CHAPTER 1

Overview of Operating Systems

INTRODUCTION

Computer is an electronic device that is used to process data. Everyone in this world is aware of computer and its uses in various sectors. A computer mainly consists of software and hardware. Various kinds of software and hardware are available.

This chapter focuses your attention on the most important software that is required in every computer *i.e.* system software. It explains the needs, types and functions of system software. The chapter starts with the definition and types of software. It explains the types of system software and the functions. The main motive is to make you understand the functioning and benefits of operating system.

1.1. SOFTWARE

We know that the computer performs tasks as instructed by the user. User supplies these orders or instructions by using different softwares. Software is a set of programs, which performs a well-defined function. A program is a set of instructions of a computer language, written in a sequence to solve a particular problem. Several related programs may be grouped to create software. These software are created by programmers and distributed by using disks or the internet.

Definition Software is a set of programs, which performs a well-defined function. A program is a set of instructions of a computer language, written in a sequence to solve a particular problem.

For example, a software to automate the admission and fee system in a college, will have a program for admission in which the record of each student is inserted. The other program for fees will manage various kinds of fees as per the facilities opted by the student at the time of admission. There may be a program to keep track of the attendance record of a student and a program to manage fines collected from different student under various categories.

Hardware and software, both are interdependent. No one can perform its functionality without the presence of the other. Hardware can be considered as organs and the software is life that moves inside these organs. The software and hardware industries are developing as per the needs of each other. Whenever a new type of device is manufactured, the corresponding software to work with that device are also created. For example, to use a CD Writer, you need

to have a software for writing CDs. In India, software and hardware industries are growing rapidly. India is a leader in exporting customized software to Europe and America. You will be surprised to know that around 80% of total exports of India is done by software industry only. The programmers of south Asian countries are highly regarded in Europe and America.

1.2. TYPES OF SOFTWARE

Software is generally divided into two main types :

1. System software
2. Application software.

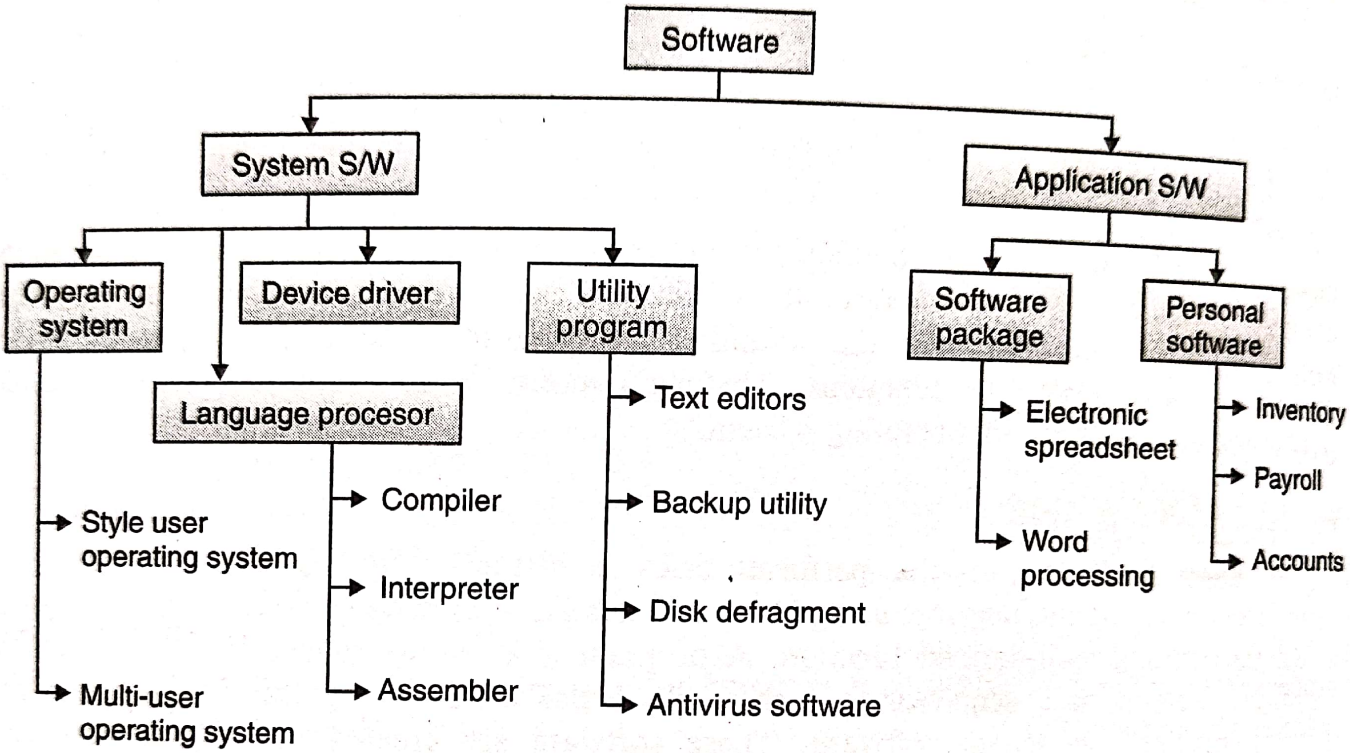


Fig. 1.1.

1.2.1. System Software

System software is a software that manages overall working of the computer system. System software checks for the availability of different devices, scans input devices for input and sends output to output devices. System software keeps track of the utilization of memory by different running applications and convert the instruction from user understandable language to computer understandable language. It controls the working of various peripheral devices attached with the computer.

Definition *System software are the software used for controlling the internal operations of the computer.*

System software are used either for the development of application software or for running the application software on the computer. The main job of the system software is to act as an interface between user and computer's hardware. It basically consists of low-level programs that interact with the system at a very basic level. It controls all the processing activities and makes sure that the resources and the power of the computer are used in most efficient manner.

It keeps an eye on each and every part of the computer. Following software are generally regarded as system software :

1. Operating systems
2. Language processors
3. Device drivers
4. Utility programs.

1.2.1.1. Operating system

Operating system is a set of master programs, which is designed to control entire operation of the computer. This is the software that is required in order to run other application software. The operating system can be regarded as a track on which the train runs. As the track acts as a strong link between earth and the train, operating system also acts as a strong link between the hardware and the user of the computer. To run the train, track has to be installed first, similarly the operating system is required in the system to make it operational. It is loaded automatically into the computer whenever we start the computer. Operating system is a general-purpose software which is involved in each and every activity of the computer. The operating system decides when and how to process a given data, where to store the processed data and from where to supply the data when user requests. Following are the main functions provided by the operating system.

Definition *Operating system is a set of master programs, which is designed to control entire operation of the computer.*

The operating system can be regarded as a manager in the sales industry that keeps trace on every salesperson and evaluates his performance from time to time. If some worker is not working properly, the manager takes appropriate action to make him do his job properly. It is the manager which fulfils the needs of different employees working under him. For example, the requirements of banners and advertisement material to promote the sales. He acts as a link between the staff working under him and the top management. The Fig. 1.2 illustrates the architecture of computer system.

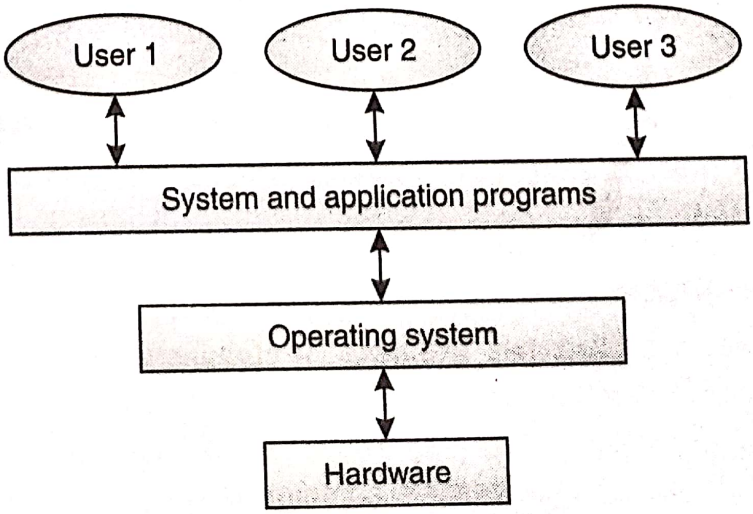


Fig. 1.2. Computer architecture.

Important *A computer cannot function without the presence of operating system.*

FUNCTIONS OF OPERATING SYSTEM

Following are the main functions of operating system :

1. Interface between user and hardware : The computer hardware does not understand the language used by humans to communicate with each other. It works on the binary language of 0s and 1s. The users work in human understandable language like English but still we get the correct result from the computer. It is the operating system which converts our input to the language understandable by the hardware, then it converts the output to user understandable language.

For this, each operating system has its own set of commands and the user supplies these commands on the terminal. These commands are interpreted by the operating system and the desired result is presented to the user. Thus, operating system acts as command interpreter.

2. Peripheral management : The operating system manages the working of different input/output devices attached to the computer. Different input – output devices like printer and scanners work under control of the operating system. For example, if there are three print commands fired from different applications, the operating system will ensure that all the printing jobs are performed efficiently and independent of each other.

3. Memory management : Memory management is one of the most critical jobs of the operating system. The operating system decides which file will be stored at which location in the memory. Both primary and secondary memories work under the control of operating system. The operating system takes data needed for processing from secondary memory and stores it into the main memory. After processing, it takes output from the main memory and stores it into the secondary memory when required by the user.

4. Process management : In operating systems, allowing multiple applications to run simultaneously, the operating system keeps track of number of processes waiting to get processed and manages the processing of each and every process by providing them whatever they want. The modern operating systems allocate a part of main memory to each running application.

Important *The operating system performs the jobs of acting as an interface between user and hardware, peripheral management, memory management and process management.*

► 1.2.1.2. Language processors

We need a language to express our feelings. To communicate with someone, you need to know a language that can be understood by both the persons who want to communicate. If you want to talk to a friend in Punjabi, who only knows Tamil then you must know Tamil to talk to him. If both the people do not understand a common language, then a person is needed who knows both the languages. In our example, a person knowing Punjabi and Tamil is required. He

will listen to you, translate your sentence from Punjabi to Tamil and will tell it to other person. Then, he will listen the answer from other person in Tamil, convert it into Punjabi and will tell it to you.

Similarly, we can communicate to computers using different languages. These languages are classified into four generations from first generation language to fourth generation languages. The similar problem lies here. The computer hardware only understands two characters 0 and 1. That is why computer is called a binary device. The programmers write instructions using syntax and grammar of programming languages like C and C++. Some mechanism is needed to translate these instructions into binary language that can be directly processed by the computer hardware.

Definition *Language processors are the class of system software that are used to convert programs written in programming languages into their equivalent binary form.*

Language processors are the class of system software that are used to convert programs written in programming languages into their equivalent binary form. The languages like C, C++, Cobol and Pascal can be understood by human beings but the computer hardware only understands two characters 0 and 1. That is why computer is called a binary device. When we write instructions in high level languages, language processors take a program written in high level or assembly language as input and give its equivalent binary form as output to the computer. There are three main types of language processors :

1. Compiler
2. Interpreter
3. Assembler.

1. Compilers : The programs to create software are written by using high level languages. These programs are written by programmers by using some special softwares called Editors. These programs are created by following the syntax and grammar of a specific computer language like C or C++. A compiler is a system software which translates programs written in high level language into the machine level language that is directly understood by hardware. It is very hard to supply instructions to the computer in the language of hardware so we use high level languages which have relatively easy syntax for writing these instructions. Then, the compilers are used to translate these high level programs into machine level language.

