CSCI 509

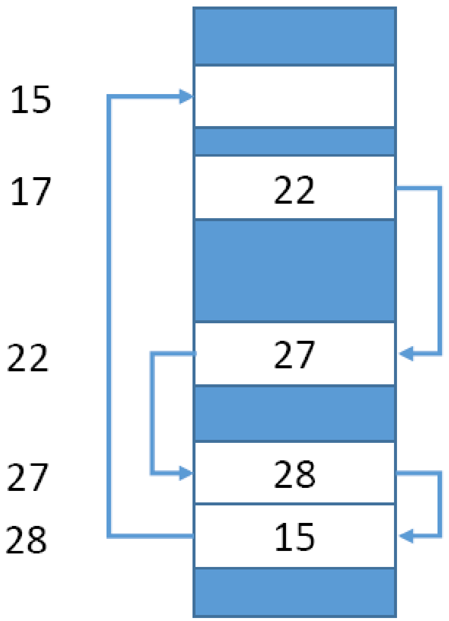
OPERATING SYSTEMS

FILE ALLOCATION TABLE

File Allocation Table

Reserve a single (or two or three) blocks, to hold a table of all blocks

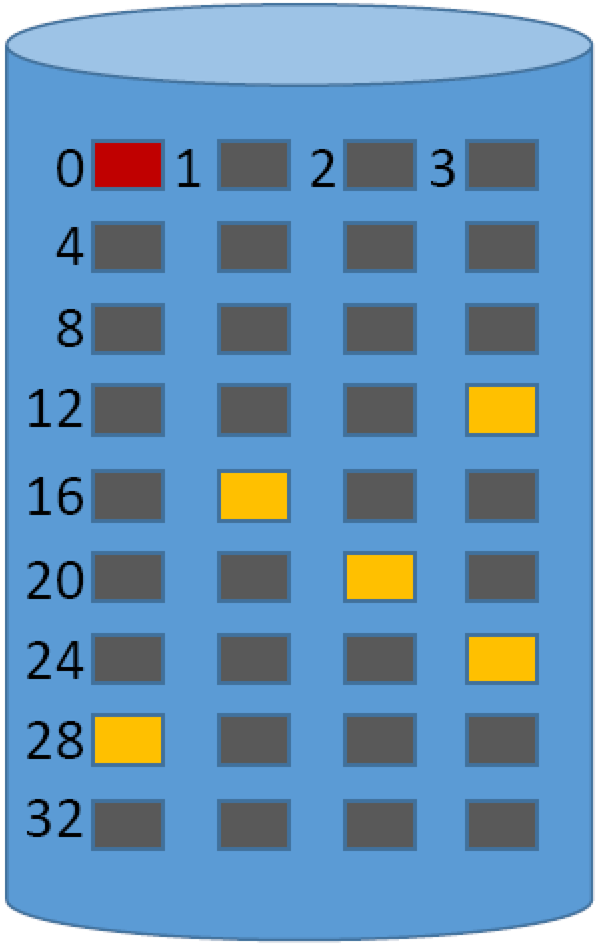
File Allocation Table



Directory entry

aFile	other	17

The directory entry contains only the start block of the data, and using the FAT, the OS can identify all of the blocks that the file occupies

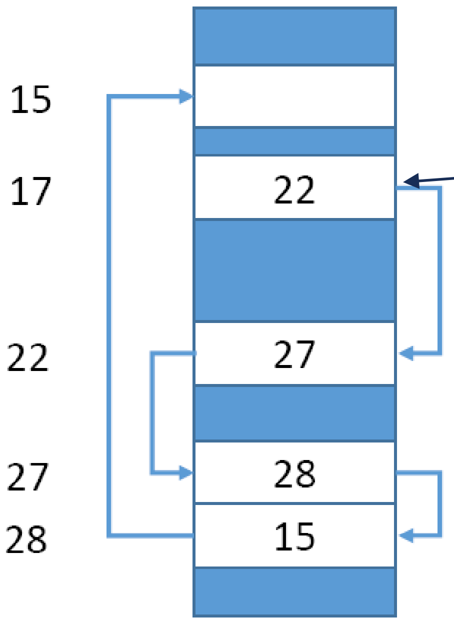


FILE ALLOCATION TABLE

File Allocation Table

Reserve a single (or two or three) blocks, to hold a table of all blocks

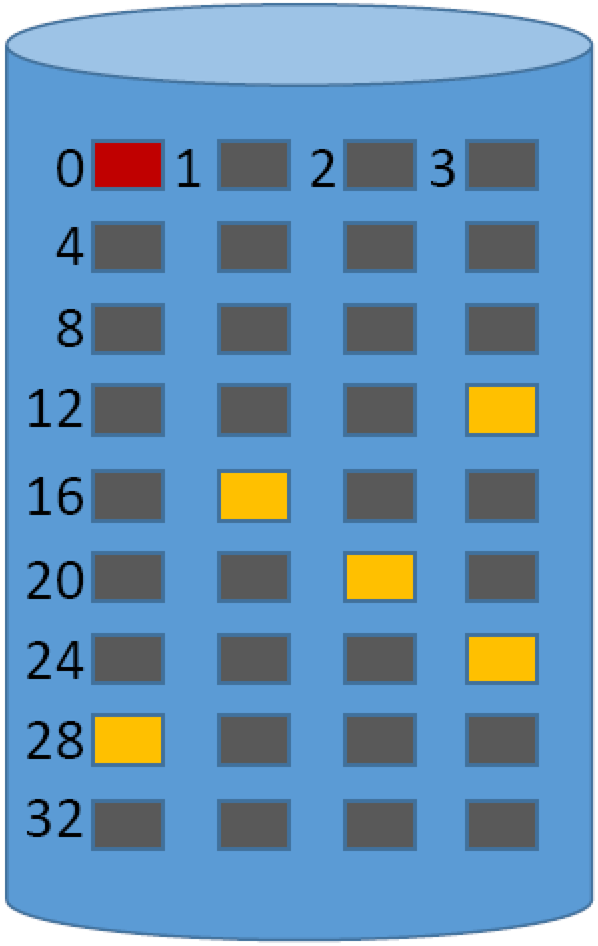
File Allocation Table



Directory entry

aFile	other	17

The directory entry contains only the start block of the data, and using the FAT, the OS can identify all of the blocks that the file occupies

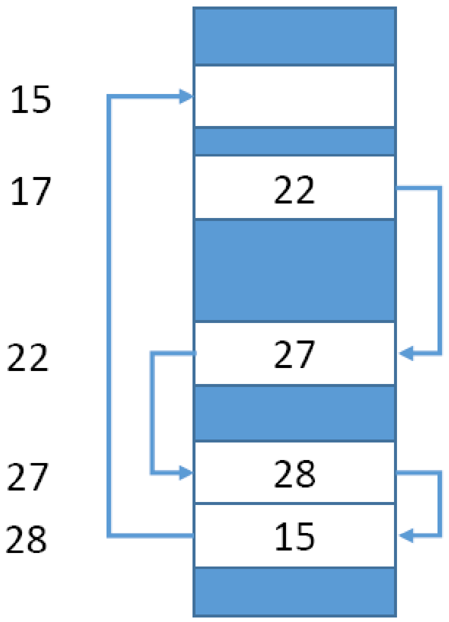


FILE ALLOCATION TABLE

File Allocation Table

Reserve a single (or two or three) blocks, to hold a table of all blocks

File Allocation Table

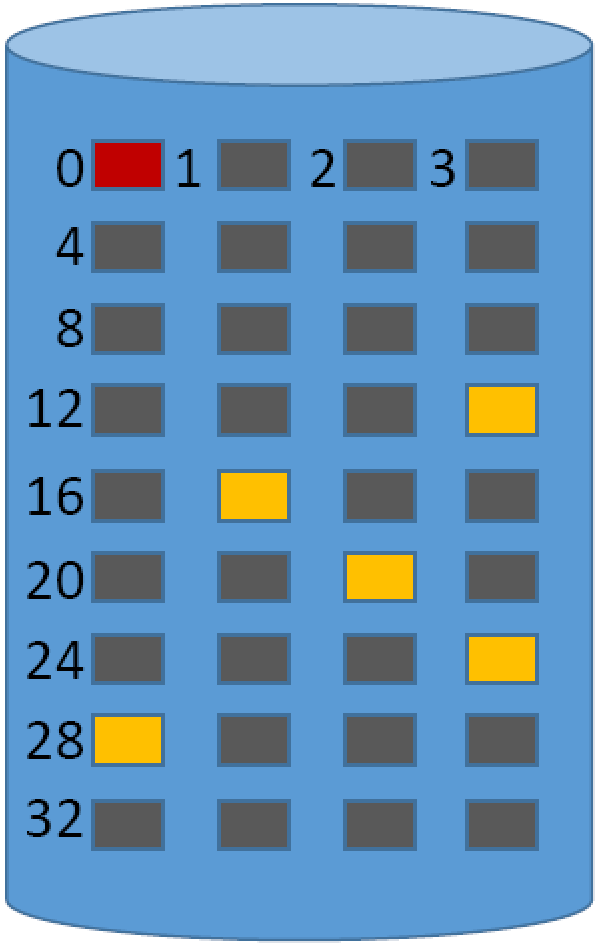


- The "chaining" is only done in the table.
- Allows for faster traverse.

