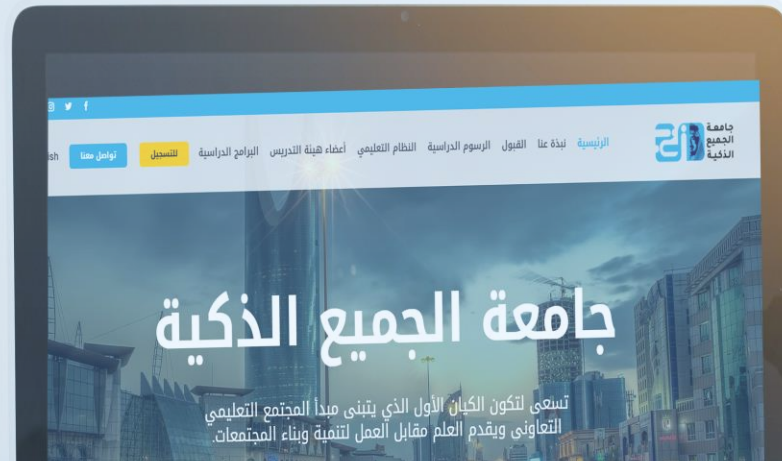


CYB 510 OPERATING SYSTEMS

Prepared by Prof. Dr. Lamiaa Fattouh

Lecture 10

System



CYB 510

File Systems Interface and Security



Outline

- File Concept
- Access Methods
- Disk and Directory Structure
- Protection
- Memory-Mapped Files



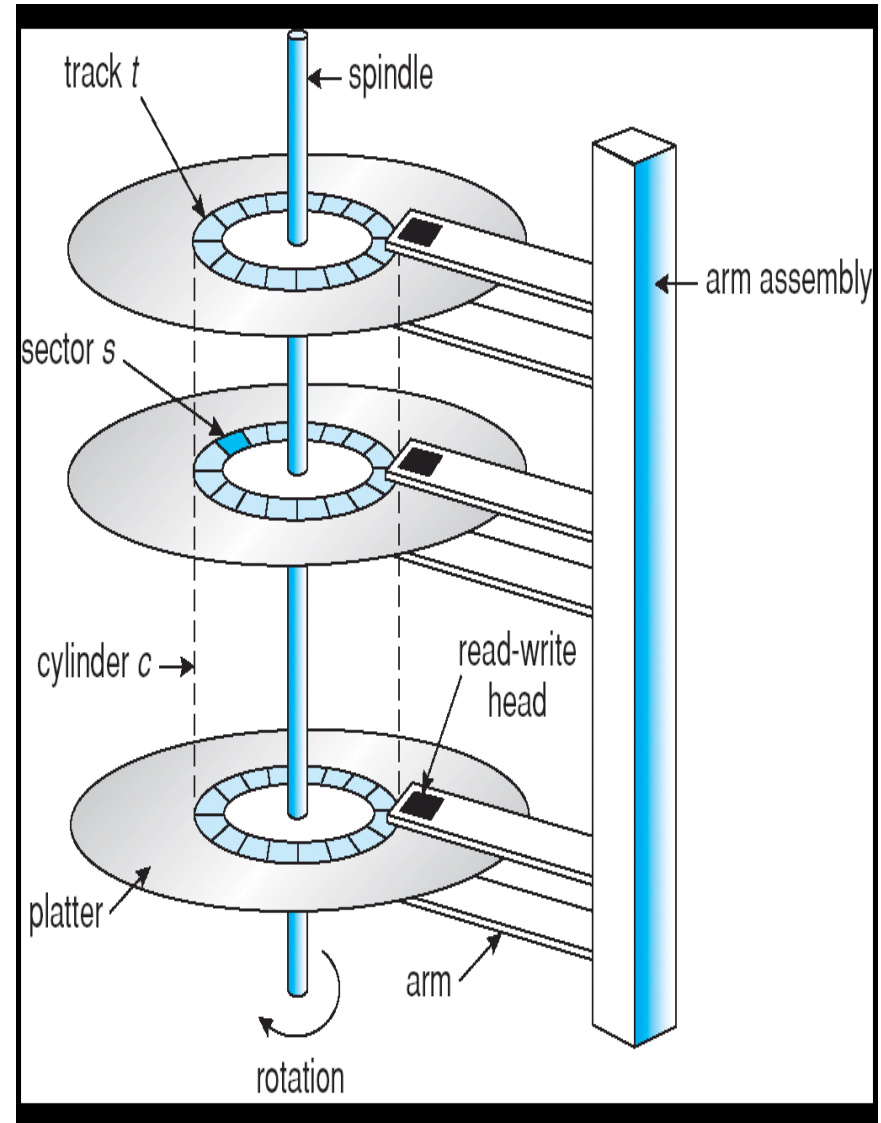
Objectives

- 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



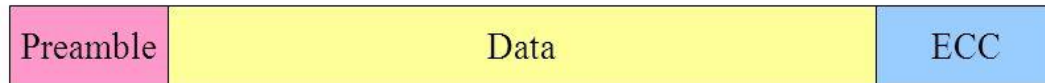
Magnetic Media (Disk)

- Disks have multiple platters
- Each platter has a head
- The different heads are attached to a single arm
- The different heads can access data in parallel
- Each platter has multiple concentric tracks
- A cylinder is same track across different platters
- Each track has multiple sectors
- A sector typically has preamble, data and ECC



Magnetic Disk (Cont.)

