Understanding File System

MODULE 6

Contents

Understanding File System

6.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- Explain the importance of file system
- Define common file systems
- Explain types of file systems

6.2 FILE SYSTEM

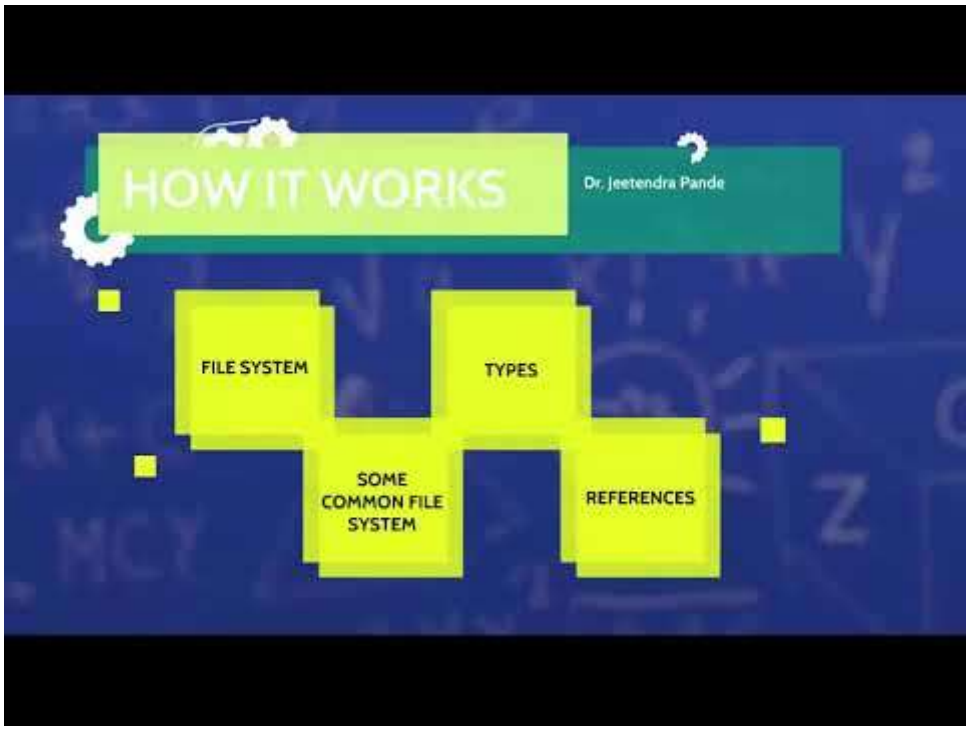
Even though hard drives can be very small, they still contain millions of bits and therefore need to be organized so that information can be located[1]. This is the purpose of the file system. Remember that a hard drive is made up of several circular platters rotating around an axis. The tracks (concentric areas written to on either side of the platter) are divided into pieces called sectors (each 512 bytes in size). Logical formatting of a disk allows a file system to be created on the disk, which in turn will allow an operating system (DOS, Windows 9x, UNIX, etc) to use the disk space to store and use files. The file system is based on management of clusters, the smallest disk unit that the operating system is able to manage.

A cluster consists of one or more sectors, so the larger the cluster size, the fewer entities the operating system will have to manage. On the other hand, since an operating system only knows how to manage whole allocation units (i.e. a file occupies a whole number of clusters), the more sectors per cluster, the more wasted space there will be. This is why the choice of file system is important.

Files systems and the operating system: In reality, the choice of file system depends first of all on the operating system that you are using. In general, the more recent the operating system, the greater the number of files it will support. So, under DOS and on the first versions of Windows 95, FAT16 is required. Starting with Windows 95 OSR2, you have the choice between FAT16 and FAT32 file systems. If the partition size is greater than 2GB, then FAT file systems are excluded and you need to use the FAT32 system (or modify the size of the partition). Below this limit, FAT16 is recommended for partitions with a capacity of less than 500Mb, otherwise it is preferable to use FAT32.

[1] http://ccm.net/contents/624-the-file-system

VIDEO LECTURE



In the case of Windows NT (up until version 4) you have the choice between the FAT16 system and NTFS, FAT32 is not supported. In general, the NTFS system is recommended as it provides higher security and better performance than the FAT system. Microsoft actually recommends using a small FAT-type partition (of between 250 and 500MB) for the operating system, so as to be able to boot from a bootable DOS floppy disk in case of a catastrophe, and to use a second partition for storing your data.

