CS 798: Digital Forensics and Incident Response

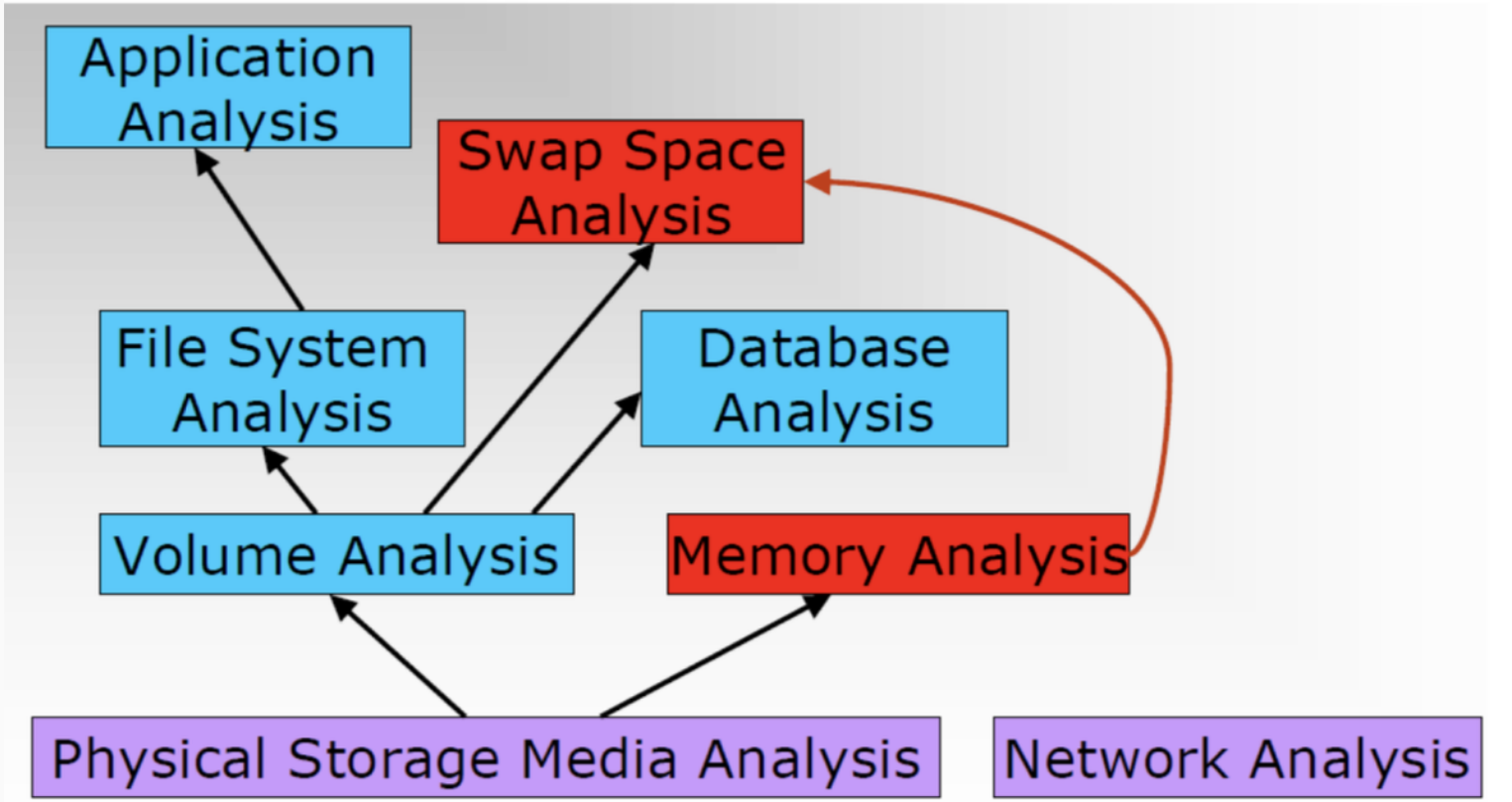
Lecture 8 - File System Analysis

Diogo Barradas

Winter 2025

University of Waterloo

We talked about storage and volumes...



Source: „File System Forensic Analysis", Brian Carrier

- Seen how to deal with / interpret volumes and partitions
- Studied the implications of storage technology in forensics

Outline

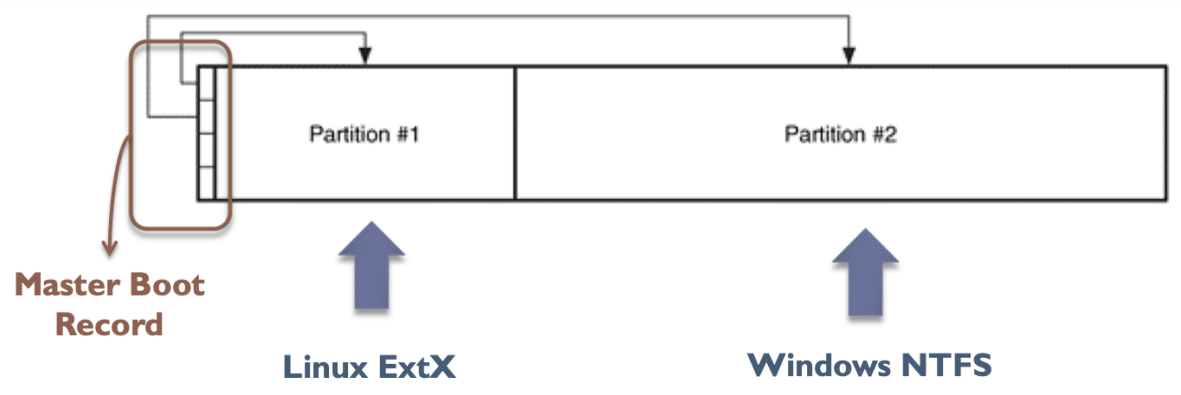
1. Evidence from file systems

2. File system analysis techniques

Evidence from file systems

Partitions have their own file systems

- A disk that is organized using DOS partitions has a Master Boot Record (MBR) in the first 512-byte sector
- The MBR has a partition table with four entries, one for each partition
  - e.g., two partitions, each formatted to a different FS

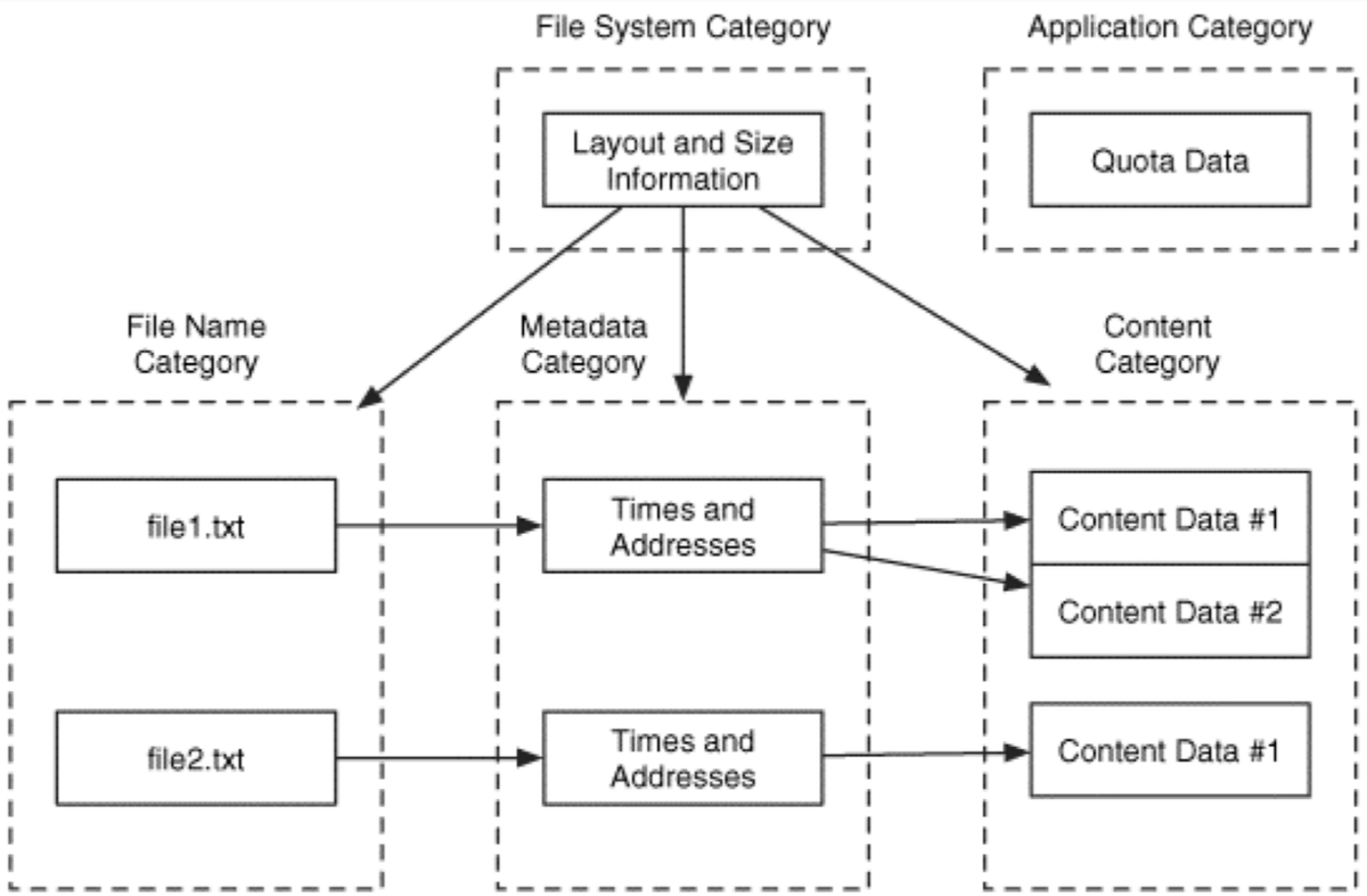


Analysis of persistent storage

- To access overt content, e.g.:
  - Folder and file contents
  - Folder and file names
  - Folder and file permissions, time stamps
- To access deleted content, e.g.:
  - Locate and recover deleted files
  - Identify names of deleted files
- To access hidden content, e.g.:
  - Data located inside file slack space

DECtape	HPFS
OS/3x0 FS	Rock Ridge
Level-D	JFS1
George 3	VxFS
Version 6 Unix file system (V6FS)	ext
RT-11 file system	AdvFS
Disk Operating System (GEC DOS)	NTFS
CP/M file system	LFS
ODS-1	ext2
GEC DOS filing system extended	Xiafs
FAT (8-bit)	UFS1
DOS 3.x	XFS
UCSD p-System	HFS
CBM DOS	FAT16X
Atari DOS	Joliet ("CDFS")
Version 7 Unix file system (V7FS)	UDF
ODS-2	FAT32, FAT32X
FAT12	QFS
ProDOS	GPFS
DFS	Be File System
ADFS	Minix V2 FS
FFS	HFS Plus
FAT16	NSS
MFS	PolyServe File System (PSFS)
Elektronika BK tape format	ODS-5
HFS	WAFL
Amiga OFS[1]	ext3
GEMDOS	ISO 9660:1999
NWFS	JFS
High Sierra	GFS
FAT16B	ReiserFS
Minix V1 FS	zFS
Amiga FFS	FATX
ISO 9660:1988	UFS2
	OCFS
	SquashFS
	VMFS2
	Lustre
	Fossil
	Google File System

FS evidence can be grouped into categories



Data categories of popular file systems

Category	FAT FS	EXTx	NTFS
File System	Boot sector	Superblock, group	$Boot, $Volume, descriptor, $AttrDef
Content	Clusters, FAT	Blocks, block bitmap	Clusters, $Bitmap
Metadata	Directory entries, FAT	Inodes, inode bitmap	$MFT,$DATA, $ATTRIBUTE_LIST
File Name	Directory entries	Directory entries	$FILE_NAME, $IDX_ROOT, $BITMAP
Application	N/A	Journal	Journal, disk quota