Structure of a disk sector



- Preamble contains information about the sector
 - Sector number & location information
- Data is usually 256, 512, or 1024 bytes
- ECC (Error Correcting Code) is used to detect & correct minor errors in the data



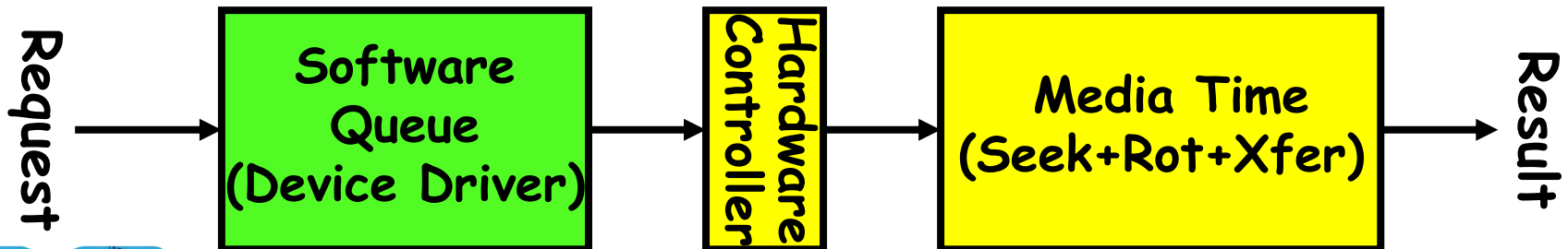
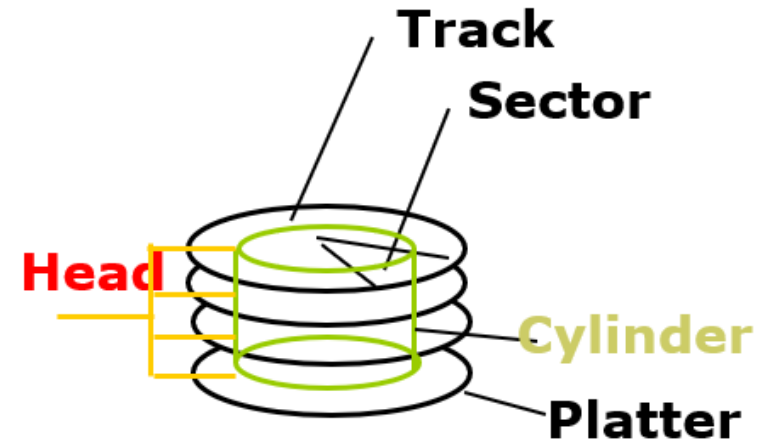
Magnetic Disk (Cont.)

- Disks are accessed at the granularity of sectors Typically, 512 bytes
- A file system allocates data in chunks called **blocks**
 - A **block** contains 2^n contiguous sectors
 - Reduces overhead of managing individual bytes
 - Large blocks improve throughput but increase internal fragmentation
- File systems view the disk as an array of blocks
- Disks can process I/O requests in *parallel*
- Many I/O requests for blocks can be pending at a given time
- **Disks are slow!**
 - ~10 ms latency to access a block
 - Versus 0.0002 ms to access to memory
 - Versus 0.000001ms to access to register



Magnetic Disk (Cont.)

- **Cylinder**: all the tracks under the heads at a given point on all surface



Magnetic Disk (Cont.)

- Read/write data is a three-stage process:
 - **Seek time**: position the head/arm over the proper **track** (into proper cylinder) => arm motion (say forth and back motion)
 - **Rotational latency**: wait for the desired **sector** to rotate under the read/write head (rotation motion)
 - **Transfer time**: transfer a block of bits (sector) under the read-write head
- **Disk Latency = Queuing Time + Controller time + Seek Time + Rotation Time + Xfer Time**
- **Highest Bandwidth**:
 - transfer large group of blocks sequentially from one track



To read/write information into a sector, a movable arm containing a read/write head is located over each surface. The term cylinder is used to refer to all the tracks under the read/write head at a given point on all surfaces.

To access data, the operating system must direct the disk through a 3-stage process.

- (a) The first step is to position the arm over the proper track. This is the seek operation and the time to complete this operation is called the seek time.
- (b) Once the head has reached the correct track, we must wait for the desired sector to rotate under the read/write head. This is referred to as the rotational latency.
- (c) Finally, once the desired sector is under the read/write head, the data transfer can begin.



Magnetic Disk (Cont.)

- Disks: *Not just slow, Complex*
 - Seeks between cylinders are *expensive*
 - ▶ Random vs. sequential access
- Sequential access – **platter rotation**
 - 80 MB/s bandwidth
 - 0.16 ms latency
- Random access – **head seeks**
 - 0.5 MB/s bandwidth
 - 10 ms latency



File Systems

- 3 criteria for long-term information storage:
 - Should be able to store very large amount of information
 - Information must persist the processes using it
 - Should provide concurrent access to multiple processes
- Solution:
 - Store information on disks in units called **files**
 - Files are **persistent**, and only **owner** can explicitly **delete** it
 - Files are managed by the OS
- File Systems: How the OS manages files!
- File systems have different types
 - File system partitions have types
 - ▶ e.g., swap space, EXT3, FAT32, NTFS



File System (Cont.)

- **File System:** Layer of OS that **transforms** block interface of disks (or other block devices) into Files, Directories, etc.
- File System Components
 - **Disk Management:** collecting disk blocks into files
 - **Naming:** Interface to find files by name, not by blocks
 - **Protection:** Layers to keep data secure
 - **Reliability/Durability:** Keeping of files durable despite crashes, media failures, attacks, etc.

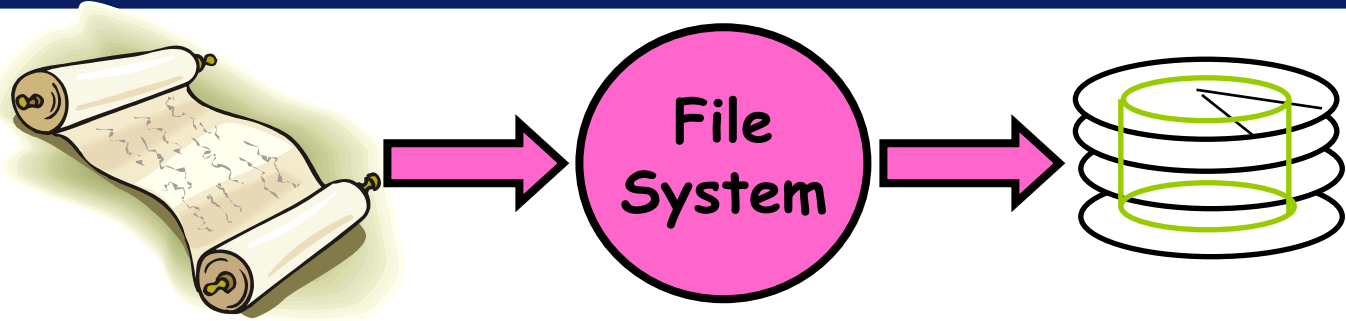


File System (Cont.)

