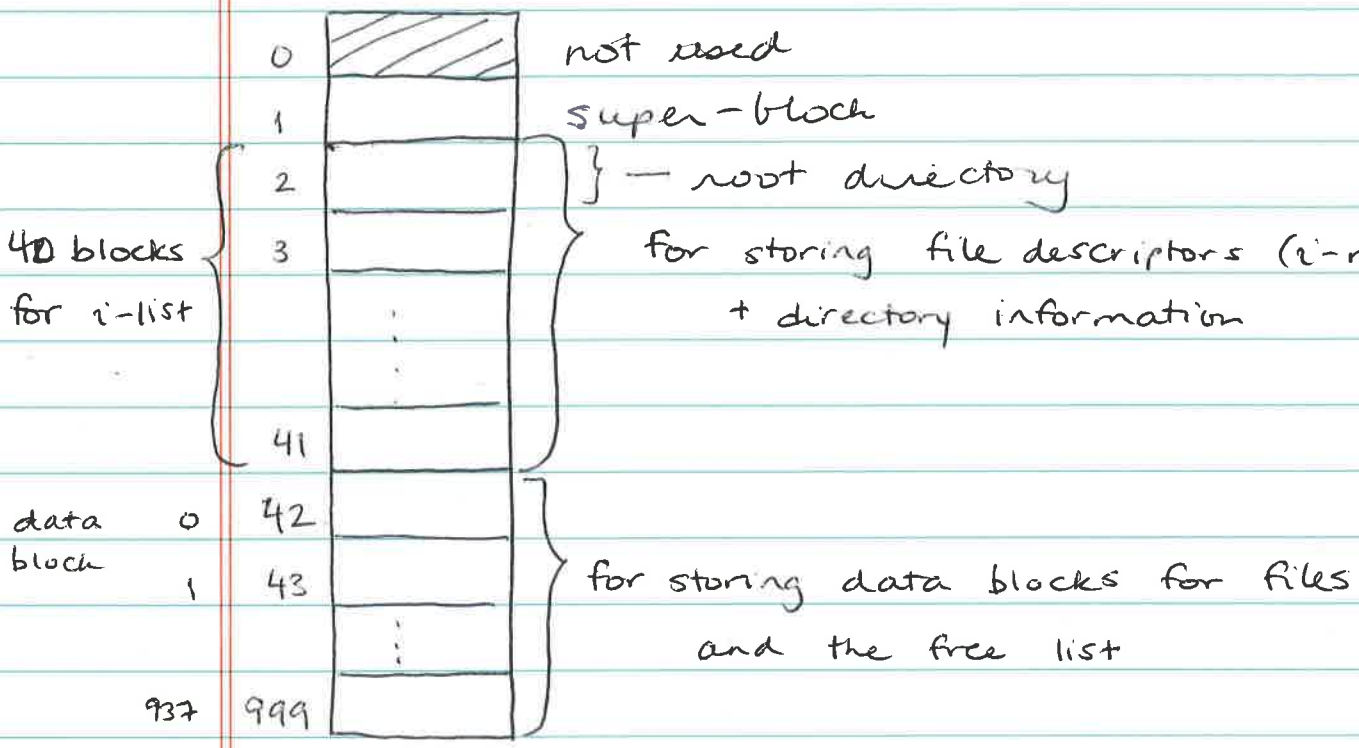
Unix V6 file system

1000 blocks of 512 bytes

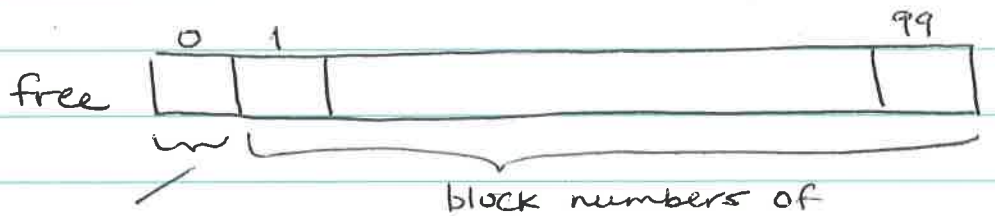


relevant fields of an super block

i-size : # blocks in the i-list, here 40

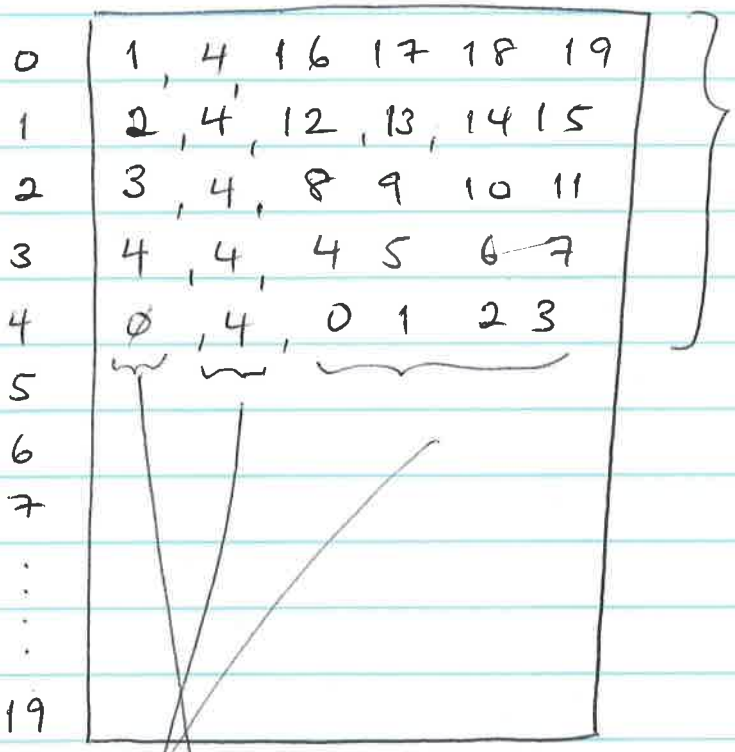
n-free : index into free

free[100] :



if = 0, no more free blocks

Suppose there were only 20 data blocks and that the free list was of length 5:



1st entry = block containg next set of 4 free blocks
= 0, then no more of them

2nd entry
= # free blocks in the list, nfree

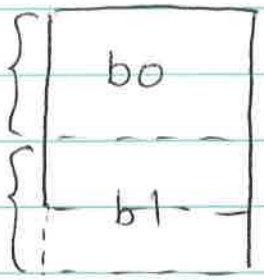
list of nfree free blocks

so free[5]

	0	1	2	3	4
free	1	16	17	18	19

nfree = 4

how to allocate a file f



suppose 720 bytes in size
since each block is 512 bytes need 2 blocks

free[4] = 19 , nfree = 4-1 = 3
copy b0 of f into block 19 , free[3] = 18
copy b1 of f into block 18 , nfree = 2

now allocate file g of size 1224 bytes
⇒ need 3 blocks
copy b0 of g into free[2] = block 17, nfree = 1
copy b1 of g into free[1] = block 16, nfree = 0
since nfree = 0 ⇒ block free[1] = 1 contains next free list

