**總分 60 分(另外 40 分為報告分數)**

1. (10 pts) File system must keep track of the locations and the order of disk blocks that are allocated to files. One possible method is the indexed allocation scheme. Indexed allocation scheme maintains an index block (or in the inode) for each file that contains pointers point to the data blocks of files. That is, in the index block (or in the inode), the $i^{th}$ entry (pointer), points to the $i^{th}$ block of the file, i.e., a **direct pointer** scheme. However, using direct pointer scheme is not easy to support large files. To address the issue, as shown in Figure 1, Linux (Unix-based OS) adopts a combined scheme. The first 12 entries are direct pointers pointing to the data block. Then, the 13th entry is an **indirect** pointer that points to an index block which in turn points to the corresponding data blocks. Finally, the 14th entry is a **double indirect** pointer. (a) Please state the performance implication of combined scheme of Linux? Hint: Please compare the performance of small files and large files and why their performances are different. (b) Please state the design implication of the combined scheme adopted of Linux? Hint: Linux designers should observer something so that they designed the above mentioned combined scheme.

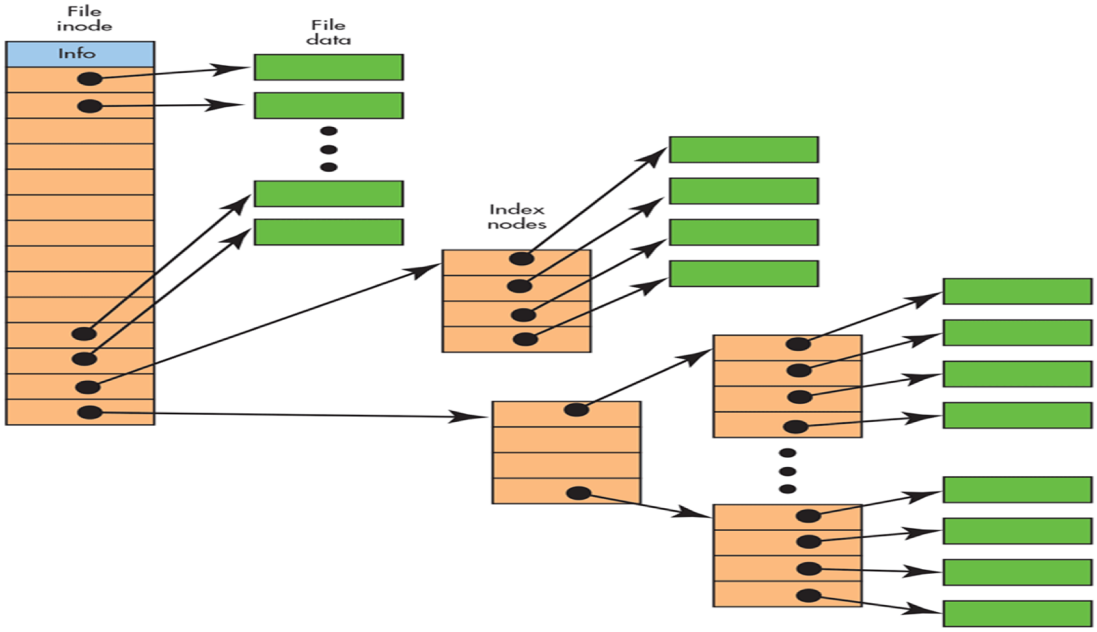


Figure 1. The combined scheme in Linux.

2. (20 pts) (a) Usually, we need to open a file before accessing the file. Why? (b) Assume the file system is ***vsfs***. Upon issuing an *open("/foo/bar.txt", O RDONLY)* system call, please show the blocks, and their corresponding order, that needs to be accessed for implementing the system call. ***Hint***: The first step must access the disk block containing the root inode. Here, we assume that the sizes of the directory files of both "/" and "/foo" directories are one block. Besides, we assume that the ***root inode number*** has been known by reading the superblock upon

mounting. (c) Besides, how many disk blocks need to be accessed? Please show the access sequence. (d) From above example, the amount of I/O generated by the *open()* is proportional to the ***length of the pathname***. Why? (e) The amount of I/O generated by the *open()* is also proportional to ***the size of all visited directories***. Why? (Here, we assume that the directory file is implemented by a linking list that links all of the directory file's entries.)

3. (15 pts) Figure 2 shows a simplified diagram of a file system. The file system maintains 8-bit inode bitmap, 8-bit data bitmap, 8 Inodes, and 8 data blocks on the disk. Besides, assume that a file is created and consists of a single block. Assume that a workload **appends** a single data block to the file. (a) How many disk writes must be issued for appending a data block? Why? (b) Assume that the system is crashed after issuing the first write. Please show the problem if only the first single disk write succeeds. (c) To address the issue, write-ahead log (journaling) is proposed. Please briefly address the idea of write-ahead log, i.e., why the read-ahead log scheme can solve the inconsistency problem?

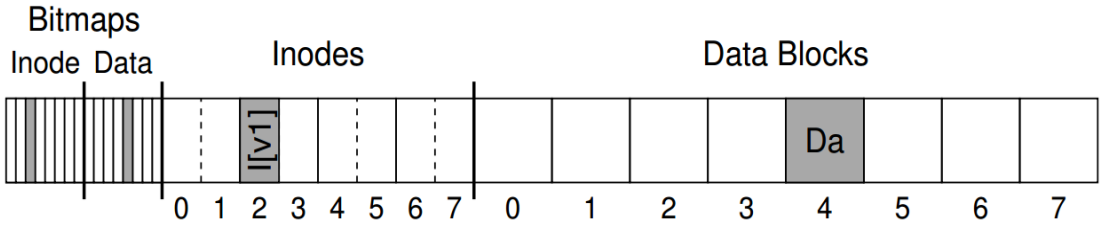


Figure 2. A simple file system.

4. (5 pts) (a) Please state the problem if without Virtual File System (VFS). (b) How the VFS solve the problem.
5. (5 pts) Please state briefly the idea of LFS (Log-structure File System).
6. (5 pts) Why the format of directory file (directory structure) is changed from Fig. 3 to Fig. 4?

| ***name*** | ***Attributes*** |
|---|---|
| *.* | *permission =..., access time = ..., size = ..., location = block 4, 5, 8, 10...* |
| *..* | *permission =..., access time = ..., size = ..., location = block 14, 15, 16, 20...* |
| *foo* | *permission =..., access time = ..., size = ..., location = block 23, 25, 28, 40...* |
| *bar* | *permission =..., access time = ..., size = ..., location = block 41, 45, 48, 50...* |
| *foobar* | *permission =..., access time = ..., size = ..., location = block 54, 55, 56, 57...* |

Fig. 3

| *name* | inode num. |
|---|---|
| . | 0 |
| .. | 1 |
| *foo* | 4 |
| *bar* | 3 |
| *foobar* | 7 |

| inode num. | *Attributes* |
|---|---|
| 0 | *permission =..., access time = ..., size = ..., location = block 4, 5, 8, 10...* |
| 1 | *permission =..., access time = ..., size = ..., location = block 14, 15, 16, 20...* |
| 2 | *permission =..., access time = ..., size = ..., location = block 23, 25, 28, 40...* |
| 3 | *permission =..., access time = ..., size = ..., location = block 41, 45, 48, 50...* |
| 4 | *permission =..., access time = ..., size = ..., location = block 54, 55, 56, 57...* |

Fig. 4