Under Windows NT5, there are more choices as it accepts FAT16, FAT32 and NTFS partitions. Once again, the more recent file system (NTFS 5) is recommended, as it offers many more features than the FAT systems. For the same reasons given above, you can still choose to have a FAT-type partition.

Table 1: Operating system and choice of file system

Operation system	File system types supported
Dos	FAT16
Windows 95	FAT16
Windows 95 OSR2	FAT16, FAT32
Windows 98	FAT16, FAT32
Windows NT4	FAT, NTFS (version 4)
Windows 2000/XP	FAT, FAT16, FAT32, NTFS (versions 4 and 5)

Linux	Ext2, Ext3, ReiserFS, Linux Swap(, FAT16, FAT32, NTFS)
MacOS	HFS (Hierarchical File System), MFS (Macintosh File System)
OS/2	HPFS (High Performance File System)
SGI IRIX	XFS
FreeBSD, OpenBSD	UFS (Unix File System)
Sun Solaris	UFS (Unix File System)
IBM AIX	JFS (Journaled File System)

Coexistence of several file systems: When several operating systems coexist on the same machine, the problem of choosing a file system is at its worse. Since the file system is tightly coupled to the operating system, when there are several operating systems you must choose a file system for each, bearing in mind that it is possible that data from one operating system may be accessed from another. One solution would be to use FAT partitions for all the systems, making sure that the partitions are no larger than 2 GB. The most appropriate solution would be to use for each OS a partition whose file system is best suited to it, and to use a dedicated FAT16 partition for data to be shared by the different operating systems.

6.3 SOME COMMON FILE SYSTEMS

File systems are your interface to store your data[2]. Modern file systems offer a hierarchical view of your data, though historical file systems have been flat.

6.3.1 FAT

FAT stands for File Allocation Table, it is a relatively old file system, files are limited to 4GB in size and file names are case insensitive. It has the benefit of being widely portable, being available on many platforms. For this reason storage devices are often pre-formatted as FAT, just so less technical users don't assume the device is broken and return it. Its portability makes it useful for USB flash drives and the partition you use for `/boot`. It would be a poor choice for your root file system.

6.3.2 NTFS

This is Microsoft's primary file system. Its data structures don't limit the maximum file size to 4GB. As a commenter pointed out, NTFS is case sensitive, but not through the Windows API, which maintains case insensitivity for compatibility with older software that assumed insensitivity, since it was dealing with FAT.
On Windows it is case preserving, so if you create a file called "Foo", you can read it as "foo", but when you list the contents of the directory, it is shown as "Foo", rather than "FOO", as FAT traditionally does. This makes it a better choice for storage media, as Linux is also able to read it. It is still inadvisable to use NTFS as your root file system on Linux, since it's primary use is

[2] https://yakking.branchable.com/posts/filesystems/

reading disks that are also used by Windows machines, rather than being an installation's root file system.

6.3.3 ext2, ext3 and ext4

The ext file systems are Linux's primary file systems and are usually the default option when installing Linux distributions. Despite having a similar name, they are different beasts. ext2 is rather primitive, only really useful with old bootloaders. **ext3** is more advanced, though active development has moved on to ext4. **ext4** supports journalling, uses extents for its storage and supports extended attributes, where additional metadata can be assigned to a file. There are third-party tools to read ext file systems from Windows, but NTFS support in Linux is better than ext support in Windows.

6.3.4 XFS

XFS is a development from Silicon Graphics. It exceeds ext4's features, including the ability to take snapshots of the logical state of the file system, and was the source of extended attributes. It is available on IRIX and Linux, so portability is not its strong point, hence would not be useful on a USB flash drive, but it is an excellent choice for your root file system.

6.3.5 ZFS

ZFS is a product of Sun Microsystems, later bought by Oracle. It is a very advanced file system, offering all the features mentioned above plus more. This is a copy-on-write file system, unlike the above, which were either journalling, or wrote to the blocks directly. Being copy-on-write allows for deduplication, since if multiple files have the same data, the file system can point both files to the same data, since if it's changed in one file, the other one won't have its data changed. Live file system checking is possible with the scrub command, so downtime is not needed to perform maintenance. It can use multiple physical devices as its storage pool. Its license is incompatible with the Linux kernel, so kernel support is provided by a third-party module, which makes it possible that a kernel update could leave your file system unreadable, since the ZFS kernel module is un-readable. Loading an external kernel module is slower than it being built in, so this impacts boot speed. Despite its complexity, ZFS is also available on the Solaris and BSD unices.

