2.2. Working with Windows and DOS Systems

Purpose and structure of file systems. Understanding File Systems

- **File system**
  - – Gives OS a road map to data on a disk
- Type of file system an OS uses determines how data is stored on the disk
- When you need to access a suspect's computer to acquire or inspect data
  - – You should be familiar with both the computer's OS and file systems

Understanding the Boot Sequence

- Complementary Metal Oxide Semiconductor (CMOS)
  - – Computer stores system configuration and date and time information in the CMOS
- When power to the system is off
- Basic Input/Output System (BIOS) or Extensible Firmware Interface (EFI)
  - – Contains programs that perform input and output at the hardware level
- **Bootstrap process**
  - Contained in ROM, tells the computer how to proceed
  - Displays the key or keys you press to open the CMOS setup screen
- CMOS should be modified to boot from a forensic floppy disk or CD

Understanding Disk Drives

- Disk drives are made up of one or more platters coated with magnetic material
- Disk drive components
  - Geometry
  - Head
  - Tracks
  - Cylinders
  - Sectors

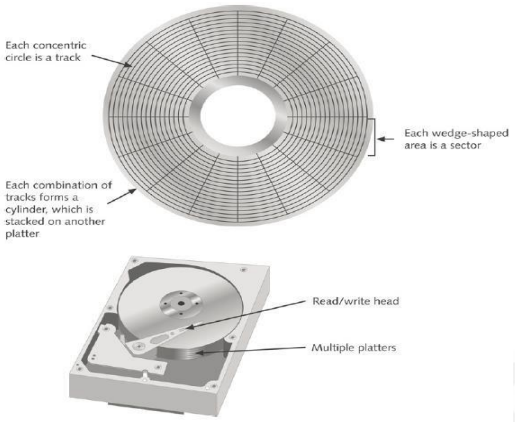

Figure 5-2 Components of a disk drive
© Cengage Learning®

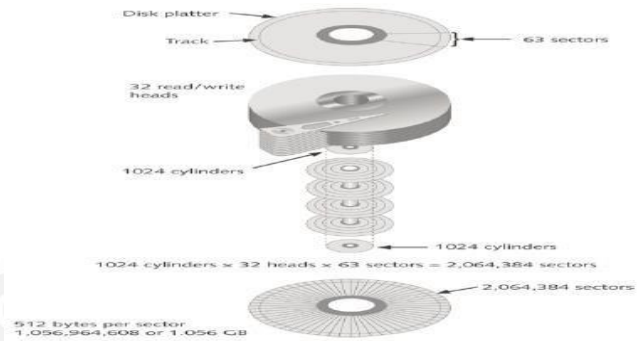

Figure 5-3 CHS calculation
© Cengage Learning®

- Properties handled at the drive's hardware or firmware level
  - Zone bit recording (ZBR)
  - Track density
  - Areal density
  - Head and cylinder skew **Solid-State Storage Devices**
- All flash memory devices have a feature called **wear-leveling**
  - An internal firmware feature used in solid-state drives that ensures even wear of read/writes for all memory cells
- When dealing with solid-state devices, making a full forensic copy as soon as possible is crucial
  - In case you need to recover data from unallocated disk space

Exploring Microsoft File Structures

- In Microsoft file structures, sectors are grouped to form **clusters**
  - Storage allocation units of one or more sectors
- Clusters range from 512 bytes up to 32,000 bytes each
- Combining sectors minimizes the overhead of writing or reading files to a disk
- Clusters are numbered sequentially starting at 0 in NTFS and 2 in FAT
  - First sector of all disks contains a system area, the boot record, and a file structure database
- OS assigns these cluster numbers, called **logical addresses**

- Sector numbers are called **physical addresses**
- Clusters and their addresses are specific to a logical disk drive, which is a disk partition

