Edinburgh Napier UNIVERSITY

CSN08101
Digital Forensics
Lecture 8: File Systems

Module Leader: Dr Gordon Russell
Lecturers: Robert Ludwiniak

Edinburgh Napier UNIVERSITY

Objectives

- Investigative Process
  - Analysis Framework
- File Systems
  - FAT
  - NTFS
  - EXT2/EXT3

last week

Edinburgh Napier
UNIVERSITY

NTFS

Edinburgh Napier UNIVERSITY

File System: NTFS

- Master File table grows, never shrinks
- B-tree algorithm used for file tree
  - re-“balances” file system tree when tree changes
  - creating or deleting a file can cause entire tree to change and can overwrite nodes that were marked as free but still had information in them
- Lots of attributes on files, can be confusing (e.g., which access time is the “official” one to use)
  - most useful attributes are MAC times
- Master File Table (MFT)
  - Contains information about all files and directories
  - Each has at least one entry in the table

Edinburgh Napier UNIVERSITY

NTFS Features

- Logging
- Transaction-based
- File and folder permissions
- Disk quotas
- Reparse points (used to link files)
- Sparse file support
- Compression
- Encryption
- Alternate data streams

Edinburgh Napier UNIVERSITY

Sparse Files

- Clusters that contain all zeros aren't written to disk
- Analysis considerations
  - A deleted sparse file is hard to recover
  - If file system metadata is deleted or corrupted, a sparse file might not be recoverable

Edinburgh Napier UNIVERSITY

File Compression

- Data is broken into equal-sized compression units (e.g. 16 clusters)
- An attempt is made to compress each unit
- Parts of a file may be compressed while other parts aren't

Edinburgh Napier UNIVERSITY

File Compression Analysis Considerations

- A single file can use different compression methods (e.g. none, sparse, or variant of LZ77)
- Recovery tools need to support decompression
- A deleted compressed file is hard to recover
- If file system metadata is deleted or corrupted, a compressed file might not be recoverable

Edinburgh Napier UNIVERSITY

Encrypting File System (EFS)

- Uses both symmetric key encryption (DESX) and asymmetric key encryption (RSA)
- Generates a single file encryption key (FEK) and encrypts file with FEK using DESX
- Stores FEK with file

Edinburgh Napier UNIVERSITY

File Encryption Key Encryption

- FEK is encrypted with user's public key
- FEK is decrypted with user's private key
- If policy allows it, FEK is also encrypted with public key of recovery agent (and decrypted with private key of recovery agent)

Edinburgh Napier UNIVERSITY

EFS Analysis Considerations

- By default a user's private key is stored in the Windows registry, encrypted with login password as key
  - Login password is susceptible to brute force attack and private key might be compromised
- EFS creates a temporary file (EFS0.TMP) with plaintext data
  - Marks it as deleted when finished but doesn't actually erase contents

Edinburgh Napier UNIVERSITY

Alternate Data Streams

- Data added to a file
- Introduced to support Macintosh files that have a data and resource fork
- Almost impossible to detect with normal file browsing techniques
- A favourite of hackers and criminals

Edinburgh Napier UNIVERSITY

File System Metadata Files

- Files that store file system administrative data
- Do not confuse with file metadata
- First 16 MFT entries reserved for files that describe the file system listed in the root directory
- Each file begins with ‘$’

Edinburgh Napier UNIVERSITY

File System Metadata Files

Entry	File Name	Description
0	$MFT	MFT entry
1	$MFTMirr	Backup of the MFT
2	$LogFile	Contains journal information for metadata transactions
3	$Volume	Volume Information: label, identifier, version
4	$AttrDef	Attribute information: identifier values, name, sizes
5	.	Root directory of the files system
6	$Bitmap	Contains allocation status for each cluster
7	$Boot	Contains the boot code
8	$BadClus	Contains clusters that have bad sectors

Edinburgh Napier UNIVERSITY

Master File Table

- Contains information about all files and directories
- Every file and directory has at least one entry in the table
- Each entry is simple
  - 1 KB in size
  - Entry header is first 42 bytes
  - Remaining bytes store attributes

Edinburgh Napier UNIVERSITY

Resident and Non-Resident Attributes

- A resident attribute stores its content in the MFT entry
- A non-resident attribute stores its content in external clusters
- Non resident attributes are stored in cluster runs
- The attribute header gives the starting cluster address and its run length

