CS2106 Operating Systems

Semester 1 2021/22

Tutorial 11

1. **(Putting it together)** This question is based on a "simple" file system built from the various components discussed in lecture 12.

Partition information (free space information):

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	1
16	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
1	0	1	0	1	0	1	0	1	1	1	0	0	1	0	0

Directory Structure + File Information:

- Directory structures are stored in 4 "directory" blocks. Directory entries (both files and subdirectories) of a directory are stored in a single directory block.

Directory entry:

- For File: Indicates the first and last data block number.
- For Subdirectory: Indicates the directory structure block number that contains the subdirectory's directory entries.
- The "/" root directory has the directory block number 0.

0	1	2	3
y |Dir | 3 f |File| 12| 2 x |Dir | 1	g |File| 0| 31 z |Dir | 2	k |File| 6| 6	i |File| 1| 3 h |File|27|28

File Data:

- Linked list allocation is used. The first value in the data block is the "next" block pointer, with "-1" to indicate the end of data block.
- Each data block is 1 KB. For simplicity, we show only a couple of letters/numbers in each block.

0	1	2	3	4	5	6	7
11 AL	9 TH	-1 S!	-1 ND	23 GS	-1 SO	-1 :)	10 TE
8	**9**	**10**	**11**	**12**	**13**	**14**	**15**
31 RE	3 EE	28 M:	31 OH	19 SE	13 AH	4 IN	17 NO
16	**17**	**18**	**19**	**20**	**21**	**22**	**23**
30 YE	2 OU	1 ON	17 RI	26 EV	14 AT	21 DA	7 YS
24	**25**	**26**	**27**	**28**	**29**	**30**	**31**
-1 HO	18 ME	0 AL	30 OP	-1 -(	5 LO	21 ER	-1 A!

a. (Basic Info) Give:
    - The current free capacity of the disk.
    - The current user view of the directory structure.
b. (File Paths) Walkthrough the file path checking for:
    - "/y/i"
    - "/x/z/i"
c. (File access) Access the entire content for the following files:
    - "/x/z/k"
    - "/y/h"
d. (Create file) Add a new file "/y/n" with 5 blocks of content. You can assume we always use the free block with the smallest block number. Indicate all changes required to add the file.

2. **(File System Overhead)** Let us find out the overhead of FAT16 and ext2 file systems. To have a meaningful comparison, we assume that there is a total of 2^{16} 1KB data blocks. Let us find out how much bookkeeping information is needed to manage these data blocks in the two file systems. Express the overhead in terms of number of data blocks.

    a. Overhead in FAT16: Give the size of the two copies of file allocation table.
    b. Overhead in ext2 is a little more involved. For simplicity, we will ignore the super block and group descriptors overhead.

        Below are some known restrictions:
        - The two bitmaps (data block and inode) each occupies a single disk block.
        - There are **184** inodes per block group.

        Calculate the following:
        i. Number of data block per block group.
        ii. Size of the inode table per block group.
        iii. Number of block groups in order to manage 2^{16} data blocks.
        iv. Combine (i – iii) to give the total overhead.

    c. Comment on the runtime overhead of the two file systems, i.e. how much memory space is needed to support file system operations during runtime.

3. **(Adapted from AY 2019/20 S 1 exam)**
   You are required to implement and optimize the storage and search of multiple fixed file sizes of 1MB. Each file has a unique numeric identifier, and it is saved on disk using ext2 file system. The number of files is unlimited (say, 10 million), but it cannot grow beyond the disk size (maximum 10TB). We want to optimize this file system to store and search fast for files by identifier. You should use some of the methods studied under file system implementation for your optimizations.

   a. First, assume that all files are stored in a directory. What would be the main disadvantage when we want to add a new file? What would be the main disadvantage when searching for a file by identifier?
   b. Assume that you do not have any restrictions in terms of how many directories you can use. How would you mitigate the disadvantages shown at point a. for storing a file? Explain your optimizations.
   c. How would you mitigate the disadvantage(s) shown at point a. for searching a file by identifier? Explain your optimizations.
   d. You are allowed to use additional data structures to speed up the storing and searching of files. What data structures would you use to implement your optimizations? Be specific about what you store in each data structure.
   e. Explain how your suggestions (b, c, d) improve the storage and search times for a file. Estimate the overhead in terms of space and time of creating, maintaining, and using the additional data structures from point d.

4. **(Adapted from AY 2020/21 S 1 exam)**
   In a particular ext2 filesystem, many data blocks belonging to different files have the same contents. We can implement a mechanism to reduce the space on disk occupied by such files.

   a. Define a new system call, `clone_file()`, that makes a copy of a file without allocating new blocks on disk. Give the parameters, return value and description for `clone_file()`, and explain how to use the system call.
   b. How would you allow for files that are cloned by `clone_file` to later be modified independently of each other? For example, if file A is cloned by `clone_file` to file B, and later on a write to file A is done, file B should still reflect the original contents. You should avoid duplication of blocks that are not yet modified.
   c. Explain how you would implement the mechanism in (b) in the ext2 filesystem. If changes to structures are required, explain those as well.

Questions for your own exploration

5. **(Adapted from AY 2016/17 S 1 exam)**
   Most OSes perform some higher-level I/O scheduling on top of just trying to minimize hard disk seeking time. For example, one common hard disk I/O scheduling algorithm is described below:
   a. User processes submit file operation requests in the form of **operation(starting hard disk sector, number of bytes)**
   b. OS sorts the requests by hard disk sector.
   c. The OS merges requests that are nearby into a larger request, e.g. several requests asking for tens of bytes from nearby sectors merged into a request that reads several nearby sectors.
   d. The OS then issues the processed requests when the hard disk is ready.

   a. How should we decide whether to merge two user requests? Suggest two simple criteria.

   b. Give one advantage of the algorithm as described.

   c. Give one disadvantage of the algorithm and suggest one way to mitigate the issue.

   d. Strangely enough, the OS tends to intentionally delay serving user disk I/O requests. Give one reason why this is actually beneficial using the algorithm in this question for illustration.

   e. In modern hard disks, algorithms to minimise disk head seek time (e.g. SCAN variants, FCFS etc.) are built into the hardware controller. i.e. when multiple requests are received by the hard disk hardware controller, the requests will be reordered to minimise seek time. Briefly explain how the high-level I/O scheduling algorithm described in this question may conflict with the hard disk's built-in scheduling algorithm.

6. **This question is not in scope for the exam.**
   (Cluster Allocation, Adapted from [SGG]) A common modification to the file allocation scheme is to allocate several contiguous blocks instead of a single disk block for every allocation. This variation is known as the disk block cluster in FAT file systems. The design of cluster can be one of the following:

   a. Fixed cluster size, e.g. one cluster = 16 disk blocks.
   b. Variable cluster size, e.g. one cluster = 1 to 32 disk blocks.
   c. Several fixed cluster size, e.g. one cluster = 2, 4, 8 or 16 disk blocks.

   Discuss the changes to the file information in order to support cluster. Briefly state the general advantage and disadvantage of the various cluster design.