CS 4410

Operating Systems

File System Implementation

Summer 2016

Cornell University

Today

- File allocation
- Unix file system
- Log-structured file system

File system: two design problems

- User interface:
  - File, directory, attributes, allowed operations.
- Hardware interface:
  - Map logical file system onto storage devices.
  - Manage free storage space.
    - Bit vector,
    - Linked list, …

File

- Data of a file is mapped to several blocks in the storage device.
- For each file, the OS maintains a structure FCB (file control block) with information about:
  - the location of data blocks,
  - size,
  - permissions,
  - owner …
- FCB is stored in yet another block.

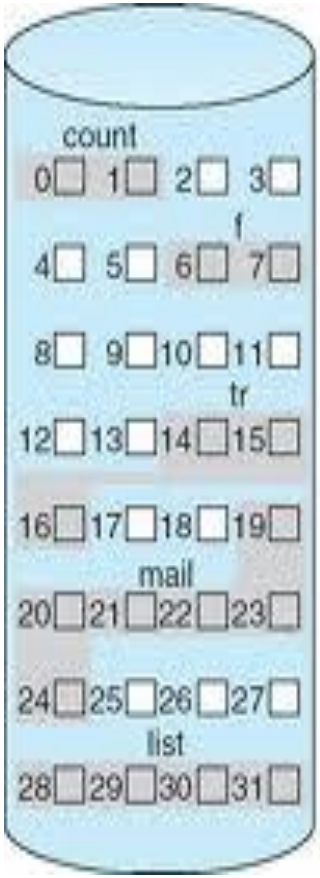
Directory

- The main function of a directory is to map the ASCII name of the file onto the information needed to locate the data.
- A directory can be implemented as a list of entries.
  - Each entry is a pair of an ASCII name and an FCB.
- Every time we open a file, which has not been opened yet in the system,
  - we find its directory and
  - search its entry.
- A directory is stored in blocks, too.
- The OS maintains an FSB for each directory.

Allocation Methods

- How do we allocate space (blocks) to the files so that:
  - Disk space is utilized effectively.
  - Files are accessed quickly.

Contiguous Allocation



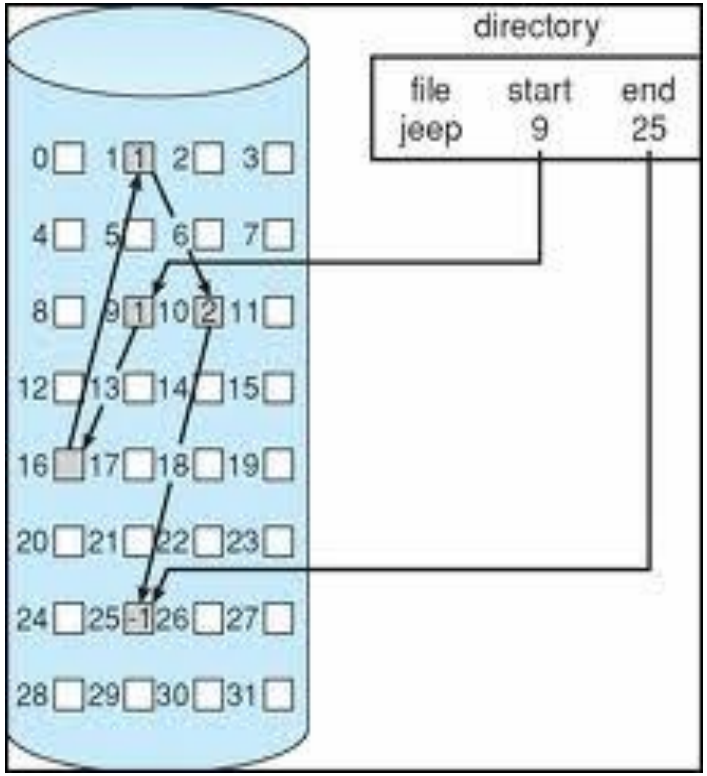
directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Contiguous Allocation

- Each file occupy a set of **contiguous blocks** on the disk.
- **Minimal disk seeks** and **seek time**.
- The **directory entry** for each file indicates the address of the **starting block** and the **number of blocks used**.
- It supports both sequential and direct access.
- Difficulty in finding free space.
  - Dynamic storage-allocation problem
  - External fragmentation
    - Solution: Compaction
  - Determine space needed for a file.

