File Systems

ECGR 6185 Spring 2006

Christina Warren
University of North Carolina at Charlotte

File system: intro

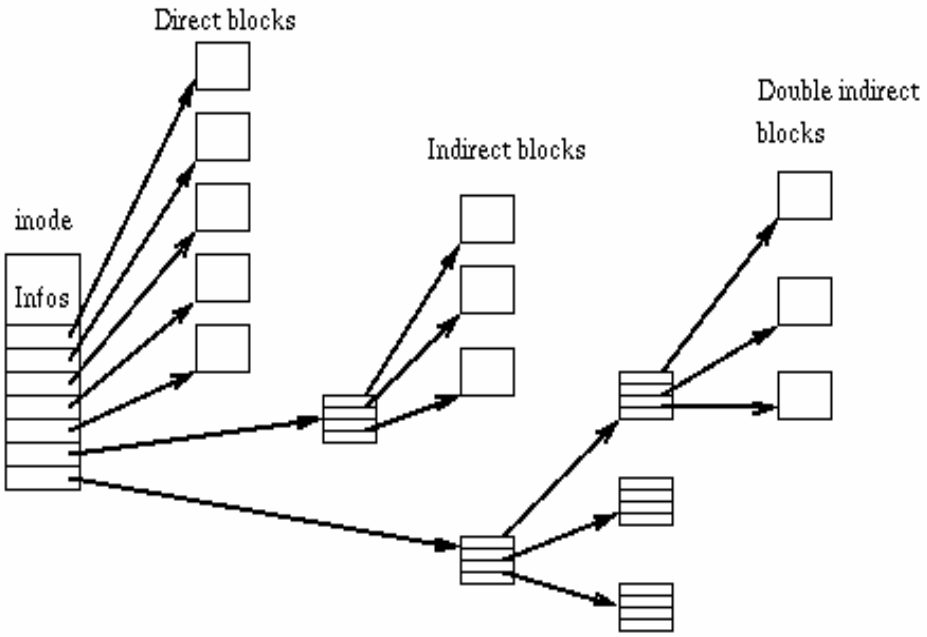
In computing, a file system

- is a method for storing and organizing computer files
- it makes it easy to find and access data
- File systems may use a storage device such as a hard disk or CD-ROM and it involves maintaining the physical location of the files, or the files may be virtual and exist only as an access method for virtual data or for data over a network (e.g. NFS).

More formally, a file system is a set of abstract data types that are implemented for

- storage
- hierarchical organization
- manipulation
- navigation
- access
- and retrieval of data.

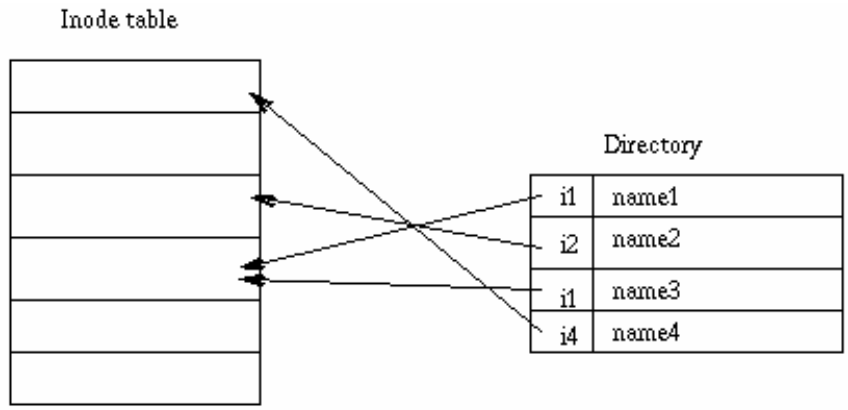
Inode



- Each file is represented by a structure, called an inode.
- Each inode contains the description of the file: file type, access rights, owners, timestamps, size, pointers to data blocks.
- The addresses of data blocks allocated to a file are stored in its inode.
- When a user requests an I/O operation on the file, the kernel code converts the current offset to a block number, uses this number as an index in the block addresses table and reads or writes the physical block.

Ref: 1

Directory Entry



- Directories are structured in a hierarchical tree. Each directory can contain files and subdirectories.
- Directories are implemented as a special type of files. Actually, a directory is a file containing a list of entries. Each entry contains an inode number and a file name.