Directory entry

aFile	other	17

The directory entry contains only the start block of the data, and using the FAT, the OS can identify all of the blocks that the file occupies

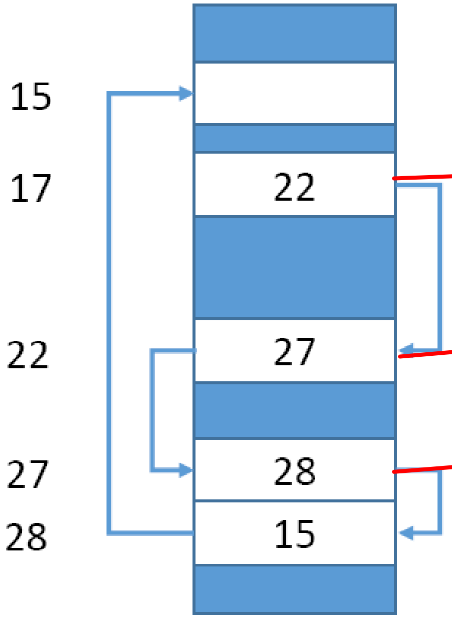


FILE ALLOCATION TABLE

File Allocation Table

Reserve a single (or two or three) blocks, to hold a table of all blocks

File Allocation Table

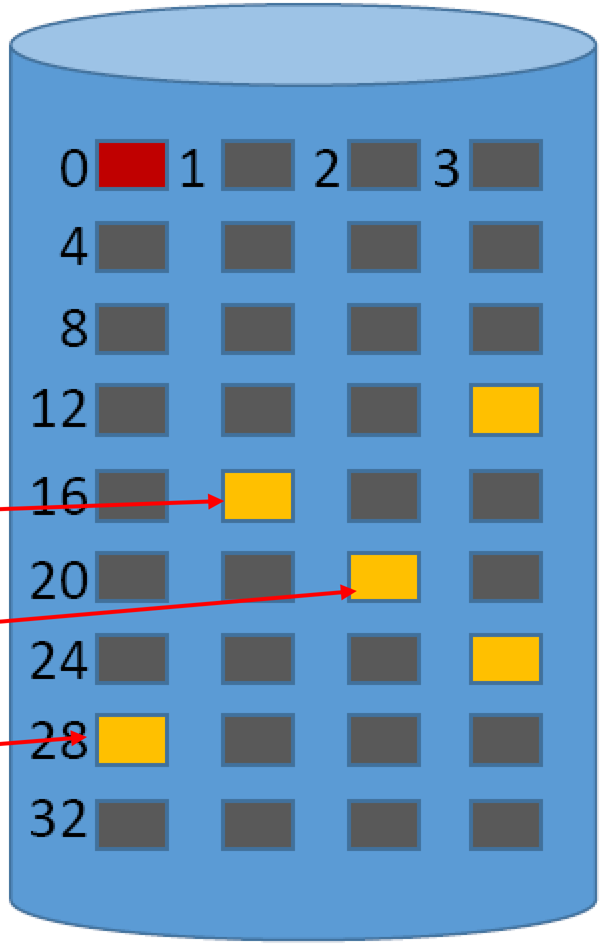


- The "chaining" is only done in the table.
- Allows for faster traverse.

Directory entry

aFile	other	17

The directory entry contains only the start block of the data, and using the FAT, the OS can identify all of the blocks that the file occupies

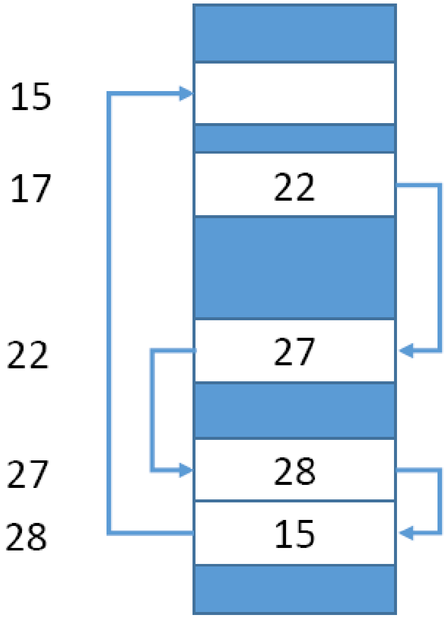


FILE ALLOCATION TABLE

File Allocation Table

Reserve a single (or two or three) blocks, to hold a table of all blocks

File Allocation Table

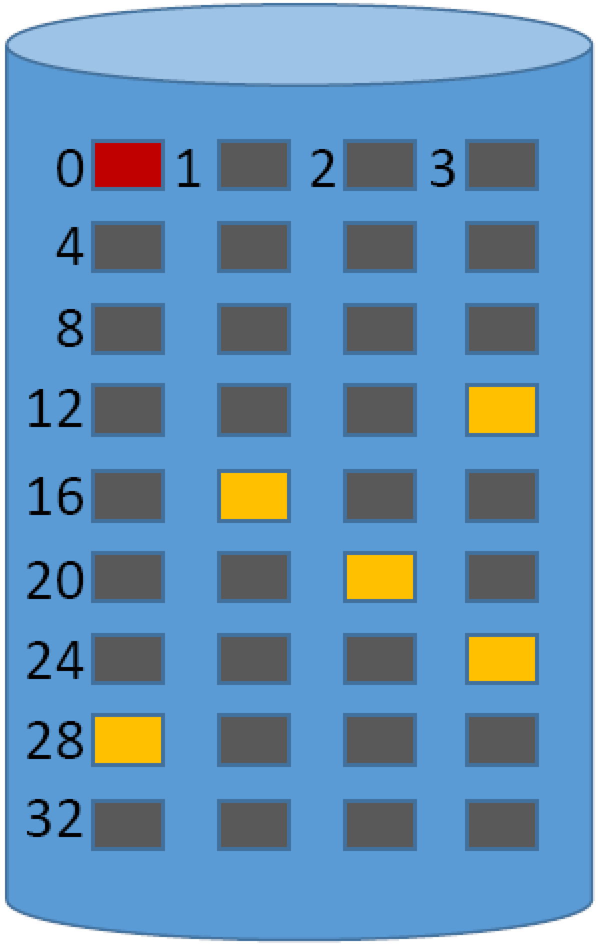


- The "chaining" is only done in the table.
- Allows for faster traverse.

Directory entry

aFile	other	17

The directory entry contains only the start block of the data, and using the FAT, the OS can identify all of the blocks that the file occupies



- Assignment 5 is now due March 10 to void having submission during prep week. Ten days should be ample time to complete this assignment which is much easier than A4.

FILE ALLOCATION TABLE

- Worksheet: Given a FAT and a file with a first block address of 25 on disk, list the blocks that belong to this file, in order.

14	-1
15	19
16	12
17	28
18	-1
19	22
20	2
21	31
22	23
23	18
24	-1
25	15

FILE ALLOCATION TABLE

- Worksheet: Given a FAT and a file with a first block address of 25 on disk, list the blocks that belong to this file, in order.

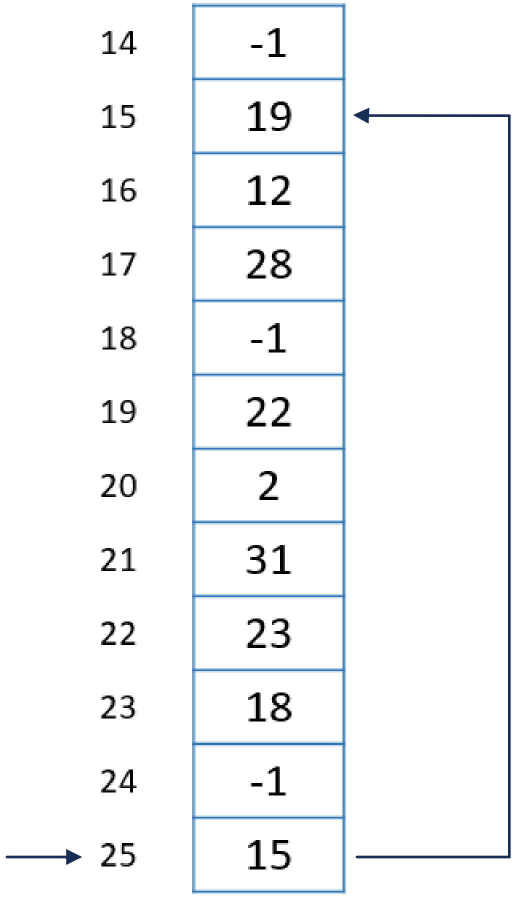
1. 25

14	-1
15	19
16	12
17	28
18	-1
19	22
20	2
21	31
22	23
23	18
24	-1
→ 25	15

FILE ALLOCATION TABLE

- Worksheet: Given a FAT and a file with a first block address of 25 on disk, list the blocks that belong to this file, in order.

1. 25
2. 15

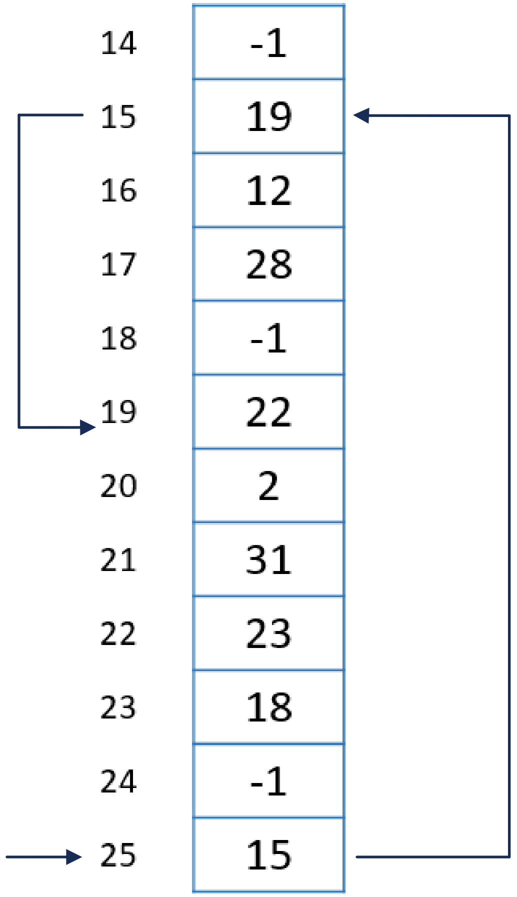


14	-1
15	19
16	12
17	28
18	-1
19	22
20	2
21	31
22	23
23	18
24	-1
25	15

FILE ALLOCATION TABLE

- Worksheet: Given a FAT and a file with a first block address of 25 on disk, list the blocks that belong to this file, in order.

1. 25
2. 15
3. 19

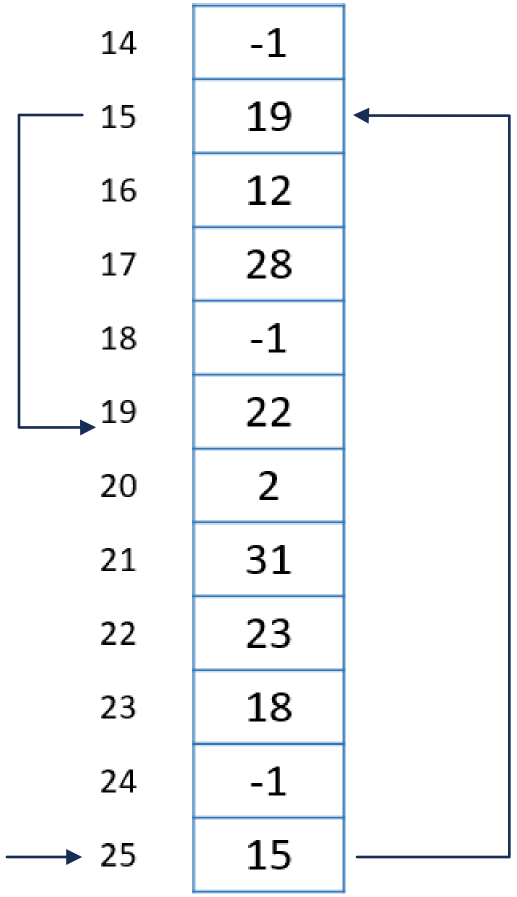


14	-1
15	19
16	12
17	28
18	-1
19	22
20	2
21	31
22	23
23	18
24	-1
25	15

FILE ALLOCATION TABLE

- Worksheet: Given a FAT and a file with a first block address of 25 on disk, list the blocks that belong to this file, in order.

