



# **FUNDAMENTALS OF OPERATING SYSTEM**

**Presented by:Mr.Vipul Gamit**

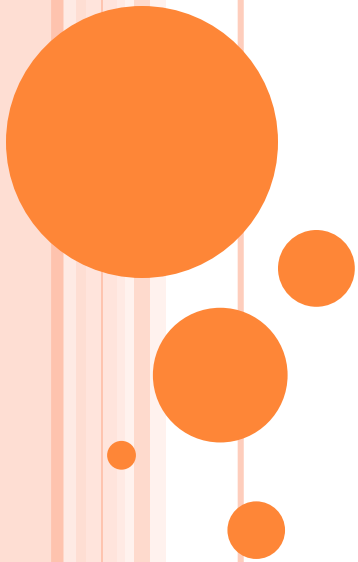
## Unit – 1

# Introduction of Operating Systems, File Systems and Management



## What is Operating System ?.

- *The software layer, nearest to hardware, which facilitates launching all the software utilities and applications is called the operating system.*
- *An Operating System is a program that acts as an intermediary between the user of a computer and the computer hardware.*



# PURPOSE OF OS

- To provide an environment in which user can execute programs in a convenient and efficient manner.

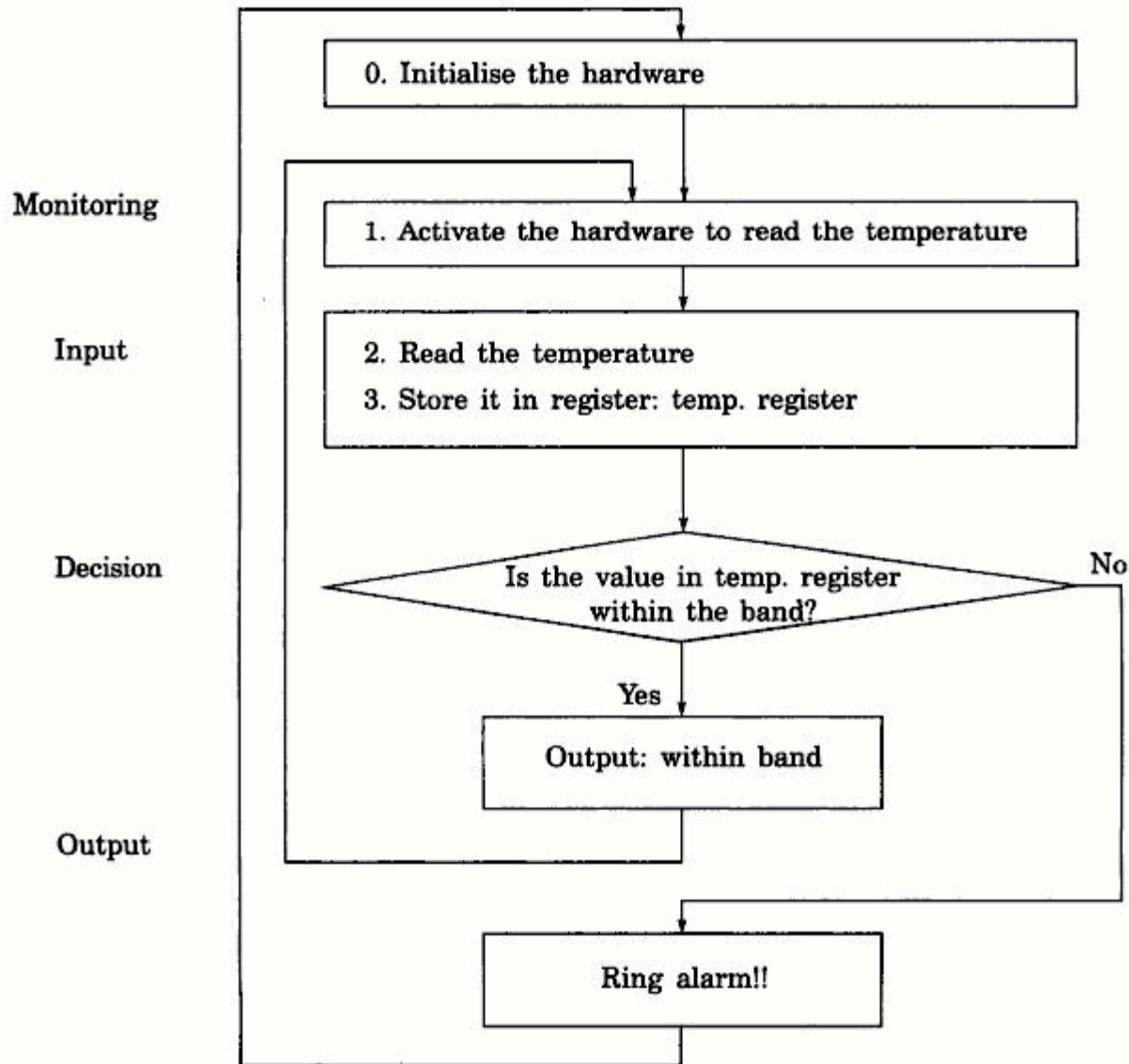


# FUNCTIONALITIES OF OS

- It offers a very wide range of general data services to input , output, store and process information in a computer system.
- To achieve this OS manages devices such as keyboard, display unit, processor, memory and other devices of i/p and o/p.

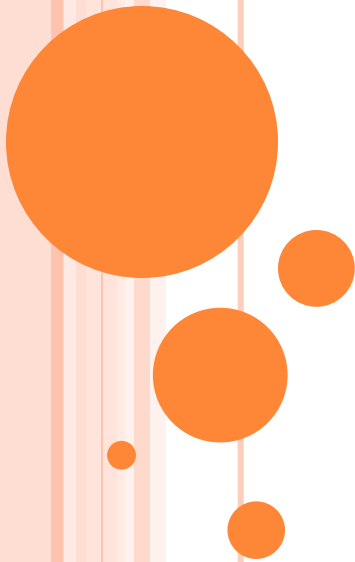


## A real time control application:-



# A Real Time Control Application (cont..)

- **Above supervisory program has one task to accept input information by reading out a sensor periodically.**
- **The next task of this program is to process this data.**
- **The final task is to prepare an output.**



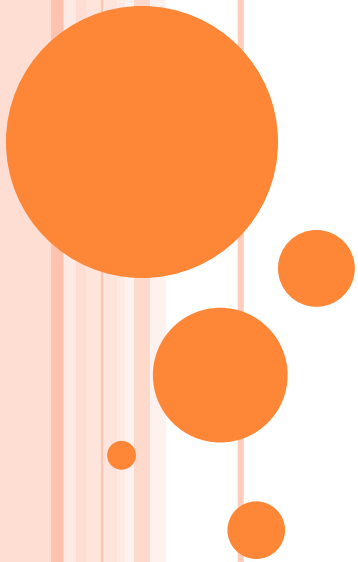
# A REAL TIME CONTROL APPLICATION (CONT..)

- The cycle of operation is repeated in the following sequence:
  1. Input task
  2. Process data
  3. Output task





- OS provide that software layer, in a computer system which schedules all the operations and manages all the computing resources as well.



# What does an OS provides?

- **User access to the system**
- **Storage and management of information including protection against an accidental or intentional misuse.**
- **Support for data processing activity**
- **Communication with a variety of IO devices.**
- **Management of all the activities in a transparent manner in multi-user operating system. Users are unaware of the presence of other users.**

# WHAT DOES AN OS DO ?

- On power-on, the OS makes sure that the system has a certain initial set-up.
- On a request from a user application, for a resource or for IO, the OS makes appropriate access checks & the requested facility or service is either made available or denied.
- Since multiple-user applications are supported at any time of operation, the OS makes sure that each user, or application, gets the resources they require. This means that OS must schedule the demands on system resources.



## WHAT DOES AN OS DO ? (CONT..)

- In case an error occurs during an operation, it should be logged & recorded. Basically, the OS records the sequence of events that lead to the error condition.
- The OS supports operations & services that require communication over a network.
- (Optional) The OS supports some deadlines so that safety critical operations run & fail gracefully.




# AN OPERATIONAL OVERVIEW

- To understand the management functions provided within an OS, to manage various resources.
- Primary resources are :-
  - processor
  - memory
  - I/O devices



# AN OPERATIONAL OVERVIEW (CONT..)

## Processor

- OS ensures that each user gets sufficient processor time.
  - Processor time needs to be shared.
  - To schedule & allocate processor time for every user & for every running application.
  - A computer executes programs & accesses data from its main memory.
- 

# AN OPERATIONAL OVERVIEW (CONT..)

## Memory

- A computer executes programs & accesses data from main memory.
- Main memory (Volatile).
- Another level of storage(secondary memory)-disk & tapes.
- They are Non-volatile.



# AN OPERATIONAL OVERVIEW (CONT..)

## Memory (cont..)

- For enhancing throughput is to have an intermediate storage which can take up different rates or timings of Input/Output.
- A processor access registers at a speed which is far in excess of the main memory access.
- Therefore, one choose to place cache at an intermediate level of storage between CPU & main memory.





