Advance Operating System

Operating System → An operating System is a program that manages the Computer hardware. It also provided a basis for the application programs and acts as an intermediary between the computer user and the computer hardware.

→ An Amazing aspect of OS is how varied they are in accomplishing these tasks. Mainframe OS are designed primarily to optimize utilization of hardware. Personal Computer (PC) OS support complex games, business applications, and everything in between. OS for handheld Computers are designed to provide an environment in which a user can easily interface with the computer to execute programs.

→ Thus, Some OS are designed to be convenient, others to be efficient, and others Some combination of the two.

* Softwares are basically classified as →
1→ Machine language
2→ System Software
3→ Application Software.

* System Software manages the operation of the computer components.
* Application SW Solves problem of user. The base upon which application SW works is OS.
* Computer HW Shielded with a layer of System S/W is called a virtual machine.

Computer HW are basically classified as →
1→ Physical Devices
2→ Micro Programming.

Microprogram → Machine language interpreter
It interpreter machine language instruction, determine location of operands, fetch them, execute them and Store the result.

* Written in low level language, so interpretation is easy.

Features of OS →
1→ Resource Sharing
2→ Provided Virtual Machine / Extended Machine
3→ Provided Interface / user friendly
4→ Memory Management
5→ Process Management.
6→ I/O operations
7→ File System manipulation
8→ Communication
9→ Error detection
10→ Protection & Security.

Resource Manager → Controlled allocation of resources → Allocation & deallocation.

Extended Machine → Virtual Machine → CHIW with the SIW.)

Virtual Memory → Secondary memory working as primary memory.
OS with Secondary memory.

User and H/W are the two main resources which are handled by the OS.

Classification of OS →
1→ Multiuser → It allows for multiple users to use the same computer at the same time and/or different time. Example → Linux, UNIX, Windows.

2→ Multiprocessing → It is that type of OS that is capable of supporting and utilizing more than one computer processor. Example → Unix, Windows

3→ Multitasking → This OS is capable of allowing multiple SIW processes to run at the same time. Allows more than one program to run concurrently. Example → Unix, Windows.

4→ Multithreading → It allow different parts of a single SIW to run concurrently. Example → Linux, Unix, Windows

Types of OS →

There are 4 types of OS categorized based on the types of computers they control and the sort of applications they support.

1→ Real-time OS → (RTOS) → They are used to control machinery, scientific instruments and industrial systems. It has a little user interface capability. The very important property of RTOS is managing the resources of the computer so that a particular operation executes in precisely the same amount of time every time it occurs.

2→ Single-user, Single Task → This OS is designed to manage the computer so that one user can effectively do one thing at a time. Example → Palm OS for Palm handheld computers.

3→ Single-user, Multi-tasking → It lets a single user have several programs in operation at the same time, Windows and Apple's MacOS are example of such OS.

4→ Multi User → It allows multiple users to take advantage of the computer resources simultaneously. It must make sure that the requirements of various users are balanced, and that each of the programs they are using has sufficient and separate resources so that a problem with one user doesn't affect the entire community of users.

OS operation → OS has two modes of operation

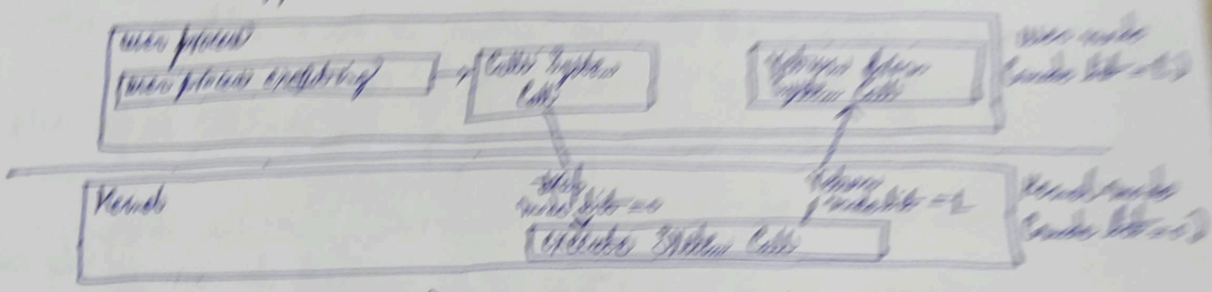
1→ User mode

2→ Kernel mode (Supervisor mode, System mode or privileged mode).

