Week 11.a
CS3650
03/18 2024
https://naizhengtan.github.io/24spring/

1. FS interface
2. Lab4: fs3650

- Lab3.
- Lab4
- Midterm

- strace → syscalls

$ strace cat file ⇒ ?

- fstat
- read
- write

FILE*
fprintf

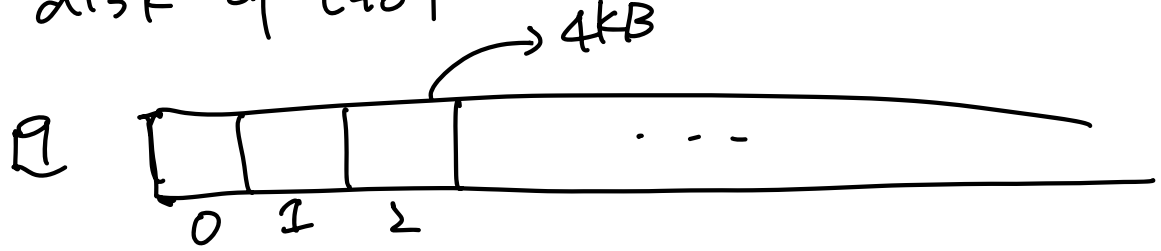
some fs syscalls:

* statfs - report file system statistics
* fstat - get attributes of a file/directory
* readdir - enumerate entries in a directory
* read - read data from a file

* rename - rename a file
* chmod - change file permissions
* creat - create a new (empty) file
* mkdir - create new (empty) directory
* unlink - remove a file
* rmdir - remove a directory
* write - write to a file
* truncate - delete the contents of a file
* utime - change access and modification times

- Lab4: CS3650 fs (fs3650)

(a) disk of lab4

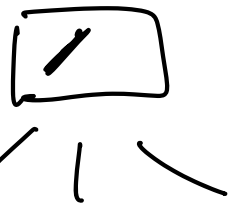



```
int block_read ( void *buf, int lba, int nblks);
```

char buf[4096]; block#lba #blocks

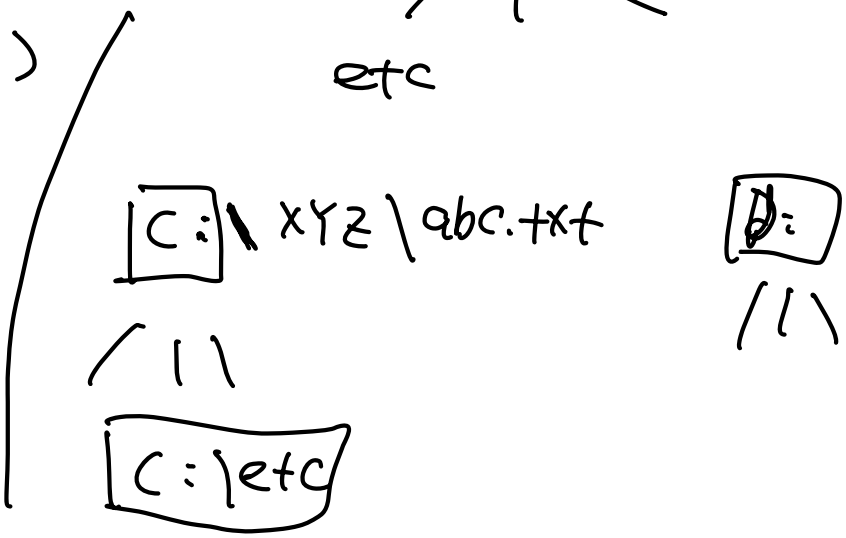
(b) fs3650 data structures

- superblock
- inode
- inode (dir)



