Challenge 3

Filip Strömbäck, Ahmed Rezine

March 6, 2024

Instructions

You have to do research on the parts you do not understand. When you solve problems, do not hesitate to:

- State your starting point. What do you know from start? Why is it important?
- State your goal. What do you need to know to reach it?
- State your assumptions. What are you taking for granted or assuming?
- Show step by step how you can go from what you have to what you want.

You will get one credit for each presentation you claim prepared. You may (at random) be selected to present any of the solutions you prepared. When you are selected for presentation:

- For each step in your solution, explain what you were thinking. How did you come up with this? What gave you the clues?
- Practice so you are ready to present more than less from memory (you can still have a note for reference, but you should not need to look at it much). **Aim for 10 minutes.**
- You are not required to have a correct solution to get credit, but **you are required to solve all parts of a problem, to make a serious attempt and to "believe" in your solution**.
- If you are not prepared you loose all credit for the seminar in question.

As audience you should think of how difficult it is to clearly present a solution. Be humble and supportive. You may put forward constructive criticism of the presented solution. Compare to your solution; I did it another way, what will be the difference? You are of course welcome to take notes.

Problem 1

A model of a high end digital cameras will store still images on an attached 64 GiB NAND flash card (145 MiB/s sequential transfer, 500 µs access time, 4 KiB block size). Pictures are stored in raw format at 6048x4032 resolution with 4 bytes per pixel (for best quality), or as jpeg with a compression ratio of at least 10:1 (for space efficiency). The CCD, memory and CPU architecture, and jpeg compression allow for up to 25 full frame pictures per second.

To allow photographers to easier capture "the right moment" it is important to be able to take high speed photo sequences. The typical user will take photos until the card is full, then transfer all images to a computer, and finally erase the card completely.

(a) Assume overhead from finding free blocks and writing index blocks are negligible. If the file system use indexed allocation, how many pictures per second will be feasible if:

  (1) The free blocks to fill are contiguous on disk.

  (2) The free blocks to fill are fragmented in many small regions. Assume a new seek (access time) is required for 50% of all blocks.

(b) Which allocation method would you use for the camera? Why?

(c) Do your solution have any possible drawbacks?

Problem 2

A hand held music player can play most standard audio file formats (mp3, wav, flac). Typical file sizes ranges from a few MiB (mp3) up to GiB for long play high definition lossless audio (24 bit, 192 KHz flac). It has a 64 GiB flash memory with 500 µs access time and 512 byte blocks. You are not bound to use 512 bytes blocks as logical blocks can be multiples of that.

The typical user will transfer files, listen to them until bored, keep the best and replace the rest with new songs, over and over again.

It is important to be able to search in a (possibly a GiB) song without delay, and all reported free capacity must be immediately usable (you should not have to use a defragmentation program).

(a) Suggest and motivate a suitable file allocation method for the player.

(b) Assume each file is directly indexed via a single logical block. Assume 512 KiB logical blocks and 32 bits pointers. Can you index all logical blocks? What is the maximum file size?

(c) Assume again 512 KiB logical blocks. If we fill the flash disk with 5000 music files, how much space do you expect to be wasted due to internal fragmentation? How could you change this amount?

Problem 3

A modern multicore machine serves a member site with high resolution photo albums. Popular images appear in many albums and are frequently accesses by many processes at once.

It is essential to minimize the amount of random disk accesses and the amount of disk to host transfers.

(a) Explain how OS system-wide and process-wide open file tables are used to minimize overall disk access and file transfer time from the array.

It is essential that the OS can unload any cached but no longer needed file content as soon as a process terminate.

For optimization purpose it is desirable if the OS can keep track of where each process will read next in each file.

(b) To support your internal organization, which file system calls must be provided for processes to use?

(c) Demonstrate how your structure handles a file request. What will be performed when the first process wants to read one entire file for the first time?

(d) Demonstrate how a simultaneous read request for the same file by another process is handled, and what benefits your structure create in terms of reduced disk accesses and transfer time.

