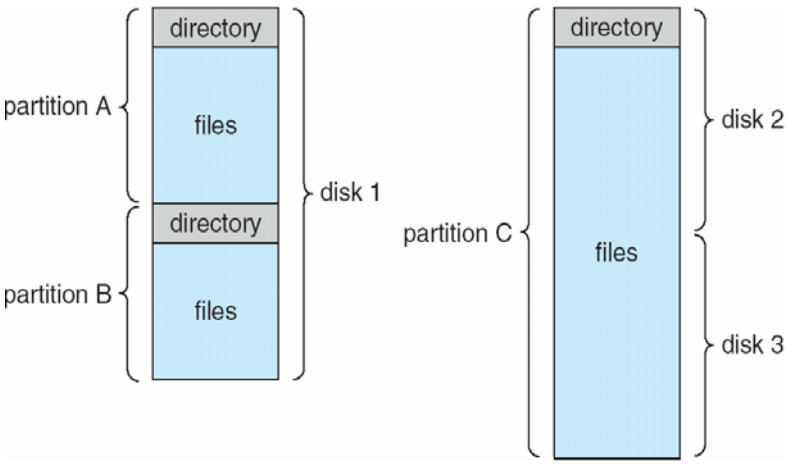


Both the directory structure and the files reside on disk
Backups of these two structures are kept on tapes

Disk Structure

- Disk can be subdivided into partitions
- Disks or partitions can be RAID protected against failure
- Disk or partition can be used raw – without a file system, or formatted with a file system
- Partitions also known as minidisks (IBM world)), slices
- Entity containing file system known as a **volume**
- Each volume containing file system also tracks that file system's info in **device directory** (simply as **directory**) or **volume table of contents**
- As well as general-purpose file systems there are many special-purpose file systems, frequently all within the same operating system or computer

A Typical File-system Organization



Storage Structure

- In the Solaris example
  - tmpfs: Temporary file system
  - objfs: Virtual file system
  - ctfs: virtual file system maintains “Contract” info.
  - lofs: Loop back file system
  - procfs: Processes file system
  - ufs, zfs: General-purpose file system
  - Others.

Operations Performed on Directory

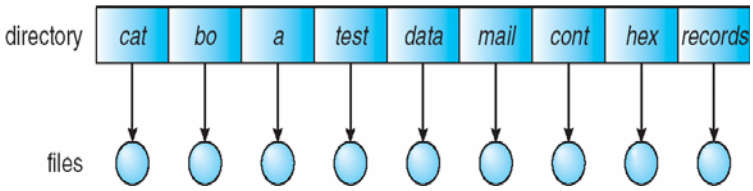
- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file
- Traverse the file system

Organize the Directory (Logically) to Obtain

- Efficiency – locating a file quickly
- Naming – convenient to users
  - Two users can have same name for different files
  - The same file can have several different names
- Grouping – logical grouping of files by properties, (e.g., all Java programs, all games, …)

Single-Level Directory

- A single directory for all users

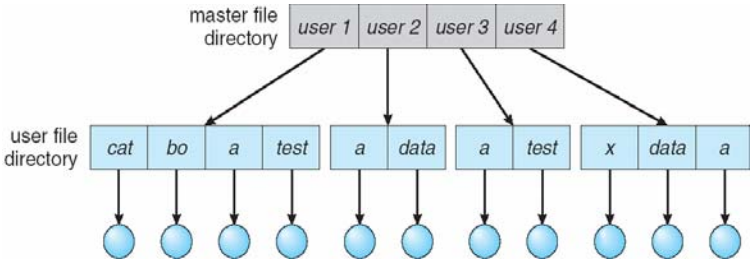


Naming problem

Grouping problem

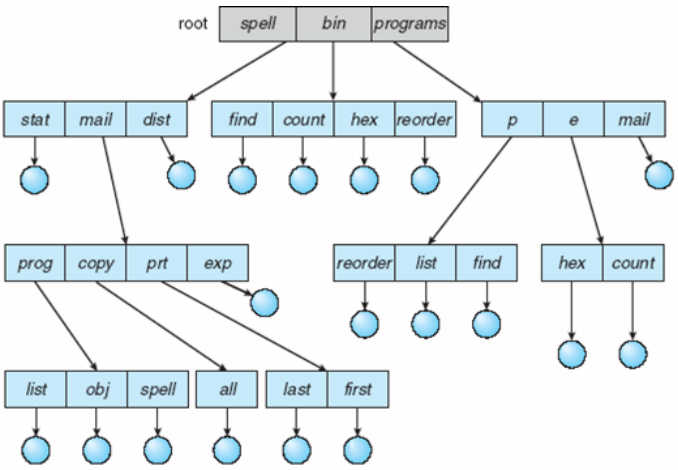
Two-Level Directory

- Separate directory for each user



- Path name
- Can have the same file name for different user
- Efficient searching
- No grouping capability

Tree-Structured Directories



Tree-Structured Directories (Cont)

- Efficient searching
- Grouping Capability
- Current directory (working directory)
  - cd /spell/mail/prog
  - type list

Tree-Structured Directories (Cont)

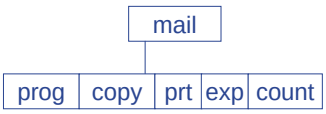
- **Absolute** or **relative** path name
- Creating a new file is done in current directory
- Delete a file

  rm <file-name>
- Creating a new subdirectory is done in current directory

  mkdir <dir-name>

  Example: if in current directory /mail

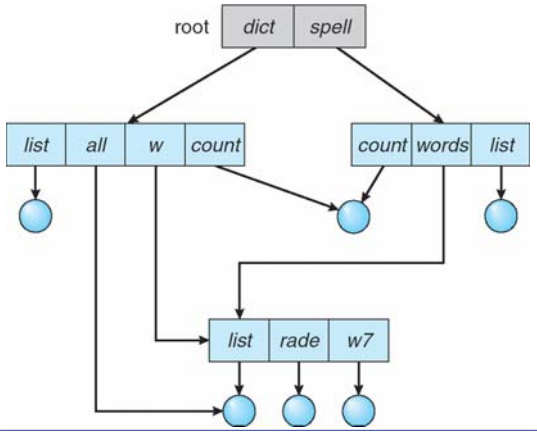
  mkdir count



Deleting “mail” ⇒ deleting the entire subtree rooted by “mail”