1. 25
2. 15
3. 19
4. 22

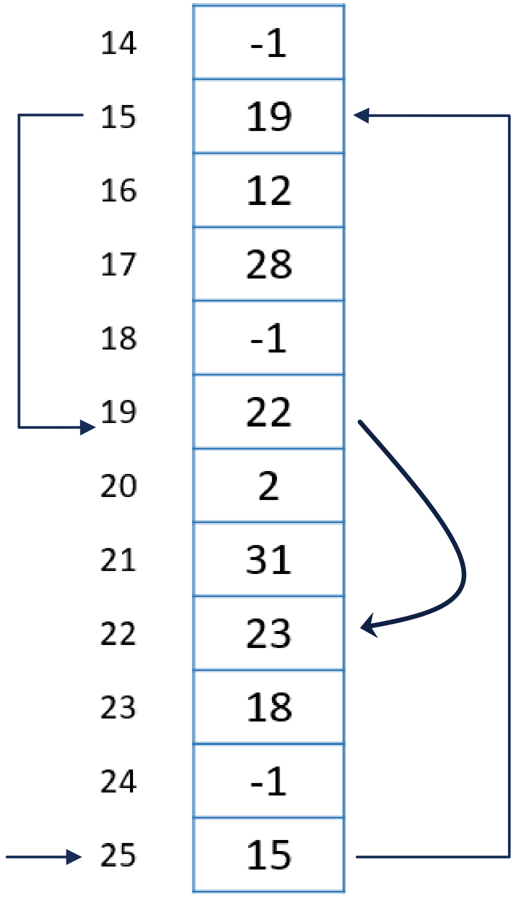


14	-1
15	19
16	12
17	28
18	-1
19	22
20	2
21	31
22	23
23	18
24	-1
25	15

FILE ALLOCATION TABLE

- Worksheet: Given a FAT and a file with a first block address of 25 on disk, list the blocks that belong to this file, in order.

1. 25
2. 15
3. 19
4. 22

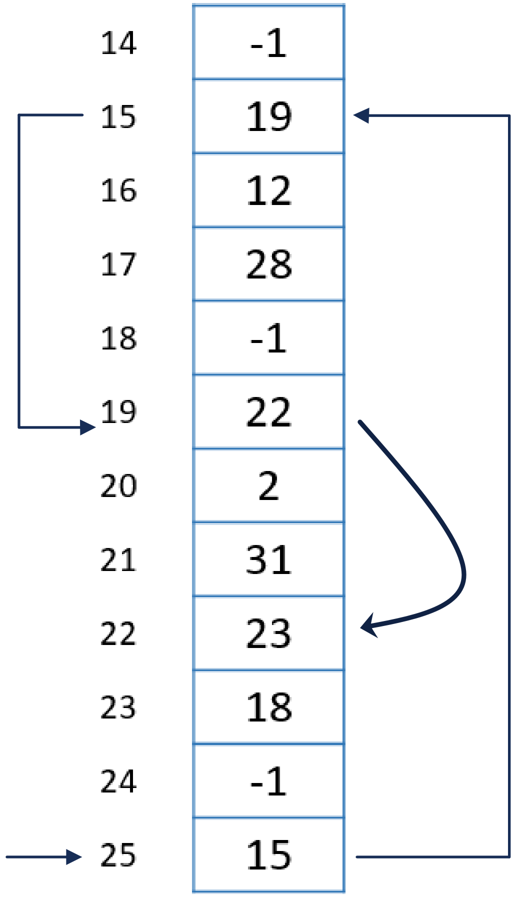


14	-1
15	19
16	12
17	28
18	-1
19	22
20	2
21	31
22	23
23	18
24	-1
25	15

FILE ALLOCATION TABLE

- Worksheet: Given a FAT and a file with a first block address of 25 on disk, list the blocks that belong to this file, in order.

1. 25
2. 15
3. 19
4. 22
5. 23

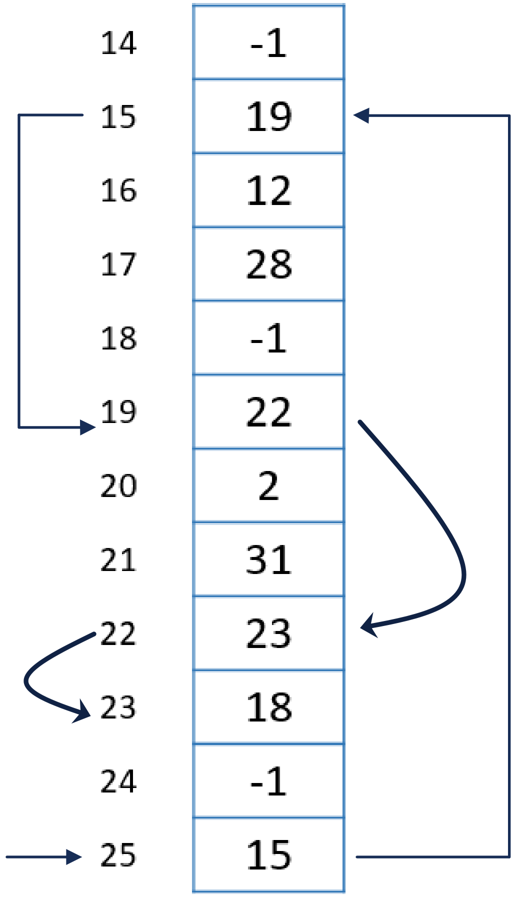


14	-1
15	19
16	12
17	28
18	-1
19	22
20	2
21	31
22	23
23	18
24	-1
25	15

FILE ALLOCATION TABLE

- Worksheet: Given a FAT and a file with a first block address of 25 on disk, list the blocks that belong to this file, in order.

1. 25
2. 15
3. 19
4. 22
5. 23

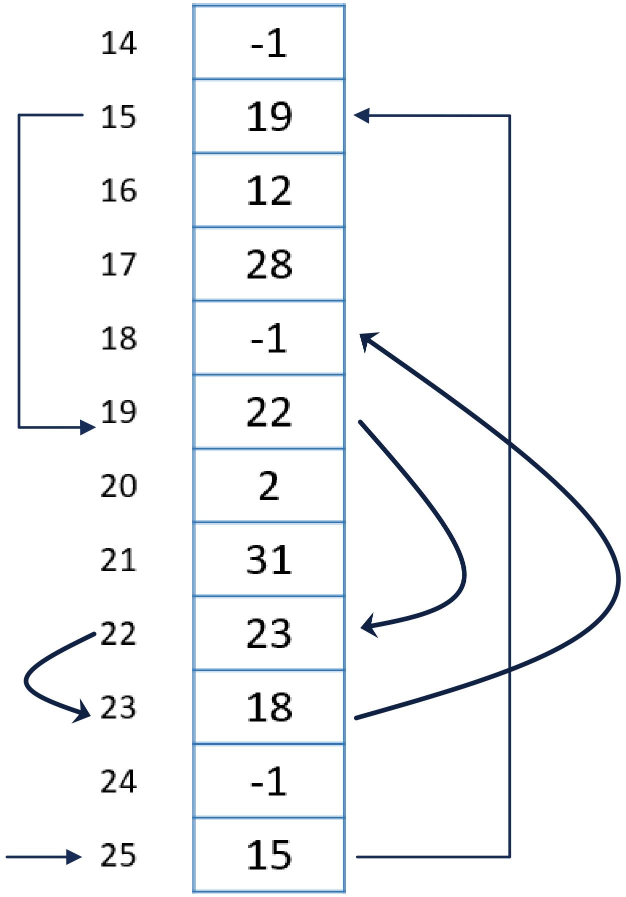


14	-1
15	19
16	12
17	28
18	-1
19	22
20	2
21	31
22	23
23	18
24	-1
25	15

FILE ALLOCATION TABLE

- Worksheet: Given a FAT and a file with a first block address of 25 on disk, list the blocks that belong to this file, in order.

1. 25
2. 15
3. 19
4. 22
5. 23
6. 18

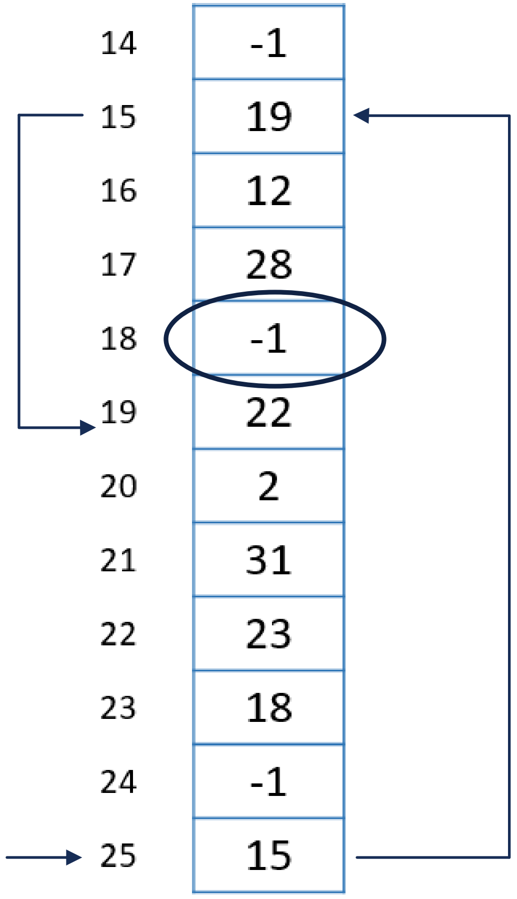


14	-1
15	19
16	12
17	28
18	-1
19	22
20	2
21	31
22	23
23	18
24	-1
25	15

FILE ALLOCATION TABLE

- Worksheet: Given a FAT and a file with a first block address of 25 on disk, list the blocks that belong to this file, in order.