Disk Partitions

- A partition is a logical drive
- Windows OSs can have three primary partitions followed by an extended partition that can contain one or more logical drives
- Hidden partitions or voids
  – Large unused gaps between partitions on a disk
- Partition gap
  – Unused space between partitions

- The partition table is in the **Master Boot Record (MBR)**
  – Located at sector 0 of the disk drive
- MBR stores information about partitions on a disk and their locations, size, and other important items
- In a hexadecimal editor, such as WinHex, you can find the first partition at offset 0x1BE
  – The file system's hexadecimal code is offset 3 bytes from 0x1BE for the first partition

Examining FAT Disks

- File Allocation Table (FAT)

– File structure database that Microsoft originally designed for floppy disks

- FAT database is typically written to a disk's outermost track and contains:
  – Filenames, directory names, date and time stamps, the starting cluster number, and file attributes
- Three current FAT versions
  – FAT16, FAT32, and exFAT (used by Xbox game systems)
- Cluster sizes vary according to the hard disk size and file system

Drive size	Sectors per cluster	FAT16
8–32 MB	1	512 bytes
32–64 MB	2	1 KB
64–128 MB	4	2 KB
128–256 MB	8	4 KB
256–512 MB	16	8 KB
512–1024 MB	32	16 KB
1024–2048 MB	64	32 KB
2048–4096 MB	128	64 KB

Fig :Sectors and bytes per duster

- Microsoft OSs allocate disk space for files by clusters – Results in **drive slack**
- Unused space in a cluster between the end of an active file and the end of the cluster
- Drive slack includes:
  - **RAM slack** and **file slack**
- An unintentional side effect of FAT16 having large clusters was that it reduced fragmentation

– As cluster size increased

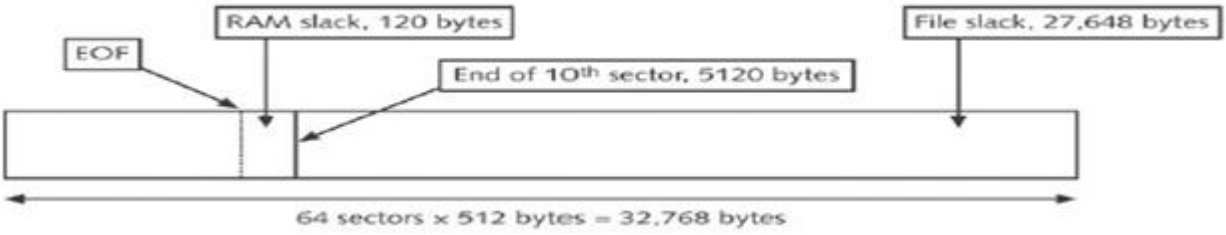

Fig: File Stack space

- When you run out of room for an allocated cluster
  - OS allocates another cluster for your file, which creates more slack space on the disk
- As files grow and require more disk space, assigned clusters are chained together
  - The chain can be broken or fragmented
- When the OS stores data in a FAT file system, it assigns a starting cluster position to a file

Data for the file is written to the first sector of the first assigned cluster

- When this first assigned cluster is filled and runs out of room
  – FAT assigns the next available cluster to the file
- If the next available cluster isn't contiguous to the current cluster
  – File becomes fragmented

Deleting FAT Files

- In Microsoft OSs, when a file is deleted
  – Directory entry is marked as a deleted file
- With the HEX E5 character replacing the first letter of the filename
- FAT chain for that file is set to 0
- Data in the file remains on the disk drive
- Area of the disk where the deleted file resides becomes **unallocated disk space**
  – Available to receive new data from newly created files or other files needing more space

NTFS Disks

- **NT File System (NTFS)**
  - Introduced with Windows NT
  - Primary file system for Windows 8
- Improvements over FAT file systems
  - NTFS provides more information about a file
  - NTFS gives more control over files and folders
- NTFS was Microsoft's move toward a journaling file system
  – It records a transaction before the system carries it out
- In NTFS, everything written to the disk is considered a file
- On an NTFS disk
  - First data set is the **Partition Boot Sector**
  - Next is **Master File Table (MFT)**
- NTFS results in much less file slack space
- Clusters are smaller for smaller disk drives

