CS302
Operating System
Lab 10

File System

Xinxun Zeng, Shiqi Zhang,Chuang Yang

Overview

- Concept: File, Directory, File System
- File System Layout
  - contiguous allocation
  - linked allocation
  - indexed allocation

File System

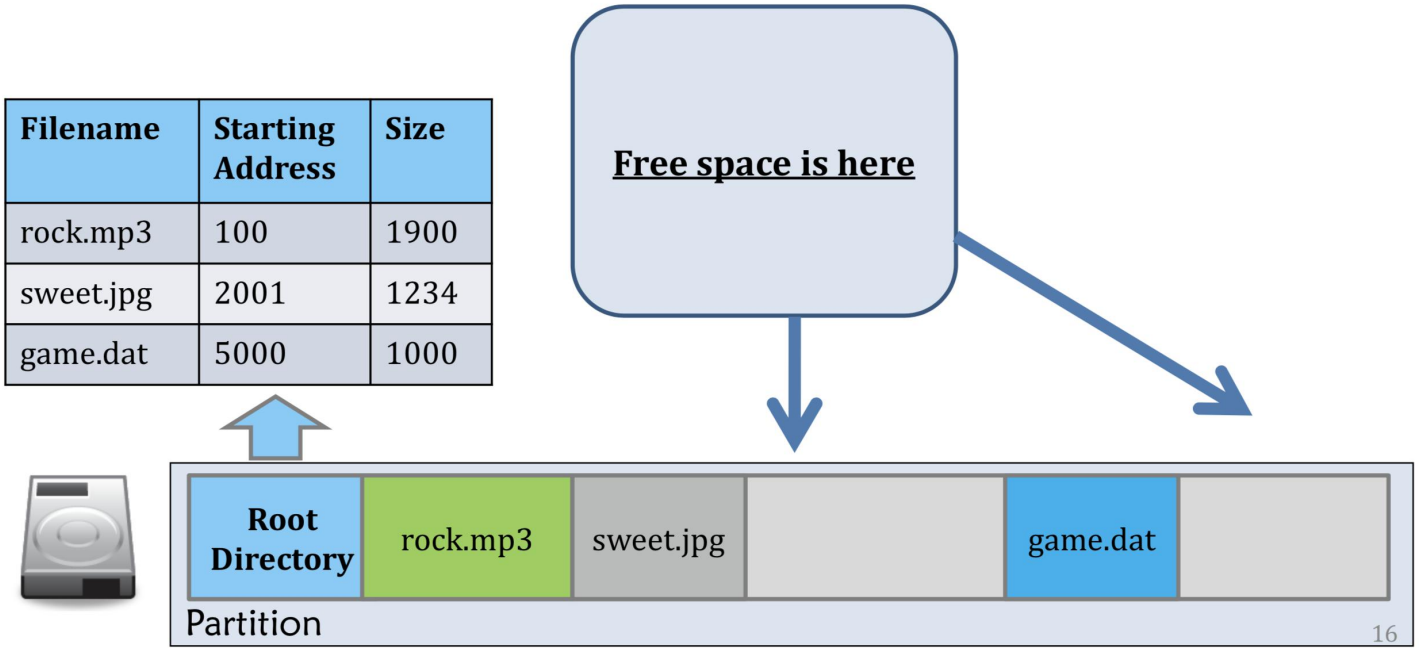
- File systems
  - Layer of OS that transforms block interface of disks (or other block devices) into Files, Directories, etc.
  - Keep track of files in secondary storage:
  - Organize files logically (directories)

File System Layout

- File systems define a block size (e.g., 4KB)
  - Disk space is allocated in granularity of blocks
- A "Master Block" determines location of root directory
- Remaining disk blocks used to store files (and dirs)
  - There are many ways to do this

Contiguous Allocation

- Contiguous allocation
  - Like main memory (base&bound)
  - Fast, simplifies directory access
  - Easy file deletion



