OPERATING SYSTEMS

Input - Output Management and Security - 9

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

OPERATING SYSTEMS

Case Study : Windows File System - 1

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

OPERATING SYSTEMS

Course Syllabus - Unit 5

10 Hours

Unit-5:Unit 5: IO Management and Security

I/O Hardware, polling and interrupts, DMA, Kernel I/O Subsystem and Transforming I/O Requests to Hardware Operations - Device interaction, device driver, buffering System Protection: Goals, Principles and Domain of Protection, Access Matrix, Access control, Access rights. System Security: The Security Problem,Program Threats, System Threats and Network Threats. Case Study: Windows 7/Windows 10

OPERATING SYSTEMS

Course Outline

47	I/O Hardware, polling and interrupts	13.1,13.2
48	DMA	13.2.3
49	Transforming I/O Requests to Hardware Operations, Device interaction, device driver, buffering.	13.5
50	Goals, Principles and Domain of Protection	14.1-14.3
51	Access Matrix	14.4
52	Access control, Access rights	14.5-14.7
53	The Security Problem	15.1
54	Program Threats	15.2
55	System Threats and Network Threats	15.3
56	Case Study : Windows File System	17.5

- Case study : Windows File System - 1

Case study : Windows File System

- The native file system in Windows is **NTFS** .
- It is used for all local volumes.
- Associated USB thumb drives, flash memory on cameras, and external disks may be formatted with the 32-bit FAT file system for portability.
- A disadvantage is that the FAT file system **does not restrict** file access to authorized users.
- The only solution for securing data with FAT is to run an application to encrypt the data before storing it on the file system.

Case study : Windows File System

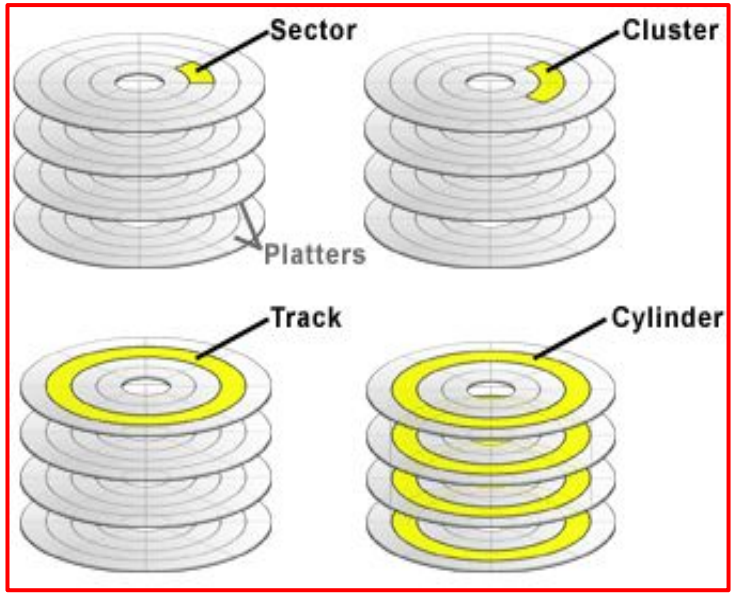
- NTFS uses ACLs to control access to individual files and supports implicit encryption of individual files or entire volumes using Windows BitLocker feature