- NTFS also uses **Unicode**
  - An international data format

Drive size	Sectors per cluster	Cluster size
7–512 MB	8	4 KB
512 MB–1 GB	8	4 KB
1–2 GB	8	4 KB
2 GB–2 TB	8	4 KB
2–16 TB	8	4 KB
16–32 TB	16	8 KB
32–64 TB	32	16 KB
64–128 TB	64	32 KB
128–256 TB	128	64 KB

Fig: Cluster sizes in an NTFS disk

NTFS System Files

- MFT contains information about all files on the disk
  - Including the system files the OS uses
- In the MFT, the first 15 records are reserved for system files
- Records in the MFT are called metadata

Filename	System file	Record position	Description
$Mft	MFT	0	Base file record for each folder on the NTFS volume; other record positions in the MFT are allocated if more space is needed.
$MftMirr	MFT 2	1	The first four records of the MFT are saved in this position. If a single sector fails in the first MFT, the records can be restored, allowing recovery of the MFT.
$LogFile	Log file	2	Previous transactions are stored here to allow recovery after a system failure in the NTFS volume.
$Volume	Volume	3	Information specific to the volume, such as label and version, is stored here.
$AttrDef	Attribute definitions	4	A table listing attribute names, numbers, and definitions.
$	Root filename index	5	This is the root folder on the NTFS volume.
$Bitmap	Cluster bitmap	6	A map of the NTFS partition shows which clusters are in use and which are available.
$Boot	Boot sector	7	Used to mount the NTFS volume during the bootstrap process; additional code is listed here if it's the boot drive for the system.
$BadClus	Bad cluster file	8	For clusters that have unrecoverable errors, an entry of the cluster location is made in this file.
$Secure	Security file	9	Unique security descriptors for the volume are listed in this file. It's where the access control list (ACL) is maintained for all files and folders on the NTFS volume.
$Upcase	Upcase table	10	Converts all lowercase characters to uppercase Unicode characters for the NTFS volume.
$Extend	NTFS extension file	11	Optional extensions are listed here, such as quotas, object identifiers, and reparse point data.
		12–15	Reserved for future use.

Fig: Metedata records in the MFT

MFT and File Attributes

- In the NTFS MFT
  – All files and folders are stored in separate records of 1024 bytes each
- Each record contains file or folder information
  – This information is divided into record fields containing metadata
- A record field is referred to as an attribute ID
- File or folder information is typically stored in one of two ways in an MFT record:
  – Resident and nonresident
- Files larger than 512 bytes are stored outside the MFT
  – MFT record provides cluster addresses where the file is stored on the drive's partition
- Referred to as **data runs**
- Each MFT record starts with a header identifying it as a resident or nonresident attribute
- When a disk is created as an NTFS file structure
  – OS assigns logical clusters to the entire disk partition
- These assigned clusters are called **logical cluster numbers (LCNs)**
  – Become the addresses that allow the MFT to link to nonresident files on the disk's partition
- When data is first written to nonresident files, an LCN address is assigned to the file

  This LCN becomes the file's **virtual cluster number (VCN)**

MFT Structures for File Data

- For the header of all MFT records, the record fields of interest are as follows:
  - *At offset 0x00* - the MFT record identifier FILE
  - *At offset 0x1C to 0x1F* - size of the MFT record
  - *At offset 0x14* - length of the header (indicates where the next attribute starts)
  - *At offset 0x32 and 0x33* - the update sequence array, which stores the last 2 bytes of the first sector of the MFT record

NTFS Alternate Data Streams

- **Alternate data streams**
  - – Ways data can be appended to existing files
    - Can obscure valuable evidentiary data, intentionally or by coincidence
- In NTFS, an alternate data stream becomes an additional file attribute
  - – Allows the file to be associated with different applications
- You can only tell whether a file has a data stream attached by examining that file's MFT entry

