Storage & File Systems

- Non volatile long term storage is essential for any modern computer system.
- Data are stored as files on devices.
- Examples of storage devices include tape, hard disk, optical disks & usb storage devices.
- In the past, data was also stored on floppy disks, zip disks and cassette tape.

Storage & File Systems (cont.)

- In order to store files a device has to be formatted i.e. a file system has to be written to it.
- Many devises are formatted as part of the manufacture process
- Many companies offer on-line file hosting services e.g. Dropbox, Google Drive & Microsoft OneDrive. Usually, a modest amount of storage space is offered for free.
- This has the advantage that it can be accessed anywhere and backup is also provided.
- Some concerns about security.

Storage & File Systems

- A file system is the method by which a set of files are stored and catalogued. Different devices may have different file systems. All file systems need organization, the ability to manipulate files, access data, etc.

- Specific files can be created, deleted, retrieved, and modified when the computer or user issues a command - the file system is a set of conventions which guides this.

Disk Space Allocation

- Disks have a set block size and the file system allocates disk space by the block.

- Allocating disk space by the block will always result in some internal fragmentation. In general, larger blocks result in more fragmentation - smaller blocks, however, means more disk work.

Disk Space Allocation

- **Internal Fragmentation –** a block that is not fully used results in some unusable space e.g. if the block is 4k and only 355 bytes are used then the internal fragmentation is 4096 – 355 = 3741bytes (4k = 4 x 1024 bytes)

- Files must be allocated portions of disk space for storage.

- The command **ls –s** lists files and their block sizes in the current directory.

4 -rw-rw-r-- 1 cadmin cadmin 86 Mar 20 16:25 myFirstScript

Free Space List

As files are deleted and created, different areas of the disk are occupied or freed.

The disk manager must keep track of this **free space list**.

There are different ways in which this free space list can be implemented: Bit map, linked list and free sequence list.

Allocation Objectives

- Efficient utilization of disk space.
- Speed of access to files.
- These objectives are not necessarily compatible i.e. the more data on a device – sometimes the slower it is to retrieve it.
- There are three common methods: *contiguous, chained (linked)* and *indexed.* Whatever method is used, disk management requires information about free disk space. Operating system keeps a record of free space.

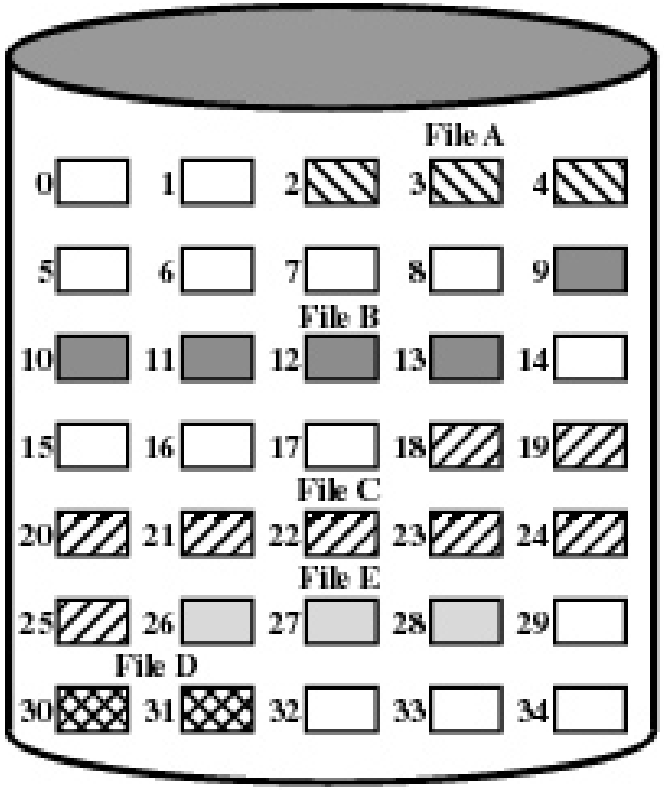
Contiguous Allocation

- Each file is allocated contiguous disk blocks. For each file, the device directory contains the location of its starting block and the number of blocks.