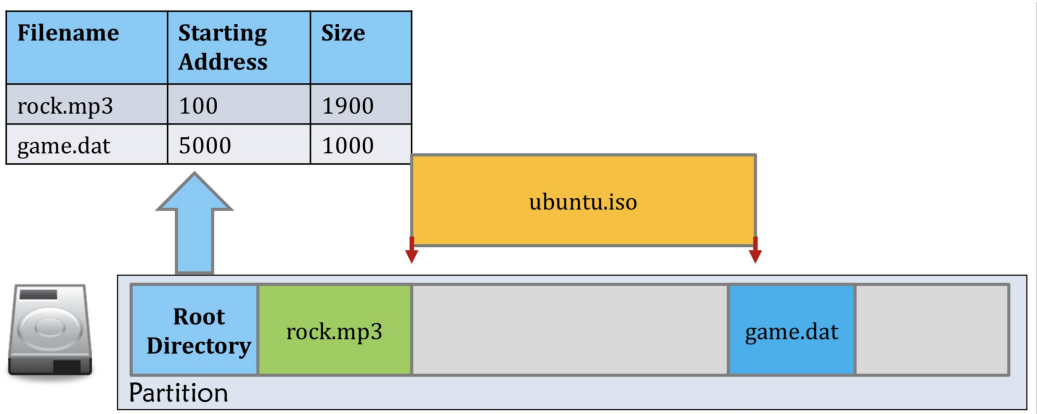
Question

- Which contiguous free block to be allocated ?

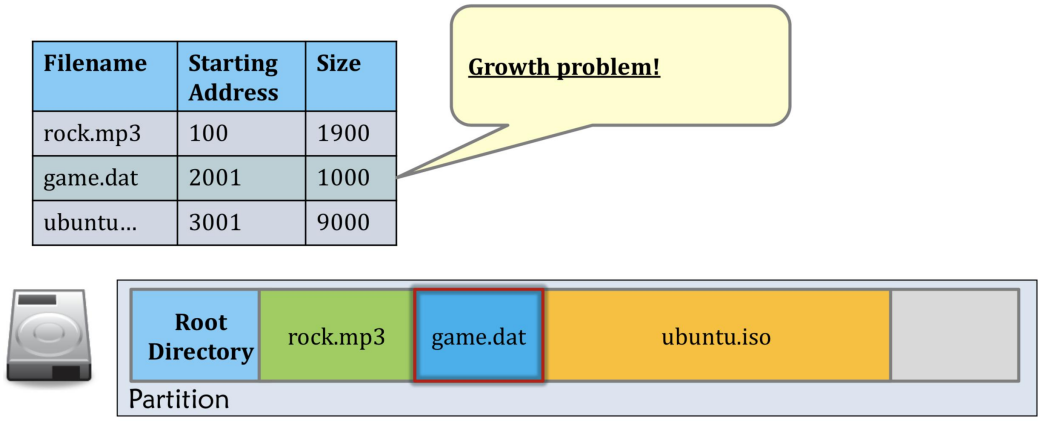
  - First Fit
  - Best Fit
  - Worst Fit

