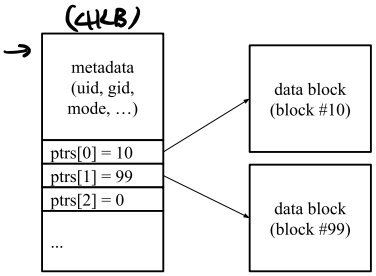
```
CS5600, Handout Week12.b

// 1. Read/write disk in Lab4 (CS5600 file system)
// borrowed from Lab4, fs5600.c

    /* disk access.
     * All access is in terms of 4KB blocks; read and
     * write functions return 0 (success) or -EIO.
     *
     * read/write "nblks" blocks of data
     *   starting from block id "lba"
     *   to/from memory "buf".
     *     (see implementations in misc.c)
     */
extern int block_read(void *buf, int lba, int nblks);
extern int block_write(void *buf, int lba, int nblks);

    /*
     * below are two toy examples of
     * reading from and writing to a disk block
     */

    // Reading one block

    char buf[FS_BLOCK_SIZE];  // FS_BLOCK_SIZE=4096; see panel 2
    int inum = 100;           // block number to read from
    int ret = block_read(&buf, inum, 1);
    if (ret < 0) {  // error; ret should be -EIO
        return ret;
    }


    // Writing one block

    char buf[FS_BLOCK_SIZE];  // again, 4KB buffer
    ...                       // update buf

    int ret = block_write(&buf, 99, 1);  // update the 99th block
    if (ret < 0) {  // error; ret should be -EIO
        return ret;
    }
```



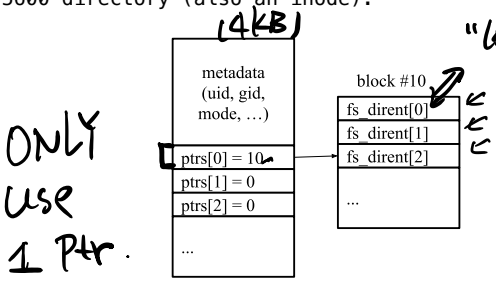
```


// 2. CS5600 file system data structures
// borrowed from fs5600.h with minor changes

    /*
     * file:        fs5600.h
     * description: Data structures for CS5600 file system.
     *
     * CS 5600, Computer Systems, Northeastern CCIS
     * Peter Desnoyers,  November 2016
     *
     * Modified by CS5600 staff in fall 2021.
     */

    #define FS_BLOCK_SIZE 4096
    #define FS_MAGIC 0x30303635

    #define INODE_NUM_PTRS (FS_BLOCK_SIZE/4 - 5)

    /* Superblock - holds file system parameters.
     */
struct fs_super {
    uint32_t magic;
    uint32_t disk_size;       /* in blocks */

    /* pad out to an entire block */
    char pad[FS_BLOCK_SIZE - 2 * sizeof(uint32_t)];
};


/* inode = 4096 bytes */
struct fs_inode {
    uint16_t uid;
    uint16_t gid;
    uint32_t mode;
    uint32_t ctime;
    uint32_t mtime;
     int32_t  size;
    uint32_t ptrs[INODE_NUM_PTRS];
};


    /* Entry in a directory
     */
struct fs_dirent {
    uint32_t valid : 1;
    uint32_t inode : 31;
    char name[28];        /* with trailing NUL */
};
```

1. fs5600 visualization

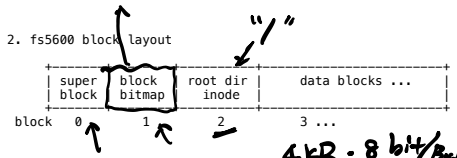
fs5600 file inode:



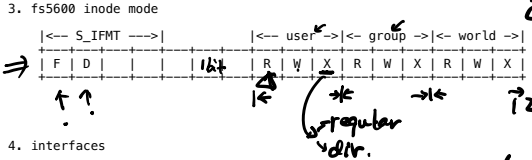
fs5600 directory (also an inode):



2. fs5600 block layout



3. fs5600 inode mode



4. interfaces

```
fs_init - constructor (i.e. put your init code here)
fs_statfs - report file system statistics
fs_getattr - get attributes of a file/directory
fs_readdir - enumerate entries in a directory
fs_read - read data from a file
fs_rename - rename a file
fs_chmod - change file permissions

fs_create - create a new (empty) file
fs_mkdir - create new (empty) directory
fs_unlink - remove a file
fs_rmdir - remove a directory
fs_write - write to a file
fs_truncate - delete the contents of a file
fs_utime - change access and modification times
```

