**Milan Kumar Nayak**

**Lect.In Comp.Sc**

**Operating System**

**Operating System**

**(Unit-1)**

**Introduction to Operating System**

An operating system acts as an intermediary between the user of a computer and computer hardware. The purpose of an operating system is to provide an environment in which a user can execute programs in a convenient and efficient manner.

An operating system is a software that manages the computer hardware. The hardware must provide appropriate mechanisms to ensure the correct operation of the computer system and to prevent user programs from interfering with the proper operation of the system.

**Operating System –** Definition:

* An operating system is a program that controls the execution of application programs and acts as an interface between the user of a computer and the computer hardware.
* A more common definition is that the operating system is the one program running at all times on the computer (usually called the kernel), with all else being application programs.
* An operating system is concerned with the allocation of resources and services, such as memory, processors, devices, and information. The operating system correspondingly includes programs to manage these resources, such as a traffic controller, a scheduler, memory management module, I/O programs, and a file system.

**Functions of Operating system –**

Operating system performs three functions:

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 and introduction of new system functions at the same time without interfering with service.
4. An **Operating System** acts as a communication bridge (interface) between the user and computer hardware. The purpose of an operating system is to provide a platform on which a user can execute programs in a convenient and efficient manner.
5. An operating system is a piece of software that manages the allocation of computer hardware. The coordination of the hardware must be appropriate to ensure the correct working of the computer system and to prevent user programs from interfering with the proper working of the system.
6. **Security –**  
   The operating system uses password protection to protect user data and similar other techniques. it also prevents unauthorized access to programs and user data.
7. **Control over system performance –**  
   Monitors overall system health to help improve performance. records the response time between service requests and system response to have a complete view of the system health. This can help improve performance by providing important information needed to troubleshoot problems.
8. **Job accounting –**  
   Operating system Keeps track of time and resources used by various tasks and users, this information can be used to track resource usage for a particular user or group of user.
9. **Error detecting aids –**  
   Operating system constantly monitors the system to detect errors and avoid the malfunctioning of computer system.
10. **Coordination between other software and users –**  
    Operating systems also coordinate and assign interpreters, compilers, assemblers and other software to the various users of the computer systems.
11. **Memory Management –**  
    The operating system manages the Primary Memory or Main Memory. Main memory is made up of a large array of bytes or words where each byte or word is assigned a certain address. Main memory is a fast storage and it can be accessed directly by the CPU. For a program to be executed, it should be first loaded in the main memory. An Operating System performs the following activities for memory management:
12. **Device Management –**  
    An OS manages device communication via their respective drivers. It performs the following activities for device management. Keeps tracks of all devices connected to system. designates a program responsible for every device known as the Input/Output controller. Decides which process gets access to a certain device and for how long. Allocates devices in an effective and efficient way. Deallocates devices when they are no longer required.
13. **File Management –**  
    A file system is organized into directories for efficient or easy navigation and usage. These directories may contain other directories and other files. An Operating System carries out the following file management activities. It keeps track of where information is stored, user access settings and status of every file and more… These facilities are collectively known as the file system.

**Resources Abstraction**

Main purposes of an operating system are to provide an interface between the hardware and the application programs, execute user programs, make solving user problems easier, and make the computer system convenient to use and to manage the various pieces that make up a computer. Precisely, these pieces are called resources. A resource is any object which can be used by any process and can be allocated to any process within a system. Some examples of resources are processors (CPUs), disk space, input/output devices, files, memory (RAM), and so on. Thus, we can restate the purpose of the operating system in terms of resources. Operating system can be termed as resource manager because it manages resources (also called resource allocation and resource management) and provides an interface to resources for application programs (also called resource abstraction). Operating system must keep track of the status of each resource, decide which process is to get the resource (how much and when), allocate it, and eventually reclaim it. Viewing the operating system as a resource manager, OS must do the following: 1. Keep track of the resource. 2. Enforce policy that determines who gets what, when and for how much. [Which process gets which resource during what time of its execution and for how much

