Unit 1 - Operating System Overview

* Operating System Objectives and Functions :-

An OS is a program that controls the execution programs and acts as an interface between applications and the computer hardware. It can be thought of as having three objectives -

1) Convenience - An OS makes a computer more convenient to use.

2) Efficiency - An OS allows the computer system resources to be used in an efficient manner.

3) Ability to evolve - An OS should be constructed in such a way as to permit the effective development, testing & introduction of new system functions without interfering with service.

The OS typically provides services in the following areas.

* Program development - The OS provides a variety of facilities & services, such as editors and debuggers, to assist the programmer in creating programs. Typically, these services are in the form of utility programs that, while not strictly part of the core of the OS, are supplied with the OS and are referred to as application program development tools.

* Program execution - A number of steps need to be performed to execute a program. Instructions and data must be loaded into main memory, I/O devices and files must be initialized and other resources must be prepared. The OS handles these scheduling duties for the user.

* Access to I/O devices - Each I/O devices requires its own peculiar set of instructions or control signals for operation. The OS provides a uniform interface that hides these details so that programmers can

DATE :

can access such devices using simple reads & writes.

* Controlled access to files — For file access, the OS must reflect a detailed understanding of not only the nature of the I/O device (disk drive, tape drive) but also the structure of the data contained in the files on the storage medium. In the case of a system with multiple users, the OS may provide protection mechanisms to control access to the files.

* System access — For shared or public systems, the OS controls access to the system as a whole and to specific system resources. The access function must provide protection of resources and data from unauthorized users and must resolve conflicts for resource contention.

* Error detection and response — A variety of errors can occure while a computer system is running. These include internal & external hardware errors, such as a memory error, or a device failure or malfunction; and various software errors, such as division by zero, attempt to access forbidden memory location, and inability of the OS to grant the request of an application. In each case, the OS must provide a response that clear the error condition with the least impact on running applications. The response may range from ending the program that caused the error, to retrying the operation, to simply reporting the error to the application.

* Accounting — A good OS will collect usage statistics for various resources & monitor performance parameters such as response time. On any system, this information is useful in anticipating the need for future enhancements and in tuning the system to improve performance. On a multiuser system, the information can be used for billing purposes.

PAGE NO.
DATE :

* The Evolution of Operating Systems —

*] Serial Processing :- With the earliest computers, from the late 1940s to the mid-1950s, the programmer interacted directly with the computer hardware; there was no OS. These computers were run from a console consisting of display lights, toggle switches, some form of input device and a printer. Programs in a machine code were loaded via the input device (e.g. - a card reader). If an error halted the program, the error condition was indicated by the lights. If the program proceeded to a normal completion, the output appeared on the printer.

*] Simple Batch Systems :- The simple idea behind the simple batch processing scheme is the use of a piece of software known as monitor. With this type of OS, the user no longer has direct access to the processor. Instead, the user submits the job on cards or tape to a computer operator, who batches the jobs together sequentially and places the entire batch on an input device, for use by the monitor. Each program is constructed to branch back to the monitor when it completes processing, at which point the monitor automatically begins loading the next program.

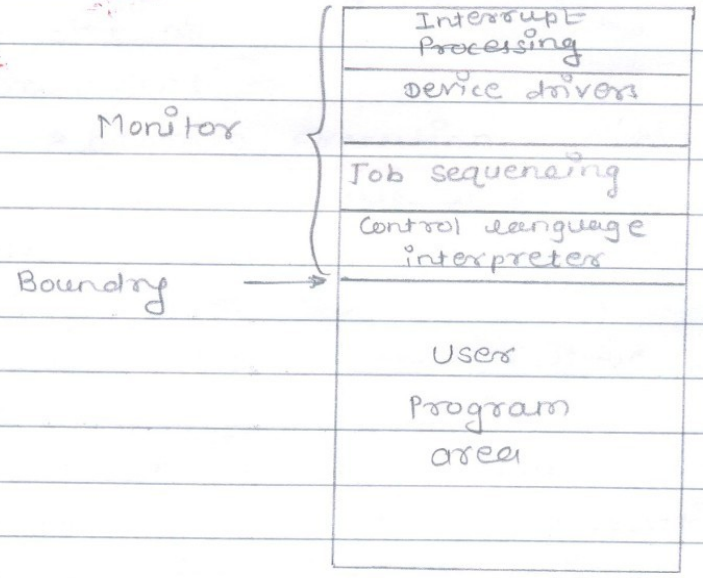


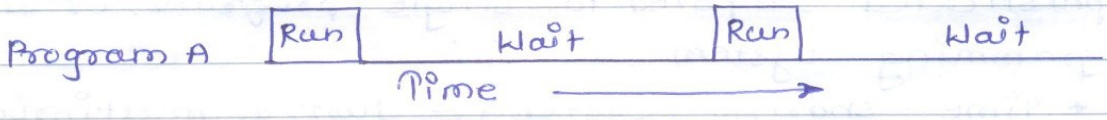
Fig :- Memory Layout For a Resident Monitor

The monitor performs a scheduling function: A batch of jobs is queued up, and jobs are executed as rapidly as possible, with no intervening idle time. The monitor improves job setup time as well. With each job, instructions are included in a primitive form of job control language (JCL). This is a special type of programming language used to provide instructions to the monitor.