When a process uses a pathname, the kernel code searches in the directories to find the corresponding inode number. After the name has been converted to an inode number, the inode is loaded into memory and is used by subsequent requests.

Ref: 1

Metadata

- Nearly all file systems keep metadata about files.
- Some systems keep metadata in
  - directory entries;
  - others in specialized structure like inodes
  - or even in the name of a file.
- Metadata can range from simple timestamps, mode bits, and other special-purpose information used by the implementation itself, to icons and free-text comments, to arbitrary attribute-value pairs.
- Linux implements file metadata using extended file attributes.
- Extended file attributes is a file system feature that enables users to associate arbitrary metadata with computer files, whereas regular attributes have a strictly defined purpose (such as permissions or records of creation and modification times).
- Typical uses can be storing the author of a document, the character encoding of a plain-text document, or a checksum.

Master Boot Record (MBR)

- In the IBM PC architecture the **Master Boot Record (MBR)**, or **partition sector**, is the 512-byte boot sector,
- i.e. the sector on the logical beginning of a hard disk that contains the sequence of commands necessary for booting the operating system(s) (OSes).
- The values in the partition table (contained in the MBR) depend directly on the size of the physical disk and on the logical partitioning on that disk.

Layout of Master Boot Record

- address function

|===|
| 0x0000 Code Area (440 Bytes max.) |
|===|
|0x01B8 4 byte disk serial number |
| 2 bytes null (0) |
|===|
| 0x01BE 16 byte partition table entry|
|===|
| 0x01CE 16 byte partition table entry|
|===|
| 0x01DE 16 byte partition table entry|
|===|
| 0x01EE 16 byte partition table entry|
|===|
| 0x01FE 2 byte MBR signature (0xAA55)|
|===|

Partition table and Boot Code

- **Master Partition Table:** This small table contains the descriptions of the partitions that are contained on the hard disk.

  One of the partitions is marked as active, indicating that it is the one that the computer should use for booting up.

- **Master Boot Code:** The master boot record contains the small initial boot program that the BIOS loads and executes to start the boot process.

  This program eventually transfers control to the boot program stored on whichever partition is used for booting the PC.

Types of file systems

- **Disk file systems**

- A *disk file system* is a file system designed for the storage of files on a data storage device, most commonly a disk drive, which might be directly or indirectly connected to a computer.

- Examples of disk file systems include FAT, NTFS, HFS, ext2, ISO 9660, ODS-5, and UDF.

- Some disk file systems are also journaling file systems or versioning file systems.

Types of file systems - 1

- **Database file systems**

- Files are identified by their characteristics,
  - type of file,
  - topic,
  - author,
  - or metadata.

- Examples include Gnome VFS, BFS, and WinFS.

Types of file systems - 2

- **Transactional file systems**
  - It logs events or transactions to files.
  - Each operation that you do may involve changes to a number of different files and disk structures.
  - In many cases, these changes are related, meaning that it is important that they all be executed at the same time.
  - Take for example a bank sending another bank some money electronically. The bank's computer will "send" the transfer instruction to the other bank and also update its own records to indicate the transfer has occurred. If for some reason the computer crashes before it has had a chance to update its own records, then on reset, there will be no record of the transfer but the bank will be missing some money. A transactional system can rebuild the actions by resynchronizing the "transactions" on both ends to correct the failure. All transactions can be saved, as well, providing a complete record of what was done and where.
  - This type of file system is designed and intended to be fault tolerant and necessarily, incurs a high degree of overhead.