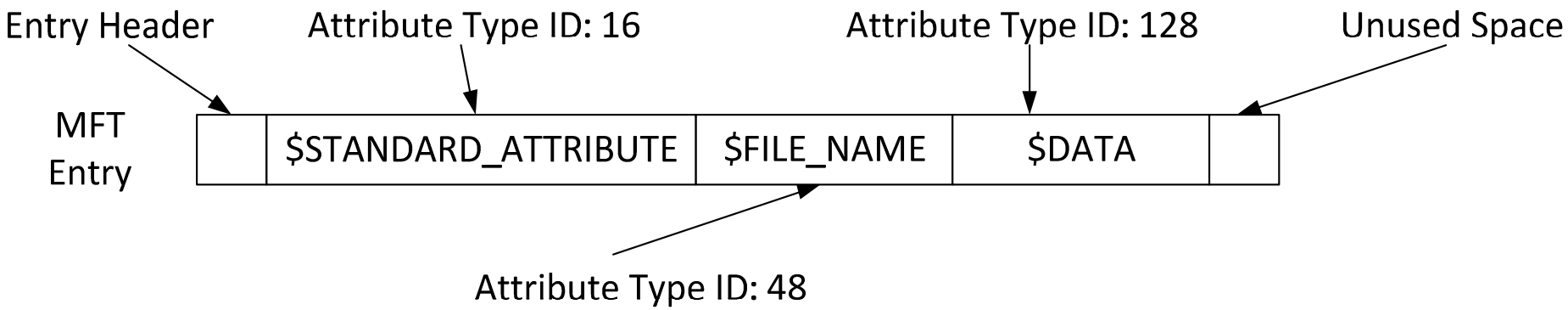
Edinburgh Napier UNIVERSITY

Data Structure Categories

	File System	Content	Metadata	File Name	Application
NTFS	$Boot, $Volume, $AttrDef	Clusters, $Bitmap	$MFT, $MFTMirr, $STANDARD_ INFORMATION, $DATA, $ATTRIBUTE_ LIST, $SECURITY_ DESCRIPTOR	$FILE_NAM E$IDX_ROO T, $IDX_ ALLOCATIO N, $BITMAP	Disk Quota, Journal, Change Journal

Edinburgh Napier UNIVERSITY

NTFS Record Layout



Edinburgh Napier UNIVERSITY

NTFS Record

- Each MFT record is addressed by a 48 bit MFT entry value.
- First entry has address 0.
- Each MFT entry has a 16 bit sequence number that is incremented when the entry is allocated.
- MFT entry value and sequence number combined yield 64b file reference address.

Edinburgh Napier UNIVERSITY

NTFS Record

- MFT entry attributes are loosely defined
- Each attribute is preceded by the attribute header
- The attribute header identifies
  - Type of attribute
  - Size
  - Name

Edinburgh Napier UNIVERSITY

MFT Record Structure

0x00-0x03: Magic Number: "FILE"

0x04-0x05: Offset to the update sequence.

0x06-0x07: Number of entries in fixup array

0x08-0x0f: $LogFile Sequence Number (LSN)

0x10-0x11: Sequence number

0x12-0x13: Hard link count

0x14-0x15: Offset to first attribute

Edinburgh Napier UNIVERSITY

MFT Record Structure

0x16-0x17: Flags: 0x01 record in use, 0x02 directory.
0x18-0x1B: Used size of MFT entry
0x1C-0x1F: Allocated size of MFT entry.
0x20-0x27: File reference to the base FILE record
0x28-0x29: Next attribute ID
0x2A-0x2B: (XP) Align to 4B boundary
0x2C-0x2F: (XP) Number of this MFT record
0x30-0x100: Attributes and fixup value

Edinburgh Napier UNIVERSITY

MFT Attribute Layout

- MFT Header is always the same:
  - 0x00 Attribute Type Identifier
  - 0x04 Length of Attribute
  - 0x08 non-resident flag
  - 0x09 length of name
  - 0x0a offset to name
  - 0x0c flags