time period]. 3. Allocate the resource. 4. Reclaim the resource. The OS is a manager of system resources. A computer system has many resources as stated above. Executing a job on a computer system often requires several of its resources such as CPU time, memory space, file storage space, I/O devices and so on. The operating system acts as manager of the various resources of a computer system and allocates them to specific program and users to execute their job successfully. When a computer system is used to simultaneously handle several applications, there can be many, possibly conflicting, requests for the resources. Here, the OS must decide which requests are to be allocated resources to operate the computer system fairly & efficiently. The efficient and fair sharing of resources among users is a key goal of most operating systems. Therefore, OS is responsible to manage the system’s resources efficiently & to resolve conflicts arising from competition among the various users. Resources can be used in space-multiplexed and time multiplexed fashion. A space-multiplexed means that the resource can be divided into two or more distinct units of the resource. Different executing programs can be allocated exclusive control of different units of a resource at the same time. Memory and disks are examples of space-multiplexed resources. In this figure, one can see that three processes P1, P2, P3 are sharing single resource (memory) in space multiplexed format. Also we can relate it to an example. We can take an example of escalators in a shopping mall used by different people at same time as shown in figure. Here, escalator can be considered as space multiplexed resource and people can be considered as processes. Time multiplexed means a resource is not divided into units. Instead, a program is allocated exclusive control of the entire resource for a short period of time. After that time has elapsed, the resource is de-allocated from the program and allocated to another program. Processor (CPU), which is an important resource, is always used in time-multiplexing. In this figure, one can see that six users require CPU resource. Currently, CPU is working for user 1, after that it will serve user 2, user 3 and so on. In another example, a weighting machine can be used to measure weight by different people one by one as shown in figure. Here weighting machine can be considered as time multiplexed resource and people can be considered as processes. Resource Management Resource allocation in a computer system by operating system is a difficult task. If a computer has only one program for execution, then all the available resources can be allocated to that program. However, this situation is not very useful. To increase the utilization of a computer system, operating systems support multiprogramming. Multiprogramming means execution of multiple programs at the same time. Concurrent execution of multiple programs is not possible due to existence of single processor. Actually, the operating system distributes the resources between the competing programs in such a wonderful manner that all programs see the system as if it was dedicated to itself. Multiprogramming can only be implemented by use of resource management, deciding which running program will get what resources at what time for how long. Most computers, even personal computers, run several programs at once. In this situation, it is much more efficient for a computer to share its resources rather than dedicating all the resources to any single program until it finishes execution. You might think of this situation to a teacher having 10 different colored pencils providing them to 10 students who are painting their multicolor drawing. Every student needs all 10 different colors to complete the painting. The teacher provides all colors to a single student and let him complete the painting. Although the student who has been given all the colors cannot possibly use all the colors at once, the student keeps them all while the other students wait. In this situation, the color pencils represent the available resources, and the students represent the programs that need to use those resources. The best solution is to distribute the color pencils among the students, so that most of the students are occupied all time. When one student finishes his task with one color, he returns it to teacher and request for different unused color. Similarly, the operating system in a computer has various strategies for distributing resources to programs. Resource abstraction Along with resource allocation, resource abstraction is one of the major responsibilities of the operating

