```
CS5600 file system

// 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;
    }
```

mem 100 1

/a/b/c


```


// 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 */
    };
```

meta data

file size

100 (block #100, inode #100)

dir inode

data block

fs_dirent

4KB

sizeof(fs_dirent)

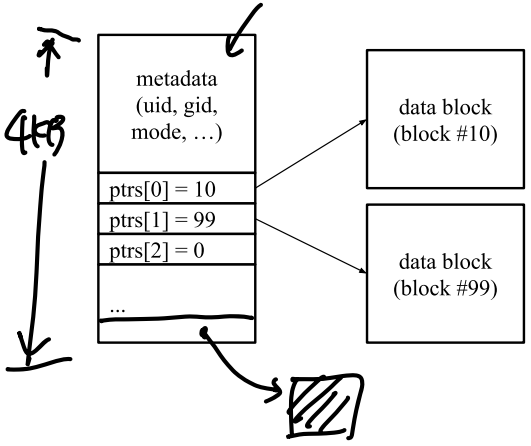
= 4B + 28B

= 32B

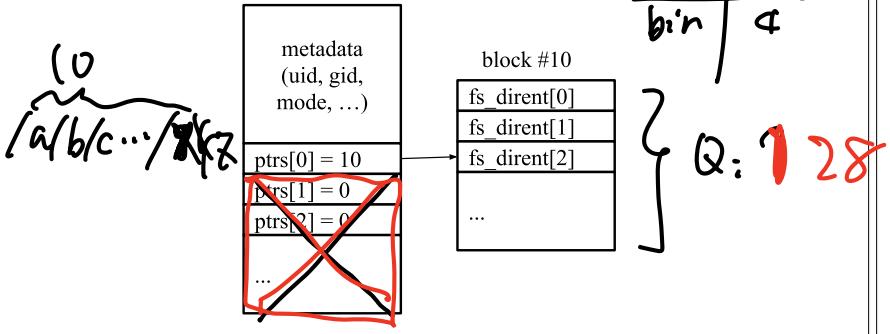
#files = 4KB / 32B = 128

1. fs5600 visualization

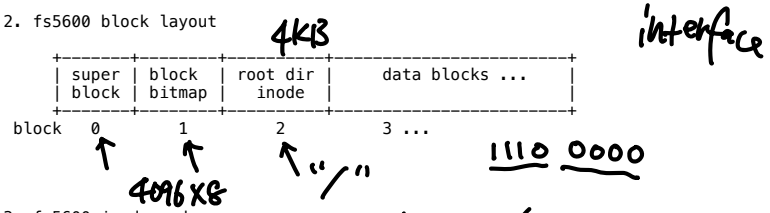
fs5600 file inode:



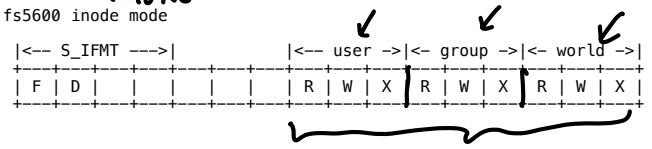
fs5600 directory (also an inode):



