Introduction

An operating system (OS) is a set of programs that control the execution of application programs and act as an intermediary between a user of a computer and the computer hardware.

An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.

OS is software that manages the computer hardware as well as providing an environment for application programs to run.

Examples of OS are: Windows, Windows/NT, OS/2 and

1.2 Operating System Operation

i) Memory Management

Memory management refers to management of Primary Memory or Main Memory.

Main memory provides a fast storage that can be accessed directly by the CPU.

For a program to be executed, it must be in the main memory.

An Operating System does the following activities for memory management:

Keeps tracks of primary memory, i.e., what part of it are in use by

whom, what part are not in use.

In multiprogramming, the OS decides which process will get memory when and how much.

Allocates the memory when a process requests it to do so.

De-allocates the memory when a process no longer needs it or has been terminated.

ii) Processor Management

In multiprogramming environment, the OS decides which process gets the processor when and for how much time.

This function is called process scheduling.

An Operating System does the following activities for processor management:

Keeps tracks of processor and status of process.

Allocates the processor (CPU) to a process.

De-allocates processor when a process is no longer required.

iii) Device Management

An Operating System manages device communication via their respective drivers.

It does the following activities for device management:

Keeps tracks of all peripheral devices.

Decides which process gets the device when and for how much time.

Allocates the device in the most efficient way.

De-allocates devices.

iv) File Management

A file is normally organized into directories for easy navigation and usage.

These directories may contain files and other directories.

An Operating System does the following activities for file management:

Keeps track of information, location, uses, status etc.

Decides who gets the resources.

Allocates the resources.

De-allocates the resources.

1.3 Operating System Services

An Operating System provides services to both the users and to the programs.

It provides programs an environment to execute.

It provides users the services to execute the programs in a

convenient manner.

Some of the services are:

□ Security -- By means of password and similar other techniques, it prevents unauthorized access to programs and data.

□ Job accounting -- Keeping track of time and resources used by various jobs.

□ Error detecting -- Production of error messages, and other debugging and error detecting aids.

□ Coordination between other software and users -- Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

□ Communication-- In case of distributed systems which are a collection of processors that do not share memory, peripheral devices, or a clock, the operating system manages communications between all the processes.

Other Important services are:

□ Program execution

□ I/O operations

▯ File System manipulation

▯ Resource Allocation

▯ Control over system performance

1.4 Operating System Structure

Operating systems are broadly classified into following categories, based on their structuring mechanism as follows:

▯ Monolithic System

▯ Layered System

▯ Virtual Machine

▯ Exo-kernels

▯ Client-Server Model(microkernel)

1. Monolithic System

▯ The monolithic operating system controls all aspects of the operating system's operation, including file management, memory management, device management, and operational operations.

▯ The components of monolithic operating system are organized haphazardly and any module can call any other module without any reservation.

▯ The operating system code runs in a privileged processor mode (referred to as kernel mode), with access to system data and to the hardware;

 Applications run in a non-privileged processor mode (called the user mode), with a limited set of interfaces available and with limited access to system data.

 The operating system is written as a collection of procedures, each of which can call any of the other ones whenever it needs to.

 This approach might well be subtitled " The Big Mess.& quot; The structure is that there is no structure.

 The monolithic operating system structure with separate user and kernel processor mode is shown in Figure. Example Systems: CP/M and MS-DOS

2. Layered Operating System

 The layered approach consists of breaking the operating system into the number of layers(level), each built on the top of lower layers.

 The bottom layer is the hardware layer; the highest layer is the user interface.

 The main advantages of the layered approach is modularity. The layers are selected such that each uses functions (operations) and services of only lower-level layers.

 In this approach, the Nth layer can access services provided by the (N-1)th layer and provide services to the (N+1)th layer.

 This structure also allows the operating system to be debugged

starting at the lowest layer, adding one layer at a time until the whole system works correctly.

⍰ Layering also makes it easier to enhance the operating system.

⍰ If an error is found during the debugged of particular layer, the layer must be on that layer, because the layer below it already debugged.

