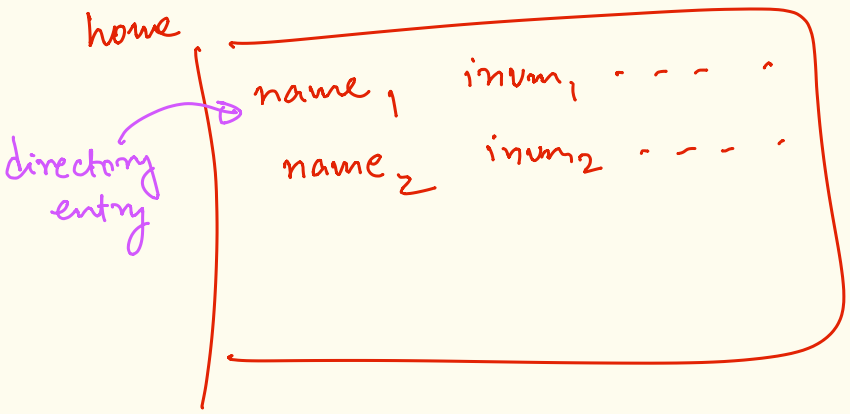
CS219 Lecture 34

* Lab Quiz 4
  9th April
  Thu.

* CS236 Help Sessions
  Today: 5 pm (Tue)
  tomorrow: 2.30 pm (Wed)
  KR 125

* FS design.

(i) directories are files



(ii) FS uses a set of in-memory caches/objects
for efficiency

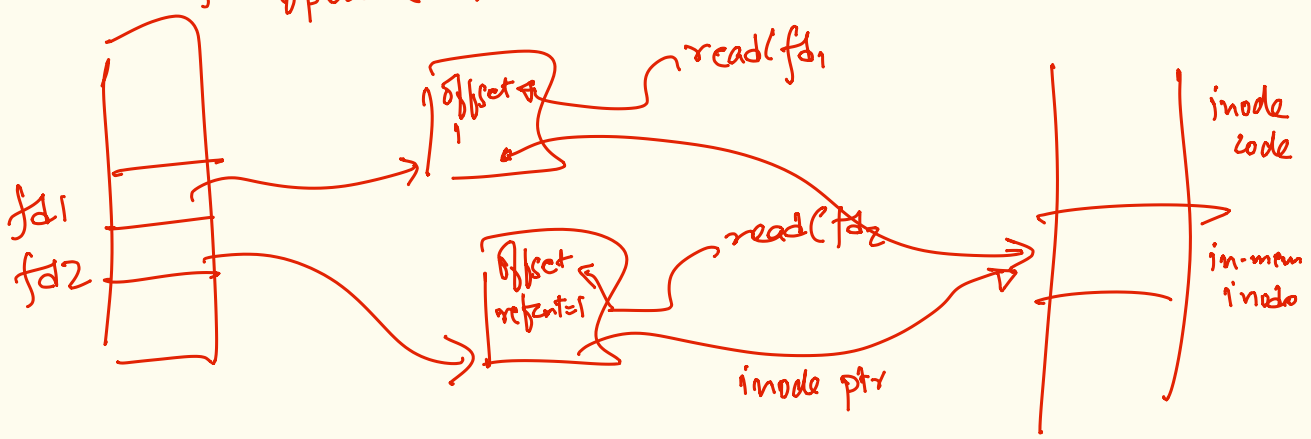
dentry cache — dir filename — inode/inum

inode cache — inum — inode contents (disk)

page/block cache — (fd, offset) — block contents
inode

struct file — offset, refcnt, inode ptr.

fd table — index, ptr. to file



H.W

DIRENTRY size = 64 bytes

size of disk inode = 64 bytes

BSIZE = 1024 bytes.

Baddr = 4 bytes

inode

DIRECT = 2

INDIRECT = 2

```
fd = open("/home/foo.txt");
lseek(400);
n = read(fd, buf, 1);
lseek(4000);
n = read(fd, buf, 1);
n = read(fd, buf, 2);
```

determine # of disk accesses.

(#) reads are great for caching but what about writes?

FS

write-back

- update in-memory state & return from FS
- faster!
- risk of lost updates.
- [Kflushd] ~ Kernel daemon which periodically flushes updated FS objects

write-through

- writes return only on update to disk (& write is through cache till disk)
- slower
- potentially more consistent!

#1 write system call

write(fd, buf, N); → single write FS syscall

- potentially reserve/allocate new datablocks
- update inode on disk
- update data bitmap – (set the bit assoc. w/ the chosen data block)
- content on datablock
  copy

} 3 diff. writes on the disk.

#2 problem for the FS.

- if all 3 writes to disk not done, FS is an inconsistent state.

		inode update	data bitmap update	datablock update
FS consistent		✓	✓	✓
FS consistent		x	x	x
data loss	(IC)	x	x	✓
	IC	✓	x	x
		x	✓	x
		✓	✓	x
		✓	x	✓
		x	✓	✓