update

free	2	~~12~~	~~13~~	~~14~~	15

nfree = 4

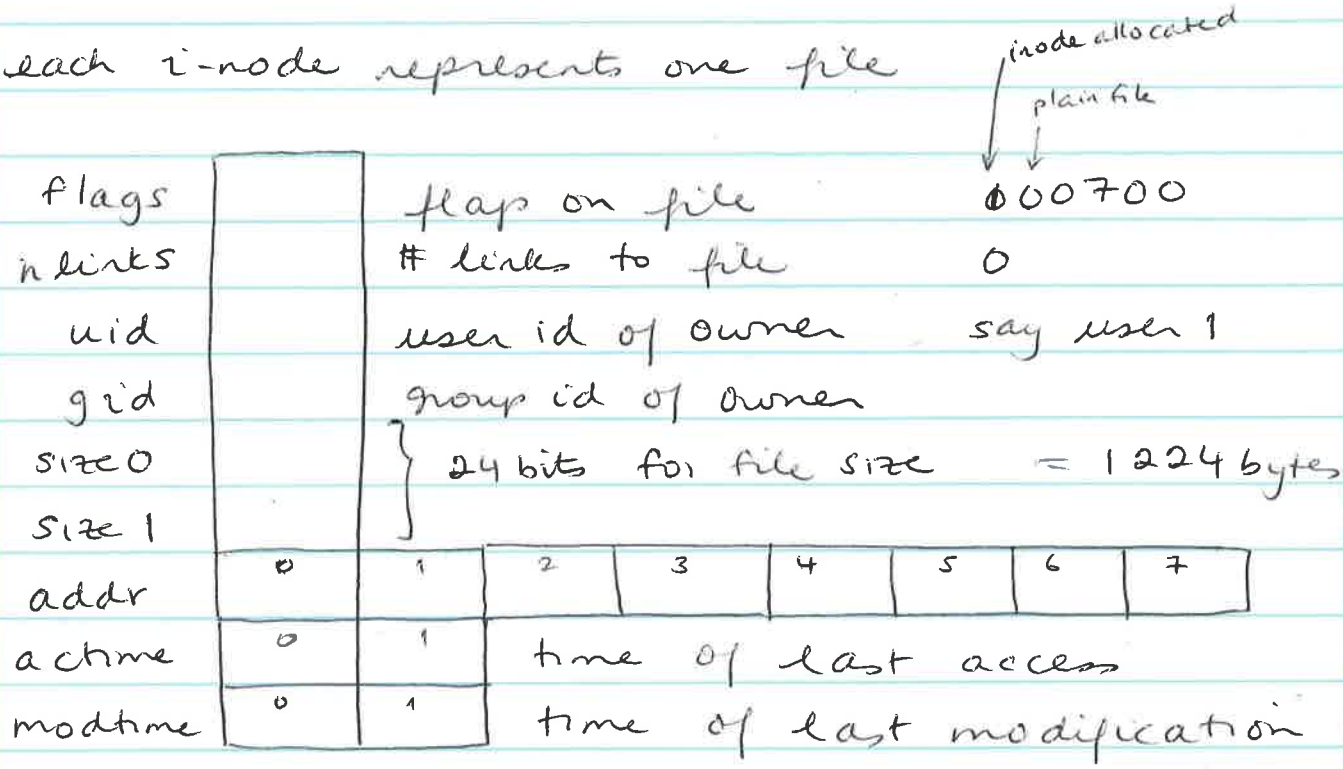
now allocate (copy) b2 of g into free[4] = block 15
nfree = 3

i-nodes (file descriptors) + directories
i-numbers begin at 1; storage starts in block 2

i-nodes are 32 bytes long so. (32 × 16 = 512)
16 of them fit in one block

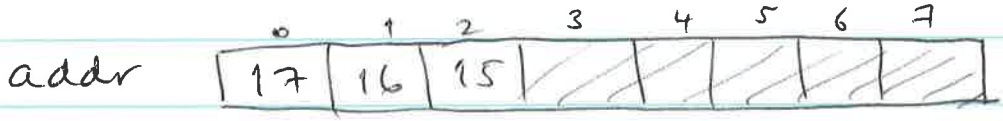
root directory in block 2

each i-node represents one file



will only deal with small files ($8 \times 512b = 4K$)
⇒ so addr[i], $0 \le i \le 7$ are direct address blocks, i.e, give block # of file contents