Acyclic-Graph Directories

- Have shared subdirectories and files

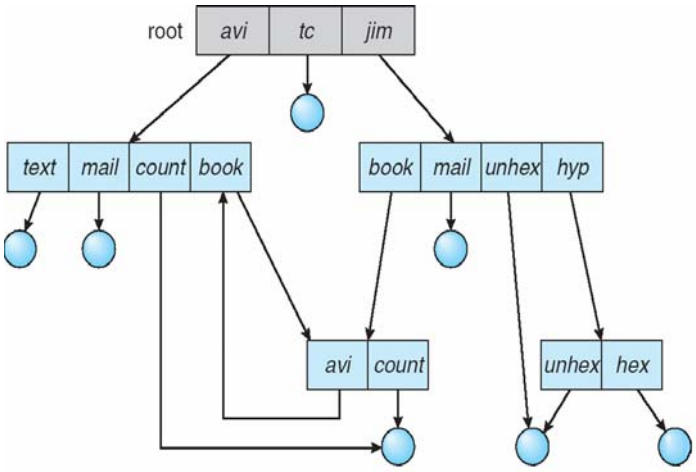


Acyclic-Graph Directories (Cont.)

- Two different names (aliasing)
- If *dict* deletes *list* ⇒ dangling pointer

  Solutions:
  - Backpointers, so we can delete all pointers
    Variable size records a problem
  - Backpointers using a daisy chain organization
  - Entry-hold-count solution
- New directory entry type
  - **Link** – another name (pointer) to an existing file
  - **Resolve the link** – follow pointer to locate the file

General Graph Directory



General Graph Directory (Cont.)

- How do we guarantee no cycles?
  - Allow only links to file not subdirectories
  - Garbage collection
  - Every time a new link is added use a cycle detection algorithm to determine whether it is OK

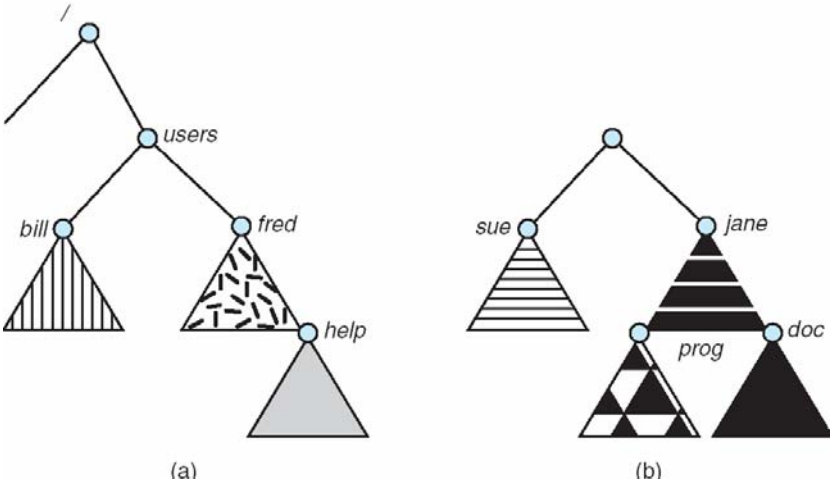
File System Mounting

- A file system must be **mounted** before it can be accessed
- A unmounted file system is mounted at a **mount point**

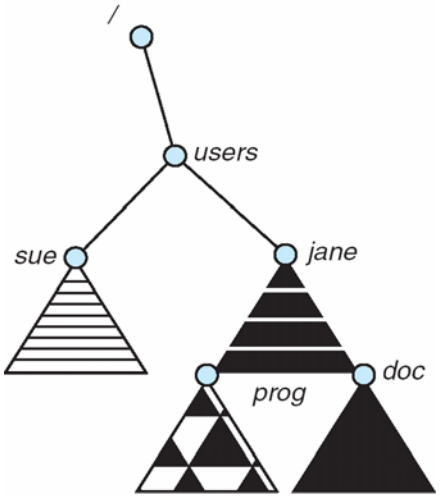
File System Mounting

- Mount Procedure:
  - Give OS the Name of the device and the Mount Point
    - Mount point: the location within the file structure
  - OS verifies that the device contains a valid file system
  - OS notes in its directory structure that a file system is mounted at the specified mount point.

(a) Existing. (b) Unmounted Partition



Mount Point



File Sharing

- Sharing of files on multi-user systems is desirable
- Sharing may be done through a **protection** scheme
- On distributed systems, files may be shared across a network
- Network File System (NFS) is a common distributed file-sharing method

File Sharing – Multiple Users

- **User IDs** identify users, allowing permissions and protections to be per-user
- **Group IDs** allow users to be in groups, permitting group access rights

File Sharing – Remote File Systems

- Uses networking to allow file system access between systems
  - Manually via programs like FTP
  - Automatically, seamlessly using **distributed file systems**
  - Semi automatically via the **world wide web**
- **Client-server** model allows clients to mount remote file systems from servers
  - Server can serve multiple clients
  - Client and user-on-client identification is insecure or complicated
  - **NFS** (Network File System) is standard UNIX client-server file sharing protocol
  - **CIFS** (Common Internet File System) is standard Windows protocol
  - Standard operating system file calls are translated into remote calls

File Sharing – Remote File Systems

- Distributed Information Systems
  - Called Also Distributed Naming Services
  - Solve the problems of Client-Server Model
  - **DNS** (Domain Name System) provides host-name-to-network-address translation for the entire Internet
  - Sun applies **NIS** (Network Information Service) mechanism
  - Microsoft uses **CIFS** as the protocol
  - Software Industry is moving **LDAP** (Lightweight Directory-Access Protocol) as a Secure distributed naming mechanism.
  - Active Directory implement unified access to information needed for remote computing

File Sharing – Failure Modes

- File systems can fail for a variety of reasons:
  - Disk which contains the file system is failure
  - Corruption of the directory structure
  - Corruption of the disk-management information
  - Disk-controller failure
  - Cable failure
  - Host-adapter failure

