- [x] 1. Last time
- [x] 2. Directories
- [x] 3. Performance

---

2. Directories

(see sheet at the end)

- Intro to directories
- Hierarchical Unix

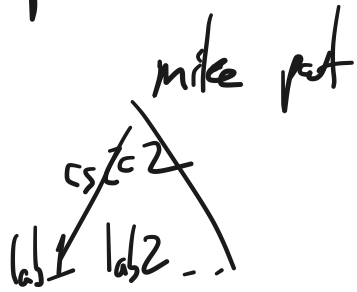
/usr/mike/cs202/lab2/...

$ ls /

/

bin dev sbin tmp usr

ls grep



Directory is a special kind of file:

struct dirent

name	inode #
bin	1021
dev	1001
sbin	2011
⋮	

← can be another directory. This turns the FS into a hierarchical tree.

this data (the table) can either live in the data blocks of a file (as in lab 5) or else in the inode of the directory (as in many of the examples that we will go over).

bootstrapping: root dir (/) always inode #2

special names: /, ., ..

can navigate the name space with:

$ cd name

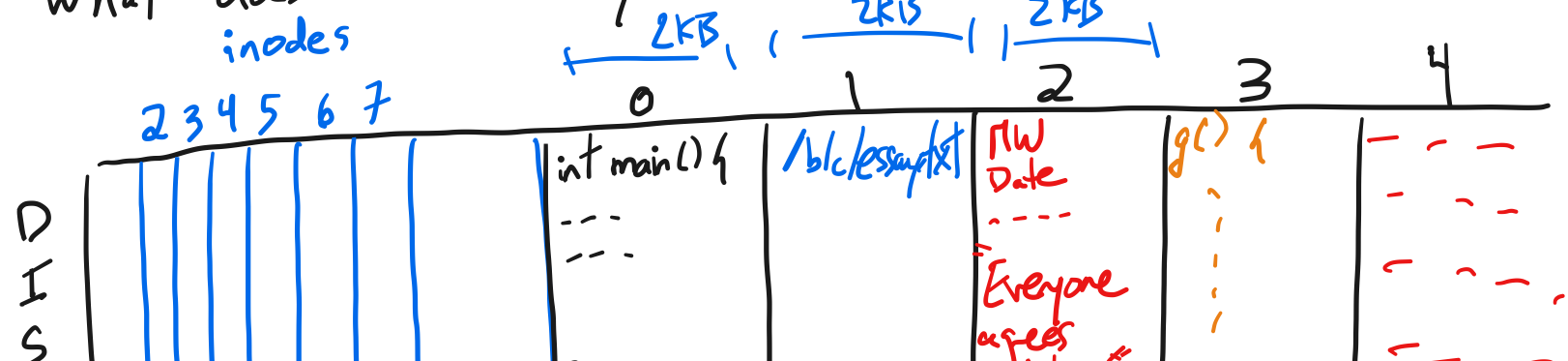
$ ls

Example: two files

/a/foo.c

/b/c/essay.txt

what does the file system look like?



/a/hello.txt to be another name for essay.txt

Links

Hard: $ ln x y creates a synonym ("y") for ("x")

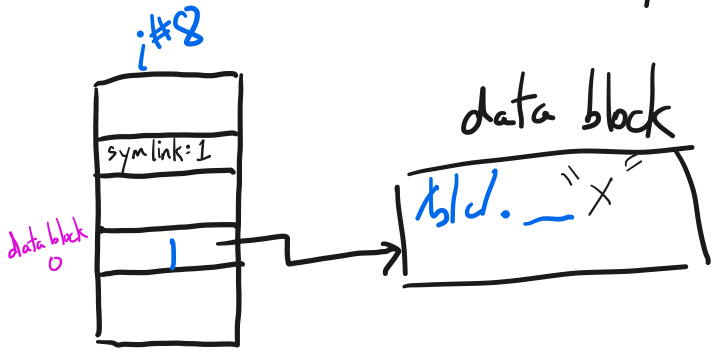
Question: what is the result of:

$ ln /b/c/essay.txt /a/hello.txt

?