- User vs. System View of a File
 - User's view:
 - ▶ Durable Data Structures: directories and files.
 - System's view (system call interface):
 - ▶ Collection of Bytes (UNIX)
 - ▶ Doesn't matter to system what kind of data structures you want to store on disk!
 - System's view (inside OS):
 - ▶ Collection of blocks (a **block** is a logical transfer unit, while a **sector** is the physical transfer unit)
 - ▶ **Block size $\geq$ sector size**; in UNIX, block size is 4KB



Translating from User to System View



- What happens if user says: give me bytes 2—12?
 - Fetch block corresponding to those bytes
 - Return just the correct portion of the block
 - What about: write bytes 2—12?
 - Fetch block
 - Modify portion
 - Write out Block
 - Everything inside File System is in whole size blocks
- file is a collection of blocks** (i.e. systems view inside OS)



File Systems (Cont.)

- A **file system** provides an **abstraction** for **storing**, **organizing** and **accessing persistent data**. It is a Layer of OS that **transforms** **block interface of disks (or other block devices)** into **Files, Directories**, etc.
 - File-system data is organized as logical files
 - ▶ Accessed via system calls
 - ▶ Files have names and are organized as directories
 - **Directory structure maintains info of all files in the volume**
 - ▶ Name, location, size, type, ...
 - ▶ The list of blocks containing the data
 - ▶ Other information such as access control list or permissions, owner, time of access, etc?



What is a File?

- A file is a sequence of bytes contained in multiple, ordered, sectors (blocks) on non-volatile devices such as disks
 - Disk space is always allocated in **blocks**
- A sector is a fixed size block of data
 - E.g., 512 bytes
- Example
 - With a 512 byte sector size, a 64 MB file (MB = 1024* 1024 bytes) would have how many sectors?

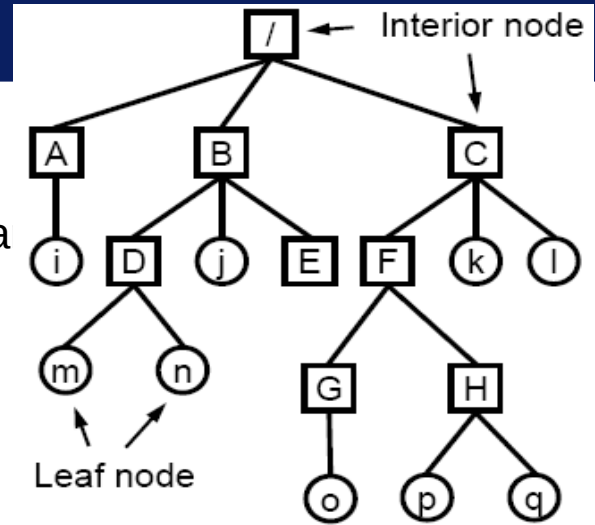
$$\frac{64 \times 1024 \times 1024 \text{ bytes}}{512 \text{ bytes/sector}} = 131,072 \text{ sectors}$$



Remember Block size $\geq$ sector size

Directories

- **A Directory** Store a list of directory entries that point to files or subdirectory. Method for naming a locating a file
 - Efficiently locate and identify files
 - Allow intuitive *grouping of objects like files together*
 - Useful for both OS and users
- Modern systems use hierarchical directories
 - A directory contains files or sub-directories
 - E.g., B contains entries for D, j and E
- Files are accessed with pathnames
 - **Absolute pathname**. path of file from the root directory e.g. cat /B/D/n
 - **Relative pathname** path from the current working directory e.g. cd B/D; cat n
 - two special entries in each directory:
 - “.” for current directory and “..” for parent



Operations Performed on Directory

- 1- Search for a file
- 2- Create a file
- 3- Delete a file
- 4- Truncate a file
- 5- List a directory
- 6- Rename a file
- 7- Traverse file system



File Structure and Attributes

- The OS typically treats **files as an unstructured sequence of bytes**
- Programs create files and they can impose any format on files
 - E.g., application programs may look for specific file extension to indicate the file's type
- However, OS needs to understand the format of executable files to execute programs(text, data, heap, stack sections)
- Files have various **attributes** associated with them. These attributes are called **file metadata or File Control Block (FCB) or i-node** and is **maintained in file system** data structures for each file. They vary a lot across different operating systems. Examples:
 - **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



File info Window on Mac OS X



File Operations

- 1- **Create a file**: create new file in a directory
 - Find space in the file system, add directory entry.
- 2- **Open file**: fetch existing file attributes and disk addresses in memory
 - System call specifying name of file.
 - system searches directory structure to find file.
 - System keeps **current file position pointer** to the location where next write/read occurs
 - System call returns **file descriptor** (a **handle**) to user process
- 3- **Write in file**: write content of buffer in memory into disk blocks- Increase size
 - System call specifying file descriptor and information to be written
 - Writes information at location pointed by the files current pointer
- 4- **Read from a file**: read disk blocks into buffer in memory
 - System call specifying file descriptor and number of bytes to read (and possibly where in memory to stick contents).
- 5- **Repositioning within a file**: move to specific block
 - System call specifying file descriptor and new location of current pointer. Also called a file **seek** even though does not interact with disk



File Operations (Cont.)

