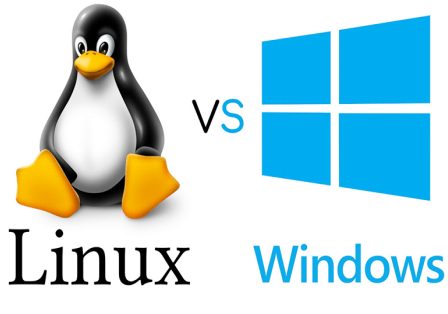
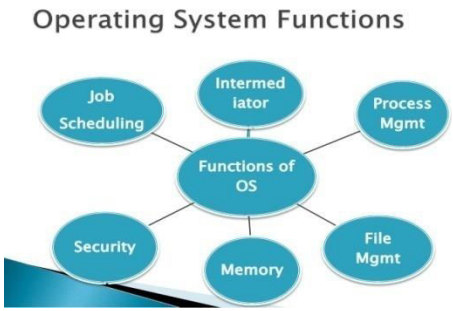
CIT210 OP. SYS. MANAG.

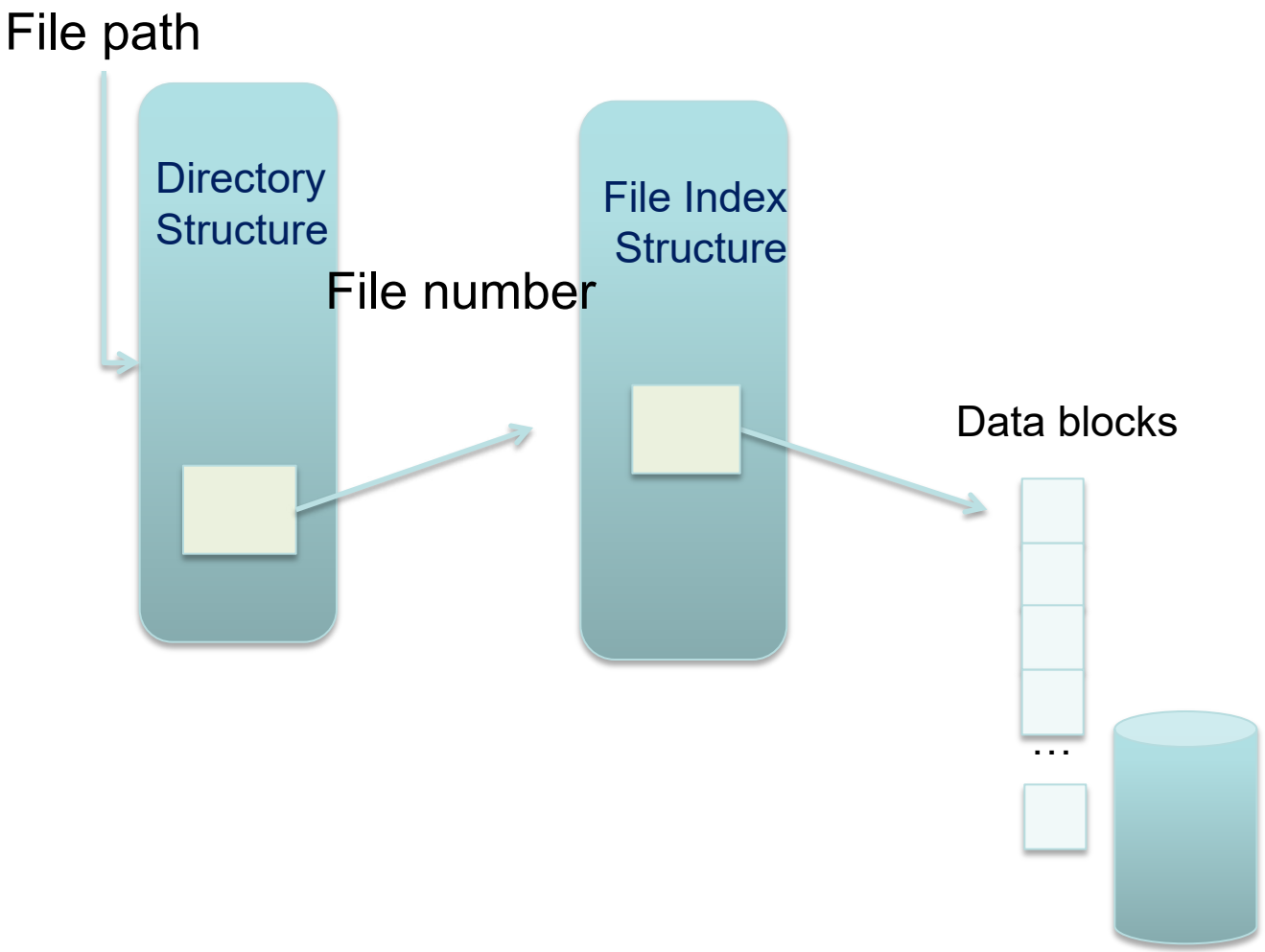
CSUN
COLLEGE OF ENGINEERING AND COMPUTER SCIENCE

Week5Day1 : Operating Systems: File-System Structure, Partition and Mounting, File-System Layers, Different File Systems (FAT, NTFS, HFS, ext3) Maintaining Consistency, Setting up RAID

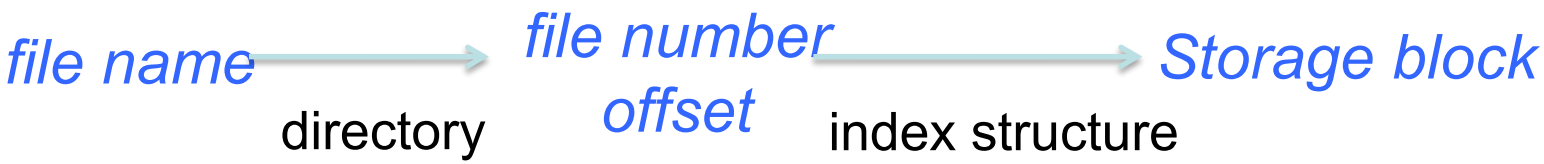




Basic File System Components



Components of a file system



- **Open performs Name Resolution**
  - **Translates pathname into a "file number"**
    - Used as an "index" to locate the blocks
  - **Creates a file descriptor in PCB (Process Control Block) within kernel**
  - **Returns a file descriptor (int) to user process**

- **`read`, `write`, `seek`, and `sync` operate on handle**
  - **Mapped to file descriptor and to blocks**

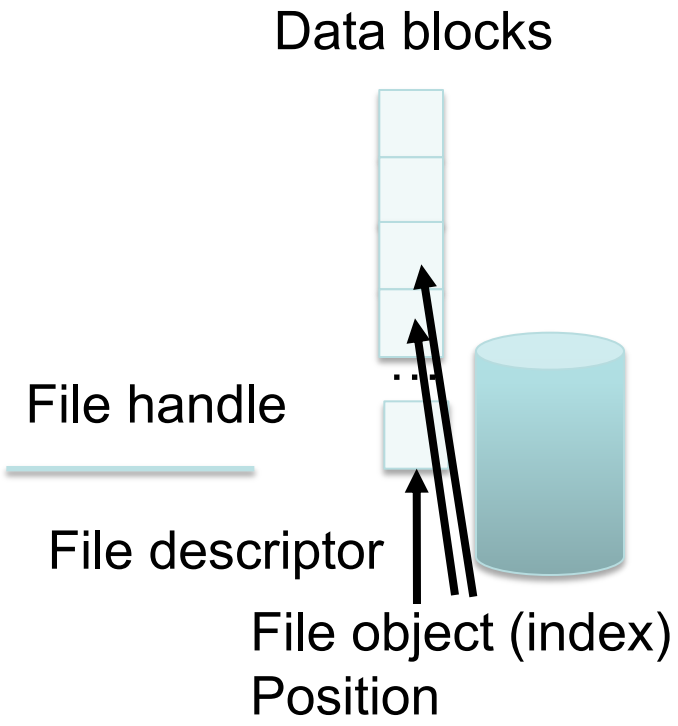
Directory

- A hierarchical structure
- Each directory entry is a collection of file name to file number mappings
  - Some of these mappings could be subdirectories: simply a link to another collection of mappings
- Directories actually form a DAG, not just a tree. Why?
  - Hard Link: Mapping from name to file number
  - File number could be pointed to by multiple names (in different directories)
  - Soft link: Mapping from one name to another

A DAG (Database Availability Group) is initially created as an empty object in Active Directory. This directory object is used to store relevant information about the DAG, such as server membership information and some DAG configuration settings. When you add the first server to a DAG, a failover cluster is automatically created for the DAG.