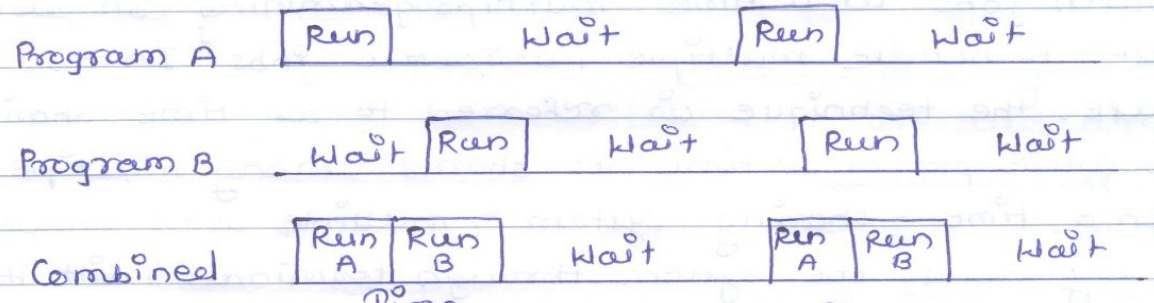
The overall format of the job looks like this:

```
$ JOB
$ FTN
.
.          } FORTRAN instructions
.
$ LOAD
$ RUN
.
.          } Data
.
$ SEND
```

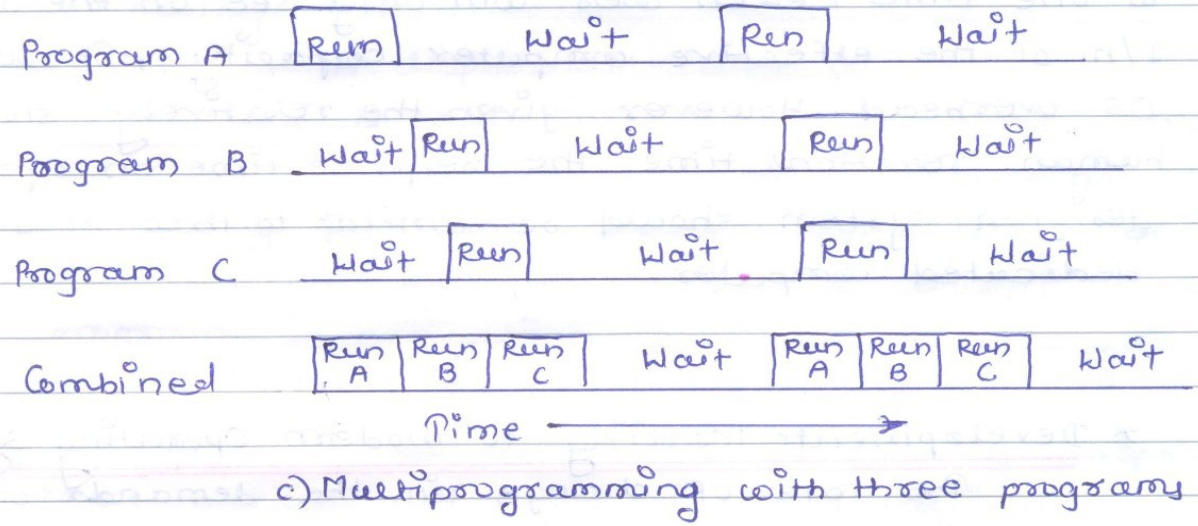
* Multiprogrammed Batch systems –



a) Uniprogramming



b) Multiprogramming with two programs



c) Multiprogramming with three programs

Fig :– Multiprogramming Example

Table – sample program Execution Attributes –

	JOB 1	JOB 2	JOB 3
Type of job	Heavy Compute	Heavy I/O	Heavy I/O
Duration	5 min	15 min	10 min
Memory Required	50M	100M	75 M
Need disk ?	No	No	Yes
Need terminal ?	No	Yes	No
Need printer ?	No	No	Yes

Multiprogramming Operating systems are fairly sophisticated compared to single-program, or uni-programming system.

