 # OS DAY-10

- Directory structure:
- from user point of view contents of the directory file are name of its files and sub dir's, whereas from filesystem point of view contents of directory files are directory entries of its files & sub dir's.
- **"directory entry" = inode number + filename**

- fileio program: (CRUD operations):
1. write record into a file:
step1 - open a file/create a new file: "rb+"/read as well as write
fopen() -> open()

- fopen() is a library function which internally makes call to open() system call api.

open() open a file or create a new file

0644 - access perms can be assigned for a file while creation in octal format

- first digit : leading 0 indicates it is an octal constant
- second digit : indicates access perms for owner/ user.
- third digit : indicates access perms for group members
- fourth digit : indicates access perms for others

4 - read
2 - write
1 - execute

0544 ->
0 : octal constant
5 : user can read as well execute
4 : grp members can read only
4 : others can read only

0744:
0 : octal constant
7 : user/owner - r+w+x
4 : grp members can only read
4 : other can only read

- "chmod" is used to change mode bits i.e. to change access perms

- **perror()** is a C library function which prints user message as well as actual system error message.

step2 - write a record into a file:
fwrite() -> write()

step3 - close a file:
fclose() -> close()

2. read records from a file:
step1 - open a file (file must exists already)
step2 - read all the records from a file one by one and display
step3 - close a file

+ fileio system calls/system call api : open(), write(), read(), close()

3. update record into a file
4. delete record from a file

+ for a text file
1. "w" - write only
2. "w+" - write as well read
3. "a" - append only
4. "a+" - append as well as read
5. "r" - read only
6. "r+" - read as well write

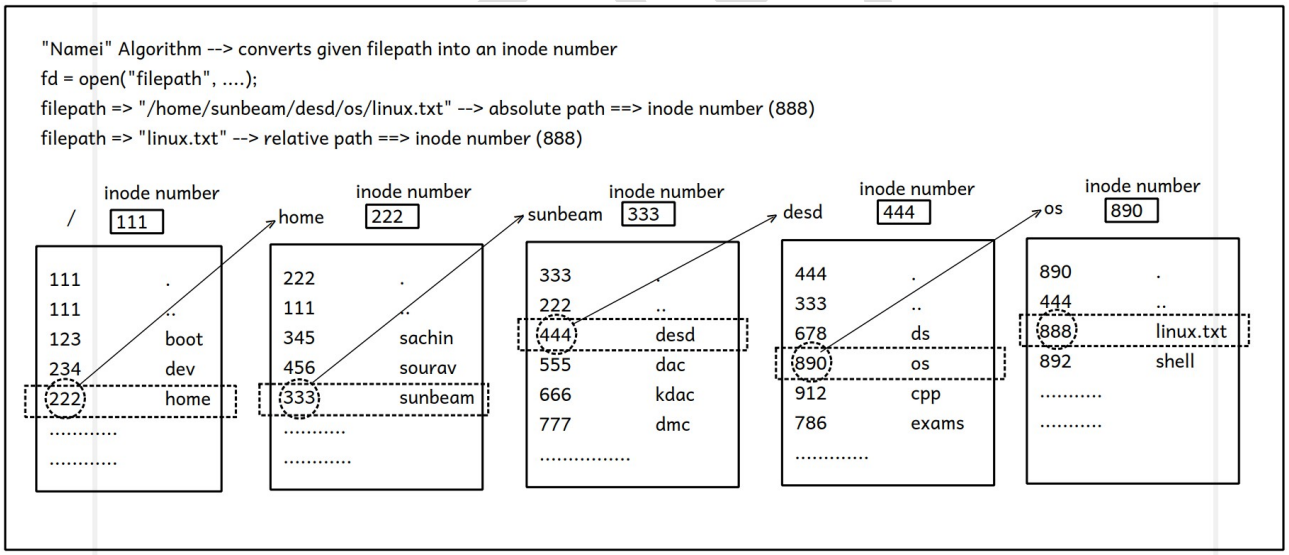
+ for a binary file:

1. "wb" - write only
2. "wb+" - write as well read
3. "ab" - append only
4. "ab+" - append as well as read
5. "rb" - read only
6. "rb+" - read as well write

 # open() system call internals:
int fd = open("filepath", O_RDONLY, 0644);

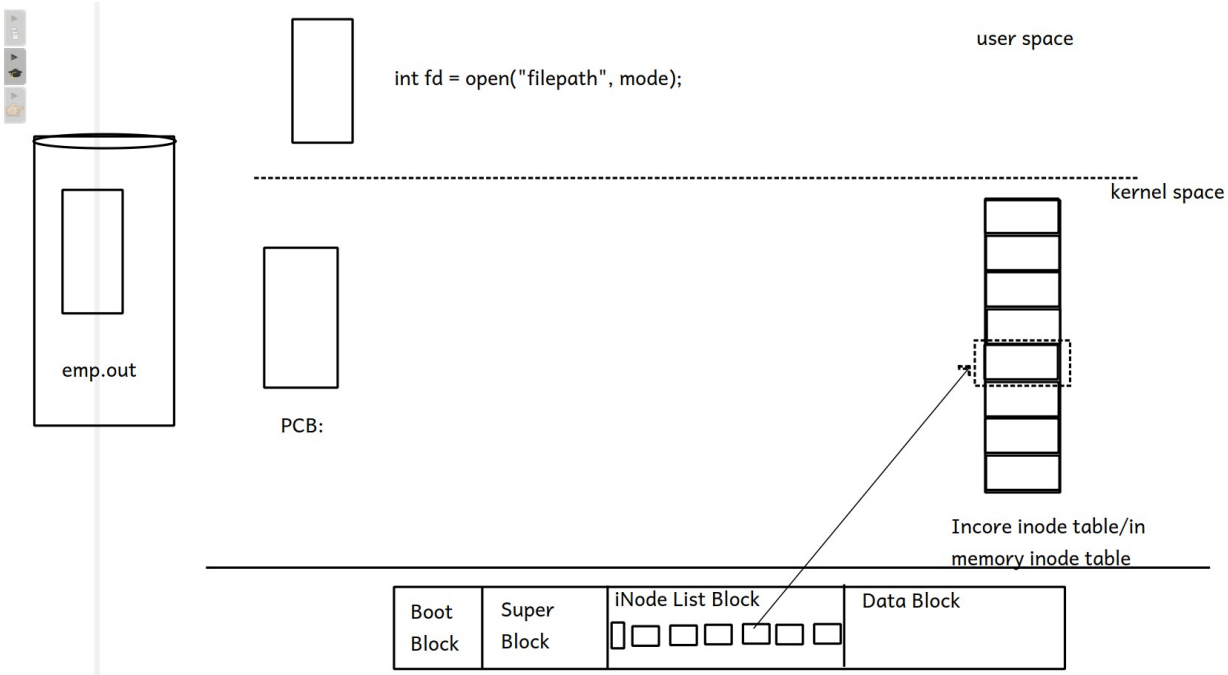
step1: it converts given filepath into its inode number.
- filepath may be absolute path or it may relative path
- algo which converts given filepath into its inode number is reffered as **"namei" algorithm**, it is called as **pathname translation.**

For Example:



step2: once open() syscall got the inode number of that file, its corresponding inode gets loaded into the main memory by searching it from inode list block, and inode entry of that file got added into one global table (kernel data structure) reffered as **"in-memory inode table".**

- **In-Memory/In-Core Inode Table:** contains list of inodes of all recently opened files.


user space
int fd = open("filepath", mode);
kernel space
emp.out
PCB:
Incore inode table/in
memory inode table
Boot
Block
Super
Block
iNode List Block
Data Block