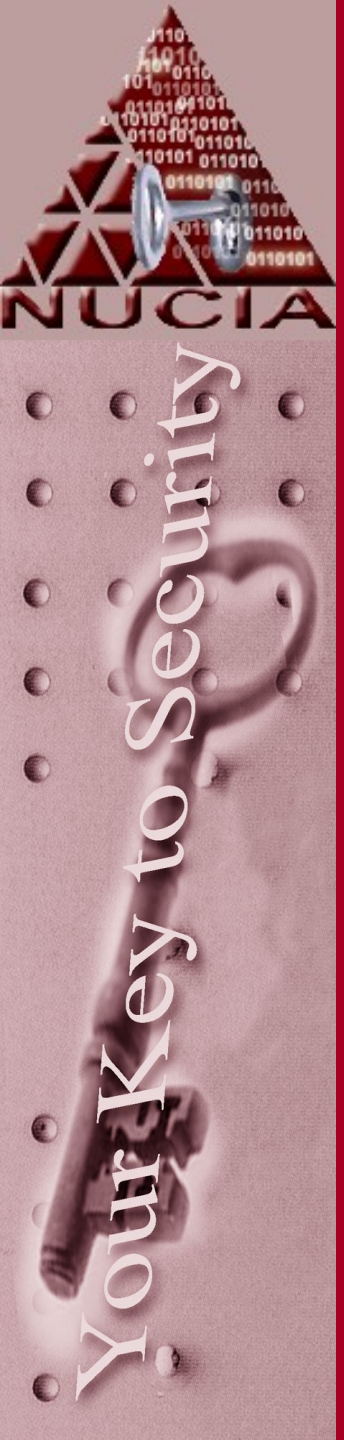



# Cyber-Forensics Intermediate Topics

CERTConf2006

Tim Vidas

Duplication is a Science



# Who am I?

- Tim Vidas
  - Sr. Tech. Research Fellow
  - UNO/PKI/NUCIA
  - Certs: CISSP, 40xx, Guidance, AccessData etc.
  - Instructor: UNO, Guidance, LM RRCF



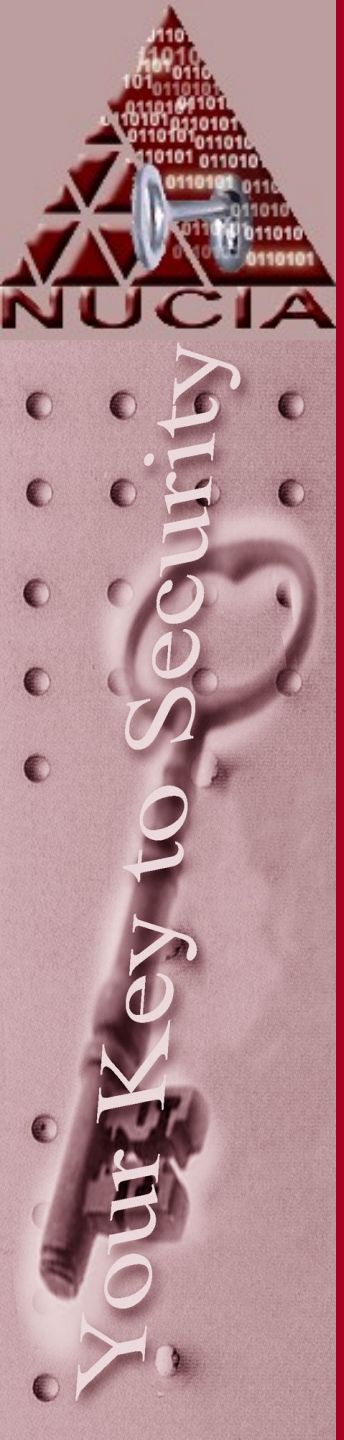
# NUCIA

- Nebraska University Consortium on Information Assurance
- IA full time
- Traditional university coursework in IA, Crypto, Forensics, Secure Administration, Certification and Accreditation, etc
- STEAL Labs
- “Other work”
- Most of us are ‘around’ CERTconf.



# Who are you?

- Who are you?
- Where do you work?
- What do you do?
- How many of you are planning on attending all “Forensics” sessions?
- What are you expecting to get out of them? (I’ll try to be accommodating)




# Disclaimer

- Even though this class touches on quite a few legal topics – nothing should be construed as advice or legal instruction
- Before performing many of the skills learned this week on a computer other than your own, you may need to seek permission (possibly written) and or seek advice from your own legal counsel.



# Why?

- Why even create a copy? Why not perform analysis on the actual hard disk?
  - Evidence
    - Bagged n sealed
    - Chain of custody
    - Basically
      - what if you accidentally made a mistake?
      - To show that no evidence was “planted”
    - Preserving the Integrity of the Evidence



# Interface

- IDE / ATA
- SCSI  (scuzzy)
- Serial ATA

Various limitations (usually addressing) create limits on hard drive sizes. Commonly 2.1, 8.4, 32 and 137 GB.



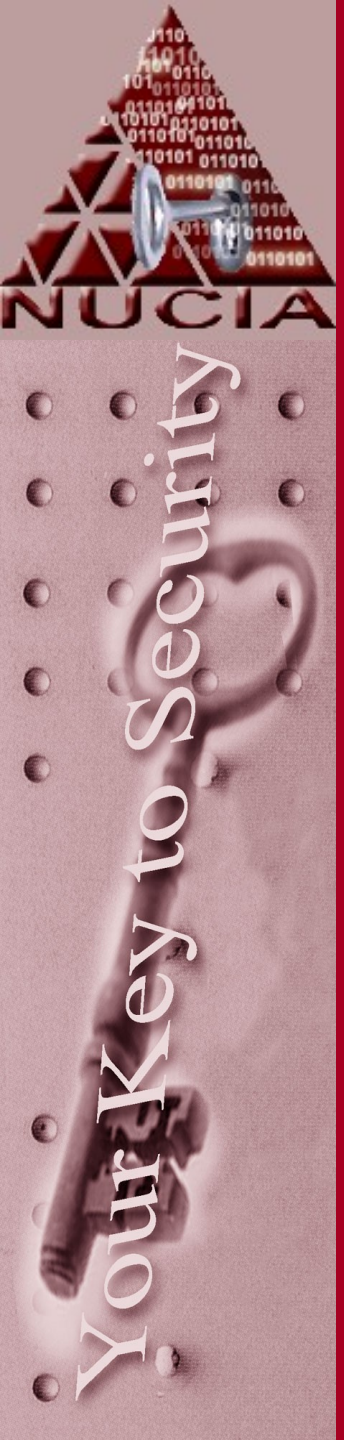
# The $#%* cable

- ATA 33 and lower can use a 40 pin ribbon cable
- _Anything_ higher requires an 80 pin to reduce crosstalk between the wires
- The ribbon cables are sometimes color coded.
- Typically the “Master Drive” is on one end, and the “Slave Drive” is in the middle.



# Bridging

- Similar to a networking bridge, you can circumvent interface standards using ATA bridges. Something like a Firewire – ATA bridge.
- This has a couple benefits from a forensic point of view:
  - External, swappable
  - Write blocking in hardware
  - Ability to use the same interface every time



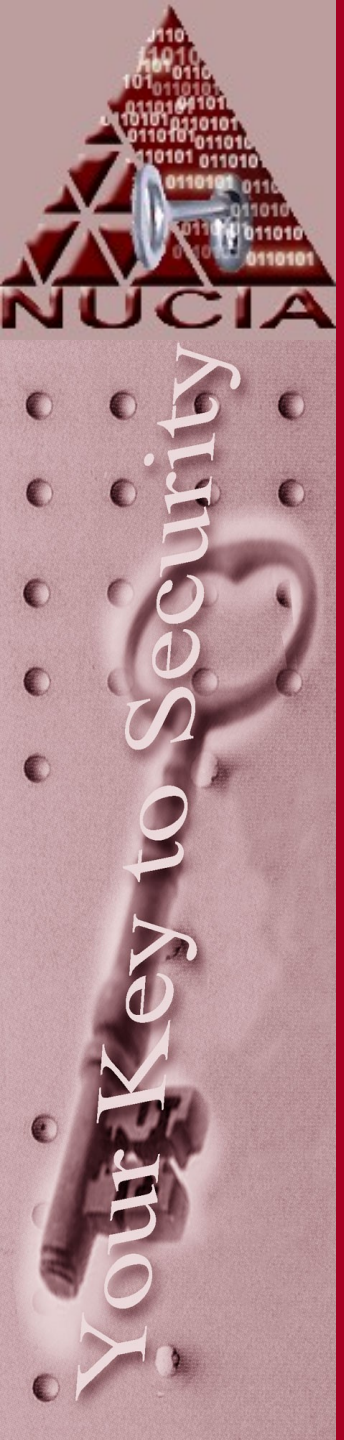

# SCSI

- Still mainly found in servers as opposed to desktops
- Physically the hardware is very similar – usually the controller is the difference
- Raid is almost always used
  - What does this mean to us as examiners?



# SCSI

- SCSI's main advantage is on the bus:
  - Each ATA device controls the entire bus for actions (write / read / etc)
  - SCSI devices can share, queue, etc
- SCSI devices are IDed 0-7 or 0-14 and can all be chained together on a single controller (but must be terminated on the ends)




# SCSI

- While ATA took a sequential approach to versions (1,2,3,4,5) scsi created many variations: LVD, LVD/SE, DIFF, Ultra, Ultrawide, Ultra4 (also Ultra320), etc.
- You must have matching controllers and devices as well as the correct cable for each.