* Time-sharing Systems – Just as multiprogramming allows the processor to handle multiple batch jobs at a time, multiprogramming can also be used to handle multiple interactive jobs. In this latter case, the technique is referred to as time sharing, because processor time is shared among multiple users. In a time-sharing system, multiple users simultaneously access the system through terminals, with the OS interleaving the execution of each user program in a short burst or quantum of computation. Thus, if there are n users actively requesting service at one time, each user will only see on the average 1/n of the effective computer capacity, not counting OS overhead. However, given the relatively slow human reactions time, the response time on a properly designed system should be similar to that on a dedicated computer.

* Developments leading to modern Operating Systems – The rate of change in the demands on OS requires not just modifications and enhancements to existing architectures but new ways of organising the OS. A wide range of different approaches and design elements has been tried in both experimental & commercial operating systems, but much of the work fits into the following categories:

- Microkernel architecture
- Multithreading
- Symmetric multiprocessing
- Distributed Operating System

DATE :

Most OS, until recently, featured a large monolithic kernel. Most of what is thought of as OS functionality is provided in these large kernels, including scheduling, file system, networking, device drivers, memory management and more. Typically, a monolithic kernel is implemented as a single process, with all elements sharing the same address space. A microkernel Architecture assigns only a few essential functions to the kernel, including address spaces, interprocess communication (IPC), and basic scheduling.

Multithreading is a technique in which a process, executing an application, is divided into threads, that can run concurrently, We can make the following distinction:

Thread :- A dispatchable unit of work

Process :- A collection of one or more threads and associated system resources (such as memory containing both code & data, open files & devices) This corresponds closely to the concept of a program in execution.

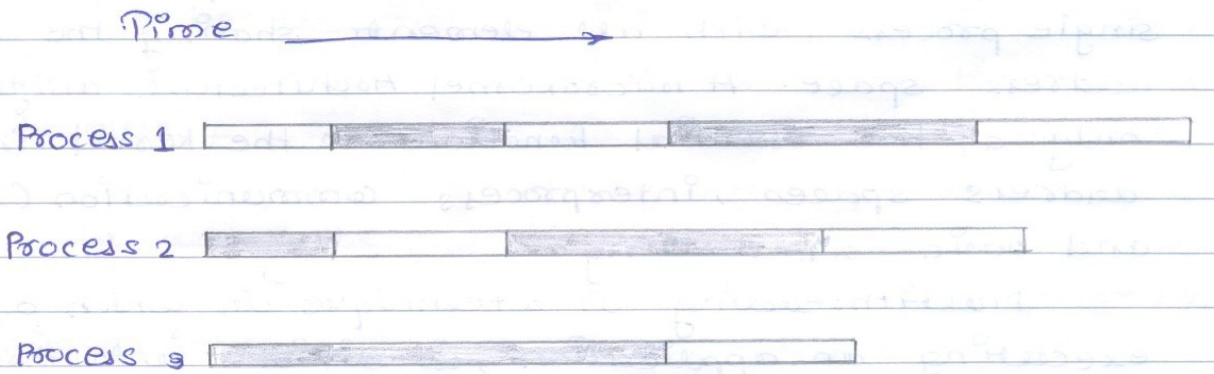
Multithreading is useful for applications that perform a number of essentially independent tasks that do not need to be serialized. An example is a database server that listens for and processes numerous client requests.

To achieve greater efficiency & reliability one technique is to employ symmetric multi-processing (SMP), a term that refers to a computer hardware architecture & also to the OS behaviour that exploits that architecture. A symmetric micro-processor can be defined as a standalone computer system with the following characteristics.

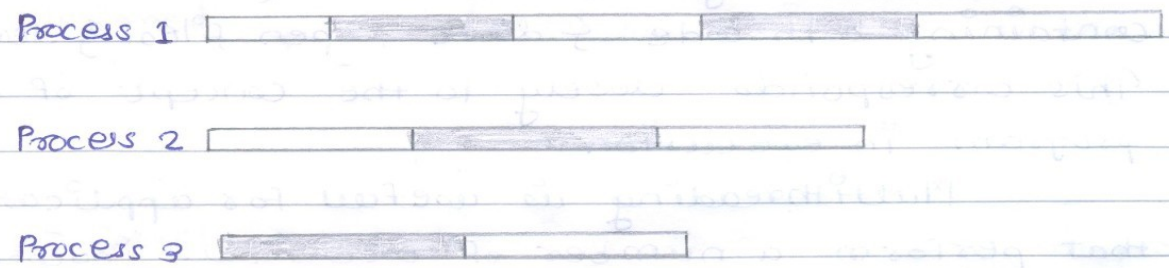
1] There are multiple processors.

2] These processors share the same main memory and I/O facilities, interconnected by a communication bus or other internal connection scheme.

3] All processors can perform the same functions.



a) Interleaving (multiprogramming, one processor)



b) Interleaving & overlapping (Multiprocessing; two processors)

Blocked Running.