- Access is easy. Sequential and direct access is possible. Notice also that this allocation method reduces head movement during file access - most blocks are on the same cylinder.

- Finding space for a new file is awkward. The free space list must be searched for a big enough chunk of contiguous space.

Contiguous Allocation (cont.)



File Allocation Table

File Name	Start Block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3

Contiguous Allocation (cont.)

- At any time a disk is made up of allocated and free sequences of blocks. Free block sequences are called *holes*. The dynamic allocation problem is that of finding the most suitable hole for a file of size *n* blocks.
- Direct access is possible – easy to locate a single block.
- This is prone to external fragmentation. A possible solution is compaction – however this can be a heavy load for the processor.
- **External Fragmentation –** when whole blocks are unusable e.g. in the previous diagram block 29 is probably unusable.

Contiguous Allocation (cont.)

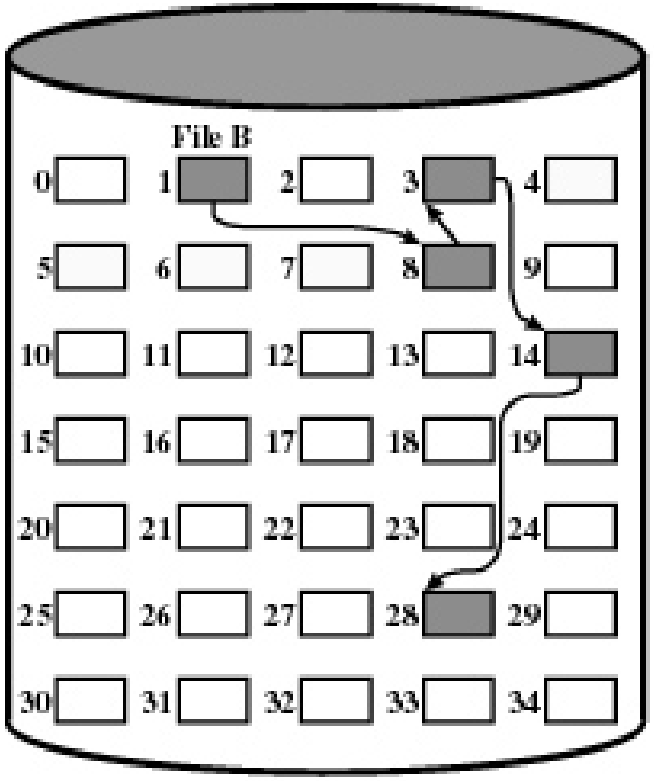
Contiguous allocation has other problems. The size of the file must be known. However, most of the time this is not the case. Thus, as a file grows it may have to be moved about the disk a lot in order to find a big enough space for its new size.

Anticipating growth by over-allocating space leads to more fragmentation, the extra space may never be used.

Chained (Linked) Allocation

- There are problems with allocating contiguous space.
- Linked allocation avoids those problems. This is achieved by allowing the disk space to be allocated randomly.
- A file's space is a linked list of blocks. Pointers are maintained to the first and last blocks. Each block points to its successor. For each file, the device directory just contains the location of its starting block.

Chained (Linked) Allocation



File Allocation Table

File Name	Start Block	Length
• • •	• • •	• • •
File B	1	5
• • •	• • •	• • •

Chained (Linked) Allocation

- To access a random block (i.e. any individual block within the file) the access must begin at the start and follow the pointers. Finding space for files is easy. Blocks are chosen from the free space list and linked to the previous blocks. There is no external fragmentation.