File

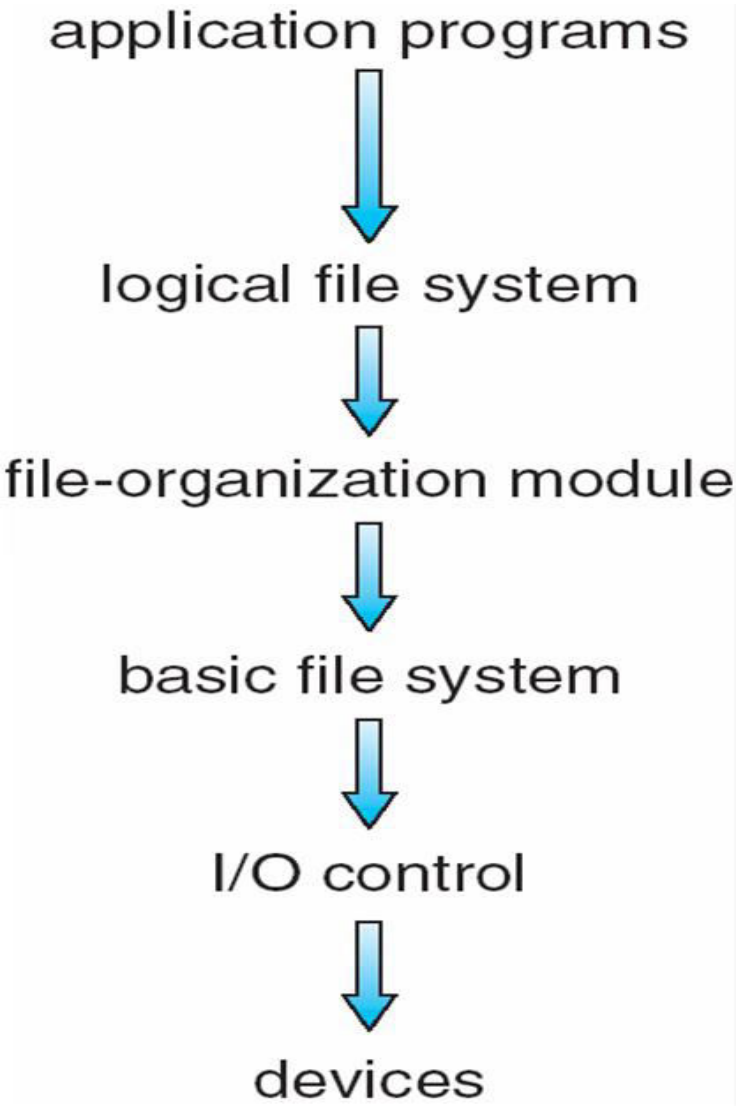
- **Named permanent storage**
- **Contains**
  - **Data: Blocks on disk somewhere**
  - **Metadata (Attributes)**
    - Owner, size, last opened, …
    - Access rights



File-System Structure

- File structure
  - Logical storage unit
  - Collection of related information
- File system resides on secondary storage (disks)
  - Provided user interface to storage, mapping logical to physical
  - Provides efficient and convenient access to disk by allowing data to be stored, located retrieved easily
- Disk provides in-place rewrite and random access
  - I/O transfers performed in blocks of sectors (usually 512 bytes)
- File control block – storage structure consisting of information about a file
- Device driver controls the physical device
- File system organized into layers

Layered File System



File System Layers

- **Device drivers** manage I/O devices at the I/O control layer
  - Given commands like “read drive1, cylinder 72, track 2, sector 10, into memory location 1060” outputs low-level hardware specific commands to hardware controller
- **Basic file system** given command like “retrieve block 123” translates to device driver
  - Also manages memory buffers and caches (allocation, freeing, replacement)
    - `Buffers hold data in transit`
    - `Caches hold frequently used data`

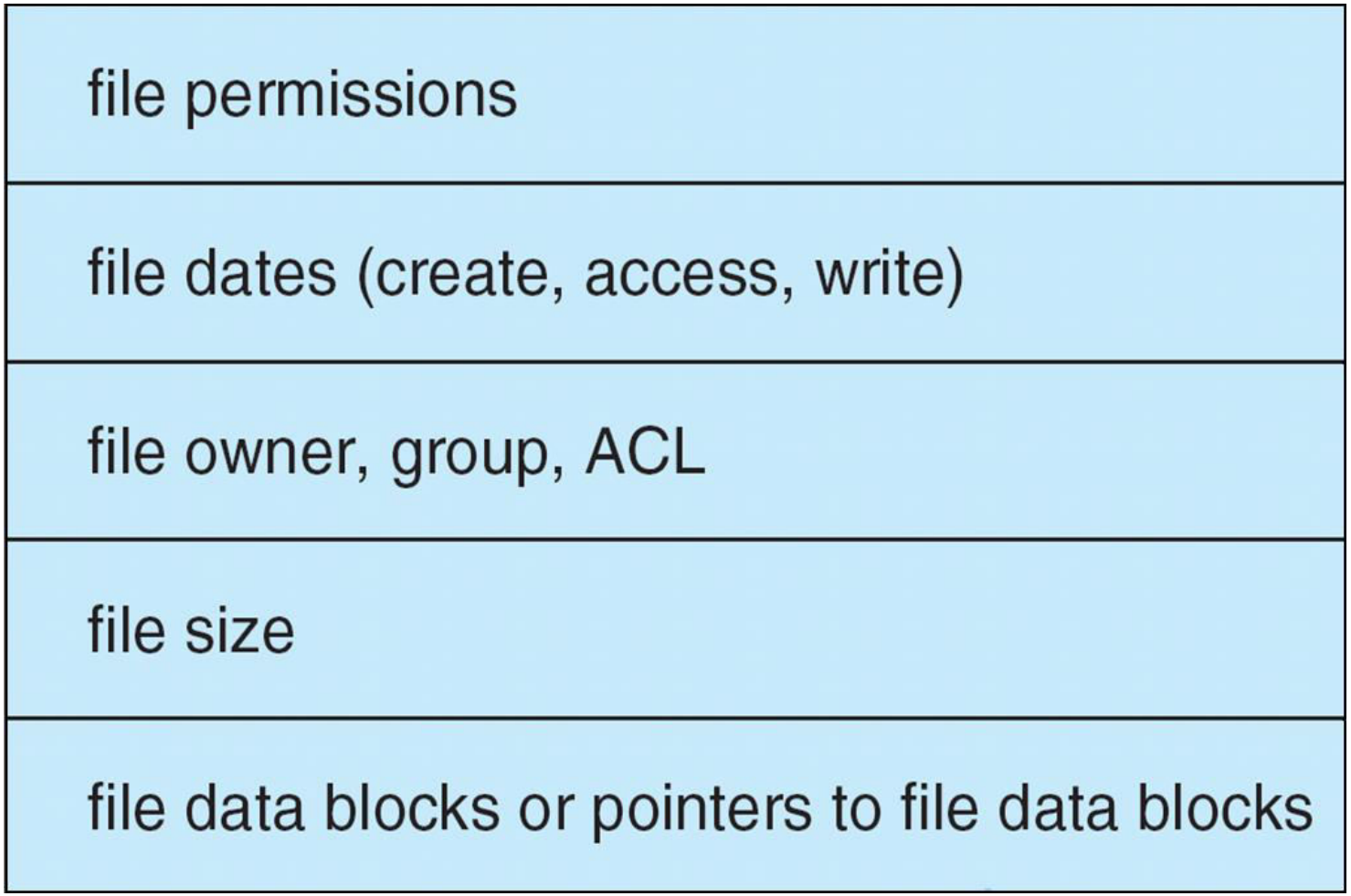
File System Layers

- **File organization module understands files, logical address, and physical blocks**
  - **Translates logical block # to physical block #**
  - **Manages free space, disk allocation**
  - **Logical file system manages metadata information**
  - **Translates file name into file number, file handle, location by maintaining file control blocks (inodes in Unix)**
  - **Directory management**
  - **Protection**
- **Layering useful for reducing complexity and redundancy, but adds overhead and can decrease performance**
  - Logical layers can be implemented by any coding method according to OS designer

File-System Implementation

- We have system calls at the API level, but how do we implement their functions?
  - On-disk and in-memory structures
- Boot control block contains info needed by system to boot OS from that volume
  - Needed if volume contains OS, usually first block of volume
- Volume control block (superblock, master file table) contains volume details
  - Total # of blocks, # of free blocks, block size, free block pointers or array
- Directory structure organizes the files
  - Names and inode numbers, master file table
- Per-file File Control Block (FCB) contains many details about the file
  - Inode number, permissions, size, dates
  - NTFS stores into in master file table using relational DB structures

A Typical File Control Block



Partitions and Mounting

- Partition can be a volume containing a file system ("cooked") or raw – just a sequence of blocks with no file system
- Boot block can point to boot volume or boot loader set of blocks that contain enough code to know how to load the kernel from the file system
  - Or a boot management program for multi-os booting
- Root partition contains the OS, other partitions can hold other Oses, other file systems, or be raw
  - Mounted at boot time
  - Other partitions can mount automatically or manually
- At mount time, file system consistency checked
  - Is all metadata correct?
    - If not, fix it, try again
    - If yes, add to mount table, allow access

Filesystems

- **A *filesystem* organizes data in a storage device in a way that makes it easy to store and retrieve data from the device.**
- **Many file systems, sometimes many within an operating system**
  - Each with its own format (CD-ROM is ISO 9660; Unix has **UFS**, FFS; Windows has FAT, FAT32, NTFS as well as floppy, CD, DVD Blu-ray, Linux has more than 40 types, with **extended file system** ext2 and ext3 leading; plus distributed file systems, etc)
  - New ones still arriving – ZFS, GoogleFS, Oracle ASM, FUSE

Filesystems

- Some file systems are used on local data storage devices; others provide file access via a network protocol.
- Data storage devices often have *partitions*, and the separate partitions may have different filesystems on them; when dual-booting from the a hard disk, one operating system uses one partition while the other uses another.
- The disk stores the information about the partitions' locations and sizes in an area known as the *partition table*, which is often in the first sector of the disk.