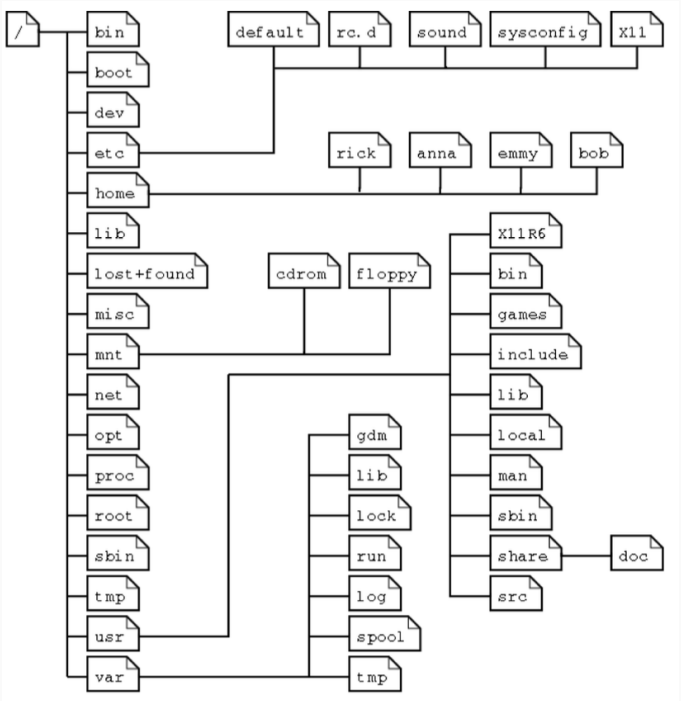
The ExtX file system family

- Ext2, Ext3, and Ext4 file systems (ExtX) are the default file systems for many distributions of Linux
  - Ext3 = Ext2 + journaling
- An ExtX file system manages the storage space of a given partition
  - Most space is used for storing files' contents
  - Some space used for ExtX metadata

ExtX volume (crude representation)

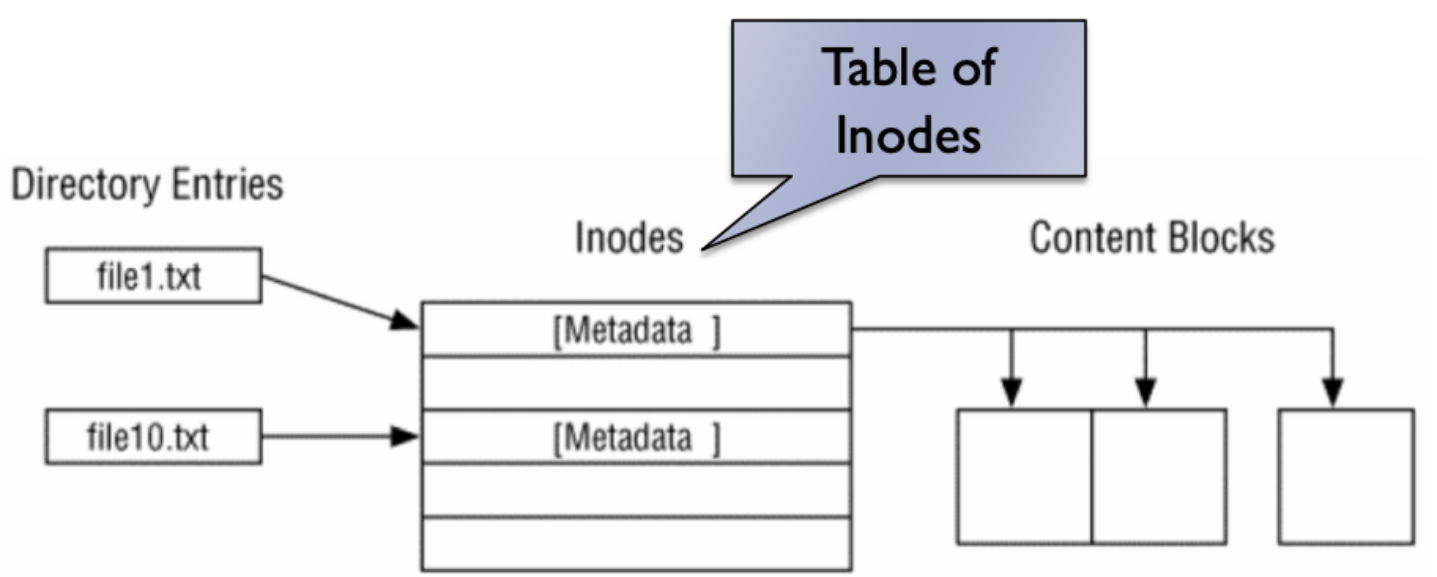
A typical Linux file system structure

- Linux organizes its file system in a hierarchical structure, similar to a tree
- The root directory, denoted as '/', is the topmost directory and serves as the base for the entire file system.



Overview of main ExtX data structures

- File contents are stored inside blocks (e.g., 4KB)
  - Each block has an address within the partition (0 up to max #blocks -1)
- The blocks allocated to a file are kept by a record called inode
  - ExtX keeps track of all inodes inside a table
- Directory entries associate the file name with the file's inode

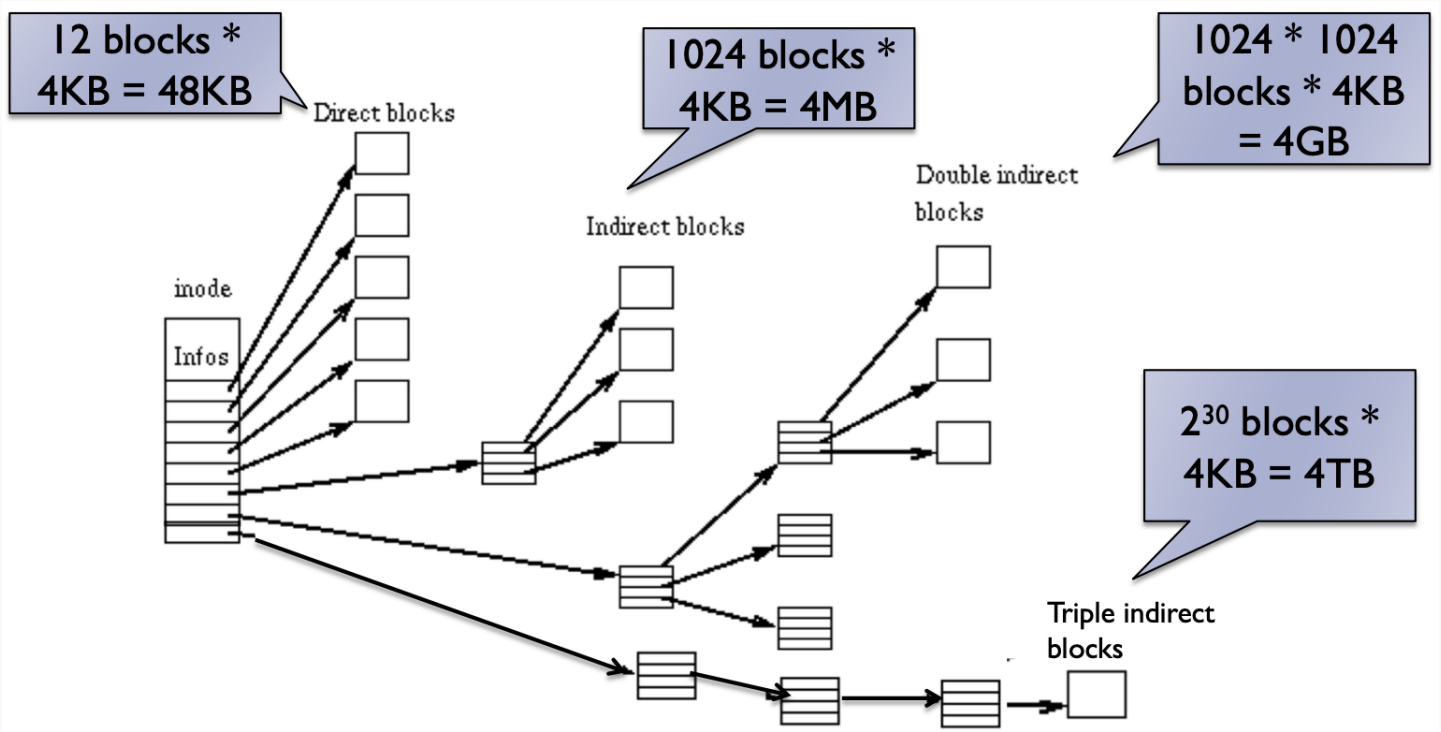


Overview of main ExtX data structures

Size (bytes)	Name	What is this field for?
2	mode	Read/write/execute permissions
2	uid	User ID of the file owner
4	size	Size of the file in bytes
4	time	Last access time
4	ctime	Creation time
4	mtime	Last modification time
4	dtime	Deletion time
2	gid	Group ID of the file
2	links_count	Number of hard links pointing to this file
4	blocks	Number of data blocks allocated to this file
4	flags	File or directory indicator and other flags
60	block	Pointers to data blocks (15 direct and indirect)

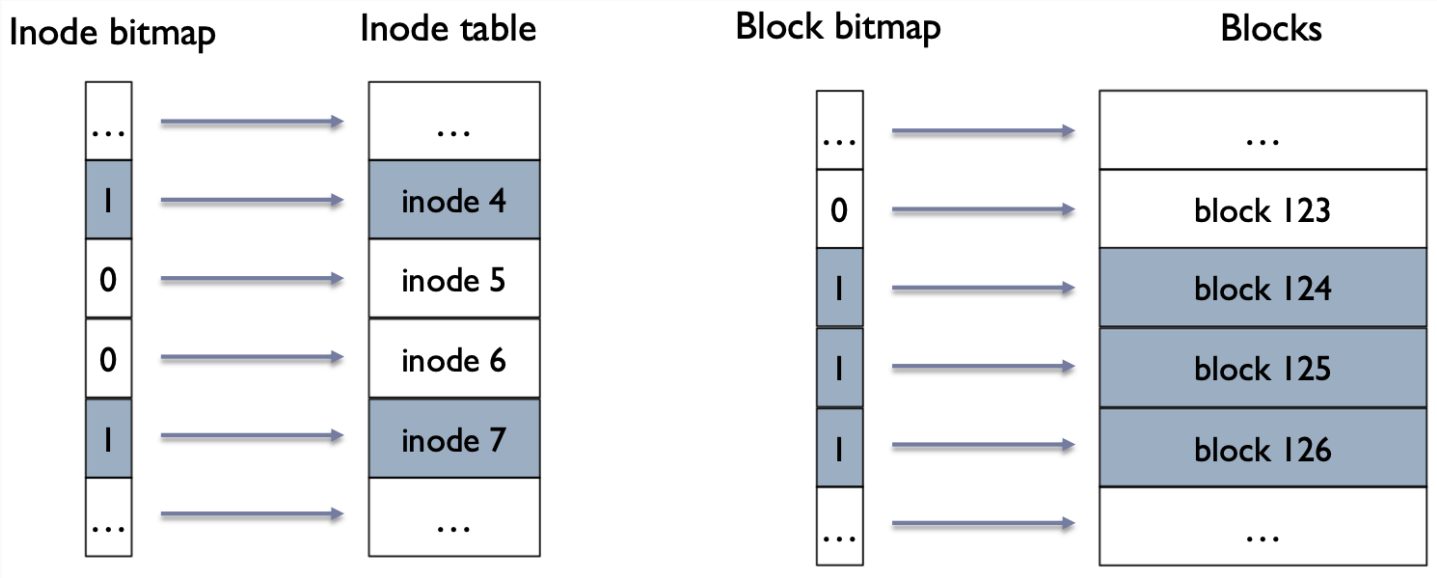
Inode block pointers

- Consider a file with size 4097 bytes
  - Block size is 4 KB (4096 bytes)



Keeping track of inode and block allocation

- Using bitmaps: bit arrays, each bit indicates allocation status
  - Inode bitmap: tells which inodes are allocated to files
  - Block bitmap: tells which data blocks are allocated to files



