CS140 Review Session
Project 4 – File Systems

Varun Arora

Based on Vincenzo Di Nicola's slide

General Points

- May build project 4 on top of Project 2 or Project 3
- Extra Credit if built on top of Project 3
- Good News: “Easier than PP3”
- Not so good news: Probably the biggest assignment in terms of code lines.
- Read the Design Document before starting. It will help you design your project better.
- Still, Open ended
  - You have to figure out the design to get the required functionality.

Requirements

- Indexed and Extensible Files
- Subdirectories
- Buffer Cache
- Synchronization at a finer level.

Indexed and Extensible Files

- Current file system is an extent based file system.
- The size of file is specified at file creation time.
- It suffers from external fragmentation.

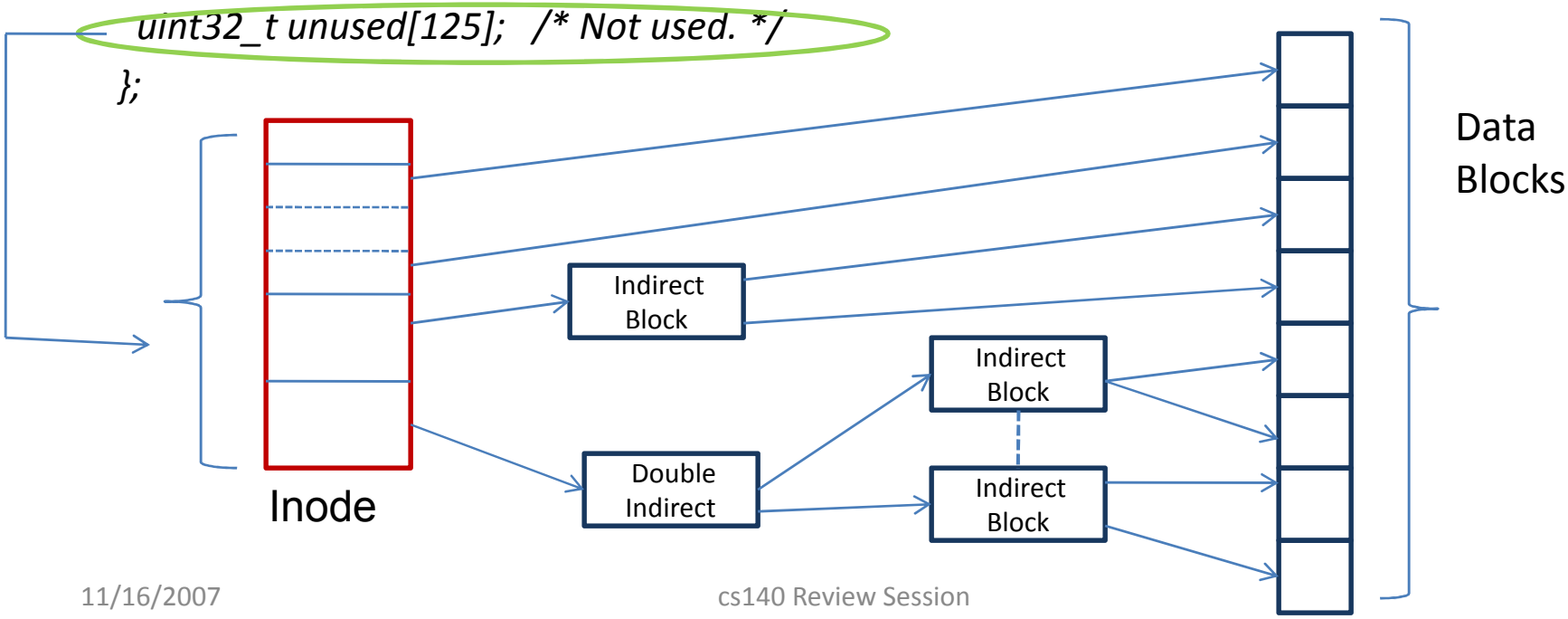
```
struct inode_disk
  {
  disk_sector_t start;          /* First data sector. */
  off_t length;                 /* File size in bytes. */
  unsigned magic;               /* Magic number. */
  uint32_t unused[125];         /* Not used. */
 };
```

- Continuous sectors allocated on disk for the file.

Indexed and Extensible Files (cont..)

- Modify the on-disk inode structure, to remove external fragmentation.
- Generally some indexed structure of direct, indirect and double indirect blocks is used.

```
struct inode_disk {
disk_sector_t start;      /* First data sector. */
off_t length;             /* File size in bytes. */
unsigned magic;           /* Magic number. */
uint32_t unused[125];     /* Not used. */
};
```



Indexed and Extensible Files (cont..)