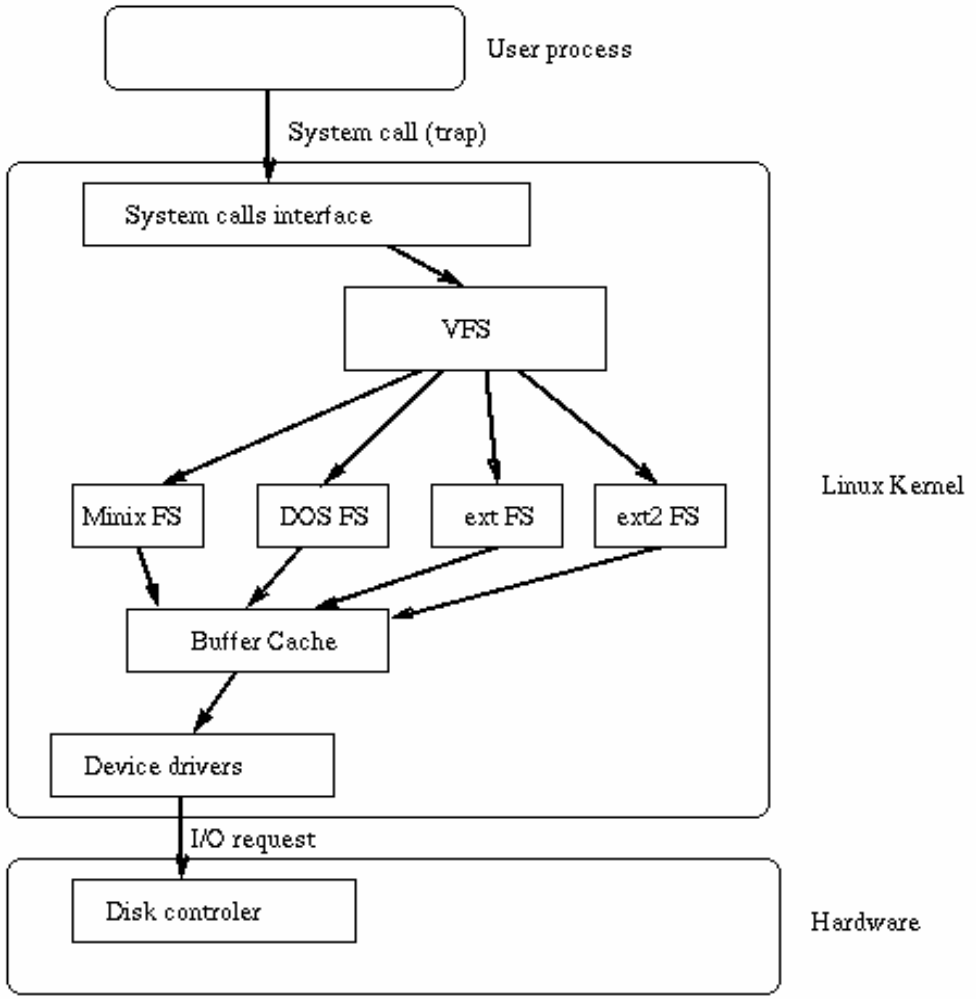
Types of file systems - 3

- **Special purpose file systems**

- It is any file system that is not a disk file system or a network file system.
- Here files are arranged dynamically by software.
- Special purpose file systems are most commonly used by file-centric operating systems such as Unix. Examples include the '/proc' file system used by some Unix variants, which grants access to information about processes and other operating system features.
- Deep space science exploration craft, like Voyager I & II used digital tape based special file systems. Most modern space exploration craft like Cassini-Huygens used Real-time operating system file systems or RTOS influenced file systems. The Mars Rovers are one such example of an RTOS file system, important in this case because they are implemented in flash memory.

Virtual File System (VFS)

- When a process issues a file oriented system call, the kernel calls a function contained in the VFS.
- This function handles the structure independent manipulations and redirects the call to a function contained in the physical filesystem code, which is responsible for handling the structure dependent operations.
- Filesystem code uses the buffer cache functions to request I/O on devices.



Ref: 13

The VFS structure

- The VFS defines a set of functions that every file system has to implement. This interface is made up of a set of operations associated to three kinds of objects: file systems, inodes, and open files.
- The VFS knows about file system types supported in the kernel.

  It uses a table defined during the kernel configuration.

  Each entry in this table describes a file system type: it contains the name of the file system type and a pointer on a function called during the mount operation.

  When a file system is to be mounted, the appropriate mount function is called.
- Two other types of descriptors are used by the VFS: an inode descriptor and an open file descriptor.

  Each descriptor contains information related to files in use and a set of operations provided by the physical file system code.

FAT: File Allocation System

- **File Allocation Table** (**FAT**) is a patented file system developed by Microsoft for MS-DOS and is the primary file system for consumer versions of Microsoft Windows.

The most common implementations have a serious drawback in that when files are deleted and new files written to the media, their fragments tend to become scattered over the entire media making reading and writing a slow process.

De-fragmentation is one solution to this, but is often a lengthy process in itself and has to be repeated regularly to keep the FAT file system clean.

There are 3 types of FAT: *FAT12, FAT16 and FAT32*

FAT (contd.)