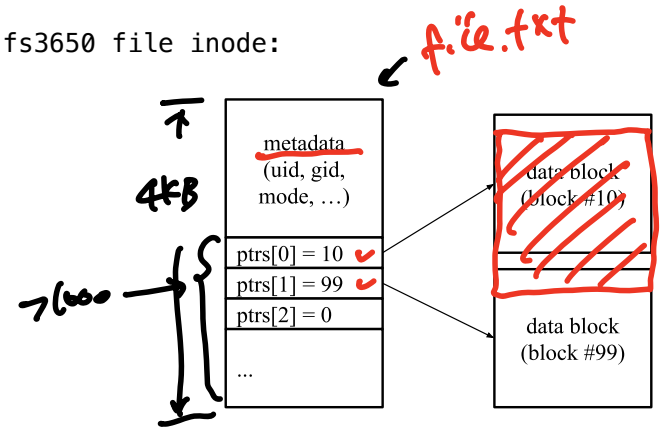
cd /etc; pwd

/etc

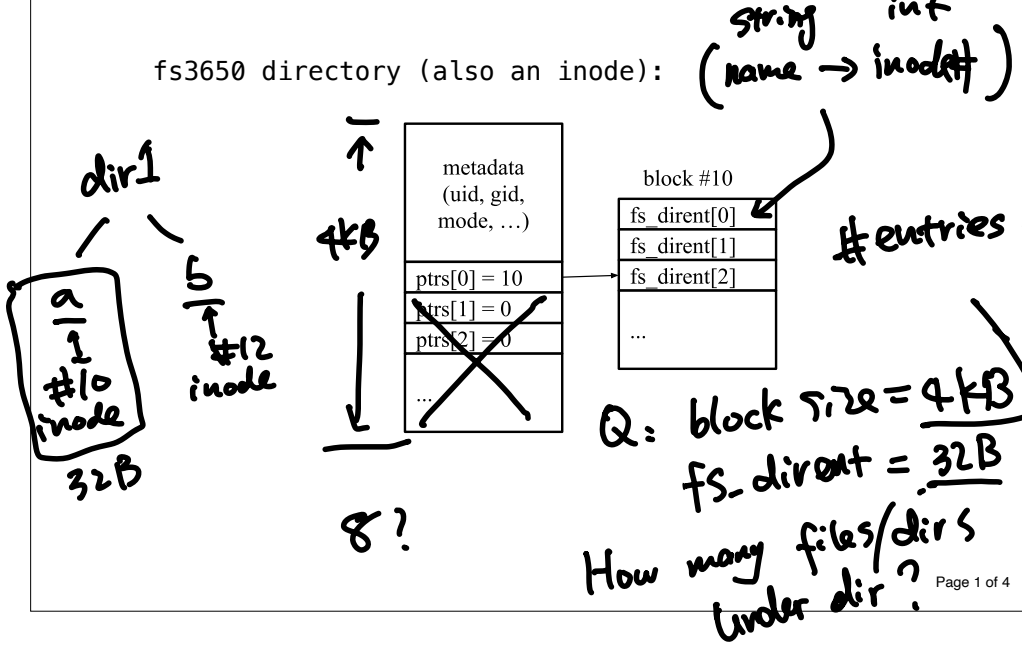


1. fs3650 visualization

fs3650 file inode:



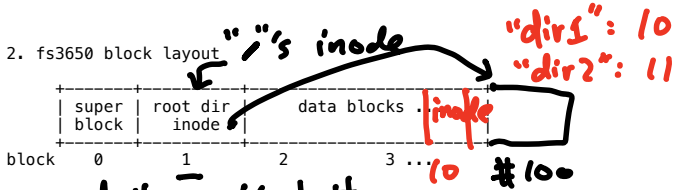
fs3650 directory (also an inode):



Q: block size = 4kB
fs_dirent = 32B
How many files/dirs under dir?

