CSE 3201

OPERATING SYSTEMS

File

- A file is an abstraction mechanism.
- It provides a way to store information on the disk and read it back later.
- This must be done in such a way as to shield the user from the details of how and where the information is stored, and how the disks actually work.
- Some popular file systems are – FAT-16, FAT-32, NTFS, ReFS, exFAT.

File Attributes

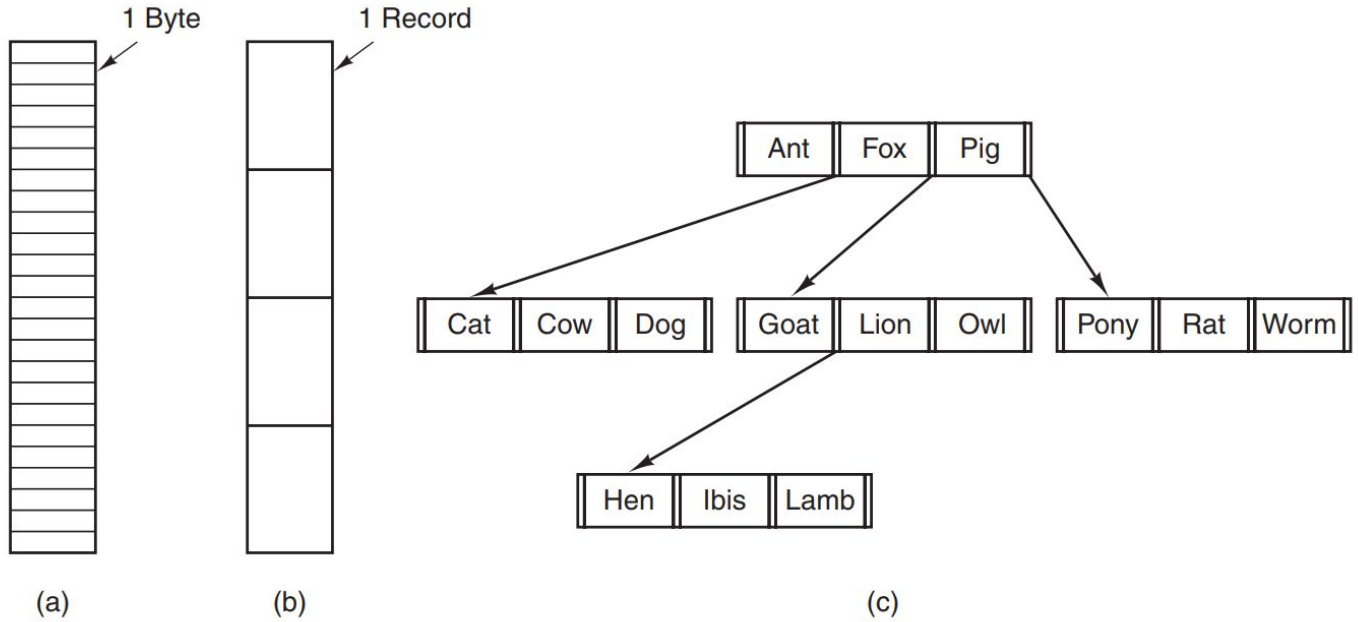
- **Name:** The symbolic file name is the only information kept in humanreadable form.
- **Identifier:** This unique tag, usually a number, identifies the file within the file system; it is the non-human-readable name for the file.
- **Type:** This information is needed for systems that support different types of files.
- **Location:** This information is a pointer to a device and to the location of the file on that device.
- **Size:** The current size of the file (in bytes, words, or blocks) and possibly the maximum allowed size are included in this attribute.
- **Protection:** Access-control information determines who can do reading, writing, executing, and so on.

File Naming

- 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.
- Usually all present systems support 1-8 letters file name. Many file systems support names as long as 255 characters.
- Some system are case sensitive and some are not. UNIX is case sensitive, MS-DOS is not case sensitive. Filename, FileName, FILENAME are 3 different files in UNIX system, whereas they are same in MS-DOS system.
- Many operating systems support two-part file names, with the two parts separated by a period, as in *prog.c*. The part following the period is called the **file extension** and usually indicates something about the file.

