Ungraded Pre-Lecture Quiz (*To Help Refresh*)

- What's the difference between a "***cluster chain***" in FAT file system and a "***cluster run***" in NTFS?
- What are the involved key ***file system's components*** in FAT file system and NTFS, respectively?

Quick Reminders/Announcements

- ***No* graded lab tasks** from today's Lab 5
- Please register your **group** (of 4 members) by **19 February** by responding to a Canvas discussion/forum thread
- After that, I will randomly-assign those unassigned and possibly merge partial groups by **26 February**

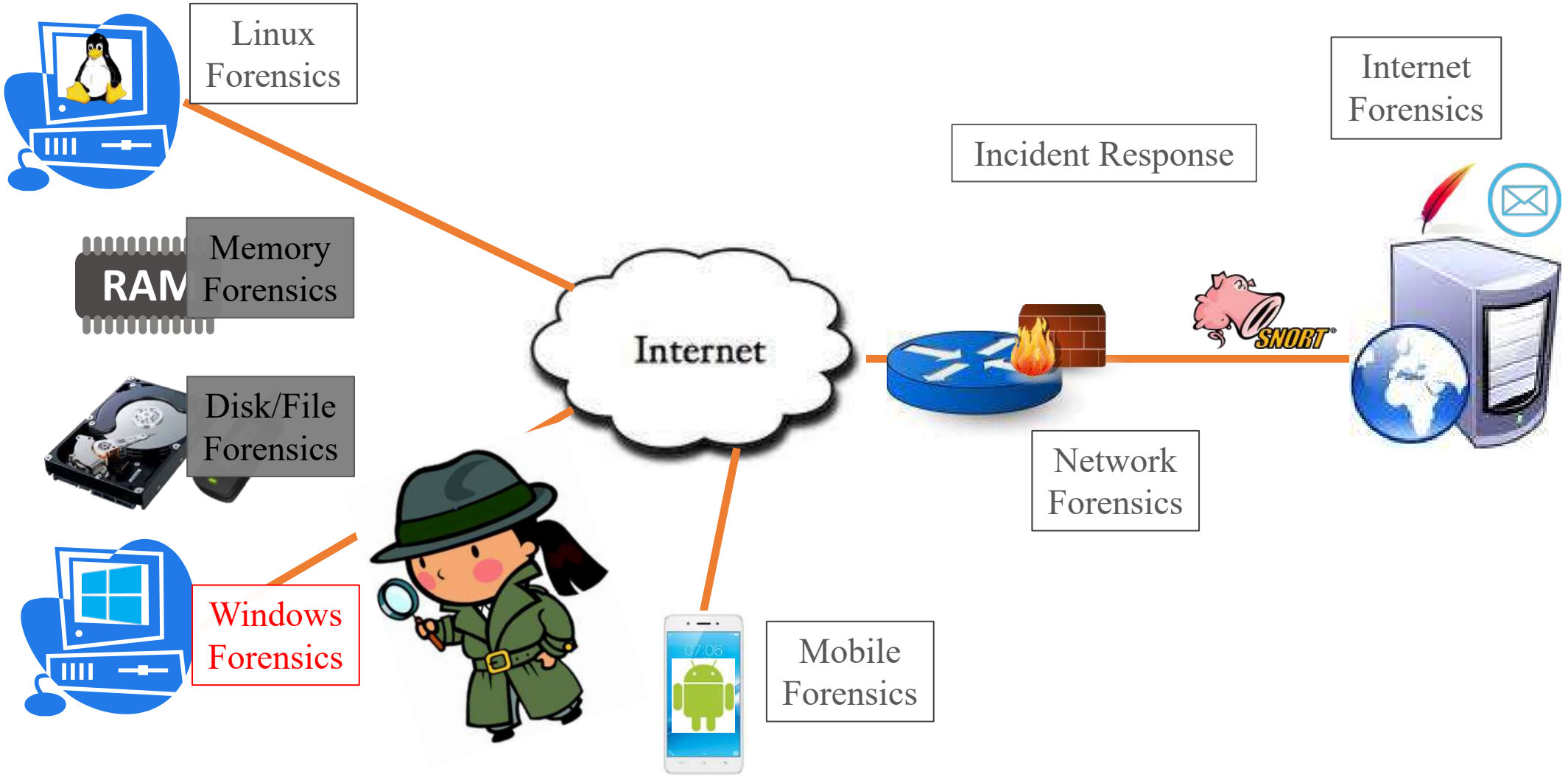
IFS4102: Digital Forensics

Lecture 6: Windows (Part 2) & Linux Forensics

Outline

- Timestamps in Windows
- Windows event log analysis
- Windows artefact analysis: prefetch-file, shortcut, jump-list, user assists, thumbnail cache, shellbags & recycle-bin analyses
- Linux/UNIX forensics
- Linux partition & files
- Linux ext file system
- Live vs offline Linux analysis
- Linux log analysis
- Lab 6 exercises

This Lecture's Focus



Timestamps in Windows

Timestamps in Windows

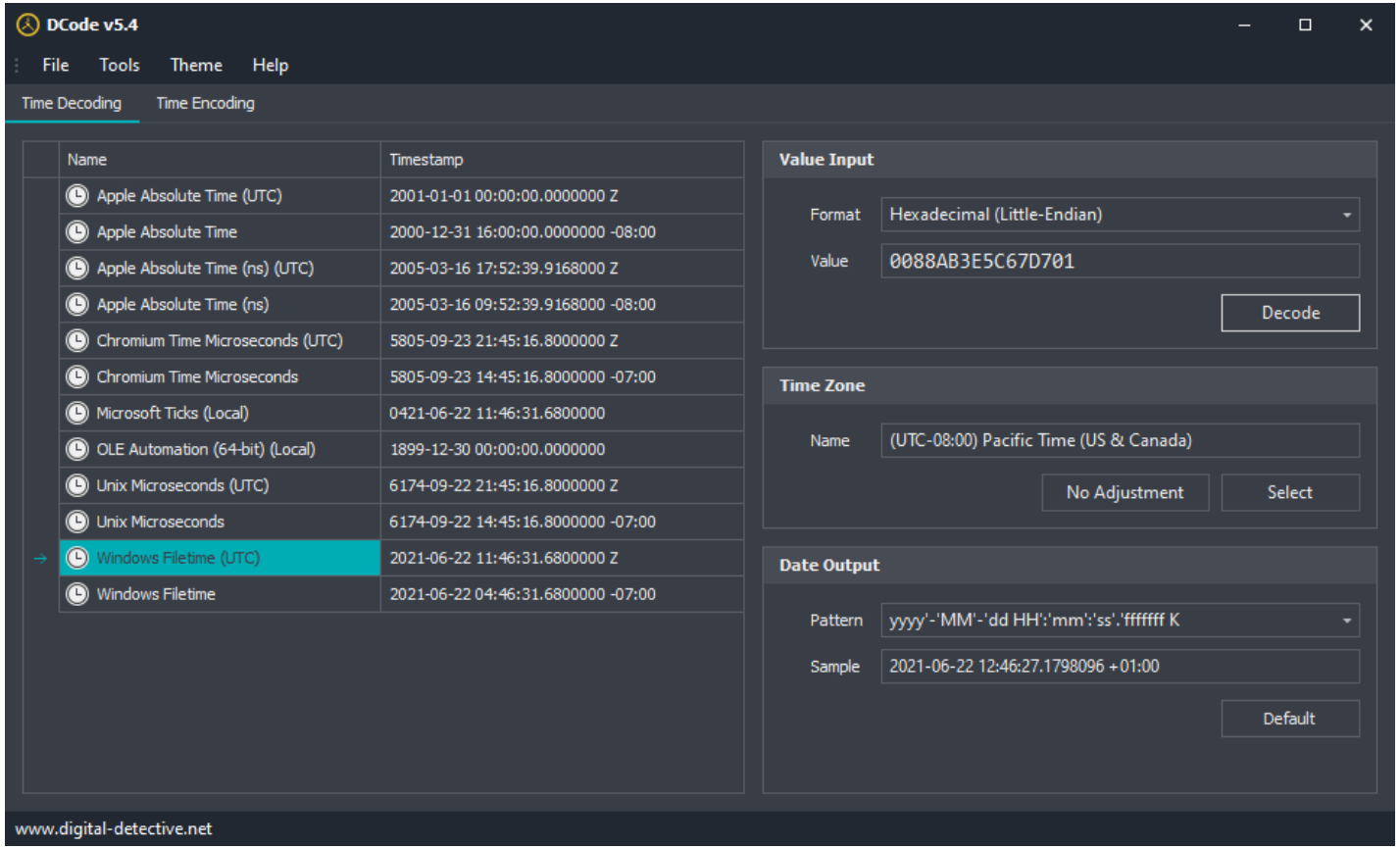
- Time-based **evidence** in Windows: in registry, file system, etc.
- Timestamps in **several formats**, including:
  - 32-bit **Unix/epoch time**: *discussed earlier*
  - 64-bit ***Windows File Time*** **(*FILETIME*)**
  - *(Sometimes)* 128-bit ***Windows System Time*** (***SYSTEMTIME***): https://msdn.microsoft.com/en-us/library/ms724950(VA.85).aspx
  - **Human-readable string**: e.g. “01/10/2010”
- Let’s discuss ***Windows File Time*** format

Windows File Times

- A ***file time***: a 64-bit value representing the no of 100-nanosecond intervals that have elapsed since 12:00 A.M. January 1, 1601 UTC
- The system **records file times** when applications create, access, and write to files
- NTFS vs FAT:
  - NTFS stores time values in ***UTC format***
  - FAT stores time values based on the ***local time*** of the computer
  - **Example**: a file saved at 3:00pm PST in Washington is seen as
    → 6:00pm EST in New York (on an NTFS file system)
    → 3:00pm EST in New York (on a FAT file system)
- Ref: https://docs.microsoft.com/en-us/windows/win32/sysinfo/file-times

Windows File Times: Conversion Tool

- A good **time-conversion tool**:
  DCode (https://www.digital-detective.net/dcode/)



From: https://www.digital-detective.net/dcode/)

Windows Event Log Analysis

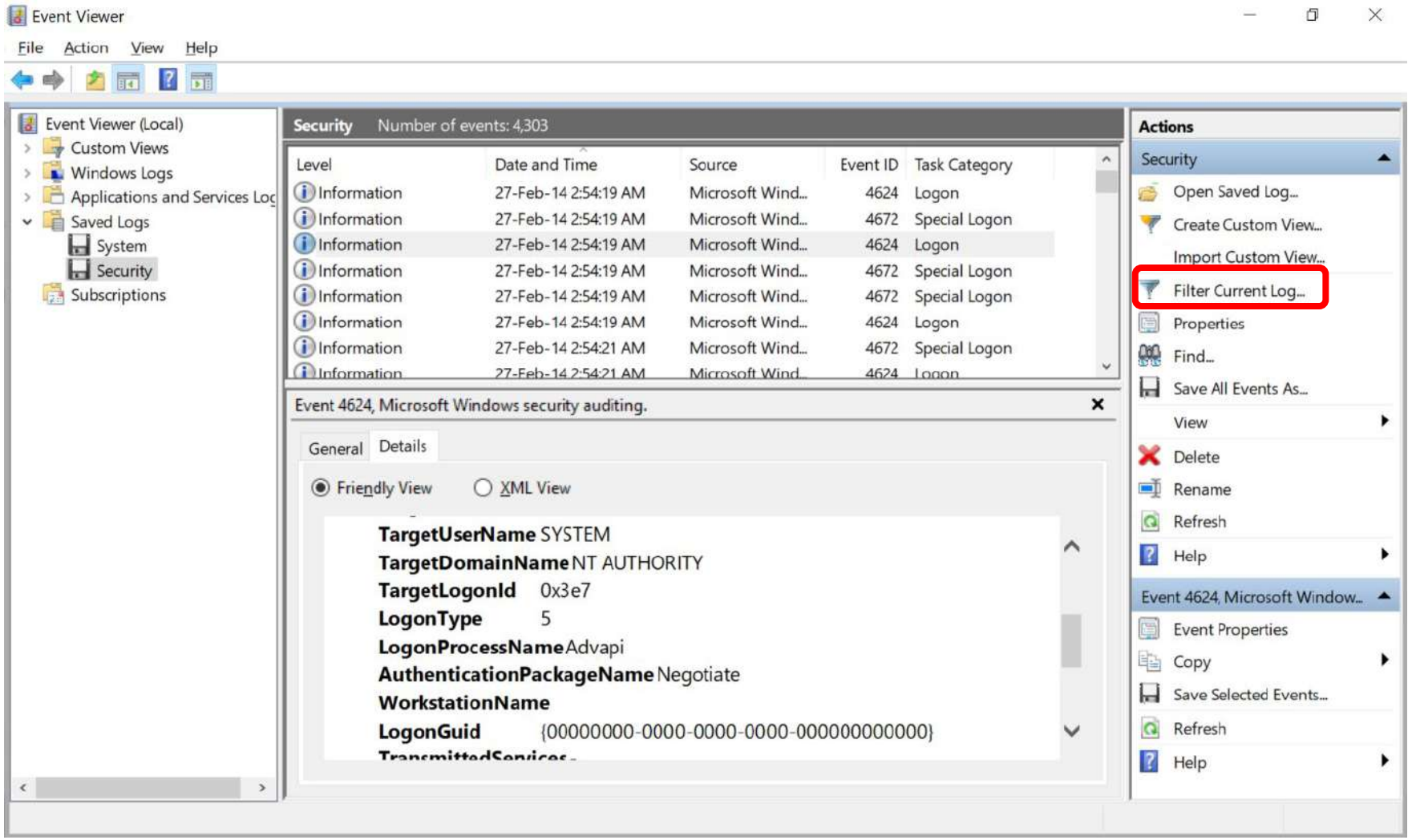
Windows Event Logs

- Very **important & useful** information!
- **Objects tracked** include:
  login/off (both successes and failures), user activities, Internet access, various significant events
- **File formats**:
  - **.evt**: up until Windows XP
  - **.evtx**: from Windows Vista and newer, with many **new features/ enhancements**, including:
    - New event **properties**
    - XML **format**
    - A new Event Viewer
    - A rewritten Windows Event Log service

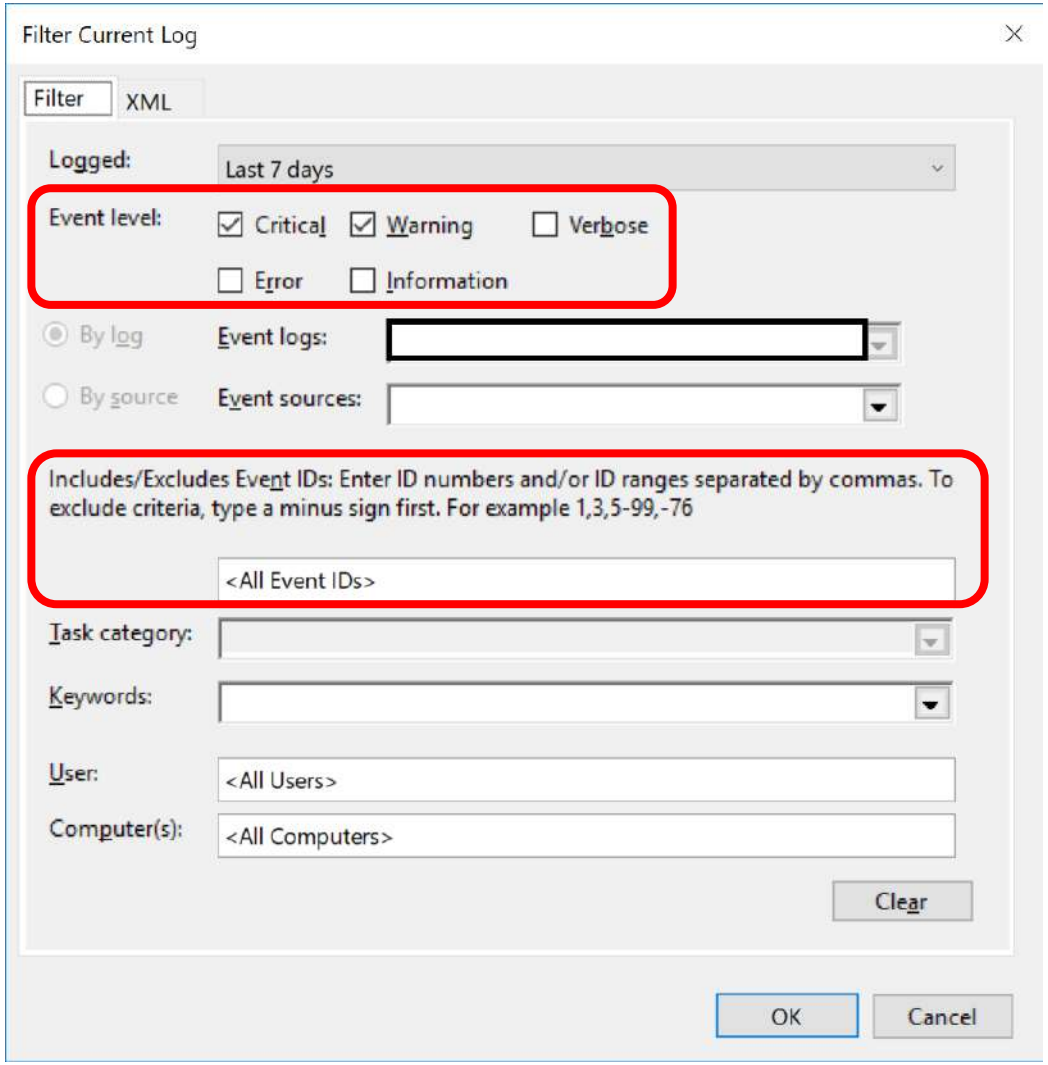
Windows Event Logs: Locations and Filenames

- Log files are usually **stored** under the %SystemRoot%\`System32\Config` folder (e.g. `C:\Windows\System32\Config`)
- **File folder locations & names** are set inside these **registry keys**:
  - **Security**: SYSTEM\ControlSet001\services\eventlog\Security
  - **System**: SYSTEM\ControlSet001\services\eventlog\System

Windows Event Viewer



Windows Event Viewer: Log Filtering Feature



Event Properties & Sample Event Analysis

- ***Event levels***: Information, Warning, Error, Critical
- ***Event ID***: a number identifying the particular **event type**
- ***Logon* activities**:
  - Event ID **4624(S)**: an account was **successfully logged on**
    See: https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4624
  - Event ID **4625(F)**: an account **failed to log on**
    See: https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4625
  - Also see different logon types: Interactive, Network, Batch, Service, …
- ***Logoff* activities**:
  - Event ID **4634(S)**: an account was **logged off**
    See: https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4634
  - Event ID **4647(S)**: **user-initiated logoff**
    See: https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4647

Successful Logon (Event ID 4624): Logon Types

Logon Type	Description
2	Interactive (logon at keyboard and screen of system)
3	Network (i.e. connection to shared folder on this computer from elsewhere on network)
4	Batch (i.e. scheduled task)
5	Service (Service startup)
7	Unlock (i.e. unnattended workstation with password protected screen saver)
8	NetworkCleartext (Logon with credentials sent in the clear text. Most often indicates a logon to IIS with "basic authentication") See this article for more information.
9	NewCredentials such as with RunAs or mapping a network drive with alternate credentials. This logon type does not seem to show up in any events. If you want to track users attempting to logon with alternate credentials see 4648. MS says "A caller cloned its current token and specified new credentials for outbound connections. The new logon session has the same local identity, but uses different credentials for other network connections."
10	RemoteInteractive (Terminal Services, Remote Desktop or Remote Assistance)
11	CachedInteractive (logon with cached domain credentials such as when logging on to a laptop when away from the network)