Fig - Multiprogramming & Multiprocessing

DATE :

A distributed operating system provides the illusion of a single main memory space and a single secondary memory space, plus other unified access facilities, such as a distributed file system. Although clusters are becoming increasingly popular, and there are many clusters products on the market, the state of the art for distributed operating systems lags that of uniprocessor and SMP operating systems.

Another innovation in OS design is the use of object-oriented technologies. Object oriented design lends discipline to the process of adding modular extensions to a small kernel.

* Traditional Unix Systems —

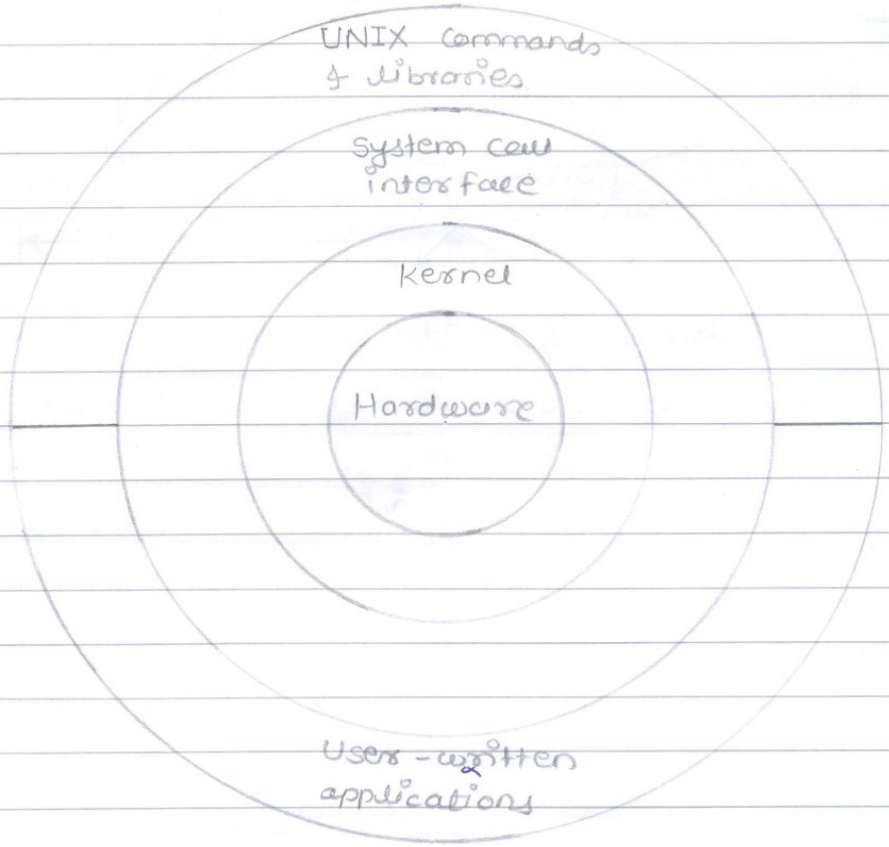