# Why?

- No matter what tool you are using, a “write blocking” adapter will prevent costly mistakes

- This allows writes to succeed according to the system, but actually blocks the interrupts from reaching the HDD



# In fact…

## LAPD CCU Field Response Checklist:

- Field Computer
- FireWire Write Blocker
- FastBloc Write Blocker
- Crossover cable
- Laplink cable
- Tool kit
- Encase Boot Disks – Network/CD/Floppy
- Gdisk
- Encase Install Disks
- Keyboard
- Mouse
- Power Cables
- Drive Bay Keys
- Exam Logs
- Blank Floppies
- Blank CD's
- Blank DVD's
- Clip Board
- Forensic Computer Boot Drive(s)
- Drivers for – USB/Parallel Dongles, SCSI cards, USB Zip drive
- Flat Monitor
- SCSI cables
- Camera and Charger
- 2.5 – 3.5 adapter
- Laptop
- IDE Cables
- Power Splitter – Molex connectors
- Flashlight
- ESD Equipment – Mat, Wristband
- Encase/FTK Dongles
- Smart Media Reader
- Serial ATA cables
- SCSI Write block
- Software
- Ghost Images of Forensic Boot disk



# Evidence

- Federal Rules of Evidence (FRE)
- To prove the content of a writing, recording, or photograph, the original writing, recording or photograph is required, except as otherwise provided in these rules or by Act of Congress
- FRE #1002—item or information presented in court must be original
- FRE 1001(3) outlines one of these exceptions:
- Definitions and Duplicates: If data are stored in a computer or similar device, any printout or other output readable by sight, shown to reflect accurately, is an original



# Evidence

- Admissibility of Duplicates FRE #1003, A duplicate is admissible to the same extent as an original unless 1) a genuine question is raised as to the authenticity of the original ,or 2) in the circumstances it would be unfair to admit the duplicate in lieu of the original



# Chain o Custody

- Evidence tag for each hard drive or media
  - Time and date of action
  - Number we assigned to that case
  - Number of this particular tag
  - Consent required? Signature of person owning information
  - Whom evidence belonged? Who provided information
  - Complete description of evidence including quantity
  - Who rec'd evidence and signature of recipient



# Chain o Custody

- Back of evidence tag
  - Who the evidence was rec'd from and location it was in
  - Date of receipt
  - Reason the evidence was given to another person
  - Who rec'd evidence and where was evidence was rec'd or located



# Initial Response: Live Sys

- Volatile data before forensic image
  - Volatile data
  - Registers, cache contents
  - Memory contents
  - State of networks
  - State of running processes
  - Contents of storage media
  - Contents of removable and backup media



# Initial Response: Live Sys

- Create a step-by-step plan, document it:
  - Establish a new shell: cmd.exe (W), bash (U)
  - Record the system date and time: date, time (W), w (U)
  - Who is logged on: loggedon (W), w (U)
  - Record open sockets: netstat (W), netstat –anp (U)
  - Processes that open sockets: fport (W), lsof (U)
  - Currently running processes: pslist (W), ps (U)
  - System that recently connected: nbtstat (W), netstat (U)
  - Record system time: date, time (W), w (U)
  - Record step taken: doskey (W), script, vi, history (U)
- This stuff can be scripted!
  - More on this later..



# Terminology

- *Forensic Duplication:* bit for bit copy (dd, dfcldd, odd)
- *Qualified Forensic Duplicate:* file that contains every bit, but is stored in an altered form (encase)
- *Restored Image:* a Forensic Dup or Qualified Forensic Dup that has been restored to a drive (dd, encase, etc)
- *Mirror Image:* a duplicate created with hardware (SF-5000, Solo-2)



# Offline duplication

- Assuming the computer is ‘offline’
  - There are a couple avenues to take
    - Logical copy
      - Windows drag n drop / cut n paste
      - Here we are talking about things like files and directories
    - Physical copy
      - Bit for bit copy
      - Here we are obviously talking about bits
    - Advantages / Disadvantages to the two methods?



# Offline Duplication

- Physical copies of the hard drive contain more information than a logical copy. Things like:
  - Information left in virtual memory / page files, swap space, etc
  - Files and directories marked as deleted then partially written over, slack space, etc
  - “unallocated” space that is actually in use



# Integrity

- When writing an image to a new hard disk for analysis it's a good idea to 'clean' the disk first
- This is where the 'writing zeros' or 'zero-filling' comes into play

- Let's talk about this for a bit…



# Cleaning up

- Using dd…
- dd if=/dev/zero of=/dev/hda
- dd if=/dev/random of=/dev/hda1

- Darik's Boot-n-nuke
- Ritedisk?
- wipe



# Cleaning up

- Sometimes we can clean up too much..
- dd if=/dev/zero of=/dev/fd0
- All the ‘meta information’ is gone, we also wrote 0’s to the traditionally non addressable portions of the disk
- mkfs.dos /dev/fd0



# Proving Integrity

- As a professional, your word might not be enough to establish that the copy is an exact copy of evidence obtained earlier.
- This can be mitigated using a HASH like MD5 or SHA-1 – some forensic packages may even use things like CRC.

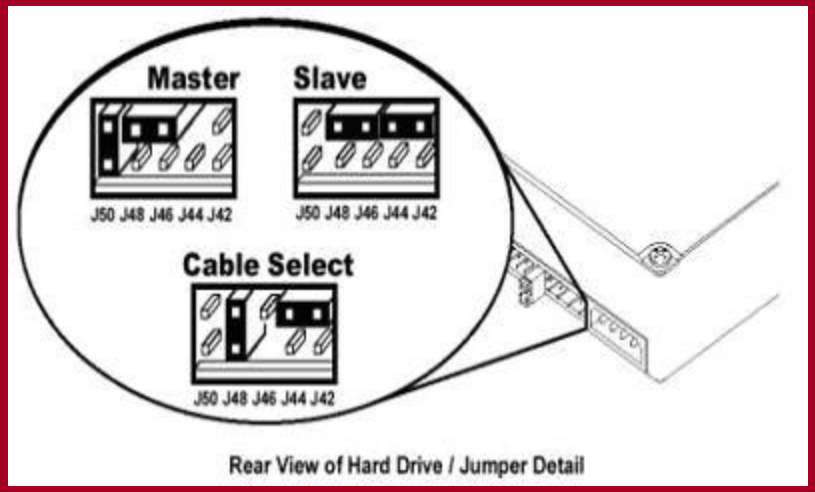


# Duplication gotchas

- Some file systems have limits to maximum file size…like the 2.1 GB barrier Hard/Soft or the 8.4 Hardware barriers
  - In such cases, the image would have to segmented into multiple images that can later be restored into one
  - Or you could use a filesystem that supports larger files :-)

# Duplication gotchas

- Be aware of jumper settings
- Master / Slave / CS etc
- This will vary based on Drive and should be noted on the physical drive somewhere
- Some hardware will require certain settings to attach correctly.



Rear View of Hard Drive / Jumper Detail



# Duplication gotchas

- dd (and it's variants) is VERY powerful.
  - This can do physical duplication of a device, or a particular partition or a _____
  - What does this mean to you?

(this will be important for you in just a little bit ;-)


Your Key to Security

# dd

- dd has *many* flags (options)
- You must first understand:

dd if=/*source* of=/*destination*

if = infile, or evidence you are copying (a hard disk, tape, etc.)
source = source of evidence
of = outfile, or copy of evidence
destination = where you want to put the copy



# dd

- dd if=/dev/hda of=/dev/ImageCopy1
- In addition to hard drives, dd works well restoring block-oriented devices, such as tapes.
- Some useful optoins are:

  ibs = input block size
  obs = output block size
  bs = block size

  count = number of blocks to copy
  skip = # of blocks to skip at start of input
  seek = # of blocks to skip at start of output



# dcfldd

- An enhanced version of dd – DOD Computer Forensics Lab dd
- Able to generate hashes as the image is created – otherwise works just like dd
- Readily available to the public

```
dcfldd if=/dev/hdd of=/mnt/disk.dd bs=2k
    hashwindow=2M hashlog=/mnt/disk.md5
```



# DCCIdd

- Updated version of dcfldd
  - Restricted distribution – somewhat
- Still DCFL maintained
- Works the same as dcfldd

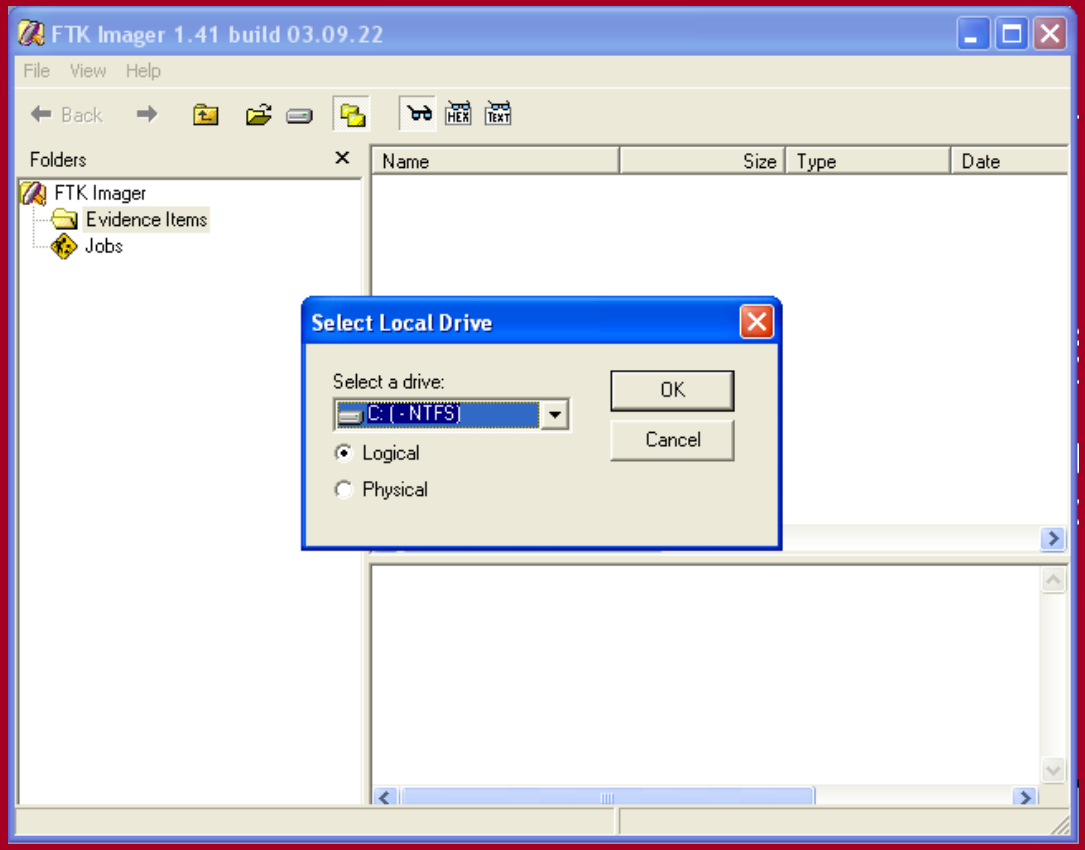


# odd

- Odessa – Open Digital Evidence Search and Seizure Architecture
- ODD – open data duplicator
  - Client / server
    - Both can be on one machine
  - Plugin based
    - Auto extract images
    - Auto hashing
    - Auto string search