1. Last time ←
2. Files
3. Directories
4. fs5600 (Lab4) ←
   A. fs5600 disk
   B. data structures
   C. interfaces

bottom ↓ up

- SSD
  - read/erase/program.
  - Cannot program twice w/o erasing.
  - Wear-out   FTL

- fs.
  1. persistence
  2. named bytes ⇒ files
  3. locate bytes ⇒ directories

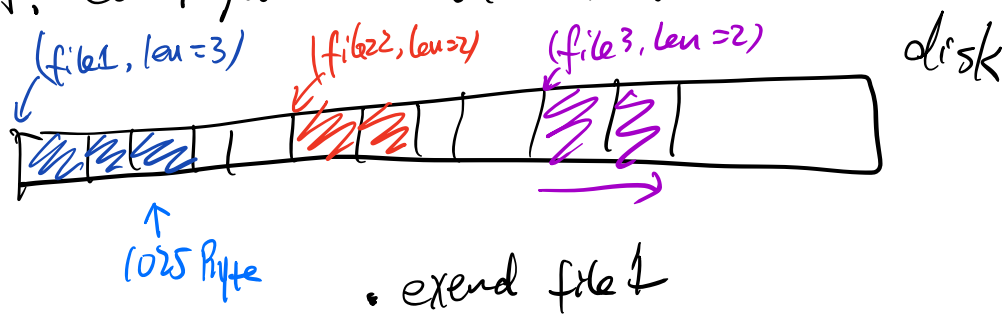
- files.

user		FS.
an array of bytes	⟶	a set of blocks.
(file, offset)	⟶ ?	disk address. (LBA, 48 bits)
(Process, VA)	⟶ PT	PA
path (string)	⟶ directories.	? = inode

/tmp/aaa ...

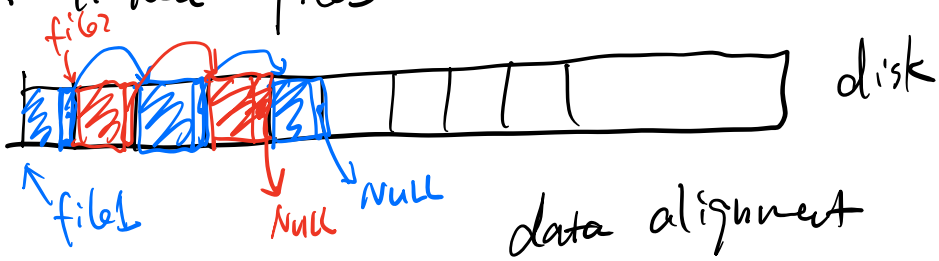
- FS. design parameters.
  - small files vs. large files.
  - access patterns:
    - sequential access.
    - random access.
    - keyed access.
  - disk utilization
- Candidate design? (files)

