File Concept

- **Files are logical units of information created by processes**.
- A disk usually contains thousands or even millions of files, each one independent of others.
- The part of the OS dealing with files is known as **file system**.
- It is **the most important concepts relating to the OS**.
- Processes can read existing files and create new ones if need be.
- Information stored in the file must be persistent, i.e., not be affected by process creation and termination.
- A file should be disappeared only when its owner removes it.
- Reading and writing are the two most common operations for files.

File Concept

Need/Purpose of using Files

- It is possible to store a large amount of information.
- Multiple processes must be able to access the information at once.
- Without a files
  - While a process is running, it can store a limited amount of information within its own address space.
  - When the process terminates, the information is lost.
  - The information stored in a process's address space will be used by that process only.

File Concept

File Naming

- A file stores information on the disk and read it back later.
- When a process creates a file, it gives the file a name.
- When the process terminates, the file continues to exist and can be accessed by other processes using its name.
- The exact rues for naming a file varies from system to system. Some supports a string of one to eight letters, many file systems support up to 255 characters. Digits and special characters are permitted also. Sometimes uppercase and lowercase letters are distinguished.
- Older version of Windows use FAT-16 (File Allocation Table). Newer versions use advanced file system NTFS (New Technology File System). How a file is named is inherited from these file systems.
- Two part file names: file name.extension.
- Figure 4.1: Typical File Extensions
- In contrast to Unix, **meanings are assigned to the extensions in Windows**. When a user double clicks on a file name, the program assigned to its file extension is launched with the file as parameter.

File Concept

File Structure

- Files can be structured in several ways. Three most common ways are:

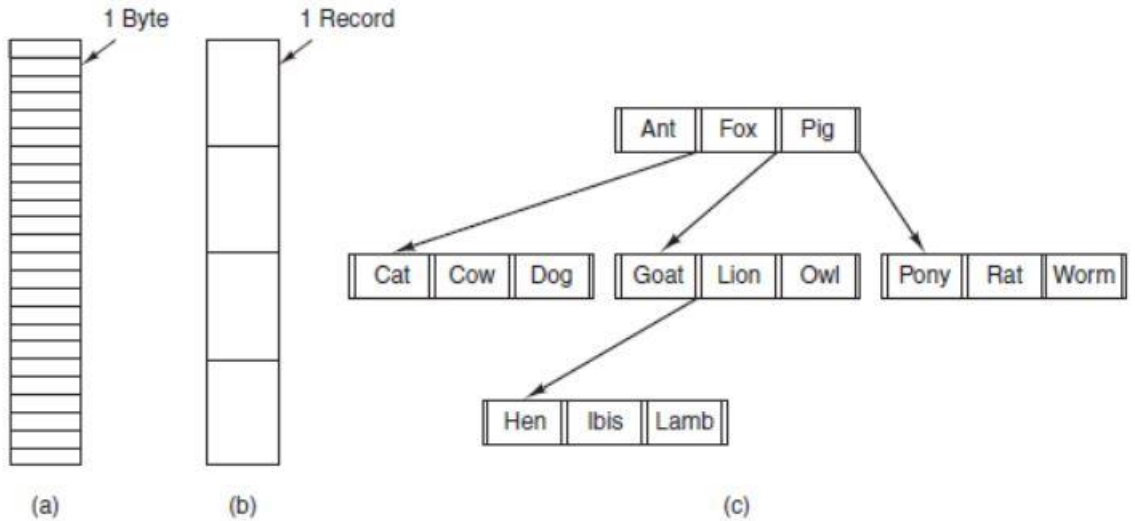


Figure 4-2. Three kinds of files. (a) Byte sequence. (b) Record sequence. (c) Tree.

File Concept

- The OS does not care about what is in a file. **It considers bytes only**.
- **A file is a sequence of fixed length of records (Figure 4.2 (b))**, each with some internal structure. Here, the read operation returns one record and the write operation writes or appends one record.
- **A file consists of a tree of records (Figure 4.2 (c))**, not necessarily all the same length, each containing a key field in a fixed position in the record. The tree is sorted on the key field, to allow rapid searching for a particular key. Basically used in some large mainframe computers for commercial data processing.