NTFS Compressed Files

- NTFS provides compression similar to FAT DriveSpace 3 (a Windows 98 compression utility)
- Under NTFS, files, folders, or entire volumes can be compressed
- Most computer forensics tools can uncompress and analyze compressed Windows data

NTFS Encrypting File System (EFS)

- **Encrypting File System (EFS)**
  - – Introduced with Windows 2000
    - Implements a **public key** and **private key** method of encrypting files, folders, or disk volumes
- When EFS is used in Windows 2000 and later
  - – A **recovery certificate** is generated and sent to the local Windows administrator account
- Users can apply EFS to files stored on their local workstations or a remote server

EFS Recovery Key Agent

- Recovery Key Agent implements the recovery certificate
  - – Which is in the Windows administrator account

- Windows administrators can recover a key in two ways: through Windows or from an MS-DOS command prompt
- MS-DOS commands
  - cipher
  - copy
  - efsrecvr (used to decrypt EFS files) **Deleting NTFS Files**
- When a file is deleted in Windows NT and later
  - The OS renames it and moves it to the Recycle Bin
- Can use the Del (delete) MS-DOS command
  - Eliminates the file from the MFT listing in the same way FAT does **Resilient File System**
- Resilient File System (ReFS) - designed to address very large data storage needs – Such as the cloud
- Features incorporated into ReFS's design:
  - Maximized data availability
  - Improved data integrity
  - Designed for scalability
- ReFS uses disk structures similar to the MFT in NTFS

List some options for decrypting drives encrypted with whole disk encryption.

Understanding Whole Disk Encryption

- In recent years, there has been more concern about loss of

  Personal identity information (PII) and trade secrets caused by computer theft
- Of particular concern is the theft of laptop computers and other handheld devices
- To help prevent loss of information, software vendors now provide whole disk encryption
- Current whole disk encryption tools offer the following features:
  - Preboot authentication
  - Full or partial disk encryption with secure hibernation
  - Advanced encryption algorithms

- Key management function

- Whole disk encryption tools encrypt each sector of a drive separately
- Many of these tools encrypt the drive's boot sector
  – To prevent any efforts to bypass the secured drive's partition
- To examine an encrypted drive, decrypt it first
  - Run a vendor-specific program to decrypt the drive
  - Many vendors use a bootable CD or USB drive that prompts for a **onetime passphrase**

Examining Microsoft BitLocker

- Available Vista Enterprise/Ultimate, Windows 7 and 8 Professional/Enterprise, and Server 08 and 12
- Hardware and software requirements
  - A computer capable of running Windows Vista or later
  - The TPM microchip, version 1.2 or newer
  - A computer BIOS compliant with Trusted Computing Group (TCG)
  - Two NTFS partitions
  - The BIOS configured so that the hard drive boots first before checking other bootable peripherals **Examining Third-Party Disk Encryption Tools**
- Some available third-party WDE utilities:
  - **PGP Full Disk Encryption**
  - **Voltage SecureFile**
  - **Utimaco SafeGuard Easy**
  - **Jetico BestCrypt Volume Encryption**

    TrueCrypt

How the Windows Registry works

- Registry
  – A database that stores hardware and software configuration information, network connections, user preferences, and setup information

- To view the Registry, you can use:
  - Regedit (Registry Editor) program for Windows 9x systems
  - Regedt32 for Windows 2000, XP, and Vista
  - Both utilities can be used for Windows 7 and 8

Exploring the Organization of the Windows Registry

- **Registry terminology:**
  - Registry
  - Registry Editor
  - HKEY
  - Key
  - Subkey
  - Branch
  - Value
  - Default value
  - Hives