6.3.6 BTRFS

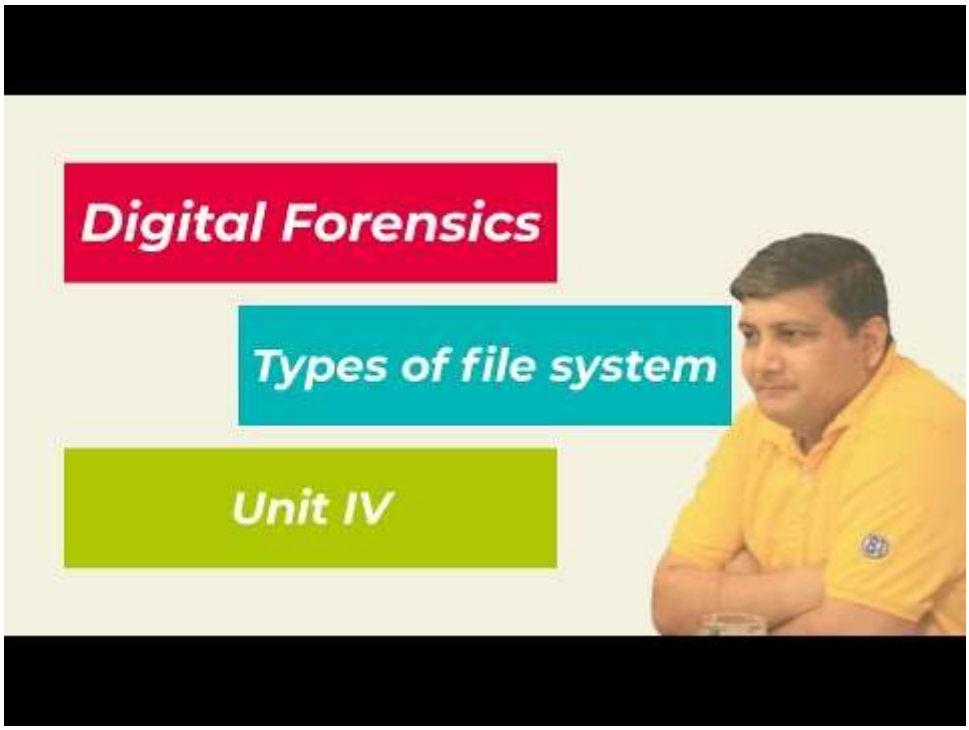
Work on BTRFS was initiated by Oracle to be a competitor to ZFS, this is no longer the motivating factor, since Oracle acquired Sun, but BTRFS is likely to become the default Linux file system in the future. It is nearly as featureful as ZFS, only missing the online deduplication, which the BTRFS developers expect to complete in a couple of Linux kernel releases. Its design allows for a transition between ext and btrfs with the `btrfs-convert` tool, by saving the ext metadata elsewhere and re-mapping ext's data to btrfs extents. It still offers the original file system's data as a read-only disk image that can be mounted. Reverting back to ext is done by reinstating the ext metadata. Unfortunately, it has a reputation of being unstable, corrupting

your data and becoming unusable. This is a natural stage of file system maturity though, and BTRFS is my preferred root file system.

6.4 TYPES OF FILE SYSTEMS

File system types can be classified into disk/tape file systems, network file systems and special-purpose file systems[3].

VIDEO LECTURE



6.4.1 Disk file systems

A *disk file system* takes advantages of the ability of disk storage media to randomly address data in a short amount of time. Additional considerations include the speed of accessing data following that initially requested and the anticipation that the following data may also be requested. This permits multiple users (or processes) access to various data on the disk without regard to the sequential location of the data. Examples include FAT (FAT12, FAT16, FAT32), exFAT, NTFS,HFS and HFS+, HPFS, UFS, ext2, ext 3, ext4, XFS, btrfs, ISO 9660, Files-11,Veritas File System, VMFS, ZFS, ReiserFS and UDF. Some disk file systems are journaling file systems or versioning file systems.

