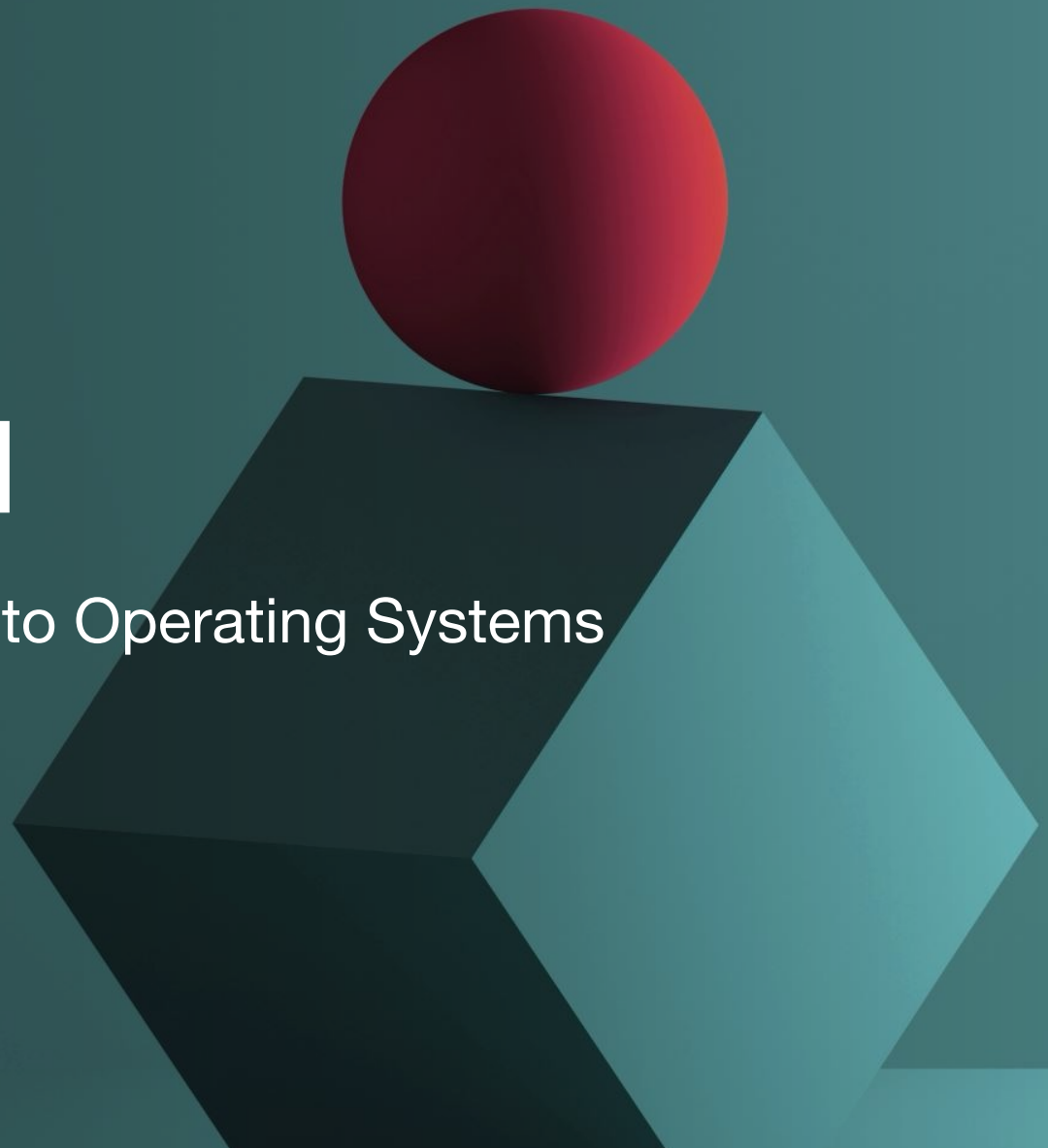


Tutorial 11

CS2106: Introduction to Operating Systems

Question 1

File System Implementation

Question 1

- This question is based on a "simple" file system built from the various components discussed in lecture 12.
- In Lecture 12, the following File Systems case studies were discussed

File System	Full Name	Distinctive Features
FAT32	File Allocation Table 32	• Files are allocated to a number of data blocks • File Allocation Table entry usually stores information about the next block. • <Block No>, EOF, Free, Bad
ext2	Extended-2 File System	• Blocks are further grouped into Block Groups • Each file / directory was described using a special structure called I-Node

Question 1: Context

- Partition Information (Free Space Information)
  - Total of 32 file data blocks
  - Bitmap of 32 bits to keep track of status (1 = Free, 0 = Occupied)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	1
16	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
1	0	1	0	1	0	1	0	1	1	1	0	0	1	0	0

Question 1: Context

- File Data:
  - Linked list allocation is used. The first value in the data block is the "next" block pointer, with "-1" to indicate the end of data block.
  - Each data block is 1 KB. For simplicity, we show only a couple of letters/numbers in each block.

