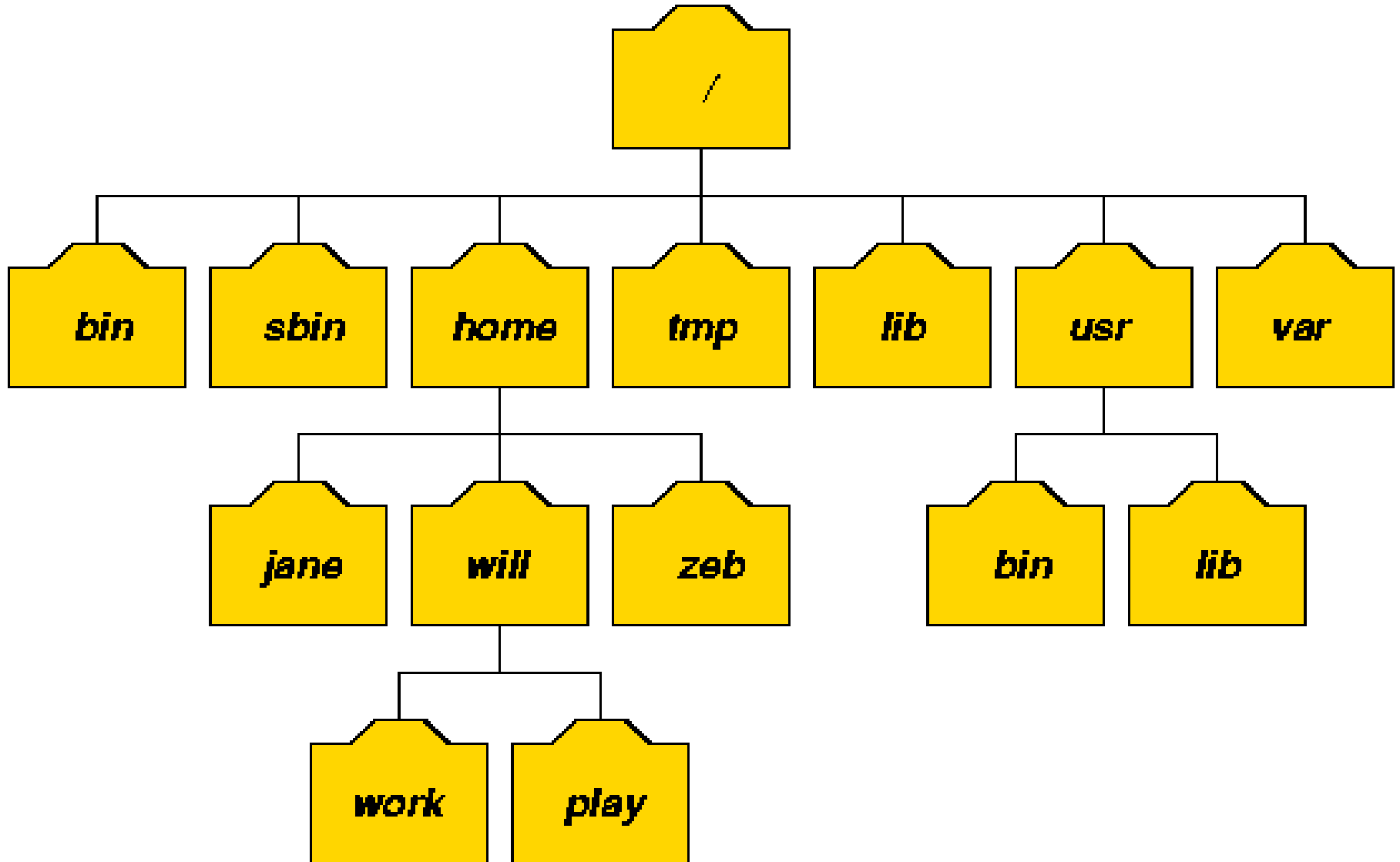


UNIX AND LINUX FILE SYSTEM



WHAT IS A FILE?

1. The file is a container for storing information. We can simply treat it as a sequence of characters. If you name a file `foo` and write three characters `a`, `b` and `c` into it, then `foo` will contain only the string `abc` and nothing else.
2. Unlike the old DOS files, a UNIX file doesn't contain the eof(end-of-file) mark. A file's size is not stored in the file, nor even its name. All file attributes are kept in a separate area of the hard disk, not directly accessible to humans, but only to the kernel.
3. UNIX treats directories and devices as files as well. A directory is simply a folder where you store filenames and other directories. All physical devices like the hard disk ,memory,CD-ROM,printer and modem are treated as files.