⍰ Because of this design, the system is simplified when operating system is broken up into layer.

⍰ Os/2 operating system is example of layered architecture of operating system another example is earlier version of Windows NT.

⍰ The main disadvantage of this architecture is that it requires an appropriate definition of the various layers & a careful planning of the proper placement of the layer.

⍰ The layer approach design was first used in the THE operating system at the Technische Hogeschool Eindhoven.

⍰ The THE system was defined in the six layers, as shown in the fig below.

Example Systems:

⍰VAX/VMS,

⍰Multics,

⍰UNIX

3. Virtual Machines:

Virtual machine is an illusion of a real machine.

It is created by a real machine operating system, which make a single real machine appears to be several real machine.

The architecture of virtual machine is shown below.

The best example of virtual machine architecture is IBM 370 computer.

In this system each user can choose a different operating system.

Actually, virtual machine can run several operating systems at once, each of them on its virtual machine.

Its multiprogramming shares the resource of a single machine in different manner

JVM (Java Virtual Machine): JVM consists of

- class loader

- class verifier

-runtime interpreter

Advantages:

The virtual-machine concept provides complete protection of system resources since each virtual machine is isolated from all other virtual machines. This isolation, however,

permits no direct sharing of resources.

A virtual-machine system is a perfect vehicle for operating-systems research and development. System development is done on the virtual machine, instead of on a physical machine and so does not disrupt normal system operation.

Disadvantage:

The virtual machine concept is difficult to implement due to the effort required to provide an exact duplicate to the underlying machine

4. Client Server Structure (microkernel)

Two classes of processes - Server and Clients

Communication between client and server is via message passing

Client and server can run on different computers connected by LAN/WAN

Servers run as user mode. Hence, no system down even if the server crashed

Well adopted in distributed system

E.g. Windows NT

Fig: The client-server model (in a distributed system.)

5. Microkernel

The new concepts in operating system design, microkernel, is aimed at migrating traditional services of an operating system out of the monolithic kernel into the user-level process.

The idea is to divide the operating system task into several processes, each of which implements a single set of services

For example, I/O servers, memory server, process server, threads interface system.

Fig: The client-server model(microkernel in a non distributed system).

Each server runs in user mode, provides services to the requested client.

The client, which can be either another operating system component or application program, requests a service by sending a message to the server.

An OS kernel (or microkernel) running in kernel mode delivers the message to the appropriate server;

The server performs the operation; and microkernel delivers the results to the client in another message, as illustrated in Figure of client server structure earlier.

6. Exo-kernel architecture

⍰ It is a further extension of the micro-kernel approach where the kernel is devoid of functionality.

⍰ This means the request for file access by one process would be passed by the kernel to the library that is directly responsible for managing file system.

Comparison as:

⍰ In monolithic everything is implemented in the kernel space

⍰ In microkernel only the lower level operating system facilities are implemented in the kernel

⍰ In Exo-kernel nothing is implemented in kernel space.

1.5 System Call:

⍰ In computing, a system call is the mechanism used by an application program requests a service from an operating system's kernel.

⍰ This may include hardware related services (e.g. accessing the hard disk), creating and executing new processes, and communicating with integral kernel services (like scheduling).

⍰ System calls provide the interface between a process and the operating system.

On Unix, Unix-like and other POSIX-compatible (Portable operating system interface) operating systems, popular system calls are open, read, write, close, wait, fork, exit, and kill.

Many of today's operating systems have hundreds of system calls.

For example, Linux has over 300 different calls.

Steps in Making a System Call (example)

There are 11 steps in making the system call read (fd, buffer, nbytes)

Push parameter into the stack (1-3)

Calls library procedure (4)

Pass parameters in registers (5)

Switch from user mode to kernel mode and start to execute (6)

Examine the system call number and then dispatch to the correct system call handler via a table of pointer (7)

Run System call Handlers (8)

Once the system call handler completed its work, control return to the library procedure (9)

This procedure then return to the user program in the usual way (10)

Increment SP to clean up the stack before call to finish the job. (11)