- A FAT file system is composed of four different sections.
- **Reserved sectors**

  The first reserved sector is the Boot Sector (aka Partition Boot Record). The total count of reserved sectors is indicated by a field inside the Boot Sector.
- The **FAT Region**

  This contains two copies of the File Allocation Table for the sake of redundancy.
- The **Root Directory Region**

  This is a Directory Table that stores information about the files and directories in the root directory.
- The **Data Region**

  This is where the actual file and directory data is stored and takes up most of the partition.

FAT (contd.)

- The **File Allocation Table** (**FAT**) is a list of entries that map to each cluster on the partition. Each entry records one of five things:

- the address of the next cluster in a chain
- a special *end of file* (*EOF*) character that indicates the end of a chain
- a special character to mark a bad cluster
- a special character to mark a reserved cluster
- a zero to note that that cluster is unused

NTFS

- Developed by Microsoft
- In NTFS,
  - file name,
  - creation date,
  - access permissions
  - and even contents is stored as metadata.
- This elegant, albeit abstract, approach allowed easy addition of file system features during the course of Windows NT's development
- It’s far more robust, it supports Unicode filenames, proper security, compression and encryption.

NTFS: features

- **Alternate data streams (ADS)**

Alternate data streams allows files to be associated with more than one data stream.

For example, a file such as text.txt can have a ADS with the name of text.txt:secret.txt (of form *filename:ads*) that can only be accessed by knowing the ADS name or by specialized directory browsing programs.

NTFS: features (contd.)

- **Quotas**

- File system quotas were introduced in NTFS 5.
- They allow the administrator of a computer that runs a version of Windows that supports NTFS to set a threshold of disk space that users may utilize.
- It also allows administrators to keep a track of how much disk space each user is using.
- An administrator may specify a certain level of disk space that a user may use before they receive a warning, and then deny access to the user once they hit their upper limit of space.
- Disk quotas do not take into account NTFS's transparent file-compression, should this be enabled.
- Applications that query the amount of free space will also see the amount of free space left to the user who has a quota applied to them.

NTFS: features (contd.)

- **Sparse files**
- An application that reads a sparse file reads it in the normal manner with the file system calculating what data should be returned based upon the file offset.
- In the case of compressed files, the actual size of sparse files are not taken into account when determining quota limits.

NTFS: features (contd.)

- **Volume mount points**

- This is used when the root of another file system is attached to a directory. In NTFS, this allows additional file systems to be mounted without requiring a separate drive letter (like C: or D:) for each.

NTFS: features (contd.)

- **Hierarchical Storage Management (HSM)**
- Hierarchical storage management is a means of transferring files that are not used for some period of time to less expensive storage media.
- When the file is next accessed the reparse point on that file determines that it is needed and retrieves it from storage.

NTFS: features (contd.)

- **File compression**
- NTFS can compress files using a variant of the *LZ77 algorithm* (also used in the popular ZIP file format).
- Although read-write access to compressed files is transparent, it is recommended to avoid compression on server systems and/or network shares holding roaming profiles because it puts a considerable load on the processor.

NTFS: features (contd.)

- **Single Instance Storage (SIS)**
- When there are several directories that have different, but similar files, some of these files may have identical content.
- *Single instance storage* allows identical files to be merged to one file and create references to that merged file.
- SIS consists of a file system filter that manages copies, modification and merges to files; and a user space service (or *groveler*) that searches for files that are identical and need merging. SIS was mainly designed for remote installation servers as these may have multiple installation images that contain many identical files; SIS allows these to be consolidated but, unlike for example hard links, each file remains distinct; changes to one copy of a file will leave others unaltered.

NTFS: features (contd.)

- **Encrypting File System (EFS)**
- Provides strong and user-transparent encryption of any file or folder on an NTFS volume.
- EFS works by encrypting a file with a bulk symmetric key (also known as the File Encryption Key, or FEK), which is used because it takes a relatively smaller amount of time to encrypt and decrypt large amounts of data than if an asymmetric key cipher is used.

NTFS: Limitations

- **Reserved File Names**

- Though the file system supports paths up to 32,000 Unicode characters with each path component (directory or filename) up to 255 characters long, certain names are unusable, since NTFS stores its metadata in regular (albeit hidden and for the most part inaccessible) files; accordingly, user files cannot use these names.

- These files are all in the root directory of a volume (and are reserved only for that directory).