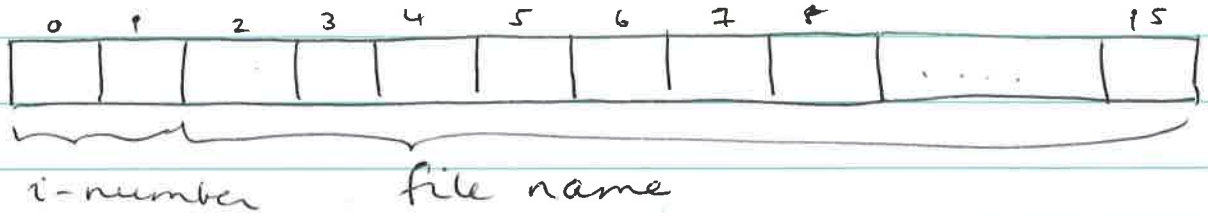
for previous example of file g



directory entries

indicated by flag bit of i-node entry

directory entries are 16 bytes long ⇒ 32 in one block



if = 0 then entry is empty
o/w it is the file number

by convention 1st 2 entries in each directory are . and ..

. → current
.. → parent

in block 2 is the root directory
i.e, can contain 16 i-nodes
flags has 140777 (inode alloc, directory, r/w/x world)
addr array for a directory is contents of the directory, i.e, if addr[0] = 57 then block 57 contains directory entries (up to 32 of them)