1. 25
2. 15
3. 19
4. 22
5. 23
6. 18 ← Last block because next is '-1'

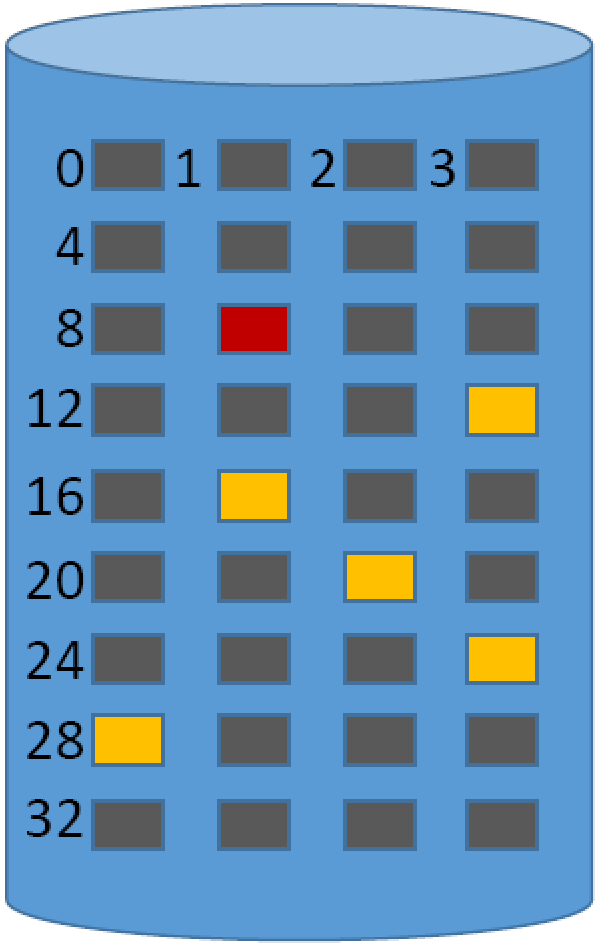


14	-1
15	19
16	12
17	28
18	-1
19	22
20	2
21	31
22	23
23	18
24	-1
25	15

INDEX ALLOCATION

Index Allocation

A single index block contains ALL of the pointers for the blocks in use by a file



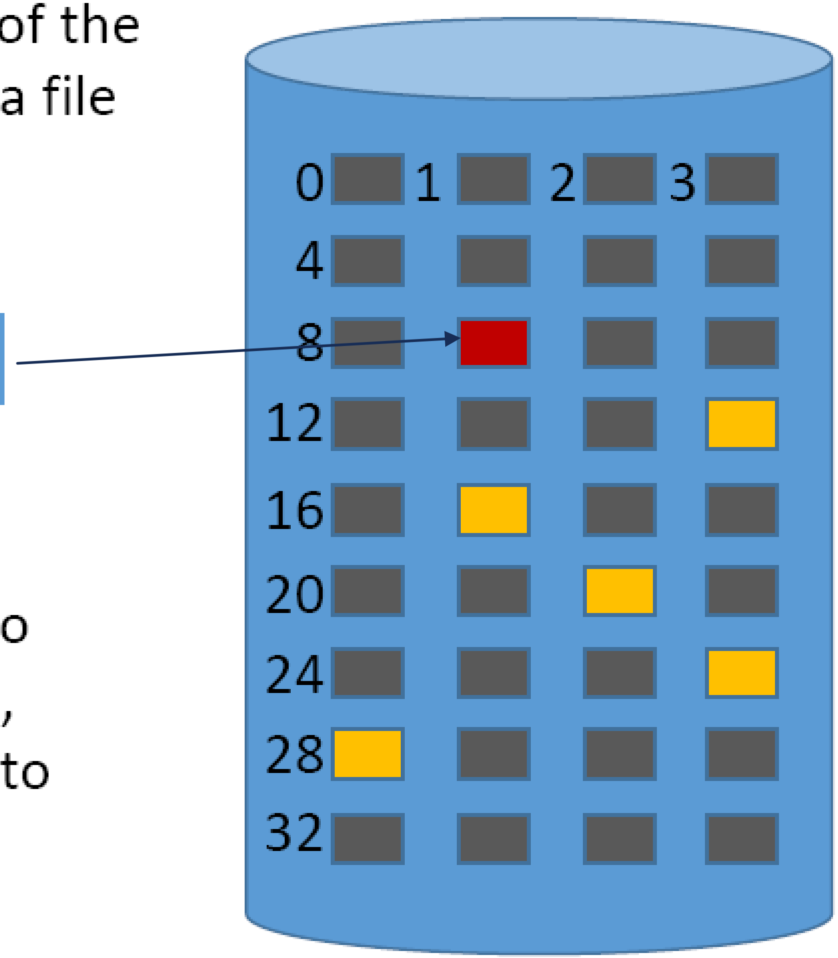
INDEX ALLOCATION

Index Allocation

A single index block contains ALL of the pointers for the blocks in use by a file

Directory entry

aFile	other	9



The block entry in the directory metadata points to the index block of file *aFile*, which contains the pointers to the data blocks for the file

INDEX ALLOCATION

Index Allocation

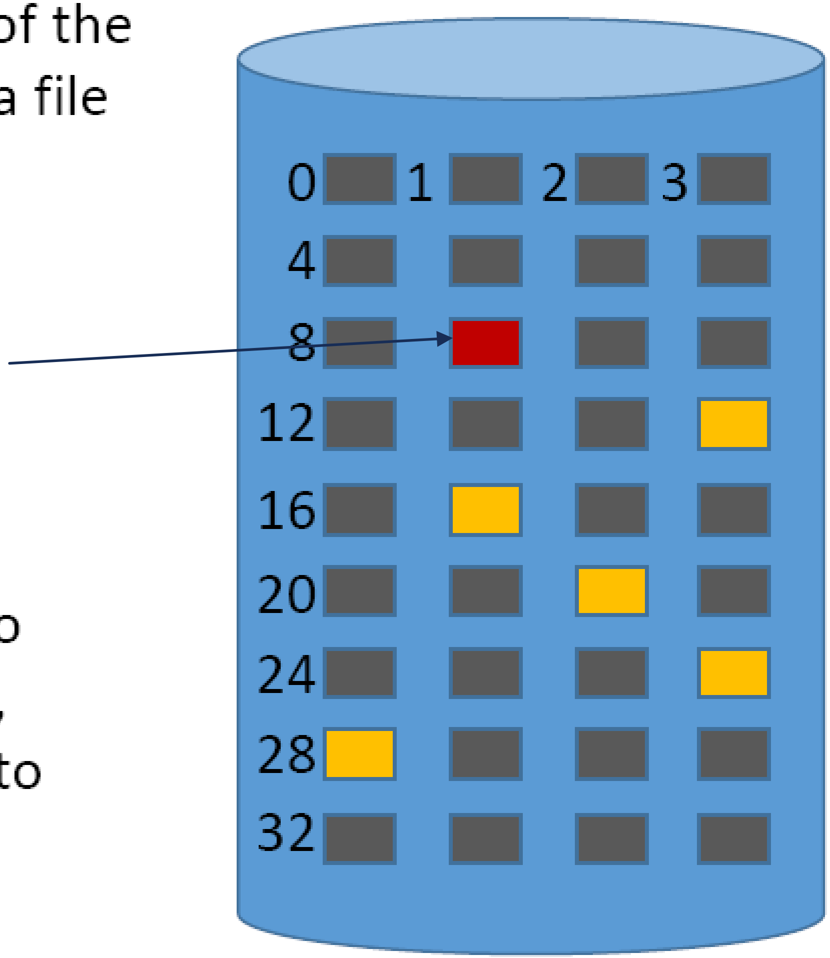
A single index block contains ALL of the pointers for the blocks in use by a file

Block 9
17
22
27
28
15
-1
-1
-1

Directory entry

aFile	other	9

The block entry in the directory metadata points to the index block of file *aFile*, which contains the pointers to the data blocks for the file



INDEX ALLOCATION

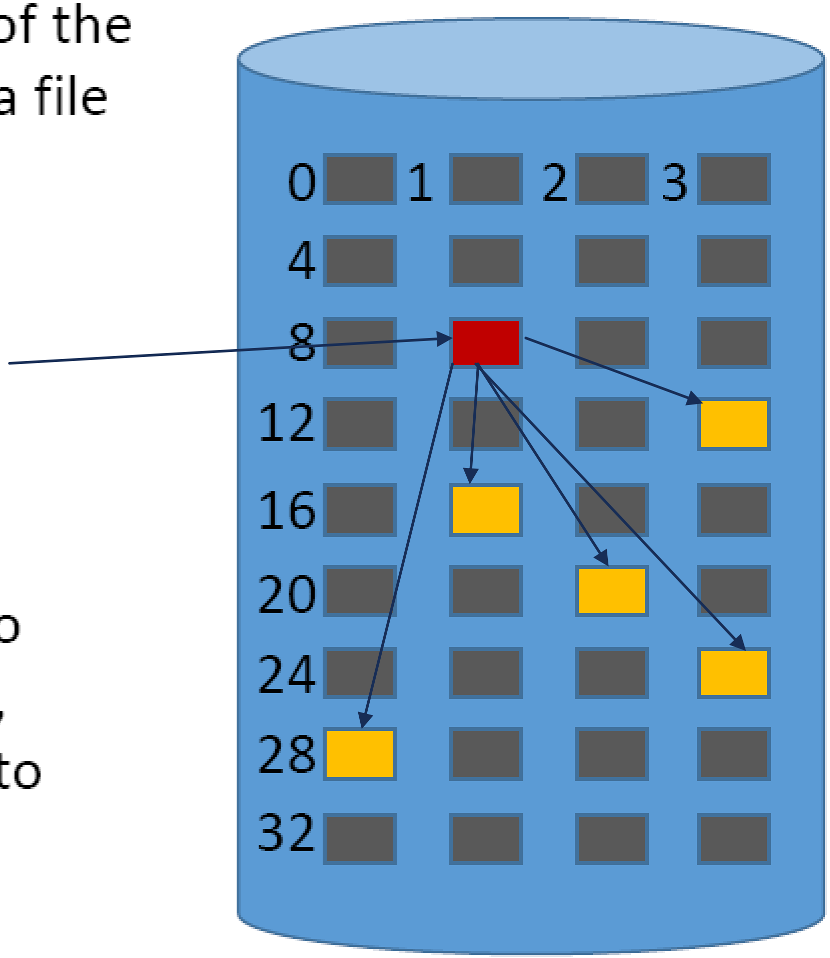
Index Allocation

A single index block contains ALL of the pointers for the blocks in use by a file