- The names are: $Mft, $MftMirr, $LogFile, $Volume, $AttrDef, . (dot), $Bitmap, $Boot, $BadClus, $Secure, $Upcase, and $Extend [9]; . and $Extend are both directories, the others are files.

NTFS: Limitations (contd.)

- **Maximum Volume Size**

- NTFS is a 64 bit filesystem and in theory its limits are fairly big, however the Microsoft implementation limits file size to 16 TB, volume size to 256 TB and the number of files to 4 billions.

- Because partition tables on master boot record (MBR) disks only support partition sizes up to 2 TiB, you must use dynamic volumes to create NTFS volumes over 2 TiB.

NTFS: Looking for volumes

fdisk -l

The output might look like:
Disk /dev/hda: 64 heads, 63 sectors,
*4465 cylindersUnits = cylinders of 4032 * 512 bytes*

Device Boot	Start	End	Blocks	Id	System
/dev/hda1	1	2125	4283968+	07	NTFS/HPFS
/dev/hda2	2126	19851	35735616	0f	Win95 Ext'd (LBA)
/dev/hda5	2126	4209	4201312+	83	Linux
/dev/hda6	4210	4465	516064+	82	Linux swap

ext2

- The **ext2** or **second extended file system** is a file system for the Linux kernel.
- Its main drawback is that it is not a journaling file system.
- Its successor, ext3, is a journaling file system and is almost completely compatible with ext2.
- The Ext2fs supports standard Unix file types: regular files, directories, device special files and symbolic links.
- Ext2fs provides long file names. It uses variable length directory entries.

  The maximal file name size is 255 characters. This limit could be extended to 1012 if needed.
- Ext2fs reserves some blocks for the super user (root). Normally, 5% of the blocks are reserved.

Journaling

- A **journaling file system** is a file system that logs changes to a journal (usually a circular log in a specially-allocated area) before actually writing them to the main file system.
- Journaling can have a severe impact on performance because it requires that all data be written twice.
- Metadata-only journaling is a compromise between reliability and performance that stores only changes to file metadata (which is usually relatively small and hence less of a drain on performance) in the journal.

  This still ensures that the file system can recover quickly when next mounted, but leaves an opportunity for data corruption because un-journal file data and journal metadata can fall out of sync with each other.

ReiserFS

- Metadata-only journaling
- Online resizing (growth only), with an underlying volume manager such as LVM(Logical Volum e Manager)

  Since then, Namesys has also provided tools to resize (both grow and shrink) ReiserFS file systems offline.
- Tail packing, a scheme to reduce internal fragmentation.

  Tail packing, however, has a significant performance impact; Namesys recommends disabling the feature in performance-critical applications.
- Reiserfs uses its balanced trees to streamline the process of finding the files and retrieving their security (and other) metadata.

ReiserFS: disadvantages

- ReiserFS v3 may become corrupt when its tree is rebuilt during a file system check.
- Some file operations are not synchronous on ReiserFS, which can cause some subtle breakage in applications relying heavily on file-based locks.
- There is no known way to defragment a ReiserFS file system, aside from a full dump and restore.

Reiser4

- Efficient journaling
- Efficient support of small files, in terms of disk space and speed
- Fast handling of very large directories with hundreds of millions of files
- Flexible plug-in infrastructure (through which special metadata types, encryption and compression will be supported)
- Atomic file system modification
- Dynamically optimized disk-layout through allocate-on-flush (also called delayed allocation in XFS)
- Transaction support

References

[1] http://web.mit.edu/tytso/www/linux/ext2intro.html
[2] http://en.wikipedia.org/wiki/Master_boot_record
[3] http://en.wikipedia.org/wiki/List_of_file_systems
[4] http://en.wikipedia.org/wiki/Journaling_file_system
[5] http://wiki.linux-ntfs.org/doku.php?id=ntfs-en
[6] http://wiki.linux-ntfs.org/doku.php?id=ntfsmount
[7] http://www.namesys.com/mount-options.html
[8] http://www.namesys.com/v4/v4.html
[9] http://www.linuxplanet.com/linuxplanet/tutorials/2926/4/
[10] http://www.tldp.org/HOWTO/Filesystems-HOWTO-5.html
[11] http://en.wikipedia.org/wiki/List_of_file_systems
[12] http://en.wikipedia.org/wiki/Comparison_of_file_systems
[13] http://www.meteck.org/ext2.htm