0	1	2	3	4	5	6	7
11 AL	9 TH	-1 S!	-1 ND	23 GS	-1 SO	-1 :)	10 TE
8	**9**	**10**	**11**	**12**	**13**	**14**	**15**
31 RE	3 EE	28 M:	31 OH	19 SE	13 AH	4 IN	17 NO
16	**17**	**18**	**19**	**20**	**21**	**22**	**23**
30 YE	2 OU	1 ON	17 RI	26 EV	14 AT	21 DA	7 YS
24	**25**	**26**	**27**	**28**	**29**	**30**	**31**
-1 HO	18 ME	0 AL	30 OP	-1 -(	5 LO	21 ER	-1 A!

Question 1: Context

- Directory Structure + File Information:
  - Directory structures are stored in 4 "directory" blocks. Directory entries (both files and subdirectories) of a directory are stored in a single directory block.
- Directory entry:
  - **For File:** Indicates the first and last data block number.
  - **For Subdirectory:** Indicates the directory structure block number that contains the subdirectory's directory entries.
  - The "/" root directory has the directory block number 0.

0 (root)	1	2	3
y |Dir | 3 f |File| 12| 2 x |Dir | 1	g |File| 0| 31 z |Dir | 2	k |File| 6| 6	i |File| 1| 3 h |File|27|28

Question 1: Overview

Directory

0 (root)	1	2	3
y |Dir | 3 f |File| 12| 2 x |Dir | 1	g |File| 0| 31 z |Dir | 2	k |File| 6| 6	i |File| 1| 3 h |File|27|28

F.Data BM

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	1
16	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
1	0	1	0	1	0	1	0	1	1	1	0	0	1	0	0

Actual File Data

0	1	2	3	4	5	6	7
11 AL	9 TH	-1 S!	-1 ND	23 GS	-1 SO	-1 :)	10 TE
8	**9**	**10**	**11**	**12**	**13**	**14**	**15**
31 RE	3 EE	28 M:	31 OH	19 SE	13 AH	4 IN	17 NO
16	**17**	**18**	**19**	**20**	**21**	**22**	**23**
30 YE	2 OU	1 ON	17 RI	26 EV	14 AT	21 DA	7 YS
24	**25**	**26**	**27**	**28**	**29**	**30**	**31**
-1 HO	18 ME	0 AL	30 OP	-1 -(	5 LO	21 ER	-1 A!

Question 1(a): Basic Info

Give the current free capacity of the disk.