# Others…

- Safeback
- Forensic ToolKit
- Encase




Your Key to Security



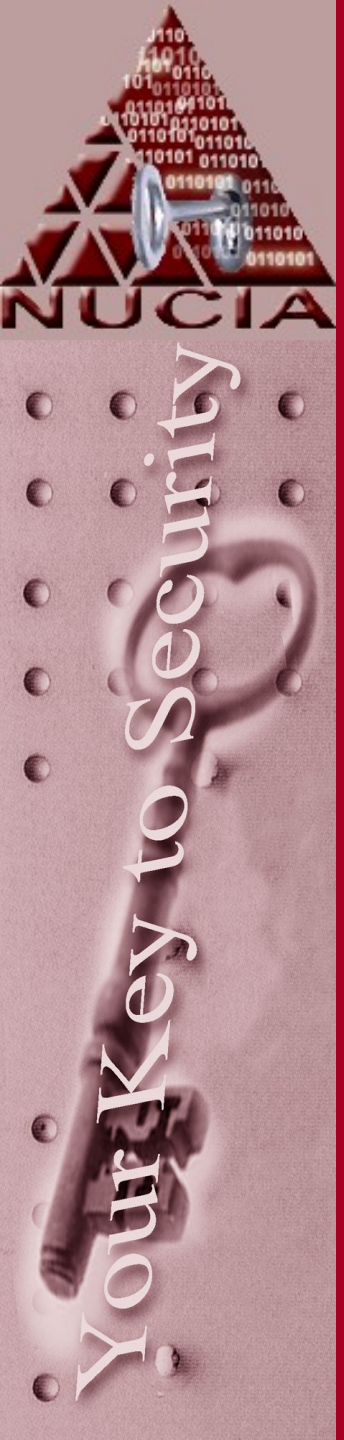
# ‘Hard’ to recover

- *Deletion* and *replacement*
  Deleting the file and replacing it immediately with another file of the same name and exactly the same size typically completely overwrites the original file.
- *Low level formatting*
  LLFing of the computer hard disk will destroy all data. LLF is usually only carried out once by the manufacturer - the Format command in DOS/Windows does NOT perform a low level format – in fact an ‘actual’ LLF can not be performed by a PC on a ‘newer’ hard disk.



# LLF – ask and ye shall receive

- Ok everyone always asks “why can the LLF only be done at the factory?”
- IDE drives have control information on track 0 or -1 that only the controller can read. This information includes bad track information, head skew factors and zone sector information. Fortunately, all newer hard drives only operate in a *translation mode*, so we can successfully do a format that could be more properly called a mid-level format.
- All hard drives are initially formatted in *native mode*, and all, except the oldest IDE drives, operate in *translation mode*. The special tracks containing information specifically for the controller cannot be accessed in translation mode. Even if you could access this information, the BIOS and operating system would not be able to read the hard drive, as they cannot handle Zoned Bit Recording.




# ‘Hard’ to recover

- *"Anti-Forensic Software“*
  Specialist software is available which claims to completely remove all trace of deleted files from a hard disk, including residual traces of old deleted files in slack space (unused space left over at the end of a cluster), by overwriting those areas multiple times with random data.
- *Encryption:*
  Encryption is an effective way to conceal incriminating evidence. An encrypted file can usually only be opened if the decryption key is obtained.



# “Easy” to Recover

- *Deletion*
  One of the easiest situations for an investigator is when a suspect has simply deleted all incriminating files just before the PC is obtained.
- When a file is deleted, the operating system simply marks the cluster(s) the file is occupying as now being available for use again in the File Allocation Table. It does not in any way destroy or damage the data in the cluster(s) itself, apart from (typically) replacing the first letter of the filename with the greek letter sigma. The file has effectively been removed from the index. The forensic investigator is easily able to recover the file by simply extracting it straight from the cluster.



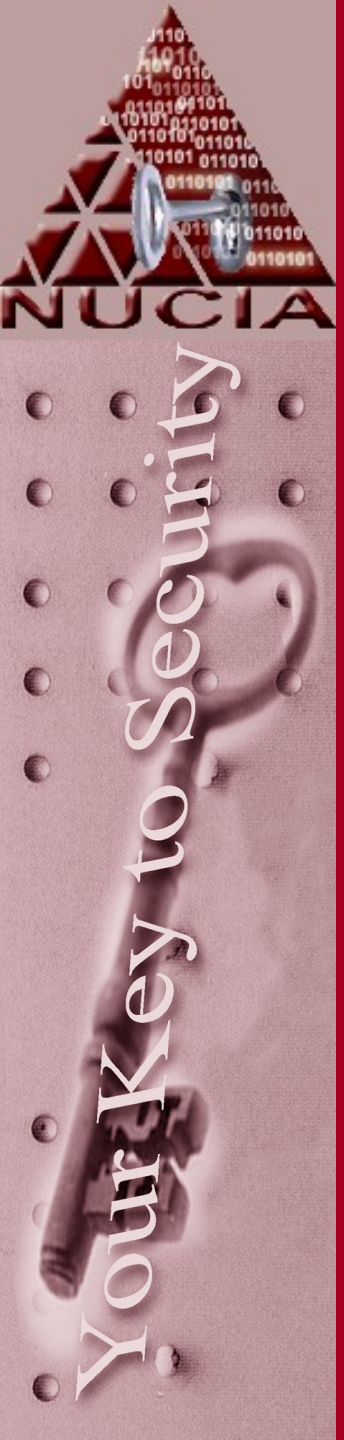
# ‘Easy’ to recover

- The situation becomes more difficult as time passes. Since deletion the operating system now sees the cluster(s) as being available for use (un-re-allocated). The next time a new file is saved onto the disk there is a danger that the file, or part of it, will be stored in the cluster containing the old deleted file.
- However, under certain circumstances it is still possible to recover some of the old file, even if a new file has been saved to the same cluster, because of the slack space.
- Consider the situation where a cluster contained an important document, 30k in length. The file is removed from the index in the FAT but the document remains in the cluster. A new document is then saved to the same cluster, however the new document is only 20k in length. The last 10k of the original document will still be present in the slack space at the end of the cluster and can be retrieved.



# “Easy” to Recover

- *Formatting*
  The process of formatting using the Format command in Windows or DOS performs a **high level format.** This is non-destructive to data on the disk. The process simply resets the index in the File Allocation Table so that operating system sees the disk as empty. The information is still there, only the operating system does not know how to get to it. Data on a disk which has been high level formatted can usually be recovered.
- *Defragmentation*
  When the operating system stores files on the hard disk, it splits them up into clusters. If the file is larger than the cluster size, several clusters will be used. These clusters are not necessarily adjacent to each other, but may be spread across the surface of the hard disk, depending on space available.
  - The process of defragging simply identifies clusters that contain parts of the same file, and moves them together so that they make, as far as possible, a contiguous block. In this process the system will use space allocated as free in the File Allocation Table, it is therefore possible that data being moved will overwrite space occupied by a deleted file, however this is by no means guaranteed.




# Slack Space

- Looking back to the email from the hex section:
- Assuming a file was stored contiguously (not fragmented)…

Habib,

Our plans are in motion and all is well. Homeland Security suspects nothing, the explosion will be grand. Attached are the coordinates of the attack saved in the usual way.

Loyally,

Samir

????????

# Slack Space

- Using tools like xxd or winhex we can see how the file is stored in hex:

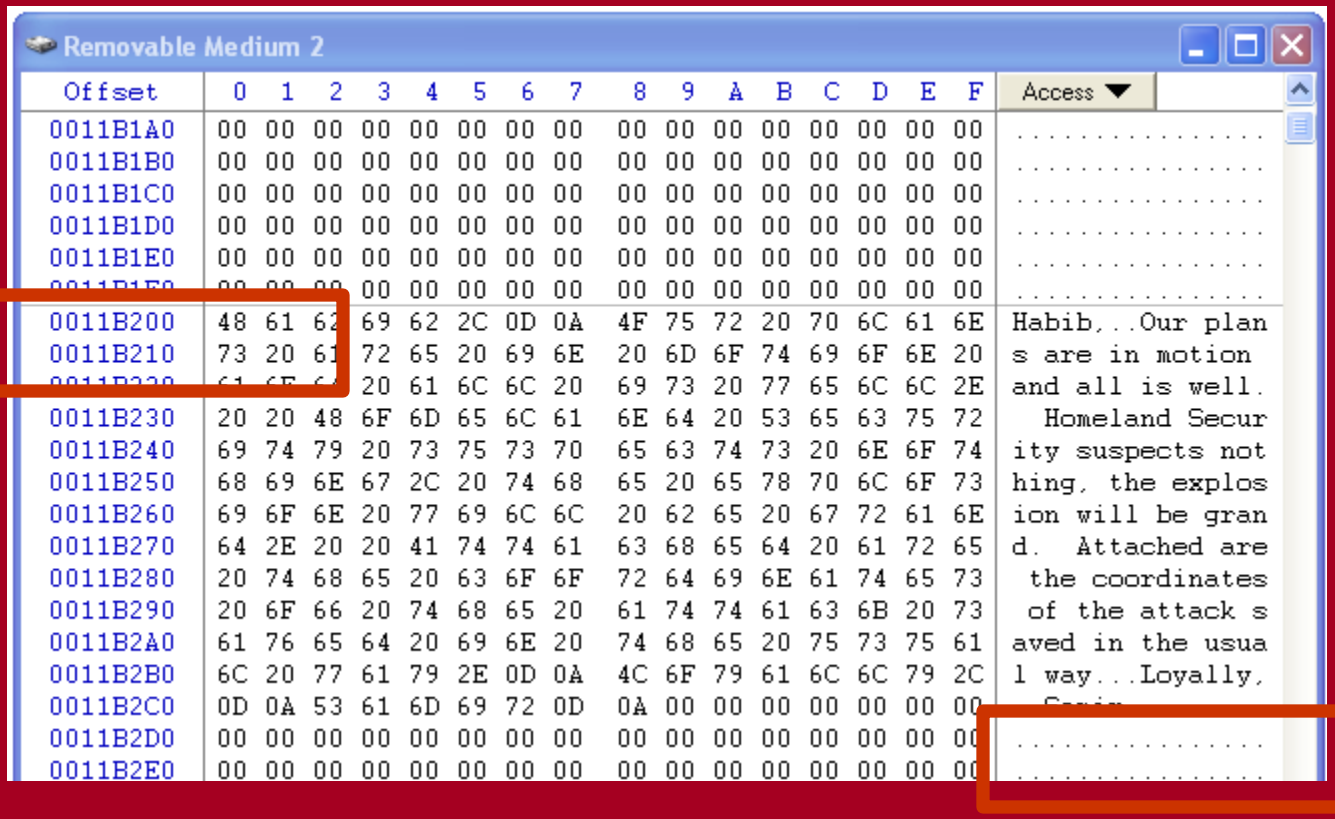
```
$ xxd SavedEmail.txt
0000000: 4861 6269 622c 0d0a 4f75 7220 706c 616e  Habib,..Our plan
0000010: 7320 6172 6520 696e 206d 6f74 696f 6e20  s are in motion
0000020: 616e 6420 616c 6c20 6973 2077 656c 6c2e  and all is well.
0000030: 2020 486f 6d65 6c61 6e64 2053 6563 7572    Homeland Secur
0000040: 6974 7920 7375 7370 6563 7473 206e 6f74  ity suspects not
0000050: 6869 6e67 2c20 7468 6520 6578 706c 6f73  hing, the explos
0000060: 696f 6e20 7769 6c6c 2062 6520 6772 616e  ion will be gran
0000070: 642e 2020 4174 7461 6368 6564 2061 7265  d.  Attached are
0000080: 2074 6865 2063 6f6f 7264 696e 6174 6573   the coordinates
0000090: 206f 6620 7468 6520 6174 7461 636b 2073   of the attack s
00000a0: 6176 6564 2069 6e20 7468 6520 7573 7561  aved in the usua
```

SavedEmail.txt

| Offset | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00000000 | 48 | 61 | 62 | 69 | 62 | 2C | 0D | 0A | 4F | 75 | 72 | 20 | 70 | 6C | 61 | 6E | Habib,..Our plan |
| 00000010 | 73 | 20 | 61 | 72 | 65 | 20 | 69 | 6E | 20 | 6D | 6F | 74 | 69 | 6F | 6E | 20 | s are in motion |
| 00000020 | 61 | 6E | 64 | 20 | 61 | 6C | 6C | 20 | 69 | 73 | 20 | 77 | 65 | 6C | 6C | 2E | and all is well. |
| 00000030 | 20 | 20 | 48 | 6F | 6D | 65 | 6C | 61 | 6E | 64 | 20 | 53 | 65 | 63 | 75 | 72 | Homeland Secur |
| 00000040 | 69 | 74 | 79 | 20 | 73 | 75 | 73 | 70 | 65 | 63 | 74 | 73 | 20 | 6E | 6F | 74 | ity suspects not |
| 00000050 | 68 | 69 | 6E | 67 | 2C | 20 | 74 | 68 | 65 | 20 | 65 | 78 | 70 | 6C | 6F | 73 | hing, the explos |
| 00000060 | 69 | 6F | 6E | 20 | 77 | 69 | 6C | 6C | 20 | 62 | 65 | 20 | 67 | 72 | 61 | 6E | ion will be gran |
| 00000070 | 64 | 2E | 20 | 20 | 41 | 74 | 74 | 61 | 63 | 68 | 65 | 64 | 20 | 61 | 72 | 65 | d.  Attached are |
| 00000080 | 20 | 74 | 68 | 65 | 20 | 63 | 6F | 6F | 72 | 64 | 69 | 6E | 61 | 74 | 65 | 73 | the coordinates |
| 00000090 | 20 | 6F | 66 | 20 | 74 | 68 | 65 | 20 | 61 | 74 | 74 | 61 | 63 | 6B | 20 | 73 | of the attack s |
| 000000A0 | 61 | 76 | 65 | 64 | 20 | 69 | 6E | 20 | 74 | 68 | 65 | 20 | 75 | 73 | 75 | 61 | aved in the usua |
| 000000B0 | 6C | 20 | 77 | 61 | 79 | 2E | 0D | 0A | 4C | 6F | 79 | 61 | 6C | 6C | 79 | 2C | l way...Loyally, |
| 000000C0 | 0D | 0A | 53 | 61 | 6D | 69 | 72 | 0D | 0A | | | | | | | | ..Samir.. |

# Slack Space

- See the difference?




# Files and File Systems



# The OSI of File Systems

| Application Storage |
| --- |
| Classification |
| Space Management |
| Allocation Units |
| Data Classification |
| Physical |


# The OSI of File Systems

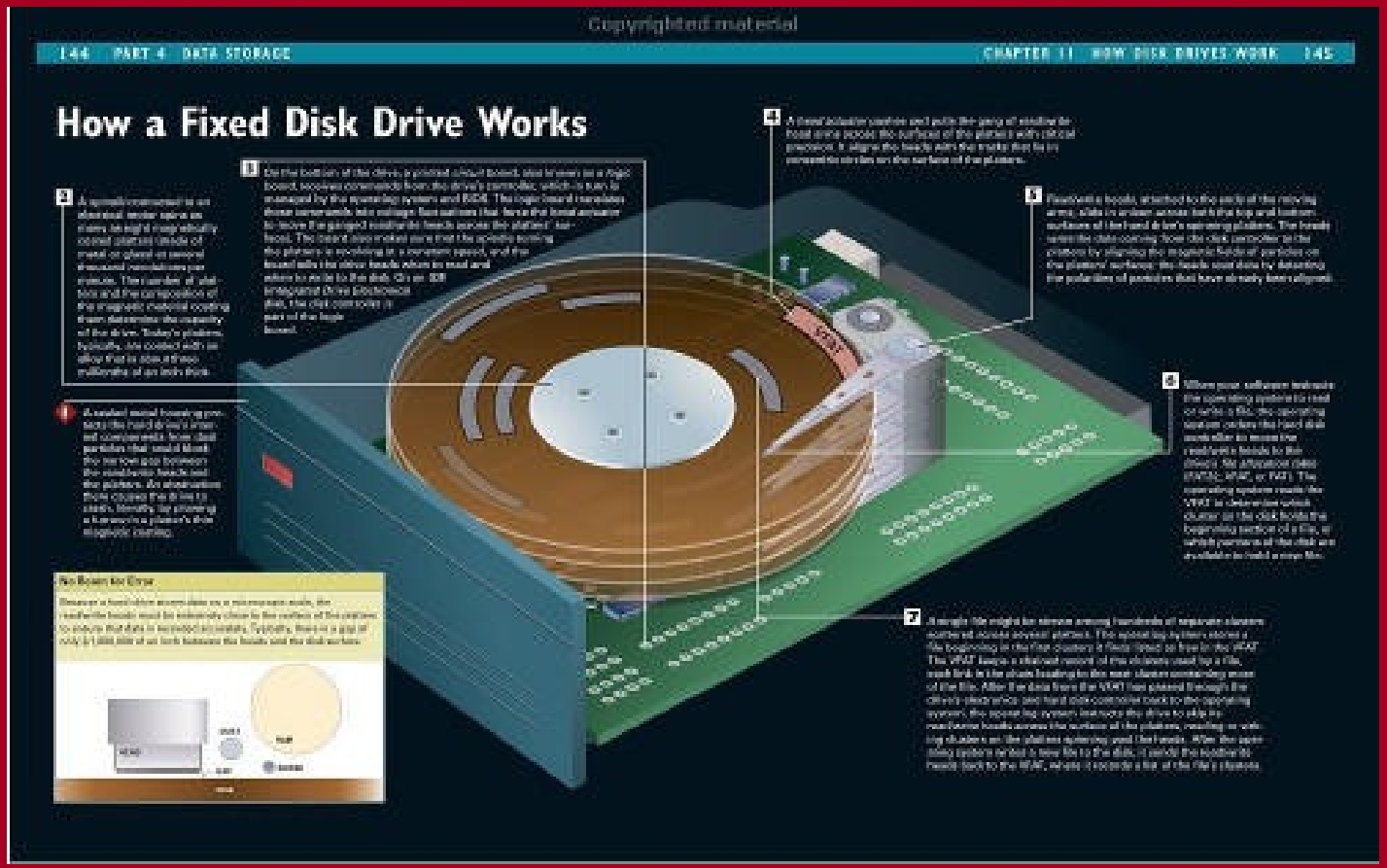
| | FAT / NTFS | EXT2/3 |
|---|---|---|
| Application Storage | Files | Files |
| Classification | Folders | Directories |
| Space Management | FAT (MFT) | Inodes |
| Allocation Units | Clusters | Blocks |
| Data Classification | Partitions | Partitions |
| Physical | Sectors | Sectors |