Creating and deleting a file on Ext2

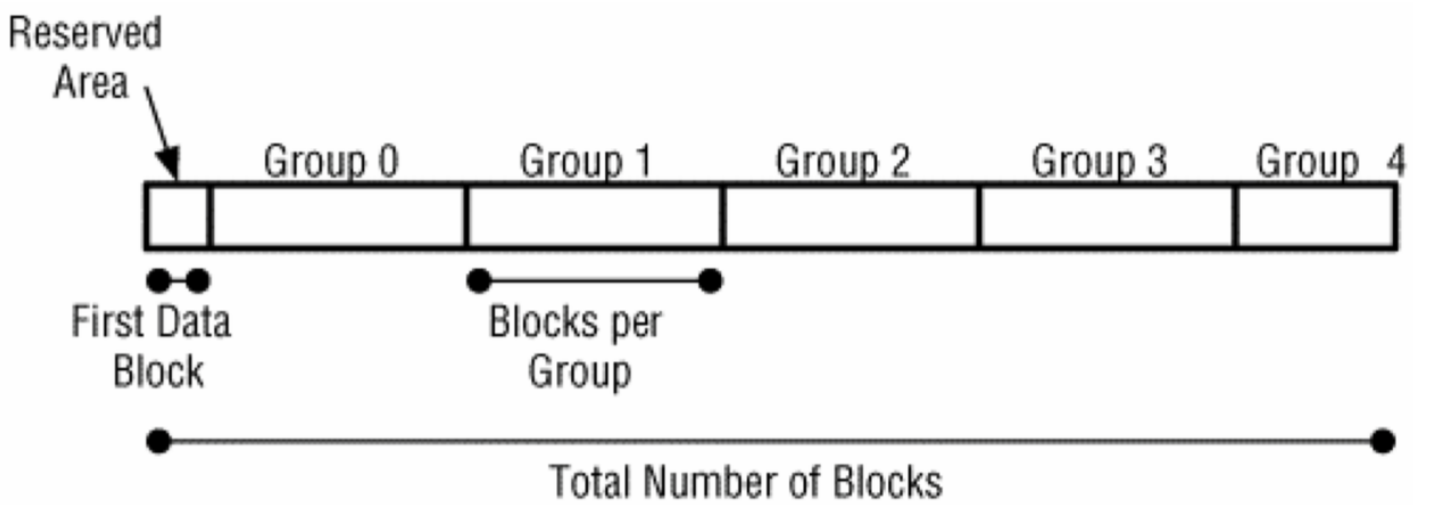
- Additional helper data structures
  - Inode bitmap: tells which inodes are allocated to files
  - Block bitmap: tells which data blocks are allocated to files
- Create a new file
  - Allocate a new inode (inode bitmap is updated)
  - Allocate required blocks (block bitmap is updated)
  - Allocate entry in directory (entry points to inode)
  - Update data blocks, inode, and directory entry
- Delete an existing file
  - Update the inode and block bitmaps, unallocate directory entry
  - Most contents of inode, data blocks, and directory entry remain intact

Journaling: Maintaining consistency

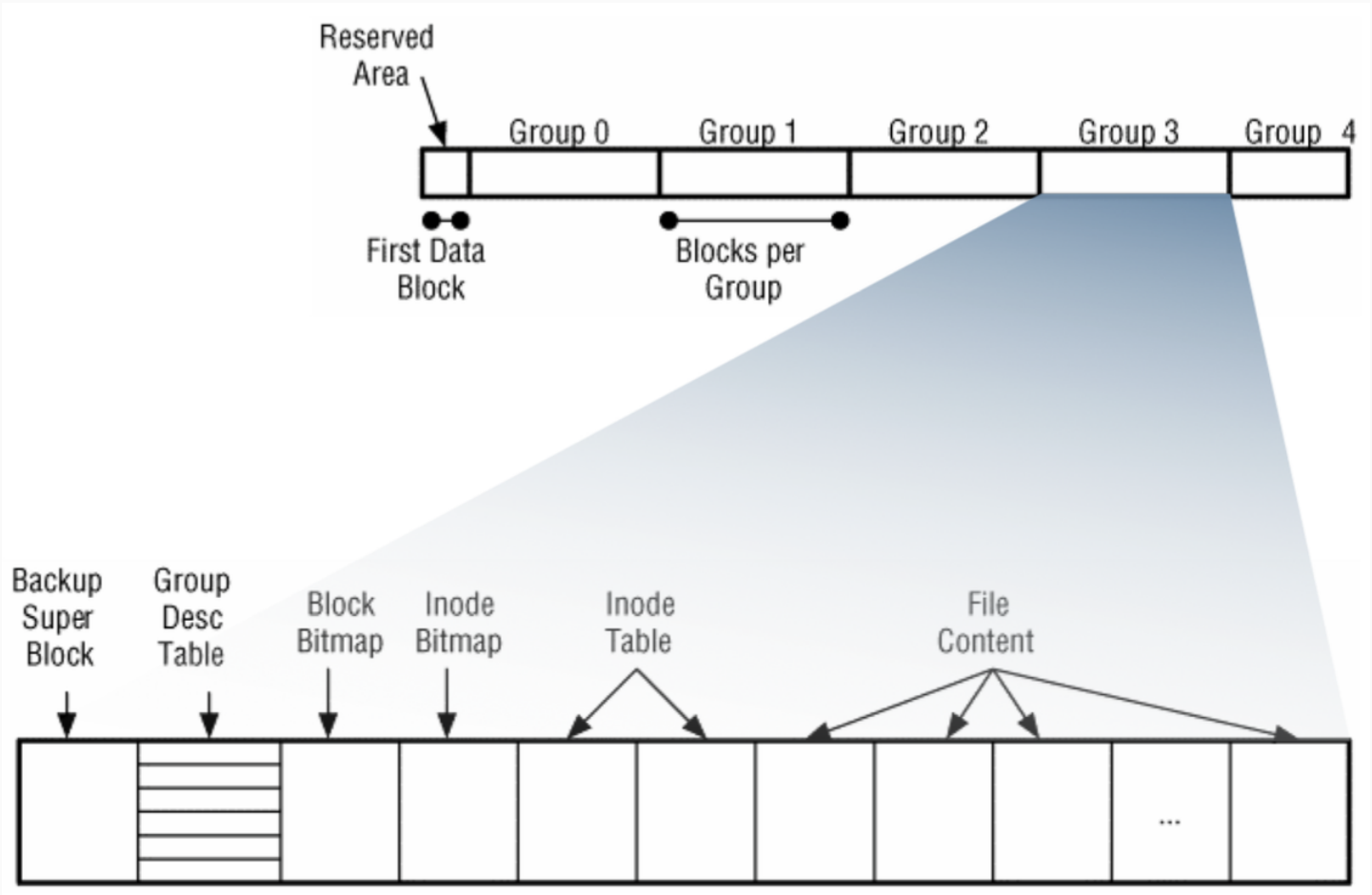
- Many operations results in multiple, independent writes to the file system
  - Example: append a block to an existing file
    1. Update the free data bitmap
    2. Update the inode
    3. Write the user data
- What if the computer crashes during this process?
  - The file system may enter an inconsistent state
- Solution: Ext3 maintains a journal in inode 8 (typically)
  - Journal records transactions of FS operations
  - Typically implemented as circular buffers
  - Only overwritten over time – provide recent history of updates to file system and, potentially, forensic evidence!

On-disk organization of a ExtX file system

- Organized as sequence of logical blocks
  - The block size is defined upon disk formatting; 1, 2, 4 or 8KB are common
- Blocks are grouped into larger units called block groups
  - All block groups have equal length possibly except the last one
- The first data block (aka boot block) is not used by the FS
  - Has a fixed 1024 byte length and may contain bootstrap code

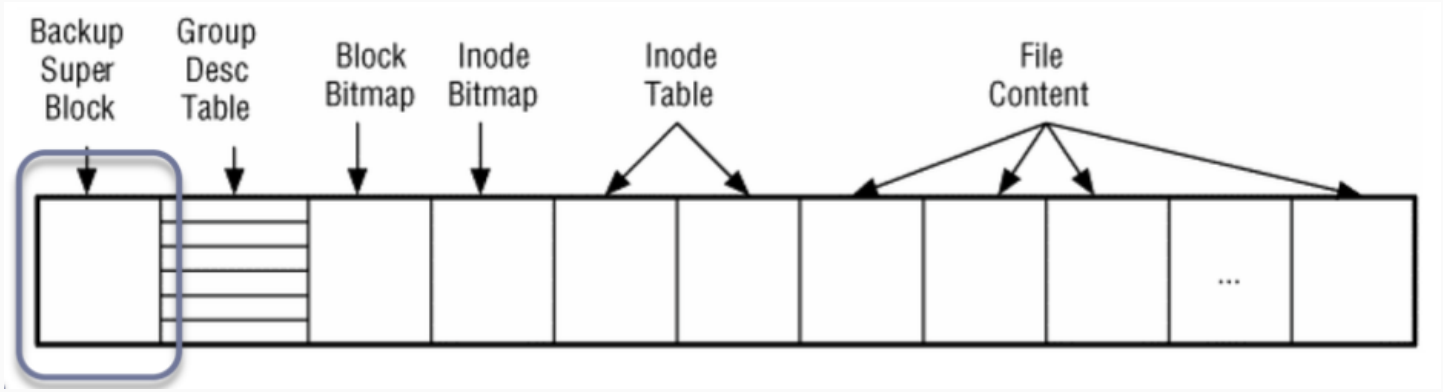


Group block internals



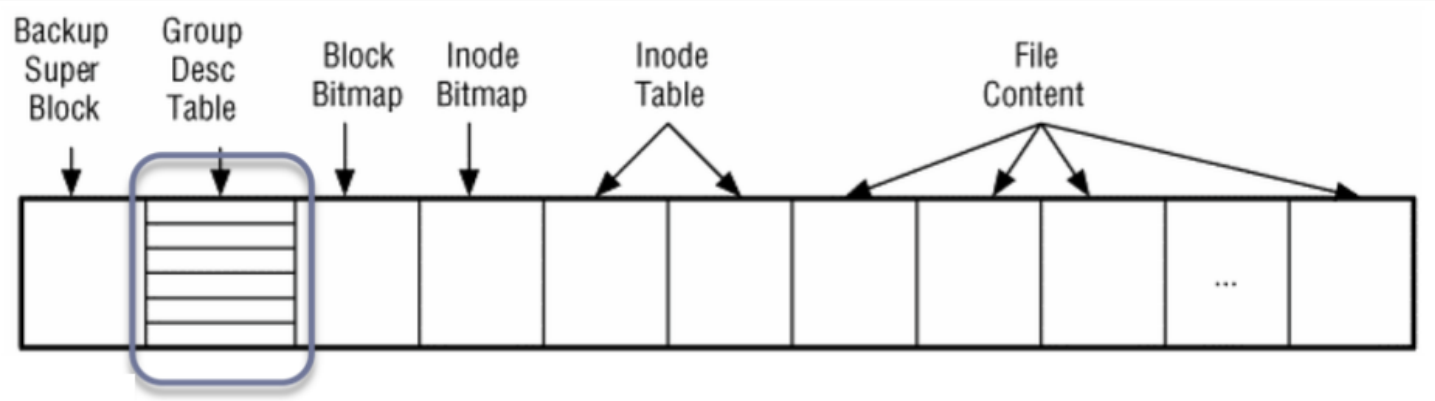
Group block internals: The superblock

- Superblock: contains fundamental info about the file system
  - block size, total number of blocks, # of blocks per group...
- The superblock is replicated in all group blocks
  - Copies of the superblock are in the first block of each block group



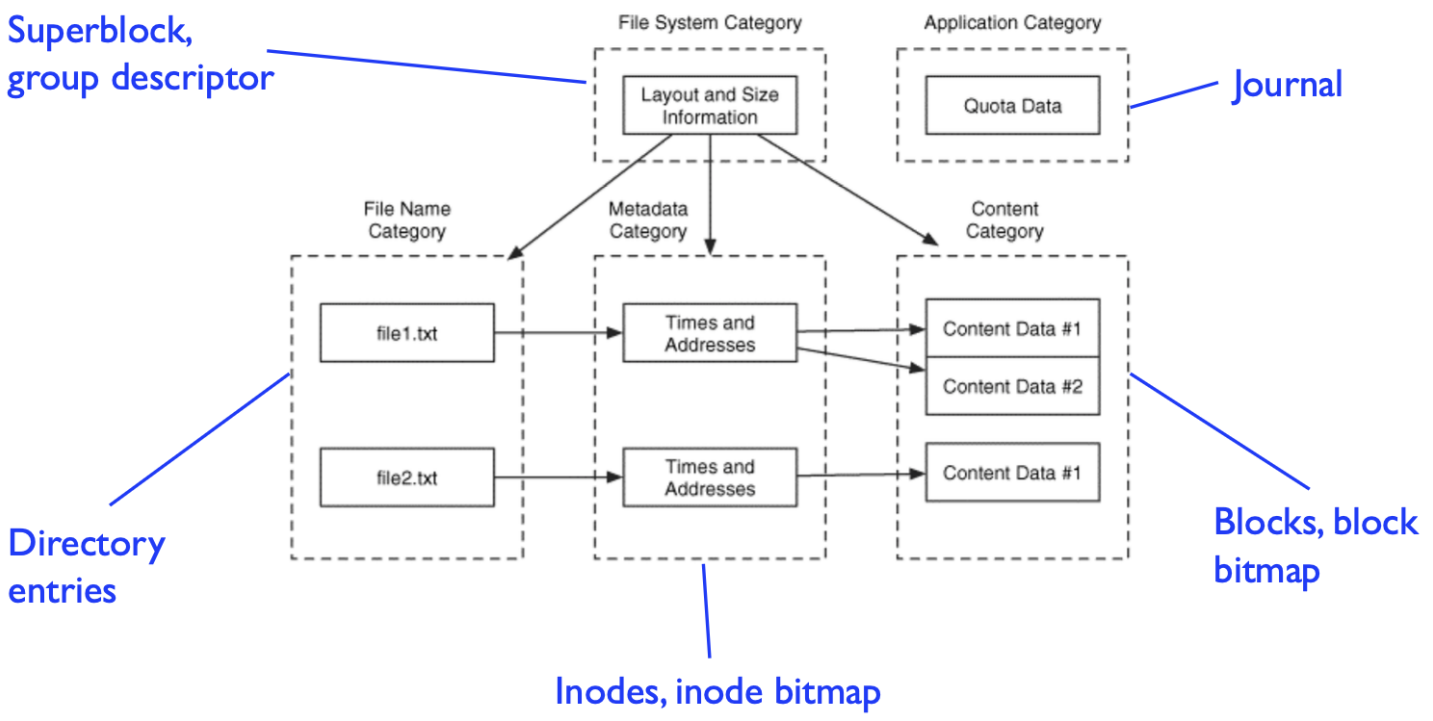
Group block internals: The grup descriptor table