TYPES OF FILE IN UNIX

ORDINARY FILES

1. Ordinary files are comprised of streams of data (bytes) stored on some physical device.
2. Examples of ordinary files include simple text files, application data files, files containing high-level source code, executable text files, and binary image files.
3. Note that unlike some other OS implementations, files do not have to be binary Images to be executable. Also known as regular file.
4. It is of two types:
 - Text File
 - Binary File

TEXT FILE

- 1. A text file contains only printable characters and you can often view the contents and make sense out of them.**
- 2. All C and JAVA program sources, shell and perl scripts are text files. A text file contains lines of characters where every line is terminated with a newline character, also known as linefeed(LF).**
- 3. When you press[Enter] while inserting text, the LF character is appended to every line. You won't see this character normally, but there is a command (od) which can make it visible.**

BINARY FILE

- 1. A binary file on the other hand, contains both printable and unprintable characters that cover the entire ASCII range(0 to 255).**
- 2. Most UNIX commands are binary files, and the object code and executables that you produce by compiling C programs are also binary files.**
- 3. Picture, sound and video files are binary files as well.**

DIRECTORY FILE

- ❖ A directory contains no data, but keeps some details of the files and subdirectories that it contains.
- ❖ The UNIX file system is organized with a number of directories and subdirectories, and you can also create them as and when you need.
- ❖ A directory file contains an entry for every file and subdirectory that it houses. If you have 20 files in a directory, there will be 20 entries in the directory. Each entry has two components:
 - ❖ **The filename.**
 - ❖ **A unique identification number for the file or directory (called the inode number).**

DIRECTORY FILE:

1. A directory contains the filename and not the file's contents.
2. You cannot write a directory file, but you can perform some action that makes the kernel write a directory.

For instance ,when you create or remove a file, the kernel automatically updates corresponding directory by adding or removing the entry (inode number and filename) associated with the file.

DEVICE FILE

1. Device file types typically include: character device files, block device files, Unix domain sockets, named pipes and symoblic links. However, not all of these file types may be present across various Unix implementations.

2. This is another special file that is used to describe a physical device, such as a printer or a zip drive.

This file contains no data whatsoever, it merely maps any data coming its way to the physical device it describes.

3. Device filenames are generally found inside a single directory structure, `/dev`.

4. The operation of a device is entirely governed by the attributes of the associated file. The kernel identifies a device from its attributes and then uses them to operate the device.

MAJOR & MINOR DEVICE NUMBER

- **Device files have no file size instead they include Major & minor device number**
- **Major number represents the device driver being referred.**
- **All hard disk have same major number if they are attached to same controller.**
- **Minor number is passed by kernel to device driver to indicate possible instance of same device.**
- **Ex:fd0135ds18 & fd196ds15 shows two devices attached to a particular controller will have same major number but different minor number**