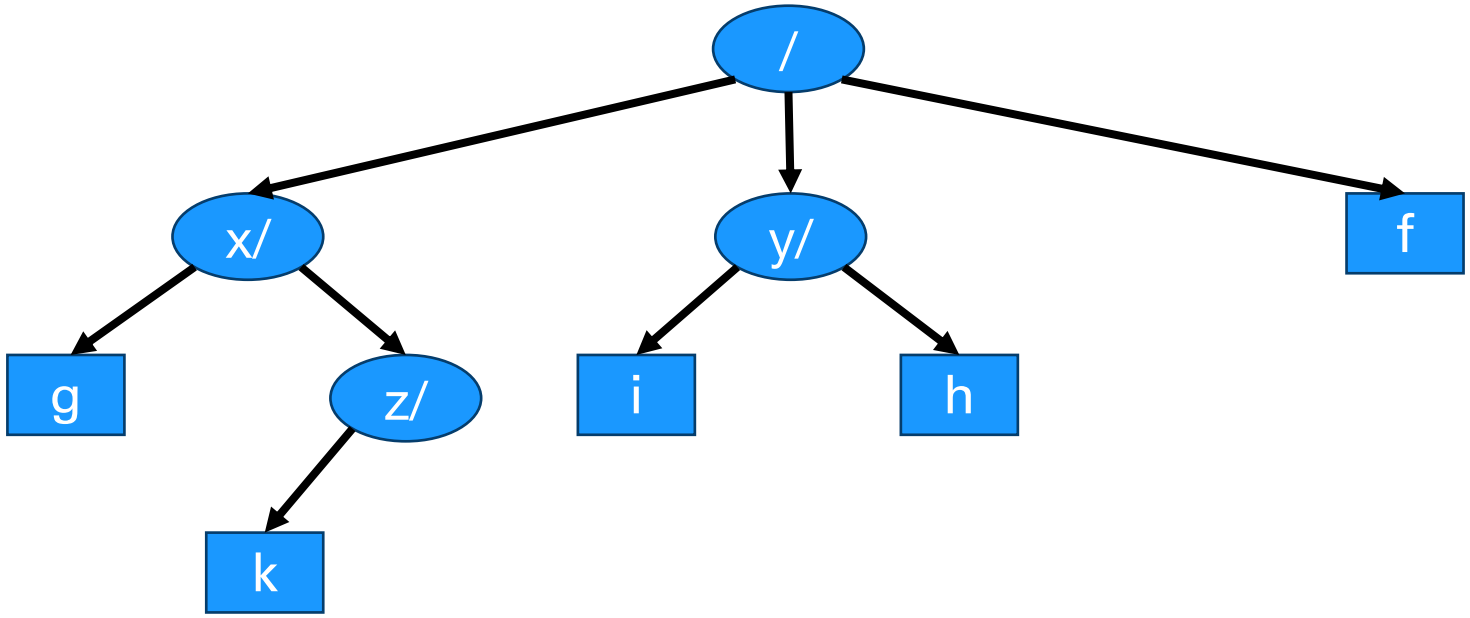
- From the Partition Info, we can see that there are 12 free blocks.
- Each data block is 1KB
- Current Free Capacity of Disk = 12KB

F.Data BM	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	1
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	1	0	1	0	1	0	1	0	1	1	1	0	0	1	0	0

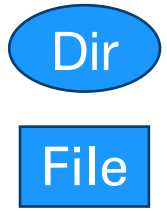
Question 1(a): Basic Info

- **For File:** Indicates the first and last data block number.
- **For Subdirectory:** Indicates the directory structure block number that contains the subdirectory's directory entries.

Give the current user view of the directory structure.



Legend



Directory	0 (root)	1	2	3
	y |Dir | 3 f |File| 12| 2 x |Dir | 1	g |File| 0| 31 z |Dir | 2	k |File| 6| 6	i |File| 1| 3 h |File|27|28

Question 1(b): File Paths

Walkthrough the file path checking for:

- "/y/i"
- "/x/z/i"

0 (root)	1	2	3
y |Dir | 3 f |File| 12| 2 x |Dir | 1	g |File| 0| 31 z |Dir | 2	k |File| 6| 6	i |File| 1| 3 h |File|27|28

Question 1(b): File Paths

Walkthrough the file path checking for:

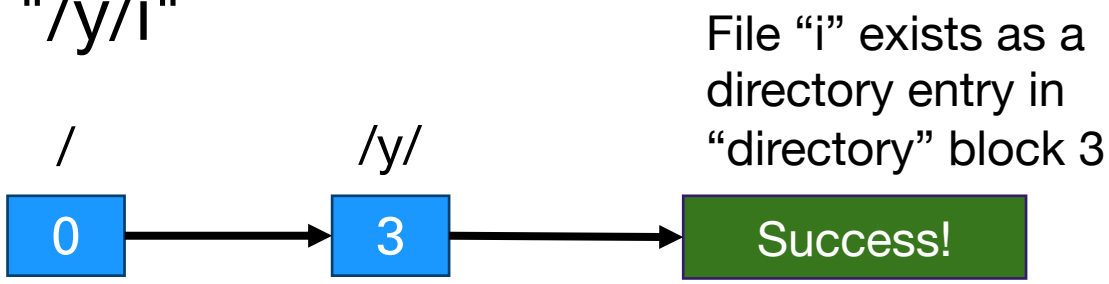
- "/y/i"

0 (root)	1	2	3
y |Dir | 3 f |File| 12| 2 x |Dir | 1	g |File| 0| 31 z |Dir | 2	k |File| 6| 6	i |File| 1| 3 h |File|27|28