Soft: $ ln -s x y (/b/c/essay.txt /a/sym.txt)

creates new inode. new file named "y". Its contents are "x".



---

3. Performance

case study: FFS (1984)

problems w/ the original:

- blocks too small (512B)
- inode array at the beginning of the disk
- free blocks stored in linked list on disk
- poor clustering of related objects

consecutive file blocks
inodes relative to pointed-to disk blocks
inodes for a given directory

result:

ls -l
grep <patt> *.c
} ⇒ slow

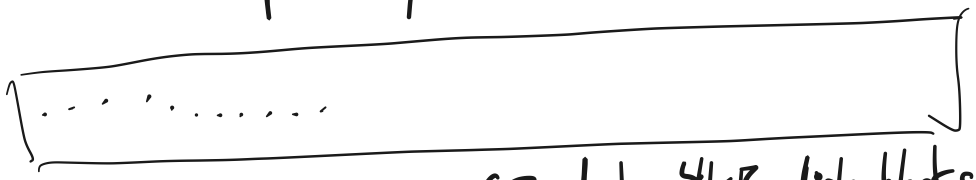
Improvements?

- Make data blocks/inodes close to each other

- Cluster files in the same directory

- make data blocks bigger (4KB, 8KB, 16 KB)

- free blocks: store separately (bitmap)

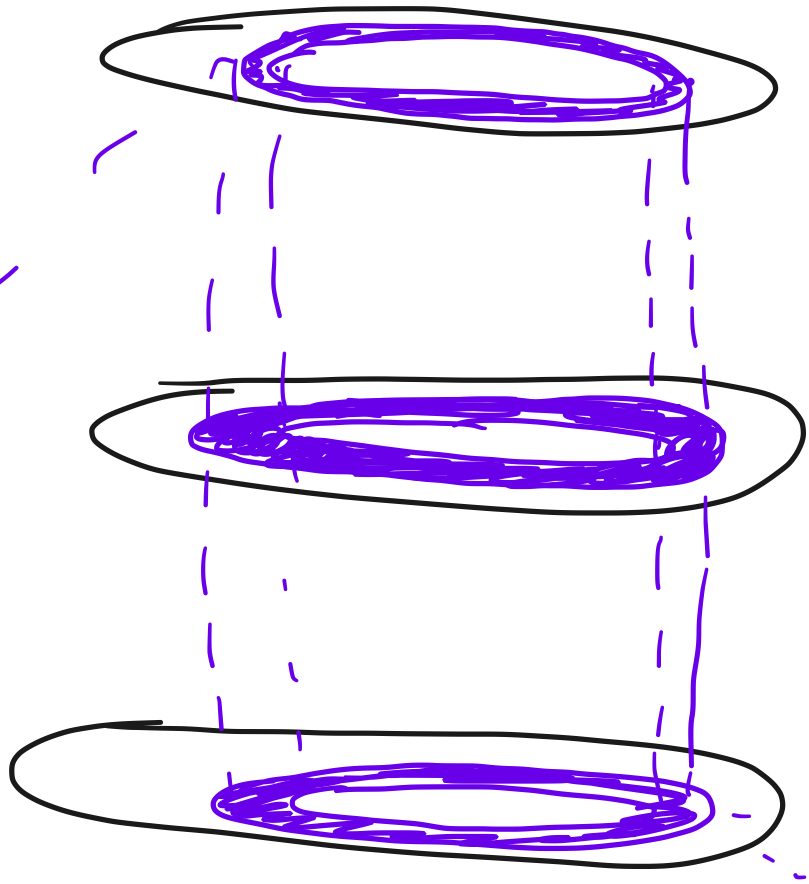


Assume 512 GB disk, 4KB disk blocks
How much space does the bitmap take up?