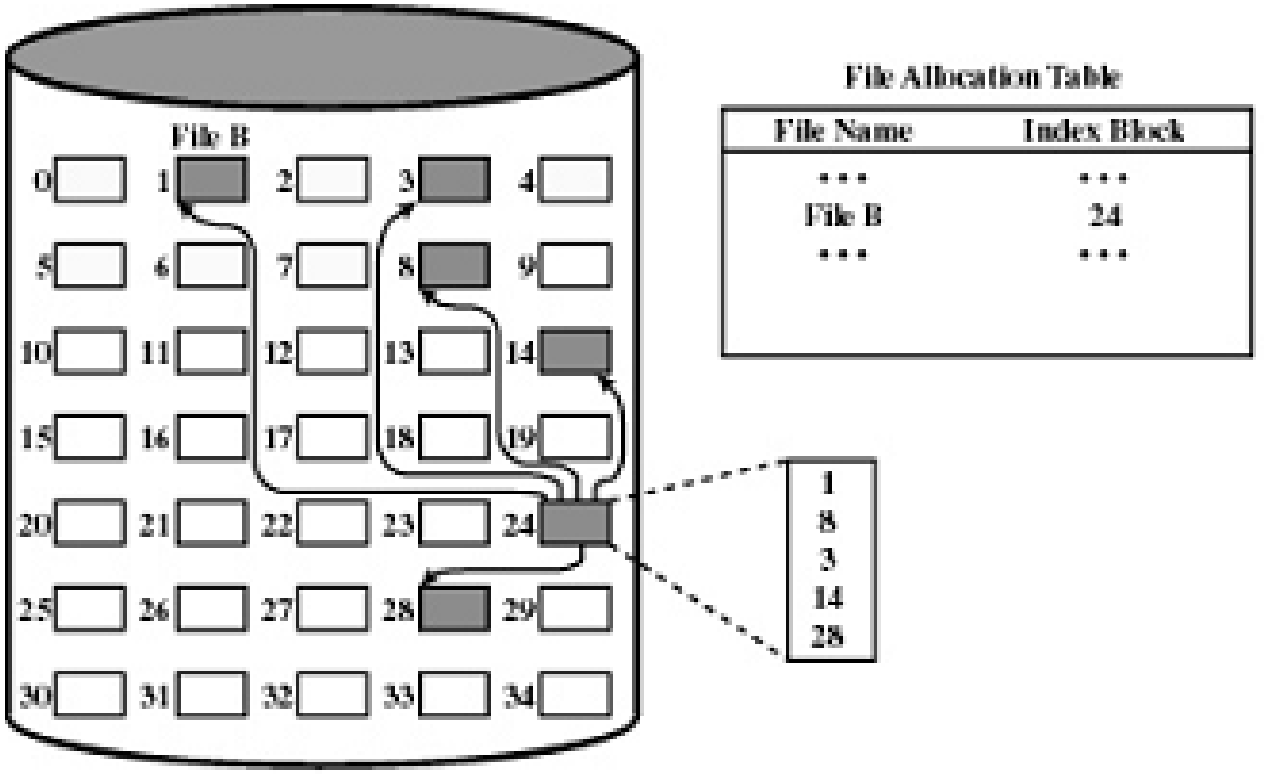
- A minor problem is the space used up by pointers. A more serious problem is with pointers scattered all over the disk, the possibility of losing a part of a file due to a corrupt pointer is high.

Indexed Allocation

- All of a file's block pointers are gathered together into its *index block* - an array of pointers to blocks stored on disk. The *ith* entry is the pointer to the file's *ith* block.

- Access is easy. Both sequential and direct access possible. No external fragmentation. Allocation is easy. Files are allocated extra blocks from the free space list as they grow.

Indexed Allocation (cont.)



Indexed Allocation (cont.)