# AN OPERATIONAL OVERVIEW (CONT..)

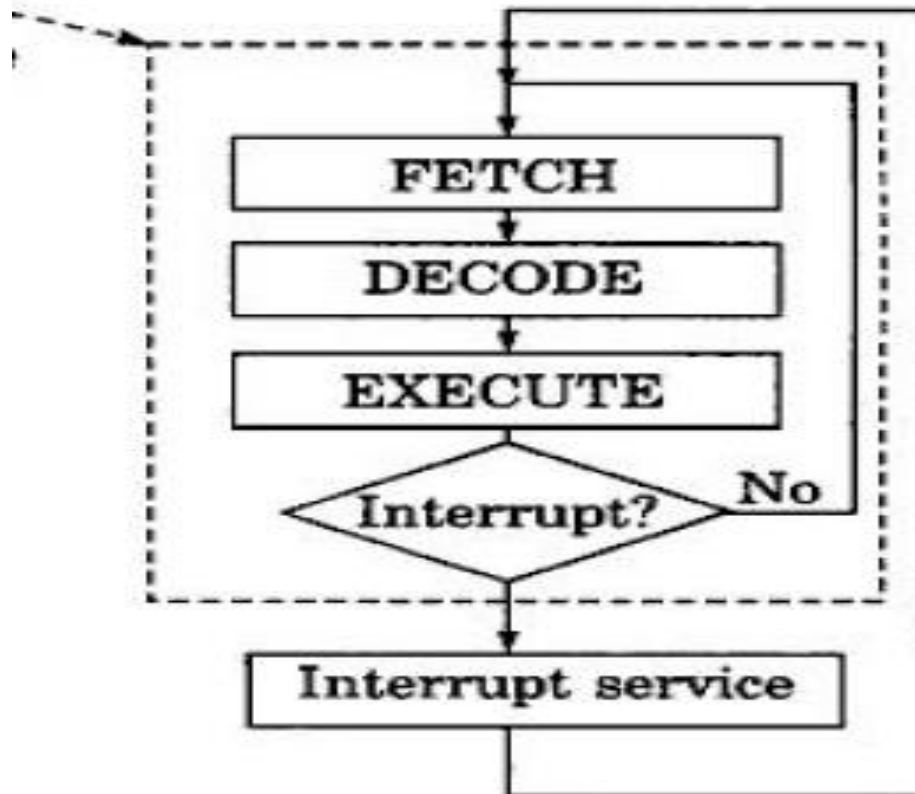
## I/O devices

- User interaction takes place through an I/O device.
- OS provide utilities for all the required communication with I/O devices.



# AN OPERATIONAL OVERVIEW (CONT..)

## I/O devices (cont..)



# AN OPERATIONAL OVERVIEW (CONT..)

## I/O devices (cont..)

- Interrupt mode of data transfer is used wherever a user, or a critical application, requires immediate attention.
- Interrupts are not used to handle large amount of data transfers.
- Solution to problem, used DMA(Direct Memory Access) mode.



# AN OPERATIONAL OVERVIEW (CONT..)

## I/O devices (cont..)

- 2 Stages of DMA mode of data transfer :-
  - Data transfer requirement – setup(how much data to transfer & where it is located)
  - After setup, device & processor compete with each other to seek memory access.



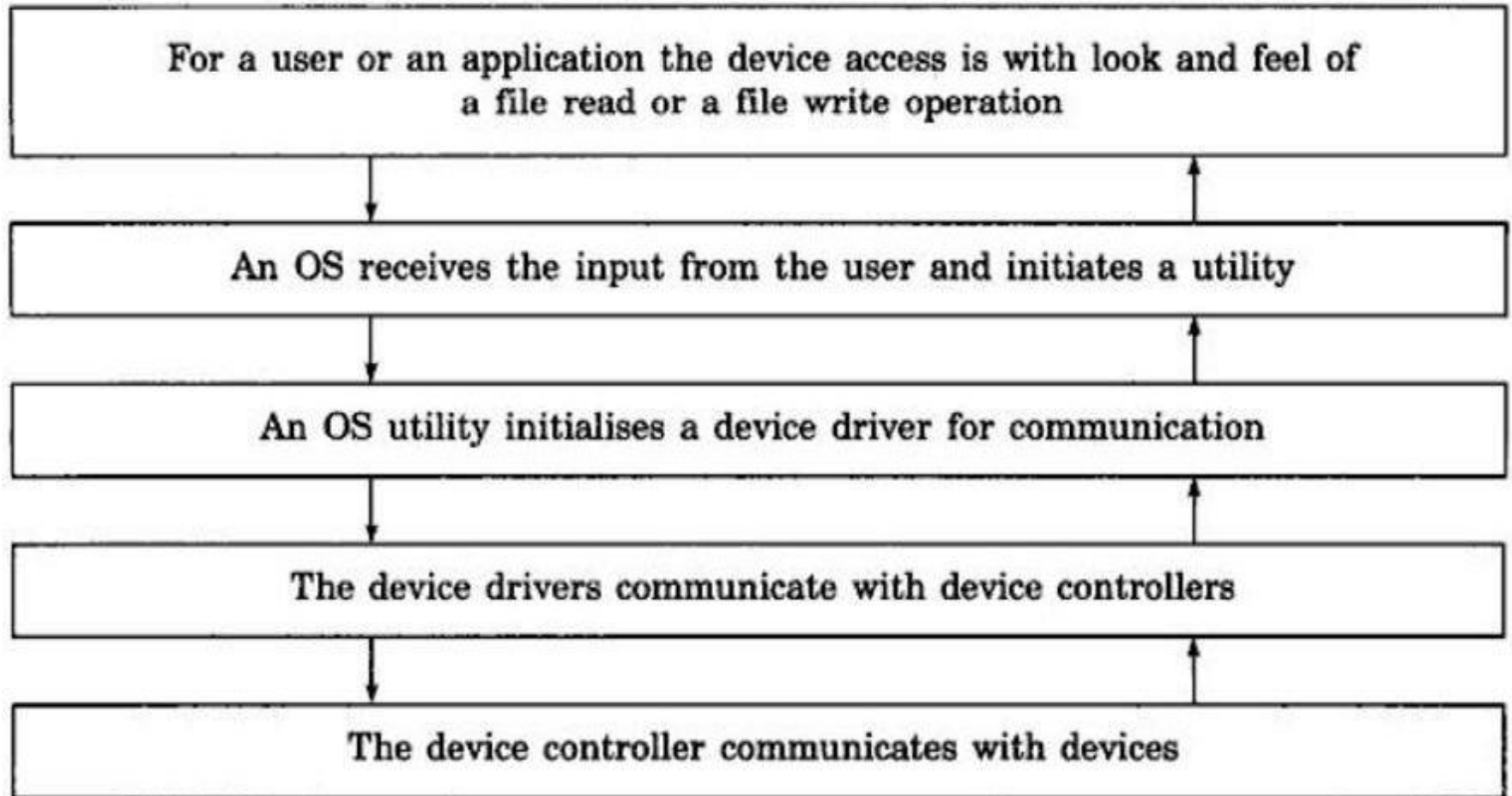
## AN OPERATIONAL OVERVIEW (CONT..)

### Communication with devices :-

- A computer may be connected to variety of devices(keyboard, monitor, printer, modem)
- OS provide device service utilities – that cater requirements of individual device characteristics.
- Utilities supported by OS – execute a sequence of instructions stored in a device driver.



## AN OPERATIONAL OVERVIEW (CONT..)



# AN OPERATIONAL OVERVIEW (CONT..)

## Mutual Exclusion

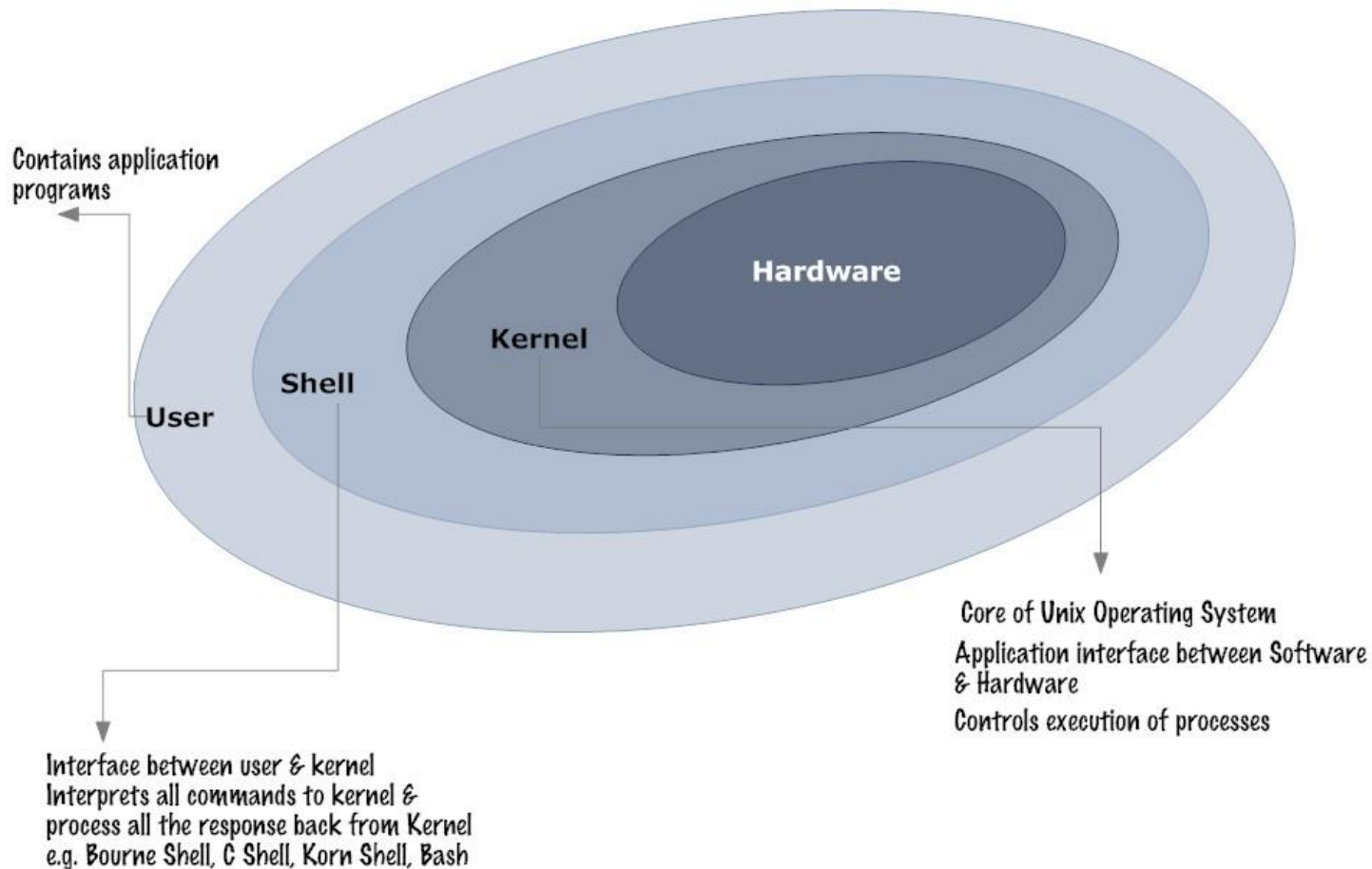
- Problem posed by devices is mutual exclusion.
- Two users want to output on printer which shared then may both documents content is interleaved.
- So OS need to schedule the use of shared resources.



