MCA 32
OPERATING SYSTEMS
UNIT : #VI
FILE SYSTEM IMPLEMENTATION

Khalid Raza
DEPARTMENT OF COMPUTER SCIENCE
Jamia Millia Islamia (Central University), New Delhi, India.
✉ kraza@jmi.ac.in | Web site: www.kraza.in

CONTENTS

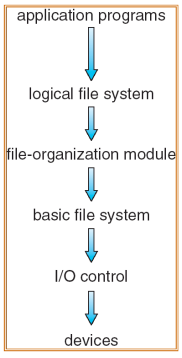
OBJECTIVES

- To describe the details of implementing local file systems and directory structures
- To describe the implementation of remote file systems
- To discuss block allocation and free-block algorithms and trade-offs

FILE-SYSTEM STRUCTURE

- File structure
  - Logical storage unit
  - Collection of related information
- File system resides on secondary storage (disks)
- File system organized into layers
- **File control block** – storage structure consisting of information about a file.

LAYERED FILE SYSTEM

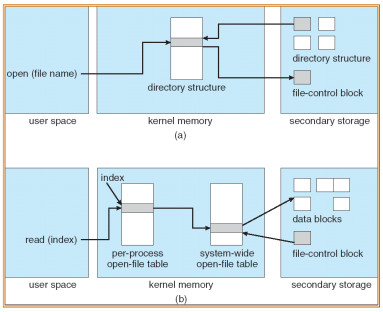


A TYPICAL FILE CONTROL BLOCK

file permissions
file dates (create, access, write)
file owner, group, ACL
file size
file data blocks or pointers to file data blocks

In-Memory File System Structures

- The following figure illustrates the necessary file system structures provided by the operating systems.



(a) refers to opening a file.

(b) refers to reading a file.

Virtual File Systems

- Virtual File Systems (VFS) provide an object-oriented way of implementing file systems.
- VFS allows the same system call interface (the API) to be used for different types of file systems.
- The API is to the VFS interface, rather than any specific type of file system.

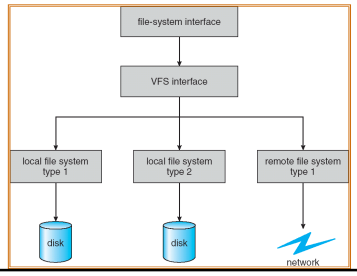


Fig. Schematic View of VFS

Directory Implementation

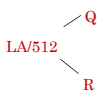
- **Linear list** of file names with pointer to the data blocks.
  - simple to program
  - time-consuming to execute
- **Hash Table** – linear list with hash data structure.
  - decreases directory search time
  - **collisions** – situations where two file names hash to the same location
  - fixed size

Allocation Methods

- An allocation method refers to how disk blocks are allocated for files:
- **Contiguous allocation**
- **Linked allocation**
- **Indexed allocation**

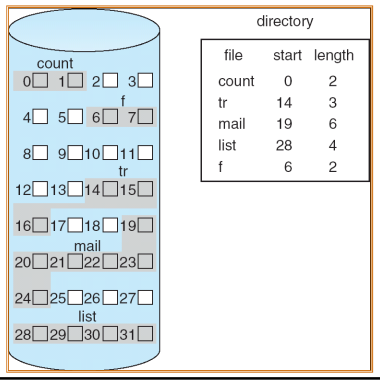
Contiguous Allocation

- Each file occupies a set of contiguous blocks on the disk
- Simple – only starting location (block #) and length (number of blocks) are required
- Random access
- Wasteful of space (dynamic storage-allocation problem)
- Files cannot grow
- Mapping from logical to physical



Block to be accessed = ! + starting address
Displacement into block = R

Contiguous Allocation of Disk Space



Linked Allocation

- Each file is a linked list of disk blocks: blocks may be scattered anywhere on the disk.
- Simple – need only starting address
- Free-space management system – no waste of space
- No random access
- Mapping
- File-allocation table (FAT) – disk-space allocation used by MS-DOS and OS/2.

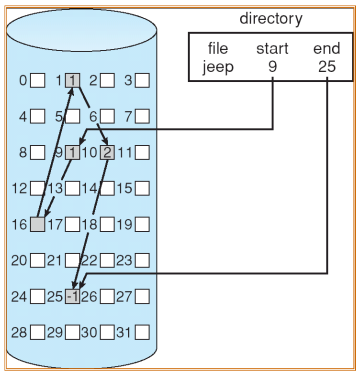
block = pointer

LA/511 → Q, R

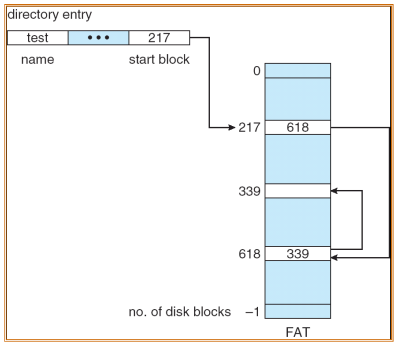
Block to be accessed is the Qth block in the linked chain of blocks representing the file.