From: https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/event.aspx?eventid=4624

Failed Logon (Event ID 4625): Failure Information

Status and Sub Status Codes	Description (not checked against "Failure Reason:")
0xC0000064	user name does not exist
0xC000006A	user name is correct but the password is wrong
0xC0000234	user is currently locked out
0xC0000072	account is currently disabled
0xC000006F	user tried to logon outside his day of week or time of day restrictions
0xC0000070	workstation restriction, or Authentication Policy Silo violation (look for event ID 4820 on domain controller)
0xC0000193	account expiration
0xC0000071	expired password
0xC0000133	clocks between DC and other computer too far out of sync
0xC0000224	user is required to change password at next logon
0xC0000225	evidently a bug in Windows and not a risk
0xc000015b	The user has not been granted the requested logon type (aka logon right) at this machine

From: https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/event.aspx?eventid=4625

User-Initiated Logoff (Event ID 4647): Example

Examples of 4647

```
User initiated logoff:

Subject:
   Security ID:  WIN-R9H529RIO4Y\Administrator
   Account Name:  Administrator
   Account Domain:  WIN-R9H529RIO4Y
   Logon ID:  0x19f4c

This event is generated when a logoff is initiated but the token reference count is not zero and the l
ogon session cannot be destroyed.  No further user-initiated activity can occur.  This event can be in
terpreted as a logoff event.
```

From: https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/event.aspx?eventid=4625

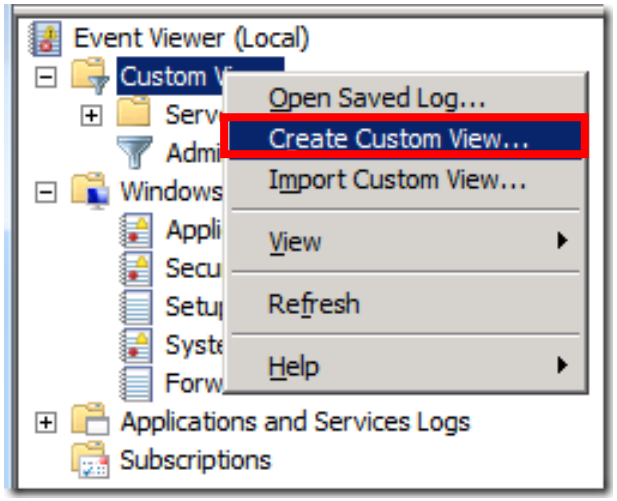
For more about events:

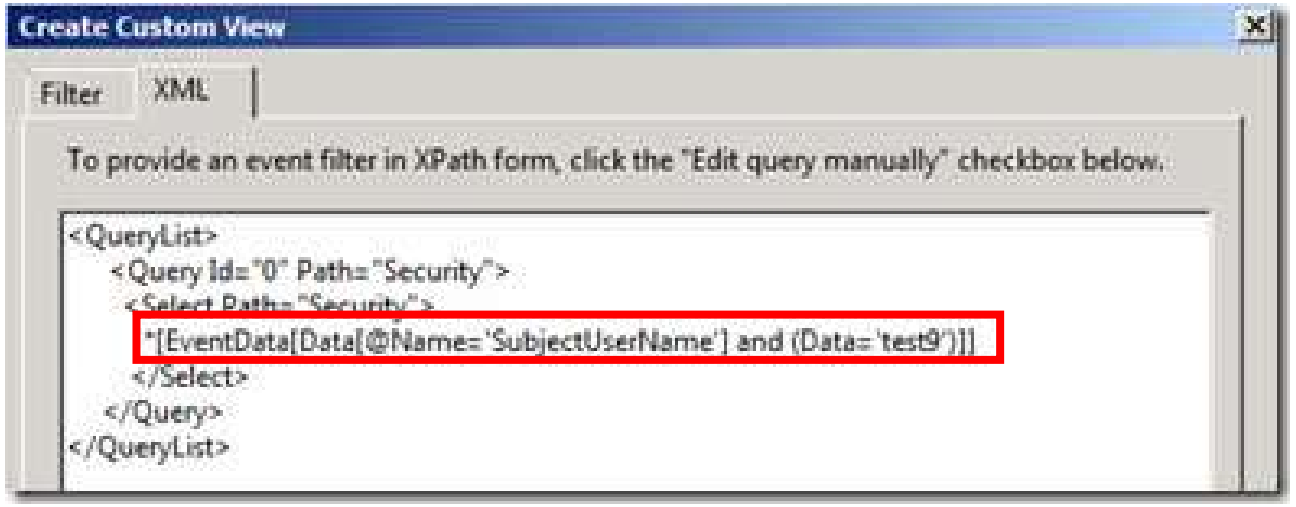
- Lab 6 Task 2
- See the list at: https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/default.aspx
- Some **other important events** will be discussed in our *Incident Response* part later!

Advanced Filtering in Windows Event Viewer

Optional

- Using **custom views** and **XML filtering**



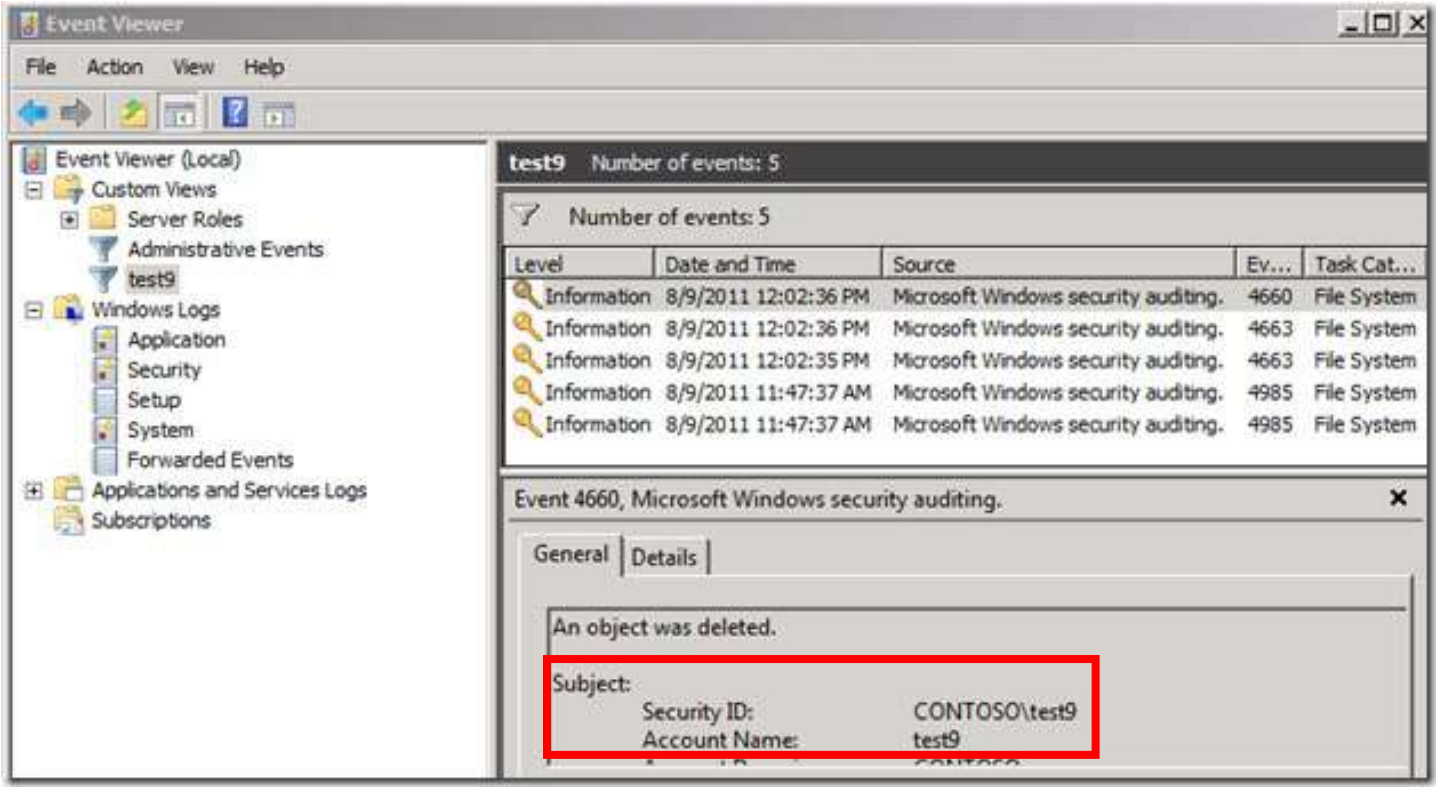


Ref: https://techcommunity.microsoft.com/t5/ask-the-directory-services-team/advanced-xml-filtering-in-the-windows-event-viewer/ba-p/399761

Advanced Filtering in Windows Event Viewer

Optional

- Using **XML filtering** and **custom views**



Ref: https://techcommunity.microsoft.com/t5/ask-the-directory-services-team/advanced-xml-filtering-in-the-windows-event-viewer/ba-p/399761

Windows Artefact Analysis

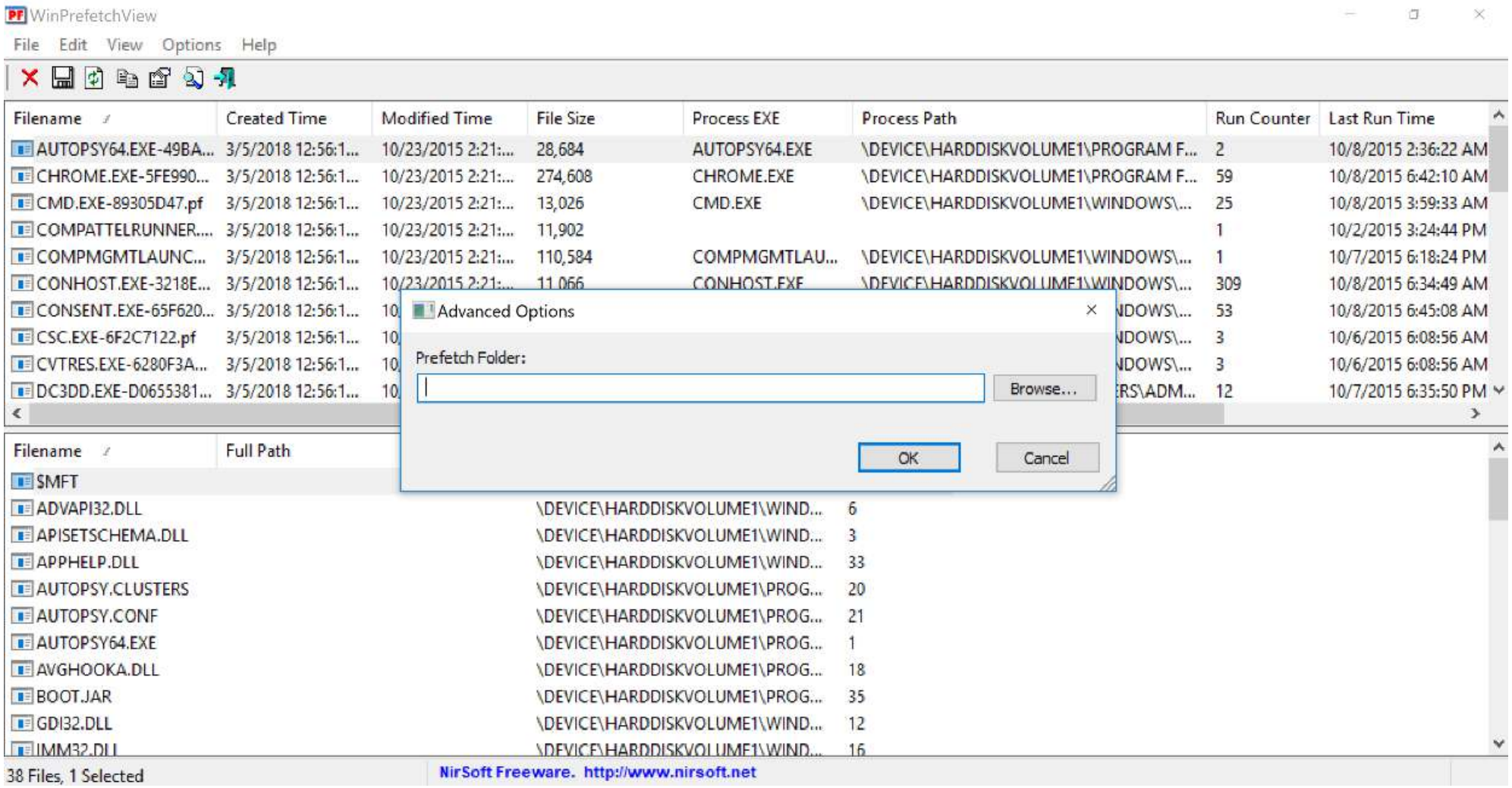
Why Windows Artefact Analysis

- Windows "***can remember***" what you previously did!
- Various **computer-generated files** for performance purposes, easier user access:
  - Prefetch files
  - Shortcuts (LNK/Link) files
  - Jump list files
  - User assists
  - Thumbnail cache files
  - Shellbags
- **Network & browser related artefacts**: *to be discussed in Lecture 7*
- And don't forget the **regular** files:
  system files, user's data files, downloaded files, deleted files

Windows Artefacts: Prefetch Files

- **Prefetch files**:
  - Used to increase **system performance**
  - **Cache Manager** monitors all files & directories read by application/process for **10 seconds**, and maps them into a <*appfilename-hash*>`.pf` file
  - *Why is **hash** appended*? The **same application filename** executed from **two separate locations** will create two prefetch files
  - Default prefetch folder: `C:\Windows\Prefetch`
  - File-number limits: 128 files (XP and Win 7), 1024 files (Win 8-10)
  - Forensic investigators can utilise prefetch files to find out **executed applications & their read files** (e.g. DLL files, configuration files, input files)
- Prefetch file analysis tool: **WinPrefetchView** (see Lab 6 Task 3)
- Reference: https://forensics.wiki/prefetch/

WinPrefetchView



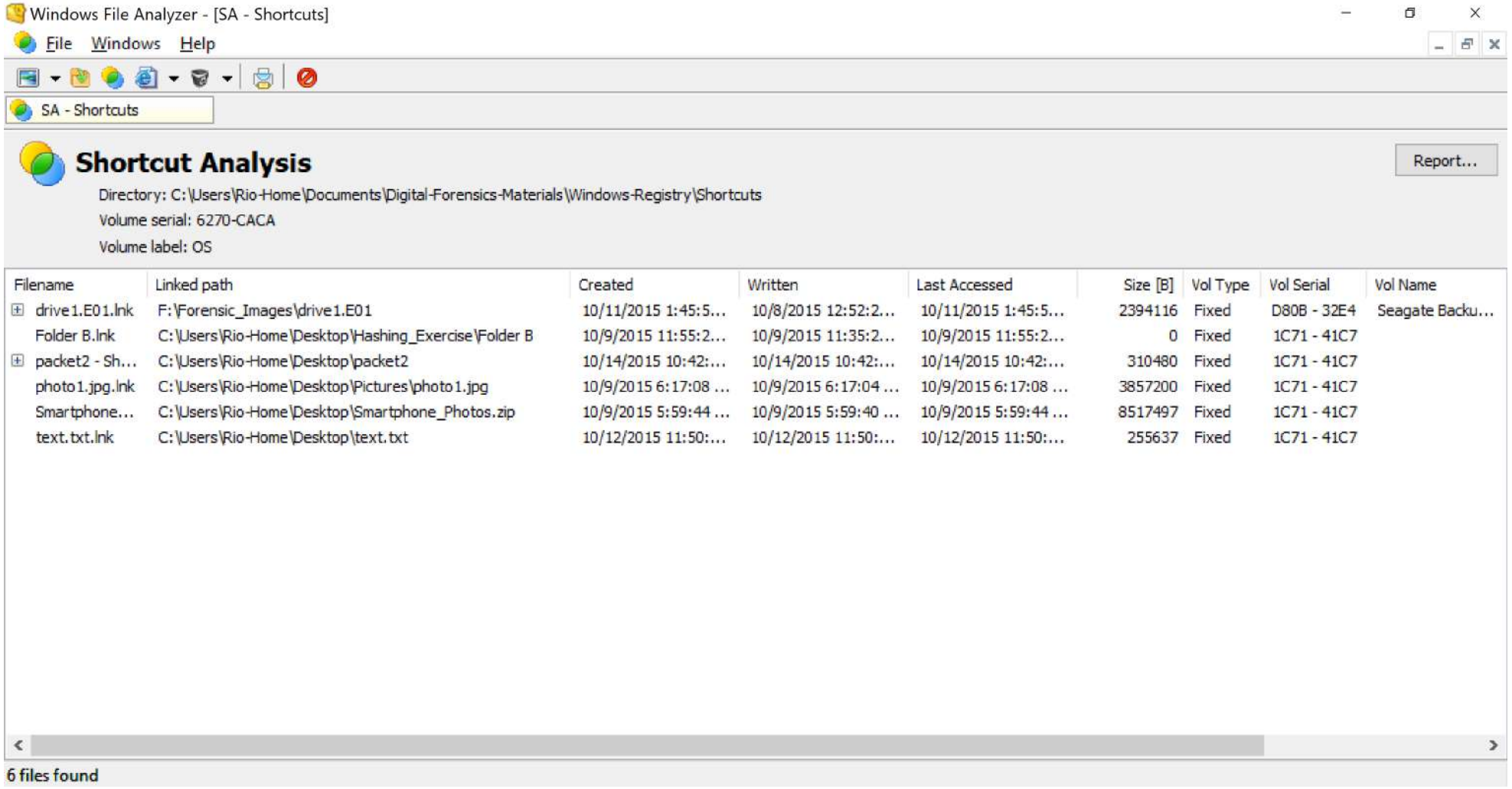
Sample Analysis using WinPrefetchView

- Information of a listed **application** (from the *upper pane*):
  - **Process file and path**: the previously-run application and its location
  - **Run counter**: how many times was the application run
  - **Last run time**: when was the last time the application run
- Information of the **files read** by the application within the first 10s after it was launched (from the *lower pane*):
  - File name
  - Full pathname

Windows Artefacts: Shortcuts (LNK/Link Files)

- **Shortcuts** for easier user access
- Created in **different** ways:
  - Software **installation**: Start menu, desktop, Quick launch
  - **Manual** shortcut created by user
  - File open via **Explorer** (recent files): %USERPROFILE%\Recent (XP), %AppData%\Microsoft\Windows\Recent\ (Win7/8/10)
  - File open by **Microsoft Office** (recent Office files): %AppData%\Microsoft\Office\Recent\
- Shortcuts are **seldom** removed by users: **valuable forensic artefacts**!
- Shortcut analysis tool: MiTec Windows File Analyzer (see Lab 6 Task 4)
- Reference: https://forensics.wiki/lnk/