Two problems however:

1. The index block may be underused. Many files are small and thus will only use a small part of the index block.

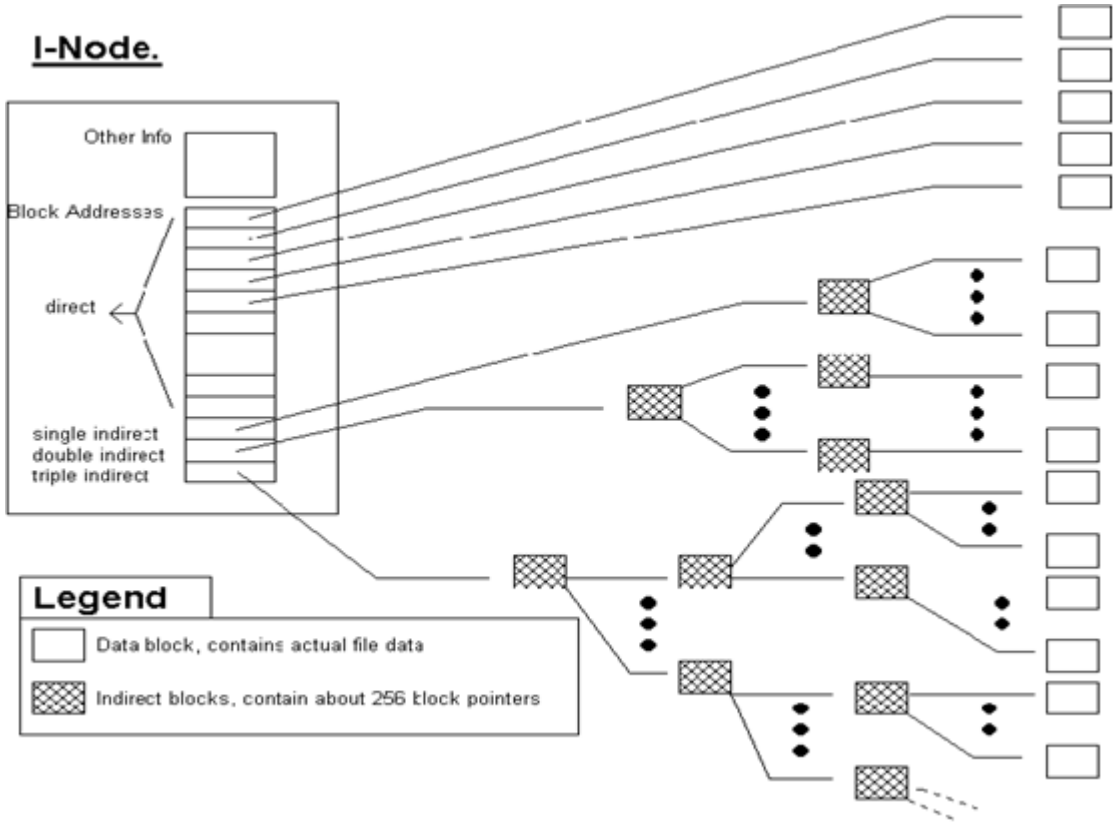
2. More seriously, the index block is of fixed size. This sets a limit on file size.

UNIX Disk Space Allocation

- All UNIX files are administered through *inodes* (i.e. information node or index node). The i-node is the unique block of descriptor information for that file (containing, for example, the file's block numbers on disk). Each file has an i-node.

- UNIX uses a variation on the indexed method. Each i-node contains a file address made up of thirteen addresses. The first 10 point to the first 10 blocks of the file. If the file is longer than 10 blocks then the remaining 3 addresses are used to implement one or more levels of indirection as needed in the following order:

UNIX Disk Space Allocation



UNIX Disk Space Allocation

Single indirection

- 11th address points to a block containing an index. Each entry in the index block points to the succeeding blocks of the file.

Double indirection

- 12th address points to a block containing an index to indexes. Each entry in the first index block points to another index block which itself points to the succeeding blocks of the file.

Triple indirection

- 13th address points to a block containing an index to indexes to indexes. Each entry in the first index block points to a second set of index blocks. Each entry in the index blocks of the second level of indexes points to a third set of index blocks which themselves point to the succeeding blocks of the file.