File Structure

Three kind of structure is present in file system-



(a) Byte sequence. (b) Record sequence. (c) Tree

File Structure

- Byte Sequence
  A process (user) can write anything as byte data. All versions of UNIX (including Linux and OS X) and Windows use this file model.

- 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.

- Tree
  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. It is used on some large mainframe computers for commercial data processing.

File Operations

- **Creating a file**
  1. Check for available space.
  2. Create an entry in directory.

- **Writing a file**
  To write a file, we make a system call specifying both the name of the file and the information to be written to the file. Given the name of the file, the system searches the directory to find the file's location. The system must keep a write pointer to the location in the file where the next write is to take place. The write pointer must be updated whenever a write occurs.

- **Reading a file**
  To read from a file, we use a system call that specifies the name of the file and where (in memory) the next block of the file should be put. Again, the directory is searched for the associated entry, and the system needs to keep a read pointer to the location in the file where the next read is to take place.

- Both the read and write operations use same pointer which is called **current-file-position pointer**, saving space and reducing system complexity.

File Operation

- **Repositioning within a file**
  The directory is searched for the appropriate entry, and the current-file-position pointer is repositioned to a given value. This is also called file seek.

- **Deleting a file**
  To delete a file, we search the directory for the named file. Having found the associated directory entry, we release all file space, so that it can be reused by other files, and erase the directory entry.

- **Truncating a file**
  The user may want to erase the contents of a file but keep its attributes. Rather than forcing the user to delete the file and then recreate it, this function allows all attributes to remain unchanged—except for file length—but lets the file be reset to length zero and its file space released.

Access Method

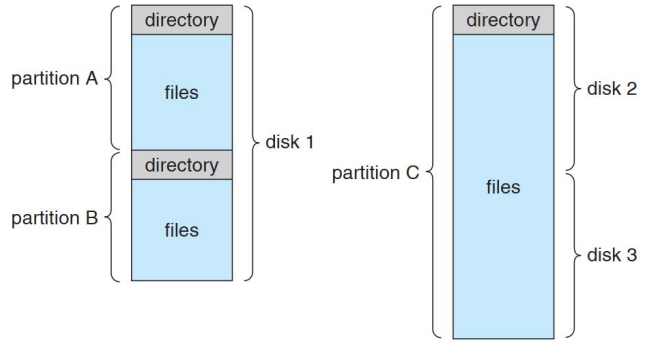
- Sequential Access
  - ✔ Reads and writes make up the bulk of the operations on a file.
  - ✔ A read operation—read next—reads the next portion of the file and automatically advances a file pointer, which tracks the I/O location.
  - ✔ Similarly, the write operation—write next—appends to the end of the file and advances to the end of the newly written material (the new end of file).
- Direct Access
  - ✔ A file is made up of fixed length logical records that allow programs to read and write records rapidly in no particular order. The direct-access method is based on a disk model of a file, since disks allow random access to any file block.
  - ✔ For the direct-access method, the file operations must be modified to include the block number as a parameter. Thus, we have read n, where n is the block number, rather than read next, and write n rather than write next. An alternative approach is to retain read next and write next, as with sequential access, and to add an operation position file to n, where n is the block number. Then, to effect a read n, we would position to n and then read next.

Access Method

sequential access	implementation for direct access
reset	*cp* = 0;
read next	*read cp*; *cp* = *cp* + 1;
write next	*write cp*; *cp* = *cp* + 1;

Directory & Disk Structure

- Files are stored on random-access storage devices, including hard disks, optical disks, and solid state (memory-based) disks
- A storage device can be used in its entirety for a file system. It can also be subdivided for finer-grained control.
- Any entity containing a file system is generally known as a volume.
- The volume may be a subset of a device, a whole device, or multiple devices linked together into a RAID set. Each volume can be thought of as a virtual disk.