A bit called mode bit is added to the hardware of the computer to indicate the current mode: kernel (0) or user (1). With the mode bit, we are able to distinguish between a task that is executed on behalf of the OS and one that is executed on behalf of the user. When the computer system is executing on behalf of a user application, the system is in user mode. However, when a user application requests a service from the OS (via

System call 3, to must transition from user to kernel mode to fulfil the request.

At boot time, the h/w starts in kernel mode. The OS is then loaded and starts user application in user mode.



Transition from user to kernel mode.

Timer:- To prevent a user program from getting stuck in an infinite loop or not calling system services and never returning control to the OS, a timer is used. A timer can be set to interrupt the computer after a specified period. The OS sets the counter and this counter is decremented every time the clock ticks. When the counter reaches 0, an interrupt occurs which transfers control automatically to the OS.

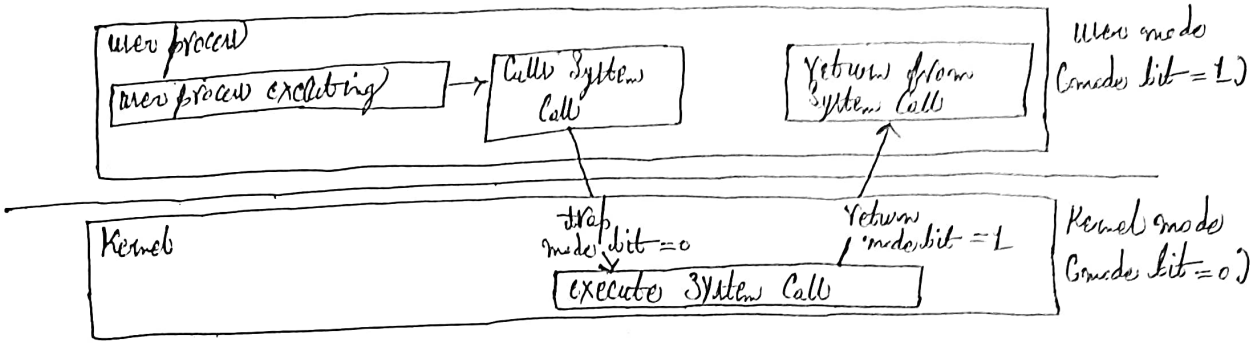
System Calls and System Programs:-

System calls provide an interface between the process and the operating system. System calls allow user-level processes to request some services from the operating system which process itself is not allowed to do. For example, for I/O a process involves a system call telling the OS to read or write particular area and this request is satisfied by the OS.

Types of System Calls are:-

System Call), it must transition from user to Kernel mode to fulfill the request.

At boot time, the H/W starts in Kernel mode. The OS is then loaded and starts user applications in user mode.



Transition from user to Kernal mode

Timer → To prevent a user program from getting stuck in an infinite loop or not calling System Services and never returning control to the OS, a timer is used. A timer can be set to interrupt the computer after a specified period. The OS sets the counter and this counter is decremented every time the clock ticks. When the counter reaches 0, an interrupt occurs which transfers control automatically to the OS.

System Calls and System Programs →

System Calls provide an interface between the process and the operating System. System calls allow user-level processes to request some services from the operating System which process itself is not allowed to do. For example, for I/O a process involves a System Call telling the OS to read or write particular area and this request is satisfied by the OS.

Types of System Calls are →

1→ Process Control →
   * end, abort
   * load, execute
   * Create process, terminate process
   * allocate and free memory
   * Get process attributes, Set process attributes.

2→ File Management
* Create file, delete file.
* open, Close
* Read, write, reposition
* Get file attributes, Set file attributes.

3→ Device Management
* Request Device, release device
* logically attach and detach devices
* Read, write, reposition
* Get device attributes, Set device attributes

4→ Communication
* Create, delete Communication Connection
* Send, receive messages
* Transfer Status information
* Attach or detach Remote devices.

System programs → System programs provide basic functioning to users so that they do not need to write their own environment for program development (eg. editors, Compilers) and program execution (Shells). In some sense, They are bundles of useful System Calls.

Unix System Calls →

System Calls	Description
access()	This checks if a calling process has access to the required file.
chdir()	The chdir command changes the current directory of the System.
chmod()	The mode of a file can be changed using this command.
chown()	This changes the ownership of a particular file.
kill()	This System call Sends kill Signal to one or more processes.
link()	A new file name is linked to an existing file using link System Calls.

System Call	Description
open()	This open a file for the reading or writing process.
pause()	The pause call Suspends a file until a particular Signal occurs.
Stime()	This System call sets the Correct time.
time()	Gets the parent and child process times
alarm()	The alarm System Call Sets the alarm clock of a process.
fork()	A new process is Created using this Command.
Chroot()	This Changes the root directory of a file.
Exit()	The exit System call is used to ~~exit~~ exit a process.