Question

- What is the problem of Contiguous allocation ?
  - External fragmentation



  - Growth Problem

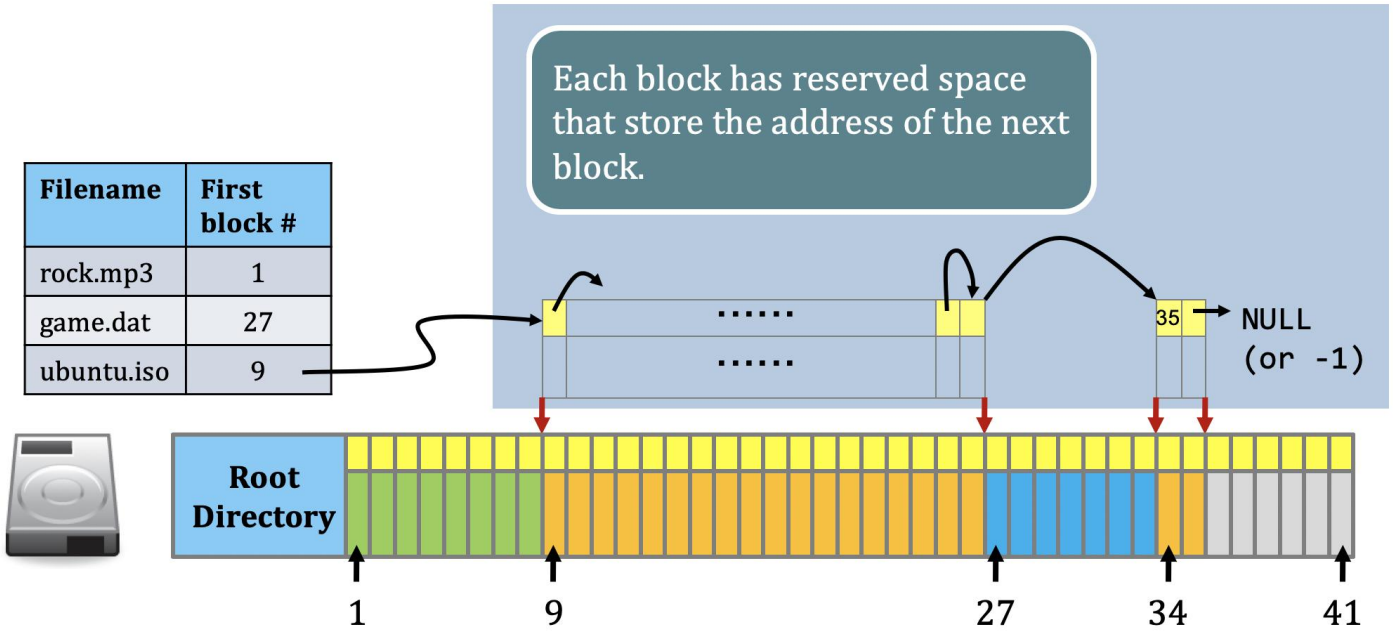


Question

- What is the advantage of Contiguous allocation ?
  - Fast Random access
  - Fast Sequential access

Linked Allocation

- Linked Allocation
  - Each block points to the next, directory entry points to the first
  - Good for sequential access,
  - Solve external fragmentation and growth problem

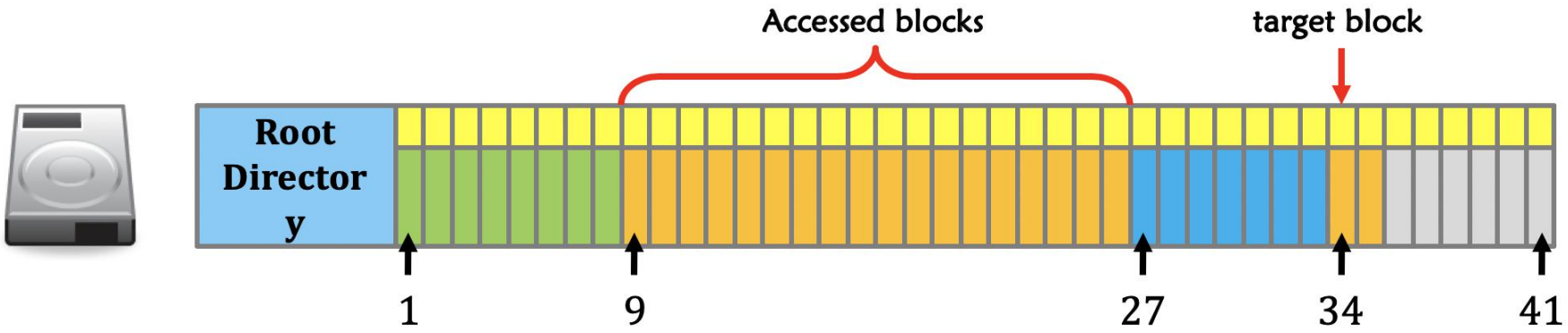


Question

- What is the problem of Linked allocation ?
  - Bad for random access
  - Reliability
  - Space Wasting

Θ: Ηοω το ρεδυχε σπαχε ωαστινγ ?