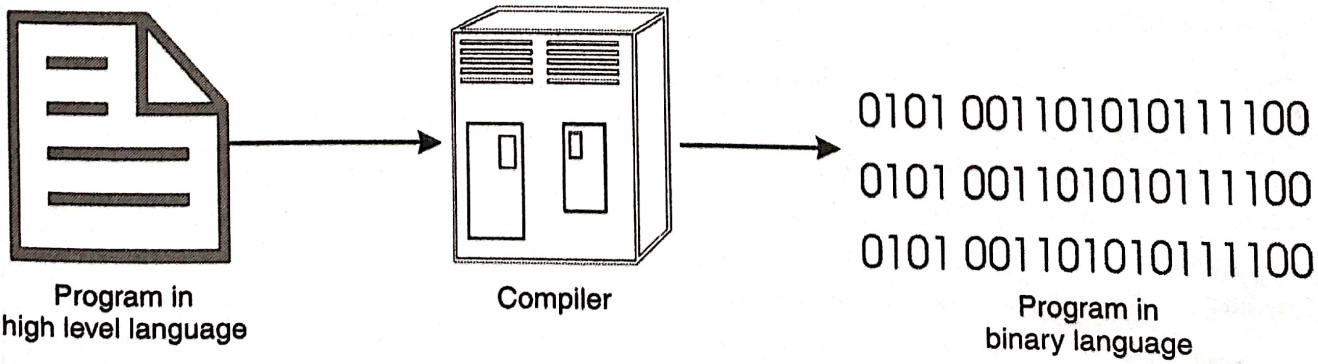


Fig. 1.3. Working of compiler.

Definition *A compiler is a system software which translates programs written in high level language into the machine level language that is directly understood by hardware.*

The program written in high level language is called as source program and the program produced by the compiler after translation is called as object program. The main advantage of compiler is that it translates the whole program at once. It takes more time to translate the program but the translated file executes rapidly leading to increased performance.

2. Interpreters : Interpreters are also used to convert a program written in high level language to machine language but they are different from compilers in performing their job of translating high level language instructions into machine level language. Interpreters translate the program line by line while the program is running whereas the compilers translate the program in one go after reading all its instructions.

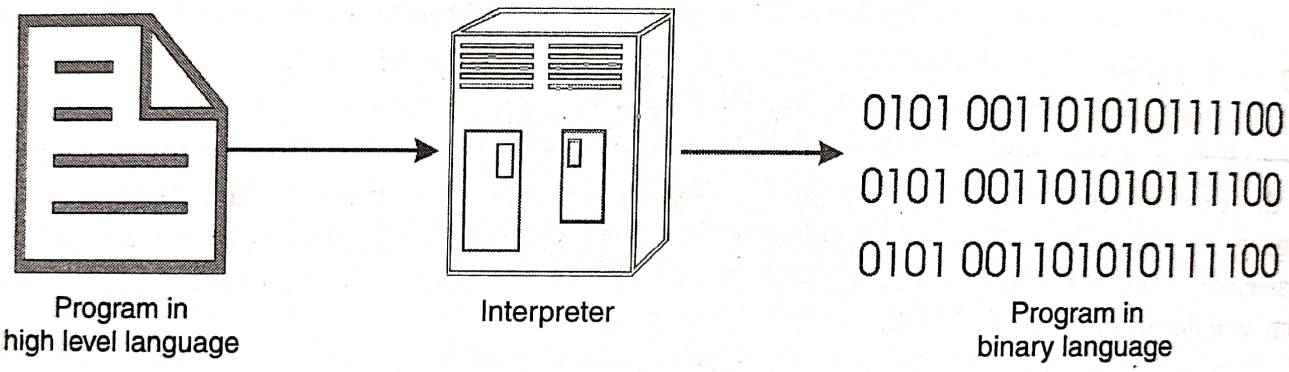


Fig. 1.4. Working of interpreters.

Interpreters have relative advantages and disadvantages over compilers. An interpreter translates line by line thus it has a greater speed of translation than that of a compiler. Another advantage is the size of the interpreter itself is smaller than the compiler and they don't generate object modules so further leading to storage capacity saving.

Important *Interpreter and compiler, both are used to convert programs written in high level language into machine level language. The difference lies in the mode of their working, the compiler translates the whole program at once so taking long time to translate whereas the interpreter translates the program line by line, so taking shorter time to translate.*

The major **disadvantage of interpreters** is that they are less efficient in translation than compilers and the programs translated by interpreters run relatively slow than programs translated by compilers. This is due to the fact that interpreters translate the instruction just before its execution, so if the number of instructions is large then a considerable amount of time is consumed and the processor is kept on waiting during this time which degrades the performance. This process is repeated whenever the program is executed again. If we see on the other side, the compiler once translates the whole program before execution and then, there is no need of re-translation for the remaining executions.

Difference between compiler and interpreter.

S.No.	Compiler	Interpreter
1.	It converts the source program instructions into object code all at once.	It converts source program instructions into machine code step by step at the time of execution.
2.	It resides on secondary memory.	It resides in RAM.
3.	It is a lengthy and complex program.	It is small and simple program.
4.	A compiled program is faster in execution.	An interpreted program is slower in execution.
5.	It responds slowly to changes in source program.	It responds faster to changes in source program.
6.	Loop statements are translated once.	Loop statements translate as many times as the loop executes.
7.	Compiler is atleast 20 times faster than interpreter.	Interpreter is much slower than compiler.

3. Assemblers : Assembler is also a system software that translates the programming language instructions into machine language. The assembler can only translate programs written in assembly language. Its working is similar to that of the compiler, it creates an object module from which further processing is done.

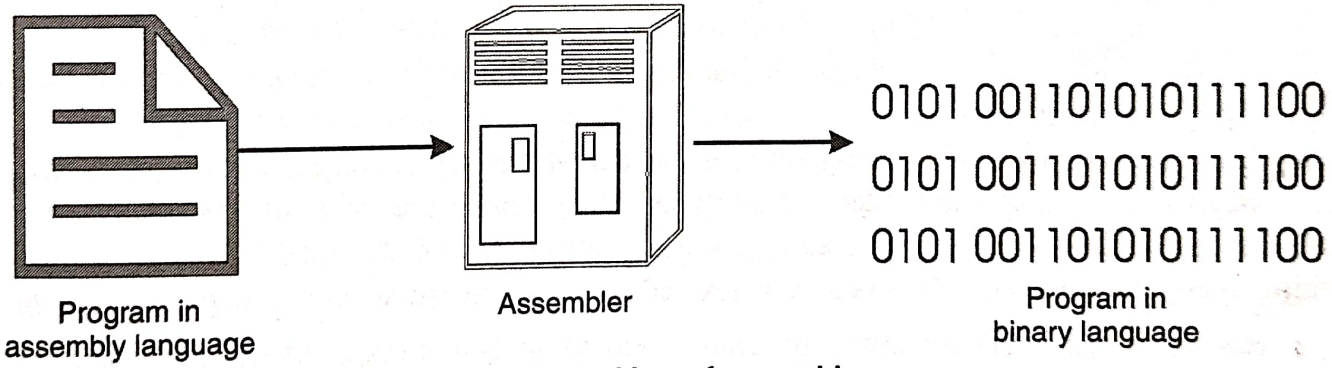


Fig. 1.5. Working of assemblers.

Different assemblers are used to program different central processing units. This is due to the fact that each new CPU has its own set of instructions that may be different from the other CPUs.

Important *Different CPUs may have different instruction sets, so need different assemblers.*

► 1.2.1.3. Device drivers

As you know that there are so many types of input and output devices. Each of these input and output devices operate by using its own set of instructions and may understand a language that might not be understood by the operating system. So, the manufacturers of these devices

create special software that operate as a link between the device and the operating system of the computer. These softwares are called as Device drivers.

Definition *Device driver is a program that acts as interface between the device and the operating system.*

Modern operating systems have inbuilt support of variety of input – output devices. These operating systems already contain the device drivers of most popular models of several companies. If you attach a device whose driver is inbuilt in the operating system, the device will start working as soon as you attach it with your computer. This is called **Plug and Play** compatibility. For example, Windows 2000, 2003 and Windows XP contains drivers of hundreds of printers, scanners and other devices.

► 1.2.1.4. Utility programs

Utility programs are the programs provided to ensure proper functioning of the computer. There are several types of utility programs those are commonly used. Some utility programs erase outdated or temporary files, some allow you to backup files, some save your computer from virus attacks and some recover the information erased accidentally. In simple words, we can say that utility programs perform housekeeping job in computer.

Following are some popular utility programs :

1. Text editors
2. Backup utility
3. Compression utility
4. Disk defragmenter
5. Antivirus software.

1. Text editors : This utility program is present in almost all the popular operating systems. It is used to create or edit text files. These types of programs are generally used for creating small text files. They allow you to create new files, save files to disk, open files for editing and printing files. For example, Notepad is a text editor available in windows operating systems.

2. Backup utility : These utility programs help us to take backup of our important data. Backups are generally taken to retrieve data in case of any damage to the original data. The files are backed up to floppies, CDs and tape drives using these programs and when there is a need, the backup is restored in the system.

3. Compression utility : Compression utilities are used to compress large sized files so that they can be stored in a storage of low capacity. Compression utilities reduce the size of the original file by compressing it. This compressed file can be easily transported using floppy or networks. At the destination, the same utility is used to uncompress the file. On uncompressing, the original file is retrieved. Winzip is a popular compression utility used in windows based desktops.

4. Disk defragmenter : Disk defragmenter utilities are used to reduce fragments created on the disks. Fragments are the small portions of unused disk space left between the used areas

of disk. On high capacity disks, several fragments are created. Due to this, parts of a single file are stored on separate locations on disk. This results into longer file access time. The disk defragmenter utilities combine the fragments to create a large unused space and rearrange fragments of files to speed up the access. Windows contains an inbuilt disk defragmenter utility.

5. Antivirus software : Antivirus software protect the computer against malicious programs called "Virus". A virus is a malicious program that *disrupts* the normal functioning of a computer. Antivirus software guard the computers against virus infections and removes if any found in the files stored on disks. Advance antivirus software remain active all the time and scan each and every file being used by the computer. They also scan every file being downloaded from internet including e-mails.

1.2.2. Application Software

Application software are the software that are designed to satisfy a particular need of a particular environment. An application software designed to satisfy one purpose may not be able to solve other purpose of the same user. For example, software for financial management can not be used for designing two-dimensional graphics. Similarly, we have word processors for letter writing and presentation software for presentations. We cannot interchange the use of two application softwares with each other.

Definition *Application software are the software that are designed to satisfy a particular need of a particular environment.*

Application software are created by programmers by using high level languages. They are created by analysing the environment and the need of the area of use. For example, to design software for maintaining stock of books for a publisher, the engineer will deeply study the way how books are brought into and taken out from godown. The application software will be different in terms of degree of sophistication, extent of coverage and complexity for two different environments.

Application software is further divided into two classes :

1. A software package
2. A personal software.

1. A software package is a software designed to fulfil the requirements of a specific user by providing additional necessary information for the purpose of controlling the operation of the package.
2. A personal software is a software designed and developed to fulfil the requirements of a single user.

1.3. DEFINITION OF OPERATING SYSTEM

An Operating System (OS) is a software program that acts as an intermediary between computer hardware and application software. It is a fundamental component of a computer system, providing the necessary services and resources to manage various hardware and software components and enabling the execution of programs.

An operating system's primary objective is to run user programs and streamline tasks. To carry out this work, a variety of application programs and hardware systems are used. An operating system is a piece of software that efficiently uses every component of a computer by managing and controlling all of its resources. The Fig. 1.6 shows how OS acts as a medium between hardware units and application programs.

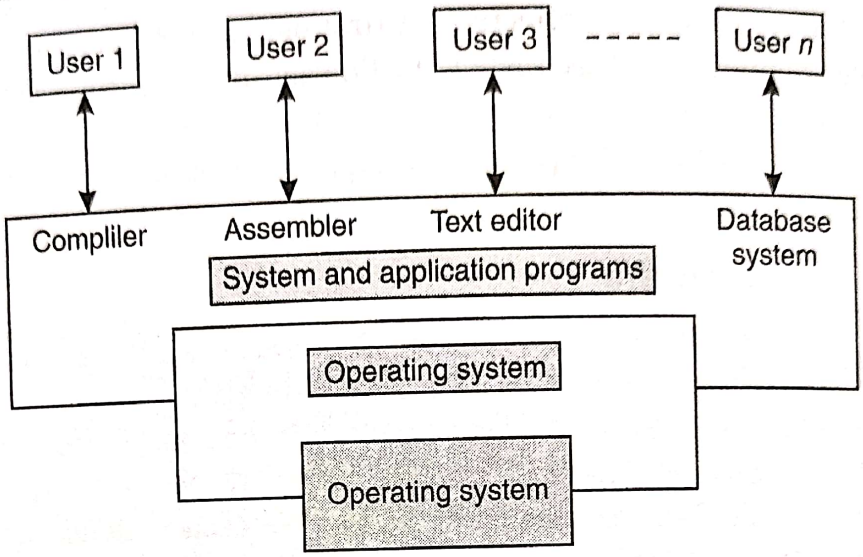


Fig. 1.6.