Question 1(b): File Paths

Walkthrough the file path checking for:

- "/y/i"

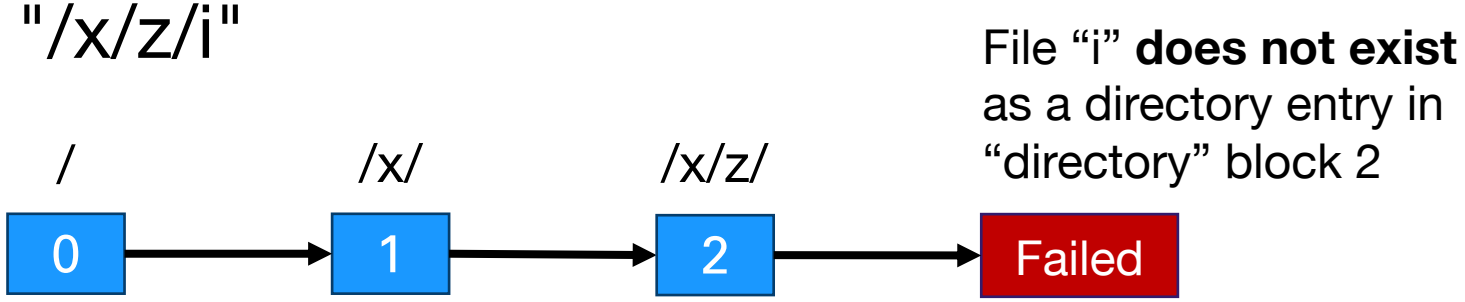


0 (root)	1	2	3
y |Dir | 3 f |File| 12| 2 x |Dir | 1	g |File| 0| 31 z |Dir | 2	k |File| 6| 6	i |File| 1| 3 h |File|27|28

Question 1(b): File Paths

Walkthrough the file path checking for:

- "/x/z/i"



0 (root)	1	2	3
y |Dir | 3 f |File| 12| 2 x |Dir | 1	g |File| 0| 31 z |Dir | 2	k |File| 6| 6	i |File| 1| 3 h |File|27|28

Question 1(c): File Access

Access the entire content for the following files:

- "/x/z/k”
- "/y/h"

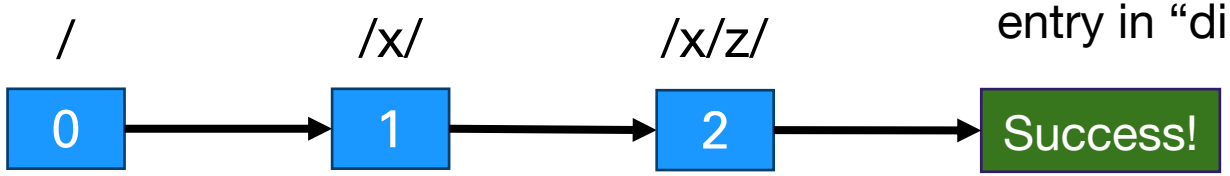
Question 1(c): File Access

- **For File:** Indicates the first and last data block number.
- **For Subdirectory:** Indicates the directory structure block number that contains the subdirectory's directory entries.

Access the entire content for the following files:

- "/x/z/k"

File "k" **exists** as a directory entry in "directory" block 2



0 (root)	1	2	3
y |Dir | 3 f |File| 12| 2 x |Dir | 1	g |File| 0| 31 z |Dir | 2	k |File| 6| 6	i |File| 1| 3 h |File|27|28

First data block for file "/x/z/k" is in file data block 6
Last data block for file "/x/z/k" is also in file data block 6

Question 1(c): File Access

Access the entire content for the following files:

- "/x/z/k”

First data block for file “/x/z/k” is in file data block 6
Last data block for file “/x/z/k” is also in file data block 6

Actual File Data

0	1	2	3	4	5	6	7
11 AL	9 TH	-1 S!	-1 ND	23 GS	-1 SO	-1 :)	10 TE
8	**9**	**10**	**11**	**12**	**13**	**14**	**15**
31 RE	3 EE	28 M:	31 OH	19 SE	13 AH	4 IN	17 NO
16	**17**	**18**	**19**	**20**	**21**	**22**	**23**
30 YE	2 OU	1 ON	17 RI	26 EV	14 AT	21 DA	7 YS
24	**25**	**26**	**27**	**28**	**29**	**30**	**31**
-1 HO	18 ME	0 AL	30 OP	-1 -(	5 LO	21 ER	-1 A!