File Sharing – Failure Modes

- Remote file systems add new failure modes, due to network failure, server failure
- Recovery from failure can involve state information about status of each remote request
- **Stateless protocols** such as NFS include all information in each request, allowing easy recovery but less security

File Sharing – Consistency Semantics

- **Consistency semantics** specify how multiple users are to access a shared file simultaneously
  - Similar to process synchronization algorithms
    - Tend to be less complex due to disk I/O and network latency (for remote file systems
  - Andrew File System (AFS) implemented complex remote file sharing semantics
  - Unix file system (UFS) implements:
    - Writes to an open file visible immediately to other users of the same open file
    - Sharing file pointer to allow multiple users to read and write concurrently
  - AFS has session semantics
    - Writes only visible to sessions starting after the file is closed

Protection

- File owner/creator should be able to control:
  - what can be done
  - by whom
- Types of access
  - **Read**
  - **Write**
  - **Execute**
  - **Append**
  - **Delete**
  - **List**

Access Lists and Groups

- Mode of access: read, write, execute
- Three classes of users

			RWX
a) **owner access**	7	⇒	1 1 1
			RWX
b) **group access**	6	⇒	1 1 0
			RWX
c) **public access**	1	⇒	0 0 1

- Ask manager to create a group (unique name), say G, and add some users to the group.
- For a particular file (say *game*) or subdirectory, define an appropriate access.

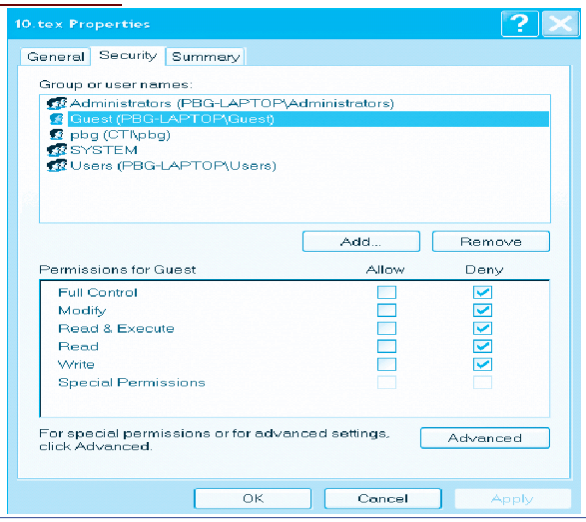
owner group public

chmod 761 game

Attach a group to a file

chgrp G game

Windows XP Access-control List Management



A Sample UNIX Directory Listing

-rw-rw-r--	1 pbg	staff	31200	Sep 3 08:30	intro.ps
drwx------	5 pbg	staff	512	Jul 8 09.33	private/
drwxrwxr-x	2 pbg	staff	512	Jul 8 09:35	doc/
drwxrwx---	2 pbg	student	512	Aug 3 14:13	student-proj/
-rw-r--r--	1 pbg	staff	9423	Feb 24 2003	program.c
-rwxr-xr-x	1 pbg	staff	20471	Feb 24 2003	program
drwx--x--x	4 pbg	faculty	512	Jul 31 10:31	lib/
drwx------	3 pbg	staff	1024	Aug 29 06:52	mail/
drwxrwxrwx	3 pbg	staff	512	Jul 8 09:35	test/

Implementing File Systems – Chapter 11

- File-System Structure
- File-System Implementation
- Directory Implementation
- Allocation Methods
- Free-Space Management

File-System Structure

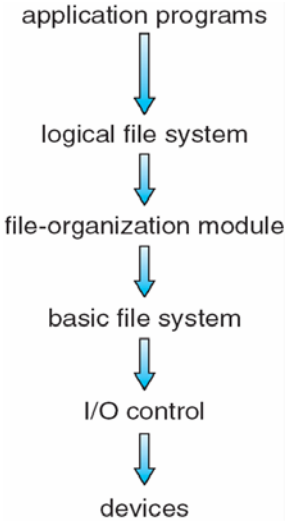
- File structure
  - Logical storage unit
  - Collection of related information
- File system resides on secondary storage (disks)
- There are two characteristics that make disk a convenient medium for storing multiple files
  - A disk can be rewritten in place
  - A disk can access directly any block of information

TSEG Tamkang Universty Software Engineering Group 淡江軟體工程實驗室 http://www.tkse.tku.edu.tw/

File-System Structure

- A file system poses two quite different design problems:
  - Define how the file system should look to the user
  - Create algorithms and data structures to map the logical file system onto the physical secondary-storage devices
- File system organized into layers
  - I/O Control
  - Basic File System
  - File-Organization Module
  - Logical File System

Layered File System



File-System Structure

- I/O Control
  - Consists of Device Drivers and Interrupt Handlers
- Basic File System
  - Needs only to issue generic commands to the appropriate device driver
- File-Organization Module
  - Knows about files and their logical blocks as well as physical blocks
- Logical File System
  - Manages metadata information

File-System Structure

- **File control block** – storage structure consisting of information about a file
  - Maintained by Logical File System
  - An inode in most UNIX file systems
  - The information including
    - Ownership
    - Permissions
    - Location of the file contents
    - Protection and Security

A Typical File Control Block (FCB)

file permissions
file dates (create, access, write)
file owner, group, ACL
file size
file data blocks or pointers to file data blocks

File-System Implementation

- On-Disk and In-Memory structures
- Information contained by On-Disk
  - A boot control block per volume
  - A volume control block per volume
  - A directory structure per file system
  - A per-file FCB

TSEG Tamkang University Software Engineering Group 淡江軟體工程實驗室 http://www.tkse.tku.edu.tw/