Fig – General Unix Architecture

* Modern Unix Systems

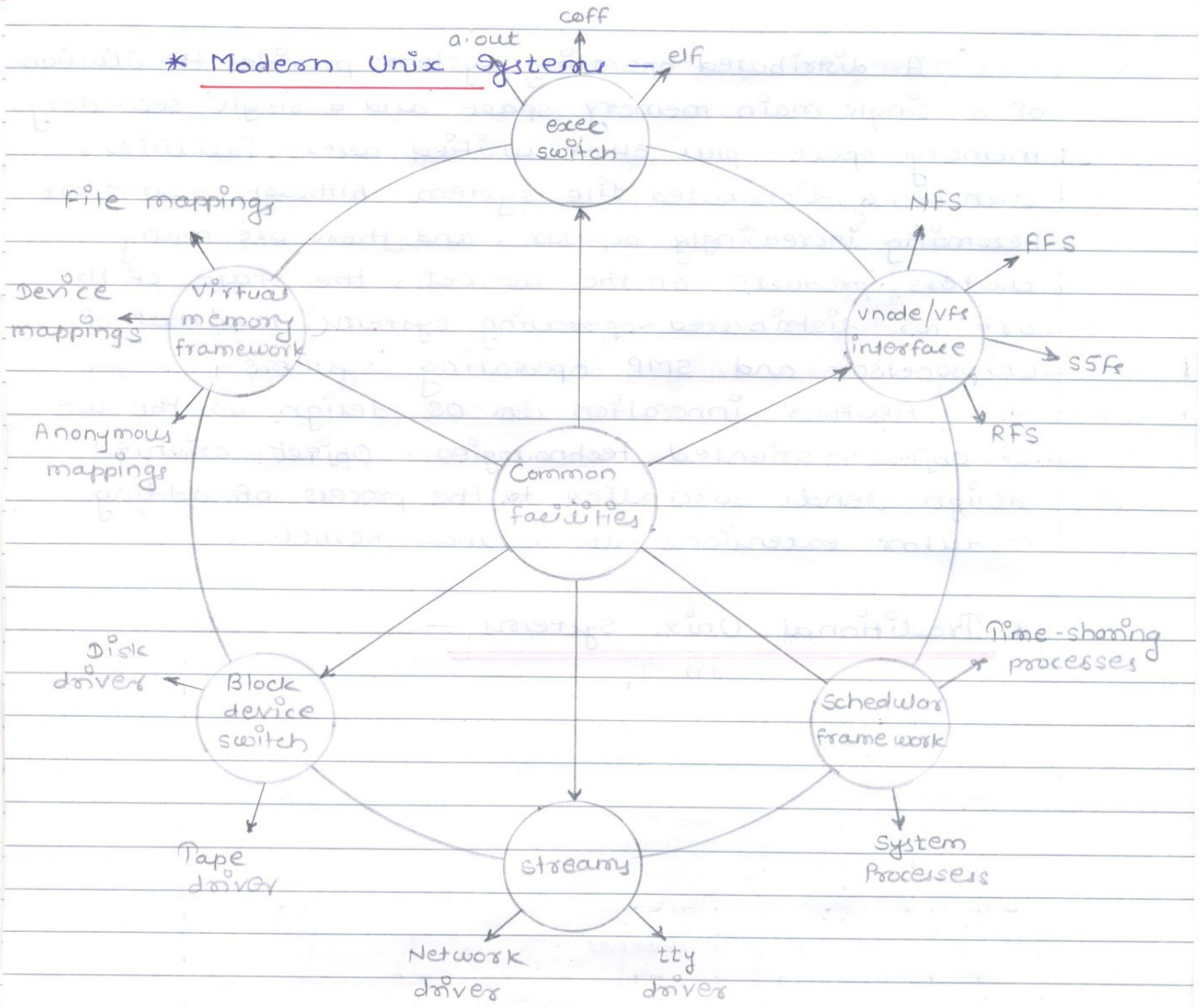


Fig – Modern UNIX kernel

* Linux kernel components —

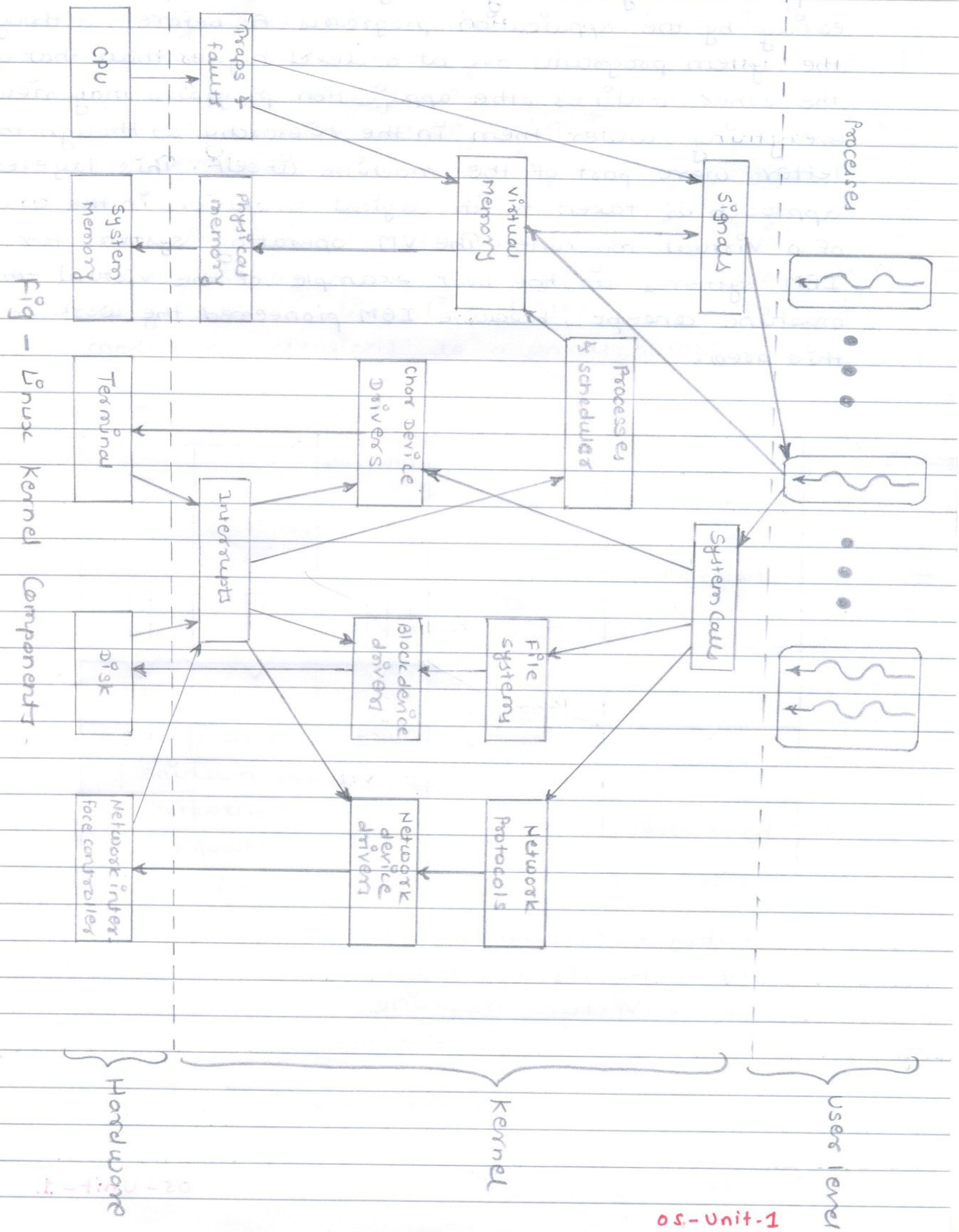


Fig - Linux kernel Component

os.-Unit-1

* Virtual Machines – Some systems carry this scheme a step further by allowing the system programs to be called easily by the application programs. As before, although the system programs are at a level higher than that of the other routines, the application programs may view everything under them in the hierarchy as though the latter were part of the machine itself. This layered approach is taken to its logical conclusion in the concept of a virtual machine. The VM operating system for IBM systems is the best example of the virtual machine concept, because IBM pioneered the work in this area.