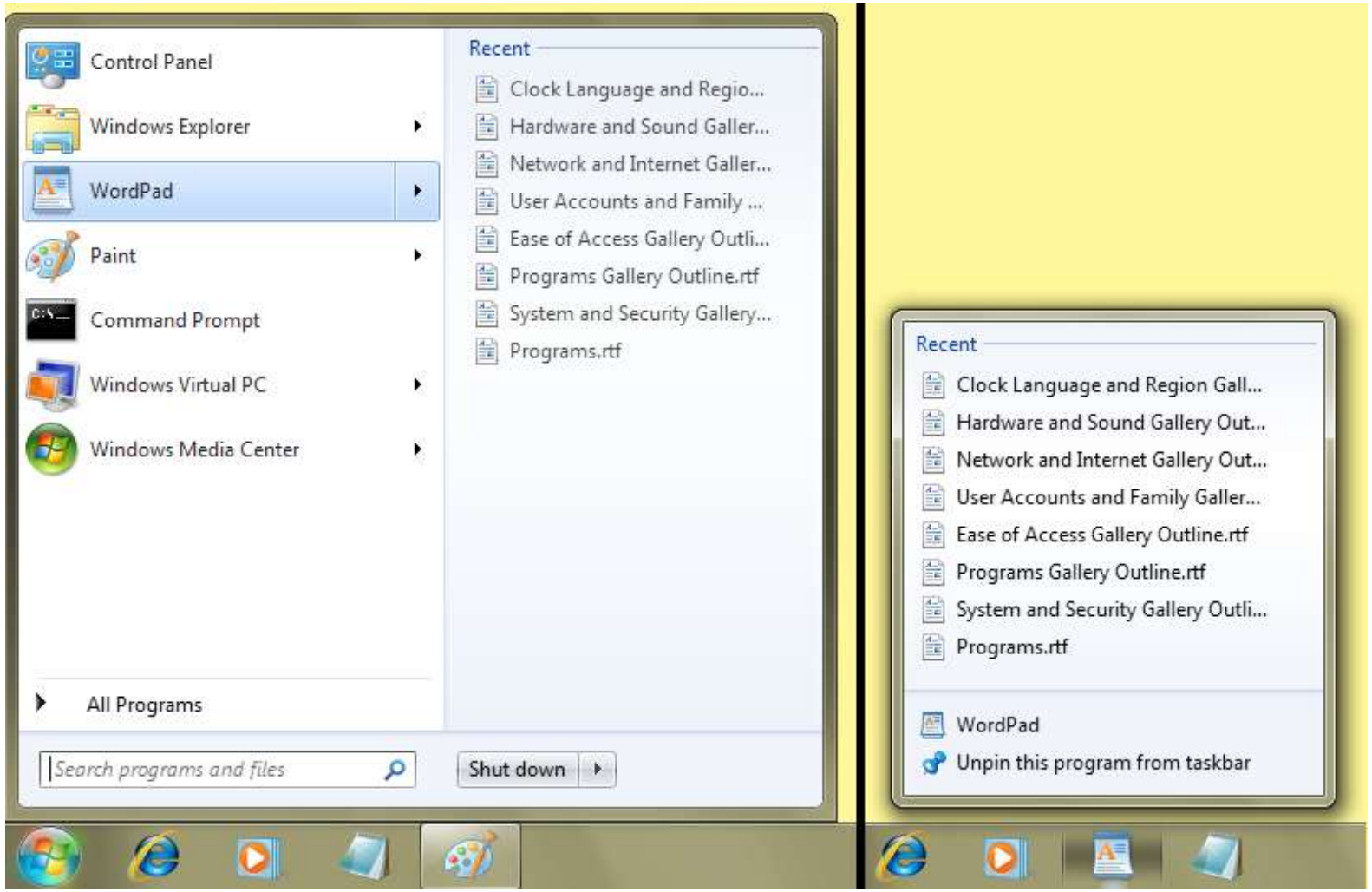
Windows File Analyzer (WFA)



Windows Artefacts: Jump Lists

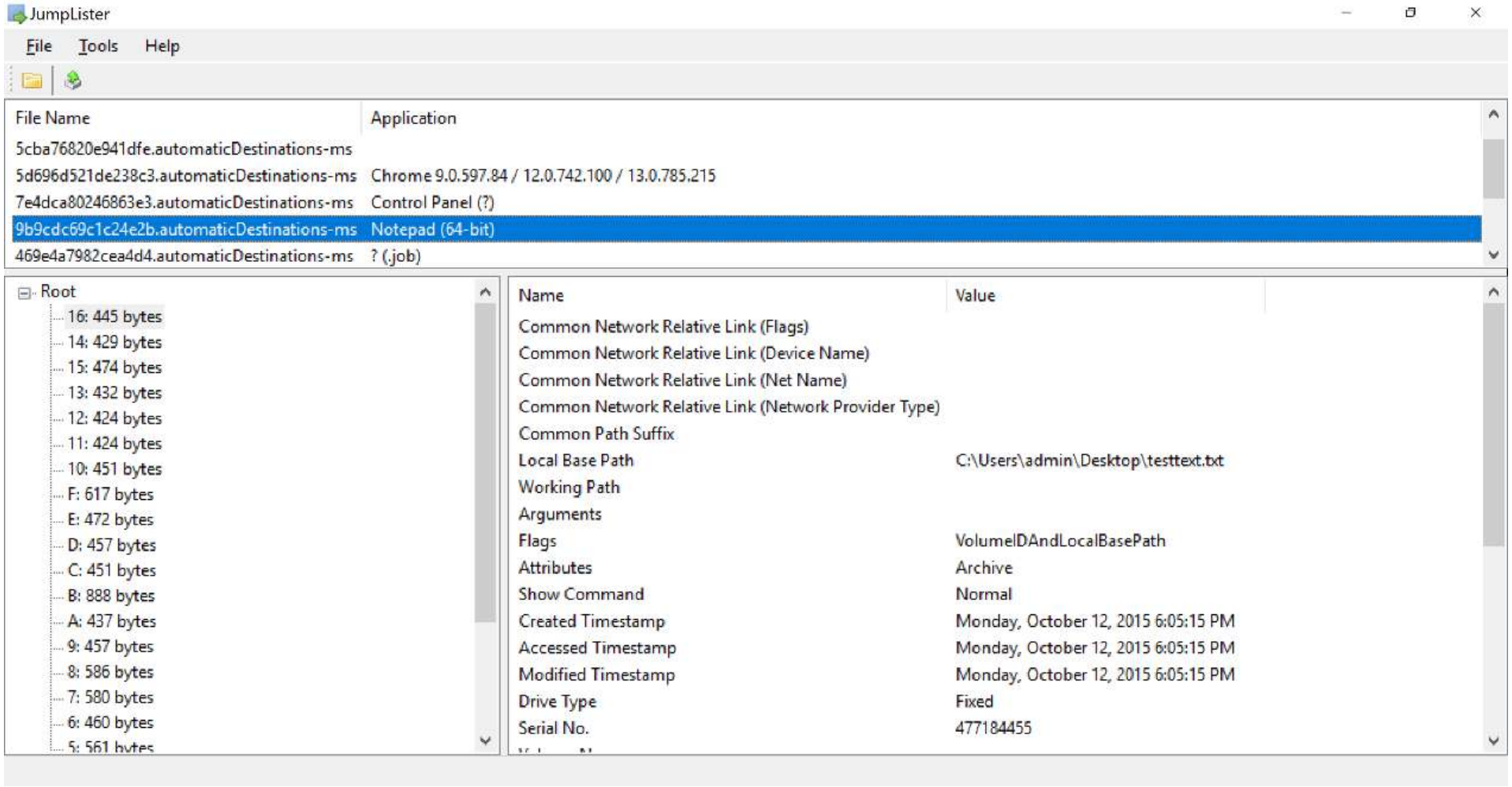
- **Jump lists**:
  - Introduced in Windows 7
  - Allows users to "**jump**"/access frequently or recently used items quickly
  - The lists are stored in the **jump list folders**:
    - %APPDATA%\Microsoft\Windows\Recent\AutomaticDestinations
    - %APPDATA%\Microsoft\Windows\Recent\CustomDestinations
  - Each has a unique file prepended with the **AppID** of the association app, and embedded with LNK files in **each stream**
- Jump list analysis tool: **JumpLister** (see Lab 6 Task 5)
- References:
  - https://forensics.wiki/jump_lists/
  - Singh & Singh, "*A forensic insight into Windows 10 Jump Lists*", Digital Investigation 17, 2016

Windows Artefacts: Jump Lists (On Task Bar)



Source: https://www.techrepublic.com/blog/windows-and-office/take-full-advantage-of-jump-lists-in-windows-7-with-these-tips/

JumpLister

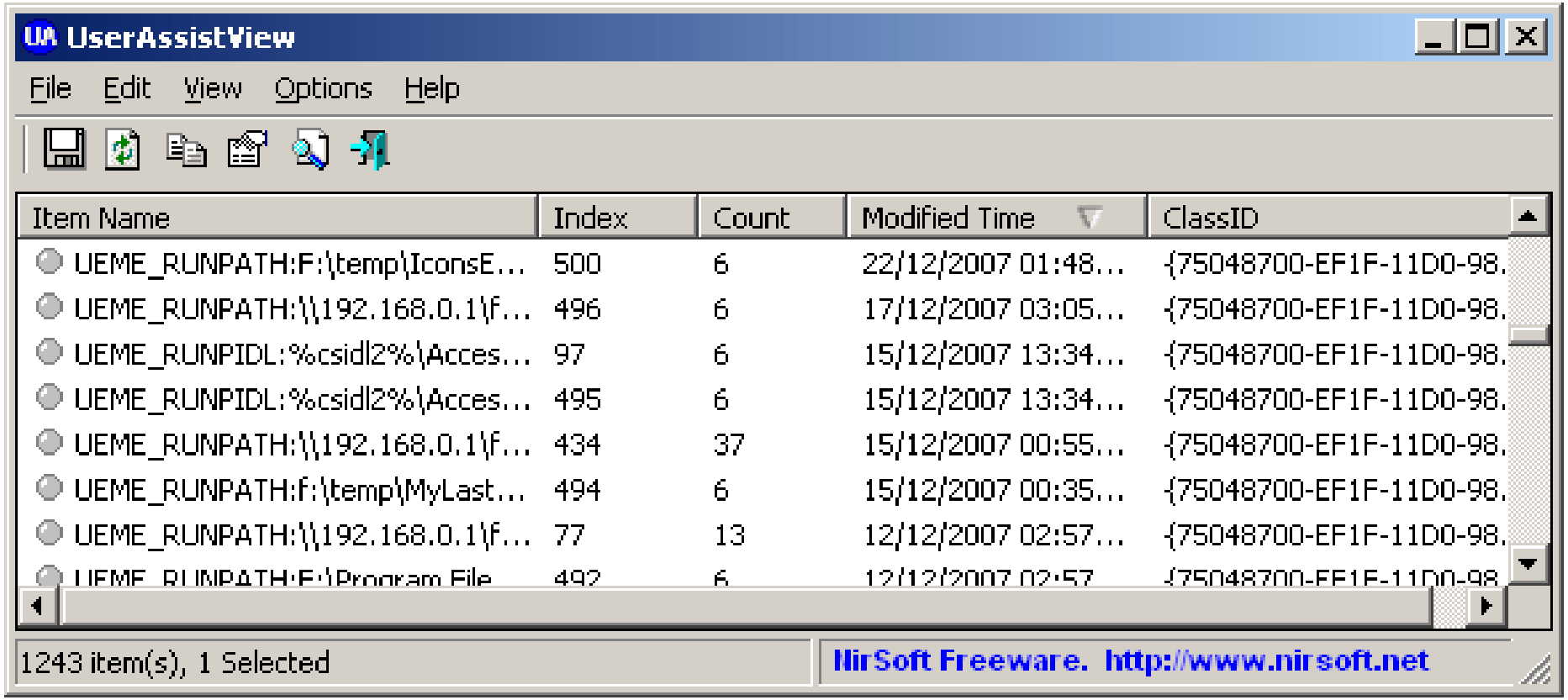

JumpLister
File Tools Help
File Name
Application
5cba76820e941dfe.automaticDestinations-ms
5d696d521de238c3.automaticDestinations-ms
Chrome 9.0.597.84 / 12.0.742.100 / 13.0.785.215
7e4dca80246863e3.automaticDestinations-ms
Control Panel (?)
9b9cdc69c1c24e2b.automaticDestinations-ms
Notepad (64-bit)
469e4a7982cea4d4.automaticDestinations-ms
? (.job)
Root
16: 445 bytes
14: 429 bytes
15: 474 bytes
13: 432 bytes
12: 424 bytes
11: 424 bytes
10: 451 bytes
F: 617 bytes
E: 472 bytes
D: 457 bytes
C: 451 bytes
B: 888 bytes
A: 437 bytes
9: 457 bytes
8: 586 bytes
7: 580 bytes
6: 460 bytes
5: 561 bytes
Name
Value
Common Network Relative Link (Flags)
Common Network Relative Link (Device Name)
Common Network Relative Link (Net Name)
Common Network Relative Link (Network Provider Type)
Common Path Suffix
Local Base Path
C:\Users\admin\Desktop\testtext.txt
Working Path
Arguments
Flags
VolumeIDAndLocalBasePath
Attributes
Archive
Show Command
Normal
Created Timestamp
Monday, October 12, 2015 6:05:15 PM
Accessed Timestamp
Monday, October 12, 2015 6:05:15 PM
Modified Timestamp
Monday, October 12, 2015 6:05:15 PM
Drive Type
Fixed
Serial No.
477184455

Windows Artefacts: User Assists

- Every **GUI-based programs** launched from the desktop are tracked in **UserAssist** registry keys
- The keys store **no of executions** and **last** execution date & time
- Locations:
  - HKEY_USERS\{SID}\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{GUID}\Count\
  - HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{GUID}\Count\
- Reference: https://www.aldeid.com/wiki/Windows-userassist-keys

UserAssistView Utility

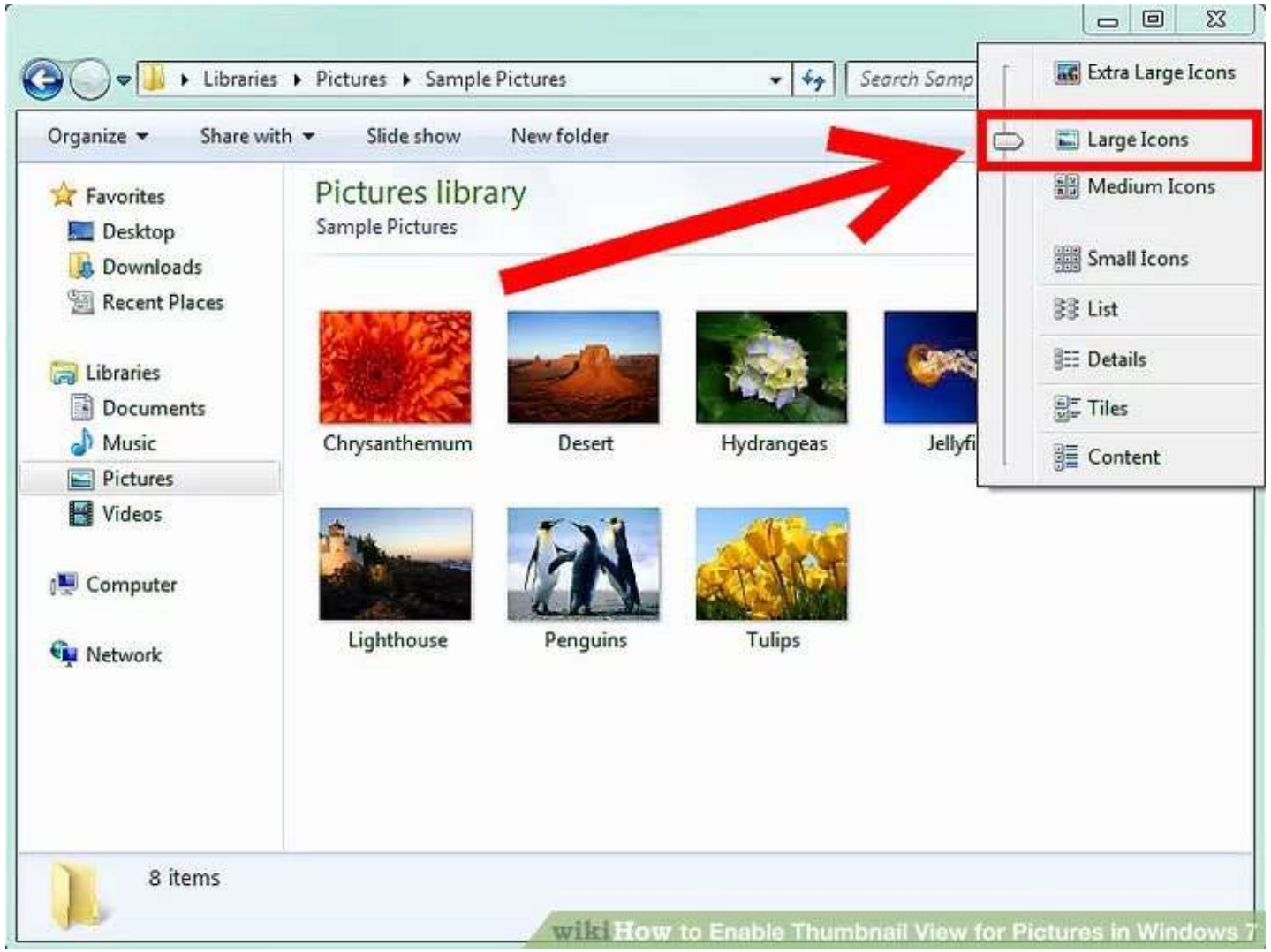
- See: https://www.nirsoft.net/utils/userassist_view.html



Windows Artefacts: Thumbnail Cache

- **Thumbnails**: reduced-size versions of pictures
- **Thumbnail cache**:
  - Thumbnail **database** of opened pictures, office documents and folders
  - **Each user** has her own database based on the thumbnail sizes viewed
  - Location: %USERPROFILE%\AppData\Local\Microsoft\Windows\Explorer
- Thumbnail cache analyzer tools:
  - MiTec Windows File Analyzer (**WFA**): see Lab 6 Task 6
  - OSForensics
- Reference: https://forensics.wiki/thumbs.db/