~~Memory~~ Memory Management Unit → (MMU)

A Computer's Memory management unit (MMU) is the physical hardware ~~that~~ handles its virtual memory and Caching operations. The MMU is usually located within the Computer's Central processing unit (CPU), but Sometimes operates in a Separate integrated chip (IC). All data request inputs are Sent to the MMU, which in turn determines whether the data needs to be retrieved from RAM or ROM Storage.

A memory management unit is also known as a paged memory management unit.

The MMU performs three major functions

→ Hardware memory management

→ operating System (OS) memory management

→ Application memory management

Hardware memory management deals with a System's RAM and cache memory, OS memory management regulates resources among objects and data structures, and application memory management allocates and optimizes memory among programs.

The MMU also includes a section of memory that holds a table that matches virtual addresses to physical addresses, called the translation lookaside buffer (TLB).

Process Address Space → The process address space is the set of logical addresses that a process references in its code. For example, when 32-bit addressing is in use, addresses can range from 0 to 0x7fffffff; that is 2^{31} possible numbers, for a total theoretical size of 2 gigabytes.

The OS takes care of mapping the logical addresses to the physical addresses at the time of memory allocation to the program. There are three types of addresses used in a program before and after memory is allocated —

1→ Symbolic addresses → The addresses used in a source code. The variable names, constants and instruction labels are the ~~base~~ basic elements of the symbolic address space.

2→ Relative addresses → At the time of compilation, a compiler converts symbolic addresses into relative addresses.

3→ Physical addresses → The loader generates these addresses at the time when a program is loaded into main memory.

Virtual and physical addresses are the same in compile-time and load-time address-binding schemes. Virtual & physical addresses differ in execution-time address binding scheme.

The set of all logical addresses generated by a program is referred to as a logical address space. The set of all physical addresses corresponding to these logical addresses is referred to as a physical address space.

The runtime mapping from virtual to physical address is done by the memory management unit (MMU) which is a hw device. MMU uses following Mechanism to Convert virtual addresses to physical address.

→ The value in the base register is added to every address generated by a user process, which is treated as offset at the time it is sent to memory. For example, if the base register value is 10000, then an attempt by the user to use address location 100 will be dynamically relocated to location 10100.

→ The user program deals with virtual addresses; it never sees the real physical addresses.

Memory Allocation → Main memory usually has two partitions:→

→ low memory — Operating System resides in this memory.

→ High memory — User processes are held in high memory.

Operating System uses the following memory allocation mechanism →

1→ Single-partition allocation → In this type of allocation, relocation-register scheme is used to protect user processes from each other, and from changing operating-System Code and data. Relocation register Contains value of Smallest physical address whereas limit register Contains range of logical addresses. Each logical address must be less than the limit register.

2→ Multiple-partition allocation → In this type of allocation, main memory is divided into a number of fixed-sized partitions where each partition should Contain only one process. When a partition is free, a process is selected from the input queue & is loaded into the free partition. When the process terminates, the partition becomes available for another process.

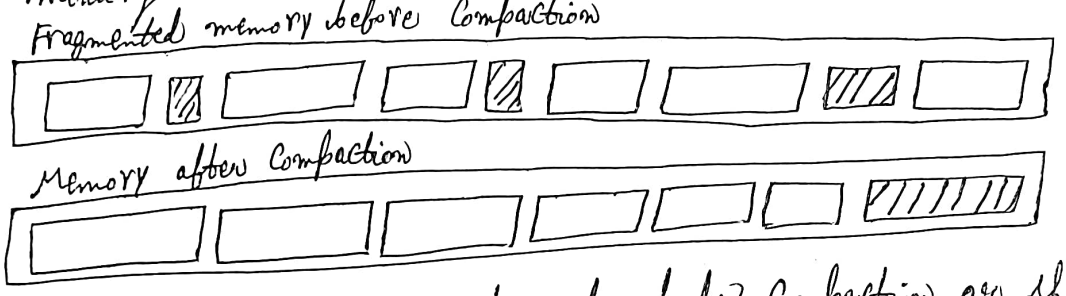
Fragmentation → As processes are loaded & removed from memory, the free memory space is broken into little pieces. It happens after sometimes that processes cannot be allocated to memory blocks considering their small size and memory blocks remains unused. This problem is known as fragmentation.

Fragmentation is of two types →

1→ External fragmentation → Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous so it cannot be used.