system. Resource abstraction is the process of hiding the details of how the hardware operates, making computer hardware relatively easy for an application programmer to use. Operating system can implement resource abstraction by providing a single abstract disk interface which will be the same for both the hard disk and floppy disk. Such an abstraction saves the programmer from need to learn the details of both hardware interfaces. Instead, the programmer only needs to learn the disk abstraction provided by the operating system. Operating system provides an abstraction layer over the concrete hardware, use the computer hardware in an efficient manner and hide the complexity of the underlying hardware. The significance of resource abstraction can be illustrated in this example: Suppose a programmer is writing an application to analyze stock market trends. For this programmer, the effort to design and debug code to read and write information to/from a disk drive would represent a significant fraction of the overall effort. The skill and experience required to write the software to control the disk drive are not the same as that to design the stock analysis portion of the program. While an application programmer must be aware of the general behavior of a disk drive, it is generally preferable to avoid learning the details of methods how disk input/output takes place. Abstraction is the perfect approach, since the application programmer uses a previously implemented abstraction to read and write the disk drive. A disk software package is an example of system software. Programmers can focus their attention on the application programming problem rather than diverting it to tasks not specific to the application domain. In other words, system software is generally transparent to the end user but is of major significance to the programmer. Resource abstraction provides an abstract model of the operation of hardware components and generalizes hardware behavior. It also limits the flexibility in which hardware can be manipulated. It also provides a more convenient working environment for applications, by hiding some of the details of the hardware, and allowing the applications to operate at a higher level of abstraction. For example, the operating system provides the abstraction of a file system, and applications don’t need to handle raw disk interfaces directly. Resource abstraction also provides isolation means that many applications can co-exist at the same time, using the same hardware devices, without falling over each other’s feet. One can view Operating Systems from two points of views: resource manager and extended machines. Form resource manager point of view operating systems manage the different parts of the system efficiently and from extended machines point of view operating systems provide a virtual machine to users that is more convenient to use. Modern operating systems generally have following three major goals. Operating systems generally accomplish these goals by running processes in low privilege and providing service calls that invoke the operating system kernel in highprivilege state.

**Operating system as User Interface –**

1. User
2. System and application programs
3. Operating system
4. Hardware

Every general-purpose computer consists of the hardware, operating system, system programs, and application programs. The hardware consists of memory, CPU, ALU, and I/O devices, peripheral device, and storage device. System program consists of compilers, loaders, editors, OS, etc. The application program consists of business programs, database programs.

# Types of Operating Systems

Following are some of the most widely used types of Operating system.

1. Simple Batch System
2. Multiprogramming Batch System
3. Multiprocessor System
4. Desktop System
5. Distributed Operating System
6. Clustered System
7. Real-time Operating System
8. Handheld System

## Simple Batch Systems

* In this type of system, there is **no direct interaction between user and the computer**.
* The user has to submit a job (written on cards or tape) to a computer operator.
* Then computer operator places a batch of several jobs on an input device.
* Jobs are batched together by type of languages and requirement.
* Then a special program, the monitor, manages the execution of each program in the batch.
* The monitor is always in the main memory and available for execution.

#### Advantages of Simple Batch Systems

1. No interaction between user and computer.
2. No mechanism to prioritise the processes.

## Multiprogramming Batch Systems

* In this the operating system picks up and begins to execute one of the jobs from memory.
* Once this job needs an I/O operation operating system switches to another job (CPU and OS always busy).
* Jobs in the memory are always less than the number of jobs on disk(Job Pool).
* If several jobs are ready to run at the same time, then the system chooses which one to run through the process of **CPU Scheduling**.
* In Non-multiprogrammed system, there are moments when CPU sits idle and does not do any work.
* In Multiprogramming system, CPU will never be idle and keeps on processing.

**Time Sharing Systems**

are very similar to Multiprogramming batch systems. In fact time sharing systems are an extension of multiprogramming systems.

In Time sharing systems the prime focus is on **minimizing the response time**, while in multiprogramming the prime focus is to maximize the CPU usage.

## Multiprocessor Systems

A Multiprocessor system consists of several processors that share a common physical memory. Multiprocessor system provides higher computing power and speed. In multiprocessor system all processors operate under single operating system. Multiplicity of the processors and how they do act together are transparent to the others.

**Advantages of Multiprocessor Systems**

1. Enhanced performance
2. Execution of several tasks by different processors concurrently, increases the system's throughput without speeding up the execution of a single task.
3. If possible, system divides task into many subtasks and then these subtasks can be executed in parallel in different processors. Thereby speeding up the execution of single tasks.

## Desktop Systems

Earlier, CPUs and PCs lacked the features needed to protect an operating system from user programs. PC operating systems therefore were neither **multiuser** nor **multitasking**. However, the goals of these operating systems have changed with time; instead of maximizing CPU and peripheral utilization, the systems opt for maximizing user convenience and responsiveness. These systems are called **Desktop Systems**and include PCs running Microsoft Windows and the Apple Macintosh. Operating systems for these computers have benefited in several ways from the development of operating systems for **mainframes**.