File Concept

File Types

- Different OSs support several file types.
- UNIX and Windows have **regular files** and **directories**.
- **Regular files** are those that contain user information.
- **Directories** are system files for maintaining the structure of the file system.
- Regular files are generally either ASCII files or binary files.
- **ASCII files** consist of lines of text. The main advantage of this type of file is that a file can be displayed and printed as it is. It can be edited with a text editor.
- **Binary files** are not ASCII files. For example, en executable file (Figure 4.3). A binary file often includes some type of header that indicates the type of file. The header might include a few human-readable characters, but a binary file as a whole requires specific software or hardware to read the file and take action. A binary file is computer-readable but not human-readable.
- **Special files** are one type of files that may be stored in a file system. A special file is sometimes also called a device file. The purpose of a special file is to expose the device as a file in the file system. A special file provides a universal interface for hardware devices.

File Concept

- **Character special files** are provide access to an input/output device. They are used to model serial I/O devices such as terminals, printers, and networks.
- **Block special files** act as direct interface to a block device. A block device is any device which performs data I/O in units of blocks.

File Concept

File Access

- **Sequential Access**: A process can read all bytes or records in a file in order, starting from the beginning, but could not skip around and read them out of order.
- **Random Access**: Out of order access. For example, database.

File Attributes

✓File name
✓ File size
✓Date and time of the file was last modified
✓Password
✓Owner
✓……
See Figure 4.4

File Concept

File Operations

- **Create**: A file is created with no data. This is an announcement.
- **Delete**: The file is deleted, when it is no longer used/needed, to free up the disk space.
- **Open**: Before using a file, a process must open it. The purpose of the open call is to allow the system to fetch the attributes and list of disk addresses into main memory for rapid access on later calls.
- **Close**: When all the accesses are finished, the attributes and disk addresses are no longer needed, so the file should be closed to free up internal table space.
- **Read**: Data are read from file. Usually, the bytes come from the current position. The caller must specify how many data are needed and must also provide a buffer to put them in.
- **Write**: Data are written to the file again, usually at the current position. If the current position is the end of the file, the file's size increases. If the current position is in the middle of the file, existing data are overwritten and lost forever.
- **Append**: This call is a restricted form of write. It can add data only to the end of the file.

File Concept

- **Seek:** For random-access files, a method is needed to specify from where to take the data. One common approach is a system call, seek, that repositions the file pointer to a specific place in the file. After this call has completed, data can be read from, or written to, that position.
- **Get attributes:** Processes often need to read file attributes to do their work. For example, the UNIX make program is commonly used to manage software development projects consisting of many source files. When make is called, it examines the modification times of all the source and object files and arranges for the minimum number of compilations required to bring everything up to date. To do its job, it must look at the attributes, namely, the modification times.
- **Set attributes:** Some of the attributes are user settable and can be changed after the file has been created. This system call makes that possible. The protection-mode information is an obvious example.
- **Rename:** It frequently happens that a user needs to change the name of an existing file. This system call makes that possible.

File Concept

Directories

- File systems have directories or folders to keep track of files.
- **Directories themselves are files**.
- **Single Level Directory System**:
  - The simplest form of directory.
  - It has one directory containing all the files, called **root directory**.
  - Early PCs had this directory system.
  - The advantage is to locate files very quickly, because there is only one place to look.
  - The first super computer had this system.
  - Still used in some devices such as digital cameras and portable music players.
  - Adequate for dedicated systems.

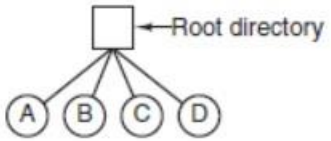


Figure 4-6. A single-level directory system containing four files.

File Concept

- **Hierarchical Directory System**:
  - Modern computers with thousands of files, it would be impossible to find anything if all files were in a single directory.
  - A hierarchy or tree of directories are appropriate for this purpose.
  - Related files are grouped together here.
  - The ability for user to create an arbitrary number of subdirectories provides a powerful structuring tool for users to organize their work.
- **Path Name**
  - When files are organized in directory system, it is needed to specify the path name.
  - **Absolute Path Name**: Consisting of the path from the root directory to the file.
    - Always starts at the root directory and is unique.
    - For example: \programming\example\test.

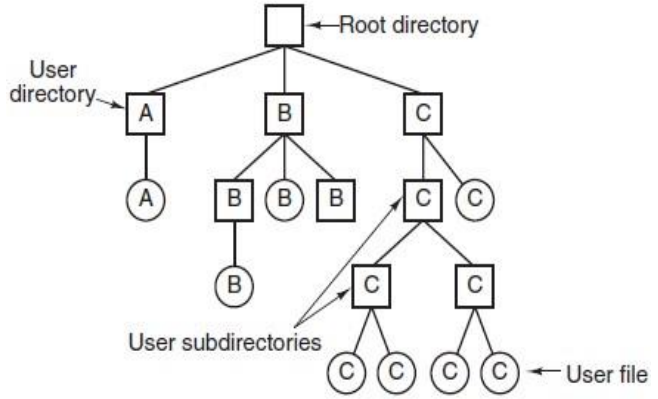


Figure 4-7. A hierarchical directory system.

File Concept

- **Relative Path Name**: This is used in conjunction with current directory or working directory.
  - A user can designate a directory as a working directory.
  - All the path names not beginning at the root directory are taken relative to the working directory.
  - Each process has it own working directory.

- Most of the OSs that support hierarchical directory system have two special entries in every directory; “.” and “..”. Dot refers to the current directory and dotdot refers to its parent.

File Concept

- **Directory Operation:** System calls to for managing directories in UNIX are
  - Create: A directory is created. It is empty.
  - Delete: A directory is created. Only an empty directory can be deleted.
  - Opendir: Directories can be read.
  - Closedir: Closes a directory.
  - Rename: A directory is renamed.
  - Link: A file can be appeared in more than one directory.

File System Layout

- File systems are stored on disks. Most disks can be divided up into one or more partitions. Each partition has independent file systems.
- Sector 0 of the disk is called the **MBR-Master Boot Record**, which is used to boot the computer. The end of the MBR contains the partition table which gives the starting and ending address of each partition.
- When the computer is booted, the BIOS reads in and executes the MBR.
- At first the MBR locates the active partition, reads in the first block of it, which is called the **boot block,** and executes it.
- The program in the boot block loads the OS contained in the partition.

File Concept

- Superblock contains all the key parameters about the file system; a number to identify the file system type, the number of blocks in the file system, and other administrative information.

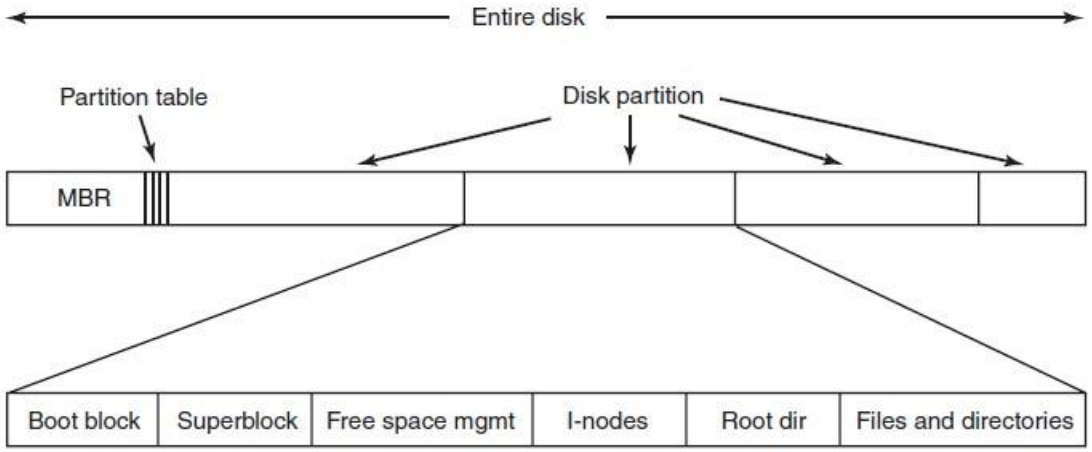


Figure 4-9. A possible file-system layout.