UNIX Disk Space Allocation

- The total number of data blocks that a UNIX file can contain depends then on the capacity of the fixed-sized blocks. For example, if the length of a block is 1Kbyte and each block that acts as an index block can hold 256 block addresses. So the maximum file size is over 16Gbytes ($10 + 256 + 256^2 + 256^3 = 16.7G$)

- As a rough guide ten percent of the file system should be i-Nodes. This figure should be increased if the partition will contain lots of small files or decreased if the partition will contain few but large files.

UNIX Disk Space Allocation

Advantages of the scheme

- The i-node is of a small fixed size and so can be kept in memory for long periods.
- Smaller files (most common) can be accessed with no indirection which reduces access time.
- The upper limit of 16 Gbytes is large enough to cope with virtually all files (current Linux systems have higher limits). Larger files require double and triple indirect addressing.

UNIX Disk Space Allocation

In addition to storing the addresses of the 13 data blocks where the contents of the file are placed, the i–node contains information about:

- The owner (UID) and group owner (GID) of the file.
- The type of file - is the file a directory or another type of special file?
- User access permissions - which users can do what with the file
- The number of hard links to the file - the same physical file may be accessed under several names; we will examine how later.
- The size of the file
- The time the file was last modified
- The time the I-Node was last changed - if permissions or information on the file change then the I-Node is changed.

File Systems

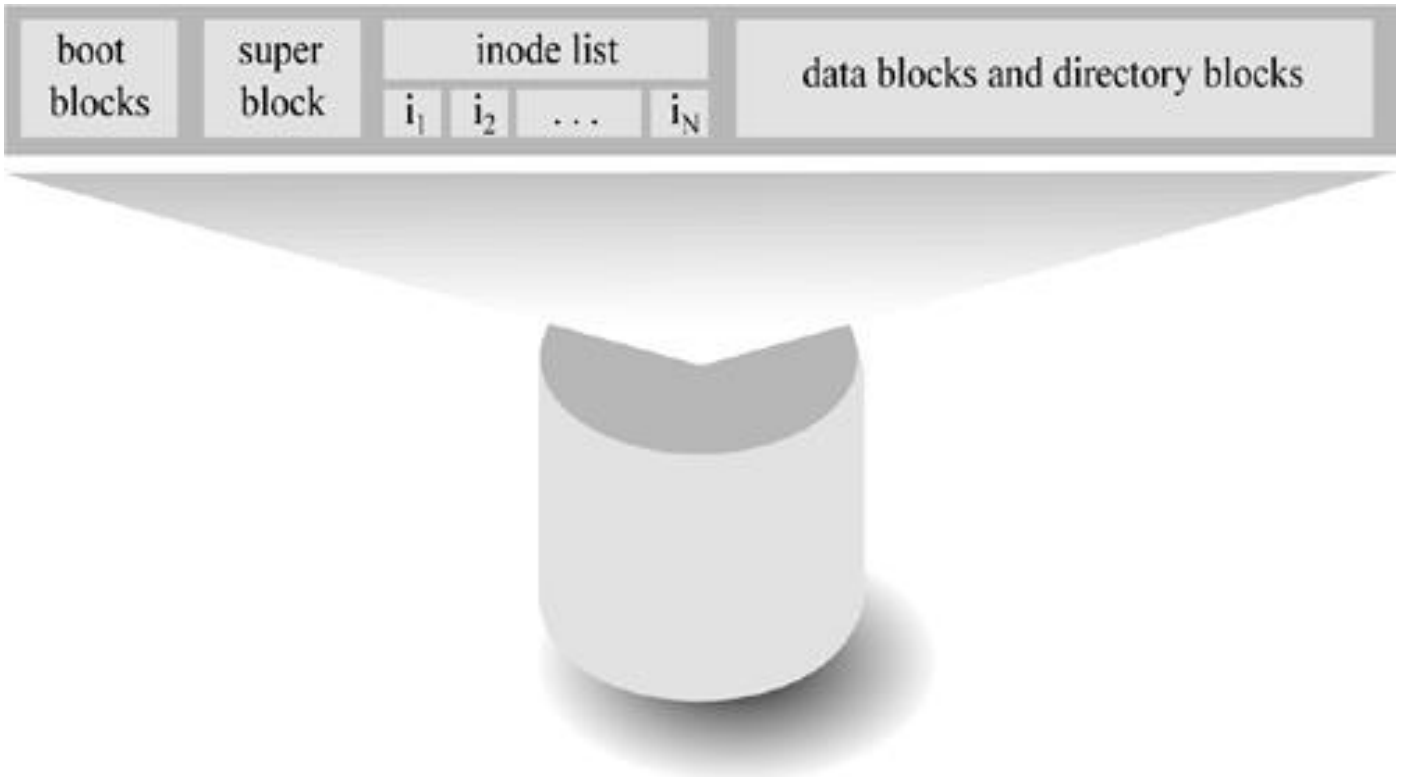
- The Linux file system (ext4 – Extended File System 4) uses the i-node block addressing scheme.