# AN OPERATIONAL OVERVIEW (CONT..)

## Elements of Unix Operating System

### Unix Architecture Layers





# AN OPERATIONAL OVERVIEW (CONT..)

## Shell of an OS

- A user never communicates directly with the OS service routines.
- OS service routines are resident in a kernel which cannot be directly accessed by a user program or by an application.
- OS services are sought by a user via a command interpreter provided by a shell around a kernel.

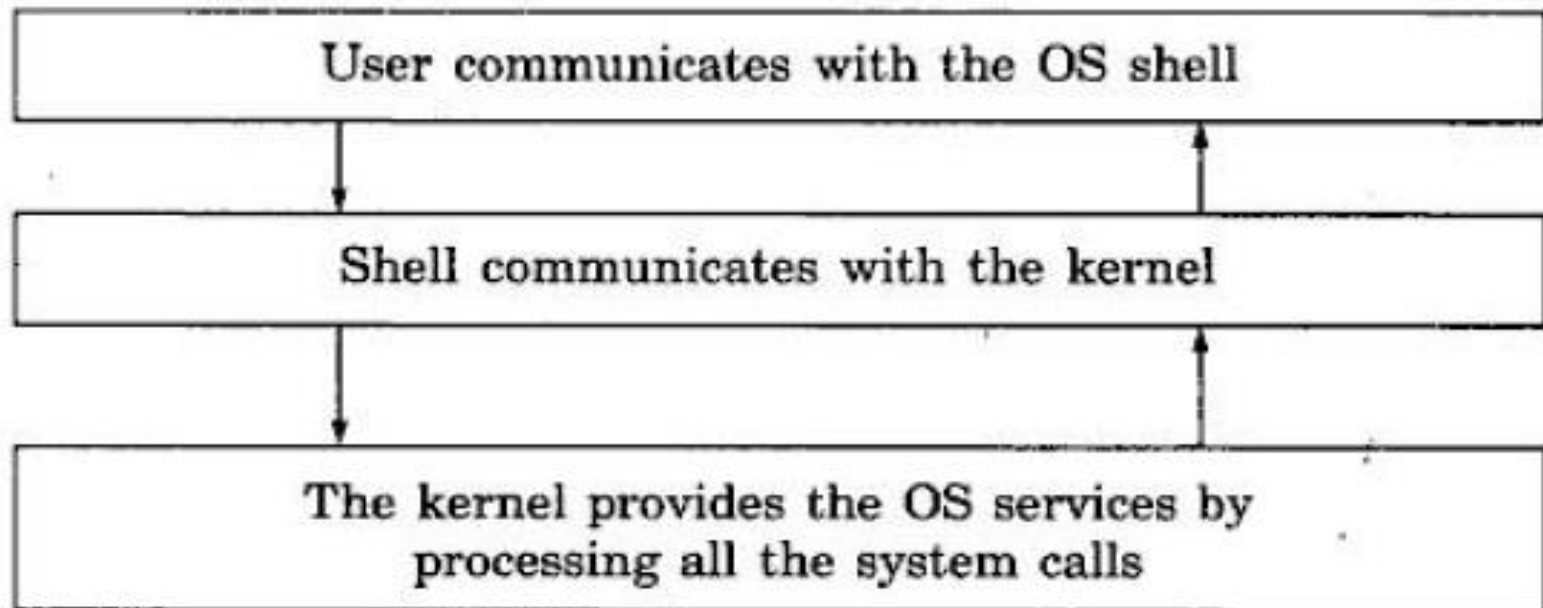


## AN OPERATIONAL OVERVIEW (CONT..)

- The shell interprets the OS commands & elicits OS services by communicating with the kernel.
- Thus a shell protects the system service routines



## AN OPERATIONAL OVERVIEW (CONT..)



Shell acts as a command interpreter. The kernel is protected from direct user access.



# AN OPERATIONAL OVERVIEW (CONT..)

## HCI(Human Computer Interface)

- Notice that we have a set of icons presented in Windows or Unix operational environment.
- This icons invite user to launch the application.
- Unix users operate in command driven environment.



# PROCESSES & TOOLS

## Processes

- A program in execution – known as process.
- With multiple applications active at the same time there are many processes that are active.
- OS needs to manage all processes.
- Often applications may create processes that need to communicate with each other. This is called interprocess communication.



# PROCESSES & TOOLS (CONT..)

## Processes

- In network one machine seeking services called the client machine & machine offering services called server.



# PROCESSES & TOOLS (CONT..)

## Tools

- Most OS provide many other general purpose utilities – as a set of packaged tools.
- Tools- helps to think in terms of higher level of operations.
- In absence of tools & toolkits we required to write fresh program each time to accomplish the task.



# PROCESSES & TOOLS (CONT..)

## Tools

- Sort utility in Unix
- Zip tools in windows environment





# FILE SYSTEMS AND MANAGEMENT



33

**Chapter:2**

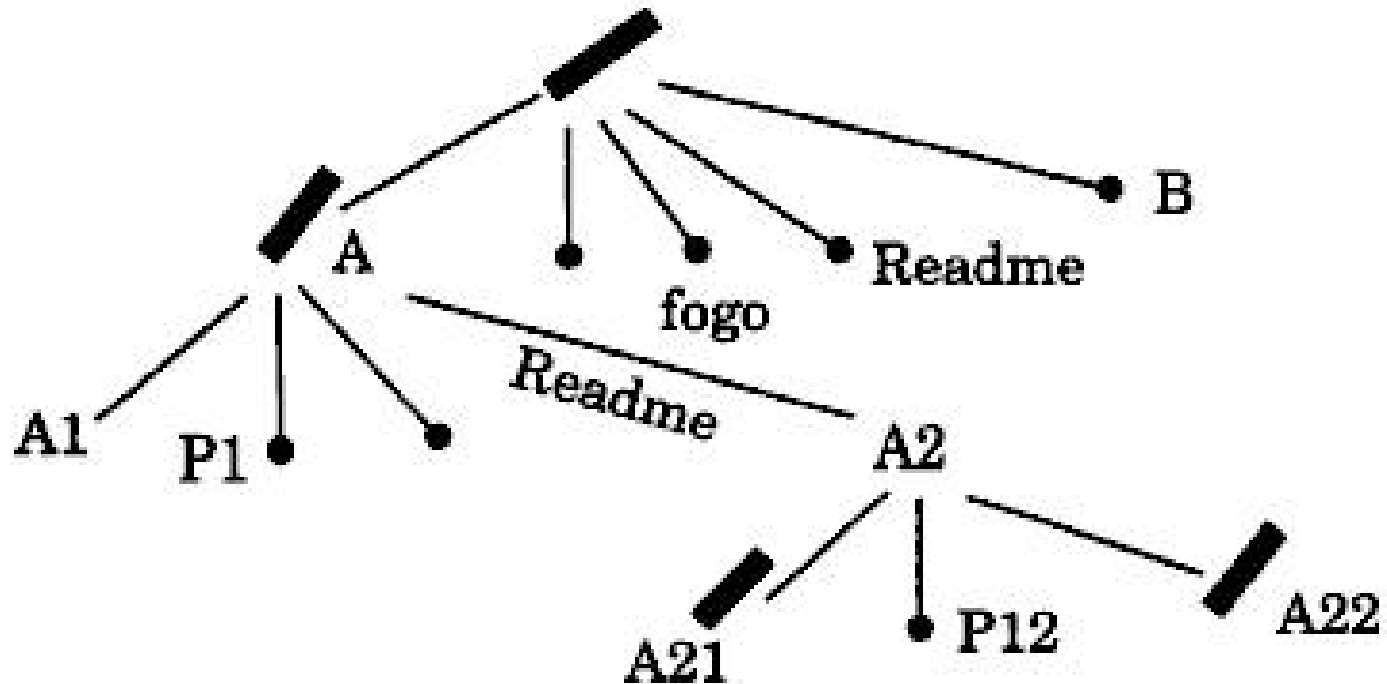
# FILE

- Any organized information is a file.
- E.g. Student Roll No.
- File contains sequence of bit(or byte).

# FILE SYSTEM

- A file system is that software which allows users and applications to organize their files.
- The organization of information may involve access, updates, and movement of information between devices.
- Files are organized in tree structure in directory.

