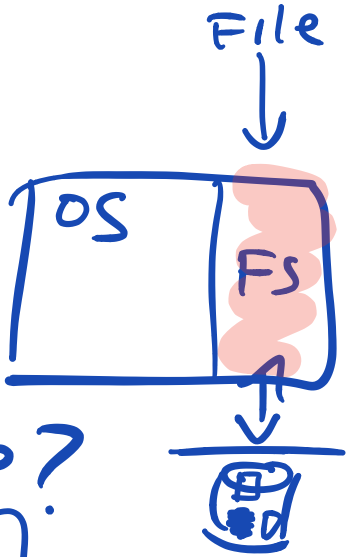
Today (4/11)

=> Files / Directories ← lecture

=> P4a : MapReduce

Files / Directories:

Abstraction:



File: what is a file?

unix: [array of bytes] ← unstructured
read/write
create/delete

has:
low-level name => number

other attributes:
=> size, permissions, when accessed/modified
=> location(s) on disk

Directory : special type of file

specific:

array of records

each record: "name" (human readable) => low-level name (inode number)

files or dirs

Arrange Files + Dirs:

in Tree, w/ root

root has special name: /

(more later)

Access:

file descriptor (integer) ⇐ open("file1", O_RDONLY);

read(fd, ...

⋮

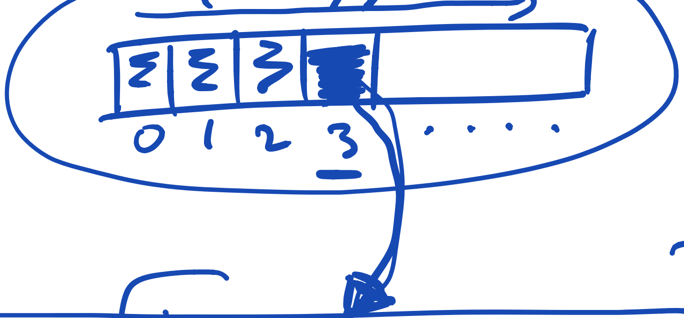
read(fd, ...

close

⇓

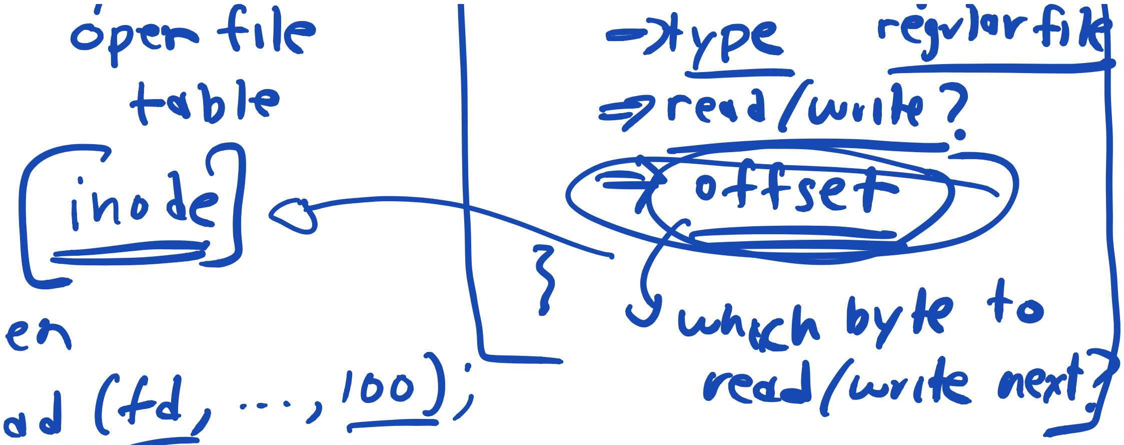
[per process: descriptor table]

(array)