Windows Artefacts: Thumbnail Cache



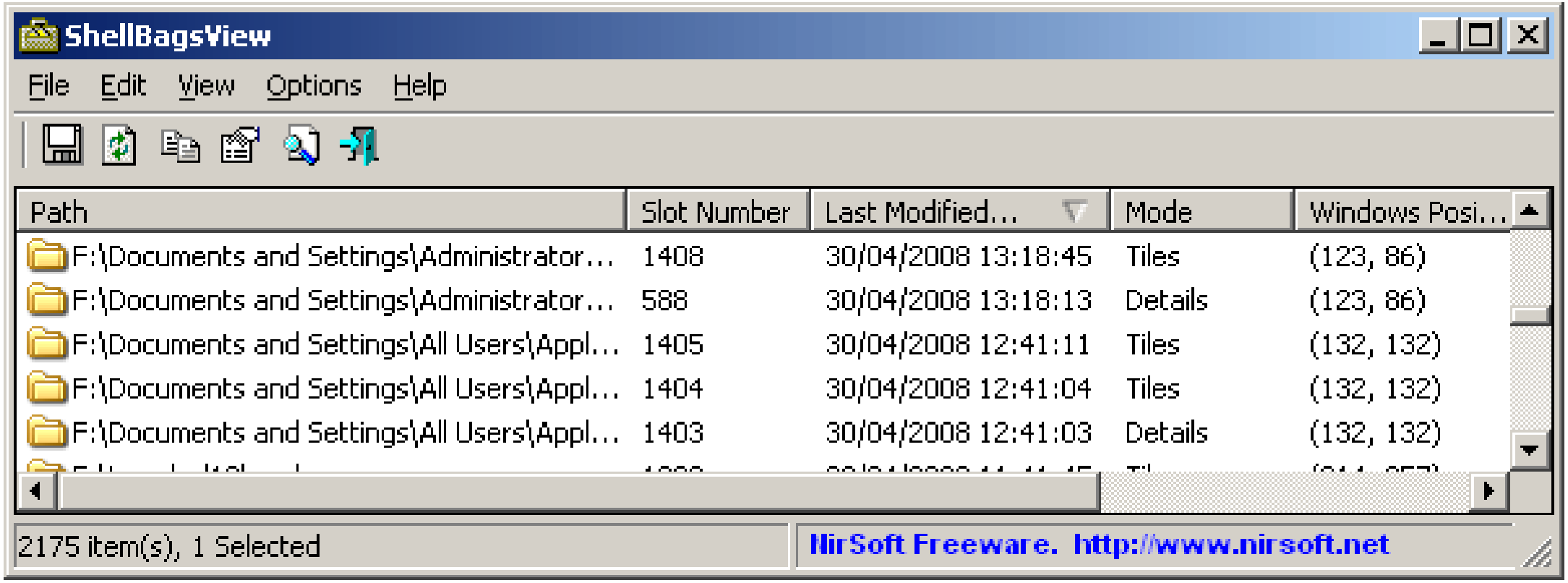
Source: https://www.wikihow.com/Enable-Thumbnail-View-for-Pictures-in-Windows-7

Shellbags

- **Shellbag registry keys**: store/remember **user settings** for GUI folders displayed within Windows Explorer
- Stored **data items**: visible columns, display mode (icons, details, list, etc.), sort order, etc.
- Usefulness to forensics: they indicate **previously opened folders**
- Locations:
  - HKEY_CURRENT_USER\Software\Microsoft\Windows\ShellNoRoam
  - HKEY_CURRENT_USER\Software\Microsoft\Windows\Shell
  - HKEY_CURRENT_USER\Software\Classes\Local Settings\Software\Microsoft\Windows\Shell (only in Windows Vista)
- Ref: https://www.sans.org/white-papers/34545/

ShellBagsView

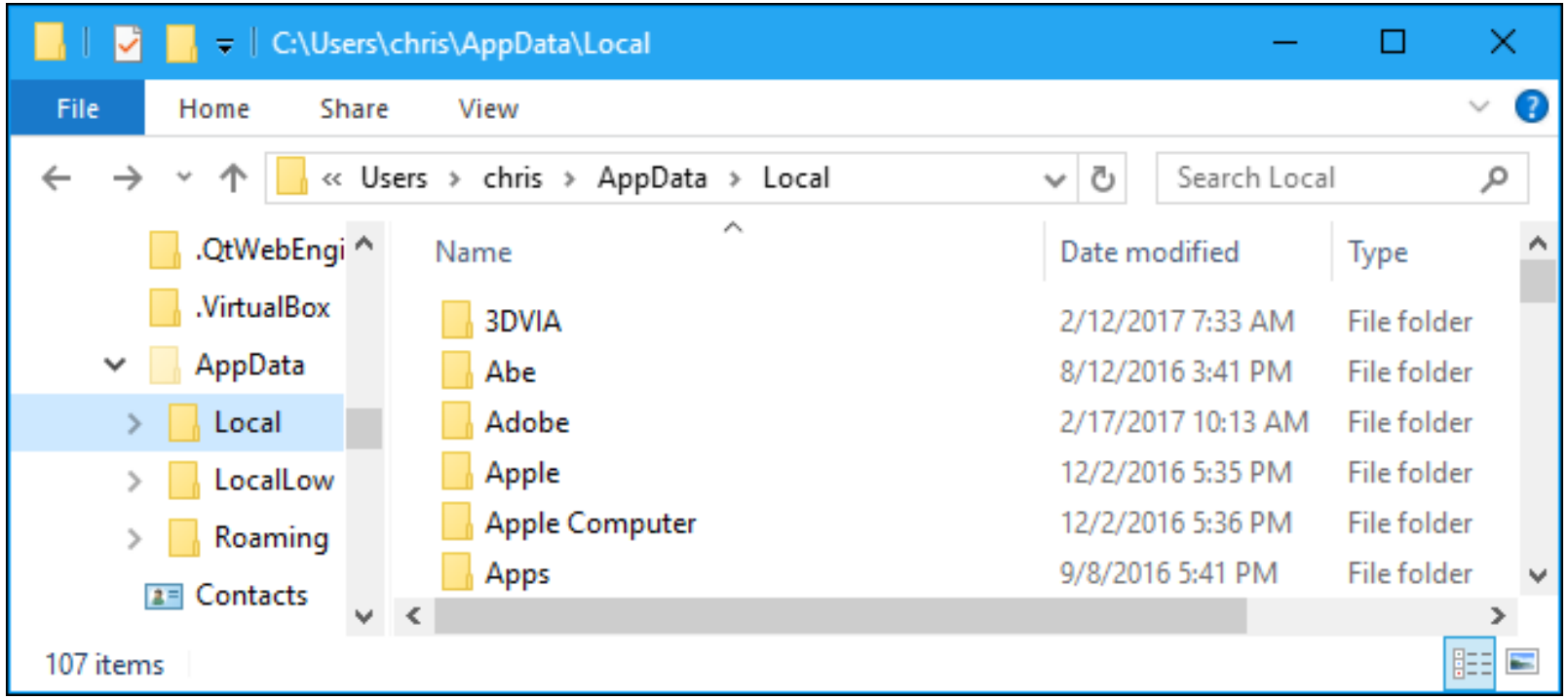
- See: https://www.nirsoft.net/utils/shell_bags_view.html



Other Windows Artefacts

- **Other** available Windows artefacts:
  See: https://resources.infosecinstitute.com/topic/understanding-critical-windows-artifacts-and-their-relevance-during-investigation-part-2/
- Also **regular files** stored inside the file system:
  - System files
  - User files:
    including the **AppData** folder: Local, LocalLow, Roaming *(see the next slide)*
  - Downloaded files
  - Deleted files: **recycle bin**, permanently-deleted files (use TSK/Autopsy)
- **Network & browser artefacts** on Windows machines:
  - *Will be covered in Lecture 7*

AppData Folder



From: https://www.howtogeek.com/318177/what-is-the-appdata-folder-in-windows/

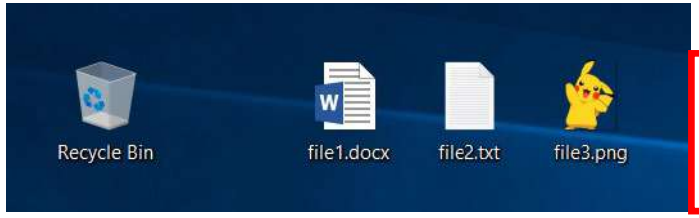
- For the folder-content explanation, see e.g.: https://www.howtogeek.com/318177/what-is-the-appdata-folder-in-windows/

Recycle Bin

- It is stored in the root of the OS drive (typically C:\), and called **$Recycle.Bin** in Windows 10
- When a file is deleted:
  - **Two files** are added into the user's folder: with 8-character file name starting with **$I** or **$R**, and followed by some **random six-character value**
  - **Examples**: C:\$Recycle.Bin\<*SID*>\$Ixxxxxx, C:\$Recycle.Bin\<*SID*>\$Rxxxxxx
  - **$I file**: contains the **meta-data** of the file, including: the original file name & path, the size of the file, and the time at which it was deleted
  - **$R file**: the actual **file content**
- References:
  - https://atalaysblog.wordpress.com/2019/03/19/recycle-bin-forensics/
  - https://df-stream.com/2016/04/fun-with-recycle-bin-i-files-windows-10/

Recycle Bin: Example

- From: https://atalaysblog.wordpress.com/2019/03/19/recycle-bin-forensics/
- Before the deletion:



Name	Original Location	Date Deleted	Size	Item type	Date modified
file1.docx	C:\Users\Guinness\Desktop	19 Mar 3:37 AM	12 KB	Microsoft Word Docu...	19 Mar 3:34 AM
file2.txt	C:\Users\Guinness\Desktop	19 Mar 3:37 AM	1 KB	Text Document	19 Mar 3:34 AM
file3.png	C:\Users\Guinness\Desktop	19 Mar 3:37 AM	43 KB	PNG File	19 Mar 3:35 AM

- After the deletion:

```
C:\>cd $Recycle.Bin

C:\$Recycle.Bin>dir /a
 Volume in drive C is OS
 Volume Serial Number is 4CD3-FB

 Directory of C:\$Recycle.Bin

16 Sep  11:21 AM    <DIR>          .
16 Sep  11:21 AM    <DIR>          ..
16 Sep  11:21 AM    <DIR>          S-1-5-18
19 Mar  03:37 AM    <DIR>          S-1-5-21-3954386123-3477195644-2237217235-1001
```

```
19 Mar  03:34 AM            11,861 $RJIETQY.docx
19 Mar  03:34 AM                13 $R7GLYIA.txt
19 Mar  03:35 AM            43,697 $RSJTV6E.png
19 Mar  03:37 AM               102 $IJIETQY.docx
19 Mar  03:37 AM               100 $I7GLYIA.txt
19 Mar  03:37 AM               100 $ISJTV6E.png
              92 File(s)         68,419 bytes
               2 Dir(s)  397,528,186,880 bytes free

C:\$Recycle.Bin\S-1-5-21-3954386123-3477195644-2237217235-1001>
```

- File copying & viewing:

```
C:\$Recycle.Bin\S-1-5-21-3954386123-3477195644-2237217235-1001>copy $RSJTV6E.png "C:\Users\Guinness\RecFor\"
        1 file(s) copied.
```

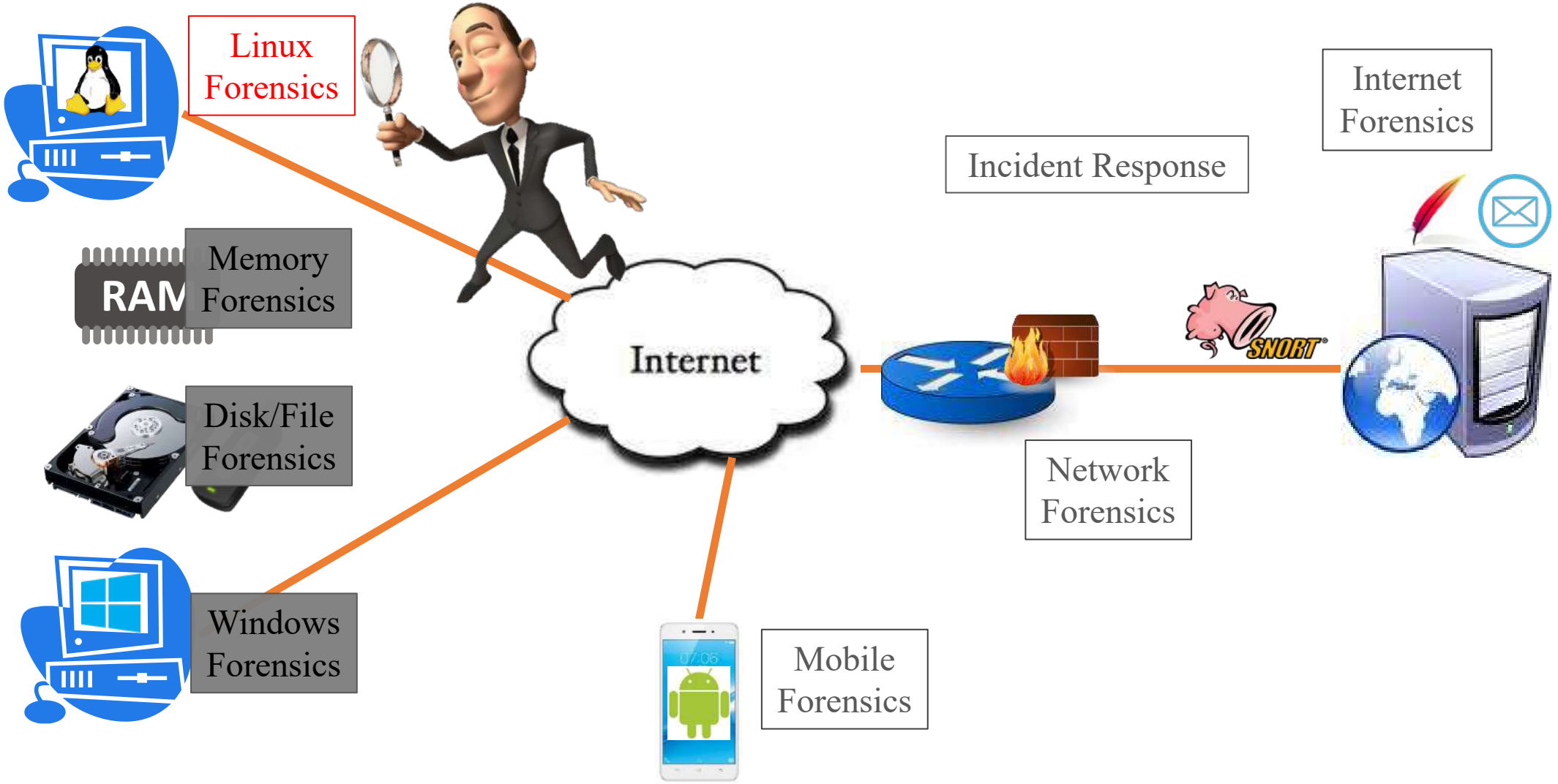
$RSJTV6E.png

Additional Resources on Windows Forensics

- Additional Windows forensics **references**:
  - Harlan Carvey, "*Windows Forensic Analysis Toolkit*", 4th Edition, 2014.
  - Poster: https://www.sans.org/security-resources/posters/windows-forensics-evidence-of/75/download

- Additional collection of **tools**:
  - https://ericzimmerman.github.io/#!index.md

This Lecture's Focus

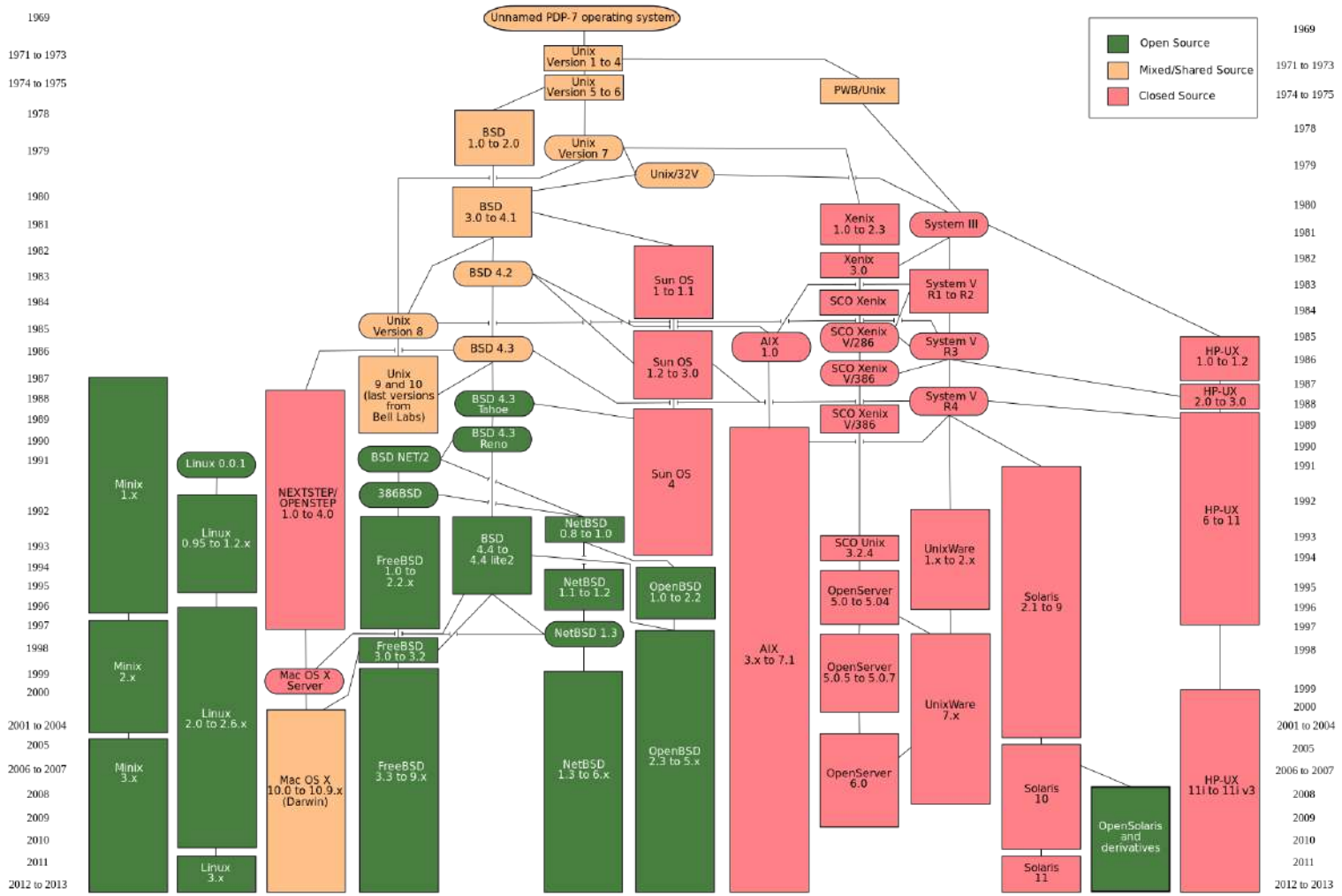


Linux/UNIX Forensics

Linux & UNIX: History Recap

- History from 1970s
- Various **versions** (Linux, **Android**, OSX + iOS, Solaris, AIX, …)
- We will focus only on **Linux**:
  - Open source (http://www.kernel.org)
  - Widely used for servers and desktops as well
  - Many distributions, which vary in setup, administration, kernel, …
  - Many tools (usually also open source)

Simplified UNIX Family Tree



Why People Use Linux/UNIX?