Φρομ Βλοχκ το Χλυστερ, α τραδε–οφφ



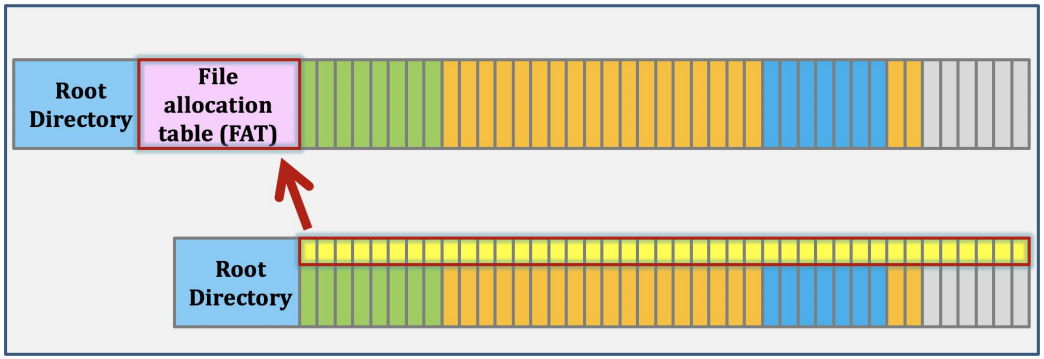
FAT, a variation of Linked Allocation

- **FAT Series: Centralize all the block links as File Allocation Table**

```
struct node_type {
    float data;
     int next;
} node[100]
```

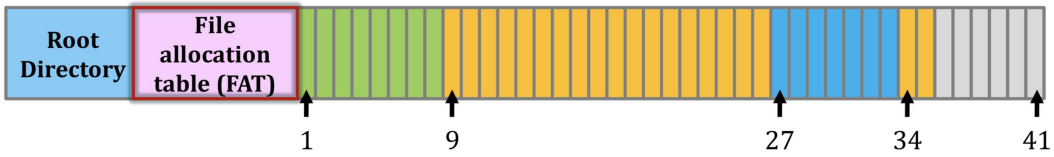
⇩

```
float node_data[100];
int node_next[100];
```



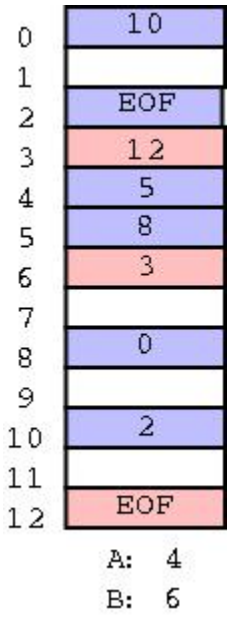
Filename	First block #	Size
rock.mp3	1	1900
game.dat	27	1000
ubuntu.iso	9	9000

Block #	1	...	7	8	9	...	26	27	...	32	33	34	35	...	41
Next Block #	2	...	8	-1	10	...	34	28	...	33	-1	35	-1	...	0



FAT, a variation of Linked Allocation

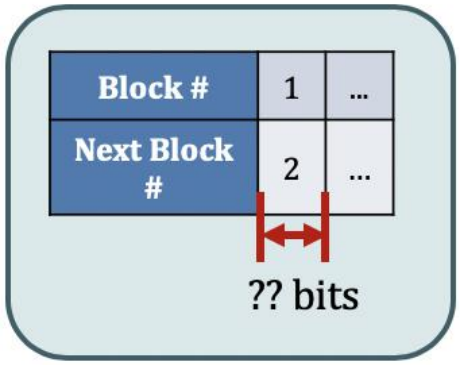
- **FAT Series: Centralize all the block links as File Allocation Table**
  - Block Data, is stored on disk;
  - The next pointers, which are small (each is an integer) are stored in memory in a File Allocation Table or FAT.
  - When the system is shut down the FAT is copied to disk and when the system is booted, the FAT is copied to memory.