2→ Internal Fragmentation → Memory block assigned to process is bigger. Some portion of memory is left unused, as it cannot be used by another process.

The following diagram shows how fragmentation can cause waste of memory and a compaction technique can be used to create more free memory out of fragmented memory.



External fragmentation can be reduced by compaction or shuffle memory contents to place all free memory together in one large block. To make compaction feasible relocation should be dynamic.

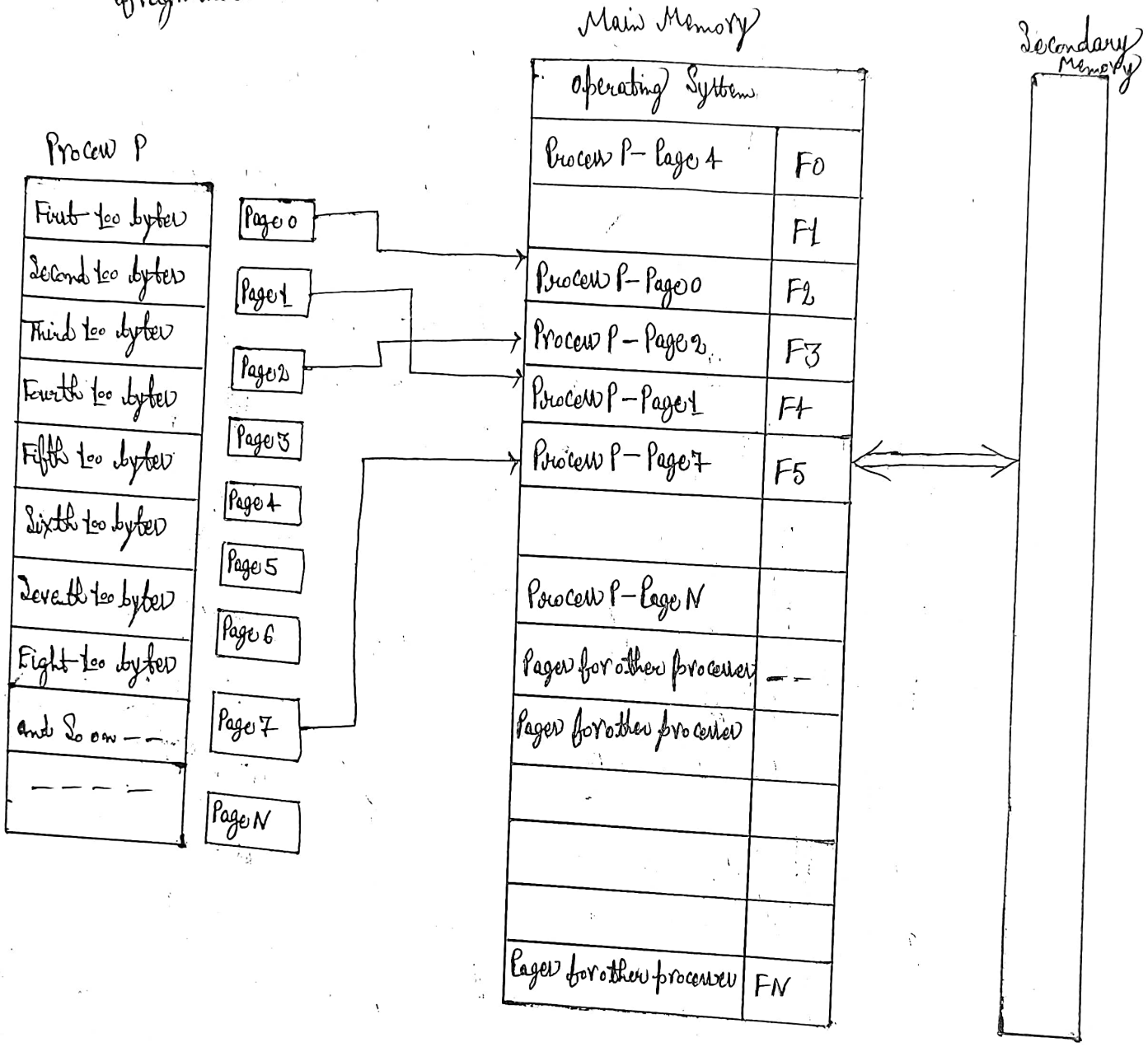
The internal fragmentation can be reduced by effectively alligning the smallest partition but large enough for the process.

~~Eg: Eg~~ Paging → A Computer can address more memory than the amount physically installed on the System. This extra memory is actually called virtual memory and it is a section of a hard drive that's set up to emulate the computer's RAM. Paging Technique plays an important role in implementing Virtual memory.