UNIX: A beautiful but strange beast

```
$ find . –name “abc” –exec rm {} \;
```

Unix philosophy: "***Swiss Army Knife***"

Linux Partition and Files

Partition & File System: Hierarchical Structure

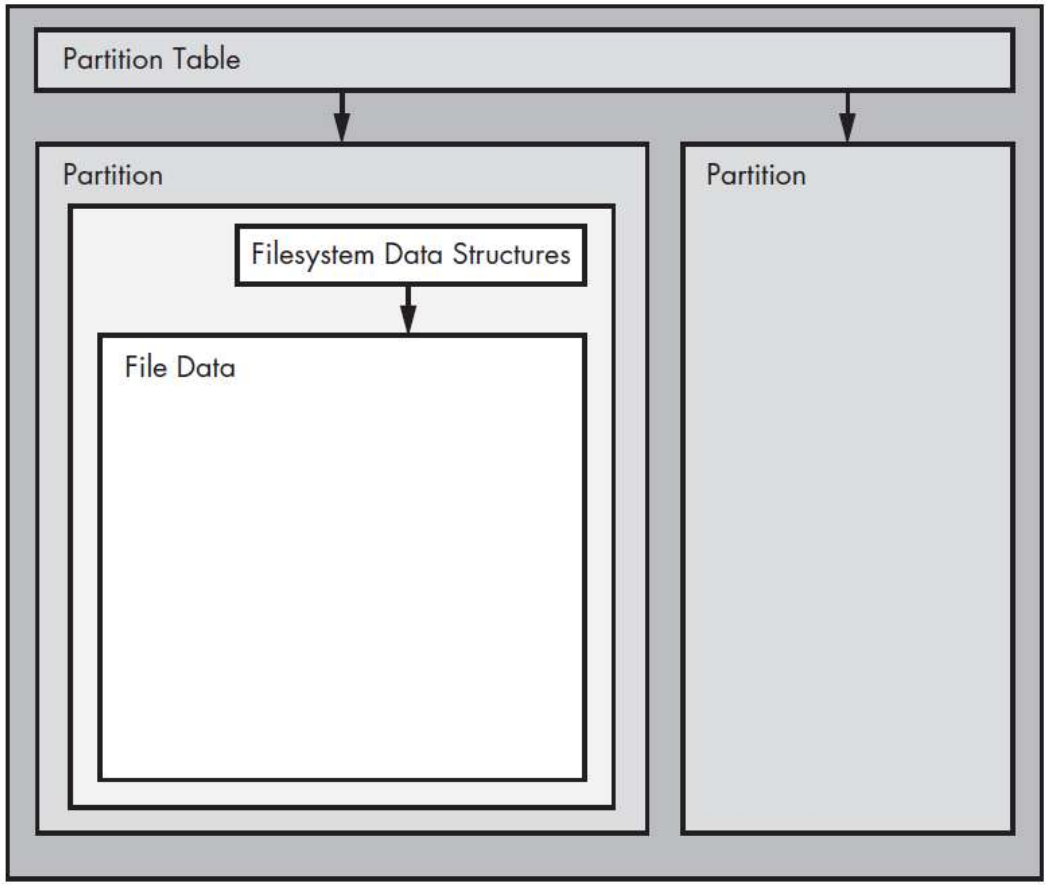


Figure 4-1: Typical Linux disk schematic

From: Ward, "How Linux Works", 2nd edition, 2014

Partition & File System: Kernel Access to Disk

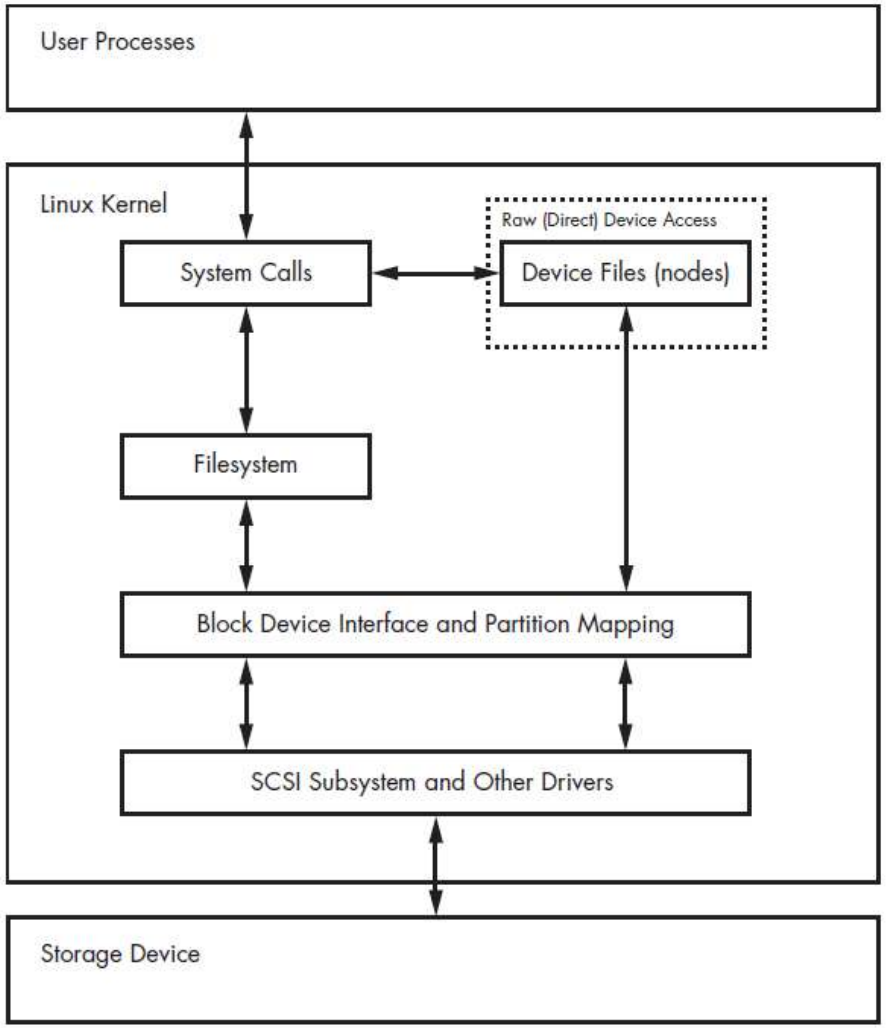


Figure 4-2: Kernel schematic for disk access

From: Ward, "How Linux Works", 2nd edition, 2014

Linux Disk & Partitions: *Recap*

- Application of UNIX philosophy: storage device (e.g. disk) as a **file**
- Files for connected SCSI/SATA disks:
  - **`/dev/sd`*`xn`***
  - *x* = a, b, c, … : (physical) drive letter
  - *n* = 1, 2, 3, … : partition number
- The file-naming scheme is used by **Linux commands** and **forensic tools** that need to refer to disks

Partition Analysis

- Some tools: fdisk, **parted**, gparted

```
# parted -l
Model: ATA WDC WD3200AAJS-2 (scsi)
Disk /dev/sda: 320GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End     Size    Type      File system     Flags
 1      1049kB  316GB   316GB   primary   ext4            boot
 2      316GB   320GB   4235MB  extended
 5      316GB   320GB   4235MB  logical   linux-swap(v1)

Model: FLASH Drive UT_USB20 (scsi)
Disk /dev/sdf: 4041MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt

Number  Start   End     Size    File system  Name        Flags
 1      17.4kB  1000MB  1000MB               myfirst
 2      1000MB  4040MB  3040MB               mysecond
```

From: Ward, "How Linux Works", 2nd edition, 2014

File System: Types & Construction

- Linux **file system types**:
  - ***Second Extended filesystem*** (**ext2**):
    inspired by traditional Unix filesystems,
    e.g. the Unix File System (UFS) and the Fast File System (FFS)
  - ***Third Extended filesystem*** (**ext3**):
    added a **journaling** feature to enhance data integrity and hasten booting
  - ***Fourth Extended filesystem*** (**ext4**): the latest
- Creating Linux file system:
  - **`mkfs*`**: can be linked to other executable files
  - Similar to `format` in Windows environment

File System Construction: Commands

- Format a partition using **`mkfs`**, such as:
  `# mkfs.ext4 /dev/sda4`

```
$ ls -l /sbin/mkfs.*
-rwxr-xr-x 1 root root 17896 Mar 29 21:49 /sbin/mkfs.bfs
-rwxr-xr-x 1 root root 30280 Mar 29 21:49 /sbin/mkfs.cramfs
lrwxrwxrwx 1 root root     6 Mar 30 13:25 /sbin/mkfs.ext2 -> mke2fs
lrwxrwxrwx 1 root root     6 Mar 30 13:25 /sbin/mkfs.ext3 -> mke2fs
lrwxrwxrwx 1 root root     6 Mar 30 13:25 /sbin/mkfs.ext4 -> mke2fs
lrwxrwxrwx 1 root root     6 Mar 30 13:25 /sbin/mkfs.ext4dev -> mke2fs
-rwxr-xr-x 1 root root 26200 Mar 29 21:49 /sbin/mkfs.minix
lrwxrwxrwx 1 root root     7 Dec 19  2011 /sbin/mkfs.msdos -> mkdosfs
lrwxrwxrwx 1 root root     6 Mar  5  2012 /sbin/mkfs.ntfs -> mkntfs
lrwxrwxrwx 1 root root     7 Dec 19  2011 /sbin/mkfs.vfat -> mkdosfs
```

From: Ward, "How Linux Works", 2nd edition, 2014

Partition Mounting: File System Hierarchy

Table 5.1 Standard Linux Directories	
/bin	essential command binaries (for all users)
/boot	files needed for the system bootloader
/dev	device files
/etc	system configuration files
/home	user home directories
/lib	essential shared libraries and kernel modules
/media	mount points for removable media (usually for automounts)
/mnt	temporary mount points (usually mounted manually)
/opt	add-on application packages (outside of system package manager)
/root	root user's home directory
/sbin	system binaries
/tmp	temporary files

Filesystem Hierarchy Standard (FHS)

- ***Filesystem Hierarchy Standard (FHS)*** from the Linux Foundation
- Defines the directory structure and directory contents in Linux distributions
- Standard file system structure only **by convention**:
  - Many distributions declare a policy to maintain FHS compliance
  - Yet, everything can just be changed!
- See: https://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard

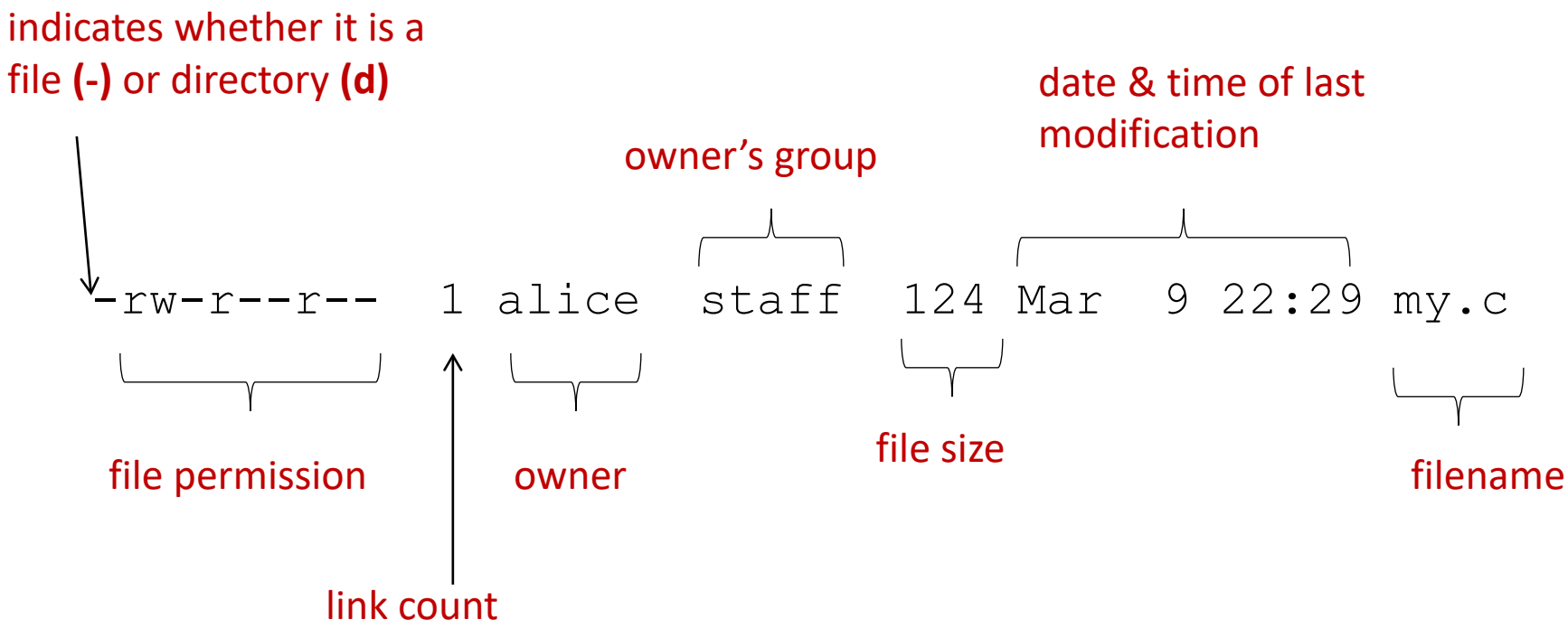
File Types in Linux/UNIX

- **Regular file**: normal ASCII/binary data (including executables)
- **Directories**: special file to map filenames to file blocks
  (see also implementation of ***hard* links**)
- **Special files**:
  - **Devices**: block devices (e.g. hard disks), character devices (e.g. keyboards)
  - **Symbolic links**: a ***soft link***
  - **Named pipes (FIFO)**: for uni-directional inter-process communication
  - **Unix socket**: for bi-directional inter-process communication
- Other non-native file types:
  e.g. DOS, CDROMs, NFS (network file system)
- proc: **special/pseudo file-system** like interface to kernel internals

Linux File Attributes

- Every file has:
  - **Size:** in bytes
  - **User ID** & **group ID**
  - **Link count:** the number of hard links
  - **Access permissions**
    - rwx: for owner | group | other users
    - setuid, setgid, sticky bits
  - **MAC times**:
    - **Modification** time: last **data** modification
    - **Access** time: most recent access to **data**
    - **Change** time: last modification of ***attributes***

Linux File Attributes: File/Directory Listing

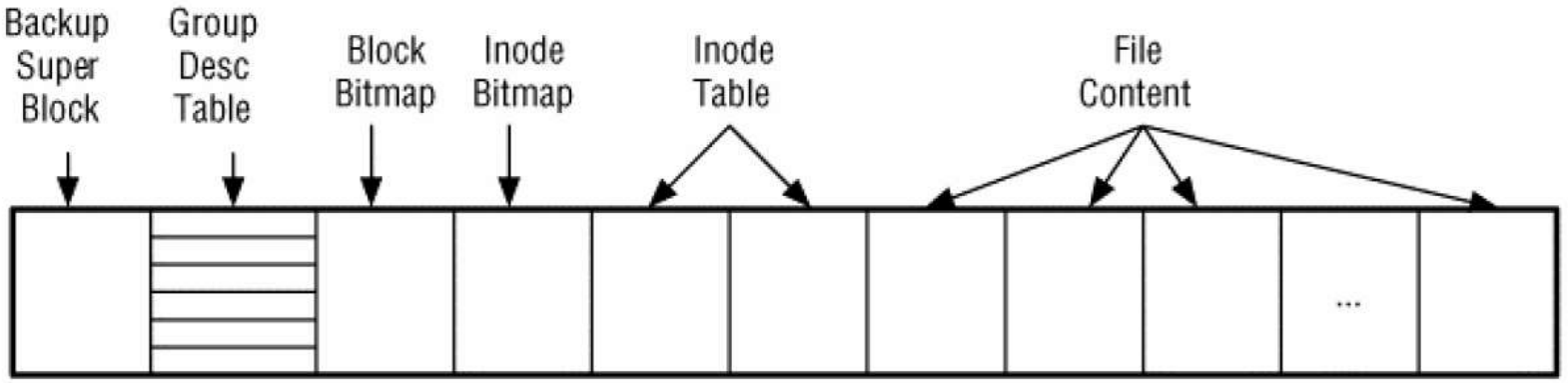


Break!

Linux Ext File System

File System Layout

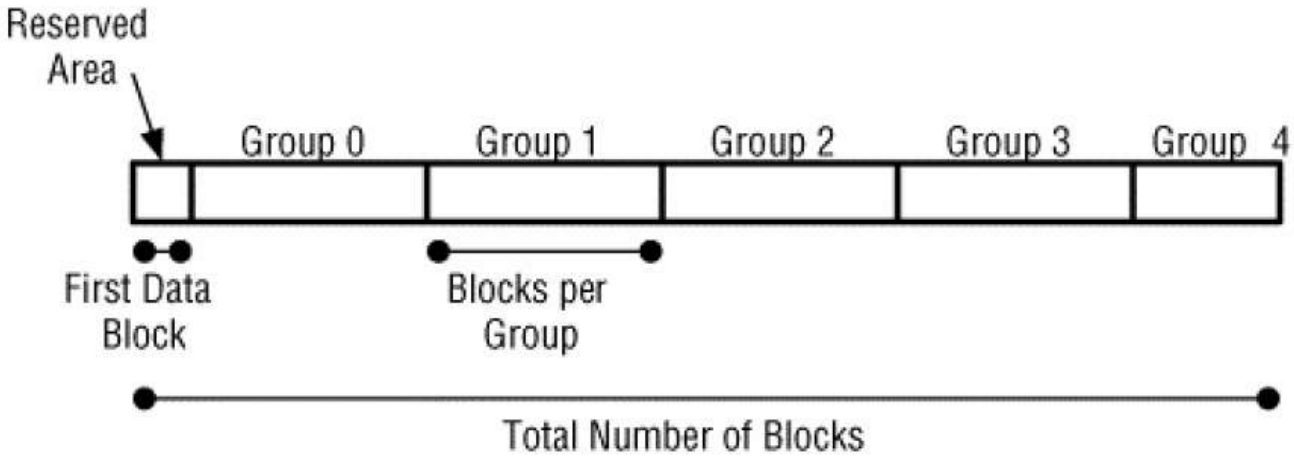
- Disk space is split into **data/content blocks**
- **A *data/content block*:**
  - Corresponds to 1 or more disk sectors
  - Similar to a **disk cluster** in FAT file system
- Data blocks are grouped into ***block group***: *described soon*



From: Brian Carrier, "File System Forensic Analysis"

File System Layout

- High-level layout of a file system: ***superblock*** and ***block groups***

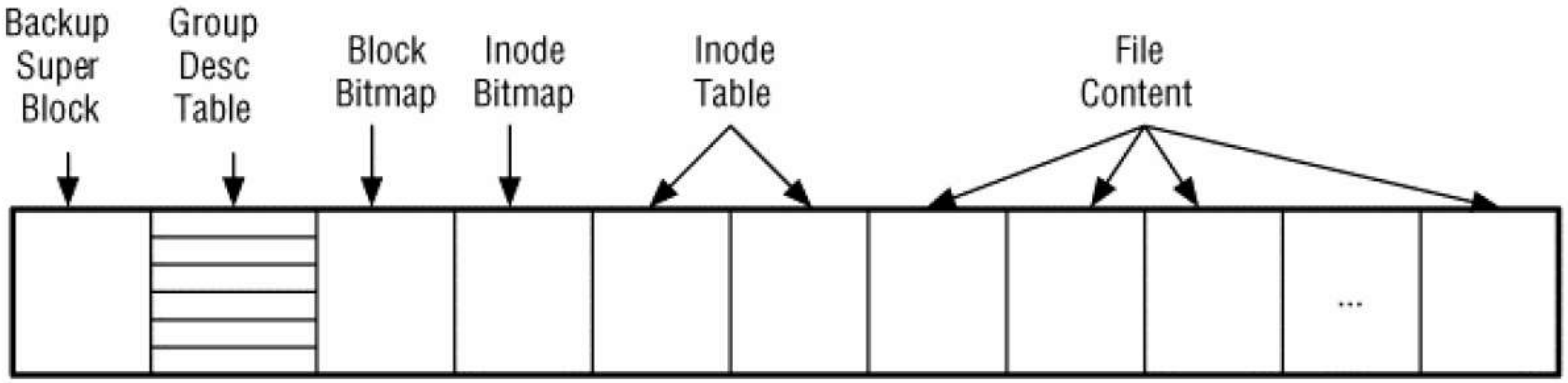


From: Brian Carrier, "File System Forensic Analysis"