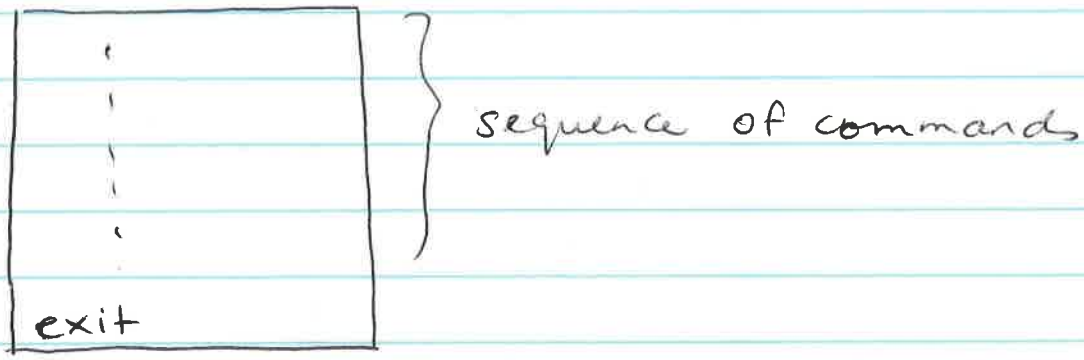
	size	
0		.
1		..
2		/
3	⋮	⋮

- main thread

u6fs disk k

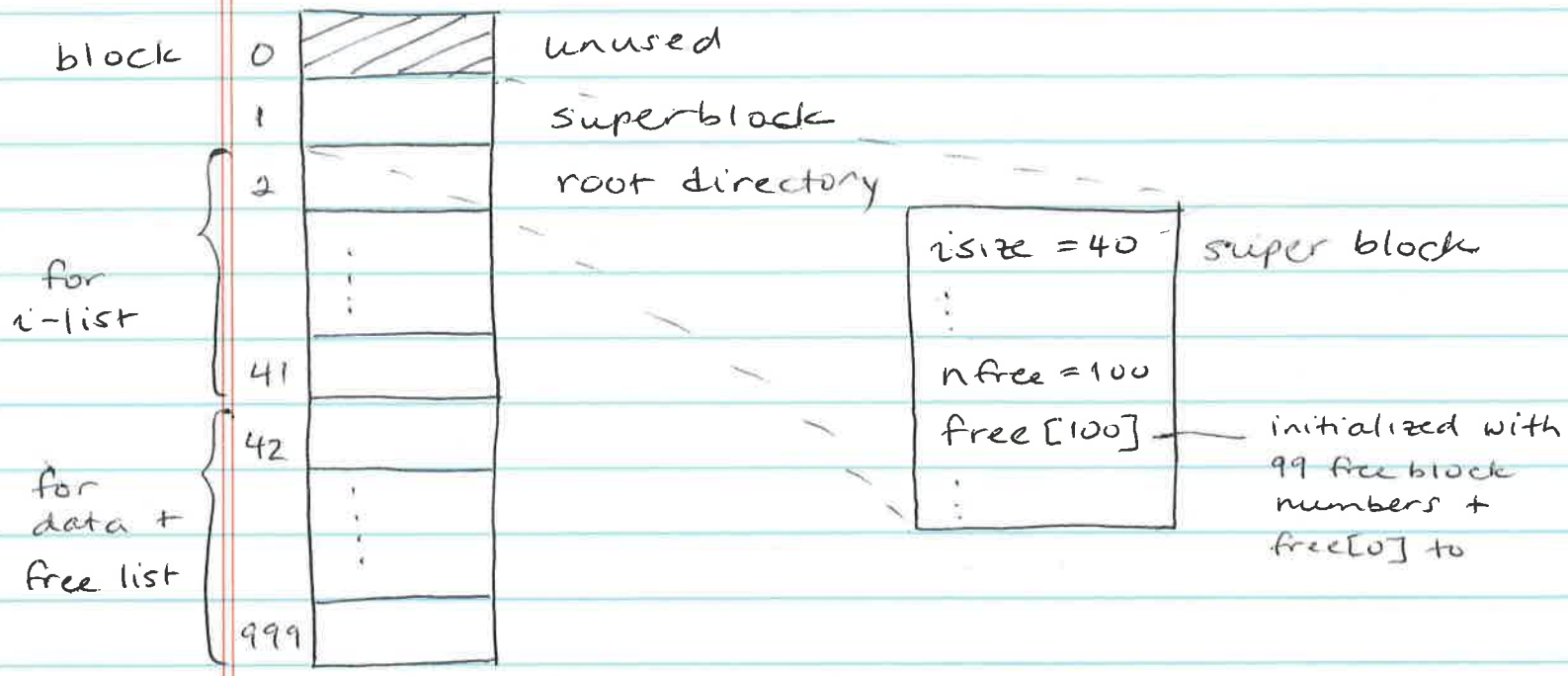
initializes the file system