**Microcomputers** were immediately able to adopt some of the technology developed for larger operating systems. On the other hand, the hardware costs for microcomputers are sufficiently **low** that individuals have sole use of the computer, and CPU utilization is no longer a prime concern. Thus, some of the design decisions made in operating systems for mainframes may not be appropriate for smaller systems.

## Distributed Operating System

The motivation behind developing distributed operating systems is the availability of powerful and inexpensive microprocessors and advances in communication technology.

These advancements in technology have made it possible to design and develop distributed systems comprising of many computers that are inter connected by communication networks. The main benefit of distributed systems is its low price/performance ratio.

**Advantages Distributed Operating System**

1. As there are multiple systems involved, user at one site can utilize the resources of systems at other sites for resource-intensive tasks.
2. Fast processing.
3. Less load on the Host Machine.

Operating System Organization

Unit-2

**Processor User mode and Kernel Mode**

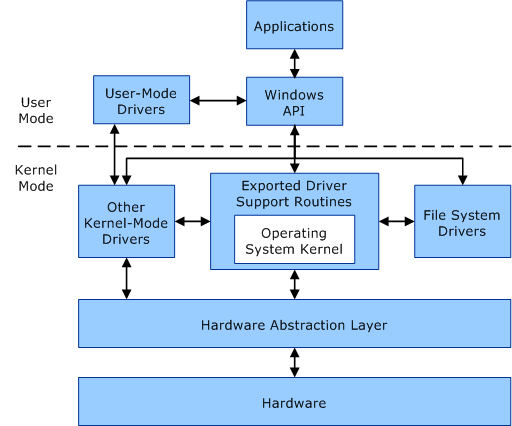
A processor in a computer running Windows has two different modes: user mode and kernel mode. The processor switches between the two modes depending on what type of code is running on the processor. Applications run in user mode, and core operating system components run in kernel mode. While many drivers run in kernel mode, some drivers may run in user mode.

When you start a user-mode application, Windows creates a process for the application. The process provides the application with a private virtual address space and a private handle table. Because an application's virtual address space is private, one application cannot alter data that belongs to another application. Each application runs in isolation, and if an application crashes, the crash is limited to that one application. Other applications and the operating system are not affected by the crash.

In addition to being private, the virtual address space of a user-mode application is limited. A processor running in user mode cannot access virtual addresses that are reserved for the operating system. Limiting the virtual address space of a user-mode application prevents the application from altering, and possibly damaging, critical operating system data.

All code that runs in kernel mode shares a single virtual address space. This means that a kernel-mode driver is not isolated from other drivers and the operating system itself. If a kernel-mode driver accidentally writes to the wrong virtual address, data that belongs to the operating system or another driver could be compromised. If a kernel-mode driver crashes, the entire operating system crashes.

This diagram illustrates communication between user-mode and kernel-mode components.



User Mode VS Kernel Mode

**User Mode**

The system is in user mode when the operating system is running a user application such as handling a text editor. The transition from user mode to kernel mode occurs when the application requests the help of operating system or an interrupt or a system call occurs.

The mode bit is set to 1 in the user mode. It is changed from 1 to 0 when switching from user mode to kernel mode.

## Kernel Mode

The system starts in kernel mode when it boots and after the operating system is loaded, it executes applications in user mode. There are some privileged instructions that can only be executed in kernel mode.These are interrupt instructions, input output management etc. If the privileged instructions are executed in user mode, it is illegal and a trap is generated.The mode bit is set to 0 in the kernel mode. It is changed from 0 to 1 when switching from kernel mode to user mode.An image that illustrates the transition from user mode to kernel mode and back again is:

## Necessity of Dual Mode (User Mode and Kernel Mode) in Operating System

The lack of a dual mode i.e user mode and kernel mode in an operating system can cause serious problems. Some of these are:

* A running user program can accidentaly wipe out the operating system by overwriting it with user data.
* Multiple processes can write in the same system at the same time, with disastrous results.