Different types of file systems

- Local file systems
  - Stored data on local hard drives, SSDs, floppy drives, optical disks or etc.
  - Examples: NTFS, EXT4, HFS+, ZFS
- Network/distributed file systems
  - Stored data on remote file server(s)
  - Example: NFS, CIFS/Samba, AFP, Hadoop DFS, Ceph
- Pseudo file systems
  - Example: procfs, devfs, tmpfs
- “List of file systems”
  - http://en.wikipedia.org/wiki/List_of_file_systems

File Allocation Tables (FAT)

- Simple file system popularized by MS-DOS
  - First introduced in 1977
  - Most devices today use the FAT32 spec from 1996
  - FAT12, FAT16, VFAT, FAT32, exFATetc.
- Still quite popular today
  - Default format for USB sticks and memory cards
  - Used for EFI boot partitions
- Name comes from the index table used to track directories and files

FAT File System

- The File Allocation Table (FAT) file system is a simple file system originally designed for small disks and simple folder structures.
- The FAT file system is named for its method of organization, the file allocation table, which resides at the beginning of the volume.
- To protect the volume, two copies of the table are kept, in case one becomes damaged. In addition, the file allocation tables and the root folder must be stored in a fixed location so that the files needed to start the system can be correctly located.

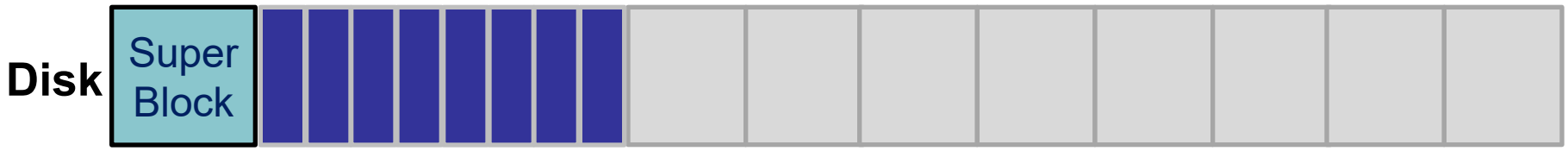
FAT32 File System

- FAT32 supports drives with over 2GB of storage.
- FAT32 drives can contain more than 65,526 clusters, smaller clusters are used than on large FAT16 drives.
- This method results in more efficient space allocation on the FAT32 drive.
- The largest possible file for a FAT32 drive is 4GB.
- The FAT32 file system includes four bytes per cluster within the file allocation table. Note that the high 4 bits of the 32-bit values in the FAT32 file allocation table are reserved and are not part of the cluster number.

- Stores basic info about the file system
- FAT version, location of boot files
- Total number of blocks
- Index of the root directory in the FAT

- File allocation table (FAT)
- Marks which blocks are free or in-use
- **Linked-list structure** to manage large files

- Store file and directory data
- Each block is a fixed size (4KB – 64KB)
- Files may span multiple blocks



FAT Limitations

FAT type	Max Clusters	Cluster sizes	Max volume size
FAT12	4,086	0.5 to 4KB	16,736,256 bytes (**16MB**)
FAT16	65,526	2KB to 32KB	2,147,483,648 bytes (**2GB**)
FAT32	268,435,456	4KB to 32KB	8,796,093,022,208 bytes (**8TB**)

NTFS System Files

- NTFS is a high-performance, self-healing file system proprietary to Windows. Features include
  - file-level security, compression, and auditing
  - support for large volumes and powerful storage solutions such as built-in RAID support
  - the ability to encrypt files and folders to protect your sensitive data
  - more efficient drive management due to its smaller cluster size capabilities
  - support for very large drives made possible by its 64-bit clustering arrangement
  - recoverable file system capabilities

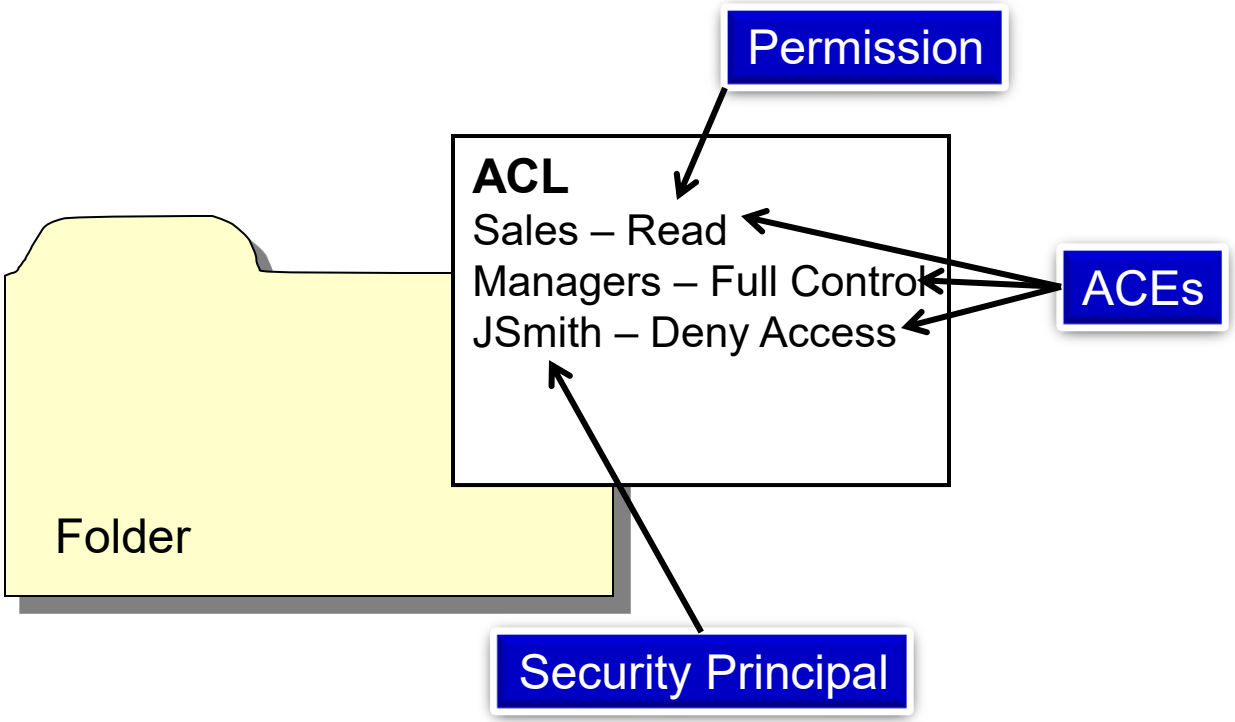
Assigning Permissions (NTFS)

The four permissions in Windows systems:

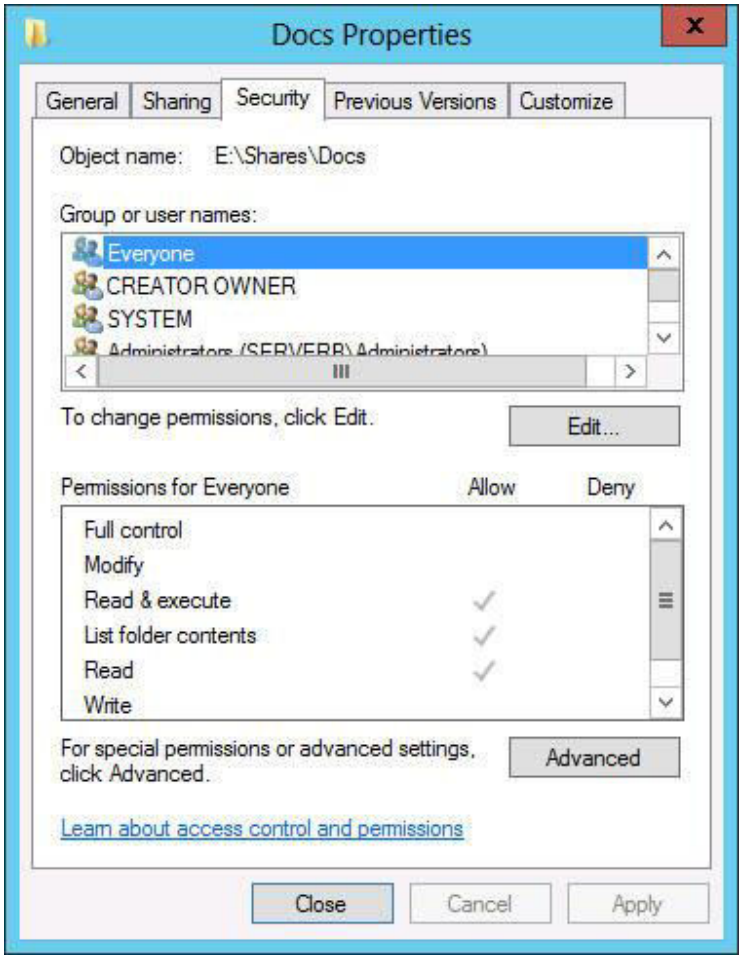
- **Share permissions: Control access to folders over a network.**
- **NTFS permissions: Control access to the files and folders stored on disk volumes formatted with the NTFS file system.**
- **Registry permissions: Control access to specific parts of the Windows registry.**
- **Active Directory permissions: Control access to specific parts of an Active Directory Domain Services (AD DS) hierarchy.**