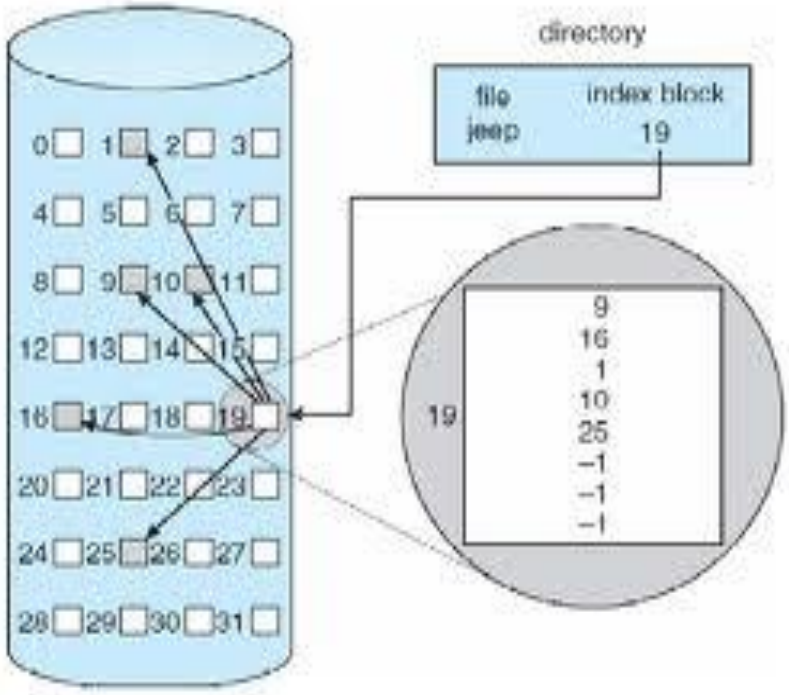
Linked Allocation



Linked Allocation

- A file is a **linked list of disk blocks**.
- The **directory entry** contains a **pointer to the first** and **last** blocks.
- Each block contains a **pointer to the next** block.
  - They consume space.
- Easy block allocation.
- Effective only for sequentially-access files.
- Solution: Allocate clusters rater than blocks.
- Another Problem: Reliability