Displacement into block = R + 1

…Linked Allocation

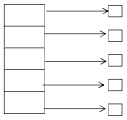


Linked Allocation: File Allocation Table

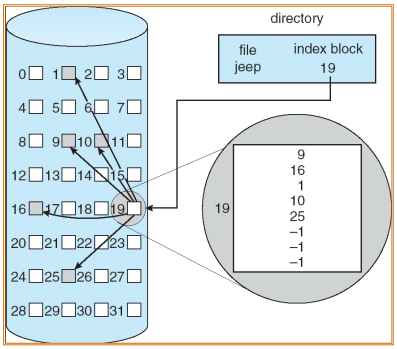


Indexed Allocation

- Brings all pointers together into the *index block*.
- Logical view.



Example of Indexed Allocation



…Indexed Allocation

- Need index table
- Random access
- Dynamic access without external fragmentation, but have overhead of index block.
- Mapping from logical to physical in a file of maximum size of 256K words and block size of 512 words. We need only 1 block for index table.

LA/512 → Q, R

Q = displacement into index table
R = displacement into block

Free-Space Management

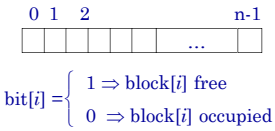
- Disk space is limited, we need to reuse the space from deleted files.
  - Write-once optical disks only allow one write to any given sector, and thus such reuse is not physically possible.)
- To keep track of free disk space, system maintains a free-space list.
- To create a file, we search the free-space list for the required amount of space and allocate that space to the new file.
- Allocated space is then removed from the free-space list.
- Similarly, when a file is deleted, its disk space is added to the free-space list.

…Free-Space Management

- The disk free-space management techniques are:
  - Bit-vector
  - Linked List
  - Grouping
  - Counting
  - Space Maps

…Free-Space Management (Bit Vector)

- Frequently used technique.
- Each block is represented by 1 bit.
- If the block is free, the bit is 1; if the block is allocated, the bit is 0.
- E.g. bit vector (n blocks)



$$bit[i] = \begin{cases} 1 \Rightarrow block[i] \text{ free} \\ 0 \Rightarrow block[i] \text{ occupied} \end{cases}$$

…Free-Space Management (Bit Vector)

- E.g., consider a disk where blocks **2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26, and 27** are free and the rest of the blocks are allocated. The free-space bit map would be

  0011110011111100011000000011100000 …
- ***Advantage:*** simplicity and its efficiency in finding the first free block or n consecutive free blocks.
- One technique for finding the first free block is to sequentially check each word in the bit map.

…Free-Space Management (Bit Vector)

- Bit vectors are inefficient unless kept in main memory (written to disk occasionally for recovery needs).
- Bit map requires extra space
  - Example:

    block size = 2^{12} bytes (4 KB)

    disk size = 2^{30} bytes (1 GB)

    $n = 2^{30}/2^{12} = 2^{18}$ bits (or 32 KB)
- Easy to get contiguous files

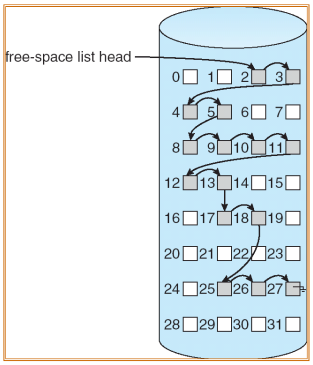
…Free-Space Management (Bit Vector)

- *Questions:* 1-TB disk with 4-KB blocks requires how much memory to store its bit map.
- ***Answer:*** n = $2^{40}/2^{12} = 2^{28}$ bits = 32 MB
- Given that disk size constantly increases, the problem with bit vectors will continue to escalate.
- A 1-PB file system would take a 32-GB bitmap just to manage its free space.

Free-Space Management (Linked List)

- Another approach is to link together all the free disk blocks, keeping a pointer to the first free block.
- This first block contains a pointer to the next free disk block, and so on.
- E.g., blocks 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26, and 27 were free and the rest of the blocks were allocated.

…Free-Space Management (Linked List)



…Free-Space Management (Linked List)

- Not efficient: to traverse the list, we must read each block, which requires I/0 time.
- Fortunately, *traversing the free list is not a frequent action*.
- Usually, OS simply needs a first free block so that it can allocate.
- The FAT method incorporates free-block accounting into the allocation data structure.
- No separate method is needed.

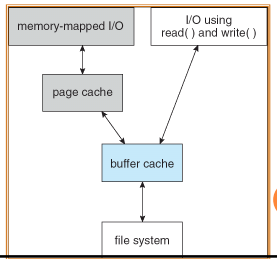
Efficiency and Performance

- Efficiency dependent on:
  - disk allocation and directory algorithms
  - types of data kept in file's directory entry
- Performance
  - disk cache – separate section of main memory for frequently used blocks.
  - free-behind and read-ahead – techniques to optimize sequential access
  - improve PC performance by dedicating section of memory as virtual disk, or RAM disk.