Understanding Microsoft Startup Tasks

- Learn what files are accessed when Windows starts
- This information helps you determine when a suspect's computer was last accessed
  - Important with computers that might have been used after an incident was reported

Startup in Windows 7 and Windows 8

- Windows 8 is a multiplatform OS
  - Can run on desktops, laptops, tablets, and smartphones
- The boot process uses a boot configuration data (BCD) store
- The BCD contains the boot loader that initiates the system's bootstrap process
  - Press F8 or F12 when the system starts to access the Advanced Boot Options

Startup in Windows NT and Later

- All NTFS computers perform the following steps when the computer is turned on:
  - Power-on self test (POST)

  - Initial startup
  - Boot loader
  - Hardware detection and configuration
  - Kernel loading
  - User logon

- Startup Files for Windows Vista:
  - The Ntldr program in Windows XP used to load the OS has been replaced with these three boot utilities:
- Bootmgr.exe
- Winload.exe
- Winresume.exe
  - Windows Vista includes the BCD editor for modifying boot options and updating the BCD registry file
  - The BCD store replaces the Windows XP boot.ini file
- Startup Files for Windows XP:
  - NT Loader (NTLDR)
  - Boot.ini
  - Ntoskrnl.exe
  - Bootvid.dll
  - Hal.dll
  - BootSect.dos
  - NTDetect.com
  - NTBootdd.sys
  - Pagefile.sys

Windows XP System Files

Filename	Description
Ntoskrnl.exe	The XP executable and kernel
Ntkrnlpa.exe	The physical address support program for accessing more than 4 GB of physical RAM
Hal.dll	The Hardware Abstraction Layer (described earlier)
Win32k.sys	The kernel-mode portion of the Win32 subsystem
Ntdll.dll	System service dispatch stubs to executable functions and internal support functions
Kernel32.dll	Core Win32 subsystem DLL file
Advapi32.dll	Core Win32 subsystem DLL file
User32.dll	Core Win32 subsystem DLL file
Gdi32.dll	Core Win32 subsystem DLL file

Fig: Windows XP System Files

- Contamination Concerns with Windows XP
  - When you start a Windows XP NTFS workstation, several files are accessed immediately
- The last access date and time stamp for the files change to the current date and time
  - Destroys any potential evidence
- That shows when a Windows XP workstation was last used **2.7 Describe MS-DOS startup tasks**

✓ MS-DOS uses three files when starting, with the same names as in Windows 9x/Me: Io.sys, Msdos.sys, and Command.com.

✓ Two other files are then used to configure MS-DOS at startup: Config.sys and Autoexec.bat.

✓ Although MS-DOS and Windows 9x use some of the same startup filenames, there are some important differences between the files in these OSs.

➢ Io.sys is the first file loaded after the ROM bootstrap loader finds the disk drive. Io.sys then resides in RAM and provides the basic input and output service for all MS-DOS functions.

➢ Msdos.sys is the second program to load into RAM immediately after Io.sys.

➢ As mentioned, this file is the actual OS kernel, not a text file as in Windows 9x

and Me.

- After Msdos.sys finishes setting up DOS services, it looks for the Config.sys file to configure device drivers and other settings.
- Config.sys is a text file containing commands that typically run only at system startup to enhance the computer's DOS configuration.
- Msdos.sys then loads Command.com, which contains the same internal DOS commands in MS-DOS 6.22 as in Windows 9x. As the loading of Command.com nears completion, Msdos.sys looks for and loads Autoexec.bat, a batch file containing customized settings for MS-DOS that runs automatically.
- In this batch file, you can define the default path and set environmental variables, such as temporary directories. MS-DOS then accesses and resets the last access dates and times on files when powered up.

- **Virtual machine**
  - – Allows you to create a representation of another computer on an existing physical computer .A virtual machine is just a few files on your hard drive
  - – Must allocate space to it
- A virtual machine recognizes components of the physical machine it's loaded on
  - – Virtual OS is limited by the physical machine's OS .