Indexed Allocation



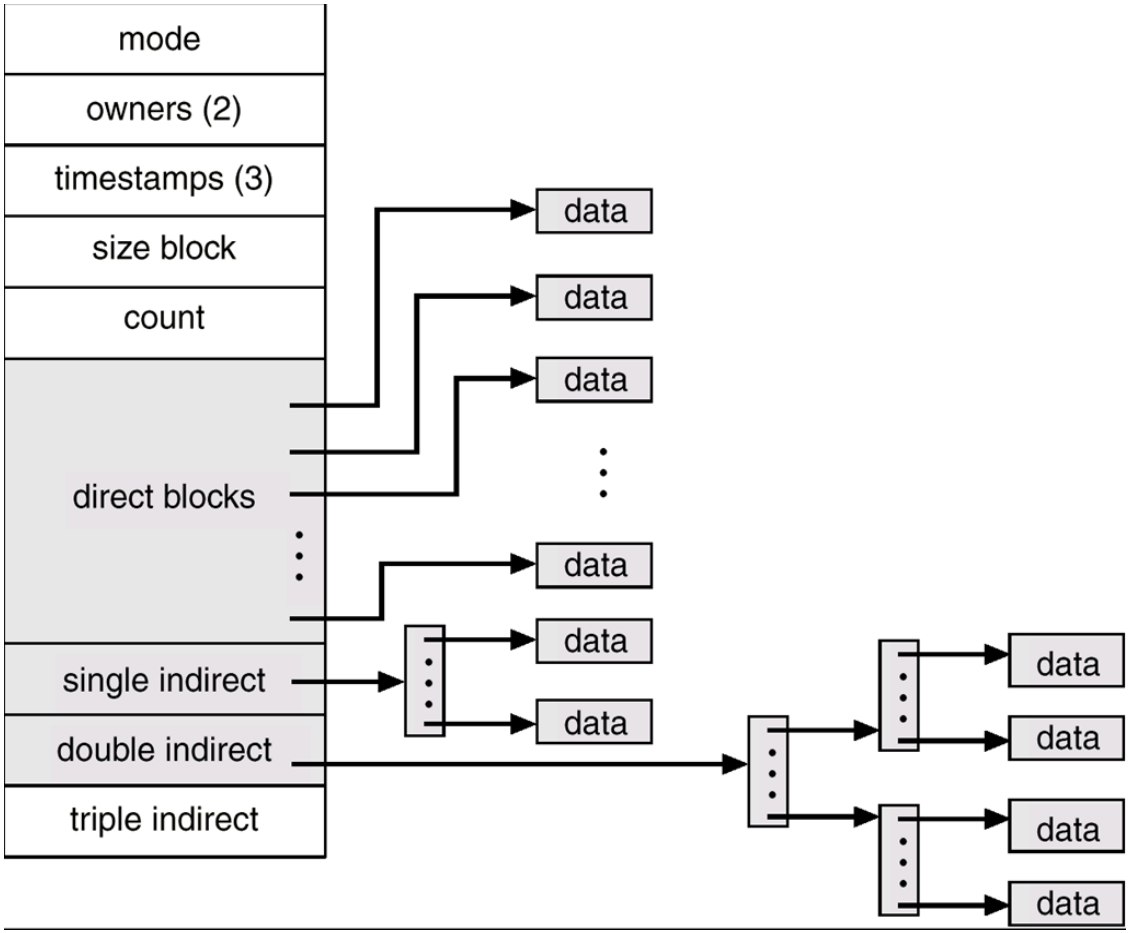
Indexed Allocation

- Bring all the pointers together into the **index block**.
- The **directory entry** contains the address of the **index block**.
- It keeps the advantages of the Linked Allocation (no external fragmentation, flexible size-declaration).
- It supports efficient **direct access**.
- It suffers from wasted space.
- How large should the index block be?
- What happens if the pointers do not fit in one block?

Indexed Allocation

- Combined scheme
  - Used in UFS
  - The directory entry has a pointer to the file's inode.
  - inode = FSB that saves additional 15 pointers.
    - 12 pointers point to direct blocks.
    - 1 pointer points to single indirect block.
    - 1 pointer points to double indirect block.
    - 1 pointer points to triple indirect block.

The Unix inode



The Unix inode

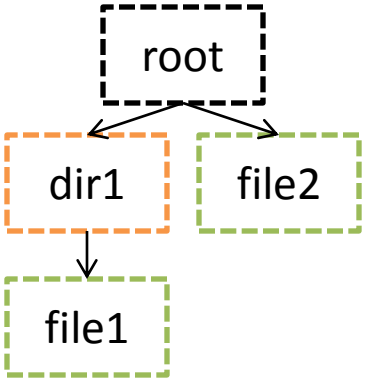
- If blocks are 4K and block references are 4 bytes ...
  - First 48K reachable from the inode.
  - Next 4MB available from single-indirect .
  - Next 4GB available from double-indirect.
  - Next 4TB available through the triple-indirect block.
- Any block (in 4TB space) can be found at 4 disk accesses.

Log-structured File System (LFS)

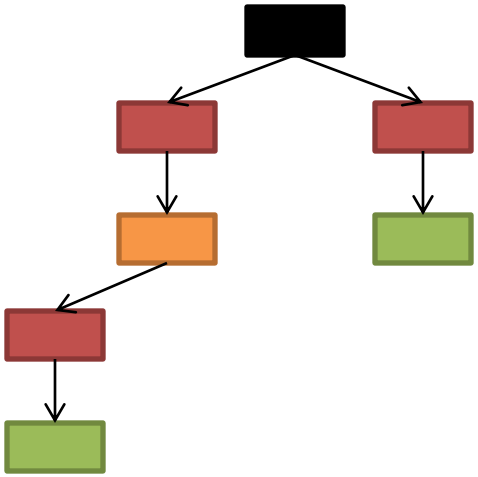
- There is a gap between CPU speeds and disk access times.
- Assumption: Files are cached in main memory.
- So, disk traffic will be dominated by writes.
- Write all new information to disk in a sequential structured called log.
- This approach increases performance dramatically by eliminating almost all seeks.
- Permanent storage. No other structure on disk.
- For a log-structured file system to operate efficiently, it must ensure that there are always large extents of free space available for writing new data. This is the most difficult challenge.
- Outperforms UFS by an order of magnitude for small writes.
- Matches or exceeds UFS performance for reads and large writes.