Edinburgh Napier UNIVERSITY

MFT List of Possible Attributes

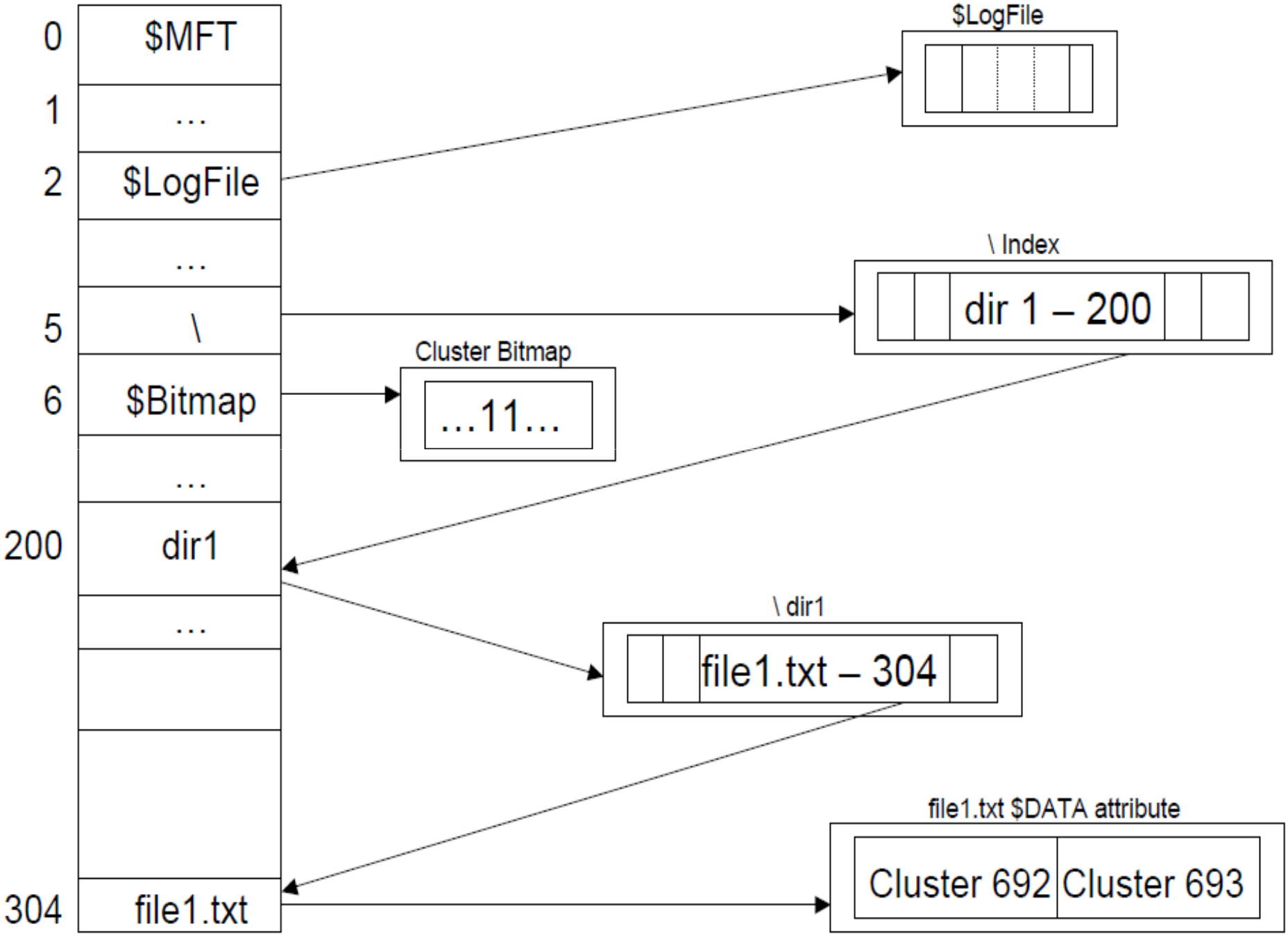
- Defined in $AttrDef entry of MFT, but default is:
  - 0x10 $STANDARD_INFORMATION
  - 0x20 $ATTRIBUTE_LIST
  - 0x30 $FILE_NAME0
  - X40 (NT) $VOLUME_VERSION (2K) $OBJECT_ID
  - 0x50 $SECURITY_DESCRIPTOR
  - 0x60 $VOLUME_NAME
  - 0x70 $VOLUME_INFORMATION
  - 0x80 $DATA
  - 0x90 $INDEX_ROOT
  - 0xA0 $INDEX_ALLOCATION
  - 0xB0 $BITMAP
  - 0xC0 (NT) $SYMBOLIC_LINK, (2K) $REPARSE_POINT
  - 0xD0 $EA_INFORMATION
  - 0xE0 $EA0xF0NT$PROPERTY_SET
  - 0x100(2K) $LOGGED_UTILITY_STREAM