Optical discs: ISO 9660 and Universal Disk Format (UDF) are two common formats that target Compact Discs, DVDs and Blu-ray discs. Mount Rainier is an extension to UDF

[3] https://en.wikipedia.org/wiki/File_system

supported since 2.6 series of the Linux kernel and since Windows Vista that facilitates rewriting to DVDs.

6.4.2 Flash file systems

A *flash file system* considers the special abilities, performance and restrictions of flash memory devices. Frequently a disk file system can use a flash memory device as the underlying storage media but it is much better to use a file system specifically designed for a flash device.

6.4.3 Tape file systems

A *tape file system* is a file system and tape format designed to store files on tape in a self-describing form. Magnetic tapes are sequential storage media with significantly longer random data access times than disks, posing challenges to the creation and efficient management of a general-purpose file system.

In a disk file system, there is typically a master file directory, and a map of used and free data regions. Any file additions, changes, or removals require updating the directory and the used/free maps. Random access to data regions is measured in milliseconds so this system works well for disks. Tape requires linear motion to wind and unwind potentially very long reels of media. This tape motion may take several seconds to several minutes to move the read/write head from one end of the tape to the other. Consequently, a master file directory and usage map can be extremely slow and inefficient with tape. Writing typically involves reading the block usage map to find free blocks for writing, updating the usage map and directory to add the data, and then advancing the tape to write the data in the correct spot. Each additional file write requires updating the map and directory and writing the data, which may take several seconds to occur for each file.

Tape file systems instead typically allow for the file directory to be spread across the tape intermixed with the data, referred to as *streaming*, so that time-consuming and repeated tape motions are not required to write new data. However, a side effect of this design is that reading the file directory of a tape usually requires scanning the entire tape to read all the scattered directory entries. Most data archiving software that works with tape storage will store a local copy of the tape catalogue on a disk file system, so that adding files to a tape can be done quickly without having to rescan the tape media. The local tape catalogue copy is usually discarded if not used for a specified period of time, at which point the tape must be re-scanned if it is to be used in the future.

IBM has developed a file system for tape called the Linear Tape File System. The IBM implementation of this file system has been released as the open-source IBM Linear Tape File System — Single Drive Edition (LTFS-SDE) product. The Linear Tape File System uses a separate partition on the tape to record the index meta-data, thereby avoiding the problems associated with scattering directory entries across the entire tape.

Tape formatting

Writing data to a tape is often a significantly time-consuming process that may take several hours. Similarly, completely erasing or formatting a tape can also take several hours. With many data tape technologies it is not necessary to format the tape before over-writing new data to the tape. This is due to the inherently destructive nature of overwriting data on sequential media. Because of the time it can take to format a tape, typically tapes are pre-formatted so that the tape user does not need to spend time preparing each new tape for use. All that is usually necessary is to write an identifying media label to the tape before use, and even this can be automatically written by software when a new tape is used for the first time.

6.4.4 Database file systems

Another concept for file management is the idea of a database-based file system. Instead of, or in addition to, hierarchical structured management, files are identified by their characteristics, like type of file, topic, author, or similar rich metadata.

IBM DB2 for i (formerly known as DB2/400 and DB2 for i5/OS) is a database file system as part of the object based IBM i operating system (formerly known as OS/400 and i5/OS), incorporating a single level store and running on IBM Power Systems (formerly known as AS/400 and iSeries), designed by Frank G. Soltis IBM's former chief scientist for IBM i. Around 1978 to 1988 Frank G. Soltis and his team at IBM Rochester have successfully designed and applied technologies like the database file system where others like Microsoft later failed to accomplish. These technologies are informally known as 'Fortress Rochester' and were in few basic aspects extended from early Mainframe technologies but in many ways more advanced from a technological perspective.

Some other projects that aren't "pure" database file systems but that use some aspects of a database file system:

- Many Web content management systems use a relational DBMS to store and retrieve files. For example, XHTML files are stored as XML or text fields, while image files are stored as blob fields; SQL SELECT (with optional XPath) statements retrieve the files, and allow the use of a sophisticated logic and more rich information associations than "usual file systems". Many CMSs also have the option of storing only metadata within the database, with the standard filesystem used to store the content of files.
- Very large file systems, embodied by applications like Apache Hadoop and Google File System, use some *database file system* concepts.