# FS Layers : Physical

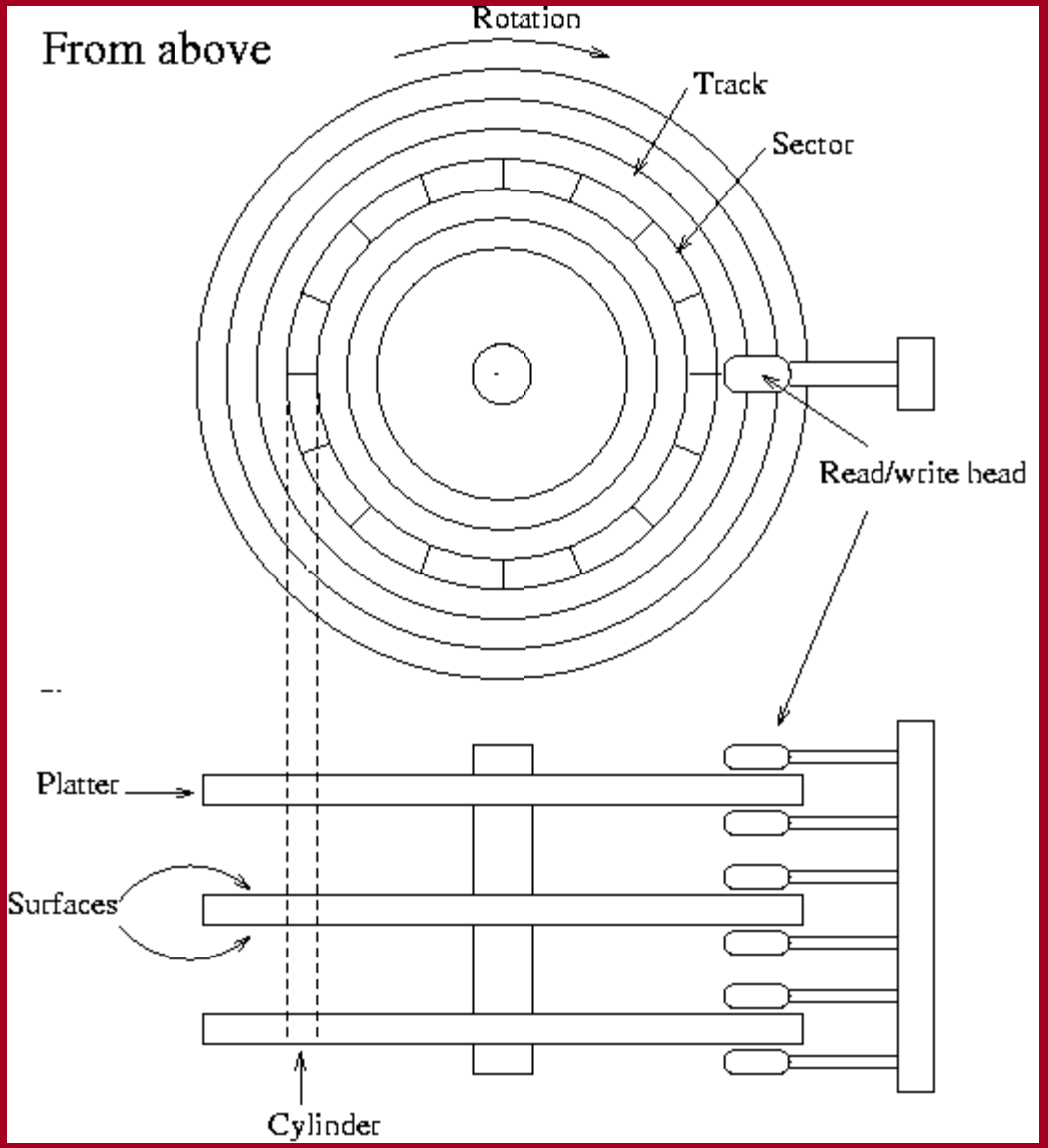
- No matter what, this layer is always present. The bits have to actually be located somewhere.
- Absolute sectors are numbered 0 and up.
- Most OS's read and write in chunks of 512 bytes.
- Some hardware actually allow access via Cylinder, head and sector values

# FS Layers: Physical






# FS Layers: Physical





# FS Layers: Data Classification

- Just above the physical layer
- Partitioning scheme set up by the OS
- Basically allows for segmentation of data
  - Security
  - Logical organization
  - Speed
  - Means to an end



# FS Layers: Data Classification

- Host Protected Access
- Usually used by vendors as part of a restoration process
- Not accessible by the OS
  - Or by earlier versions of encase/safeback/etc
  - Current versions support this
  - Great example of why you need to be an expert media analyst – not an expert encase user



# FS Layers: Data Classification

- One byte FS identifier code – used for mounting. Some OSes allow this to be specified.

00 Empty
01 DOS 12-bit FAT
02 XENIX root
03 XENIX /usr .
04 DOS 3.0+ 16-bit FAT (up to 32M)
05 DOS 3.3+ Extended Partition
06 DOS 3.31+ 16-bit FAT (over 32M)
07 OS/2 IFS (e.g., HPFS)
07 Windows NT NTFS
07 Advanced Unix
07 QNX2.x pre-1988
08 OS/2 (v1.0-1.3 only)
08 AIX boot partition
08 SplitDrive
08 Commodore DOS
08 DELL partition spanning multiple drives
08 QNX 1.x and 2.x ("qny")
09 AIX data partition
09 Coherent filesystem
09 QNX 1.x and 2.x ("qnz")
0a OS/2 Boot Manager
0a Coherent swap partition
0a OPUS
0b WIN95 OSR2 FAT32
0c WIN95 OSR2 FAT32, LBA-mapped
0e WIN95: DOS 16-bit FAT, LBA-mapped
0f WIN95: Extended partition, LBA-mapped
10 OPUS (?)
11 Hidden DOS 12-bit FAT
11 Leading Edge DOS 3.x
12 Configuration/diagnostics partition
14 Hidden DOS 16-bit FAT <32M
14 AST DOS with
16 Hidden DOS 16-bit FAT >=32M
17 Hidden IFS (e.g., HPFS)
18 AST SmartSleep Partition
19 Unused
1b Hidden WIN95 OSR2 FAT32
1c Hidden WIN95 OSR2 FAT32, LBA-mapped
1e Hidden WIN95 16-bit FAT, LBA-mapped
20 Unused
21 Reserved
21 Unused
22 Unused
23 Reserved
24 NEC DOS 3.x
26 Reserved
31 Reserved
32 NOS
33 Reserved
34 Reserved
35 JFS on OS/2 or eCS
36 Reserved
38 THEOS ver 3.2 2gb partition
39 Plan 9 partition
39 THEOS ver 4 spanned partition
3a THEOS ver 4 4gb partition
3b THEOS ver 4 extended partition
3c PartitionMagic recovery partition
3d Hidden NetWare
40 Venix 80286
41 Linux/MINIX (sharing disk with DRDOS)
41 Personal RISC Boot
41 PPC PReP (Power PC Reference Platform) Boot
42 Linux swap (sharing disk with DRDOS)
42 SFS (Secure Filesystem)
42 Windows 2000 dynamic extended partition marker
43 Linux native (sharing disk with DRDOS)
44 GoBack partition
45 Boot-US boot manager
45 Priam
5c Priam EDisk
61 SpeedStor
63 Unix System V (SCO, ISC Unix, UnixWare, ...), Mach, GNU Hu
64 PC-ARMOUR protected partition
64 Novell Netware 286, 2.xx
65 Novell Netware 386, 3.xx or 4.xx
66 Novell Netware SMS Partition
82 Solaris x86
82 Linux swap
83 Linux native partition
84 OS/2 hidden C: drive
84 Hibernation partition
85 Linux extended partition
86 Old Linux RAID partition superblock
86 NTFS volume set
87 NTFS volume set
8a Linux Kernel Partition (used by AiR-BOOT)
8b Legacy Fault Tolerant FAT32 volume
8c Legacy Fault Tolerant FAT32 volume using BIOS extd INT 13h
8d Free FDISK hidden Primary DOS FAT12 partitition
8e Linux Logical Volume Manager partition
90 Free FDISK hidden Primary DOS FAT16 partitition
91 Free FDISK hidden DOS extended partitition
92 Free FDISK hidden Primary DOS large FAT16 partitition
93 Hidden Linux native partition
a0 Laptop hibernation partition
a1 Laptop hibernation partition
a1 HP Volume Expansion (SpeedStor variant)
a5 BSD/386, 386BSD, NetBSD, FreeBSD
a6 OpenBSD
a9 NetBSD
ab Mac OS-X Boot partition
c2 Hidden Linux
c3 Hidden Linux swap

