CMPUT 379 Lab

ETLC E1003: Tuesday, 5:00 – 7:50 PM.

Tianyu Zhang, Peiran Yao

CAB 311: Thursday, 2:00 – 4:50 PM.

Max Ellis, Aidan Bush

Today’s Lab

- Sample FS simulator (vsfs.py - Very Simple File System)
- Crash consistency (fsck.py - File System Consistency Checker)
- FAQ for Assignment 3

Sample FS simulator

- This sample file system contains similar data structure with assignment 3
  - inode bitmap indicates which inodes are allocated
  - inodes table of inodes and their contents
  - data bitmap indicates which data blocks are allocated
  - data indicates contents of data blocks

- The inodes each have three fields:
  - type of file (e.g., f for a regular file, d for a directory)
  - which data block belongs to a file
  - the reference count for the file or directory

- More detail: http://pages.cs.wisc.edu/~remzi/OSTEP/file-implementation.pdf

Crash consistency

- This python file can generate a crashed disk (same format as vsfs.py)
- The possible reason of crash includes:
  - DATA BITMAP corrupt
  - INODE BITMAP corrupt
  - INODE refcnt increased
  - INODE refcnt decreased
  - INODE orphan
  - INODE points to dead block
  - INODE was type file, now dir
  - INODE directory altered to refer to unallocated inode

- More detail: http://pages.cs.wisc.edu/~remzi/OSTEP/file-journaling.pdf

Assignment 3 FAQ

Block number

```
void fs_read(char name[5], int block_num)
void fs_write(char name[5], int block_num)
```

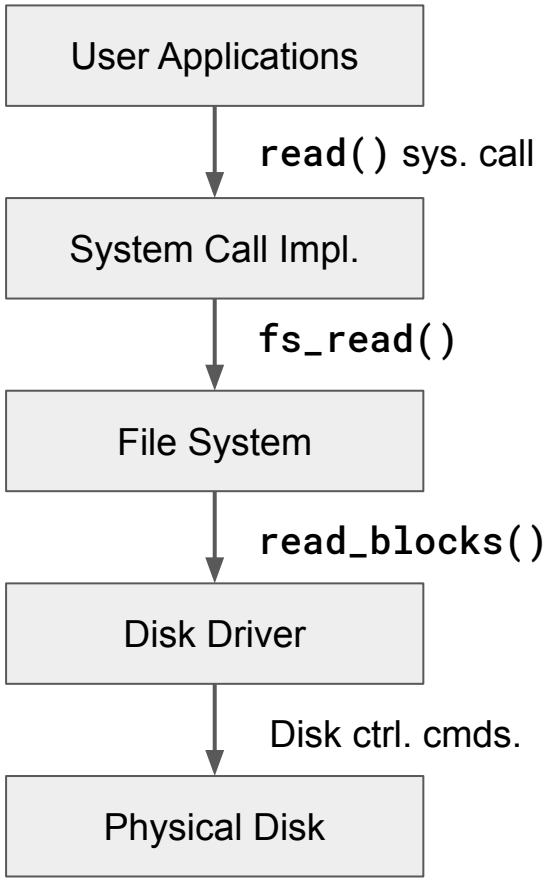
- The `block_num` parameter for `fs_read` and `fs_write` refers to the `block_num`th block **of the file**.
- They are relative to the `start_block` of the file
- Valid values of `block_num` lie between `0` and `used_size-1`

Mounting

- `fs_mount` only **simulates** the process of mounting in a real FS. It affects nothings about the real OS.
- Only the following things need to be done:
  - Check and save a reference of the virtual disk (disk0)
  - Load the superblock of the FS, by reading the virtual disk file.
  - (Optional) Organize the FS layout in certain data structures
  - Do the consistency checks (constraints 1-6)

The role of file systems

- FS provides an abstraction of files and directories that system calls can use
- Even with similar names (`read()` and `fs_read()`), system calls and FS APIs are different.



Manipulate part of a file

- To implement `fs_read`, `fs_write` etc., you may need to read or change certain range of bytes in the virtual disk (e.g. `disk0`).
- **`lseek`**(`int fildes`, `off_t offset`, `int whence`) system call can change the offset of a file descriptor.
- E.g. Read the `[k, k+n]` bytes of file `fd`.

```
int fd = open("./disk0", O_RDONLY);
uint8_t buffer[MAX_BUF];
lseek(fd, k, SEEK_SET);
read(fd, buffer, n);
```