Problem 4

A small OS provide operations to create/open/read/write/close/remove inodes. However, there is no structure in place to keep track of, or name, inodes. The system keeps track of free blocks in an inode located at block 0. Block 1 is an inode reserved for a the root of a directory tree.

(a) Suggest a detailed design of a simple (include just the most essential parts) structure to keep track of files. Show what and where you store different data.

Different users want their own home account file area where they can organize files in folders. Your simple design may restrict file name lengths and folder sizes.

(b) Show how you add an inode representing a file to your structure.

(c) Show how you add an inode representing a folder to your structure.

(d) Show how you add an inode representing a hard link to your structure.

Hint: In this problem we already have *inodes*, which can be thought of as files without names. The problem is about using the inodes to implement directories that allow us to name files. Essentially, think of what needs to be stored in an inode that is a directory, and any other data that we might need to store for each inode.

Problem 5

A standard USB stick uses a 32 bit file allocation table (FAT32) for interoperability, since FAT32 is a file system with read and write support by virtually every operating system.

FAT32 can be considered a combination of linked and indexed allocation. At its core blocks are linked, but all pointers are moved into a central index, also doubling as free space management system.

(a) Consider using a file allocation table that has to be loaded in main memory for a new 8 TiB disk. How large should the logical blocks be in order to restrict the allocaiton table size to 128 MiB?

(b) Compare the file allocation table to linked allocation. What benefits and drawbacks do you find?

(c) Compare the file allocation table to indexed allocation. What benefits and drawbacks do you find?

Problem 6

The following operations all take place within the same directory on the same partition on the same disk. Assume a very simple directory similar to the one you came up with in problem 4.

1. create the file `file_1`
2. create the file `file_2`
3. create the hard link `hardlink` referring to `file_1`
4. create the symbolic link `symlink` referring to `file_2`
5. remove the file `file_1`
6. remove the file `file_2`
7. move the file `hardlink` to `file_1`
8. move the file `symlink` to `file_2`
9. open `file_1`
10. open `file_2`

Recall we work in the same directory.

(a) Explain how the directory changes in each step.

(b) Will any operation affect location or content of file data blocks on disk?

(c) Will all operations succeed? If not, why?

Problem 7

A new 32 TiB drive will use indexed allocation of inodes. It has 10 ms access time and will present 512 byte logic block size to the OS.

(a) How much disk space and DRAM memory would be required to manage free blocks if a bitmap (or bitvector) was used with the logic block size?

(b) How much time would it take to create a new 4 GiB file (the legally bought movie "The prestige" for example) if free blocks was linked? (assume the worst case)

(c) Can you propose a better solution to manage free blocks on this drive.

(d) With your solution, assume the free block management system is lost. Suggest a way to recover the free blocks information. You can assume files are managed by a undamaged multilevel directory structure and linked allocation.

Problem 8

The following operations all take place within the same directory on the same partition on the same disk. Assume removing a file automatically opens it, erases it from the directory immediately and marks it for later deallocartion in the system-wide open file table (in its open FCB/inode). Assume this deallocation happens when all processes that have opened the file close it. This is the way that most Linux filesystems, and the Pintos educational OS works. The curios may take a look at the source code file `src/filesys/inode.c`, functions `inode_remove`, and `inode_close`, and file `src/filesys/filesys.c`, function `filesys_remove`.

The file system initially contains an empty file named `data_1`.

The processes A, B, and C are currently running in the system. None of them have any files opened.

1. process A open the file `data_1` as fd `1`
2. process B remove the file `data_1`
3. process B create the file `data_1`
4. process B open the file `data_1` as fd `1`
5. process B write `"akerueh"` to fd `1`
6. process C write `"hakuna matata"` to fd `1`
7. process B close fd `2`
8. process A write `"heureka"` to fd `1`
9. process A close fd `1`

(a) Demonstrate how the directory and open file tables change in each step.

(b) What data content will be in `data_1` after the operations?

(c) Will all operations succeed? If not, why?