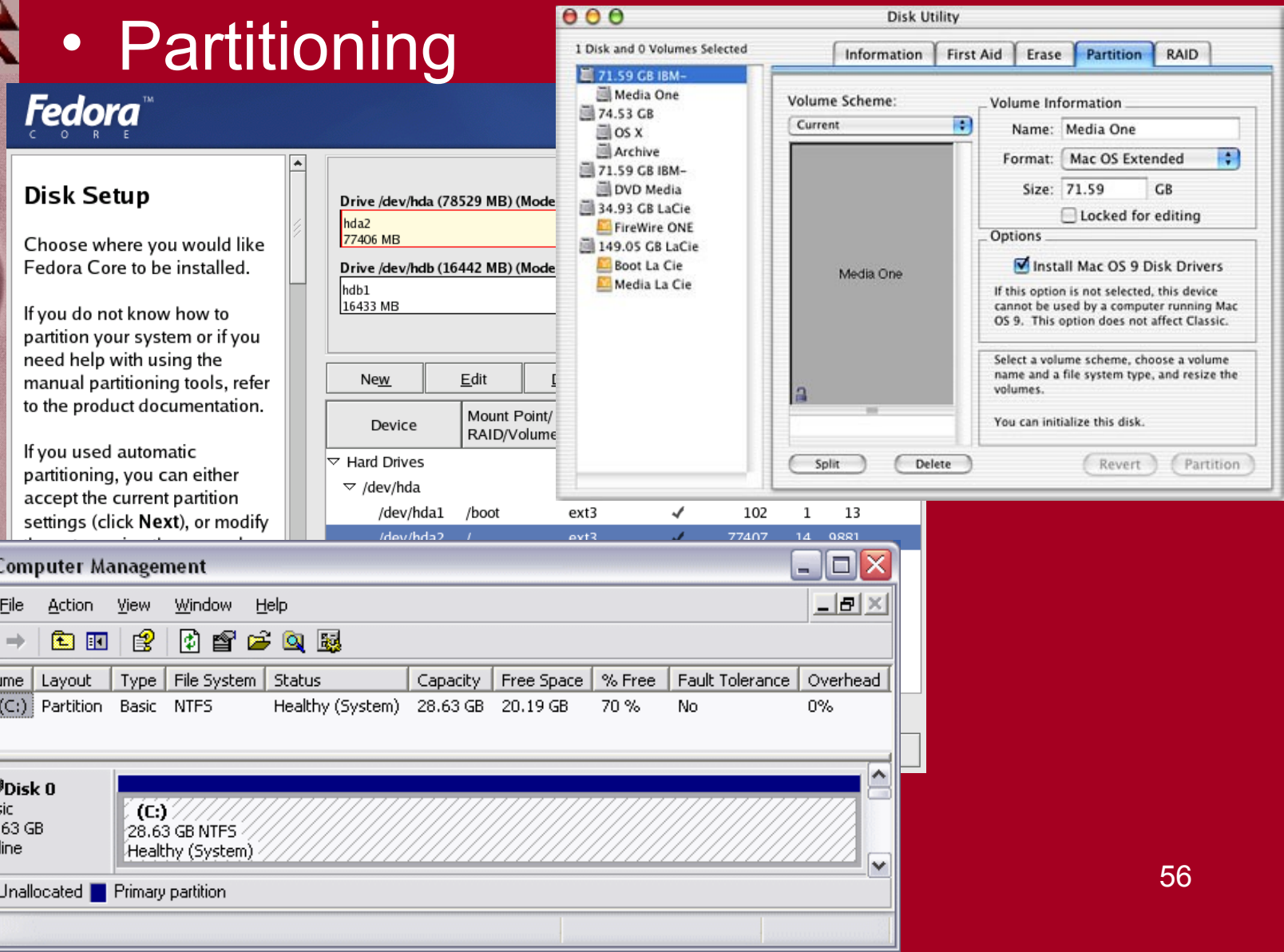


# FS Layers: Data Classification

- Most mainstream OS's have partitioning software built in (most even graphical)
- 3rd party vendors sell things like partition magic

# FS Layers: Data Classification

- Partitioning





# FS Layers: Allocation

- Allocations units (blocks) depend on:
  - FS Type
  - Partition Size
  - System Admin
- Particular applications can perform better or worse depending on the size of an allocation unit
- Things like databases, and video have known performance relations with block size.



# FS Layers: Allocation

| Hard Disk Size | FAT12 | FAT16 | FAT32 | NTFS | Ext2 |
|---|---|---|---|---|---|
| 0 to 16MB | 4,096 bytes | 2,048 bytes | 512 bytes | 512 bytes | 4,096 bytes |
| 16 to 128MB | n/a | 2,048 bytes | 512 bytes | 512 bytes | 4,096 bytes |
| 128 to 256MB | n/a | 4,096 bytes | 512 bytes | 512 bytes | 4,096 bytes |
| 256 to 512MB | n/a | 8,192 bytes | 4,096 bytes | 512 bytes | 4,096 bytes |
| 512 to 1,024MB | n/a | 16,384 bytes | 4,096 bytes | 1,024 bytes | 4,096 bytes |
| 1,024 to 2,048MB | n/a | 32,768 bytes | 4,096 bytes | 4,096 bytes | 4,096 bytes |
| 2,048 to 6,128MB | n/a | n/a | 4,096 bytes | 4,096 bytes | 4,096 bytes |

- Why are there N/A‘s?



# FS Layers: Management

- Space Management
  - This layer logically keeps track of all the blocks from the Allocation layer below.
- FAT (file allocation table) uses…a table …to track all the allocation units….in the file system…



# FS Layers: Management

- Files that are larger than a single allocation unit span multiple units
- The FAT table has an entry for each block possible values are
  - Address for next block of the file (contiguous or not)
  - EOF
  - Bad Block



# FAT

- MSWIN4.1 partitions (string Search)



# The inode

- The Ext2 file system until recently could easily be argued to be the most used FS in linux. Now more and more FS's are being used, (Ext3, XFS, Reiserfs….)



# The inode

- Inodes contain meta-data for files
- One piece of data is a link-count
  - Every link to the file ups the count
  - Every deletion of a link to the file decrements
  - If the count is 0 – the file is ‘deleted’



# The inode

- The Inode contians:
  - Mode of the file (everything is a file)
  - Link-count
  - UID
  - GID
  - Size
  - Access time
  - Mod time
  - Address for the file (data portion)
  - Number of blocks
  - version



# Inode

- Unlike FAT, the inode keeps pointers to the blocks that contain the information for the file.
  - Up to three (or so ) levels of pointers:
    - Direct (original 13 pointers)
    - Indirect (first ptr layer – 128 more ea)
    - Double indirect (128 more ea)
    - Triple indirect (128 more ea)

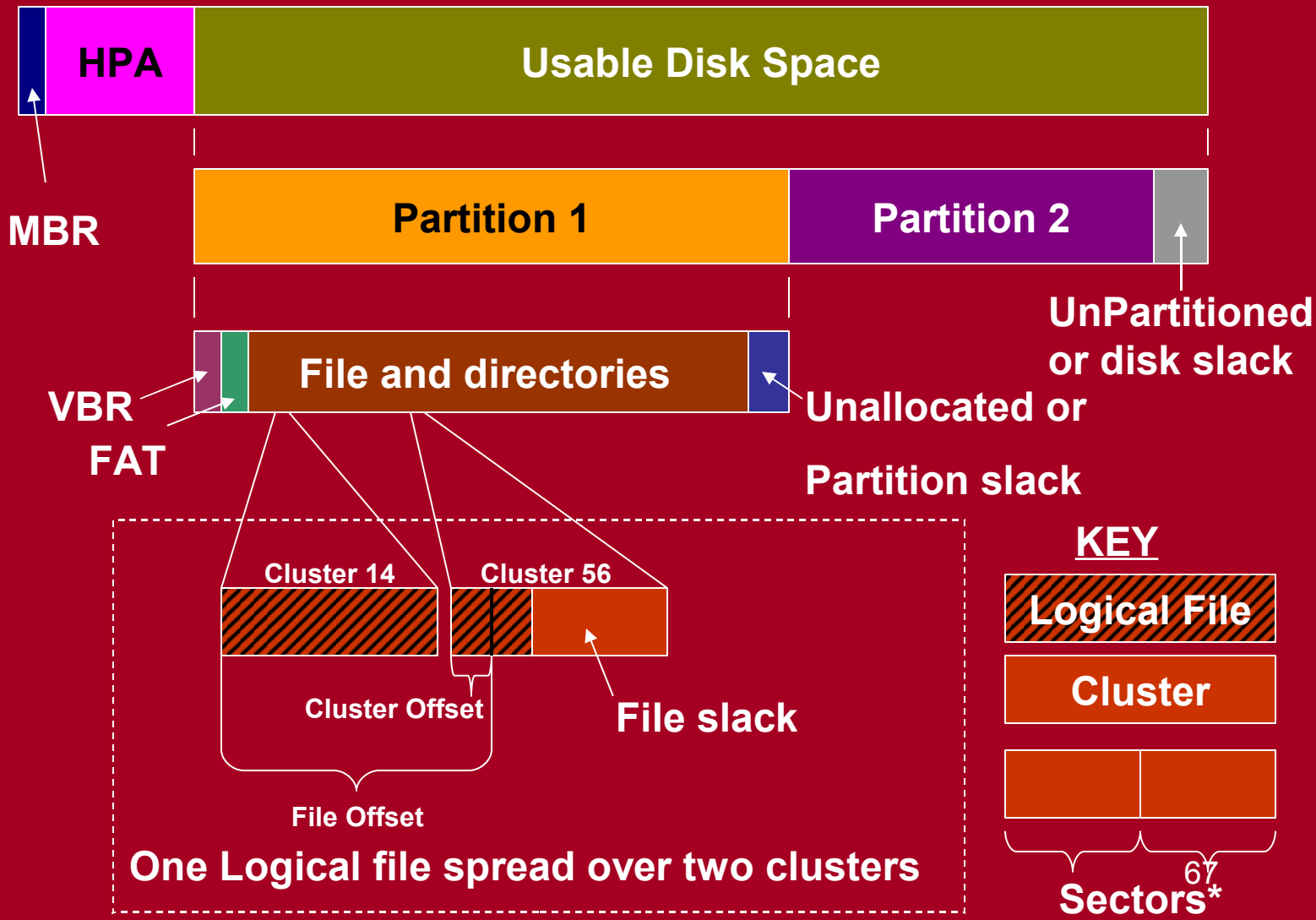


# FS Layers: Directories / files

- Logically used by users to segment data.
- Some systems have very little to distinguish between a file and a directory
- Most files will have metadata about the file itself – much like a header in network data structures or email


Your Key to Security

# Graphically…





# Partition finding

- Look for strings like MSWIN4.1 or NTFS to locate the beginning of FAT/NTFS partitions (existing or deleted)



# Files and Paths

- C:\data\pictures\vacation.jpg
- C:\data\pictures\ is the path
- vacation.jpg is the file

- What about:
- /home/timv/Desktop/dilbert.tgz



# Logical file systems

- Windows
  - A:
  - C:
    - Docs n set..
    - Windows
      - Sys32
        - » drivers
      - sys
  - D:
  - E:
- Linux
  - /
    - Bin
    - Etc
      - fstab
    - Dev
      - sda1
    - Boot
    - Home
    - Initrd
    - Mnt
      - Floppy
      - usb
    - Usr
    - var



# Drives

- Drives are files too
  - /dev/hda  /dev/hdb
- SCSI (and firewire, and usb …)
  - /dev/sda
- There are other ‘oddball ones..”
  - /dev/xd* for XT disks
  - /dev/ed* for ESDI disks
  - ….



# Partitions

- Partitions are part of drives
  - /dev/hda1 /dev/hda2

  (notice the number after the device)
- For Example
  - hdb2 is the second partition on the second IDE hard drive
  - What is?
    - sda3? sdb5? hda2?