Windows Permissions Architecture

- Access Control List (ACL)
- Access Control Entries (ACEs)
- Security principal



Windows Permissions



Allowing and Denying Permissions

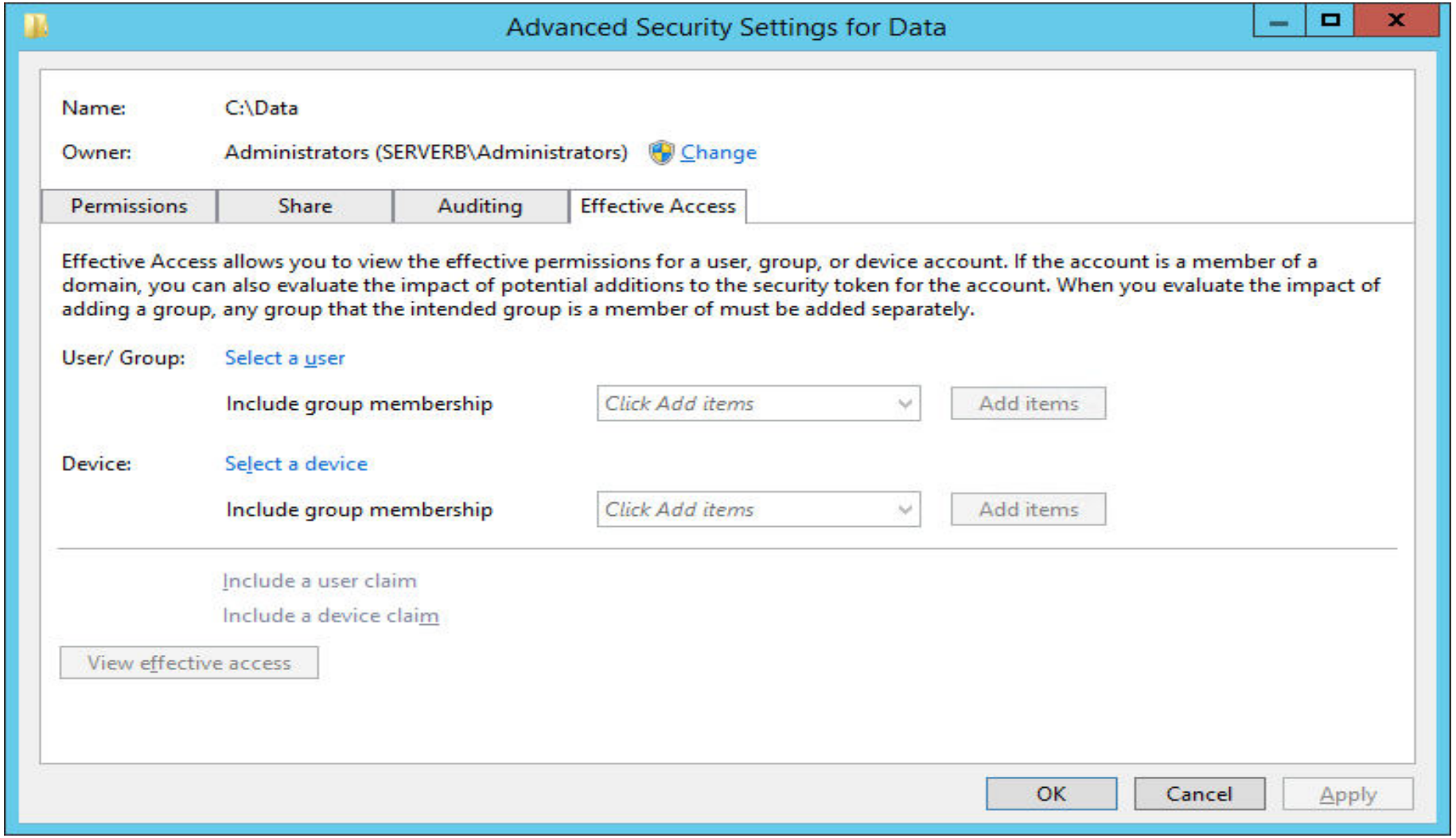
- **Additive**
  - **Start with no permissions and then grant Allow permissions (preferred method).**
- **Subtractive**
  - **Start by granting Allow permissions and then grant Deny permissions.**

Effective Access

The combination of Allow permissions and Deny permissions that a security principal receives for a system element:

- Allow permissions are cumulative.
- Deny permissions override Allow permissions.
- Explicit permissions take precedence over inherited permissions.

Effective Access



NTFS Authorization

- NTFS and ReFS support permissions.
- Every file and folder on an NTFS or ReFS drive has an ACL with ACEs, each of which contains a security principal and their permissions.
- Security Principals are users and groups identified by Windows using security identifiers (SIDs).
- During authorization, when a user accesses a file/folder, the system compares the user's SIDs to those stored in the element's ACEs to determine that user's access.

NTFS Quotas

- Enable administrators to set a storage limit for users of a particular volume.
- Users exceeding the limit can be denied access or just receive a warning.
- Space consumed by users is measured by the size of the files they own or create.

Fast File System (FFS)

- FFS developed at Berkeley in 1984
  - First attempt at a disk aware file system
  - i.e. optimized for performance on spinning disks
- Observation: processes tend to access files that are in the same (or close) directories
  - Spatial locality
- Key idea: place groups of directories and their files into cylinder groups

Cross-development

- Linux: first developed on a minix system
- Both OSs shared space on the same disk
- So Linux reimplemented minix file system
- Two severe limitations in the minix FS
  - Block addresses are 16-bits (64MB limit)
  - Directories use fixed-size entries (w/filename)

Extended File System

- Originally written by Chris Provenzano
- Extensively rewritten by Linux Torvalds
- Initially released in 1992
- Removed the two big limitations in minix
- Used 32-bit file-pointers (filesizes to 2GB)
- Allowed long filenames (up to 255 chars)
- Question: How to integrate ext into Linux?

Limitations in Ext

- **Some problems with the Ext filesystem**
  - **Lacked support for 3 timestamps**
    - Accessed, Inode Modified, Data Modified
  - **Used linked-lists to track free blocks/inodes**
    - Poor performance over time
    - Lists became unsorted
    - Files became fragmented
  - **Did not provide room for future extensibility**

Xia and Ext2 filesystems

- Two new filesystems introduced in 1993
- Both tried to overcome Ext's limitations
- Xia was based on existing minix code
- Ext2 was based on Torvalds' Ext code
- Xia was initially more stable (smaller)
- But flaws in Ext2 were eventually fixed
- Ext2 soon became a 'de facto' standard

Definition of Inodes

- Every file has a unique inode
- Contain the information necessary for a process to access a file
- Exist in a static form on disk
- Kernel reads them into an in-core inode to manipulate them.

Contents of Disk inodes

- File owner identifier (individual/group owner)
- File type (regular, directory,..)
- File access permission (owner,group,other)
- File access time
- Number of links to the file
- Table of contents for the disk address of data in a file (byte stream vs discontiguous disk blocks)
- File size

* inode does not specify the path name that access the file

Sample Disk Inode

- File owner identifier
- File type
- File access permission
- File access time
- Number of links to the file
- Table of contents for the disk address of data in a file
- File size

Owner mjb
Group os
Type regular file
Perms rwxr-xr-x
Accessed Oct 23 1984 1:45 P.M
Modified Oct 22 1984 10:3 A.M
Inode Oct 23 1984 1:30 P.M
Size 6030 bytes
Disk addresses

Linux ext2 inodes

Size (bytes)	Name	What is this field for?
2	mode	Read/write/execute?
2	uid	User ID of the file owner
4	size	Size of the file in bytes
4	time	Last access time
4	ctime	Creation time
4	mtime	Last modification time
4	dtime	Deletion time
2	gid	Group ID of the file
2	links_count	How many hard links point to this file?
4	blocks	How many data blocks are allocated to this file?
4	flags	File or directory? Plus, other simple flags
60	block	15 direct and indirect pointers to data blocks

Status Check

- At this point, we have a full featured file system
  - Directories
  - Fine-grained data allocation
  - Hard/soft links
- File system is optimized for spinning disks
  - inodes are optimized for small files
  - Block groups improve locality
- What's next?
  - Consistency and reliability

Maintaining Consistency

- Many operations results in multiple, independent writes to the file system
  - Example: append a block to an existing file
  1. Update the free data bitmap
  2. Update the inode
  3. Write the user data
- What happens if the computer crashes in the middle of this process?