6.4.5 Transactional file systems

Some programs need to update multiple files "all at once". For example, a software installation may write program binaries, libraries, and configuration files. If the software installation fails, the program may be unusable. If the installation is upgrading a key system utility, such as the command shell, the entire system may be left in an unusable state.

Transaction processing introduces the isolation guarantee, which states that operations within a transaction are hidden from other threads on the system until the transaction commits, and that

interfering operations on the system will be properly serialized with the transaction. Transactions also provide the atomicity guarantee, ensuring that operations inside of a transaction are either all committed or the transaction can be aborted and the system discards all of its partial results. This means that if there is a crash or power failure, after recovery, the stored state will be consistent. Either the software will be completely installed or the failed installation will be completely rolled back, but an unusable partial install will not be left on the system.

Windows, beginning with Vista, added transaction support to NTFS, in a feature called Transactional NTFS, but its use is now discouraged. There are a number of research prototypes of transactional file systems for UNIX systems, including the Valor file system, Amino, LFS, and a transactional ext3 file system on the TxOS kernel, as well as transactional file systems targeting embedded systems, such as TFFS.

Ensuring consistency across multiple file system operations is difficult, if not impossible, without file system transactions. File locking can be used as a concurrency control mechanism for individual files, but it typically does not protect the directory structure or file metadata. For instance, file locking cannot prevent TOCTTOU race conditions on symbolic links. File locking also cannot automatically roll back a failed operation, such as a software upgrade; this requires atomicity.

Journaling file systems are one technique used to introduce transaction-level consistency to file system structures. Journal transactions are not exposed to programs as part of the OS API; they are only used internally to ensure consistency at the granularity of a single system call.

Data backup systems typically do not provide support for direct backup of data stored in a transactional manner, which makes recovery of reliable and consistent data sets difficult. Most backup software simply notes what files have changed since a certain time, regardless of the transactional state shared across multiple files in the overall dataset. As a workaround, some database systems simply produce an archived state file containing all data up to that point, and the backup software only backs that up and does not interact directly with the active transactional databases at all. Recovery requires separate recreation of the database from the state file, after the file has been restored by the backup software.

6.4.6 Network file systems

A *network file system* is a file system that acts as a client for a remote file access protocol, providing access to files on a server. Programs using local interfaces can transparently create, manage and access hierarchical directories and files in remote network-connected computers. Examples of network file systems include clients for the NFS, AFS, SMB protocols, and file-system-like clients for FTP and WebDAV.

6.4.7 Shared disk file systems

A *shared disk file system* is one in which a number of machines (usually servers) all have access to the same external disk subsystem (usually a SAN). The file system arbitrates access to that

subsystem, preventing write collisions. Examples include GFS2 from Red Hat, GPFS from IBM, SFS from DataPlow, CXFS from SGI and StorNext from Quantum Corporation.

6.4.8 Special file systems

A *special file system* presents non-file elements of an operating system as files so they can be acted on using file system APIs. This is most commonly done in Unix-like operating systems, but devices are given file names in some non-Unix-like operating systems as well.

Device file systems

A *device file system* represents I/O devices and pseudo-devices as files, called device files. Examples in Unix-like systems include devfs and, in Linux 2.6 systems, udev. In non-Unix-like systems, such as TOPS-10 and other operating systems influenced by it, where the full filename or pathname of a file can include a device prefix, devices other than those containing file systems are referred to by a device prefix specifying the device, without anything following it.

Other special file systems

- In the Linux kernel, *configfs* and *sysfs* provide files that can be used to query the kernel for information and configure entities in the kernel.
- *procfs* maps processes and, on Linux, other operating system structures into a filespace.

6.4.9 Minimal file system / audio-cassette storage

The late 1970s saw the development of the microcomputer. Disk and digital tape devices were too expensive for hobbyists. An inexpensive basic data storage system was devised that used common audio cassette tape. When the system needed to write data, the user was notified to press "RECORD" on the cassette recorder, then press "RETURN" on the keyboard to notify the system that the cassette recorder was recording. The system wrote a sound to provide time synchronization, then modulated sounds that encoded a prefix, the data, a checksum and a suffix. When the system needed to read data, the user was instructed to press "PLAY" on the cassette recorder. The system would *listen* to the sounds on the tape waiting until a burst of sound could be recognized as the synchronization. The system would then interpret subsequent sounds as data. When the data read was complete, the system would notify the user to press "STOP" on the cassette recorder. It was primitive, but it worked (a lot of the time). Data was stored sequentially, usually in an unnamed format, although some systems (such as the Commodore PET series of computers) did allow the files to be named. Multiple sets of data could be written and located by fast-forwarding the tape and observing at the tape counter to find the approximate start of the next data region on the tape. The user might have to listen to the sounds to find the right spot to begin playing the next data region. Some implementations even included audible sounds interspersed with the data.