Block 9
17
22
27
28
15
-1
-1
-1

Directory entry

aFile	other	9



The block entry in the directory metadata points to the index block of file *aFile*, which contains the pointers to the data blocks for the file

INDEX ALLOCATION

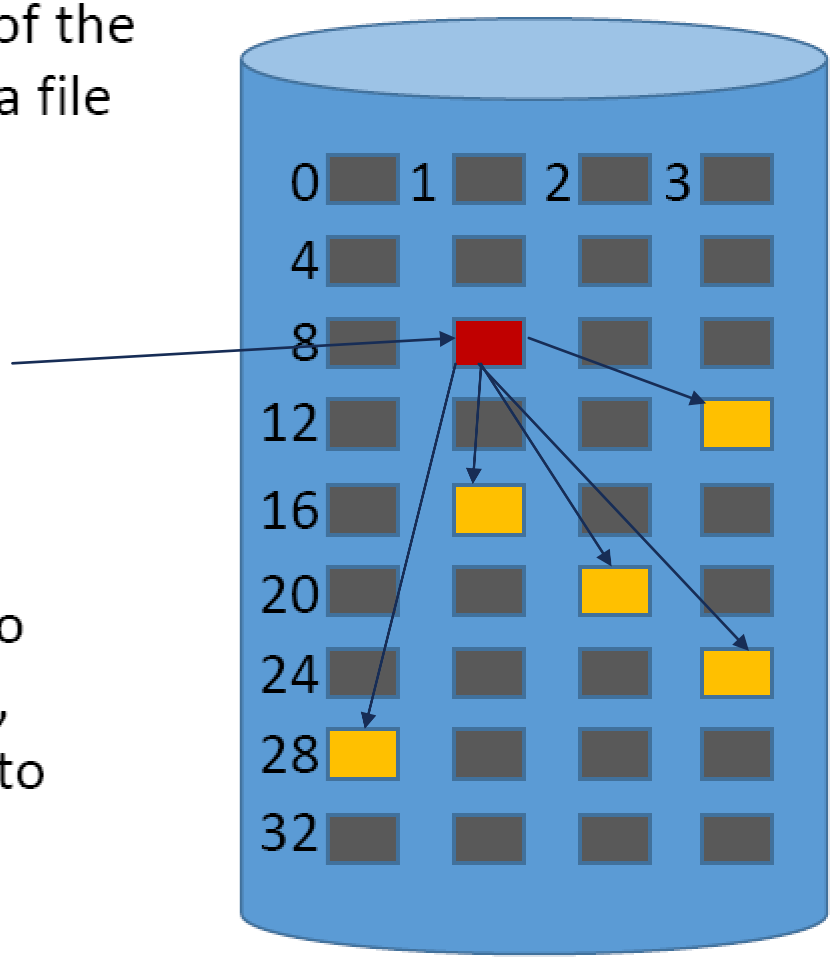
Index Allocation

A single index block contains ALL of the pointers for the blocks in use by a file

Block 9
17
22
27
28
15
-1
-1
-1

Directory entry

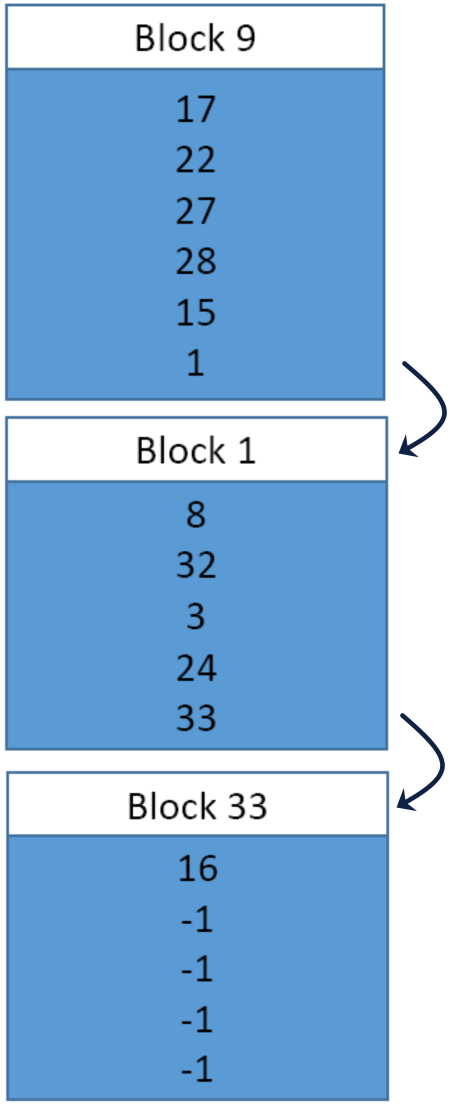
aFile	other	9



The block entry in the directory metadata points to the index block of file *aFile*, which contains the pointers to the data blocks for the file

Q: What if the index table is larger than a single block?

INDEX ALLOCATION



Block 9

17
22
27
28
15
1

Block 1

8
32
3
24
33

Block 33

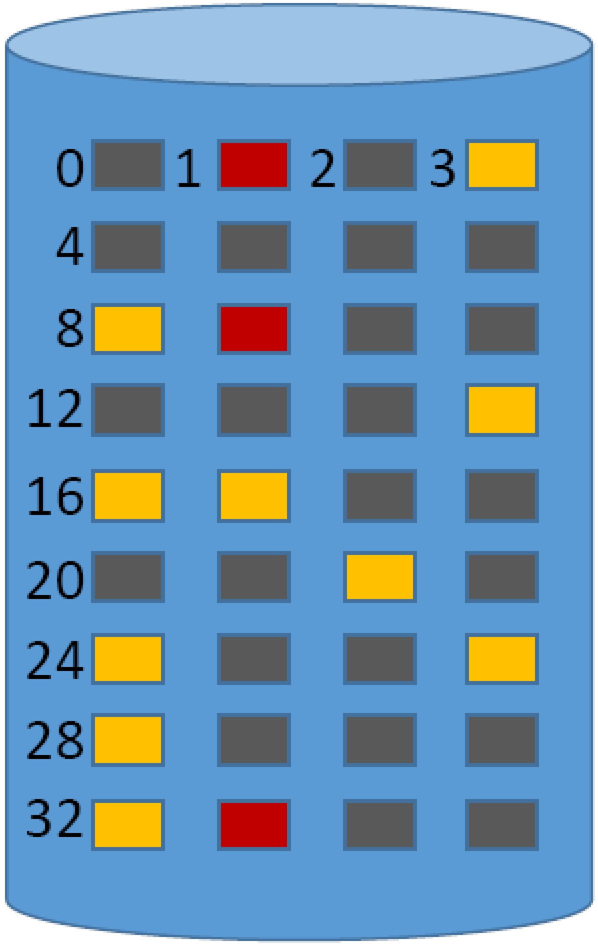
16
-1
-1
-1
-1

A block contains references (addresses) to data blocks, and the last reference in a block refers to ANOTHER index block

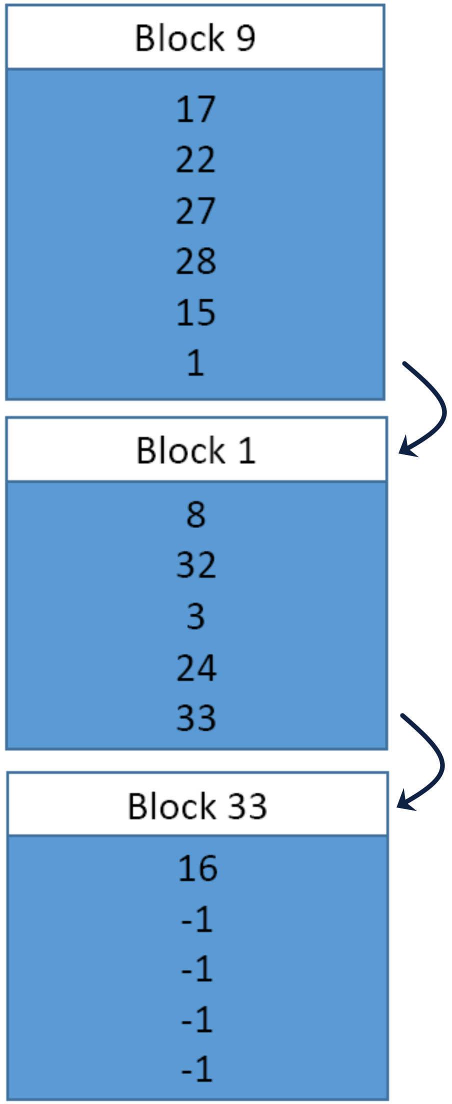
Directory entry

aFile	other	9

The “last” index block in the list of index blocks contains -1s to indicate “no more”



INDEX ALLOCATION


Block 9

17
22
27
28
15
1

Block 1

8
32
3
24
33

Block 33

16
-1
-1
-1
-1

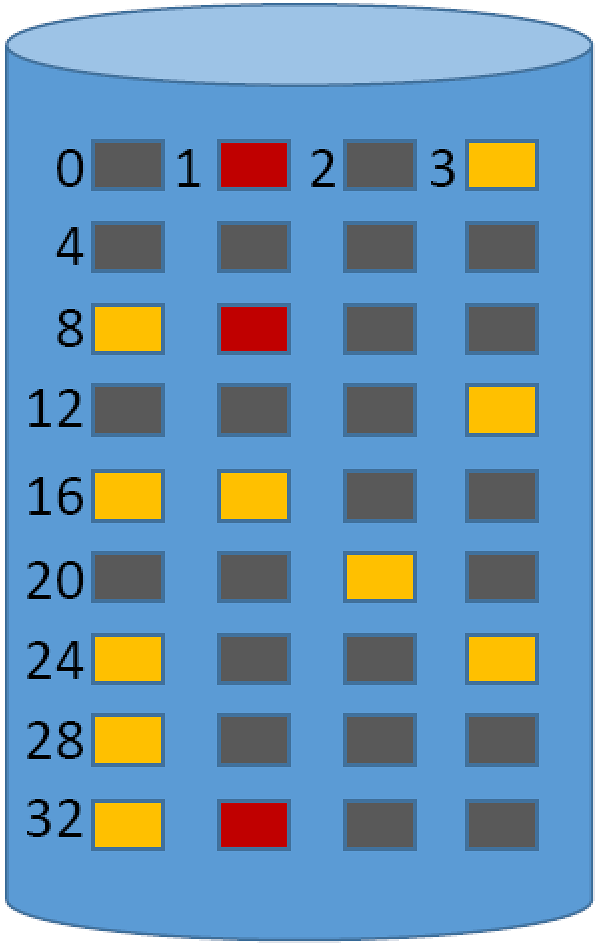
A block contains references (addresses) to data blocks, and the last reference in a block refers to ANOTHER index block