## FILE SYSTEM (CONT...)



# FILE TYPES

- Files tend to use file type information within a name.
- File descriptor – A file descriptor is the value containing information necessary to use a file : a pointer to the file, the access rights, the access modes (read or write), the current position in the file, etc.
- A file descriptor is kept within the file structure and is often used by the file system software to help OS provide management services.

## FILE TYPES (CONT..)

- In windows, if we create a file suppose using notepad or word and save it.
- Then an icon appears with file
- This icon of specific application environment is displayed by OS represented the type of the file created.
- Now , to open this file we just double click on file or other way right click on file icon and select open.

## FILE TYPES (CONT..)

- For a user the extension in the name of a file helps to identify the file type.
- When a user has a very large number of files, it is very helpful to know the type of a file from its name extensions.

# FILE TYPES (CONT..)

<i>Usage</i>	<i>File extension used</i>	<i>Associated functionality</i>
An ASCII text file	.txt, .doc	A simple text file
A word-processing file	.wp, .tex	Usually for structured documents
Program files	.c, .p, .f77, .asm	C, Pascal, Fortran, or assembly code
Print or view	.ps, .gif, .dvi	Printing and viewing images, documents
Scripting	.pl, .BAT, .sh	For shell scripts or Web CGI
Program library	.lib	Library routines in packages
Archive generation	.arc, .zip, .tar	Compression and long-term storage
Files that execute	.exe, .out, .bin	Compiler generated executable files
Object codes	.o	Often need linking to execute



# FILE ATTRIBUTES

- In addition to the file types, a file system must have many other pieces of information that are important which are termed as File Attributes.
  - location at which file available on disk
  - size of file
  - when created, i.e. date and time of creation
  - owner of the file
  - access permissions on file(i.e. read, write, execute)

# FILE OPERATIONS

- Files are stored in secondary storage(physical view of a file).
- A file system(as a layer of software) provides a logical view of files to a user or to an application.
- OS gets all the information it needs to physically locate, access, and do other file-based operations whenever needed.

## FILE OPERATIONS (CONT..)

- From operational point of view, a user should be able to create a file.
- We will considered him/her as owner of the file.

CON..

- The owner can save and store the file.
- He should be able to read the contents or even write into the file.
- He should also be able to display, rename, and append to this file.
- He may even wish to copy or even delete the file.

## FILE OPERATIONS (CONT..)

- OS provides service for operations such as below
  - indicating who else has an authorization of an access to read or write or execute this file.

# FILE OPERATIONS (CONT..)

<i>Usage</i>	<i>Editor-based operation</i>	<i>OS terminology and description</i>
Create	Under FILE menu NEW	A CREATE command is available with explicit read/write option
Open	Under FILE menu OPEN	An OPEN command is available with explicit read write option
Close	Under FILE menu CLOSE Also when you choose QUIT	A file CLOSE option is available
Read	Open to read	Specified at the time of open
Write	Save to write	Specified at the time of open
Rename or copy	Save as file name	Can copy using a copy command
Cut and Paste	Via a buffer	Uses desk top environment CDE
Join files		Concatenation possible or uses an append at shell level
Delete	Under FILE use delete	Use remove or delete command
Relocate		A move command is available
Alias		A symbolic link is possible
List files	OPEN offers selection	Use a list command in a shell

## FILE OPERATIONS (CONT..)

- One may need information on file sizes.
- One may wish to determine the number of lines, words or characters in a file.
- For such requirements, a shell may have a suite of word counting programs.
- File names may bear a common stem to help us categorize them.
- For e.g. I tend to use prog for my program text files.

## FILE OPERATIONS (CONT..)

- A programmer derives considerable support through use of regular expressions within file names.
- Use of a regular expression enhances a programmer's productivity in checking or accessing file names.



# FILE OPERATIONS (CONT..)

<i>Usage</i>	<i>Unix shell command</i>	<i>MS DOS command</i>
Copy a file	cp	COPY
Rename a file	mv	RENAME
Delete a file	rm	DEL
List files	ls	DIR
Make a directory	mkdir	MKDIR
Change current directory	cd	CHDIR

## FILE OPERATIONS (CONT..)

Makes a copy of file 'debian' and call it 'Debian'

```
$ cp -i debian Debian
```

Rename file 'unix' to 'Unix'

```
$ mv -i unix Unix
```

Move file Unix from your home directory to /tmp.

```
$ mv -i newhw/hw1 oldhw
```

```
$ rm -i prgSum.c
```

```
$ mkdir foo
```

```
$ ls -l
```

```
$ cd foobar
```

# FILE OPERATIONS (CONT..)

## The ls command

Unix's ls command which lists files and subdirectories.

11/15/2017

-a	list hidden files
-d	list the name of the current directory
-g	show group ownership of file in long listing
-i	print the inode number of each file
-l	long listing giving details about files and directories

# FILE OPERATIONS (CONT..)

## Using regular expressions

- Most OS allow use of regular expression operators in conjunction with the commands.
- One may input a partial pattern and complete the rest by a \* or a ? Operator.
- This not only saves on typing but also helps you when you are searching a file after a long time gap and you do not remember exact file name completely.
- C\*p\*2 or \*2\*
- my\_prog?

# FILE OPERATIONS (CONT..)

## Using regular expressions

- Regular Expression is a set of Characters that specify a pattern.
- Used when you want to search for specific lines of text containing a particular pattern.

Pattern	Matches
<code>^A</code>	"A" at the beginning of a line
<code>A\$</code>	"A" at the end of a line
<code>A^</code>	"A^" anywhere on a line
<code>\$A</code>	"\$A" anywhere on a line
<code>^^</code>	"^" at the beginning of a line
<code>\$S</code>	"\$" at the end of a line

# FILE OPERATIONS (CONT..)

## Using regular expressions

- When same file needed in two separate directories.
- First solution : Keep two copies, one in each directory.
- Second solution : Use ln command for creating symbolic link regardless of directory locations in file system.
- But a shared file link should be in the same disk partition.

# FILE ACCESS RIGHTS

## Access permission:-

- A file system manages access by checking a file's *permissions*.
- *File* may be accessed to *read* or *write* or *execute*.
- The usage is determined by the context in which it is created.
- Example:-  
Consider a *file supporting an application of bus schedules*.  
It shall contain a *bus time-table with the following* restrictions
  - *read-only permission for the general public,*
  - *read, write permission for the supervisor,*
  - *read, write and execute permissions for the management.*

# FILE ACCESS RIGHTS (CONT..)

## Who all can access a File?

- Unix recognizes three category of users – *user/owner*, *group* and *others*.
- Owner may be a person or a program or an application or a system based utility.
- The notion of a group comes from the software engineering team operations.
- Others has the public usage.
- Organization of this information is as 9 bits

*r w x r w x r w x*

for each owner, group and others where each *r w x* is an octal number, e.g 100 100 100 gives read permission for owner, group and others.



# FILE ACCESS AND SECURITY

In Unix, owner of a file can change its permissions using `chmod` command which has a syntax as follows:

`chmod pattern fileName` as in `chmod 644 myFile`

e.g : 644 corresponds to the pattern `rw-r--r--`

664 corresponds to the pattern `rw-rw—r--`

`ls -l` command displays the access permissions of a file.

# SECURITY CONCERNS

- File permissions are the most elementary and effective form of security measure in a stand alone single user system.
- Some systems provide security by having passwords for files.
- Enhanced security would be to encrypt a file with some key.
- Unix provides crypt command to encrypt files.  
Syntax is :

*crypt EncryptKey <InputFile> OutputFile*

# INFORMATION REQUIRED FOR FILE MANAGEMENT

Nature of Information	Its significance	Its use in management
File Name	Chosen by its creator user or program.	To check its uniqueness within a directory
File Type	Text, binary, program etc.	To check its correct usage
Date of creation and last usage	Time and date	Useful for recording identity of user(s) also
Current usage	Time and date	Identity of all current users
Back-up information	Time and date	Useful for recovery following a crash

# INFORMATION REQUIRED FOR FILE MANAGEMENT

11/15/2017

Permission	rxw information	Controls read, write, execute, useful for network access also
Starting address	Physical mapping	Useful for access
Size	User must allocate within allocated space.	Internal allocation of disc blocks
File Structure	Useful in data manipulation	To check its usage

# FILE STORAGE MANAGEMENT

- An OS needs to maintain several pieces of information for file management, e.g. access and modification times of a file.
- *Audit Trail gives who accessed which file and did what.*
- In Unix, these trails are maintained in the *syslog file*
- useful to recover files after a system crash and to detect unauthorized accesses to a system.

# FILE STORAGE MANAGEMENT

- Unix internally recognizes 4 different file types  
*(1)ordinary(2) directory(3) special and (4)named.*
- Ordinary files are those created by the users programs or utilities.
- Directory files organize the files hierarchically, they are different from ordinary files.
- *Inode* structure is used by Unix to maintain information about the named files.



# INODE STRUCTURE IN UNIX

11/15/2017

File Type	16 bit information Bits 14-12 : File type (ordinary, directory, character etc.) Bits 11-9 : Execution flags. Bits 8-6 : Owners r w x information Bits 5-3 : Groups r w x information Bits 2-0 : Others r w x information
Link Count	Number of symbolic references to this file
Owner's ID	Log in ID of the person who owns this file.
Group's ID	Group ID of the user

# INODE STRUCTURE IN UNIX

File Size	Expressed in number of bytes.
File Address	39 bytes of addressing information.
Last access to file	Date and time of last access.
Last modified	Date and time of last modification.
Last inode modification	Date and time of last inode modification.