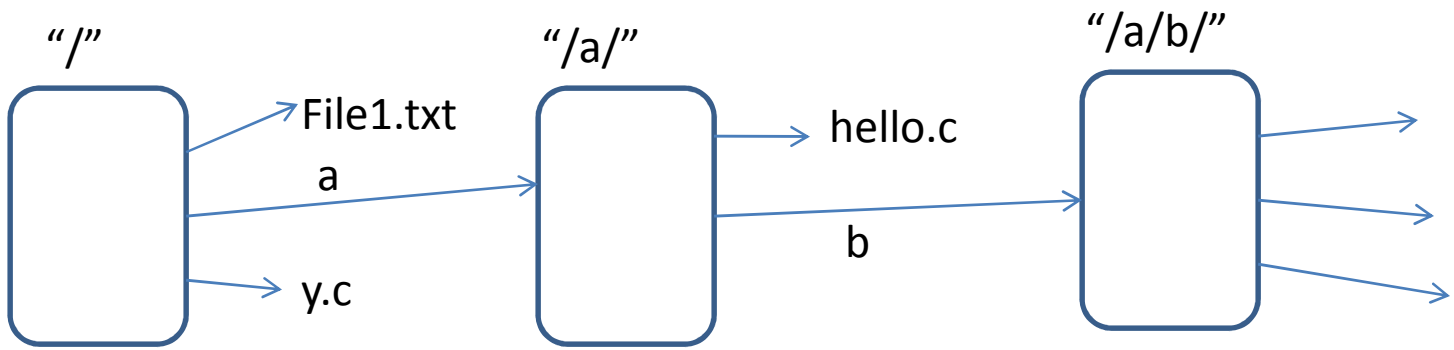
- Similar to hierarchical paging structure for VM.
- Size of the on disk inode structure exactly equal to DISK_SECTOR_SIZE.
- Size of each block is 512B.
  - Each block can store 512B/4B = 128 block addresses.
- Assume that disk will not be larger than 8MB(minus metadata).
- Must support files are large as the disk.
- Don't keep any upper limit on the number of files which can reside in the FS.

Indexed and Extensible Files(cont..)

- Implement File Growth.
- Writing past the EOF should be possible and should extend the file to that position.
  - Might need to get additional data blocks and update the inode data structure.
- A read from a position past the EOF should return zero bytes.
- Handle race between a reader reading past EOF and the writer writing past EOF

Subdirectories

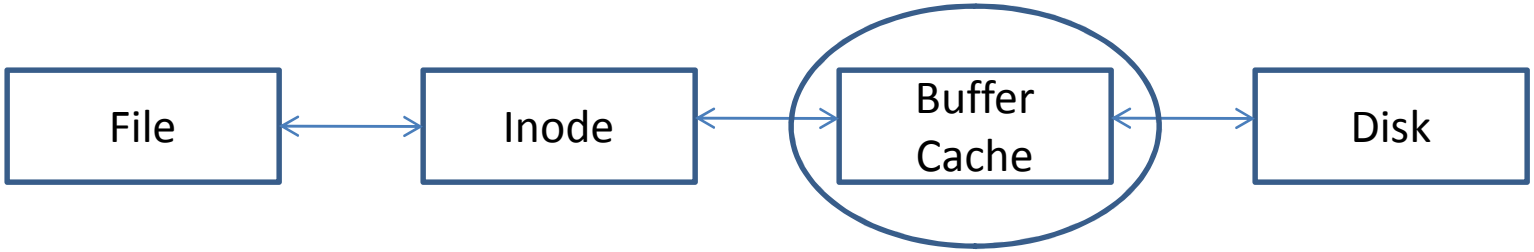
- In current FS, all files live in a single directory.
- Directories are files, but have a different layout.
  - Consist of an array of directory entries.
- Ensure directories can expand just like any other file.
- Extend the directory structure in directory.c so that directory entries can be both files and other directories.



Subdirectories (cont…)

- Update existing system calls, to allow relative or absolute path wherever a file name is required.
  - Each process should have a separate cwd.
  - May use strtok_r() to parse path names
- Implement new system calls
  - bool **chdir** (const char **dir*)
  - bool **mkdir** (const char **dir*)
  - bool **readdir** (int *fd*, char **name*)
  - …

Buffer Cache



- Integrate buffer cache early into you design.
- Similar to VM concept
  - Keep a cache of file blocks in main memory.
  - Read/Write call must first check the cache and then go to disk if not present in cache.
- Cache is limited to 64 sectors in size.
- Can keep the free map against the cache size.
- Implement a cache replacement policy as good as "clock" algorithm. (may give preference to metadata)

Buffer Cache (cont…)

- Write-behind policy:
  - don’t write the dirty block immediately to disk.
  - Flush the dirty blocks when they are evicted.
  - Flush the entire cache are a periodical interval and also in filesys_done().
  - Can use timer_sleep from PP1, for periodic flushing.
- Read ahead policy:
  - When one block is read, automatically read the next block from disk.
  - Should be done asynchronously (can use a helper thread).

Synchronization

- Crucial and the harder part of the assignment.
- Remove the external lock on whole FS.
  - Multiple readers can read the same file simultaneously.
  - Multiple writers can write to different block of same file simultaneously
  - Extending a file should be atomic. If A and B trying to write past the EOF, only one should be allowed.
  - Operations on difference directories should be concurrent.
    Operations on same directory can be serialized.

Questions?