Directory entry

aFile	other	9

The "last" index block in the list of index blocks contains -1s to indicate "no more"

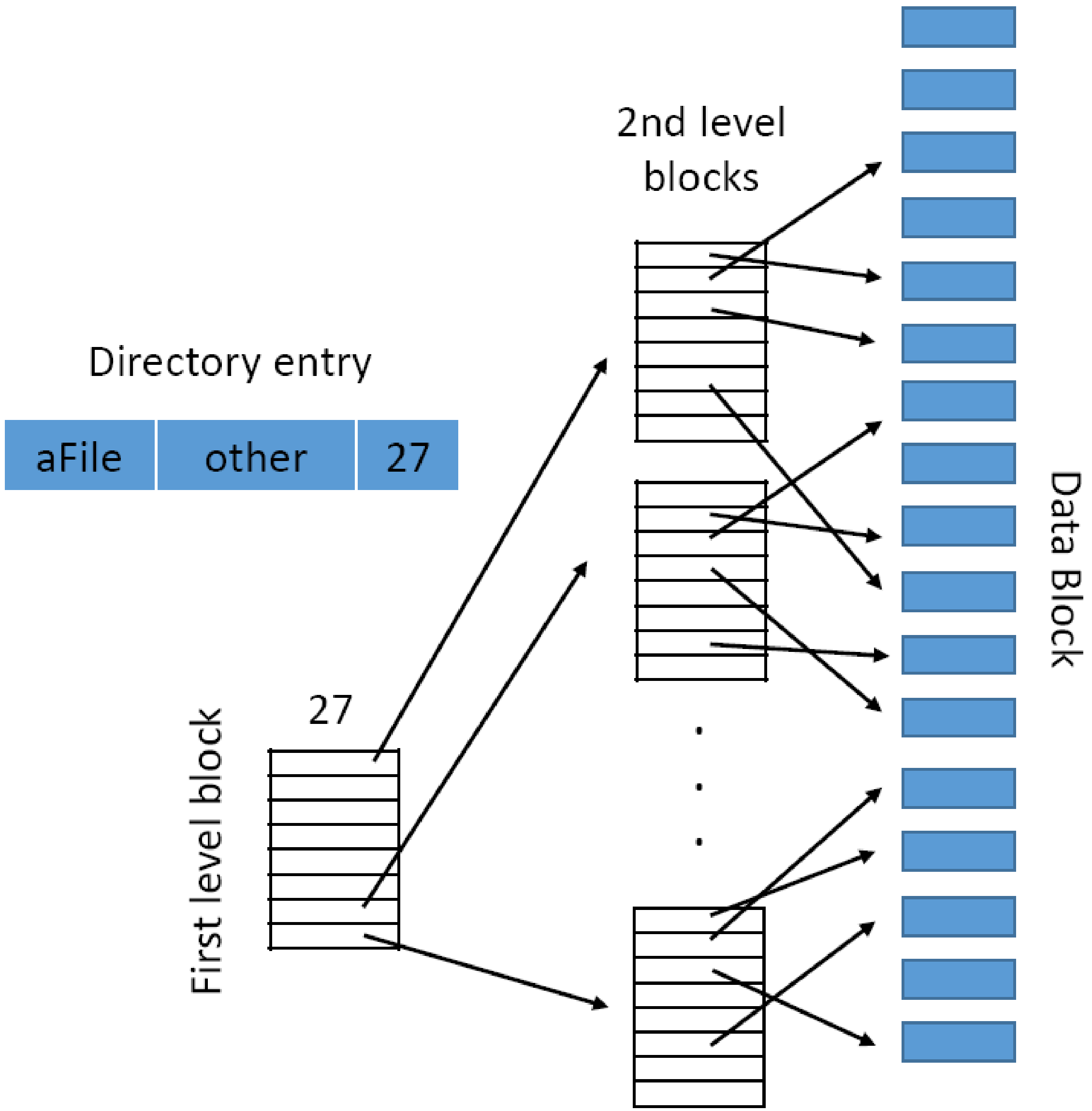
Sequential Access?
Direct Access?



MULTI-LEVEL INDEXING

Multilevel Index blocks

The first level contains references to second level index blocks, and THEY contair the references to data blocks.

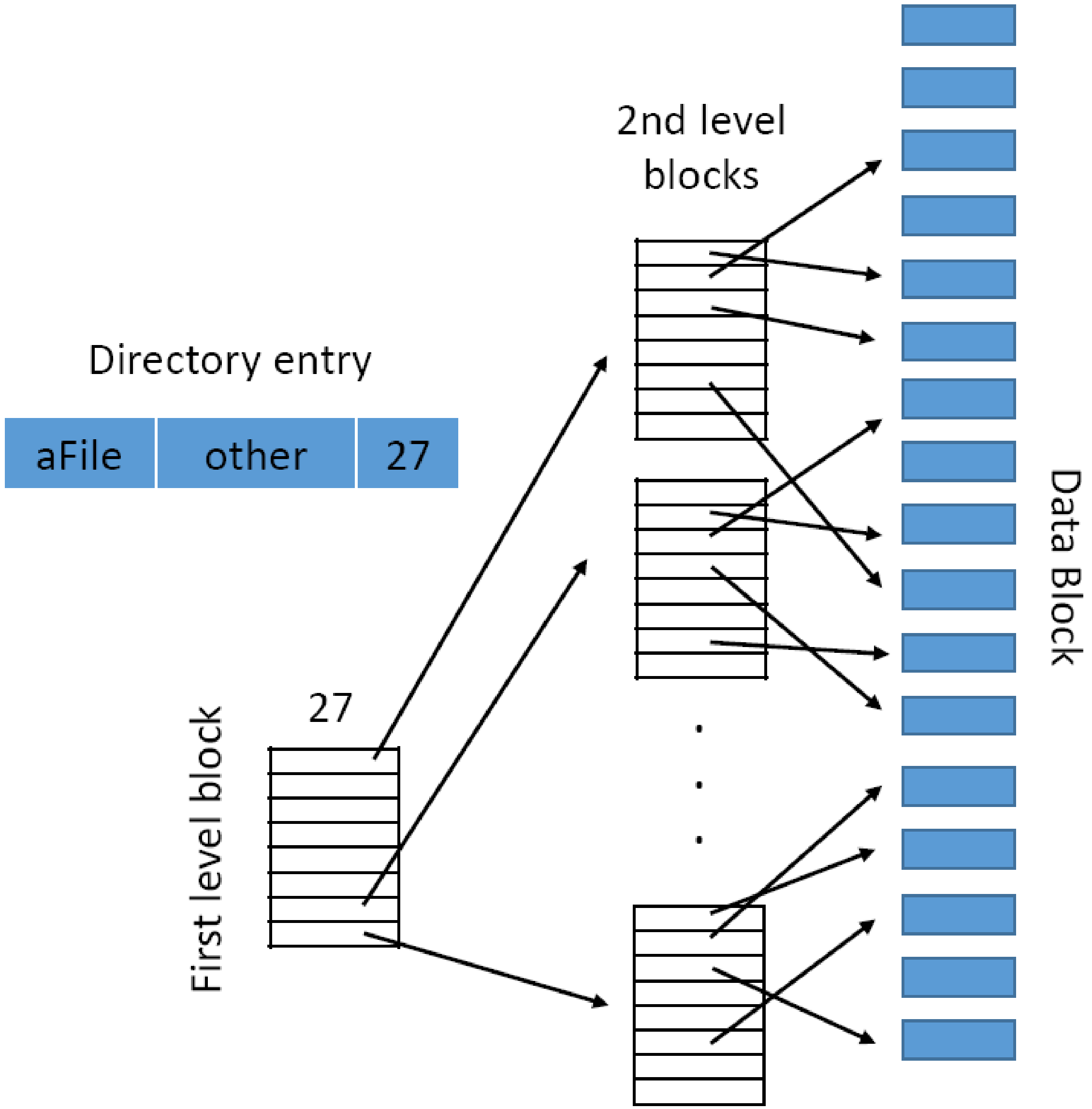


MULTI-LEVEL INDEXING

Multilevel Index blocks

The first level contains references to second level index blocks, and THEY contair the references to data blocks.