1.4. FUNCTIONS OF OPERATING SYSTEM

The primary functions of an operating system include ;

1. Process management : It manages and co-ordinates the execution of multiple processes or tasks running concurrently on the computer system, allocating system resources and scheduling their execution.

2. Memory management : It controls the allocation and deallocation of memory resources to programs, ensuring efficient utilization and preventing conflicts between processes.

3. File system management : It provides a hierarchical structure for organizing and storing files on storage devices, managing file access, and enforcing security and permissions.

4. Device management : It handles the communication and interaction between the computer system and various Input/Output (I/O) devices, such as keyboards, mice, printers, disks, and network interfaces.

5. User interface : It provides a means for users to interact with the computer system, offering graphical or command-line interfaces that allow users to execute commands, launch applications, and access system resources.

6. Security and protection : It enforces security measures to protect the system from unauthorized access, viruses, and other malicious activities. It also ensures data integrity and provides mechanisms for user authentication and authorization.

1.5. NEED FOR OPERATING SYSTEM

1. OS as a platform for application programs : The operating system provides a platform, on top of which, other programs, called application programs can run. These

Q.5. What is the need of language processors ?

Ans. The language processors are needed to translate programs written in high level computer languages and assembly language in their equivalent binary form. They are needed as the computer hardware cannot directly understand the languages of humans.

Q.6. What are the different types of language processors ?

Ans. Compilers, interpreters and assemblers are the different types of language processors.

Q.7. What is the use of assemblers ?

Ans. Assemblers are used to convert a program written in assembly language into equivalent binary language so that it can be processed by computer hardware.

Q.8. What are device drivers ?

Ans. Device driver is a program that acts as interface between the device and the operating system.

Q.9. What do you mean by application software ?

Ans. Application software are the software that are designed to satisfy a particular need of a particular environment.

Q.10. What is the use of compression utility ?

Ans. Compression utilities are used to compress large sized files so that they can be stored in a storage of low capacity. Compression utilities reduce the size of the original file by compressing it. This compressed file can be easily transported using floppy or networks. At the destination, the same utility is used to uncompress the file. On uncompressing, the original file is retrieved. Winzip is a popular compression utility used in Windows based desktops.

Q.11. Explain the various functions of operating system.

Ans. Following are the main functions of operating system :

(*i*) Interface between user and hardware : The computer hardware does not understand the language used by humans to communicate with each other. It works on the binary language of 0s and 1s. Operating system converts our input to the language understandable by the hardware then converts the output to user understandable language.

(*ii*) Peripheral management : The operating system manages the working of different input/ output devices attached to the computer. Different input – output devices like printer and scanners work under the control of the operating system.

(*iii*) Memory management : The operating system decides which file will be stored at which location in the memory. Both primary and secondary memories work under control of operating system. The operating system takes data needed for processing from secondary memory and stores into the main memory. Then, it takes output from the main memory and stores it into the secondary memory when required by the user.

(*iv*) Process management : In operating systems, allowing multiple applications to run simultaneously, the operating system keeps track of number of processes waiting to get processed and manages the processing of each and every process by providing them whatever they want. The modern operating systems allocate a part of main memory to each running application.

Q.12. Differentiate between compiler and assembler.

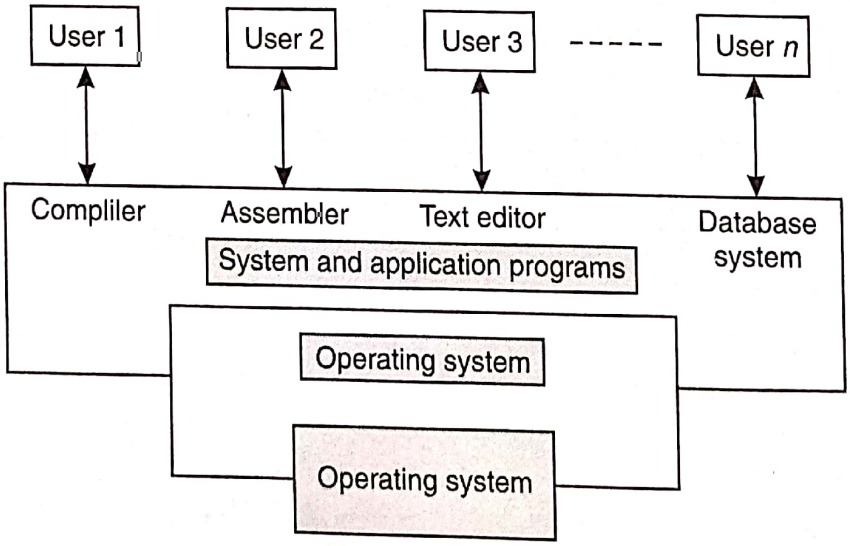
Ans. An assembler converts assembly instructions into executable machine language. A compiler converts higher level programming language instructions into assembly instructions and then those are turned into executable machine language. Some older compilers allow for the assembly instructions to be fine tuned by the programmer. Compiled programming languages typically generate many lines of assembly instructions for each program statement. Some programming languages, such as ANSIC, are very close to assembly, while others such as Java, result in many assembly instructions per program statement.

Compiling is the process of taking higher level languages files and combining then with libraries and necessary dependency files. Most compilers will then use a "linker" to link these files together and then use an "assembler" to turn that code into assembly languages and then, if necessary into machine code.

Q.13. What is operating system ?

Ans. An Operating System (OS) is a software program that acts as an intermediary between computer hardware and application software. It is a fundamental component of a computer system, providing the necessary services and resources to manage various hardware and software components and enabling the execution of programs.

An operating system's primary objective is to run user programs and streamline tasks. To carry out this work, a variety of application programs and hardware systems are used. An operating system is a piece of software that efficiently uses every component of a computer by managing and controlling all of its resources. The figure shows how OS acts as a medium between hardware units and application programs.



Q.14. Write some advantages and disadvantages of batch operating system.

Ans. Advantages :

(*i*) It is very difficult to guess or know the time required for any job to complete. Processors of the batch systems know how long the job would be when it is in the queue.

(*ii*) The batch systems are accessible to several users.

(*iii*) The batch system has a very low amount of idle time.

(*iv*) It is easy to manage large work repeatedly in batch systems.

Disadvantages :

(*i*) The computer operators should be well known with batch systems.

(*ii*) Batch systems are hard to debug.

(*iii*) It is sometimes costly.

(*iv*) The other jobs will have to wait for an unknown time if any job fails.

Q.15. Give some examples of time-sharing operating system with explanation.

Ans. Examples of time-sharing operating system with explanation :

(*i*) IBM VM/CMS : IBM VM/CMS is a time-sharing operating system that was first introduced in 1972. It is still in use today, providing a virtual machine environment that allows multiple users to run their own instances of operating systems and applications.

(*ii*) TSO (Time Sharing Option) : TSO is a time-sharing operating system that was first introduced in the 1960s by IBM for the IBM System/360 mainframe computer. It allowed multiple users to access the same computer simultaneously, running their own applications.

(*iii*) Windows Terminal Services : Windows Terminal Services is a time-sharing operating system that allows multiple users to access a Windows server remotely. Users can run their own applications and access shared resources, such as printers and network storage, in real-time.

Objective Questions

Fill in the Blanks

1. Software is a ..... .
2. Software can be classified as ..... and ..... .
3. A program is a set of ..... .
4. ..... is not a language processor.
5. ..... is a first generation language.
6. ..... translates the high level language program into machine level language.
7. High level languages are ..... .
8. ..... converts a program written in assembly language into its equivalent binary form.
9. ..... language is suitable for artificial intelligent program.
10. Disk defragmenter is the type of ..... .

True or False

1. The operating system manages the working of different input/output devices attached to the computer.
2. The programs to create software are written by using binary language.
3. An compiler translates line by line thus it has a greater speed of translation than that of a compiler.
4. Compiler is atleast 20 times faster than interpreter.

5. Utility programs are the programs provided to ensure proper functioning of the computer.
6. Notepad is a text editor available in windows operating systems.
7. Compression utilities are used to reduce fragments created on the disks.
8. A virus is a malicious program that disrupts the normal functioning of a computer.
9. An operating system's primary objective is to run user programs and streamline tasks.
10. IBM VM/CMS is a time-sharing operating system.

ANSWERS

Fill in the Blanks :

1. set of programs
2. system software, application software
3. instructions
4. Operating system
5. Machine language
6. Compiler
7. machine independent
8. Assembler
9. IISP
10. utility program

True or False :

1. True
2. False
3. False
4. True
5. True
6. True
7. False
8. True
9. True
10. True

Review Questions

1. Is it necessary to install operating systems on computer ?
2. Explain how operating system acts as an interface between user and computer ?
3. Why do we need antivirus software ?
4. What is the difference between interpreter and compiler ?
5. What are the various components of system software ?
6. State some important functions of operating system.
7. What is the need for operating system ?
8. State and explain the various types of operating systems.
9. What is real-time operating system ? Explain.

CHAPTER 2

Process Management

2.1. PROCESS CONCEPTS

In operating systems, the "process" is a fundamental concept that refers to an independent, executable unit of work. It represents a running program along with its current execution state. Processes allow an operating system to manage and execute multiple tasks concurrently, providing the illusion that multiple programs are running simultaneously on a single processor.

2.2. COMPONENTS OF A PROCESS

A process is divided into the following four sections :

1. Stack : Temporary data like method or function parameters return address, and local variables are stored in the process stack.

2. Heap : This is the memory that is dynamically allocated to a process during its execution.

3. Text : This comprises the contents present in the processor's registers as well as the current activity reflected by the value of the program counter.

4. Data : The global as well as static variables are included in this section.

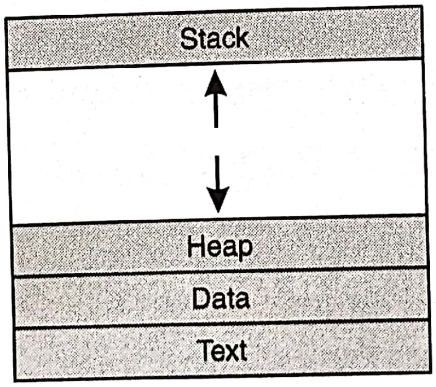


Fig. 2.1.

2.3. PROCESS STATE DIAGRAM

In an operating system, a process can exist in several different states during its life cycle. The process states represent the various stages a process goes through from its creation to termination or suspension.

The typical process states include :

1. **New** : This is the initial state when a process is first created. The operating system initializes the process, allocates necessary resources, and prepares it for execution.

2. **Ready** : Once the process has been initialized and is ready to run, it enters the ready state. In this state, the process is waiting to be assigned to a processor by the scheduler.

3. **Running :** When the scheduler selects the process for execution and the processor starts executing its instructions, the process enters the running state.

4. **Blocked (Waiting) :** While a process is running, it may encounter certain events, such as waiting for input/output operations to complete or for certain resources to become available. In such cases, the process is blocked or waiting, and it enters the blocked state. The operating system will not schedule a blocked process for execution until the blocking condition is resolved.

5. **Terminated :** When a process completes its execution or is explicitly terminated by the user or the operating system, it enters the terminated state. At this point, the process is removed from the list of active processes, and its resources are deallocated.