- Group descriptor table: array of descriptors for all block groups
  - Provides the location of the inode bitmap and inode table, block bitmap, number of free blocks and inodes, etc.
  - The size of the descriptor table depends on how many groups are defined
  - Every block group descriptor table contains all info about all block groups



On-disk organization of a ExtX file system

- Data evidence categories of the ExtX file system family



File system analysis techniques

TSK provides analysis tools in each category

- Data evidence categories of the ExtX file system family

File System Layer Tools

These file system tools process general file system data, such as the layout, allocation structures, and boot blocks

- fsstat: Shows file system details and statistics including layout, sizes, and labels.

File Name Layer Tools

These file system tools process the file name structures, which are typically located in the parent directory.

- ffind: Finds allocated and unallocated file names that point to a given meta data structure.
- fls: Lists allocated and deleted file names in a directory.

Meta Data Layer Tools

These file system tools process the meta data structures, which store the details about a file. Examples of this structure include directory entries in FAT, MFT entries in NTFS, and inodes in ExtX and UFS.

- icat: Extracts the data units of a file, which is specified by its meta data address (instead of the file name).
- ifind: Finds the meta data structure that has a given file name pointing to it or the meta data structure that points to a given data unit.
- ils: Lists the meta data structures and their contents in a pipe delimited format.
- istat: Displays the statistics and details about a given meta data structure in an easy to read format.

Data Unit Layer Tools

These file system tools process the data units where file content is stored. Examples of this layer include clusters in FAT and NTFS and blocks and fragments in ExtX and UFS.

- blkcat: Extracts the contents of a given data unit.
- blkls: Lists the details about data units and can extract the unallocated space of the file system.
- blkstat: Displays the statistics about a given data unit in an easy to read format.
- blkcalc: Calculates where data in the unallocated space image (from blkls) exists in the original image. This is used when evidence is found in unallocated space.

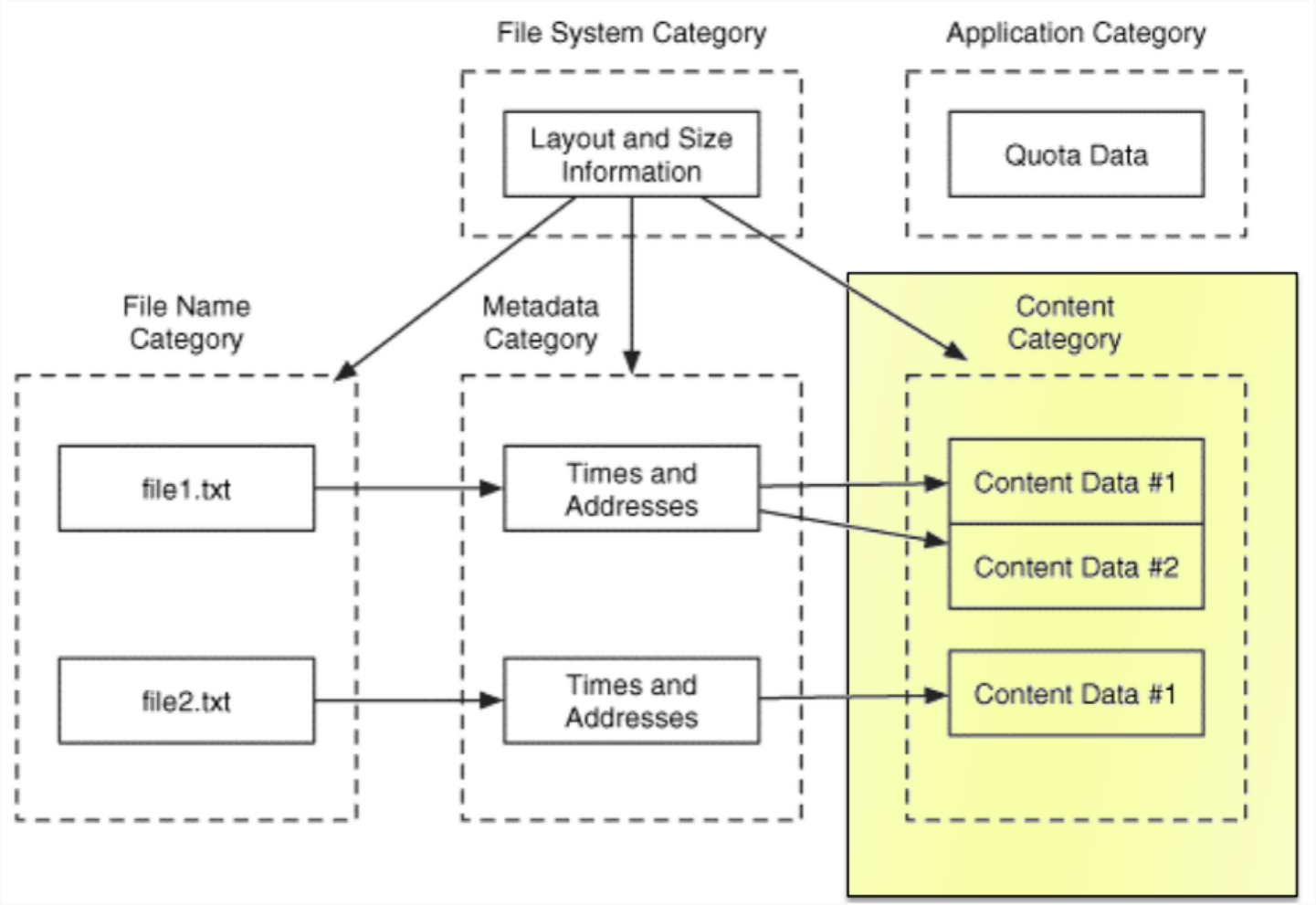
File System Journal Tools

These file system tools process the journal that some file systems have. The journal records the metadata (and sometimes content) updates that are made. This could help recover recently deleted data. Examples of file systems with journals include Ext3 and NTFS.

- jcat: Display the contents of a specific journal block.
- jls: List the entries in the file system journal.

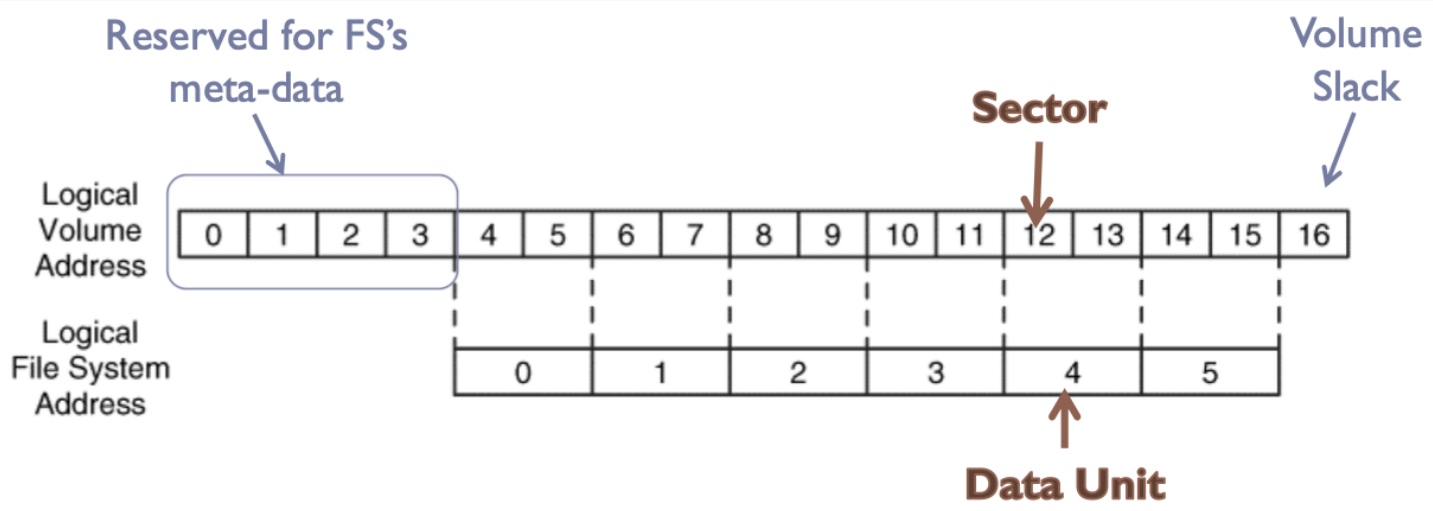
`https://wiki.sleuthkit.org/index.php?title=TSK_Tool_Overview`

File system evidence: Content category



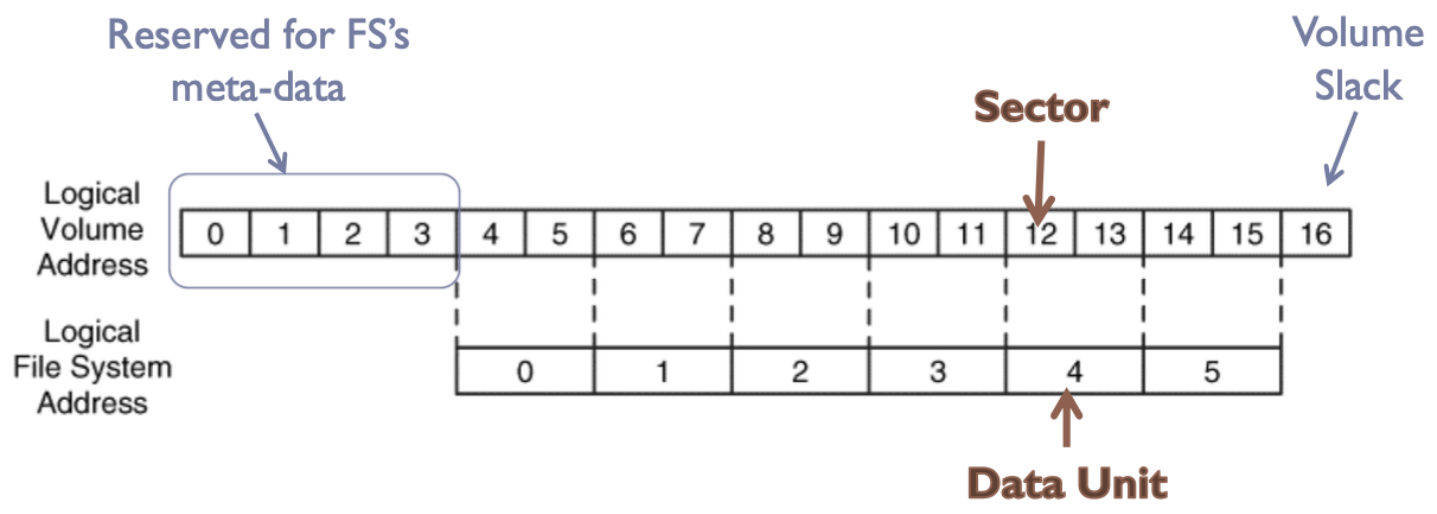
Evidence in the content category

- Content category includes the storage locations allocated to files: data units (e.g., named blocks in ExtX)
  - A data unit is a set of consecutive sectors of the volume
  - A sector has: physical address and a logical FS address
- In ExtX, evidence in this category includes: blocks and the block bitmap