spawns k threads
each thread i, $1 \leq i \leq k$

file user i.txt



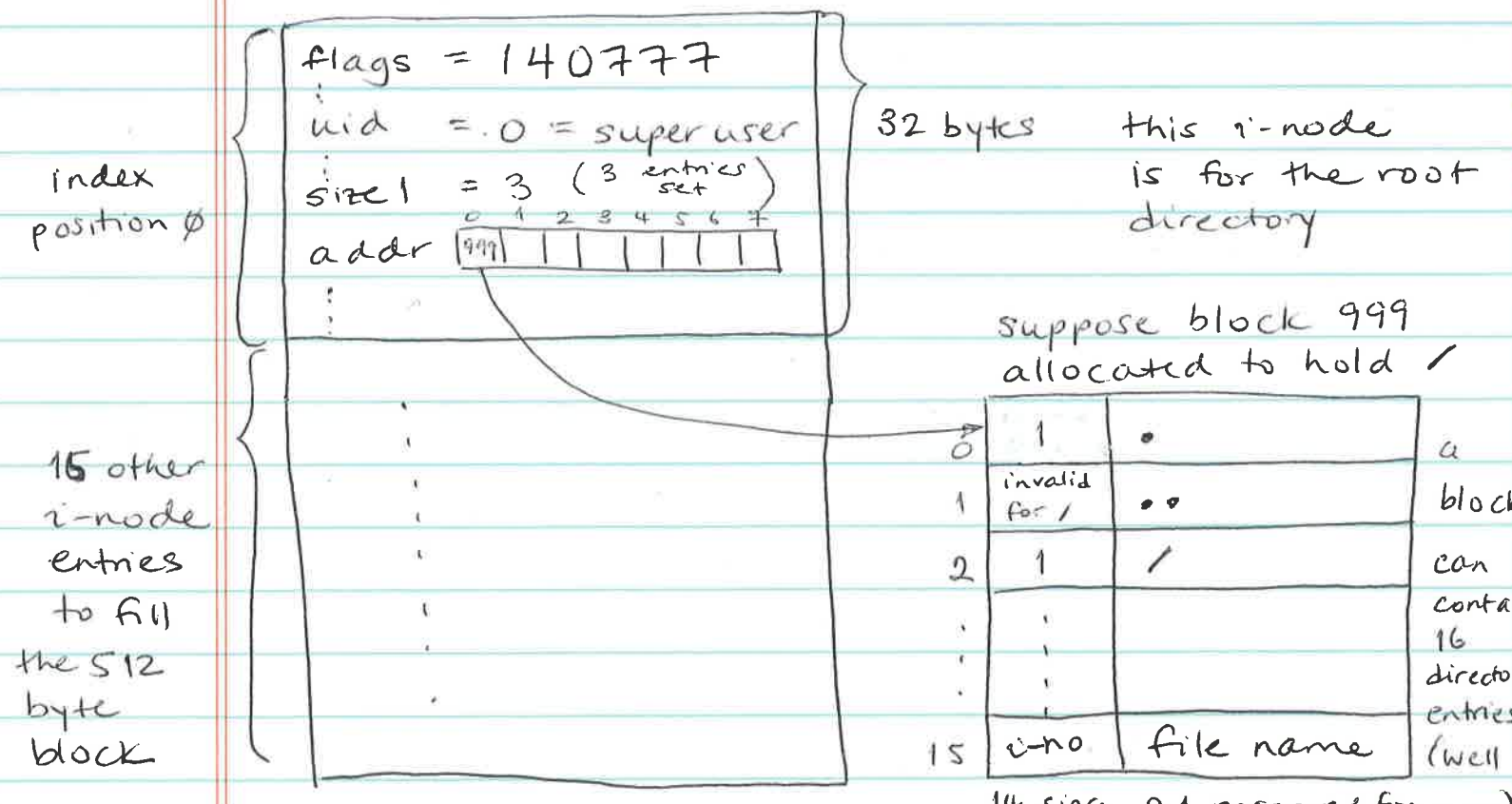
- file system is allocated in memory
  - work on it in memory