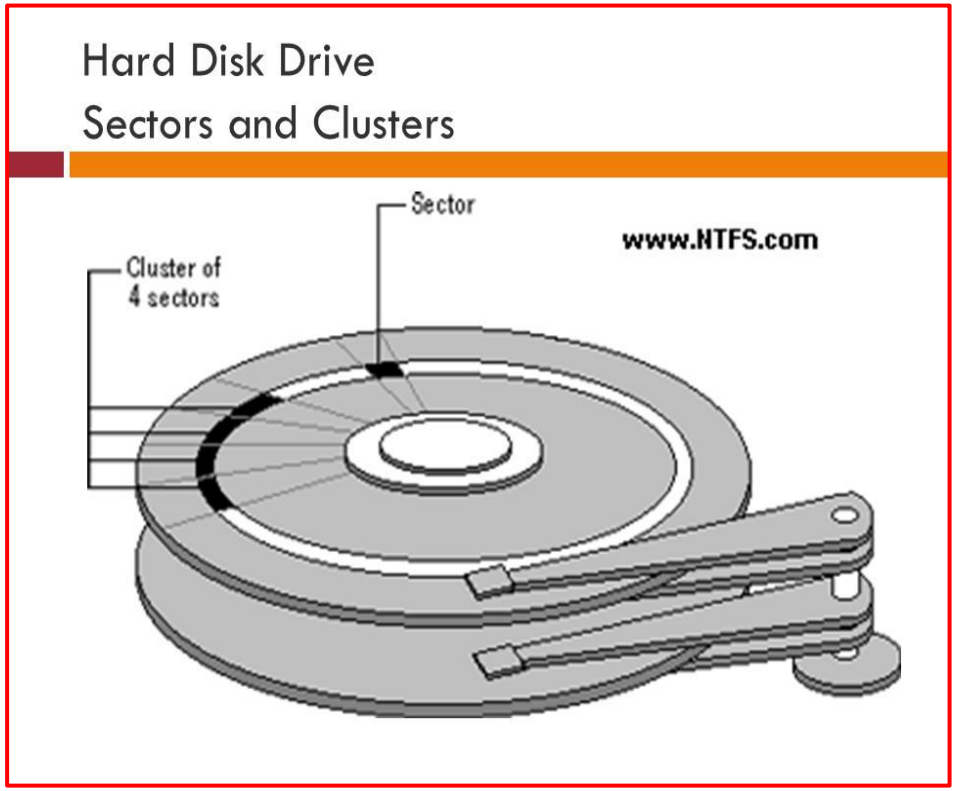
- NTFS implements many other features as well, including
  - Data recovery
  - Fault tolerance
  - Very large files and file systems
  - Multiple data streams
  - UNICODE names
  - Sparse files
  - Journaling
  - Volume shadow copies
  - File compression

Case study : NTFS Internal Layout

- The fundamental entity in NTFS is a **volume**.
- A volume is created by the Windows logical disk management utility and is based on a logical disk partition.
- A volume may occupy a portion of a disk or an entire disk, or may span several disks.
- NTFS does not deal with individual sectors of a disk but instead uses **clusters** as the units of disk allocation.
- A **cluster** is a number of disk sectors that is a **power of 2.**
- The cluster size is configured when an NTFS file system is formatted.
- The default cluster size is based on the volume size —**4 KB** for volumes larger than 2 GB .