Edinburgh Napier UNIVERSITY

MFT Attribute Example

Standard Info Attribute Layout

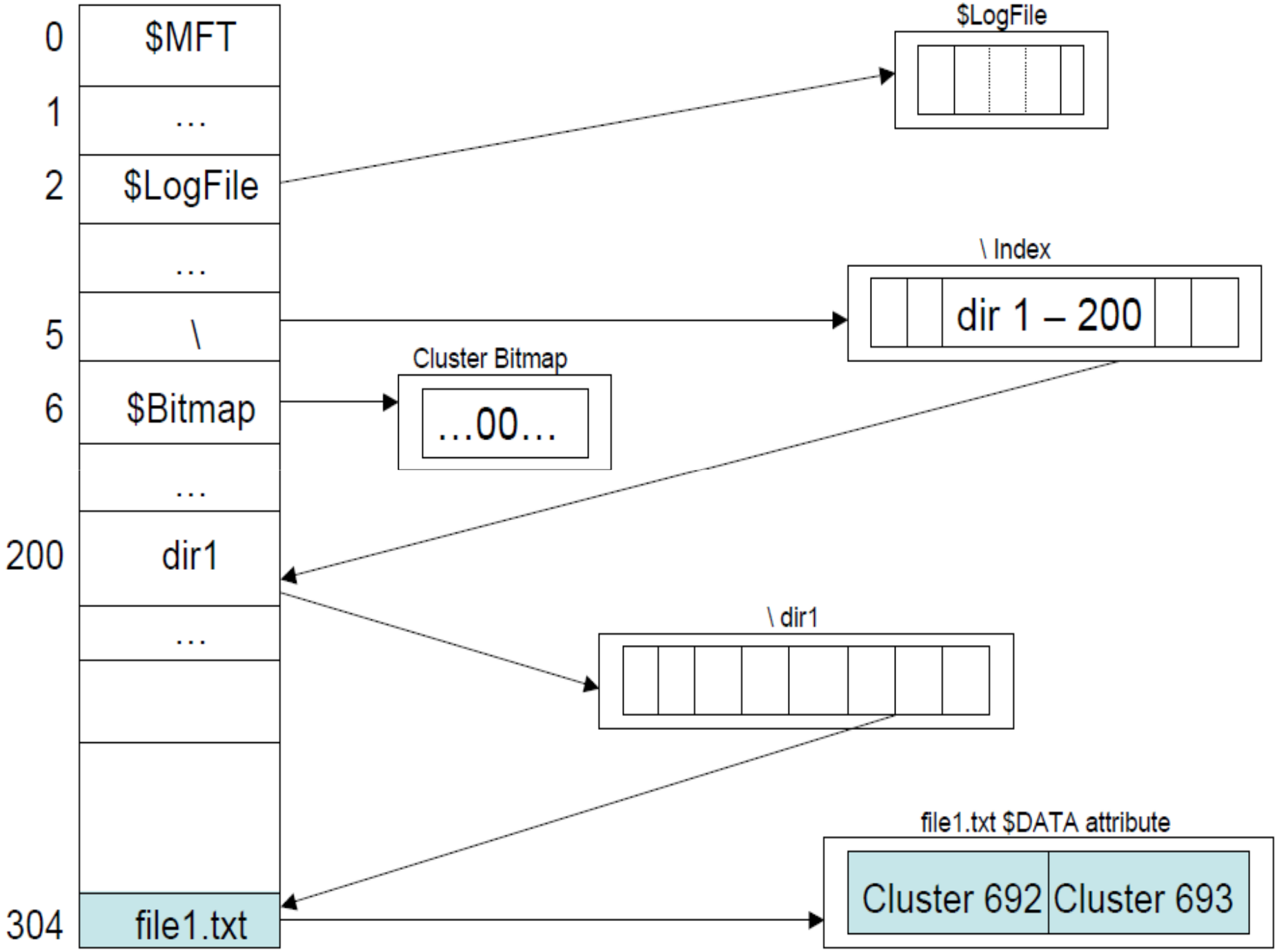
0x00	8		File Creation Time
0x08	8		File Alteration Time
0x10	8		MFT Change
0x18	8		File Read Time
0x20	4		DOS File Permissions
0x24	4		Maximum number of versions
0x28	4		Version number
0x2C	4		Class ID
0x30	4	2K	Owner ID


0
$MFT
1
…
2
$LogFile
…
5
\
6
$Bitmap
…
200
dir1
…
304
file1.txt
$LogFile
\ Index
dir 1 – 200
Cluster Bitmap
…11…
\ dir1
file1.txt – 304
file1.txt $DATA attribute
Cluster 692
Cluster 693

Creating File

Edinburgh Napier UNIVERSITY

1. Read volume boot sector to locate MFT.
2. Read first entry in MFT to determine layout of MFT.
3. Allocate an MFT entry for the new file.
4. Initialize MFT entry with $STANDARD_INFORMATION, etc.
5. Check MFT $Bitmap to find free clusters, using best-fit algorithm.
6. Set corresponding $Bitmap bits to 1.
7. Write file content to clusters and update $DATA attribute with starting address of cluster run and run length.
8. Read root directory (MFT entry 5), traverse index, and find dir1.
9. Read $INDEX_ROOT attribute for dir1 and determine where file1.txt should go.
10. Create new index entry; resort index tree.
11. Enter steps in $LogFile (as each step is take


0
$MFT
1
…
2
$LogFile
…
5
\
6
$Bitmap
…
200
dir1
…
304
file1.txt
$LogFile
\ Index
dir 1 – 200
Cluster Bitmap
…00…
\ dir1
file1.txt $DATA attribute
Cluster 692
Cluster 693

Edinburgh Napier UNIVERSITY

Deleting File

1. Read volume boot sector to locate MFT.
2. Read first entry in MFT to determine layout of MFT.
3. Read root directory (MFT entry 5), traverse index, and find dir1.
4. Read $INDEX_ROOT for dir1 entry and find file1.txt entry.
5. Remove filename entry from index; move other entries over.
6. Set MFT $Bitmap entries to 0.
7. Enter steps in $LogFile (as each step is taken).

fsstat - display general details of a file system

Edinburgh Napier UNIVERSITY

```
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: NTFS
Volume Serial Number: 2A040574040543F3
OEM Name: NTFS
Version: Windows XP

METADATA INFORMATION
--------------------------------------------
First Cluster of MFT: 324570
First Cluster of MFT Mirror: 486856
Size of MFT Entries: 1024 bytes
Size of Index Records: 4096 bytes
Range: 0 - 14519
Root Directory: 5

CONTENT INFORMATION
--------------------------------------------
Sector Size: 512
Cluster Size: 2048
Total Cluster Range: 0 - 973711
Total Sector Range: 0 - 3894847
```

fsstat (continue)

