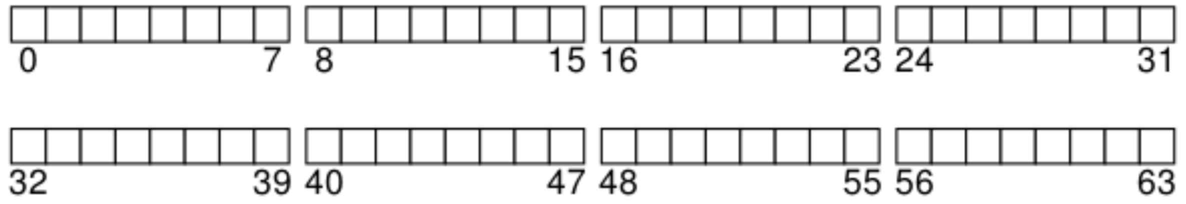
OSTEP Chapter 40

ECE 3600, Fall 2022

Table of Contents

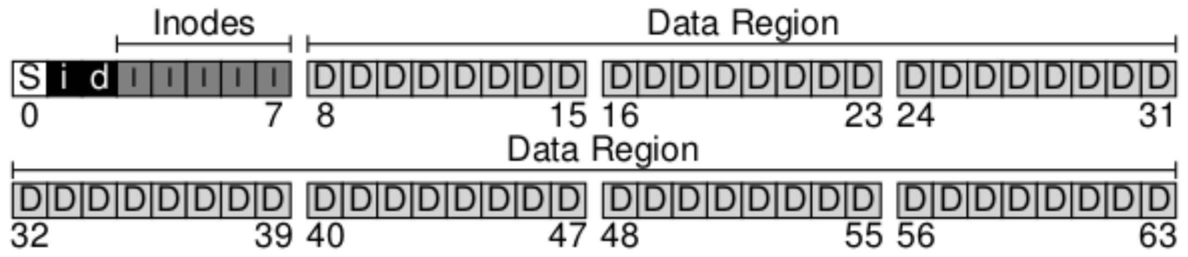
1. File System Implementation

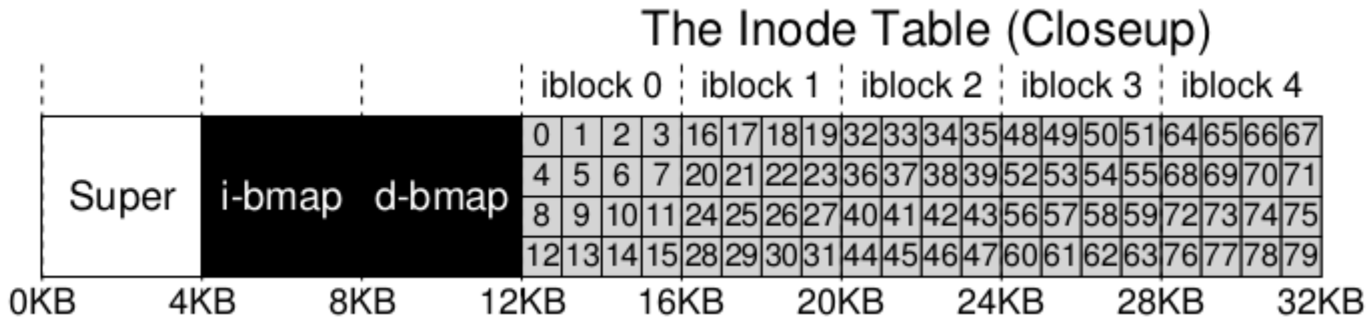
Example with 64 blocks, block size 4 KB:



One block each for superblock, inode bitmap, data bitmap.

5 blocks for inode table (256 bytes per inode, 16 inodes per block), 56 blocks for file data:



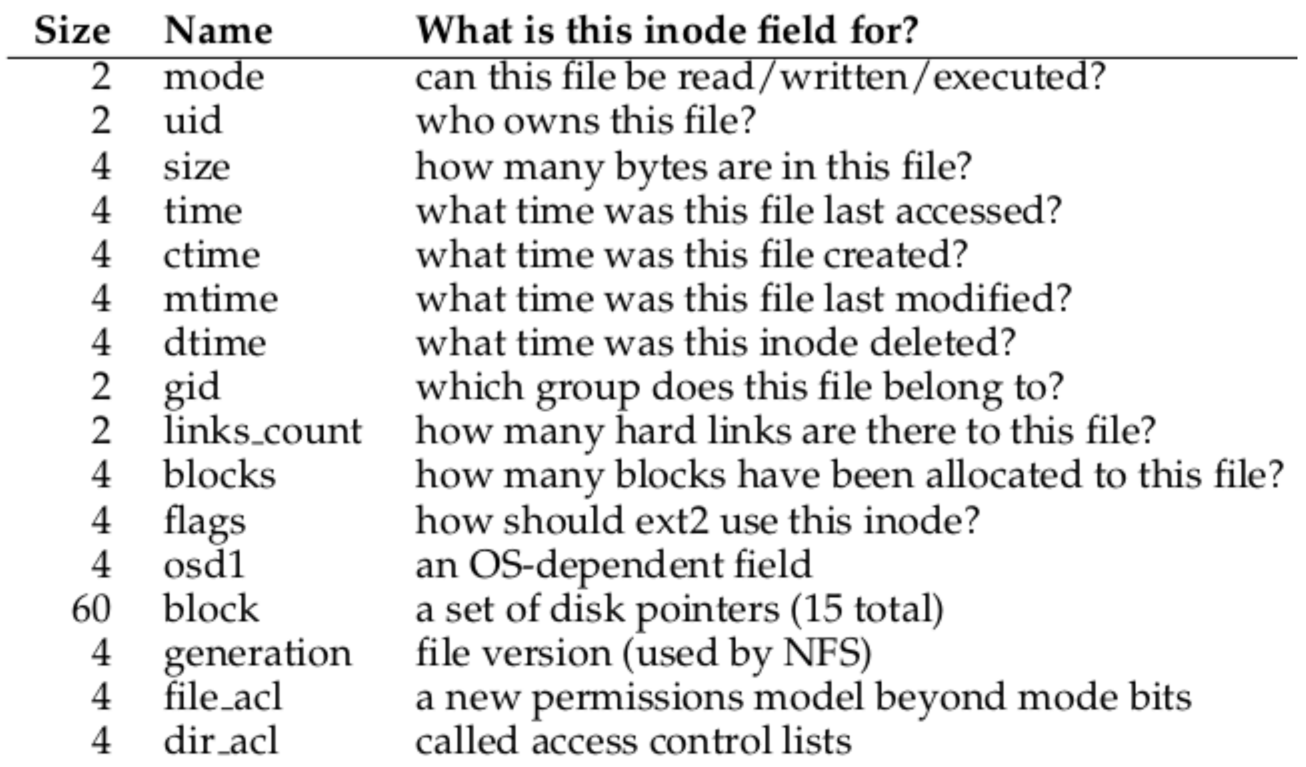


Reading an inode: `address = 12KB + 256 * inumber; block = address / 4KB; offset = address % 4KB`

2. Inode Contents

stat() shows subset of the inode contents; also see inode(7)

mode includes file type



Size	Name	What is this inode field for?
2	mode	can this file be read/written/executed?
2	uid	who owns this file?
4	size	how many bytes are in this file?
4	time	what time was this file last accessed?
4	ctime	what time was this file created?
4	mtime	what time was this file last modified?
4	dtime	what time was this inode deleted?
2	gid	which group does this file belong to?
2	links_count	how many hard links are there to this file?
4	blocks	how many blocks have been allocated to this file?
4	flags	how should ext2 use this inode?
4	osd1	an OS-dependent field
60	block	a set of disk pointers (15 total)
4	generation	file version (used by NFS)
4	file_acl	a new permissions model beyond mode bits
4	dir_acl	called access control lists

Figure 40.1: **Simplified Ext2 Inode**

Multi-level indexing for larger files: indirect block pointers, double indirect, triple indirect