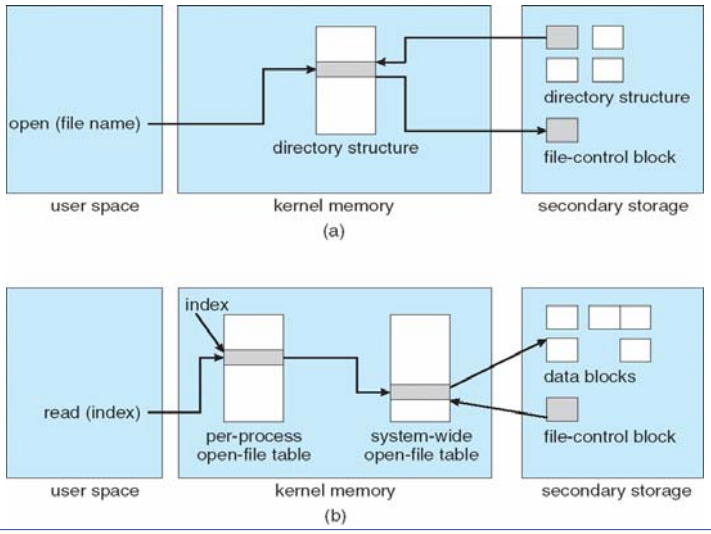
File-System Implementation

- Information contained by In-Memory
  - An in-memory mount table
  - An in-memory directory-structure cache
  - The system-wide open-file table
  - The per-process open-file table
  - Buffers

In-Memory File System Structures

- The following figure illustrates the necessary file system structures provided by the operating systems.
- Figure (a) refers to opening a file.
- Figure (b) refers to reading a file.

In-Memory File System Structures



Partitions and Mounting

- Partitions (or Volume) vs. Disks
- Status of Partition
  - Raw
    - Containing no file system
  - Cooked
    - Containing a file system

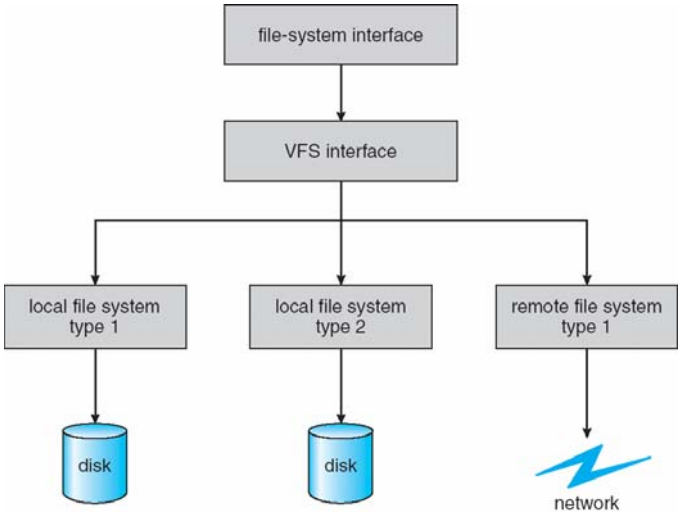
Virtual File Systems

- Virtual File Systems (VFS) provide an object-oriented way of implementing file systems.
- VFS allows the same system call interface (the API) to be used for different types of file systems.
- The API is to the VFS interface, rather than any specific type of file system.

Virtual File Systems

- The file-system implementation consists of three major layers:
  - File-System Interface
    - based on the open(), read(), write(), and close() call and on file descriptor.
  - Virtual File System (VFS)
  - Local file system or Remote file system

Schematic View of VFS



Virtual File Systems

- Virtual File System (VFS) serves Two important functions:
  - Separate file-system-generic operations from their implementation by defining a clean VFS interface
  - Provides a mechanism for uniquely representing a file throughout a network

Directory Implementation

- **Linear List** of file names with pointer to the data blocks.
  - simple to program
  - time-consuming to execute
- **Hash Table** – linear list with hash data structure.
  - decreases directory search time
  - **collisions** – situations where two file names hash to the same location
  - fixed size

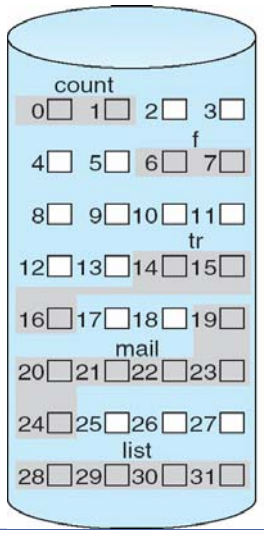
Allocation Methods

- Objectives
  - Disk Space is utilized effectively
  - Files can be accessed quickly
- An allocation method refers to how disk blocks are allocated for files. There are three kinds of Allocation method:
  - **Contiguous allocation**
  - **Linked allocation**
  - **Indexed allocation**

Contiguous Allocation

- Each file occupies a set of contiguous blocks on the disk
- Simple – only starting location (block #) and length (number of blocks) are required
- Support to both of Sequential access and Random access
- Wasteful of space (dynamic storage-allocation problem)
- Suffer External Fragmentation
- Files cannot grow

Contiguous Allocation of Disk Space



directory

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

Extent-based Systems

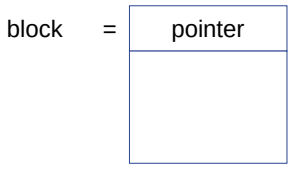
- Many newer file systems (I.e. Veritas File System) use a modified contiguous allocation scheme
- Extent-based file systems allocate disk blocks in **extents**
- An **extent** is a contiguous block of disks
  - Extents are allocated for file allocation
  - A file consists of one or more extents.

Extent-based Systems

- When a file is created newly, it is allocated one contiguous block, the first *Extent*.
- After the file is growing and the original Extent does not enough, file system can assign *Another* extent for it.
- The file location record is consisted by Address + Number of Extent + Link to next Extent.

Linked Allocation

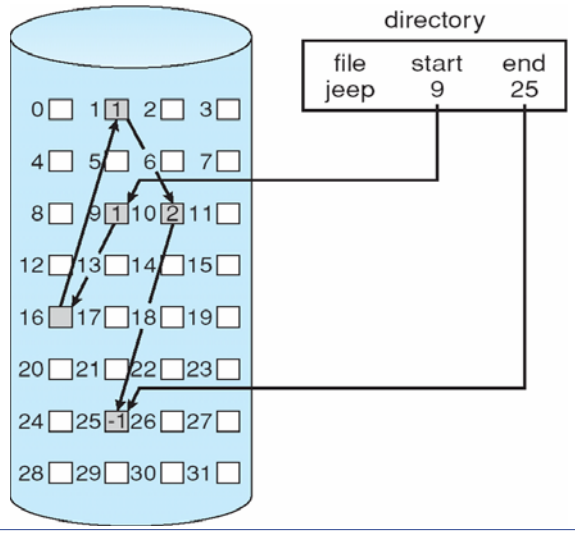
- Each file is a linked list of disk blocks: blocks may be scattered anywhere on the disk.