Question

- What do 12, 16 and 32 mean in FAT12, FAT16 and FAT32

  - Block(cluster) address



	FAT12	FAT16	FAT32
Cluster address length	12 bits	16 bits	28 bits
Number of clusters	2^{12} (4,096)	2^{16} (65,536)	2^{28}

Question

- Although fasten the random access **What is the problem of FAT?**

Many file seeks unless entire FAT is in memory.

- 1TB (2^40 bytes) disk,
- 4KB (2^12) block size,
- FAT has 256 million (2^28) entries.
- If 4 bytes(32bit) used per entry ⇒ 1GB (2^30) of main memory required for FAT,
- which is a sizeable overhead

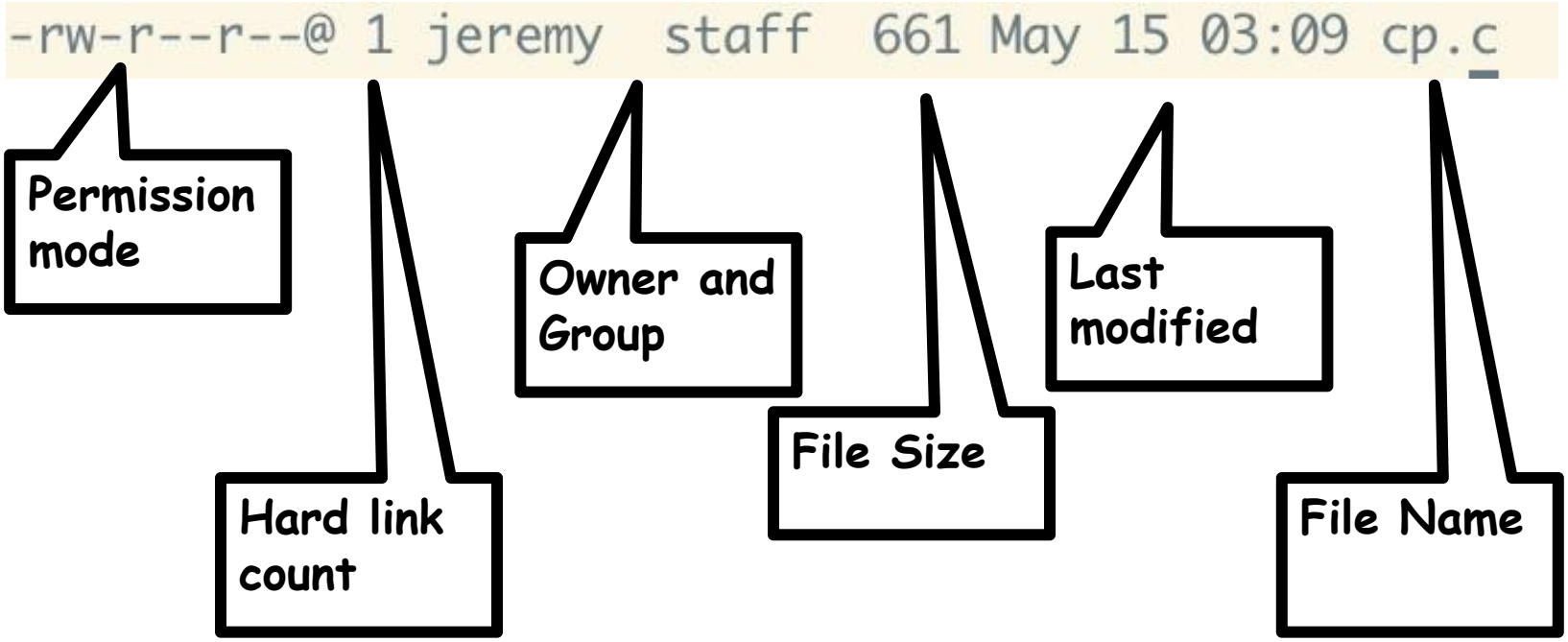
Question

- If block size is 4KB in a FAT32 file system, what is the total size of the file system?