6.4.10 Flat file systems

In a flat file system, there are no subdirectories. When floppy disk media was first available this type of file system was adequate due to the relatively small amount of data space

available. CP/M machines featured a flat file system, where files could be assigned to one of 16 *user areas* and generic file operations narrowed to work on one instead of defaulting to work on all of them. These user areas were no more than special attributes associated with the files, that is, it was not necessary to define specific quota for each of these areas and files could be added to groups for as long as there was still free storage space on the disk. The early Apple Macintosh also featured a flat file system, the Macintosh File System. It was unusual in that the file management program (Macintosh Finder) created the illusion of a partially hierarchical filing system on top of EMFS. This structure required every file to have a unique name, even if it appeared to be in a separate folder. While simple, flat file systems become awkward as the number of files grows and makes it difficult to organize data into related groups of files.

A recent addition to the flat file system family is Amazon's S3, a remote storage service, which is intentionally simplistic to allow users the ability to customize how their data is stored. The only constructs are buckets (imagine a disk drive of unlimited size) and objects (similar, but not identical to the standard concept of a file). Advanced file management is allowed by being able to use nearly any character (including '/') in the object's name, and the ability to select subsets of the bucket's content based on identical prefixes

6.5 SUMMARY

1. A hard disk drive (HDD), hard disk, hard drive or fixed disk is a data storage device used for storing and retrieving digital information using one or more rigid rapidly rotating disks (platters) coated with magnetic material.
2. Data is accessed in a random-access manner, meaning that individual blocks of data can be stored or retrieved in any order rather than sequentially. HDDs retain stored data even when powered off.
3. The primary characteristics of an HDD are its capacity and performance.
4. An HDD records data by magnetizing a thin film of ferromagnetic material on a disk.
5. A typical HDD design consists of a spindle that holds flat circular disks, also called platters, which hold the recorded data.
6. The two most common form factors for modern HDDs are 3.5-inch, for desktop computers, and 2.5-inch, primarily for laptops.
7. HDDs are connected to systems by standard interface cables such as PATA (Parallel ATA), SATA (Serial ATA), USB or SAS (Serial attached SCSI) cables.
8. The purpose of low-level formatting is to divide the disk surface into basic elements viz. tracks, sectors and cylinders.
9. Operating systems use different file systems, so the type of logical formatting will depend on the operating system you install.
10. The partitioning of a hard drive occurs after the drive has been physically formatted but before it is logically formatted.
11. There are three types of partitions: primary partitions, extended partitions and logical drives.
12. Partitioning is the process of writing the sectors that will make up the partition table.
13. The unused space at the end of a file in a file system that uses fixed size.

14. A lost cluster is a series of clusters on the hard disk drive that are not associated with a particular file.
15. A bad sector is a sector on a computer's disk drive or flash memory that is either inaccessible or unwriteable due to permanent damage, such as physical damage to the disk surface or failed flash memory transistors.
16. The boot sector is the first sector of a hard drive (cylinder 0, head 0, sector 1), it contains the main partition table and the code, called the boot loader, which, when loaded into memory, will allow the system to boot up.
17. Firmware is the software that is programmed into Electrically Erasable Programmable Read-Only Memory (EEPROM).
18. The purpose of a bootloader is to load the initial kernel and supporting modules into memory.
19. The kernel is the main component of any operating system. The kernel acts as the lowest-level intermediary between the hardware on your computer and the applications running on your computer.