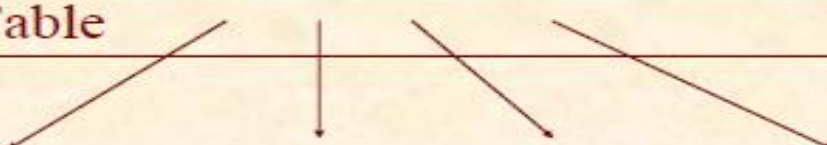
# FILE CONTROL BLOCKS

- The Microsoft counterpart of an inode is a File Control Block (FCB).
- The FCBs store file name, location of secondary storage, length of a file in bytes, date and time of its creation etc.

# FILE CONTROL BLOCKS (CONT..)

11/15/2017

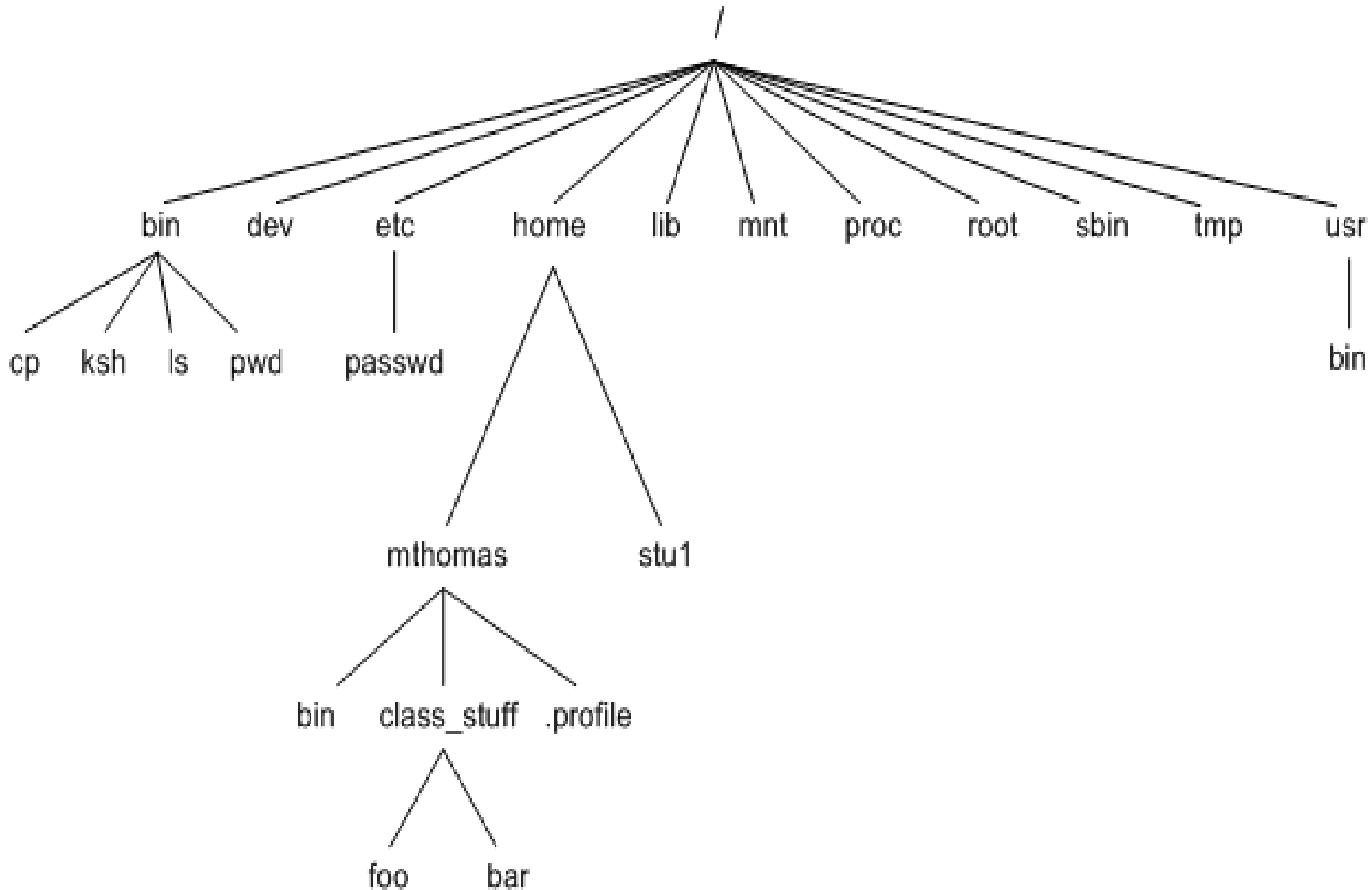
## Organization of Inodes (Similarly for FCBs)

Label for the disc	System
Boot Area	Information
System Boot Information	
Inode Table	
 Data block area (Inodes point to this area)	
Back up area	

# THE ROOT FILE SYSTEM

- When an OS is installed initially, it creates a root file system.
- The OS not only ensures but also specifies how the system and user files shall be distributed for space allocation on the disk storage.
- Almost always the root file system has a directory tree structure.
- In OS with unix flavours the root of the root file system is a directory.
- The root is identified by “/”. The root file system has several subdirectories.

# THE ROOT FILE SYSTEM (CONT..)



## CONVENTIONS FOLLOWED IN UNIX

- Subdirectory *usr* contains binaries sharable by system and user - *read-only usage mode*.
- Executable programs are found in subdirectory *bin* found at any level.
- Subdirectory *sbin* contains some system *executable* files – used during boot time and power on.

# CONVENTIONS FOLLOWED IN UNIX (CONT..)

- Subdirectory *lib* contains libraries – found at several places in the file system. Subdirectory *etc* contains host related files – has many subdirectories to store configuration (*config*), *internet (hosts)* and *device (dev)* information.
- Subdirectory *mnt* contains device mount information (in Linux).
- Subdirectory *tmp* contains temporary files created during file operation.
- Subdirectory *var* contains files that have variable data  
E.g. *mail* and *system log* contain variable data. It may also have subdirectories for spools.

# CONVENTIONS FOLLOWED IN UNIX

## (CONT..)

- Each user *name* will find his home directory as */home/name*
- Subdirectory *include* contains all C header files.

# THE ROOT FILE SYSTEM (CONT..)

- The OS creates disk partition to allocate files for specific usages.
- A certain disk partition may have system files and some others may have other user files or utilities.
- The system files are usually programs that are executable with **.bin** in unix and **.exe** extension in MS environment.
- In unix Subdirectories are like:- **/usr, /bin, /sbin, /lib, /etc/hosts, /etc/config, /mnt, /tmp, /var.**

## ADVANTAGE:-

- The system knows exactly where to look for some specific routines.
- Users can customize their operational environment by providing a definition for an environment variable **PATH** which guides the sequence in which the OS searches for the commands.



# BLOCK-BASED FILE ORGANISATION

- Data transfer take place from disks in blocks as large as 512 or 1024 bytes at a time. Any file which a user generates moves in blocks.
- Each operating system has its own block management policies. These policies map each linear byte stream into disk blocks.
- **NOTE:-** A policy on storage management can heavily influence the performance of a file system.

# CONTIGUOUS ALLOCATION

- Let us assume we know apriori(well in advance) the size of files before their creation.
- So this information can always be given to OS before a file is created and the OS can simply make space available.
- In such situation it is possible to follow pre-allocation find a suitable starting block so that the file can be accommodated in a contiguous sequence of disk blocks.

# CONTIGUOUS ALLOCATION (CONT..)

## Advantage

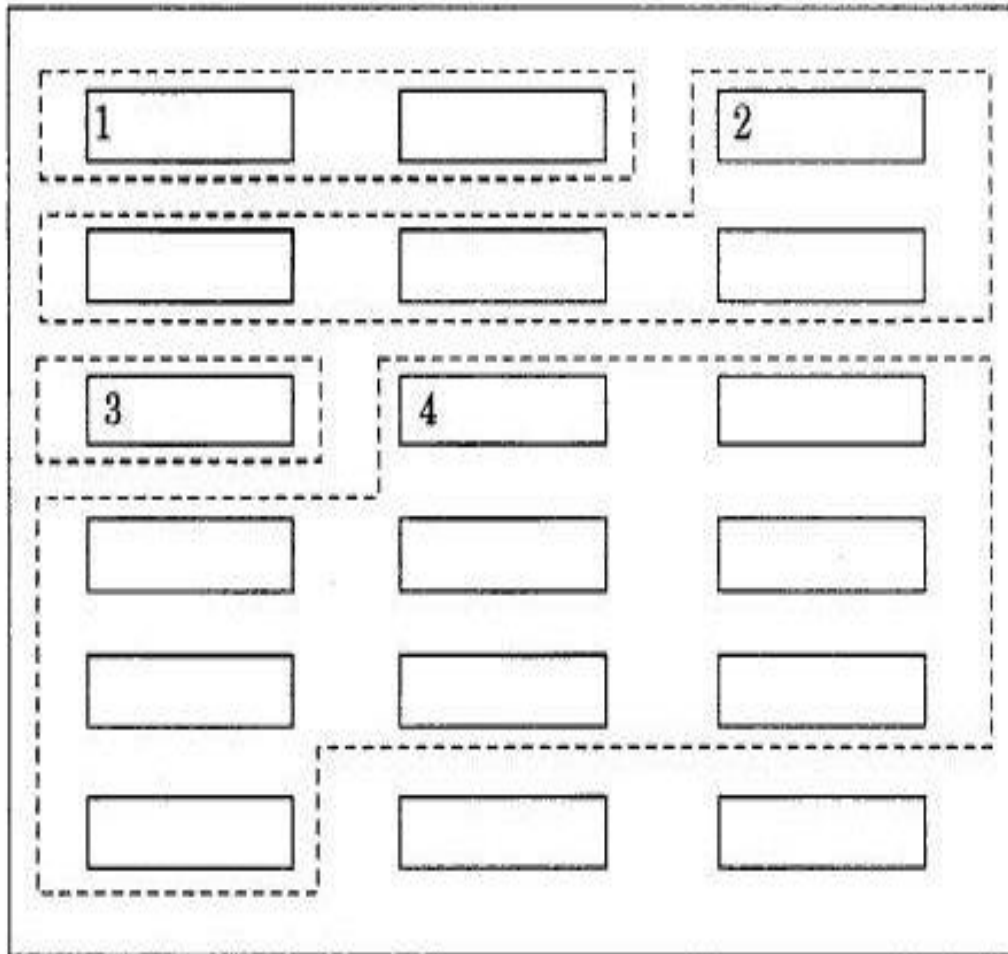
Retrieval of information is very fast.