- ***Superblock***:
  - **Size**: 1,024 bytes; location: offset 1,024 bytes (after the boot "*block*")
  - Contains **FS basic info**: block size, no of blocks/inodes, volume name, …
  - Backup copies: typically stored in the first block of each block group

Block Group

- A ***block group*** contains:
  - **Backup superblock**: for redundancy
  - **Group description table**: describes its block group
  - **Block bitmap:** keeps track of the allocation status of its content blocks
  - **Inode bitmap**: keeps track of the allocation status of its inodes
  - **Inode table**: an array of inodes of this data block



From: Brian Carrier, "File System Forensic Analysis"

File System Layout

- TSK command for checking file system structure:

```
# fsstat -f linux-ext3 ext3.dd
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: Ext3
Volume Name:
Volume ID: e4636f48c4ec85946e489517a5067a07

Last Written at: Wed Aug 4 09:41:13 2004
Last Checked at: Thu Jun 12 10:35:54 2003
```

From: Brian Carrier, “File System Forensic Analysis”

File System Layout

- TSK command for checking file system structure (continued):

```
Last Mounted at: Wed Aug 4 09:41:13 2004
Unmounted properly
Last mounted on:

Source OS: Linux
Dynamic Structure
Compat Features: Journal,
InCompat Features: Filetype, Needs Recovery,
Read Only Compat Features: Sparse Super, Has Large Files,

Journal ID: 00
Journal Inode: 8

METADATA INFORMATION
--------------------------------------------
Inode Range: 1 - 1921984
Root Directory: 2
Free Inodes: 1917115

CONTENT INFORMATION
--------------------------------------------
Block Range: 0 - 3841534
Block Size: 4096
Free Blocks: 663631
```

From: Brian Carrier, "File System Forensic Analysis"

File System Layout

- TSK command for checking file system structure (continued):

```
BLOCK GROUP INFORMATION
--------------------------------------------
Number of Block Groups: 118
Inodes per group: 16288
Blocks per group: 32768

Group: 0:
  Inode Range: 1 - 16288
  Block Range: 0 - 32767
  Layout:
    Super Block: 0 - 0
    Group Descriptor Table: 1 - 1
    Data bitmap: 2 - 2
    Inode bitmap: 3 - 3
    Inode Table: 4 - 512
    Data Blocks: 513 - 32767
  Free Inodes: 16245 (99%)
  Free Blocks: 0 (0%)
  Total Directories: 11

Group: 1:
  Inode Range: 16289 - 32576
  Block Range: 32768 - 65535
[REMOVED]
```

From: Brian Carrier, "File System Forensic Analysis"

More on Data/Content Block

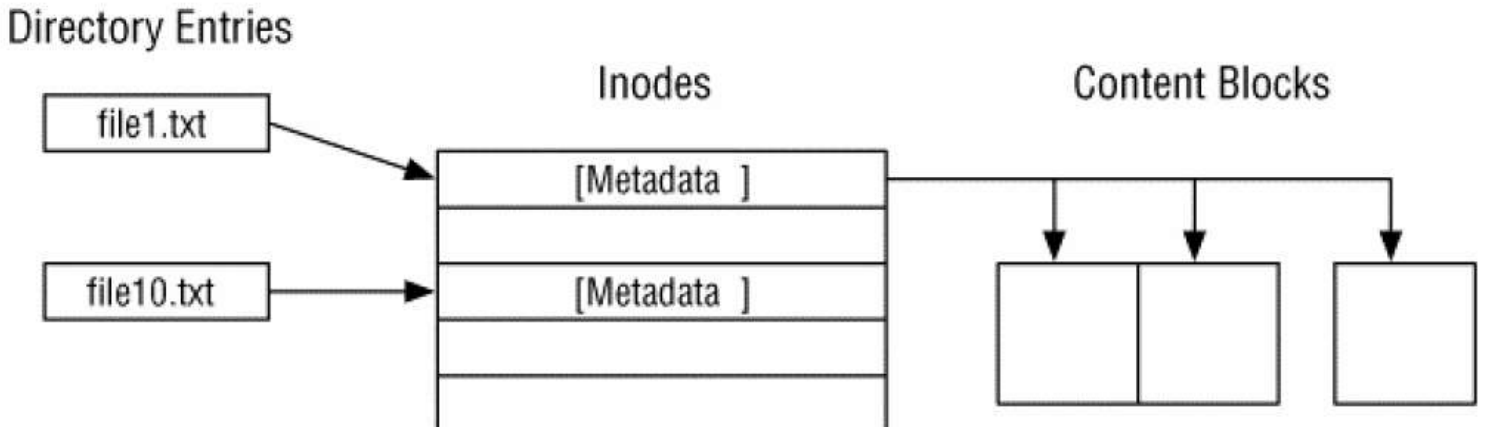
- **Block size**: 1,024; 2,048; or 4,096 bytes; given in the superblock
- All blocks are given an **address**:
  - Address starts with 0
  - Block 0 is located in the first sector of the *file system*
- A block's **allocation status**: kept by the group's block ***bitmap***
- The TSK's `dstat` tool: shows the **allocation status** of a block and the **block group** to which it belongs

```
# dstat -f linux-ext3 ext3.dd 14380

Block: 14380
Allocated
Group: 0
```

From: Brian Carrier, "File System Forensic Analysis"

File System Structure

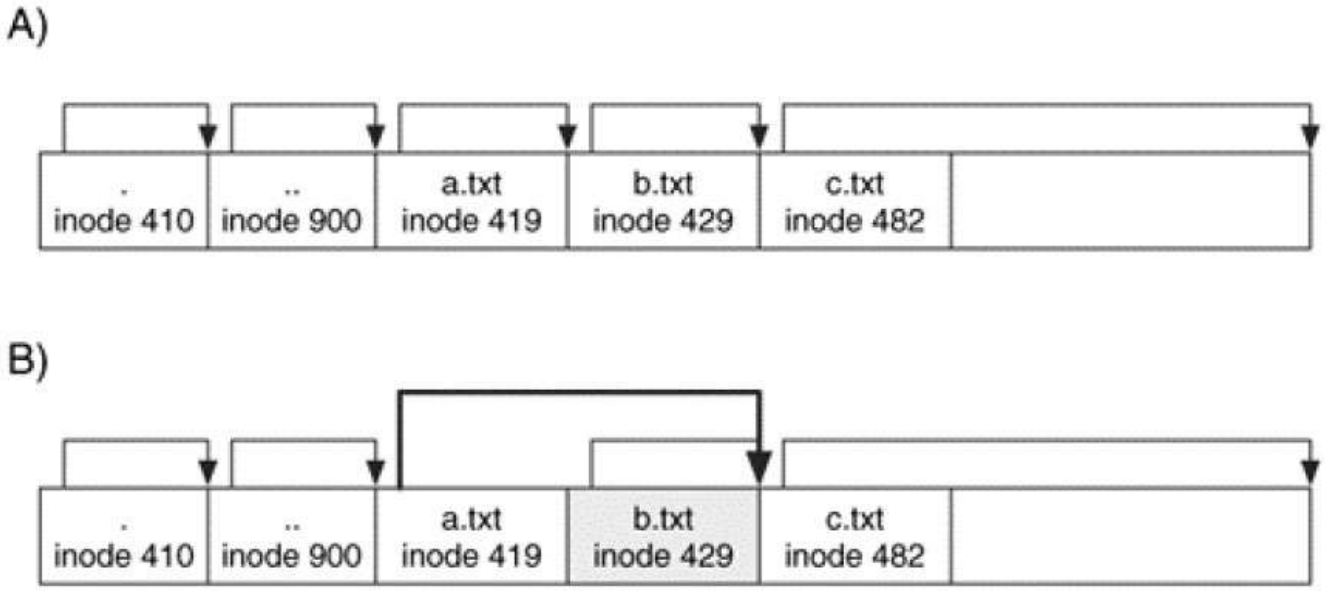
- File system structures for a file/sub-directory:
  - ***Directory entries***: inside a content block for a **directory**
  - ***Inodes***
  - ***Content blocks***
- Relationship among them:



From: Brian Carrier, "File System Forensic Analysis"

Directory

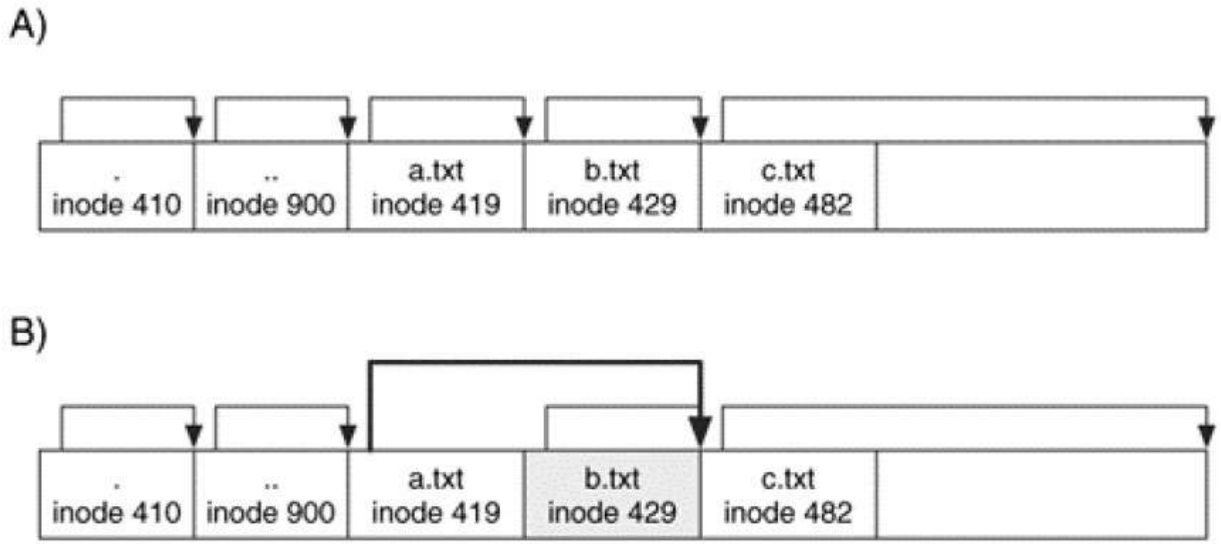
- A ***directory***:
  - Just like a regular file but with a **special type value** in its inode
  - Its **data blocks** store:
    a **linked list** of ***directory entries*** for all files/sub-directories within it, including the '.' and '..' directories



From: Brian Carrier, "File System Forensic Analysis"

Directory Entries

- Each ***directory entry*** contains:
  - File/sub-directory **name**: from 1 to 255
  - **Length** of the file/sub-directory name
  - **Type**: file, sub-directory, or other special file type
  - **Inode number** for the file/sub-directory
  - **Size** (i.e. **record length**): for *locating the next entry*



From: Brian Carrier, "File System Forensic Analysis"

Directory Entries

- TSK command for checking directory entries: fls

```
# fls -f linux-ext3 -a ext3.dd 69457
d/d 69457:      .
d/d 53248:      ..
r/r 69458:      abcdefg.txt
r/r * 69459:    file two.dat
d/d 69460:      subdir1
r/r 69461:      RSTUVWXY
```

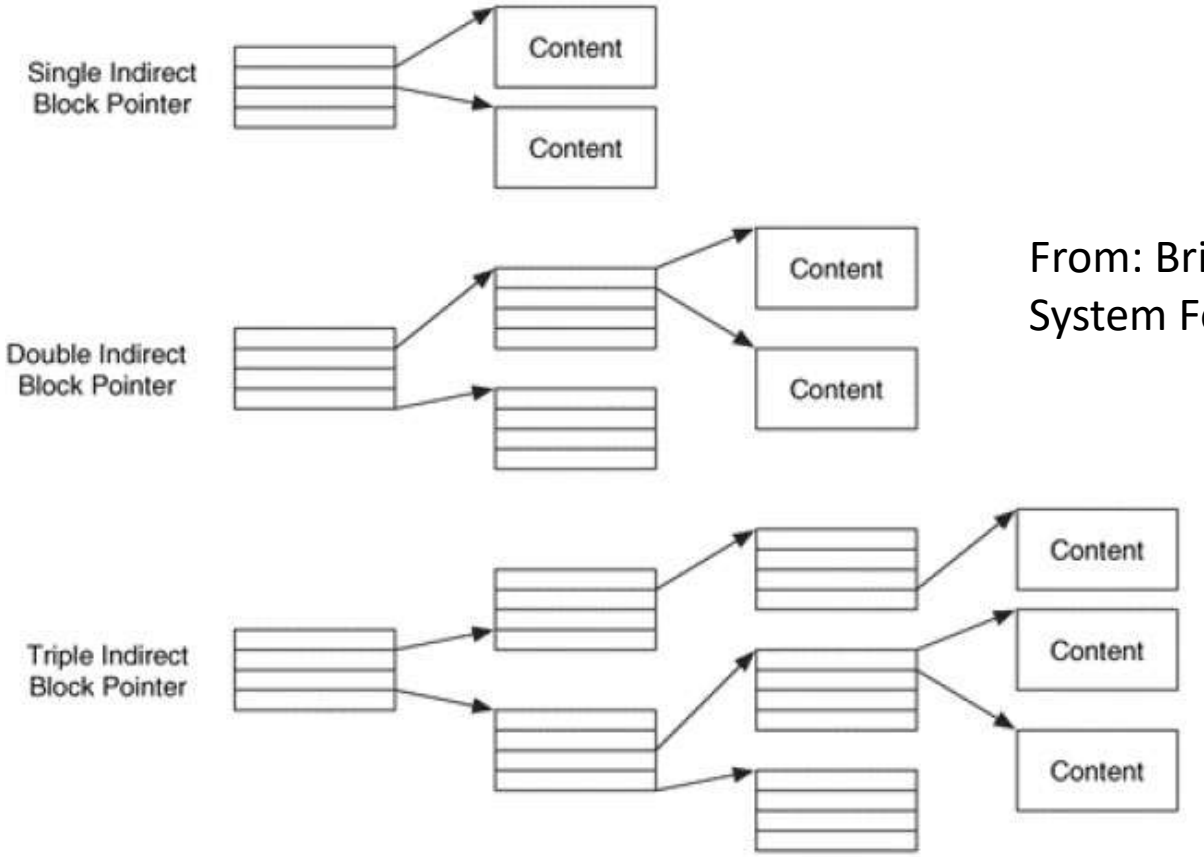
From: Brian Carrier, “File System Forensic Analysis”

Inode

- An ***inode*** contains:
  - The metadata of a file or directory
  - Points to the **content block(s)** of the file or directory using block pointers
- Inode detailed **fields**:
  - **Mode**: file type (regular, directory, special, etc.), basic permission values
  - **Owner info**: user ID, group ID
  - **File size**
  - **Timestamp**: times for the last modification, access, change and deletion; stored as the **number of seconds** since January 1, 1970 UTC
  - **Link/reference count**: no of file names that point to the inode
  - **Data block pointers**
- Supporting ***inode bitmap***:
  keeps track of allocated and unallocated inodes

Inode's 15 Block Pointers

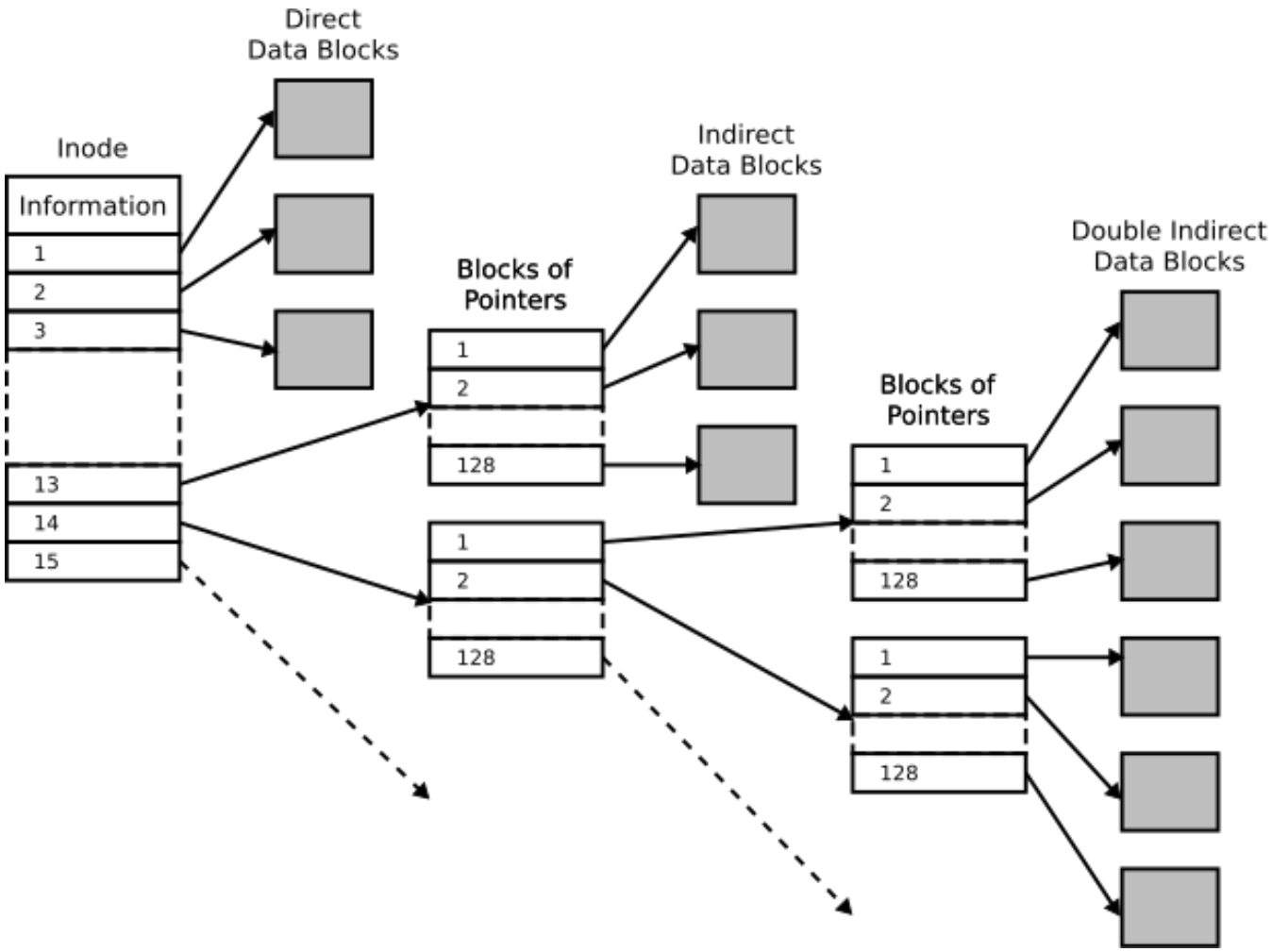
- Inodes and content blocks: 12 **direct**; 1 **single**, 1 **double**, and 1 **triple indirect block pointers**
- *Why direct and indirect (multi-level) indexing?*
  - Fast access to **small files**
  - Flexibility in handling **huge files**



From: Brian Carrier, "File System Forensic Analysis"