Edinburgh Napier UNIVERSITY

$AttrDef Attribute Values:

$STANDARD_INFORMATION (16) Size: 48-72 Flags: Resident
$ATTRIBUTE_LIST (32) Size: No Limit Flags: Non-resident
$FILE_NAME (48) Size: 68-578 Flags: Resident,Index
$OBJECT_ID (64) Size: 0-256 Flags: Resident
$SECURITY_DESCRIPTOR (80) Size: No Limit Flags: Non-resident
$VOLUME_NAME (96) Size: 2-256 Flags: Resident
$VOLUME_INFORMATION (112) Size: 12-12 Flags: Resident
$DATA (128) Size: No Limit Flags:
$INDEX_ROOT (144) Size: No Limit Flags: Resident
$INDEX_ALLOCATION (160) Size: No Limit Flags: Non-resident
$BITMAP (176) Size: No Limit Flags: Non-resident
$REPARSE_POINT (192) Size: 0-16384 Flags: Non-resident
$EA_INFORMATION (208) Size: 8-8 Flags: Resident
$EA (224) Size: 0-65536 Flags:
$LOGGED_UTILITY_STREAM (256) Size: 0-65536 Flags: Non-resident

fls - list file and directory names

Edinburgh Napier UNIVERSITY

```
r/r 4-128-4:    $AttrDef
r/r 8-128-2:    $BadClus
r/r 8-128-1:    $BadClus:$Bad
r/r 6-128-1:    $Bitmap
r/r 7-128-1:    $Boot
d/d 11-144-4:   $Extend
r/r 2-128-1:    $LogFile
r/r 0-128-1:    $MFT
r/r 1-128-1:    $MFTMirr
r/r 9-128-8:    $Secure:$SDS
r/r 9-144-11:   $Secure:$SDH
r/r 9-144-14:   $Secure:$SII
r/r 10-128-1:   $UpCase
r/r 3-128-3:    $Volume
r/r 6679-128-1: AUTOEXEC.BAT
r/r 3052-128-3: boot.ini
r/r 5505-128-1: CONFIG.SYS
d/d 3058-144-6: Documents and Settings
r/r 3039-128-1: hiberfil.sys
r/r 13377-128-3:        hpfr5550.log
r/r 6680-128-1: IO.SYS
r/r 6681-128-1: MSDOS.SYS
r/r 3020-128-3: NTDETECT.COM
r/r 3016-128-3: ntldr
r/r 27-128-1:   pagefile.sys
d/d 3457-144-6: Program Files
d/d 12728-144-1:        RECYCLER
d/d 7611-144-1: System Volume Information
d/d 28-144-6:   WINDOWS
d/d 14519:      $OrphanFiles
```

Edinburgh Napier UNIVERSITY

istat - Display details of a meta-data structure

```
MFT Entry Header Values:
Entry: 5505         Sequence: 11
$LogFile Sequence Number: 15634044
Allocated File
Links: 1

$STANDARD_INFORMATION Attribute Values:
Flags: Archive
Owner ID: 0
Security ID: 258  ()
Created:        Mon Oct 21 16:56:44 2002
File Modified:  Mon Oct 21 16:56:44 2002
MFT Modified:   Mon Oct 21 16:56:44 2002
Accessed:       Mon Oct 21 16:56:44 2002
```

Edinburgh Napier UNIVERSITY

istat - Display details of a meta-data structure

```
$FILE_NAME Attribute Values:
Flags: Archive
Name: CONFIG.SYS
Parent MFT Entry: 5     Sequence: 5
Allocated Size: 0       Actual Size: 0
Created:        Mon Oct 21 16:56:44 2002
File Modified:  Mon Oct 21 16:56:44 2002
MFT Modified:   Mon Oct 21 16:56:44 2002
Accessed:       Mon Oct 21 16:56:44 2002

Attributes:
Type: $STANDARD_INFORMATION (16-0)   Name: N/A   Resident   size: 72
Type: $FILE_NAME (48-2)   Name: N/A   Resident   size: 86
Type: $DATA (128-1)   Name: N/A   Resident   size: 0
```