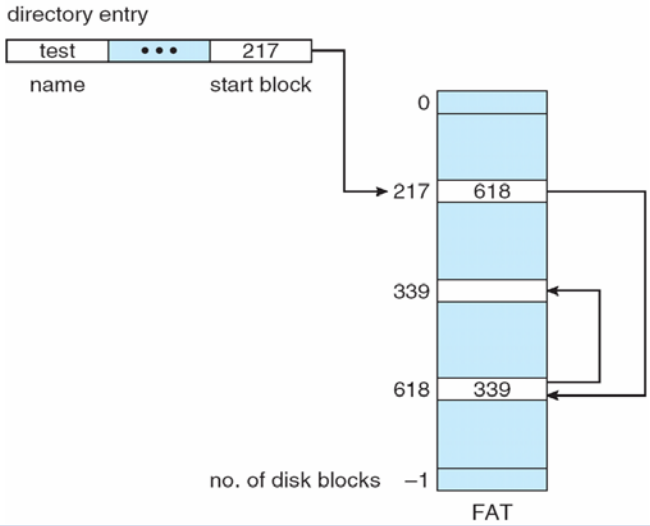
Linked Allocation

- Simple – need only starting address
- Free-space management system – no waste of space
- No random access
- The necessary space for pointer
- Reliability problem
- File-allocation table (FAT) – disk-space allocation used by MS-DOS and OS/2

Linked Allocation of Disk Space

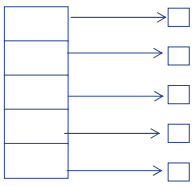


File Allocation Table (FAT)



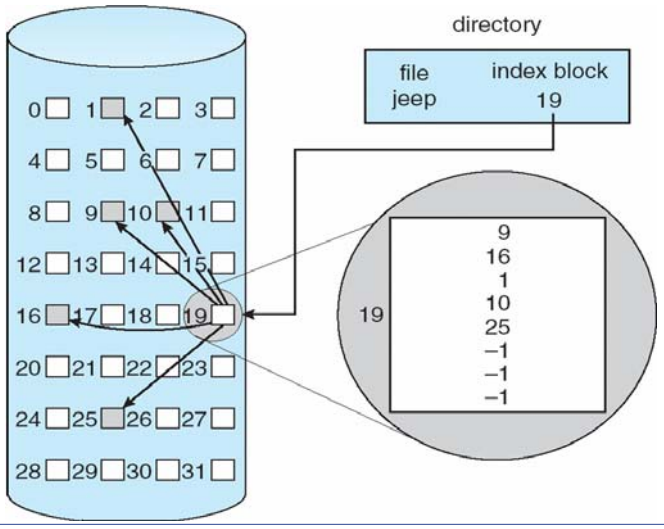
Indexed Allocation

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



index table

Indexed Allocation of Disk Space



Indexed Allocation

- Support Direct Access
- No External Fragmentation
- Need index table (waste space)
  - Regularly, one indexed block is larger than all pointer spaces
- Dynamic access without external fragmentation, but have overhead of index block.
- The issue of How large the Indexed block should be?
  - The problem of Larger
  - The problem of Smaller

Scheme of Mechanism

- A Mechanism to deal with this issue
- Scheme of the Mechanism
  - Linked scheme
  - Multilevel index scheme
  - Combined scheme

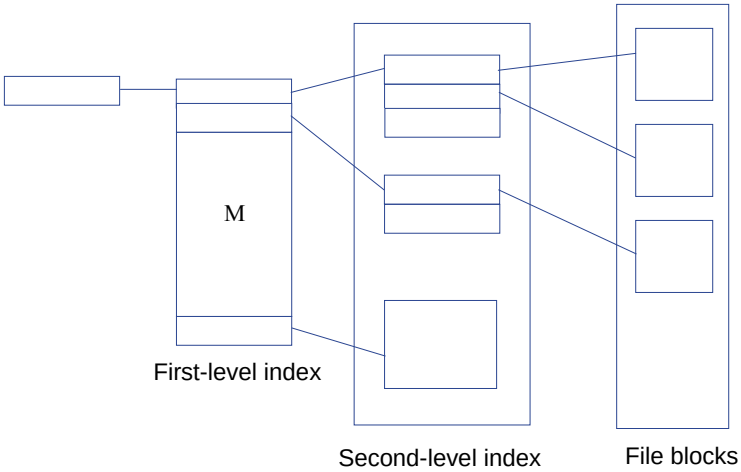
TSEG Tamkang Universty Software Engineering Group 淡江軟體工程實驗室 http://www.tkse.tku.edu.tw/

Scheme of Mechanism

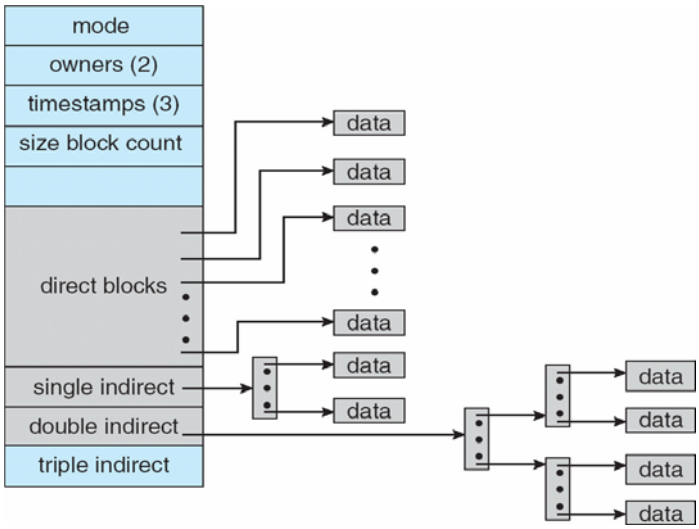
- Linked scheme
  - An index block is one disk block
  - Link several index blocks together for large files
- Multilevel Index scheme
  - First-level index block points to a set of Second-level index blocks
  - Which in turn point to file blocks
  - It can be continued to a third or fourth levels.
- Combined scheme
  - Keep first most of pointers to file blocks
  - Final several pointers to create Multilevel Index scheme