$$4 * 2^{10} * 2^{28} = 2^{40}B = 1TB$$

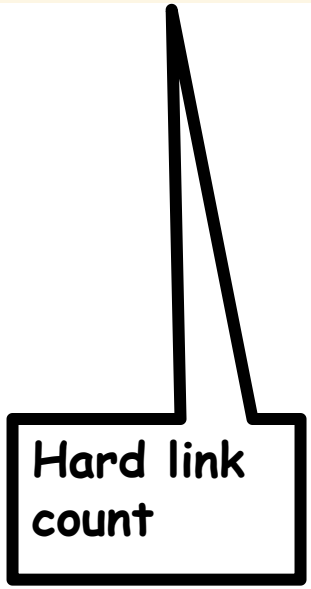
File Permission (Linux)

- Use **ls –l filename** to view the permission of a file



What is hard link?

```
-rw-r--r--@ 1 jeremy  staff  661 May 15 03:09 cp.c
```



Background: INODE

- INODE
  - Contains all info about a file(except filename and content)
  - Metadata of a file

* 文件的字节数

* 文件拥有者的User ID

* 文件的Group ID

* 文件的读、写、执行权限

* 文件的时间戳，共有三个：ctime指inode上一次变动的时间，mtime指文件内容上一次变动的时间，atime指文件上一次打开的时间。

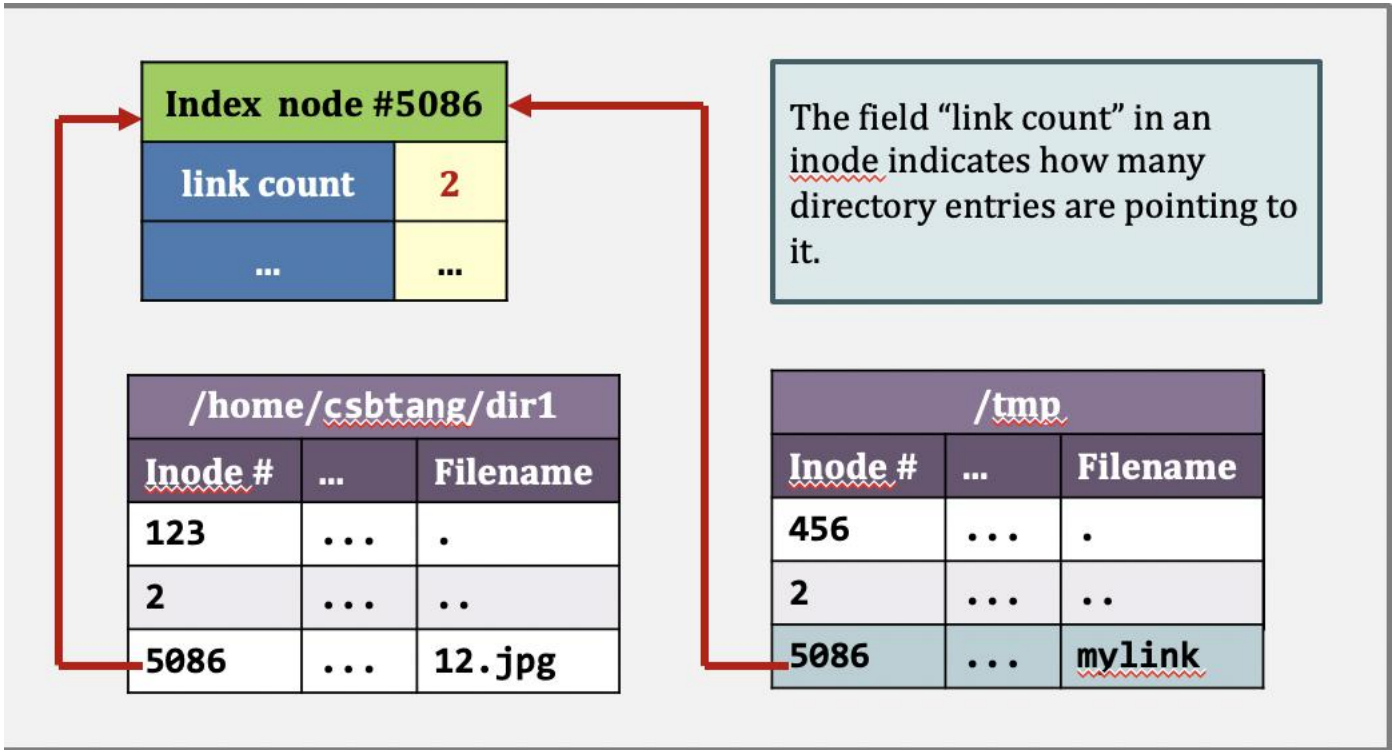
* 链接数，即有多少文件名指向这个inode

* 文件数据block的位置

http://www.ruanyifeng.com/blog/2011/12/inode.html

Hard Link

- Each hard linked file is assigned the **same Inode** value as the original, therefore they reference the same physical file location.