1. Data unit viewing

- Used when the investigator knows the direct address where evidence may be located, e.g., a DU allocated to a specific file or that has special meaning
  - E.g, in many FAT32 file systems, sector 3 is not used and is all zeros; viewing it shows if there are non-zero hidden data



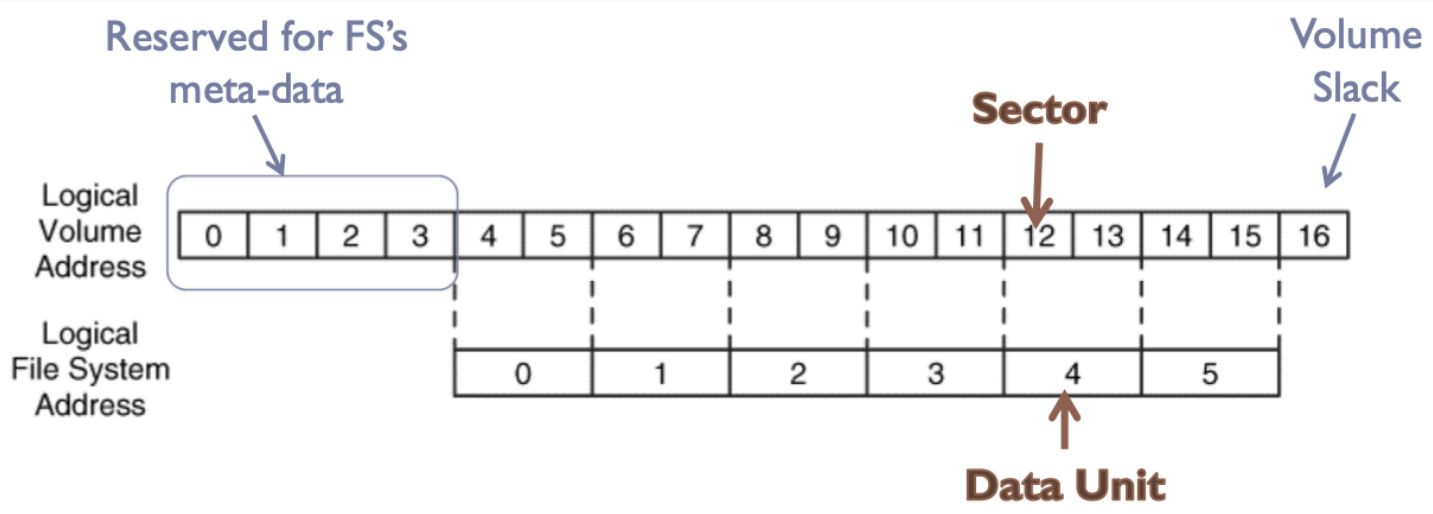
Example of dcat

- Dump the content of block 1 from a Ext3 FS image

```
 #dcat -f linux-ext3 ext3.dd 1 | xxd
0000000:  0200 0000 0300 0000 0400 0000 d610 7b3f
0000016:  0a00 0000 0000 0000 0000 0000 0000 0000
0000032:  0280 0000 0380 0000 0480 0000 0000 8e3f
0000048:  0100 0000 0000 0000 0000 0000 0000 0000
REMOVED
```

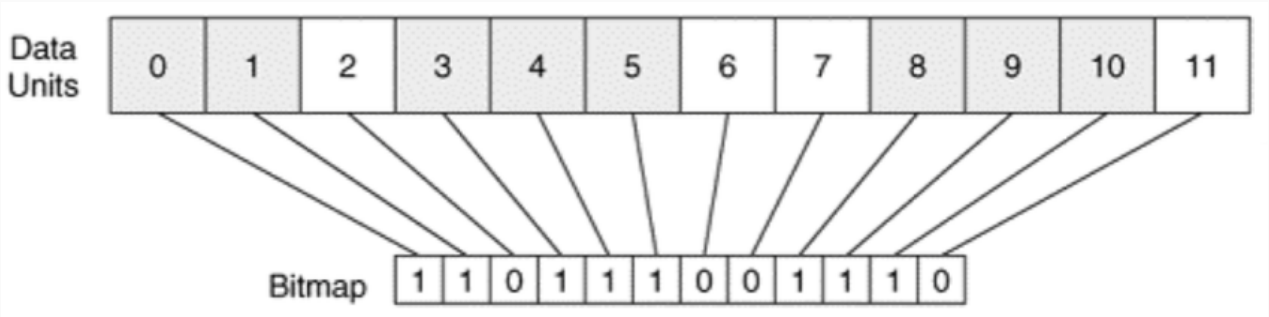
2. Logical file system-level searching

- We know what to look for, but not the place: a logical file system search looks in each DU for specific values
  - E.g., search for "forensics" or a specific file header value
- Note: if the value is located in two non-consecutive DUs of a fragmented file, search will not find it



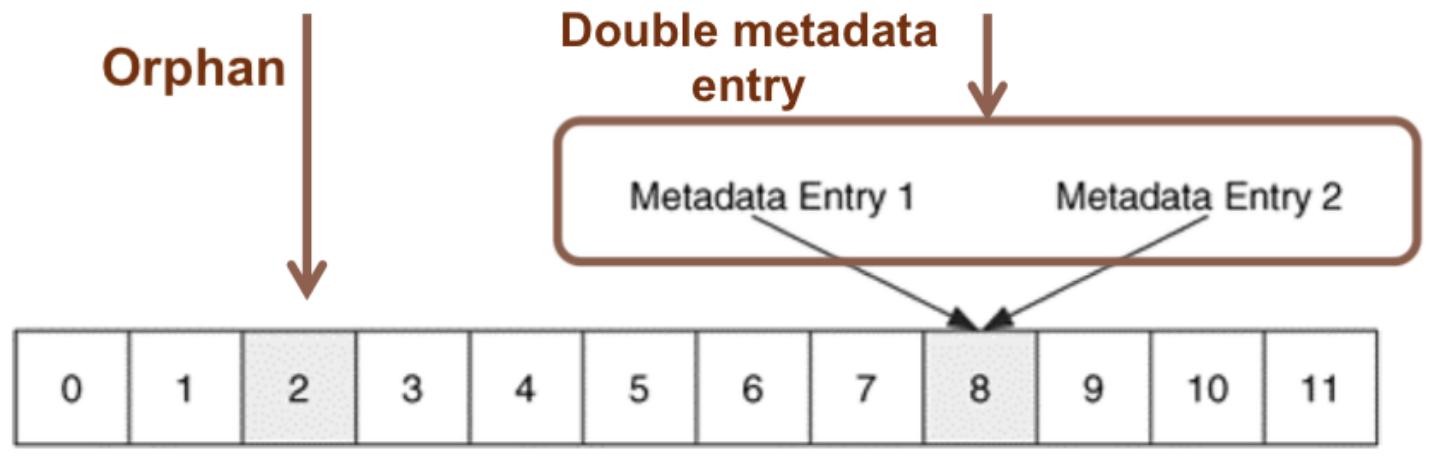
3. Unallocated data unit searching

- If we do not know the location of evidence, but we know that it is unallocated, we can focus our attention there
- Some tools can extract all unallocated DUs to a separate file; others restrict analysis to only the unallocated areas
- The definition of unallocated space may vary: you need to know what your analysis tool considers unallocated data
  - `gparted` shows drive space not assigned to any partition as unnalocated
  - In this class context, unnalocated means file system space not currently assigned



4. Consistency checking

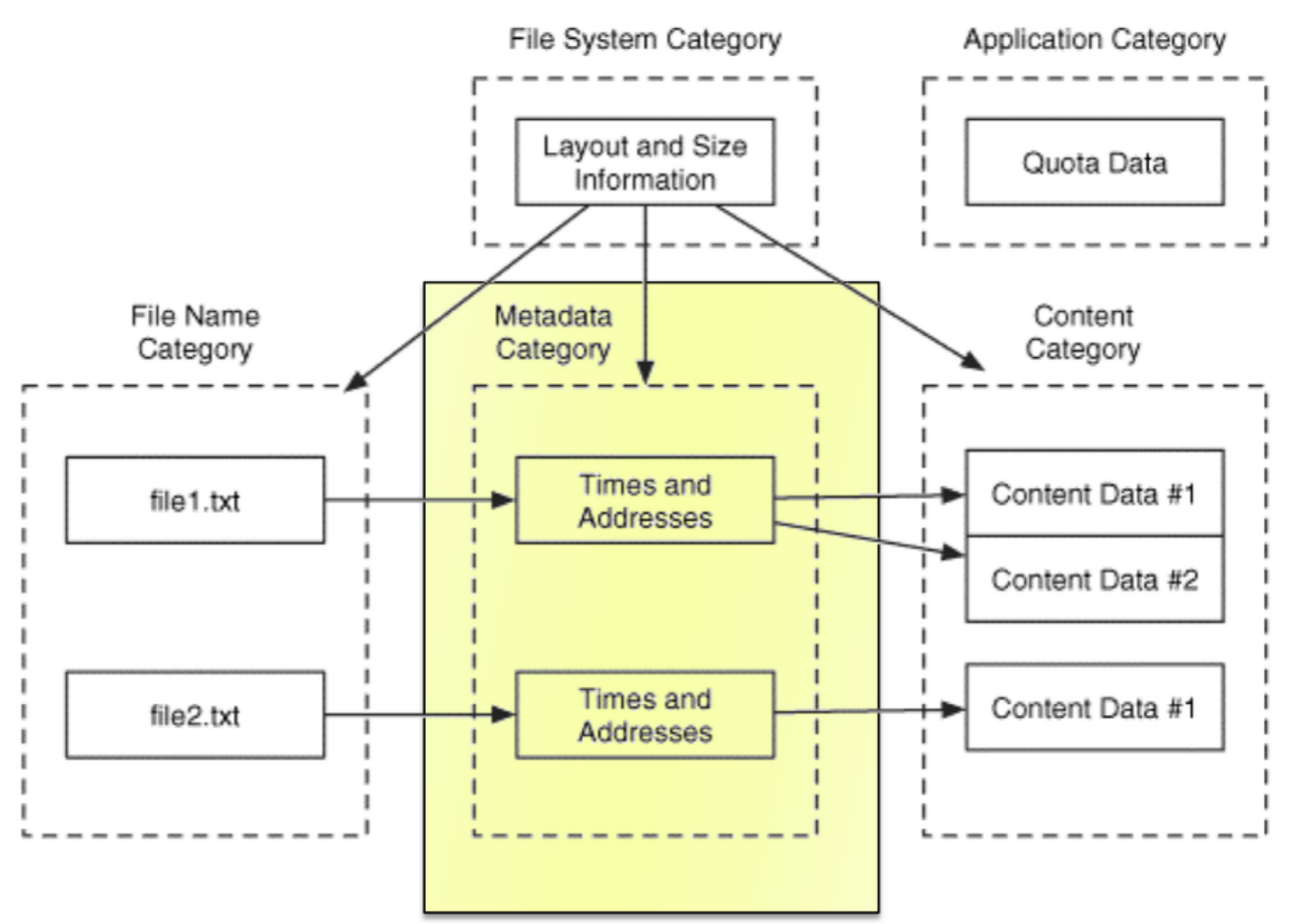
- Consistency checks allow us to determine if the file system is in a suspicious state
- Example: orphans and double metadata entries



Summary of techniques in the content category

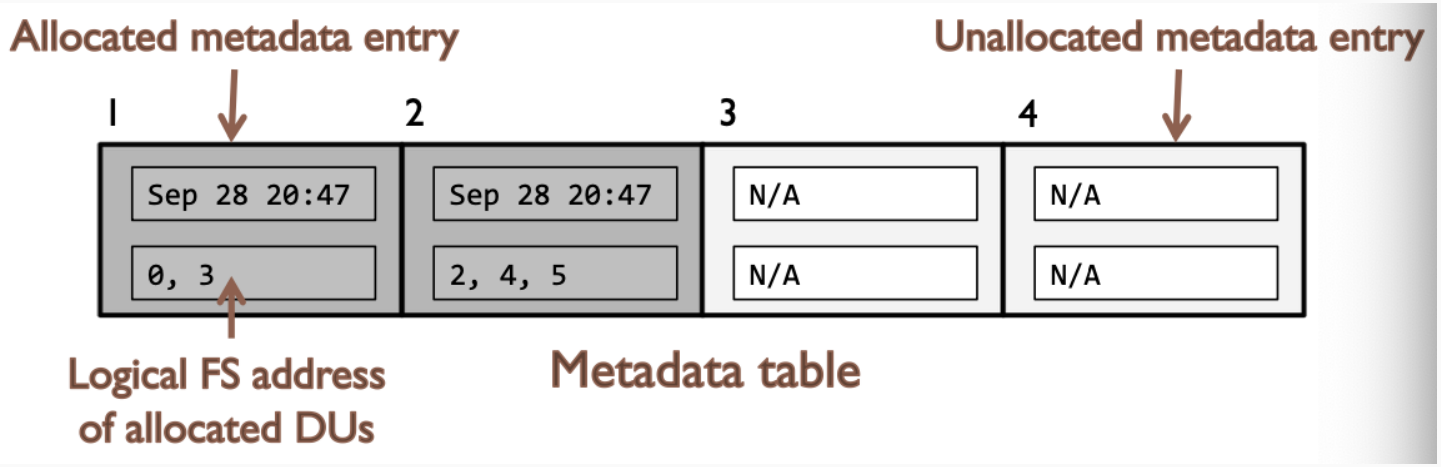
- Data unit viewing
- Logical file system-level searching
- Unallocated data unit searching
- Consistency analysis
- Data carving (next class)