## Disadvantages

Requires apriori knowledge.

It leaves no room for changes.

# CONTIGUOUS ALLOCATION (CONT..)



File\_1 (size 1145 bytes)

File\_2 (size 4060 bytes)

File\_3 (size 700 bytes)

File\_4 (size 9000 bytes)

# CHAINED ALLOCATION:- (DYNAMIC BLOCK ALLOCATION)

## Why we need this type of allocation?

- .
- It is not always possible to know apriori the size of file being created
- There are some files that already exist and it is not easy to find contiguous region(area).
- For example, there may be enough space available on disk but you may not find a single large enough chunk to accommodate an incoming file.

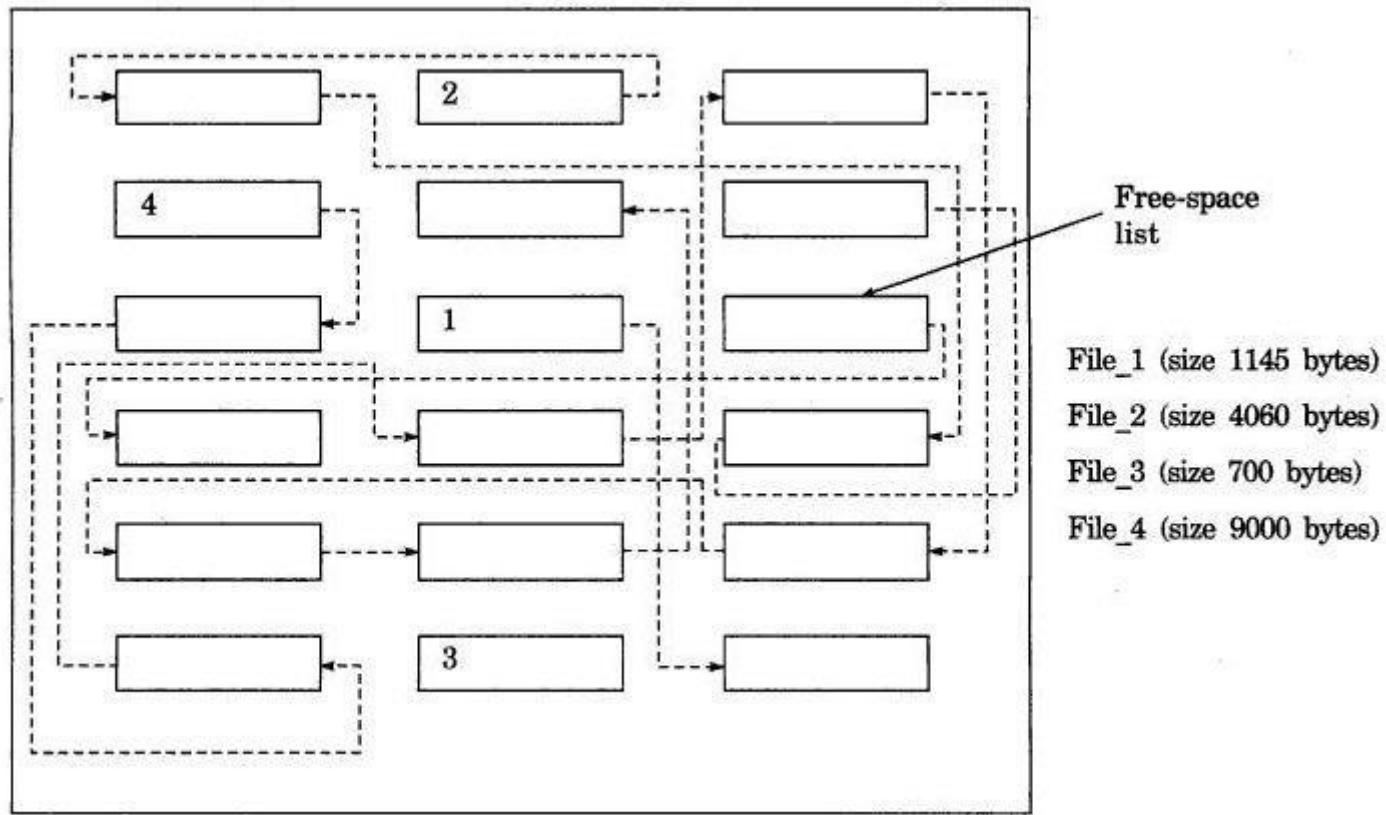
# CHAINED ALLOCATION (CONT..)

## How allocation is done?

- A list of free blocks is maintained. Allocation is made as need arises. We may allocate a block at a time from a free space list.
- The OS maintains a chain of free blocks and allocates the next free block in the chain to an incoming file. This way the finally allocated files may be located at various positions on the disk.
- **Advantage:-**  
Free space management system(no waste space).  
Simple because need only starting address.
- **Disadvantage:-**  
Random access to blocks is not possible.  
Overhead - Maintenance of chained links.

# CHAINED ALLOCATION (CONT..)

11/15/2017



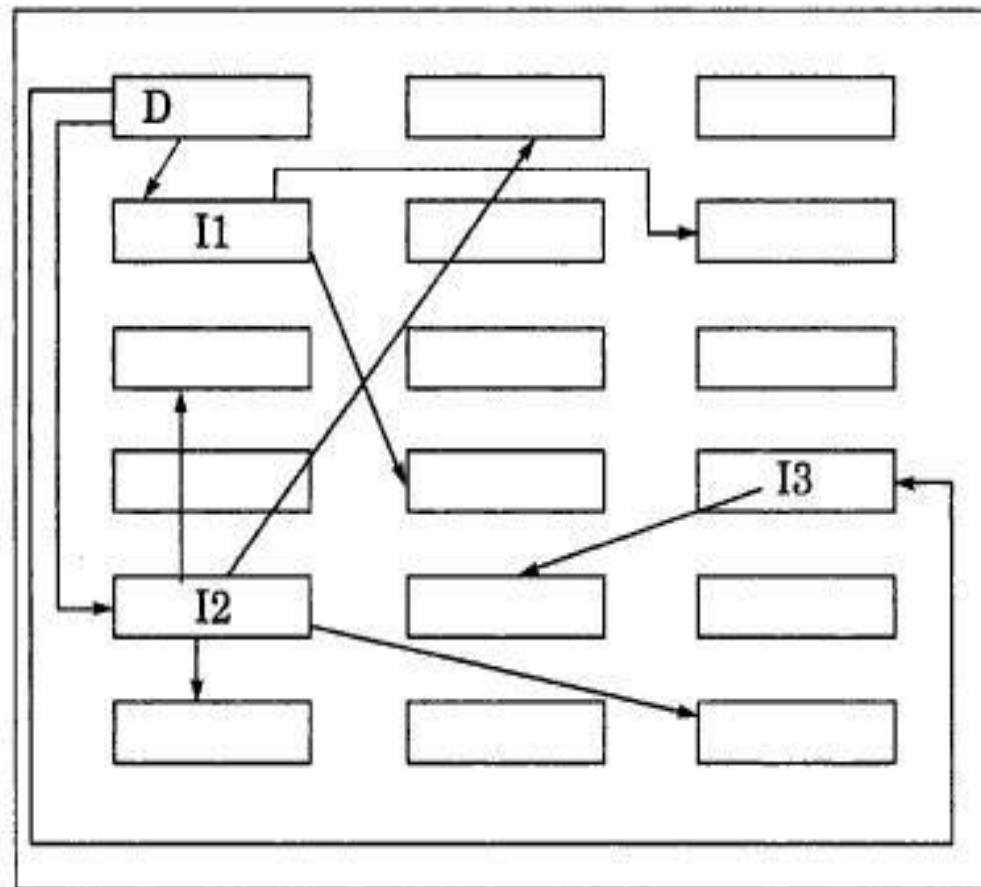
# INDEXED ALLOCATION

- We maintain an index table for each file in its very first block.
- So, it is possible to obtain the address information for each of the blocks with only one level of indirection.
- **Advantage** over dynamic allocation method is a direct access to every block of file.



# INDEXED ALLOCATION (CONT..)

11/15/2017



File\_1 (size 1145 bytes)

File\_2 (size 4060 bytes)

File\_3 (size 700 bytes)

# INTERNAL & EXTERNAL FRAGMENTATION

## Internal Fragmentation

- In mapping byte streams to blocks we assumed a block size of 1024.
- In previous example, file-1 of size 1145 byte was allocated two blocks.
- The two blocks together have 2048 bytes capacity.
- We will fill the first block completely; the second block will remain mostly empty as only 121 bytes out of 1024 bytes can be used.
- As the assignment storage is by blocks in size of 1024 bytes, the remaining bytes in the second block cannot be used. Such non-utilization of space caused internally (as it is within a file's space) is termed ***internal fragmentation***.