Case study : NTFS Internal Layout

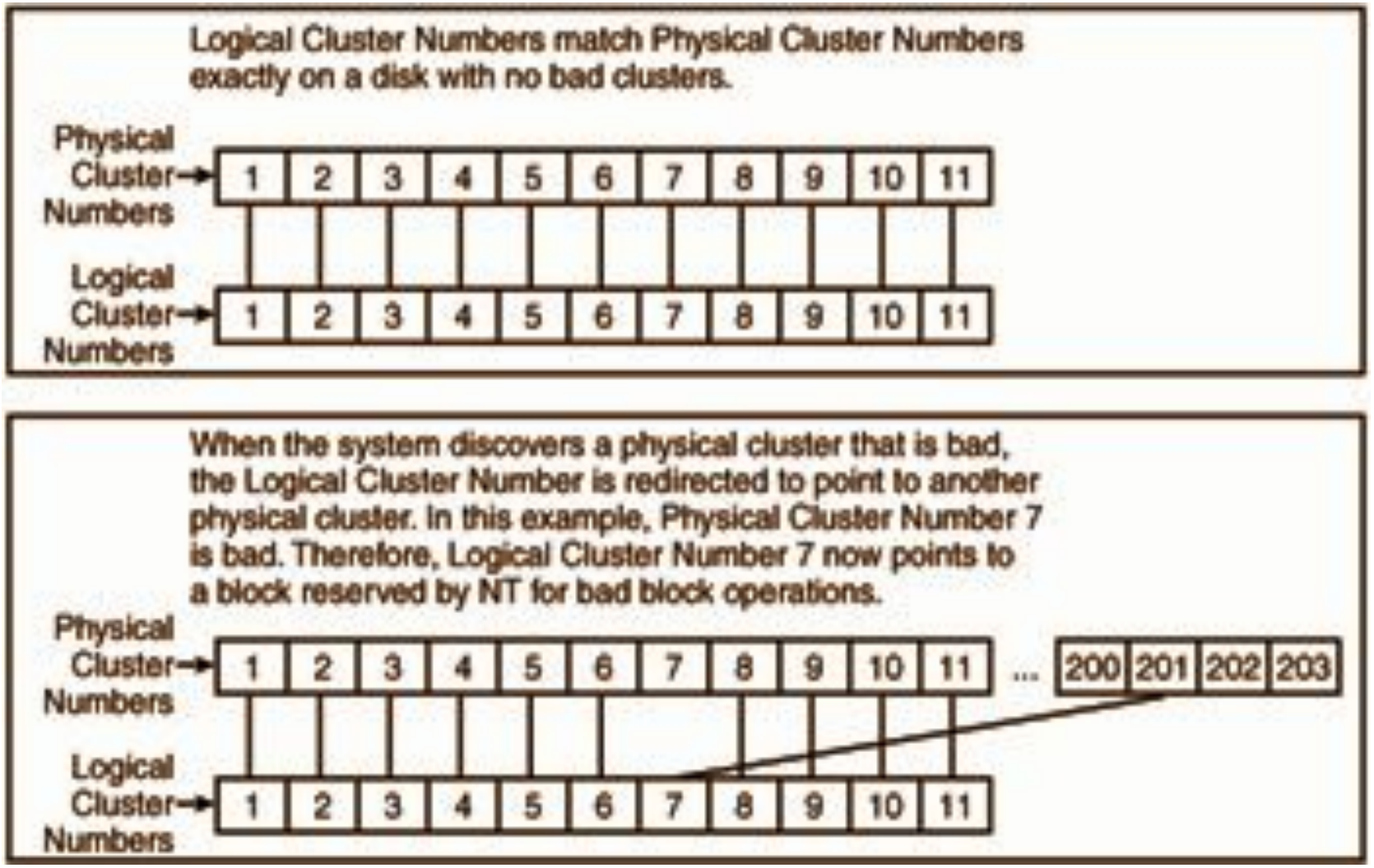




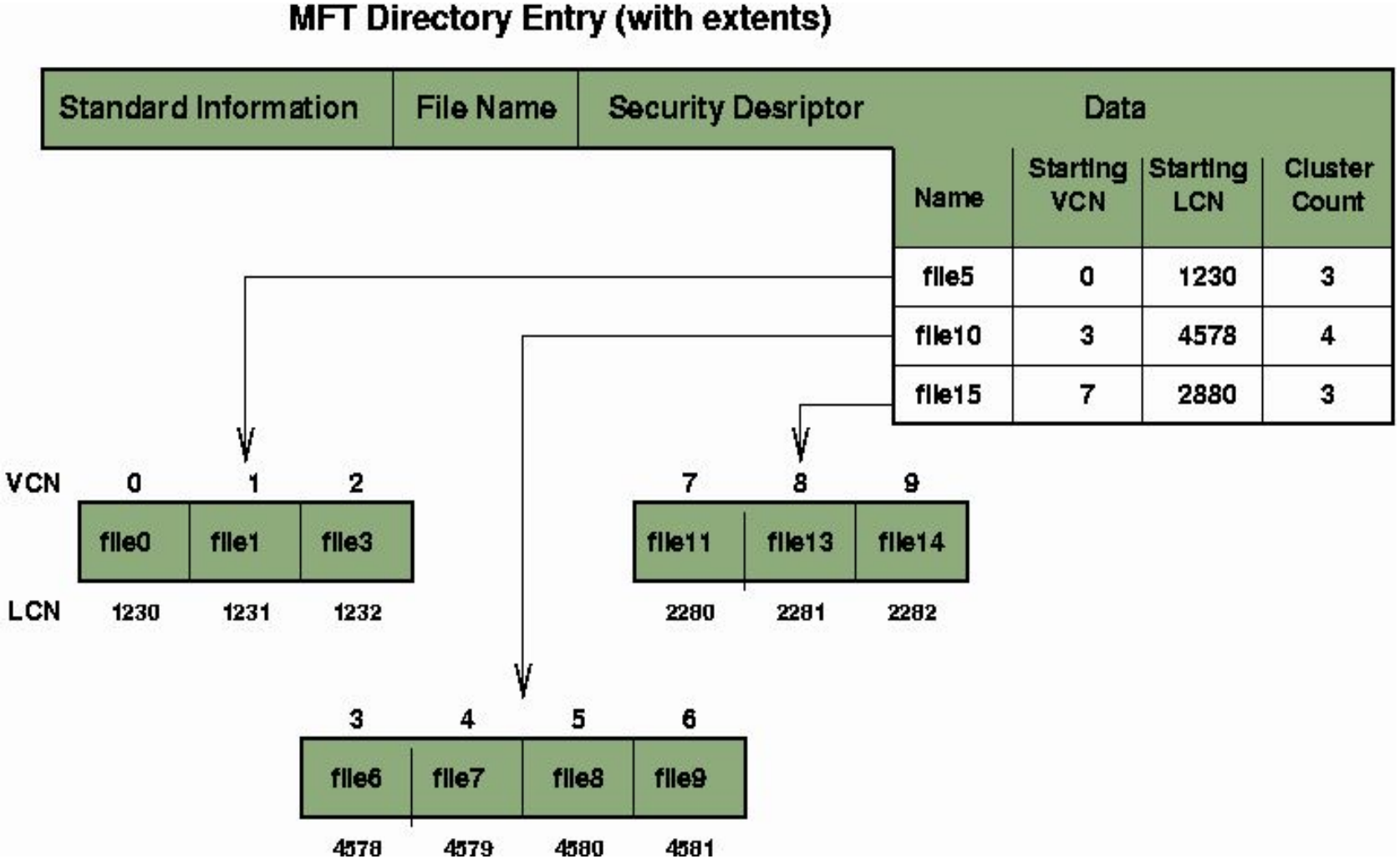
Case study : NTFS Internal Layout

- NTFS uses logical cluster numbers (LCNs) as disk addresses.
- It assigns them by numbering clusters from the beginning of the disk to the end.
- A file in NTFS is not a simple byte stream as it is in UNIX ; rather, it is a structured object consisting of typed attributes.
- Each attribute of a file is an independent byte stream that can be created, deleted, read, and written.
- User data are stored in **data attributes**.
- Attributes may be added as necessary and are accessed using a file-name:attribute syntax.