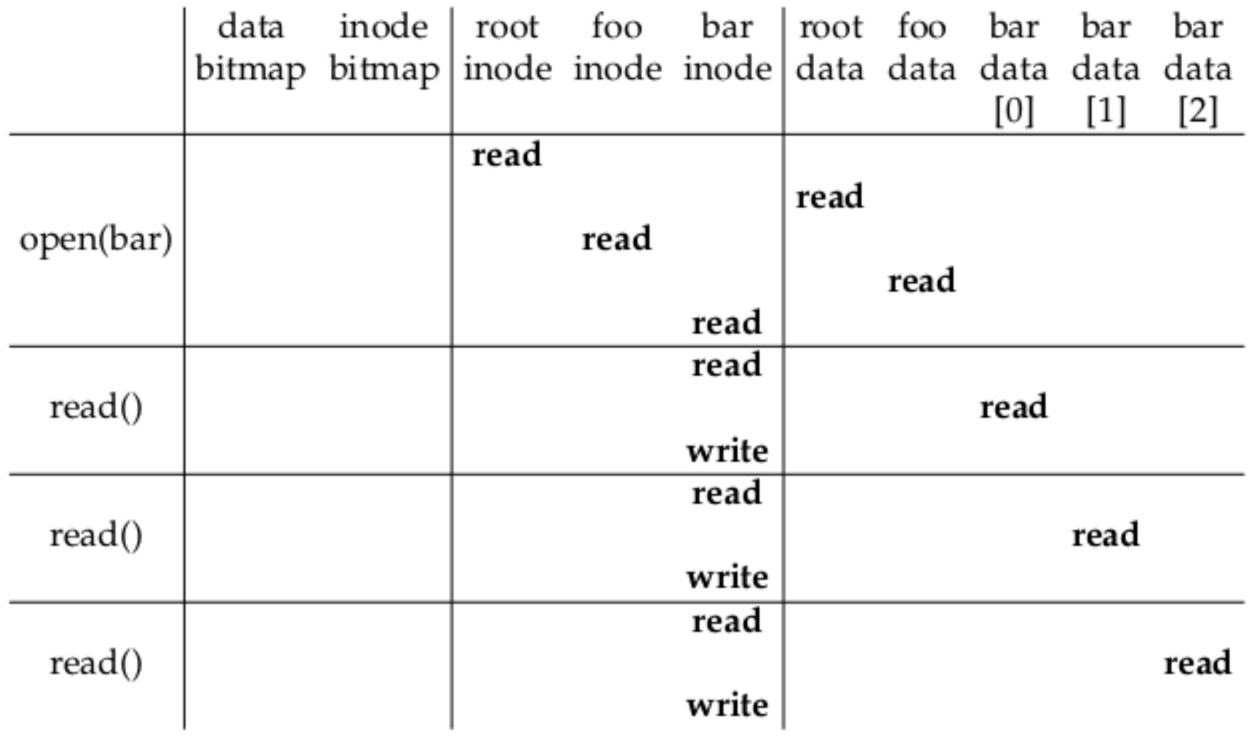
3. Directories

Directories are just files with a special structure.

inum	reclen	strlen	name
5	12	2	.
2	12	3	..
12	12	4	foo
13	12	4	bar
24	36	28	foobar_is_a_pretty_longname