The Crash Consistency Problem

- **The disk guarantees that sector writes are atomic**
  - **No way to make multi-sector writes atomic**
- **How to ensure consistency after a crash?**
  1. **Don't bother to ensure consistency**
     - Accept that the file system may be inconsistent after a crash
     - Run a program that fixes the file system during bootup
     - File system checker (*fsck*)
  2. **Use a transaction log to make multi-writes atomic**
     - Log stores a history of all writes to the disk
     - After a crash the log can be "replayed" to finish updates
     - Journaling file system

File System Security

- The file system is crucial to data integrity.
- Main method of protection is through access control
- Accessing file system operations (ex. modifying or deleting a file) are controlled through access control lists or capabilities
- Capabilities are more secure so they tend to be used by operating systems on file systems like NTFS or ext3/4.
- Secondary method of protection is through the use of backup and recovery systems

Attacks on the file system

- Race Condition Attacks
- Using ADS to hide files
- Directory traversal

Race Condition Attacks

- *A ``**race condition**" can be defined as ``Anomalous behavior due to unexpected critical dependence on the relative timing of events". **Race conditions** generally involve one or more processes accessing a shared resource (such a **file** or variable), where this multiple access has not been properly controlled.*
- **Occurs when a process performs a sequence of operations on a file, under the assumption that they are executed atomically.**
- **Can be used by the attacker to change the characteristics of that file *between* two successive operations on it resulting in the victim process to operate on the modified file.**

Avoiding Race Conditions

- A *race condition* exists when changes to the order of two or more events can cause a change in behavior.
- If the correct order of execution is required for the proper functioning of the program, this is a bug.
- If an attacker can take advantage of the situation to insert malicious code, change a filename, or otherwise interfere with the normal operation of the program, the race condition is a security vulnerability.
- Attackers can sometimes take advantage of small time gaps in the processing of code to interfere with the sequence of operations, which they then exploit.

Avoiding Race Conditions

- *macOS, like all modern operating systems, is a multitasking OS; that is, it allows multiple processes to run or appear to run simultaneously by rapidly switching among them on each processor.*
- *The advantages to the user are many and mostly obvious; the disadvantage, however, is that there is no guarantee that two consecutive operations in a given process are performed without any other process performing operations between them.*
- *In fact, when two processes are using the same resource (such as the same file), there is no guarantee that they will access that resource in any particular order unless both processes explicitly take steps to ensure it.*

Avoiding Race Conditions

- For example, if you open a file and then read from it, even though your application did nothing else between these two operations, some other process might alter the file after the file was opened and before it was read.
- If two different processes (in the same or different applications) were writing to the same file, there would be no way to know which one would write first and which would overwrite the data written by the other. Such situations cause security vulnerabilities.

Using ADS to hide Files

- Alternate Data Streams(ADS) allows multiple data streams to be attached to a single file.
- A file can be hidden behind a file as an attached stream that could be hundreds of megabytes in size, however a directory listing will only display the file's normal size.

Directory Traversal

- An exploit caused by lack of insufficient security validation of user supplied input file names
- For example the attacker would pass this as input. ../../../../../../../../../../etc/password to retrieve the password file from the server.

How does the file system ensure data integrity?

- There are various methods of protecting the files on a file system.
  - Access Controls
  - Encryption
  - RAID
  - Recovery when data is corrupted

Access Control

- Access Control plays a huge part in file system security
- The system should only allow access to files that the user is permitted to access
- Almost all major file systems support ACL's or capabilities in order to prevent malicious activity on the file system
- Depending on the users rights they can be allowed to read, write and/or execute and object. In some file systems schemes only certain users are allowed to alter the ACL on a file or see if a file even exists.
- Ultimately the less the user has access to the less that can go wrong and the integrity of the disk can be more guaranteed.

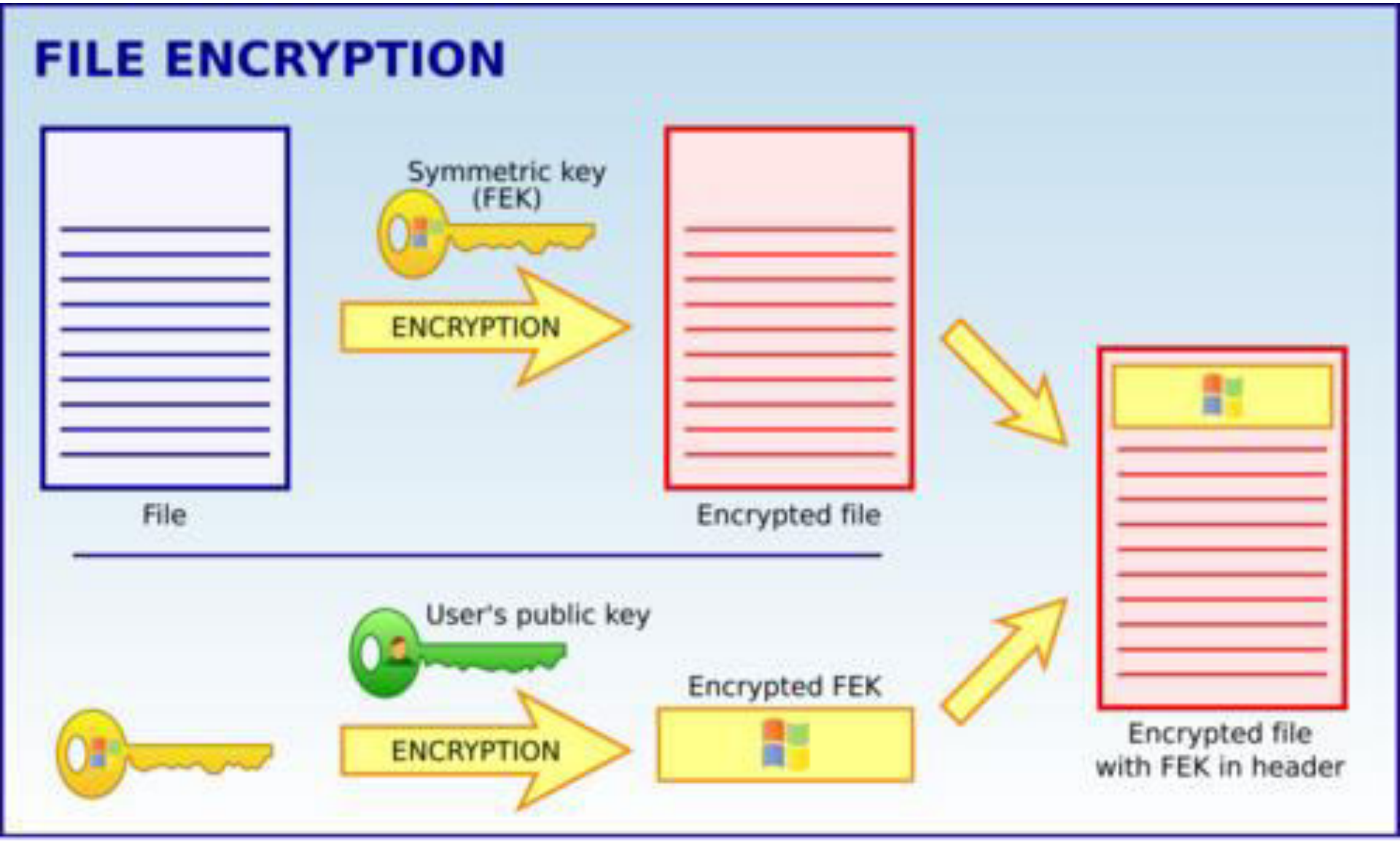
General File System Encryption

- Encryption is also a method used by file systems to secure data, NTFS for example offers file encryption using DESX
- Two method of disk encryption
  - Full Disk Encryption
  - File System Encryption
- File system encryption has a few advantages over full disk encryption for example
  1. File based key management
  2. Individual management of encrypted files
  3. Access control can be further strengthened through the use of public key cryptography
  4. Keys are only held in memory while the file is being used

Encrypting File System(EFS)

- Provides security beyond user authentication and access control lists. For example when the attacker has physical access to the computer.
- EFS uses public key cryptography however it is susceptible to brute-force attacks against the user account passwords.
- EFS works by encrypting a file with a bulk symmetric key, aka File Encryption Key or FEK.
- The FEK is encrypted with a public key that is associated with the user that encrypted the file.