<inhon@mail.tku.edu.tw> May 22, 2011

Indexed Allocation - Mapping



Combined Scheme: UNIX (4k bytes per Block)



Free-Space Management

- Free Space List (but doesn't be implemented by List)
- Bit vector (n blocks) or called Bit maps

bit[i] = 678
- 0 ⇒ block[i] free
- 1 ⇒ block[i] occupied

Block number calculation

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

Free-Space Management

- Bit map requires extra space
  - Example:

    block size = 2^{12} bytes

    disk size = 2^{30} bytes (1 gigabyte)

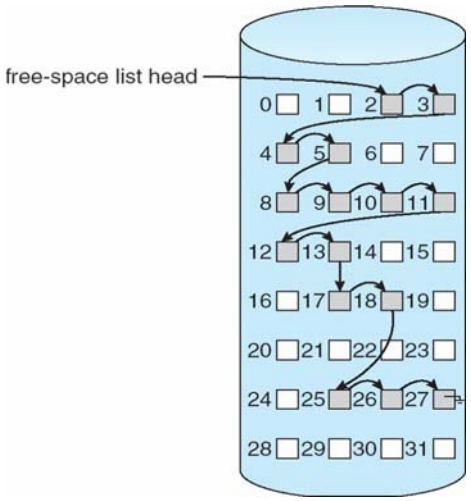
    $n = 2^{30}/2^{12} = 2^{18}$ bits (or 32K bytes)
- Inefficient to manage
- Easy to get contiguous files

Free-Space Management

- Linked List
  - Keeping a pointer to the first free block
  - Cannot get contiguous space easily
  - Inefficient for traversing on the list
  - No waste of space

Free-Space Management

- Linked List



Free-Space Management

- Grouping
  - Storing the address of *n* free blocks in the first free block
  - The first *n-1* of these blocks are actually free
  - The last block contains the addresses of another *n* free blocks
  - Advantage:
    - A large number of free blocks can be found quickly
  - Disadvantage:
    - Some free blocks are used to record the address of other free blocks

Free-Space Management

- Counting
  - Assume the most of free blocks are contiguous
  - Keep the address of the first free block and the number (n) of free contiguous blocks
  - Each **entry** in the free-space list consists of a disk address and a count value
  - All **entries** can be stored in a **B-Tree**, rather than a linked list

Free-Space Management

- Space maps
  - Apply by Sun's ZFS file system
  - Create **metalabs**
  - Each metalab has an associated Space Map
  - The space map is a log of all block activity (allocating and freeing)
  - The associated Space map is loaded into memory in a balanced-tree structure
  - The free-space list is updated on disk

Secondary Storage Structure – Chapter 12

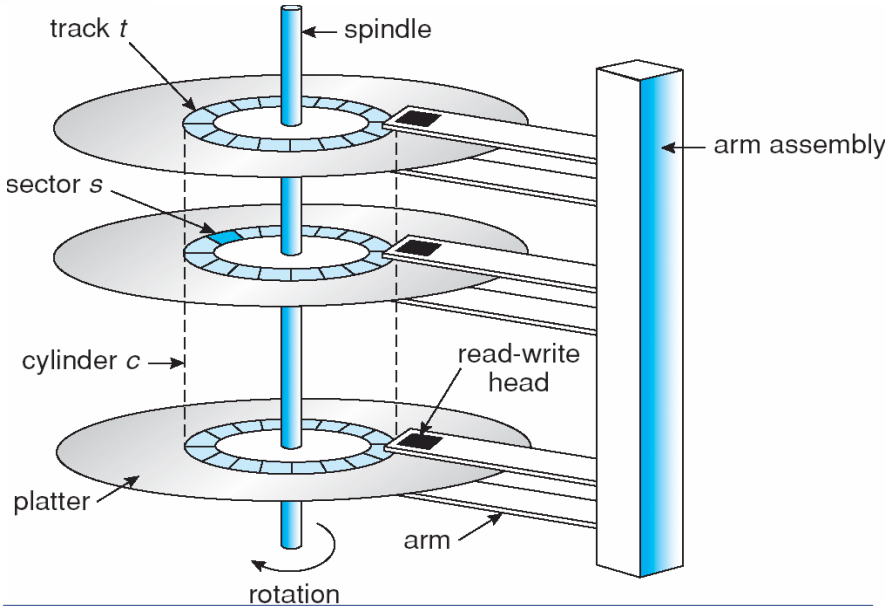
- Overview of Mass Storage Structure
- Disk Structure
- Disk Scheduling
- Swap-Space Management
- RAID Structure

Overview of Mass Storage Structure - Disk

- Magnetic disks provide bulk of secondary storage of modern computers
  - Drives rotate at 60 to 200 times per second
  - Disk Speed = Transfer rate + Random-Access time
  - **Transfer rate** is rate at which data flow between drive and computer
  - **Positioning time** (**random-access time**) is time to move disk arm to desired cylinder (**seek time**) and time for desired sector to rotate under the disk head (**rotational latency**)
  - **Head crash** results from disk head making contact with the disk surface
    - That's bad

Overview of Mass Storage Structure - Disk

- Disks can be removable
- Drive attached to computer via **I/O bus**
  - Busses vary, including **EIDE, ATA, SATA, USB, Fiber Channel, SCSI**
  - **Host controller** in computer uses bus to talk to **disk controller** built into drive or storage array



Disk Structure

- Disk drives are addressed as large 1-dimensional arrays of *logical blocks*, where the logical block is the smallest unit of transfer.
- The 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially.
  - Sector 0 is the first sector of the first track on the outermost cylinder.
  - Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost.

Disk Structure

- Disk drives are addressed as large 1-dimensional arrays of *logical blocks*, where the logical block is the smallest unit of transfer.
- The 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially.
  - Sector 0 is the first sector of the first track on the outermost cylinder.
  - Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost.