Question 1(c): File Access

Access the entire content for the following files:

- "/x/z/k”
  - Block 6 is accessed.
  - Content: " :) "

First data block for file “/x/z/k” is in file data block 6
Last data block for file “/x/z/k” is also in file data block 6

Actual File Data

0	1	2	3	4	5	6	7
11 AL	9 TH	-1 S!	-1 ND	23 GS	-1 SO	-1 :)	10 TE
8	**9**	**10**	**11**	**12**	**13**	**14**	**15**
31 RE	3 EE	28 M:	31 OH	19 SE	13 AH	4 IN	17 NO
16	**17**	**18**	**19**	**20**	**21**	**22**	**23**
30 YE	2 OU	1 ON	17 RI	26 EV	14 AT	21 DA	7 YS
24	**25**	**26**	**27**	**28**	**29**	**30**	**31**
-1 HO	18 ME	0 AL	30 OP	-1 -(	5 LO	21 ER	-1 A!

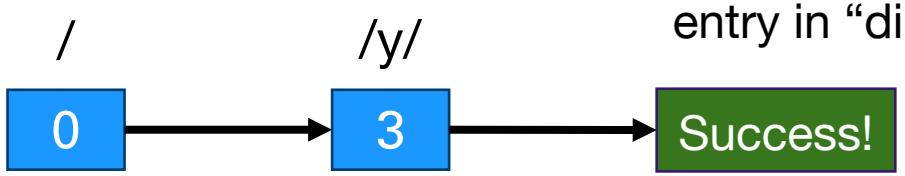
Question 1(c): File Access

- **For File:** Indicates the first and last data block number.
- **For Subdirectory:** Indicates the directory structure block number that contains the subdirectory's directory entries.

Access the entire content for the following files:

- "/y/h"

File "h" **exists** as a directory entry in "directory" block 3



0 (root)	1	2	3
y |Dir | 3 f |File| 12| 2 x |Dir | 1	g |File| 0| 31 z |Dir | 2	k |File| 6| 6	i |File| 1| 3 h |File|27|28

First data block for file "/y/h" is in file data block 27
Last data block for file "/y/h" is also in file data block 28

Question 1(c): File Access

Access the entire content for the following files:

- "/x/z/k”

First data block for file “/y/h” is in file data block 27
Last data block for file “/y/h” is also in file data block 28

Actual File Data

0	1	2	3	4	5	6	7
11 AL	9 TH	-1 S!	-1 ND	23 GS	-1 SO	-1 :)	10 TE
8	**9**	**10**	**11**	**12**	**13**	**14**	**15**
31 RE	3 EE	28 M:	31 OH	19 SE	13 AH	4 IN	17 NO
16	**17**	**18**	**19**	**20**	**21**	**22**	**23**
30 YE	2 OU	1 ON	17 RI	26 EV	14 AT	21 DA	7 YS
24	**25**	**26**	**27**	**28**	**29**	**30**	**31**
-1 HO	18 ME	0 AL	30 OP	-1 -(	5 LO	21 ER	-1 A!

Question 1(c): File Access

Access the entire content for the following files:

- "/x/z/k”
  - Blocks 27, 30, 21, 14, 4, 23, 7, 10, 28 are accessed.
  - Content: **"OPERATINGSYSTEM:-("**

First data block for file “/y/h” is in file data block 27
Last data block for file “/y/h” is also in file data block 28

Actual File Data

0	1	2	3	4	5	6	7
11 AL	9 TH	-1 S!	-1 ND	23 GS	-1 SO	-1 :)	10 TE
8	**9**	**10**	**11**	**12**	**13**	**14**	**15**
31 RE	3 EE	28 M:	31 OH	19 SE	13 AH	4 IN	17 NO
16	**17**	**18**	**19**	**20**	**21**	**22**	**23**
30 YE	2 OU	1 ON	17 RI	26 EV	14 AT	21 DA	7 YS
24	**25**	**26**	**27**	**28**	**29**	**30**	**31**
-1 HO	18 ME	0 AL	30 OP	-1 -(	5 LO	21 ER	-1 A!

Question 1(d): Create File

Add a new file "/y/n" with 5 blocks of content. You can assume we always use the free block with the smallest block number. Indicate all changes required to add the file.