- **Allows for fast direct access.**

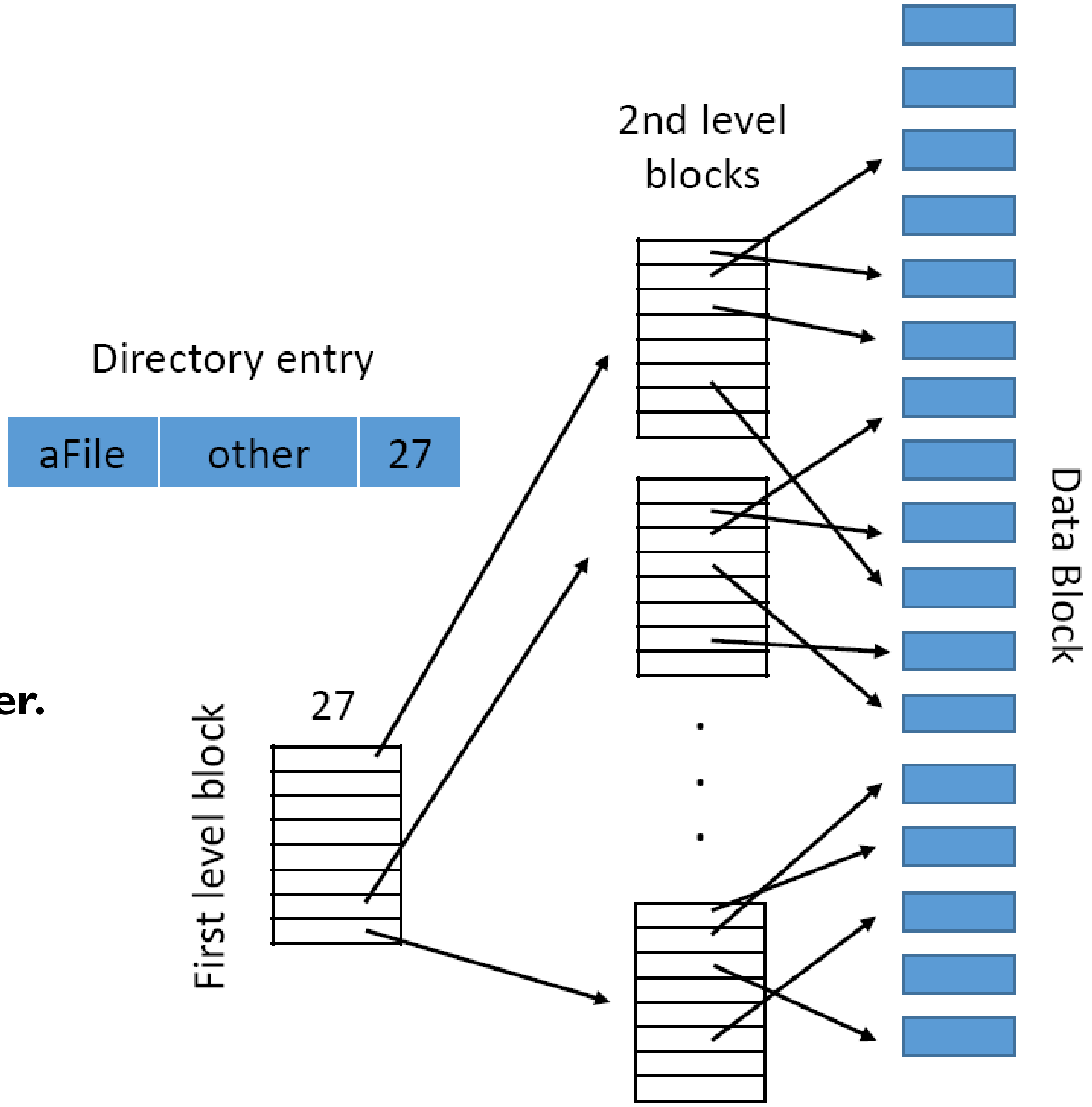


MULTI-LEVEL INDEXING

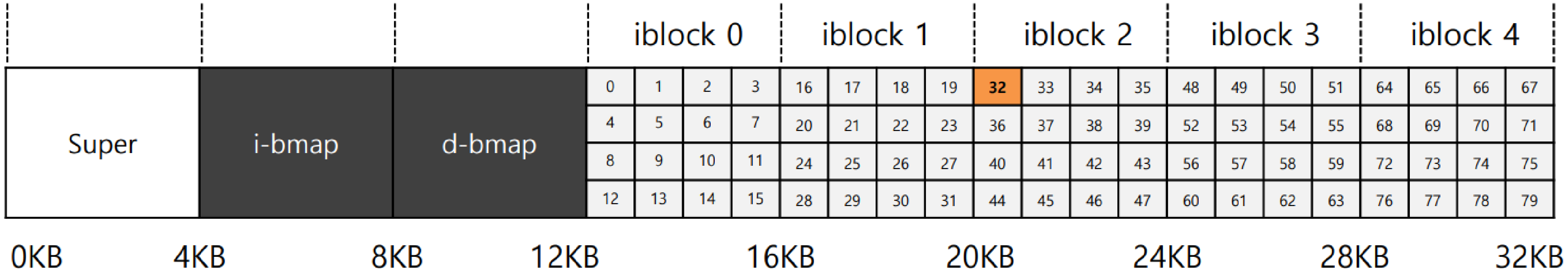
Multilevel Index blocks

The first level contains references to second level index blocks, and THEY contair the references to data blocks.

- **Allows for fast direct access.**
- **Sequential access can be slightly slower.**

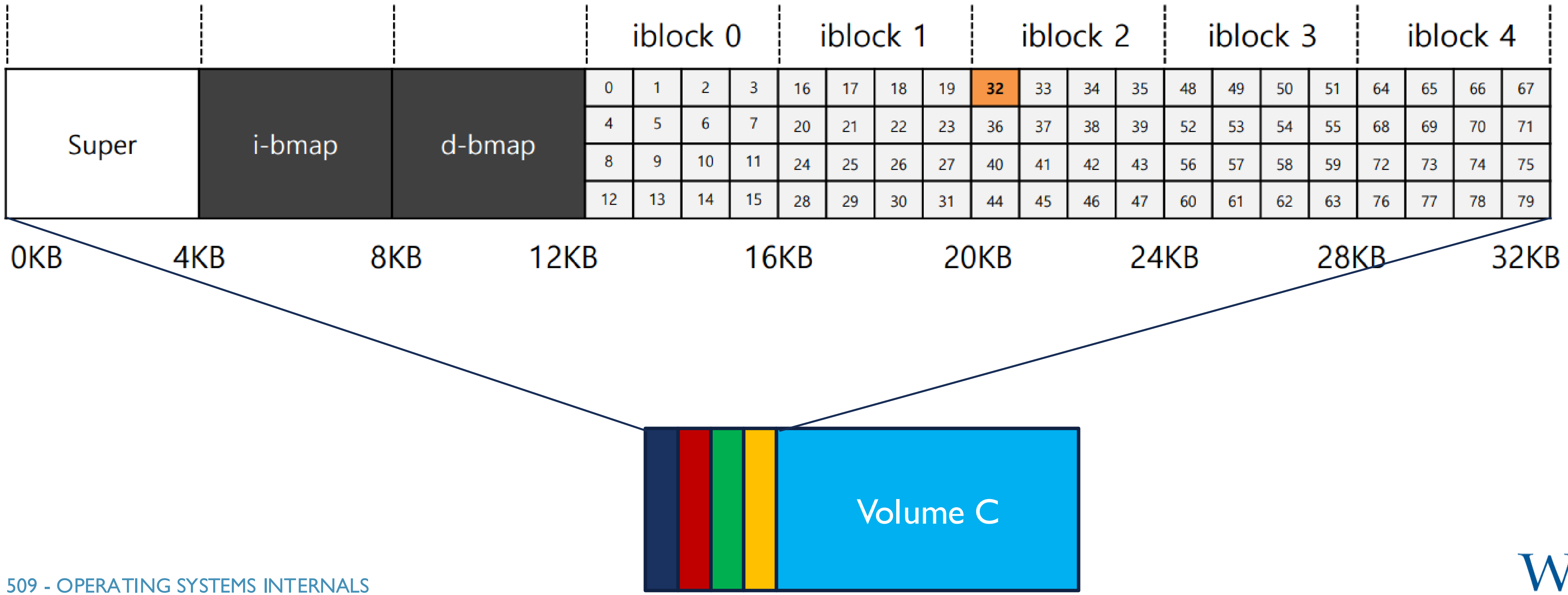


LINUX FILE SYSTEMS



iblock 0				iblock 1				iblock 2				iblock 3				iblock 4			
0	1	2	3	16	17	18	19	**32**	33	34	35	48	49	50	51	64	65	66	67
4	5	6	7	20	21	22	23	36	37	38	39	52	53	54	55	68	69	70	71
8	9	10	11	24	25	26	27	40	41	42	43	56	57	58	59	72	73	74	75
12	13	14	15	28	29	30	31	44	45	46	47	60	61	62	63	76	77	78	79

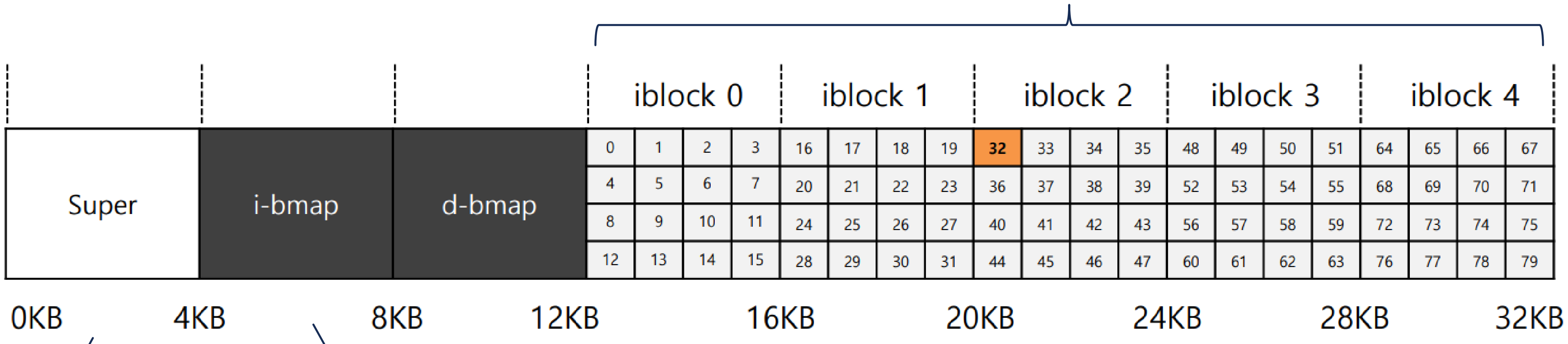
LINUX FILE SYSTEMS



iblock 0				iblock 1				iblock 2				iblock 3				iblock 4			
0	1	2	3	16	17	18	19	**32**	33	34	35	48	49	50	51	64	65	66	67
4	5	6	7	20	21	22	23	36	37	38	39	52	53	54	55	68	69	70	71
8	9	10	11	24	25	26	27	40	41	42	43	56	57	58	59	72	73	74	75
12	13	14	15	28	29	30	31	44	45	46	47	60	61	62	63	76	77	78	79

LINUX FILE SYSTEMS

inodes can be spread around
the disk



			iblock 0				iblock 1				iblock 2				iblock 3				iblock 4			
Super	i-bmap	d-bmap	0	1	2	3	16	17	18	19	**32**	33	34	35	48	49	50	51	64	65	66	67
			4	5	6	7	20	21	22	23	36	37	38	39	52	53	54	55	68	69	70	71
			8	9	10	11	24	25	26	27	40	41	42	43	56	57	58	59	72	73	74	75
			12	13	14	15	28	29	30	31	44	45	46	47	60	61	62	63	76	77	78	79