- **6- Sharing a file:**
 - Sharing may be done through a **protection scheme**
 - Some systems provide file sharing by maintaining a single copy of a file; other systems maintain several copies, one for each of the users
 - ▶ **With a single copy**, several concurrent updates to a file may result in user obtaining **incorrect information**, and the file being left in an **incorrect state**.
 - ▶ **With multiple copies**, there is **storage waste** and the various copies may not be consistent with respect to each other.
- **7- Delete (unlink) a file:** remove directory entry, free up space
- **8- Truncate a file:** free up space, set file length to zero
- **9- Closing a file:** entry removed from open file table
 - System call specifying file descriptor
 - Call removes current file position pointer and file descriptor associated with process and file



Multiple users of a file

- 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 (in memory) when last processes closes it
 - **Disk location of the file**: cache of data access information
 - **Access rights**: per-process access mode information
- OS typically keeps the above information in **two levels of internal tables**:
- **Per-process table**
 - Information about the use of the file by the user (e.g. **current file position pointer**).
- **System wide table**
 - Gets created by first process which opens the file
 - Location of file on disk
 - Access dates



File size

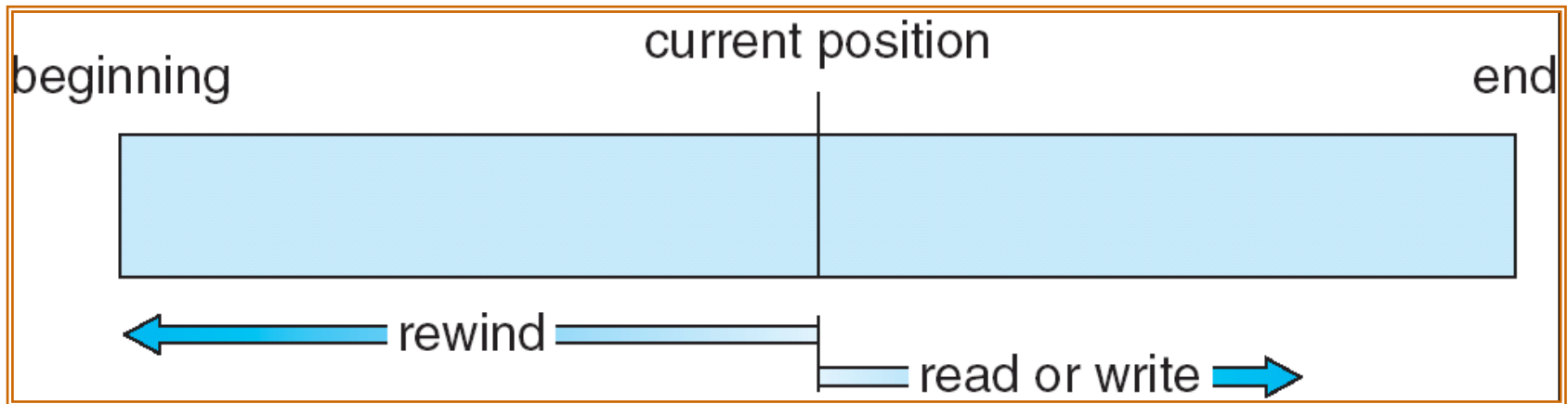
Count of how many processes have the file open (used for deletion)