Question 1(d): Create File

Add a new file "/y/n" with 5 blocks of content. You can assume we always use the free block with the smallest block number. Indicate all changes required to add the file.

1. Update bits, 5, 8, 13, 15, 16 in the bitmap to 0.

Before

F.Data BM	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	1
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	1	0	1	0	1	0	1	0	1	1	1	0	0	1	0	0

After

F.Data BM	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	0	0	1	0	1	0	1	0	1	1	1	0	0	1	0	0

Question 1(d): Create File

Add a new file "/y/n" with 5 blocks of content. You can assume we always use the free block with the smallest block number. Indicate all changes required to add the file.

2. Add new entry in “directory” block 3: **`"n |File| 5|16"`**

Before

Directory	0 (root)	1	2	3
	y |Dir | 3 f |File| 12| 2 x |Dir | 1	g |File| 0| 31 z |Dir | 2	k |File| 6| 6	i |File| 1| 3 h |File|27|28

After

Directory	0 (root)	1	2	3
	y |Dir | 3 f |File| 12| 2 x |Dir | 1	g |File| 0| 31 z |Dir | 2	k |File| 6| 6	i |File| 1| 3 h |File|27|28 n |File| 5|16

Question 1(d): Create File

Add a new file "/y/n" with 5 blocks of content. You can assume we always use the free block with the smallest block number. Indicate all changes required to add the file.

3. Update File Data Blocks 5 → 8 → 13 → 15 → 16 → -1

Before

Actual File Data

0	1	2	3	4	5	6	7
11 AL	9 TH	-1 S!	-1 ND	23 GS	-1 SO	-1 :)	10 TE
8	**9**	**10**	**11**	**12**	**13**	**14**	**15**
31 RE	3 EE	28 M:	31 OH	19 SE	13 AH	4 IN	17 NO
16	**17**	**18**	**19**	**20**	**21**	**22**	**23**
30 YE	2 OU	1 ON	17 RI	26 EV	14 AT	21 DA	7 YS
24	**25**	**26**	**27**	**28**	**29**	**30**	**31**
-1 HO	18 ME	0 AL	30 OP	-1 -(	5 LO	21 ER	-1 A!

Question 1(d): Create File

Add a new file "/y/n" with 5 blocks of content. You can assume we always use the free block with the smallest block number. Indicate all changes required to add the file.

3. Update File Data Blocks 5 → 8 → 13 → 15 → 16 → -1

After

Actual File Data

0	1	2	3	4	5	6	7
11 AL	9 TH	-1 S!	-1 ND	23 GS	8 SO	-1 :)	10 TE
8	**9**	**10**	**11**	**12**	**13**	**14**	**15**
13 RE	3 EE	28 M:	31 OH	19 SE	15 AH	4 IN	16 NO
16	**17**	**18**	**19**	**20**	**21**	**22**	**23**
-1 YE	2 OU	1 ON	17 RI	26 EV	14 AT	21 DA	7 YS
24	**25**	**26**	**27**	**28**	**29**	**30**	**31**
-1 HO	18 ME	0 AL	30 OP	-1 -(	5 LO	21 ER	-1 A!

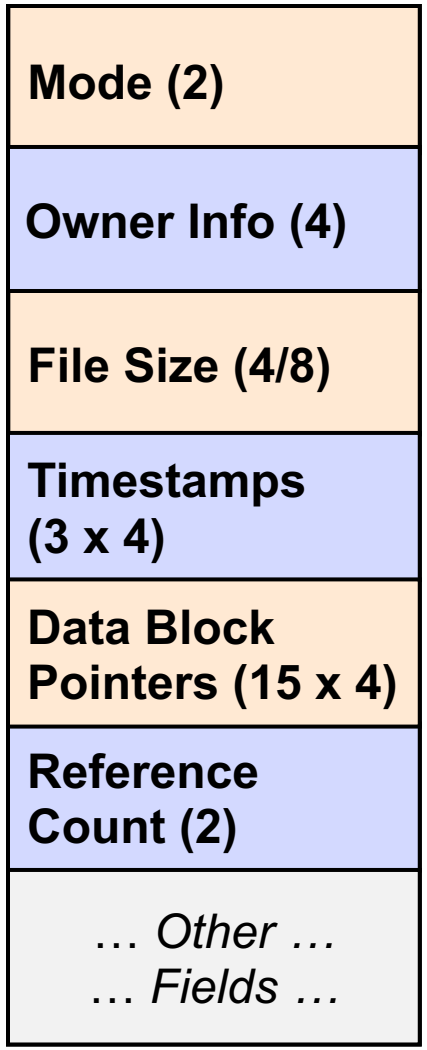
Question 1(d): Create File

Add a new file "/y/n" with 5 blocks of content. You can assume we always use the free block with the smallest block number. Indicate all changes required to add the file.

