IT5002 Computer Systems and Applications
Tutorial 9

1. We consider a simple file system illustrated below:

Partition Information (Free Space Information)
- Bitmap is used, with a total of 32 file data blocks (1 = Free, 0 = Occupied):

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`"

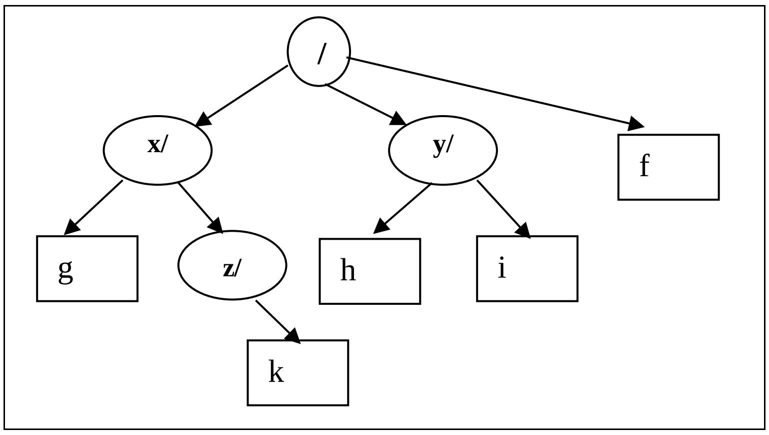
c. (File access) Access the entire content for the following files:
   - "`/x/z/k`"
   - "`/y/h`"

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.**

ANS:

a.

- **~12KB free space (12 '1' in Bitmap, each data block is 1KB). Due to the linked list file allocation overhead, the actual capacity is a little bit smaller.**



b.

Only the directory block number is indicated:

- `/y/i`: 0, 3 (successful)
- `/x/z/i`: 0, 1, 2 (failed)

c.

- `/x/z/k`: block 6, content = "`:)`"
- `/y/h`: blocks 27, 30, 21, 14, 4, 23, 7, 10, 28, content = "`OPERATINGSYSTEM:-(`"

d.

Bitmap updated: Bit 5, 8, 13, 15, 16 changed to 0

Directory block 3 (for `/y`) updated: "`n |File| 5|16`" added

Data Blocks 5, 8, 13, 15, 16 (next block pointer changed, with -1 in block 16).

2. Let us compare and contrast the various ways of implementing files in this question.

Below are the hardware parameters:

a. Number of disk blocks = 2^{16} = 65,536 blocks (i.e. each disk block number = 2 bytes).
b. Disk block size = 512 bytes.

Points of comparison:

- **Total Number of Data Blocks:** Number of disk blocks needed to store the file data.
- **Overhead**: Extra bookkeeping information in bytes. Focus only on information directly related to keep track of file data. You can ignore the space needed for file name and other metadata for this question.
- **Worst Case Disk Access:** The worst case number of disk accesses needed to get to a particular block in the file. May include accessing book keeping information if they are in disk. Specify the block which causes the worst case.

File data allocation schemes under consideration:

A. Contiguous
B. Linked list
C. File allocation table
D. Index block

Fill in the following table for the indicated file size. State assumptions for your calculation (if any).

File size: 100 b	**Total Number of Data Blocks**	**Overhead**	**Worst Case Disk Access**
A			
B			
C			
D			

File size: 132,000 b	**Total Number of Data Blocks**	**Overhead**	**Worst Case Disk Access**
A			
B			
C			
D			

File size: 33,554,432 b	**Total Number of Data Blocks**	**Overhead**	**Worst Case Disk Access**
A			

B			
C			
D			

ANS:

Note that with different assumptions, you may need to adjust your answer accordingly.
Assumptions:
For A, the length of blocks is 2 bytes (i.e. can take all disk blocks if needed).
For B, we keep pointers to both first and last block.
For D, we use the "linked" scheme to chain up the index blocks when needed.

Basic Calculation:
For B, one disk block can store 512-2 (the "next" disk block pointer) = 510 bytes only. Although the "next" disk block pointer overhead is already reflected by the larger number of disk blocks required, we still list them explicitly for learning purpose.
For C, the whole FAT cost $2^{16} * 2 = 2^{17}$.
For D, each index block can store 512 / 2 = 256 disk addresses. We use the last disk address to chain to the next index block, i.e. 255 disk addresses can be stored for file data blocks.

File size: 100 b	**Total Number of Data Blocks**	**Overhead**	**Worst Case Disk Access**
A	**1**	**2 (start) + 2 (length)**	**1**
B	**1**	**2 (first) + 2 (last) + 2 (1 pointer)**	**1**
C	**1**	**2^{17} (FAT) + 2 (start)**	**1**
D	**1**	**512 (Index Block) + 2 (Location of Idx Blk)**	**1 (Index block) + 1**

File size: 132,000 b	**Total Number of Data Blocks**	**Overhead**	**Worst Case Disk Access**
A	**258**	**2 (start) + 2 (length)**	**1**
B	**259**	**2 (first) + 2 (last) + 518 (259 pointers)**	**258 access for the 2nd last block.**
C	**258**	**2^{17} (FAT) + 2 (start)**	**1**
D	**258**	**1024 (2 × Index Block) + 2 (Location of Idx Blk)**	**2 (Index block) + 1 for any block beyond 255th.**

File size: 33,554,432 b	**Total Number of Data Blocks**	**Overhead**	**Worst Case Disk Access**
A	**65,536**	**2 (start) + 2 (length)**	**1**

B	**Cannot store (Need 65,794)**	**---**	**---**
C	**65,536**	**2^{17} (FAT) + 2 (start)**	**1**
D	**Cannot store (Need 65,794)**	**---**	**---**

3. We are given the following table of contents in an inode. Suppose that each data block holds 2048 bytes of data. List down the block numbers that would be read/written by the following operations. Assume that fp points to a validly open file with this TOC.

Table of Contents
15
18
12
22

Operation	Block number read/written
fseek(fp, 1121, SEEK_SET)	-
fwrite(buff, sizeof(char), 128, fp)	**15**
fseek(fp, 5523, SEEK_SET)	-
fread(buff, sizeof(char), 256, fp)	**12**
fseek(fp, 8121, SEEK_SET)	-
fwrite(buff, sizeof(int), 25, fp)	**22**