6.6 CHECK YOUR PROGRESS

1. Fill in the blanks

i. The primary characteristics of an HDD are its _____ and _____.
ii. An HDD records data by magnetizing a thin film of _____ material on a disk.
iii. SCSI stands for _____.
iv. A _____ refers to all the data located on the same track of different platters.
v. _____ is the process of writing the sectors that will make up the partition table.
vi. BIOS stands for _____.
vii. _____is the software that is programmed into Electrically Erasable Programmable Read-Only Memory.
viii. CMOS stands for _____.
ix. The purpose of a _____is to load the initial kernel and supporting modules into memory.
x. _____is a development from Silicon Graphics.
xi. IBM has developed a file system for tape called the _____.
xii. A _____is a file system that acts as a client for a remote file access protocol, providing access to files on a server.

2. State True or False

i. The data is read from the disk by detecting the transitions in magnetization. True
ii. The platters are made from a non-magnetic material, usually aluminum alloy, glass, or ceramic.
iii. The 40-pin IDE/ATA connection transfers 32 bits of data at a time on the data cable.

iv. EIDE was an unofficial update (by Western Digital) to the original IDE standard.
v. EIDE have DMA transfer functionality.
vi. Physical level formatting is also known as high level formatting.
vii. The tracks are the concentric areas written on both sides of a platter.
viii. The partitioning of a hard drive occurs after the drive has been physically formatted but before it is logically formatted.
ix. The partition table stores information about how the drive is logically laid out.
x. The file system is based on management of clusters.
xi. The choice of file system does not depends on the operating system that you are using.
xii. NTFS system provides higher security and better performance than the FAT system.
xiii. Several operating systems coexist on the same machine.
xiv. The BIOS does not have its own memory storage.

6.7 ANSWERS TO CHECK YOUR PROGRESS

1. Fill in the blanks

i. capacity , performance.
ii. ferromagnetic
iii. Small Computer System Interface.
iv. cylinder
v. Partitioning
vi. Basic Input/Output System.
vii. Firmware .
viii. Complimentary Metal Oxide Semiconductor.
ix. bootloader
x. XFS
xi. Linear Tape File System.
xii. network file system

2. State True or False

i. True
ii. True
iii. False
iv. True
v. True
vi. False
vii. True
viii. True

ix. True

x. True

xi. False

xii. True

xiii. True

xiv. False

6.8 FURTHUR READING

CARRIER, B. FILE SYSTEM FORENSIC ANALYSIS.

Computer Storage. (n.d.). Retrieved Sep. 21, 2015, from http://www.utilizewindows.com/: http://www.utilizewindows.com/pc-fundamentals/storage/333-hardware-and-software-disk-optimization

Cooper, M. (2014, March). *General overview of the Linux file system*. Retrieved Oct. 22, 2015, from Linux Documentation Project : http://www.tldp.org/LDP/intro-linux/html/sect_03_01.html

Different Types Of PC Hard Disk Drives (HDD). (n.d.). Retrieved Oct. 21, 2015, from SAYPOINT: http://www.saypoint.net/2012/05/different-types-of-pc-hard-disk.html

File Systems: FAT, NTFS, and HFS+. (n.d.). Retrieved Oct. 22, 2015, from www.study.com: http://study.com/academy/lesson/files-systems-fat-ntfs-hfs-and-ffs.html

Hagen, W. V. (2002). *Linux Filesystems.* SAMS.

How Things Work/Hard Drive. (2012, April 2012). Retrieved Oct. 22, 2015, from Wikibooks: https://en.wikibooks.org/wiki/Wikijunior:How_Things_Work/Hard_Drive

Silberschatz, Galvin, & Gagne. (2006). *Operating System Principles.* Wiley.

Singh, A. (2006). *Mac OS X Internals: A Systems Approach.*

Smidsrød, R. (n.d.). *The fundamentals of network booting*. Retrieved Oct. 22, 2015, from Networkboot: http://networkboot.org/fundamentals/.

Stages of linux boot process. (2012, Aug. 11). Retrieved Oct. 21, 2015, from https://pacesettergraam.wordpress.com: https://pacesettergraam.wordpress.com/2012/08/11/stages-of-linux-boot-process/

Thakur, D. (n.d.). *What is Booting? Type of Booting*. Retrieved Oct. 22, 2015, from Computer Notes: http://ecomputernotes.com/fundamental/disk-operating-system/what-is-booting-type-of-booting

Windows Disk File Systems. (2010). General Books LLC.