1. Bitmap updated: Bit 5, 8, 13, 15, 16 changed to 0
2. Directory block 3 (for /y) updated: "n |File| 5|16" added
3. Data Blocks 5, 8, 13, 15, 16 (next block pointer changed, with -1 in block 16).

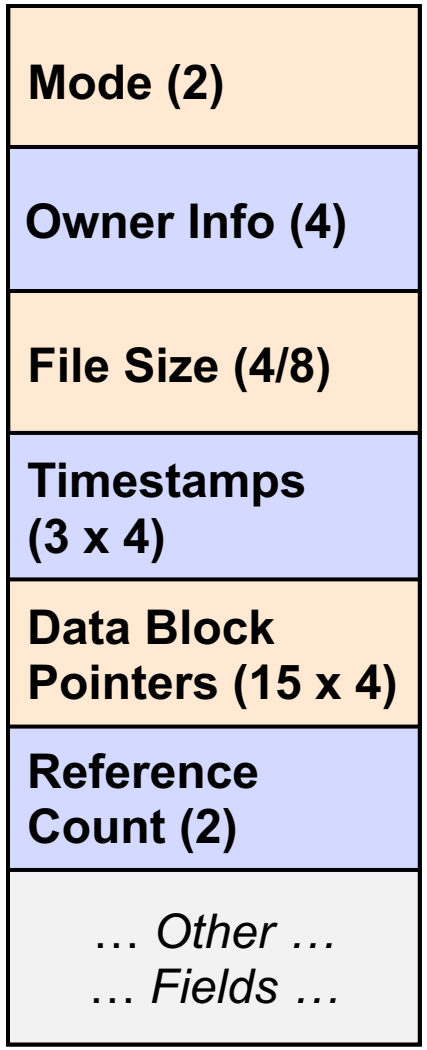
ADDITIONAL CONTENT

Worked Example for I-Node in EXT2



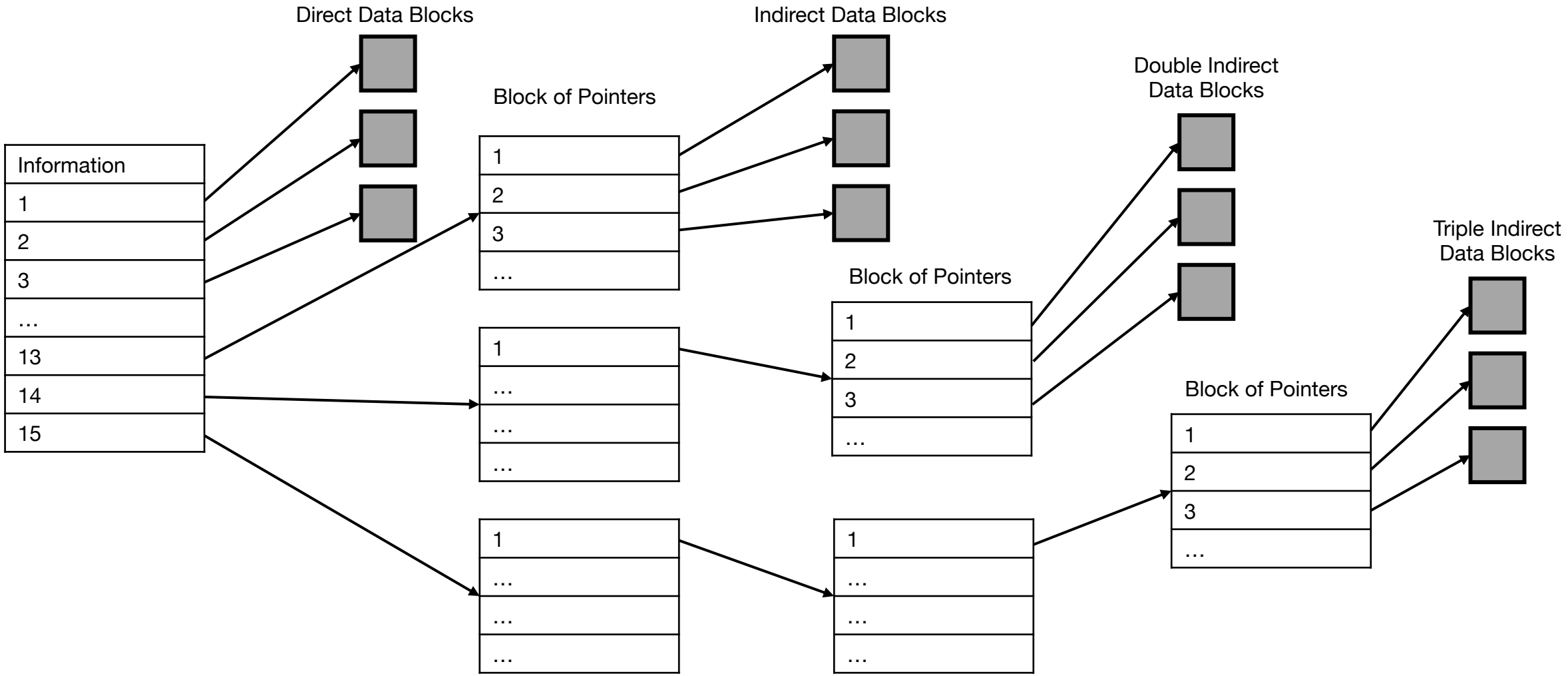
- In EXT2, each file and directory is represented by an I-Node.
- In this worked example, lets find out the maximum size of data that can be contained within an I-Node in EXT2, given the size of a data block.
  1. First calculate the maximum number of
     - Direct Blocks
     - Single Indirect Blocks
     - Double Indirect Blocks
     - Triple Indirect Blocks
  2. Then multiply by the data block size
- Let's look in detail what does an I-Node contain.