Hard Link

- Link to same inode
- Removing any link, just reduces the link count, but doesn't affect other links.
- If original file is removed then the link will still show the content of the file.
- Command to create a hard link is:

```
ln  [original filename] [link name]
```

Soft Link

- Each soft linked file contains a **separate** Inode value that **points to the original file.**

```
# ln -s /home/csbtang/dir1/12.jpg /tmp/mylink
```

create another inode...

/home/csbtang/dir1		
Inode #	...	Filename
123	...	.
2	...	..
5086	...	12.jpg

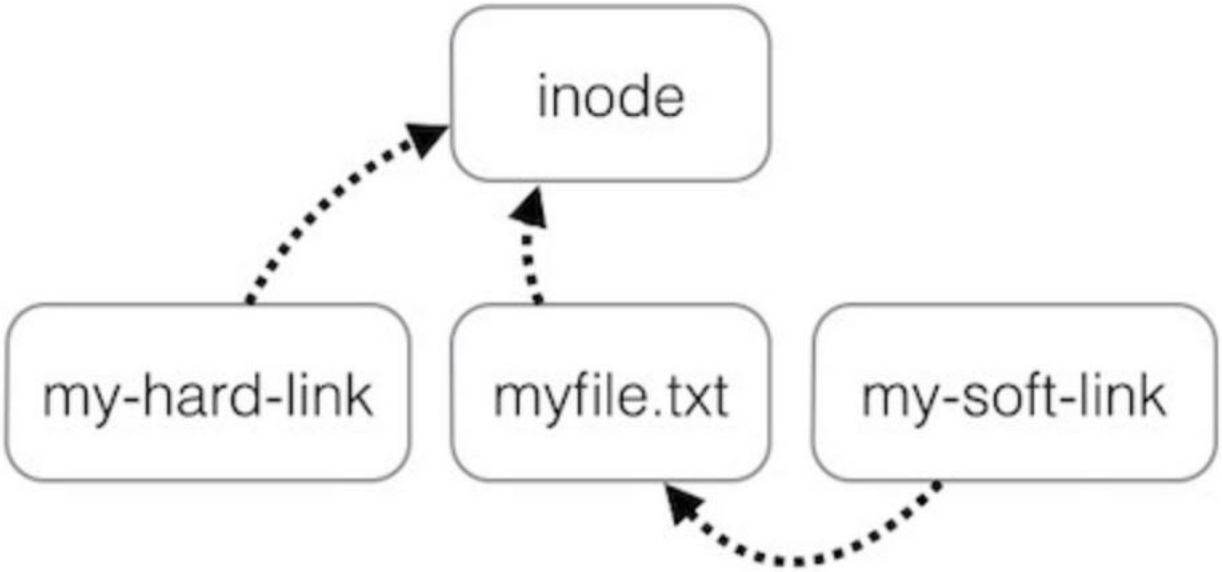
/tmp		
Inode #	...	Filename
456	...	.
2	...	..
6120	...	mylink