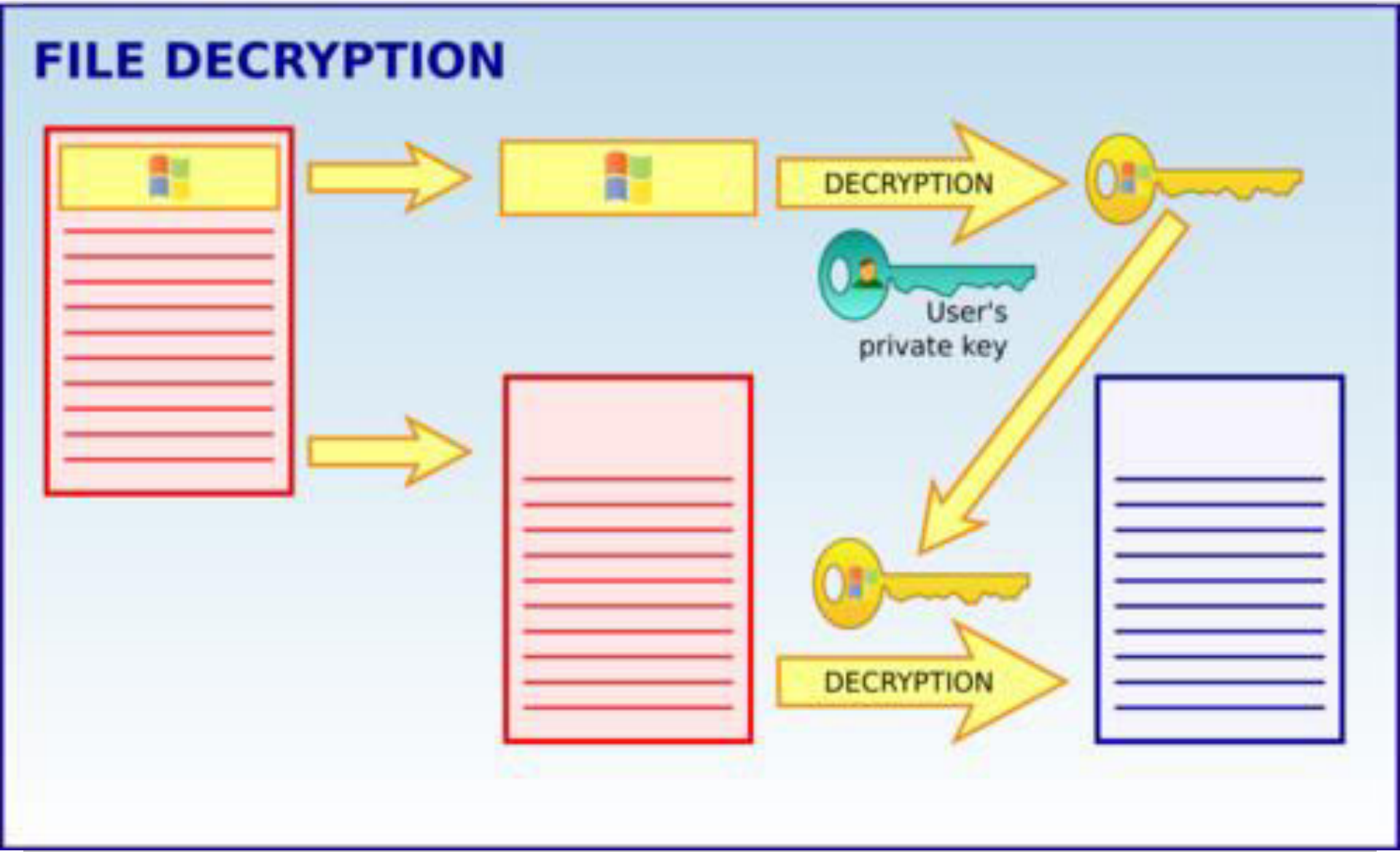
EFS Encryption



EFS Decryption

- The EFS uses the private key that matches the EFS digital certificate (that was used to encrypt the file) to decrypt the symmetric key.
- The resulting symmetric key is then used to decrypt the file.

EFS Decryption



RAID

- RAID stands for Redundant Array of Independent Disks
- Offers drawbacks and advantages over a single disk, each with different applications
- Types of RAID
  - RAID 0 “Striping set without parity”
  - RAID 1 “Mirrored set without parity”
  - RAID 3 “Striped set with byte level parity”
  - RAID 4 “Striped set with block level parity”
  - RAID 5 “Striped set with distributed parity”
  - RAID 6 “Striped set with dual distributed parity”

RAID Introduction

- RAID stands for Redundant Array of Independent Disks
- A system of arranging multiple disks for redundancy (or performance)
- Term first coined in 1987 at Berkley
- Idea has been around since the mid 70's
- RAID is now an umbrella term for various disk arrangements
  - Not necessarily redundant

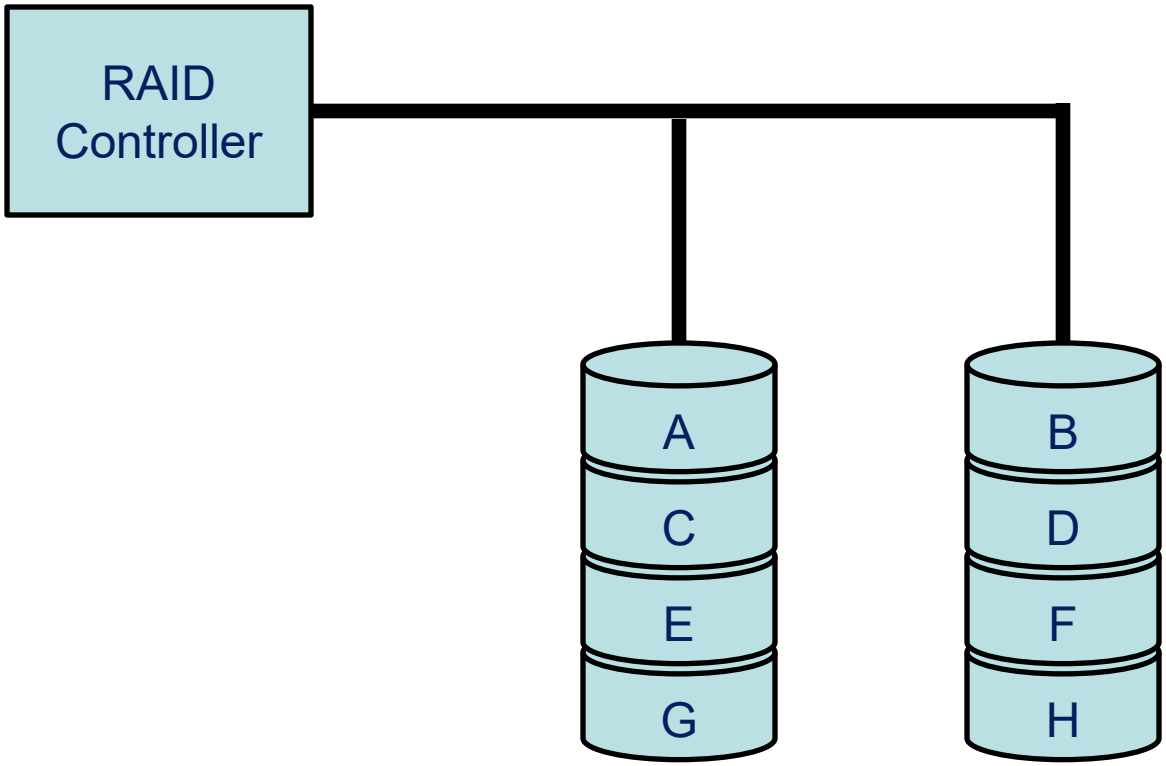
RAID 0

- Also known as “Striping”
- Data is striped across the disks in the array
- Each subsequent block is written to a different disk

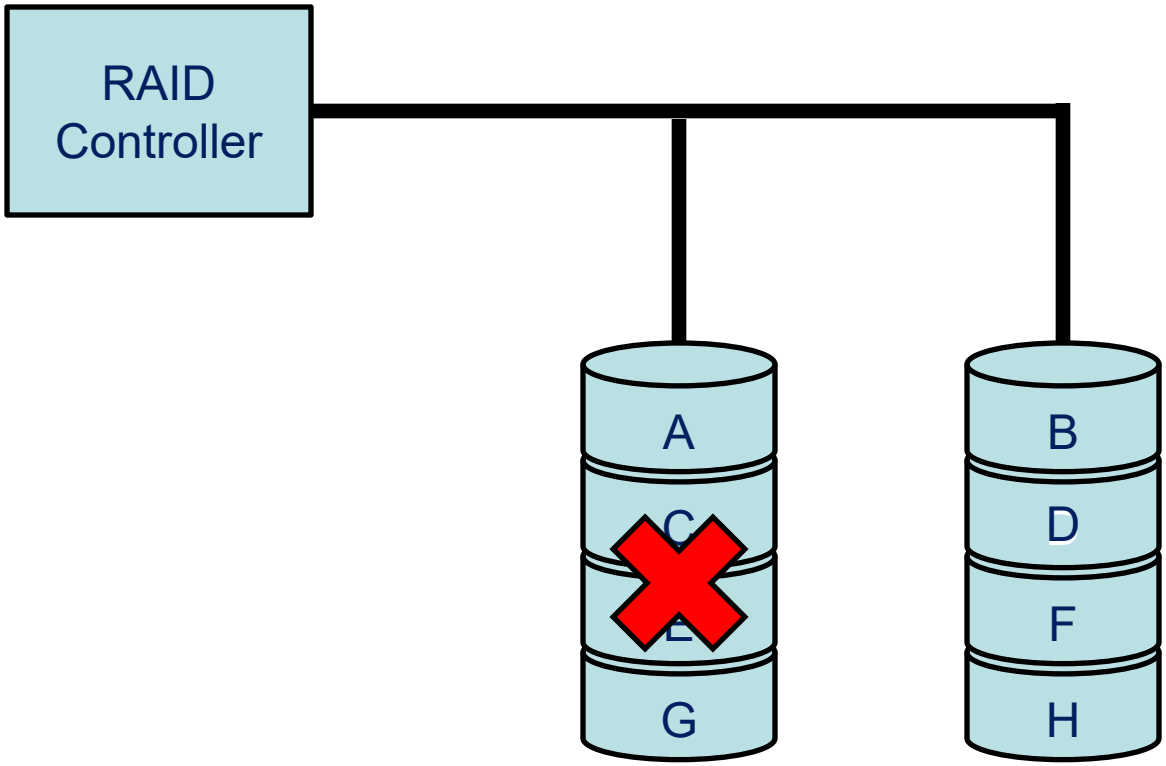
Using a Striped Volume (RAID-0)

- Reasons for using a RAID level 0 or a striped volume in Windows Server :
  - Reduce the wear on multiple disk drives by equally spreading the load
  - Increase disk performance compared with other methods for configuring dynamic disk volumes
- To create a striped volume, right-click the unallocated space for the volume and click New Striped Volume
- Only dynamic disks can be striped volumes