Disk Structure

- Where logical block means Physical block in Chapter 10, and Disk block in Chapter 11.
- Disk Address = Cylinder No. + Track No. + Sector No.
- Mapping between *Logical Block No.* and *Disk Address*

Disk Structure

- CLV
  - Constant Linear Velocity
  - The density of bits per track is uniform.
  - Thus, the farther a track is from the center of the disk, the greater its length, so more sectors it can hold.
  - Used in CD-ROM and DVD-ROM
- CAV
  - Constant Angular Velocity
  - The disk rotation speed can stay constant
  - The density of bits decreases from inner to outer tracks
  - Used in general Hard Disk

Disk Scheduling

- The operating system is responsible for using hardware efficiently — for the disk drives, this means having a fast access time and disk bandwidth.
- Access time has two major components
  - *Seek time* is the time for the disk are to move the heads to the cylinder containing the desired sector.
  - *Rotational latency* is the additional time waiting for the disk to rotate the desired sector to the disk head.
- Traditional concept
  - Access time = Seek time + Rotational Delay time + Transfer time

Disk Scheduling

- Minimize seek time
- Seek time ≈ seek distance
- Disk bandwidth is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer.
- ***Decreasing*** the totally seek time will *Increase* the **Performance** of Disk Access.

Disk Scheduling

- Several algorithms exist to schedule the servicing of disk I/O requests.
- We illustrate them with a request queue (0-199).

98, 183, 37, 122, 14, 124, 65, 67

Head pointer 53

Disk Scheduling

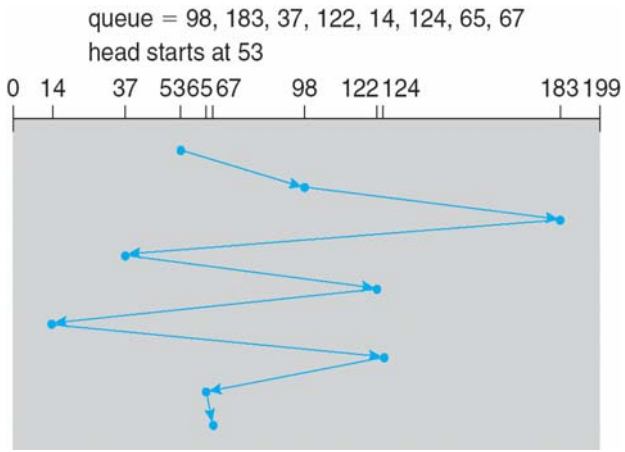
- Several algorithms exist to schedule the servicing of disk I/O requests.
- We illustrate them with a request queue (0-199).

98, 183, 37, 122, 14, 124, 65, 67

Head pointer 53

FCFS-First Come First Service

Illustration shows total head movement of 640 cylinders.

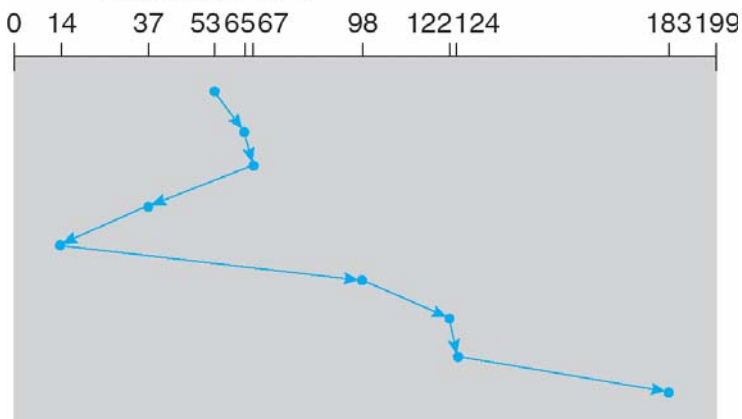


SSTF-Shortest Seek Time First

- Selects the request with the minimum seek time from the current head position.
- SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests.
- Illustration shows total head movement of 236 cylinders.

SSTF-Shortest Seek Time First

queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53



SCAN

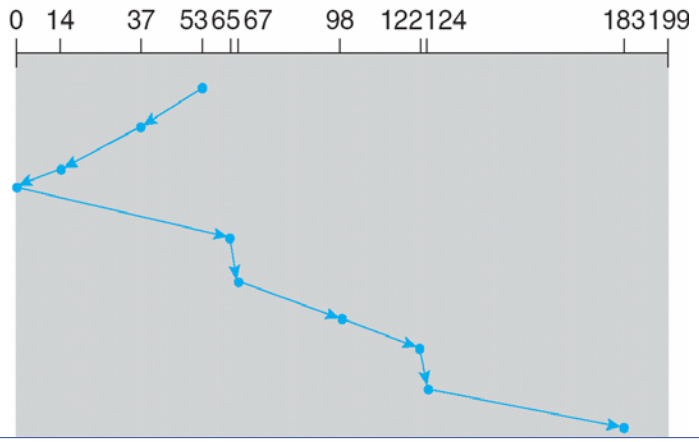
- The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues.
- Sometimes called the *elevator algorithm*.
- Illustration shows total head movement of 208 cylinders.

TSEG Tamkang University Software Engineering Group 淡江軟體工程實驗室 http://www.tkse.tku.edu.tw/

SCAN

queue = 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53



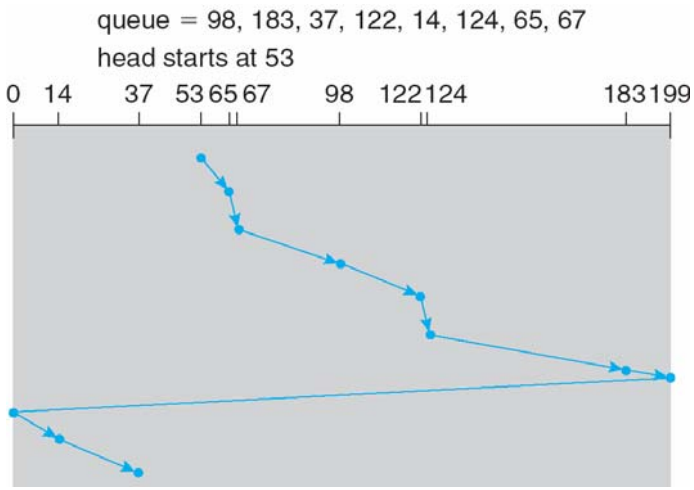
<inhon@mail.tku.edu.tw> May 22, 2011

C-SCAN

- Circular SCAN
- Provides a more uniform wait time than SCAN.
- The head moves from one end of the disk to the other. servicing requests as it goes. When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip.
- Treats the cylinders as a circular list that wraps around from the last cylinder to the first one.
- Illustration shows total head movement of 382 cylinders.