A. contiguous allocation



- exend file ?
- fragmentation

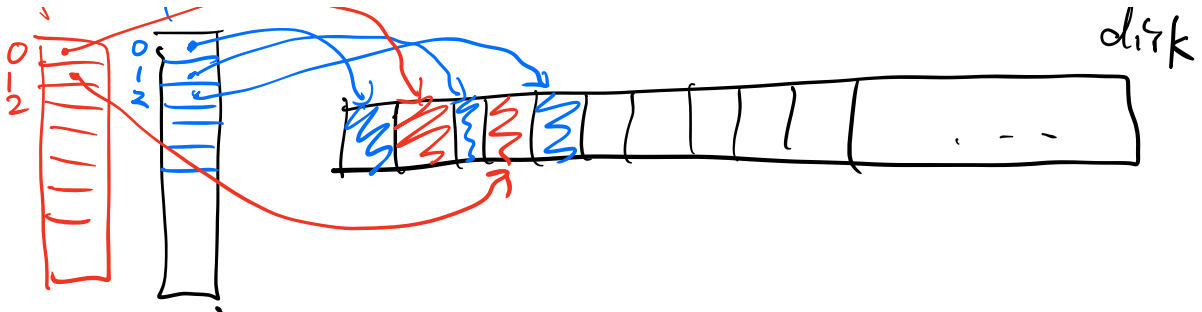
B. linked files



data alignment

C. indexed files.

file2 file1

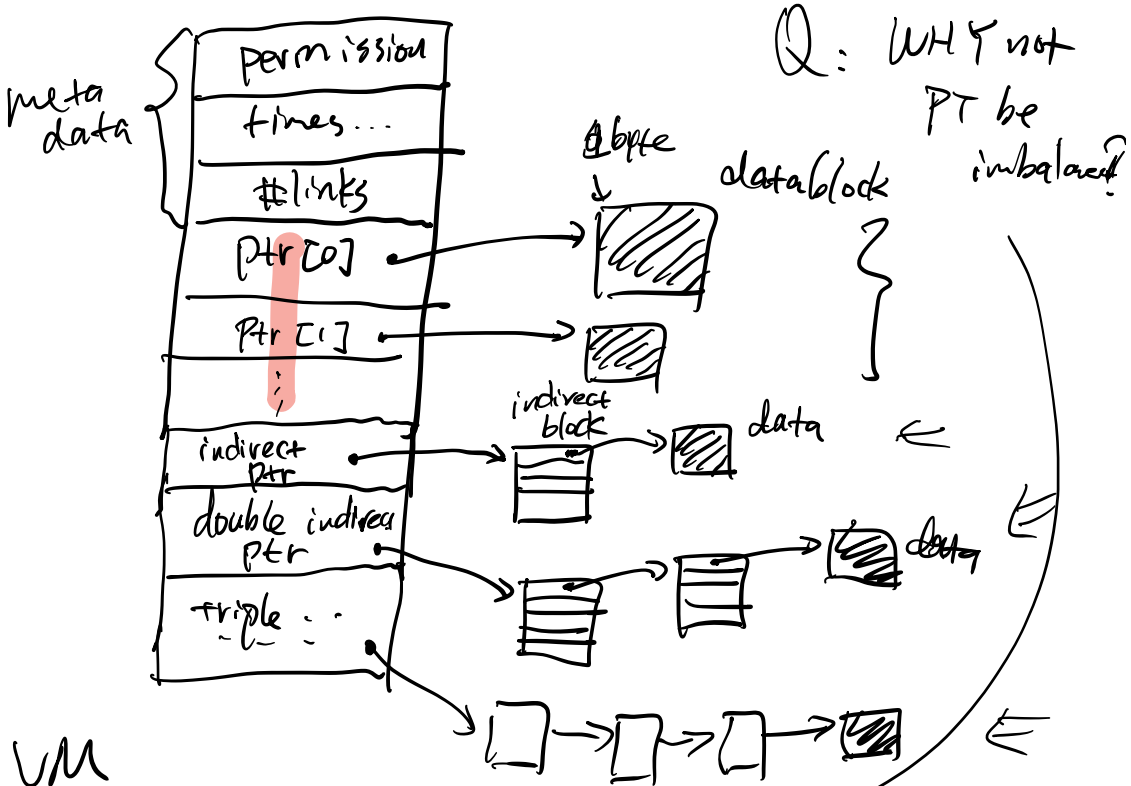


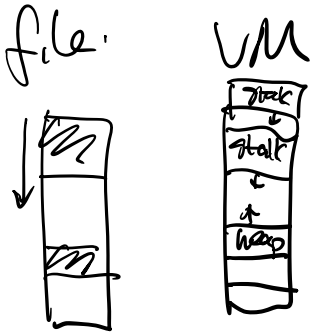
where is this array?
How large is this?

Unix inode

Q: WHY?

Q: WHY not PT be imbalanced?





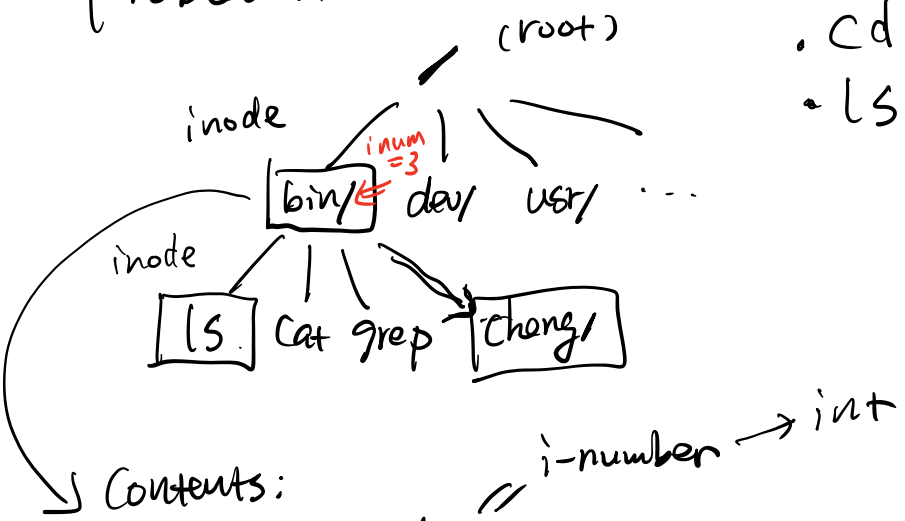
① TLB

② HW.