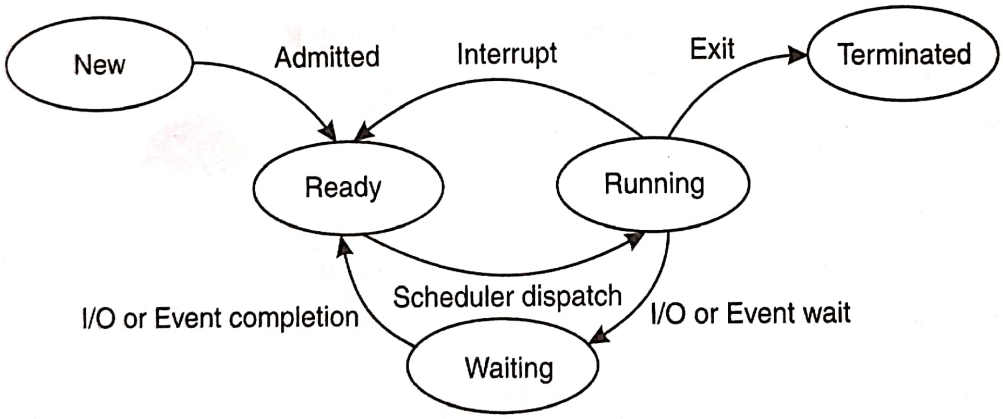


Fig. 2.2.

2.4. PROCESS CONTROL BLOCK

In operating systems, PCB stands for Process Control Block. It is a data structure used by the operating system to manage information about a specific process. Each process in the system has its own unique PCB, and the OS uses it to keep track of the process's current state, resource usage, and other essential details. The PCB serves as the central repository of information for a process and is crucial for the OS to manage and control the execution of processes efficiently.

The PCB typically contains the following information :

1. **Process ID (PID) :** A unique identifier assigned to each process in the system.

2. **Process state :** Indicates the current state of the process (*e.g.*, ready, running, blocked, terminated, etc.).

3. **Program Counter (PC) :** Keeps track of the address of the next instruction to be executed by the process.

4. **Registers :** The values of the CPU registers, which represent the process's current execution context.

5. **CPU scheduling information :** Contains data related to the process's priority, scheduling algorithm, and time spent executing on the CPU.

6. **Memory management information :** Details about the memory allocation of the process, including the base address, limit, and page tables.

7. **Input/Output information :** Stores the status of I/O operations associated with the process.

8. **Parent process ID :** Identifies the PID of the parent process that created this process (useful in process hierarchies).

9. **Child process pointers :** Keeps track of any child processes created by this process.

10. **File descriptors :** A list of open files and their associated file descriptors for the process.

11. **Accounting information :** Records resource usage statistics, such as CPU time consumed, execution time, etc.

12. **Signals and signal handlers :** Information about the signals that the process can handle and their associated handlers.

The PCB allows the operating system to switch between processes efficiently during multi-tasking or context switching. When the OS decides to pause the execution of a running process and start another, it saves the state of the current process in its PCB and then loads the saved state of the next process to be executed. This way, the OS can seamlessly switch between multiple processes and provide the illusion of concurrency on a single processor system.

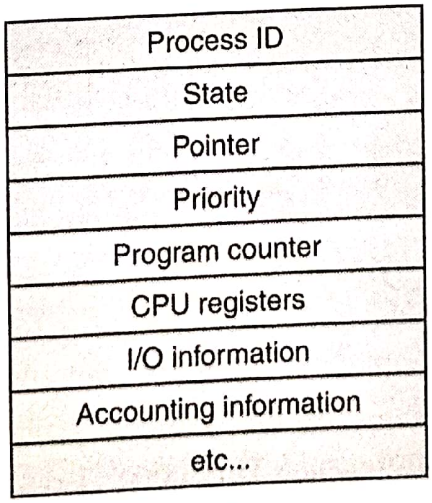


Fig. 2.3.

2.5. PROCESS SCHEDULING

The process scheduling is a process of selecting the next ready process for processing in a multitasking operating system. In multitasking or multiprogramming operating systems, several ready processes are kept in ready queue. These processes have acquired all the resources for their processing except the processor. When the processor processes a particular process for a time slice, it invokes the scheduler to select the next ready process from ready queue. Various scheduling algorithms are used by various operating systems to process multiple applications simultaneously by a single central processing unit.

2.5.1. Scheduling Queues

The job queue is where the system's incoming processes are stored. Assume that processes that are ready are typically added to the ready queue. The processes waiting for a device are placed in device queues. There are unique device queues which are available for every I/O device. A new process must first be added to the ready queue, where it waits until it is chosen to be executed. Any one of the following things can happen once the process has been assigned to the CPU and is running :

Once the process is assigned to the CPU and is executing, any one of the following events occur :

1. The process issues an I/O request, and then placed in the I/O queue.
2. The process may create a new sub-process and wait for termination.
3. The process may be removed forcibly from the CPU, which is an interrupt, and it is put back in the ready queue.

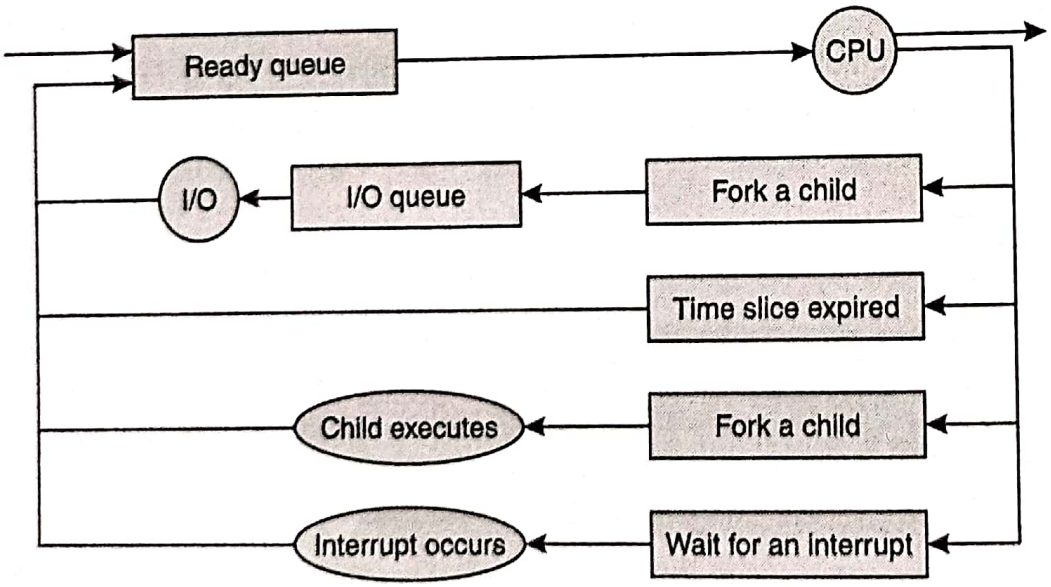


Fig. 2.4.

The method moves the first two instances from the waiting state to the ready state before returning them to the ready queue. This cycle is repeated until a process terminates, at which point its PCB and resources are reallocated and it is deleted from all queues.

2.6. OPERATIONS ON PROCESSES

In an operating system, processes are fundamental units of execution that represent running programs and tasks. The operating system provides several operations to manage processes effectively. Some of the common operations on processes include ;

1. Creation : The creation operation allows the operating system to initiate a new process. This operation involves allocating the necessary resources for the process, including memory space, setting up the Process Control Block (PCB), and initializing various process attributes.

2. Termination : The termination operation is used to end the execution of a process. When a process completes its tasks or is no longer needed, the operating system terminates it. This involves releasing all the resources associated with the process, deallocating memory, and removing its PCB.

3. Scheduling : The scheduling operation involves determining which process should be executed next by the CPU. The process scheduler uses scheduling policies and algorithms to decide the order of process execution and allocate CPU time to each process fairly.

4. Context switching : Context switching is the operation of saving the current execution context of a process and loading the context of another process. It is performed during a context switch when the operating system switches the CPU from one process to another to allow multi-tasking.

5. Waiting (Blocking) : When a process is waiting for a particular event, such as Input/Output (I/O) completion or a resource becoming available, it enters a waiting state. During this time, the process does not consume CPU time and remains in the blocked state until the event occurs.

6. Resuming : The operating system can resume the execution of a previously blocked process when the event it was waiting for has occurred. This operation involves restoring the process's context and making it ready to run again.

7. Synchronization : Processes often need to synchronize their activities to avoid conflicts and ensure orderly execution. The operating system provides mechanisms, such as semaphores, mutexes, and condition variables, to enable process synchronization.

8. Inter-Process Communication (IPC) : Processes may need to communicate with each other to share data or co-ordinate their activities. The operating system provides IPC mechanisms, such as pipes, sockets, and message queues, to facilitate communication between processes.

9. Priority adjustment : The operating system may allow priority adjustments for processes to influence their execution order. Higher-priority processes are given precedence over lower-priority ones, affecting the scheduling decisions.

10. Forking (for processes in UNIX-like systems) : In UNIX-like systems, the fork operation creates a new process (child process) as an exact copy of the calling process (parent process). The child process inherits the resources and execution context of the parent process and can then execute a different program using the exec system call.

2.7. INTER-PROCESS COMMUNICATION

Inter-Process Communication (IPC) refers to the mechanisms and techniques used by processes in an operating system to exchange data, share resources, and co-ordinate their activities. IPC is essential for enabling collaboration and communication between multiple processes running concurrently in a system.

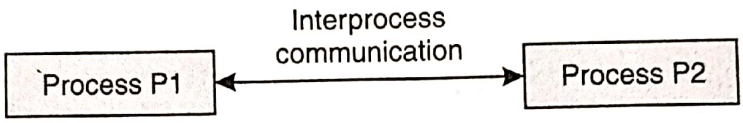


Fig. 2.5.

There are several IPC mechanisms provided by operating systems to facilitate inter-process communication, and some of the common ones include ;

1. Shared memory : Shared memory is a fast and efficient IPC mechanism where multiple processes can access the same region of memory. Processes can read from and write to this shared memory area, allowing them to communicate and share data directly without the need for copying data between processes.

2. Message-passing : Message-passing involves processes exchanging messages with each other through the operating system. Messages can contain data or signals to co-ordinate activities. This method provides a more structured and controlled way of communication, but it can be slower than shared memory due to data copying and OS involvement.

3. Pipes : Pipes are a one-way communication channel that allows processes to communicate in a producer-consumer fashion. One process writes data to the pipe, and another process reads the data from it. Pipes are typically used for sequential communication between related processes.

4. Named Pipes (FIFOs) : Named pipes are similar to regular pipes but have a unique file system name, allowing unrelated processes to communicate through them. Named pipes provide bidirectional communication.

5. Sockets : Sockets are a communication mechanism commonly used in network programming. Processes can communicate with each other over a network or locally on the same machine using TCP/IP or UDP protocols.

6. Signals : Signals are a form of asynchronous inter-process communication, where one process can send a signal to another process to notify it of an event or request it to perform a specific action.

7. Semaphores : Semaphores are synchronization mechanisms used to control access to shared resources between processes. They allow processes to signal and wait for conditions to co-ordinate their activities and prevent race conditions.

8. Mutexes (Mutual exclusion) : Mutexes are used to ensure that only one process can access a shared resource or critical section at a time, preventing data corruption or inconsistencies due to concurrent access.

9. Condition variables : Condition variables are used in conjunction with mutexes to allow processes to wait for a particular condition to be met before proceeding with their execution.

The choice of IPC mechanism depends on the specific requirements of the application and the nature of communication needed between processes. Properly implemented IPC is crucial for maintaining data integrity, preventing conflicts, and facilitating efficient collaboration between processes in a multi-process or multi-threaded environment.

2.7.1. Shared Memory Systems

A Shared Memory System, also known as Shared Memory Architecture, is a type of computer system that allows multiple processors or cores to access the same physical memory space simultaneously. In such systems, all processors can read and write to the shared memory region, making it an efficient mechanism for inter-processor communication and co-ordination.

Shared memory systems are commonly used in parallel and multi-core computing environments, as they provide a simple and straightforward way for processors to exchange information and work collaboratively on tasks. This contrasts with distributed memory systems, where each processor has its own separate memory and communication between processors requires explicit message-passing.

Working : In the shared memory system, the co-operating processes communicate, to exchange the data with each other. Because of this, the co-operating processes establish a shared region in their memory. The processes share data by reading and writing the data in the shared segment of the processes.

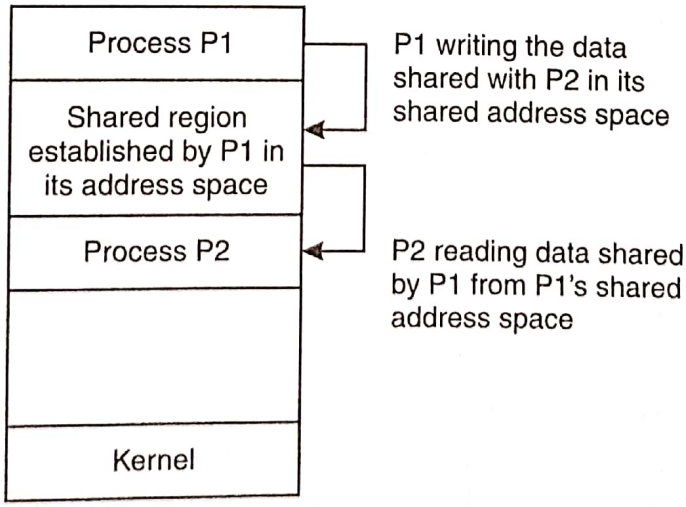


Fig. 2.6.

Let the two co-operating processes P1 and P2. Both the processes P1 and P2, have their different address spaces. Now let us assume, P1 wants to share some data with P2.

So, P1 and P2 will have to perform the following steps :

Step 1 : Process P1 has some data to share with process P2. First P1 takes initiative and establishes a shared memory region in its own address space and stores the data or information to be shared in its shared memory region.

Step 2 : Now, P2 requires the information stored in the shared segment of P1. So, process P2 needs to attach itself to the shared address space of P1. Now, P2 can read out the data from there.

Step 3 : The two processes can exchange information by reading and writing data in the shared segment of the process.

Advantages :

The advantages of shared memory are as follows :

1. It is a faster inter-process communication system.
2. It allows co-operating processes to access the same pieces of data concurrently.
3. It speeds up the computation power of the system and divides long tasks into smaller sub-tasks and can be executed in parallel.
4. Modularity is achieved in a shared memory system.
5. Users can perform multiple tasks at a time.

2.7.2. Message-passing Systems

Message-passing systems in operating systems refer to a method of Inter-Process Communication (IPC) where processes or threads exchange data and information by sending and receiving messages. This approach allows separate processes or threads to communicate and collaborate without sharing a common address space, which is the case in shared memory systems.

Message-passing systems are commonly used in distributed systems, multi-process systems, and parallel computing environments. The key components of message-passing systems in operating systems include ;

1. Message : A message is a unit of data that processes or threads exchange with each other. It typically contains information like the destination process ID, source process ID, message type, and the actual data payload.

2. Communication channels : Communication channels act as a conduit for passing messages between processes or threads. These channels can be either direct or indirect.

***(i)* Direct communication :** In direct communication, processes or threads must explicitly name the recipient(s) and sender(s) of the message. The sender sends the message directly to the recipient, and both processes have to know each other's identifiers.

***(ii)* Indirect communication :** Indirect communication involves using message queues or mailboxes that act as intermediaries for sending and receiving messages. Processes or Threads deposit messages into a mailbox and retrieve messages from a mailbox without needing to know the identity of the sender or recipient.

3. Synchronization : In message-passing systems, synchronization mechanisms are crucial to prevent race conditions and ensure that messages are sent and received in a co-ordinated

manner. Semaphore, mutexes, or other synchronization primitives may be used to control access to shared communication channels or resources.

4. Message-Passing Interface (MPI) : MPI is a widely used standard for message-passing in parallel computing environments, particularly in distributed memory systems. It provides a standardized set of functions that allow processes to send and receive messages, manage communication, and co-ordinate parallel tasks effectively.

Advantages :

1. Processes do not need to share memory, making it suitable for distributed systems or systems with security concerns.
2. They are generally more scalable as they don't suffer from contention issues associated with shared memory systems.
3. They allow for better modularity and encapsulation as processes remain isolated from each other.

Disadvantages :

1. Message-passing can introduce higher overhead compared to shared memory access due to the need to package and transmit data explicitly.
2. Designing and implementing communication protocols can be more complex than shared memory systems.

▶ 2.8. PROCESS SYNCHRONIZATION

In the previous section of this chapter, you have studied about the scheduling of the processes and various scheduling algorithms generally used to process multiple processes by a single processor. Now, we will be discussing about the synchronization of different processes running simultaneously in a system. The simultaneous processing of multiple processes generally results into performance enhancement of the system but it also have some side effects. The need for generating synchronization among multiple processes is one of the major side effect of multiprogramming.

2.8.1. Inter-process Interaction

You have already studied that the processing running simultaneously in a system can be competitive or cooperative in nature. The cooperative processes generally interact with each other.

This interaction between cooperative processes is mainly of three types :

1. Inter-process synchronization
2. Inter-process signaling
3. Inter-process communication.

1. Inter-process synchronization : Inter-process synchronization is a set of protocols and mechanisms used to preserve system integrity and consistency when concurrent processes share a serially usable resource. A resource is called as serially usable resource if it can be used by one process at one time. The state and operation of a serially usable resource may be corrupted when it is manipulated concurrently by more than one process without synchronization. For example, a printer is a serially usable device, which can process the print command of one user at the moment. It can start printing the document of the other user when the print job of one user is over. What will happen if two processes want to have a control over the same printer at the same time ? The synchronization methods are derived to deal with such kind of problems. Processes that are running simultaneously, interact with each other to achieve inter-process synchronization.

2. Inter-process signaling : Inter-process signaling refers to the exchange of timing signals between the concurrent processes or threads. Inter-process signaling is used to coordinate the collective progress of process or tasks. It is the very basic but common form of inter-process synchronization.

3. Inter-process communication : Concurrent cooperating processes must communicate with each other for exchanging data, reporting the progress and gathering collective results. Shared memory provides simple and common means of inter-process communication as all processes can access it. Every process uses the shared memory to communicate with the other process.

If we don't derive an appropriate synchronization and enforce its use by each process that is using the shared resource, it will result into an erratic system behaviour. The concurrent processing may result in performance enhancement but the same pollutes system if improper inter-process synchronization occurs.

2.8.2. Need for Inter-process Synchronization

The cooperating processes use shared variables for inter-process communications. When a set of processes has to access common address space, they can use shared variables for a number of purposes such as signaling flags. In this process, if there is no restriction on the updation of shared variables then it can lead to inconsistencies. These inconsistencies are dependent upon specific timing and interleaving of process actions. Errors generated due to concurrency are extremely difficult to find and debug. The following example shows an error of such kind.

Suppose Keyboard and Display are two processes, which are concerned with input of information from keyboard and displaying it on the monitor. Both the processes share common buffer for storing the input. The process Keyboard works in response to user input and places the input in buffer. From this buffer, the process Display, displays the character on the visual display unit. For effective working, both processes maintain their private pointer

to mark current working position in the buffer. Let us assume that Echo is the name of the variable that is shared by both the processes. The processes bodies can be assumed as under.

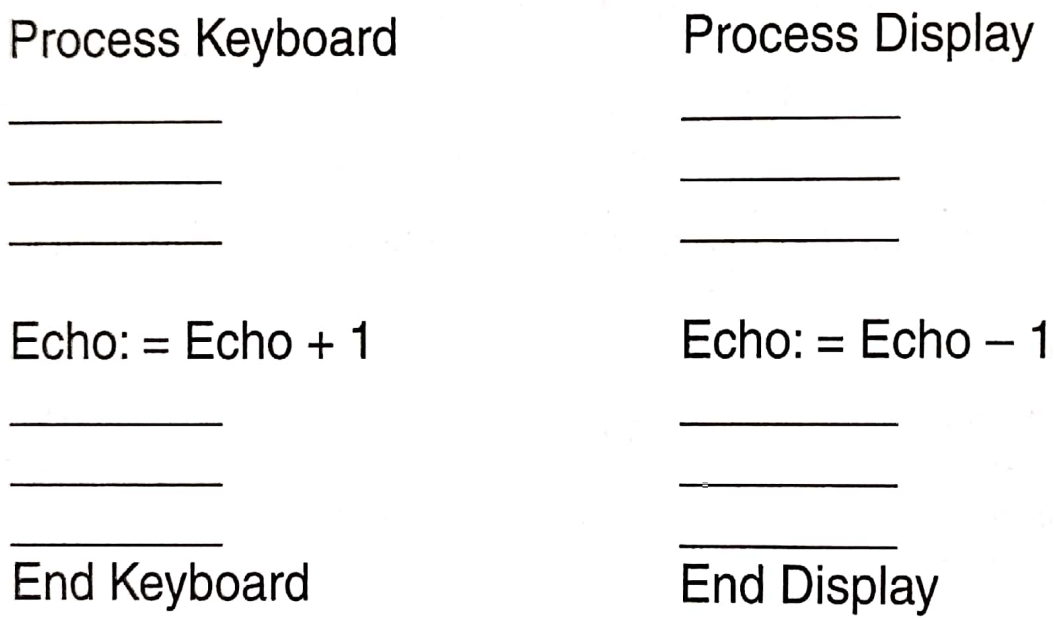


Fig. 2.7. Keyboard and display processes.

Whenever a character is read by the process Keyboard, it increments the value of the shared variable Echo by 1. The process Display displays a character and decrements the value of Echo by 1. As Display will always follow Keyboard, Echo will always contain a non negative value. Thus, Echo contains the running difference between the Keyboard and Display.

Assume that Keyboard is ahead than Display by 1 character and Echo contains 1. Now, a display interrupt occurs. The display process displays the character but berfore it should decrement Echo, an input interrupt occurs and decrement step is skipped. Thus, the value of echo will be 1 greater than the required value.

In the same way, Display can contain one value less than the required value if Display is called all of sudden, skipping the Echo increment step. Therefore, the problems are faced in an unsynchronized concurrent execution of cooperating processes.

To avoid such unsynchronized circumstances, synchronizing methods like mutual exclusion are used in operating systems.

2.9. SEMAPHORES IN PROCESS SYNCHRONIZATION

Semaphores are just normal variables used to co-ordinate the activities of multiple processes in a computer system. They are used to enforce mutual exclusion, avoid race conditions, and implement synchronization between processes.

The process of using Semaphores provides two operations :

wait (P) and **signal (V).**

Wait : It decrements the value of its A argument in case it is positive. In case S is zero or negative, then no operation would be performed.

```
wait(A)
{
while (A<=0);
A-;
}
```

Signal : This operation increments the actual value of its argument A.

```
signal(A)
{
A++;
}
```

The **wait** operation decrements the value of the semaphore, and the **signal** operation increments the value of the semaphore. When the value of the semaphore is zero, any process that performs a wait operation will be blocked until another process performs a signal operation.

Semaphores are used to implement critical sections, which are regions of code that must be executed by only one process at a time. A semaphore is a special kind of synchronization data that can be used only through specific synchronization primitives.

Semaphores are of two types :

1. **Binary semaphore :** This is also known as a mutex lock. It can have only two values *i.e.*, 0 and 1. Its value is initialized to 1. It is used to implement the solution of critical section problems with multiple processes.

2. **Counting semaphore :** Its value can range over an unrestricted domain. It is used to control access to a resource that has multiple instances.

Important and Expected Questions

Q.1. What is process in operating systems ?

Ans. In operating systems, the "process" is a fundamental concept that refers to an independent, executable unit of work. It represents a running program along with its current execution state. Processes allow an operating system to manage and execute multiple tasks concurrently, providing the illusion that multiple programs are running simultaneously on a single processor.

Q.2. What is process scheduling ?

Ans. The process scheduling is a process of selecting the next ready process for processing in a multitasking operating system. In multitasking or multiprogramming operating systems, several ready processes are kept in ready queue. These processes have acquired all the resources for their processing except the processor. When the processor processes a particular process for a time slice, it invokes the scheduler to select the next ready process from ready queue. Various

scheduling algorithms are used by various operating systems to process multiple applications simultaneously by a single central processing unit.

Q.3. What is inter-process synchronization ?

Ans. Inter-process synchronization is a set of protocols and mechanisms used to preserve system integrity and consistency when concurrent processes share a serially usable resource. A resource is called as serially usable resource if it can be used by one process at one time. The state and operation of a serially usable resource may be corrupted when it is manipulated concurrently by more than one process without synchronization.

Q.4. Briefly explain process state diagram.

Ans. In an operating system, a process can exist in several different states during its life cycle. The process states represent the various stages a process goes through from its creation to termination or suspension.

The typical process states include ;

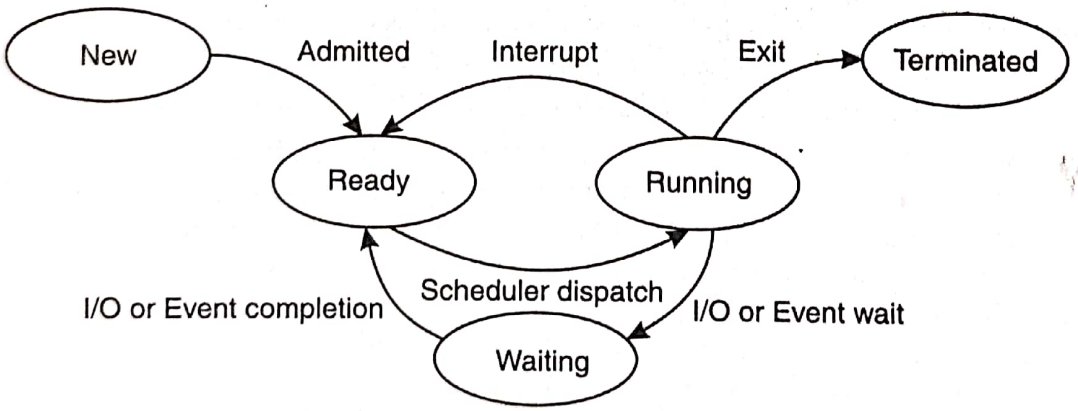
(*i*) New : This is the initial state when a process is first created. The operating system initializes the process, allocates necessary resources, and prepares it for execution.

(*ii*) Ready : Once the process has been initialized and is ready to run, it enters the ready state. In this state, the process is waiting to be assigned to a processor by the scheduler.

(*iii*) Running : When the scheduler selects the process for execution and the processor starts executing its instructions, the process enters the running state.

(*iv*) Blocked (Waiting) : While a process is running, it may encounter certain events, such as waiting for input/output operations to complete or for certain resources to become available. In such cases, the process is blocked or waiting, and it enters the blocked state. The operating system will not schedule a blocked process for execution until the blocking condition is resolved.

(*v*) Terminated : When a process completes its execution or is explicitly terminated by the user or the operating system, it enters the terminated state. At this point, the process is removed from the list of active processes, and its resources are deallocated.



Q.5. State some of the common operations on processes.

Ans. In an operating system, processes are fundamental units of execution that represent running programs and tasks. The operating system provides several operations to manage processes effectively. Some of the common operations on processes include ;

(*i*) Creation : The creation operation allows the operating system to initiate a new process. This operation involves allocating the necessary resources for the process, including memory space, setting up the Process Control Block (PCB), and initializing various process attributes.

(*ii*) **Termination :** The termination operation is used to end the execution of a process. When a process completes its tasks or is no longer needed, the operating system terminates it. This involves releasing all the resources associated with the process, deallocating memory, and removing its PCB.

(*iii*) **Scheduling :** The scheduling operation involves determining which process should be executed next by the CPU. The process scheduler uses scheduling policies and algorithms to decide the order of process execution and allocate CPU time to each process fairly.

(*iv*) **Context switching :** Context switching is the operation of saving the current execution context of a process and loading the context of another process. It is performed during a context switch when the operating system switches the CPU from one process to another to allow multi-tasking.

(*v*) **Waiting (Blocking) :** When a process is waiting for a particular event, such as Input/Output (I/O) completion or a resource becoming available, it enters a waiting state. During this time, the process does not consume CPU time and remains in the blocked state until the event occurs.

(*vi*) **Resuming :** The operating system can resume the execution of a previously blocked process when the event it was waiting for has occurred. This operation involves restoring the process's context and making it ready to run again.

(*vii*) **Synchronization :** Processes often need to synchronize their activities to avoid conflicts and ensure orderly execution. The operating system provides mechanisms, such as semaphores, mutexes, and condition variables, to enable process synchronization.

(*viii*) **Inter-Process Communication (IPC) :** Processes may need to communicate with each other to share data or co-ordinate their activities. The operating system provides IPC mechanisms, such as pipes, sockets, and message queues, to facilitate communication between processes.

Q.6. Write a short note on shared memory system.

Ans. A Shared Memory System, also known as Shared Memory Architecture, is a type of computer system that allows multiple processors or cores to access the same physical memory space simultaneously. In such systems, all processors can read and write to the shared memory region, making it an efficient mechanism for inter-processor communication and co-ordination.

Shared memory systems are commonly used in parallel and multi-core computing environments, as they provide a simple and straightforward way for processors to exchange information and work collaboratively on tasks. This contrasts with distributed memory systems, where each processor has its own separate memory and communication between processors requires explicit message-passing.

Objective Questions

Fill in the Blanks

1. ..... is the memory that is dynamically allocated to a process during its execution.
2. The ..... is a process of selecting the next ready process for processing in a multitasking operating system.

3. ..... refers to the mechanisms and techniques used by processes in an operating system to exchange data, share resources, and co-ordinate their activities.
4. A Shared Memory System, also known as ..... .
5. Binary semaphore is also known as a ..... .

True or False

1. In operating systems, PCB stands for Process Control Block.
2. Process counter indicates the current state of the process.
3. The termination operation involves determining which process should be executed next by the CPU.
4. Inter-process signaling refers to the exchange of timing signals between the concurrent processes or threads.
5. Semaphores are just normal variables used to co-ordinate the activities of multiple processes in a computer system.
6. Counting semaphore is used to control access to a resource that has multiple instances.

ANSWERS

Fill in the Blanks :

1. Heap
2. process scheduling
3. Inter-Process Communication (IPC)
4. Shared Memory Architecture
5. mutex lock

True or False :

1. True
2. False
3. False
4. True
5. True
6. True

Review Questions

1. State the various components of a process.
2. What is process control block in operating systems ?
3. Write a short note on scheduling queues.
4. What is inter-process communication ? Explain.
5. Write a short note on message-passing systems.
6. What is process synchronization ? Explain.
7. State the need for inter-process synchronization.
8. Write a short note on semaphores in process synchronization.

CHAPTER 3

CPU Scheduling

3.1. INTRODUCTION TO CPU SCHEDULING

CPU scheduling is the process of deciding which process will own the CPU to use while another process is suspended. The main function of the CPU scheduling is to ensure that whenever the CPU remains idle, the OS has at least selected one of the processes available in the ready-to-use line. It is the process of determining which processes in the system are allocated CPU time and in what order. It's an essential component for efficient resource utilization and system performance.

3.2. SCHEDULING QUEUES

CPU scheduling queues are data structures used by the operating system to organize processes based on their scheduling characteristics. These queues facilitate the efficient implementation of various scheduling algorithms.

Here are some common types of CPU scheduling queues :

1. Ready queue : The ready queue holds processes that are ready to execute but are waiting for CPU time. Processes in this queue are usually prioritized based on the scheduling algorithm being used. For example, in priority scheduling, processes with higher priority are placed at the front of the queue.

2. Blocked queue (or I/O queue) : The blocked queue (or I/O queue) holds processes that are blocked or waiting for some I/O operation to complete, such as reading from disk or waiting for user input. Processes are moved from the ready queue to the blocked queue when they need to perform I/O operations. When an I/O operation completes, the process is moved back to the ready queue to await CPU execution.

3. Device queues : Device queues hold processes waiting for a particular I/O device, such as a printer or disk drive. Each I/O device typically has its own queue to manage processes waiting for that device. Processes waiting for I/O are moved from the blocked queue to the appropriate device queue.

4. Expired queue (for preemptive algorithms) : In preemptive scheduling algorithms like Round Robin, processes are given a fixed time slice (quantum) to execute. If a process's

time slice expires before it completes, it is moved to the expired queue to wait for its turn to execute again. The expired queue ensures fairness by allowing processes that didn't complete within their time slice to get another chance to execute.

5. Feedback queue (for multilevel feedback queue scheduling) : In multilevel feedback queue scheduling, processes can move between different queues based on their behaviour. The feedback queue holds processes that have been demoted or promoted between different priority levels based on their execution history. Processes with high CPU burst times may be demoted to lower-priority queues, while processes with short burst times may be promoted to higher-priority queues.

▶ 3.3. CPU SCHEDULER

The CPU scheduler is a key component of an operating system responsible for determining which process from the ready queue should be allocated the CPU (Central Processing Unit) for execution. Its primary goal is to maximize CPU utilization, throughput, and overall system performance while ensuring fairness and responsiveness to various processes.

When a process becomes ready to execute, it enters the ready queue, and the CPU scheduler decides which process should run next based on a scheduling algorithm. There are several scheduling algorithms, each with its own characteristics and performance trade-offs.

■ 3.3.1. Scheduling Criteria

CPU scheduling criteria are the factors and metrics used to evaluate the performance and effectiveness of different CPU scheduling algorithms. These criteria help in selecting the most suitable scheduling algorithm for a given system and workload. There are several important scheduling criteria, and the choice of which criterion is prioritized depends on the specific requirements and goals of the system. Some of the commonly used scheduling criteria are ;

1. CPU utilization : The CPU should be kept as busy as possible to achieve maximum utilization. A good scheduling algorithm aims to minimize CPU idle time and keep the CPU busy executing processes.

2. Throughput : Throughput refers to the total number of processes completed per unit of time. A scheduling algorithm that maximizes throughput can lead to higher overall system performance.

3. Turnaround time : Turnaround time is the time taken for a process to complete execution from the time of submission. A good scheduling algorithm should strive to minimize the turnaround time, as it affects the responsiveness of the system.

4. Waiting time : Waiting time is the total time a process spends waiting in the ready queue before it gets CPU time for execution. Lower waiting times indicate better scheduling efficiency.

5. Response time : Response time is the time taken from when a process is submitted until it starts responding and producing output. Short response times are critical for interactive systems to provide a quick user experience.

6. Fairness : Fairness refers to giving each process a fair share of the CPU time. A scheduling algorithm should ensure that no process is continuously starved of CPU resources.

7. Predictability : Predictability refers to the ability to estimate and control the behavior of the system. A scheduling algorithm should provide consistent and predictable results to ensure reliable system behavior.

8. Overhead : Scheduling overhead is the additional time and computational resources required to manage the scheduling process itself. A good scheduling algorithm should have minimal overhead.

9. Context switching : Context switching involves saving the state of a running process and restoring the state of the next process to be executed. A scheduling algorithm should aim to minimize the frequency of context switches to reduce overhead.

10. Preemption overhead : For preemptive scheduling algorithms, preemption overhead refers to the cost associated with frequently interrupting and resuming processes. Minimizing preemption overhead is essential for efficient CPU utilization.

The importance of each criterion may vary based on the specific system requirements. For example, in real-time systems, meeting strict deadlines and ensuring low response times may be more critical than maximizing throughput. In contrast, in a batch processing system, maximizing throughput and CPU utilization might be the primary goals. The choice of the best scheduling algorithm requires a careful consideration of these criteria and their impact on the system's overall performance and behavior.

▶ 3.4. SCHEDULING ALGORITHMS

The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.

Process scheduling is an essential part of a multi-programming operating system. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing.

There are two categories of scheduling ;

1. Non-preemptive : Here the resource can't be taken from a process until the process completes execution. The switching of resources occurs when the running process terminates and moves to a waiting state.

2. Preemptive : Here the OS allocates the resources to a process for a fixed amount of time. During resource allocation, the process switches from running state to ready state

or from waiting state to ready state. This switching occurs as the CPU may give priority to other processes and replace the process with higher priority with the running process.

The scheduling algorithms explained in this section, may be suited to any of the three schedulers. Some algorithms are better suited to the needs of a particular type of scheduler.

In general, scheduling discipline may be preemptive or non preemptive. In terms of long term scheduler, non preemption means that once scheduled, a selected job runs till its completion. In terms of short term scheduler, non preemption means a running process will retain the ownership of its resources unless it is completed. Other process is scheduled either it is completed or suspended. In preemptive scheduling, a running process is replaced by a higher priority process at any time. This is done in response to events that change global system state. Following are the commonly used scheduling algorithms in the modern operating systems :

3.4.1. First Come First Serve (FCFS) Scheduling

First come first serve is the simplest scheduling algorithm available in operating systems. In this, the workload is simply processed in the order of its arrival. The process which requests the CPU first, is given the CPU first. The first come first serve algorithm is non preemptive *i.e.* once the CPU has been allocated to a process, it keeps it till its completion or till it is swapped out due to some input/output request or any other reason.

The first come first serve algorithm may result in the poor performance as it does not consider the state of the system and resource requirements of individual scheduling entities. As there is no preemption, component utilization and system throughput may be quite low.

Important *In the first come first serve scheduling, the workload is simply processed in the order of its arrival. The process, which requests the CPU, first, is given the CPU first. The first come first serve algorithm is non preemptive.*

There is no discrimination on the basis of the required services, so the shorter jobs may suffer turnaround delays and waiting times when one or more jobs are in queue. Consider a system with two jobs J1 and J2 with the total execution time of 20 and 2 time units respectively. Suppose that they arrive in the order J1 and then J2 so that J2 must wait for J1 to complete. Then, only it can be processed. Both the processes will be processed in 20 + 22 = 42 seconds. The average execution time will be 42/2 = 21 time units. The waiting times for both the processes will be 0 and 20 time units respectively as J2 has to wait for J1 to complete. The average waiting time will be 0 + 20 = 10 time units.

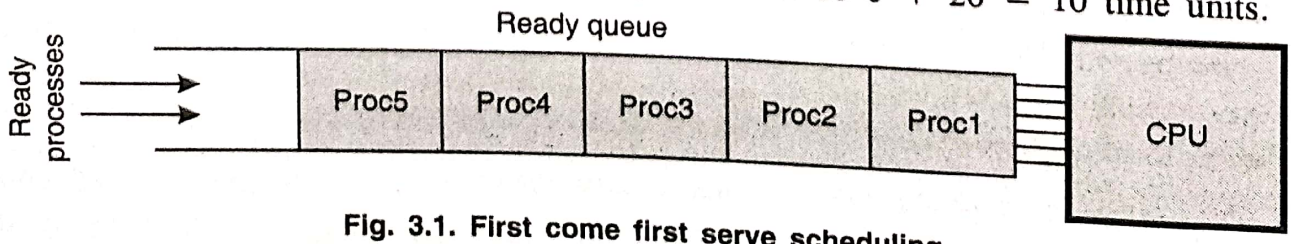


Fig. 3.1. First come first serve scheduling.

2. Symmetric Multiprocessing (SMP) : In symmetric multiprocessing, all processors are treated equally, and there is no master-slave hierarchy. Each processor can execute processes independently, and all processors contribute to system-wide task execution. SMP systems are more commonly used in modern desktops, servers, and high-performance computing clusters.

Schedulers in SMP systems : Schedulers in SMP systems need to make decisions about which process should be executed on which CPU or core at any given time. There are various CPU scheduling algorithms designed for SMP systems to achieve efficient load balancing and resource utilization. Some common scheduling algorithms include ;

1. Round Robin (RR) : A preemptive algorithm that allocates CPU time to each process in fixed time slices (time quantum) in a circular order. This ensures fair sharing of CPU time among processes.

2. Multilevel queue scheduling : Processes are classified into different priority queues based on their characteristics. Each queue may have its own scheduling algorithm, allowing different priority levels for different types of tasks.

3. Completely Fair Scheduler (CFS) : A modern scheduling algorithm used in the Linux kernel for SMP systems. It assigns each process a weight based on its priority and the available CPU resources, ensuring fairness and near-optimal CPU utilization.

4. Load balancing : SMP systems often use load balancing techniques to distribute tasks evenly across all available CPUs. Load balancing aims to prevent situations where some CPUs are heavily loaded while others remain idle.

The choice of the scheduling algorithm depends on the system's workload, performance requirements, and the level of fairness desired between different processes. Effective multiprocessing scheduling is crucial for exploiting the full potential of multi-core systems and achieving high-performance computing.

Important and Expected Questions

Q.1. What do you mean by the term CPU scheduling ?

Ans. CPU scheduling is the process of deciding which process will own the CPU to use while another process is suspended. The main function of the CPU scheduling is to ensure that whenever the CPU remains idle, the OS has at least selected one of the processes available in the ready-to-use line. It is the process of determining which processes in the system are allocated CPU time and in what order. It's an essential component for efficient resource utilization and system performance.

Q.2. Write a short note on CPU scheduler.

Ans. The CPU scheduler is a key component of an operating system responsible for determining which process from the ready queue should be allocated the CPU (Central Processing Unit) for

execution. Its primary goal is to maximize CPU utilization, throughput, and overall syst performance while ensuring fairness and responsiveness to various processes.

When a process becomes ready to execute, it enters the ready queue, and the CPU schedu decides which process should run next based on a scheduling algorithm. There are seve scheduling algorithms, each with its own characteristics and performance trade-offs.

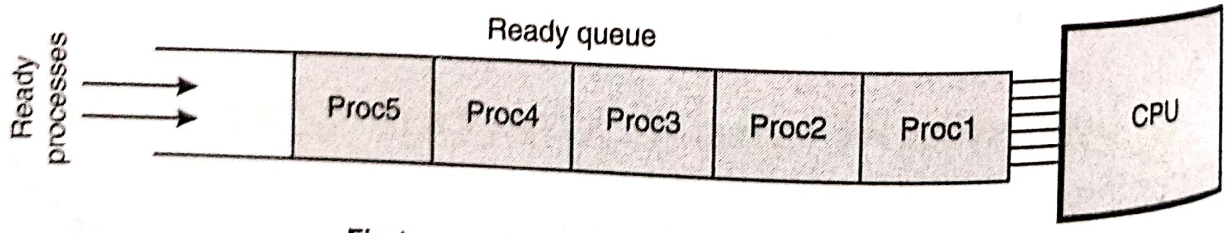
Q.3. Briefly explain FCFS scheduling.

Ans. First come first serve is the simplest scheduling algorithm available in operating systems this, the workload is simply processed in the order of its arrival. The process which reque the CPU first, is given the CPU first. The first come first serve algorithm is non preemptiv *i.e.* once the CPU has been allocated to a process, it keeps it till its completion or till it swapped out due to some input/output request or any other reason.

The first come first serve algorithm may result in the poor performance as it does not consid the state of the system and resource requirements of individual scheduling entities. As there no preemption, component utilization and system throughput may be quite low.

Important : In the first come first serve scheduling, the workload is simply processed in th order of its arrival. The process, which requests the CPU, first, is given the CPU first. Th first come first serve algorithm is non preemptive.

There is no discrimination on the basis of the required services, so the shorter jobs may suffe turnaround delays and waiting times when one or more jobs are in queue. Consider a system with two jobs J1 and J2 with the total execution time of 20 and 2 time units respectively Suppose that they arrive in the order J1 and then J2 so that J2 must wait for J1 to complete Then, only it can be processed. Both the processes will be processed in 20 + 22 = 4 seconds. The average execution time will be 42/2 = 21 time units. The waiting times for both the processes will be 0 and 20 time units respectively as J2 has to wait for J1 to complete The average waiting time will be 0 + 20 = 10 time units.



First come first serve scheduling.

Suppose the processes arrive in the order J2 and then, J1. In this case, the average execution time will be 2 + 22 = 24/2 = 12 time units. The average waiting time will be 0 + 2 = 2/ 2 = 1 time unit. This example shows that how shorter jobs are hurt by longer ones. The figure explains the process of first come first serve scheduling.

Q.4. Write a short note on multilevel feedback queue scheduling.

Ans. Multilevel feedback queue scheduling is similar to multilevel queue scheduling except it provides movement of processes between various queues. This means a process can move from one to the other queue. In multilevel feedback queue scheduling, the processes are separated on the

basis of their CPU requirements. If a process uses too much of CPU time, it is moved to the low priority queue. Input – output bound and interactive processes are kept in high priority.

Q.5. Differentiate between non-preemptive and preemptive SJF.

Ans. (*i*) **Non-preemptive SJF :** In the non-preemptive SJF algorithm, once a process starts executing, it runs until it completes or voluntarily yields the CPU (*e.g.*, through an I/O request). The scheduler selects the process with the smallest burst time from the ready queue and allows it to run until it finishes. The next process is then selected based on the remaining processes' burst times in the ready queue.

(*ii*) **Preemptive SJF :** In the preemptive SJF algorithm, a newly arriving process with a shorter burst time can preempt the currently running process if its burst time is longer. The preempted process is moved back to the ready queue, and the shorter job is allowed to execute. This approach helps to handle cases where shorter jobs arrive after the current process has started executing, potentially reducing overall waiting times.

Objective Questions

Fill in the Blanks

1. ..... is the process of deciding which process will own the CPU to use while another process is suspended.
2. The non preemptive version of shortest remaining time next is called ..... .
3. In round robin scheduling, the processing time is divided into small units called ..... or ..... .
4. ..... scheduling in operating systems refers to the management of multiple processes that run on multiple CPUs or cores simultaneously.
5. In asymmetric multiprocessing, there is a ..... relationship between processors.

True or False

1. Expired queues hold processes waiting for a particular I/O device.
2. The first come first serve algorithm is non preemptive.
3. In the first come first serve scheduling, the workload is simply processed in the order of its arrival.
4. Round robin scheduling is regarded as "Fair" scheduling.
5. Event driven or Priority scheduling is a general class, priority based scheduling algorithm.
6. Multiple level queue scheduling partitions the ready queue into three separate queues.
7. In the preemptive SJF algorithm, once a process starts executing, it runs until it completes or voluntarily yields the CPU.

CHAPTER 4

Deadlocks

4.1. DEADLOCK MODEL

A deadlock is a situation in multiprogramming systems when each process of a group acquires some of resources needed for its completion while it is waiting for other resources that are acquired by other processes of same group to be released. This situation permanently blocks all the processing and the system may come to a halt.

Deadlock is the major side effect of synchronization and concurrent processing. It occurs as a result of uncontrolled granting of system resources to requesting processes. For example, consider three tape drives. Suppose each of three processes hold one tape drive. Every process now requests the other tape drive. Three processes will be now in deadlock state. Another example is a process having control of the tape drive, requires printer to complete its job but the other process acquiring printer needs tape drive to complete its printing job, a deadlock occurs in which both the processes are waiting for the resources.

Deadlocks can occur as a result of competition over any shared device. During deadlock, processes never finish and all system resources are blocked. This prevents new jobs from starting.

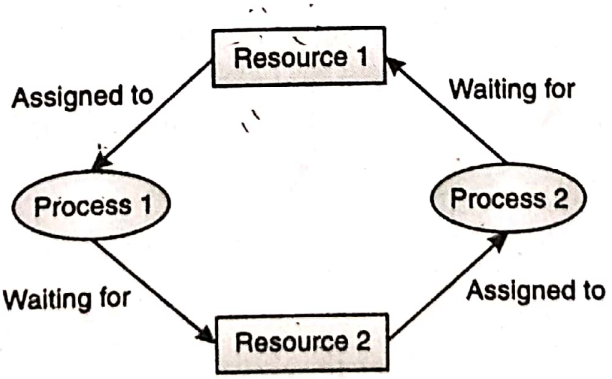


Fig. 4.1. Deadlock in operating system.

System model :

1. For the purposes of deadlock discussion, a system can be modeled as a collection of limited resources that can be divided into different categories and allocated to a variety of processes, each with different requirements.