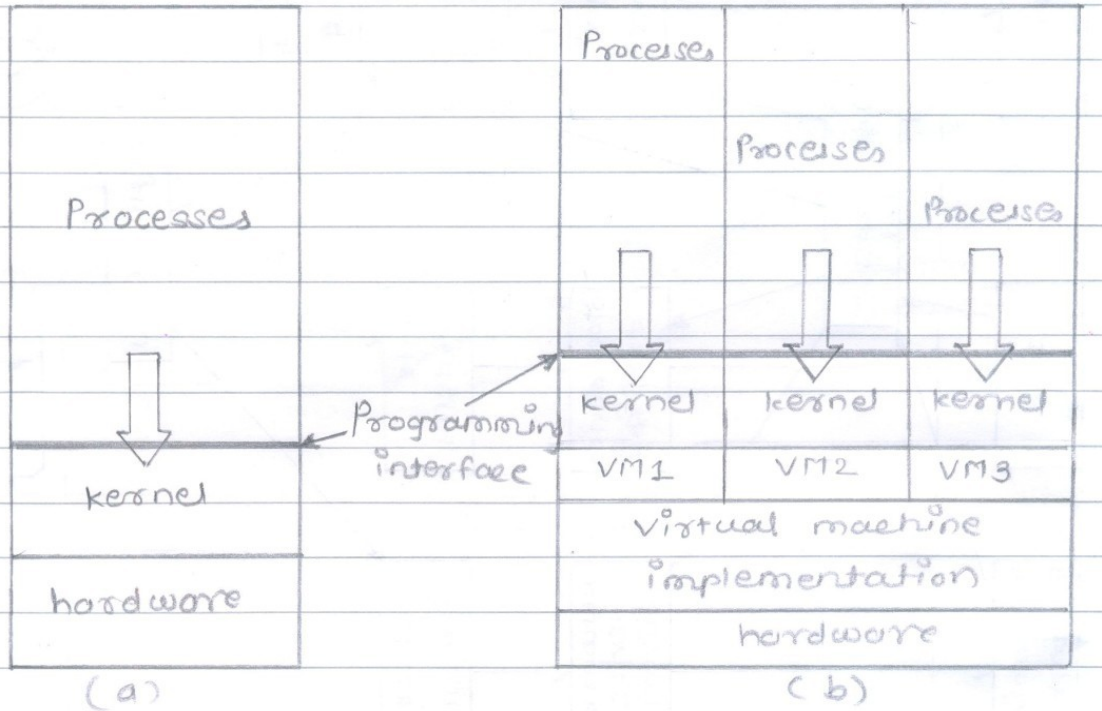


Fig - System models
(a) – Nonvirtual machine
(b) - Virtual machine

DATE :

* Benefits — The virtual machine provides a robust level of security. Second, the virtual machine allows system development to be done without disrupting normal system operation.

A disadvantage of this environment is that there is no direct sharing of resources.

* Java — Java is a very popular object-oriented language introduced by Sun Microsystems in late 1995. In addition to a language specification & a large API library, Java also provides a specification for a Java Virtual Machine (JVM).

Java objects are specified with the class construct; A Java program consists of one or more classes. For each Java class, the Java compiler produces an architecture-neutral bytecode output (.class) file that will run on any implementation of the JVM.

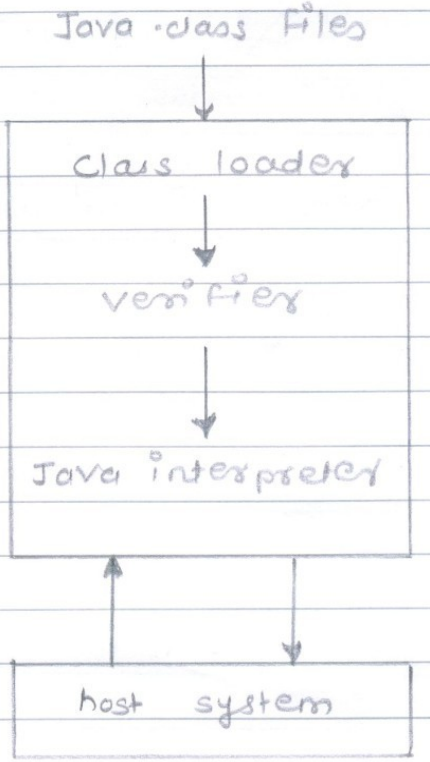


Fig - The Java virtual machine

* Windows booting Process →

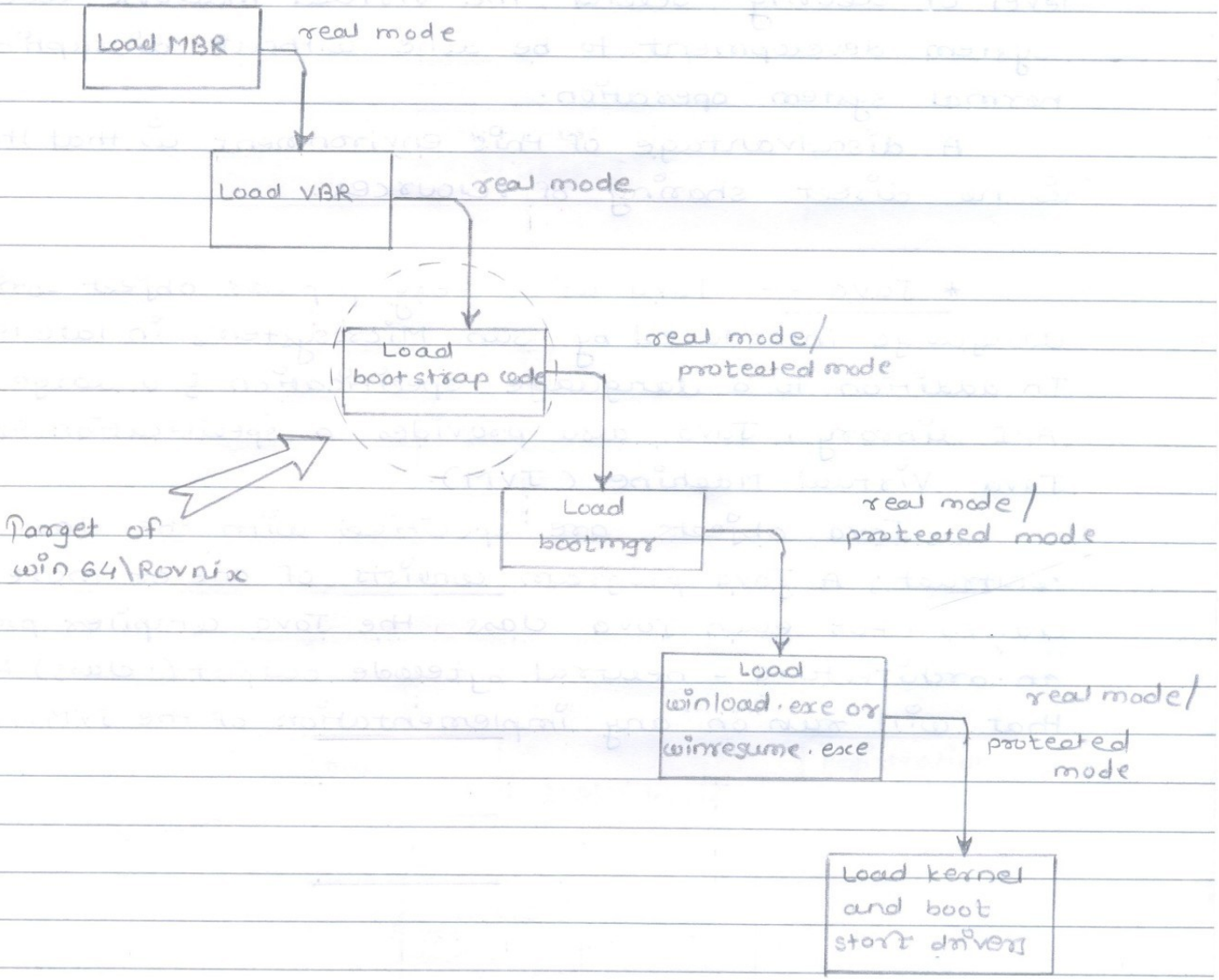


Fig- Windows booting process