Witjes, C., & Stremlau, A. (n.d.). *The Windows 7 Boot Process (sbsl)*. Retrieved Oct. 22, 2015, from http://social.technet.microsoft.com/: http://social.technet.microsoft.com/wiki/contents/articles/11341.the-windows-7-boot-process-sbsl.aspx

6.9 MODEL QUESTIONS

1. What is Hard Disk Drive? What are its main characteristics?
2. Explain the working of HDD.
3. Explain various interfaces of HDD in detail.
4. Differentiate between high level formatting and low level formatting.
5. What is cyclic redundancy check (CRC)?
6. Define the terms:
    a. Slack space
    b. Lost cluster
    c. Bad sector
7. What is master boot record?
8. What is booting? Explain the booting process of Window 7 in detail.
9. What is a file system? Why it is used?
10. Give the details of file systems that different Operating System supports.
11. What is journaling?
12. What are different types of File Systems? Explain in detail.
13. What is flat file system?

REFERENCES, ARTICLE SOURCE & CONTRIBUTORS

Bad Sector. (2015, Sep. 26). Retrieved Oct. 21, 2015, from Wikipedia: https://en.wikipedia.org/wiki/Bad_sector

BootX (Apple). (2015, May 25). Retrieved Oct. 21, 2015, from Wikipedia: https://en.wikipedia.org/wiki/BootX_(Apple)

Computer Storage. (n.d.). Retrieved Sep. 21, 2015, from http://www.utilizewindows.com/: http://www.utilizewindows.com/pc-fundamentals/storage/333-hardware-and-software-disk-optimization

Different Types Of PC Hard Disk Drives (HDD). (n.d.). Retrieved Oct. 21, 2015, from SAYPOINT: http://www.saypoint.net/2012/05/different-types-of-pc-hard-disk.html

File system. (2015, Oct. 18). Retrieved Oct. 21, 2015, from Wikipedia: https://en.wikipedia.org/wiki/File_system

File systems. (2014, Jan.). Retrieved Oct. 21, 2015, from https://yakking.branchable.com/: https://yakking.branchable.com/posts/filesystems/

Formatting - Formatting a hard drive. (2015, Oct. 01). Retrieved Oct. 21, 2015, from ccm.net: http://ccm.net/contents/626-formatting-formatting-a-hard-drive

Formatting - Formatting a hard drive. (2015, Sep.). Retrieved Oct. 21, 2015, from ccm.net: http://ccm.net/contents/626-formatting-formatting-a-hard-drive

Glossary of digital forensics terms. (2015, Sep. 09). Retrieved Oct. 21, 2015, from Wikipedia: https://en.wikipedia.org/wiki/Glossary_of_digital_forensics_terms

Hard disk drive. (2015, Oct. 16). Retrieved Oct. 21, 2015, from Wikipedia: https://en.wikipedia.org/wiki/Hard_disk_drive

Stages of linux boot process. (2012, Aug. 11). Retrieved Oct. 21, 2015, from https://pacesettergraam.wordpress.com: https://pacesettergraam.wordpress.com/2012/08/11/stages-of-linux-boot-process/

The file system. (2015). Retrieved Oct. 21, 2015, from ccm.net: http://ccm.net/contents/624-the-file-system

Vig, A. (2013). *The boot process*. Retrieved Oct. 21, 2015, from http://ops-school.readthedocs.org: http://ops-school.readthedocs.org/en/latest/boot_process_101.html

EXPERT PANEL

Dr. Jeetendra Pande, Associate Professor- Computer Science, School of Computer Science & IT, Uttarakhand Open University, Haldwani

Dr. Ajay Prasad, Sr. Associate Professor, University of Petroleum and Energy Studies, Dehradun

Dr. Akashdeep Bharadwaj, Professor, University of Petroleum and Energy Studies, Dehradun

Mr. Sridhar Chandramohan Iyer, Assistant Professor- Universal College of Engineering, Kaman, Vasai, University of Mumbai

Mr. Rishikesh Ojha, Digital Forensics and eDiscovery Expert

Ms. Priyanka Tewari, IT Consultant

Mr. Ketan Joglekar, Assistant Professor, GJ College, Maharastra

Dr. Ashutosh Kumar Bhatt, Associate Professor, Uttarakhand Open University, Haldwani

Dr. Sangram Panigrahi, Assistant Professor, Siksha 'O' Anusandhan, Bhubaneswar

This MOOC has been prepared with the support of