#entries = block size / size of entry = 4kB / 32B = 128

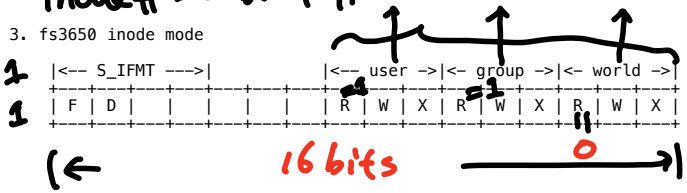
2. fs3650 block layout



inode# == block#

3. fs3650 inode mode

file
F:1
D:1
dir



4. interfaces

(a) fs_getattr - get attributes of a file/directory

```
int fs_getattr(const char *path, struct stat *sb,
               struct fuse_file_info *fi)
```

(b) fs_readdir - enumerate entries in a directory

```
int fs_readdir(const char *path, void *ptr, fuse_fill_dir_t filler, off_t offset,
               struct fuse_file_info *fi, enum fuse_readdir_flags flags)
```

(c) fs_read - read data from a file

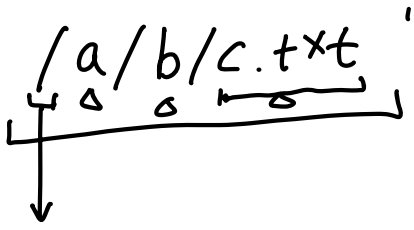
```
int fs_read(const char *path, char *buf, size_t len, off_t offset,
            struct fuse_file_info *fi)
```

```
CS3650 file system

// 1. Read/write disk in Lab4 (CS3650 file system)
// borrowed from Lab4, homework.c

   /* disk access. All access is in terms of 4KB blocks; read and
    * write functions return 0 (success) or -EIO.
    */
   extern int block_read(void *buf, int lba, int nblks);
   extern int block_write(void *buf, int lba, int nblks);

   /* below is a toy example of reading from a disk block
    */

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


// 2. CS3650 file system data structures
// borrowed from fs3650.h with minor changes


   #define FS_BLOCK_SIZE 4096
   #define FS_MAGIC 0x33363530

   #define MAX_PATH_NAMES 20    /* max depth of a path */
   #define MAX_PATH_BYTES 1024  /* max length of a path */

   /* how many buckets of size M do you need to hold N items?
    */
   #define DIV_ROUND_UP(N, M) ((N) + (M) - 1) / (M)

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

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

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

```



```
   struct fs_inode {
       uint16_t uid;
       uint16_t gid;
       uint32_t mode;
       uint32_t ctime;
       uint32_t mtime;
       int32_t  size;
       uint32_t ptrs[FS_BLOCK_SIZE/4 - 5]; /* inode = 4096 bytes */
   };


// 3. "struct stat" from Linux
// borrowed from "man fstat"

   struct stat {
       dev_t     st_dev;         /* ID of device containing file */
       ino_t     st_ino;         /* Inode number */
       mode_t    st_mode;        /* File type and mode */
       nlink_t   st_nlink;       /* Number of hard links */
       uid_t     st_uid;         /* User ID of owner */
       gid_t     st_gid;         /* Group ID of owner */
       dev_t     st_rdev;        /* Device ID (if special file) */
       off_t     st_size;        /* Total size, in bytes */
       blksize_t st_blksize;     /* Block size for filesystem I/O */
       blkcnt_t  st_blocks;      /* Number of 512B blocks allocated */

       /* Since Linux 2.6, the kernel supports nanosecond
          precision for the following timestamp fields.
          For the details before Linux 2.6, see NOTES. */

       struct timespec st_atim;  /* Time of last access */
       struct timespec st_mtim;  /* Time of last modification */
       struct timespec st_ctim;  /* Time of last status change */

   #define st_atime st_atim.tv_sec      /* Backward compatibility */
   #define st_mtime st_mtim.tv_sec
   #define st_ctime st_ctim.tv_sec
   };
```

- fs3650 critical functions
  - walking path



① inode#1
② find "a"'s inode#
③ load "a"'s inode
④ find b's inode#
⑤ load b's inode
⑥ return c.txt's inode#

- read("/a/file.txt", buf, 10, 4096);

① walk "a/file.txt" ⇒ inode#

② ptr[0]
ptr[1]