Access Methods

- **Sequential Access**
- **Direct Access**

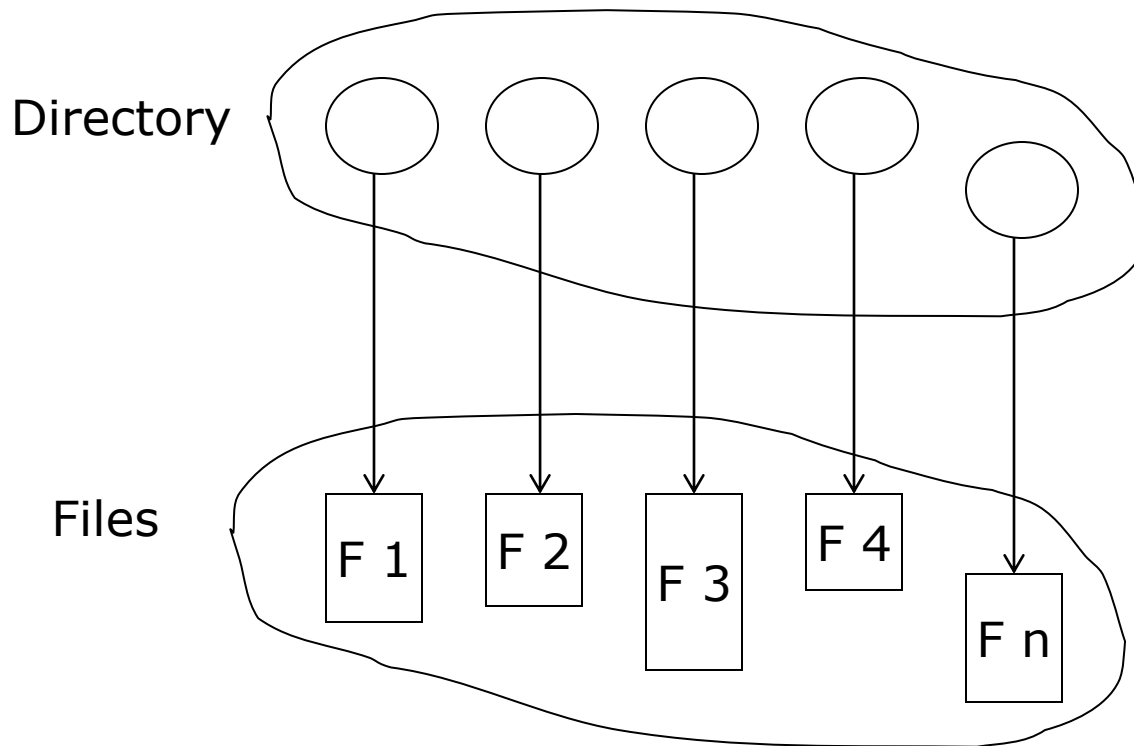


Sequential-access File



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

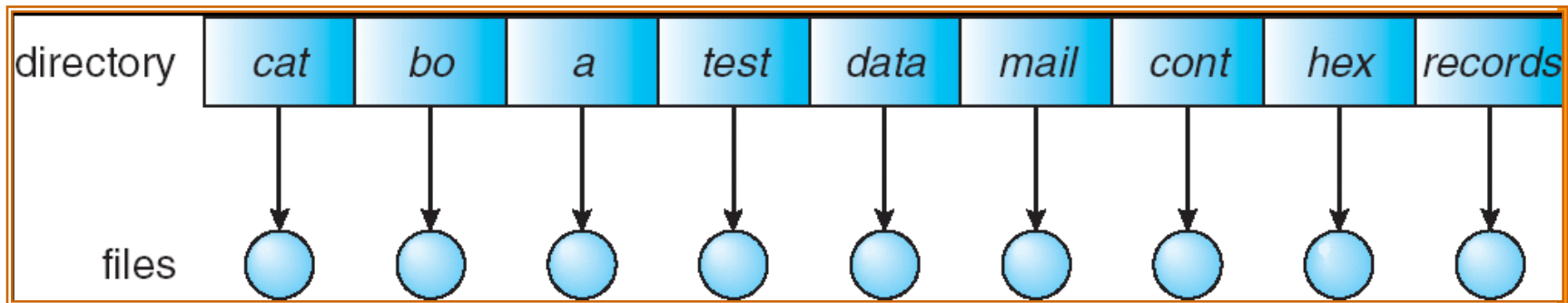
Organize the Directory (Logically) to Obtain