# Devices

- /dev/fd0 or /dev/floppy is a floppy
- /dev/cdrom is cdrom
  - Do this at a linux prompt:

    cd /dev

    ls –al | less



# Fdisk

- Typically lives in regular only /sbin might have to “which” it or “locate” it if it’s not in your path
- Called the Partition Table Manipulator for Linux
- Allows you to view (and change) information in the partition table



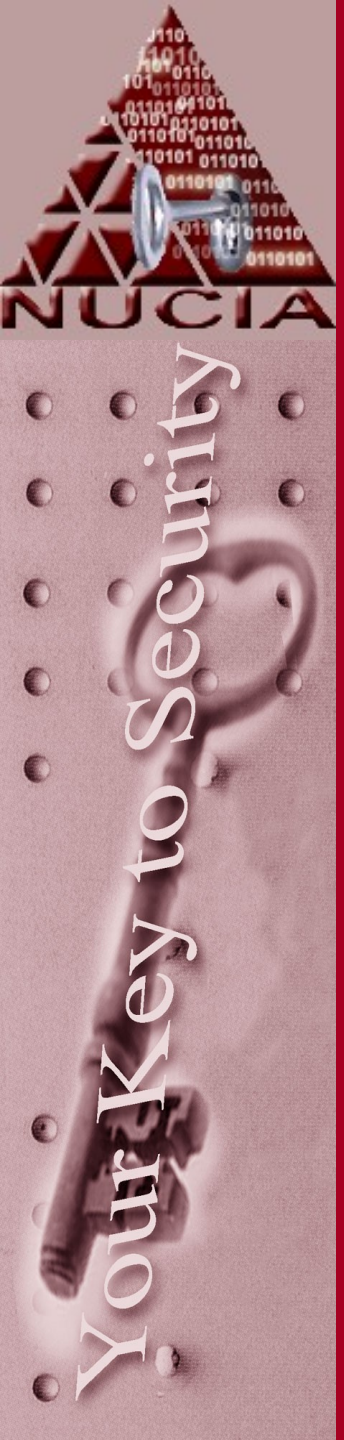
# Fdisk

- Popular options
  - l  list the partition table – by far the one used most – uses /proc/partitions
  - u  when listing, use sector sizes instead of cylinders
  - s  size of a partition in blocks
  - b  sector size – uhm 512 and high powers of 2…typically this setting is done by the OS



# Example

- On a downed machine
- Insert your trusty bootable distro
- Turn on machine
  - BEFORE boot finishes verify BIOS settings, otherwise you may inadvertently taint evidence
- Open terminal
- Switch to superuser with su
- /sbin/fdisk -l




# Class activity

- Helix CD's
- On the computer in front of me:
  - 1) How many Hard Disks present?
  - 2) How many Partitions?
  - 3) What type of Drives is it/are they?
  - 4) What file systems are indicated?



# Delving a bit deeper

- The partition table is part of the MBR
  - Master Boot Record
  - Boot Record
  - Boot Sector
- Traditionally located at cylinder 0, head 0, sector 1.
- Don’t confuse this with a “volume boot record” which exists inside the partition..



# MBR

- 4 primary partitions can reside in the primary partition table, if you need more than that – you have to use extended partitions (these are bootable partitions)
- Also the bootstrap code – Master boot code – exists here to enable the disk to boot, basically just transfers control to the boot partition when executed



# MBR

- Damaging or corrupting the MBR can be catastrophic to the well being of your hard data
- Luckily there are ways to recover ‘lost’ partitions
  - There is even freeware!
  - Testdisk by cg security (theultimatebootdisk)



# Partition Table

- So…open up your favorite hex viewer
- The first 446 sectors are for executable code
- The preceding 64 bytes before the 0x55AA is the table.
- The first byte is always 80, the last two are always 55 and AA
- Each Partition table entry is 16 bytes long.



# Example MBR

**Entire MBR record in hex and ASCII**

```
    OFFSET 0 1 2 3  4 5 6 7  8 9 A B  C D E F  *0123456789ABCDEF*
    000000 fa33c08e d0bc007c 8bf45007 501ffbfc *.3.....|..P.P...*
    000010 bf0006b9 0001f2a5 ea1d0600 00bebe07 *................*
    000020 b304803c 80740e80 3c00751c 83c610fe *.....t....u.....*
    000030 cb75efcd 188b148b 4c028bee 83c610fe *.u......L.......*
    000040 cb741a80 3c0074f4 be8b06ac 3c00740b *.t....t.......t.*
    000050 56bb0700 b40ecd10 5eebf0eb febf0500 *V.......^.......*
    000060 bb007cb8 010257cd 135f730c 33c0cd13 *..|...W.._s.3...*
    000070 4f75edbe a306ebd3 bec206bf fe7d813d *Ou...........}.=*
    000080 55aa75c7 8bf5ea00 7c000049 6e76616c *U.u.....|..Inval*
    000090 69642070 61727469 74696f6e 20746162 *id partition tab*
    0000a0 6c650045 72726f72 206c6f61 64696e67 *le.Error loading*
    0000b0 206f7065 72617469 6e672073 79737465 * operating syste*
    0000c0 6d004d69 7373696e 67206f70 65726174 *m.Missing operat*
    0000d0 696e6720 73797374 656d0000 00000000 *ing system......*
    0000e0 00000000 00000000 00000000 00000000 *................*

    0000f0 TO 0001af SAME AS ABOVE  - EMPTY

    0001b0 00000000 00000000 00000000 00008001 *................*
    0001c0 0100060d fef83e00 00000678 0d000000 *..........x....*
    0001d0 00000000 00000000 00000000 00000000 *................*
    0001e0 00000000 00000000 00000000 00000000 *................*
    0001f0 00000000 00000000 00000000 000055aa *..............U.*
```



# Partition Table demystified

| 80 | 01010006 | 0ebe94 | 3e000000 | 0c610900 |
|---|---|---|---|---|

**1 byte active partition flag**

**3 bytes CHS starting point (int 13 format)**

**1 byte partition type**

**3 bytes CHS ending point (int 13)**

**4 bytes starting LBA**

**4 bytes sector size**

**=**

**16 bytes total per entry**



# Linux Topics for research

ls –al

Sticky bit

Set uid

Set gid

Permission bits

mke2fs

e2fsck

badblocks

mount


# FAT

File Allocation Tables and the people that groom them.



# FAT

- FAT12
- FAT16
- FAT32
- VFAT



# FAT

- As mentioned previously..
- The FAT is a Table used to keep track of File Allocation
  - It's right after the volume boot record
  - There's a backup right after that
  - Each cluster has an entry



# FAT32

- The two tables…
  - Either can be primary or backup
  - The method of backup can be disabled
  - Allows for protection of the backup



# Directories

- Each directory is nothing more than a specially formatted file
- Directories are 32 bytes long:
  - File name and extension
    - 11 char dos 8.3 based
  - File attribute byte
    - From left to right: Readonly, hidden, system, vol label, directory, archive
  - Change date/time
  - File size
  - Pointer to starting cluster



# Directories

- Special directories:
  - Navigation
    - The . is the current directory
    - The .. is the previous directory
  - Root
    - “base of the tree”
    - Only one per volume
    - Directly after the two FATs
    - Under FAT12/FAT16/VFAT fixed size…



# Root dir size limit

| Vol Type | # of root entries |
|---|---|
| 360 kB 5.25" Floppy Disk | 112 |
| 720 kB 5.25" Floppy Disk | 112 |
| 1.2 kB 5.25" Floppy Disk | 224 |
| 1.44 kB 5.25" Floppy Disk | 224 |
| 2.88 kB 5.25" Floppy Disk | 448 |
| Hard Disk | 512 |



# FAT32

- No root dir size limit



# Lil history

- Dos 8.3
  - Nothing to do with version number
  - Eight chars for filename, 3 for extension
- Long filenames became desirable around windows 95
- …along came VFAT



# VFAT

- Allows file names up to 255 chars
- Predecessor to FAT32
- Used in win95, win98, ME
- Also assigned an 8.3 alias
  - First 6 ‘normal’ chars
  - Followed by a ~
  - Followed by a unique sequential number
  - Followed by the original 3 char extension



# Crazyness

- In win95
  - Long filenames are stored in multiple directory entries
  - To keep legacy programs from accidentally thinking they are actually files – or conversely empty and allocatable space – the readonly, hidden, system, and vol label bits are set….uhm, ok
  - Creates weird coping collision characteristics…
    - Old copy util – delete data (only understands short name)
    - New copy util – confuse user (creates new filenames from scratch)



# Cluster Chaining

- For each cluster that is in use, the corresponding entry contains a pointer to the next cluster that holds information for the file in question
- Or a special EOF bitstring
- Or a special not-in-use bitstring
- Or a special bad-cluster bitstring



# Open a file

- Ok, so there is a file called mypics.zip that is 20,000 bytes
- The OS is using 4,096 byte clusters (8 sectors per cluster)
- The file will require 5 clusters 20,000 / 4096 = 4.88 ( ~ 100 bytes of slack ;-)



# Open a file cont.

- Using winzip to open the file.. The application asks the OS to locate the file.
- The directory entry is queried for the initial cluster number
  - 21241
- To find the second, the FAT entry for 21241 is queried
  - 32423
- This continues until the last one is found and the FAT entry is EOF
- EOF is typically all one's…12, 16, etc



# FAT12

- The oldest type of FAT uses a 12-bit binary number to hold the cluster number.
- A volume formatted using FAT12 can hold a maximum of 4,086 clusters, which is 2^12 minus a few values (to allow for reserved values to be used in the FAT).
- FAT12 is therefore most suitable for very small volumes, and is used on floppy disks and hard disk partitions smaller than about 16 MB.



# FAT16

- The FAT used for most older systems, and for small partitions on modern systems, uses a 16-bit binary number to hold cluster numbers.
- When you see someone refer to a "FAT" volume generically, they are usually referring to FAT16, because it is the de facto standard for hard disks, even with FAT32 now more popular than FAT16.
- A volume using FAT16 can hold a maximum of 65,526 clusters, which is 2^16 less a few values (again for reserved values in the FAT).
- FAT16 is used for hard disk volumes ranging in size from 16 MB to 2,048 MB. VFAT is a variant of FAT16.