③ 1 level vs. 4-level doesn't make much diff

- directories.

Problem.



. cd
. ls

"." ".."

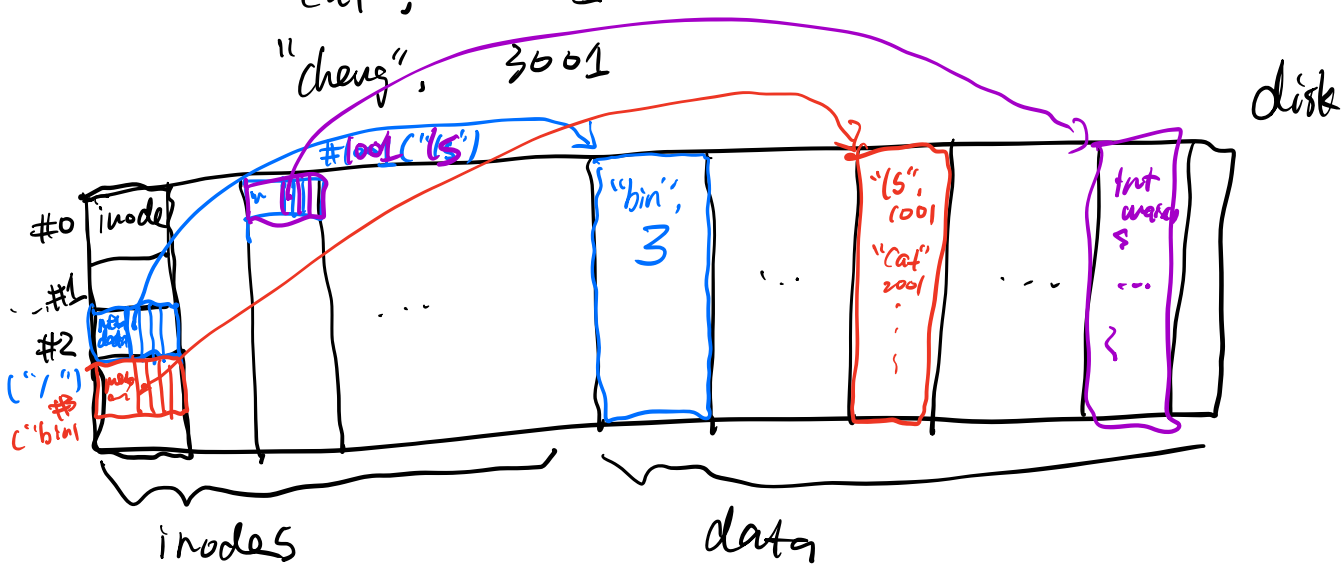
Contents: // i-number → int

<filename, inode number>

"ls", 1001

"cat", 2001

"cheng", 3001



* links.

① • hard link

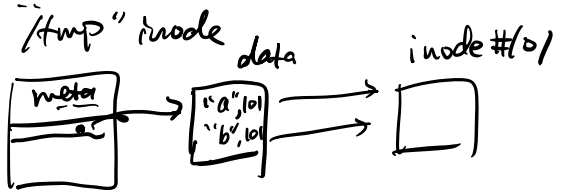
$ ln /tmp/a /tmp/b (alias)

② • soft link.

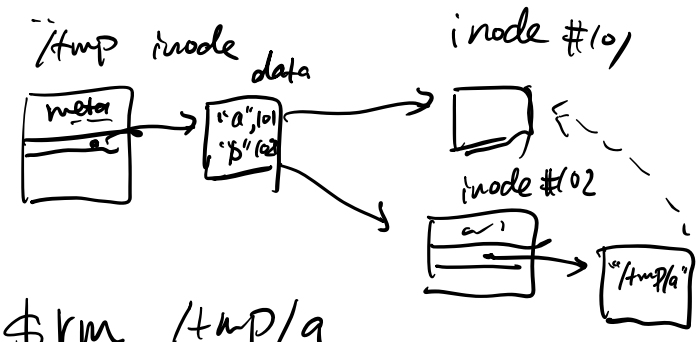
$ ln -s /tmp/a /tmp/sb

(create an inode. link to /tmp/a)

①



②



$rm /tmp/a

① $cat /tmp/b ? → okay.

② $cat /tmp/sb ? → error