Paging is a memory management technique in which process address space is broken into blocks of the same size called pages (Size is power of 2,

between 512 bytes and 8192 bytes). The size of the process is measured in the number of pages.

Similarly, main memory is divided into small fixed-sized blocks of (physical) memory called frames and the size of a frame is kept the same as that of a page to have optimum utilization of the main memory and to avoid external fragmentation.



Address translation :-

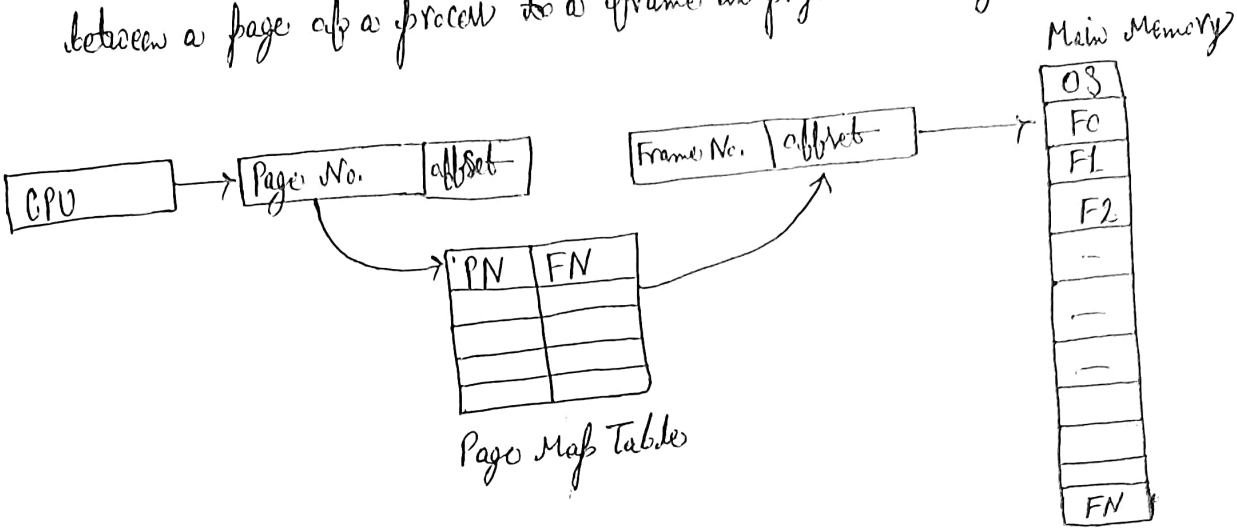
Page address is called logical address and represented by page number and the offset

Logical Address = Page number + page offset

Frame address is called physical address and represented by a frame number and the offset.

Physical Address = Frame number + page offset

A data Structure called page map table is used to keep track of the relation between a page of a process to a frame in physical Memory.



When the System allocates a frame to any page, it translates this logical address into a physical address and create entry into the page table to be used throughout execution of the program.

When a process is to be executed, its corresponding pages are loaded into any available memory frames. Suppose you have a program of 8 Kb but your memory can accommodate only 5 Kb at a given point in time, then the paging concept will come into picture. When a Computer runs out of RAM, the OS will move idle or unwanted pages of memory to Secondary memory to free up RAM for other process and bring them back when needed by the program.

This process continues during the whole execution of the program where the OS keeps removing idle pages from the main memory and write them onto the Secondary memory and bring them back when required by the program.

Advantages and Disadvantages of Paging →

→ Paging reduces external fragmentation, but still suffer from internal fragmentation

→ Paging is Simple to implement & assumed as an efficient memory management technique.

→ Due to equal size of the pages and frames, Swapping becomes very easy.

→ Page table requires extra memory Space, So may not be good for a System having Small RAM.

Segmentation → Segmentation is a memory management technique in which each job is divided into several segments of different sizes, one for each module that contains pieces that perform related functions. Each segment is actually a different logical address space of the program.

When a process is to be executed, its corresponding segmentation is loaded into non-contiguous memory though every segment is loaded into a contiguous block of available memory.

Segmentation memory management works very similar to paging but here segments are of variable-length where as in paging pages are of fixed size.

A program segment contains the program's main function, utility functions, data structures, and so on. The operating system maintains a segment map table for every process and a list of free memory blocks along with segment number, their size and corresponding memory locations in main memory. For each segment, the table stores the starting address of the segment and the length of the segment. A reference to a memory location includes a value that identifies a segment and an offset.