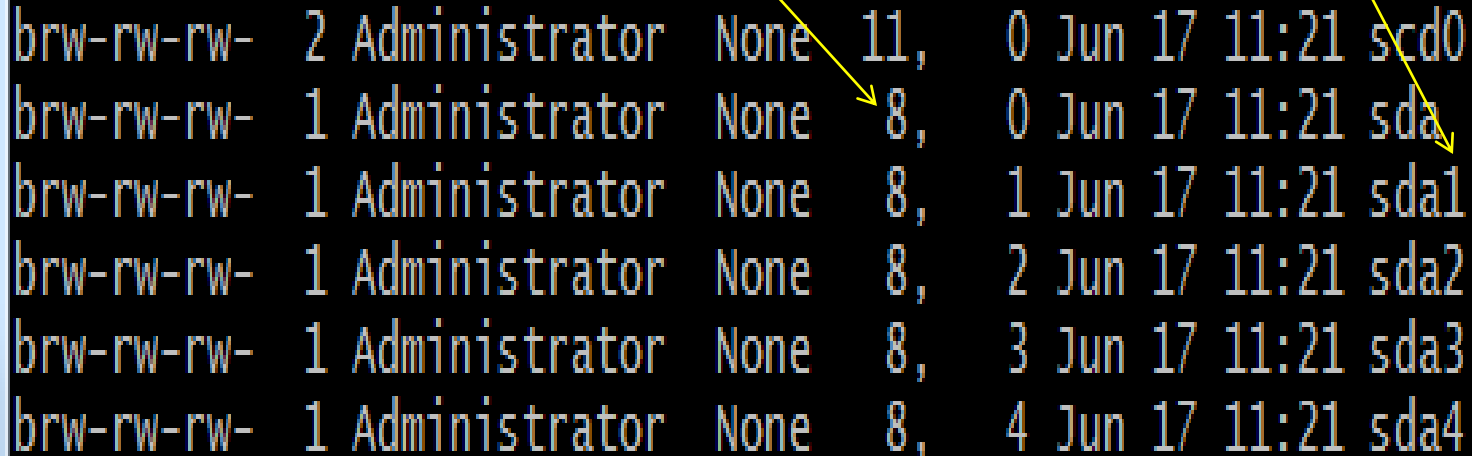
MAJOR & MINOR DEVICE NUMBER

e.g.: `$ cd /dev`

`[dev means devise file]`

`$ ls -l`

Here major number is **8** and minor number is **1,2,3,4**



```
brw-rw-rw- 2 Administrator None 11, 0 Jun 17 11:21 sdc0
brw-rw-rw- 1 Administrator None 8, 0 Jun 17 11:21 sda
brw-rw-rw- 1 Administrator None 8, 1 Jun 17 11:21 sda1
brw-rw-rw- 1 Administrator None 8, 2 Jun 17 11:21 sda2
brw-rw-rw- 1 Administrator None 8, 3 Jun 17 11:21 sda3
brw-rw-rw- 1 Administrator None 8, 4 Jun 17 11:21 sda4
```

SOME MORE FILE TYPES:

4. **SYMBOLIC LINK FILE:**

It is a logical file that defines the location of another file somewhere else in the system.

unix defines two types of links: hard links and soft links.

5. **FIFO FILE:**

a first in first out file also known as pipe is used for inter-process communication.

- **6. SOCKET FILE:**

It is a special file that is used for network communication.