RAID 0 Reads



RAID 0 Recovery



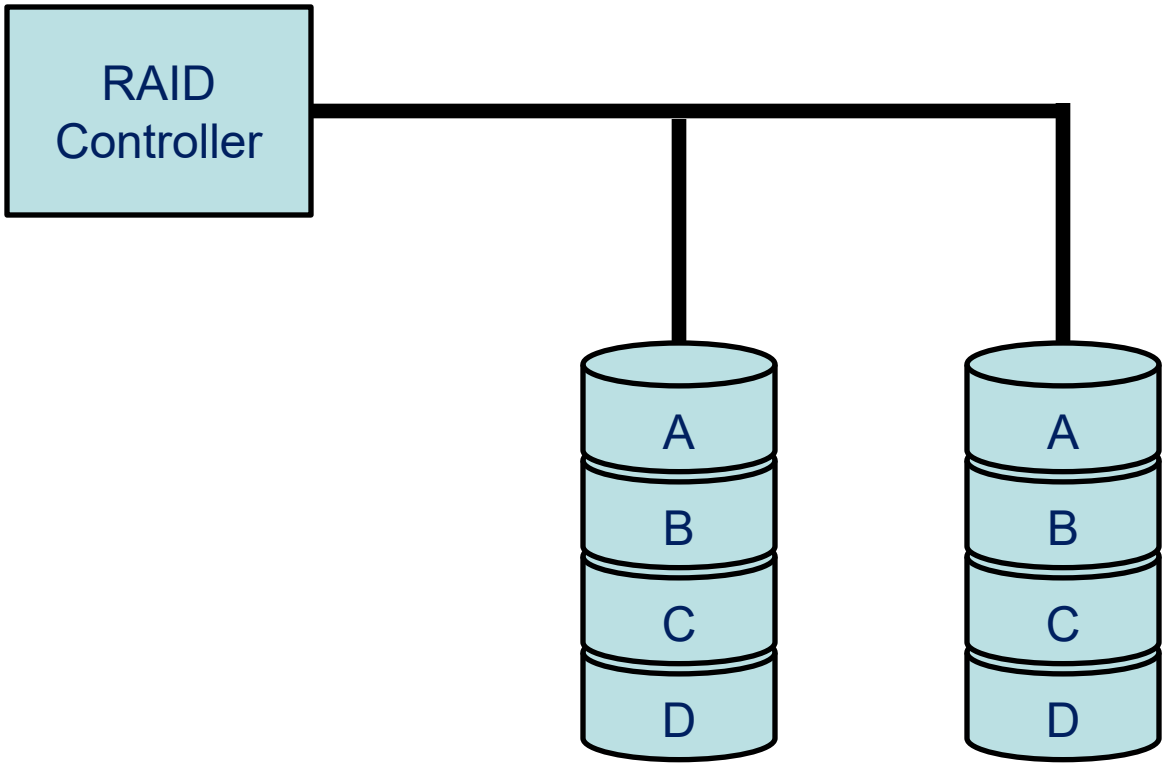
RAID 1

- Known as “Mirroring”
- Data is written to two disks concurrently
- The first type of RAID developed

Using a Mirrored Volume (RAID-1)

- Disk mirroring involves creating a shadow copy of data on a backup disk
- Only dynamic disks can be set up as a mirrored volume in Windows Server
- One of the most guaranteed forms of disk fault tolerance
- Disk read performance is the same as reading data from any single disk drive
- A mirrored volume is well suited for situations in which data is mission-critical and must not be lost under any circumstances
  - Such as customer files at a bank
- A mirrored volume is created through the Disk Management tool

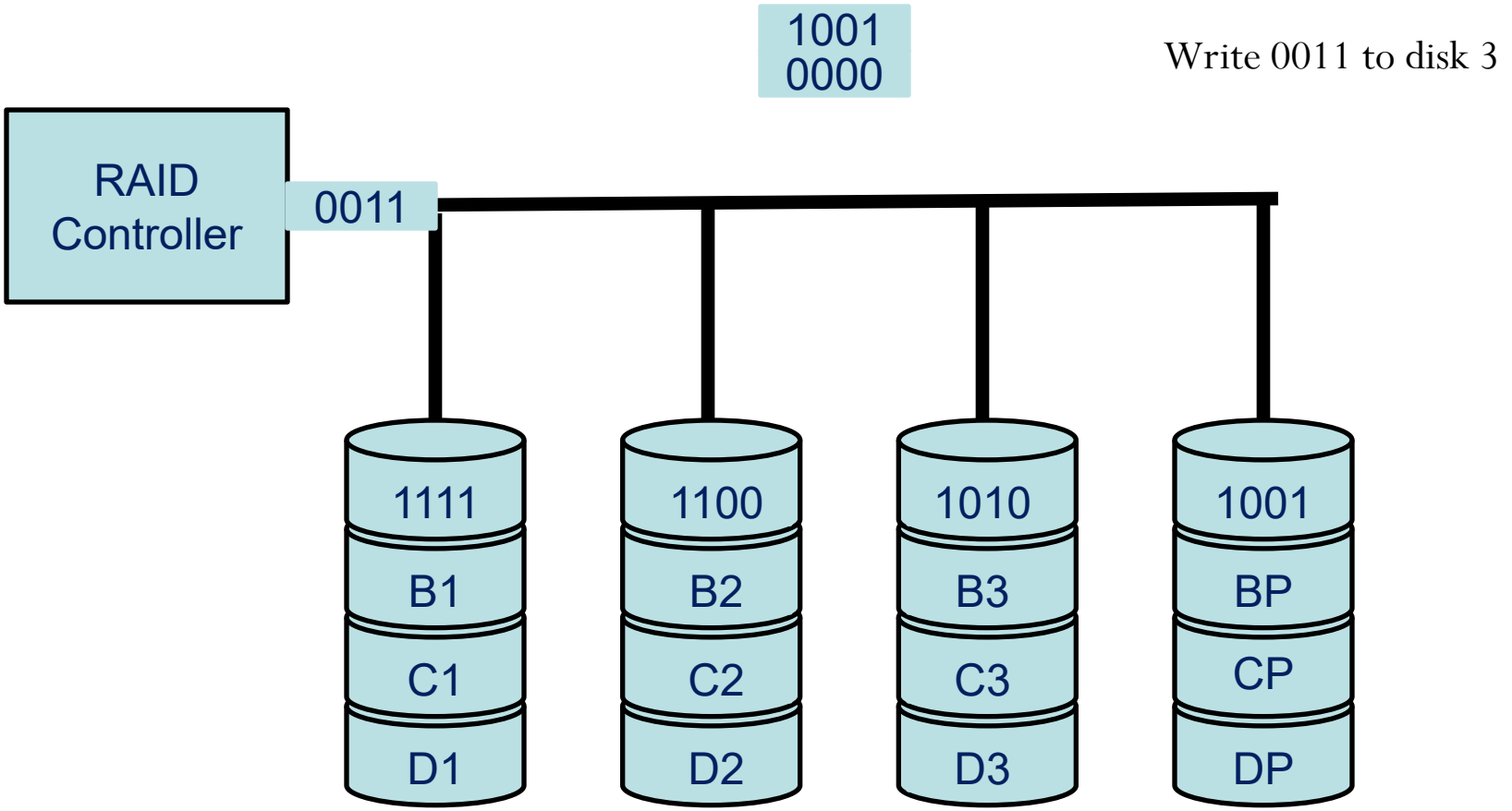
RAID 1 Reading



RAID 4

- Striping with a dedicated parity disk
- Blocks are written to each subsequent disk
- Each block of the parity disk is the XOR value of the corresponding blocks on the data disks
- Not used often in the real world

RAID 4


1001
0000
Write 0011 to disk 3
RAID
Controller
0011
1111
B1
C1
D1
1100
B2
C2
D2
1010
B3
C3
D3
1001
BP
CP
DP