- 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



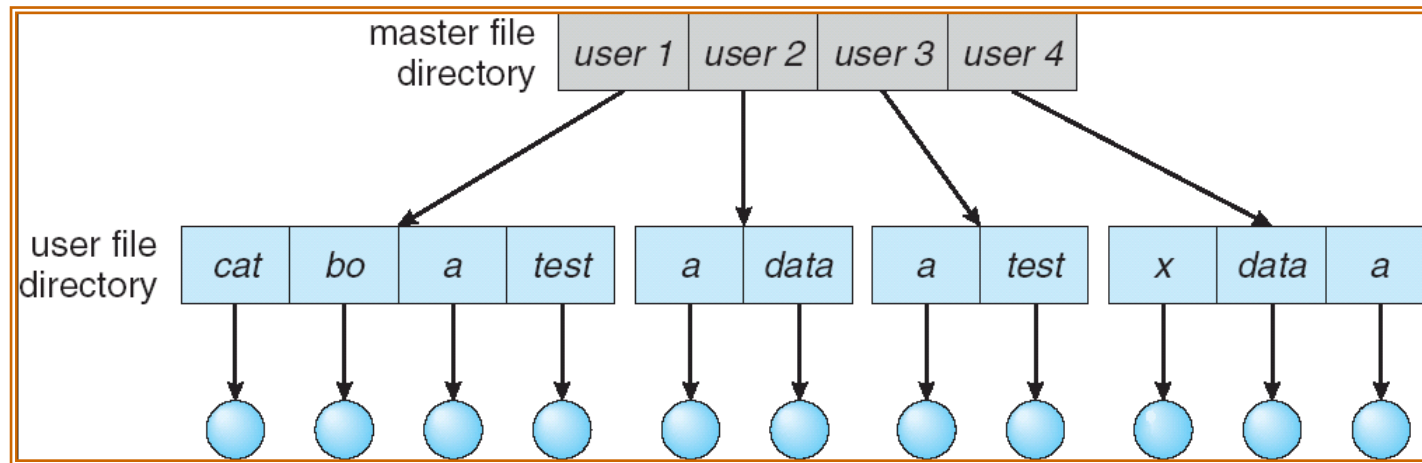
Naming problem

Grouping problem



Two-Level Directory

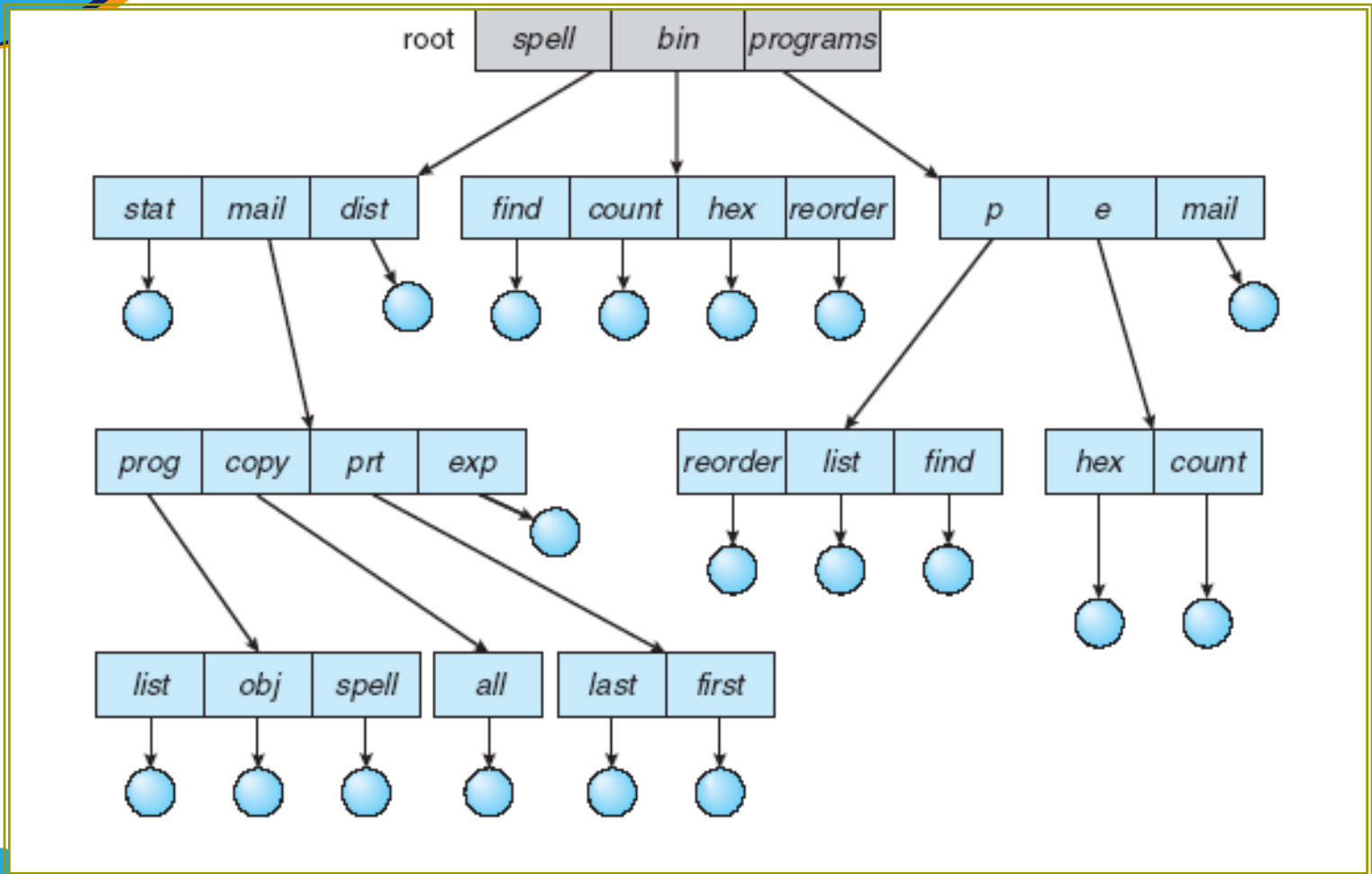
- Separate directory for each user



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

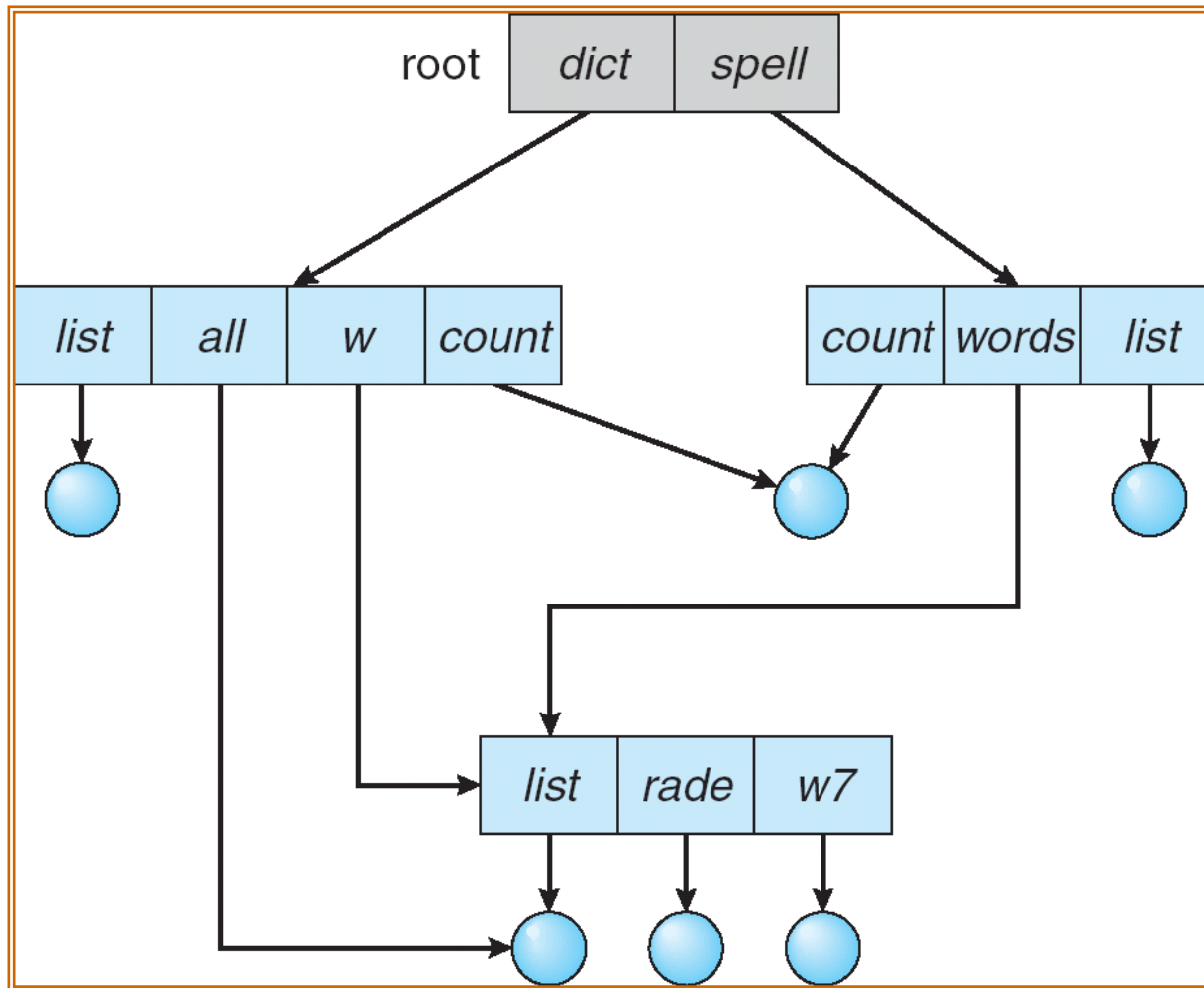


Tree-Structured Directories

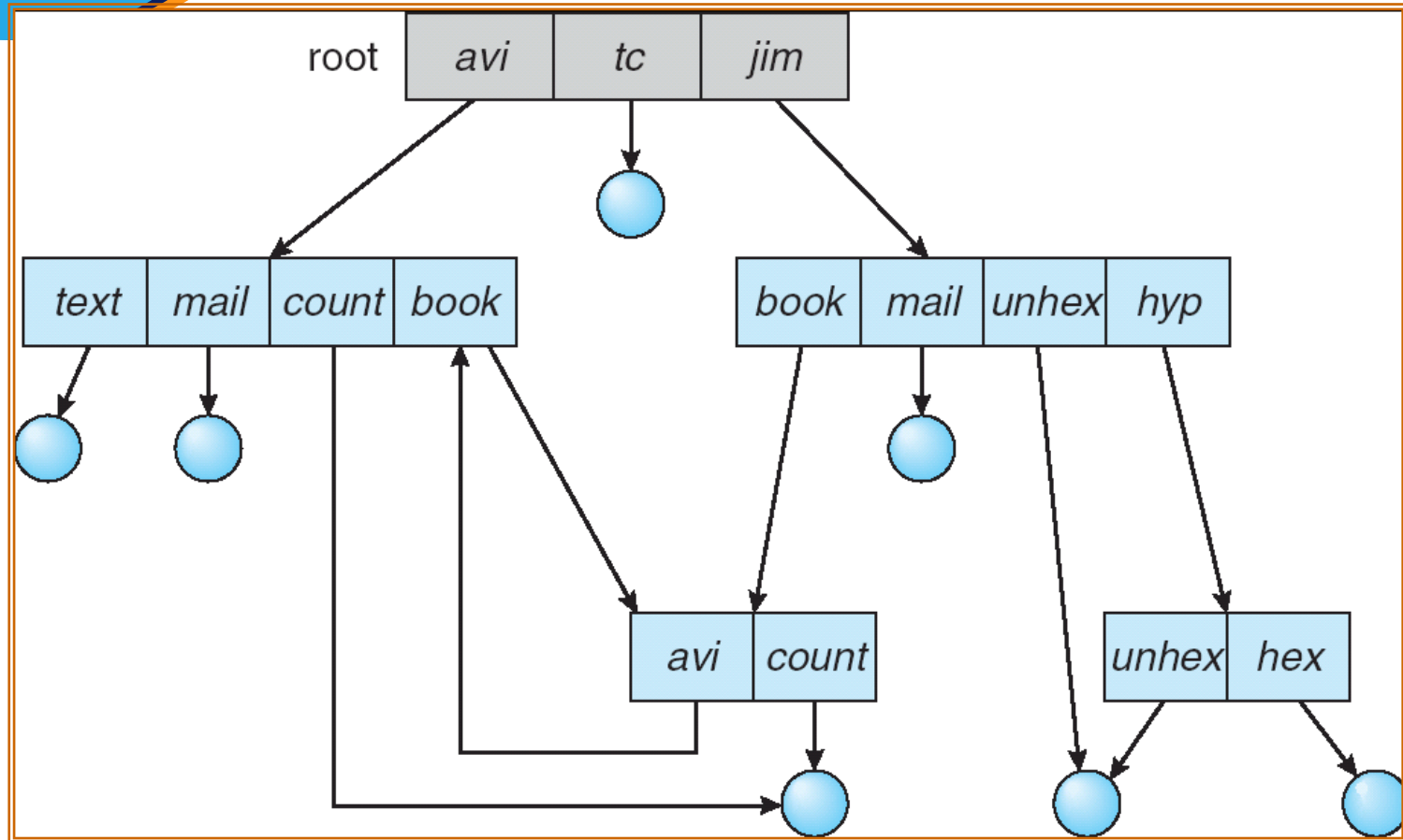


Acyclic-Graph Directories

- Have shared subdirectories and files



General Graph Directory



File Protection(Cont.)

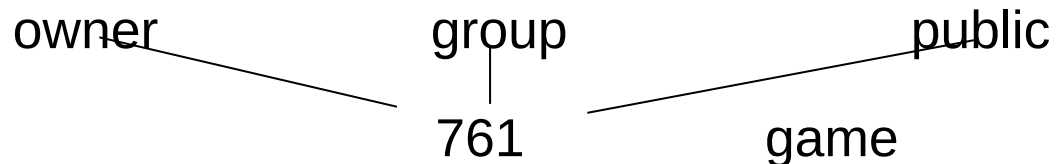
- **Linux Access Rights (chmod, uid, grpuid....)**

- **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) other access	1	⇒	0 0 1

- **For a particular file (say *game*) or subdirectory, define an appropriate access.**



Chapter 16: Security



The Security Problem

- System **secure** if resources used and accessed as intended under all circumstances
 - Unachievable
- **Intruders** (**crackers**) attempt to breach security
- **Threat** is potential security violation
- **Attack** is attempted to breach security
- Attack can be accidental or malicious
- Easier to protect against accidental than malicious misuse



Security Violation Categories

■ Breach of confidentiality

This type of violation involves **unauthorized reading of data** (or theft of information). Typically, a breach of confidentiality is the goal of an intruder. Capturing secret data from a system or a data stream, such as credit-card information or identity information for identity theft, or unreleased movies or scripts, can result directly in money for the intruder and embarrassment for the hacked institution.

Breach of integrity

This violation involves unauthorized modification of data. Such attacks can, for example, result in passing of liability to an innocent party or modification of the source code of an important commercial or open-source application.



Breach of availability

This violation involves unauthorized destruction of data. Some attackers would rather wreak havoc and get status or bragging rights than gain financially. Website defacement is a common example of this type of security breach.



Theft of service

This violation involves unauthorized use of resources. For example, an intruder (or intrusion program) may install a daemon on a system that acts as a file server.