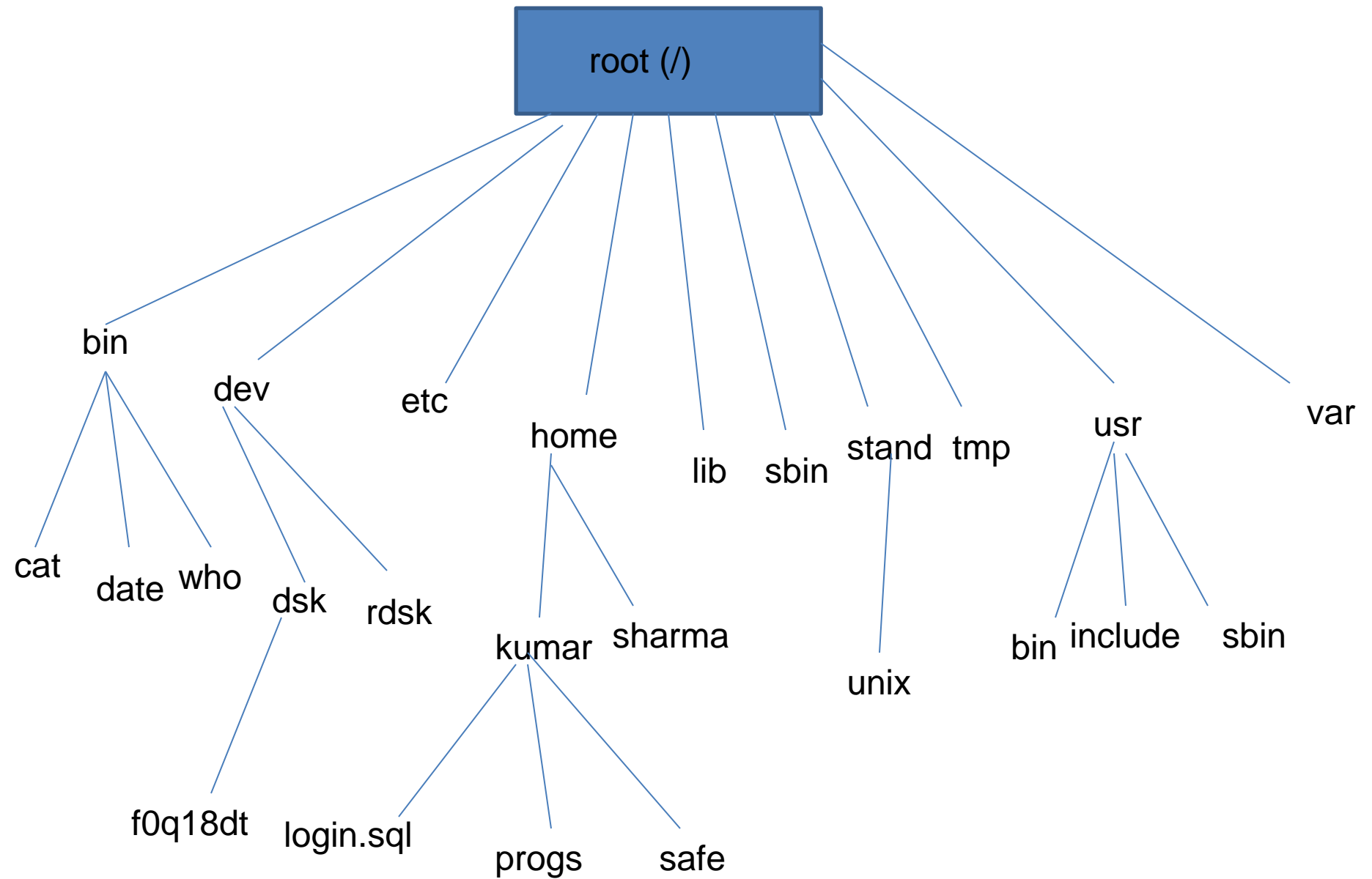
there are 2 types of socket files

(i) *Unix domain socket for IPC*: they can only be used for processes running locally on the same unix system to communicate with each other .

(ii) *Unix Internet socket*: it support protocol which allows to connect processes between different unix system

Characteristics of unix file system

- Hierarchical file structure
- Files system structure is dynamic.
- File have access permissions
- Structure less file
- All devices are implemented as files.



THE UNIX FILE SYSTEM TREE

❖ **/bin and /usr/bin**-These are the directories where all the commonly used UNIX commands are found. bin stands for binary, hence this directory contains binary files.

❖ **/sbin and /usr/sbin**-If there's a command that you can't execute but the system administrator can, then it would probably be in one of these directories.

❖ **/etc**-This directory contains the configuration files of the system. You can change a very important aspect of system functioning by editing a text file in this directory. It also has special files such as /etc/inittab, /etc/gettydef for booting, configuration file /etc/profile, /etc/passwd for registered users info., /etc/motd for storing messages etc.

❖ **/dev**-This directory contains all device files. These files don't occupy space on the disk. There could be more subdirectories like pts, dsk and rsk in this directory.

❖ **/lib and /usr/lib**-Contains all library functions provided by unix for programmers in binary form. You'll need to link your C programs with files in these directories.

/tmp-The directories where users are allowed to create temporary files. These files are wiped away regularly by the system.

/var-The variable part of the file system. Contains all your print jobs and your outgoing and incoming mail.

/usr/share/man-This is where the man pages are stored. There are separate subdirectories here (like man1,man2,etc) that contains the pages for each section. For instance, the man page of ls can be found in /usr/share/man/man1, where the 1 in man1 represents Section 1 of the UNIX manual.

/usr/src-it contains the source code for many of the utilities.

/home-it is the home directory of all registered users of unix system.

/usr/include-Contains the standard header files used by C programs. The statement `#include<stdio.h>` used in most C programs refers to the file `stdio.h` in this directory.

FEATURES OF SHELL:

- Interactive Environment
- Shell scripts
- Input/output redirection
- Piping mechanism
- Metacharacter facilities/filename substitution
- Background Processing
- Customized environment(fullfills personnel needs)
- Programming language constructs
- Shell variables

THE ARRANGEMENT OF DISK BLOCKS IN UNIX



BB : Boot Block

IL : inode List

SB : Super Block

DB: Data Blocks

BOOT BLOCK

It is the smallest part of Unix file system.

Boot block or block0 is the first block of the file system. It is normally reserve for booting process. And also called Master Boot Record(MBR).

When the system is booted, the system BIOS checks for the existence of the hard disk and loads the entire segment of the boot block into main memory.

Now, the control hands over to the bootstrapping program. Which is responsible to load the kernel into main memory.

The kernel of a unix system is usually stored in root directory of file system.

BOOT BLOCK

The boot block contains the code to bootstrap the OS. A boot block *is* located in the first few sectors of a file system. The boot block contains the initial bootstrap program used to load the operating system. Typically, the first sector contains a bootstrap program that reads in a larger bootstrap program from the next few sectors, and so forth.



SUPER BLOCK

Super block or block1 is the balance sheet of every unix file system.

It stores the characteristics of a file system.