Ref: https://en.wikipedia.org/wiki/Inode_pointer_structure

Inode's 15 Block Pointers: Illustration



From: Wikipedia

File System Structure: Inode

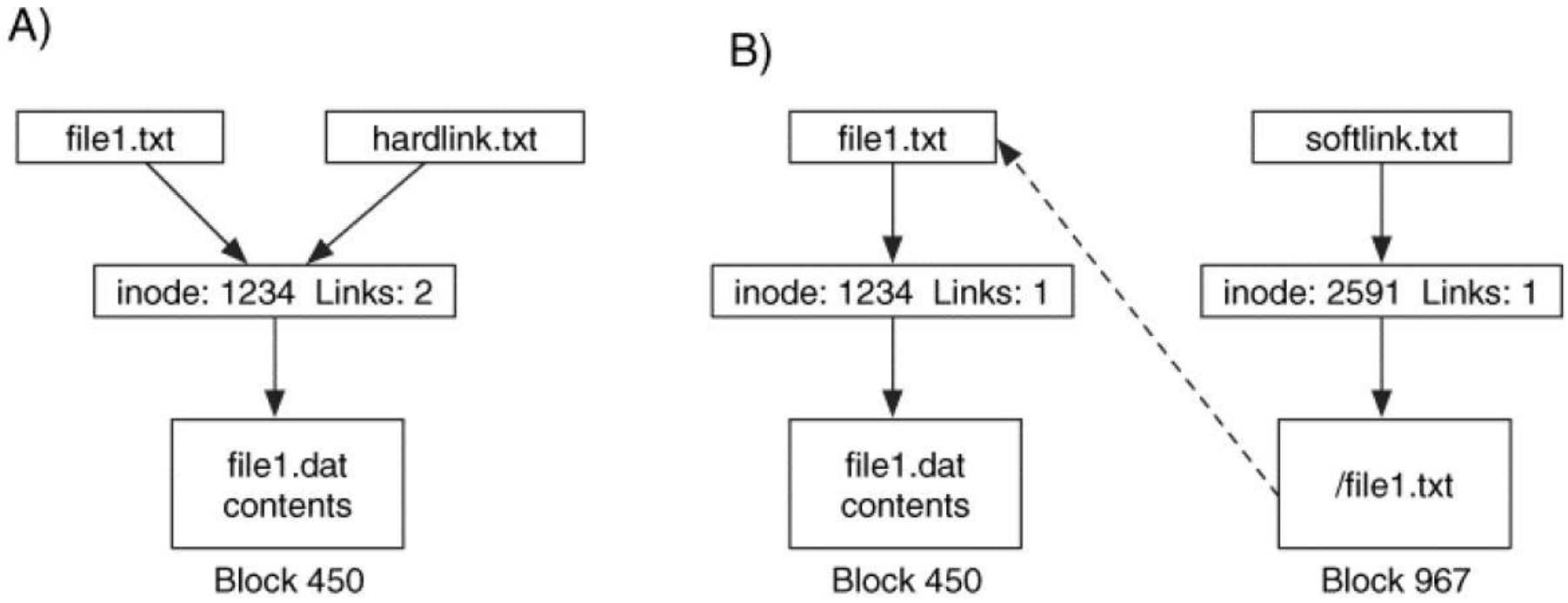
- TSK command for checking inode: istat

```
# istat -f linux-ext3 ext3.dd 16
inode: 16
Allocated
Group: 0
Generation Id: 199922874
uid / gid: 500 / 500
mode: -rw-r--r--
size: 10240000
num of links: 1

Inode Times:
Accessed:        Fri Aug  1 06:32:13 2003
File Modified:   Fri Aug  1 06:24:01 2003
Inode Modified:  Fri Aug  1 06:25:58 2003

Direct Blocks:
14380 14381 14382 14383 14384 14385 14386 14387
14388 14389 14390 14391 14393 14394 14395 14396
[REMOVED]
16880 16881 16882 16883

Indirect Blocks:
14392 15417 15418 16443
```

From: Brian Carrier, “File System Forensic Analysis”

Hard and Soft/Symbolic Links

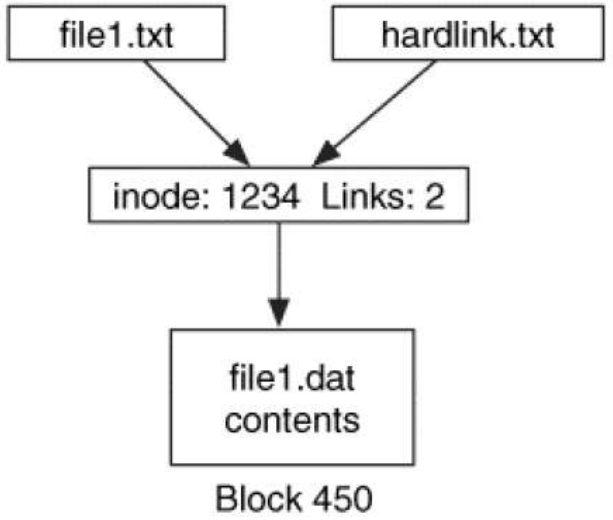
- Hard links vs soft links:



From: Brian Carrier, "File System Forensic Analysis"

Hard Links

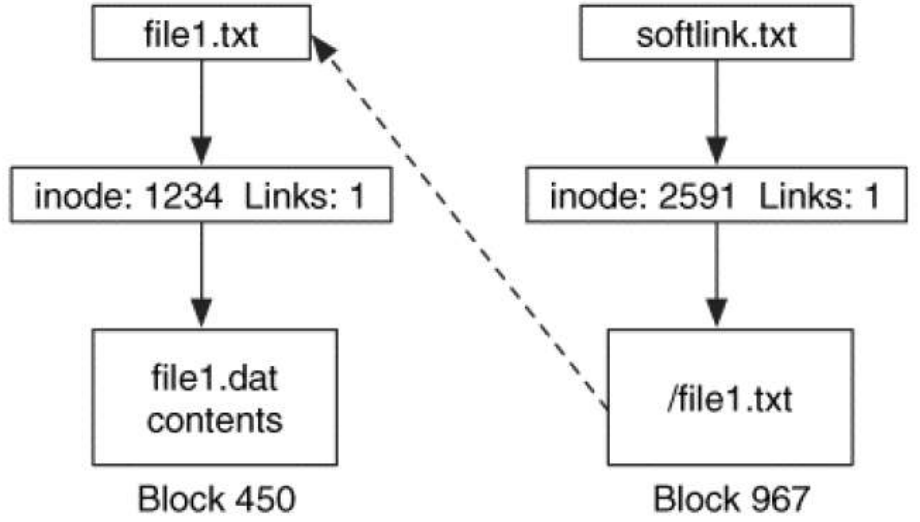
- A ***hard link***:
  - An additional file name for a file/directory in **the same file system**
  - Creates **a new directory entry** that points it to the original inode
  - The **link count** in the inode is incremented by one
  - A file will not be deleted until **all its hard links** are deleted
  - After a hard link creation, we can't tell the original name & the link



From: Brian Carrier, "File System Forensic Analysis"

Soft/Symbolic Links

- A ***soft link***:
  - Is also a second name for a file or directory
  - Creates **a new file** that contains the pathname of the original file/directory
  - The pathname is **stored** in either:
    - blocks allocated to the file
    - the inode: if the pathname is less than 60 characters long



From: Brian Carrier, "File System Forensic Analysis"

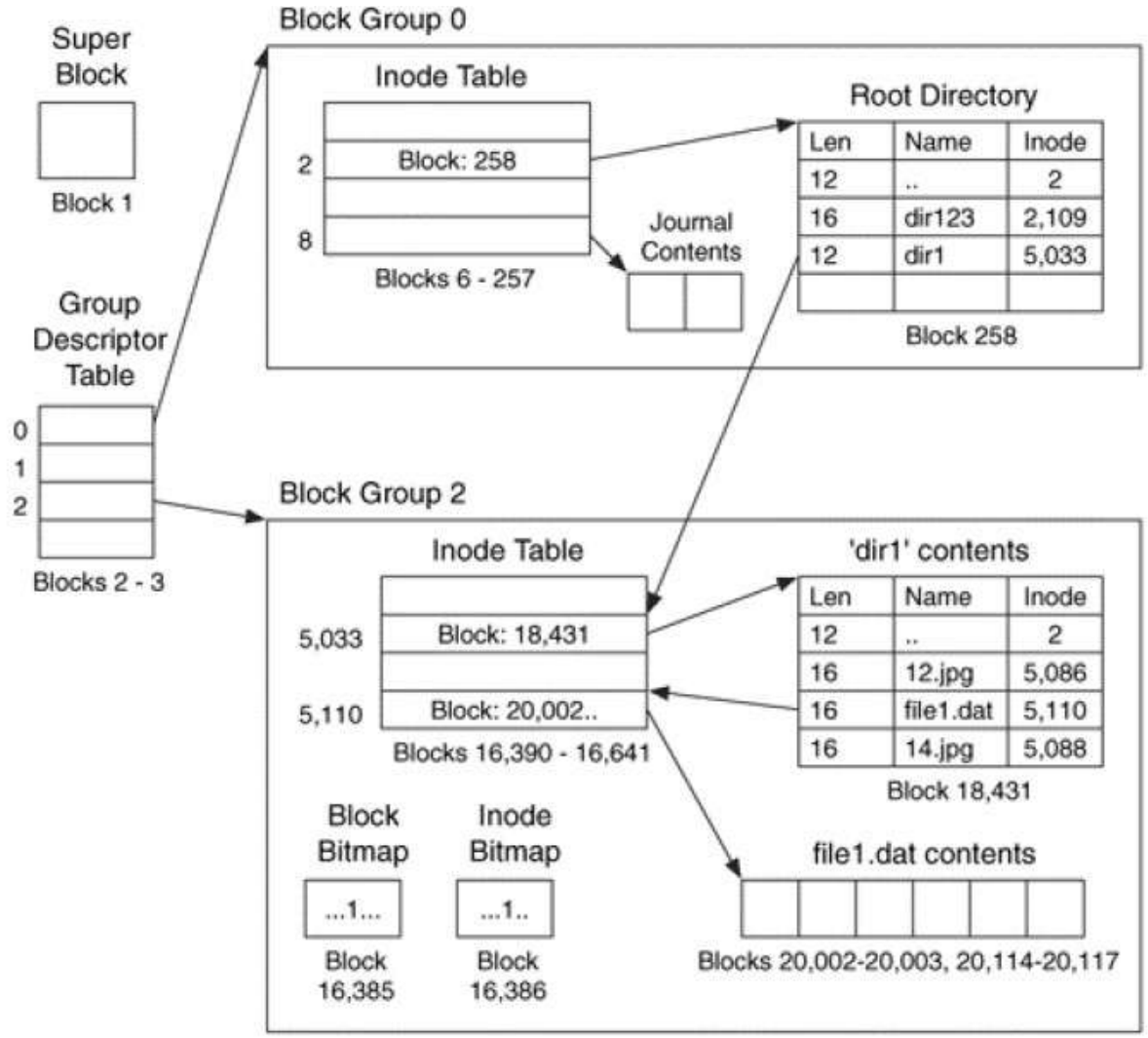
Soft/Symbolic Links

- Notice that only a pathname is stored for a soft link
- **Good** feature:
  - A soft link can span **different** file systems
  - *Why?*
- Potential **problem**:
  - The link can be invalidated: file name change, file deletion, etc.
  - *Why?*

- Sample commands (including some inode operations): https://linoxide.com/linux-command/linux-inode/

File System Structure: Overall Layout Example

For file:
/dir1/file1.dat



From: Brian Carrier, "File System Forensic Analysis"

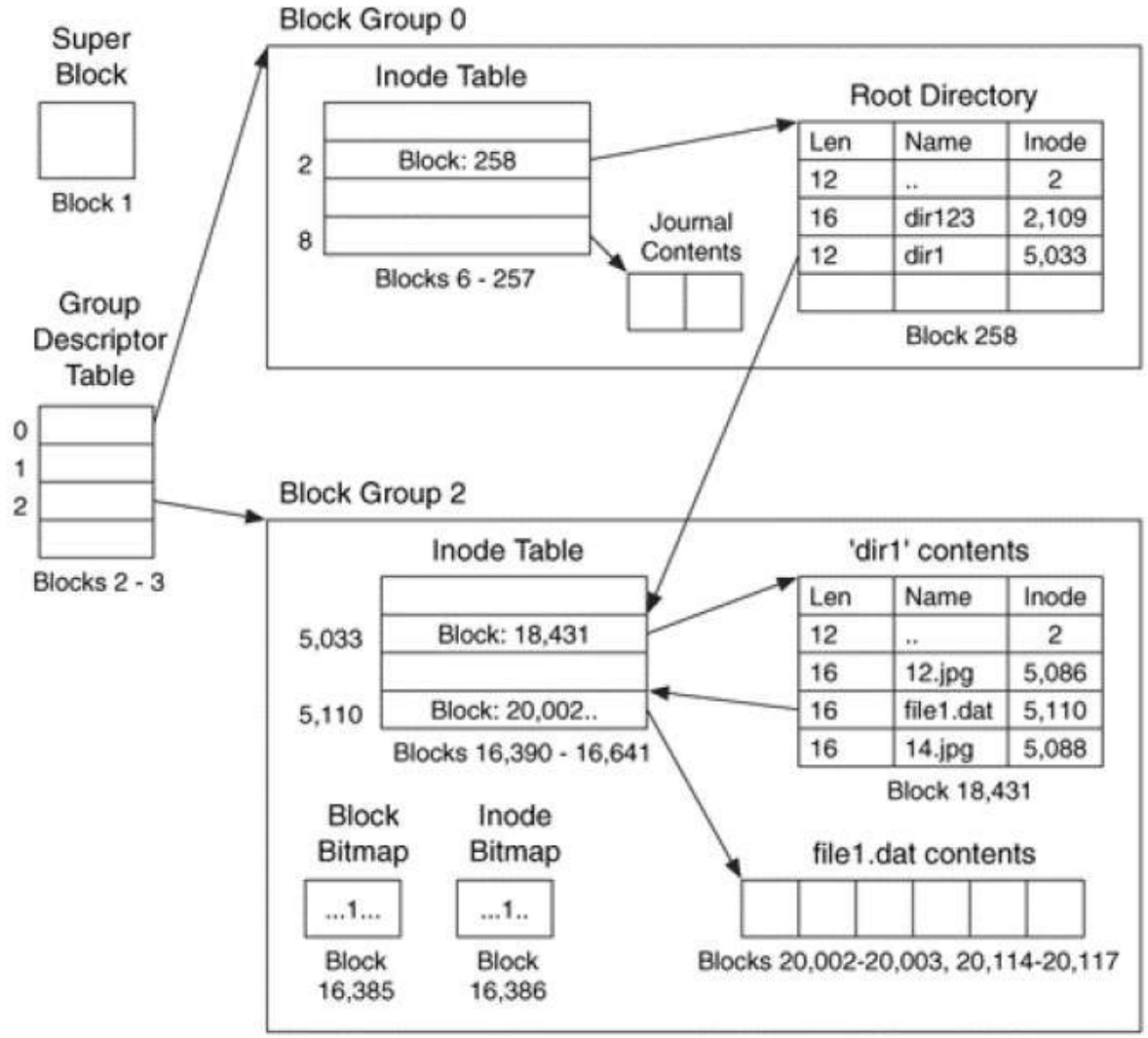
File Deletion Steps

- The following **steps** are done when a file is **deleted**:
  - Its **directory entry**: gets unallocated by adding its **record length** to the record length field in the previous directory entry
  - Its inode's **link count:** gets decremented by 1

    If the link count becomes 0: the **inode** gets deallocated by setting the **inode bitmap** accordingly, and updating the **free inode counts** in the group descriptor & superblock
  - Its **data blocks** gets deallocated by: setting the **block bitmap** accordingly, and by clearing the **block pointers** in the inode
- The directory entry and data are ***not*** actually deleted, and **may be *recoverable*** if not overwritten
- Example: deletion of "/dir1/file1.dat" in the next slide

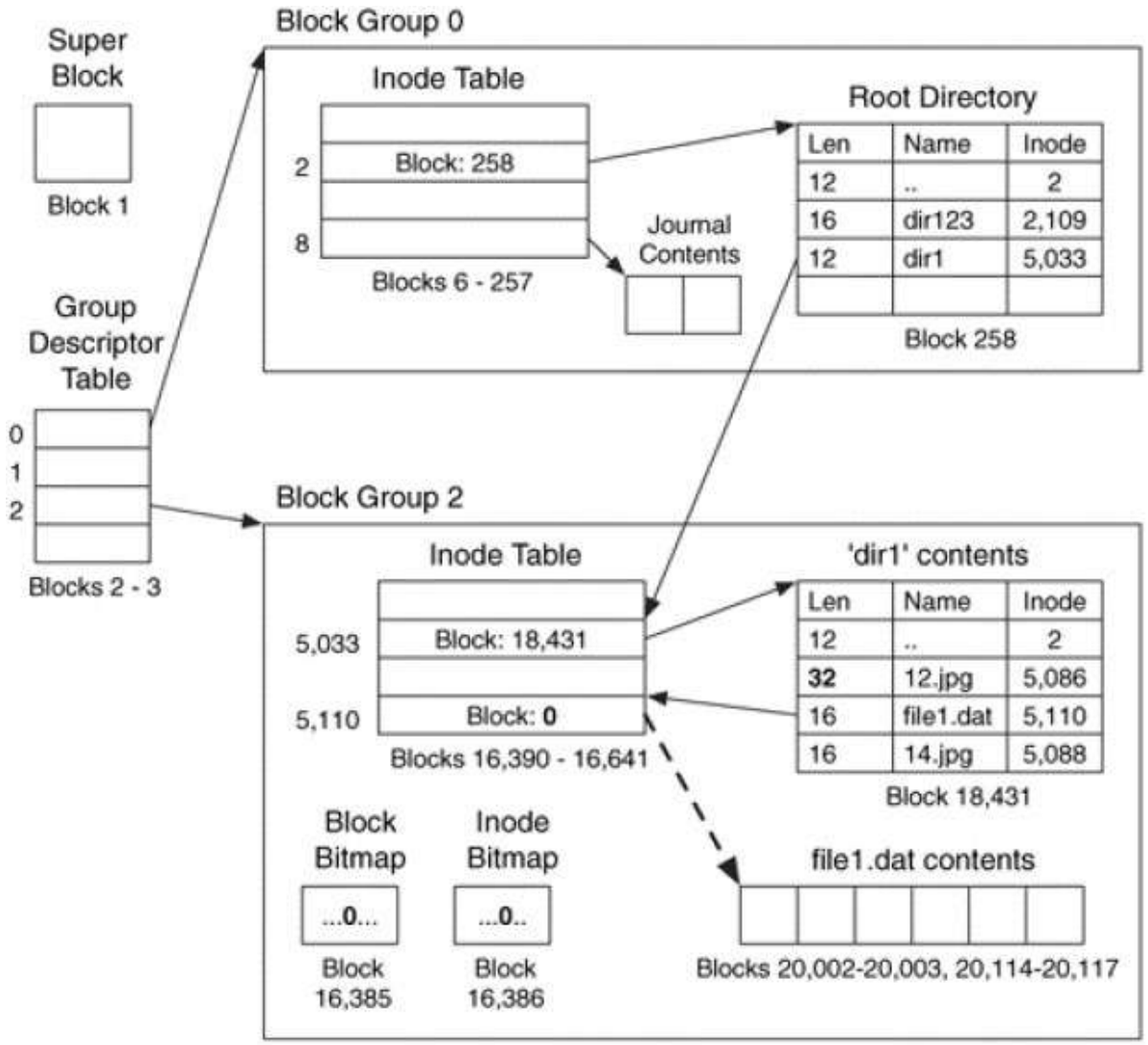
File System Structure: Overall Layout Example