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
```

Week 13.b
CS 5600
04/05 2023

1. last time
2. Directories
3. fs5600 (Lab5)
   A. fs5600 disk
   B. data structures
   C. interfaces

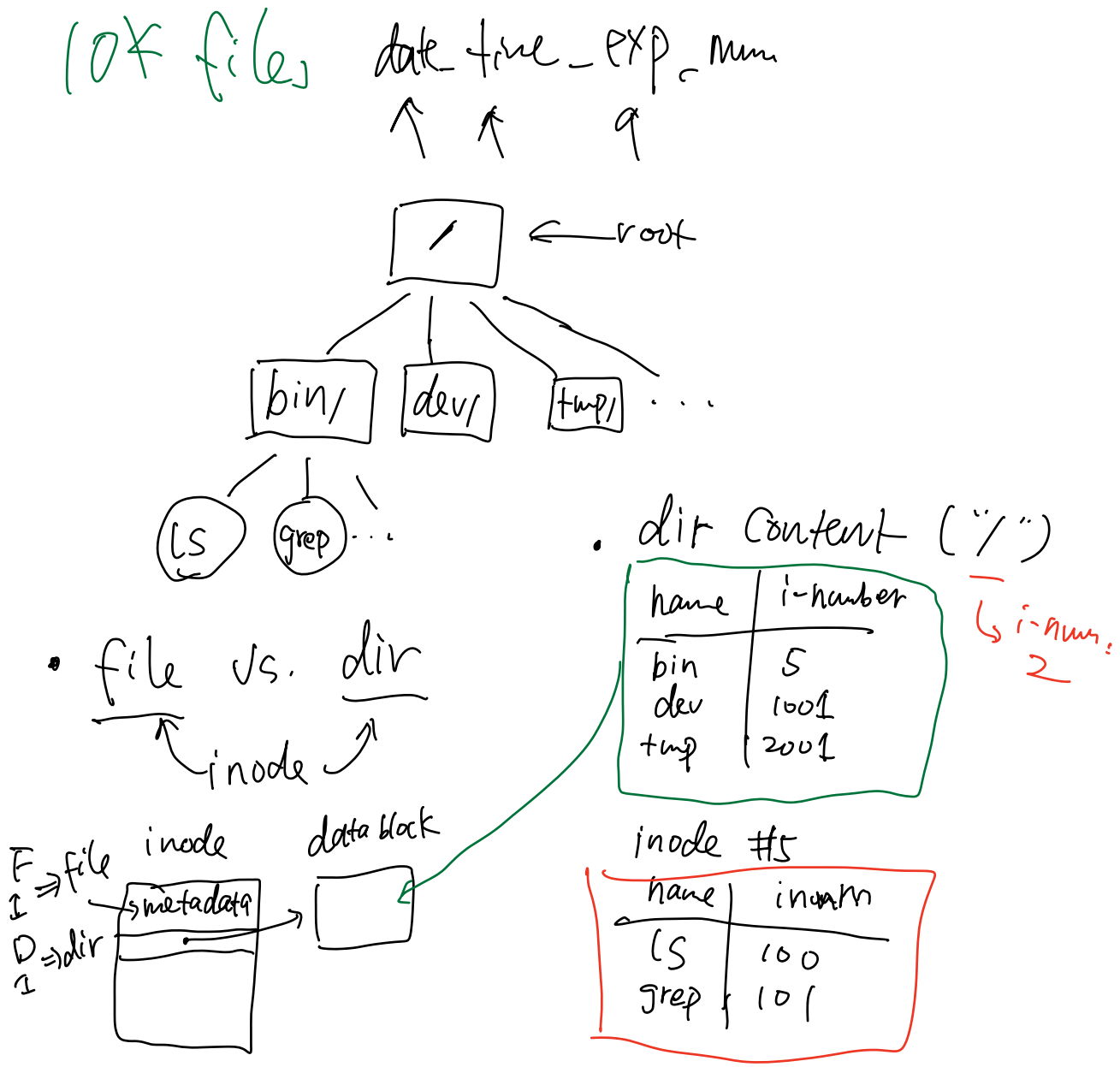
---

projector Failer during this 30 min

Question: how many files can the following toy-fs has?
  sizeof(inode) = 128B
  sizeof(block) = 512B
toy-fs uses 1000 blocks to store inodes

2. Directories

* Problem:
  "Spend all day generating data, come back the next morning, want to use it." F. Corbato, on why files/dirs invented.



• /bin/ls
↳ load inode #2 ("/")
↳ "bin" ⇒ inode #5 ("bin")
↳ "ls" ⇒ inode #100 ("ls")

Question:

```
$ touch /tmp/a
$ ln /tmp/a /tmp/b; ln -s /tmp/a /tmp/sb
$ rm /tmp/a
```

What is the output of this cmd ?

```
$ cat /tmp/b
```

→ work

What is the output of this cmd?

```
$ cat /tmp/sb
```

→ won't work ERROR

and why?

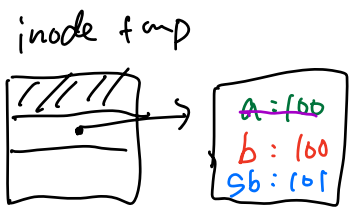
- link, hard / soft

```
$ echo abc > /tmp/a
$ ln /tmp/a /tmp/b
```

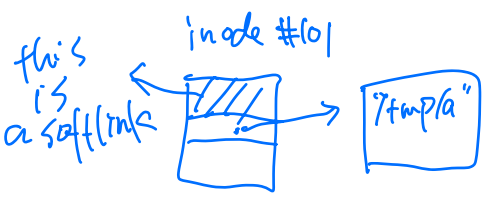
(existing) (new) (alias)

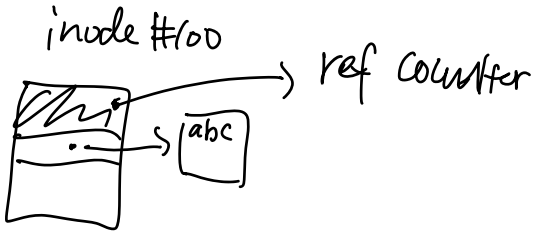
```
$ ln -s /tmp/a /tmp/sb
```

(alias to "name")




```
$ rm /tmp/a
```

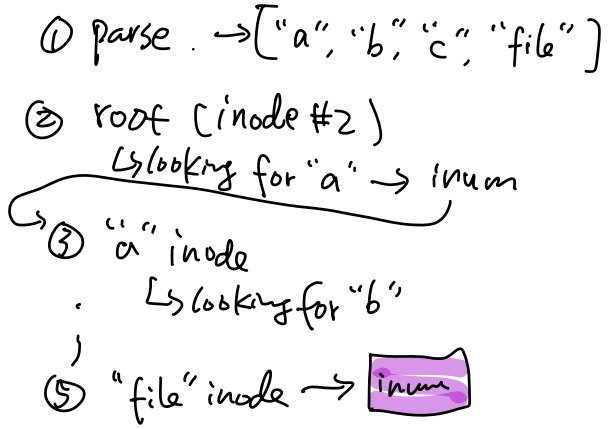




* path walk

  int inum = path2inum(char *pat);

  --how? for example, "/a/b/c/file"



* fs_read - read data from a file

  --how? for example, "read("/a/file", buf, len, offset)" (pseudocode)