These problems could have occurred in the MS-DOS operating system which had no mode bit and so no dual mode.

**Introduction of System Call**

System calls provide an interface between the process an 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. In handling the trap, the operating system will enter in the kernel mode, where it has access to privileged instructions, and can perform the desired service on the behalf of user-level process. It is because of the critical nature of operations that the operating system itself does them every time they are needed. For example, for I/O a process involves a system call telling the operating system to read or write particular area and this request is satisfied by the operating system.System programs provide basic functioning to users so that they do not need to write their own environment for program development (editors, compilers) and program execution (shells). In some sense, they are bundles of useful system calls.

**System Call**

In computing, a **system call** is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on. A system call is a way for programs to **interact with the operating system**. A computer program makes a system call when it makes a request to the operating system’s kernel. System call **provides** the services of the operating system to the user programs via Application Program Interface(API). It provides

an interface between a process and operating system to allow user-level processes to request services of the operating system. System calls are the only entry points into the kernel system. All programs needing resources must use system calls.

**Services Provided by System Calls :**

1. Process creation and management
2. Main memory management
3. File Access, Directory and File system management
4. Device handling(I/O)
5. Protection
6. Networking, etc.

**Types of System Calls :** There are 5 different categories of system calls –

* 1. **Process control:** end, abort, create, terminate, allocate and free memory.
  2. **File management:** create, open, close, delete, read file etc.
  3. Device management
  4. Information maintenance
  5. Communication

**Examples of Windows and Unix System Calls –**

|  |  |  |
| --- | --- | --- |
|  | **WINDOWS** | **UNIX** |
| Process Control | CreateProcess() ExitProcess() WaitForSingleObject() | fork() exit() wait() |
| File Manipulation | CreateFile() ReadFile() WriteFile() CloseHandle() | open() read() write() close() |
| Device Manipulation | SetConsoleMode() ReadConsole() WriteConsole() | ioctl() read() write() |

**System Program:**

A system program is nothing but a special utility program that creates a *user-friendly environment* where the user can perform his desired work efficiently.

*Operating system, interpreter, compiler, editor* etc. are all system programs. They provide a built-in environment to the user where several necessary functions like the ones for compiling, editing, interrupt handling, memory management etc. are already defined and hence the user does not need to write any code for such functions.

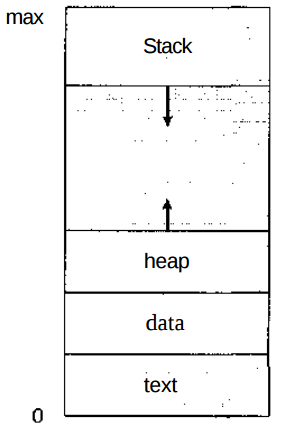
**Process Management**

**Unit-3**

A question that arises in discussing operating systems involves what to call all the CPU activities. A batch system executes jobs, whereas a time-shared system  
has user programs, or tasks. Even on a single-user system such as Microsoft Windows, a user may be able to run several programs at one time: a word processor, a web browser, and an e-mail package. Even if the user can execute only one program at a time, the operating system may need to suppoft its own internal programmed activities, such as memory management. In many respects, all these activities are similar, so we call all of them processes. The terms job and process are used almost interchangeably in this text. Although we personally prefer the term process, much of operating-system theory and terminology was developed during a time when the major activity of operating systems was job processing. It would be misleading to avoid the use of commonly accepted terms that include the word job (such as job scheduling) simply because process has superseded job.