File system organization

Storage Structure

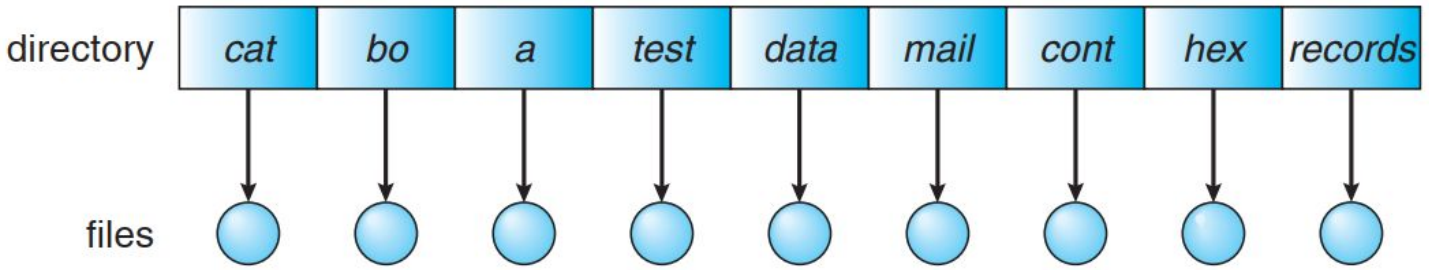
- **tmpfs**—a “temporary” file system that is created in volatile main memory and has its contents erased if the system reboots or crashes
- **objfs**—a “virtual” file system (essentially an interface to the kernel that looks like a file system) that gives debuggers access to kernel symbols
- **ctfs**— a virtual file system that maintains “contract” information to manage which processes start when the system boots and must continue to run during operation
- **lofs**—a “loop-back” file system that allows one file system to be accessed in place of another one
- **procfs**—a virtual file system that presents information on all processes as a file system
- **ufs, zfs**—general-purpose file systems

Directory Overview

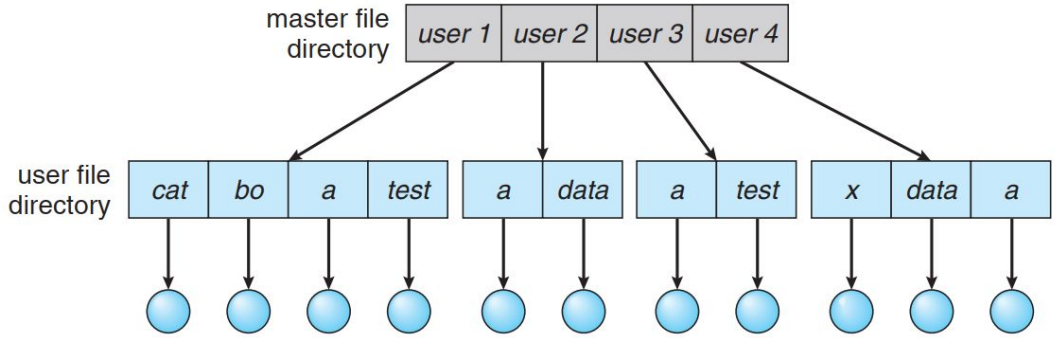
The Operations that must be performed in a directory -

- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file
- Traverse the file system

