FILE OPERATION AND STRUCTURE

File Operations

Files exist to store information and allow it to be retrieved later. Different systems provide different operations to allow storage and retrieval. Below is a discussion of the most common system calls relating to files.

1. **Create.** The file is created with no data. The purpose of the call is to announce that the file is coming and to set some of the attributes.

2. **Delete**. When the file is no longer needed, it has to be deleted to free up disk space. A system call for this purpose is always provided.

3. **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.

4. **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 some internal table space. Many systems encourage this by imposing a maximum number of open files on processes. A disk is written in blocks, and closing a file forces writing of the file's last block, even though that block may not be entirely full yet.

5. **Read.** Data are read from file. Usually, the bytes come from the current position. The caller must specify how much data are needed and must also provide a buffer to put them in.

6. **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.

7. **Append.** This call is a restricted form of write. It can only add data to the end of the file. Systems that provide a minimal set of system calls do not generally have append, but many systems provide multiple ways of doing the same thing, and these systems sometimes have append.

8. **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.

9. **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.

10. **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. Most of the flags also fall in this category.

11. **Rename.** It frequently happens that a user needs to change the name of an existing file. This system call makes that possible. It is not always strictly necessary, because the file can usually be copied to a new file with the new name, and the old file then deleted.

12. **Lock.** Locking a file or a part of a file prevents multiple simultaneous access by different process. For an airline reservation system, for instance, locking the database while making a reservation prevents reservation of a seat for two different travelers.

File Structure:
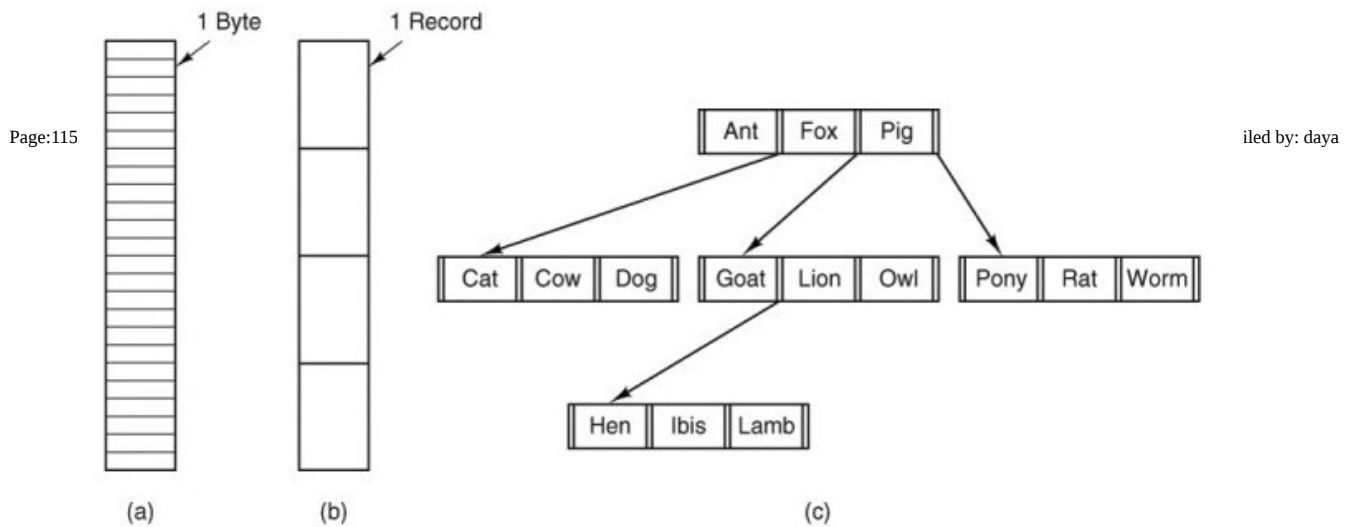
Three kinds of files. (a) Byte sequence. (b) Record sequence. (c) Tree.

a. Byte Sequence:
The file in Fig. (a) is just an unstructured sequence of bytes. In effect, the operating system does not know or care what is in the file. All it sees are bytes. Any meaning must be imposed by user-level programs. Both UNIX and Windows 98 use this approach.

b. Record Sequence:
In this model, a file is a sequence of fixed-length records, each with some internal structure. Central to the idea of a file being a sequence of records is the idea that the read operation returns one record and the write operation overwrites or appends one record. As a historical note, when the 80-column punched card was king many (mainframe) operating systems based their file systems on files consisting of 80-character records, in effect, card images

c. Record Sequence:

In this organization, a file consists of a tree of records, 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.

Files Organization and Access Mechanism: