7 Operating Systems

Learning Objectives

In this chapter, we will study the importance of an operating system in a computer system and understand the functions that the operating system performs for managing a computer's resources. The chapter will also explain the evolution of operating systems, starting from the batch processing system to the modern-day real-time and virtual operating systems. The chapter ends with descriptions of the features of the commonly used operating systems such as disk operating system (DOS), Windows, Unix, and Linux.

7.1 INTRODUCTION

An operating system is a group of computer programs that controls a computer's resources such as the central processing unit (CPU), memory, and input/output (I/O) devices and provides the users with an interface that makes it easy to use the computer. It provides users an environment to execute programs conveniently and efficiently.

The operating system coordinates all the activities of computer hardware devices. It is usually the first program loaded into the computer during booting and it remains in the memory at all times.

Note The process of starting or restarting the computer is called booting. There are two types of booting. Cold booting is done when a completely turned off computer is turned on. Warm booting takes place when the operating system is used to restart the computer.

The basic functions of an operating system (shown in Figure 7.1) are as follows:

Performs basic computer tasks The operating system performs a computer's basic tasks such as managing peripheral devices—keyboard, mouse, printer, and so on. The operating

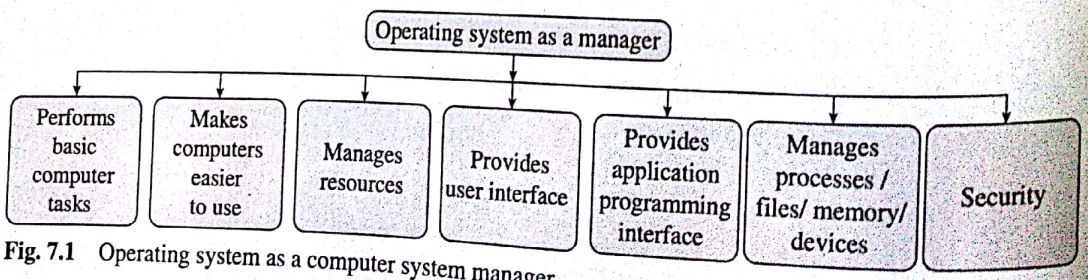


Fig. 7.1 Operating system as a computer system manager

Scanned by CamScanner

systems available today are based on the plug and play concept in which a new device when connected will be automatically detected and configured without any user's intervention.

Makes computer easy to use A computer system comprises one or more CPUs, memory, and I/O devices such as disks, tapes, printers, and terminals. Writing programs to correctly and efficiently use a computer is an extremely challenging job and requires an in-depth knowledge of the functioning of these resources. Hence, to make computers easy to use and manage, the operating system is added as a layer in between the application programs and the computer hardware (Figure 7.2). This shields the users and programmers from the complexities of the underlying hardware.

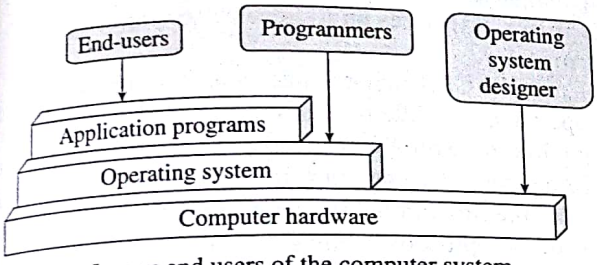


Fig. 7.2 Layers and users of the computer system

Manages computer's resources The operating system controls and efficiently utilizes hardware components such as the CPU, memory, and I/O devices. It also facilitates sharing of the CPU time amongst different user's applications or peripheral devices. Usually, users simultaneously run different programs (such as creating a word document, sending email, and listening to music) and the programs constantly compete for the CPU's time, memory, and I/O bandwidth until they finish their execution. The operating system, therefore, ensures that each program gets the necessary resources it needs to maximize the functionality of the overall system. At a given instant of time, it decides (a) which program will be executed by the CPU, (b) what all programs will be present in the memory, and (c) which program will use the I/O device.

Provides user interface The operating system enables users to easily interact with the computer hardware. For example, the Windows operating system displays icons through which the users can interact with the system. In general, a user can interact with the computer system in two ways: command line interface (CLI) and graphical user interface (GUI). With the CLI, the user interacts with the operating system by typing commands to perform specific tasks. Disk operating system (DOS) is a popular example of a CLI. With the GUI, the user interacts with the operating system by using a pointing device such as a mouse to select windows, icons, and menus. Windows XP, Windows 7, and Windows 8 are all examples of GUI.

Provides application programming interface The operating system provides a consistent application interface that ensures programmers that the applications developed on one computer will also run on other computers with the same or different hardware configuration. Since the operating system and not the application is in charge of managing the hardware and the utilization of its resources, it can easily manage hardware from the numerous vendors manufacturing computer equipment. For example, in Windows, the steps to add a new printer will be the same irrespective of whether the printer was manufactured by HP, Cannon, or some other company.

Manages processes A program in execution is called a process. The operating system enables users to execute at the same time more than one process to enhance productivity. Multiple processes being executed simultaneously calls for efficient utilization of the system's resources by the operating system. Moreover, while managing processes, besides taking care of scheduling of resources to different processes, the operating system also takes care of the creation and deletion of processes and provides the mechanism for synchronization and communication among processes.

Manages memory A program to be executed must be loaded in the main memory along with its data. To improve CPU utilization and provide better response time to the users, a computer usually stores more than one program in the memory. Hence, the operating system has to find vacant space in the main memory, load the appropriate data and programs in the located space (also known as memory allocation), execute the programs, and de-allocate the memory space for a program when it has completed its execution.

Manages files All users store, retrieve, and work on the information stored in files. The operating system, therefore, enables the users to create, copy, delete, move, and rename a file. It also keeps track of the location, access time, modified time, and other details of files and directories (folders) saved on the computer disk.

Manages security Computer systems store a large amount of data, which can be confidential and very valuable to its users. To ensure privacy of the stored data, the operating system protects the stored information from unauthorized users and denies them access to data. Besides this, it also ensures that one process does not interfere with other processes or with the operating system itself.

Manages devices The operating system manages and controls all I/O devices such as disks, tapes, terminals, printer, and keyboard to ensure correct data transmission to and from devices. It also provides an intuitive interface so that users can easily work with them.

Note The operating system also keeps an account of the system resources used by the processes and maintains an internal time clock and a log of system usage by the users.

Scanned by CamScanner

7.2 EVOLUTION OF OPERATING SYSTEMS

The evolution of operating systems in a chronological order, shown in Figure 7.3, is discussed in this section.

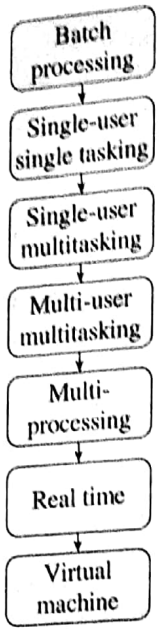


Fig. 7.3 Evolution of operating systems

Batch Processing Operating System

Batch processing operating system allows very limited or no interaction between the user and the processor during the execution of work. Data and programs that need to be processed are bundled and collected as a 'batch' and executed together (Figure 7.4).

Batch processing operating systems perform very well when a large amount of data has to be processed and either the data or the processing is similar in nature. Batch processing is performed automatically without any user intervention. For example, an organization uses batch processing to automate its payroll. The process would identify each employee, calculate his/her monthly salary (with tax deductions), and print the corresponding payslip. Batch processing is useful for this purpose since these procedures are repeated for each employee every month.

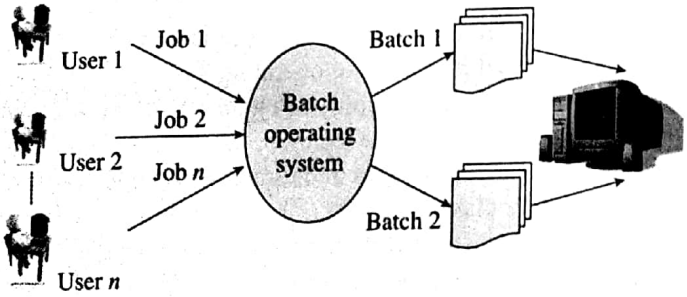


Fig. 7.4 Batch operating system

In the batch operating system, users had to prepare their job on an offline device such as a punch card and submit it to the computer operator. To speed up processing, the computer operator would group the jobs with similar needs to form a batch, which were then executed by the computer. The results of execution and errors (if any) were recorded on the punched cards and then given to the user of the job. The following were the typical problems with the batch operating system:

- There was little or no interaction between the user and the job.
- The CPU remained idle most of the times as the speed of the I/O devices was far slower than that of the CPU.
- Assigning priorities to jobs was difficult.

Single-user Single-tasking Operating System

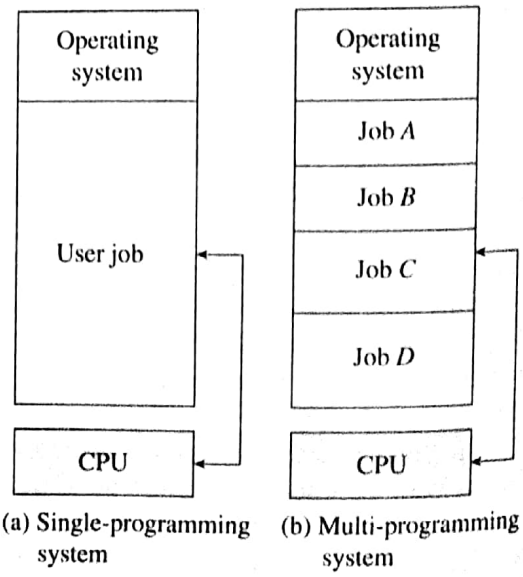
As the name indicates, the single-user single-tasking operating system allowed only one program to be executed at a time. It was designed to manage the computer to enable a single user to do a single job effectively at any point of time. The Palm OS for palm hand-held computers is an example of a modern single-user single-task operating system.

Multiprogramming Operating System

In the batch processing system, only one job was stored in the memory and executed by the CPU. The next job would be loaded for execution only after the first job was completely executed. Thus, the entire main memory and CPU time were exclusively reserved for a single job (Figure 7.5a). However, a job does not need the CPU all the time during its execution. For some time, it needs the CPU for processing, and at other times, it requires an I/O device for performing an I/O operation.

Hence, the CPU remains idle when the job has to do an I/O operation. The idle time of the CPU can be significant if the job has to do 80–90 per cent of I/O operations and only 10–20 per cent of processing. To reduce the idle time of the CPU, multiprogramming was introduced for interleaved execution of multiple jobs by the same CPU.

In the multiprogramming operating system, more than one job resides in the main memory so that when one job is busy with an I/O operation, the operating system can allocate the CPU to another job waiting for execution by the CPU. Figure 7.5(b) shows a multiprogramming system in which more than one job is in the main memory. When Job *A* is busy with some I/O operation, the operating system allocates the CPU to Job *B* to enhance the utilization of the CPU. Jobs *C* and *D* are in the memory, waiting to be executed by the CPU. Figure 7.6 illustrates the concept of a uniprogramming system and a multiprogramming system.



(a) Single-programming system (b) Multi-programming system

Fig. 7.5 Operating system

Scanned by CamScanner

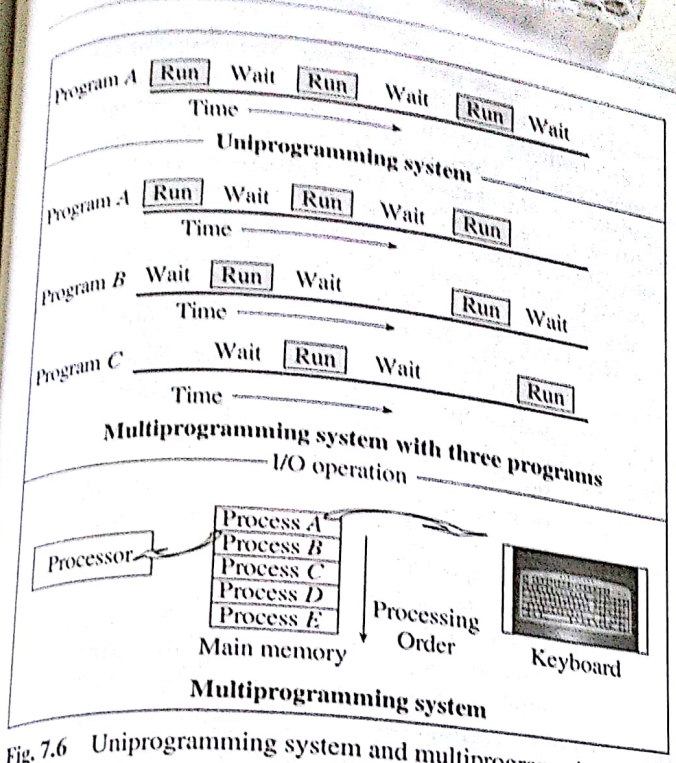


Fig. 7.6 Uniprogramming system and multiprogramming system

Single-user Multitasking Operating System

The single-user multitasking operating system (a form of multiprogramming) allows a single user to simultaneously perform several tasks. This is the operating system normally used in our desktops and laptops. The single-user multitasking operating system enhances the productivity of users, as they can complete more than one job at the same time.

For example, when we are typing a document in Microsoft Word while listening to a song and downloading a file from the Internet, we are actually doing three jobs at the same time with the help of a multitasking operating system. Without multitasking, the user had to close the word document to listen to the music and then close the music player to download the file from the Internet.

Microsoft's Windows and Apple's Mac OS platforms are examples of single-user multitasking operating systems.

Multi-user Multitasking Operating System

A multi-user operating system enables multiple users on different computers to access a single system (with one operating system). In simple terms, it allows more than one user to connect to the main computer (which has only one CPU and one operating system) to perform more than one job at a time. Hence, users on multiple terminals can access the same data and application programs that are stored on the main computer.

The multi-user multitasking operating system is quite complicated, as it must efficiently manage the jobs and resources required by the different users connected to it. Users will

typically be at terminals or computers that access the main system through a network. In a multi-user operating system (Figure 7.7), each user accesses the same operating system at different machines. This type of system is often used on mainframes and similar machines. However, if the main system fails, it can affect several users.

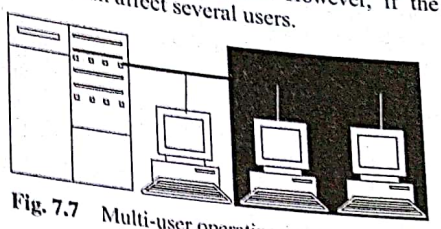


Fig. 7.7 Multi-user operating system

Time-sharing Operating System

Time sharing is a logical extension of multiprogramming and enables multiple users to simultaneously share the CPU's time. Time-sharing systems were developed to provide an interactive use of the computer system by several users. In such a system, the CPU switches between multiple jobs so frequently that every user receives an immediate response (in a few seconds).

The time-sharing operating system makes use of CPU scheduling and multiprogramming to provide each user with a small slot of the processor's time for execution. When the time slot allotted to a user's job expires, the operating system allocates the CPU time to the next user on the system. This timeslot allotted is extremely small and gives the users an impression that each of them has his/her own CPU. The short period of time, which is usually between 10–100 ms, during which a user gets the CPU is known as a *time slice* or a *quantum*. The concept of a time-sharing system is shown in Figure 7.8. In this figure, the CPU is shared amongst five users, of which two have been allotted the CPU; after the time slot expires, user 3 will be given a chance to execute his/her job.

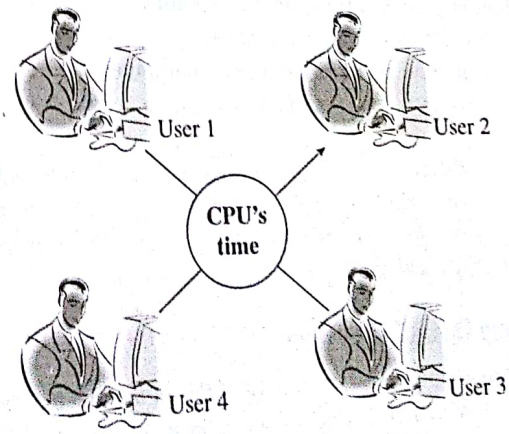


Fig. 7.8 Time-sharing operating system

Note: The main difference between multiprogramming and time-sharing is that the former focuses on maximizing processor use whereas the latter focuses on minimizing response time.

Scanned by CamScanner

Multiprocessing Operating System

Multiprocessing means using two or more processors (CPUs) within a single computer system. It refers to the coordinated processing of programs by more than one processor. In multiprocessing systems, a complex program can be divided into smaller parts and then be executed concurrently by multiple processors.

Figure 7.9 shows a multiprocessing system that has two processors for carrying out arithmetic and logical operations, two I/O processors for carrying out I/O operations concurrently, and a shared main memory. Here, the operating system manages communication between two processors.

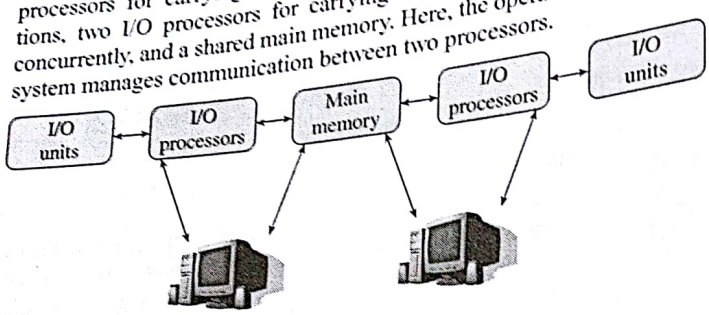


Fig. 7.9 Multi-processing system

In general, there are two types of multiprocessing systems: *tightly coupled systems* in which all processors share the same primary memory and *loosely coupled systems* in which each processor has its own local memory and the processors do not share a system-wide memory.

However, the term multiprocessing should not be confused with multiprogramming, which means the interleaved execution of two or more programs by a single processor. Today, most of the modern computers support multiprogramming. Moreover, a system can be both multiprocessing and multiprogramming, only one of the two, or neither of the two. Typical advantages of the multiprocessing system include the following:

- Users can share resources.
- If one processor fails, the other processors can do load balancing and continue with operations.
- There is improved response time.
- There is reduction of load on one computer.
- There are less delays in data processing.

Note Real-time processing is always online, whereas online systems may or may not be real time. The response time of a real-time system is very less when compared to that of online processing.

Real-time Operating System

A real-time operating system (RTOS) is a data processing system in which the time interval required to process and respond to inputs is very small and highly critical.

An RTOS has well-defined, fixed time constraints; otherwise, the system will fail. Let us consider two examples that will help us to understand the utility of RTOS in real-world applications.

- RTOS is used in embedded systems like cell phones, air conditioners, digital homes, cars, and so on. For example, an RTOS can be used in car air bags. Now, in case of a severe accident, if the air bag does not open at the right time, then the person in the car will be either injured or may even die. Thus, an RTOS has strict constraints on time; otherwise, the system will fail (no use of deploying the air bag).
- Another area in which the RTOS is commonly used is an HDTV receiver and display. This device continuously reads a digital signal, decodes, and displays it in real time. Any delay in this process can lead to a jerky or pixelated video and/or garbled audio.

RTOS is often used when there is a rigid time requirement on data processing. For example, real-time systems are widely used to control a device in a dedicated application, in industry to control machinery, and in scientific experiments, medical imaging systems, weapon systems, robots, home appliances controllers, air traffic control systems, and so on.

An RTOS has very little user interface capability and does not support end-user utilities. Like other operating systems, it manages the hardware resources of a computer and the host applications that run on the computer. However, unlike other operating systems, an RTOS is specially designed to run applications with very precise timing and a high degree of reliability. This is particularly useful in measurement and automation systems where down-time is costly or a program delay could cause a safety hazard.

There are two types of real-time operating systems: *hard real-time system* and *soft real-time system*, which are summarized in Table 7.1.

Table 7.1 Hard real-time and soft real-time systems

Hard real-time system	Soft real-time system
• Hard real-time systems guarantee that critical tasks are completed on time.	• Soft real-time systems are less restrictive but they give priority to critical tasks and retain the priority until the task is completed.
• In hard real-time systems, secondary storage is limited or missing. They store data in the read-only memory.	• Soft real-time systems have limited utility than hard real-time systems.
• In these systems, virtual memory is almost never found.	• Applications of soft real-time systems include multimedia, virtual reality, and advanced scientific projects such as undersea exploration and planetary rovers.
• Applications of hard real-time systems include controlling a device in a dedicated application, robotics, and air traffic control systems.	

Scanned by CamScanner

Virtual Machine Operating System

Virtual machine operating system enables several users of a computer to use it as if they are using it individually. When using a virtual machine operating system, several operating system environments can coexist on the same computer. For example, Figure 7.10 shows three operating system environments existing on the same computer. These operating systems are Windows, Linux, and Solaris.

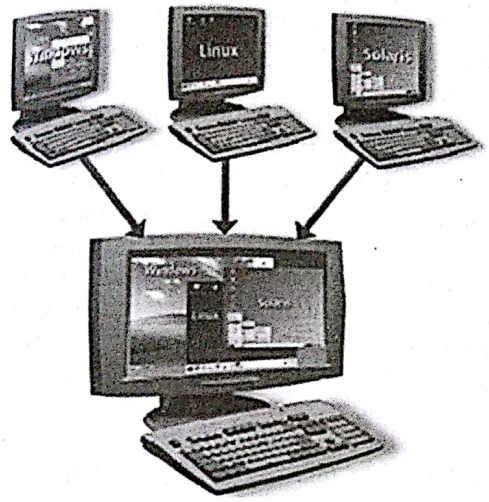


Fig. 7.10 Virtual operating system

7.3 PROCESS MANAGEMENT

A process is a program in execution. Process management is an integral part of the operating system. Before proceeding further, consider Figures 7.11 and 7.12, which show the components and states of a process.

Process components include code, data, resources, and status of the process, whereas process states include new, ready, running, waiting, and terminated. A process state, which is defined as the current activity of the process, keeps changing as the process is executed.

According to Figure 7.12, a process is in the *new* state when it is created. Then, the process enters the *ready* state in which it waits for the operating system to allocate it to the processor for execution. Once the process gets the CPU time, it starts running or executing the program code. At this stage, two events can happen. Either the process gets completely executed and thus gets terminated or it may change state from running to waiting in case it needs to perform some I/O operation. After it completes the I/O operation, the process again enters the ready state so that it can again be allotted the CPU to complete its execution.

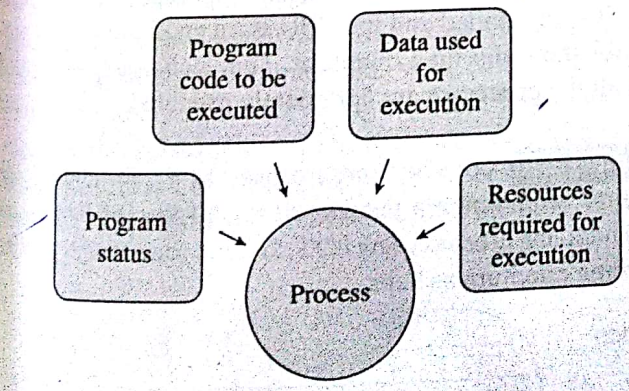


Fig. 7.11 Components of a process

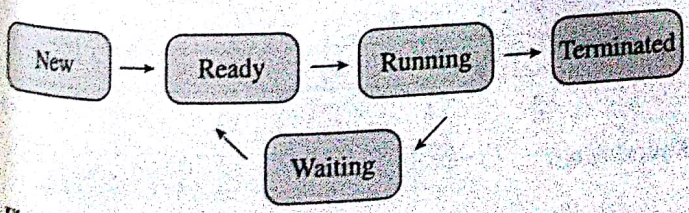


Fig. 7.12 Process states

Note A process can terminate only when all the requested resources have been allocated to the process.

7.3.1 Process Control Block

As a part of process management, the operating system allocates resources to processes, enables different processes to share and exchange information, protects the resources of each process from other processes, and allows synchronization among processes. In order to do this, the operating system maintains a data structure for each process that describes the state and resource ownership of that process. All this information together enables the operating system to exert control over each process. The process control block (PCB), also called a *task control block* (shown in Figure 7.13), stores all types of information that the operating system needs about a particular process. This information includes the following:

- Process number, which identifies the process
- Pointers, which point to the PCB of other processes
- Current state of the process
- Program counter, which stores the location of the next instruction to be executed for this process
- Contents of CPU registers

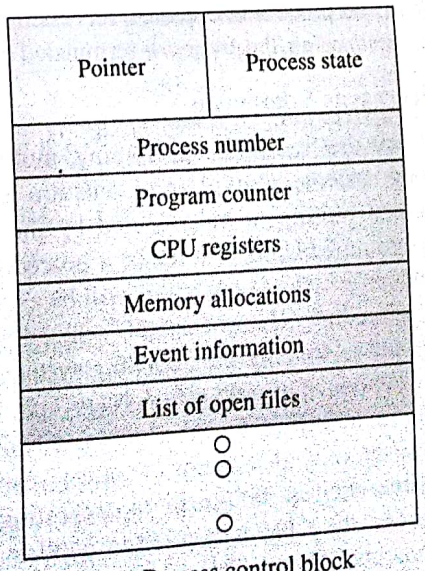


Fig. 7.13 Process control block

Scanned by CamScanner

- States of various flags and switches
- Upper and lower bounds of the memory required for the process
- List of files opened by the process
- Priority of the process
- Status of all I/O devices needed by the process

7.3.2 Process Operations

The operating system primarily carries out two main types of process operations: *process creation* and *process termination*. When a process is executing, it may request the operating system to create a new process. In such a case, the process that requests the creation of a new process is called the *parent process* and the new process is called the *child process*. Note that a child process can also further create new processes, thereby forming a hierarchical structure of processes, as shown in Figure 7.14.

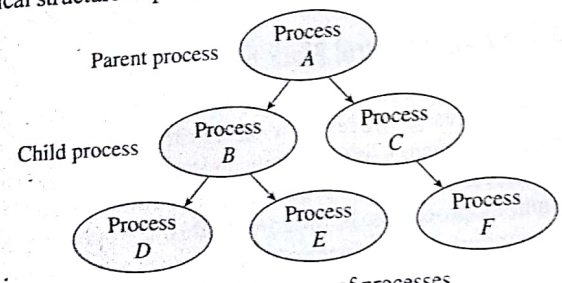


Fig. 7.14 Hierarchical structure of processes

Process creation is very common while executing a program, as corresponding to every task that has to performed, there is a process associated with it. For example, a new process is created whenever a user logs on to the system or whenever the user opens a word document. When a process is executed to its completion, it is terminated by the operating system.

During process creation, the operating system allocates the required resources to the child process. Similarly, during process termination, it de-allocates and releases all the resources allocated to the process that is being terminated. The termination of a process may be normal or abnormal. In *normal termination*, the process is executed successfully, whereas in *abnormal termination*, the process is terminated due to an error.

7.3.3 Processes Scheduling

We have seen that in multiprogramming, multitasking, and time-sharing systems, multiple programs are present in the main memory waiting to be executed by the CPU. Since there is only one CPU, there must be a procedure to decide the sequence in which the processes will be executed by the CPU. This procedure is known as CPU scheduling or process scheduling and is done by the operating system. In general, there are two types of process scheduling:

Preemptive scheduling In preemptive scheduling, each process is given a small amount of time to execute. When the time slice expires, the currently executing process is suspended, and the CPU is allotted to another process for the same amount of time. This process repeats until all the processes are executed. Moreover, when a higher priority process comes, the lower priority process is suspended and the higher priority process is executed.

Non-preemptive scheduling In non-preemptive scheduling, the first process is completely executed, and then the CPU executes the next process. The process is repeated until all the processes are executed. There is no need for a CPU scheduling policy in a non-preemptive system, as the execution of the second job will strictly follow the execution of the first.

There are a number of ways to decide the process that will be executed next in case of preemptive scheduling, some of which are as follows:

First come, first serve In the first come, first serve (FCFS) technique, the process that enters the memory first will be executed first. A ready queue is maintained such that it contains all the processes that need the CPU time. The scheduling policy is very simple to implement as it just takes the first process, executes it, and then allots the CPU to the next process in sequence. The main drawback of the FCFS policy is poor performance and long waiting times for small processes.

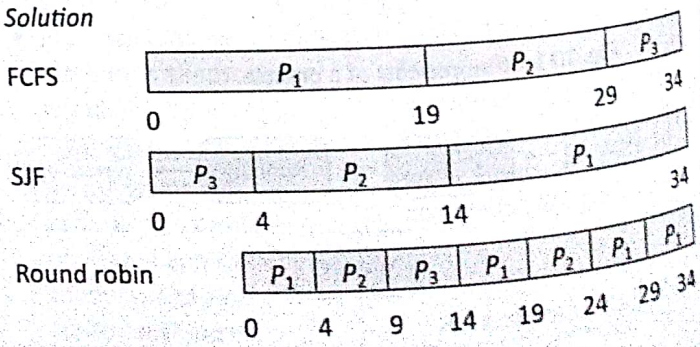
For example, if the first process needs 23 ms to execute and the second process needs only 5 ms, the second process will have to wait for 23 ms for the complete execution of the first process.

Shortest job first In this technique, the CPU time requirements of each process must be known in advance so that a queue can be maintained such that it stores the processes in increasing order of their CPU time requirements. Though this is the best approach to follow, it is very difficult to implement, as prior knowledge of the time required cannot always be known in advance. The technique stresses on executing the shortest job first (SJC), and hence, it may lead to starvation of long jobs that had entered very early in the system.

Round robin In this technique, the CPU is allotted to each process on the FCFS basis for a specified time (called the time quantum). If the first process is completely executed in this time quantum, it is removed from the ready queue; otherwise, it is suspended and the CPU is given to the second process for the same time quantum. This technique is repeated until all the processes are completely executed.

Example 7.1 Consider three processes P_1, P_2, and P_3 with CPU time requirements given as 20 ms, 10 ms, and 5 ms. Show the scheduling process using FCFS, SJF, and round robin policies. Take time quantum as 5 ms.

Solution



Scanned by CamScanner

7.3.4 Process Synchronization

Process synchronization includes synchronizing the execution of processes in such a way that no two processes have access to the same shared data or resource. In general, when multiple processes are running concurrently, they may need the same data or resource at the same time. Such a situation will always lead to inconsistency of data and ambiguous and arbitrary use of the system's resources.

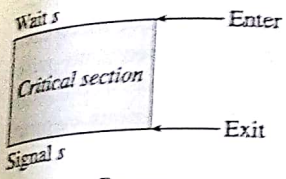


Fig. 7.15 Process synchronization

Usually, every program has a critical section (Figure 7.15), which includes a set of instructions that may change the value of a global variable, write modified data to a file, and so on. In order to ensure that only a single process executes the critical section at a given instant of time, the processes must coordinate with each other so that when one process is in the critical section, no other process can enter the critical section.

One solution to the critical solution problem is that the process must acquire a lock on the system's resources, which it needs before it enters the critical section. After it exits from the critical section, the lock can be released so that other processes waiting to enter the critical section can proceed with execution.

7.3.5 Interprocess Communication

Interprocess communication (IPC) allows interaction between processes. The processes communicating with each other may be running on the same computer or on different computers connected through a network. IPC allows one process to exhibit control over another process and enables several processes to share the same data without interfering with one another.

The two commonly used techniques of IPC are *shared memory* and *message passing*. In shared memory, the processes that want to communicate with each other share a part of memory, as shown in Figure 7.16. The process that wants to share data with other processes will write it in the shared memory. Correspondingly, the processes that want to read that data will read it from the shared memory. However,

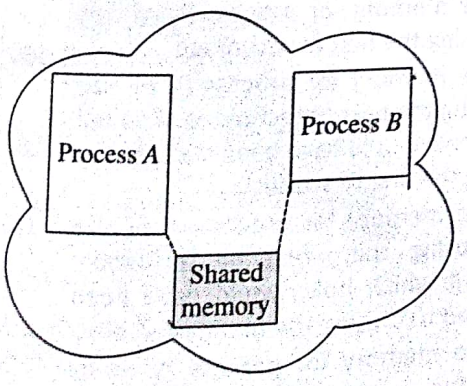


Fig. 7.16 Shared memory

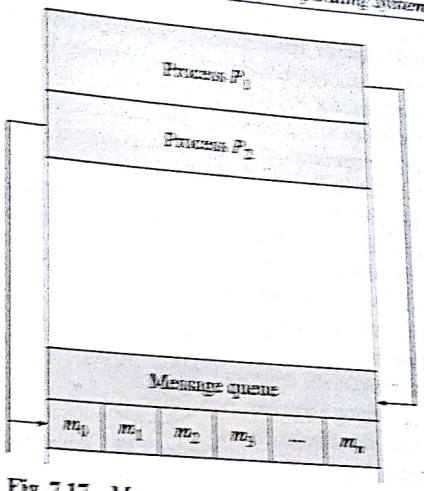


Fig. 7.17 Message passing

once shared memory is established, all accesses to the shared memory are treated as routine memory accesses, and no special help is required from the operating system.

In message passing technique (Figure 7.17), the process that wants to share data or communicate with other processes sends messages to those processes. For this communication to take place, the operating system maintains a message queue such that a message sent by one process can be received by other interacting processes in the system.

Message passing is easier to implement than shared memory and is very helpful for exchanging smaller amounts of data. On the other hand, shared memory is faster and allows maximum speed and convenience of communication as it can be done at memory speeds when within a computer.

7.3.6 Deadlock

A deadlock is a situation that occurs when multiple processes are waiting for each other to free the resources they have acquired, and as a consequence, all of them remain halted (Figure 7.18). For example, suppose process P_1 is waiting for a resource (say, a printer) acquired by process P_2 to complete its execution and process P_2 is waiting for a shared data that has been locked by process P_1. In this scenario, both the processes are waiting for each other to free a resource so that its execution can be completed but no process is able to proceed.

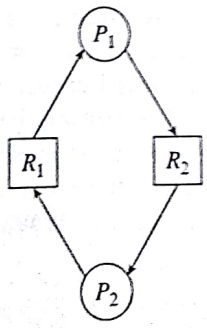


Fig. 7.18 Deadlock

Usually, four conditions result in a deadlock.

Mutual exclusion In the mutual exclusion mode, processes cannot share resources. Hence, if one process has acquired a resource, then the other process trying to access that resource must wait until the process currently using that resource completes its execution.

Scanned by CamScanner

Hold and wait In this situation, a process holds a resource and waits for another resource that is currently being acquired by another process.

No preemption If a process needs a resource that is currently held by another process, then that process cannot force the other process to release that resource. The resource can only be voluntarily released by the process that has currently acquired it.

Circular wait In a system, when process P_i is waiting for process P_j where $i = (i + 1)\%(n + 1)$, a circular wait of processes exists. Here, n is the number of processes in the system.

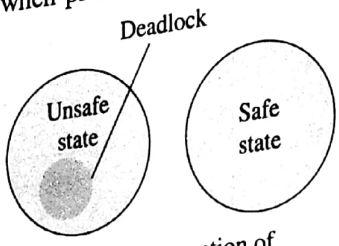


Figure 7.19 Prevention of deadlocks by the operating system

The operating system can prevent deadlocks in the system (Figure 7.19) by following any of the following three strategies:

Deadlock prevention or avoidance When following this technique, the operating system does not allow the system to get into a deadlock state. It maintains all necessary information about all processes such as the resources required, the number of units of resources required, and the schedule or the sequence in which these resources are required. With the help of this information, the operating system ensures that at least one of the aforementioned conditions for the deadlock to happen does not persist in the system. Therefore, when the operating system identifies that starting a process or granting resources to a process may lead to future deadlocks, then either the process will not be started or its request for resources will not be granted.

Deadlock detection and recovery When nothing is done to avoid a deadlock and a deadlock has been detected in the system, the operating system either aborts a process or preempts some resources from that process. Although deadlock detection is fairly simple and straightforward, deadlock recovery, which may call for aborting processes or preempting resources, is not a good alternative.

Deadlock ignorance In this strategy, it is assumed that deadlock will never occur. This is because deadlocks seldom occur in the system, for example, once in a year. Hence, it is better to simply let them happen and reboot the system when the need arises than to incur constant overhead and impact system performance while following deadlock prevention or detection strategies. This approach is followed in both Windows and Unix operating systems.

Note All deadlocks are unsafe but not all unsafe states are deadlocks.

7.4 MEMORY MANAGEMENT

In a single-user single-task operating system such as DOS, only one process can be executed at a time. Hence, the memory space is available for only one process. After that process is completely executed, it frees the memory, which is then allocated to another process. However, in multiprogramming systems, multiple programs reside in the memory to be executed by the CPU. Therefore, every process that needs to be executed requires a certain amount of space in the main memory. It is the job of the operating system to allocate sufficient memory to each process and correspondingly de-allocate the previously allocated memory when the process is completely executed. The de-allocated space is then made available for use by other processes.

Besides allocation and de-allocation of memory, the operating system also ensures that processes do not share each other's memory space and applies defragmentation techniques to utilize small chunks of vacant spaces in the main memory. An operating system can support various memory allocation schemes, which are explained in this section.

7.4.1 Multiple Partition Allocation

In the multiple partition allocation technique, the operating system keeps a list of the parts (also called partitions, blocks, or holes) of memory that are free and thus can be allocated to the user's programs. When a process requires memory, the list of holes is scanned and a hole that is large enough for the process is allocated to it. Correspondingly, when a process is completely executed and it releases its allocated memory, the released block of memory is added in the set of holes.

There are three ways to decide which hole should be allocated to the process requesting it:

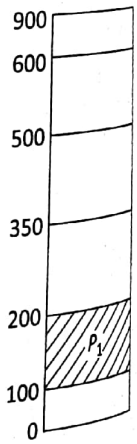
First fit According to this technique, the first hole that is big enough for the process is allocated to the requesting process.

Best fit This technique allocates to the process the smallest hole that is big enough to meet its memory requirements.

Worst fit In this technique, the process is allocated the largest hole available in the memory.

Though the multiple partition allocation scheme seems to be simple, its major drawback is that it results in *memory fragmentation*. Fragmentation occurs during memory allocation and de-allocation when the free memory is broken into small pieces and remains unused by any process.

Example 7.2 Consider a scenario of memory of a computer system as in the below given figure where only one process P_1 resides in the main memory.



(a) Allocate memory for process P_2 of size 200K using the first fit technique.

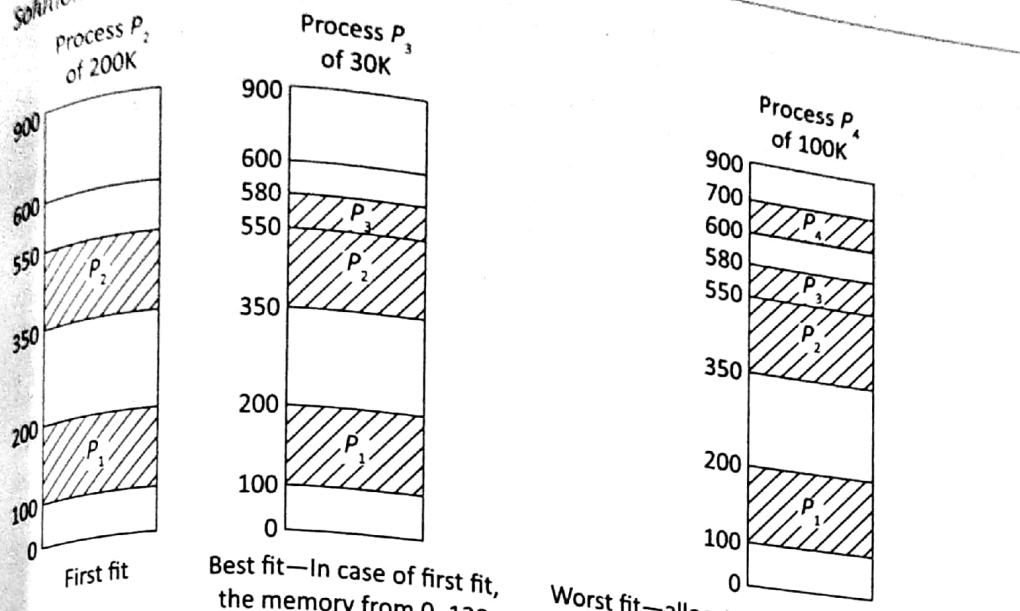
(b) Allocate memory for process P_3 of size 30K using the best fit technique. Also tell which hole would have been allocated to it using the first fit strategy.

(c) Allocate memory for process P_4 of size 100K using the worst fit technique. Also tell which hole would have been allocated to it using the first fit and best fit strategies.

(d) Allocate memory for process P_5 of size 500K using any technique.

Scanned by CamScanner

Solution



First fit

Best fit—In case of first fit, the memory from 0–130 would have been allocated.

Worst fit—allocated memory from 600–700K, In case of first fit 0–100K, position will be allocated. In case of best fit, again 0–100k role would have been selected.

Note that even though 500K memory space is available in the system, it cannot be allocated to process P_5 because the available memory is fragmented. This problem is called memory fragmentation and can be handled well by paging and segmentation techniques.

7.4.2 Paging

An address generated by the CPU is called a logical address or a virtual address, and the set of all logical addresses generated by a program is called the logical address space. Correspondingly, the address actually available in the main memory is called the physical address, and the set of all physical addresses is called the physical address space.

The operating system provides memory protection with the help of two registers. The first is the base register, which holds the smallest physical memory address allocated to a process. The second register is the limit register, which specifies the size of the address allocated to the process. For example, if the base register holds 2000 and the limit register is 1000, then the program can legally access all addresses from 2000 through 3000 (Figure 7.20). The paging and segmentation schemes strictly apply the implementation of base and limit registers to protect the memory from illegal access.

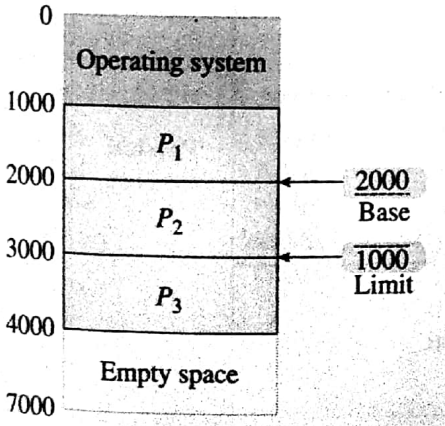


Fig. 7.20 Base and limit registers

In paging, the main memory is divided into fixed size blocks called *frames*. Similarly, the user's program is divided into fixed size blocks called *pages*. In general, the size of a page is equal to the size of a frame, which usually varies from 1 KB to 8 KB. When a process has to be executed, its pages are loaded into the available memory frames. The operating system keeps track of all the frames available in the memory. To execute a program consisting of n pages, the operating system needs n free frames in the memory. In paging technique, the logical address generated by the CPU is divided into two parts:

Page number (p) It acts as an index to the page table, which contains the base address of each page in the main memory.

Page offset (d) The page offset is added to the base address to calculate the physical memory address.

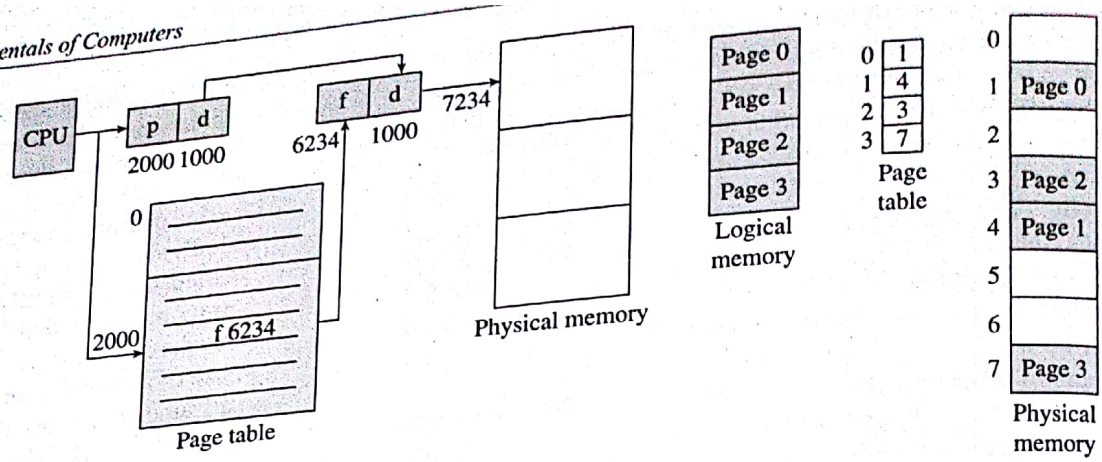
The paging technique handles the problem of fragmentation by allowing a program's pages to be loaded anywhere in the memory (not necessarily in consecutive frames). Consider Figure 7.21(a) in which the logical address has 2000 as the page number and 1000 as the page offset. The page table is scanned to find the entry corresponding to page number 2000. In the page table, we find the value 6234. This value is added with the page offset to generate the physical address.

Figure 7.21(b) shows a scenario in which a program has four pages. These pages are loaded one each in frames 1, 4, 3, and 7. Note that in the page table, the page number can be used as an index to locate the frame number in the physical memory.

7.4.3 Segmentation

The paging technique suffers from internal fragmentation, which means a certain amount of space is wasted within the memory frame. This occurs when the allocated memory or frame is slightly larger than the requested memory. For example, if a page (frame) size is 4K but the program needs only 2K, then the entire frame of 4K will be allocated to the program out of which 2K will be used and 2K will be wasted. This problem of internal fragmentation can be avoided by using another important technique to manage memory, namely *segmentation*.

Scanned by CamScanner



(a) Paging

(b) Pages can be loaded in any frame in memory

Fig. 7.21 Paging technique

Another reason for using segmentation is that programmers view an application as a collection of related program units, where each unit has the following characteristics:

- It is an independent entity.
- It contains closely related functions or data.
- It may use the services of other units.
- It may vary in size.

Segmentation supports the user's view of an application by breaking the memory into logical pieces, where each piece represents a group of related information, as shown in Figure 7.22. It can be seen from the figure that each segment has a base address and a limit. Like in paging, in segmentation the CPU generates a logical address, which consists of two parts:

Segment number (s) It is used as an index to the segment table, which contains the base address of each segment in the physical memory and its limit.

Segment offset (o) It is first checked to determine whether it is within the limit specified in the segment table. If yes, then the offset is added to the base address to generate the physical address (refer Figure 7.23).

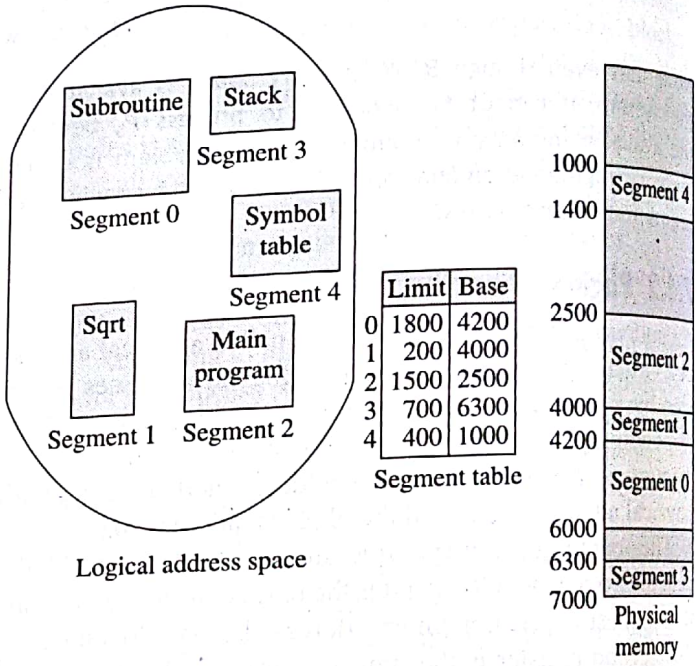


Fig. 7.22 Segmentation

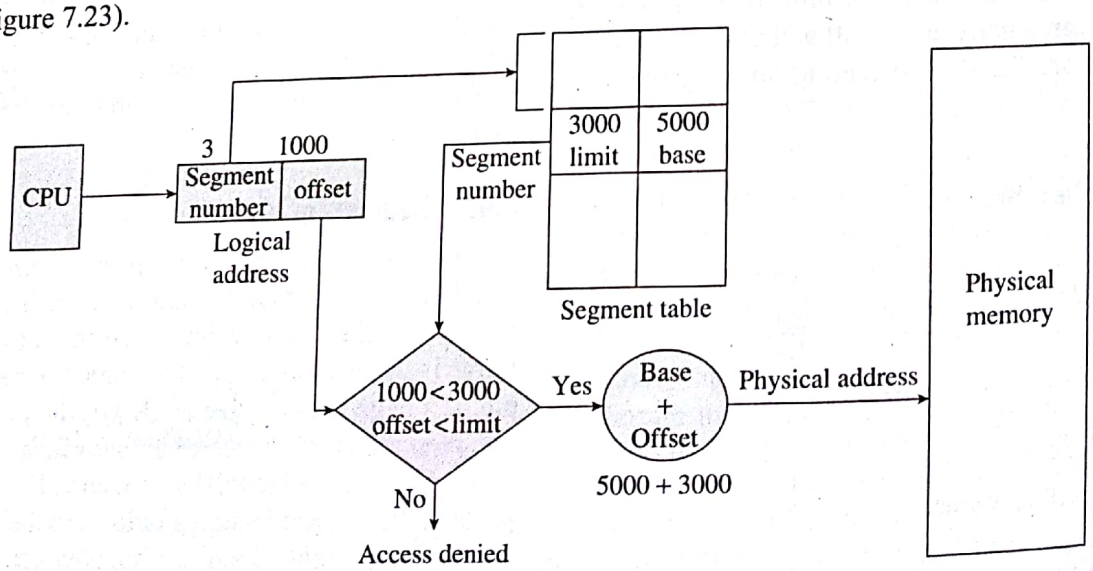


Fig. 7.23 Example for segmentation

Scanned by CamScanner

7.5 FILE MANAGEMENT

In computer terminology, a file is a block of useful data that is available to a computer program and is usually stored on a persistent storage medium such as a hard disk. Storing a file on a persistent storage medium ensures the availability of the file for future use. Today, the files on computers are a good alternative to store documents that were once stored in offices and libraries.

7.5.1 Data Hierarchy

Every file contains data. This data can be organized in a hierarchy to present a systematic organization. Data hierarchy includes terms such as fields, records, files, and database. These terms can be defined as follows:

- A *data field* is an elementary unit that stores a single fact. It is usually characterized by its type and size. For example, student's name is a data field that stores the name of a student. This field is of type characters and its size can be set to be of maximum 20 or 30 characters, depending on the need.
- A *record* is a collection of related data fields that is seen as a single unit from the application point of view. For example, a student record may contain data fields such as the name of the student, address, phone number, roll number, and marks obtained.
- A *file* is a collection of related records. For example, if there are 60 students in a class, then there are 60 records of the students. All these related records are stored in a student's file. Similarly, we can have a file of all the employees working in an organization, a file of all the customers of a company, a file of all the suppliers, and so on.
- A *directory* stores information of related files. A directory organizes information so that the user can find it easily. For example, consider Figure 7.24 that shows how multiple related files are stored in a student directory.

7.5.2 File Attributes

Every file in a computer system is stored in a directory. Each file has a list of attributes associated with it; these attributes give more information to the operating system and the application software about the file and its intended use.

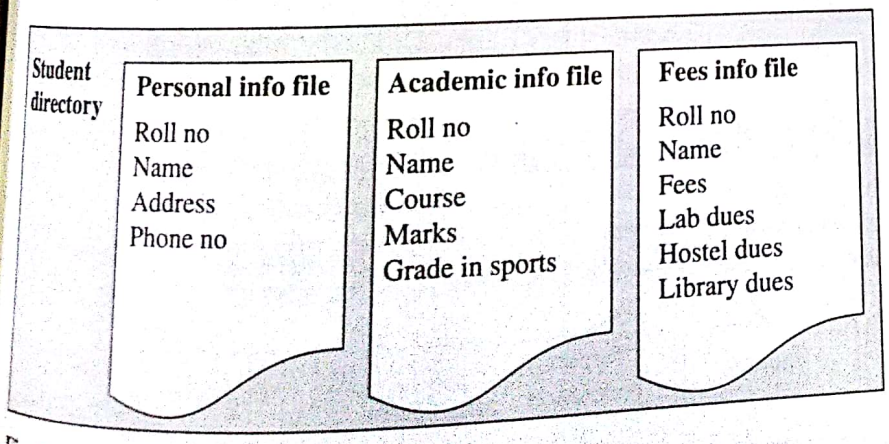


Fig. 7.24 Example of a student directory

A software program looks up in the directory entry to discern the attributes of a file and to take decisions about how to treat the file. For example, if a program attempts to write to a file that has been marked as a *read-only* file, then the program prints an appropriate message to notify the user that he/she is trying to write to a file that is meant only for reading.

Similarly, there is an attribute called *hidden*. When a user attempts to execute the DIR command in DOS or sees a list of all the files in Windows, the files whose hidden attribute is set will not be displayed. The various attributes of a file can be explained as follows:

File name This is a string of characters that stores the name of the file. File naming conventions vary from one operating system to the other.

File position This is a pointer that points to the position at which the next read/write operation will be performed.

File structure This indicates whether the file is a text file or a binary file. In a text file, the numbers (integers or floating points) are stored as a string of characters; a binary file, on the other hand, stores numbers in the same way as they are represented in the main memory.

File access method This indicates whether the records in the file will be accessed sequentially or randomly. In the sequential access mode, records are read one by one that is, if records of 60 students are stored in a *Student* file, then to read the record of the 39th student, the user has to go through the records of the first 38 students. However, in random access, records can be accessed in any order.

Table 7.2 Attribute flag

Attribute	Attribute byte
Read-only	00000001
Hidden	00000010
System	00000100
Volume label	00001000
Directory	00010000
Archive	00100000

Attributes flag A file can have six additional attributes attached to it. These attributes are usually stored in a single byte, with each bit of the byte representing a specific attribute. Each bit that is set to a one means that the file has that attribute turned on. Table 7.2 shows the list of attributes and their position in the attribute flag or attribute byte.

Therefore, if a system file is set to hidden and is a read-only file, then its attribute byte can be given as 00000111. We will discuss all these attributes in detail in this section. Note that directory is treated as a special file in the operating system. Hence, all these attributes are applicable for files as well as for directories.

Scanned by CamScanner

Read-only A file marked as read-only cannot be deleted or modified. For example, if an attempt to either delete or modify a read-only file is made, then a message 'Access denied' is displayed to the user.

Hidden A file marked as hidden is not displayed in the directory listing.

System A file marked as a system file indicates that it is an important file used by the system and should not be altered or removed from the disk. Hence, it is not wrong to say that it is like a 'more serious' read-only flag.

Volume label Every disk volume is assigned a label for identification. The label can be assigned at the time of formatting the disk or later through various tools such as the DOS command 'LABEL'.

Directory In directory listing, the files and the sub-directories of the current directory are differentiated by a directory-bit. Hence, files that have the directory-bit turned on are actually sub-directories containing one or more files. Theoretically, one can convert a file to a directory by changing this bit. However, practically, trying to do this would lead to complications.

Archive The archive bit is useful for communication between programs that modify files and programs that are used for backup. Most of the backup programs available today enable the users to do an incremental backup, which selects for backup only the files that were modified since the last backup.

When a file is created, the archive bit is turned on. When the backup program takes the backup of a file, or in other words, when the program archives the file, it clears the archive bit (sets it to 0). Subsequently, if any program modifies the file, it turns on the archive bit (sets it to 1). Then, whenever the backup program is run, the backup program would check the archive bit of each file to identify the files that have been modified since its last run. It will then archive only those files that were modified. After taking the backup, it would turn off the archive bit. Thus, the archive attribute is used for taking backup.

7.5.3 Basic File Operations

The basic operations that can be performed on a file are given in Figure 7.25.

Creation A file is created by specifying its name and mode. Then, the file is opened for writing the records that are read from some input device (e.g., a keyboard). Once all the records have been written into the file, the file is closed. The file is now available for future read/write operations by any program that has been designed to use it.

Updation Updating a file means changing the contents of the file to reflect a more current picture of reality. A file can be updated in the following ways.

- Insert a new record in the file; for example, if a new student joins the course, we will need to add his/her record as well in the Student file.
- Delete an existing record; for example, if a student quits a course in the middle of the session, his/her record needs to be deleted from the Student file.
- Modify an existing record; for example, if the name of a student was initially wrongly spelt, then correcting the name will be a modification of the existing record.

Retrieval Retrieving from a file means extracting useful data from a given file. Information can be retrieved from a file either for an enquiry or for report generation. An enquiry for some data retrieves low volume of data, whereas report generation may retrieve a large amount of data from the file.

Maintenance Maintenance of a file involves restructuring or reorganization of the file to improve the performance of the programs that access that file. Restructuring the file keeps the file organization unchanged and changes only the structural aspects of the file (e.g., changing the field width, adding or deleting fields); on the other hand, file reorganization may involve changing the entire organization of the file. File organization is discussed in detail in Section 7.5.4.

7.5.4 File Organization

As mentioned earlier, a file is a collection of related records. The main issue in file management is the way in which the records are organized inside the file because organization of records heavily affects system performance. Here, by the term *organization of records*, we mean the *logical* arrangement of the records in the file (e.g., based on their ordering or the placement of related records close to each other in the file) and not the physical layout of the file as stored on a storage media.

Choosing an appropriate file organization is a design decision; hence, it must be done keeping in mind the priority of achieving good performance with respect to the most likely usage of

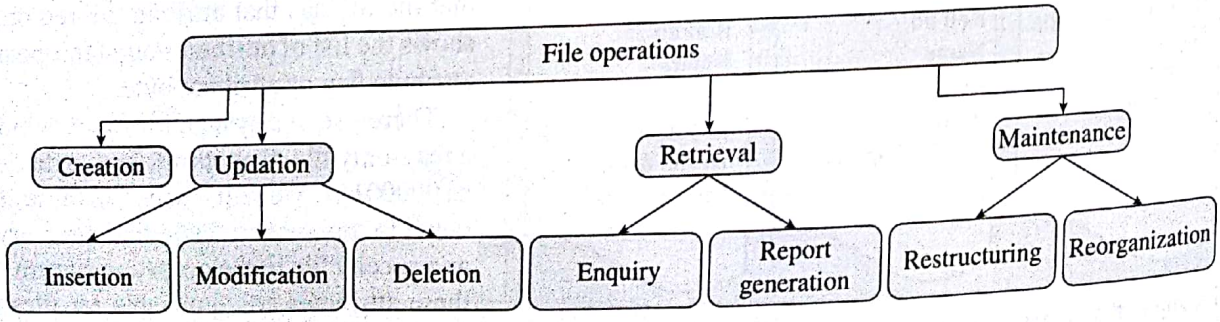


Fig. 7.25 Basic file operations

Scanned by CamScanner

the file. Therefore, the following criteria need to be considered before selecting an appropriate file organization method:

- Rapid access to one or more records
- Ease of inserting, updating, or deleting one or more records without disrupting the speed of accessing record(s)
- Efficient storage of records
- Using redundancy to ensure data integrity

Though these requirements are in contrast with each other, it is the designer's job to find a good compromise among them to get an adequate solution to the problem at hand. For example, ease of addition of records can be compromised to get fast access of data.

In this section, we will discuss some of the file organization techniques that can be considered for a particular situation at hand.

Sequential Organization

A sequentially organized file stores records in the order in which they are entered. Thus, the first record that is entered is written as the first record in the file, the second record entered is written as the second record in the file, and so on. As a result, new records are added only at the end of the file.

Sequential files can be read only sequentially, starting with the first record in the file. Sequential file organization is the most basic way to organize a large collection of records in a file. Figure 7.26 shows n records numbered from 0 to $n-1$ stored in a sequential file.

Once the records are stored in the file, we cannot make them shorter or longer. We cannot even delete the records from a sequential file. However, a file can be updated but only if the length does not change. In case we need to delete or update one or more records, we have to replace the records by creating a new file.

In sequential file organization, all the records have the same size and the same field format, and every field has a fixed size. The records in the file are sorted based on the value

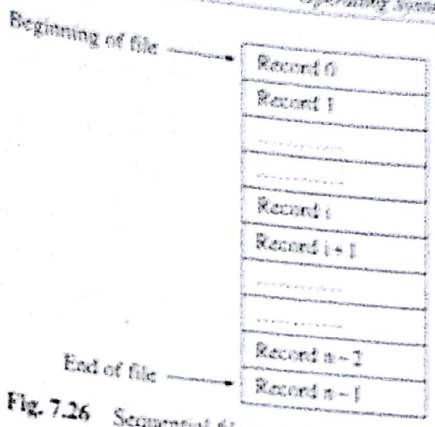


Fig. 7.26 Sequential file organization

of one field or a combination of two or more fields. This field is known as the *key*. Every key uniquely identifies the records in the file. Thus, every record has a different value for the key field. Records can be sorted in ascending or descending order.

Sequential files are usually used for report printing or sequential reads of a large amount of data, which some programs prefer to do, for example, payroll processing of all the employees of an organization. Sequential files can be easily stored on both disks and tapes. Table 7.3 summarizes the features, advantages, and disadvantages of sequential file organization.

Relative File Organization

Relative file organization provides an effective way to directly access individual records. In this type, the records in a file are ordered by their *relative key*. Thus, the record number represents the location of the record relative to the beginning of the file. The records are numbered from 0 to $n-1$, where n is the number of records in the file. For example, the record with number 0 is the first record in the file. The records in a relative file are of fixed length.

Therefore, in relative files, records are organized on the basis of ascending relative record number. A relative file is similar to a single-dimension table stored on a disk, in which the relative record number is the index to the table.

Table 7.3 Features, advantages, and disadvantages of sequential file organization

Features	Advantages	Disadvantages
• Records are written in the order in which they were entered. • Records are read and written sequentially. • Deleting or updating one or more records calls for replacing the original file with a new file that contains the desired changes. • Records have the same size and the same field format. • Records are sorted on a key value. • It is usually used for report printing or sequential reads.	• It is simple and easy to handle. • No extra overheads are involved. • Sequential files can be stored on magnetic disks as well as magnetic tapes. • It works well for batch-oriented applications.	• Records can be read only sequentially. If an ith record has to be read then all $i-1$ records must be read. • It does not support updating records in the same file. A new file has to be created and the original file has to be replaced with the new file that contains the desired changes. • It cannot be used for interactive applications.

Scanned by CamScanner

Relative files can be used for both random accesses of data and sequential access. For sequential access, records are simply read one after the other.

Relative files provide support for only one key, that is, the relative record number. This key must be numeric and must take a value between 0 and the current highest relative record number −1. Hence, enough space must be allocated for the file to contain records with relative record numbers between 0 and the highest record number −1. For example, if the highest relative record number is 1000, then space must be allocated to store 1000 records in the file.

Figure 7.27 shows a schematic representation of a relative file that has been allocated enough space to store 100 records. Although it has space to accommodate 100 records, not all the locations are occupied. The locations marked as FREE are yet to store records in them. Therefore, every location in the table either stores a record or is marked as FREE.

Relative file organization provides random access of data by directly jumping to the record that has to be accessed. For example, if the records are of fixed length and we know that each record occupies 20 bytes and the base address of the file is 1000, then any record *i* can be accessed using the following formula.

```
Address of ith record = Base_address + (i - 1) *
record_length
```

Therefore, if we have to access the fifth record, then the address of the fifth record can be given as

```
1000 + (5 - 1) * 20
= 1000 + 80
= 1080
```

Note that the base address of the file means the starting address of the file. We took *i* −1 in the formula because the record numbers start from 0 rather than 1. Hence, the fifth record is actually the record with record number 4. Table 7.4 summarizes the features, advantages, and disadvantages of relative file organization.

Relative record number	Records stored in memory
0	Record 0
1	Record 1
2	FREE
3	FREE
4	Record 4
.....	.....
98	FREE
99	Record 99

Fig. 7.27 Relative file organization

Indexed Sequential File Organization

Indexed sequential file organization stores data for fast retrieval. The records in an indexed sequential file are of fixed length, and every record is uniquely identified by a key field. We maintain a table known as an index table to store the record number and the address of each record in the file. Every file has an index table. This type of file organization is called an indexed sequential file organization, because physically the records may be stored anywhere but the addresses of those records are stored in the index table.

The *i*th entry in the index table points to the *i*th record of the file. Initially, when the file is created, each entry in the

Table 7.4 Features, advantages, and disadvantages of relative file organization

Features	Advantages	Disadvantages
• It provides an effective way to access individual records. • The record number represents the location of the record relative to the beginning of the file. • Records are of fixed length. • Relative files can be used for both random accesses of data and sequential access. • Every location in the table either stores a record or is marked as FREE.	• Processing is easy. • If the relative record number of the record that has to be accessed is known, then the record can be accessed instantaneously. • Random access of records helps fast access to relative files. • It allows deletions and updates in the same file. • It provides random as well as sequential access of records with low overhead. • New records can be easily added in the free locations based on the relative record number of the record to be inserted. • It is well suited for interactive applications.	• Use of relative files is restricted to disk devices. • Records can be of fixed length only. • For random access of records, the relative record number must be known in advance.

Scanned by CamScanner

index table contains NULL. When the ith record of the file is written, any free space is obtained from the free space manager and its address is stored in the ith location of the index table. Figure 7.28 shows this scheme.

Record no.	Address of the record
1	765
2	27
3	876
4	742
5	Null
6	Null
7	Null
8	Null
9	Null

Fig. 7.28 Indexed sequential file

Now, if one has to read the fourth record, then there is no need to access the first three records. The address of the fourth record can be obtained from the index table and the record can be directly read from the specified address (742, in our example). Conceptually, the indexed sequential file organization can be visualized as shown in Figure 7.29.

The indexed sequential file uses the concept of sequential files as well as that of relative files. The index table is read sequentially to find the address of the desired record, whereas a direct access is made to the address of the specified record in order to access it randomly.

Indexed sequential files perform well in situations where sequential access as well as random access is made to the data. They can be stored only on devices that support random access of data, such as a magnetic disk. This type of organization is used in situations such as a college where the details of

Record no.	Pointer to the record
1	
2	
3	
4	
5	Null
6	Null
7	Null
8	Null
9	Null

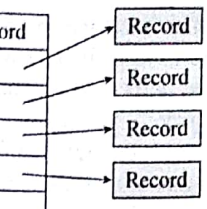


Fig. 7.29 Indexed sequential file organization

its students are stored as an indexed sequential file. The file can be accessed in either of the following ways:

- Sequentially to print the aggregate marks obtained by each student in a particular course
- Randomly, for example, to modify the name of a particular student

Table 7.5 summarizes the features, advantages, and disadvantages of indexed sequential file organization.

7.6 DEVICE MANAGEMENT

A computer can be connected to different types of peripheral devices such as a keyboard, mouse, hard disk, printer, and plotter. Some of these devices may send or receive data character by character, whereas others may do it block by block; some allow sequential access, whereas others support random access of data; some operate in a dedicated way, whereas others may be shared amongst two or more computers. Hence, in such a big diversity of implementation, it is the job of the operating system to manage and control the

Table 7.5 Features, advantages, and disadvantages of indexed sequential file organization

Features	Advantages	Disadvantages
• It provides fast data retrieval. • Records are of fixed length. • The index table stores the address of the records in the file. • The ith entry in the index table points to the ith record of the file. • The index table is read sequentially to find the address of the desired record, whereas a direct access is made to the address of the specified record in order to access it randomly. • Indexed sequential files perform well in situations where sequential access as well as random access is made to the data.	• The key improvement is that indexes are small and can be searched quickly, allowing the database to access only the records it needs. • It supports applications that require both batch and interactive processing. • Records can be accessed sequentially as well as randomly. • It updates the records in the same file.	• Indexed sequential files can be stored only on disks. • It needs extra space and overhead to store indices. • Handling these files is more complicated than handling sequential files. • It supports only fixed length records.

Scanned by CamScanner

devices connected to the computer. Go to Device Manager in the Control Panel to see how interactively the operating system allows users to manipulate the devices.

Figure 7.30 shows that the operating system communicates with the peripheral devices via the *device driver software* that comes along with each device. The device drivers reduce the complexity of handling the devices and present a uniform interface to the operating system. The device is plugged into a computer on a *port* (a serial or a parallel port). Other important terms in device management include bus and

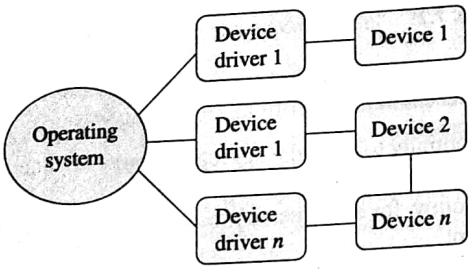


Fig. 7.30 Interaction of operating system with devices through device drivers

device controller. A *bus* is a set of wires that connects multiple devices. It follows a defined protocol that specifies a set of messages that can be sent on the wires.

A device controller operates a port, bus, or device and acts like a bridge between the device and the operating system. It accepts data from the device, stores it temporarily, and then sends it to its device driver.

Besides managing the peripheral devices, the operating system also provides various services related to I/O such as I/O scheduling, buffering, spooling, and error handling.

Scheduling I/O requests The operating system reschedules or reorders the I/O requests sent by the processes to improve the performance of the system. While rescheduling, it takes enough care to ensure that fair access is provided to all the processes. A device request queue is maintained in which the order of the elements can be rearranged to schedule the I/O requests.

Buffering Buffer is a small area of memory that stores the data while it is being transferred between two devices or between a device and an application. Since the speed of the CPU is much higher than that of the I/O devices, the I/O devices transfer data at a relatively slow speed than the speed at which the data is processed. Therefore, to handle the speed mismatch, the operating system stores the data in a buffer until the complete data has arrived and then sends it to the processor in a single write operation.

Simultaneous peripheral operation online (spooling) The operating system takes care of spooling to store the jobs in a spool, which the device can access when it is ready. This is frequently observed when taking printouts. The user may give several print commands together and then continue working with other applications. The printer will take each print command one by and one and process it when it is ready.

Polling In polling, the CPU waits for a device to be ready for an I/O operation. It repeatedly reads the status register of the device and waits until the *busy bit* in the status register is cleared. The busy bit when set indicates that the device is busy performing some operation and thus cannot respond to any other command. Once the device is ready, the device controller clears the error bit in the status register to indicate that the device I/O has succeeded and clears the busy bit to indicate that it is finished.

Direct memory transfer When a large amount of data has to be transferred between a peripheral device and the memory, it is better to transfer the data directly between the two units without unnecessarily involving the CPU. This approach is called direct memory access (DMA). DMA is often useful on devices such as disks, as several bytes of information can be transferred in a single I/O operation.

Interrupt handling An interrupt is a signal sent by the peripheral devices to the CPU to notify it to suspend the current process and execute the appropriate part of the operating system. The operating system has an *interrupt handler* that prioritizes the interrupts and saves them in a queue when more than one interrupt occurs in the system (Figure 7.31). The operating system also decides which program should be executed next. There are three types of interrupts:

Hardware interrupts are generated by hardware devices to signal the CPU that they need some attention from the operating system. These interrupts are generated when an I/O operation is completed such as reading some data into the computer from a tape drive or receiving some keystrokes on the keyboard or when DMA is completed.

Software interrupts are generated by programs in two cases: (a) when the program has terminated and (b) when the program requests some services from the operating system.

Traps are generated by the CPU itself to signal the operating system that an error has occurred and the CPU needs its assistance to resolve the problem.

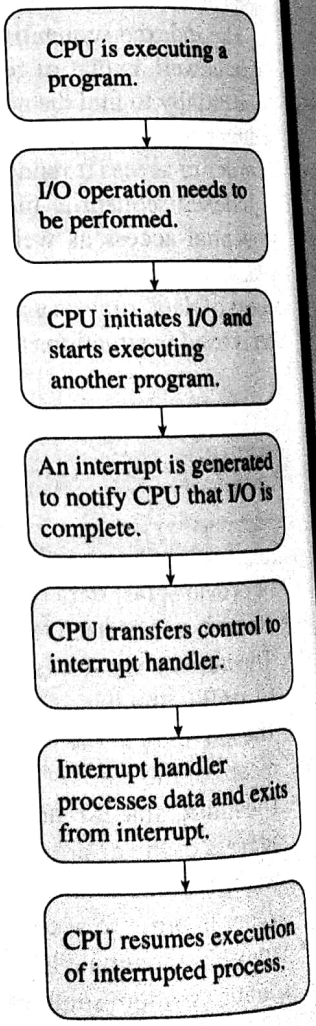


Fig. 7.31 Interrupt handling

Scanned by CamScanner

7.7 SECURITY MANAGEMENT

In operating systems, security means providing protection to the computer's resources such as the CPU, memory, disk, programs, and data. The operating system provides security at two levels (Figure 7.32).

Internal security Protection through internal security ensures that the resources of the computer are used in a consistent way. Thus, while implementing internal security, the operating system protects one process from another. In addition, it ensures that only one process is using a particular resource at a given instant of time. The operating system also assigns a privilege level to each process. If two or more processes simultaneously request for a resource, then the process with the highest privilege level will be serviced first.

External security External security must be implemented by the operating system especially when the system is connected to a network or to the Internet. This can be provided through the following ways:

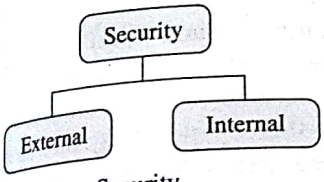


Fig. 7.32 Security

Authentication It refers to identifying every single user of the system so that only authentic users can execute programs on the computer. Operating systems identify or authenticate users in the following three ways:

Username and password Users are asked to enter a username and password to log on to the system.

User card or key Users who want to log on to the system must punch card in the card slot or enter a key generated by the key generator.

Use physical and unique attributes To log on to the system and access its resources, users must pass some physical and unique attribute such as the fingerprint, eye retina pattern, or signature via the designated input device.

Program threats It is the responsibility of the operating system to control and manage the devices, data, and resources of the computer. Nevertheless, a user can write and execute programs that can send the user credentials via the network to some hacker. Such programs are called program threats; they include the following:

- Trojan horse traps user login credentials and stores them so that they can be sent to another user, thereby allowing him/her to log on to the computer and access the system resources and use them inappropriately.
- Trap door is a program that is designed to perform a specific task. However, it has a security hole in its code that performs illegal actions without the knowledge of the user.
- Logic bomb is a program that misbehaves only when certain conditions hold true; otherwise, it works as a genuine program. Such programs are difficult to detect.
- Virus is a piece of code embedded in a program that can replicate itself on a computer system and modify or delete user files and even crash systems. When a user accesses the program, the virus gets embedded in other files or programs and makes the system unusable.
- Worm is a program that is used to choke down a system's performance by using its resources to extreme levels. A worm program regenerates its code multiple times where each copy of the code uses the system's resources, thereby preventing other important programs to get their share of the resources. In the worst case, worms can even shut down an entire network.
- Port scanning is a mechanism by which a hacker can identify vulnerabilities to attack a system.
- Denial of service attacks prevents a user to make legitimate use of the system.

7.8 COMMAND INTERPRETER

The command interpretation module (also called the command interpreter) of the operating system provides a set of commands that the users can execute. These commands are often called the system calls. When a user executes a system call, the command interpreter interprets the instructions and allocates the system's resources to handle the user's request. The command interpreter also provides an easy to use intuitive interface to the users, thereby hiding the internal complexities and fulfilling the operating system's objective of 'ease of use'.

The user interface of the command interpreter can be broadly classified into two groups—CLI and GUI (Figure 7.33).

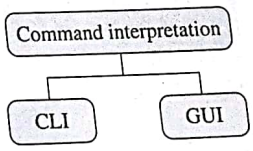


Fig. 7.33 Command interpreter

Note The command interpreter acts as the interface between the user and the operating system.

The CLI, also called the command line user interface, console user interface, character user interface, or text interface, is a way by which users can interact with a program by typing commands at the prompt. Such an interface makes use of only the keyboard to issue commands.

Although the CLI has been widely replaced by the GUI, advanced computer users still prefer CLI because of the following advantages:

- It provides a more concise and powerful means to control a program or operating system.
- Programs with the CLI can be easily automated through scripting.
- It is very valuable for some types of operations. For example, renaming 100 files in Windows based on the GUI

Scanned by CamScanner

will take a long time, but with the CLI, the same operation can be performed within a few seconds.

- When sharing resources over a network, the CLI is widely used because some networking devices can be manipulated only using the CLI.
- The CLI is stable in the sense that once a user has learnt the executing commands, the procedure will never change (as compared to GUI). Even if new commands are introduced, the original ones will always remain the same.

Figure 7.34 shows a menu and shortcut keys driven CLI through which the users can easily navigate using their keyboard. However, the major drawback of the CLI is that users need to memorize commands to interact with the programs.

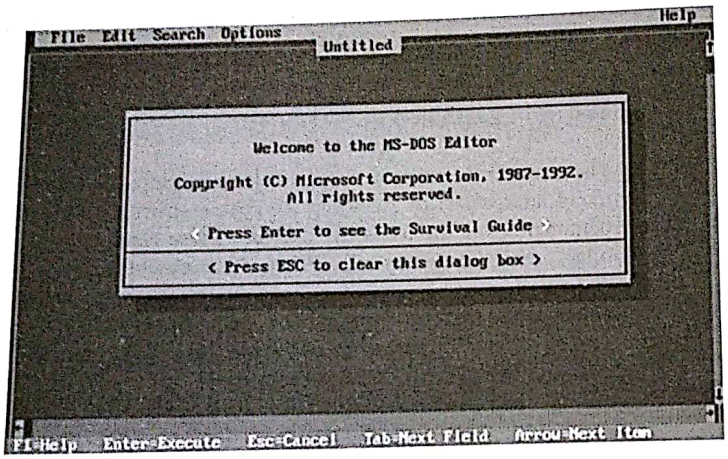


Fig. 7.34 Command line interface

The GUI is an interface that makes use of the following graphical components to allow users to easily interact with the computer (or system):

Pointer A pointer is a small, angled arrow that enables users to select commands and objects on the screen.

Pointing device It is a device (e.g., mouse or trackball) that enables users to select objects on the screen.

Icons These are small pictures that represent commands, files, or windows. Users can execute commands or open an application by moving the pointer to the icon and clicking the mouse button. Common examples of icons that we use are small rectangles (for files), file folders (for directories), a trash can (to indicate a place where we dispose unwanted files), and the direction buttons on web browsers that help users to navigate to the previous or next page.

Menus Menu is a GUI element that displays a list of available commands. Users can execute commands by selecting a choice from the menu.

Window A window is a rectangular portion of the screen that displays applications, menus, icons, files, and so on. The GUI facilitates users to simultaneously work in multiple windows, where each window displays a different application or each window can display different files that have been opened or created by the same application.

Advantages of Graphical User Interface

The following are some advantages of the GUI:

- With well-designed GUIs, users can get rid of the burden of learning complex commands to perform even small operations.
- The interface is intuitive. Thus, users take less time to learn and work with the system. For example, it is far simpler to move a file from one directory to another by dragging its icon with the mouse than remembering and typing the complete command without any mistake to perform the same operation.
- GUIs ease the task of transferring data from one application to another. However, the two applications must be running the same GUI. For example, we can easily incorporate a pie chart created in Microsoft Excel into a document created using Microsoft Word.
- GUIs give users an immediate visual feedback of the effect of their action. For example, when a file is deleted, the icon of that file immediately disappears, thereby notifying the user that the file has been removed from its current location.

Applications of Graphical User Interface

In a GUI, users can directly manipulate the graphical elements to perform the desired task. This feature makes the GUI applicable not only in computers but also in the following devices:

- Hand-held devices such as mobile phones, MP3 players, portable media players, and gaming devices
- Automated teller machines (ATMs)
- Point-of-sale kiosks
- Self-service checkouts used in a retail store
- Airline self-ticketing and check-in
- Information kiosks in public places such as a railway station or a museum
- In household appliances, office, and industry equipment

Drawbacks of Graphical User Interface

The following are some of the drawbacks of the GUI:

- A GUI requires more system resources than a CLI. This is because a number of graphical elements such as icons, menus, and fonts have to be loaded. In addition to these elements, video drivers, mouse drivers, and other drivers must also be loaded, which will use even more resources.
- Even today, more advanced tasks are performed using the CLI despite the fact that the GUI also offers enough control over the file system and operating system.

Scanned by CamScanner

- Although it is easier for users to work with the GUI, it is much slower than using a command line.
- Remote graphical access is possible with the GUI but not all computers and network devices support manipulation using the GUI.
- Each GUI system differs in design and structure from the other systems. Even different versions of the same GUI can have many variations. For example, the GUI of Windows 7 is completely different from that of Windows 8.

7.9 POPULAR OPERATING SYSTEMS

Microsoft DOS (MS-DOS), Microsoft Windows family of operating systems, Unix, Linux, and Mac OS X are some examples of the commonly used operating systems. Each operating system has specific characteristics. In this section, we will discuss the features of some of these operating systems.

7.9.1 Microsoft DOS

Microsoft DOS is a non-graphical command line operating system, which was introduced by Microsoft in August 1981. It was the first widely installed operating system in personal computers in the 1980s. MS-DOS was not only one of the most powerful operating systems of that time but was also easy to load and install. It requires neither much memory nor a very sophisticated computer to run on.

Since MS-DOS supports a CLI (refer Figure 7.36), users need to remember the commands and know where the programs and data are stored. The *command.com* module of the MS-DOS command mode interprets the command and executes it. As compared to the operating systems of today, MS-DOS has relatively a very small number of commands. The following are some of the frequently used commands:

- CD to change the current directory
- COPY to copy a file
- DEL to delete a file
- DIR to list directory contents
- EDIT to start an editor to create or edit text files
- FORMAT to format a disk
- HELP to display information about a command
- MKDIR to create a new directory
- RD to remove a directory
- REN to rename a file
- TYPE to display contents of a file on the screen

MS-Dos is basically a single-user single-tasking operating system. Therefore, at a given instant of time, only one user can use it to perform only one task. Another drawback of MS-DOS is that it does not have a built-in support for networking.

Moreover, MS-DOS is a 16-bit operating system. Therefore, it can send, receive, or process only 16 bits of data at a time. It is unable to take advantage of the 32-bit or 64-bit processors available in the market today. All these factors consequently led to the decline in the popularity of

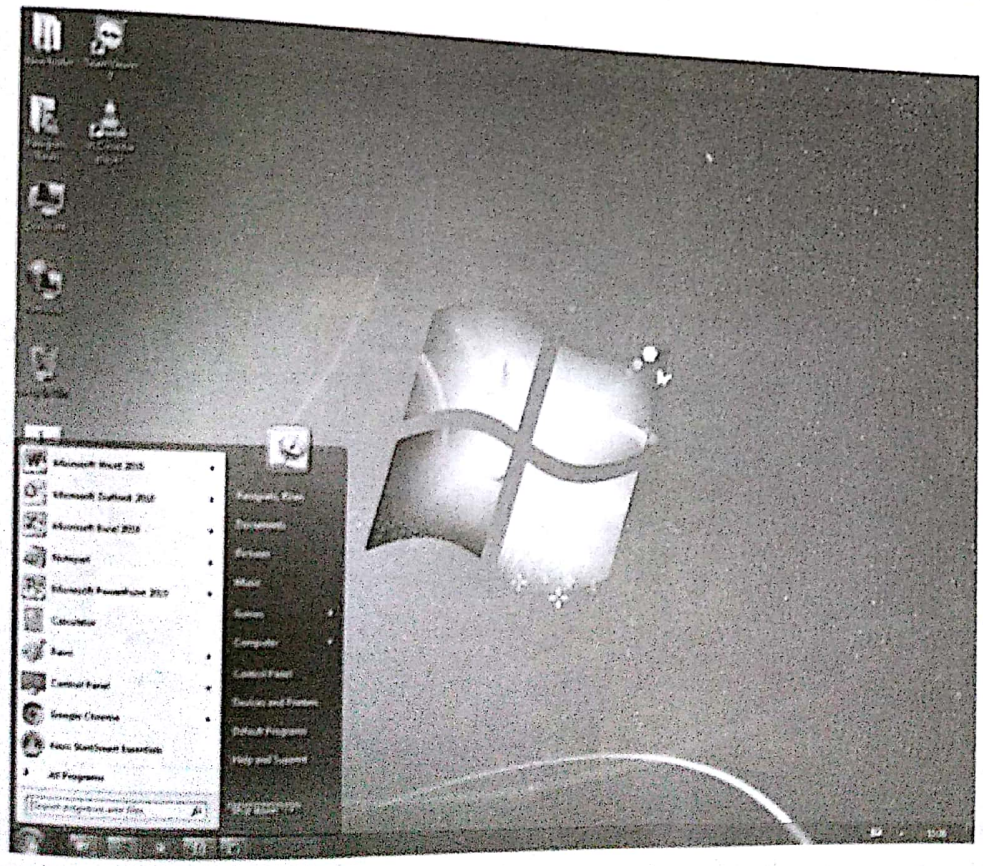


Fig. 7.35 Graphical user interface

Scanned by CamScanner

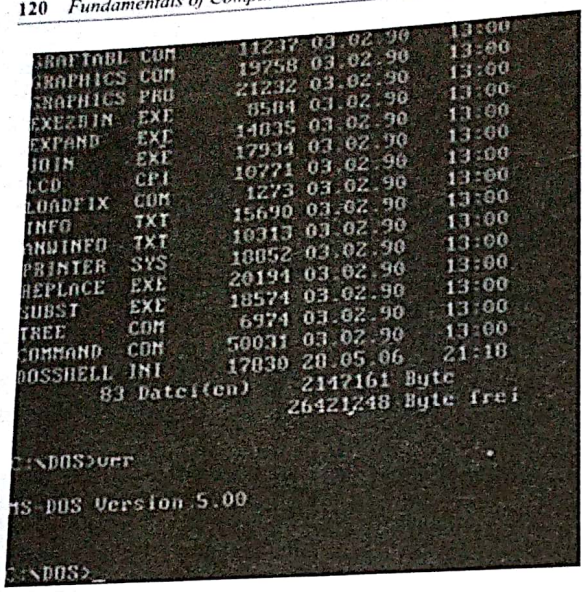


Fig. 7.36 MS-DOS

MS-DOS. Its last version, MS-DOS 6.22, was released in 1994. Although users do not use MS-DOS today, its command shell (now called Windows command line) is still popular among users.

7.9.2 Microsoft Windows

Windows operating system was developed by Microsoft. It supports a GUI, which enables even a naïve user to easily use a computer. It is a single-user multitasking operating system, so a user can run more than one application at a time. Microsoft Windows is not merely an operating system but is also a complete operating environment, because it not just supports the functions of an operating system but also has programs that conform to a standard way of working. For example, the working of Microsoft Word is similar to that of Microsoft Excel. Therefore, if a user learns one program of Microsoft Windows, he/she can easily learn the other programs developed by it.

When designing this operating system, Microsoft has taken two separate approaches, in which one is suited for home users and the other is intended for information technology professionals. The home edition supports more user functionalities and multimedia features but has limited support for security and networking. The professional edition, on the other hand, is well suited for server environment and has limited multimedia features but offers enhanced networking capability and security (Figure 7.37).

The first version of Windows (version 1.0), which was released in November 1985, was not very popular as it lacked functionality compared to the Apple operating system. After two years, version 2.0 was released, which achieved slightly more popularity than its predecessor. Then, in January 1998, version 2.03 was released. This offered an entirely different look and resulted in Apple filing a lawsuit against Microsoft with accusations of infringement.

Version 3.0 was released in 1990. This became the first edition to experience commercial success by selling two million copies within its first six months. Windows 3.0 had

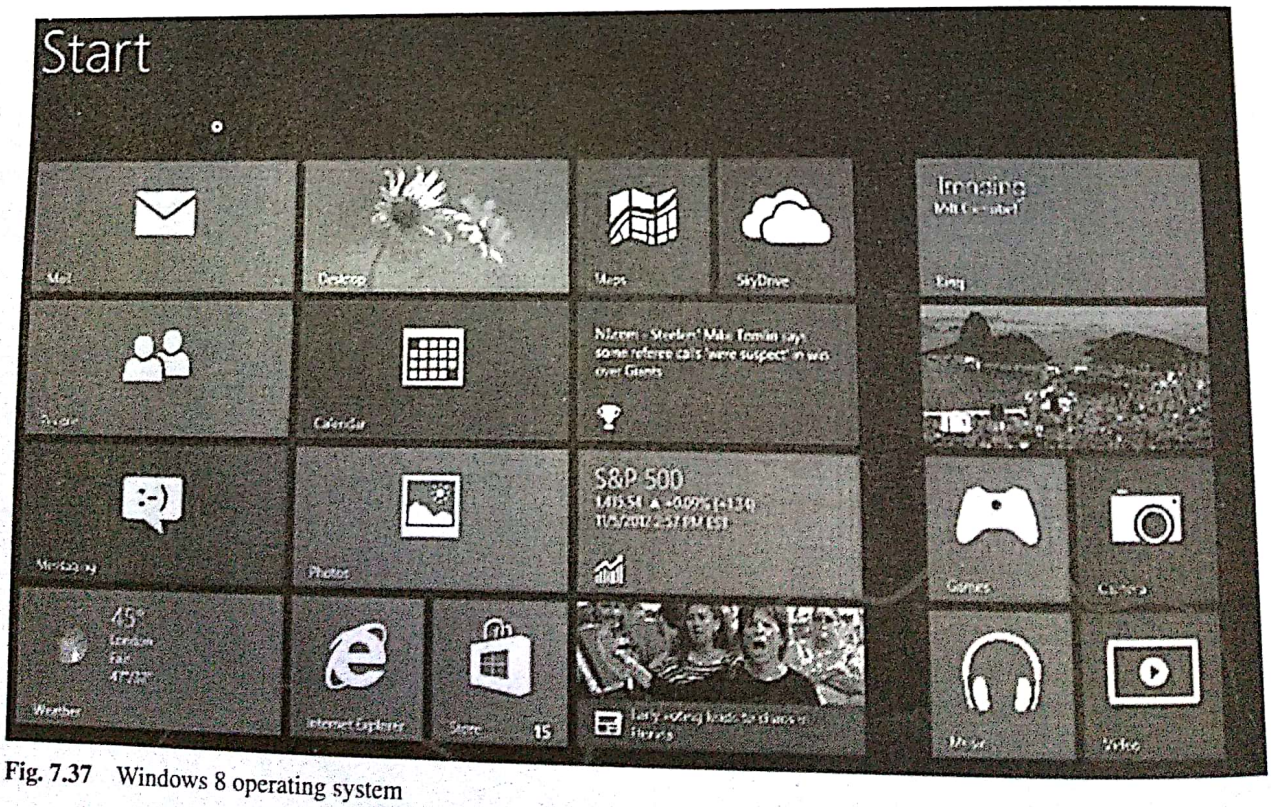


Fig. 7.37 Windows 8 operating system

Scanned by CamScanner

highly improved user interface along with new multitasking capabilities. Later, in March 1992, version 3.1 was released, which offered a facelift.

In July 1993, Windows NT was released and became the first operating system to be designed for a professional platform.

In August 1995, Windows 95 was released as a replacement for Windows 3.1. This operating system provided support for preemptive multitasking and offered a consumer solution with significant changes to the user interface. It was the first operating system by Microsoft to use the plug and play system. Windows 95 revolutionized the desktop platform and achieved mass popularity.

Later, Windows 98 was released in June 1998. However, it was criticized for being slower and less reliable than Windows 95. Hence, many of those issues were addressed a year later with the release of Windows 98 Second Edition.

In February 2000, Microsoft brought out another professional operating system known as Windows 2000, the consumer version of which was released as Windows ME in September. October 2001 witnessed the release of Windows XP, which was based on the NT kernel and managed to retain the extreme functionality of its home-based predecessors. XP became very popular among the masses and was available in two different editions: home and professional. Windows XP was then succeeded by Windows Vista, which included several new features with an emphasis on security.

Two of the widely used Windows versions are discussed below.

Windows 7

Windows 7, which is widely used today, has advanced features for touch, speech, and handwriting recognitions. It supports additional file formats and provides improved performance on multi-core processors. It requires less memory and provides quicker USB device detection. With Windows 7, it is very easy to share files and printers.

Windows 8

Windows 8 based on a touch-optimized user interface has brought in a revolution by drastically changing the operating system's platform and user interface to make it easy to be used on tablets. It has a number of special features. Some of them include the following:

- Windows Defender and Windows SmartScreen guard the computer against viruses, spyware, and other malicious software.
- Facilitates users to sign in their Microsoft account (by providing an email address and password) in to any of the devices running Windows 8.1, and experience the same settings, layout and access to applications.
- By signing in with Microsoft enables users to access a world of applications from the Windows Store.

- Signing with Microsoft also allows users to save, view, and edit their documents and photos online with SkyDrive (a free online storage). When the files are saved on SkyDrive, the data will be protected even if something happens to the user's PC. Moreover, users can easily collaborate and share their files with colleagues working on the same project.

7.9.3 Unix

Unix is an operating system that first developed in the 1960s by Ken Thompson and Dennis Ritchie at Bell Laboratories; since then it has been under constant development. It is a stable multi-user multitasking operating system for servers, desktops, and laptop computers. It was the first operating system that was written in a higher language (C); the earlier operating systems were written in the assembly language.

Like Windows, Unix supports a GUI to provide an easy-to-use environment. It is made up of three parts: kernel, shell, and files/processes (refer Figure 7.38).

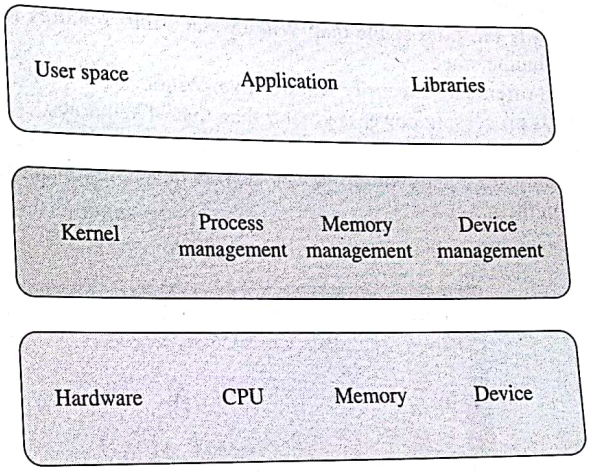


Fig. 7.38 Unix operating system

Kernel The kernel of Unix is the hub of the operating system as it allocates the CPU time and memory to the programs and handles the communications when a system call is invoked. It contains all modules necessary for process, memory, file, device, and security management.

Shell The shell acts as an interface between the user and the kernel. When a user logs in, the login program authenticates the user by verifying the username and password, and then starts a program called the shell. The shell is a command line interpreter. It interprets the commands typed by the user and executes them with the help of the kernel. The commands are themselves programs, and when they terminate, the shell displays another prompt (%) on the screen.

Let us take an example to see how the shell and kernel work together to execute the commands typed by the user. When the user types `rm myfile` (which means removing the file `myfile`), the shell searches the file containing the program `rm` and then uses system calls to request the kernel to execute the program `rm` on `myfile`.

Scanned by CamScanner

The Unix shell provides a large number of commands and each command performs a specific task. To perform complex tasks, users can combine several commands in a shell script and save it as an executable file.

Files and processes Everything in Unix is considered either a file or a process. A process is a program under execution and a file is a collection of data. Users can create their own files using a text editor. Examples of files include a document (e.g., report), the text of a program written in some high-level programming language, and a directory containing information about its contents, which may contain other directories (subdirectories) and ordinary files.

Advantages of Unix

The following are the advantages of Unix:

- It is flexible and can be easily installed on different types of computers (e.g., supercomputers, microcomputers, mainframes).
- It is far more stable than Windows and thus requires less maintenance.
- It offers more security features than Windows.
- Its processing power is greater than that of Windows.
- It is the most widely used operating system for web servers. Almost 90 per cent of the web servers have Unix installed on them.

7.9.4 Linux

Linux is a very powerful, free open-source operating system based on Unix. It was originally created by Linus Torvalds with the assistance of developers from around the globe. Users can freely download Linux from the Internet and also make changes to it. The advantages of Linux include the following:

Low cost Linux is freely available on the Internet, so users need not spend huge amount of money to obtain licences. Moreover, users can also edit its source code to develop a customized operating system.

Stable It is a stable operating system. It rarely freezes up or slows down and has continuous up-times of hundreds of days or more.

Performance It gives very high performance on various networks. It can simultaneously handle large numbers of users.

Networking capabilities It is widely used in networks as it provides a strong support for network functionality. A computer running Linux can easily be connected to another computer to form a client–server model. Moreover, Linux is known to perform tasks such as network backup faster than any other operating system.

Flexible Linux is a flexible operating system as it can be used for high-performance server applications, desktop applications, and embedded systems. The user has the

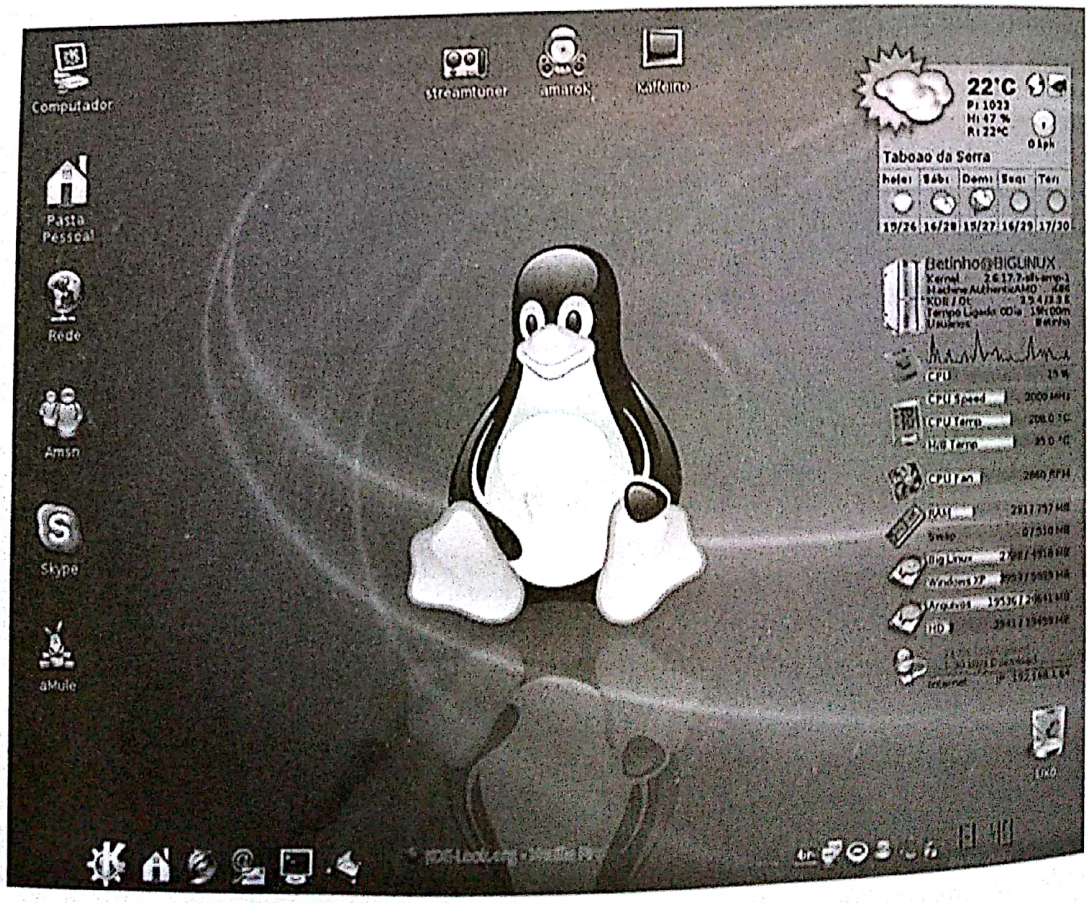


Fig. 7.39 Linux operating system

Scanned by CamScanner

option to install only the needed components for a particular use.

Compatible It is a Unix-based operating system and is therefore compatible with it. Thus, it can run all common Unix software packages and can process all common file formats.

Easy to install It comes with user-friendly installation.

Better use of hard disk Linux uses its resources well enough even when the hard disk is almost full.

Multitasking Linux is a multitasking operating system so it can simultaneously execute several jobs.

Security Linux is a secure operating system, as it supports many options for file ownership and permissions.

Overall, Linux is a popular operating system used by home and office users. It is mainly used for high-performance business and in web servers. Moreover, Linux can be installed on a wide variety of computers ranging from mobile phones, tablet computers, routers, and video game consoles to desktop computers, mainframes, and supercomputers. Today, Linux is the most popular server operating system and runs the 10 fastest supercomputers in the world.

SUMMARY

- An operating system is a group of computer programs that controls a computer's resources such as the CPU, memory, and I/O devices and provides users with an interface that makes it easier to use.
- A batch processing operating system allows very limited or no interaction between the user and the processor during the execution of work. Data and programs that need to be processed are bundled and collected as a *batch* and executed together.
- A single-user operating system allows only one program to be executed at a time. A multi-user operating system enables multiple users on different computers to access a single system (with one operating system). Time-sharing systems were developed to provide an interactive use of the computer system by several users.
- In a multiprocessing system, a complex program can be divided into smaller parts and then be executed concurrently by multiple processors. Real-time system is a data processing system in which the time interval required to process and respond to inputs is very small and highly critical. By using a virtual machine operating system, several operating system environments can coexist on the same computer.
- A process is a program in execution, and process management is an integral part of the operating system. Process components include code, data, resources, and status of the process; process states, on the other hand, include new, ready, running, waiting, and terminated.
- The operating system allocates resources to processes, enables processes to share and exchange information, protects the resources of each process from other processes, and enables synchronization among processes.
- Besides allocation and de-allocation of memory, the operating system ensures that processes do not interfere in each other's memory space and applies defragmentation techniques to utilize small chunks of vacant spaces in the main memory.
- Paging suffers from internal fragmentation, which means a certain amount of space is wasted within the memory frame. This occurs when the allocated memory or frame is slightly larger than the requested memory.
- A sequentially organized file stores records in the order in which they are entered. In relative file organization, records in a file are ordered by their *relative key*. In indexed sequential file, every record is uniquely identified by a key field. An index table is maintained to store the record number and the address of the record in the file.
- The operating system communicates with the peripheral devices via the *device driver software* that comes along with each device. It supports buffering to handle the speed mismatch by storing data in a buffer until the complete data has arrived and then sends it to the processor in a single write operation.
- Security means providing protection to a computer's resources such as the CPU, memory, disk, programs, and data. The operating system provides security at two levels: internal and external.
- The command interpretation module (also called the command interpreter) of the operating system provides a set of commands that the users can execute. These commands are often called system calls. When a user executes a system call, the command interpreter interprets the instructions and allocates the system's resources to handle the user's request.
- Microsoft DOS, Windows family of operating systems, Unix, Linux, and Mac OS X are some examples of commonly used operating systems.

Scanned by CamScanner

Authentication Identifying every single user of the system so that only authentic users can execute programs on the computer.

Booting The process of starting or restarting the computer.

Bus A set of wires that connects multiple devices.

Deadlock A situation that occurs when multiple processes are waiting for each other to free the resources they have acquired, and as a consequence, all of them remain halted.

File A block of useful data that is available to a computer program and is usually stored on a persistent storage medium.

Hacker A person who seeks and exploits the weaknesses in a system or in a computer network.

Icons Small pictures that represent commands, files, or windows.

Interrupt A signal sent by peripheral devices to the CPU to notify it to suspend the current process and execute the appropriate part of the operating system.

Logical address An address generated by the CPU.

Operating system A group of computer programs that controls a computer's resources and provides the users with an easy to use interface.

Parent process The process that requests the creation of a new process.

Physical address space The address actually available in the main memory.

Pointer A small, angled arrow that enables the users to select commands and objects on the screen.

Process A program in execution.

Process control block A data structure that describes the state and resource ownership of a process.

Response time The time taken from submission of a program to its first response (not the output).

Time quantum The short period of time, which is usually between 10–100 ms, during which a user gets the CPU.

Window A rectangular portion of the screen that displays applications, menus, icons, files, and so on.

Fill in the Blanks

1. _____ provides an environment in which users can execute programs conveniently and efficiently.
2. Operating system is added as a layer between _____ and _____.
3. _____ schedules processes running in the system.
4. A program in execution is called a _____.
5. _____ takes care of the creation and deletion of processes.
6. Users store, retrieve, and work on the information stored in _____.
7. _____ is an example of a single-user operating system.
8. Microsoft Windows is an example of a _____ operating system.
9. Time-sharing operating systems make use of _____ and _____ to provide each user with a small slot of the processor's time for execution.
10. _____ operating systems use two or more processors within a single computer system.
11. The commonly used techniques of IPC are _____ and _____.
12. _____ allocates the smallest hole to the process that is big enough to meet its memory requirements.
13. Paging suffers from _____ fragmentation.
14. _____ is an elementary unit that stores a single fact.
15. _____ is a pointer that points to the position at which the next read/write operation will be performed.
16. A _____ uniquely identifies the records in a file.
17. Index table stores _____ and _____.
18. _____ hide the complexity of handling devices.
19. _____ are generated by the CPU itself to signal the operating system that an error has occurred.
20. While implementing _____ security, the operating system protects one process from another.
21. _____ is the interface between the user and the operating system.
22. _____ is a non-graphical command line operating system.
23. _____ is a stable multi-user multitasking operating system.
24. Unix is made of three parts: _____, _____, and _____.

Multiple Choice Questions

1. The part of the computer system in charge of managing the hardware and the utilization of its resources is:
   (a) Application programs
   (b) Utility programs
   (c) Operating system
   (d) System software

Scanned by CamScanner

(b) Secondary memory
(c) Tertiary memory
(d) Auxiliary memory

3. The information files saved on the computer disk that an operation system keeps track of is:
   (a) Location (b) Access time
   (c) Modified time (d) All of these
4. The operating system that allows a single user to simultaneously perform several tasks is:
   (a) Multiprogramming (b) Multitasking
   (c) Time sharing (d) Multi-user
5. The type of machines that use the multi-user operating system are:
   (a) Supercomputers
   (b) Mainframe computers
   (c) Microcomputers
   (d) Minicomputers
6. The operating system used in the air traffic control system is:
   (a) Multitasking (b) Multi-user
   (c) Real time (d) Multiprocessing
7. The operating system that has very little user interface capability and does not support end-user utilities is:
   (a) Multitasking (b) Real time
   (c) Single user (d) Multiprocessing
8. The operating system that allows several operating system environments to coexist on the same computer is:
   (a) Time sharing
   (b) Virtual operating system
   (c) Real-time operating system
   (d) Multiprocessing
9. The scheduling policy that has long waiting times for small processes is:
   (a) SJF (b) Round robin
   (c) FCFS (d) All of these
10. The scheduling policy that leads to starvation of long jobs is:
   (a) SJF (b) Round robin
   (c) FCFS (d) All of these
11. The number of programs that can be loaded in the memory in a multiprogramming system is:
   (a) 1 (b) 2 (c) 3 (d) *n*
12. The operation that does not include while updating a file is:
   (a) Adding (b) Deleting
   (c) Modifying (d) Creating
13. The file organization in which all the records have the same size and field format and a fixed size is:
   (a) Sequential
   (b) Indexed sequential
   (c) Relative
   (d) All of these
14. A small area of memory that stores the data while it is being transferred between two devices is:
   (a) Device driver (b) Device controller
   (c) Buffer (d) Device
15. The operation in which the CPU waits for a device to be ready for an I/O operation is:
   (a) Buffering (b) Spooling
   (c) Polling (d) DMA
16. The program that misbehaves only when certain conditions hold true is:
   (a) Trap door (b) Trojan horse
   (c) Virus (d) Logic bomb
17. The 16-bit operating system is:
   (a) Windows (b) DOS
   (c) Unix (d) Linux
18. The part of Unix that contains all modules necessary for the processes, memory, files, devices, and security management is:
   (a) Kernel (b) Shell
   (c) Files (d) Processes

State True or False

1. Utility program is the first program loaded into the computer during booting.
2. Warm booting is done when a completely turned off computer is turned on.
3. Multiprogramming was introduced for interleaved execution of multiple jobs by the same CPU.
4. We use time-sharing operating system in our laptops.
5. In a multi-user operating system, each user accesses the same operating system at different machines.
6. In a tightly coupled system, each processor has its own local memory and the processors do not share a system-wide memory.
7. Hard real-time operating system gives priority to critical tasks.
8. There is a single PCB for all the processes running in a system.
9. In non-preemptive scheduling, the lower priority process is suspended to execute the higher priority process.
10. There is no need for a CPU scheduling policy in a non-preemptive system.
11. In the FCFS technique, the CPU time requirements of each process must be known in advance.

Scanned by CamScanner

EXERCISES

12. When one process is in the critical section, no other process can enter the critical section at that time.
13. In the mutual exclusion mode, processes can share the resources.
14. All unsafe states are deadlocks.
15. To execute a program consisting of *n* pages, the operating system needs $n-1$ free frames in the memory.
16. When a backup program archives the file, it sets the archive bit to 1.
17. Relative files can be used for both random access of data and sequential access.
18. The records in an indexed sequential file are of fixed length.
19. The device controller acts like a bridge between the device and the operating system.
20. Direct memory access is preferred when a large amount of data has to be transferred between a peripheral device and the memory.
21. Trap door chokes down a system's performance by using its resources to extreme levels.
22. The *command.exe* module of the MS-DOS command mode interprets the command and executes it.
23. Disk operating system is a single-user multitasking operating system.

Review Questions

1. Define operating system.
2. Explain the functions of the operating system.
3. Differentiate between the following:
   (a) Cold booting and warm booting
   (b) Multiprogramming and time-sharing operating systems
   (c) Tightly coupled and loosely coupled systems
   (d) Multiprogramming and multiprocessing
   (e) Hard and soft real-time operating systems
   (f) Normal termination and abnormal termination
   (g) Preemptive scheduling and non-preemptive scheduling
   (h) Shared memory and message passing for IPC
   (i) Text file and binary file
   (j) Internal and external security
   (k) GUI and CLI
   (l) Unix and Windows
4. Write short notes on the following types of operating systems:
   (a) Batch processing operating system
   (b) Single-tasking operating system
   (c) Single-user multitasking operating system
   (d) Multi-user multitasking operating system
   (e) Time-sharing operating system
   (f) Multiprocessing operating system
   (g) Real-time operating system
5. Explain the states of a process with the help of a diagram.
6. Explain the role of an operating system in process management
7. What type of information is stored in the PCB?
8. What are the techniques to decide the next process to be executed?
9. Explain the term *process synchronization*.
10. What are deadlocks? Explain the necessary conditions for a deadlock to occur.
11. Give the techniques to handle deadlock in a computer system.
12. Explain the role of the operating system in memory management.
13. Make a comparison between the different techniques for memory allocation.
14. Explain the concept of paging.
15. How does segmentation support a programmer's viewpoint?
16. Why is internal fragmentation unavoidable in the paging scheme?
17. Explain data hierarchy.
18. What are file attributes?
19. How is the archive flag useful for data backups?
20. Write a short note on file operations.
21. Make a comparison between sequential, relative, and indexed sequential file organizations.
22. What is the role of the operating system in device management?
23. Discuss the role of the operating system in securing a computer's resources.
24. How does the operating system provide external security?
25. Explain the features of the following operating systems:
    (a) MS-DOS (b) Windows
    (c) Unix (d) Linux

Scanned by CamScanner