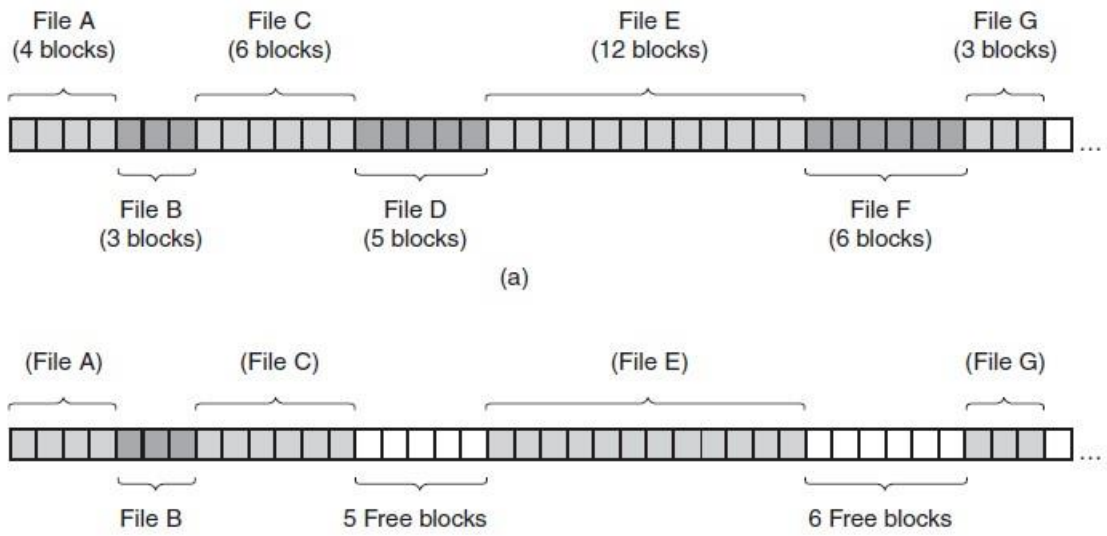
File Concept

Implementing Files

- **Contiguous Allocation**
  - The simplest allocation scheme.
  - It stores each file as a contiguous run of disk blocks. For example, a 100 KB file would be allocated 100 consecutive blocks.
  - Two significant advantages are: i. keeping track of where a files blocks are, is reduced to two numbers; the disk address of the first block and the number of blocks in the file. ii. The performance of read operation is excellent because the entire file can be read from the disk in a single operation. Only one seek is needed.
  - Drawback: Over the time, the disk becomes fragmented. As a result, the disk ultimately consists of files and holes.

File Concept

- This system is still used in CD-ROMs. Because, all the file sizes are known in advance and will never change during the subsequent use of the file system.



File Concept

- **Linked List Allocation**
  - This system keeps each file as a linked list of disk blocks.
  - The first word of each block is used as a pointer to the next one. The rest of the blocjk is for data.
  - Every disk block can be used in this method. No space is lost to disk fragmentation.
  - Drawback: To get to block n, the OS has to start at the be and read the n- blocks prior to it, one at a time. So many read operations will be slow.
  - Drawback: The amount of data storage in a block is no longer a power of two, because the pointer takes up a few bytes.