TSEG Tamkang Universty Software Engineering Group 淡江軟體工程實驗室 http://www.tkse.tku.edu.tw/

C-SCAN

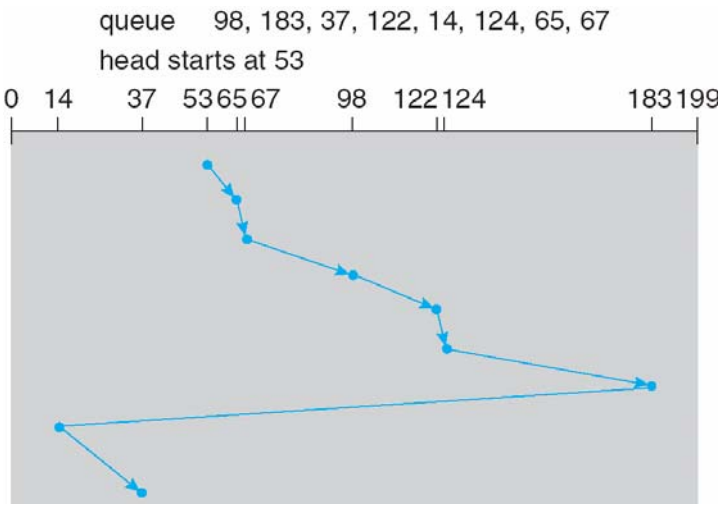


<inhon@mail.tku.edu.tw> May 22, 2011

Look & C-LOOK

- Variety from SCAN and C-SCAN
- Arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk.
- Illustration shows total head movement of 354 Cylinders in C-LOOK

C-LOOK



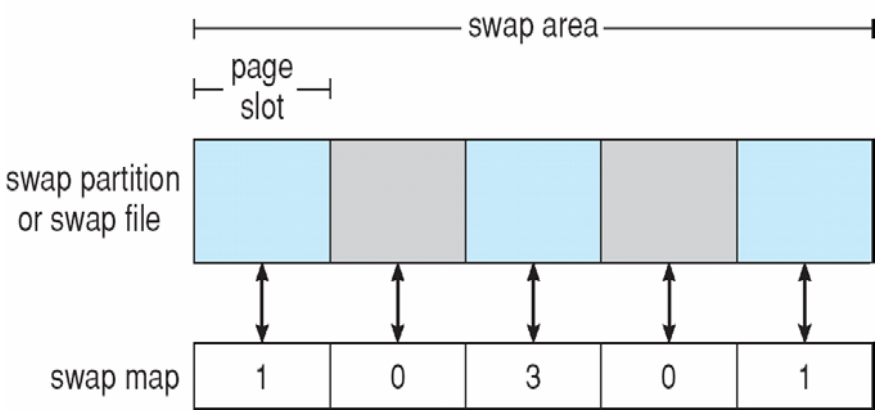
Selecting a Disk-Scheduling Algorithm

- SSTF is common and has a natural appeal
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk.
- Performance depends on the number and types of requests.
- Requests for disk service can be influenced by the file-allocation method.
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary.
- Either SSTF or LOOK is a reasonable choice for the default algorithm.

Swap-Space Management

- Swap-space — Virtual memory uses disk space as an extension of main memory.
- Swap-space can be carved out of the normal file system,or, more commonly, it can be in a separate disk partition.
- Swap-space management
  - 4.3BSD allocates swap space when process starts; holds *text segment* (the program) and *data segment.*
  - Kernel uses *swap maps* to track swap-space use.
  - Solaris 2 allocates swap space only when a page is forced out of physical memory, not when the virtual memory page is first created.

Swapping on Linux Systems



RAID Structure

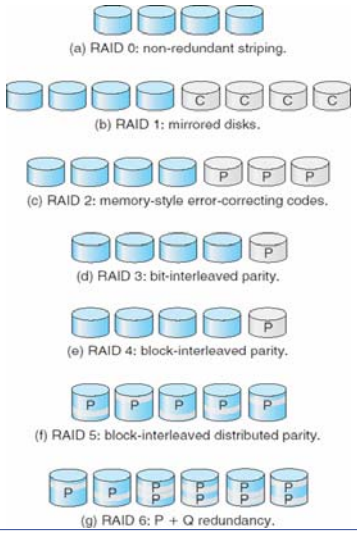
- **RAID**-Redundancy Array Independent Disks
- **RAID** – multiple disk drives provides **reliability** via **redundancy**.
- RAID is arranged into six different levels.

RAID Structure

- Several improvements in disk-use techniques involve the use of multiple disks working cooperatively.
- Disk striping uses a group of disks as one storage unit.
- RAID schemes improve performance and improve the reliability of the storage system by storing redundant data.
  - *Mirroring* or *shadowing* keeps duplicate of each disk.
  - *Block interleaved parity* uses much less redundancy.

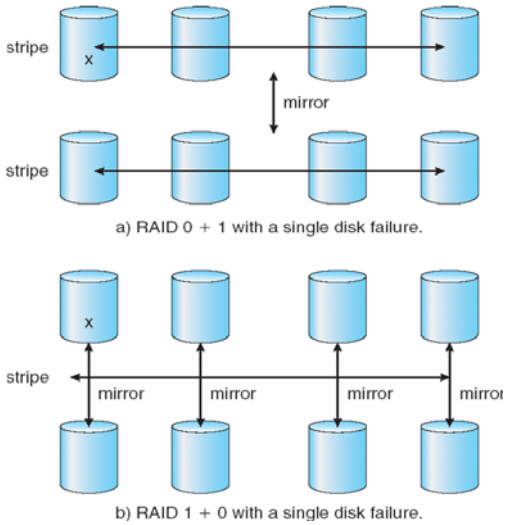
TSEG Tamkang University Software Engineering Group 淡江軟體工程實驗室 http://www.tkse.tku.edu.tw/

RAID Structure



<inhon@mail.tku.edu.tw> May 22, 2011

RAID (0 + 1) and (1 + 0)



a) RAID 0 + 1 with a single disk failure.

b) RAID 1 + 0 with a single disk failure.