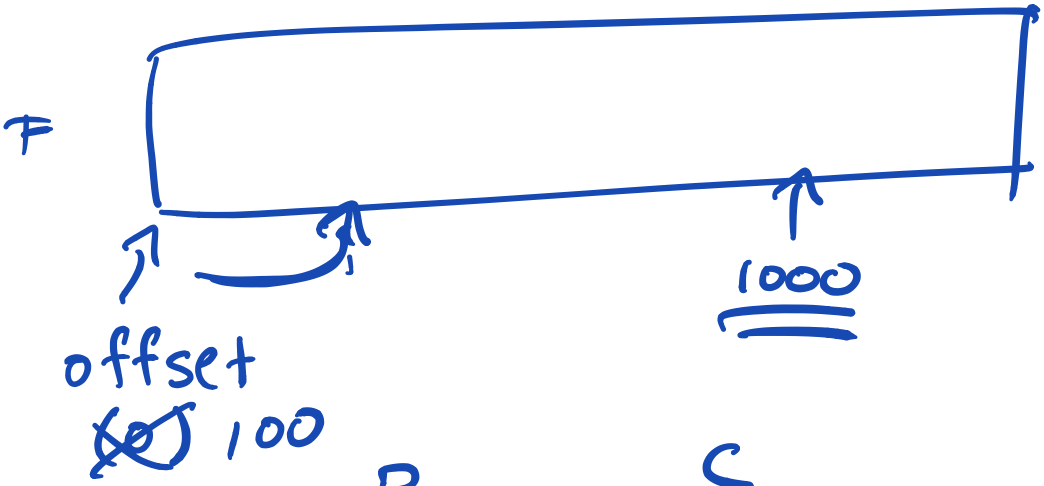


system-wide

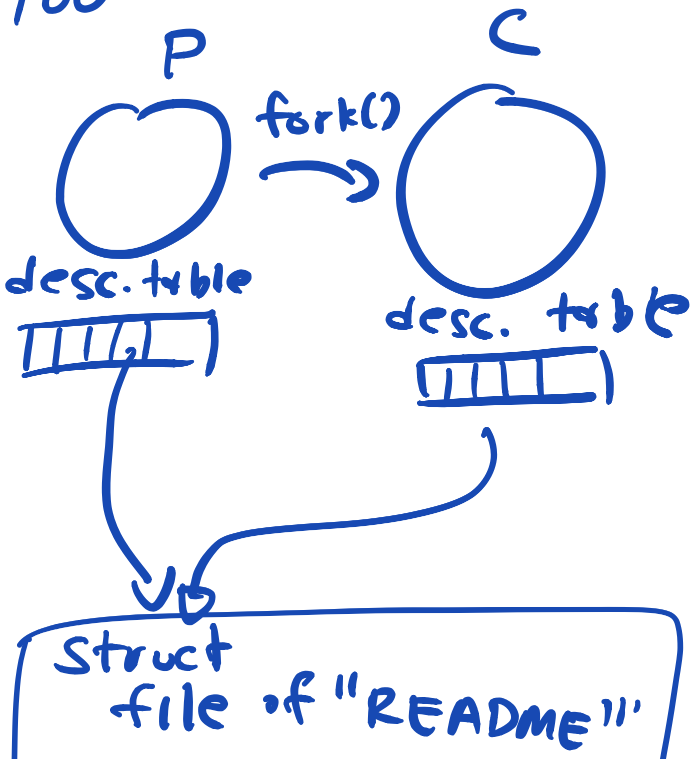
struct file {



```
fd ← open
→ read (fd, ..., 100);
→ lseek (        )
```



fork:



offset

remove file:

rm command line program

unlink() (**sys call**)

why?

rename

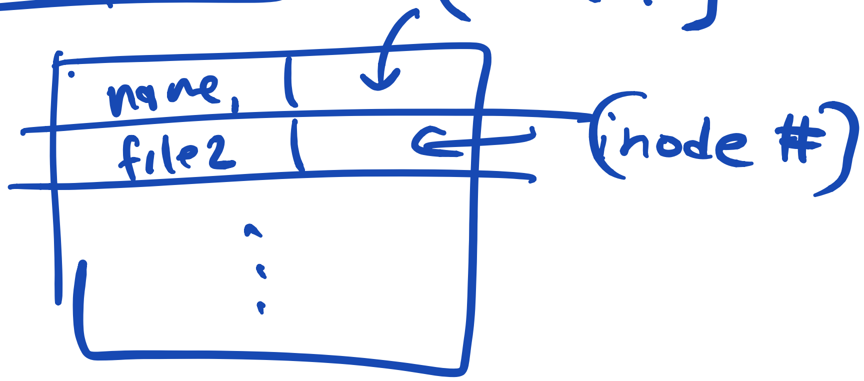
mv cmd line

rename()

"~~name~~"
name2 → low-level name (number)

Directory

(inode #)



Pathname:

absolute: starts at root of dir tree

sep.

/dir1/dir2/foo.c

root dir

dir inside root

dir2 dir inside roo71

foo.c : file inside dir2

Open ("/dir1/dir2/foo.c", ...)

opens root dir
  looks for dir1 => low-level name
opens dir1
  looks for dir2
etc.

relative: don't start at root

"file1"   [current working dir]

"../foo"

.. ↳ parent dir

special: dir entries

. current dir

.. parent

Hard Links:

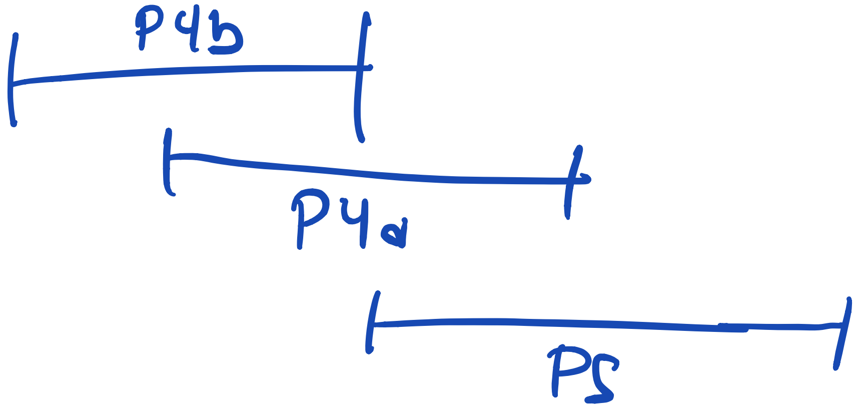
L

cmd line prog : **ln**

ln existing newname

file	7090
file2	7090

P4a :



input file:

[illegible]

os class is fun → class.

```
map (file)
    foreach w in file:
        emit(w, "1");
```

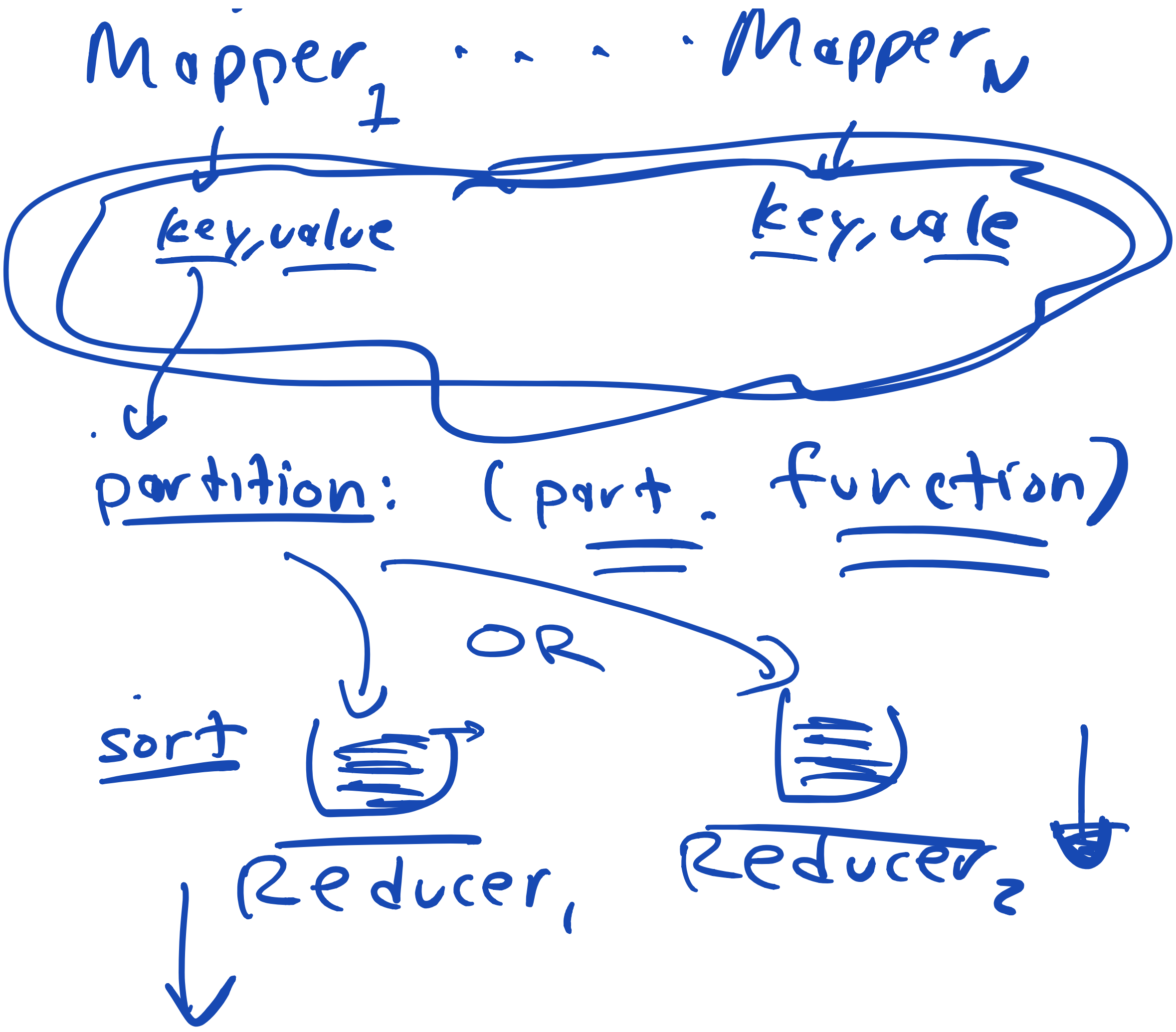
os, 1
class, 1
is, 1
fun, 1
class, 1

out of map()

```
reduce ((os,1)
    print "os", 1

reduce class, 1, 1
    print class, 2
```

{ {


Mapper$_1$ . . . . Mapper$_N$
key, value
key, vale
partition: (part. function)
OR
sort
Reducer$_1$
Reducer$_2$