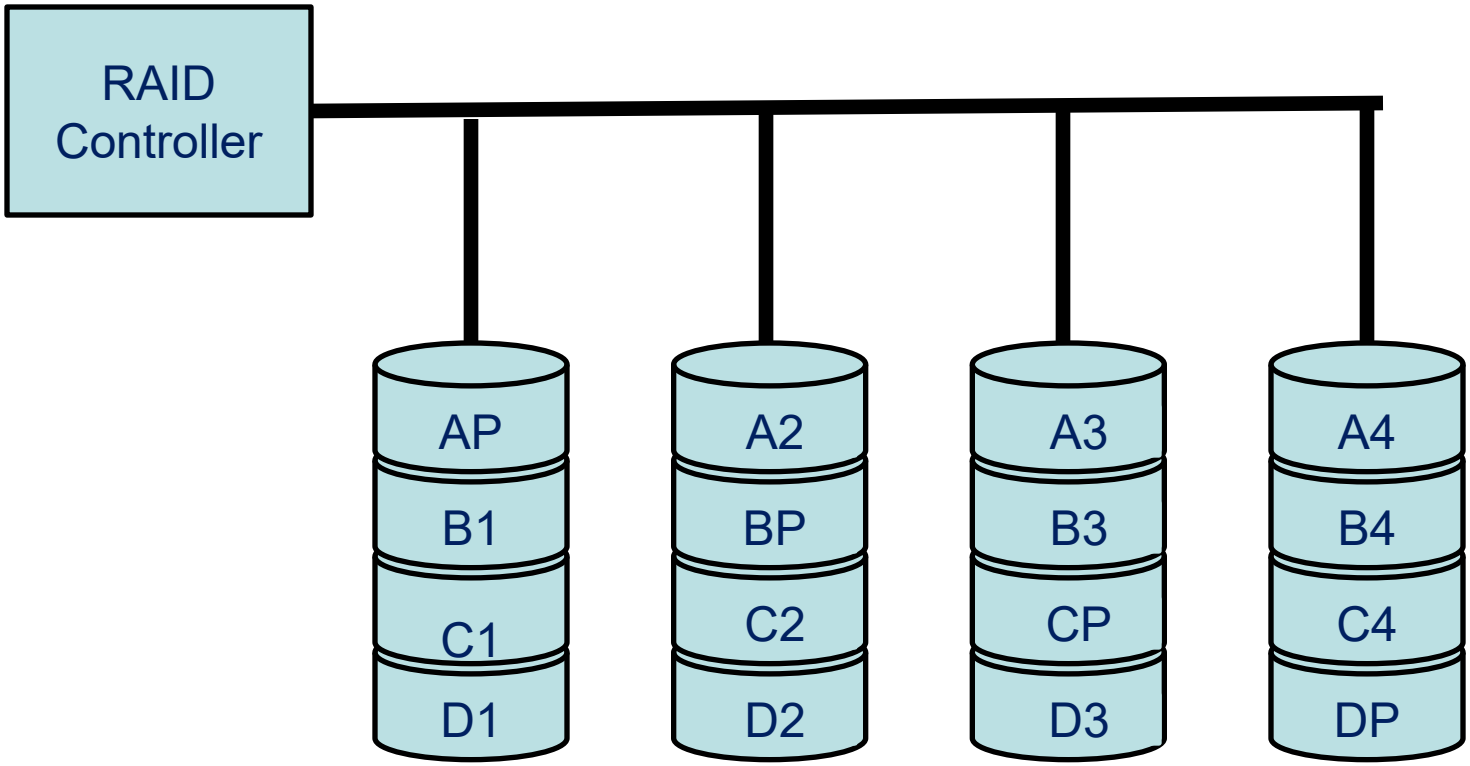
RAID 5

- Striped disks with interleaved parity
- Much like RAID 4 except that parity blocks are spread over every disk

Using a RAID-5 Volume

- Fault tolerance is better for a RAID-5 volume
  - Than for a simple striped volume
- A RAID-5 volume requires a minimum of three disk drives
- Parity information is distributed on each disk
  - If one disk fails, the information on that disk can be reconstructed
  - The parity used by Microsoft is Boolean (true/false, one/zero) logic

RAID 5



Using a RAID-5 Volume

- The performance is not as fast as with a striped volume
  - Takes longer to write the data and calculate the parity block for each row
- Accessing data through disk reads is as fast as a striped volume
- A RAID-5 volume is particularly useful in a client/server system that uses a separate database for queries and creating reports
- Use the Disk Management tool to create a RAID-5 volume

Data Storage Network

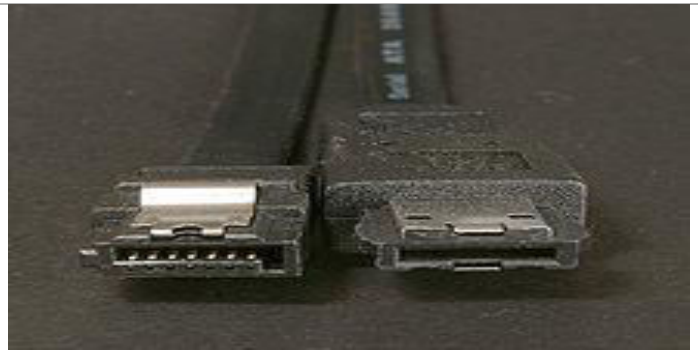
- Direct Access Storage (DAS)
  - SCSI
  - RAID
- Network Attached Storage (NAS)
- Storage Area Network (SAN)
  - Fiber Channel and
  - Fiber Channel Switch

Quick Overview

	DAS	NAS	SAN
Storage Type	sectors	shared files	blocks
Data Transmission	IDE/SCSI	TCP/IP, Ethernet	Fibre Channel
Access Mode	clients or servers	clients or servers	servers
Capacity (bytes)	10^9	10^9 - 10^{12}	> 10^{12}
Complexity	Easy	Moderate	Difficult
Management Cost (per GB)	High	Moderate	Low

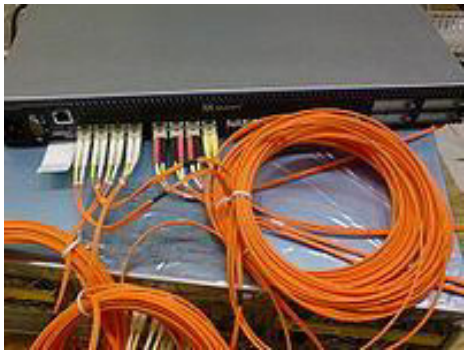
Data Storage Network

- SATA connectors



- SCSI

- FC with SAN-switch

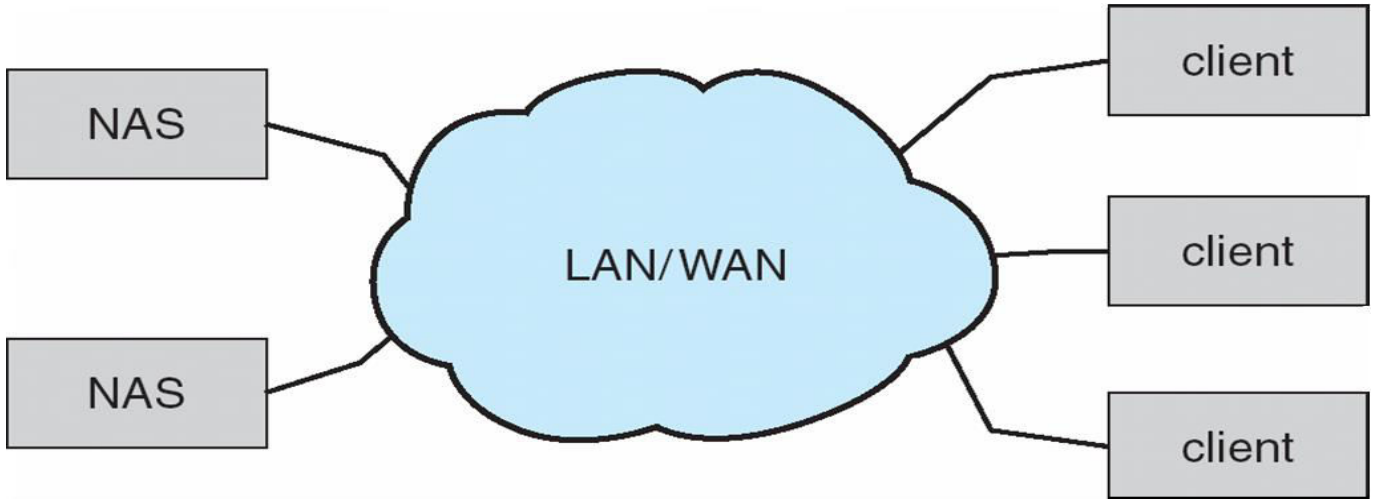


Small Computer System Interface (SCSI)

- From Shugart's 1979 SASI implementation
- An I/O bus for peripheral device, such as hard drives, tape drives, CD-ROM, scanners, etc.
  - an improvement over IDE
- A single SCSI bus connects multiple elements (max 7 or 15).
- High speed data transfer:
  - 5, 10, 20, 100, 320MB/sec, …
- Overlapping I/O capability:
  - Multiple read & write commands can be outstanding simultaneously
  - Different SCSI drives to be processing commands concurrently rather than serially. The data can then be buffered and transferred over the SCSI bus at very high speeds

Network-Attached Storage

- Network-attached storage (NAS) is storage made available over a network rather than over a local connection (such as a bus)
- NFS and CIFS are common protocols
- Implemented via remote procedure calls (RPCs) between host and storage
- New iSCSI protocol uses IP network to carry the SCSI protocol



Network Attached Storage (NAS)

- NAS is a dedicated storage device, and it operates in a client/server mode.
- NAS is connected to the file server via LAN.
- Protocol: NFS (or CIFS) over an IP Network
  - Network File System (NFS) – UNIX/Linux
  - Common Internet File System (CIFS) – Windows Remote file system (drives) mounted on the local system (drives)
    - evolved from Microsoft NetBIOS, NetBIOS over TCP/IP (NBT), and Server Message Block (SMB)
  - SAMBA: SMB on Linux (Making Linux a Windows File Server)
- Advantage: no distance limitation
- Disadvantage: Speed and Latency
- Weakness: Security

Storage Area Network (SAN)

- A Storage Area Network (SAN) is a specialized, dedicated high speed network joining servers and storage, including disks, disk arrays, tapes, etc.
- Storage (data store) is separated from the processors (and separated processing).
- High capacity, high availability, high scalability, ease of configuration, ease of reconfiguration.
- Fiber Channel is the de facto SAN networking architecture, although other network standards could be used.

Storage Area Network (SAN)

- Special/dedicated network for accessing block level data storage
- Multiple hosts attached to multiple storage arrays - flexible