Page Cache

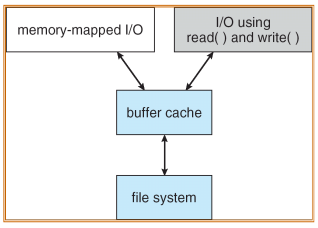
- A **page cache** caches pages rather than disk blocks using virtual memory techniques.
- Memory-mapped I/O uses a page cache.
- Routine I/O through the file system uses the buffer (disk) cache.
- Figure:

Some systems maintain a separate section of main memory for a buffer cache where blocks are kept under the assumption that will be used again shortly.



Unified Buffer Cache

- A unified buffer cache uses the same page cache to cache both memory-mapped pages and ordinary file system I/O.



Recovery

- Files and directories are kept both in MM and disk to recovery from system failure.
- A system crash can cause inconsistencies among on-disk file-system data structures, such as :
  - directory structures,
  - free-block pointers, and
  - free FCB pointers.
- File systems have varying methods to deal with these issue:
- Consistency checking
- Log-Structured File Systems
- Other Solutions
- **Backup and Restore**
  - Consistency checking: compares data in directory structure with data blocks on disk, and tries to fix inconsistencies.
  - Use system programs to **back up** data from disk to another

…Recovery (Consistency Checking)

- Consistency checking: compares data in directory structure with data blocks on disk, and tries to fix inconsistencies.
- Use system programs to **back up** data from disk to another storage device.
- Recover lost file or disk by **restoring** data from backup.

Log Structured File Systems

- **Log structured** file systems record each update to the file system as a **transaction.**
- All transactions are written to a **log**
  - A transaction is considered **committed** once it is written to the log.
  - However, the file system may not yet be updated
- If the file system crashes, all remaining transactions in the log must still be performed.

Network File System (NFS)

- Developed by Sun Microsystem.
- An implementation and a specification of a software system for accessing remote files across LANs (or WANs)
- The implementation is part of the Solaris and SunOS operating systems running on Sun workstations using an unreliable datagram protocol (UDP/IP protocol and Ethernet

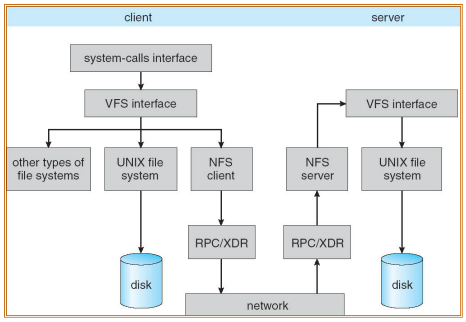
…Network File System (NFS)

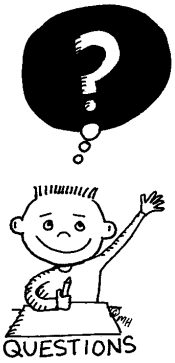
- Interconnected workstations viewed as a set of independent machines with independent file systems.
- Allows sharing among these file systems in a transparent manner:
  - A remote directory is mounted over a local file system directory
  - Specification of the remote directory for the mount operation is nontransparent;
  - Subject to access-rights accreditation,
  - Potentially any file system

…Network File System (NFS)

- NFS is designed to operate in a heterogeneous environment of different machines, OS, and network architectures;
- NFS specifications independent of these media
- This independence is achieved through the use of RPC primitives built on top of an External Data Representation (XDR) protocol used between two implementation-independent interfaces
- The NFS specification distinguishes between the services provided by a mount mechanism and the actual remote-file-access services.

…Network File System (NFS)





Conclusions

- File-System Structure
- File-System Implementation
- Directory Implementation
- Allocation Methods
- Free-Space Management
- Efficiency and Performance
- Recovery
- Log-Structured File Systems
- NFS

Self Assessment Questions

1. In what situations would using memory as a RAM disk be more useful than using it as a disk cache?
2. Some file systems allow disk storage *to* be allocated at different levels of granularity. For instance, a file system could allocate 4 KB of disk space as a single 4-KB block or as eight 512-byte blocks. How could we take advantage of this flexibility to improve performance? What modifications would have to be made to the free-space management scheme in order to support this feature?
3. Consider a file system that uses inodes to represent files. Disk blocks are 8 KB in size, and a pointer to a disk block requires 4 bytes. This file system has 12 direct disk blocks, as well as single, double, and triple indirect disk blocks. What is the maximum size of a file that can be stored in this file system?
4. Explain why logging metadata updates ensures recovery of a file system after a file-system crash.

References

- Abraham Silberschatz, Peter B. Galvin, Operating system concepts, 8th Ed., Wiley India.
- Andrew S. Tanenbaum, "Modern Operating System", PHI.
- Other freely available web resources.