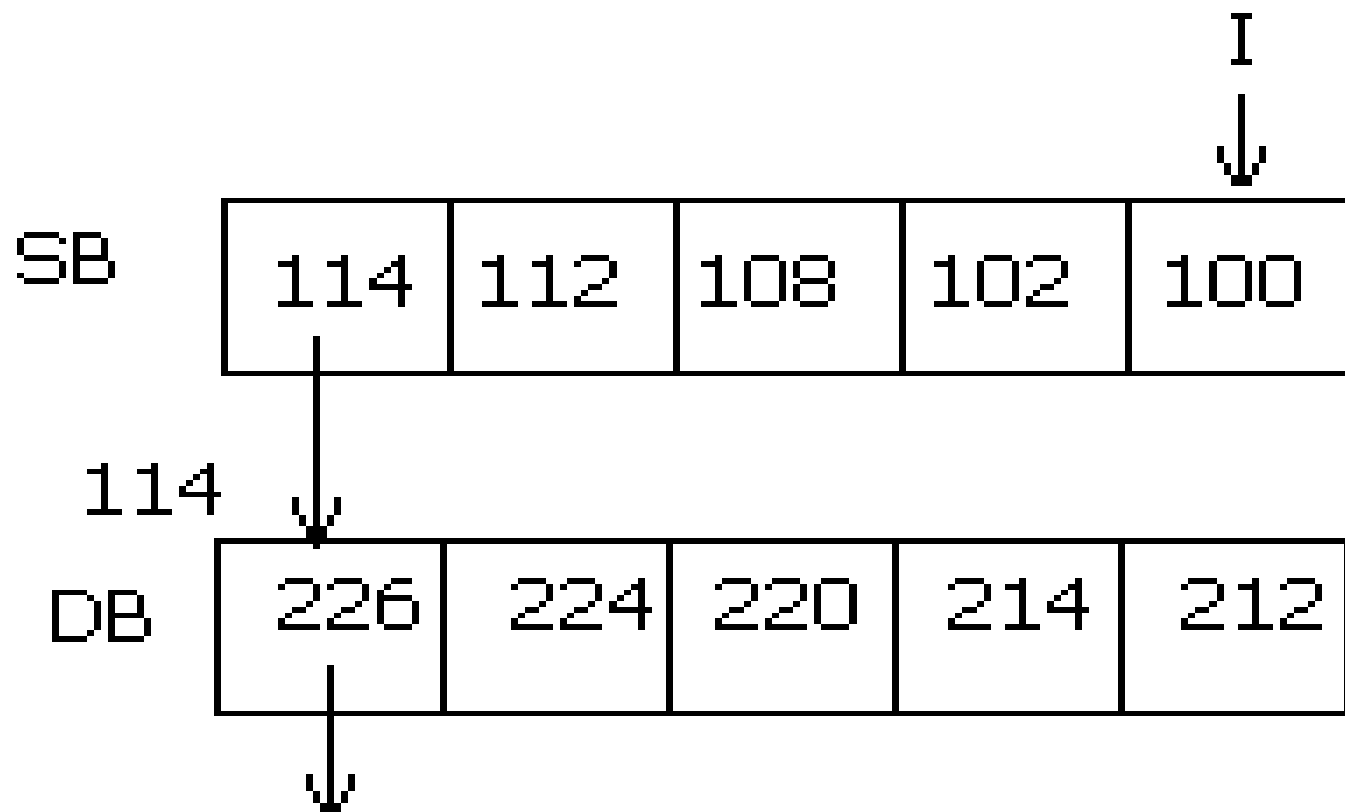
It mainly contain:

- The size of file system
- The size of logical block of the system
- Last time updating
- Free data blocks such as number of free data blocks available, list of free data blocks, pointer to first free data block on free data list.
- Free i-node such as the number of free I-node available, list of free i-nodes, pointer to first free I nodes on free i-node list.
- The state of the file system whether the file system is 'clear' or 'dirty'

SUPER BLOCK

A super block describes the **state of the file system**: the total size of the partition, the block size, pointers to a list of free blocks, the inode number of the root directory etc. The super block contains an array of free disk block numbers, one of which points to the next entry in the list. That entry in turn will be a data block, which contains an array of some other free blocks and a next pointer. When process requests for a block, it searches the free block list returns the available disk block from the array of free blocks in the super block. If the super block contains only one entry which is a pointer to a data block, which contains a list of other free blocks, all the entries from that block will be copied to the super block free list and returns that block to the process. Freeing of a block is reverse process of allocation. If the list of free blocks in super block has enough space for the entry then, this block address will be marked in the list. If the list is full, all the entries from the super block will be copied to the freed block and mark an entry for this block in the super block. Now the list in super block contains only this entry.