After file system is initialized it will contain



root directory (block 2)

overlay an array of 16 i-nodes onto block 2



First, suppose a user1 creates a directory d1; since the current directory is initialized as /, d1 gets created in /

To create a directory:

1) Create an entry in the current directory for it
   ⇒ into block 999 (holding /)

block 999

0	1	.
1	-1	..
2	1	/
3	2	d1
	⋮	

also update size1 of / i-node to be 4 (for 4 entries are set)

assume i-number = -1 means invalid (can't go up from /)

2) Create an i-node for the directory d1
   ⇒ into block 2, add an i-node for d1 at position 1

block 2

0 { update size1 = 4 } i-node for /

index pos. 1 { flags = 140777
uid = 1
size1 = 2 (2 entries set)
addr [998 | | | | | | |]
⋮ } i-node for d1

14 other i-node entries to fill

Suppose block 998 allocated for d

0	2	.
1	1	..

directory is empty

Note that creating the directory allocated a block to hold entries for the directory (initialized with . and ..)

Suppose user 1 now did cd to d1, and then executed "cpin f1.txt f1v6.txt"

The cd now has block 998 as the current directory. Hence the cpin will create the directory entry for the file here

block 998 (directory d1)

also update size1 of d1 to 3

2	.
1	..
3	f1v6.txt
⋮	⋮

1) First create a directory entry for the file
2) Create an i-node for the file (suppose f1.txt has 720 bytes)

block 2

i-node

0	for /
1	for d1
2	flags = 100777 uid = 1 size1 = 720 addr [997][996] ...

inode for f1v6.txt

block 997

1st 512b of f1.txt

1st block of f1v6.txt

block 996

720 - 512 = 208 bytes

remaining bytes of f1v6.txt

internal fragmentation

If move back up to / and create 1 entries in block 999, would need to allocate another block to directory entries and update addr[1] in block 2 (1st i-node in block 2)

For directories size1 = number of directory entries in the block

For files, size1 = the file size in bytes; since our max. file size is 4K, the size0 field is unused.