# INTERNAL & EXTERNAL FRAGMENTATION (CONT..)

## External Fragmentation

- After a number of file insertions and deletions or modifications the free-space list becomes smaller in size.
- For instance, we have a file which was initially spread over 7 blocks. Now after few edits the file needs only 4 blocks.
- This space of 3 blocks which got released is now not connected anywhere. It is not connected with the storage list either.
- As a result we end up with a hole of 3 blocks which is not connected anywhere. After many file edits and operations, many such holes of various sizes get created.

# INTERNAL & EXTERNAL FRAGMENTATION (CONT..)

## External Fragmentation

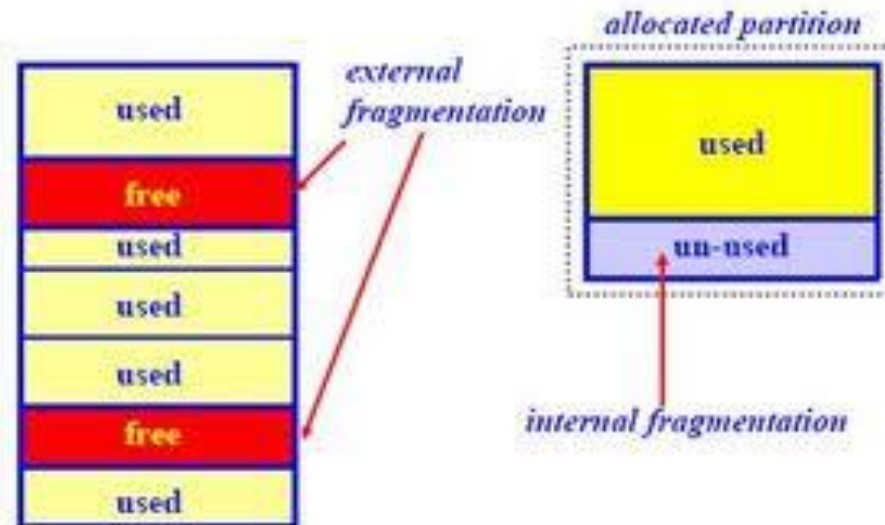
- Suppose we now wish to insert a moderately large-sized file, thinking that adequate space should be still available.
- Then it may happen that the free-space list has shrunk so much that enough space is not available. This may be because there are many unutilized holes in the disk. Such non-utilization, which is outside of file space, is regarded as *external fragmentation*.

# INTERNAL & EXTERNAL FRAGMENTATION (CONT..)

- A file system therefore, must periodically perform an operation to rebuild free storage list by collecting all the unutilized holes and linking them back to free storage list. This process is called compaction.
- When you boot your system, often the compaction gets done automatically. This is usually a part of file system management check.
- Sometimes this is also referred to as a ***garbage collection***.

# INTERNAL & EXTERNAL FRAGMENTATION (CONT..)

11/15/2017



# DISK PARTITION

- It allows better management of disk space.
- In most cases the disk partitions are created at the time the disk is formatted.
- So a formatted disk has information about the partition size.
- Every partition has its own file system management information.
- This information is about the files in that partition which populate the file system.

## DISK PARTITION (CONT..)

- Unix ensures that the partition for the system kernel and the users files are located in the different partition (or file systems).
- Unix systems identify specific partitions to store the root file system, usually in root partition.
- Another use in the PC world is to house two different operating systems, one in each partition.



## DISK PARTITION (CONT..)

- A disk partition, with all it's essentially a set of organized information. It has its own directory structure(tree structure).
- This tree gets connected to some node in overall tree structure of the file system and forks out.
- The network file system may have remote partitions which are mounted on it. It offers seamless file access as if all of the storage was on the local disk.

# PORTABLE STORAGE

- There are external media types like tapes, media, floppies.
- They can be physically ported.
- Most file system recognize these as online files when these are mounted on an IO device like tape drive or floppy drive.
- UNIX treats these as special files.
- PCs and MAC OS show an icon when these are mounted.

## PORTABLE STORAGE (CONT..)

- Files are the entities that users deal with all the time.
- Users create files, manage them, and seek system support in their file management activity.

# FLASH MEMORY

- Flash memory is a type of electrically erasable programmable read-only memory (EEPROM).
- The name comes from how the memory is designed -- a section of memory cells can be erased in a single action or in a "flash".
- Flash memory is a non-volatile computer storage chip that can be electrically erased and reprogrammed.
- It is used as an extension to secondary memory.
- It is used as an external memory.

## THE EVOLUTION:-

- It was the discovery from TOSHIBA.
- Flash memory is preferred media storage in hand-held and embedded systems.
- Flash memory cards used for digital cameras, camcorders, cellular phones, video games networking hardware, and PC cards.

# FLASH MEMORY (CONT..)

The development of flash memory in last two decades

<i>Year</i>	<i>The nature of development</i>
Mid 1970s	Intel developed and perfected EEPROMs
1984	Toshiba announced first NOR flash memory, <i>inventor</i> : Fuji Masuoka
1987	Toshiba developed first NAND flash memory
1992	First memory cards came in market to support ROM-less CPUs
1995	Memory cards incorporated with digital cameras
1996	Flash memory used as media card with PCs and other appliances like TV
1998	Flash memory delivered in multi-chip packages: bulk storage capability
2000 onwards	Bulk storage media for MP3 players, mobile handsets, consumer electronic appliances and also used for large file storages in servers like Web servers

# FLASH MEMORY (CONT..)

- Structure of flash memory:-

“cell” is capable of storing one bit(0 or 1) information, would be the smallest physical as well as logical structure. Cells are constructed from basic transistor circuitry.

With cell as a structural unit, it is possible to study characteristics that impact operational aspects like access for read, write or execute operations.

- There are two kinds of cells:-

NOR logic structure

NAND logic structure

respectively referred as NOR flash and NAND flash.

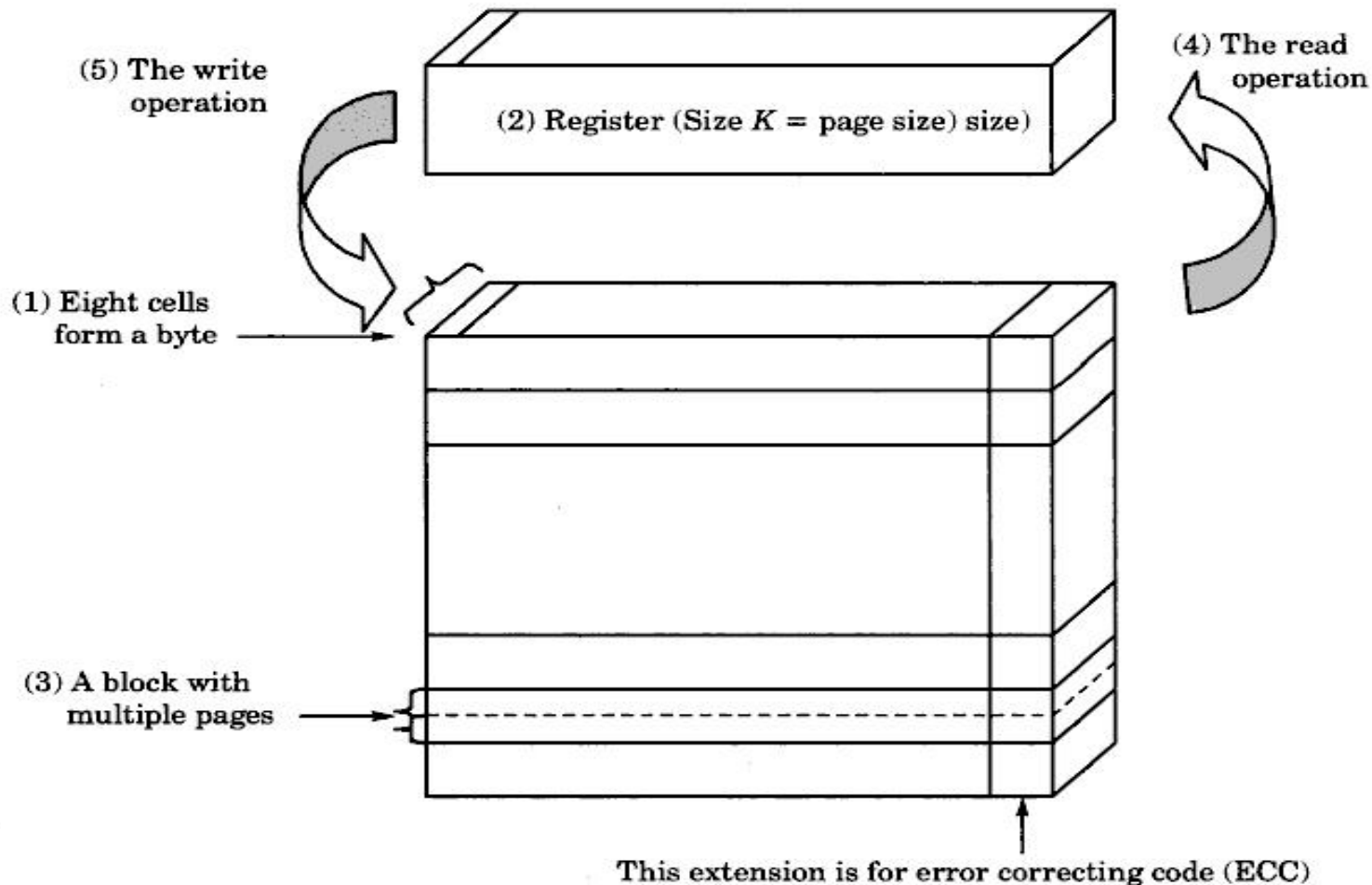
## FLASH MEMORY (CONT..)