Directory Structure

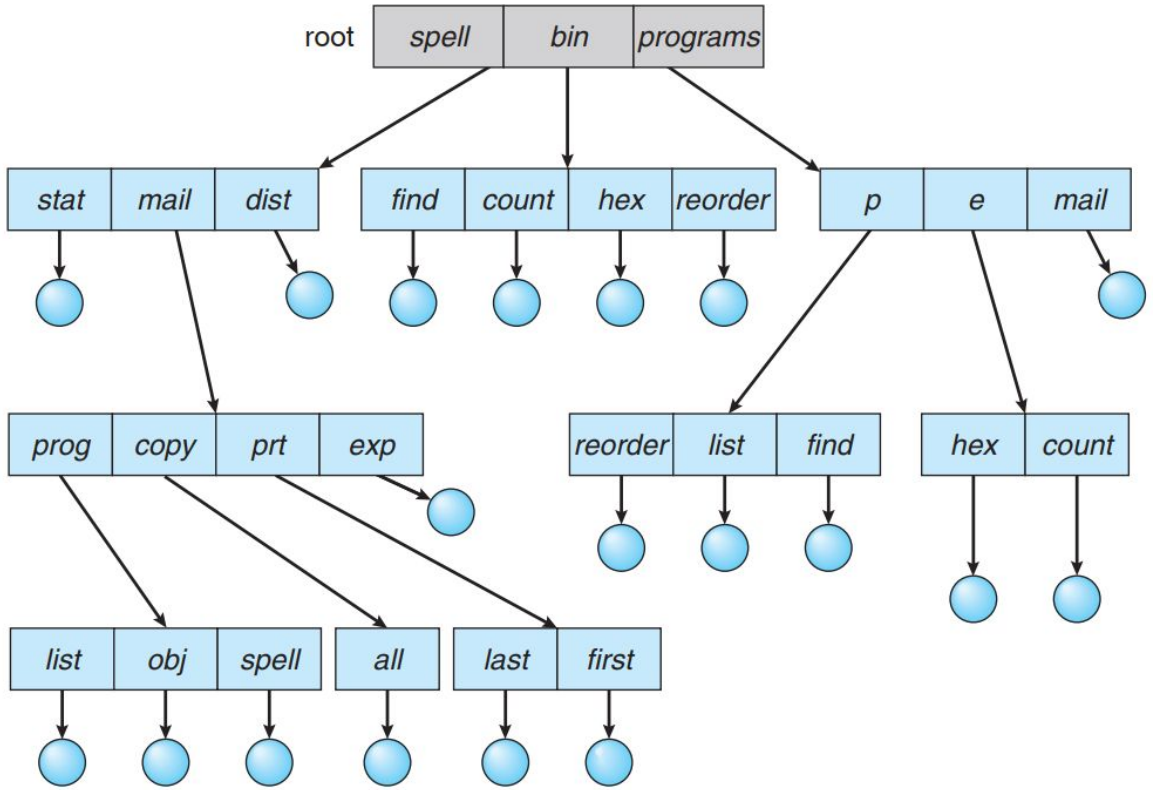


Single Level Directory



Two Level Directory

Directory Structure



Tree Structured Directory

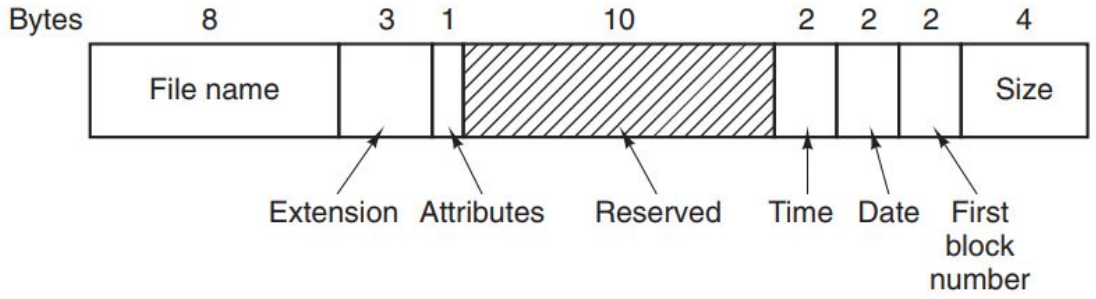
Directory Structure

- In normal use, each process has a current directory.
- When reference is made to a file, the current directory is searched.
- If a file is needed that is not in the current directory, then the user usually must either specify a path name or change the current directory to be the directory holding that file.
- To change directories, a system call is provided that takes a directory name as a parameter and uses it to redefine the current directory
- Path names can be of two types: absolute and relative.
- An absolute path name begins at the root and follows a path down to the specified file, giving the directory names on the path.
- A relative path name defines a path from the current directory

MS-DOS File System

Opening of a file:

- To read a file, an MS-DOS program must first make an open system call to get a handle for it.
- The open system call specifies a path, which may be either absolute or relative to the current working directory.
- The path is looked up component by component until the final directory is located and read into memory. It is then searched for the file to be opened.
- MS-DOS directories are variable sized but they use a fixed-size 32-byte directory entry.



MS-DOS Directory Entry

MS-DOS File System

Block size	FAT-12	FAT-16	FAT-32
0.5 KB	2 MB		
1 KB	4 MB		
2 KB	8 MB	128 MB	
4 KB	16 MB	256 MB	1 TB
8 KB		512 MB	2 TB
16 KB		1024 MB	2 TB
32 KB		2048 MB	2 TB

Maximum partition size for different block sizes