Soft Link

- Like file shortcut in Window OS
- Link to same original file
- If the original file is deleted or moved, the soft linked file will not work correctly (called hanging link).
- Command to create a Soft link is:

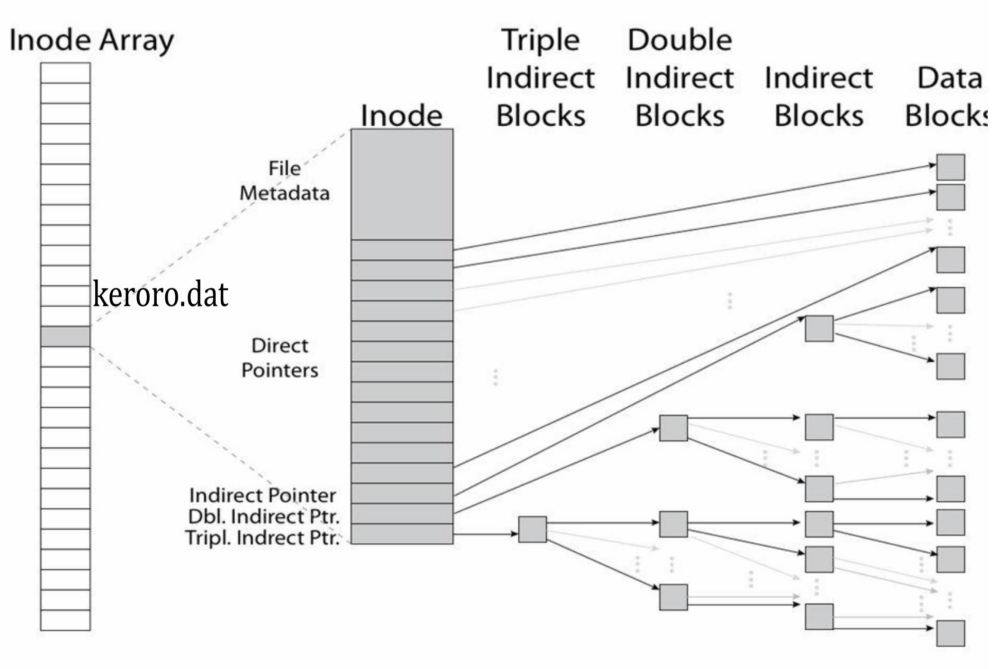
```
ln  -s [original filename] [link name]
```

Hard Link and Soft Link



Indexed Allocation

- Indexed Allocation(Unix Inode)
  - iNode Table is an array of iNodes
  - Handles random access better, still good for sequential



iNode Structure (128 bytes long)	
Bytes	Value
0-1	File type and permission
2-3	User ID
4-7	Lower 32 bits of file sizes in bytes
8-23	Time information
24-25	Group ID
26-27	Link count (will discuss later)
…	…
40-87	12 direct data block pointers
88-91	Single indirect block pointer
92-95	Double indirect block pointer
96-99	Triple Indirect block pointer
…	…
108-111	Upper 32 bits of file sizes in bytes

Question

- If block size is 4KB, address length is 4B, what is the maximum size for a file in Unix inode

$$12 * 2^{12} + 1 * (2^{12}/2^2) * 2^{12} + 1 * (2^{12}/2^2)^2 * 2^{12} + 1 * (2^{12}/2^2)^3 * 2^{12}$$

Number of direct blocks	12
Number of indirect blocks	1
Number of double indirect blocks	1
Number of triple indirect blocks	1