$\frac{512GB}{4KB} = \frac{2^{39}}{2^{12}} = 2^{27} = 128M$---

$2^{27}/2^3 = 2^{24}B = 16MB$

- reserve space (lie to the user about free space)
- symbolic links
- atomic rename (`$ mv abc.txt def.txt`)
- cylinder groups (for clustering)



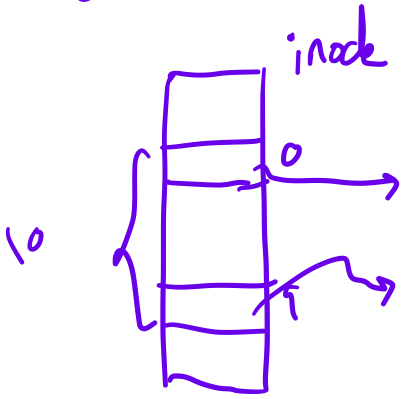
[superblock | bookkeeping | inodes | bitmap | data blocks (512 bytes each)]

attempt to: put inodes and their data blocks in the same cyl. group

attempt to: put inodes of files in the same dir in the same cyl. group.

new directory: place in cyl. group w/ higher than avg. # of free inodes

as a file grows, after it crosses 40KB, spill to next cyl. group, and do likewise for every 1MB thereafter.



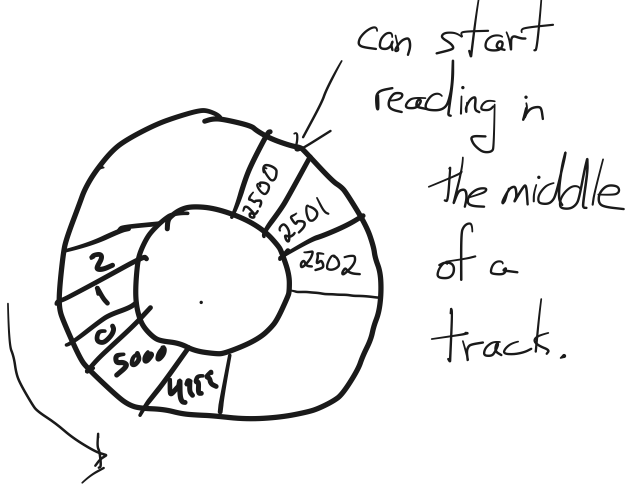
total perf:

20-40% of disk bandwidth for large files

10-20x improvement on the predecessor

Other things they did:

- buffer cache
- read entire track
- write in big chunks
- read ahead in big chunks (64KB)



Directories

Approach 1: Single dir for whole system

map : <name, inumber>

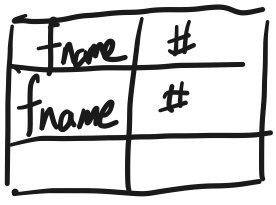
mike-todo.txt, 64
chris-todo.txt, 121
⋮

Approach 2: Single dir for each user

mkdir

/mike

/jo



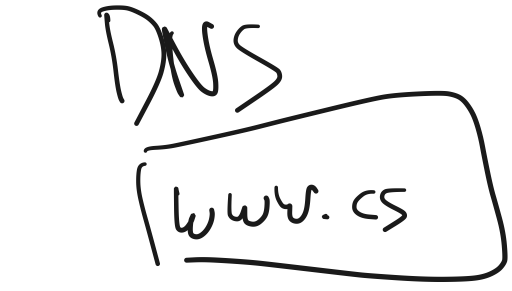
Approach 3: Hierarchical name space

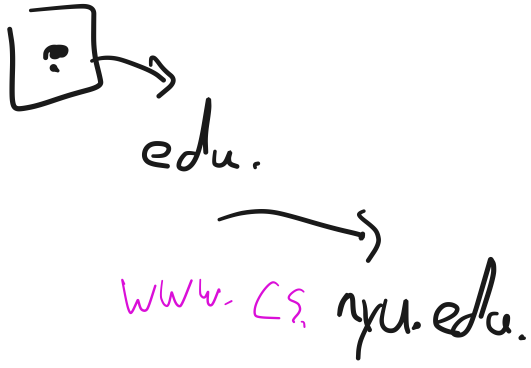
Directory maps from names to files OR other directories

File system then forms a tree or graph.

Large name spaces tend to be hierarchical
Ex: IP addresses, domain names
Cloud computing infrastructure has changed that! Google docs, for example.

www.cs.nyu.edu.





/cs201/___
/usr/mike/cs202/lab3/4.cpp
/usr/jo/___