For file:
/dir1/file1.dat



From: Brian Carrier, "File System Forensic Analysis"

File Deletion: The File System Layout After



From: Brian Carrier, "File System Forensic Analysis"

MACD Times in UNIX/Linux

- ***M-time***: updated when the ***content*** of a file or directory changes
  - **File**: when any of the file content changes
  - **Directory**: when a file is created or deleted inside of it
- **File operations** and M-time:
  - **Move**: M-time on the destination file is ***the same*** as the original because the file content did *not* change during the move
  - **Move (to a network attached drive)**: M-time on the destination file might be updated since the network server considers it **a new file**
  - **Copy**: M-time on the destination file is ***updated*** to the current time because it is considered new file content

MACD Times in UNIX/Linux

- ***A-time***: updated when the *content* of the file or directory is read
  - **File**: when a process reads the contents, when the file is copied, and when the file is moved to a new volume
  - **Directory**: when a directory listing is done on its contents or when a file or subdirectory is opened
- ***C-time***: corresponds to the **last *inode change***
  - When the *metadata* for a file or directory changes
  - Examples:
    - A file or directory is created
    - The permissions or ownership of a file are changed
    - The content of a file or directory changes
    - A file is moved

MACD Times in UNIX/Linux

- ***D-time***: when a file was deleted
  - It is set only when a file has been deleted
  - It is cleared when the inode is allocated
  - Many deleted files have the M-time and C-time equal to the D-time
- **Summary**:
  - **If a file is *created***:
    - MAC times are updated to the creation time
    - D-time is set to 0
    - The parent directory's M-time and C-time are updated
  - **If a file is *copied***:
    - The original file and parent directory have an updated A-time
    - The destination file has new MAC times
    - The destination parent directory has new M-time and C-time

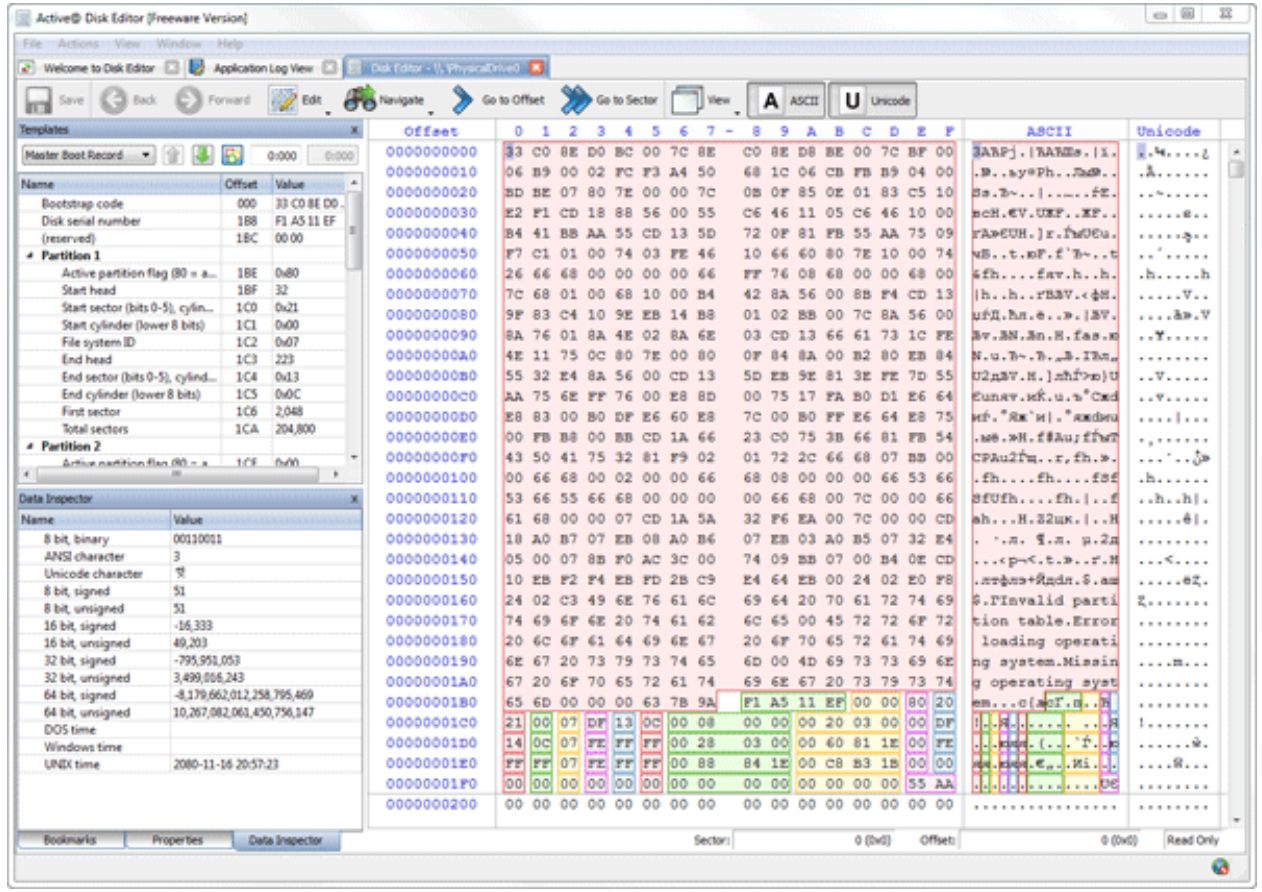
MACD Times in UNIX/Linux

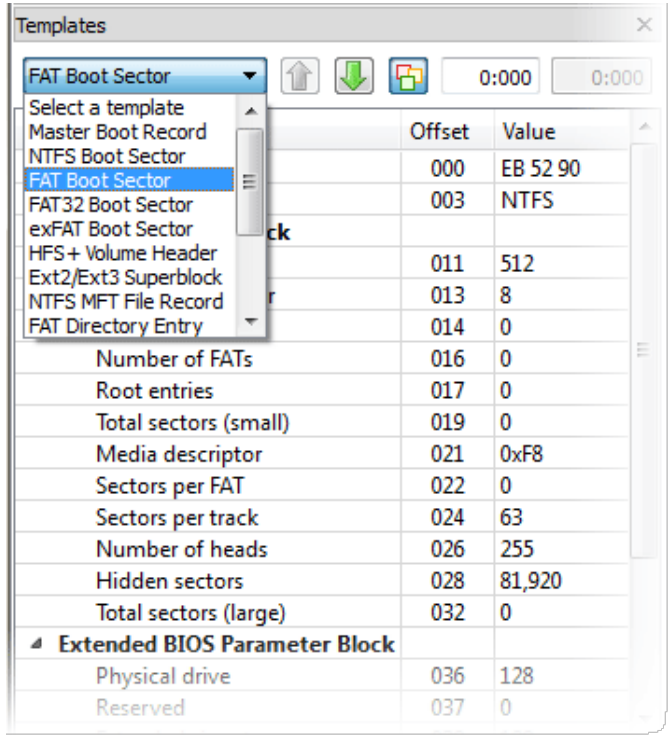
- **If a file is *moved***:
  - The parent directory of the original file has updated MAC times
  - The parent directory of the destination file has updated M-time and C-time
  - The destination file has the same M-time and A-time as the original and an updated C-time
- **Notes on a file move operation**:
  - If the move is inside of **the same volume**, the same inode will be used
  - If the move is to a **new volume**, the source inode will be unallocated and the MACD times will be updated

- *How about **B-time**?*

Inspecting Ext File System using Disk Editor

- One popular product: Active@ (https://www.disk-editor.org/)





File System Comparison

	File System	Content	Metadata	File Name	Application
ExtX	Superblock, group descriptor	Blocks, block bitmap	Inodes, inode bitmap, extended attributes	Directory entries	Journal
FAT	Boot sector, FSINFO	Clusters, FAT	Directory entries, FAT	Directory entries	N/A
NTFS	$Boot, $Volume, $AttrDef	Clusters, $Bitmap	$MFT, $MFTMirr, $STANDARD_ INFORMATION, $DATA, $ATTRIBUTE_LIST, $SECURITY_DESCRIPTOR	$FILE_NAME, $IDX_ROOT, $IDX_ALLOCATION, $BITMAP	Disk Quota, Journal, Change Journal
UFS	Superblock, group descriptor	Blocks, fragments, block bitmap, fragment bitmap	Inodes, inode bitmap, extended attributes	Directory entries	N/A

From: Brian Carrier, “File System Forensic Analysis”

Live vs Offline Linux Analysis

Linux System Analysis

- ***Live analysis***:
  - Analysis of **live accessible** Linux machine
  - **Evidence items** to be gathered & **commands** to use include:
    - Date & time: `date`, `uptime`
    - System information: `uname`
    - Network configuration: `ifconfig`, `netstat`, `route`, …
    - Process: `ps`, `top`
    - Memory: `free`
    - Users: `who`, `users`
    - Disk usage: `df`
    - Stored files : `ls`, `find`
    - Open files: `lsof`
    - Stored logs: *to be discussed more later!*
  - For the **usage of relevant commands**:
    - Check man pages ($ man man) and available resources : books & online resources

Linux System Analysis

- General **principle** for forensic analysis:
  minimize changes on the target system
  - Avoid installing any software whenever possible
  - Minimize memory utilization/footprint
  - Avoid writing into the system's file system:
    store new files to an external USB drive, use netcat for a network transfer
- **References** for Linux live analysis:
  - A handy reference for forensic purposes:
    Barry J. Grundy, "*The Law Enforcement and Forensic Examiner's Introduction to Linux*", June 2018
  - See also (inode operations):
    https://linoxide.com/linux-command/linux-inode/

Linux System Analysis: *Recap*

- **Static/offline analysis**:
  - Analysis of memory dump (e.g. acquired using LiME, Pmem):
    - **Volatility**
  - Analysis of disk image (e.g. acquired using dd, dcfldd, …):
    - **TSK**
    - Forensics suites: FTK, Encase, **Autopsy**, …
    - Disk and hex editors
    - Various analyses of Linux artefacts

Linux Log Analysis

Logging in UNIX/Linux

- Virtually every **system event**, including log-on and log-off, is **logged somewhere** in one or more system logs, depending on how the system is configured
- **Log analysis tools** are available to correlate various log entries when reconstructing system events
- Some **log files** (under /var folder):
  - /var/run/utmp, /var/log/wtmp, /var/log/lastlog: last command
  - /var/log/messages
  - /var/log/boot
  - /var/log/auth.log
  - /var/log/faillog
  - /var/log/lpr.log
  - /var/log/mail.*
  - /var/log/mysql.*

Logging in UNIX/Linux

Table 5.4 Common Log Files of Interest

/var/log/messages	Catch-all, nonspecified logs
/var/log/auth.log	User authentication successes/failures
/var/log/secure	
/var/log/sulog	"su" attempts/success
/var/log/httpd/*	Apache Web Server
/var/log/samba/smbd.log	Samba (Windows File Sharing)
/var/log/samba/nmbd.log	
/var/log/audit/audit.log	Auditd/SELinux
/var/log/maillog	Mail servers (sendmail/postfix)
/var/log/cups/access_log	CUPS Printer Services
/var/log/cron	Anacron/cron
/var/log/xferlog	FTP servers

Types of Logs (Ubuntu)

- **System** logs:
  - Authorization log
  - Daemon log
  - Debug log
  - Kernel kog
  - Kernel ring buffer
  - System log
- **Application** logs:
  - Apache HTTP server logs
  - CUPS print system logs
  - …

Types of Logs (Ubuntu)

- **Binary (non-human-readable)** logs:
  - Login failures log: at `/var/log/faillog`, can be opened using **faillog**
  - Last logins log: `/var/log/lastlog`, can be opened using **lastlog**
  - Login records log: `/var/log/wtmp`, can be opened using **who**
- System logging daemon (**syslogd**):
  see also https://www.loggly.com/ultimate-guide/linux-logging-basics/
- References:
  - https://help.ubuntu.com/community/LinuxLogFiles
  - https://stackify.com/linux-logs/
  - https://www.eurovps.com/blog/important-linux-log-files-you-must-be-monitoring/

Text Log-File Inspection

- **Literal/keyword searches**:
  - Look for **exact string** of characters
  - Good for looking for occurrences of unique strings
  - Bad if there are relevant **inexact/similar strings**: e.g. diary vs diaries
- ***Grep*** (**g**lobally-search a **r**egular **e**xpression and **p**rint):
  - Standard UNIX/Linux tools: **grep**, **egrep** (**grep -E)**
  - Useful for searching strings within **text** files, including various Linux log files
  - Won't work on binary/non-text and encrypted files
  - User needs to understand **regular expression** to search effectively

Grep Command

Some **common usages**:

- Search and display any line inside file that contains the sought **string** “word”:
  `grep 'word' file`
- Do a **case-insensitive** search for the string “word”:
  `grep -i 'word' file`
- Display lines that **do NOT contain** the string “word”, i.e. invert the match:
  `grep -v 'word' file`
- **Recursively** search all files in the current directory and all of its subdirectories for the string “word”:
  `grep -R 'word'`
- Search for the **whole word** “word”, and not word123:
  `grep -w 'word' file`

Regular Expression (Regex):

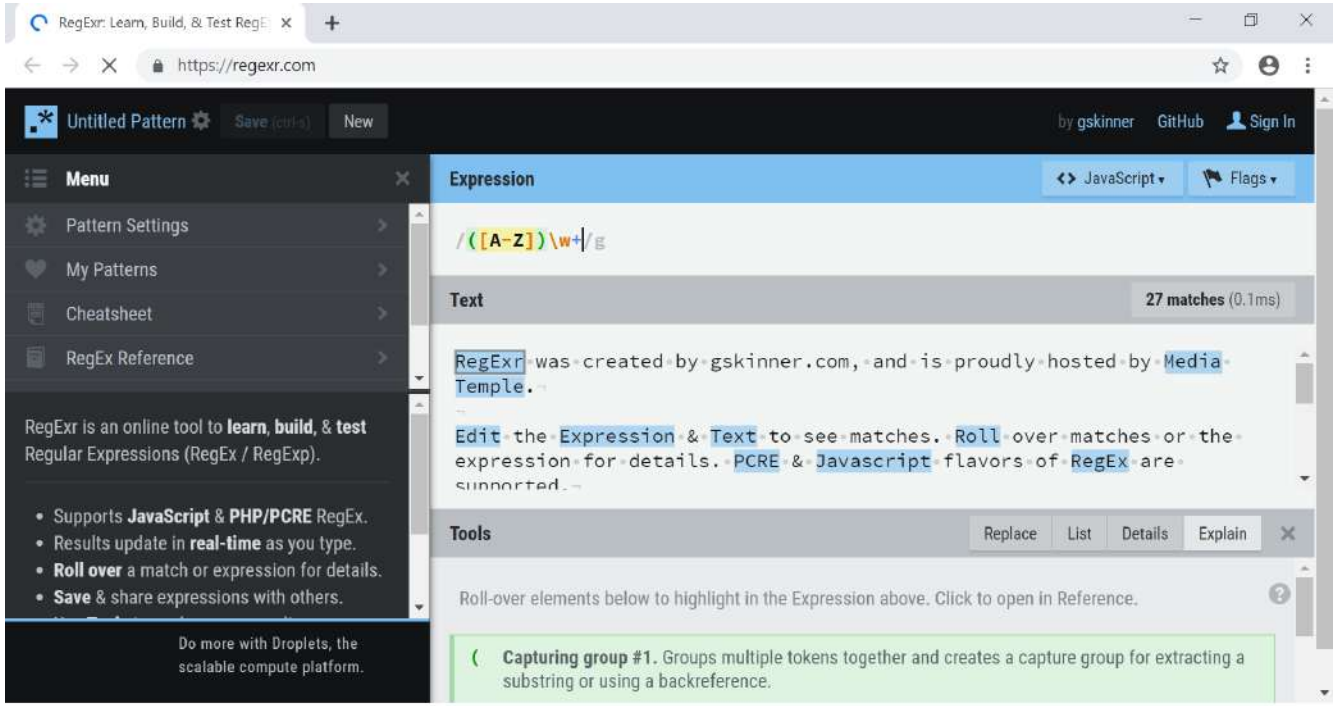
- A quick reference guide: https://docs.microsoft.com/en-us/dotnet/standard/base-types/regular-expression-language-quick-reference
- **Character classes**:
  - [character_group]: e.g. [ae]
  - [^character_group]: e.g. [^ae]
  - [first-last] **range**: e.g. [A-Z]
  - . : matches any **single character** (except \n)
  - \d: matches any **decimal digit**
- **Anchors**:
  - ^: match the beginning of the line
  - $: match the end of the line (before \n)

Regular Expression (Regex):

- **Quantifiers**:
  - *: matches the previous element **zero or more times**
  - +: matches the previous element **one or more times**
  - ?: matches the previous element **zero or one time**
  - {n}: matches the previous element exactly **n times**
- **Alternation** construct:
  - |: matches any one element separated by |
- You can **practice** grep using your own regular expressions:
  - Sample Linux logs: https://www.ossec.net/docs/log_samples/linux/index.html
  - Regex examples: https://docs.microsoft.com/en-us/dotnet/standard/base-types/regular-expression-examples
- See Lab 6 Task 7

Web-based Regular Expression Editors

- An online regex editor: https://regexr.com/:



- Other editors: https://regex101.com/, https://www.regextester.com/, …

Text Log-File Inspection

- Other useful **UNIX/Linux *tools***:
  - awk
  - sed
- Additional relevant **UNIX/Linux *commands***:
  - cut
  - sort
  - uniq
  - head
  - tail
- For Windows forensic workstation: PowerShell
- Language independent: Python, …

Binary Log-File Inspection

- *How about inspecting **binary logs**?*
- **App-specific** log-viewer commands and tools: *mentioned earlier*
- Some useful ***general-purpose*** tools:
  - TSK's **sigfind**
    - Usage: **sigfind [-b** *bsize* **] [-o** *offset* **] [-t** *template* **] [-lV] [** *hex_signature* **]** *file*
    - Example:

```
# sigfind -o 56 -l ef53 disk-8.dd
Block size: 512 Offset: 56
Block: 298661 (-)
Block: 315677 (+17016)
Block: 353313 (+37636)
Block: 377550 (+24237)
```

From: Brian Carrier, "File System Forensic Analysis"

  - **Hex editors**

Other Linux Artefact Analysis

- Linux **shell history**:
  - E.g. .bash_history
  - Find out the last executed **shell commands** of users
  - See: https://www.youtube.com/watch?v=wv1xqOV2RyE&list=PLfouvuAjspToi20WLhZwCe9gDMQCJk_Jz
- *How about **unknown binary files**?*
  - Related to **malware analysis/forensics**
  - See: https://www.youtube.com/watch?v=3xAEsDT-4NA, https://www.youtube.com/watch?v=SjDH_vTuefM

Lab 6 Exercises

Lab 6

- Task 1: Inspecting a file hidden inside **NTFS's Alternate Data Stream**
- Task 2: Analyzing **Windows event log** files using Event Viewer
- Task 3: Analyzing **prefetch files** using WinPrefetchView
- Task 4: Analyzing **shortcuts files** using Windows File Analyzer
- *(Optional)* Task 5: Analyzing **jump list files** using JumpLister
- *(Optional)* Task 6: Analyzing **thumbnail caches** using Windows File Analyzer
- *(Optional)* Task 7: Performing a Linux **log analysis** using grep

- ***No* graded lab tasks: *have a good recess week!*** ☺

Questions?

See you two weeks from now!