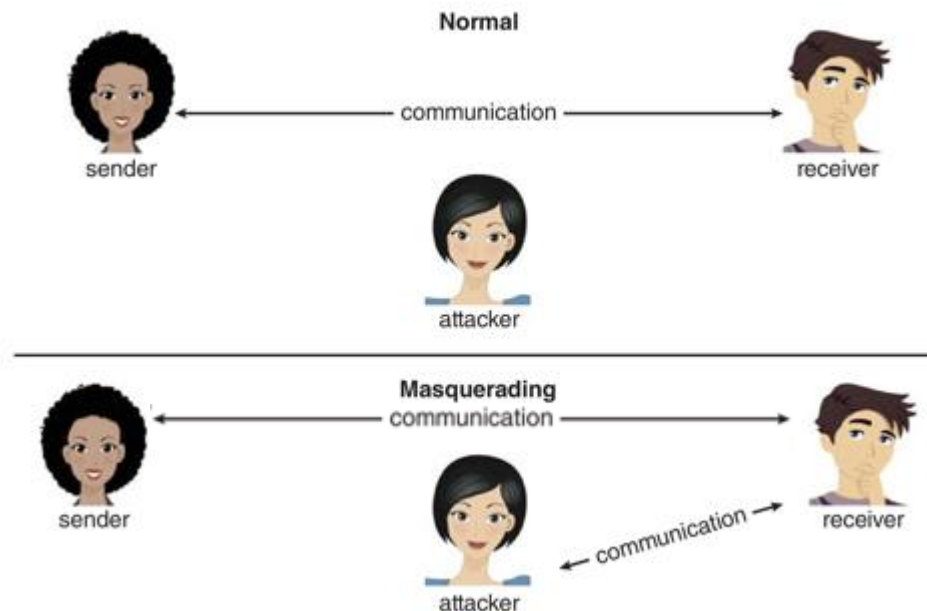
Denial of service (DOS)

This violation involves preventing legitimate use of the system. **Denial-of-service (DOS)** attacks are sometimes accidental. The original Internet worm turned into a DOS attack when a bug failed to delay its rapid spread.



Security Violation Methods

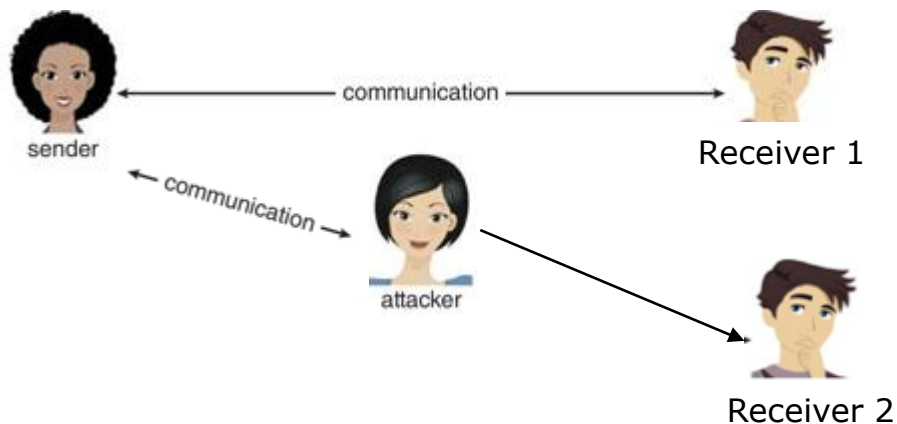
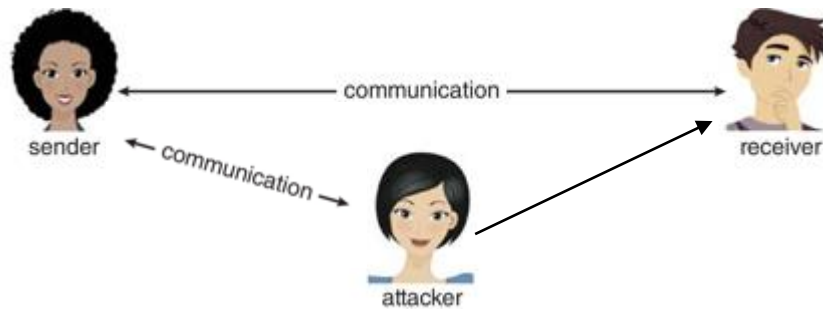
- **Masquerading** (breach **authentication**)
 - Pretending to be an authorized user to escalate privileges



Lecture 10

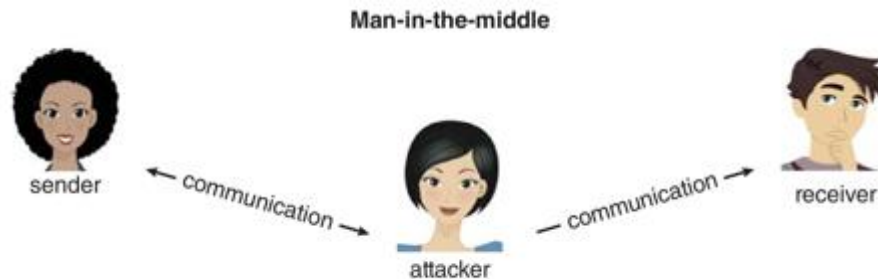
Replay attack

As is or with **message modification**



Man-in-the-middle attack

Intruder sits in data flow, masquerading as sender to receiver and vice versa



Session hijacking

Intercept an already-established session to bypass authentication

Privilege escalation

Common attack type with access beyond what a user or resource is supposed to have



Security Measure Levels

- Impossible to have absolute security, but make cost to perpetrator sufficiently high to deter most intruders
- Security must occur at four levels to be effective:
 - **Physical**
 - ▶ Data centers, servers, connected terminals
 - **Application**
 - ▶ Benign or malicious apps can cause security problems
 - **Operating System**
 - ▶ Protection mechanisms, debugging
 - **Network**
 - ▶ Intercepted communications, interruption, DOS
- Security is as weak as the weakest link in the chain
- Humans a risk too via **phishing** and **social-engineering** attacks
- But can too much security be a problem?



End of Lecture 10