File system evidence: Metadata category



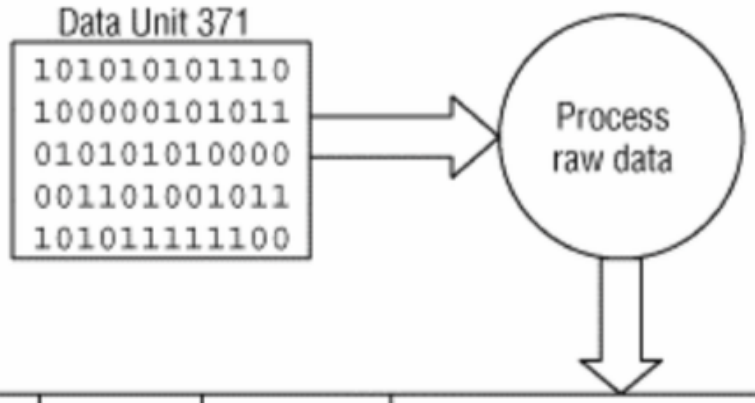
Evidence in the metadata category

- The metadata category contain the descriptive data
  - e.g., last accessed time, addresses of DUs allocated to a file and each entry has an address
- Many metadata structures are stored in a fixed or dynamic-length table, and each entry has an address
- In ExtX, evidence in this category includes: inodes and inode bitmap



1. Metadata lookup

- We found the name of a file that points to a specific metadata and we want to learn about the file
- We just need to locate the metadata and process it
  - e.g., interpret metadata located in DU 371 and shows two metadata entries: a deleted file, and an allocated directory



Del	Type	Size	Last Written	Data Unit 1	Data Unit 2
Yes	File	1,389	Jan 03, 2004 03:12:03	300,140	0
No	Dir	315,66 8	Jan 03, 2004 03:12:15	300,147	300,148

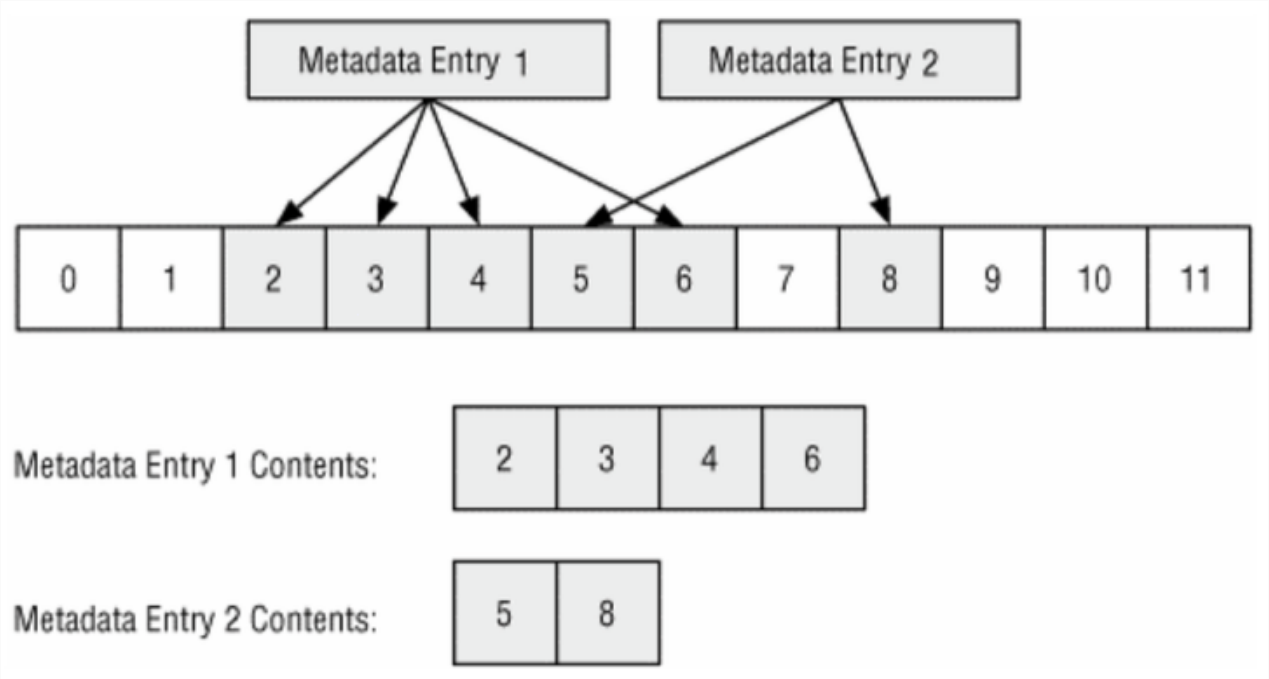
Example of istat

- Metadata of file associated with inode #16
  - The file is 10MB
  - Required four indirect blocks to point to all the allocated blocks

```
 #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
16880 16881 16882 16883

Indirect Blocks:
14392 15417 15418 16443
```

2. Logical file viewing

- After we look up the metadata for a file, we can view the file contents by reading the DUs allocated to the file
  - We do this when searching for evidence in the content of a file
  - This process occurs in the metadata and content categories
- During this process, we need to keep slack space in mind



Example of icat

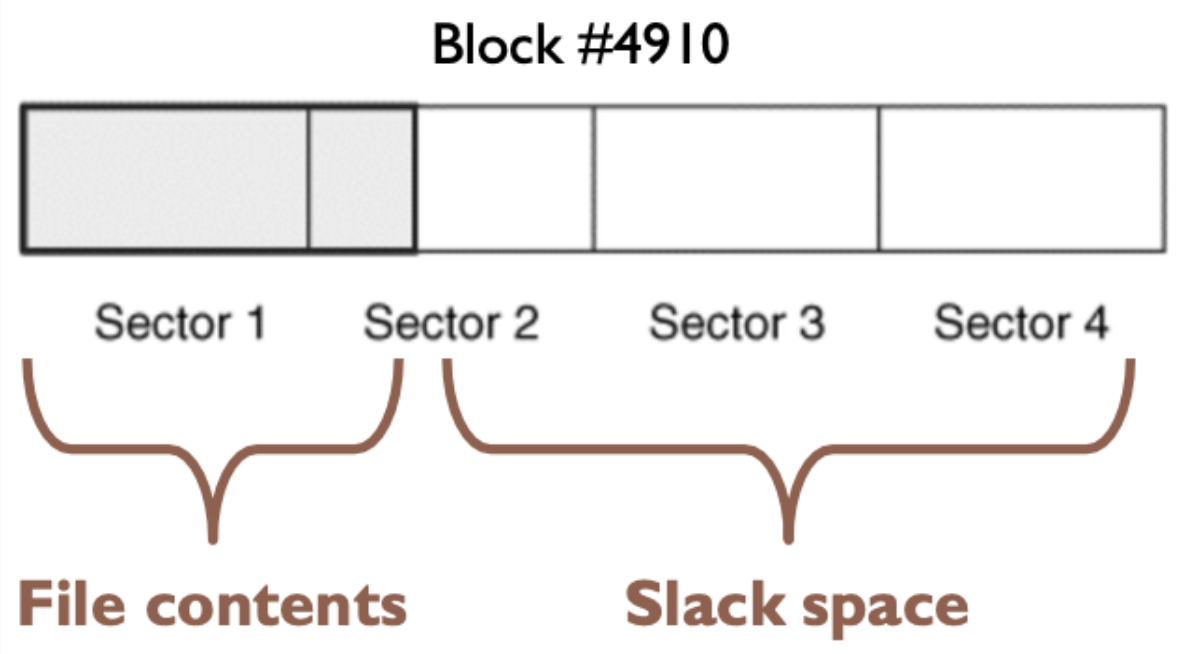
- The `icat` tool allows you to view the contents of the data units that are allocated to a metadata structure
  - If the `-s` flag is given, the slack space is shown
  - If the `-r` flag is given, it attempts to recover deleted files

```
 #icat -f linux-ext3 ext3.dd 69457 | xxd
0000000:  510f 0100 0c00 0102 2e00 0000 00d0 0000 Q...............
0000016:  0c00 0202 2e2e 0000 520f 0100 2800 0b01 ........R...(...
0000032:  6162 6364 6566 672e 7478 7400 530f 0100 abcdefg.txt.S...
0000048:  1400 0c01 6669 6c65 2074 776f 2e64 6174 ....file two.dat
0000064:  540f 0100 1000 0702 7375 6264 6972 3100 T.......subdir1.
0000080:  550f 0100 b003 0801 5253 5455 5657 5859 U.......RSTUVWXY
0000096:  0000 0000 0000 0000 0000 0000 0000 0000 ................
```

- Lists the raw contents of file associated with inode #69457
  - This is a directory

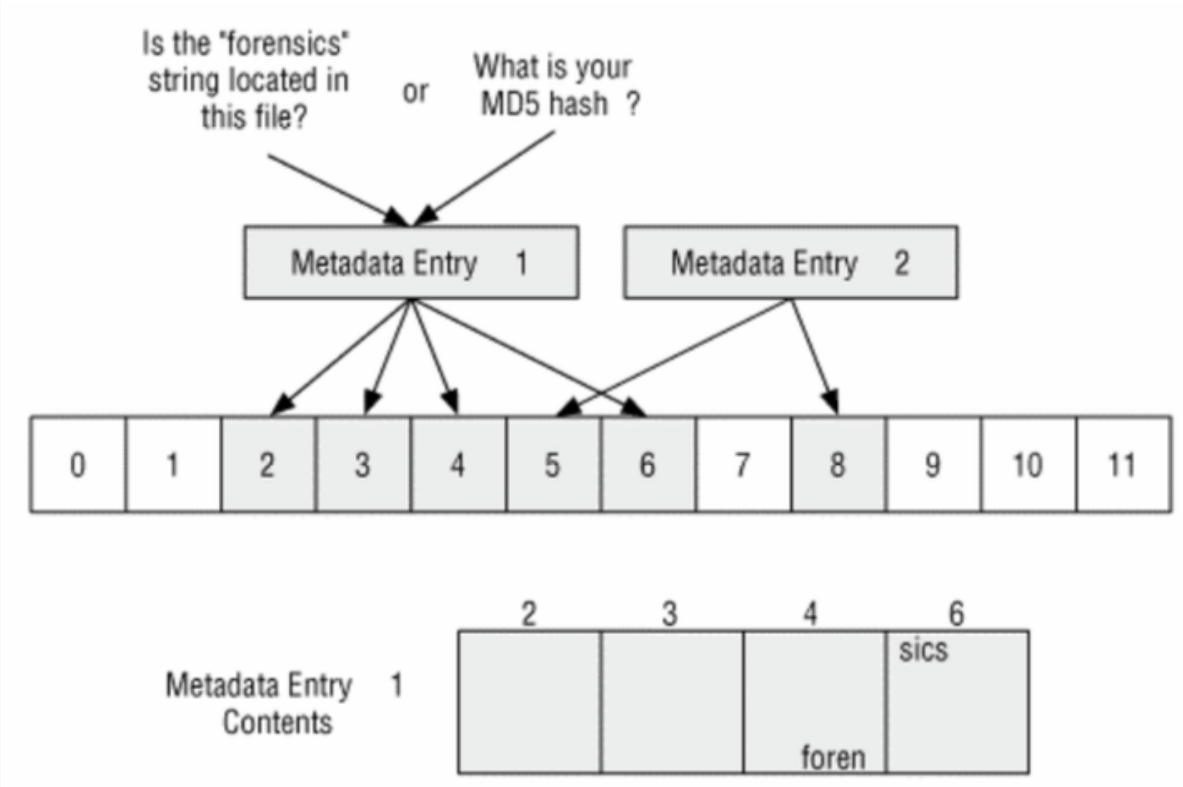
DU's slack space

- A file must allocate a full DU, even if it needs part of it
- The unused bytes in the last DU are called slack space
  - If unused bytes are not wiped, they may contain data from previous files (or memory)