icat - Output the contents of a file based on its inode number

```
0000000: 4649 4c45 3000 0300 2497 2b02 0000 0000  FILE0...$.+.....
0000010: 0100 0100 3800 0100 9801 0000 0004 0000  ....8...........
0000020: 0000 0000 0000 0000 0600 0000 0000 0000  ................
0000030: f000 0000 0000 0000 1000 0000 6000 0000  ............`...
0000040: 0000 1800 0000 0000 4800 0000 1800 0000  ........H.......
0000050: c08e 3200 e478 c201 c08e 3200 e478 c201  ..2..x....2..x..
0000060: c08e 3200 e478 c201 c08e 3200 e478 c201  ..2..x....2..x..
0000070: 0600 0000 0000 0000 0000 0000 0000 0000  ................
0000080: 0000 0000 0001 0000 0000 0000 0000 0000  ................
0000090: 0000 0000 0000 0000 3000 0000 6800 0000  ........0...h...
00000a0: 0000 1800 0000 0300 4a00 0000 1800 0100  ........J.......
00000b0: 0500 0000 0000 0500 c08e 3200 e478 c201  ..........2..x..
00000c0: c08e 3200 e478 c201 c08e 3200 e478 c201  ..2..x....2..x..
00000d0: c08e 3200 e478 c201 0040 0000 0000 0000  ..2..x...@......
00000e0: 0040 0000 0000 0000 0600 0000 0000 0000  .@..............
00000f0: 0403 2400 4d00 4600 5400 0000 0000 0000  ..$.M.F.T.......
0000100: 8000 0000 4800 0000 0100 4000 0000 0100  ....H.....@.....
0000110: 0000 0000 0000 0000 5f1c 0000 0000 0000  ........_.......
0000120: 4000 0000 0000 0000 0000 e300 0000 0000  @...............
0000130: 00dc e200 0000 0000 00dc e200 0000 0000  ................
0000140: 3260 1cda f304 0080 b000 0000 4800 0000  2`..........H...
0000150: 0100 4000 0000 0500 0000 0000 0000 0000  ..@.............
0000160: 0000 0000 0000 0000 4000 0000 0000 0000  ........@.......
0000170: 0008 0000 0000 0000 1807 0000 0000 0000  ................
0000180: 1807 0000 0000 0000 3101 d9f3 0400 0000  ........1.......
0000190: ffff ffff 0000 0000 006c 0000 0000 0000  .........l......
00001a0: 006c 0000 0000 0000 3110 daf3 0400 0000  .l......1.......
00001b0: b000 0000 4800 0000 0100 4000 0000 0500  ....H.....@.....
00001c0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00001d0: 4000 0000 0000 0000 0008 0000 0000 0000  @...............
00001e0: 0800 0000 0000 0000 0800 0000 0000 0000  ................
00001f0: 3101 d9f3 0400 0000 ffff ffff 0000 f000  1...............
```

Edinburgh Napier
UNIVERSITY

EXT2/EXT3

Edinburgh Napier UNIVERSITY

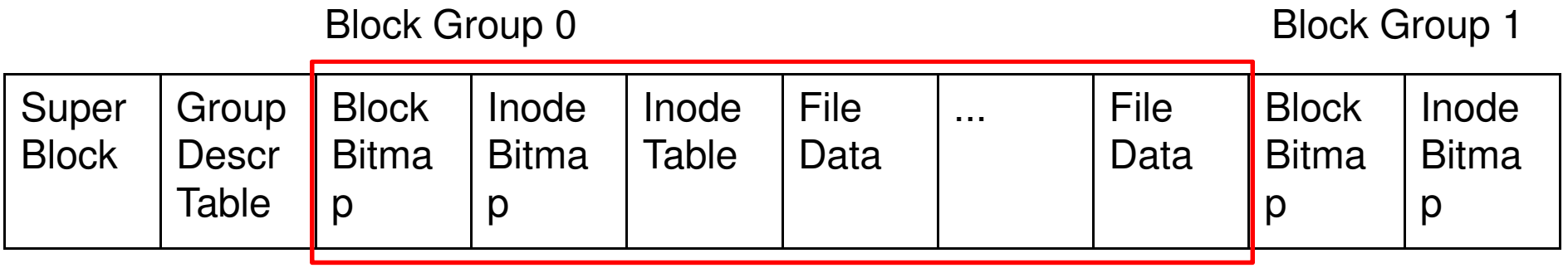
Introduction

- Ext2 and Ext3 are the default Linux file system.
- Ext3 is the new version of Ext2 and adds journaling mechanism, but the basic structures are the same.
- The metadata is stored throughout the file system, and the metadata which is associated with a file are stored “near” it.

Edinburgh Napier UNIVERSITY

File System Layout

- The whole area is divided into several block groups, and block groups contains several blocks.
- A block group is used to store file metadata and file content



Edinburgh Napier UNIVERSITY

Metadata Concepts

- Superblock:
  - The Ext2 superblock is located 1024 bytes from the start of the file system and is 1024 bytes in size. (The first two sectors are used to store boot code if necessary)
  - Back up copies are typically stored in the first file data block of each block group
  - It contains basic information of the file system, such as the block size, the total number of blocks, etc.