2. Memory, printers, CPUs, open files, tape drives, CD-ROMs, and other resources are examples of resource categories.
3. By definition, all resources within a category are equivalent, and any of the resources within that category can equally satisfy a request from that category. If this is not the case (*i.e.* if there is some difference between the resources within a category), then that category must be sub-divided further. For example, the term "printers" may need to be sub-divided into "laser printers" and "color inkjet printers."
4. Some categories may only have one resource.
5. The kernel keeps track of which resources are free and which are allocated, to which process they are allocated, and a queue of processes waiting for this resource to become available for all kernel-managed resources. Mutexes or wait( ) and signal( ) calls can be used to control application-managed resources (*i.e.* binary or counting semaphores).
6. When every process in a set is waiting for a resource that is currently assigned to another process in the set, the set is said to be deadlocked.

▶ 4.2. DEADLOCK CHARACTERISTICS

For a deadlock to occur, following conditions must be true :

1. Mutual exclusion : Shared resources are acquired and used in a mutually exclusive manner *i.e.* only one at a time.

2. Hold and wait : Each process holds the resources allocated to it while waiting for other resources.

3. No preemptions : Once allocated, resources can only be released back by that process. The system can't forcefully revoke them.

4. Circular waiting : Deadlock processes are invoked in a circular chain such that each process holds some resources needed by others.

All the above stated conditions must be there for a deadlock to occur.

■ 4.2.1. Deadlock Characterization

A deadlock happens in operating system when two or more processes need some resource to complete their execution that is held by the other process. A deadlock occurs if the four Coffman conditions hold true. But these conditions are not mutually exclusive.

Here's a detailed look at deadlock characterization :

1. Mutual exclusion : There should be a resource that can only be held by one process at a time. In Fig. 4.2, there is a single instance of Resource 1 and it is held by Process 1 only.

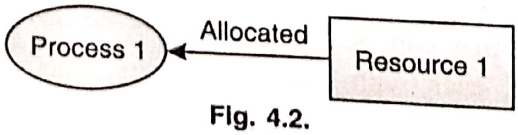


Fig. 4.2.

2. Hold and wait : A process can hold multiple resources and still request more resources from other processes which are holding them. In Fig. 4.3, Process 2 holds Resource 2 and Resource 3 and is requesting the Resource 1 which is held by Process 1.

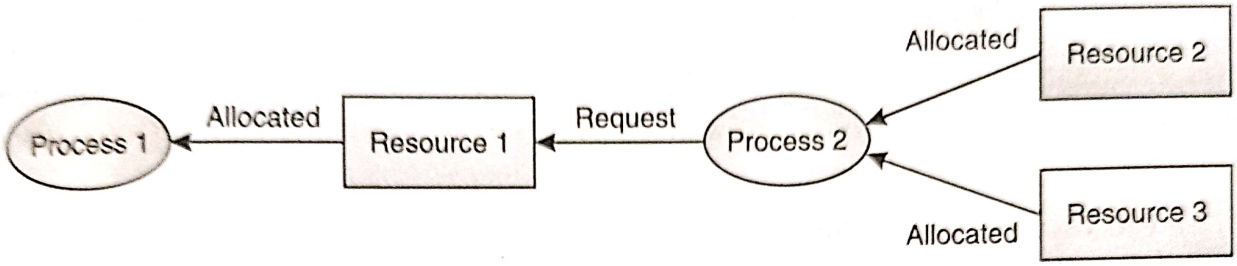


Fig. 4.3.

3. No preemption : A resource cannot be preempted from a process by force. A process can only release a resource voluntarily. In Fig. 4.4, Process 2 cannot preempt Resource 1 from Process 1. It will only be released when Process 1 relinquishes it voluntarily after its execution is complete.

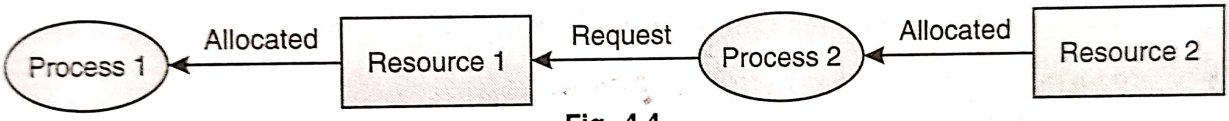


Fig. 4.4.

4. Circular wait : A process is waiting for the resource held by the second process, which is waiting for the resource held by the third process and so on, till the last process is waiting for a resource held by the first process. This forms a circular chain. For example : Process 1 is allocated Resource 2 and it is requesting Resource 1. Similarly, Process 2 is allocated Resource 1 and it is requesting Resource 2. This forms a circular wait loop.

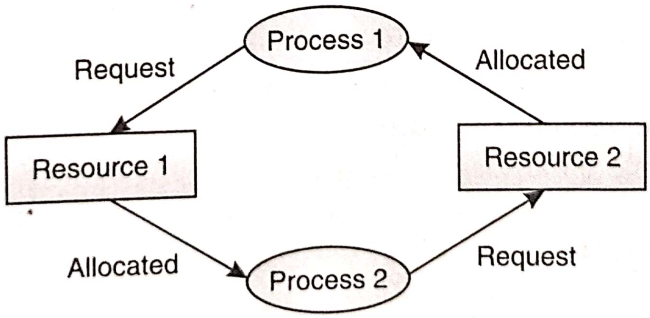


Fig. 4.5.

4.3. METHODS FOR HANDLING DEADLOCKS

As deadlocks are very dangerous for the performance of the computer system, several deadlock handling techniques are designed to save the system from deadlock state. Most of deadlock handling techniques can fall into one of these three classes :

1. Deadlock prevention
2. Deadlock avoidance
3. Deadlock detection and recovery.

9. Complexity of resource dependencies : Deadlock detection and recovery may become less effective in systems with intricate resource dependencies, as accurately identifying and resolving deadlocks in such environments can be more challenging.

Important and Expected Questions

Q.1. What is deadlock ?

Ans. A deadlock is a situation in multiprogramming systems when each process of a group acquires some of resources needed for its completion while it is waiting for other resources that are acquired by other processes of same group to be released. This situation permanently blocks all the processing and the system may come to a halt.

Deadlock is the major side effect of synchronization and concurrent processing. It occurs as a result of uncontrolled granting of system resources to requesting processes. For example, consider three tape drives. Suppose each of three processes hold one tape drive. Every process now requests the other tape drive. Three processes will be now in deadlock state. Another example is a process having control of the tape drive, requires printer to complete its job but the other process acquiring printer needs tape drive to complete its printing job, a deadlock occurs in which both the processes are waiting for the resources.

Deadlocks can occur as a result of competition over any shared device. During deadlock, processes never finish and all system resources are blocked. This prevents new jobs from starting.

Q.2. Write a short note on deadlock detection and recovery.

Ans. In deadlock detection and recovery, the system grants the access to each requesting process freely. It occasionally, checks for deadlocks in order to reclaim resources held by processes in deadlock.

At a stage, when system is checked for deadlock, the detection algorithm examines all possible sequences for incomplete processes. If competition sequence exists then the system is not in deadlock stage otherwise the system is in deadlock stage and all incomplete processes are blocked.

Deadlock detection is only a part of deadlock detection and recovery process. Deadlock detection only reveals a problem does not solves it. Now, the system must break the deadlock so that processes may be processed.

The first step in deadlock recovery is to identify deadlocked processes. The next step is to roll back or restart one or more processes causing deadlock. Restarting leads to the loss of work done by the particular process. So, generally those processes are chosen which are less costly to roll back. The process is rolled back to the point where the deadlock is released.

Such facility is required by systems needing high reliability and or availability but these algorithms will be dangerous when processes have made changes, which can be rolled back.

Q.3. State the various advantages of deadlock detection and recovery in operating systems.

Ans. Deadlock detection and recovery mechanisms in operating systems offer several advantages that contribute to the stability and reliability of the system.

Here are some of the key advantages ;

***(i)* Deadlock resolution :** Deadlock detection and recovery mechanisms provide a way to resolve deadlocks that may occur in the system. By identifying the specific processes and

resources involved in the deadlock, the system can take appropriate actions to break the deadlock and allow the processes to proceed, preventing a system-wide freeze.

(ii) **Graceful handling :** Deadlock recovery mechanisms offer a more graceful way of dealing with deadlocks compared to other methods, such as system restarts or complete termination of all processes. Deadlock resolution techniques can target specific processes or resources involved in the deadlock, minimizing the impact on the overall system.

(iii) **Minimal disruption :** Deadlock recovery aims to minimize the disruption caused by a deadlock. Instead of abruptly terminating processes, the system can choose to preempt specific resources or rollback processes to a consistent state, allowing for a smoother recovery process.

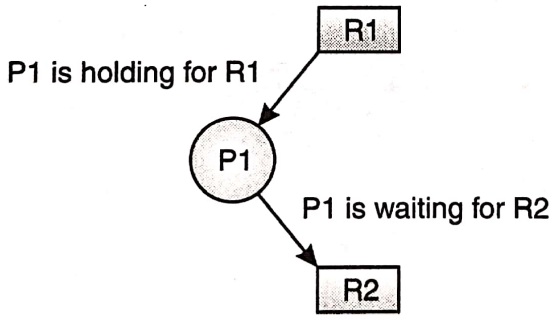
(iv) **Improved system availability :** By detecting and resolving deadlocks promptly, the system's availability and responsiveness can be improved. Deadlocks can be addressed as soon as they occur, reducing the time during which critical resources remain unavailable.

Q.4. Write some common deadlock prevention techniques.

Ans. Some common deadlock prevention techniques include ;

(i) **Mutual exclusion :** Ensure that resources, such as critical sections or shared data, are accessible by only one process at a time. By preventing multiple processes from simultaneously accessing the same resource, you eliminate the possibility of a circular waiting condition, one of the necessary conditions for a deadlock to occur.

(ii) **Hold and wait :** To prevent the hold and wait condition, a process requesting resources must not hold any resource while waiting for additional resources. This can be achieved by adopting a policy that requires a process to request and acquire all the necessary resources before starting execution.



Hold and wait.

(iii) **No preemption :** Avoid resource preemption, meaning that a resource cannot be forcibly taken away from a process. If a process holds a resource and requests additional resources that are currently unavailable, it will be allowed to hold onto its currently acquired resources without interruption. This prevents the circular waiting condition from arising.

(iv) **Circular wait :** Enforce a total ordering of resources and ensure that processes always request resources in a specific order. By requiring processes to follow a pre-determined order while requesting resources, the potential for circular waiting is eliminated.

While deadlock prevention techniques are effective at avoiding deadlocks, they may introduce inefficiencies in resource utilization and process execution. For instance, requiring a process to acquire all necessary resources upfront could lead to resource wastage if a process acquires

resources it may not ultimately need. Additionally, enforcing strict ordering of resource requests can lead to decreased parallelism and reduced system performance.

In practice, deadlock prevention is not always practical or desirable for all systems. As an alternative, systems may employ deadlock detection and recovery mechanisms to identify and resolve deadlocks if they occur. Deadlock detection allows a system to identify deadlocks and take corrective actions such as process termination, resource preemption, or process rollback to break the deadlock and restore normal operation. These approaches strike a balance between prevention and resolution to ensure system stability and performance.

Objective Questions

Fill in the Blanks

1. ..... is the major side effect of synchronization and concurrent processing.
2. The resource allocater keeps track of number of ..... and ..... resources of each type.
3. The ..... is used to visualize the system's current state as a graph.

True or False

1. Deadlocks occur as a result of uncontrolled granting of system resources to requesting processes.
2. Circular wait can be avoided by linear ordering of resources in a system.
3. The basic idea of deadlock avoidance is to grant only those resource requests that can possibly result into a state of deadlock.
4. Deadlock prevention is a proactive approach of avoiding the occurrence of deadlocks in a computer system.

ANSWERS

Fill in the Blanks :

1. Deadlock
2. allocated, free
3. resource allocation graph

True or False :

1. True
2. True
3. False
4. True

Review Questions

1. Define deadlock.
2. State and explain the various methods for handling deadlocks.
3. Write a short note on deadlock prevention.
4. What are the disadvantages of deadlock detection and recovery in operating systems ?
5. Explain how deadlocks can be prevented.
6. What is deadlock ? How it degrades the performance ?