### ****The Process****

Informally, as mentioned earlier, a process is a program in execution. A process is more than the program code, which is sometimes known as the text section.  
It also includes the current activity, as represented by the value of the program counter and the contents of the processor's registers. A process generally also  
includes the process stack, which contains temporary data (such as function parameters, return addresses, and local variables), and a data section, which  
contains global variables. A process may also include a heap, which is memory that is dynamically allocated during process run time. The structure of a process  
in memory is shown in Figure. We emphasize that a program by itself is not a process; a program is a passive entity, such as a file containing a list of instructions stored on disk (often called an executable file), whereas a process is an active entity, with a program counter specifying the next instruction to execute and a set of associated resources. A program becomes a process when an executable file is loaded into memory. Two common techniques for loading executable files are double-clicking an icon representing the executable file and entering the name of the executable file on the command line (as in prog. exe or a. out.)  
Although two processes may be associated with the same program, they are nevertheless considered two separate execution sequences. For instance,

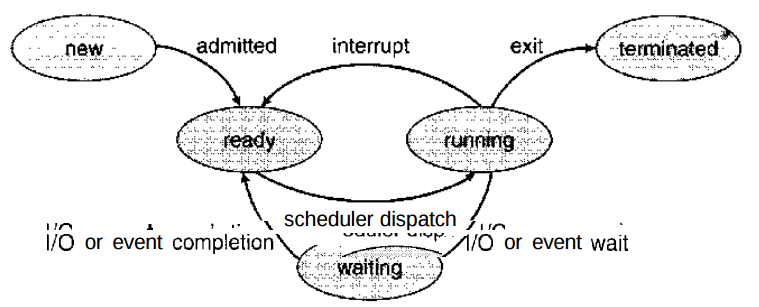


**Process in memory**

several users may be running different copies of the mail program, or the same user may invoke many copies of the web browser program. Each of these is a  
separate process; and although the text sections are equivalent, the data, heap, and stack sections vary. It is also common to have a process that spawns many  
processes as it runs.

### ****Process State****

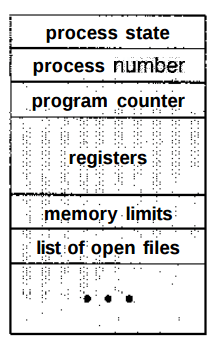
As a process executes, it changes state. The state of a process is defined in part by the current activity of that process. Each process may be in one of the  
following states:  
 • New. The process is being created.  
• Running. Instructions are being executed.  
• Waiting. The process is waiting for some event to occur (such as an I/O completion or reception of a signal).  
• Ready. The process is waiting to be assigned to a processor.  
• Terminated. The process has finished execution. These names are arbitrary, and they vary across operating systems. The states that they represent are fotind on all systems, however. Certain operating systems also more finely delineate process states. It is important to realize that only one process can be running on any processor at any instant. Many processes may be ready and limiting, however. The state diagram corresponding to these states is presented in Figure 3.2.

****

**Process states**

### ****Process Control Block****

Each process is represented in the operating system by a process control block (PCB)—also called a task control block. A PCB is shown in Figure . It contains  
many pieces of information associated with a specific process, including these:  
• Process state. The state may be new, ready, running, waiting, halted, and so on. Program counter. The counter indicates the address of the next instruction  
to be executed for this process.  
• CPU registers. The registers vary in number and type, depending on the computer architecture. They include accumulators, index registers, stack pointers, and general-purpose registers, plus any condition-code information. Along with the program counter, this state information must be saved when an interrupt occurs, to allow the process to be continued correctly afterward .  
• CPU-scheduling information. This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters.   
• Memory-management information. This information may include such information as the value of the base and limit registers, the page tables, or the segment tables, depending on the memory system used by the operating system .  
• Accounting information. This information includes the amount of CPU and real time used, time limits, account mimbers, job or process numbers, and so on.  
• I/O status information. This information includes the list of I/O devices allocated to the process, a list of open files, and so on. In brief, the PCB simply serves as the repository for any information that may vary from process to process.



### ****Threads****

The process model discussed so far has implied that a process is a program that performs a single thread of execution. For example, when a process is running a word-processor program, a single thread of instructions is being executed. This single thread of control allows the process to perform only one task at one time. The user cannot simultaneously type in characters and run the spell checker within the same process, for example. Many modern operating systems have extended the process concept to allow a process to have multiple threads of execution and thus to perform more than one task at a time.