Start



SB : Super Block

DB : Data Block

I : Index

INODE

Internal representation of a file is called an "inode" (contraction of term - index node) which contains all the required information and description of the file data and its layout on disk.

This article deals in detail with the information stored in i-node and the way it is represented in the kernel.

I-nodes resides on the disk and the kernel reads them into an into memory which we can call as in-core inodes.

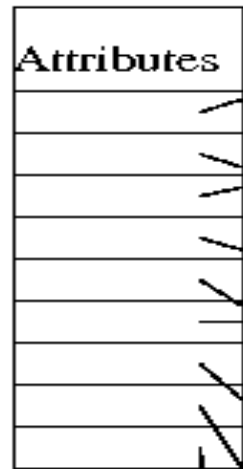
INODE

- ❖ An inode is the "handle" to a file and contains the following information:
- ❖ file ownership indication
- ❖ file type (e.g., regular, directory, special device, pipes, etc.)
- ❖ file access permissions.

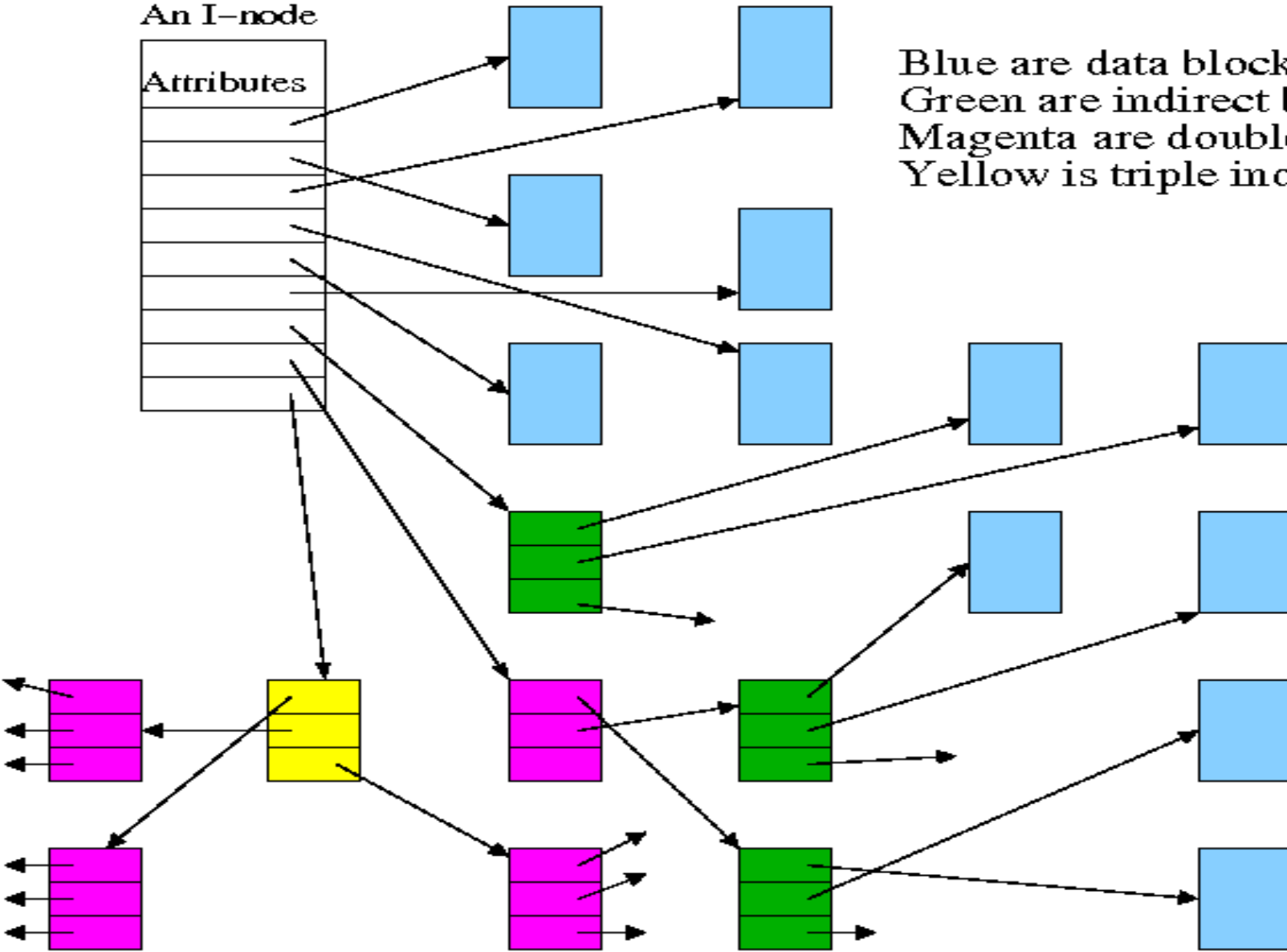
- ❖ **time of last access, and modification**
- ❖ **number of links (aliases) to the file**
- ❖ **pointers to the data blocks for the file**
- ❖ **size of the file in bytes (for regular files), major and minor device numbers for special devices.**
- ❖ **An integral number of inodes fits in a single data block.**