3. Logical file searching

- Previous technique assumed you had a specific file to inspect
- Oftentimes, we need to find a file based on its content
  - e.g., we want all files including the term “forensics”
- That is when we use a logical file search



4. (Un)allocated metadata analysis

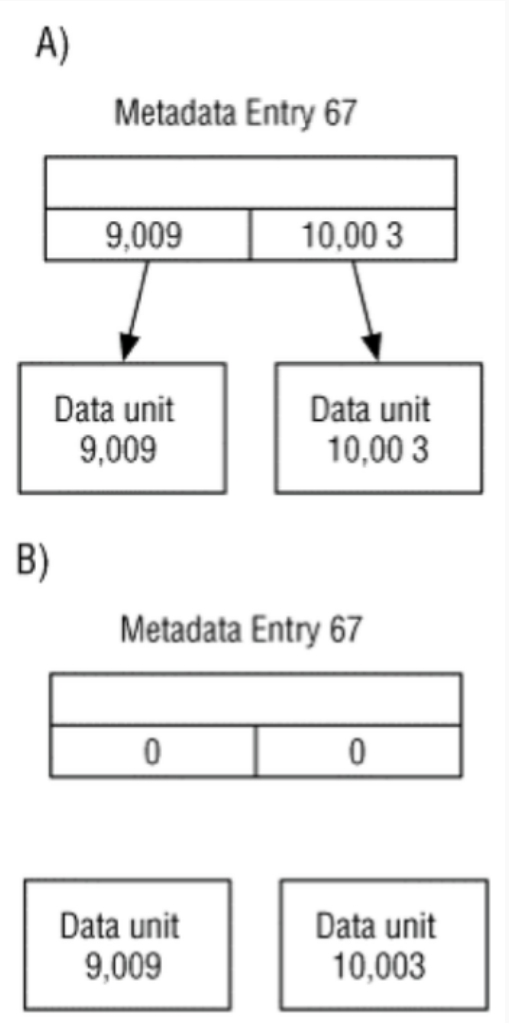
- When searching for deleted content, your evidence could be sitting in an unallocated metadata entry
  - You cannot see it because it no longer has a name
  - Some tools can list the unallocated entries for you
- The example below lists the allocated (-a) inodes within a given range (grep to select directories only)

```
 #ils -f linux-ext3 -m -a ext3-8.dd 32577-48864 | grep "Id"
<ext 3-8.dd-alive-32577>|0|32577|16893|drwxrwxr-x |4 |500|500|0|4096|
<ext3-8.dd-alive-32655>|0|32655|16893|drwxrwxr-x|2|500|500|0|4096|
<ext3-8.dd-alive-32660>|0|32660|16877|drwxr-xr-x |2|500|500|0|4096|
```

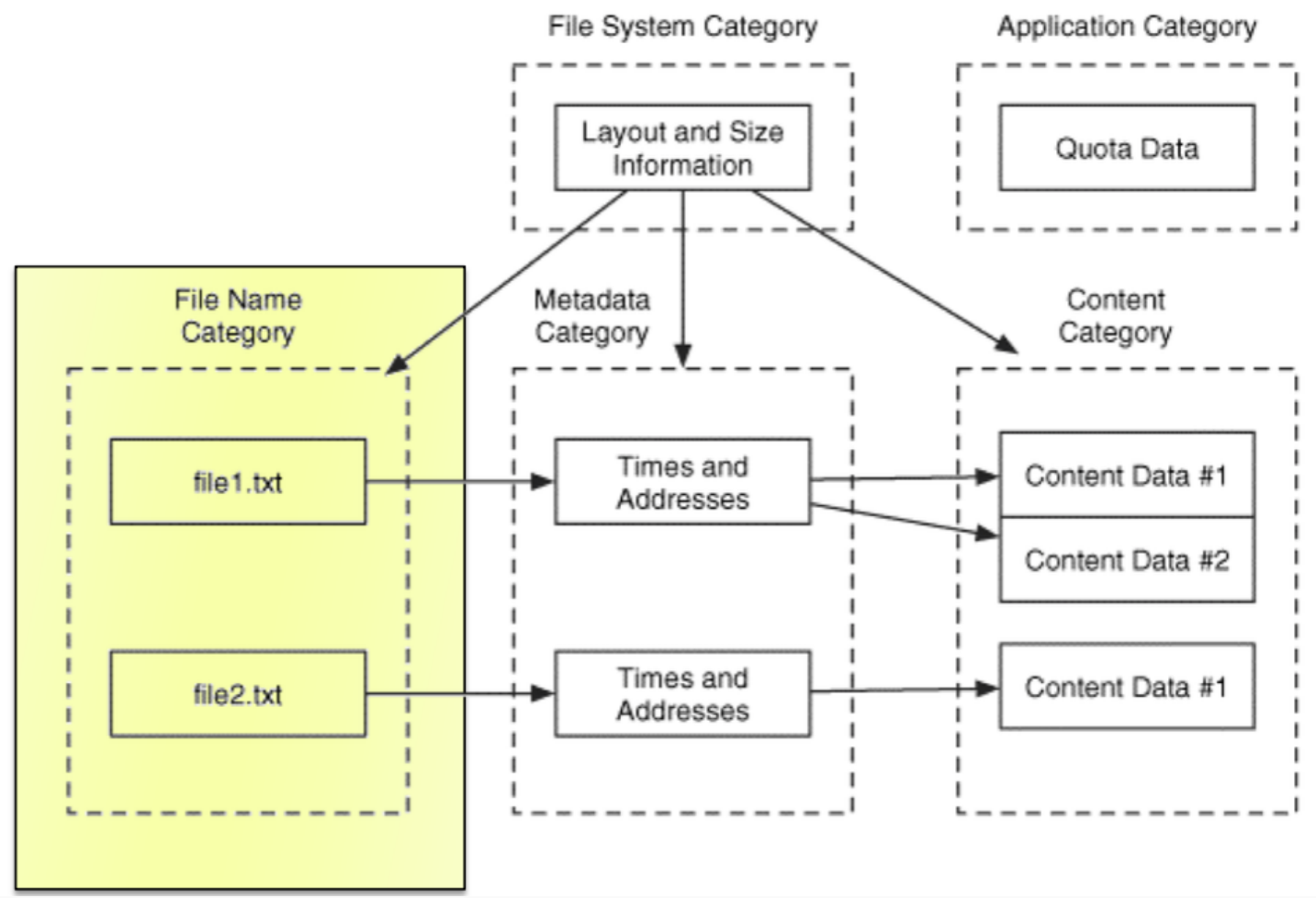
- Lists the raw contents of file associated with inode #69457
  - This is a directory

Deleted file recovery

- In some cases, you might want to search for evidence in deleted files
- A major method is metadata-based
  - Metadata-based recovery works when metadata from the deleted file still exists
  - Does not work if the metadata was wiped or reallocated to a new file
- Note: metadata and data units can become out of sync because the data units are allocated to new files



File system evidence: File name category

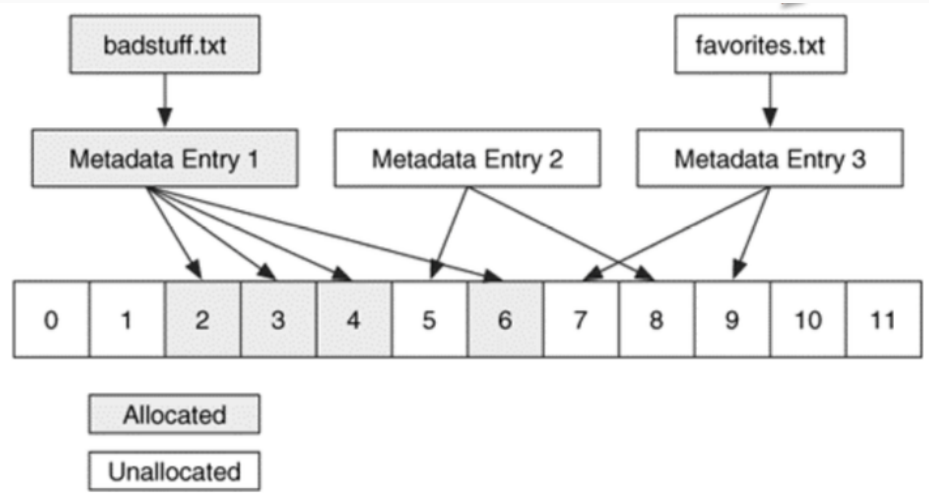


Evidence in the file name category

- Includes the names of files and allows the user to refer to a file by its name instead of its metadata address
  - In other words: we are analyzing directories
- In ExtX, evidence in this category includes: directory entries
- An important part of file name analysis is to determine where the root directory is located, e.g., / in ExtX
  - Each file system has its own way of defining the location of the root directory

File system evidence: File name category

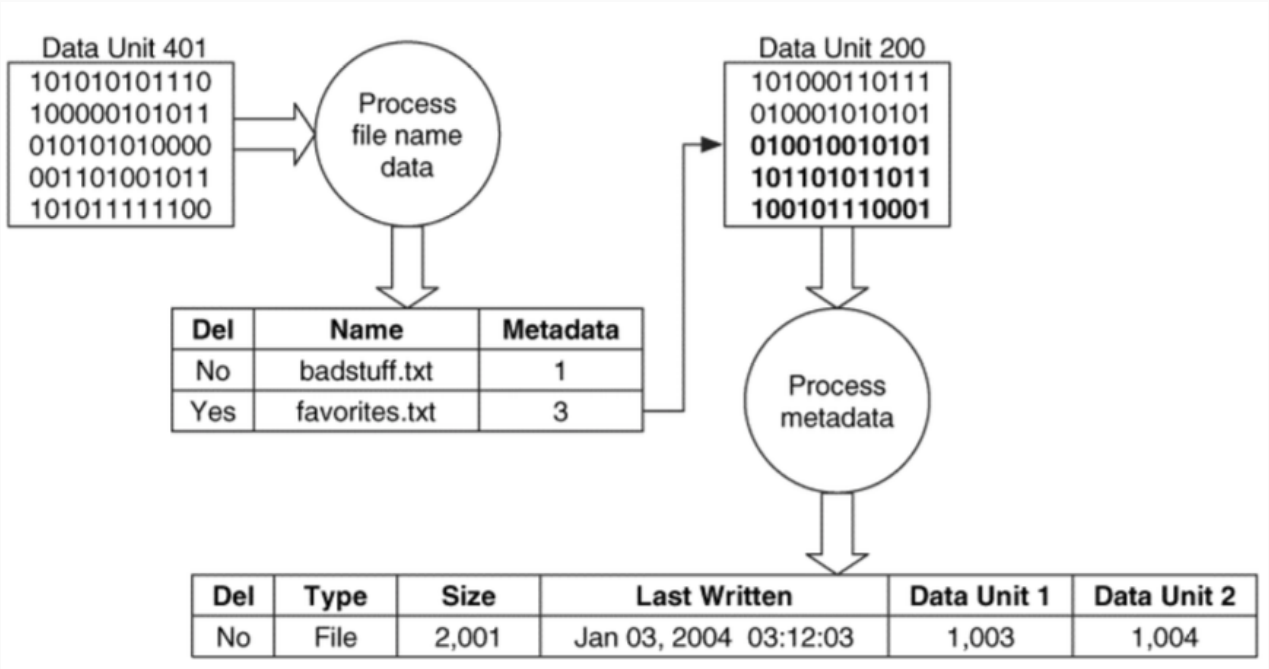
- With file name-based recovery, we use a deleted file name and its corresponding metadata address to recover the file content



- The `favorites.txt` file is deleted, and its name points to an unallocated metadata entry
- We can try to recover its contents using the metadata-based recovery techniques

1. File name listing

- List the names of the files and directories when searching for evidence based on name, path, or extension of a file
  - First locate the root directory of the file system and metadata
  - Then, obtain file list and corresponding metadata



Del	Name	Metadata
No	badstuff.txt	1
Yes	favorites.txt	3

Del	Type	Size	Last Written	Data Unit 1	Data Unit 2
No	File	2,001	Jan 03, 2004 03:12:03	1,003	1,004

2. File name searching

- Listing file names works well if we know what file we are looking for
- If we don't know the full file name, we can search for the part that we do know
    - e.g., based on the file's extension
- The process required to search for a name is similar to what we saw for file name listing