Worked Example for I-Node in EXT2



- The size of each I-Node in EXT2 is 128 Bytes.
  - Mode: 2 bytes
  - Owner Info: 4 bytes (Owner ID / Group ID)
  - File Size (4 or 8 bytes):
    - 4 Bytes (32-bit Systems), supports up to 4GB file size
    - 8 Bytes (64-bit Systems), supports up to 16 Exa-Bytes file size
  - Time Stamps (3 × 4 bytes)
    - Access Time: When was file last read
    - Modification: When file content was last changed
    - Change Time: When I-Node metadata was last changed
  - Data Blocks: (15 pointers × 4 bytes)
    - 12 Direct Data Block Pointers
    - 1 Single Indirect Data Block Pointers
    - 1 Double Indirect Data Block Pointers
    - 1 Triple Indirect Data Block Pointers
  - Reference Count: 2 bytes
    - Which means a maximum of 2^{16} = 65536 hard links!

Worked Example for I-Node in EXT2



Worked Example for I-Node in EXT2

- Suppose, each data block is 1KB
- What is the size of the block of pointers?
  - The block of pointers pointed pointed by pointers 13, 14 and 15 are the same size as each data block which is also 1KB.
- How many pointers can be contained in a data block?
  - $\frac{1KB}{4\ bytes} = \frac{2^{10}}{2^2} = 256$ pointers

Worked Example for I-Node in EXT2

- Number of pointers in data block = 256
- Calculate the number of
  - Direct Blocks = 12
  - Single Indirect Blocks = 256
  - Double Indirect Blocks = $256^2 = 2^{16} = 65536$
  - Triple Indirect Blocks = $256^3 = 2^{24} = 16777216$
- Total number of data blocks = 16843020
- Maximum Data Size in I-Node with 1KB Data Blocks
  - 16843020 KB = 16.062GB

Reference: https://man7.org/linux/man-pages/man3/readdir.3.html

Question

- If an I-Node is for a directory, then what does its data block pointers point to?
  - It still points to the data blocks (Direct, Single Indirect, …)
  - Except that each data block now contains directory entries.
  - Each directory entry contains:
    - I-Node number for that file / subdirectory
    - Size (Length) of directory entry
    - Length of the file / subdirectory name
    - Type: File, Subdirectory, Symbolic Link, etc…
    - File / subdirectory name (e.g. `"Cat.png"`)

Directory entries are stored as a Linked List where each directory entry is of variable length (one reason is because of differing lengths of filenames).

So, by storing the size of the directory entry, we can quickly jump ahead or seek to the next entry.

END OF TUTORIAL