INODE STRUCTURE

An I-node



Blue are data blocks
Green are indirect blocks
Magenta are double indirect
Yellow is triple indirect



- Directories have filename and inodes of file with it. So we can view a directory as a table containing inode number and filename.
- Whenever any file requested its inode number is taken from directory table,then from it inode we can reach to the data blocks of file with the help of data pointers in inode.

- ❖ The inode table contains a listing of all inode numbers for the respective file system. When users search for or access a file, the UNIX system searches through the inode table for the correct inode number. When the inode number is found, the command in question can access the inode and make the appropriate changes if applicable.
- ❖ The inode consists of the following information:
 1. Inode number

- ❖ Number of links to the file
- ❖ UID of the owner
- ❖ Group ID (GID) of the owner
- ❖ Size of the file
- ❖ Actual number of blocks that the file uses
- ❖ Time last modified
- ❖ Time last accessed
- ❖ Time last changed

Stat command is used to have information of files like:

```
[nidhi@linux nidhi]$ stat a
```

```
File: `a'
```

```
Size: 4096      Blocks: 8      IO Block: 4096  directory
```

```
Device: 302h/770d  Inode: 471470  Links: 2
```

```
Access: (0775/drwxrwxr-x) Uid: ( 796/  nidhi)  Gid: ( 796/  
nidhi)
```

```
Access: 2016-08-19 02:53:13.0000000000 +0530
```

```
Modify: 2016-08-09 02:32:36.0000000000 +0530
```

```
Change: 2016-08-09 02:32:36.0000000000 +0530
```


Administrator@ADMIN ~

\$ stat neha

File: neha

Size: 0 Blocks: 4 IO Block: 65536 directory

Device: 1e1d096fh/505219439d Inode: 5629499534368670 Links: 1

Access: (0775/drwxrwxr-x) Uid: (197108/Administrator) Gid: (197121/ None)

Access: 2018-06-22 11:29:19.339159400 +0530

Modify: 2018-06-22 11:29:19.339159400 +0530

Change: 2018-06-22 11:29:19.339159400 +0530

Birth: 2018-05-11 11:12:13.001760300 +0530

Command `ls -i` is used to get inode numbers of each file and directory.

```
$ls -i
```

```
309649 1          309605 d11      309333 f1       309631 s1.sh    309676 y11
309652 1cd       309337 d2       309333 f12      309635 s2.sh    309670 y2
471470 a        34261 d3       309639 ft.sh   309629 s3.sh    309690 y22
309685 a*      534752 d4      309628 if.sh   309634 s4.sh    892100 z1
309686 a1     568543 d5      309642 m1      309637 s5.sh    309598 z13
309589 a1.lnk  293349 d6      309617 m2      309602 t1       309636 z3
```

Type or select values in entry fields.

Press Enter AFTER making all desired changes.

	[Entry Fields]
File system name	/usr
NEW mount point	[/usr]
SIZE of file system	
Unit Size	512bytes
Number of units	[16084320]
Mount GROUP	[bootfs]
Mount AUTOMATICALLY at system restart?	yes
PERMISSIONS	read/write
Mount OPTIONS	[]
Start Disk Accounting?	no
Block Size (bytes)	4096
Inline Log?	no
Inline Log size (MBytes)	[0]
Extended Attribute Format	[v1]
ENABLE Quota Management?	no
Allow Small Inode Extents?	<input checked="" type="checkbox"/> no

LISTING SHOWS INODE INFORMATION FOR THE FILE /USR/BIN/KSH IN AIX.

DATA BLOCK

This is where the file data itself is stored.

Since a directory is simply a specially formatted file, directories are also contained in the data blocks.

An allocated data block can belong to one and only one file in the system.

If a data block is not allocated to a file, it is free and available for the system to allocate when needed.