Case study : NTFS Internal Layout



Case study : NTFS Internal Layout



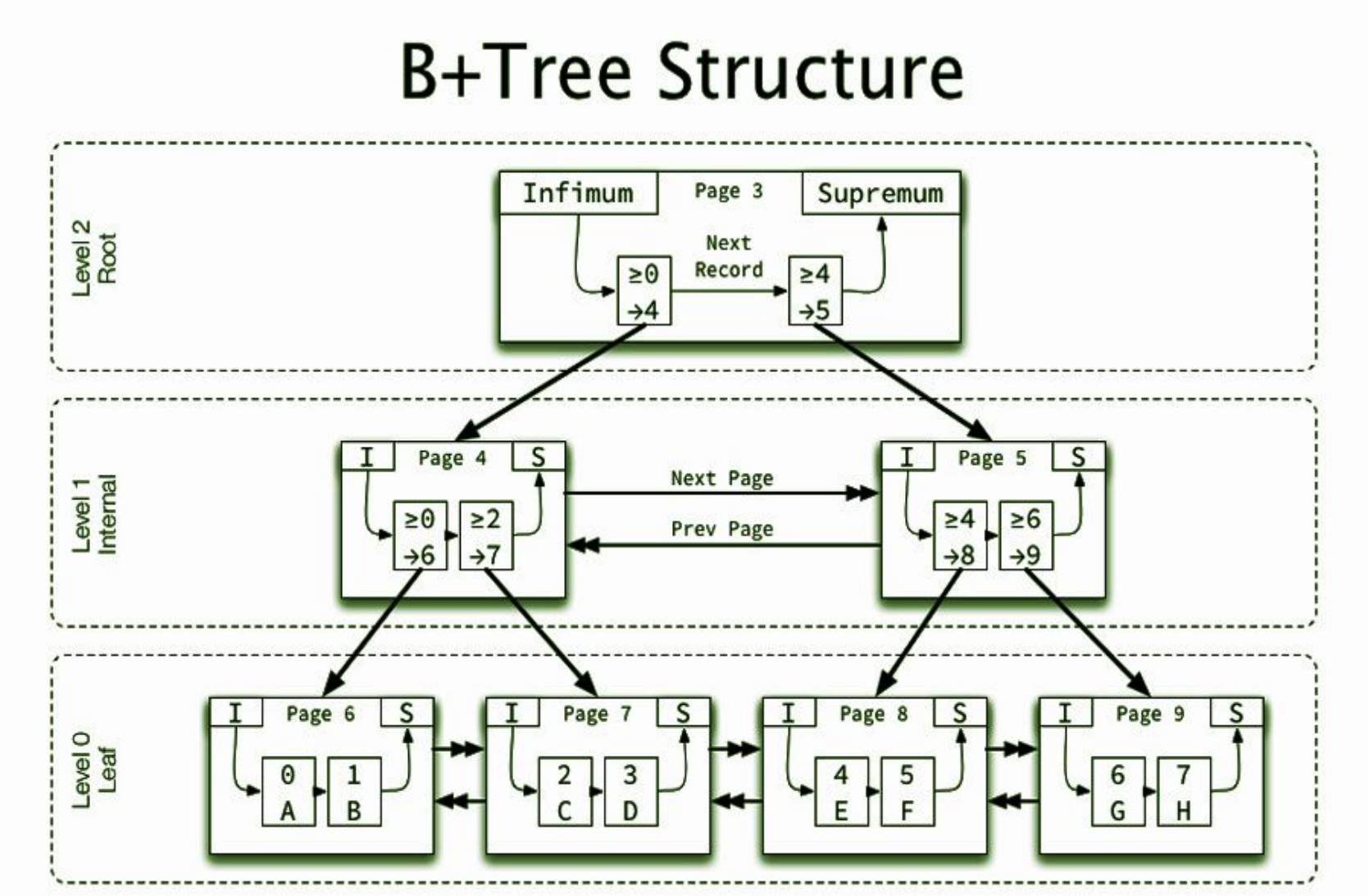
Case study : NTFS Internal Layout

- Every file in NTFS is described by one or more records in an array stored in a special file called the **master file table (MFT)**.
- Small attributes are stored in the MFT record itself and are called **resident attributes.**
- Large attributes, such as the unnamed bulk data, are called **nonresident attributes** and are stored in one or more contiguous extents on the disk.
- Each file in an NTFS volume has a unique ID called a **file reference**.
- The file reference is a 64-bit quantity that consists of a **48-bit file number** and a **16-bit sequence number**.
- The sequence number enables NTFS to perform internal consistency checks, such as catching a stale reference to a deleted file after the MFT entry has been reused for a new file.

Case study : NTFS B+ Tree

- The NTFS namespace is organized as a hierarchy of directories.
- Each directory uses a data structure called a **B+ tree** to store an index of the file names in that directory.
- In a B+ tree, the length of every path from the root of the tree to a leaf is the same, and the cost of reorganizing the tree is eliminated.
- The index root of a directory contains the top level of the B+ tree.
- For a large directory, this top level contains pointers to disk extents that hold the remainder of the tree.
- Each entry in the directory contains the name and file reference of the file, as well as a copy of the update timestamp and file size taken from the file's resident attributes in the MFT.
- Copies of this information are stored in the directory so that a directory listing can be efficiently generated.

Case study : NTFS B+ Tree



Case study : NTFS Metadata

- The NTFS volume **metadata** are all stored in files.
- The first file is the **MFT** .
- The second file, which is used during recovery if the MFT is damaged, contains a copy of the first 16 entries of the MFT .

Case study : NTFS Metadata

- The next few files are also special in purpose
  - The **log file** records all metadata updates to the file system
  - The **volume file** contains the name of the volume, the version of NTFS that formatted the volume, and a bit that tells whether the volume may have been corrupted and needs to be checked for consistency using the **chkdsk** program.

Case study : NTFS Metadata

- The next few files are also special in purpose
  - The **attribute-definition table** indicates which attribute types are used in the volume and what operations can be performed on each of them.
  - The **root directory** is the top-level directory in the file-system hierarchy.

Case study : NTFS Metadata

- The next few files are also special in purpose
  - The **bitmap file** indicates which clusters on a volume are allocated to files and which are free.
  - The **boot file** contains the startup code for Windows and must be located at a particular disk address so that it can be found easily by a simple ROM bootstrap loader. The boot file also contains the physical address of the MFT .
  - The **bad-cluster file** keeps track of any bad areas on the volume; NTFS uses this record for error recovery.

- Case study : Windows File System - 1

THANK YOU

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

nitin.pujari@pes.edu

For Course Deliverables by the Anchor Faculty click on www.pesuacademy.com