* Linux booting process –

BIOS	Basic Input / output system executes MBR
MBR	Master Boot Record executes GRUB
GRUB	Grand Unified Bootloader exceeutes kernel
kernel	kernel executes /sbin/init
Init	Init executes runlevel programs
Runlevel	Runlevel programs are executed from /etc/rc.d/rc*.d/

* Linux booting process

* Android booting process →

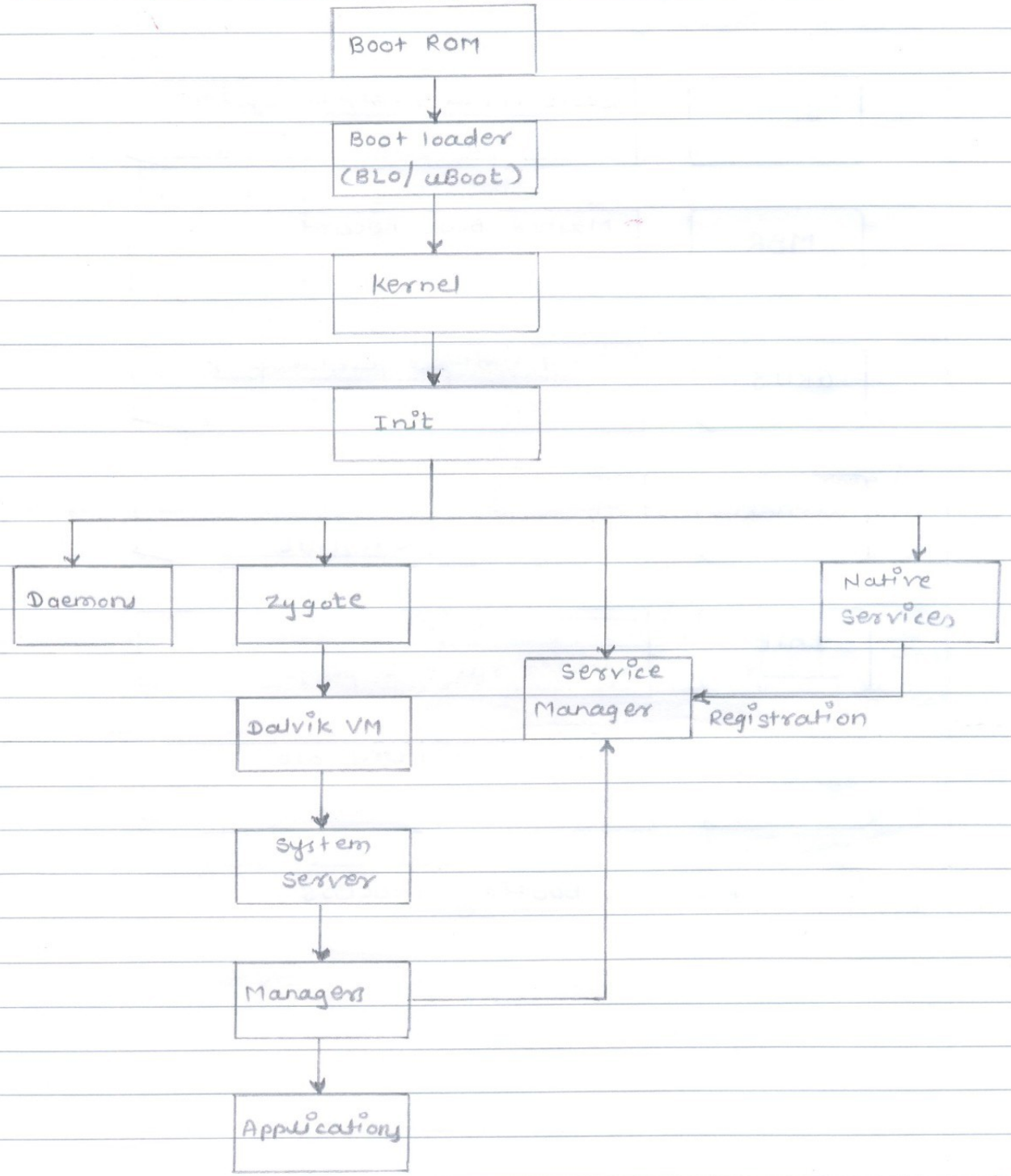


Fig - Android Booting Process