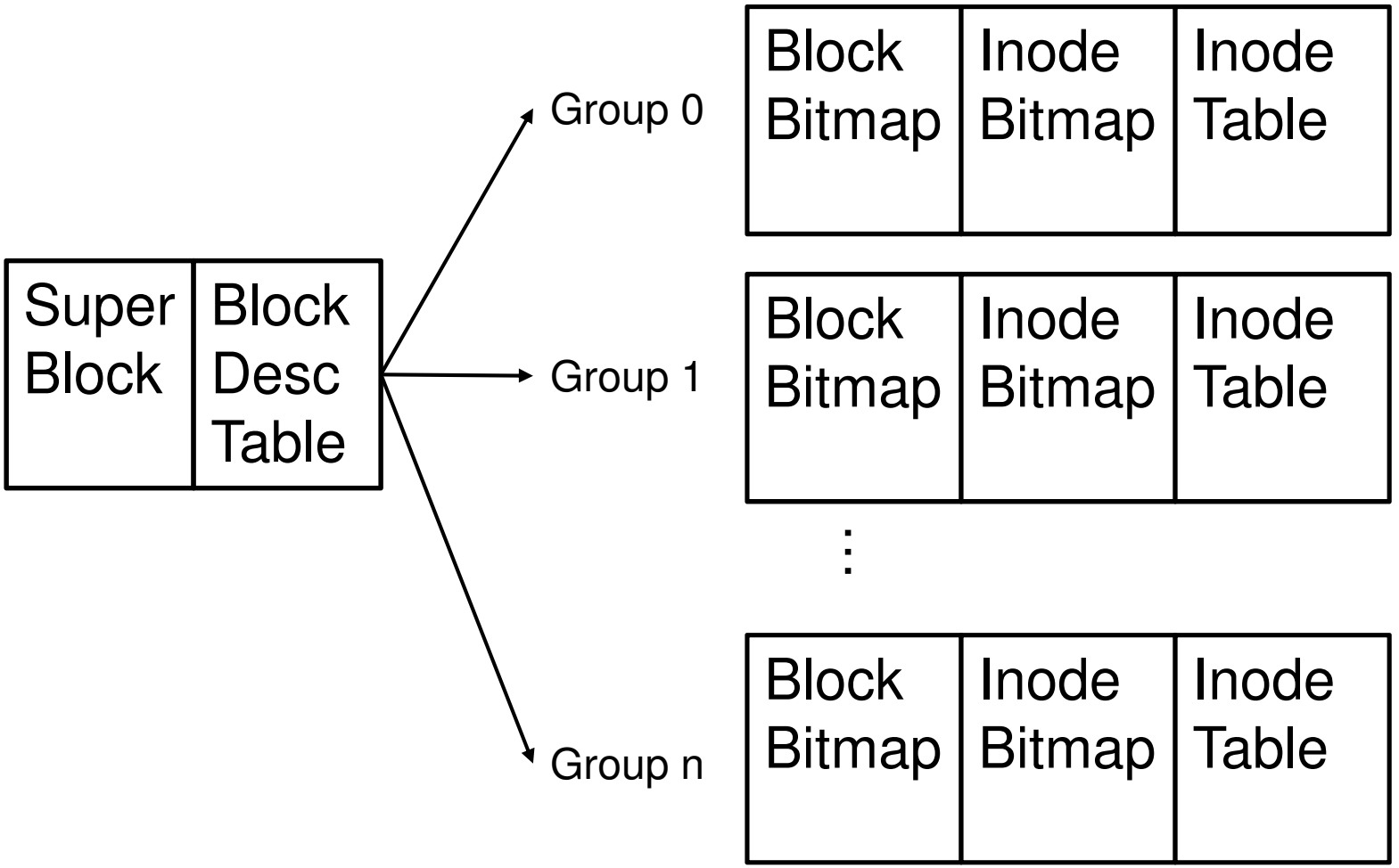
Edinburgh Napier UNIVERSITY

Metadata Concepts

- Block Group Descriptor Table:
  - It contains a group descriptor data structure for every block group.
  - The group descriptor stores the address of block bitmap and inode bitmap for the block group.
- Bitmaps:
  - The block bitmap manages the allocation status of the blocks in the group.
  - The inode bitmap manages the allocation status of the inodes in the group.