- Each device or partition can hold only one file system. A hard disk can be divided into a number of partitions each effectively treated as an independent storage area with its own i-node table.

- Windows systems generally have FAT or NTFS systems. In many cases there is a level of compatibility between different systems i.e. a Linux system can read data on an NTFS system, etc.

File Systems

The General structure of a file system



File Systems

When partitions are created, the first block of every partition is reserved as the **boot block**. Here, the bootstrap loader is executed to launch the OS. Systems that contain two or more operating systems use the boot block to house small programs (e.g. **lilo** and **grub**) that ask the user to chose which OS they wish to boot.

The second block on the partition is called the **superblock**. It contains all the information about the partition including information on:

- The size of the partition
- The physical address of the first data block
- The number and list of free blocks
- Information of what type of file system uses the partition
- When the partition was last modified

File Systems

Other File Systems

- **FAT File System** The File Allocation Table, or FAT, file system is often associated with MS-DOS, but was used in Windows until recently. It is still used by many forms of removable media, for example on USB flash drives, floppy disks, etc and it is supported by almost every operating system.

- **NTFS File System** New Technology File System, or NTFS, is the file system used by current versions of Windows. NTFS has replaced FAT as Window's primary file system.

- **Apple Mac OS Extended File System** Apple's OS uses a file system known as HFS Plus, or Hierarchical File System Plus, though it is sometimes referred to as Mac OS Extended.

- **Network files systems** include Sun's NFS, Distributed File System, and AppleShare.

File Systems

What corrupts a File System?

When a file system is being manipulated and the process is unexpectedly interrupted corruption is likely to occur for example - the loss of power at a time of change. As the changes where not fully made, an operating system or application cannot make sense the data which was written because it is incomplete.

Partitioning a Disk

In order to write a file system to a disk, one must first create a partition for it.

An Intel hard disk can be divided into 4 primary partitions which means a number of file systems can exist on the same disk.

Primary partitions can also be marked as extended – and can be further divided into a number of logical partitions.

Advantages of Partitioning a Disk

- Multiple Operating Systems - Having more than one operating system is an obvious advantage on any machine. It's possible to join both to a local network and a wide area network including the Internet. The relatively low cost of disk space and installing open source operating systems makes this a viable option.

- Separation Issues Different directory branches can be kept on different partitions for many reasons including - certain directories may contain data that will only need to be read (e.g. application software), others will need to be both read and written (e.g. users' home directories & configuration files). It is possible to mount these partitions restricting such operations. There are also security advantages of this – read only partitions can not be infected by viruses; also, it's easy to un-mount partitions (i.e. make them unavailable) when they are not required.

Advantages of Partitioning a Disk

- Certain directories can fill up with files very quickly.
- Management - the logical division of system software, local software and home directories all lend themselves to separate partitions
- Backup Issues - separating directories can make the process backups and upgrades easier.
- Performance Issues - by spreading the file system over several partitions and devices, the IO load is spread around.