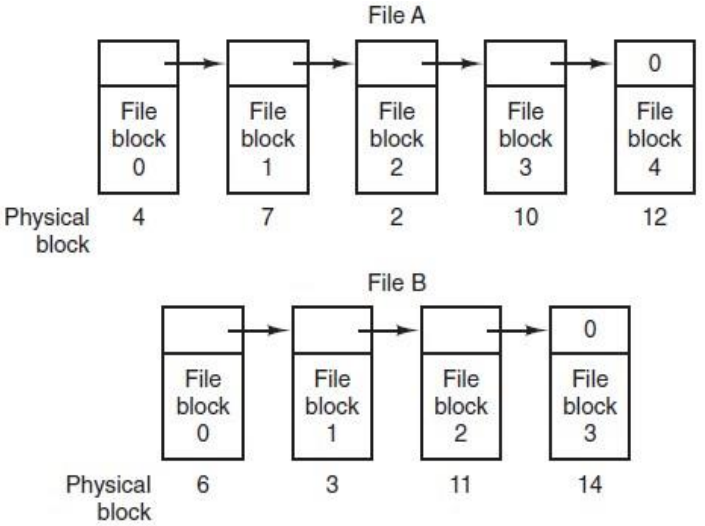


Figure 4-11. Storing a file as a linked list of disk blocks.

File Concept

- **Linked List File Allocation using a Table in Memory**
  - Both drawbacks of the linked list file allocation can be eliminated by taking the pointer word from each disk block and putting it in a table in memory.
  - Such a table in the main memory is called a FAT-File Allocation Table.
  - The entire chain is in the memory, so it can be followed without making any disk reference.
  - The disadvantage is that the entire table must be in the memory all the time to make it work.
  - With a 1 TB disk and 1 KB block size, the table needs one billion entries.

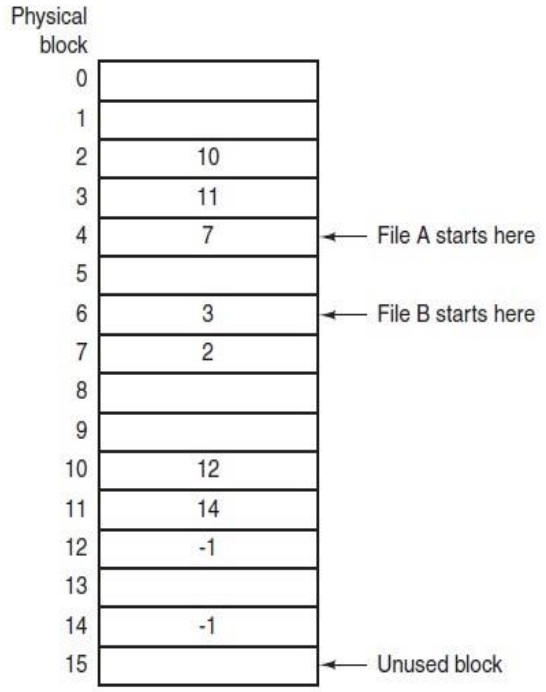


Figure 4-12. Linked-list allocation using a file-allocation table in main memory.

File Concept

- **I-nodes**
  - Here each file is associated with a data structure called an I-node- Index-node which lists the attributes and disk addresses of the files blocks.
  - The advantage over linked list method is that I-node needs to be in memory only when the corresponding file is open.
  - If node occupies n bytes and a maximum of k files may be open at once, the total memory occupied by the array holding the I-nodes for the open files is only kn bytes.

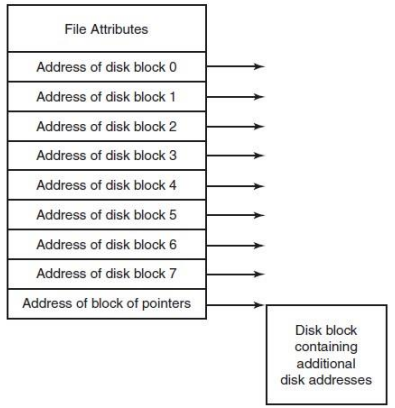


Figure 4-13. An example i-node.