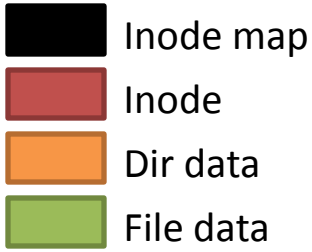
LFS

Logical file system

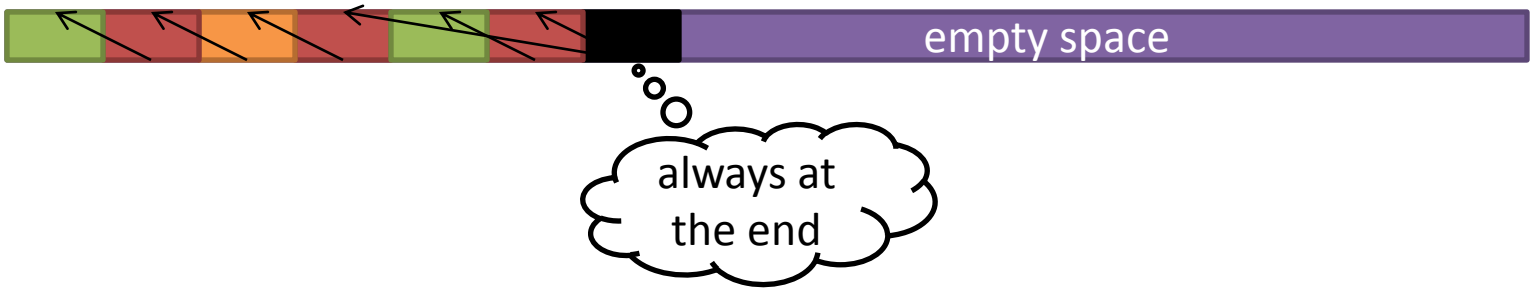


Block-level representation of file system





Disk layout



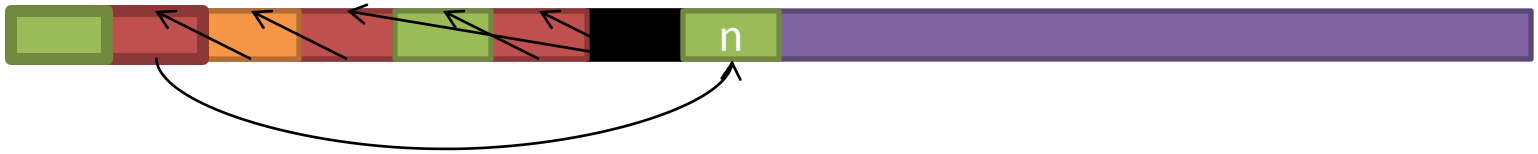
LFS: write

- Remember: even if a few bytes of a file are written/modified, the entire block that includes these bytes should be written/modified.
- LFS treats a block that needs to be modified or written in the same way.
  - This block is written at the end of the log structure.

Assume that file data  is modified and becomes 

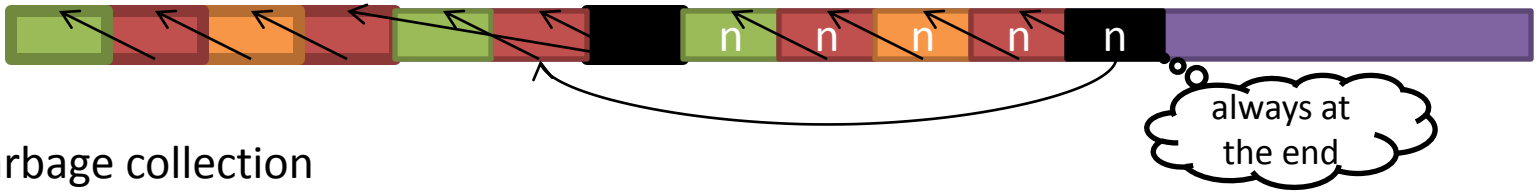
.

The inode of that file should change to point to the new data block.

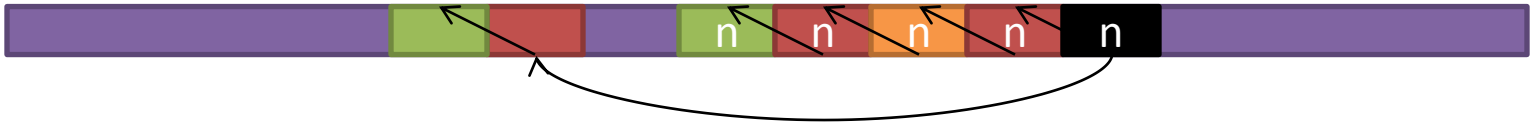


The inode of that file is modified…

In turn, the directory, its inode, and the inode map are modified…



Garbage collection



Cleaning (in segments)

Today

- File allocation
- Unix file system
- Log-structured file system

Coming up…

- Next lecture: Networking – Introduction
- HW4 is due on Tuesday
- Exam on Thursday
- Office hours moved from today to Tuesday.