# FAT32

- The newest type of FAT is supported by newer versions of Windows, including Windows 95's SR2 release, as well as Windows 98, Windows ME and Windows 2000.
- FAT32 uses a 28-bit binary cluster number--*not* 32, because 4 of the 32 bits are "reserved".
- 28 bits is still enough to permit huge volumes--FAT32 can theoretically handle volumes with over 268 million clusters, and will support drives up to 2 TB  (yeah right) in size.
- However to do this the size of the FAT grows very large…



# The file allocation table table

| Drive Size (logical volume) | FAT Type | Sectors /Cluster | Cluster Size |
|---|---|---|---|
| 0 MB - 15 MB | 12-bit | 8 | 4K |
| 16 MB - 127 MB | 16-bit | 4 | 2K |
| 128 MB - 255 MB | 16-bit | 8 | 4K |
| 256 MB - 511 MB | 16-bit | 16 | 8K |
| 512 MB - 1023 MB | 16-bit | 32 | 16K |
| 1024 MB - 2048 MB | 16-bit | 64 | 32K |
| 2048 MB - 4096 MB | 16-bit | 128 | 64K* |
| 4096 MB - 8192 MB | 16-bit | 256 | 128K* |
| 8192 MB - 16384 MB | 16-bit | 512 | 256K* |

*not supported in some software



# Varying amounts of slack

- As disk size increases so does the size of FAT needed to keep track of the clusters
- Actually, on the same disk, if the cluster size decreases, the number of FAT entries increases…FAT32 allows sort of an ‘adapting’ to drive



# Happy FAT

- This auto-reduction is going to keep the FAT at around 8MB instead if increasing with HD size…



# Happy FAT

| Part Size | 4KB cluster | 8 KB | 16kB | 32kB |
|---|---|---|---|---|
| 8 GB | 8 MB | 4 MB | 2MB | 1 MB |
| 16 GB | 16 MB | 8 MB | 4 MB | 2 MB |
| 32 GB | 32 MB | 16 MB | 8 MB | 4 MB |
| 54 GB | 64 MB | 32 MB | 16 MB | 8 MB |
| 2 TB | | 1,024 MB | 512 MB | 256 MB |



# Rules of Thumb

| | FAT12 | FAT16 | FAT32 |
|---|---|---|---|
| Typ. Use | Floppy, small HD | Small-med hd | Med-large HD |
| Each FAT entry | 12 bits | 16 bits | 28 bits |
| Max clusters | 4,086 | 65,526 | 268,435,456 |
| Cluster size | .5-4KB | 2-32KB | 4-32KB |
| Max Vol size | 16,736,256 | 2,147,123,200 | 2^41 |


Your Key to Security

| | FAT12 | FAT16 | FAT32 |
|---|---|---|---|
| **Developer** | Microsoft | | |
| **Full Name** | File Allocation Table | | |
| | (12-bit version) | (16-bit version) | (32-bit version) |
| **Introduced** | 1977 (Microsoft Disk BASIC) | July 1988 (MS-DOS 4.0) | August 1996 (Windows 95 OSR2) |
| **Partition identifier** | 0x01 (MBR) | 0x04, 0x06, 0x0E (MBR) | 0x0B, 0x0C (MBR)<br>EBD0A0A2-B9E5-4433-87C0-68B6B72699C7 (GPT) |
| **Structures** | | | |
| **Directory contents** | Table | | |
| **File allocation** | Linked List | | |
| **Bad blocks** | Linked List | | |
| **Limits** | | | |
| **Max file size** | 32 MiB | 2 GiB | 4 GiB |
| **Max number of files** | 4,077 | 65,517 | 268,435,437 |
| **Max filename size** | 8.3 or 255 characters when using LFNs | | |
| **Max volume size** | 32 MiB | 2 GiB<br>4 GiB with some implementations | 2 TiB |
| **Features** | | | |
| **Dates recorded** | Creation, modified, access | | |
| **Date range** | January 1, 1980 - December 31, 2107 | | |
| **Forks** | Not natively | | |
| **Attributes** | Read-only, hidden, system, volume label, subdirectory, archive | | |
| **Permissions** | No | | |
| **Transparent compression** | Per-volume, Stacker, DoubleSpace, DriveSpace | | No |
| **Transparent encryption** | Per-volume only with DR-DOS | | No |

Rules of Thumb wikipedia.org



# A note on endianess

- Lil endian vs big endian:

  You want to store A1B2C3D4 (4 bytes)

  Big – endian A1B3C3D4

  Lil – endian D4C3B2A1

  The storage for all 4 bytes is located, then each byte is stored started at the “back” (bytes are generally atomic)

  Why? Because x86 does that, because conversion from int to long to ___ is easier

  This is one cause of software non-portability



# Let's get techie: Boot Sector

First 512 Bytes are the boot sector:

0-2 Jump to bootstrap (E.g. eb 3c 90; on i86: JMP 003E NOP. One finds either eb xx 90, or e9 xx xx. The position of the bootstrap varies.)

3-10 OEM name/version (E.g. "IBM 3.3", "IBM 20.0", "MSDOS5.0", "MSWIN4.0". Various format utilities leave their own name, like "CH-FOR18". Sometimes just garbage. Microsoft recommends "MSWIN4.1".)

11-12 Number of bytes per sector (512) Must be one of 512, 1024, 2048, 4096.

13 Number of sectors per cluster (1) Must be one of 1, 2, 4, 8, 16, 32, 64, 128. A cluster should have at most 32768 bytes. In rare cases 65536 is OK.

14-15 Number of reserved sectors (1) FAT12 and FAT16 use 1. FAT32 uses 32.

16 Number of FAT copies (2)

17-18 Number of root directory entries (224) 0 for FAT32. 512 is recommended for FAT16.

19-20 Total number of sectors in the file system (2880) (in case the partition is not FAT32 and smaller than 32 MB)

21 Media descriptor type (f0: 1.4 MB floppy, f8: hard disk;)

22-23 Number of sectors per FAT (9) 0 for FAT32.

24-25 Number of sectors per track (12)

26-27 Number of heads (2, for a double-sided diskette)

28-29 Number of hidden sectors (0) Hidden sectors are sectors preceding the partition.

30-509 Bootstrap

510-511 end Signature of 55 aa



- 000000 eb 3f 90 49 42 4d 20 20 33 2e 33 00 02 01 01 00
  000010 02 e0 00 40 0b f0 09 00 12 00 02 00 00 00 00 00
  000020 00 00 00 00 00 00 00 00 00 00 70 00 ff ff 49 42
  000030 4d 42 49 4f 20 20 43 4f 4d 00 50 00 00 08 00 18
  000040 0a fc 33 c0 8e c0 fa 8e d0 bc 00 7c fb 33 d2 cd
  000050 13 bd 78 00 8b fc c5 76 00 89 7e 00 8c 46 02 b9
  000060 0b 00 f3 a4 91 8e d8 8b ec c6 46 04 12 8a 46 0d
  000070 89 46 24 8a 46 10 f7 66 16 03 46 0e 83 d2 00 8b
  000080 4e 0b 81 f9 00 02 74 11 72 54 d1 e9 03 c0 83 d2
  000090 00 d1 66 24 d1 66 16 eb e9 83 7e 16 0c 77 05 c7
  0000a0 46 2c ff 0f 03 46 1c 13 56 1e 50 52 8b 5e 11 53
  0000b0 83 c3 0f b1 04 d3 eb 88 5e 41 8e 46 3c 06 e8 b3
  0000c0 00 07 89 46 42 89 56 44 59 2b ff 51 57 8d 76 2e
  0000d0 b9 0b 00 f3 a6 5f 59 74 13 83 c7 20 e2 ed be d2
  0000e0 7d e8 dd 00 98 cd 16 ea 00 00 ff ff 26 ff 75 1a
  0000f0 26 8b 45 1c 05 ff 01 d1 e8 88 66 26 be c7 7d e8
  000100 bf 00 59 5a 58 51 8b 4e 16 2b c1 83 da 00 8e 46
  000110 3c 51 c6 46 41 01 e8 5b 00 59 e2 f5 5b 8e 46 2a
  000120 8b 46 24 88 46 41 28 46 26 9c 73 06 8a 56 26 00
  000130 56 41 53 4b 4b f7 e3 03 46 42 13 56 44 e8 34 00
  000140 5b 06 8b c3 d1 e3 c4 7e 3c 72 02 8e c7 81 7e 2c
  000150 ff 0f 75 12 03 d8 d1 eb 26 8b 1f 73 04 b1 04 d3
  000160 eb 80 e7 0f eb 03 26 8b 1f 07 9d 77 b3 8a 96 fd
  000170 01 ff 6e 28 33 db 50 52 e8 15 00 8c c0 05 20 00
  000180 8e c0 5a 58 05 01 00 83 d2 00 fe 4e 41 75 e5 c3
  000190 f7 76 18 42 8a ca 33 d2 f7 76 1a 8a f2 d0 cc d0
  0001a0 cc 80 e4 c0 0a cc 8a e8 8a 96 fd 01 b8 01 02 cd
  0001b0 13 73 dc fe 4e 40 75 f4 e9 23 ff b4 0e 2b db cd
  0001c0 10 ac 84 c0 75 f5 c3 4c 6f 61 64 69 6e 67 2e 2e
  0001d0 2e 00 0d 0a 43 61 6e 6e 6f 74 20 6c 6f 61 64 20
  0001e0 66 69 6c 65 0d 0a 50 72 65 73 73 20 61 20 6b 65
  0001f0 79 20 74 6f 20 72 65 74 72 79 0d 0a 00 00 55 aa



# File Date/Time stamp

- 16 bits each
  - 5 bit hour
  - 6 bit minute
  - 5 bit second
  - 7 bit year
  - 4 bit month
  - 5 bit day



# File Date/Time stamp

- Year is actually from 1980+ 0-127
- Month 1-12
- Day 1-31
- Hour 0-23
- Min 0-59
- Seconds 0-30
- (yup that's not a typo)



# Resources

- http://www.win.tue.nl/~aeb/linux/fs/fat
- http://www.pcguide.com/ref/hdd/file/
- http://www.cs.umd.edu/class/spring2
- http://support.microsoft.com/?kbid=140365



# References

- www.ata-atapi.com/hwwtab.htm
- www.ata-atapi.com/hiwmbr.htm
- www.cgsecurity.org/index.html
- http://www.pcguide.com



# Resources

- http://www.wiebetech.com/
- http://odessa.sourceforge.net/
- http://sourceforge.net/projects/dcfld
- http://www.sf-soft.de/winhex/index-
- http://www.law.cornell.edu/rules/fre