4. Open and Read Access Paths

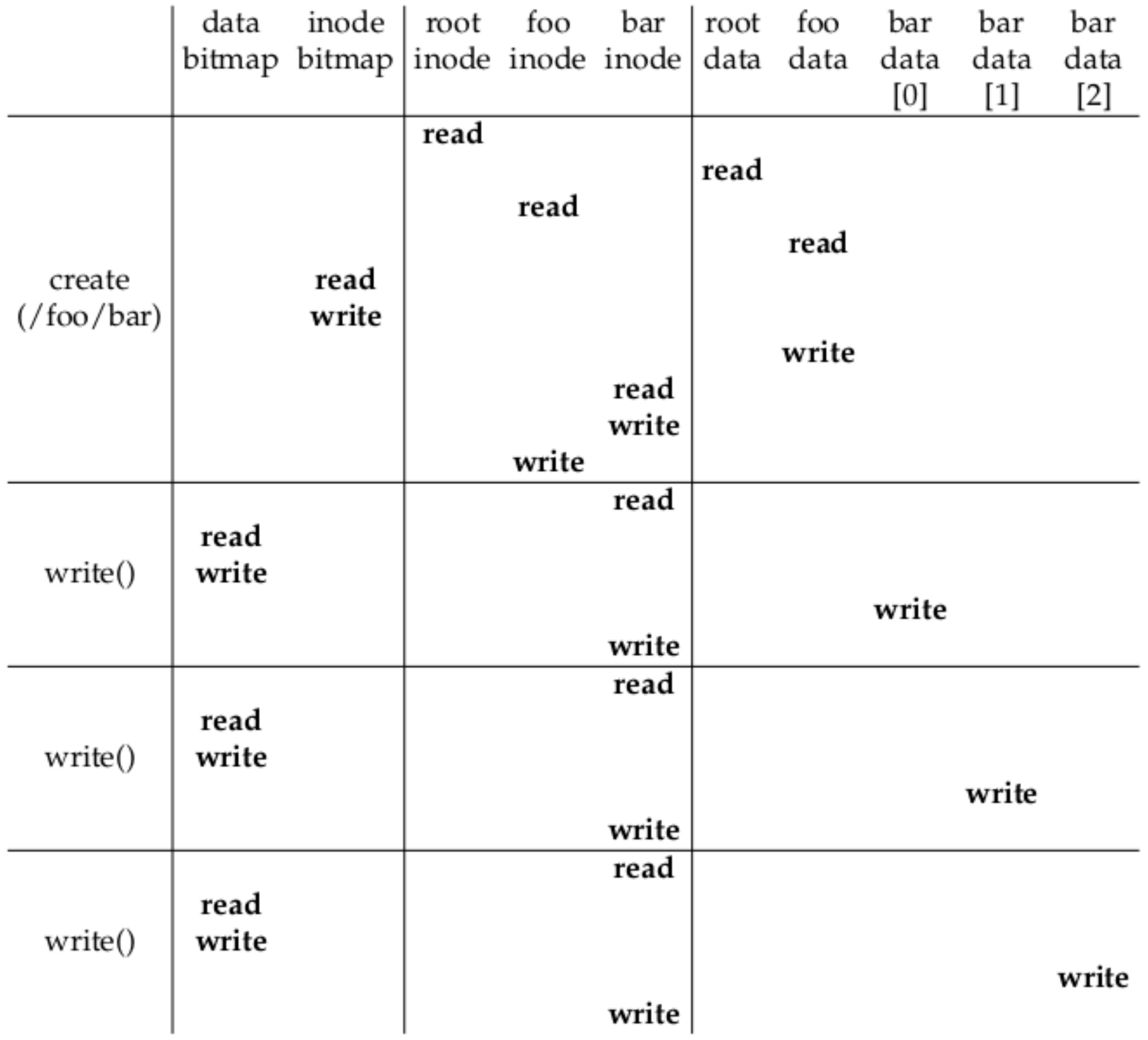
```
open("/foo/bar", O_RDONLY)
```



	data bitmap	inode bitmap	root inode	foo inode	bar inode	root data	foo data	bar data [0]	bar data [1]	bar data [2]
			read							
						read				
open(bar)				**read**						
							read			
					read					
					read					
read()								**read**		
					write					
					read					
read()									**read**	
					write					
					read					
read()										**read**
					write					

Figure 40.3: **File Read Timeline (Time Increasing Downward)**

5. Create and Write Access Paths



	data bitmap	inode bitmap	root inode	foo inode	bar inode	root data	foo data	bar data [0]	bar data [1]	bar data [2]
create (/foo/bar)			**read**							
						read				
				read						
							read			
		read								
		write								
							write			
					read					
					write					
				write						
write()					**read**					
	read									
	write									
								write		
					write					
write()					**read**					
	read									
	write									
									write	
					write					
write()					**read**					
	read									
	write									
										write
					write					

Figure 40.4: **File Creation Timeline (Time Increasing Downward)**

6. Exercises

Exercises from the book using vsfs.py:

```
$ python ./vsfs.py -n 4

Initial state

inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []

Which operation took place?

inode bitmap  11000000
inodes        [d a:0 r:3] [d a:1 r:2] [] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (g,1)] [(.,1) (..,0)] [] [] [] [] [] []

Which operation took place?

inode bitmap  11100000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (g,1) (q,2)] [(.,1) (..,0)] [] [] [] [] [] []

Which operation took place?

inode bitmap  11110000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [f a:-1 r:1] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (g,1) (q,2) (u,3)] [(.,1) (..,0)] [] [] [] [] [] []

Which operation took place?

inode bitmap  11110000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [f a:-1 r:2] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (g,1) (q,2) (u,3) (x,3)] [(.,1) (..,0)] [] [] [] [] [] []

$ python ./vsfs.py -n 4 -r

Initial state

inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []

mkdir("/g");  State of file system (inode bitmap, inodes, data bitmap, data)?

creat("/q");

creat("/u");

link("/u", "/x");
```