- NAND flash is more commonly used for storing program codes and persistent data file, memory cards, USB flash drives, solid-state drives, and similar products, for general storage and transfer of data.
- NOR flash which allows true random access and therefore direct code execution, is used as a replacement for the older EPROM and as an alternative to certain kinds of ROM applications.



# ORGANIZATION OF NAND FLASH

Below figure shows how starting with a cell, pages and blocks are defined in NAND flash



# ORGANIZATION OF NAND FLASH.

1. First see how a group of eight cells form a byte.
2. Register size is same as the page size in the memory.
3. A block consists of multiple pages. Anywhere up to 64 pages may form a block.

The flash memory may have capacity of 256 MB to 32 GB.

4. and 5. the memory controller regulates serial read/write operations.

## ORGANIZATION OF NAND FLASH (CONT..)

- Bulk read/write is at page level.
- It is possible to maintain a hash-table to identify the block in which the page is located.
- A controller can be used to regulate and manage the traffic in and out of flash.
- *NOR flash*, the unit of read/write is a byte.
- NOR flash is a random access memory.
- NAND flash is accessed serial within a page.

## FLASH MEMORY (CONT..)

- Flash is extensively used in consumer electronic appliances.
- In PCs it is used either as a physically detachable storage device.
- Removable device (pen drive) uses flash.

## FLASH MEMORY (CONT..)

- Flash is used as an intermediate storage in the following three ways:-
  - As a store file system to boot the system.
  - As an intermediate storage for communication with secondary devices.
  - It is also used to store code and programs that are frequently accessed in an embedded system.

# FLASH USED TO STORE FILE SYSTEM

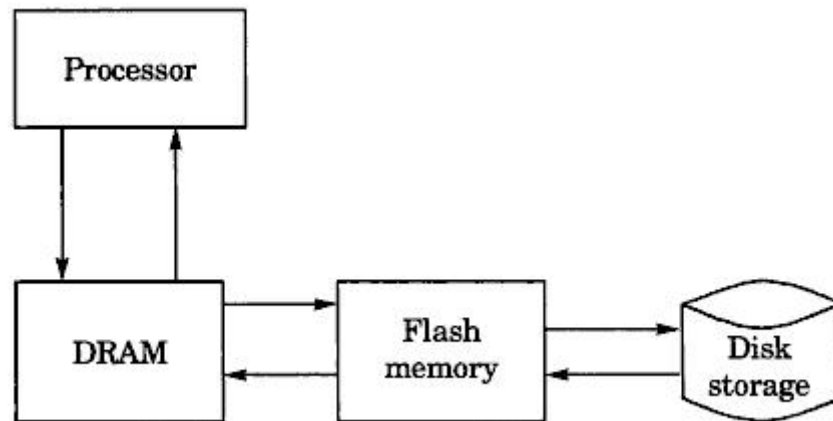
- Some embedded Linux versions may use initrd to store full-fledged file systems in compressed form in flash. To operationalize file system, the following steps come in fore :
  - The compressed file system together with the initrd image is loaded into RAM.
  - The boot sequence transfers control next to the kernel to execute the initialization sequence.
  - The kernel decompresses the file system & mounts it appropriately in RAM using the RAM disk driver.

# FLASH USED AS CACHE

- Flash memory used as intermediate storage between DRAM and hard disk.
- Requires 2 stages of data transfer :- from cache to flash and disk to flash.
- Both of these data transfer would have to be in page size of flash.
- Traditionally, the cache is between primary memory and hard disk drive.

# FLASH USED AS CACHE (CONT..)

11/15/2017





## FLASH USED AS CACHE (CONT..)

- One way of facilitating the access from the cache is to maintain a hash table in the DRAM.
- The hash table in DRAM holds the tags for each block of flash.
- These tags direct access to pages in the flash.
- Both NOR and NAND flash memory management systems must be wear-aware.
- Because flash has limited count of erase cycle somewhere between 10,000 to 100000 depending on the technology.
- The erase count maintained in DRAM for each block to ensure that the cache is still reliable.

## FLASH USED AS CACHE (CONT..)

- Alternatively, flash may itself stores the erase count.
- In fact, for this is the reason why an update is done as a new copy instead of writing back in the same block.
- This avoids having a “hot spot” with too frequent erasures.

# FLASH MEMORY COMMANDS

Typical commands for flash memory pins are as follows :-

CE : Chip Enable

WE : Write Enable

OE : Output Enable

ADE : Address Latch Enable

CDE : Command Latch Enable

## FLASH MEMORY COMMANDS (CONT..)

- With this command structure a write would require the steps to disable output, enable address latch, enable write lines followed by chip, enable to perform the write operation.
- Most suppliers of flash memory in embedded environment have development kits which would allow one to write drivers for the memory in high level language like C.

## FLASH MEMORY COMMANDS (CONT..)

- The C language program is ultimately translated into the low level enable commands which regulate the voltage levels on respective pins on the chip.
- These voltage signals achieve the desired function of read/write/erase, etc.

# FLASH MEMORY FILE SYSTEM

- File systems like ext2 (Linux) use 4 K block size.
- This does not match with the page size in flash.
- So, it requires matching file system for the flash.
- The first file system, FFS2, for flash memory was given by Microsoft.
- However, under the aegis of PCMCIA, the international industry association, definition of file translation layer(FTL) for flash evolved.

# DESIGNING FLASH MEMORY FILE SYSTEMS

- 1) To obtain good response time & also for reliability, it is recommended to spread the writes over the entire media.

By observing this write once & read often policy we can expect better response times for NOR flash, which has a higher erase & write times.

- 2) With NOR flash architectures, which are inherently parallel, it is possible to have simultaneous read & write operations.

On NAND flash, this is not possible due to serial control.

# DESIGNING FLASH MEMORY FILE SYSTEMS

- 3) Usually, when the flash is updated one writes the updated copy in a new block. Once the new block has been written the pointers to the file are updated. This is particularly useful for NAND memories where the updates are frequent.
- 4) NAND flash is often described as hard-disk-drive on a chip. The NAND flash is organized as “pages” akin to sectors on hard disk. Also, the serial control makes it suited for storing & retrieving data which has sequential character like images audio or data structures for PC.

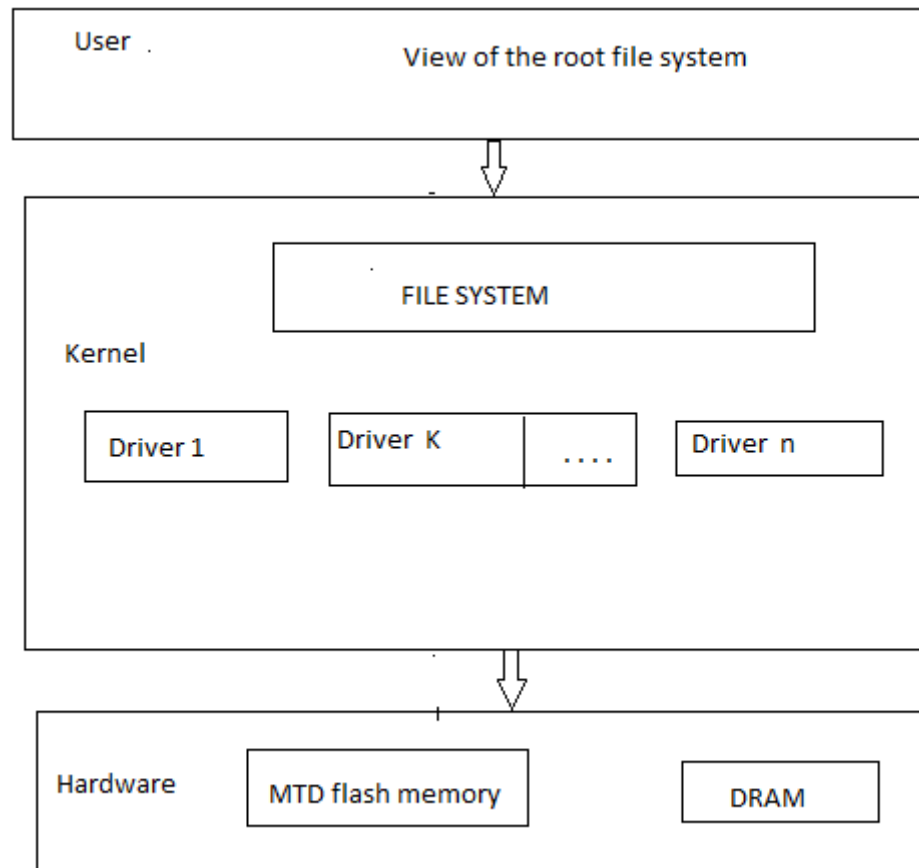


# DESIGNING FLASH MEMORY FILE SYSTEMS

- 5) For NAND flash stored data one can get the effect of random access at system level by “shadowing” data to RAM. But it occupies RAM storage.
- 5) NAND flash use error-correcting code(ECC) to maintain data integrity.

In short, NAND flash may have bad data blocks but with ECC we can still hope to use it. So NAND flash is more fault tolerant.

# FLASH FILE SYSTEM ARCHITECTURE



# FLASH FILE SYSTEM ARCHITECTURE

- At the top level (hierarchical - tree directory structure).
- This is uniform view regardless of the device.
- So user view for flash also continues to be hierarchical within directory structure i.e. file have names and host directory.
- Kernel shown at intermediate level in figure, handles IO.
- IO gets appropriately identified with buffers.
- File system translation layer finally selects an appropriate driver for device.

# FLASH FILE SYSTEM ARCHITECTURE

- In figure, MTD(memory technology device) flash is driven by “driver-k”.
- Drivers finally manage the actual data transfer with the protocol preparing the chip to receive or send data using a buffer of appropriate size.