A ``block" is a 1024-byte unit of data stored on the disk.

A data block can contain either directory entries or file data.

A directory entry consists of an inode number, a filename, and a version number for [undelete\(C\)](#) (file versioning).

DATA BLOCK

Inodes include pointers to the data blocks.

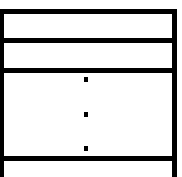
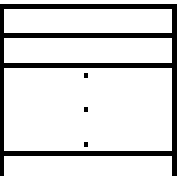
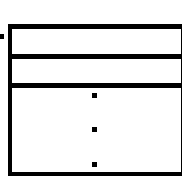
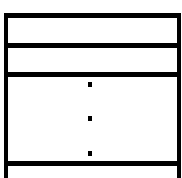
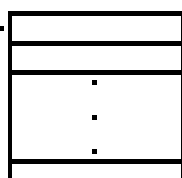
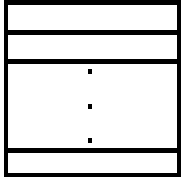
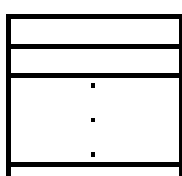
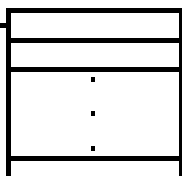
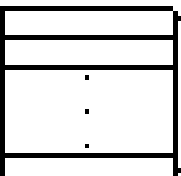
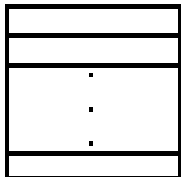
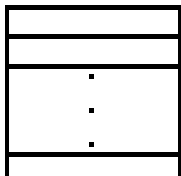
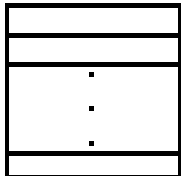
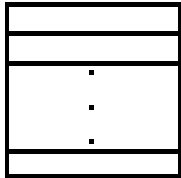
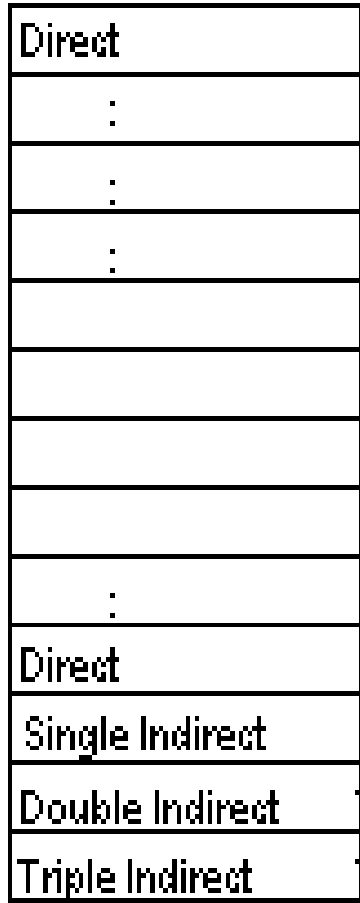
Each inode contains 15 pointers:

the first 12 pointers point directly to data blocks

the 13th pointer points to an indirect block, a block containing pointers to data blocks

the 14th pointer points to a doubly-indirect block, a block containing 128 addresses of singly indirect blocks

the 15th pointer points to a triply indirect block (which contains pointers to doubly indirect blocks, etc.)



DATA BLOCKS

HANDLING FILE SYSTEM AND ORDINARY FILES

❖ Cat:DISPLAYING AND CREATING FILES

It is mainly used to display the contents of a small file on the terminal.

cat , like several other UNIX commands,also accepts more than one filename as arguments.

In other words cat concatenates the two files-hence its name.

cat is also used to create files.

❖ Cp:Copying a File

The cp(copy) command copies a file or a group of files.It creates an exact image of the file on the disk with a different name.The syntax requires at least two filenames to be specified in the command line.When both are ordinary files, the first is copied into the second.

❖ Rm:Deleting Files

The rm(remove) command deletes one or more files.To delete all files in a directory the * is used to represent all the files to be deleted.

❖ Mv:Renaming Files

The mv command has two distinct functions:

It renames a file (or directory).

It moves a group of files to a different directory.

❖ Man:The man command displays its output a page at a time.To view the file enter the command with the filename.

❖ Wc:It is used for counting lines, words and characters.

❖ Cmp:This command is used for comparing two files.

❖ Comm:It requires two sorted files,and lists the differing entries in different columns.



**THANK
YOU**