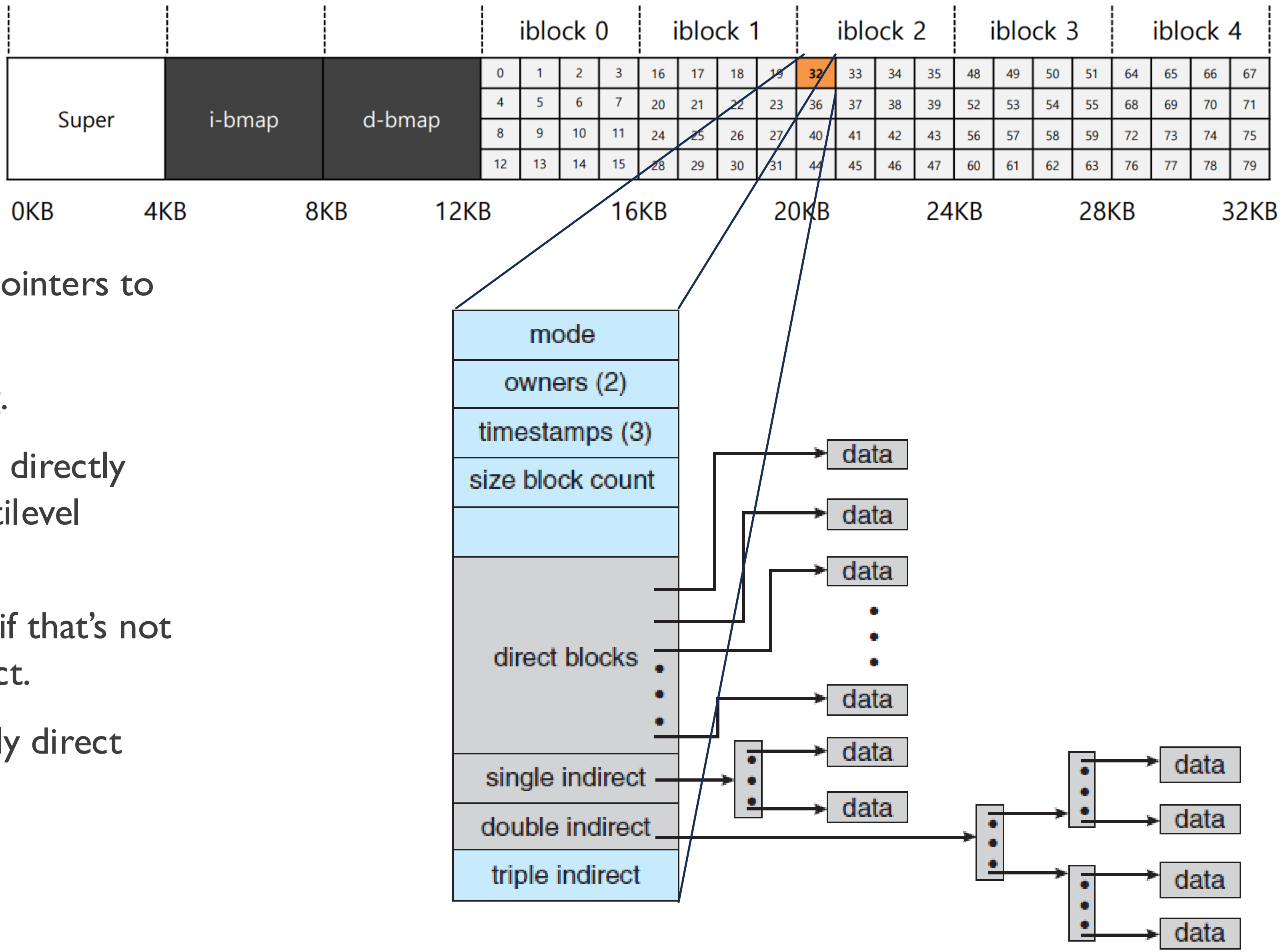
Contains volume
information and
directory structure

Which inodes
are 'vacant'

Which data
blocks are vacant

LINUX INODE

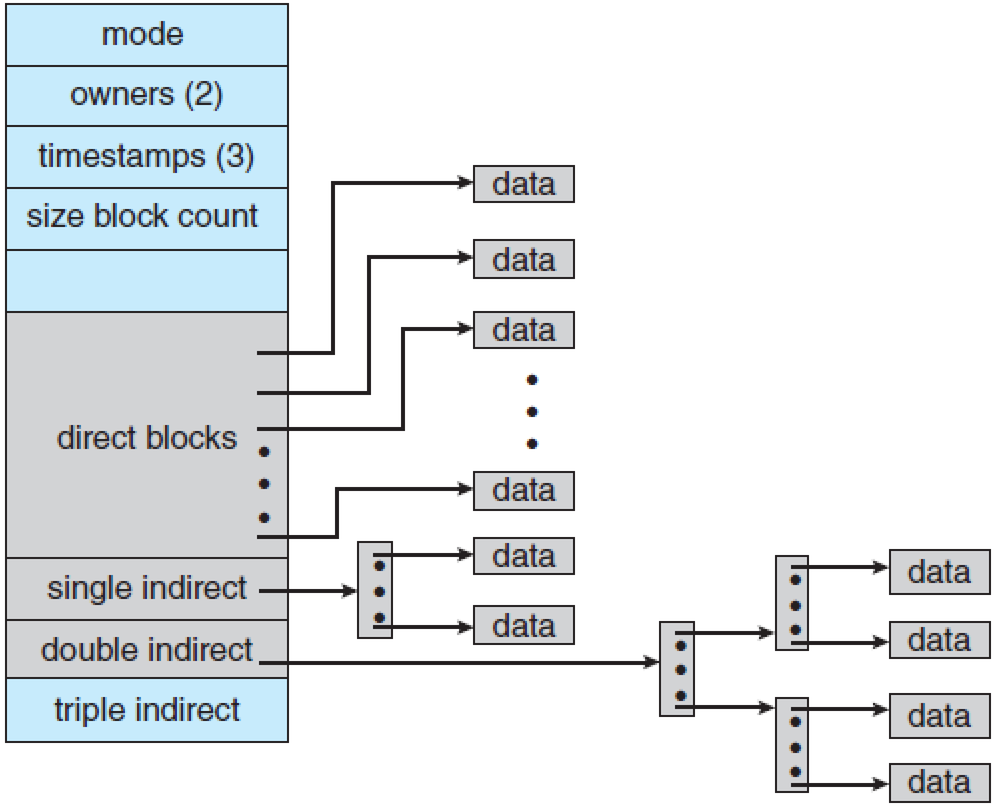
- Contain both metadata and pointers to blocks used.
- Uses various type of indexing.
- First blocks can be addressed directly others could have utilize multilevel indexing.
- Start with using direct block, if that's not enough for the file use indirect.
- Most files are small and usually direct blocks would suffice.



iblock 0				iblock 1				iblock 2				iblock 3				iblock 4			
0	1	2	3	16	17	18	19	32	33	34	35	48	49	50	51	64	65	66	67
4	5	6	7	20	21	22	23	36	37	38	39	52	53	54	55	68	69	70	71
8	9	10	11	24	25	26	27	40	41	42	43	56	57	58	59	72	73	74	75
12	13	14	15	28	29	30	31	44	45	46	47	60	61	62	63	76	77	78	79

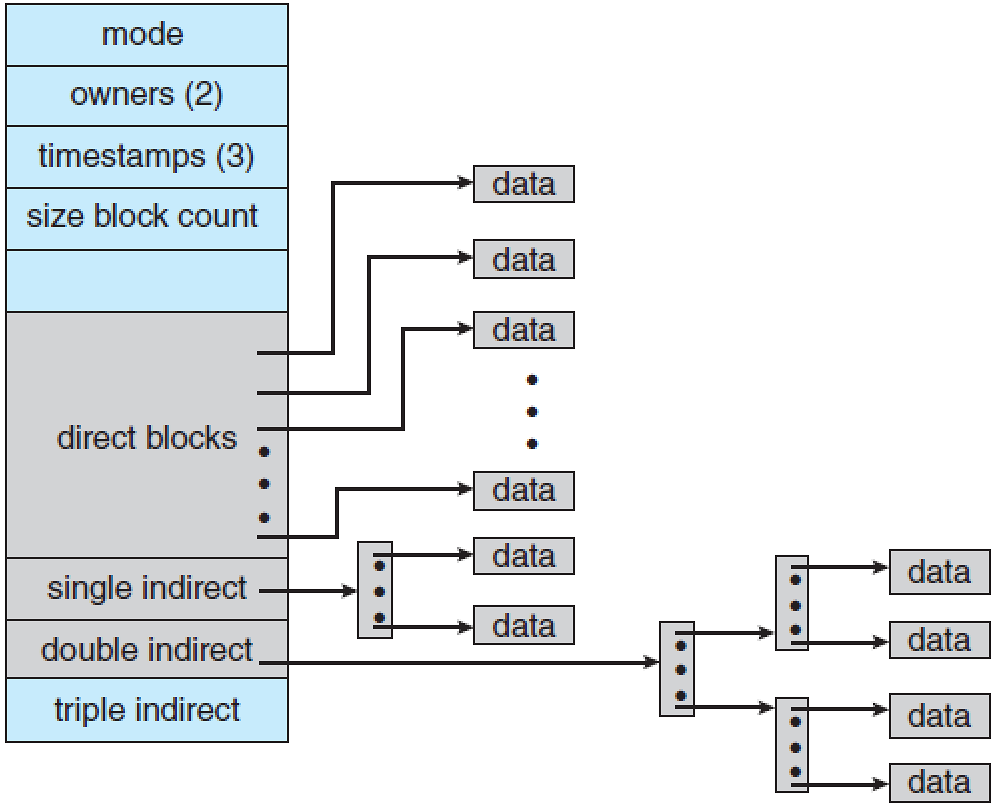
WORKSHEET

- In an ext2 file system an inode consists of only 15 block pointers.
- The first 12 block pointers are direct block pointers.
- The 13th pointer is an indirect pointer.
- The 14th pointer is a double indirect pointer.
- The 15th pointer is a triple indirect pointer.
- Block size of 4KB
- 32-bit addressing for the blocks
- Which of these pointers will be utilized when the inode represents a file of size 64 KB?
- Which of these pointers will remain unutilized?



WORKSHEET

- How many blocks do we need for the file?
- How much "size on disk" does each direct block pointer support?
- How much "size on disk" does a single indirect pointer can support?
  - How many direct pointers can a block on disk hold?



mode
owners (2)
timestamps (3)
size block count
direct blocks
single indirect
double indirect
triple indirect
data
data
data
data
data
data
data
data
data
data