Edinburgh Napier UNIVERSITY

Metadata Concepts



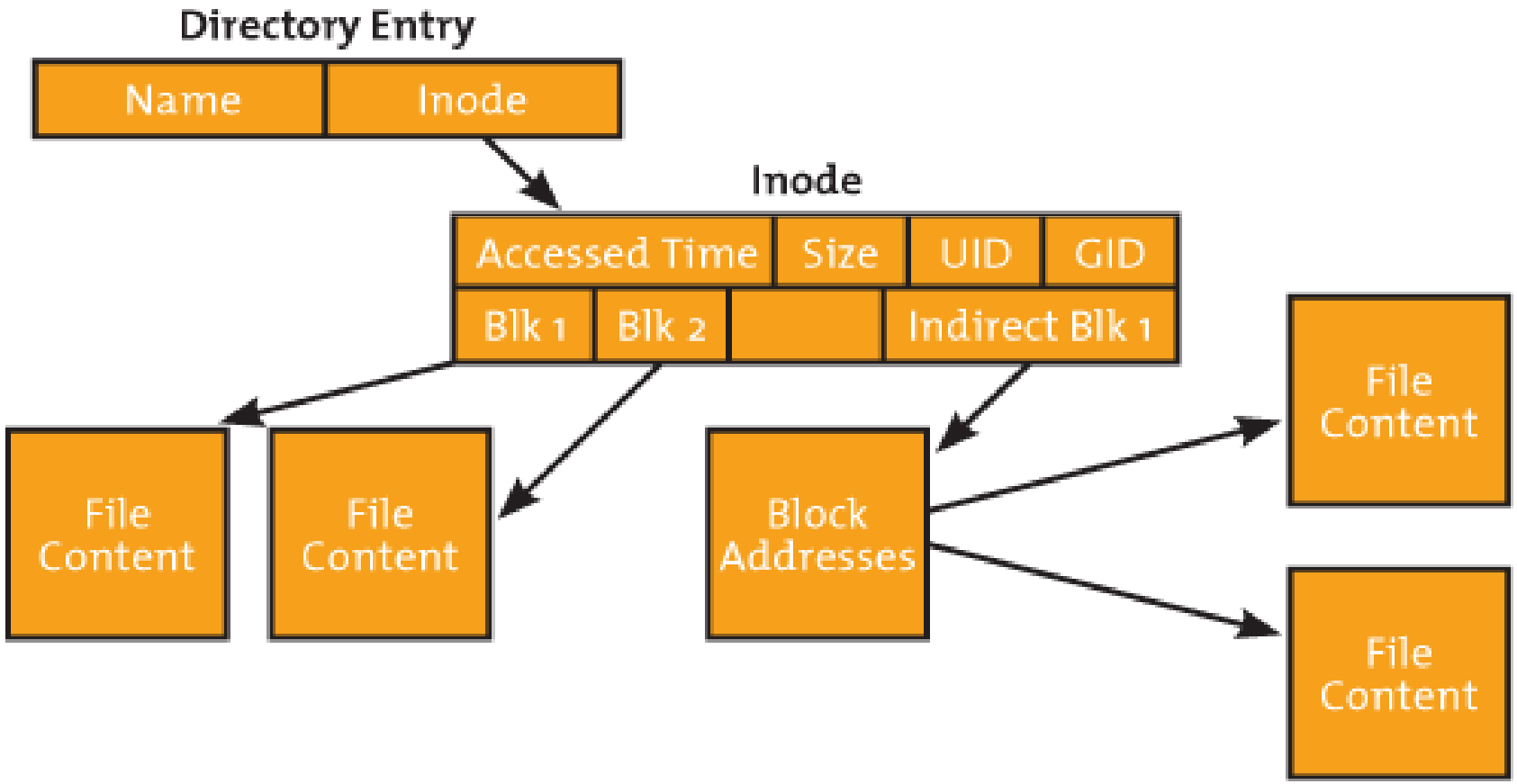
Edinburgh Napier UNIVERSITY

Metadata Concepts

- Inode Tables:
  - Inode table contains the inodes that describes the files in the group
- Inodes:
  - Each inode corresponds to one file, and it stores file's primary metadata, such as file's size, ownership, and temporal information.
  - Inode is typically 128 bytes in size and is allocated to each file and directory

Edinburgh Napier UNIVERSITY

Inode Structure



Edinburgh Napier UNIVERSITY

Metadata Concepts

- Inode Allocation:
  - If a new inode is for a non-directory file, Ext2 allocates an inode in the same block group as the parent directory.
  - If that group has no free inode or block, Ext2 uses a quadratic search (add powers of 2 to the current group)
  - If quadratic search fails, Ext2 uses linear search.

Edinburgh Napier UNIVERSITY

Metadata Concepts

- Inode Allocation:
  - If a new inode is for a directory, Ext2 tries to place it in a group that has not been used much.
  - Using total number of free inodes and blocks in the superblock, Ext2 calculates the average free inodes and blocks per group.
  - Ext2 searches each of the group and uses the first one whose free inodes and blocks are less than the average.
  - If the pervious search fails, the group with the smallest number of directories is used.

Edinburgh Napier UNIVERSITY

Indexing and Directories

- An Ext2 is just like a regular file except it has a special type value.
- The content of directories is a list of directory entry data structure, which describes file name and inode address.
- The length of directory entry varies from 1 to 255 bytes.
- There are two fields in the directory entry:
  - Name length: the length of the file name
  - Record length: the length of this directory entry

Edinburgh Napier UNIVERSITY

Indexing and Directories

- Directory entry allocation:
  - Ext2 starts at the beginning of the directory and examines each directory entry.
  - Calculate the actual record length and compare with the record length field.
  - If they are different, Ext2 can insert the new directory entry at the end of the current entry.
  - Else, Ext2 appends the new entry to the end of the entry list.

Edinburgh Napier UNIVERSITY

Journaling

- A file system journaling records updates to the file system can be recovered after a crash.
- There are two modes of journaling:
  - Only metadata updates are recorded
  - All updates are recorded
- Journaling in Ext3 is done at block level
- The first block in the journal is journal superblock, and it contains the first logging data address and its sequence number.

Edinburgh Napier UNIVERSITY

Journaling

- Updates are done in transactions, and each transaction has a sequence number.
- Each transaction starts with a descriptor block that contains the transaction sequence number and a list of what blocks are being updated.
- Following the descriptor block are the updated blocks.
- When the updates have been written to disk, a commit block is written with the same sequence number.

Edinburgh Napier
UNIVERSITY

ANY QUESTIONS ...