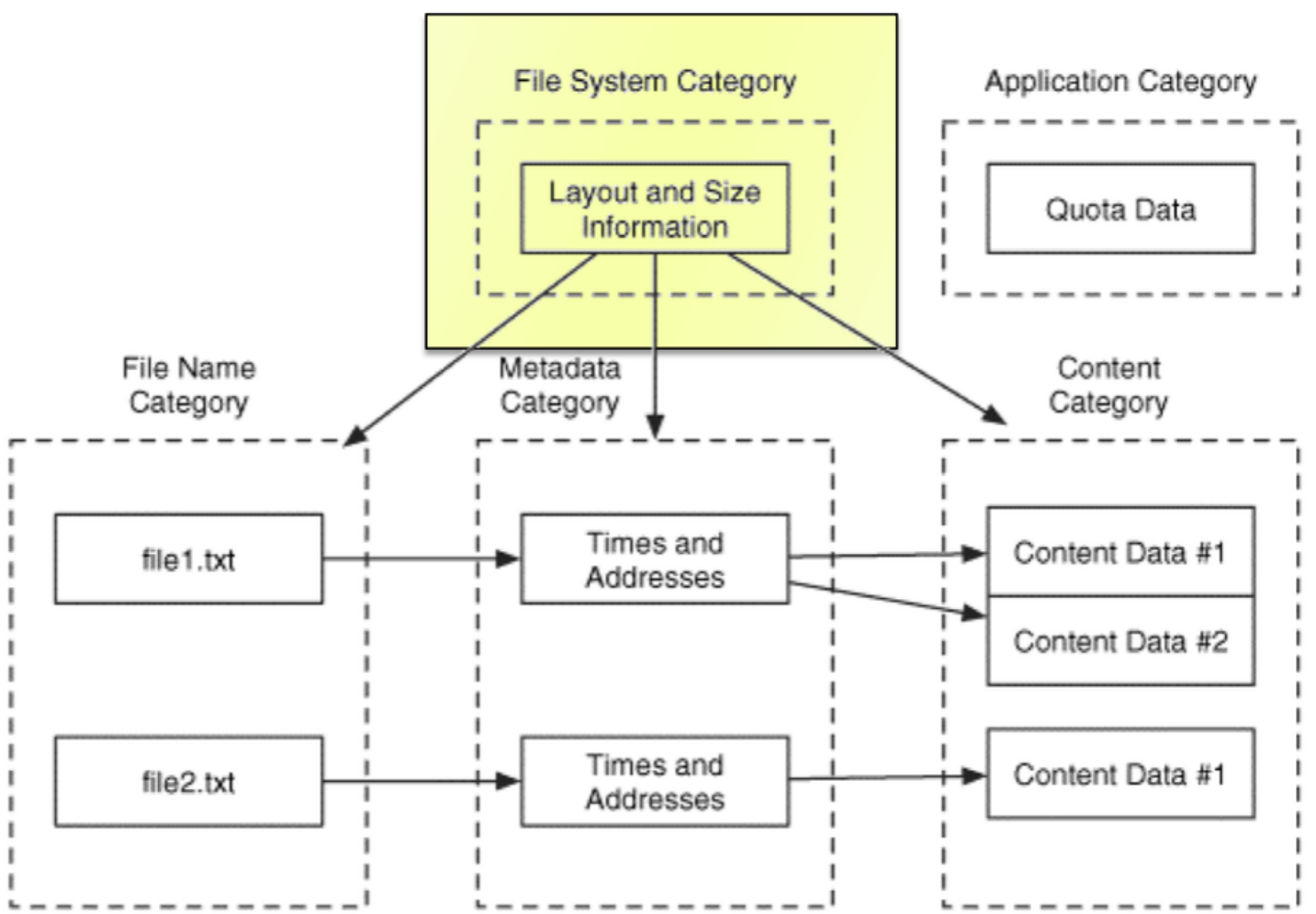
3. Consistency checking

- Consistency checks for the file name data include verifying that all allocated names point to allocated metadata structures
- It is valid for some file systems to have multiple file names for the same file, and many of them implement this functionality by having more than one file name entry with the same metadata address

Summary of techniques in file name category

- File name listing
- File name searching
- Consistency checking

File system evidence: File system category



Example of fsstat

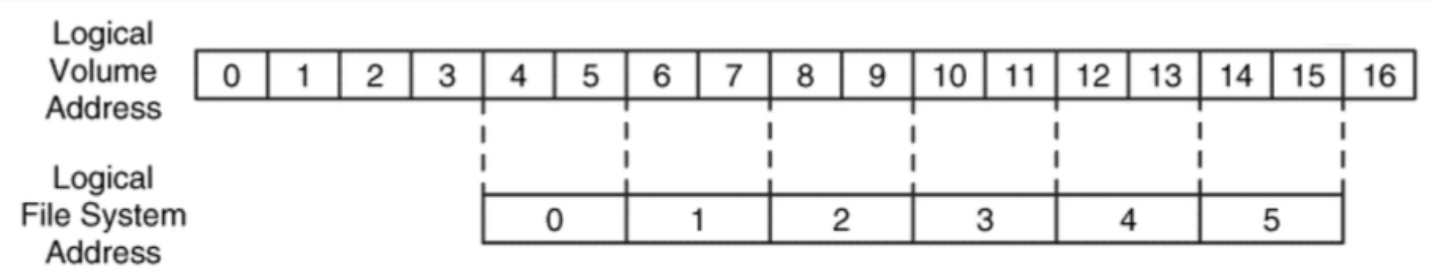
- Output of `fsstat` applied to Ext3 file system image

```
 #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
Last Mounted at:  Wed Aug 4 09:41:13 2004
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

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%)
```

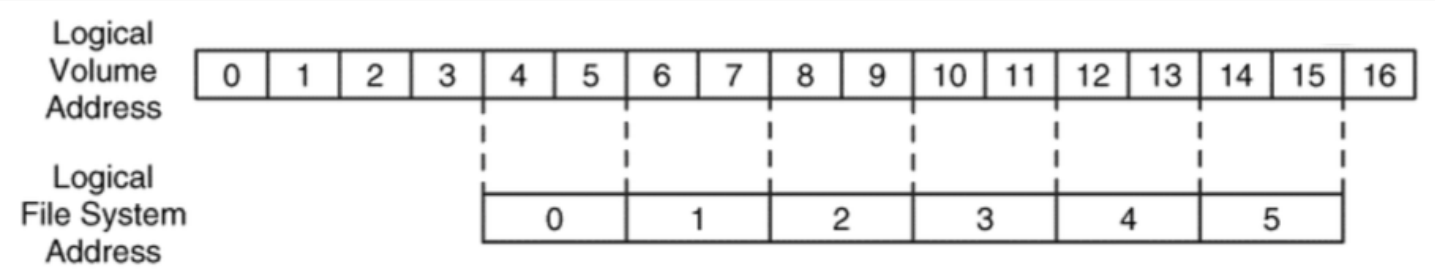
The volume slack

- The data structures in this category frequently have unused values that can be used to hide small amounts of data
- A consistency check in this category is to compare the size of the file system with the size of the volume in which it is located
- If the volume is larger, the sectors after the file system are called volume slack and could be used to hide data

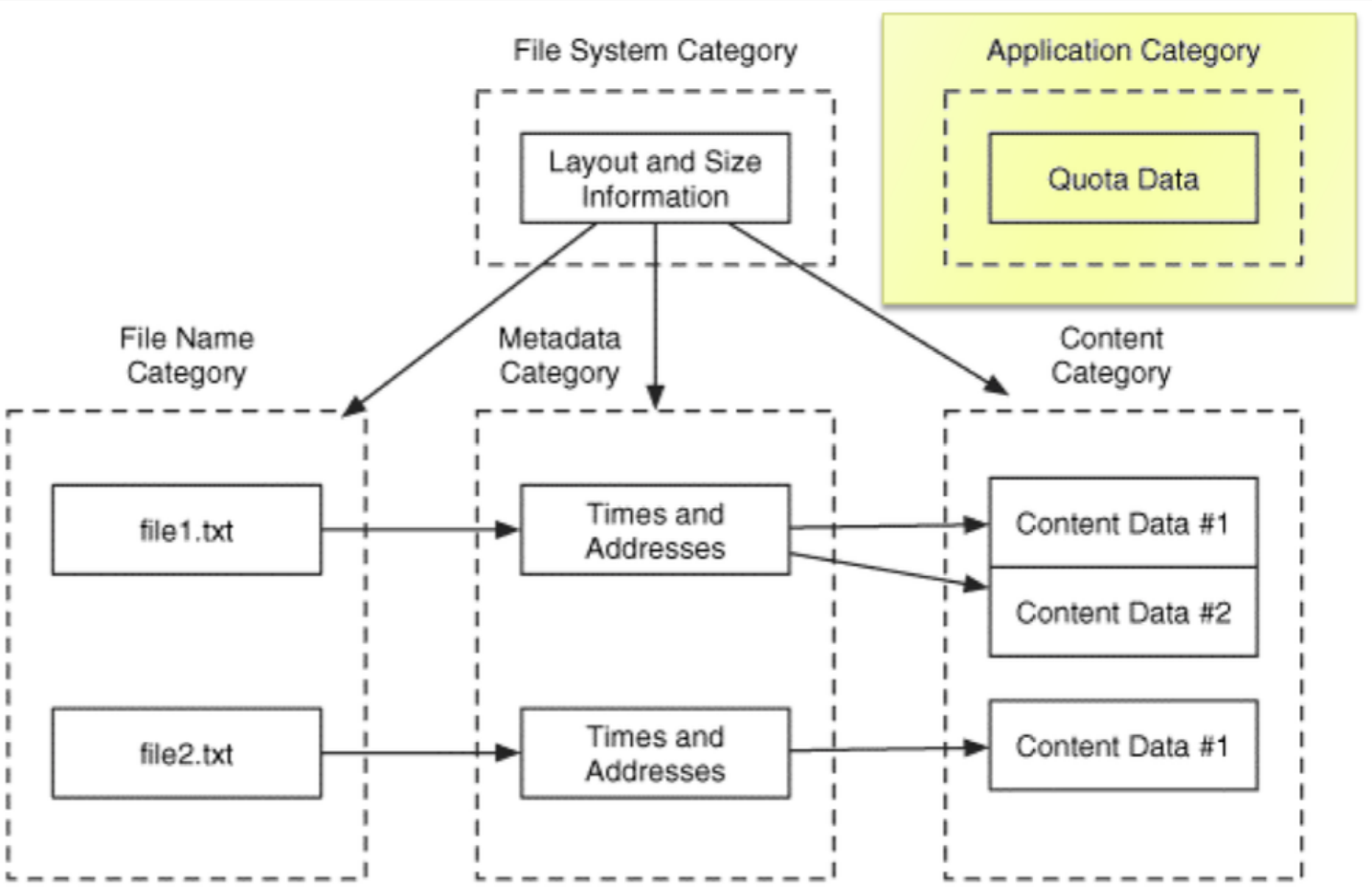


Evidence in damaged data units

- Older hard disks did not have the capability to handle errors: writing data to bad sectors could result in data loss
- To prevent this, FS allow DUs to be marked as bad: The OS marks bad DUs, and prevents their allocation to a file
- Modern hard disks can detect a bad sector and replace it with a spare, so the FS functionality is not needed
- Still, it is a great place to hide data: a user could manually add a DU to the damaged list and place data in it



File system evidence: Application category



Evidence in the application category

- Some file systems contain data that belongs in the application category
  - e.g., Acme Software decided that its OS would be faster if an area of the FS were reserved for an address book
- These data are not essential to the FS, and exist as special FS data instead of living inside a normal file
- One of the most common application category features is called journaling

Summary of TSK tools

Tool Prefix	Layer/Function	Tools
disk	(Disk Tools)	disk_sreset, diskstat
img	(Image File Tools)	img_stat
mm	(Media Management Tools)	mmls
fs	File System Layer	fsstat
j	(File System Journal Tools)	jcat, jls
i	Metadata/inode Layer	ils, icat, istat, ifind
d	Content/Data Layer	dls, dcat, dstat, dcalc
f	Human Interface/File Layer	fls, ffind

Takeaways

- File systems play a fundamental role in preserving huge amounts of evidence
- To analyze such evidence, there are several general techniques that can be applied based on the data category of the file system
- A real investigation normally requires chaining several of such techniques

Pointers

- **Textbook:**
  - Carrier – Chapter 8
- **Acknowledgements:**
  - Slides adapted from Nuno Santos's Forensics Cyber-Security course at Técnico Lisbon